数据仓库服务
8.0.x
开发指南

文档版本 034

发布日期 2023-01-13

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 欢迎

1.1 文档面向的读者对象

数据库开发指南重点面向数据库的设计者、应用程序开发人员或DBA，提供设计、构建、查询和维护数据仓库所需的信息。

作为数据库管理员和应用程序开发人员，至少需要了解以下知识：

- 操作系统知识。这是一切的基础。
- SQL语法。这是操作数据库的必备能力。

声明

GaussDB(DWS)的作者们在进行文档写作时努力基于商用角度，从使用场景和任务完成角度给出内容指引。即使这样，文档中依然可能存在对Postgres内容的引用和参考。对于这类内容，遵从如下的Postgres Copyright：

1.2 阅读指引

对于首次接触 GaussDB(DWS)的用户，建议先阅读以下部分：

- **产品介绍**——介绍GaussDB(DWS)服务的特点、功能和适用场景。
- **快速入门**——GaussDB(DWS)入门包含一个示例，引导您完成创建数据仓库集群、创建数据库表、上传数据和测试查询这一过程。

如果打算或正在将应用程序从其他数据仓库向GaussDB(DWS)迁移，您可能想知道GaussDB(DWS)在实施方式上有什么区别。

GaussDB(DWS)进行数据库应用程序开发过程中，下表将帮您找到对应的信息。

如果要..	查阅建议
快速开始使用GaussDB(DWS)。	首先，按照**《数据仓库服务快速入门》**中的步骤快速部署集群、连接到数据库并尝试进行一些查询。 准备好构建数据库后，将数据加载到表中并编写查询内容以操作数据仓库中的数据后，可以回到《数据库开发指南》。
了解GaussDB(DWS)数据仓库的内部架构。	**产品架构**概括介绍了GaussDB(DWS)的架构。 如果您想要更全面地了解GaussDB(DWS)服务，请转到GaussDB(DWS)产品首页。
了解如何设计表以实现良好性能。	**开发设计建议**介绍数据库应用程序开发过程中，应当遵守的设计规范。依据这些规范进行建模，能够更好的契合GaussDB(DWS)的分布式处理架构，输出更高效的业务SQL代码。 对业务的执行效率不满意，期望通过调优加快业务执行的情况下，可以参考**优化查询性能**进行调优。性能调优是一项复杂的工程，有些时候无法系统性地说明和解释，而是依赖于DBA的经验判断。尽管如此，**优化查询性能**一节还是期望能尽量系统性的对性能调优方法加以说明，方便应用开发人员和刚接触GaussDB(DWS)的DBA参考。
加载数据。	**导入数据**介绍数据入库GaussDB(DWS)的方法和途径。 **导入最佳实践**提供有关快速高效数据导入的经验提示。
管理用户、组和数据库安全。	**管理数据库安全**涵盖数据库安全主题。
监控和优化系统性能。	**系统表和系统视图**详细介绍您可以从中查询数据库状态并监控查询内容与流程的系统表和视图。 您还应该查阅**管理指南**了解如何使用GaussDB(DWS)管理控制台检查系统运行状况、监控指标。

1.3 文档表达约定

举例约定

内容	说明
dbadmin	表示创建集群时指定的运行和维护GaussDB(DWS)的用户。
8000	表示GaussDB(DWS)监听客户端连接请求的端口号。

手册中的SQL示例是基于TPC-DS模型开发的，如果需要运行手册中的示例，请先参考官网说明（**http://www.tpc.org/tpcds/**），安装TPC-DS benchmark。

SQL 语法文本格式约定

为了方便对语法使用的理解，在文档中对SQL语法文本按如下格式进行表述。

格式	意义
大写	语法关键字（语句中保持不变、必须照输的部分）采用大写表示。
小写	参数（语句中必须由实际值进行替代的部分）采用小写表示。
[ ]	表示用“[ ]”括起来的部分是可选的。
...	表示前面的元素可重复出现。
[x | y | ...]	表示从两个或多个选项中选取一个或者不选。
{ x | y | ... }	表示从两个或多个选项中选取一个。
[x | y | ...] [...]	表示可选多个参数或者不选，如果选择多个参数，则参数之间用空格分隔。
[x | y | ...] [,...]	表示可选多个参数或者不选，如果选择多个参数，则参数之间用逗号分隔。
{ x | y | ... } [...]	表示可选多个参数，至少选一个，如果选择多个参数，则参数之间以空格分隔。
{ x | y | ... } [,...]	表示可选多个参数，至少选一个，如果选择多个参数，则参数之间用逗号分隔。

1.4 前置条件

使用本指南前，需要完成以下任务。

- 创建GaussDB(DWS)集群。

- 安装SQL客户端。
- 将SQL客户端连接到集群的默认数据库。

关于上述任务的详细指导，请参见**《数据仓库服务快速入门》**。

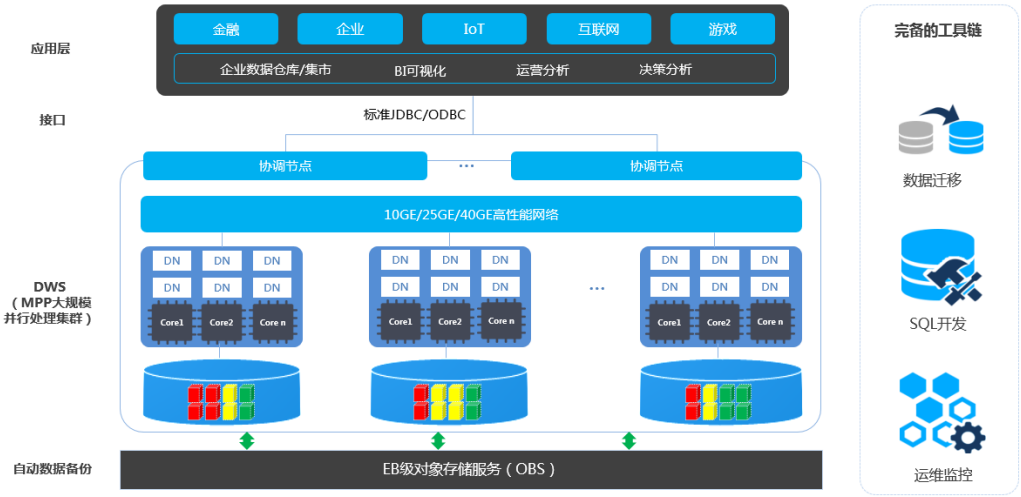
您还应该知道如何使用SQL客户端，并且对SQL语言有基本的了解。

2 系统概述

2.1 产品架构

GaussDB(DWS)是基于Share-nothing架构的分布式、并行数据库集群，其产品架构请参见图2-1。

图 2-1 GaussDB(DWS)产品架构图



- 应用层

  数据加载工具、ETL（Extract-Transform-Load）工具、以及商业上的BI工具、数据挖掘和分析工具，均可以通过标准接口与GaussDB(DWS)集成。GaussDB(DWS)兼容PostgreSQL生态，且SQL语法进行了兼容Oracle和Teradata的处理。应用只需做少量改动即可向GaussDB(DWS)平滑迁移。

- 接口

  支持应用程序通过标准JDBC 4.0和ODBC 3.5连接GaussDB(DWS)。

- GaussDB(DWS)（MPP大规模并行处理集群）

  由实施数据管理的模块组成，有关集群的组成及各模块的功能请参见图2-2和表2-1。

- 自动数据备份

  支持将集群快照自动备份到EB级对象存储服务OBS（Object Storage Service）中，方便利用业务空闲期对集群做周期备份以保证集群异常后的数据恢复。

  快照是GaussDB(DWS)集群在某一时间点的完整备份，记录了这一时刻指定集群的所有配置数据和业务数据。

- 工具链

  提供了数据并行加载工具GDS（General Data Service）、语法迁移工具DSC、SQL开发工具Data Studio，并支持通过控制台对集群进行运维监控。

GaussDB(DWS)集群逻辑架构如图2-2所示。实例的详细介绍请参见表2-1。

图 2-2 集群逻辑架构图

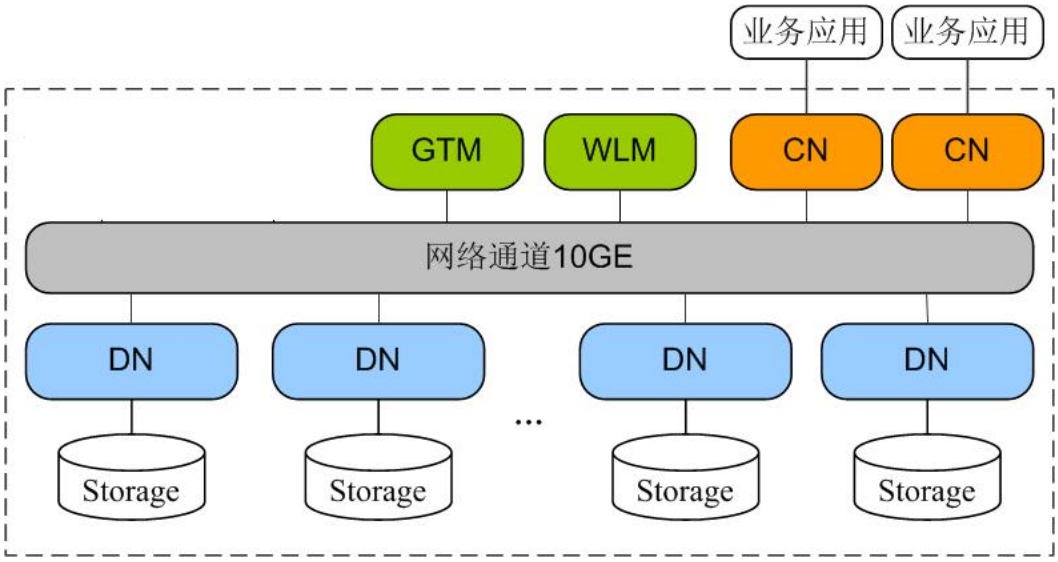


表 2-1 集群架构说明

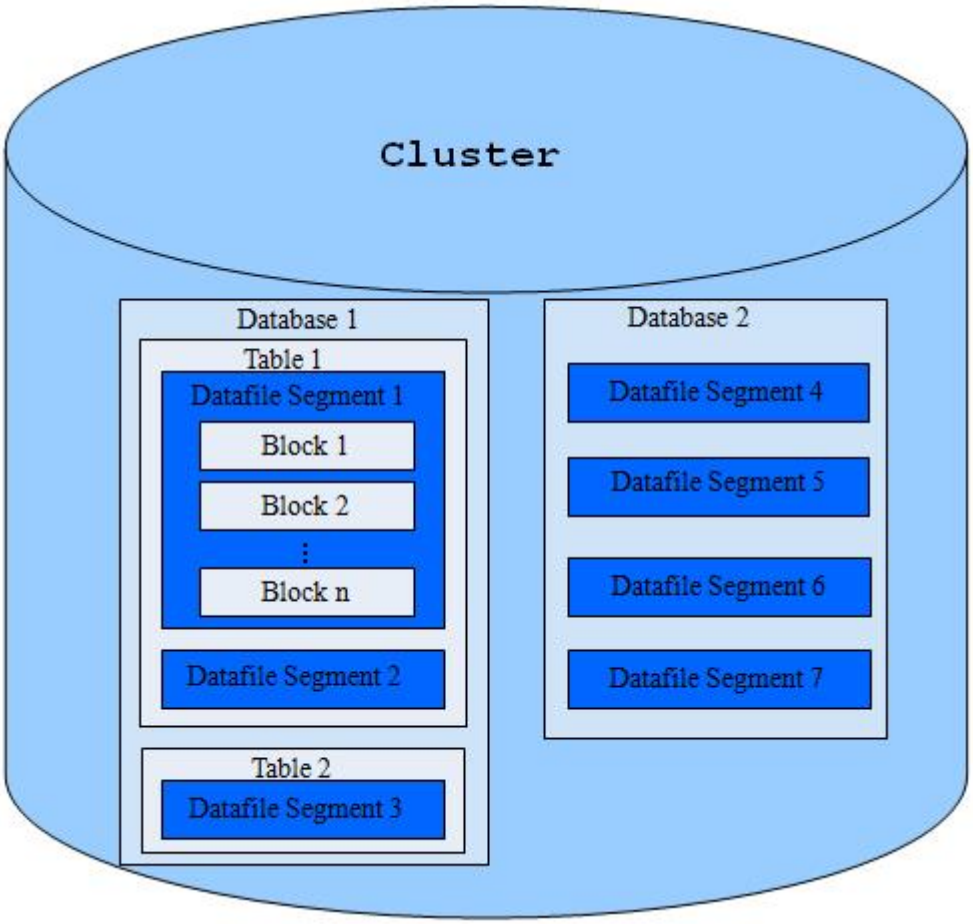
名称	描述
GTM	全局事务管理器（Global Transaction Manager），负责生成和维护全局事务ID、事务快照、时间戳等全局唯一的信息。
WLM	工作负载管理器（Workload Manager）。控制系统资源的分配，防止过量业务负载对系统的冲击而导致业务拥塞和系统崩溃。
CN	协调节点（Coordinator Node）。负责接收来自应用的访问请求，并向客户端返回执行结果；负责分解任务，并调度任务分片在各DN上并行执行。
DN	数据节点（Datanode）。负责存储业务数据（支持行存、列存、混合存储）、执行数据查询任务以及向CN返回执行结果。
Storage	服务器的本地存储资源，持久化存储数据。

集群的每个DN上负责存储数据，其存储介质也是磁盘。图2-3从逻辑上介绍了每个DN上都有哪些对象，以及这些对象之间的关系，其中：

- Database，即数据库，用于管理各类数据对象，各数据库间相互隔离。
- Datafile Segment，即数据文件，通常每张表只对应一个数据文件。如果某张表的数据大于1GB，则会分为多个数据文件存储。
- Table，即表，每张表只能属于一个数据库。
- Block，即数据块，是数据库管理的基本单位，默认大小为8KB。

数据有三种分布方式，可以在建表的时候指定：REPLICATION、ROUNDROBIN 、HASH。ROUNDRONIN只适用于外表

图 2-3 数据库逻辑结构图



2.2 高可靠事务处理

GaussDB(DWS)提供集群事务管理功能，此功能是集群HA、集群故障切换的基础，负责保证集群所有节点间事务的ACID特性，保证故障可恢复，以及恢复后满足数据的ACID（Atomicity，Consistency，Isolation，Durability）特性，并负责节点的并发控制。

故障恢复

为了在集群出现故障时尽可能地不中断服务，GaussDB(DWS)提供了高可靠机制。通过保护关键用户程序对外不间断提供服务，把因为硬件、软件和人为造成的故障对业务的影响程度降到最低，以保证业务的持续性。

- 硬件级高可靠：磁盘Raid、交换机堆叠及网卡bond、不间断电源UPS（Uninterruptible Power Supply）。
- 软件级高可靠：GaussDB(DWS)集群CN、GTM、DN等全方位HA。

事务管理

- 支持事务块，用户可以通过start transaction命令显式启动一个事务块。
- 支持单语句事务，用户不显式启动事务，则单条语句就是一个事务。
- 分布式事务管理。支持全局事务信息管理，包括gxid、snapshot、timestamp的管理，分布式事务状态管理，gxid溢出的处理。
- 分布式事务支持ACID特性。
- 支持分布式死锁预防，保证在出现死锁时自动解锁或者预防死锁。

2.3 查询高性能

GaussDB(DWS)通过如下功能来努力实现查询的高性能。

全并行的数据查询处理

GaussDB(DWS)是采用Shared-nothing架构的MPP系统，其由众多拥有独立且互不共享CPU、内存、存储等系统资源的逻辑节点组成。在这样的系统架构中，业务数据被分散存储在多个节点上，数据分析任务被推送到数据所在位置就近执行，并行地完成大规模的数据处理工作，实现对数据处理的快速响应。

GaussDB(DWS)后台还通过算子并行执行、指令在寄存器并行执行、及LLVM动态编译剪枝冗余的条件逻辑判断，助力数据查询性能提升。

行列混合存储

GaussDB(DWS)支持行存储和列存储两种存储模型，用户可以根据应用场景，建表的时候选择行存储还是列存储表。

行列混合存储引擎可以同时为用户提供更优的数据压缩比（列存）、更好的索引性能（列存）、更好的点更新和点查询（行存）性能。

列存下的数据压缩

对于非活跃的早期数据可以通过压缩来减少空间占用，降低采购和运维成本。

GaussDB(DWS)列存储压缩支持Delta Value Encoding、Dictionary、RLE 、LZ4、ZLIB等压缩算法，且能够根据数据特征自适应的选择压缩算法，平均压缩比7:1。压缩数据可直接访问，对业务透明，极大缩短历史数据访问的准备时间。

2.4 技术指标

表 2-2 8.0.x 版本技术指标

技术指标	最大值
数据容量	10PB

技术指标	最大值
集群节点数	256
单表大小	1PB
单行数据大小	1GB
每条记录单个字段的大小	1GB
单表记录数	2^{55}
单表列数	1600
单表中的索引个数	无限制
单表索引包含列数	32
单表约束个数	无限制
并发连接数	分析型长事务60；交易型短事务5000。
分区表的分区个数	32768
分区表的单个分区大小	1PB
分区表的单个分区记录数	2^{55}

2.5 SQL 语法说明

SQL（Structured Query Lanuage），即结构化查询语言，是关系数据库的标准语言，SQL是一种通用的、功能极强的关系数据库语言。

GaussDB(DWS)支持大部分SQL标准。

相较SQL标准，GaussDB(DWS)主要的不兼容点如下：

- 不支持“可串行化”事务隔离级别；
- PREPARE语句不支持复杂的查询；
- 行级触发器不能与COPY一起工作；
- CREATE TABLE AS EXECUTE不支持；
- WHERE CURRENT OF不支持；
- 游标不支持MOVE BACKWARD ；
- FOREIGN DATA WRAPPER不支持；
- LISTEN、UNLISTEN和NOTIFY只在CN本地支持；
- SECURITY LABEL不支持；
- 表的分布列不能进行更新；

2.6 相关概念

实例

实例在GaussDB(DWS)中是运行在内存中的一组数据库进程，一个实例可以管理一个或多个数据库，这些数据库组成一个集簇。集簇是存储磁盘上的一个区域，这个区域在安装时初始化并由一个目录组成，所有数据都存储在这个目录中，这个目录被称为数据目录，使用initdb创建。理论上来说一个服务器上可以在不同的端口启动多个实例，但是GaussDB(DWS)一次只能管理一个实例，启动和停止都是依赖于具体的数据目录。以后由于兼容的需要不排除引入实例名这个概念的可能。

数据库

数据库用于管理各类数据对象，与其他数据库隔离。创建数据库时可以指定对应的表空间，如果不指定相应的表空间，相关的对象会默认保存在PG_DEFAULT空间中。数据库管理的对象可分布在多个表空间上。

模式

GaussDB(DWS)的模式是对数据库做一个逻辑分割。所有的数据库对象都建立在模式下面。GaussDB(DWS)的模式和用户是弱绑定的，所谓的弱绑定是指虽然创建用户的同时会自动创建一个同名模式，但用户也可以单独创建模式，并且为用户指定其他的模式。

用户和角色

GaussDB(DWS)使用用户和角色来控制对数据库的访问。根据角色自身的设置不同，一个角色可以看做是一个数据库用户，或者一组数据库用户。在GaussDB(DWS)中角色和用户之间的区别只在于角色默认是没有LOGIN权限的。在GaussDB(DWS)中一个用户唯一对应一个角色，不过可以使用角色叠加来更灵活地进行管理。

事务管理

在事务管理上，GaussDB(DWS)采取了MVCC（多版本并发控制）结合两阶段锁的方式，其特点是读写之间不阻塞。GaussDB(DWS)的MVCC没有将历史版本数据统一存放，而是和当前元组的版本放在了一起。GaussDB(DWS)没有回滚段的概念，但是为了定期清除历史版本数据GaussDB(DWS)引入了一个VACUUM进程。一般情况下用户不用关注它，除非要做性能调优。此外，GaussDB(DWS)是自动提交事务。

3 Teradata 和 Oracle 语法兼容性差异

GaussDB(DWS)支持Teradata和Oracle两种兼容模式，分别兼容Teradata语法和Oracle语法，不同兼容模式下的语法行为有一些差异。

表 3-1 兼容项差异

兼容项	Teradata兼容	Oracle兼容
数据类型date	只有年月日	date会转为timestamp，包含年月日时分秒
空串	区分空串和NULL	只有NULL
空串转int	转换为0	NULL
超长字符自动截断	支持（GUC参数td_compatible_truncation打开）	不支持
varchar + int运算	转为numeric + numeric计算	转为bigint + int计算
case和coalesce表达式	兼容TD行为，支持数字和字符串之间的类型转换，比如coalesce参数输入int和varchar类型，解析成varchar类型。	报错

4 管理数据库安全

4.1 管理用户及权限

4.1.1 默认权限机制

数据库对象创建后，进行对象创建的用户就是该对象的所有者。集群安装后的默认情况下，未开启**三权分立**，数据库系统管理员具有与对象所有者相同的权限。也就是说对象创建后，默认只有对象所有者或者系统管理员可以查询、修改和销毁对象，以及通过GRANT将对象的权限授予其他用户。

为使其他用户能够使用对象，必须向用户或包含该用户的角色授予必要的权限。

GaussDB(DWS)支持以下的权限：SELECT、INSERT、UPDATE、DELETE、TRUNCATE、REFERENCES、CREATE、CONNECT、EXECUTE和USAGE。不同的权限与不同的对象类型关联。有关各权限的详细信息，请参见GRANT。

要撤消已经授予的权限，可以使用REVOKE。对象所有者的权限（例如ALTER、DROP、GRANT和REVOKE）是隐式的，无法授予或撤消。即只要拥有对象就可以执行对象所有者的这些隐式权限。对象所有者可以撤消自己的普通权限，例如，使表对自己以及其他人只读。

系统表和系统视图要么只对系统管理员可见，要么对所有用户可见。标识了需要系统管理员权限的系统表和视图只有系统管理员可以查询。 有关信息，请参考**系统表和系统视图**。

数据库提供对象隔离的特性，对象隔离特性开启时，用户只能查看有权限访问的对象(表、视图、字段、函数)，系统管理员不受影响。有关信息，请参考ALTER DATABASE。

4.1.2 系统管理员

系统管理员是指具有SYSADMIN属性的帐户。集群安装后，默认情况下系统管理员具有与对象所有者相同的权限。

在启动GaussDB(DWS)集群时创建的用户 dbadmin是系统管理员。

要创建新的数据库管理员，请以管理员用户身份连接数据库，并使用带SYSADMIN选项的CREATE USER语句或 ALTER USER语句进行设置。

```
CREATE USER sysadmin WITH SYSADMIN password "Bigdata@123";
```

或者

```
ALTER USER joe SYSADMIN;
```

ALTER USER时，要求用户已存在。

4.1.3 三权分立

默认权限机制和**系统管理员**两节的描述基于的是集群创建之初的默认情况。从前面的介绍可以看出，默认情况下拥有SYSADMIN属性的系统管理员，具备系统最高权限。

在实际业务管理中，为了避免系统管理员拥有过度集中的权利带来高风险，可以设置三权分立，将系统管理员的权限分立给安全管理员和审计管理员。

三权分立后，系统管理员将不再具有CREATEROLE属性（安全管理员）和AUDITADMIN属性（审计管理员）能力。即不再拥有创建角色和用户的权限，并不再拥有查看和维护数据库审计日志的权限。关于CREATEROLE属性和AUDITADMIN属性的更多信息请参考CREATE ROLE。

三权分立后，系统管理员只会对自己作为所有者的对象有权限。

三权分立的设置办法请参考**设置三权分立**章节。

三权分立前的权限详情及三权分立后的权限变化，请分别参见**表4-1**和**表4-2**。

表 4-1 默认的用户权限

对象名称	系统管理员	安全管理员	审计管理员	普通用户
表空间	对表空间有创建、修改、删除、访问、分配操作的权限。	不具有对表空间进行创建、修改、删除、分配的权限，访问需要被赋权。		
表	对所有表有所有的权限。	仅对自己的表有所有的权限，对其他用户的表无权限。		
索引	可以在所有的表上建立索引。	仅可以在自己的表上建立索引。		
模式	对所有模式有所有的权限。	仅对自己的模式有所有的权限，对其他用户的模式无权限。		
函数	对所有的函数有所有的权限。	仅对自己的函数有所有的权限，对其他用户放在public这个公共模式下的函数有调用的权限，对其他用户放在其他模式下的函数无权限。		
自定义视图	对所有的视图有所有的权限。	仅对自己的视图有所有的权限，对其他用户的视图无权限。		
系统表和系统视图	可以查看所有系统表和视图。	只可以查看部分系统表和视图。详细请参见**系统表和系统视图**。		

表 4-2 三权分立较非三权分立权限变化说明

对象名称	系统管理员	安全管理员	审计管理员	普通用户
表空间	无变化	无变化。		
表	权限缩小。 只对自己的表有所有权限，对其他用户放在属于各自模式下的表无权限。	无变化。		
索引	权限缩小。 只可以在自己的表上建立索引。	无变化。		
模式	权限缩小。 只对自己的模式有所有的权限，对其他用户的模式无权限。	无变化。		
函数	权限缩小。 只对自己的函数有所有的权限，对其他用户放在属于各自模式下的函数无权限。	无变化。		
自定义视图	权限缩小。 只对自己的视图及其他用户放在public模式下的视图有所有的权限，对其他用户放在属于各自模式下的视图无权限。	无变化。		
系统表和系统视图	无变化。	无变化。	无变化。	无权查看任何系统表和视图。

4.1.4 用户

使用CREATE USER和ALTER USER可以创建和管理数据库用户。数据库集群包含一个或多个已命名数据库。用户和角色在整个集群范围内是共享的，但是其数据并不共享。即用户可以连接任何数据库，但当连接成功后，任何用户都只能访问连接请求里声明的那个数据库。

非三权分立下，GaussDB(DWS)用户帐户只能由系统管理员或拥有CREATEROLE属性的安全管理员创建和删除。三权分立时，用户帐户只能由安全管理员创建。

在用户登录GaussDB(DWS)时会对其进行身份验证。用户可以拥有数据库和数据库对象（例如表），并且可以向用户和角色授予对这些对象的权限以控制谁可以访问哪个对象。除系统管理员外，具有CREATEDB属性的用户可以创建数据库并授予对这些数据库的权限。

创建、修改和删除用户

- 要创建用户，请使用SQL语句CREATE USER。
  例如：创建用户joe，并设置用户拥有CREATEDB属性。

```
CREATE USER joe WITH CREATEDB PASSWORD "Bigdata@123";
CREATE USER
```

- 要创建系统管理员，请使用带有SYSADMIN选项的CREATE USER语句 。
- 要删除现有用户，请使用DROP USER。
- 要更改用户帐户（例如，重命名用户或更改密码），请使用ALTER USER。
- 要查看用户列表，请查询视图**PG_USER**：
  ```
  SELECT * FROM pg_user;
  ```
- 要查看用户属性，请查询系统表**PG_AUTHID**：
  ```
  SELECT * FROM pg_authid;
  ```

私有用户

对于有多个业务部门，各部门间使用不同的数据库用户进行业务操作，同时有一个同级的数据库维护部门使用数据库管理员进行维护操作的场景下，业务部门可能希望在未经授权的情况下，管理员用户只能对各部门的数据进行控制操作（DROP、ALTER、TRUNCATE），但是不能进行访问操作（INSERT、DELETE、UPDATE、SELECT、COPY）。即针对管理员用户，表对象的控制权和访问权要能够分离，提高普通用户数据安全性。

三权分立情况下，管理员对其他用户放在属于各自模式下的表无权限。但是，这种无权限包含了无控制权限，因此不能满足上面的诉求。为此，GaussDB(DWS)提供了私有用户方案。即在非三权分立模式下，创建具有INDEPENDENT属性的私有用户。

```
CREATE USER user_independent WITH INDEPENDENT IDENTIFIED BY "1234@abc";
```

针对该用户的对象，数据库管理员在未经其授权前，只能进行控制操作（DROP、ALTER、TRUNCATE），无权进行INSERT、DELETE、SELECT、UPDATE、COPY、GRANT、REVOKE、ALTER OWNER操作。

4.1.5 角色

角色是一组权限的集合。通过GRANT把角色授予用户后，用户即具有了角色的所有权限。推荐使用角色进行高效权限分配。例如，可以为设计、开发和维护人员创建不同的角色，将角色GRANT给用户后，再向每个角色中的用户授予其工作所需数据的差异权限。在角色级别授予或撤消权限时，这些更改将作用到角色下的所有成员。

GaussDB(DWS)提供了一个隐式定义的拥有所有角色的组PUBLIC，所有创建的用户和角色默认拥有PUBLIC所拥有的权限。关于PUBLIC默认拥有的权限请参考GRANT。要撤销或重新授予用户和角色对PUBLIC的权限， 可通过在GRANT和REVOKE指定关键字PUBLIC实现。

要查看所有角色，请查询系统表PG_ROLES：

```
SELECT * FROM PG_ROLES;
```

创建、修改和删除角色

非**三权分立**时，只有系统管理员和具有CREATEROLE属性的用户才能创建、修改或删除角色。三权分立下，只有具有CREATEROLE属性的用户才能创建、修改或删除角色。

- 要创建角色，请使用CREATE ROLE。
- 要在现有角色中添加或删除用户，请使用ALTER ROLE。
- 要删除角色，请使用DROP ROLE。DROP ROLE只会删除角色，并不会删除角色中的成员用户帐户。

4.1.6 Schema

Schema又称作模式。通过管理Schema，允许多个用户使用同一数据库而不相互干扰，可以将数据库对象组织成易于管理的逻辑组，同时便于将第三方应用添加到相应的Schema下而不引起冲突。

每个数据库包含一个或多个Schema。数据库中的每个Schema包含表和其他类型的对象。数据库创建初始，默认具有一个名为public的Schema，且所有用户都拥有此Schema的权限。可以通过Schema分组数据库对象。Schema类似于操作系统目录，但Schema不能嵌套。

相同的数据库对象名称可以应用在同一数据库的不同Schema中，而没有冲突。例如，a_schema和b_schema都可以包含名为mytable的表。具有所需权限的用户可以访问数据库的多个Schema中的对象。

在当前数据库中创建用户时，系统会在当前数据库中为新用户创建一个同名schema。

数据库对象是创建在数据库搜索路径中的第一个Schema内的。有关默认情况下的第一个Schema情况及如何变更Schema顺序等更多信息，请参见**搜索路径**。

创建、修改和删除 Schema

- 要创建Schema，请使用CREATE SCHEMA。任何用户都可以创建Schema。
- 要更改Schema名称或者所有者，请使用ALTER SCHEMA。只有Schema所有者可以更改Schema。
- 要删除Schema及其对象，请使用DROP SCHEMA。只有Schema所有者可以删除Schema。
- 要在Schema内创建表，请以schema_name.table_name格式创建表。不指定schema_name时，对象默认创建到**搜索路径**中的第一个Schema内。
- 要查看Schema所有者，请对系统表PG_NAMESPACE和PG_USER执行如下关联查询。语句中的schema_name请替换为实际要查找的Schema名称。
  ```
  SELECT s.nspname,u.usename AS nspowner FROM pg_namespace s, pg_user u WHERE nspname='schema_name' AND s.nspowner = u.usesysid;
  ```
- 要查看所有Schema的列表，请查询PG_NAMESPACE系统表。
  ```
  SELECT * FROM pg_namespace;
  ```
- 要查看属于某Schema下的表列表，请查询系统视图PG_TABLES。例如，以下查询会返回Schema PG_CATALOG中的表列表。
  ```
  SELECT distinct(tablename),schemaname from pg_tables where schemaname = 'pg_catalog';
  ```

搜索路径

搜索路径定义在**search_path**参数中，参数取值形式为采用逗号分隔的Schema名称列表。如果创建对象时未指定目标Schema，则将该对象会被添加到搜索路径中列出的第一个Schema中。当不同Schema中存在同名的对象时，查询对象未指定Schema的情况下，将从搜索路径中包含该对象的第一个Schema中返回对象。

- 要查看当前搜索路径，请使用SHOW。
  ```
  SHOW SEARCH_PATH;
   search_path
  ----------------
   "$user",public
  (1 row)
  ```
  search_path参数的默认值为："$user"，public。$user表示与当前会话用户名同名的Schema名，如果这样的模式不存在，$user将被忽略。所以默认情况下，用

户连接数据库后，如果数据库下存在同名Schema，则对象会添加到同名Schema下，否则对象被添加到Public Schema下。

- 要更改当前会话的默认Schema，请使用SET命令。

  执行如下命令将搜索路径设置为myschema、public，首先搜索myschema。

  ```
  SET SEARCH_PATH TO myschema, public;
  SET
  ```

4.1.7 用户权限设置

- 给用户直接授予某对象的权限，请使用GRANT。

  将Schema中的表或者视图对象授权给其他用户或角色时，需要将表或视图所属Schema的USAGE权限同时授予该用户或角色。否则用户或角色将只能看到这些对象的名字，并不能实际进行对象访问。

  例如，下面示例将Schema tpcds的权限赋给用户joe后，将表tpcds.web_returns的select权限赋给用户joe。

  ```
  GRANT USAGE ON SCHEMA tpcds TO joe;
  GRANT SELECT ON TABLE tpcds.web_returns to joe;
  ```

- 给用户指定角色，使用户继承角色所拥有的对象权限。

  a. 创建角色。

     新建一个角色lily，同时给角色指定系统权限CREATEDB：

     ```
     CREATE ROLE lily WITH CREATEDB PASSWORD "Bigdata@123";
     ```

  b. 给角色赋予对象权限，请使用GRANT。

     例如，将模式tpcds的权限赋给角色lily后，将表tpcds.web_returns的select权限赋给角色lily。

     ```
     GRANT USAGE ON SCHEMA tpcds TO lily;
     GRANT SELECT ON TABLE tpcds.web_returns to lily;
     ```

  c. 将角色的权限赋予用户。

     ```
     GRANT lily to joe;
     ```

     说明

     当将角色的权限赋予用户时，角色的属性并不会传递到用户。

- 回收用户权限，请使用REVOKE。

4.1.8 行级访问控制

行级访问控制特性将数据库访问控制精确到数据表行级别，使数据库达到行级访问控制的能力。不同用户执行相同的SQL查询操作，读取到的结果是不同的。

用户可以在数据表创建行访问控制(Row Level Security)策略，该策略是指针对特定数据库用户、特定SQL操作生效的表达式。当数据库用户对数据表访问时，若SQL满足数据表特定的Row Level Security策略，在查询优化阶段将满足条件的表达式，按照属性(PERMISSIVE | RESTRICTIVE)类型，通过AND或OR方式拼接，应用到执行计划上。

行级访问控制的目的是控制表中行级数据可见性，通过在数据表上预定义Filter，在查询优化阶段将满足条件的表达式应用到执行计划上，影响最终的执行结果。当前受影响的SQL语句包括SELECT，UPDATE，DELETE。

场景一：某表中汇总了不同用户的数据，但是不同用户只能查看自身相关的数据信息，不能查看其他用户的数据信息。

```
--创建用户alice, bob, peter
CREATE ROLE alice PASSWORD 'Gauss@123';
```

```
CREATE ROLE bob PASSWORD 'Gauss@123';
CREATE ROLE peter PASSWORD 'Gauss@123';

--创建表public.all_data，包含不同用户数据信息
CREATE TABLE public.all_data(id int, role varchar(100), data varchar(100));

--向数据表插入数据
INSERT INTO all_data VALUES(1, 'alice', 'alice data');
INSERT INTO all_data VALUES(2, 'bob', 'bob data');
INSERT INTO all_data VALUES(3, 'peter', 'peter data');

--将表all_data的读取权限赋予alice，bob和peter用户
GRANT SELECT ON all_data TO alice, bob, peter;

--打开行访问控制策略开关
ALTER TABLE all_data ENABLE ROW LEVEL SECURITY;

--创建行访问控制策略，当前用户只能查看用户自身的数据
CREATE ROW LEVEL SECURITY POLICY all_data_rls ON all_data USING(role = CURRENT_USER);

--查看表详细信息
 \d+ all_data
                              Table "public.all_data"
 Column |          Type          | Modifiers | Storage  | Stats target | Description
--------+------------------------+-----------+----------+--------------+-------------
 id     | integer                |           | plain    |              |
 role   | character varying(100) |           | extended |              |
 data   | character varying(100) |           | extended |              |
Row Level Security Policies:
    POLICY "all_data_rls"
      USING (((role)::name = "current_user"()))
Has OIDs: no
Distribute By: HASH(id)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, enable_rowsecurity=true

--切换至用户alice，执行SQL"SELECT * FROM all_data"
SET ROLE alice PASSWORD 'Gauss@123';
SELECT * FROM all_data;
 id | role  |    data
----+-------+------------
  1 | alice | alice data
(1 row)

EXPLAIN(COSTS OFF) SELECT * FROM all_data;
                          QUERY PLAN
------------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
         Filter: ((role)::name = 'alice'::name)
 Notice: This query is influenced by row level security feature
(5 rows)

--切换至用户peter，执行SQL"SELECT * FROM all_data"
SET ROLE peter PASSWORD 'Gauss@123';
SELECT * FROM all_data;
 id | role  |    data
----+-------+------------
  3 | peter | peter data
(1 row)

 EXPLAIN(COSTS OFF) SELECT * FROM all_data;
                          QUERY PLAN
------------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on all_data
         Filter: ((role)::name = 'peter'::name)
```

```
 Notice: This query is influenced by row level security feature
(5 rows)
```

4.1.9 数据脱敏

GaussDB(DWS)数据库权限访问控制可限制到表的列级别和行级别，在当前数据库权限系统下，若需要对某个用户隐藏敏感信息，只能收回表或列上的权限，对于用户越权查询只能提供报错的形式结束。

GaussDB(DWS)支持数据脱敏，以更高的可用性来保护敏感信息。用户可以在表对象上创建数据脱敏（data redaction）策略，策略是指针对特定数据库用户、特定数据对象、特定的SQL操作、特定的重编方式的一个组合。通过数据脱敏特性，可以对一些用户将敏感信息隐藏，以保证敏感信息安全。

数据脱敏的目的是为了控制表中某些数据列的隐私泄露，通过将列对象绑定Redac策略，在查询重写阶段将满足条件的重编策略应用到执行计划上，影响最终SELECT的执行结果。

为保证数据状态的安全，insert into select，update from，merge into等包含脱敏信息时，会报错提示拒绝执行。

GaussDB(DWS)提供函数接口用于创建、修改及删除脱敏策略，具体请参考数据脱敏函数。

场景示例：某表中汇总了个人信息，个人信息中很多属性属于敏感数据，对于敏感列创建脱敏策略。

创建用户alice，bob和peter。

```
CREATE ROLE alice PASSWORD 'Gauss@123';
CREATE ROLE bob PASSWORD 'Gauss@123';
CREATE ROLE peter PASSWORD 'Gauss@123';
```

用户alice在public模式下创建表customer用于记录所用用户信息（名字，工资）。

```
CREATE TABLE public.customer(id int, name varchar(100), salary int);
```

向数据表插入三个用户。

```
INSERT INTO customer VALUES(1, 'Michael', 10000);
INSERT INTO customer VALUES(2, 'bryant', 20000);
INSERT INTO customer VALUES(3, 'harden', 15000);
```

alice将表customer的读取权限赋予bob和peter。

```
GRANT SELECT ON customer TO bob, peter;
```

alice不希望bob和peter看到工资数据，为customer表添加脱敏策略。

```
SELECT
DBMS_REDACT.add_policy(object_schema=>'public',object_name=>'customer',policy_name=>'pol_1',column_
name=>'salary',expression=>'1=1');
```

切换用户至bob和peter，无法看到customer表的工资数据。

```
set role bob password 'Gauss@123';
SET
select * from customer;
id |  name   | salary
----+---------+--------
1 | Michael |
2 | bryant  |
3 | harden  |
(3 rows)
```

```
set role peter password 'Gauss@123';
SET
select * from customer;
id |  name   | salary
----+---------+--------
1 | Michael |
2 | bryant  |
3 | harden  |
(3 rows)
```

alice改变主意，不希望peter查看到工资数据，修改策略。

```
SELECT
DBMS_REDACT.alter_policy(object_schema=>'public',object_name=>'customer',policy_name=>'pol_1',action=
>'3',expression=>'SYS_CONTEXT(''USERENV'', ''CURRENT_USER'') = ''peter''');
```

切换至用户bob，可以查看到customer表的工资数据。

```
set role bob password 'Gauss@123';
SET
SELECT * FROM customer;
id |  name   | salary
----+---------+--------
1 | Michael |  10000
2 | bryant  |  20000
3 | harden  |  15000
(3 rows)
```

查看已有的脱敏策略和脱敏列信息。

```
select * from redaction_policies;
 object_schema | object_owner | object_name | policy_name |
expression                          | enable | policy_description
---------------+--------------+-------------+-------------
+-------------------------------------------------------------------------+--------+--------------------
 public        | yxw          | customer    | pol_1       | (sys_context('USERENV'::text, 'CURRENT_USER'::text) =
'peter'::text) | t      |
(1 row)

select * from redaction_columns;
 object_schema | object_owner | object_name | column_name | function_type | function_parameters |
regexp_pattern | regexp_replace_string | regexp_position | regexp_occurrence | regexp_match_parameter |
colum
n_description
---------------+--------------+-------------+-------------+---------------+---------------------+----------------
+----------------------+-----------------+-------------------+------------------------+------
--------------
 public        | yxw          | customer    | salary      |             1 |                     |                |
|               |                 |                        |
(1 row)
```

4.2 设置帐号安全策略说明

4.2.1 设置帐户安全锁定策略

背景信息

GaussDB(DWS)为帐户提供了自动锁定和解锁帐户、手动锁定和解锁异常帐户和删除不再使用的帐户等一系列的安全措施，保证数据安全。

自动锁定和解锁帐户

- 为了保证帐户安全，如果连接数据库时输入密码次数超过10次，系统将自动锁定该帐户。

- 帐户被锁定时间超过1天后，自动解锁。

手动锁定和解锁帐户

若管理员发现某帐户被盗、非法访问等异常情况，可手动锁定该帐户。

当管理员认为帐户恢复正常后，可手动解锁该帐户。

以手动锁定和解锁用户user_read为例，命令格式如下：

- 手动锁定
  ALTER USER *user_read* **ACCOUNT LOCK;**

  显示结果如下表示锁定成功：

  ```
  ALTER USER
  ```
- 手动解锁
  ALTER USER *user_read* **ACCOUNT UNLOCK;**

  显示结果如下表示解锁成功：

  ```
  ALTER USER
  ```

删除不再使用的帐户

当确认帐户不再使用，管理员可以删除帐户。该操作不可恢复。

当删除的用户正处于活动状态时，此会话状态不会立马断开，用户在会话状态断开后才会被完全删除。

以删除帐户user_read为例，命令格式如下：

DROP USER *user_read* **CASCADE;**

显示如下结果表示删除成功：

```
DROP USER
```

4.2.2 设置帐号有效期

创建新用户时，需要限制用户的操作期限（有效开始时间和有效结束时间）。

不在有效操作期内的用户需要重新设定帐号的有效操作期。

操作步骤

步骤1 创建用户并指定用户的有效开始时间和有效结束时间。

CREATE USER *joe* **WITH PASSWORD** '*Bigdata@123*' **VALID BEGIN** '*2015-10-10 08:00:00*' **VALID UNTIL** '*2016-10-10 08:00:00*';

显示如下信息表示创建用户成功。

```
CREATE USER
```

步骤2 用户已不在有效使用期内，需要重新设定帐号的有效期，这包括有效开始时间和有效结束时间。

ALTER USER *joe* **WITH VALID BEGIN** '*2016-11-10 08:00:00*' **VALID UNTIL** '*2017-11-10 08:00:00*';

显示如下信息表示重新设定成功。

```
ALTER USER
```

----结束

说明

若在“CREATE ROLE”或“ALTER ROLE”语法中不指定“VALID BEGIN”，表示不对用户的开始操作时间做限定；若不指定“VALID UNTIL”，表示不对用户的结束操作时间做限定；若两者均不指定，表示该用户一直有效。

4.2.3 设置用户密码

用户密码存储在系统表pg_authid中，为防止用户密码泄露，GaussDB(DWS)对用户密码进行加密存储。同时，限定了密码的安全策略。

- 密码复杂度

  帐户密码的复杂度要求如下：

  - 包含大写字母（A-Z）的个数为0~999，包含小写字母（a-z）的个数为0~999，包含数字（0-9）的个数为0~999，包含特殊字符的个数为0~999（特殊字符的列表请参见**表4-3**）。

    说明

    帐户密码至少包含上述四类字符中的三类。
  - 密码的最小长度6，默认值为8。
  - 密码的最大长度999，默认值为32。
  - 不能和用户名、用户名倒写相同。
  - 不能和当前密码、当前密码的倒写相同。
- 密码重用

  用户修改密码时，只有持续未使用天数超过60天的历史密码才可以被重新使用。
- 密码有效期限

  数据库用户的密码都有密码有效期（默认值为90天），当达到密码到期提醒天数（7天）时，系统会在用户登录数据库时提示用户修改密码。

  说明

  考虑到数据库使用特殊性及业务连续性，密码过期后用户还可以登录数据库，但是每次登录都会提示修改密码，直至修改为止。
- 密码修改
  - 在安装数据库时，会新建一个和初始化用户重名的操作系统用户，为了保证帐户安全，请定期修改操作系统用户的密码。

    以修改用户user1密码为例，命令格式如下：

    ```
    passwd user1
    ```

    根据提示信息完成修改密码操作。
  - 建议系统管理员和普通用户都要定期修改自己的帐户密码，避免帐户密码被非法窃取。

    以修改用户user1密码为例，使用管理员用户连接数据库并执行如下命令：

    ```
    ALTER USER user1 IDENTIFIED BY "1234@abc" REPLACE "5678@def";
    ```

    说明

    1234@abc、5678@def分别代表用户user1的新密码和原始密码，这些密码要符合规则，否则会执行失败。
  - 管理员可以修改自己的或者其他帐户的密码。通过修改其他帐户的密码，解决用户密码遗失所造成无法登录的问题。

以修改用户joe帐户密码为例，命令格式如下：

```
ALTER USER joe IDENTIFIED BY "abc@1234";
```

说明

- 系统管理员之间不允许互相修改对方密码。
- 系统管理员可以修改普通用户密码且不需要用户原密码。
- 系统管理员可以修改自己的密码但需要管理员原密码。

- 密码验证

  设置当前会话的用户和角色时，需要验证密码。如果输入密码与用户的存储密码不一致，则会报错。

  以设置用户joe为例，命令格式如下：

  ```
  SET ROLE joe PASSWORD "abc@1234";
  ```

  显示如下命令表示设置成功：

  ```
  SET ROLE
  ```

表 4-3 特殊字符

编号	字符	编号	字符	编号	字符	编号	字符
1	~	9	*	17	|	25	<
2	!	10	(	18	[	26	.
3	@	11	)	19	{	27	>
4	#	12	-	20	}	28	/
5	$	13	_	21	]	29	?
6	%	14	=	22	;	-	-
7	^	15	+	23	:	-	-
8	&	16	\	24	,	-	-

4.3 查看审计信息

前提条件

- 需要审计的审计项开关已开启。各审计项及其开启办法，请参考设置操作信息审计章节。
- 数据库正常运行，并且对数据库执行了一系列增、删、改、查操作，保证在查询时段内有审计结果产生。
- 数据库各个节点审计日志单独记录，如果使用了LVS负载管理机制，需根据LVS日志追溯到具体的执行节点，并直接连接该节点查询相关审计日志。

背景信息

- 只有拥有AUDITADMIN属性的用户才可以查看审计记录。有关数据库用户及创建用户的办法请参见**用户**。

- 审计查询命令是数据库提供的sql函数pg_query_audit，其原型为：
  pg_query_audit(timestamptz *startime*,**timestamptz** *endtime*,*audit_log*)

  参数startime和endtime分别表示审计记录的开始时间和结束时间，audit_log表示所查看的审计日志信息所在的物理文件路径，当不指定audit_log时，默认查看连接当前实例的审计日志信息。

  通过sql函数pgxc_query_audit可以查询所有CN节点的审计日志，其原型为：

  pgxc_query_audit(timestamptz *startime*,**timestamptz** *endtime*)

  说明

  startime和endtime的差值代表要查询的时间段，其有效值为从startime日期中的00:00:00开始到endtime日期中的23:59:59之间的任何值。请正确指定这两个参数，否则将查不到需要的审计信息。

操作步骤

步骤1 使用SQL客户端工具连接数据库。

步骤2 查询审计记录。

```
SELECT * FROM pg_query_audit('2015-07-15 08:00:00','2015-07-15 09:47:33');
```

查询结果如下：

```
         time          |     type      | result | username |    database    | client_conninfo |  object_name
|                       detail_info                | node_name |            thread_id            | local_port | remote_port
------------------------+---------------+--------+----------+----------------+-----------------+---------------
+-------------------------------------------------------------------+-----------+---------------------------------+------------
+-------------
 2015-07-15 08:03:55+08 | login_success | ok     | dbadmin | postgres       | gs_clean@::1    | postgres       |
login db(postgres) success,the current user is:dbadmin       | cn_5003   |
139808902997776@490233835920483 | 9000       | 55805
```

该条记录表明，用户dbadmin在2015-07-15 08:03:55+08登录数据库postgres。其中client_conninfo字段在log_hostname启动且IP连接时，字符@后显示反向DNS查找得到的主机名。

步骤3 查询所有CN节点审计记录。

```
SELECT * FROM pgxc_query_audit('2019-01-10 17:00:00','2019-01-10 19:00:00') where type =
'login_success' and username = 'user1';
```

查询结果如下：

```
         time          |     type      | result | username | database | client_conninfo | object_name |
detail_info                    |  node_name   |            thread_id            |
local_port | remote_port
------------------------+---------------+--------+----------+----------+-----------------+-------------
+-------------------------------------------------------+--------------+---------------------------------+-
-----------+-------------
 2019-01-10 18:06:08+08 | login_success | ok     | user1    | postgres | gsql@[local]    | postgres    | login
db(postgres) success,the current user is:user1 | coordinator1 | 139965149210368@600429968516954 |
17560      | null
 2019-01-10 18:06:22+08 | login_success | ok     | user1    | postgres | gsql@[local]    | postgres    | login
db(postgres) success,the current user is:user1 | coordinator1 | 139965149210368@600429982697548 |
17560      | null
 2019-01-10 18:06:54+08 | login_success | ok     | user1    | postgres | gsql@[local]    | postgres    | login
db(postgres) success,the current user is:user1 | coordinator2 | 140677694355200@600430014804280 |
17562      | null
(3 rows)
```

查询结果显示，用户user1在CN1和CN2的成功登录记录。

----结束

5 开发设计建议

5.1 开发设计建议概述

本开发设计建议约定数据库建模和数据库应用程序开发过程中，应当遵守的设计规范。依据这些规范进行建模，能够更好的契合GaussDB(DWS)的分布式处理架构，输出更高效的业务SQL代码。

本开发设计建议中所陈述的“建议”和“关注”含义如下：

- **建议**：用户应当遵守的设计规则。遵守这些规则，能够保证业务的高效运行；违反这些规则，将导致业务性能的大幅下降或某些业务逻辑错误。
- **关注**：在业务开发过程中客户需要注意的细则。用于标识容易导致客户理解错误的知识点（实际上遵守SQL标准的SQL行为），或者程序中潜在的客户不易感知的默认行为。

5.2 数据库对象命名

数据库对象命名需要满足约束：长度不超过63个字符，以字母或下划线开头，中间字符可以是字母、数字、下划线、$、#。

- 【建议】避免使用保留或者非保留关键字命名数据库对象。

  说明

  可以使用select * from pg_get_keywords()查询GaussDB(DWS)的关键字，或者在关键字章节中查看。

- 【建议】避免使用双引号括起来的字符串来定义数据库对象名称，除非需要限制数据库对象名称的大小写。数据库对象名称大小写敏感会使定位问题难度增加。
- 【建议】数据库对象命名风格务必保持统一。
  - 增量开发的业务系统或进行业务迁移的系统，建议遵守历史的命名风格。
  - 数据库对象名称由字母、数字和下划线组成，并且不能由数字开头。建议使用多个单词组成，以下划线分割。
  - 数据库对象名称最好能够望文知意，尽量避免使用自定义缩写（可以使用通用的术语缩写进行命名）。例如，在命名中可以使用具有实际业务含义的英文词汇或汉语拼音，但规则应该在集群范围内保持一致。

  - 变量名的关键是要具有描述性，即变量名称要有一定的意义，变量名要有前缀标明该变量的类型。
- 【建议】表对象的命名应该可以表征该表的重要特征。例如，在表对象命名时区分该表是普通表、临时表还是非日志表：
  - 普通表名按照数据集的业务含义命名。
  - 临时表以“tmp_+后缀”命名。
  - 非日志表以“ul_+后缀”命名。
  - 外表以“f_+后缀”命名。

5.3 数据库对象设计

5.3.1 Database 和 Schema 设计

GaussDB(DWS)中可以使用Database和Schema实现业务的隔离，区别在于Database的隔离更加彻底，各个Database之间共享资源极少，可实现连接隔离、权限隔离等，Database之间无法直接互访。Schema隔离的方式共用资源较多，可以通过grant与revoke语法便捷地控制不同用户对各Schema及其下属对象的权限。

- 从便捷性和资源共享效率上考虑，推荐使用Schema进行业务隔离。
- 建议系统管理员创建Schema和Database，再赋予相关用户对应的权限。

Database 设计建议

- 【建议】在实际业务中，根据需要创建新的Database，不建议直接使用集群默认的postgres数据库。
- 【建议】一个集群内，用户自定义的Database数量建议不超过3个。
- 【建议】为了适应全球化的需求，使数据库编码能够存储与表示绝大多数的字符，建议创建Database的时候使用UTF-8编码。
- 【关注】创建Database时，需要重点关注字符集编码(ENCODING)和兼容性(DBCOMPATIBILITY)两个配置项。GaussDB(DWS)支持Teradata和Oracle两种兼容模式，分别兼容Teradata语法和Oracle语法，不同兼容模式下的语法行为可能有一些差异。详细内容可参考**Teradata和Oracle语法兼容性差异**。
- 【关注】Database的owner默认拥有该Database下所有对象的所有权限，包括删除权限。删除权限影响较大，请谨慎使用。

Schema 设计建议

- 【关注】如果该用户不具有sysadmin权限或者不是该Schema的owner，要访问Schema下的对象，需要同时给用户赋予Schema的usage权限和对象的相应权限。
- 【关注】如果要在Schema下创建对象，需要授予操作用户该Schema的create权限。
- 【关注】Schema的owner默认拥有该Schema下对象的所有权限，包括删除权限。删除权限影响较大，请谨慎使用。

5.3.2 表设计

GaussDB(DWS)是分布式架构。数据分布在各个DN上。总体上讲，良好的表设计需要遵循以下原则：

- 【关注】将表数据均匀分布在各个DN上。数据均匀分布，可以防止数据在部分DN上集中分布，从而导致因存储倾斜造成集群有效容量下降。通过选择合适的分布列，可以避免数据倾斜。
- 【关注】将表的扫描压力均匀分散在各个DN上。避免扫描压力集中在部分DN上，而导致性能瓶颈。例如，在事实表上使用等值过滤条件时，将会导致扫描压力不均匀。
- 【关注】减少需要扫描的数据量。通过分区表的剪枝机制可以大幅减少数据的扫描量。
- 【关注】尽量减少随机I/O。通过聚簇/局部聚簇可以实现热数据的连续存储，将随机I/O转换为连续I/O，从而减少扫描的I/O代价。
- 【关注】尽量避免数据shuffle。shuffle，是指在物理上，数据从一个节点，传输到另一个节点。shuffle占用了大量宝贵的网络资源，减小不必要的数据shuffle，可以减少网络压力，使数据的处理本地化，提高集群的性能和可支持的并发度。通过对关联条件和分组条件的仔细设计，能够尽可能的减少不必要的数据shuffle。

选择存储方案

【建议】表的存储类型是表定义设计的第一步，客户业务类型是决定表的存储类型的主要因素，表存储类型的选择依据请参考**表5-1**。

表 5-1 表的存储类型及场景

存储类型	适用场景
行存	• 点查询(返回记录少，基于索引的简单查询)。 • 增、删、改操作较多的场景。
列存	• 统计分析类查询 (关联、分组操作较多的场景)。 • 即席查询（查询条件不确定，行存表扫描难以使用索引）。

选择分布方案

【建议】表的分布方式的选择一般遵循以下原则：

表 5-2 表的分布方式及使用场景

分布方式	描述	适用场景
Hash	表数据通过Hash方式散列到集群中的所有DN上。	数据量较大的事实表。
Replication	集群中每一个DN都有一份全量表数据。	维度表、数据量较小的事实表。

选择分区方案

当表中的数据量很大时，应当对表进行分区，一般需要遵循以下原则：

- 【建议】使用具有明显区间性的字段进行分区，比如日期、区域等字段上建立分区。
- 【建议】分区名称应当体现分区的数据特征。例如，关键字+区间特征。
- 【建议】将分区上边界的分区值定义为MAXVALUE，以防止可能出现的数据溢出。

典型的分区表定义如下：

```
CREATE TABLE staffS_p1
(
  staff_ID       NUMBER(6) not null,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(4,2),
  MANAGER_ID     NUMBER(6),
  section_ID     NUMBER(4)
)
PARTITION BY RANGE (HIRE_DATE)
(
   PARTITION HIRE_19950501 VALUES LESS THAN ('1995-05-01 00:00:00'),
   PARTITION HIRE_19950502 VALUES LESS THAN ('1995-05-02 00:00:00'),
   PARTITION HIRE_maxvalue VALUES LESS THAN (MAXVALUE)
);
```

选择分布键

Hash表的分布键选取至关重要，如果分布键选择不当，可能会导致数据倾斜，从而导致查询时，I/O负载集中在部分DN上，影响整体查询性能。因此，在确定Hash表的分布策略之后，需要对表数据进行倾斜性检查，以确保数据的均匀分布。分布键的选择一般需要遵循以下原则：

- 【建议】选作分布键的字段取值应该比较离散，以便数据能在各个DN上均匀分布。当单个字段无法满足离散条件时，可以考虑使用多个字段一起作为分布键。一般情况下，可以考虑选择表的主键作为分布键。例如，在人员信息表中选择证件号码作为分布键。
- 【建议】在满足第一条原则的情况下，尽量不要选取在查询中存在常量过滤条件的字段作为分布键。例如，在表dwcjk相关的查询中，字段zqdh存在常量过滤条件“zqdh='000001'”，那么就应当尽量不选择zqdh字段做为分布键。
- 【建议】在满足前两条原则的情况，尽量选择查询中的关联条件为分布键。当关联条件作为分布键时，Join任务的相关数据都分布在DN本地，将极大减少DN之间的数据流动代价。

5.3.3 字段设计

选择数据类型

在字段设计时，基于查询效率的考虑，一般遵循以下原则：

- 【建议】尽量使用高效数据类型。

  选择数值类型时，在满足业务精度的情况下，选择数据类型的优先级从高到低依次为整数、浮点数、NUMREIC。

- 【建议】当多个表存在逻辑关系时，表示同一含义的字段应该使用相同的数据类型。

- 【建议】对于字符串数据，建议使用变长字符串数据类型，并指定最大长度。请务必确保指定的最大长度大于需要存储的最大字符数，避免超出最大长度时出现字符截断现象。除非明确知道数据类型为固定长度字符串，否则，不建议使用CHAR(n)、BPCHAR(n)、NCHAR(n)、CHARACTER(n)。

  关于字符串类型的详细说明，请参见**常用字符串类型介绍**。

常用字符串类型介绍

在进行字段设计时，需要根据数据特征选择相应的数据类型。字符串类型在使用时比较容易混淆，下表列出了GaussDB(DWS)中常见的字符串类型：

表 5-3 常用字符串类型

名称	描述	最大存储空间
CHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
CHARACTER(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
NCHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
BPCHAR(n)	定长字符串，n描述了存储的字节长度，如果输入的字符串字节格式小于n，那么后面会自动用空字符补齐至n个字节。	10MB
VARCHAR(n)	变长字符串，n描述了可以存储的最大字节长度。	10MB
CHARACTER VARYING(n)	变长字符串，n描述了可以存储的最大字节长度；此数据类型和VARCHAR(n)是同一数据类型的不同表达形式。	10MB
VARCHAR2(n)	变长字符串，n描述了可以存储的最大字节长度，此数据类型是为兼容Oracle类型新增的，行为和VARCHAR(n)一致。	10MB

名称	描述	最大存储空间
NVARCHAR2(n)	变长字符串，n描述了可以存储的最大字符长度。	10MB
TEXT	不限长度(不超过1GB-8203字节)变长字符串。	1GB-8203字节

5.3.4 约束设计

DEFAULT 和 NULL 约束

- 【建议】如果能够从业务层面补全字段值，那么，就不建议使用DEFAULT约束，避免数据加载时产生不符合预期的结果。
- 【建议】给明确不存在NULL值的字段加上NOT NULL约束，优化器会在特定场景下对其进行自动优化。
- 【建议】给可以显式命名的约束显式命名。除了NOT NULL和DEFAULT约束外，其他约束都可以显式命名。

局部聚簇

Partial Cluster Key（局部聚簇，简称PCK）是列存表的一种局部聚簇技术，在GaussDB(DWS)中，使用PCK可以通过min/max稀疏索引实现事实表快速过滤扫描。PCK的选取遵循以下原则：

- 【关注】一张表上只能建立一个PCK，一个PCK可以包含多列，但是一般不建议超过2列。
- 【建议】在查询中的简单表达式过滤条件上创建PCK。这种过滤条件一般形如col op const，其中col为列名，op为操作符 =、>、>=、<=、<，const为常量值。
- 【建议】在满足上面条件的前提下，选择distinct值比较少的列上建PCK。

唯一约束

- 【关注】行存表支持唯一约束，而列存表不支持。
- 【建议】从命名上明确标识唯一约束，例如，命名为“UNI+构成字段”。

主键约束

- 【关注】行存表支持主键约束，而列存表不支持。
- 【建议】从命名上明确标识主键约束，例如，将主键约束命名为 “PK+字段名”。

检查约束

- 【关注】行存表支持检查约束，而列存表不支持。
- 【建议】从命名上明确标识检查约束，例如，将检查约束命名为 “CK+字段名”。

5.3.5 视图和关联表设计

视图设计

- 【建议】除非视图之间存在强依赖关系，否则不建议视图嵌套。
- 【建议】视图定义中尽量避免排序操作。

关联表设计

- 【建议】表之间的关联字段应该尽量少。
- 【建议】关联字段的数据类型应该保持一致。
- 【建议】关联字段在命名上，应该可以明显体现出关联关系。例如，采用同样名称来命名。

5.4 JDBC 配置

目前，GaussDB(DWS)相关的第三方工具都是通过JDBC进行连接的，此部分将介绍工具配置时的注意事项。

连接参数

- 【关注】第三方工具通过JDBC连接GaussDB(DWS)时，JDBC向GaussDB(DWS)发起连接请求，会默认添加以下配置参数，详见JDBC代码ConnectionFactoryImpl类的实现。

```
params = {
{ "user", user },
{ "database", database },
{ "client_encoding", "UTF8" },
{ "DateStyle", "ISO" },
{ "extra_float_digits", "2" },
{ "TimeZone",  createPostgresTimeZone() },
};
```

  这些参数可能会导致JDBC客户端的行为与gsql客户端的行为不一致，例如，Date数据显示方式、浮点数精度表示、timezone显示。

  如果实际期望和这些配置不符，建议在java连接设置代码中显式设定这些参数。

- 【建议】通过JDBC连接数据库时，应该保证下面两个时区设置一致：
  - JDBC客户端所在主机的时区。
  - GaussDB(DWS)集群所在主机的时区。

fetchsize

【关注】在应用程序中，如果需要使用fetchsize，必须关闭autocommit。开启autocommit，会令fetchsize配置失效。

autocommit

【建议】在JDBC向GaussDB(DWS)申请连接的代码中，建议显式打开autocommit开关。如果基于性能或者其它方面考虑，需要关闭autocommit时，需要应用程序自己来保证事务的提交。例如，在指定的业务SQL执行完之后做显式提交，特别是客户端退出之前务必保证所有的事务已经提交。

释放连接

【建议】推荐使用连接池限制应用程序的连接数。每执行一条SQL就连接一次数据库，是一种不好SQL的编写习惯。

【建议】在应用程序完成作业任务之后，应当及时断开和GaussDB(DWS)的连接，释放资源。建议在任务中设置session超时时间参数。

【建议】使用JDBC连接池，在将连接释放给连接池前，需要执行以下操作，重置会话环境。否则，可能会因为历史会话信息导致的对象冲突。

- 如果在连接中设置了GUC参数，那么在将连接归还连接池之前，必须使用“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”将连接的状态清空。
- 如果使用了临时表，那么在将连接归还连接池之前，必须将临时表删除。

CopyManager

【建议】在不使用ETL工具，数据入库实时性要求又比较高的情况下，建议在开发应用程序时，使用GaussDB(DWS) JDBC驱动的copyManger接口进行微批导入。

5.5 SQL 编写

DDL

- 【建议】在GaussDB(DWS)中，建议DDL（建表、comments等）操作统一执行，在批处理作业中尽量避免DDL操作。避免大量并发事务对性能的影响。
- 【建议】在非日志表（unlogged table）使用完后，立即执行数据清理（truncate）操作。因为在异常场景下，GaussDB(DWS)不保证非日志表(unlogged table)数据的安全性。
- 【建议】临时表和非日志表的存储方式建议和基表相同。当基表为行存（列存）表时，临时表和非日志表也推荐创建为行存（列存）表，可以避免行列混合关联带来的高计算代价。
- 【建议】索引字段的总长度不超过50字节。否则，索引大小会膨胀比较严重，带来较大的存储开销，同时索引性能也会下降。
- 【建议】不要使用DROP…CASCADE方式删除对象，除非已经明确对象间的依赖关系，以免误删。

数据加载和卸载

- 【建议】在insert语句中显式给出插入的字段列表。例如：
  ```
  INSERT INTO task(name,id,comment) VALUES ('task1','100','第100个任务');
  ```
- 【建议】在批量数据入库之后，或者数据增量达到一定阈值后，建议对表进行analyze操作，防止统计信息不准确而导致的执行计划劣化。
- 【建议】如果要清理表中的所有数据，建议使用truncate table方式，不要使用delete table方式。delete table方式删除性能差，且不会释放那些已经删除了的数据占用的磁盘空间。

类型转换

- 【建议】在需要数据类型转换（不同数据类型进行比较或转换）时，使用强制类型转换，以防隐式类型转换结果与预期不符。

- 【建议】在查询中，对常量要显式指定数据类型，不要试图依赖任何隐式的数据类型转换。
- 【关注】在ORACLE兼容模式下，在导入数据时，空字符串会自动转化为NULL。如果需要保留空字符串需要新建兼容性为TD的数据库。

查询操作

- 【建议】除ETL程序外，应该尽量避免向客户端返回大量结果集的操作。如果结果集过大，应考虑业务设计是否合理。
- 【建议】使用事务方式执行DDL和DML操作。例如，truncate table、update table、delete table、drop table等操作，一旦执行提交就无法恢复。对于这类操作，建议使用事务进行封装，必要时可以进行回滚。
- 【建议】在查询编写时，建议明确列出查询涉及的所有字段，不建议使用“SELECT *”这种写法。一方面基于性能考虑，尽量减少查询输出列；另一方面避免增删字段对前端业务兼容性的影响。
- 【建议】在访问表对象时带上schema前缀，可以避免因schema切换导致访问到非预期的表。
- 【建议】超过3张表或视图进行关联（特别是full join）时，执行代价难以估算。建议使用WITH TABLE AS语句创建中间临时表的方式增加SQL语句的可读性。
- 【建议】尽量避免使用笛卡尔积和Full join。这些操作会造成结果集的急剧膨胀，同时其执行性能也很低。
- 【关注】NULL值的比较只能使用IS NULL或者IS NOT NULL的方式判断，其他任何形式的逻辑判断都返回NULL。例如：NULL<>NULL、NULL=NULL和NULL<>1返回结果都是NULL，而不是期望的布尔值。
- 【关注】需要统计表中所有记录数时，不要使用count(col)来替代count(*)。count(*)会统计NULL值（真实行数），而count(col)不会统计。
- 【关注】在执行count(col)时，将“值为NULL”的记录行计数为0。在执行sum(col)时，当所有记录都为NULL时，最终将返回NULL；当不全为NULL时，“值为NULL”的记录行将被计数为0。
- 【关注】count(多个字段)时，多个字段名必须用圆括号括起来。例如，count((col1,col2,col3))。注意：通过多字段统计行数时，即使所选字段都为NULL，该行也被计数，效果与count(*)一致。
- 【关注】count(distinct col)用来计算该列不重复的非NULL的数量，NULL将不被计数。
- 【关注】count(distinct (col1,col2,...))用来统计多列的唯一值数量，当所有统计字段都为NULL时，也会被计数，同时这些记录被认为是相同的。
- 【关注】通过常量来过滤数据时，会根据常量的数据类型和匹配列的数据类型来查找用于这两种数据类型计算的函数，如果找不到对应的函数，则会相应的进行隐式数据类型转化，然后再根据转化后的数据类型查找用于转化后的数据类型计算的函数。

  ```
  SELECT * FROM test WHERE timestamp_col = 20000101;
  ```

  上述例子中，假设timestamp_col是timestamp类型，则会先查找支持timestamp类型和int类型（常量数字认为是int类型）“等于”运算的函数，如果找不到，则把timestamp_col和常量数字隐式类型转化成text类型来计算。
- 【建议】尽量避免标量子查询语句的出现。标量子查询是出现在select语句输出列表中的子查询，在下面例子中，下划线部分即为一个标量子查询语句：

  ```
  SELECT id, (SELECT COUNT(*) FROM films f WHERE f.did = s.id) FROM staffs_p1 s;
  ```

  标量子查询往往会导致查询性能急剧劣化，在应用开发过程中，应当根据业务逻辑，对标量子查询进行等价转换，将其写为表关联。

- 【建议】在where子句中，应当对过滤条件进行排序，把选择读较小（筛选出的记录数较少）的条件排在前面。
- 【建议】where子句中的过滤条件，尽量符合单边规则。即把字段名放在比较条件的一边，优化器在某些场景下会自动进行剪枝优化。形如col op expression，其中col为表的一个列，op为‘=’、‘>’的等比较操作符，expression为不含列名的表达式。例如，

  ```
  SELECT id, from_image_id, from_person_id, from_video_id FROM face_data WHERE
  current_timestamp(6) - time < '1 days'::interval;
  ```

  改写为：

  ```
  SELECT id, from_image_id, from_person_id, from_video_id FROM face_data where time >
  current_timestamp(6) - '1 days'::interval;
  ```

- 【建议】尽量避免不必要的排序操作。排序需要耗费大量的内存及CPU，如果业务逻辑许可，可以组合使用order by和limit，减小资源开销。GaussDB(DWS)默认按照ASC & NULL LAST进行排序。
- 【建议】使用ORDER BY子句进行排序时，显式指定排序方式（ASC/DESC），NULL的排序方式（NULL FIRST/NULL LAST）。
- 【建议】不要单独依赖limit子句返回特定顺序的结果集。如果部分特定结果集，可以将ORDER BY子句与Limit子句组合使用，必要时也可以使用offset跳过特定结果。
- 【建议】在保障业务逻辑准确的情况下，建议尽量使用UNION ALL来代替UNION。
- 【建议】如果过滤条件只有OR表达式，可以将OR表达式转化为UNION ALL以提升性能。使用OR的SQL语句经常无法优化，导致执行速度慢。例如，将下面语句

  ```
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 300 AND inline=301) OR (cdp= 301 AND inline=302) OR (cdp= 302 AND inline=301);
  ```

  转换为：

  ```
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 300 AND inline=301)
  union all
  SELECT * FROM scdc.pub_menu
  WHERE (cdp= 301 AND inline=302)
  union all
  SELECT * FROM tablename
  WHERE (cdp= 302 AND inline=301)
  ```

- 【建议】当in(val1, val2, val3…)表达式中字段较多时，建议使用in (values (va11), (val2),(val3)…)语句进行替换。优化器会自动把in约束转换为非关联子查询，从而提升查询性能。
- 【建议】在关联字段不存在NULL值的情况下，使用(not) exist代替(not) in。例如，在下面查询语句中，当T1.C1列不存在NULL值时，可以先为T1.C1字段添加NOT NULL约束，再进行如下改写。

  ```
  SELECT * FROM T1 WHERE T1.C1 NOT IN (SELECT T2.C2 FROM T2);
  ```

  可以改写为：

  ```
  SELECT * FROM T1 WHERE NOT EXISTS (SELECT  * FROM T1,T2 WHERE T1.C1=T2.C2);
  ```

  说明

  - 如果不能保证T1.C1列的值为NOT NULL的情况下，就不能进行上述改写。
  - 如果T1.C1为子查询的输出，要根据业务逻辑确认其输出是否为NOT NULL。

- 【建议】通过游标进行翻页查询，而不是使用LIMIT OFFSET语法，避免多次执行带来的资源开销。游标必须在事务中使用，执行完后务必关闭游标并提交事务。

6 教程：使用 JDBC 或 ODBC 开发

6.1 开发规范

如果用户在APP的开发中，使用了连接池机制，那么需要遵循如下规范：

- 如果在连接中设置了GUC参数，那么在将连接归还连接池之前，必须使用“SET SESSION AUTHORIZATION DEFAULT;RESET ALL;”将连接的状态清空。
- 如果使用了临时表，那么在将连接归还连接池之前，必须将临时表删除。

否则，连接池里面的连接就是有状态的，会对用户后续使用连接池进行操作的正确性带来影响。

6.2 驱动下载

请参见**下载JDBC或ODBC驱动**。

6.3 使用 JDBC 开发

6.3.1 JDBC 开发过程

请参见**使用JDBC连接数据库**。

6.3.2 JDBC 包与驱动类

JDBC 包

从发布包中获取。包名为dws_8.0.x_jdbc_driver.zip。

解压后有两个JDBC的驱动jar包：

- gsjdbc4.jar：与PostgreSQL保持兼容的驱动包，其中类名、类结构与PostgreSQL驱动完全一致，曾经运行于PostgreSQL的应用程序可以直接移植到当前系统使用。
- gsjdbc200.jar：如果同一JVM进程内需要同时访问PostgreSQL及GaussDB(DWS)请使用此驱动包，它的主类名为“com.huawei.gauss200.jdbc.Driver”（即将

“org.postgresql”替换为“com.huawei.gauss200.jdbc”），数据库连接的URL前缀为“jdbc:gaussdb”，其余与gsjdbc4.jar相同。

驱动类

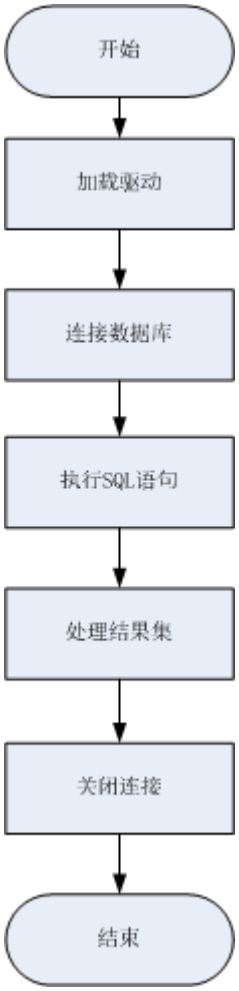
在创建数据库连接之前，需要加载数据库驱动类“org.postgresql.Driver”（对应包gsjdbc4.jar）或者“com.huawei.gauss200.jdbc.Driver”（对应gsjdbc200.jar）。

说明

由于GaussDB(DWS)在JDBC的使用上与PG的使用方法保持兼容，所以同时在同一进程内使用两个JDBC的驱动的时候，可能会类名冲突。

6.3.3 开发流程

图 6-1 采用 JDBC 开发应用程序的流程



6.3.4 加载驱动

在创建数据库连接之前，需要先加载数据库驱动程序。

加载驱动有两种方法：

- 在代码中创建连接之前任意位置隐含装载：
  Class.forName("org.postgresql.Driver");
- 在JVM启动时参数传递：java -Djdbc.drivers=org.postgresql.Driver jdbctest

  说明

  - 上述jdbctest为测试用例程序的名称。
  - 当使用gsjdbc200.jar时，上面的Driver类名相应修改为"com.huawei.gauss200.jdbc.Driver"

6.3.5 连接数据库

在创建数据库连接之后，才能使用它来执行SQL语句操作数据。

函数原型

JDBC提供了三个方法，用于创建数据库连接。

- DriverManager.getConnection(String url);
- DriverManager.getConnection(String url, Properties info);
- DriverManager.getConnection(String url, String user, String password);

参数

表 6-1 数据库连接参数

参数	描述
url	gsjdbc4.jar数据库连接描述符。格式如下： ● jdbc:postgresql:database ● jdbc:postgresql://host/database ● jdbc:postgresql://host:port/database ● jdbc:postgresql://host:port[,host:port][...]/database **说明** 使用gsjdbc200.jar时，将“jdbc:postgresql”修改为“jdbc:gaussdb” ● database为要连接的数据库名称。 ● host为数据库服务器名称或IP地址。 连接GaussDB(DWS)的机器与GaussDB(DWS)不在同一网段时，host指定的IP地址应为Manager界面上所设的mppdb.coo.cooListenIp2（应用访问IP）的取值。 由于安全原因，数据库CN禁止集群内部其他节点无认证接入。如果要在集群内部访问CN，请将JDBC程序部署在CN所在机器，host使用"127.0.0.1"。否则可能会出现“FATAL: Forbid remote connection with trust method!”错误。 建议业务系统单独部署在集群外部，否则可能会影响数据库运行性能。 ● port为数据库服务器端口。 ● 支持多ip端口配置形式，以","隔开，例如jdbc:postgresql://10.10.0.13:25308,10.10.0.14:25308/database 缺省情况下，会尝试连接到localhost的8000端口的database。

参数	描述
info	数据库连接属性。常用的属性如下： ● user：String类型。表示创建连接的数据库用户。 ● password：String类型。表示数据库用户的密码。 ● ssl：Boolean类型。表示是否使用SSL连接。 ● loggerLevel：string类型。为LogStream或LogWriter设置记录进DriverManager当前值的日志信息量。目前支持"OFF"、"DEBUG"和"TRACE"。 值为"DEBUG"时，表示只打印DEBUG级别以上的日志，将记录非常少的信息。值等于TRACE时，表示打印DEBUG和TRACE级别的日志，将产生详细的日志信息。默认值为OFF，表示不打印日志。 ● prepareThreshold：integer类型。用于确定在转换为服务器端的预备语句之前，要求执行方法PreparedStatement的次数。缺省值是5。 ● batchMode : boolean类型，用于确定是否使用batch模式连接。 ● fetchsize： integer类型，用于设置数据库链接所创建statement的默认fetchsize。 ● ApplicationName：string类型。应用名称，在不做设置时，缺省值为PostgreSQL JDBC Driver。 ● allowReadOnly:boolean类型，用于设置connection是否允许设置readonly模式，默认为false，若该参数不被设置为true，则执行connnection.setReadOnly不生效。 ● blobMode:string类型，用于设置setBinaryStream方法为不同的数据类型赋值，设置为on时表示为blob数据类型赋值，设置为off时表示为bytea数据类型赋值，默认为on。 ● connectionExtraInfo：Boolean类型。表示驱动是否上报当前驱动的部署路径、进程属主用户到数据库。 **说明** 取值范围：true或false，默认值为false。设置connectionExtraInfo为true，JDBC驱动会将当前驱动的部署路径、进程属主用户上报到数据库中，记录在connection_info参数（参见**connection_info**）里；同时可以在**PG_STAT_ACTIVITY**和**PGXC_STAT_ACTIVITY**中查询到。
user	数据库用户。
password	数据库用户的密码。

示例

此示例将演示如何基于GaussDB(DWS) 提供的JDBC接口开发应用程序。

说明

在完成以下示例前，需要先创建存储过程。

```
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
```

```
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。
//以下代码将获取数据库连接操作封装为一个接口，可通过给定用户名和密码来连接数据库。

public static Connection GetConnection(String username, String passwd)
  {
    //驱动类。
    String driver = "org.postgresql.Driver";
    //数据库连接描述符。
    String sourceURL = "jdbc:postgresql://10.10.0.13:8000/postgres?currentSchema=test";
    Connection conn = null;

    try
    {
        //加载驱动。
        Class.forName(driver);
    }
    catch( Exception e )
    {
        e.printStackTrace();
        return null;
    }

    try
    {
         //创建连接。
        conn = DriverManager.getConnection(sourceURL, username, passwd);
        System.out.println("Connection succeed!");
    }
    catch(Exception e)
    {
        e.printStackTrace();
        return null;
    }

    return conn;
  };
```

6.3.6 执行 SQL 语句

执行普通 SQL 语句

应用程序通过执行SQL语句来操作数据库的数据（不用传递参数的语句），需要按以下步骤执行：

步骤1 调用Connection的createStatement方法创建语句对象。

```
Statement stmt = con.createStatement();
```

步骤2 调用Statement的executeUpdate方法执行SQL语句。

```
int rc = stmt.executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name VARCHAR(32));");
```

说明

数据库中收到的一次执行请求（不在事务块中），如果含有多条语句，将会被打包成一个事务，事务块中不支持vacuum操作。如果其中有一个语句失败，那么整个请求都将会被回滚。

步骤3 关闭语句对象。

```
stmt.close();
```

----结束

执行预编译 SQL 语句

预编译语句是只编译和优化一次，然后可以通过设置不同的参数值多次使用。由于已经预先编译好，后续使用会减少执行时间。因此，如果多次执行一条语句，请选择使用预编译语句。可以按以下步骤执行：

步骤1 调用Connection的prepareStatement方法创建预编译语句对象。

```
PreparedStatement pstmt = con.prepareStatement("UPDATE customer_t1 SET c_customer_name = ? WHERE c_customer_sk = 1");
```

步骤2 调用PreparedStatement的setShort设置参数。

```
pstmt.setShort(1, (short)2);
```

步骤3 调用PreparedStatement的executeUpdate方法执行预编译SQL语句。

```
int rowcount = pstmt.executeUpdate();
```

步骤4 调用PreparedStatement的close方法关闭预编译语句对象。

```
pstmt.close();
```

----结束

调用存储过程

GaussDB(DWS)支持通过JDBC直接调用事先创建的存储过程，步骤如下：

步骤1 调用Connection的prepareCall方法创建调用语句对象。

```
CallableStatement cstmt = myConn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
```

步骤2 调用CallableStatement的setInt方法设置参数。

```
cstmt.setInt(2, 50);
cstmt.setInt(1, 20);
cstmt.setInt(3, 90);
```

步骤3 调用CallableStatement的registerOutParameter方法注册输出参数。

```
cstmt.registerOutParameter(4, Types.INTEGER);  //注册out类型的参数，类型为整型。
```

步骤4 调用CallableStatement的execute执行方法调用。

```
cstmt.execute();
```

步骤5 调用CallableStatement的getInt方法获取输出参数。

```
int out = cstmt.getInt(4);  //获取out参数
```

示例：

```
//在数据库中已创建了如下存储过程，它带有out参数。
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
```

```
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
```

步骤6 调用CallableStatement的close方法关闭调用语句。

```
cstmt.close();
```

说明

- 很多的数据库类如Connection、Statement和ResultSet都有close()方法，在使用完对象后应把它们关闭。要注意的是，Connection的关闭将间接关闭所有与它关联的Statement，Statement的关闭间接关闭了ResultSet。
- 一些JDBC驱动程序还提供命名参数的方法来设置参数。命名参数的方法允许根据名称而不是顺序来设置参数，若参数有默认值，则可以不用指定参数值就可以使用此参数的默认值。即使存储过程中参数的顺序发生了变更，也不必修改应用程序。目前GaussDB(DWS)数据库的JDBC驱动程序不支持此方法。
- GaussDB(DWS)数据库不支持带有输出参数的函数，也不支持存储过程和函数参数默认值。

----结束

须知

- 当游标作为存储过程的返回值时，如果使用JDBC调用该存储过程，返回的游标将不可用。
- 存储过程不能和普通SQL在同一条语句中执行。

执行批处理

用一条预处理语句处理多条相似的数据，数据库只创建一次执行计划，节省了语句的编译和优化时间。可以按如下步骤执行：

步骤1 调用Connection的prepareStatement方法创建预编译语句对象。

```
PreparedStatement pstmt = con.prepareStatement("INSERT INTO customer_t1 VALUES (?)");
```

步骤2 针对每条数据都要调用setShort设置参数，以及调用addBatch确认该条设置完毕。

```
pstmt.setShort(1, (short)2);
pstmt.addBatch();
```

步骤3 调用PreparedStatement的executeBatch方法执行批处理。

```
int[] rowcount = pstmt.executeBatch();
```

步骤4 调用PreparedStatement的close方法关闭预编译语句对象。

```
pstmt.close();
```

说明

在实际的批处理过程中，通常不终止批处理程序的执行，否则会降低数据库的性能。因此在批处理程序时，应该关闭自动提交功能，每几行提交一次。关闭自动提交功能的语句为：
conn.setAutoCommit(false);

----结束

6.3.7 处理结果集

设置结果集类型

不同类型的结果集有各自的应用场景，应用程序需要根据实际情况选择相应的结果集类型。在执行SQL语句过程中，都需要先创建相应的语句对象，而部分创建语句对象的方法提供了设置结果集类型的功能。具体的参数设置如表6-2所示。涉及的Connection的方法如下：

```
//创建一个Statement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
createStatement(int resultSetType, int resultSetConcurrency);

//创建一个PreparedStatement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
prepareStatement(String sql, int resultSetType, int resultSetConcurrency);

//创建一个CallableStatement对象，该对象将生成具有给定类型和并发性的ResultSet对象。
prepareCall(String sql, int resultSetType, int resultSetConcurrency);
```

表 6-2 结果集类型

参数	描述
resultSetType	表示结果集的类型，具体有三种类型： ● ResultSet.TYPE_FORWARD_ONLY：ResultSet只能向前移动。是缺省值。 ● ResultSet.TYPE_SCROLL_SENSITIVE：在修改后重新滚动到修改所在行，可以看到修改后的结果。 ● ResultSet.TYPE_SCROLL_INSENSITIVE：对可修改例程所做的编辑不进行显示。 **说明** 结果集从数据库中读取了数据之后，即使类型是ResultSet.TYPE_SCROLL_SENSITIVE，也不会看到由其他事务在这之后引起的改变。调用ResultSet的refreshRow()方法，可进入数据库并从其中取得当前游标所指记录的最新数据。
resultSetConcurrency	表示结果集的并发，具体有两种类型： ● ResultSet.CONCUR_READ_ONLY：如果不从结果集中的数据建立一个新的更新语句，不能对结果集中的数据进行更新。 ● ResultSet.CONCUR_UPDATEABLE：可改变的结果集。对于可滚动的结果集，可对结果集进行适当的改变。

在结果集中定位

ResultSet对象具有指向其当前数据行的光标。最初，光标被置于第一行之前。next方法将光标移动到下一行；因为该方法在ResultSet对象没有下一行时返回false，所以可以在while循环中使用它来迭代结果集。但对于可滚动的结果集，JDBC驱动程序提供更多的定位方法，使ResultSet指向特定的行。定位方法如表6-3所示。

表 6-3 在结果集中定位的方法

方法	描述
next()	把ResultSet向下移动一行。
previous()	把ResultSet向上移动一行。
beforeFirst()	把ResultSet定位到第一行之前。
afterLast()	把ResultSet定位到最后一行之后。
first()	把ResultSet定位到第一行。
last()	把ResultSet定位到最后一行。
absolute(int)	把ResultSet移动到参数指定的行数。
relative(int)	向前或者向后移动参数指定的行。

获取结果集中光标的位置

对于可滚动的结果集，可能会调用定位方法来改变光标的位置。JDBC驱动程序提供了获取结果集中光标所处位置的方法。获取光标位置的方法如表6-4所示。

表 6-4 获取结果集光标的位置

方法	描述
isFirst()	是否在一行。
isLast()	是否在最后一行。
isBeforeFirst()	是否在第一行之前。
isAfterLast()	是否在最后一行之后。
getRow()	获取当前在第几行。

获取结果集中的数据

ResultSet对象提供了丰富的方法，以获取结果集中的数据。获取数据常用的方法如表6-5所示，其他方法请参考JDK官方文档。

表 6-5 ResultSet 对象的常用方法

方法	描述
int getInt(int columnIndex)	按列标获取int型数据。
int getInt(String columnLabel)	按列名获取int型数据。
String getString(int columnIndex)	按列标获取String型数据。

方法	描述
String getString(String columnLabel)	按列名获取String型数据。
Date getDate(int columnIndex)	按列标获取Date型数据
Date getDate(String columnLabel)	按列名获取Date型数据。

6.3.8 关闭连接

在使用数据库连接完成相应的数据操作后，需要关闭数据库连接。

关闭数据库连接可以直接调用其close方法即可。如：**conn.close()**

6.3.9 示例：常用操作

示例 1

此示例将演示如何基于GaussDB(DWS)提供的JDBC接口开发应用程序。

在完成以下示例前，需要先创建存储过程。

```
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
//DBtest.java
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。
//演示基于JDBC开发的主要步骤，会涉及创建数据库、创建表、插入数据等。

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.CallableStatement;

public class DBTest {

 //创建数据库连接。
 public static Connection GetConnection(String username, String passwd) {
   String driver = "org.postgresql.Driver";
   String sourceURL = "jdbc:postgresql://localhost:8000/postgres";
   Connection conn = null;
   try {
     //加载数据库驱动。
     Class.forName(driver).newInstance();
   } catch (Exception e) {
     e.printStackTrace();
     return null;
   }

   try {
     //创建数据库连接。
```

```
    conn = DriverManager.getConnection(sourceURL, username, passwd);
    System.out.println("Connection succeed!");
  } catch (Exception e) {
    e.printStackTrace();
    return null;
  }

  return conn;
};

//执行普通SQL语句，创建customer_t1表。
public static void CreateTable(Connection conn) {
  Statement stmt = null;
  try {
    stmt = conn.createStatement();

    //执行普通SQL语句。
    int rc = stmt
        .executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name
VARCHAR(32));");

    stmt.close();
  } catch (SQLException e) {
    if (stmt != null) {
      try {
        stmt.close();
      } catch (SQLException e1) {
        e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }
}

//执行预处理语句，批量插入数据。
public static void BatchInsertData(Connection conn) {
  PreparedStatement pst = null;

  try {
    //生成预处理语句。
    pst = conn.prepareStatement("INSERT INTO customer_t1 VALUES (?,?)");
    for (int i = 0; i < 3; i++) {
      //添加参数。
      pst.setInt(1, i);
      pst.setString(2, "data " + i);
      pst.addBatch();
    }
    //执行批处理。
    pst.executeBatch();
    pst.close();
  } catch (SQLException e) {
    if (pst != null) {
      try {
        pst.close();
      } catch (SQLException e1) {
      e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }
}

//执行预编译语句，更新数据。
public static void ExecPreparedSQL(Connection conn) {
  PreparedStatement pstmt = null;
  try {
    pstmt = conn
        .prepareStatement("UPDATE customer_t1 SET c_customer_name = ? WHERE c_customer_sk = 1");
    pstmt.setString(1, "new Data");
```

```
    int rowcount = pstmt.executeUpdate();
    pstmt.close();
  } catch (SQLException e) {
    if (pstmt != null) {
      try {
        pstmt.close();
      } catch (SQLException e1) {
        e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }
 }


//执行存储过程。
 public static void ExecCallableSQL(Connection conn) {
   CallableStatement cstmt = null;
   try {

     cstmt=conn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
     cstmt.setInt(2, 50);
     cstmt.setInt(1, 20);
     cstmt.setInt(3, 90);
     cstmt.registerOutParameter(4, Types.INTEGER);  //注册out类型的参数，类型为整型。
     cstmt.execute();
     int out = cstmt.getInt(4);  //获取out参数
     System.out.println("The CallableStatment TESTPROC returns:"+out);
     cstmt.close();
   } catch (SQLException e) {
     if (cstmt != null) {
       try {
         cstmt.close();
       } catch (SQLException e1) {
         e1.printStackTrace();
       }
     }
     e.printStackTrace();
   }
 }


 /**
  * 主程序，逐步调用各静态方法。
  * @param args
 */
 public static void main(String[] args) {
   //创建数据库连接。
   Connection conn = GetConnection("tester", "Password1234");

   //创建表。
   CreateTable(conn);

   //批插数据。
   BatchInsertData(conn);

   //执行预编译语句，更新数据。
   ExecPreparedSQL(conn);

   //执行存储过程。
   ExecCallableSQL(conn);

   //关闭数据库连接。
   try {
     conn.close();
   } catch (SQLException e) {
     e.printStackTrace();
   }
```

```
 }

}
```

示例 2 客户端内存占用过多解决

此示例主要使用setFetchSize来调整客户端内存使用，它的原理是通过数据库游标来分批获取服务器端数据，但它会加大网络交互，可能会损失部分性能。

由于游标事务内有效，故需要先关闭自动提交。

```
// 关闭掉自动提交
conn.setAutoCommit(false);
Statement st = conn.createStatement();

// 打开游标，每次获取50行数据
st.setFetchSize(50);
ResultSet rs = st.executeQuery("SELECT * FROM mytable");
while (rs.next())
{
    System.out.print("a row was returned.");
}
rs.close();

// 关闭服务器游标。
st.setFetchSize(0);
rs = st.executeQuery("SELECT * FROM mytable");
while (rs.next())
{
    System.out.print("many rows were returned.");
}
rs.close();

// Close the statement.
st.close();
```

6.3.10 示例：重新执行应用 SQL

当主DN故障且40s未恢复时，GaussDB(DWS)会自动将对应的备DN升主，使集群正常运行。备升主期间正在运行的作业会失败；备升主后启动的作业不会再受影响。如果要做到DN主备切换过程中，上层业务不感知，可参考此示例构建业务层SQL重试机制。

在完成以下示例前，需要先创建存储过程。

```
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
```

```
 *
 *
 */

class ExitHandler extends Thread {
    private Statement cancel_stmt = null;

    public ExitHandler(Statement stmt) {
        super("Exit Handler");
        this.cancel_stmt = stmt;
    }
    public void run() {
        System.out.println("exit handle");
        try {
            this.cancel_stmt.cancel();
        } catch (SQLException e) {
            System.out.println("cancel query failed.");
            e.printStackTrace();
        }
    }
}

public class SQLRetry {
   //创建数据库连接。
   public static Connection GetConnection(String username, String passwd) {
     String driver = "org.postgresql.Driver";
     String sourceURL = "jdbc:postgresql://10.131.72.136:8000/postgres";
     Connection conn = null;
     try {
       //加载数据库驱动。
       Class.forName(driver).newInstance();
     } catch (Exception e) {
       e.printStackTrace();
       return null;
     }

     try {
       //创建数据库连接。
       conn = DriverManager.getConnection(sourceURL, username, passwd);
       System.out.println("Connection succeed!");
     } catch (Exception e) {
       e.printStackTrace();
       return null;
     }

     return conn;
}

   //执行普通SQL语句，创建jdbc_test1表。
   public static void CreateTable(Connection conn) {
     Statement stmt = null;
     try {
       stmt = conn.createStatement();

       // add ctrl+c handler
       Runtime.getRuntime().addShutdownHook(new ExitHandler(stmt));

       //执行普通SQL语句。?
       int rc2 = stmt
          .executeUpdate("DROP TABLE if exists jdbc_test1;");

       int rc1 = stmt
          .executeUpdate("CREATE TABLE jdbc_test1(col1 INTEGER, col2 VARCHAR(10));");

       stmt.close();
     } catch (SQLException e) {
       if (stmt != null) {
         try {
           stmt.close();
```

```
      } catch (SQLException e1) {
        e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }
}

//执行预处理语句，批量插入数据。
public static void BatchInsertData(Connection conn) {
  PreparedStatement pst = null;

  try {
    //生成预处理语句。
    pst = conn.prepareStatement("INSERT INTO jdbc_test1 VALUES (?,?)");
    for (int i = 0; i < 100; i++) {
     //添加参数。
      pst.setInt(1, i);
      pst.setString(2, "data " + i);
      pst.addBatch();
    }
    //执行批处理。
    pst.executeBatch();
    pst.close();
  } catch (SQLException e) {
    if (pst != null) {
      try {
        pst.close();
      } catch (SQLException e1) {
      e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }
}

//执行预编译语句，更新数据。
private static boolean QueryRedo(Connection conn){
  PreparedStatement pstmt = null;
  boolean retValue = false;
  try {
    pstmt = conn
        .prepareStatement("SELECT col1 FROM jdbc_test1 WHERE col2 = ?");

        pstmt.setString(1, "data 10");
        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {
            System.out.println("col1 = " + rs.getString("col1"));
        }
        rs.close();

    pstmt.close();
     retValue = true;
   } catch (SQLException e) {
    System.out.println("catch..... retValue " + retValue);
    if (pstmt != null) {
      try {
       pstmt.close();
     } catch (SQLException e1) {
       e1.printStackTrace();
      }
    }
    e.printStackTrace();
  }

   System.out.println("finesh.....");
  return retValue;
}
```

```
 //查询语句，执行失败重试，重试次数可配置。
 public static void ExecPreparedSQL(Connection conn) throws InterruptedException {
        int maxRetryTime = 50;
        int time = 0;
        String result = null;
        do {
            time++;
            try {
System.out.println("time:" + time);
boolean ret = QueryRedo(conn);
if(ret == false){
 System.out.println("retry, time:" + time);
 Thread.sleep(10000);
 QueryRedo(conn);
}
            } catch (Exception e) {
                e.printStackTrace();
            }
        } while (null == result && time < maxRetryTime);

 }

 /**
  * 主程序，逐步调用各静态方法。
  * @param args
 * @throws InterruptedException
 */
 public static void main(String[] args) throws InterruptedException {
   //创建数据库连接。
   Connection conn = GetConnection("testuser", "test@123");

   //创建表。
   CreateTable(conn);

   //批插数据。
   BatchInsertData(conn);

   //执行预编译语句，更新数据。
   ExecPreparedSQL(conn);

   //关闭数据库连接。
   try {
     conn.close();
   } catch (SQLException e) {
     e.printStackTrace();
   }

 }

}
```

6.3.11 示例：通过本地文件导入导出数据

在使用JAVA语言基于GaussDB(DWS)进行二次开发时，可以使用CopyManager接口，通过流方式，将数据库中的数据导出到本地文件或者将本地文件导入数据库中，文件格式支持CSV、TEXT等格式。

样例程序如下，执行时需要加载GaussDB(DWS) jdbc驱动。

在完成以下示例前，需要先创建存储过程。

```
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
```

```
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。
import java.sql.Connection;
import java.sql.DriverManager;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.sql.SQLException;
import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Copy{

    public static void main(String[] args)
    {
     String urls = new String("jdbc:postgresql://10.180.155.74:8000/postgres"); //数据库URL
     String username = new String("jack");            //用户名
     String password = new String("Gauss@123");             //密码
     String tablename = new String("migration_table"); //定义表信息
     String tablename1 = new String("migration_table_1"); //定义表信息
     String driver = "org.postgresql.Driver";
     Connection conn = null;

     try {
          Class.forName(driver);
          conn = DriverManager.getConnection(urls, username, password);
        } catch (ClassNotFoundException e) {
            e.printStackTrace(System.out);
        } catch (SQLException e) {
            e.printStackTrace(System.out);
        }

     // 将SELECT * FROM migration_table查询结果导出到本地文件d:/data.txt
     try {
    copyToFile(conn, "d:/data.txt", "(SELECT * FROM migration_table)");
  } catch (SQLException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
  } catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
  }
     //将d:/data.txt中的数据导入到migration_table_1中。
     try {
     copyFromFile(conn, "d:/data.txt", tablename1);
  } catch (SQLException e) {
 // TODO Auto-generated catch block
       e.printStackTrace();
} catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}

     // 将migration_table_1中的数据导出到本地文件d:/data1.txt
     try {
     copyToFile(conn, "d:/data1.txt", tablename1);
  } catch (SQLException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
  } catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
}
    }
```

```
public static void copyFromFile(Connection connection, String filePath, String tableName)
        throws SQLException, IOException {

    FileInputStream fileInputStream = null;

    try {
        CopyManager copyManager = new CopyManager((BaseConnection)connection);
        fileInputStream = new FileInputStream(filePath);
        copyManager.copyIn("COPY " + tableName + " FROM STDIN", fileInputStream);
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

public static void copyToFile(Connection connection, String filePath, String tableOrQuery)
        throws SQLException, IOException {

    FileOutputStream fileOutputStream = null;

    try {
        CopyManager copyManager = new CopyManager((BaseConnection)connection);
        fileOutputStream = new FileOutputStream(filePath);
        copyManager.copyOut("COPY " + tableOrQuery + " TO STDOUT", fileOutputStream);
    } finally {
        if (fileOutputStream != null) {
            try {
                fileOutputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

6.3.12 示例：从 MYSQL 向 GaussDB(DWS)进行数据迁移

下面示例演示如何通过CopyManager从mysql向GaussDB(DWS)进行数据迁移的过程。

在完成以下示例前，需要先创建存储过程。

```
create or replace procedure testproc
(
    psv_in1 in integer,
    psv_in2 in integer,
    psv_inout in out integer
)
as
begin
    psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
//以下用例以gsjdbc4.jar为例，如果要使用gsjdbc200.jar，请替换驱动类名（将代码中的“org.postgresql”替换成“com.huawei.gauss200.jdbc”）与连接URL串前缀（将“jdbc:postgresql”替换为“jdbc:gaussdb”）。
import java.io.StringReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Migration{

   public static void main(String[] args) {
      String url = new String("jdbc:postgresql://10.180.155.74:8000/postgres"); //数据库URL
      String user = new String("jack");            //mppdb用户名
      String pass = new String("Gauss@123");             //mppdb密码
      String tablename = new String("migration_table"); //定义表信息
      String delimiter = new String("|");              //定义分隔符
      String encoding = new String("UTF8");            //定义字符集
      String driver = "org.postgresql.Driver";
      StringBuffer buffer = new StringBuffer();       //定义存放格式  化数据的缓存

      try {
         //获取源数据库查询结果集
         ResultSet rs = getDataSet();

         //遍历结果集，逐行获取记录
         //将每条记录中各字段值，按指定分隔符分割，由换行符结束，拼成一个字符串
         //把拼成的字符串，添加到缓存buffer
         while (rs.next()) {
            buffer.append(rs.getString(1) + delimiter
                  + rs.getString(2) + delimiter
                  + rs.getString(3) + delimiter
                  + rs.getString(4)
                  + "\n");
         }
         rs.close();

         try {
            //建立目标数据库连接
            Class.forName(driver);
            Connection conn = DriverManager.getConnection(url, user, pass);
            BaseConnection baseConn = (BaseConnection) conn;
            baseConn.setAutoCommit(false);

            //初始化表信息
            String sql = "Copy " + tablename + " from STDIN DELIMITER " + "'" + delimiter + "'" + "
ENCODING " + "'" + encoding + "'";

            //提交缓存buffer中的数据
            CopyManager cp = new CopyManager(baseConn);
            StringReader reader = new StringReader(buffer.toString());
            cp.copyIn(sql, reader);
            baseConn.commit();
            reader.close();
            baseConn.close();
         } catch (ClassNotFoundException e) {
            e.printStackTrace(System.out);
         } catch (SQLException e) {
            e.printStackTrace(System.out);
         }

      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   //**********************************
   // 从源数据库返回查询结果集
   //**********************************
   private static ResultSet getDataSet() {
      ResultSet rs = null;
      try {
         Class.forName("com.mysql.jdbc.Driver").newInstance();
         Connection conn = DriverManager.getConnection("jdbc:mysql://10.119.179.227:3306/jack?
useSSL=false&allowPublicKeyRetrieval=true", "jack", "Gauss@123");
```

```
            Statement stmt = conn.createStatement();
            rs = stmt.executeQuery("select * from migration_table");
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return rs;
    }
}
```

6.4 JDBC 接口参考

JDBC接口是一套提供给用户的API方法，本节将对部分常用接口做具体描述，若涉及其他接口可参考JDK1.6（软件包）/JDBC4.0中相关内容。

6.4.1 java.sql.Connection

java.sql.Connection是数据库连接接口。

表 6-6 对 java.sql.Connection 接口的支持情况

方法名	返回值类型	支持JDBC 4
close()	void	Yes
commit()	void	Yes
createStatement()	Statement	Yes
getAutoCommit()	boolean	Yes
getClientInfo()	Properties	Yes
getClientInfo(String name)	String	Yes
getTransactionIsolation()	int	Yes
isClosed()	boolean	Yes
isReadOnly()	boolean	Yes
prepareStatement(String sql)	PreparedStatement	Yes
rollback()	void	Yes
setAutoCommit(boolean autoCommit)	void	Yes
setClientInfo(Properties properties)	void	Yes
setClientInfo(String name,String value)	void	Yes

须知

接口内部默认使用自动提交模式，若通过setAutoCommit(false)关闭自动提交，将会导致后面执行的语句都受到显式事务包裹，数据库中不支持事务中执行的语句不能在此模式下执行。

6.4.2 java.sql.CallableStatement

java.sql.CallableStatement是存储过程执行接口。

表 6-7 对 java.sql.CallableStatement 的支持情况

方法名	返回值类型	支持JDBC 4
registerOutParameter(int parameterIndex, int type)	void	Yes
wasNull()	boolean	Yes
getString(int parameterIndex)	String	Yes
getBoolean(int parameterIndex)	boolean	Yes
getByte(int parameterIndex)	byte	Yes
getShort(int parameterIndex)	short	Yes
getInt(int parameterIndex)	int	Yes
getLong(int parameterIndex)	long	Yes
getFloat(int parameterIndex)	float	Yes
getDouble(int parameterIndex)	double	Yes
getBigDecimal(int parameterIndex)	BigDecimal	Yes
getBytes(int parameterIndex)	byte[]	Yes
getDate(int parameterIndex)	Date	Yes
getTime(int parameterIndex)	Time	Yes
getTimestamp(int parameterIndex)	Timestamp	Yes
getObject(int parameterIndex)	Object	Yes

说明

- 不允许含有OUT参数的语句执行批量操作。
- 以下方法是从java.sql.Statement继承而来：close，execute，executeQuery，executeUpdate，getConnection，getResultSet，getUpdateCount，isClosed，setMaxRows , setFetchSize。
- 以下方法是从java.sql.PreparedStatement继承而来：addBatch，clearParameters，execute，executeQuery，executeUpdate，getMetaData，setBigDecimal，setBoolean，setByte，setBytes，setDate，setDouble，setFloat，setInt，setLong，setNull，setObject，setString，setTime，setTimestamp 。

6.4.3 java.sql.DatabaseMetaData

java.sql.DatabaseMetaData是数据库对象定义接口。

表 6-8 对 java.sql.DatabaseMetaData 的支持情况

方法名	返回值类型	支持JDBC 4
getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)	ResultSet	Yes
getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)	ResultSet	Yes
getTableTypes()	ResultSet	Yes
getUserName()	String	Yes
isReadOnly()	boolean	Yes
nullsAreSortedHigh()	boolean	Yes
nullsAreSortedLow()	boolean	Yes
nullsAreSortedAtStart()	boolean	Yes
nullsAreSortedAtEnd()	boolean	Yes
getDatabaseProductName()	String	Yes
getDatabaseProductVersion()	String	Yes
getDriverName()	String	Yes
getDriverVersion()	String	Yes
getDriverMajorVersion()	int	Yes
getDriverMinorVersion()	int	Yes
usesLocalFiles()	boolean	Yes

方法名	返回值类型	支持JDBC 4
usesLocalFilePerTable()	boolean	Yes
supportsMixedCaseIdentifiers()	boolean	Yes
storesUpperCaseIdentifiers()	boolean	Yes
storesLowerCaseIdentifiers()	boolean	Yes
supportsMixedCaseQuotedIdentifiers()	boolean	Yes
storesUpperCaseQuotedIdentifiers()	boolean	Yes
storesLowerCaseQuotedIdentifiers()	boolean	Yes
storesMixedCaseQuotedIdentifiers()	boolean	Yes
supportsAlterTableWithAddColumn()	boolean	Yes
supportsAlterTableWithDropColumn()	boolean	Yes
supportsColumnAliasing()	boolean	Yes
nullPlusNonNullIsNull()	boolean	Yes
supportsConvert()	boolean	Yes
supportsConvert(int fromType, int toType)	boolean	Yes
supportsTableCorrelationNames()	boolean	Yes
supportsDifferentTableCorrelationNames()	boolean	Yes
supportsExpressionsInOrderBy()	boolean	Yes
supportsOrderByUnrelated()	boolean	Yes
supportsGroupBy()	boolean	Yes
supportsGroupByUnrelated()	boolean	Yes
supportsGroupByBeyondSelect()	boolean	Yes
supportsLikeEscapeClause()	boolean	Yes
supportsMultipleResultSets()	boolean	Yes

方法名	返回值类型	支持JDBC 4
supportsMultipleTransactions()	boolean	Yes
supportsNonNullableColumns()	boolean	Yes
supportsMinimumSQLGrammar()	boolean	Yes
supportsCoreSQLGrammar()	boolean	Yes
supportsExtendedSQLGrammar()	boolean	Yes
supportsANSI92EntryLevelSQL()	boolean	Yes
supportsANSI92IntermediateSQL()	boolean	Yes
supportsANSI92FullSQL()	boolean	Yes
supportsIntegrityEnhancementFacility()	boolean	Yes
supportsOuterJoins()	boolean	Yes
supportsFullOuterJoins()	boolean	Yes
supportsLimitedOuterJoins()	boolean	Yes
isCatalogAtStart()	boolean	Yes
supportsSchemasInDataManipulation()	boolean	Yes
supportsSavepoints()	boolean	Yes
supportsResultSetHoldability(int holdability)	boolean	Yes
getResultSetHoldability()	int	Yes
getDatabaseMajorVersion()	int	Yes
getDatabaseMinorVersion()	int	Yes
getJDBCMajorVersion()	int	Yes
getJDBCMinorVersion()	int	Yes

6.4.4 java.sql.Driver

java.sql.Driver是数据库驱动接口。

表 6-9 对 java.sql.Driver 的支持情况

方法名	返回值类型	支持JDBC 4
acceptsURL(String url)	boolean	Yes
connect(String url, Properties info)	Connection	Yes
jdbcCompliant()	boolean	Yes
getMajorVersion()	int	Yes
getMinorVersion()	int	Yes

6.4.5 java.sql.PreparedStatement

java.sql.PreparedStatement是预处理语句接口。

表 6-10 对 java.sql.PreparedStatement 的支持情况

方法名	返回值类型	支持JDBC 4
clearParameters()	void	Yes
execute()	boolean	Yes
executeQuery()	ResultSet	Yes
excuteUpdate()	int	Yes
getMetaData()	ResultSetMetaData	Yes
setBoolean(int parameterIndex, boolean x)	void	Yes
setBigDecimal(int parameterIndex, BigDecimal x)	void	Yes
setByte(int parameterIndex, byte x)	void	Yes
setBytes(int parameterIndex, byte[] x)	void	Yes
setDate(int parameterIndex, Date x)	void	Yes
setDouble(int parameterIndex, double x)	void	Yes
setFloat(int parameterIndex, float x)	void	Yes

方法名	返回值类型	支持JDBC 4
setInt(int parameterIndex, int x)	void	Yes
setLong(int parameterIndex, long x)	void	Yes
setNString(int parameterIndex, String value)	void	Yes
setShort(int parameterIndex, short x)	void	Yes
setString(int parameterIndex, String x)	void	Yes
addBatch()	void	Yes
executeBatch()	int[]	Yes
clearBatch()	void	Yes

说明

- addBatch()、execute()必须在clearBatch()之后才能执行。
- 调用executeBatch()方法并不会清除batch。用户必须显式使用clearBatch()清除 。
- 在添加了一个batch的绑定变量后，用户若想重用这些值(再次添加一个batch)，无需再次使用set*()方法 。
- 以下方法是从java.sql.Statement继承而来：close，execute，executeQuery，executeUpdate，getConnection，getResultSet，getUpdateCount，isClosed，setMaxRows, setFetchSize。

6.4.6 java.sql.ResultSet

java.sql.ResultSet是执行结果集接口。

表 6-11 对 java.sql.ResultSet 的支持情况

方法名	返回值类型	支持JDBC 4
findColumn(String columnLabel)	int	Yes
getBigDecimal(int columnIndex)	BigDecimal	Yes
getBigDecimal(String columnLabel)	BigDecimal	Yes

方法名	返回值类型	支持JDBC 4
getBoolean(int columnIndex)	boolean	Yes
getBoolean(String columnLabel)	boolean	Yes
getByte(int columnIndex)	byte	Yes
getBytes(int columnIndex)	byte[]	Yes
getByte(String columnLabel)	byte	Yes
getBytes(String columnLabel)	byte[]	Yes
getDate(int columnIndex)	Date	Yes
getDate(String columnLabel)	Date	Yes
getDouble(int columnIndex)	double	Yes
getDouble(String columnLabel)	double	Yes
getFloat(int columnIndex)	float	Yes
getFloat(String columnLabel)	float	Yes
getInt(int columnIndex)	int	Yes
getInt(String columnLabel)	int	Yes
getLong(int columnIndex)	long	Yes
getLong(String columnLabel)	long	Yes
getShort(int columnIndex)	short	Yes
getShort(String columnLabel)	short	Yes
getString(int columnIndex)	String	Yes
getString(String columnLabel)	String	Yes
getTime(int columnIndex)	Time	Yes

方法名	返回值类型	支持JDBC 4
getTime(String columnLabel)	Time	Yes
getTimestamp(int columnIndex)	Timestamp	Yes
getTimestamp(String columnLabel)	Timestamp	Yes
isAfterLast()	boolean	Yes
isBeforeFirst()	boolean	Yes
isFirst()	boolean	Yes
next()	boolean	Yes

说明

- 一个Statement不能有多个处于“open”状态的ResultSet。
- 用于遍历结果集(ResultSet)的游标(Cursor)在被提交后不能保持“open”的状态。

6.4.7 java.sql.ResultSetMetaData

java.sql.ResultSetMetaData是对ResultSet对象相关信息的具体描述。

表 6-12 对 java.sql.ResultSetMetaData 的支持情况

方法名	返回值类型	支持JDBC 4
getColumnCount()	int	Yes
getColumnName(int column)	String	Yes
getColumnType(int column)	int	Yes
getColumnTypeName(int column)	String	Yes

6.4.8 java.sql.Statement

java.sql.Statement是SQL语句接口。

表 6-13 对 java.sql.Statement 的支持情况

方法名	返回值类型	支持JDBC 4
close()	void	Yes

方法名	返回值类型	支持JDBC 4
execute(String sql)	boolean	Yes
executeQuery(String sql)	ResultSet	Yes
executeUpdate(String sql)	int	Yes
getConnection()	Connection	Yes
getResultSet()	ResultSet	Yes
getQueryTimeout()	int	Yes
getUpdateCount()	int	Yes
isClosed()	boolean	Yes
setQueryTimeout(int seconds)	void	Yes
setFetchSize(int rows)	void	Yes
cancel()	void	Yes

说明

通过setFetchSize可以减少结果集在客户端的内存占用情况。它的原理是通过将结果集打包成游标，然后分段处理，所以会加大数据库与客户端的通信量，会有性能损耗。

由于数据库游标是事务内有效，所以，在设置setFetchSize的同时，需要将连接设置为非自动提交模式，setAutoCommit(false)。同时在业务数据需要持久化到数据库中时，在连接上执行提交操作。

6.4.9 javax.sql.ConnectionPoolDataSource

javax.sql.ConnectionPoolDataSource是数据源连接池接口。

表 6-14 对 javax.sql.ConnectionPoolDataSource 的支持情况

方法名	返回值类型	支持JDBC 4
getLoginTimeout()	int	Yes
getLogWriter()	PrintWriter	Yes
getPooledConnection()	PooledConnection	Yes
getPooledConnection(String user,String password)	PooledConnection	Yes
setLoginTimeout(int seconds)	void	Yes

方法名	返回值类型	支持JDBC 4
setLogWriter(PrintWriter out)	void	Yes

6.4.10 javax.sql.DataSource

javax.sql.DataSource是数据源接口。

表 6-15 对 javax.sql.DataSource 接口的支持情况

方法名	返回值类型	支持JDBC 4
getConneciton()	Connection	Yes
getConnection(String username,String password)	Connection	Yes
getLoginTimeout()	int	Yes
getLogWriter()	PrintWriter	Yes
setLoginTimeout(int seconds)	void	Yes
setLogWriter(PrintWriter out)	void	Yes

6.4.11 javax.sql.PooledConnection

javax.sql.PooledConnection是由连接池创建的连接接口。

表 6-16 对 javax.sql.PooledConnection 的支持情况

方法名	返回值类型	支持JDBC 4
addConnectionEventListener (ConnectionEventListener listener)	void	Yes
close()	void	Yes
getConnection()	Connection	Yes
removeConnectionEventListener (ConnectionEventListener listener)	void	Yes
addStatementEventListener (StatementEventListener listener)	void	Yes
removeStatementEventListener (StatementEventListener listener)	void	Yes

6.4.12 javax.naming.Context

javax.naming.Context是连接配置的上下文接口。

表 6-17 对 javax.naming.Context 的支持情况

方法名	返回值类型	支持JDBC 4
bind(Name name, Object obj)	void	Yes
bind(String name, Object obj)	void	Yes
lookup(Name name)	Object	Yes
lookup(String name)	Object	Yes
rebind(Name name, Object obj)	void	Yes
rebind(String name, Object obj)	void	Yes
rename(Name oldName, Name newName)	void	Yes
rename(String oldName, String newName)	void	Yes
unbind(Name name)	void	Yes
unbind(String name)	void	Yes

6.4.13 javax.naming.spi.InitialContextFactory

javax.naming.spi.InitialContextFactory是初始连接上下文工厂接口。

表 6-18 对 javax.naming.spi.InitialContextFactory 的支持情况

方法名	返回值类型	支持JDBC 4
getInitialContext(Hashtable<?,?> environment)	Context	Yes

6.4.14 CopyManager

CopyManager是GaussDB(DWS) JDBC驱动中提供的一个API接口类，用于批量向GaussDB(DWS)集群中导入数据。

CopyManager 的继承关系

CopyManager类位于org.postgresql.copy Package中，继承自java.lang.Object类，该类的声明如下：

```
public class CopyManager
extends Object
```

构造方法

public CopyManager(BaseConnection connection)

throws SQLException

常用方法

表 6-19 CopyManager 常用方法

返回值	方法	描述	throws
CopyIn	copyIn(String sql)	-	SQLException
long	copyIn(String sql, InputStream from)	使用COPY FROM STDIN从InputStream中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, InputStream from, int bufferSize)	使用COPY FROM STDIN从InputStream中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, Reader from)	使用COPY FROM STDIN从Reader中快速向数据库中的表加载数据。	SQLException,IOException
long	copyIn(String sql, Reader from, int bufferSize)	使用COPY FROM STDIN从Reader中快速向数据库中的表加载数据。	SQLException,IOException
CopyOut	copyOut(String sql)	-	SQLException
long	copyOut(String sql, OutputStream to)	将一个COPY TO STDOUT的结果集从数据库发送到OutputStream类中。	SQLException,IOException

返回值	方法	描述	throws
long	copyOut(String sql, Writer to)	将一个COPY TO STDOUT的结果集从数据库发送到Writer类中。	SQLException,IOException

6.5 使用 ODBC 开发

6.5.1 ODBC 包及依赖的库和头文件

Linux 下的 ODBC 包

从发布包中获取，包名为dws_8.0.x_odbc_driver_for_xxx_xxx.zip。Linux环境下，开发应用程序要用到unixODBC提供的头文件（sql.h、sqlext.h等）和库libodbc.so。这些头文件和库可从unixODBC-2.3.0的安装包中获得。

Windows 下的 ODBC 包

从发布包中获取，包名为dws_8.1.x_odbc_driver_for_windows.zip。Windows环境下，开发应用程序用到的相关头文件和库文件都由系统自带。

6.5.2 Linux 下配置数据源

将GaussDB(DWS)提供的ODBC DRIVER（psqlodbcw.so）配置到数据源中便可使用。配置数据源需要配置“odbc.ini”和“odbcinst.ini”两个文件（在编译安装unixODBC过程中生成且默认放在“/usr/local/etc”目录下），并在服务器端进行配置。

操作步骤

步骤1 获取unixODBC源码包。

获取参考地址：http://sourceforge.net/projects/unixodbc/files/unixODBC/2.3.0/unixODBC-2.3.0.tar.gz/download

步骤2 目前不支持unixODBC-2.2.1版本。以unixODBC-2.3.0版本为例，在客户端执行如下命令安装unixODBC。默认安装到“/usr/local”目录下，生成数据源文件到 “/usr/local/etc”目录下，库文件生成在“/usr/local/lib”目录。

```
tar zxvf unixODBC-2.3.0.tar.gz
cd unixODBC-2.3.0
#修改configure文件(如果不存在，那么请修改configure.ac)，找到LIB_VERSION
#将它的值修改为"1:0:0"，这样将编译出*.so.1的动态库，与psqlodbcw.so的依赖关系相同。
vim configure

./configure --enable-gui=no #如果要在TaiShan服务器上编译，请追加一个configure参数： --build=aarch64-unknown-linux-gnu
make
#安装可能需要root权限
make install
```

安装unixODBC。如果机器上已经安装了其他版本的unixODBC，可以直接覆盖安装。

步骤3 替换客户端GaussDB(DWS)驱动程序。

将dws_8.0.x_odbc_driver_for_xxx_xxx.zip解压

- 在“/dws_8.0.x_odbc_driver_for_xxx_xxx/odbc/lib”目录下得到“psqlodbcw.la”和“psqlodbcw.so”两个文件。
- 在“/dws_8.0.x_odbc_driver_for_xxx_xxx/lib”目录下得到lib库文件。

步骤4 配置数据源。

1. 配置ODBC驱动文件。

   在“/usr/local/etc/odbcinst.ini”文件中追加以下内容。

   ```
   [GaussMPP]
   Driver64=/usr/local/lib/psqlodbcw.so
   setup=/usr/local/lib/psqlodbcw.so
   ```

   odbcinst.ini文件中的配置参数说明如表6-20所示。

   表 6-20 odbcinst.ini 文件配置参数

参数	描述	示例
[DriverName]	驱动器名称，对应数据源DSN中的驱动名。	[DRIVER_N]
Driver64	驱动动态库的路径。	Driver64=/xxx/odbc/lib/psqlodbcw.so
setup	驱动安装路径，与Driver64中动态库的路径一致。	setup=/xxx/odbc/lib/psqlodbcw.so

2. 配置数据源文件。

   在“/usr/local/etc/odbc.ini ”文件中追加以下内容。

   ```
   [MPPODBC]
   Driver=GaussMPP
   Servername=10.10.0.13（数据库Server IP）
   Database=postgres （数据库名）
   Username=dbadmin （数据库用户名）
   Password= （数据库用户密码）
   Port=8000（数据库监听端口）
   Sslmode=allow
   ```

   odbc.ini文件配置参数说明如表6-21所示。

   表 6-21 odbc.ini 文件配置参数

参数	描述	示例
[DSN]	数据源的名称。	[MPPODBC]
Driver	驱动名，对应odbcinst.ini中的DriverName。	Driver=DRIVER_N
Servername	服务器的IP地址。	Servername=10.145.130.26
Database	要连接的数据库的名称。	Database=postgres

参数	描述	示例
Username	数据库用户名称。	Username=dbadmin
Password	数据库用户密码。	Password= **说明** ODBC驱动本身已经对内存密码进行过清理，以保证用户密码在连接后不会再在内存中保留。 但是如果配置了此参数，由于UnixODBC对数据源文件等进行缓存，可能导致密码长期保留在内存中。 推荐在应用程序连接时，将密码传递给相应API，而非写在数据源配置文件中。同时连接成功后，应当及时清理保存密码的内存段。
Port	服务器的端口号。	Port=8000
Sslmode	开启SSL模式	Sslmode=allow
UseServerSidePrepare	是否开启数据库端扩展查询协议。 可选值0或1，默认为1，表示打开扩展查询协议。	UseServerSidePrepare=1
UseBatchProtocol	是否开启批量查询协议（打开可提高DML性能）；可选值0或者1，默认为1。 当此值为0时，不使用批量查询协议（主要用于与早期数据库版本通信兼容）。 当此值为1，并且数据库support_batch_bind参数存在且为on时，将打开批量查询协议。	UseBatchProtocol=1
ConnectionExtraInfo	GUC参数connection_info（参见**connection_info**）中显示驱动部署路径和进程属主用户的开关。	ConnectionExtraInfo=1 **说明** 默认值为0。当设置为1时，ODBC驱动会将当前驱动的部署路径、进程属主用户上报到数据库中，记录在connection_info参数（参见**connection_info**）里；同时可以在**PG_STAT_ACTIVITY**和**PGXC_STAT_ACTIVITY**中查询到。

参数	描述	示例
ForExtensionConnector	ETL工具性能优化参数，可进行内存优化，降低对端的CN内存占用，避免因CN内存使用过多导致系统不稳定。 可选值0或者1，默认为0，表示不开启优化项。 请勿在数据库系统之外的其他业务中配置此参数，以免影响业务的正确性。	ForExtensionConnector=1

其中关于Sslmode的选项的允许值，具体信息见下表：

表 6-22 sslmode 的可选项及其描述

sslmode	是否会启用SSL加密	描述
disable	否	不使用SSL安全连接。
allow	可能	如果数据库服务器要求使用，则可以使用SSL安全加密连接，但不验证数据库服务器的真实性。
prefer	可能	如果数据库支持，那么首选使用SSL安全加密连接，但不验证数据库服务器的真实性。
require	是	必须使用SSL安全连接，但是只做了数据加密，而并不验证数据库服务器的真实性。
verify-ca	是	必须使用SSL安全连接，并且验证数据库是否具有可信证书机器签发的证书。
verify-full	是	必须使用SSL安全连接，在verify-ca的验证范围之外，同时验证数据库所在主机的主机名是否与证书内容一致。GaussDB(DWS)不支持此模式。

步骤5 SSL模式

如果需要使用SSL证书连接，那么请将GaussDB(DWS)安装包中的SSLCERT的证书包解压，在shell环境下，执行“source sslcert_env.sh”，即在**当前会话**完成证书的默认位置的部署。

或者手动声明如下环境变量，同时保证client.key*系列文件为600权限：

```
export PGSSLCERT="/YOUR/PATH/OF/client.crt" #请修改该路径到client.crt的绝对路径
export PGSSLKEY="/YOUR/PATH/OF/client.key"  #请修改该路径到client.key的绝对路径
```

同时将数据源中的Sslmode选项调整至“verify-ca”。

步骤6 将客户端所在主机的IP网段加入GaussDB(DWS)的安全组规则，确保客户端主机与GaussDB(DWS)网络互通。

步骤7 配置环境变量。

```
vim ~/.bashrc
```

在配置文件中追加以下内容。

```
export LD_LIBRARY_PATH=/usr/local/lib/:$LD_LIBRARY_PATH
export ODBCSYSINI=/usr/local/etc
export ODBCINI=/usr/local/etc/odbc.ini
```

步骤8 执行如下命令使设置生效。

```
source ~/.bashrc
```

----结束

测试数据源配置

执行isql -v GaussODBC(数据源名称)命令。

- 如果显示如下信息，表明配置正确，连接成功。

```
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

- 若显示ERROR信息，则表明配置错误。请检查上述配置是否正确。

常见问题处理

- [UnixODBC][Driver Manager]Can't open lib 'xxx/xxx/psqlodbcw.so' : file not found.

  此问题的可能原因：

  - odbcinst.ini文件中配置的路径不正确

    确认的方法：'ls'一下错误信息中的路径，以确保该psqlodbcw.so文件存在，同时具有执行权限。

  - psqlodbcw.so的依赖库不存在，或者不在系统环境变量中

    确认的办法：ldd一下错误信息中的路径，如果是缺少libodbc.so.1等UnixODBC的库，那么按照“操作步骤”中的方法重新配置UnixODBC，并确保它的安装路径下的lib目录添加到了LD_LIBRARY_PATH中；如果是缺少其他库，请将ODBC驱动包中的lib目录添加到LD_LIBRARY_PATH中。

- [UnixODBC]connect to server failed: no such file or directory

  此问题可能的原因：

  - 配置了错误的/不可达的数据库地址，或者端口

    请检查数据源配置中的Servername及Port配置项。

  - 服务器监听不正确

    如果确认Servername及Port配置正确，请根据“操作步骤”中数据库服务器的相关配置，确保数据库监听了合适的网卡及端口。

  - 防火墙及网闸设备

    请确认防火墙设置，将数据库的通信端口添加到可信端口中。

    如果有网闸设备，请确认一下相关的设置。

- [unixODBC]The password-stored method is not supported.

  此问题可能原因：

  数据源中未配置sslmode配置项。

  解决办法：

  请配置该选项至allow或以上选项。此配置的更多信息，见**表6-22**。

- Server common name "xxxx" does not match host name "xxxxx"

  此问题的原因：

  使用了SSL加密的“verify-full”选项，驱动程序会验证证书中的主机名与实际部署数据库的主机名是否一致。

  解决办法：

  碰到此问题可以使用“verify-ca”选项，不再校验主机名；或者重新生成一套与数据库所在主机名相同的CA证书。

- Driver's SQLAllocHandle on SQL_HANDLE_DBC failed

  此问题的可能原因：

  可执行文件（比如UnixODBC的isql，以下都以isql为例）与数据库驱动（psqlodbcw.so）依赖于不同的odbc的库版本：libodbc.so.1或者libodbc.so.2。此问题可以通过如下方式确认：

  ```
  ldd `which isql` | grep odbc
  ldd psqlodbcw.so | grep odbc
  ```

  这时，如果输出的libodbc.so最后的后缀数字不同或者指向不同的磁盘物理文件，那么基本就可以断定是此问题。isql与psqlodbcw.so都会要求加载libodbc.so，这时如果它们加载的是不同的物理文件，便会导致两套完全同名的函数列表，同时出现在同一个可见域里（UnixODBC的libodbc.so.*的函数导出列表完全一致），产生冲突，无法加载数据库驱动。

  解决办法：

  确定一个要使用的UnixODBC，然后卸载另外一个（比如卸载库版本号为.so.2的UnixODBC），然后将剩下的.so.1的库，新建一个同名但是后缀为.so.2的软链接，便可解决此问题。

- FATAL: Forbid remote connection with trust method!

  由于安全原因，数据库CN禁止集群内部其他节点无认证接入。

  如果要在集群内部访问CN，请将ODBC程序部署在CN所在机器，服务器地址使用"127.0.0.1"。建议业务系统单独部署在集群外部，否则可能会影响数据库运行性能。

- [unixODBC][Driver Manager]Invalid attribute value

  在使用SQL on other GaussDB功能时碰到此问题，有可能是unixODBC的版本并非推荐版本，建议通过“odbcinst --version”命令排查环境中的unixODBC版本。

- authentication method 10 not supported.

  使用开源客户端碰到此问题，可能原因：

  数据库中存储的口令校验只存储了SHA256格式哈希，而开源客户端只识别MD5校验，双方校验方法不匹配报错。

说明

- 数据库并不存储用户口令，只存储用户口令的哈希码。
- 早期版本（V100R002C80SPC300之前的版本）的数据库只存储了SHA256格式的哈希，并未存储MD5的哈希，所以无法使用MD5做用户口令校验。
- 新版本（V100R002C80SPC300及之后版本）的数据库当用户更新用户口令或者新建用户时，会同时存储两种格式的哈希码，这时将兼容开源的认证协议。
- 但是当老版本升级到新版本时，由于哈希的不可逆性，所以数据库无法还原用户口令，进而生成新格式的哈希，所以仍然只保留了SHA256格式的哈希，导致仍然无法使用MD5做口令认证。

要解决该问题，可以更新用户口令；或者新建一个用户，赋于同等权限，使用新用户连接数据库。

- unsupported frontend protocol 3.51: server supports 1.0 to 3.0

  目标数据库版本过低，或者目标数据库为开源数据库。请使用对应版本的数据库驱动连接目标数据库。

6.5.3 Windows 下配置数据源

Windows操作系统自带ODBC数据源管理器，无需用户手动安装管理器便可直接进行配置。

操作步骤

步骤1 替换客户端GaussDB(DWS)驱动程序

将GaussDB-A-8.0.0-Windows-Odbc.tar.gz解压后，根据需要，点击psqlodbc.msi（32位）或者psqlodbc_x64.msi（64 位）进行驱动安装。

步骤2 打开驱动管理器

在配置数据源时，请使用对应的驱动管理器（假设操作系统安装盘符为C:盘，如果是其他盘符，请对路径做相应修改）：

- **64**位操作系统上进行**32**位程序开发，安装**32**位驱动程序后，使用**32**位的驱动管理器：C:\Windows**SysWOW64**\odbcad32.exe

  请勿直接使用控制面板->管理工具->数据源(ODBC)。

  说明

  WoW64的全称是"Windows 32-bit on Windows 64-bit"，C:\Windows\SysWOW64\存放的是64位系统上的32位运行环境。而C:\Windows\System32\存放的是与操作系统一致的运行环境，具体的技术信息请查阅Windows的相关技术文档。

- **64**位操作系统上进行**64**位程序开发，安装**64**位驱动程序后，使用**64**位的驱动管理器：C:\Windows**System32**\odbcad32.exe

  请勿直接使用控制面板->管理工具->数据源(ODBC)。

- 32位操作系统请使用：C:\Windows\System32\odbcad32.exe

  或者点击计算机->控制面板->管理工具->数据源(ODBC)打开驱动管理器。

步骤3 配置数据源

在打开的驱动管理器上，选择用户DSN->添加->PostgreSQL Unicode（如果是64位驱动，将会有64位标识），然后进行配置：

须知

此界面上配置的用户名及密码信息，将会被记录在Windows注册表中，再次连接数据库时就不再需要输入认证信息。但是出于安全考虑，建议在单击"Save"按钮保存配置信息前，清空相关敏感信息；在使用ODBC的连接API时，再传入所需的用户名、密码信息。

步骤4 SSL模式

如果需要使用SSL证书连接，那么请将GaussDB(DWS)安装包中的SSLCERT的证书包解压，双击"sslcert_env.bat"文件，即可完成证书的默认位置的部署。

须知

该sslcert_env.bat为了保证证书环境的纯净，在%APPDATA%\postgresql目录存在时，会提示是否需要移除相关目录。如果有需要，请备份该目录中的文件。

或者手动将client.crt、client.key、client.key.cipher、client.key.rand文件放至%APPDATA%\postgresql(该目录需手动建立)目录下，并且将文件名中的client改为postgres，例如client.key修改为postgres.key；将cacert.pem文件放至%APPDATA%\postgresql目录，并更名为root.crt。

同时将步骤2中的设置窗口的“SSL Mode”选项调整至“verify-ca”。

表 6-23 sslmode 的可选项及其描述

sslmode	是否会启用SSL加密	描述
disable	否	不使用SSL安全连接。
allow	可能	如果数据库服务器要求使用，则可以使用SSL安全加密连接，但不验证数据库服务器的真实性。

sslmode	是否会启用SSL加密	描述
prefer	可能	如果数据库支持，那么首选使用SSL安全加密连接，但不验证数据库服务器的真实性。
require	是	必须使用SSL安全连接，但是只做了数据加密，而并不验证数据库服务器的真实性。
verify-ca	是	必须使用SSL安全连接，并且验证数据库是否具有可信证书机器签发的证书。
verify-full	是	必须使用SSL安全连接，在verify-ca的验证范围之外，同时验证数据库所在主机的主机名是否与证书内容一致。

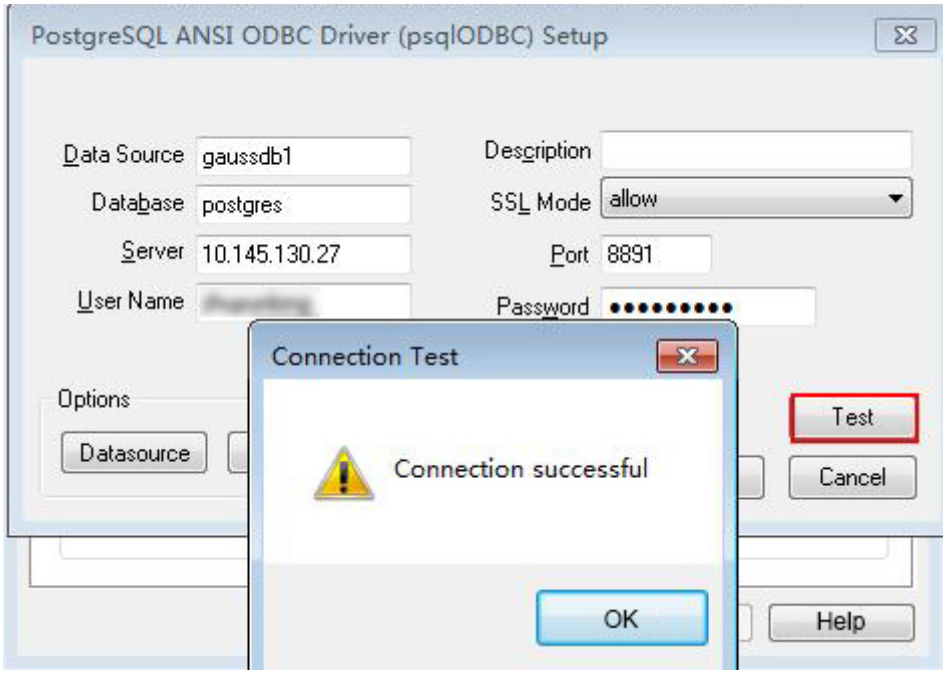
步骤5 将客户端所在主机的IP网段加入GaussDB(DWS)的安全组规则，确保客户端主机与GaussDB(DWS)网络互通。

----结束

测试数据源配置

点击Test进行测试。

- 如果显示如下，则表明配置正确，连接成功。



- 若显示ERROR信息，则表明配置错误。请检查上述配置是否正确。

常见问题处理

- Server common name "xxxx" does not match host name "xxxxx"

  此问题的原因是使用了SSL加密的“verify-full”选项，这时驱动程序会验证证书中的主机名与实际部署数据库的主机名是否一致。碰到此问题可以使用“verify-

ca”选项，不再校验主机名；或者重新生成一套与数据库所在主机名相同的CA证书。

- connect to server failed: no such file or directory

  此问题可能的原因：

  - 配置了错误的/不可达的数据库地址，或者端口

    请检查数据源配置中的Servername及Port配置项。

  - 服务器监听不正确

    如果确认Servername及Port配置正确，请根据“操作步骤”中数据库服务器的相关配置，确保数据库监听了合适的网卡及端口。

  - 防火墙及网闸设备

    请确认防火墙设置，将数据库的通信端口添加到可信端口中。

    如果有网闸设备，请确认一下相关的设置。

- 在指定的DSN中，驱动程序和应用程序之间的体系结构不匹配

  此问题可能的原因：在64位程序中使用了32位驱动，或者相反。

  C:\Windows\SysWOW64\odbcad32.exe：这是32位ODBC驱动管理器。

  C:\Windows\System32\odbcad32.exe：这是64位ODBC驱动管理器。

- The password-stored method is not supported.

  此问题可能原因：

  数据源中未配置sslmode配置项，请调整此项至allow或以上级别，允许SSL连接，此选项的更多说明，请见**表6-23**。

- authentication method 10 not supported.

  使用开源客户端碰到此问题，可能原因：

  数据库中存储的口令校验只存储了SHA256格式哈希，而开源客户端只识别MD5校验，双方校验方法不匹配报错。

  说明

  - 数据库并不存储用户口令，只存储用户口令的哈希码。
  - 早期版本（V100R002C80SPC300之前的版本）的数据库只存储了SHA256格式的哈希，并未存储MD5的哈希，所以无法使用MD5做用户口令校验。
  - 新版本（V100R002C80SPC300及之后版本）的数据库当用户更新用户口令或者新建用户时，会同时存储两种格式的哈希码，这时将兼容开源的认证协议。
  - 但是当老版本升级到新版本时，由于哈希的不可逆性，所以数据库无法还原用户口令，进而生成新格式的哈希，所以仍然只保留了SHA256格式的哈希，导致仍然无法使用MD5做口令认证。

  要解决该问题，可以更新用户口令；或者新建一个用户，赋于同等权限，使用新用户连接数据库。

- unsupported frontend protocol 3.51: server supports 1.0 to 3.0

  目标数据库版本过低，或者目标数据库为开源数据库。请使用对应版本的数据库驱动连接目标数据库。

6.5.4 ODBC 开发过程

请参见**使用ODBC连接数据库**。

6.5.5 ODBC 开发示例

常用功能示例代码

```
// 此示例演示如何通过ODBC方式获取GaussDB(DWS)中的数据。
// DBtest.c (compile with: libodbc.so)
#include <stdlib.h>
#include <stdio.h>
#include <sqlext.h>
#ifdef WIN32
#include <windows.h>
#endif
SQLHENV       V_OD_Env;        // Handle ODBC environment
SQLHSTMT      V_OD_hstmt;      // Handle statement
SQLHDBC       V_OD_hdbc;       // Handle connection
char          typename[100];
SQLINTEGER    value = 100;
SQLINTEGER    V_OD_erg,V_OD_buffer,V_OD_err,V_OD_id;
SQLLEN        V_StrLen_or_IndPtr;
int main(int argc,char *argv[])
{
    // 1. 申请环境句柄
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&V_OD_Env);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        printf("Error AllocHandle\n");
        exit(0);
    }
    // 2. 设置环境属性（版本信息）
    SQLSetEnvAttr(V_OD_Env, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0);
    // 3. 申请连接句柄
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_DBC, V_OD_Env, &V_OD_hdbc);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
        exit(0);
    }
    // 4. 设置连接属性
    SQLSetConnectAttr(V_OD_hdbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_ON, 0);
    // 5. 连接数据源，这里的“userName”与“password”分别表示连接数据库的用户名和用户密码，请根据实际情况修改。
    // 如果odbc.ini文件中已经配置了用户名密码，那么这里可以留空（""）；但是不建议这么做，因为一旦odbc.ini权限管理不善，将导致数据库用户密码泄露。
    V_OD_erg = SQLConnect(V_OD_hdbc, (SQLCHAR*) "gaussdb", SQL_NTS,
                          (SQLCHAR*) "userName", SQL_NTS,  (SQLCHAR*) "password", SQL_NTS);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
       printf("Error SQLConnect %d\n",V_OD_erg);
       SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
       exit(0);
    }
    printf("Connected !\n");
    // 6. 设置语句属性
    SQLSetStmtAttr(V_OD_hstmt,SQL_ATTR_QUERY_TIMEOUT,(SQLPOINTER *)3,0);
    // 7. 申请语句句柄
    SQLAllocHandle(SQL_HANDLE_STMT, V_OD_hdbc, &V_OD_hstmt);
    // 8. 直接执行SQL语句。
    SQLExecDirect(V_OD_hstmt,"drop table IF EXISTS customer_t1",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name VARCHAR(32));",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"insert into customer_t1 values(25,'li')",SQL_NTS);
    // 9. 准备执行
    SQLPrepare(V_OD_hstmt,"insert into customer_t1 values(?)",SQL_NTS);
    // 10. 绑定参数
    SQLBindParameter(V_OD_hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_INTEGER,0,0,
                     &value,0,NULL);
    // 11. 执行准备好的语句
    SQLExecute(V_OD_hstmt);
```

```
    SQLExecDirect(V_OD_hstmt,"select id from testtable",SQL_NTS);
    // 12. 获取结果集某一列的属性
    SQLColAttribute(V_OD_hstmt,
1,SQL_DESC_TYPE_NAME,typename,sizeof(typename),NULL,NULL);
    printf("SQLColAtrribute %s\n",typename);
    // 13. 绑定结果集
    SQLBindCol(V_OD_hstmt,1,SQL_C_SLONG, (SQLPOINTER)&V_OD_buffer,150,
            (SQLLEN *)&V_StrLen_or_IndPtr);
    // 14. 通过SQLFetch取结果集中数据
    V_OD_erg=SQLFetch(V_OD_hstmt);
    // 15. 通过SQLGetData获取并返回数据。
    while(V_OD_erg != SQL_NO_DATA)
    {
        SQLGetData(V_OD_hstmt,1,SQL_C_SLONG,(SQLPOINTER)&V_OD_id,0,NULL);
        printf("SQLGetData ----ID = %d\n",V_OD_id);
        V_OD_erg=SQLFetch(V_OD_hstmt);
    };
    printf("Done !\n");
    // 16. 断开数据源连接并释放句柄资源
    SQLFreeHandle(SQL_HANDLE_STMT,V_OD_hstmt);
    SQLDisconnect(V_OD_hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC,V_OD_hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
    return(0);
}
```

批量绑定示例代码

```
/************************************************************************
* 请在数据源中打开UseBatchProtocol，同时指定数据库中参数support_batch_bind
* 为on
* CHECK_ERROR的作用是检查并打印错误信息。
* 此示例将与用户交互式获取DSN、模拟的数据量，忽略的数据量，并将最终数据入库到test_odbc_batch_insert
中
************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sql.h>
#include <sqlext.h>
#include <string.h>

#include "util.c"

void Exec(SQLHDBC hdbc, SQLCHAR* sql)
{
    SQLRETURN retcode;                  // Return status
    SQLHSTMT hstmt = SQL_NULL_HSTMT;    // Statement handle
    SQLCHAR     loginfo[2048];

    // Allocate Statement Handle
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_STMT)",
                hstmt, SQL_HANDLE_STMT);

    // Prepare Statement
    retcode = SQLPrepare(hstmt, (SQLCHAR*) sql, SQL_NTS);
    sprintf((char*)loginfo, "SQLPrepare log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);

    retcode = SQLExecute(hstmt);
    sprintf((char*)loginfo, "SQLExecute stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);

    retcode = SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
    sprintf((char*)loginfo, "SQLFreeHandle stmt log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmt, SQL_HANDLE_STMT);
}

int main ()
```

```
{
    SQLHENV  henv  = SQL_NULL_HENV;
    SQLHDBC  hdbc  = SQL_NULL_HDBC;
    int      batchCount = 1000;
    SQLLEN   rowsCount = 0;
    int      ignoreCount = 0;

    SQLRETURN   retcode;
    SQLCHAR     dsn[1024] = {'\0'};
    SQLCHAR     loginfo[2048];

    // 交互获取数据源名称
    getStr("Please input your DSN", (char*)dsn, sizeof(dsn), 'N');
    // 交互获取批量绑定的数据量
    getInt("batchCount", &batchCount, 'N', 1);
    do
    {
        // 交互获取批量绑定的数据中，不要入库的数据量
        getInt("ignoreCount", &ignoreCount, 'N', 1);
        if (ignoreCount > batchCount)
        {
            printf("ignoreCount(%d) should be less than batchCount(%d)\n", ignoreCount, batchCount);
        }
    }while(ignoreCount > batchCount);

    retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_ENV)",
              henv, SQL_HANDLE_ENV);

    // Set ODBC Verion
    retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
                                (SQLPOINTER*)SQL_OV_ODBC3, 0);
    CHECK_ERROR(retcode, "SQLSetEnvAttr(SQL_ATTR_ODBC_VERSION)",
              henv, SQL_HANDLE_ENV);

    // Allocate Connection
    retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_DBC)",
              henv, SQL_HANDLE_DBC);

    // Set Login Timeout
    retcode = SQLSetConnectAttr(hdbc, SQL_LOGIN_TIMEOUT, (SQLPOINTER)5, 0);
    CHECK_ERROR(retcode, "SQLSetConnectAttr(SQL_LOGIN_TIMEOUT)",
              hdbc, SQL_HANDLE_DBC);

    // Set Auto Commit
    retcode = SQLSetConnectAttr(hdbc, SQL_ATTR_AUTOCOMMIT,
                                (SQLPOINTER)(1), 0);
    CHECK_ERROR(retcode, "SQLSetConnectAttr(SQL_ATTR_AUTOCOMMIT)",
              hdbc, SQL_HANDLE_DBC);

    // Connect to DSN
    sprintf(loginfo, "SQLConnect(DSN:%s)", dsn);
    retcode = SQLConnect(hdbc, (SQLCHAR*) dsn, SQL_NTS,
                         (SQLCHAR*) NULL, 0, NULL, 0);
    CHECK_ERROR(retcode, loginfo, hdbc, SQL_HANDLE_DBC);

    // init table info.
    Exec(hdbc, "drop table if exists test_odbc_batch_insert");
    Exec(hdbc, "create table test_odbc_batch_insert(id int primary key, col varchar2(50))");

    // 下面的代码根据用户输入的数据量，构造出将要入库的数据:
    {
        SQLRETURN retcode;
        SQLHSTMT hstmtinesrt = SQL_NULL_HSTMT;
        int          i;
        SQLCHAR      *sql = NULL;
        SQLINTEGER   *ids  = NULL;
        SQLCHAR      *cols = NULL;
```

```
    SQLLEN       *bufLenIds = NULL;
    SQLLEN       *bufLenCols = NULL;
    SQLUSMALLINT *operptr = NULL;
    SQLUSMALLINT *statusptr = NULL;
    SQLULEN      process = 0;

    // 这里是按列构造，每个字段的内存连续存放在一起。
    ids = (SQLINTEGER*)malloc(sizeof(ids[0]) * batchCount);
    cols = (SQLCHAR*)malloc(sizeof(cols[0]) * batchCount * 50);
    // 这里是每个字段中，每一行数据的内存长度。
    bufLenIds = (SQLLEN*)malloc(sizeof(bufLenIds[0]) * batchCount);
    bufLenCols = (SQLLEN*)malloc(sizeof(bufLenCols[0]) * batchCount);
    // 该行是否需要被处理，SQL_PARAM_IGNORE 或 SQL_PARAM_PROCEED
    operptr = (SQLUSMALLINT*)malloc(sizeof(operptr[0]) * batchCount);
    memset(operptr, 0, sizeof(operptr[0]) * batchCount);
    // 该行的处理结果。
    // 注：由于数据库中处理方式是同一语句隶属同一事务中，所以如果出错，那么待处理数据都将是出错的，并不会部分入库。
    statusptr = (SQLUSMALLINT*)malloc(sizeof(statusptr[0]) * batchCount);
    memset(statusptr, 88, sizeof(statusptr[0]) * batchCount);

    if (NULL == ids || NULL == cols || NULL == bufLenCols || NULL == bufLenIds)
    {
        fprintf(stderr, "FAILED:\tmalloc data memory failed\n");
        goto exit;
    }

    for (int i = 0; i < batchCount; i++)
    {
        ids[i] = i;
        sprintf(cols + 50 * i, "column test value %d", i);
        bufLenIds[i] = sizeof(ids[i]);
        bufLenCols[i] = strlen(cols + 50 * i);
        operptr[i] = (i < ignoreCount) ? SQL_PARAM_IGNORE : SQL_PARAM_PROCEED;
    }

    // Allocate Statement Handle
    retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmtinesrt);
    CHECK_ERROR(retcode, "SQLAllocHandle(SQL_HANDLE_STMT)",
                hstmtinesrt, SQL_HANDLE_STMT);

    // Prepare Statement
    sql = (SQLCHAR*)"insert into test_odbc_batch_insert values(?, ?)";
    retcode = SQLPrepare(hstmtinesrt, (SQLCHAR*) sql, SQL_NTS);
    sprintf((char*)loginfo, "SQLPrepare log: %s", (char*)sql);
    CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAMSET_SIZE, (SQLPOINTER)batchCount, sizeof(batchCount));
    CHECK_ERROR(retcode, "SQLSetStmtAttr", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLBindParameter(hstmtinesrt, 1, SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, sizeof(ids[0]), 0,&(ids[0]), 0, bufLenIds);
    CHECK_ERROR(retcode, "SQLBindParameter for id", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLBindParameter(hstmtinesrt, 2, SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR, 50, 50, cols, 50, bufLenCols);
    CHECK_ERROR(retcode, "SQLBindParameter for cols", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAMS_PROCESSED_PTR, (SQLPOINTER)&process, sizeof(process));
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAMS_PROCESSED_PTR", hstmtinesrt, SQL_HANDLE_STMT);

    retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAM_STATUS_PTR, (SQLPOINTER)statusptr, sizeof(statusptr[0]) * batchCount);
    CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAM_STATUS_PTR", hstmtinesrt, SQL_HANDLE_STMT);
```

```
        retcode = SQLSetStmtAttr(hstmtinesrt, SQL_ATTR_PARAM_OPERATION_PTR, (SQLPOINTER)operptr,
sizeof(operptr[0]) * batchCount);
        CHECK_ERROR(retcode, "SQLSetStmtAttr for SQL_ATTR_PARAM_OPERATION_PTR", hstmtinesrt,
SQL_HANDLE_STMT);

        retcode = SQLExecute(hstmtinesrt);
        sprintf((char*)loginfo, "SQLExecute stmt log: %s", (char*)sql);
        CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);

        retcode = SQLRowCount(hstmtinesrt, &rowsCount);
        CHECK_ERROR(retcode, "SQLRowCount execution", hstmtinesrt, SQL_HANDLE_STMT);

        if (rowsCount != (batchCount - ignoreCount))
        {
            sprintf(loginfo, "(batchCount - ignoreCount)(%d) != rowsCount(%d)", (batchCount - ignoreCount),
rowsCount);
            CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
        }
        else
        {
            sprintf(loginfo, "(batchCount - ignoreCount)(%d) == rowsCount(%d)", (batchCount - ignoreCount),
rowsCount);
            CHECK_ERROR(SQL_SUCCESS, loginfo, NULL, SQL_HANDLE_STMT);
        }

        if (rowsCount != process)
        {
            sprintf(loginfo, "process(%d) != rowsCount(%d)", process, rowsCount);
            CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
        }
        else
        {
            sprintf(loginfo, "process(%d) == rowsCount(%d)", process, rowsCount);
            CHECK_ERROR(SQL_SUCCESS, loginfo, NULL, SQL_HANDLE_STMT);
        }

        for (int i = 0; i < batchCount; i++)
        {
            if (i < ignoreCount)
            {
                if (statusptr[i] != SQL_PARAM_UNUSED)
                {
                    sprintf(loginfo, "statusptr[%d](%d) != SQL_PARAM_UNUSED", i, statusptr[i]);
                    CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
                }
            }
            else if (statusptr[i] != SQL_PARAM_SUCCESS)
            {
                sprintf(loginfo, "statusptr[%d](%d) != SQL_PARAM_SUCCESS", i, statusptr[i]);
                CHECK_ERROR(SQL_ERROR, loginfo, NULL, SQL_HANDLE_STMT);
            }
        }

        retcode = SQLFreeHandle(SQL_HANDLE_STMT, hstmtinesrt);
        sprintf((char*)loginfo, "SQLFreeHandle hstmtinesrt");
        CHECK_ERROR(retcode, loginfo, hstmtinesrt, SQL_HANDLE_STMT);
    }


exit:
    printf ("\nComplete.\n");

    // Connection
    if (hdbc != SQL_NULL_HDBC) {
        SQLDisconnect(hdbc);
        SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
    }

    // Environment
```

```
    if (henv != SQL_NULL_HENV)
        SQLFreeHandle(SQL_HANDLE_ENV, henv);

    return 0;
}
```

6.6 ODBC 接口参考

ODBC接口是一套提供给用户的API函数，本节将对部分常用接口做具体描述，若涉及其他接口可参考msdn（网址：https://msdn.microsoft.com/en-us/library/windows/desktop/ms714177(v=vs.85).aspx）中ODBC Programmer's Reference项的相关内容。

6.6.1 SQLAllocEnv

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocEnv已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

6.6.2 SQLAllocConnect

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocConnect已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

6.6.3 SQLAllocHandle

功能描述

分配环境、连接、语句或描述符的句柄，它替代了ODBC 2.x函数SQLAllocEnv、SQLAllocConnect及SQLAllocStmt。

原型

```
SQLRETURN SQLAllocHandle(SQLSMALLINT   HandleType,
                         SQLHANDLE     InputHandle,
                         SQLHANDLE     *OutputHandlePtr);
```

参数

表 6-24 SQLAllocHandle 参数

关键字	参数说明
HandleType	由SQLAllocHandle分配的句柄类型。必须为下列值之一： • SQL_HANDLE_ENV（环境句柄） • SQL_HANDLE_DBC（连接句柄） • SQL_HANDLE_STMT（语句句柄） • SQL_HANDLE_DESC（描述句柄） 申请句柄顺序为，先申请环境句柄，再申请连接句柄，最后申请语句句柄，后申请的句柄都要依赖它前面申请的句柄。

关键字	参数说明
InputHandle	将要分配的新句柄的类型。 • 如果HandleType为SQL_HANDLE_ENV，则这个值为SQL_NULL_HANDLE。 • 如果HandleType为SQL_HANDLE_DBC，则这一定是一个环境句柄。 • 如果HandleType为SQL_HANDLE_STMT或SQL_HANDLE_DESC，则它一定是一个连接句柄。
OutputHandlePtr	**输出参数**：一个缓冲区的指针，此缓冲区以新分配的数据结构存放返回的句柄。

返回值

- SQL_SUCCESS：表示调用正确，
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息，
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等、
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当分配的句柄并非环境句柄时，如果SQLAllocHandle返回的值为SQL_ERROR，则它会将OutputHandlePtr的值设置为SQL_NULL_HDBC、SQL_NULL_HSTMT或SQL_NULL_HDESC。之后，通过调用带有适当参数的**SQLGetDiagRec**，其中HandleType和Handle被设置为IntputHandle的值，可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.4 SQLAllocStmt

在ODBC 3.x版本中，ODBC 2.x的函数SQLAllocStmt已被SQLAllocHandle代替。有关详细信息请参阅**SQLAllocHandle**。

6.6.5 SQLBindCol

功能描述

将应用程序数据缓冲区绑定到结果集的列中。

原型

```
SQLRETURN SQLBindCol(SQLHSTMT       StatementHandle,
                     SQLUSMALLINT   ColumnNumber,
                     SQLSMALLINT    TargetType,
                     SQLPOINTER     TargetValuePtr,
                     SQLINTEGER     BufferLength,
                     SQLINTEGER     *StrLen_or_IndPtr);
```

参数

表 6-25 SQLBindCol 参数

关键字	参数说明
StatementHandle	语句句柄。
ColumnNumber	要绑定结果集的列号。起始列号为0，以递增的顺序计算列号，第0列是书签列。若未设置书签页，则起始列号为1。
TargetType	缓冲区中C数据类型的标识符。
TargetValuePtr	**输出参数**：指向与列绑定的数据缓冲区的指针。SQLFetch函数返回这个缓冲区中的数据。如果此参数为一个空指针，则StrLen_or_IndPtr是一个有效值。
BufferLength	TargetValuePtr指向缓冲区的长度，以字节为单位。
StrLen_or_IndPtr	**输出参数**：缓冲区的长度或指示器指针。若为空值，则未使用任何长度或指示器值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLBindCol返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用SQLGetDiagRec函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：示例

6.6.6 SQLBindParameter

功能描述

将一条SQL语句中的一个参数标志和一个缓冲区绑定起来。

原型

```
SQLRETURN SQLBindParameter(SQLHSTMT       StatementHandle,
                           SQLUSMALLINT   ParameterNumber,
                           SQLSMALLINT    InputOutputType,
                           SQLSMALLINT    ValuetType,
```

```
SQLSMALLINT    ParameterType,
SQLSMALLINT    ColumnSize,
SQLSMALLINT    DecimalDigits,
SQLPOINTER     ParameterValuePtr,
SQLINTEGER     BufferLength,
SQLINTEGER     *StrLen_or_IndPtr);
```

参数

表 6-26 SQLBindParameter

关键词	参数说明
StatementHandle	语句句柄。
ParameterNumber	参数序号，起始为1，依次递增。
InputOutputType	输入输出参数类型。
ValueType	参数的C数据类型。
ParameterType	参数的SQL数据类型。
ColumnSize	列的大小或相应参数标记的表达式。
DecimalDigits	列的十进制数字或相应参数标记的表达式。
ParameterValuePtr	指向存储参数数据缓冲区的指针。
BufferLength	ParameterValuePtr指向缓冲区的长度，以字节为单位。
StrLen_or_IndPtr	缓冲区的长度或指示器指针。若为空值，则未使用任何长度或指示器值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLBindCol返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.7 SQLColAttribute

功能描述

返回结果集中一列的描述符信息。

原型

```
SQLRETURN SQLColAttibute(SQLHSTMT        StatementHandle,
                         SQLUSMALLINT    ColumnNumber,
                         SQLUSMALLINT    FieldIdentifier,
                         SQLPOINTER      CharacterAtrributePtr,
                         SQLSMALLINT     BufferLength,
                         SQLSMALLINT     *StringLengthPtr,
                         SQLPOINTE       NumericAttributePtr);
```

参数

表 6-27 SQLColAttribute 参数

关键字	参数说明
StatementHandle	语句句柄。
ColumnNumber	要检索字段的列号，起始为1，依次递增。
FieldIdentifier	IRD中ColumnNumber行的字段。
CharacterAttributePtr	**输出参数**：一个缓冲区指针，返回FieldIdentifier字段值。
BufferLength	• 如果FieldIdentifier是一个ODBC定义的字段，而且CharacterAttributePtr指向一个字符串或二进制缓冲区，则此参数为该缓冲区的长度。 • 如果FieldIdentifier是一个ODBC定义的字段，而且CharacterAttributePtr指向一个整数，则会忽略该字段。
StringLengthPtr	**输出参数**：缓冲区指针，存放*CharacterAttributePtr中字符类型数据的字节总数，对于非字符类型，忽略BufferLength的值。
NumericAttributePtr	**输出参数**：指向一个整型缓冲区的指针，返回IRD中ColumnNumber行FieldIdentifier字段的值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLColAttribute返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用SQLGetDiagRec函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：示例

6.6.8 SQLConnect

功能描述

在驱动程序和数据源之间建立连接。连接上数据源之后，可以通过连接句柄访问到所有有关连接数据源的信息，包括程序运行状态、事务处理状态和错误信息。

原型

```
SQLRETURN  SQLConnect(SQLHDBC        ConnectionHandle,
                      SQLCHAR        *ServerName,
                      SQLSMALLINT    NameLength1,
                      SQLCHAR        *UserName,
                      SQLSMALLINT    NameLength2,
                      SQLCHAR        *Authentication,
                      SQLSMALLINT    NameLength3);
```

参数

表 6-28 SQLConnect 参数

关键字	参数说明
ConnectionHandle	连接句柄，通过SQLAllocHandle获得。
ServerName	要连接数据源的名称。
NameLength1	ServerName的长度。
UserName	数据源中数据库用户名。
NameLength2	UserName的长度。
Authentication	数据源中数据库用户密码。
NameLength3	Authentication的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。

- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLConnect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_DBC和ConnectionHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.9 SQLDisconnect

功能描述

关闭一个与特定连接句柄相关的连接。

原型

```
SQLRETURN SQLDisconnect(SQLHDBC    ConnectionHandle);
```

参数

表 6-29 SQLDisconnect 参数

关键字	参数说明
ConnectionHandle	连接句柄，通过SQLAllocHandle获得。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当调用SQLDisconnect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_DBC和ConnectionHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.10 SQLExecDirect

功能描述

使用参数的当前值，执行一条准备好的语句。对于一次只执行一条SQL语句，SQLExecDirect是最快的执行方式。

原型

```
SQLRETURN SQLExecDirect(SQLHSTMT         StatementHandle,
                        SQLCHAR          *StatementText,
                        SQLINTEGER       TextLength);
```

参数

表 6-30 SQLExecDirect 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。
StatementText	要执行的SQL语句。不支持一次执行多条语句。
TextLength	StatementText的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_NEED_DATA：在执行SQL语句前没有提供足够的参数。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。
- SQL_NO_DATA：表示SQL语句不返回结果集。

注意事项

当调用SQLExecDirect函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.11 SQLExecute

功能描述

如果语句中存在参数标记的话，SQLExecute函数使用参数标记参数的当前值，执行一条准备好的SQL语句。

原型

```
SQLRETURN SQLExecute(SQLHSTMT    StatementHandle);
```

参数

表 6-31 SQLExecute 参数

关键字	参数说明
StatementHandle	要执行语句的语句句柄。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_NEED_DATA：表示在执行SQL语句前没有提供足够的参数。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当SQLExecute函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，可通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.12 SQLFetch

功能描述

从结果集中取下一个行集的数据，并返回所有被绑定列的数据。

原型

```
SQLRETURN SQLFetch(SQLHSTMT    StatementHandle);
```

参数

表 6-32 SQLFetch 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLFetch函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.13 SQLFreeStmt

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeStmt已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

6.6.14 SQLFreeConnect

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeConnect已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

6.6.15 SQLFreeHandle

功能描述

释放与指定环境、连接、语句或描述符相关联的资源，它替代了ODBC 2.x函数SQLFreeEnv、SQLFreeConnect及SQLFreeStmt。

原型

```
SQLRETURN SQLFreeHandle(SQLSMALLINT   HandleType,
                        SQLHANDLE     Handle);
```

参数

表 6-33 SQLFreeHandle 参数

关键字	参数说明
HandleType	SQLFreeHandle要释放的句柄类型。必须为下列值之一： • SQL_HANDLE_ENV • SQL_HANDLE_DBC • SQL_HANDLE_STMT • SQL_HANDLE_DESC 如果HandleType不是这些值之一，SQLFreeHandle返回SQL_INVALID_HANDLE。
Handle	要释放的句柄。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

如果SQLFreeHandle返回SQL_ERROR，句柄仍然有效。

示例

参见：**示例**

6.6.16 SQLFreeEnv

在ODBC 3.x版本中，ODBC 2.x的函数SQLFreeEnv已被SQLFreeHandle代替。有关详细信息请参阅**SQLFreeHandle**。

6.6.17 SQLPrepare

功能描述

准备一个将要进行的SQL语句。

原型

```
SQLRETURN SQLPrepare(SQLHSTMT      StatementHandle,
                     SQLCHAR       *StatementText,
                     SQLINTEGER    TextLength);
```

参数

表 6-34 SQLPrepare 参数

关键字	参数说明
StatementHandle	语句句柄。
StatementText	SQL文本串。
TextLength	StatementText的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当SQLPrepare返回的值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数分别设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.18 SQLGetData

功能描述

SQLGetData返回结果集中某一列的数据。可以多次调用它来部分地检索不定长度的数据。

原型

```
SQLRETURN SQLGetData(SQLHSTMT       StatementHandle,
                     SQLUSMALLINT   Col_or_Param_Num,
                     SQLSMALLINT    TargetType,
                     SQLPOINTER     TargetValuePtr,
                     SQLLEN         BufferLength,
                     SQLLEN         *StrLen_or_IndPtr);
```

参数

表 6-35 SQLGetData 参数

关键字	参数说明
StatementHandle	语句句柄，通过SQLAllocHandle获得。
Col_or_Param_Num	要返回数据的列号。结果集的列按增序从1开始编号。书签列的列号为0。
TargetType	TargetValuePtr缓冲中的C数据类型的类型标识符。若TargetType为SQL_ARD_TYPE，驱动使用ARD中SQL_DESC_CONCISE_TYPE字段的类型标识符。若为SQL_C_DEFAULT，驱动根据源的SQL数据类型选择缺省的数据类型。
TargetValuePtr	**输出参数**：指向返回数据所在缓冲区的指针。
BufferLength	TargetValuePtr所指向缓冲区的长度。
StrLen_or_IndPtr	**输出参数**：指向缓冲区的指针，在此缓冲区中返回长度或标识符的值。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_NO_DATA：表示SQL语句不返回结果集。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。
- SQL_STILL_EXECUTING：表示语句正在执行。

注意事项

当调用SQLFetch函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过调用**SQLGetDiagRec**函数，并将HandleType和Handle参数分别设置为SQL_HANDLE_STMT和StatementHandle，可得到一个相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.19 SQLGetDiagRec

功能描述

返回诊断记录的多个字段的当前值，其中诊断记录包含错误、警告及状态信息。

原型

```
SQLRETURN  SQLGetDiagRec(SQLSMALLINT    HandleType
                    SQLHANDLE      Handle,
                    SQLSMALLINT    RecNumber,
                    SQLCHAR        *SQLState,
                    SQLINTEGER     *NativeErrorPtr,
                    SQLCHAR        *MessageText,
                    SQLSMALLINT    BufferLength
                    SQLSMALLINT    *TextLengthPtr);
```

参数

表 6-36 SQLGetDiagRec 参数

关键字	参数说明
HandleType	句柄类型标识符，它说明诊断所要求的句柄类型。必须为下列值之一： ● SQL_HANDLE_ENV ● SQL_HANDLE_DBC ● SQL_HANDLE_STMT ● SQL_HANDLE_DESC
Handle	诊断数据结构的句柄，其类型由HandleType来指出。如果HandleType是SQL_HANDLE_ENV，Handle可以是共享的或非共享的环境句柄。
RecNumber	指出应用从查找信息的状态记录。状态记录从1开始编号。
SQLState	**输出参数**：指向缓冲区的指针，该缓冲区存储着有关RecNumber的五字符的SQLSTATE码。
NativeErrorPtr	**输出参数**：指向缓冲区的指针，该缓冲区存储着本地的错误码。
MessageText	指向缓冲区的指针，该缓冲区存储着诊断信息文本串。
BufferLength	MessageText的长度。
TextLengthPtr	**输出参数**：指向缓冲区的指针，返回MessageText中的字节总数。如果返回字节数大于BufferLength，则MessageText中的诊断信息文本被截断成BufferLength减去NULL结尾字符的长度。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

SQLGetDiagRec不发布自己的诊断记录。它用下列返回值来报告它自己的执行结果：

- SQL_SUCCESS：函数成功返回诊断信息。
- SQL_SUCCESS_WITH_INFO：*MessageText太小以致不能容纳所请求的诊断信息。没有诊断记录生成。
- SQL_INVALID_HANDLE：由HandType和Handle所指出的句柄是不合法句柄。
- SQL_ERROR：RecNumber小于等于0或BufferLength小于0。

如果调用ODBC函数返回SQL_ERROR或SQL_SUCCESS_WITH_INFO，可调用SQLGetDiagRec返回诊断信息值SQLSTATE，SQLSTATE值的如下表。

表 6-37 SQLSTATE 值

SQLSATATE	错误	描述
HY000	一般错误	未定义特定的SQLSTATE所产生的一个错误。
HY001	内存分配错误	驱动程序不能分配所需要的内存来支持函数的执行或完成。
HY008	取消操作	调用SQLCancel取消执行语句后，依然在StatementHandle上调用函数。
HY010	函数系列错误	在为执行中的所有数据参数或列发送数据前就调用了执行函数。
HY013	内存管理错误	不能处理函数调用，可能由当前内存条件差引起。
HYT01	连接超时	数据源响应请求之前，连接超时。
IM001	驱动程序不支持此函数	调用了StatementHandle相关的驱动程序不支持的函数

示例

参见：示例

6.6.20 SQLSetConnectAttr

功能描述

设置控制连接各方面的属性。

原型

```
SQLRETURN SQLSetConnectAttr(SQLHDBC      ConnectionHandle
                            SQLINTEGER   Attribute,
                            SQLPOINTER   ValuePtr,
                            SQLINTEGER   StringLength);
```

参数

表 6-38 SQLSetConnectAttr 参数

关键字	参数说明
StatementttHandle	连接句柄。
Attribute	设置属性。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr是32位无符号整型值或指向以空结束的字符串。注意，如果ValuePtr参数是驱动程序指定值。ValuePtr可能是有符号的整数。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetConnectAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_DBC的HandleType和ConnectionHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.21 SQLSetEnvAttr

功能描述

设置控制环境各方面的属性。

原型

```
SQLRETURN SQLSetEnvAttr(SQLHENV      EnvironmentHandle
                        SQLINTEGER   Attribute,
                        SQLPOINTER   ValuePtr,
                        SQLINTEGER   StringLength);
```

参数

表 6-39 SQLSetEnvAttr 参数

关键字	参数说明
EnviromentHandle	环境句柄。
Attribute	需设置的环境属性，可为如下值： • SQL_ATTR_ODBC_VERSION：指定ODBC版本。 • SQL_CONNECTION_POOLING：连接池属性。 • SQL_OUTPUT_NTS：指明驱动器返回字符串的形式。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr可能是32位整型值，或为以空结束的字符串。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetEnvAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_ENV的HandleType和EnvironmentHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

6.6.22 SQLSetStmtAttr

功能描述

设置相关语句的属性。

原型

```
SQLRETURN SQLSetStmtAttr(SQLHSTMT      StatementHandle
                         SQLINTEGER    Attribute,
                         SQLPOINTER    ValuePtr,
                         SQLINTEGER    StringLength);
```

参数

表 6-40 SQLSetStmtAttr 参数

关键字	参数说明
StatementttHandle	语句句柄。
Attribute	需设置的属性。
ValuePtr	指向对应Attribute的值。依赖于Attribute的值，ValuePtr可能是32位无符号整型值，或指向以空结束的字符串，二进制缓冲区，或者驱动定义值。注意，如果ValuePtr参数是驱动程序指定值。ValuePtr可能是有符号的整数。
StringLength	如果ValuePtr指向字符串或二进制缓冲区，这个参数是*ValuePtr长度，如果ValuePtr指向整型，忽略StringLength。

返回值

- SQL_SUCCESS：表示调用正确。
- SQL_SUCCESS_WITH_INFO：表示会有一些警告信息。
- SQL_ERROR：表示比较严重的错误，如：内存分配失败、建立连接失败等。
- SQL_INVALID_HANDLE：表示调用无效句柄。其他API的返回值同理。

注意事项

当SQLSetStmtAttr的返回值为SQL_ERROR或SQL_SUCCESS_WITH_INFO时，通过借助SQL_HANDLE_STMT的HandleType和StatementHandle的Handle，调用**SQLGetDiagRec**可得到相关的SQLSTATE值，通过SQLSTATE值可以查出调用此函数的具体信息。

示例

参见：**示例**

7 导入数据

7.1 导入方式说明

GaussDB(DWS)提供了灵活的数据入库方式，可以将多种数据源的数据导入到GaussDB(DWS)中，如图7-1所示。各导入方式具有不同的特点，如表7-1所示，用户可以根据其特点自行选择。建议用户配合云数据迁移（Cloud Data Migration，简称CDM）和数据湖工厂（Data Lake Factory，简称DLF）一起使用，CDM用于批量数据迁移，DLF可以对整个ETL过程进行编排调度，同时提供可视化的开发环境。

图 7-1 导入方式

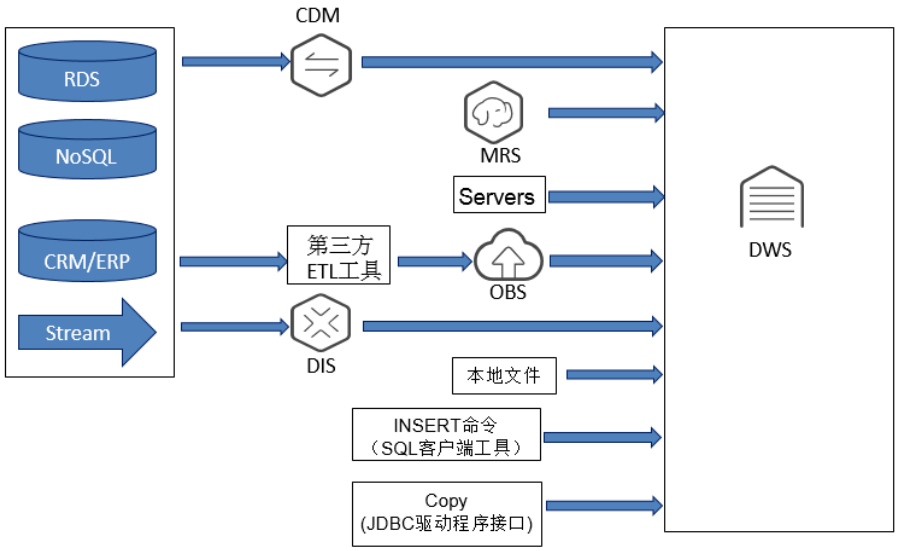


表 7-1 数据导入方式说明

数据导入方式	数据源	说明	优势
从OBS并行导入数据	OBS	支持将存储在OBS上的TXT、CSV、ORC及CARBONDATA格式的数据并行导入到GaussDB(DWS)，支持导入后查询数据，也支持远程读OBS上的数据。 GaussDB(DWS)优先推荐的导入方式。	并行拉取方式，性能好，横向扩展。
使用GDS从远端服务器导入数据	Servers（即远端服务器）	使用GaussDB(DWS)提供的GDS工具，利用多DN并行的方式，将数据从远端服务器导入到GaussDB(DWS)。这种方式导入效率高，适用于大批量数据入库。	
从MRS导入数据到集群	MRS（HDFS）	配置一个GaussDB(DWS)集群连接到一个MRS集群，然后将数据从MRS的HDFS中读取到GaussDB(DWS)。	并行拉取方式，性能好，横向扩展。
使用CDM迁移数据到Gauss(DWS)	数据库、NoSQL、文件系统、大数据平台	CDM提供同构/异构数据源之间批量数据迁移的功能，帮助您实现从多种类型的数据源迁移数据到GaussDB(DWS)。CDM在迁移数据到GaussDB(DWS)时，采用的是Copy方式和GDS并行导入方式。	数据源丰富，操作简单。
第三方ETL工具	数据库、NoSQL、文件系统、大数据平台	请参考第三方ETL工具的相关文档。 GaussDB(DWS)提供了DSC工具，可以将Teradata/Oracle脚本迁移到GaussDB(DWS)。 **使用DSC工具迁移SQL脚本**	通过OBS中转，数据源丰富，数据转换能力强。
通过INSERT语句直接写入数据	-	使用SQL客户端工具或JDBC/ODBC驱动连接GaussDB(DWS)数据库时，执行INSERT语句插入一行或多行数据，以及从指定表插入数据。	INSERT是最简单的一种数据写入方式，适合数据写入量不大，并发度不高的场景。

数据导入方式	数据源	说明	优势
使用COPY FROM STDIN导入数据	其他文件或数据库	使用Java语言开发应用程序时，通过调用JDBC驱动的CopyManager接口，从文件或其他数据库向GaussDB(DWS)写入数据。	从其他数据库直接写入GaussDB(DWS)的方式，具有业务数据无需落地成文件的优势。
使用gsql元命令\copy导入数据	本地文件	与直接使用SQL语句COPY不同，该命令读取/写入的文件只能是gsql客户端所在机器上的本地文件。	操作简单，适用于小批量数据入库。

7.2 从 OBS 并行导入数据

须知

- OBS导入导出数据时，不支持中文路径。
- OBS导入导出数据时，暂不支持跨Region进行OBS数据导入导出，必须确保OBS和DWS集群在同一个Region中。

7.2.1 关于 OBS 并行导入

对象存储服务OBS（Object Storage Service）是公有云上提供的一个基于对象的海量存储服务，为客户提供安全、高可靠、低成本的数据存储能力。OBS为用户提供了超大存储容量的能力，适合存放任意类型的文件。

数据仓库服务GaussDB(DWS)使用OBS作为集群数据与外部数据互相转化的平台，实现安全、高可靠和低成本的存储需求。

GaussDB(DWS)支持将OBS上TXT、CSV、ORC、CARBONDATA格式的数据导入到集群进行查询，也支持远程读OBS上的数据。因此对于经常查询的热数据建议直接导入GaussDB(DWS)后再做查询。偶尔查询的冷数据可以存储在OBS上直接远程读以节省成本。

目前，ORC、CARBONDATA格式数据的操作与TXT、CSV有差异。

- TXT、CSV格式的数据导入或查询

  无需用户自定义外部服务器。TXT和CSV格式的远程读较慢。**关于OBS并行导入**主旨在介绍将OBS上TXT或CSV格式的数据导入集群。如果要远程读只需要对**创建OBS外表**一节中创建的外表执行SELECT即可。
- ORC格式的数据导入或查询

  需要用户自定义外部服务器。**查询OBS上的数据概述**中介绍了OBS上ORC格式的数据如何操作。该节的**通过外表查询OBS上的数据**小节，既包含了通过外表直接查询ORC格式数据，又包含了将数据导入后再执行查询的指导。
- CARBONDATA格式的数据导入或查询

需要用户自定义外部服务器。导入和查询方式与ORC一致。

概述

在数据迁移、ETL（Extract-Transform-Load）过程中，需要向GaussDB(DWS)并行导入海量数据，使用普通方式会耗费大量的时间。GaussDB(DWS)提供了OBS（Object Storage Service）及外表接口，通过OBS外表设置的导入URL路径、导入数据格式等信息来识别数据源文件，利用多DN（Datanode）并行的方式，实现了数据的快速并行导入。

优势：

- CN只负责任务的规划及下发，把数据导入的工作交给了DN，释放了CN的资源，使其有能力处理外部请求。
- 通过让各个DN都参与数据导入，充分利用各个设备的计算能力及网络带宽。
- 支持导入过程中对数据做预处理。
- 支持在导入过程中，针对数据格式错误设置导入容错性，并可在导入结束后根据错误信息定位错误数据。

劣势：

需要创建OBS外表，并且要在OBS服务器上存放导入数据。

适用场景：

高并发、大数据量导入。

相关概念

- **数据源文件**：存储有数据的TEXT、CSV、ORC、CARBONDATA文件。文件中保存的是待并行导入数据库的数据。
- **OBS**：对象存储服务，是一种可存储文档、图片、影音视频等非结构化数据的云存储服务。向GaussDB(DWS)并行导入数据时，数据对象放置在OBS服务器上。
- **桶（Bucket）**：对OBS中的一个存储空间的形象称呼，是存储对象的容器。
  - 对象存储是一种非常扁平化的存储方式，桶中存储的对象都在同一个逻辑层级，去除了文件系统中的多层级树形目录结构。
  - 在OBS中，桶名必须是全局唯一的且不能修改，即用户创建的桶不能与自己已创建的其他桶名称相同，也不能与其他用户创建的桶名称相同。每个桶在创建时都会生成默认的桶ACL（Access Control List），桶ACL列表的每项包含了对被授权用户授予什么样的权限，如读取权限、写入权限、完全控制权限等。用户只有对桶有相应的权限，才可以对桶进行操作，如创建、删除、显示、设置桶ACL等。
  - 一个用户最多可创建100个桶，但每个桶中存放的总数据容量和对象/文件数量没有限制。
- **对象**：是存储在OBS中的基本数据单位。用户上传的数据以对象的形式存储在OBS的桶中。对象的属性包括名称Key，Metadata，Data。

  通常，我们将对象等同于文件来进行管理，但是由于OBS是一种对象存储服务，并没有文件系统中的文件和文件夹概念。为了使用户更方便进行管理数据，OBS提供了一种方式模拟文件夹。通过在对象的名称中增加“/”，如tpcds1000/stock.csv，tpcds1000可以等同于文件夹，stock.csv就可以等同于文件名，而对象名称（key）仍然是tpcds1000/stock.csv、对象的内容就是stock.csv数据文件的内容。

- **Key**：对象的名称（键），为经过UTF-8编码的长度大于0且不超过1024的字符序列，一个桶里的每个对象必须拥有唯一的对象键值。用户可使用桶名+对象名来存储和获取对应的对象。
- **Metadata**：对象元数据，用来描述对象的信息。元数据又可分为系统元数据和用户元数据。这些元数据以键值对（Key-value）的形式随http头域一起上传到OBS系统。
  - 系统元数据由OBS系统产生，在处理对象数据时使用。系统元数据包括：Date, Content-length, last-modify, Content-MD5等。
  - 用户元数据由用户上传对象时指定，是用户自己对对象的一些描述信息。
- **Data**：对象的数据内容，OBS对于数据的内容是无感知的，即认为对象内的数据为无状态的二进制数据。
- **数据库普通表**：数据库中的普通表，数据源文件中的数据最终并行导入到这些表中存储，包括行存表、列存表。
- **外表**：用于识别数据源文件中的数据。外表中保存了数据源文件的位置、文件格式、编码格式、数据间的分隔符等信息。

导入数据原理

OBS导入原理如图7-2所示，CN负责任务的规划及下发，它是按文件给每个DN节点分配任务的。

分配算法如下：

例如，图7-2中，总共有4个节点DN0~DN3， OBS上有6个文件t1.data.0~t1.data.5，那么分配方式如下：

t1.data.0 -> DN0

t1.data.1 -> DN1

t1.data.2 -> DN2

t1.data.3 -> DN3

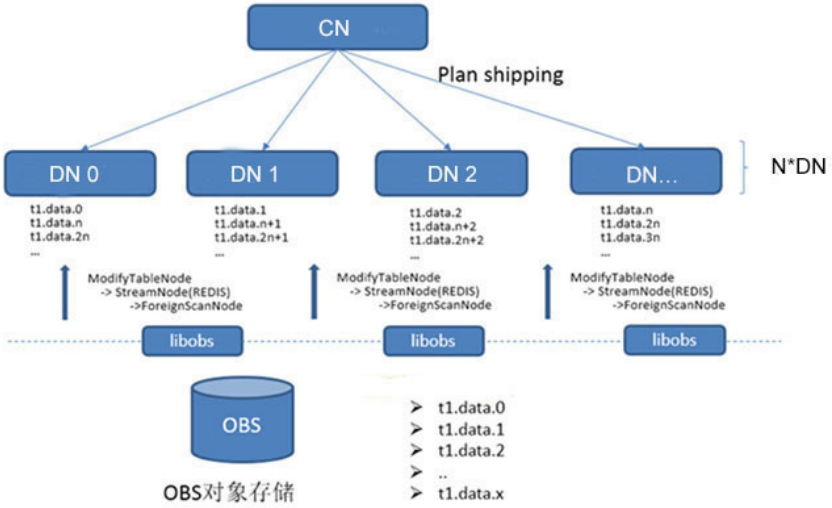
t1.data.4 -> DN0

t1.data.5 -> DN1

其中DN0 和DN1上分配了两个文件，其他DN分配了1个文件。

如果OBS上文件大小都相同时，OBS上的文件数与DN节点数的比例为1:1时导入性能为最好，因为每个DN分配的任务都相同。因此建议将数据文件存储到OBS前，尽可能均匀地将文件切分成多个，文件的数量以DN的整数倍更适合。

图 7-2 通过 OBS 外表并行导入数据



导入流程图

图 7-3 并行导入流程

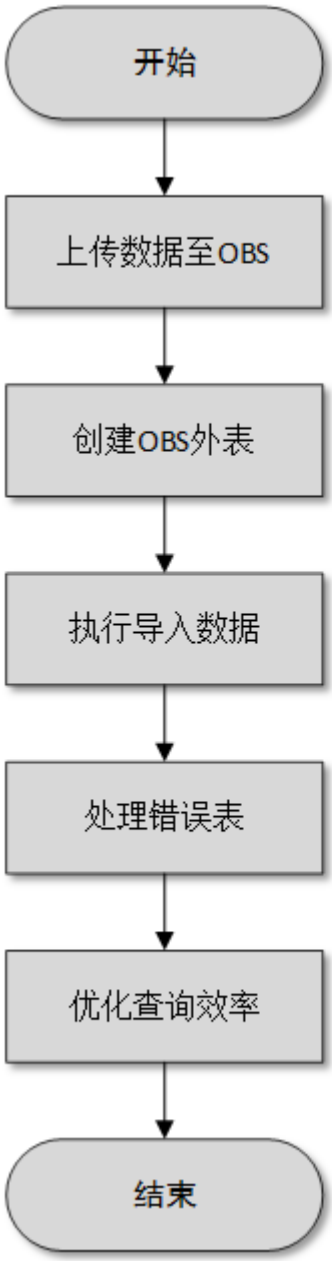


表 7-2 流程说明

流程	说明	子任务
上传数据至OBS。	在OBS服务器上规划存储路径，并上传数据文件。 详细请参见**上传数据到OBS**。	-

流程	说明	子任务
创建OBS外表。	创建外表用于识别OBS服务器上的数据源文件。在OBS外表中保存了数据源在OBS服务器上的桶名、对象名，文件格式、存放位置、编码格式、数据间的分隔符等信息。 详细请参见**创建OBS外表**。	-
执行导入数据。	在创建好外表后，通过INSERT语句，将数据快速、高效地导入到目标表中。 详细请参见**执行导入数据**。	-
处理错误表。	在数据并行导入发生错误时，请根据错误信息，**处理错误表**，以保证导入数据的完整性。	-
优化查询效率。	导入数据后，通过ANALYZE语句生成表统计信息。ANALYZE语句会将统计结果自动存储在系统表PG_STATISTIC中。执行计划生成器会使用这些统计数据，以生成最有效的查询执行计划。 详细请参见**分析表**。	-

7.2.2 创建访问密钥（AK 和 SK）

在本示例中，我们将导入OBS数据到GaussDB(DWS)集群数据库中。公有云用户通过客户端或API、SDK等方式访问OBS时，需要通过AK/SK认证方式进行认证鉴权。因此，当您需要通过客户端或JDBC/ODBC应用程序等方式连接GaussDB(DWS)数据库访问OBS时，必须先获取访问密钥（AK和SK）。

- Access Key Id（AK）：访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。
- Secret Access Key（SK）：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。

创建访问密钥（AK 和 SK）

在创建访问密钥前，请确保登录控制台的帐号已通过实名认证。

通过管理控制台创建访问密钥（AK和SK）操作步骤如下：

步骤1 登录GaussDB(DWS) 管理控制台。

步骤2 单击右上角用户名，在下拉菜单中单击“我的凭证”。

步骤3 在左侧导航树单击“访问密钥”。

如果在访问密钥列表中已经有访问密钥了，可以直接使用现有的访问密钥。但是，在访问密钥列表中只能查看到访问密钥ID（即Access Key ID），只有在新增访问密钥时，用户才可以下载到含有Access Key ID和Secret Access Key的密钥文件。如果您没有该密钥文件，可以单击“新增访问密钥”重新创建。

说明

- 每个用户最多可创建两个有效的访问密钥，如果当前已存在2个访问密钥，只能先删除现有的访问密钥，然后再重新创建。删除时，需要输入当前用户的登录密码、邮箱或手机短信的验证码，验证通过才能成功删除。
- 为了账号安全性，建议您定期更换并妥善保存访问密钥。

步骤4 单击“新增访问密钥”。

步骤5 在弹出的对话框中，输入登录密码和对应验证码，然后单击“确定”。

说明

- 用户如果未绑定邮箱和手机，则只需输入登录密码。
- 用户如果同时绑定了邮箱和手机，可以选择其中一种方式进行验证。

步骤6 在弹出的“下载确认”提示框中，单击“确定”后，密钥会直接保存到浏览器默认的下载文件夹中。

说明

- 为防止访问密钥泄露，建议您将其保存到安全的位置。
- 如果用户在此提示框中单击“取消”，则不会下载密钥，后续也将无法重新下载。如果需要使用访问密钥，可以重新创建新的访问密钥。

步骤7 打开下载下来的“credentials.csv”文件即可获取到访问密钥（AK和SK）。

----结束

注意事项

当用户发现访问密钥被异常使用（包括丢失，泄露等情况），或不再使用访问密钥时，建议在访问密钥列表中立即删除密钥或者通知管理员重置相关密钥。

删除访问密钥时，需要输入登录密码和邮箱或者手机验证码进行验证。

说明

删除的访问密钥将永久删除且无法恢复。

7.2.3 上传数据到 OBS

操作场景

从OBS导入数据到集群之前，需要提前准备数据源文件，并将数据源文件上传到OBS。如果您的数据文件已经在OBS上了，则只需完成**上传数据到OBS**中的**步骤2~步骤3**。

准备数据文件

准备需要上传到OBS的数据源文件。GaussDB(DWS)只支持CSV、TEXT、ORC和CARBONDATA格式的数据源文件。

如果用户数据无法以CSV格式保存，可以选择以文本类型保存为其他任意格式后缀的文件。

说明

根据**导入数据原理**，当数据源文件的数据量较大时，将数据文件存储到OBS前，尽可能均匀地将文件切分成多个，文件数量为DataNode的整数倍时，导入性能更好。

上传数据到 OBS

步骤1 上传数据到OBS。

将待导入的数据源文件存储在OBS桶中。

1. 登录OBS管理控制台。
   单击“服务列表”，选择“对象存储服务”，打开OBS管理控制台页面。
2. 创建桶。
   如何创建OBS桶，具体请参见《对象存储服务控制台指南》中的**创建桶**章节。
   例如，创建以下两个桶：“mybucket”和“mybucket02”。
3. 新建文件夹。
   具体请参见《对象存储服务控制台指南》中的**新建文件夹**章节。
   例如：
   - 在已创建的OBS桶“mybucket”中新建一个文件夹“input_data”。
   - 在已创建的OBS桶“mybucket02”中新建一个文件夹“input_data”。
4. 上传文件。
   具体请参见《对象存储服务控制台指南》的**上传对象**章节。
   例如：
   - 将以下数据文件上传到OBS桶“mybucket”的“input_data”目录中。
     ```
     product_info.0
     product_info.1
     ```
   - 将以下数据文件上传到OBS桶“mybucket02”的“input_data”目录中。
     ```
     product_info.2
     ```

步骤2 获取数据源文件的OBS路径。

数据源文件在上传到OBS桶之后，会生成全局唯一的访问路径。数据源文件的OBS路径用于创建外表时location参数设置。

location参数中OBS文件的路径由“obs://”、桶名和文件路径组成，即为：

obs://<bucket_name>/<file_path>

例如，在本例中，location参数中数据文件的OBS路径分别为：

```
obs://mybucket/input_data/product_info.0
obs://mybucket/input_data/product_info.1
obs://mybucket02/input_data/product_info.2
```

步骤3 为导入用户设置OBS桶的读取权限。

在从OBS导入数据到集群时，执行导入操作的用户需要取得数据源文件所在OBS桶的读取权限。通过配置桶的ACL权限，可以将读取权限授予指定的用户帐号。

具体请参见《对象存储服务控制台指南》中的**配置桶ACL**章节。

----结束

7.2.4 创建 OBS 外表

操作步骤

步骤1 根据**上传数据到OBS**中规划的路径，由此确定创建外表时使用的参数**loaction**的值。

步骤2 用户获取OBS访问协议对应的AK值和SK值。获取访问密钥，请登录管理控制台，将鼠标移至右上角的用户名，单击“我的凭证”，然后在左侧导航树单击“访问密钥”。在访问密钥页面，可以查看已有的访问密钥ID（即AK），如果要同时获取AK和SK，可以单击“新增访问密钥”创建并下载访问密钥。

步骤3 梳理待导入数据的格式信息，确定创建外表时使用的数据格式参数的值。需要收集的主要数据源格式信息如下：

- **format**：外表中数据源文件的格式。OBS外表导入支持CSV、TEXT格式。缺省值为TEXT。
- **header**：指定导出数据文件是否包含标题行，header只能用于CSV格式的文件中。
- **delimiter**：指定数据文件行数据的字段分隔符，不指定则使用默认分隔符。
- 外表可以识别的更多参数，详细使用请参见数据格式参数 。

步骤4 **规划并行导入容错性**，以控制导入过程中处理错误的方式。

- **fill_missing_fields**：数据入库时，数据源文件中某行的最后一个字段缺失时，请选择是直接将字段设为Null，还是在错误表中报错提示。
- **ignore_extra_data**：数据源文件中的字段比外表定义列数多时，请选择是忽略多出的列，还是在错误表中报错提示。
- **per node reject limit**：本次数据导入过程中每个DN实例上允许出现的数据格式错误的数量。如果有一个DN实例上录入错误表中的错误数量超过设定值时，本次导入失败，报错退出。可以选择不做限制，也可以根据所能容忍的错误数量选择一个上限值。
- **compatible_illegal_chars**：导入时遇到非法字符，选择如何处理。是将非法字符按照转换规则转换后入库，还是报错中止导入。

  非法字符容错转换规则如下：

  - 对于'\0'，容错后转换为空格。
  - 对于其他非法字符，容错后转换为问号。
  - 对非法字符进行容错转换时，如遇NULL、DELIMITER、QUOTE、ESCAPE也设置成了空格或问号，GaussDB(DWS)会通过如"illegal chars conversion may confuse COPY escape 0x20"等报错信息提示用户修改可能引起混淆的参数以避免导入错误。
- **error_table_name**：用于记录数据格式错误信息的错误表表名。并行导入结束后查询此错误信息表，能够获取详细的错误信息。
- 更多参数，详细使用请参见容错性参数。

步骤5 根据前面步骤确定的参数，**创建OBS外表**。外表的创建语法以及详细使用，请参考CREATE FOREIGN TABLE (OBS导入导出)。

----结束

示例

在GaussDB(DWS)数据库中，创建一个外表。参数信息如下所示：

- **数据格式参数访问秘钥（AK和SK）**
  - 用户获取OBS访问协议对应的AK值（access_key）。
  - 用户获取OBS访问协议对应的SK值（secret_access_key）。

  说明

  请根据用户实际获取的access_key和secret_access_key的密钥替换示例中的斜体内容。
- **设置数据格式参数**
  - **数据源文件格式（format）**为CSV。
  - **编码格式（encoding）**为UTF-8。
  - **使用加密（encrypt）**为 'off'。
  - **字段分隔符（delimiter）**为','。
  - **引号字符（quote）**使用默认值双引号。
  - **null（数据文件中空值的表示）**为“一个没有引号的空字符串”。
  - **header（指定导出数据文件是否包含标题行）**为false，当数据文件第一行不是标题行（即表头），不需要设置。

    说明

    OBS导出数据时不支持该参数为ture，使用缺省值false。
- **设置导入时的容错性参数**
  - **PER NODE REJECT LIMIT '*value*'** 为unlimited，即接受导入过程中所有数据格式错误。
  - **LOG INTO *error_table_name*** 指定为product_info_err，将数据导入过程中出现的数据格式错误信息写入表product_info_err。
  - **fill_missing_fields** 为true，即当数据加载时，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。
  - **ignore_extra_data**为true，当数据加载时，若数据源文件比外表定义列数多，则忽略行尾多出来的列，不报错。

根据以上信息，创建的外表如下所示：

```
DROP FOREIGN TABLE product_info_ext;

CREATE FOREIGN TABLE product_info_ext
(
    product_price               integer       not null,
    product_id                  char(30)      not null,
    product_time                date          ,
    product_level               char(10)      ,
    product_name                varchar(200)  ,
    product_type1               varchar(20)   ,
    product_type2               char(10)      ,
    product_monthly_sales_cnt   integer       ,
    product_comment_time        date          ,
```

```
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
SERVER gsmpp_server
OPTIONS(
location 'obs://mybucket/input_data/product_info | obs://mybucket02/input_data/product_info',
FORMAT 'CSV' ,
DELIMITER ',',
encoding 'utf8',
header 'false',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
fill_missing_fields 'true',
ignore_extra_data 'true'
)
READ ONLY
LOG INTO product_info_err
PER NODE REJECT LIMIT 'unlimited';
```

返回如下信息表示创建成功：

```
CREATE FOREIGN TABLE
```

7.2.5 执行导入数据

背景信息

在执行数据导入前，您可以参考以下优秀实践方法进行合理的设计部署，最大化的使用系统资源，以提高数据导入性能。

- OBS的数据导入性能，多数场景受限于网络的并发访问速率，因此在OBS服务器上最好部署多个桶，使用多桶并发导入，提高DN数据传输利用率。
- 并发导入场景，与单表导入相似，至少应保证I/O性能大于网络最大速率。
- 配置GUC参数“raise_errors_if_no_files”、“ partition_mem_batch”和“partition_max_cache_size”，设置导入时是否区分“导入文件记录数为空”和“导入文件不存在”、导入时的缓存个数以及数据缓存区大小。
- 若导入表存在索引，在数据导入过程中，将增量更新索引信息，影响数据导入性能。建议在执行数据导入前，先删除相关表的索引。在数据导入完成后，再重新创建索引。

操作步骤

步骤1 在GaussDB(DWS)数据库中，创建目标表，用于存储从OBS导入的数据。建表语法请参考CREATE TABLE。

目标表的表结构和OBS上将要导入的数据源文件的字段要保持一一对应，即字段个数、字段类型要一致。并且，目标表和创建的外表的表结构也要保持一致，字段名称可以不一样。

步骤2 （可选）若导入表存在索引，在数据导入过程中，将增量更新索引信息，影响数据导入性能。建议在执行数据导入前，先删除相关表的索引。在数据导入完成后，再重新创建索引。

步骤3 执行数据导入。

```
INSERT INTO [目标表名] SELECT * FROM [foreign table 表名];
```

- 若出现以下类似信息，说明数据导入成功。请查询错误信息表，查看是否存在数据格式错误，详细操作请参见处理错误表。

  ```
  INSERT 0 20
  ```

- 若出现数据加载错误，请参见**处理错误表**，并重新执行数据导入。

----结束

示例

创建一个名为product_info的表，示例如下：

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
with (
orientation = column,
compression=middle
)
DISTRIBUTE BY HASH (product_id);
```

执行以下命令将外表*product_info*_ext的数据导入到目标表*product_info* 中：

```
INSERT INTO product_info SELECT * FROM product_info_ext;
```

7.2.6 处理错误表

操作场景

当数据导入发生错误时，请根据本文指引信息进行处理。

查询错误信息

数据导入过程中发生的错误，一般分为数据格式错误和非数据格式错误。

- 数据格式错误

  在创建外表时，通过设置参数“LOG INTO error_table_name”，将数据导入过程中出现的数据格式错误信息写入指定的错误信息表error_table_name中。您可以通过以下SQL，查询详细错误信息。

  ```
  SELECT * FROM error_table_name;
  ```

  错误信息表结构如**表7-3**所示。

  表 7-3 错误信息表

列名称	类型	描述
nodeid	integer	报错节点编号。
begintime	timestamp with time zone	出现数据格式错误的时间。

列名称	类型	描述
filename	character varying	出现数据格式错误的数据源文件名。
rownum	bigint	在数据源文件中，出现数据格式错误的行号。
rawrecord	text	在数据源文件中，出现数据格式错误的原始记录。
detail	text	详细错误信息。

- 非数据格式错误

  对于非数据格式错误，一旦发生将导致整个数据导入失败。您可以根据执行数据导入过程中，界面提示的错误信息，帮助定位问题，处理错误表。

处理数据导入错误

根据获取的错误信息，请对照下表，处理数据导入错误。

表 7-4 处理数据导入错误

错误信息	原因	解决办法
missing data for column "r_reason_desc"	1. 数据源文件中的列数比外表定义的列数少。 2. 对于TEXT格式的数据源文件，由于转义字符（\）导致delimiter（分隔符）错位或者quote（引号字符）错位造成的错误。 **示例：**目标表存在3列字段，导入的数据如下所示。由于存在转义字符“\”，分隔符“|”被转义为第二个字段的字段值，导致第三个字段值缺失。 `BE|Belgium\\|1`	1. 由于列数少导致的报错，选择下列办法解决： ● 在数据源文件中，增加列“r_reason_desc”的字段值。 ● 在创建外表时，将参数“fill_missing_fields”设置为“on”。即当导入过程中，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 2. 对由于转义字符导致的错误，需检查报错的行中是否含有转义字符（\）。若存在，建议在创建外表时，将参数“noescaping”（是否不对'\'和后面的字符进行转义）设置为true。

错误信息	原因	解决办法
extra data after last expected column	数据源文件中的列数比外表定义的列数多。	• 在数据源文件中，删除多余的字段值。 • 在创建外表时，将参数“ignore_extra_data”设置为“on”。即在导入过程中，若数据源文件比外表定义的列数多，则忽略行尾多出来的列。
invalid input syntax for type numeric: "a"	数据类型错误。	在数据源文件中，修改输入字段的数据类型。根据此错误信息，请将输入的数据类型修改为numeric。
null value in column "staff_id" violates not-null constraint	非空约束。	在数据源文件中，增加非空字段信息。根据此错误信息，请增加“staff_id”列的值。
duplicate key value violates unique constraint "reg_id_pk"	唯一约束。	• 删除数据源文件中重复的行。 • 通过设置关键字“DISTINCT”，从SELECT结果集中删除重复的行，保证导入的每一行都是唯一的。 **INSERT INTO** reasons **SELECT DISTINCT * FROM** foreign_tpcds_reasons;
value too long for type character varying(16)	字段值长度超过限制。	在数据源文件中，修改字段值长度。根据此错误信息，字段值长度限制为VARCHAR2(16)。

7.2.7 OBS 导入数据示例

步骤1 在GaussDB(DWS)上，创建导入的目标表tpcds.customer_address。

```
CREATE TABLE tpcds.customer_address
(
    ca_address_sk             integer               not null,
    ca_address_id             char(16)              not null,
    ca_street_number          char(10)                      ,
    ca_street_name            varchar(60)                   ,
    ca_street_type            char(15)                      ,
    ca_suite_number           char(10)                      ,
    ca_city                   varchar(60)                   ,
    ca_county                 varchar(30)                   ,
    ca_state                  char(2)                       ,
    ca_zip                    char(10)                      ,
    ca_country                varchar(20)                   ,
    ca_gmt_offset             decimal(5,2)                  ,
```

```
    ca_location_type        char(20)
)
WITH (orientation = column,compression=middle)
DISTRIBUTE BY hash (ca_address_sk);
```

步骤2 用户通过管理控制台登录到OBS数据服务器。在OBS数据服务器上，分别创建数据文件存放的两个桶“/input-data1”和“/input-data2”，并创建每个桶下面的data目录“/input-data1/data”和“/input-data2/data”。

步骤3 将数据源文件均匀上传至OBS数据服务器的“/input-data1/data/”和“/input-data2/data/”目录中。

步骤4 在GaussDB(DWS)上，创建外表tpcds.customer_address_ext用于接收数据服务器上的数据。

假设OBS数据服务器与集群网络连接正常，OBS数据服务器IP为xxx.xxx.x.xx，数据源文件格式为CSV，规划的并行导入与示例保持一致。

其中设置的**导入信息**如下所示：

- 由于OBS服务器上的数据源文件存放目录为“/input-data1/data/ ”和/input-data2/data/ ，所以设置参数“location”为“obs://input-data1/data/ | obs://input-data2/data/”。

设置的**数据格式信息**是根据数据源文件的详细数据格式参数信息指定的，参数设置如下所示：

- 数据源文件格式（ format ）为CSV。
- 编码格式（ encoding ）为UTF-8。
- 字段分隔符（ delimiter ）为0E08。
- 引号字符（ quote ）为0x1b。
- 使用加密（ encrypt ）为 'off'。
- 用户获取OBS访问协议对应的AK值（ access_key ） 。
- 用户获取OBS访问协议对应的SK值 （ secret_access_key ）。

  说明

  请根据用户实际获取的access_key和secret_access_key的密钥替换示例中的斜体内容。

设置的**导入容错性**如下所示：

- 允许每个DN上出现数据格式错误的个数（ PER NODE REJECT LIMIT 'value' ）为unlimited，即接受导入过程中所有数据格式错误。
- 将数据导入过程中出现的数据格式错误信息（ LOG INTO error_table_name ）写入表customer_address_err。

根据以上信息，创建的外表如下所示：

```
CREATE FOREIGN TABLE tpcds.customer_address_ext
(
ca_address_sk             integer                       ,
ca_address_id             char(16)                      ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                      ,
ca_suite_number           char(10)                      ,
ca_city                   varchar(60)                   ,
ca_county                 varchar(30)                   ,
ca_state                  char(2)                       ,
```

```
ca_zip                  char(10)                ,
ca_country              varchar(20)             ,
ca_gmt_offset           decimal(5,2)            ,
ca_location_type        char(20)
)
SERVER gsmpp_server
OPTIONS(location 'obs://input-data1/data/ | obs://input-data2/data/',
FORMAT 'CSV' ,
encoding 'utf8',
DELIMITER E'\x08',
quote E'\x1b',
encrypt 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
)LOG INTO customer_address_err PER NODE REJECT LIMIT 'unlimited';
```

步骤5 在GaussDB(DWS)上，通过外表tpcds.customer_address_ext，将数据导入目标表tpcds.customer_address。

```
INSERT INTO tpcds.customer_address SELECT * FROM tpcds.customer_address_ext;
```

步骤6 查询错误信息表customer_address_err，处理数据导入错误。更多关于错误表的信息，请参见**处理错误表**。

```
SELECT * FROM customer_address_err;
```

步骤7 在数据导入完成后，执行ANALYZE语句生成表统计信息。

```
ANALYZE tpcds.customer_address;
```

----结束

7.3 使用 GDS 从远端服务器导入数据

7.3.1 关于 GDS 并行导入

INSERT和COPY方式执行数据导入时，是一个串行执行的过程，导入性能低，因此适用于小数据量的导入。对于大数据量的导入，GaussDB(DWS)支持使用GDS工具通过外表并行导入数据到集群。

概述

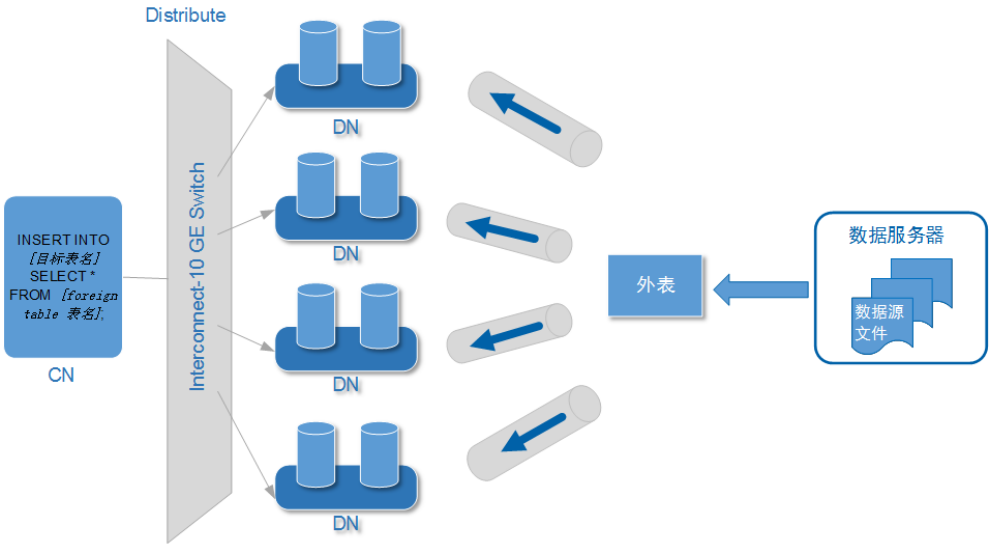
并行导入将存储在服务器普通文件系统中的数据导入到GaussDB(DWS)数据库中。暂时不支持将存储在HDFS文件系统上的数据导入GaussDB(DWS)。

并行导入功能通过外表设置的导入策略、导入数据格式等信息来识别数据源文件，利用多DN并行的方式，将数据从数据源文件导入到数据库中，从而提高整体导入性能。如**图7-4**所示：

- CN只负责任务的规划及下发，把数据导入的工作交给了DN，释放了CN的资源，使其有能力处理其他外部请求。
- 所有DN都参与数据导入，这样可以充分利用各设备的计算能力及网络带宽，提升导入效率。

外表灵活的OPTION设置，有利于在数据入库前对数据做预处理，例如非法字符替换、容错处理等。

图 7-4 数据并行导入示意图



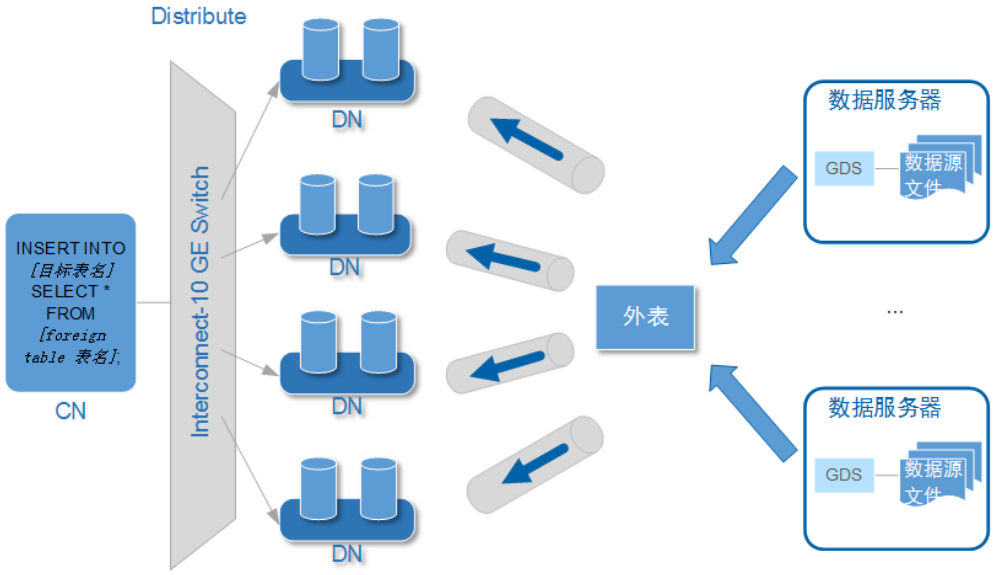
上图中所涉及的相关概念说明如下：

- **CN（Coordinator Node）**：GaussDB(DWS)协调节点。在导入场景下，接收到应用或客户端的导入SQL指令后，负责任务的规划及下发到DN。
- **DN（Datanode）**：GaussDB(DWS)数据节点。接收CN下发的导入任务，将数据源文件中的数据通过外表写入数据库目标表中。
- **数据源文件**：存有数据的文件。文件中保存的是待导入数据库的数据。
- **数据服务器**：数据源文件所在的服务器称为数据服务器。基于安全考虑，建议数据服务器和GaussDB(DWS)集群处于同一内网。
- **外表Foreign Table**：用于识别数据源文件的位置、文件格式、存放位置、编码格式、数据间的分隔符等信息。是关联数据文件与数据库实表（目标表）的对象。
- **目标表**：数据库中的实表。数据源文件中的数据最终导入到这些表中存储，包括行存表和列存表。

GDS 并发导入

- 数据量大，数据存储在多个服务器上时，在每个数据服务器上安装配置、启动GDS后，各服务器上的数据可以并行入库。如图7-5所示。

图 7-5 多数据服务器并行导入



须知

GDS进程数目不能超过DN数目。如果超过，会出现一个DN连接多个GDS进程的情形，可能会导致部分GDS异常运行。

- 数据存储在一台数据服务器上时，如果GaussDB(DWS)及数据服务器上的I/O资源均还有可利用空间时，可以采用GDS多线程来支持并发导入。

  GDS是根据导入事务并发数来决定服务运行线程数的。也就是说即使启动GDS时设置了多线程，也并不会加速单个导入事务。未做过人为事务处理时，一条INSERT语句就是一个导入事务。

  综上，多线程的使用场景如下：

  - 多表并发导入时，采用多线程充分利用资源及提升并发导入效率。
  - 对数据量大的某一事实表的导入进行提速。

    将该事实表对应的数据拆分为多个数据文件，通过多外表同时入库的方式实现多线程并发导入。注意需确保每个外表所能读取的数据文件不重复。

导入流程

图 7-6 GDS 并行导入流程

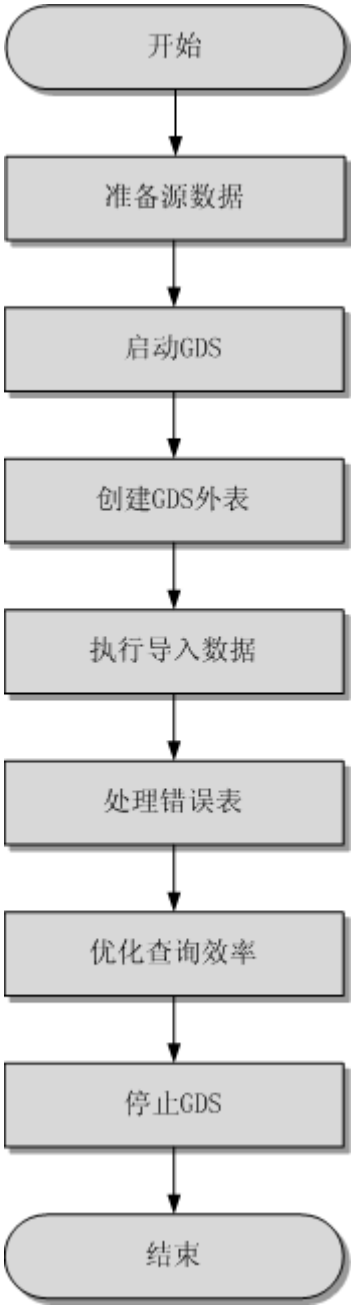


表 7-5 流程说明

流程	说明
准备源数据。	准备需要导入数据库的源数据文件，并上传至数据服务器。 详细内容请参见**准备源数据**。
启动GDS。	在数据服务器上安装配置并启动GDS。 详细内容请参见**安装配置和启动GDS**。

流程	说明
创建外表。	创建外表用于识别数据源文件中的数据。外表中保存了数据源文件的位置、文件格式、存放位置、编码格式、数据间的分隔符等信息。 详细内容请参见**创建GDS外表**。
执行导入数据。	在创建好外表后，通过INSERT语句，将数据快速、高效地导入到目标表中。详细内容请参见**执行导入数据**。
处理错误表。	在数据并行导入发生错误时，请根据具体的错误信息进行处理，以保证导入数据的完整性。 详细内容请参见**处理错误表**。
优化查询效率。	导入数据后，通过ANALYZE语句生成表统计信息。ANALYZE语句会将统计结果自动存储在系统表PG_STATISTIC中。执行计划生成器会使用这些统计数据，以生成最有效的查询执行计划。
停止GDS	待数据导入完成后，登录每台数据服务器，分别停止GDS。 GDS的停止请参见**停止GDS**。

7.3.2 准备源数据

操作场景

通常在将数据导入数据库前，即将入库的数据已经在相关主机上了。我们称这种保存着待入库数据的服务器为数据服务器。此时，只需检测以确认数据服务器和GaussDB(DWS)集群能够正常通信，并查看和记录数据在数据服务器上的存放目录备用。

如果待入库数据还没有就绪，则请先参考如下步骤，将数据上传到数据服务器上。

操作步骤

步骤1 以root用户登录数据服务器。

步骤2 创建数据文件存放目录“/input_data”。

```
mkdir -p /input_data
```

步骤3 将数据源文件上传至上一步所创建的目录中。

GDS并行导入支持CSV、TEXT格式的数据导入。请确保数据源文件符合格式要求。

----结束

7.3.3 安装配置和启动 GDS

操作场景

GaussDB(DWS)提供了数据服务工具GDS来帮助分发待导入的用户数据及实现数据的高速导入。GDS需部署到数据服务器上。

数据量大，数据存储在多个服务器上时，在每个数据服务器上安装配置、启动GDS后，各服务器上的数据可以并行入库。GDS在各台数据服务器上的安装配置和启动方法相同，本节以一台服务器为例进行说明。

背景信息

1. GDS支持在如下的操作系统中安装：

   鲲鹏平台：

   - Community Enterprise Operating System 7.6。
   - EulerOS 2.0 SP8。
   - Red Hat Enterprise Linux Server release 7.5。
   - 中标麒麟7.5/7.6。

   x86平台：

   - SUSE Linux Enterprise Server 10 SP4 x86_64。
   - SUSE Linux Enterprise Server 11 SP1/SP2/SP3/SP4 x86_64。
   - SUSE Linux Enterprise Server 12 SP0/SP1/SP2/SP3 x86_64。
   - Red Hat Enterprise Linux Server release 6.4/6.5/6.6/6.7/6.8/6.9/7.0/7.1/7.2/7.3/7.4/7.5 x86_64。
   - Community Enterprise Operating System 6.4/6.5/6.6/6.7/6.8/6.9/7.0/7.1/7.2/7.3/7.4 x86_64。

2. GDS的版本需与集群版本保持一致（如：GDS V100R008C00版本与GaussDB(DWS) 1.3.X版本配套），否则可能会出现导入导出失败或导入导出进程停止响应等情况。

   因此请勿使用历史版本的GDS进行导入。数据库版本升级后，请按照**操作步骤**中的方式下载GaussDB(DWS)软件包解压缩自带的版本的GDS进行安装配置和启动。在导入导出开始时，GaussDB(DWS)也会进行两端的版本一致性检测，不一致时会打屏显示报错信息并终止对应操作。

   GDS的版本号的查看办法为：在GDS工具的解压目录下执行如下命令。

   ```
   gds -V
   ```

   数据库版本的查看办法为：连接数据库后，执行如下SQL命令查看。

   ```
   SELECT version();
   ```

操作步骤

步骤1 在使用GDS导入/导出数据前，请先完成**使用GDS从远端服务器导入数据**中的步骤：“准备ECS作为GDS服务器”、“下载GDS工具包和SSL证书”。

步骤2 以root用户登录待安装GDS的数据服务器，创建存放GDS工具包的目录。

```
mkdir -p /opt/bin/dws
```

步骤3 将GDS工具包上传至上一步所创建的目录中。

以上传SUSE Linux版本的工具包为例，将GDS工具包“dws_client_redhat_x64.zip”上传至上一步所创建的目录中。

步骤4 （可选）如果使用SSL加密传输，请一并上传SSL证书至**步骤2**所创建的目录下。

步骤5 在工具包所在目录下，解压工具包。

```
cd /opt/bin/dws
unzip dws_client_redhat_x64.zip
```

步骤6 创建GDS专有用户及其所属的用户组。此用户用于启动GDS及读取源数据。

```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```

步骤7 分别修改工具包和数据源文件目录属主为GDS专有用户。

```
chown -R gds_user:gdsgrp /opt/bin/dws/gds
chown -R gds_user:gdsgrp /input_data
```

步骤8 切换到gds_user用户。

```
su - gds_user
```

步骤9 启动GDS服务。

GDS是绿色软件，解压后启动即可。GDS启动方式有两种。

对于集中一次性导入的场景推荐使用方法一；

对于需要隔段时间再次导入的场景，推荐使用方法二以配置文件的形式提升启动效率。

方法一：直接使用“gds”命令，在命令项中设置启动参数。

- 使用“gds”命令，启动GDS。
  - 非SSL模式传输数据的情况下，启动GDS。

    ```
    gds -d dir -p ip:port -H address_string -l log_file -D -t worker_num
    ```

    示例：

    ```
    /opt/bin/dws/gds/gds -d /input_data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -
    l /opt/bin/dws/gds/gds_log.txt -D -t 2
    ```

  - 使用SSL加密方式传输数据的情况下，启动GDS。

    ```
    gds -d dir -p ip:port -H address_string -l log_file -D
    -t worker_num --enable-ssl --ssl-dir Cert_file
    ```

    示例：

    以**步骤4**中SSL证书以上传至/opt/bin为例，命令如下。

    ```
    /opt/bin/dws/gds/gds -d /input_data/ -p 192.168.0.90:5000 -H 10.10.0.1/24 -
    l /opt/bin/dws/gds/gds_log.txt -D --enable-ssl --ssl-dir /opt/bin/
    ```

  命令中的斜体部分请根据实际替换。

  - **-d** *dir*：保存有待导入数据的数据文件所在目录。本教程中为“/input_data/”。
  - **-p** *ip:port*：GDS监听IP和监听端口。默认值为：127.0.0.1，需要替换为能跟GaussDB(DWS)通信的万兆网IP。监听端口的取值范围：1024~65535。默认值为：8098。本教程配置为：192.168.0.90:5000。
  - **-H** *address_string*：允许哪些主机连接和使用GDS服务。参数需为CIDR格式。此参数配置的目的是允许GaussDB(DWS)集群可以访问GDS服务进行数据导入。所以请保证所配置的网段包含GaussDB(DWS)集群各主机。
  - **-l** *log_file*：存放GDS的日志文件路径及文件名。本教程为“/opt/bin/dws/gds/gds_log.txt”。
  - **-D**：后台运行GDS。仅支持Linux操作系统下使用。
  - **-t** *worker_num*：设置GDS并发线程数。GaussDB(DWS)及数据服务器上的I/O资源均充足时，可以加大并发线程数。

    GDS是根据导入事务并发数来决定服务运行线程数的。也就是说即使启动GDS时设置了多线程，也并不会加速单个导入事务。未做过人为事务处理时，一条INSERT语句就是一个导入事务。
  - **--enable-ssl**：启用SSL加密方式传输数据。

- **--ssl-dir** *Cert_file*：SSL证书所在目录。需与**步骤4**中的证书保存目录保持一致。
- 关于更多参数的设置信息请参考**gds命令简介**。

方法二：将启动参数写进配置文件“gds.conf”后， 使用“gds_ctl.py”命令启动。

- 使用“gds_ctl.py”命令，启动GDS。

  a. 使用如下命令，进入GDS工具包的“config”目录下，配置“gds.conf”文件。“gds.conf”配置详细信息请参考**表7-6**。

  vim */opt/bin/gds/config/***gds.conf**

  示例：

  配置“gds.conf”文件如下：

  ```
  <?xml version="1.0"?>
  <config>
  <gds name="gds1" ip="192.168.0.90" port="5000" data_dir="/input_data/" err_dir="/err"
  data_seg="100MB" err_seg="100MB" log_file="/log/gds_log.txt" host="10.10.0.1/24"
  daemon='true' recursive="true" parallel="32"></gds>
  </config>
  ```

  配置文件信息如下：

  - 数据服务器所在IP为192.168.0.90，GDS监听端口为5000。
  - 数据文件存放在“/input_data/”目录下。
  - 错误日志文件存放在“/err”目录下。
  - 单个数据文件大小为100MB。
  - 每个错误日志大小为100MB。
  - 日志保存在“/log/gds_log.txt”文件中。
  - 只允许IP为10.10.0.*的节点进行连接。
  - GDS进程以后台方式运行。
  - 递归数据文件目录。
  - 指定并发导入工作线程数目为32。

  b. 执行如下命令启动GDS并确认GDS是否启动成功。

  python gds_ctl.py start

  示例：

  ```
  cd /opt/bin/gds
  python gds_ctl.py start
  Start GDS gds1                [OK]
  gds [options]:
   -d dir           Set data directory.
   -p port           Set GDS listening port.
      ip:port       Set GDS listening ip address and port.
   -l log_file       Set log file.
   -H secure_ip_range
                    Set secure IP checklist in CIDR notation.                  Required for GDS to start.
   -e dir           Set error log directory.
   -E size          Set size of per error log segment.(0 < si                  ze < 1TB)
   -S size          Set size of data segment.(1MB < size < 10                  0TB)
   -t worker_num     Set number of worker thread in multi-thre                  ad mode, the upper
  limit is 32. If withou                  t setting, the default value is 1.
   -s status_file    Enable GDS status report.
  ```

```
-D            Run the GDS as a daemon process.
-r            Read the working directory recursively.
-h            Display usage.
```

----结束

gds.conf 参数说明

表 7-6 gds.conf 配置说明

属性	说明	取值范围
name	标识名。	-
ip	监听ip地址。	IP需为合法IP地址。 IP的默认值：127.0.0.1
port	监听端口号。	取值范围：1024~65535，正整数。 默认值：8098。
data_dir	数据文件目录。	-
err_dir	错误日志文件目录。	默认值：数据文件目录
log_file	日志文件路径。	-
host	设置允许连接到GDS的主机IP地址（参数为CIDR格式，仅支持linux系统）。	-
recursive	是否递归数据文件目录。	取值范围： ● true：递归 。 ● false：不递归。 默认值：false。
daemon	是否以DAEMON（后台）模式运行。	取值范围： ● true：以DAEMON模式运行。 ● false：不以DAEMON模式运行。 默认值：false。
parallel	导入工作线程并发数目。	取值范围：0~32，正整数。 默认值：32。

7.3.4 创建 GDS 外表

外表中配置了数据源格式信息、GDS服务的访问信息，从而GaussDB(DWS)最终可以通过外表将数据服务器上的数据引流进数据库实表中。

操作步骤

步骤1 收集数据源格式信息、GDS服务的访问信息。

需要收集的主要数据源格式信息如下：

- **format**：GDS外表导入支持CSV、TEXT和FIXED格式。请确认存放在数据服务器上待入库数据的格式。例如，假设待入库的数据为CSV格式。
- **header（仅支持CSV,FIXED格式）**：确认数据文件是否包含标题行。
- **delimiter**：确认数据文件中，字段间的分隔符。例如，假设是以英文逗号分隔的。
- **encoding**：数据源文件的数据编码格式。例如，假设为UTF-8。
- **eol**：确认数据文件中，行间的换行符。例如，默认的换行符，如0x0D0A、0X0A，或者自定义的换行符，如字符串!@#。该参数仅支持TEXT格式导入。
- 外表可识别的其他更多格式信息请参见CREATE FOREIGN TABLE (GDS导入导出)中的数据格式参数。

需要收集的GDS服务的访问信息如下：

location：GDS服务的访问地址。例如以**安装配置和启动GDS**中的GDS信息为例。非SSL模式下的location为：**gsfs**://*192.168.0.90:5000*/*/input_data/* 。SSL模式下的location为：**gsfss**://*192.168.0.90:5000*/*/input_data/* 。其中，“192.168.0.90:5000”为GDS服务的IP及端口号；“input_data”为GDS服务管理的数据源文件所在的路径。请根据实际情况替换。

步骤2 依据数据源文件中的数据情况，设计导入容错机制。

GaussDB(DWS)支持如下的数据容错性处理，相当于数据入库前对数据做初步的简单清洗。

- **fill_missing_fields**：数据入库时，数据源文件中某行的最后一个字段缺失时，请选择是直接将字段设为Null，还是在错误表中报错提示。
- **ignore_extra_data**：数据源文件中的字段比外表定义列数多时，请选择是忽略多出的列，还是在错误表中报错提示。
- **per node reject_limit**：本次数据导入过程中每个DN实例上允许出现的数据格式错误的数量。如果有一个DN实例上录入错误表中的错误数量超过设定值时，本次导入失败，报错退出。可以选择不做限制，也可以根据所能容忍的错误数量选择一个上限值。
- **compatible_illegal_chars**：导入时遇到非法字符，选择如何处理。是将非法字符按照转换规则转换后入库，还是报错中止导入。

  非法字符容错转换规则如下：

  - 对于'\0'，容错后转换为空格。
  - 对于其他非法字符，容错后转换为问号。
  - 对非法字符进行容错转换时，如遇NULL、DELIMITER、QUOTE、ESCAPE也设置成了空格或问号，GaussDB(DWS)会通过如"illegal chars conversion may confuse COPY escape 0x20"等报错信息提示用户修改可能引起混淆的参数以避免导入错误。
- **error_table_name**：用于记录数据格式错误信息的错误表表名。并行导入结束后查询此错误信息表，能够获取详细的错误信息。
- **remote log 'name'**：数据导入过程中的数据格式错误信息是否同时在GDS服务器上以文件方式保存。name为错误数据文件的文件名前缀。
- 关于容错性参数的更多信息请参考CREATE FOREIGN TABLE (GDS导入导出)中的容错性参数。

步骤3 使用gsql或Data Studio连接数据库后，根据前面步骤所收集和规划的信息参数，创建GDS外表。

示例如下：

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
)
 SERVER gsmpp_server
 OPTIONS
(
LOCATION 'gsfs://192.168.0.90:5000/input_data | gsfs://192.168.0.91:5000/input_data',
FORMAT 'CSV' ,
DELIMITER ',',
ENCODING 'utf8',
HEADER 'false',
FILL_MISSING_FIELDS 'true',
IGNORE_EXTRA_DATA 'true'
)
LOG INTO product_info_err
PER NODE REJECT LIMIT 'unlimited';
```

示例中的各项说明如下：

- 外表字段需与数据库中即将存储数据的事实表保持一致。
- 对于GDS导入，SERVER gsmpp_server请保持不变。
- location参数请根据**步骤1**中收集的GDS服务访问信息修改。注意GDS使用SSL加密传输时，需要将“gsfs”替换为“gsfss”。
- FORMAT、DELIMITER、ENCODING、HEADER请根据**步骤1**中收集的数据源格式信息填写。
- FILL_MISSING_FIELDS、IGNORE_EXTRA_DATA、LOG INTO及PER NODE REJECT LIMIT请根据**步骤2**中设计的导入容错机制填写。注意LOG INTO是指将数据格式错误录入哪个错误表，即其取值为错误表表名。

CREATE FOREIGN TABLE语法的更多信息，请参考CREATE FOREIGN TABLE (GDS导入导出)。

----结束

任务示例

除了以下示例，更多外表创建的示例请参考**GDS导入示例**。

- **示例1：**创建GDS外表foreign_tpcds_reasons，数据格式为CSV。

  ```
  CREATE FOREIGN TABLE foreign_tpcds_reasons
  (
    r_reason_sk integer not null,
    r_reason_id char(16) not null,
    r_reason_desc char(100)
  )
   SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*',
  FORMAT 'CSV',MODE 'Normal', ENCODING 'utf8', DELIMITER E'\x08', QUOTE E'\x1b', NULL '');
  ```

- **示例2：**创建GDS导入外表foreign_tpcds_reasons_SSL，使用SSL加密传输的模式传输，数据格式为CSV。

  ```
  CREATE FOREIGN TABLE foreign_tpcds_reasons_SSL
  (
    r_reason_sk integer not null,
    r_reason_id char(16) not null,
  ```

```
 r_reason_desc char(100)
)
 SERVER gsmpp_server OPTIONS (location 'gsfss://192.168.0.90:5000/* | gsfss://192.168.0.91:5000/*',
FORMAT 'CSV',MODE 'Normal', ENCODING 'utf8', DELIMITER E'\x08', QUOTE E'\x1b', NULL '');
```

- 示例3：创建GDS外表foreign_tpcds_reasons，数据格式为TEXT。

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*',
FORMAT 'TEXT', delimiter E'\x08',  null '',reject_limit '2',EOL '0x0D') WITH err_foreign_tpcds_reasons;
```

7.3.5 执行导入数据

完成GDS的安装部署及外表创建后，本节介绍如何在GaussDB(DWS)数据库中创建事实表并将数据导入事实表中。

对于记录数超过千万条的表，建议在执行全量数据导入前，先导入部分数据，以**进行数据倾斜检查和调整分布列**，避免导入大量数据后发现数据倾斜，调整成本高。

前提条件

GDS服务器和GaussDB(DWS)集群之间网络可以互通。

- 需要创建一个弹性云服务器作为GDS服务器。
- 创建的弹性云服务器与GaussDB(DWS)集群应处于同一区域、同一虚拟私有云和子网。

操作步骤

步骤1 在GaussDB(DWS)中创建目标表，用于存储导入的数据。建表语句请参见CREATE TABLE。

步骤2 （可选）若导入表存在索引，在数据导入过程中，将增量更新索引信息，影响数据导入性能。建议在执行数据导入前，先删除相关表的索引，但是如果不能保证数据唯一性不建议删除唯一索引。在数据导入完成后，再重新创建索引。

1. 假定在导入表“product_info”上的“product_id”字段上存在普通索引“product_idx”。在执行数据导入前，请先删除相关索引。

```
DROP INDEX product_idx;
```

2. 在数据导入完成后，重建索引。

```
CREATE INDEX product_idx ON product_info(product_id);
```

步骤3 执行数据导入。

```
INSERT INTO [目标表名] SELECT * FROM [foreign table 表名];
```

- 若出现以下类似信息，说明数据导入成功。请查询错误信息表，查看是否存在数据格式错误，详细操作请参见**处理错误表**。

```
INSERT 0 9
```

- 若出现数据加载错误，请参见**处理错误表**，并重新执行数据导入。

说明

- 若执行过程中出现数据加载错误，则数据全部导入失败，没有数据导入至目标表中。
- 编写批处理任务脚本，实现并发批量导入数据。并发量视机器资源使用情况而定。可通过几个表测试，监控资源利用率，根据结果提高或减少并发量。常用资源监控命令有：内存和CPU监控top命令，IO监控命令iostat，网络监控命令sar等。相关案例请参见**示例：多线程导入**。
- 在资源许可的情况下，多台GDS服务器并发导入会很大程度上提高数据导入效率。相关案例请参见**示例：多数据服务器并行导入**。
- 对于高并发的GDS导入场景，为了保持GDS和DN间的数据连接稳定，可以将GDS服务器环境和DN所在环境的TCP Keepalive检测时间增长（推荐增长至5分钟）。调整集群环境的TCP Keepalive参数会影响故障检测的响应时间。

----结束

任务示例

1. 创建一个名为reasons的目标表。

```
CREATE TABLE reasons
(
  r_reason_sk   integer  not null,
  r_reason_id   char(16) not null,
  r_reason_desc char(100)
)
DISTRIBUTE BY HASH (r_reason_sk);
```

2. 在执行数据导入前，先删除相关表的索引。

   假定在导入表“reasons”上的“r_reason_id”字段上存在普通索引“reasons_idx”。在执行数据导入前，请先删除相关索引。

```
DROP INDEX reasons_idx;
```

3. 将数据源文件中的数据通过外表“foreign_tpcds_reasons”导入到表“reasons”中。

```
INSERT INTO reasons SELECT * FROM foreign_tpcds_reasons ;
```

4. 在数据导入完成后，再重新创建索引。

```
CREATE INDEX reasons_idx ON reasons(r_reasons_id);
```

7.3.6 处理错误表

操作场景

当数据导入发生错误时，请根据本文指引信息进行处理。

查询错误信息

数据导入过程中发生的错误，一般分为数据格式错误和非数据格式错误。

- 数据格式错误

  在创建外表时，通过设置参数“LOG INTO error_table_name”，将数据导入过程中出现的数据格式错误信息写入指定的错误信息表error_table_name中。您可以通过以下SQL，查询详细错误信息。

```
SELECT * FROM error_table_name;
```

  错误信息表结构如**表7-7**所示。

表 7-7 错误信息表

列名称	类型	描述
nodeid	integer	报错节点编号。
begintime	timestamp with time zone	出现数据格式错误的时间。
filename	character varying	出现数据格式错误的数据源文件名。
rownum	bigint	在数据源文件中，出现数据格式错误的行号。
rawrecord	text	在数据源文件中，出现数据格式错误的原始记录。
detail	text	详细错误信息。

- 非数据格式错误

  对于非数据格式错误，一旦发生将导致整个数据导入失败。您可以根据执行数据导入过程中，界面提示的错误信息，帮助定位问题，处理错误表。

处理数据导入错误

根据获取的错误信息，请对照下表，处理数据导入错误。

表 7-8 处理数据导入错误

错误信息	原因	解决办法
missing data for column "r_reason_desc"	1. 数据源文件中的列数比外表定义的列数少。 2. 对于TEXT格式的数据源文件，由于转义字符（\）导致delimiter（分隔符）错位或者quote（引号字符）错位造成的错误。 **示例：**目标表存在3列字段，导入的数据如下所示。由于存在转义字符“\”，分隔符“|”被转义为第二个字段的字段值，导致第三个字段值缺失。 `BE|Belgium\\|1`	1. 由于列数少导致的报错，选择下列办法解决： ● 在数据源文件中，增加列“r_reason_desc”的字段值。 ● 在创建外表时，将参数“fill_missing_fields”设置为“on”。即当导入过程中，若数据源文件中一行数据的最后一个字段缺失，则把最后一个字段的值设置为NULL，不报错。 2. 对由于转义字符导致的错误，需检查报错的行中是否含有转义字符（\）。若存在，建议在创建外表时，将参数“noescaping”（是否不对'\'和后面的字符进行转义）设置为true。

错误信息	原因	解决办法
extra data after last expected column	数据源文件中的列数比外表定义的列数多。	• 在数据源文件中，删除多余的字段值。 • 在创建外表时，将参数“ignore_extra_data”设置为“on”。即在导入过程中，若数据源文件比外表定义的列数多，则忽略行尾多出来的列。
invalid input syntax for type numeric: "a"	数据类型错误。	在数据源文件中，修改输入字段的数据类型。根据此错误信息，请将输入的数据类型修改为numeric。
null value in column "staff_id" violates not-null constraint	非空约束。	在数据源文件中，增加非空字段信息。根据此错误信息，请增加“staff_id”列的值。
duplicate key value violates unique constraint "reg_id_pk"	唯一约束。	• 删除数据源文件中重复的行。 • 通过设置关键字“DISTINCT”，从SELECT结果集中删除重复的行，保证导入的每一行都是唯一的。 **INSERT INTO** reasons **SELECT DISTINCT * FROM** foreign_tpcds_reasons;
value too long for type character varying(16)	字段值长度超过限制。	在数据源文件中，修改字段值长度。根据此错误信息，字段值长度限制为VARCHAR2(16)。

7.3.7 停止 GDS

操作场景

待导入数据成功后，停止GDS。

操作步骤

步骤1 以gds_user用户登录安装GDS的数据服务器。

步骤2 请根据启动GDS的方式，选择停止GDS的方式。

- 若用户使用“gds”命令启动GDS，请使用以下方式停止GDS。

a. 执行如下命令，查询GDS进程号。
```
ps -ef|grep gds
```
示例：其中GDS进程号为128954。
```
ps -ef|grep gds
gds_user 128954     1  0 15:03 ?        00:00:00 gds -d /input_data/ -p 192.168.0.90:5000 -l /log/gds_log.txt -D
gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
```
b. 使用“kill”命令，停止GDS。其中128954为上一步骤中查询出的GDS进程号。
```
kill -9 128954
```

- 若用户使用“gds_ctl.py”命令启动GDS，请使用以下命令停止GDS。
```
cd /opt/bin/dws/gds
python gds_ctl.py stop
```

----结束

7.3.8 GDS 导入示例

示例：多数据服务器并行导入

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90和192.168.0.91。数据源文件格式为CSV。

1. 创建导入的目标表tpcds.reasons。
```
CREATE TABLE tpcds.reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
);
```
2. 以root用户登录每台GDS数据服务器，在两台数据服务器上，分别创建数据文件存放目录“/input_data”。以下以IP为192.168.0.90的数据服务器为例进行操作，剩余服务器上的操作与它一致。
```
mkdir -p /input_data
```
3. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该类用户及所属用户组已存在，可跳过此步骤。
```
groupadd gdsgrp
useradd -g gdsgrp gds_user
```
4. 将数据源文件均匀分发至相应数据服务器的“/input_data”目录中。
5. 修改每台数据服务器上数据文件及数据文件目录“/input_data”的属主为gds_user。以下以IP为192.168.0.90的数据服务器为例，进行操作。
```
chown -R gds_user:gdsgrp /input_data
```
6. 以gds_user用户登录每台数据服务器上分别启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，数据文件存放在“/input_data/”目录下，数据服务器所在IP为192.168.0.90和192.168.0.91，GDS监听端口为5000，以后台方式运行。

   在IP为192.168.0.90的数据服务器上启动GDS。
```
/opt/bin/dws/gds/bin/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D
```
   在IP为192.168.0.91的数据服务器上启动GDS。
```
/opt/bin/dws/gds/bin/gds -d /input_data -p 192.168.0.91:5000 -H 10.10.0.1/24  -D
```
7. 创建外表tpcds.foreign_tpcds_reasons用于接收数据服务器上的数据。

   其中设置**导入模式信息**如下所示：

   – 导入模式为Normal模式。

- 由于启动GDS时，设置的数据源文件存放目录为“/input_data”，GDS监听端口为5000，所以设置参数“location”为“gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*”。

设置**数据格式信息**是根据导出时设置的详细数据格式参数信息指定的，参数设置如下所示：

- 数据源文件格式（format）为CSV。
- 编码格式（encoding）为UTF-8。
- 字段分隔符（delimiter）为E'\x08'。
- 引号字符（quote）为0x1b。
- 数据文件中空值（null）为没有引号的空字符串。
- 逃逸字符（escape）为默认值双引号。
- 数据文件是否包含标题行（header）为默认值false，即导入时数据文件第一行被识别为数据。

设置**导入容错性**如下所示：

- 允许出现的数据格式错误个数（PER NODE REJECT LIMIT 'value'）为unlimited，即接受导入过程中所有数据格式错误。
- 将数据导入过程中出现的数据格式错误信息（LOG INTO error_table_name）写入表err_tpcds_reasons。

根据以上信息，创建的外表如下所示：

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
)
SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/* | gsfs://192.168.0.91:5000/*',
format 'CSV',mode 'Normal', encoding 'utf8', delimter E'\x08', quote E'\x1b', null '', fill_missing_fields
'false') LOG INTO err_tpcds_reasons PER NODE REJECT LIMIT 'unlimited';
```

8. 通过外表tpcds.foreign_tpcds_reasons，将数据导入目标表tpcds.reasons。

```
INSERT INTO tpcds.reasons SELECT * FROM tpcds.foreign_tpcds_reasons;
```

9. 查询错误信息表err_tpcds_reasons，处理数据导入错误。

```
SELECT * FROM err_tpcds_reasons;
```

10. 待数据导入完成后，以gds_user用户登录每台数据服务器，分别停止GDS。

    以下以IP为192.168.0.90的数据服务器为例，停止GDS。其中GDS进程号为128954。

```
ps -ef|grep gds
gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /input_data -p 192.168.0.90:5000 -D
gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
kill -9 128954
```

示例：多线程导入

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，导入的数据源文件格式为CSV，同时导入2个目标表。

1. 在数据库中创建导入的目标表tpcds.reasons1和tpcds.reasons2。

```
CREATE TABLE tpcds.reasons1
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) ;
CREATE TABLE tpcds.reasons2
(
```

```
 r_reason_sk integer not null,
 r_reason_id char(16) not null,
 r_reason_desc char(100)
) ;
```

2. 以root用户登录GDS数据服务器，创建数据文件存放目录“/input_data”，以及子目录“/input_data/import1/”和“/input_data/import2/”。
   ```
   mkdir -p /input_data
   ```
3. 将目标表tpcds.reasons1的数据源文件存放在数据服务器“/input_data/import1/”目录下，将目标表tpcds.reasons2的数据源文件存放在目录“/input_data/import2/”下。
4. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该用户及所属用户组已存在，可跳过此步骤。
   ```
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```
5. 修改数据服务器上数据文件及数据文件目录“/input_data”的属主为gds_user。
   ```
   chown -R gds_user:gdsgrp /input_data
   ```
6. 以gds_user用户登录数据服务器上启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，数据文件存放在“/input_data/”目录下，数据服务器所在IP为192.168.0.90，GDS监听端口为5000，以后台方式运行，设定并发度为2，并设定递归文件目录。
   ```
   /opt/bin/dws/gds/bin/gds -d /input_data -p 192.168.0.90:5000 -H 10.10.0.1/24  -D -t 2 -r
   ```
7. 在数据库中创建外表tpcds.foreign_tpcds_reasons1和tpcds.foreign_tpcds_reasons2用于接收数据服务器上的数据。

   以下以外表tpcds.foreign_tpcds_reasons1为例，讲解设置的导入外表参数信息。

   其中设置的**导入模式信息**如下所示：
   - 导入模式为Normal模式。
   - 由于启动GDS时，设置的数据源文件存放目录为“/input_data/”，GDS监听端口为5000，实际存放数据源文件目录为“/input_data/import1/”，所以设置参数“location”为“gsfs://192.168.0.90:5000/import1/*”。

   设置的**数据格式信息**是根据导出时设置的详细数据格式参数信息指定的，参数设置如下所示：
   - 数据源文件格式（format）为CSV。
   - 编码格式（encoding）为UTF-8。
   - 字段分隔符（delimiter）为E'\x08'。
   - 引号字符（quote）为0x1b。
   - 数据文件中空值（null）为没有引号的空字符串。
   - 逃逸字符（escape）为默认值双引号。
   - 数据文件是否包含标题行（header）为默认值false，即导入时数据文件第一行被识别为数据。

   设置的**导入容错性**如下所示：
   - 允许出现的数据格式错误个数（PER NODE REJECT LIMIT 'value'）为unlimited，即接受导入过程中所有数据格式错误。
   - 将数据导入过程中出现的数据格式错误信息（LOG INTO error_table_name）写入表err_tpcds_reasons1。
   - 当数据源文件中一行的最后一个字段缺失（fill_missing_fields）时，自动设置为NULL。

   根据以上信息，创建的外表tpcds.foreign_tpcds_reasons1如下所示：

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons1
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/import1/*', format 'CSV',mode
'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '',fill_missing_fields 'on')LOG INTO
err_tpcds_reasons1 PER NODE REJECT LIMIT 'unlimited';
```

参考以上设置，创建的外表tpcds.foreign_tpcds_reasons2如下所示：

```
CREATE FOREIGN TABLE tpcds.foreign_tpcds_reasons2
(
  r_reason_sk integer not null,
  r_reason_id char(16) not null,
  r_reason_desc char(100)
) SERVER gsmpp_server OPTIONS (location 'gsfs://192.168.0.90:5000/import2/*', format 'CSV',mode
'Normal', encoding 'utf8', delimiter E'\x08', quote E'\x1b', null '',fill_missing_fields 'on')LOG INTO
err_tpcds_reasons2 PER NODE REJECT LIMIT 'unlimited';
```

8. 通过外表tpcds.foreign_tpcds_reasons1和tpcds.foreign_tpcds_reasons2将数据分别导入tpcds.reasons1和tpcds.reasons2。

```
INSERT INTO tpcds.reasons1 SELECT * FROM tpcds.foreign_tpcds_reasons1;
INSERT INTO tpcds.reasons2 SELECT * FROM tpcds.foreign_tpcds_reasons2;
```

9. 查询错误信息表err_tpcds_reasons1和err_tpcds_reasons2，处理数据导入错误。

```
SELECT * FROM err_tpcds_reasons1;
SELECT * FROM err_tpcds_reasons2;
```

10. 待数据导入完成后，以gds_user用户登录数据服务器，停止GDS。

    其中GDS进程号为128954。

```
ps -ef|grep gds
gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /input_data -p 192.168.0.90:5000 -D -t 2 -r
gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
kill -9 128954
```

7.4 通过 INSERT 语句直接写入数据

用户可以通过以下方式执行INSERT语句直接向GaussDB(DWS)写入数据：

- 使用GaussDB(DWS)提供的客户端工具向GaussDB(DWS)写入数据。

  请参见向表中插入数据。

- 通过JDBC/ODBC驱动连接数据库执行INSERT语句向GaussDB(DWS)写入数据。

GaussDB(DWS)支持完整的数据库事务级别的增删改操作。INSERT是最简单的一种数据写入方式，这种方式适合数据写入量不大， 并发度不高的场景。

7.5 使用 COPY FROM STDIN 导入数据

7.5.1 关于 COPY FROM STDIN 导入数据

这种方式适合数据写入量不太大， 并发度不太高的场景。

用户可以使用以下方式通过COPY FROM STDIN语句直接向GaussDB(DWS)写入数据。

- 通过键盘输入向GaussDB(DWS)写入数据。
- 通过JDBC驱动的CopyManager接口从文件或者数据库向GaussDB(DWS)写入数据。此方法支持COPY语法中copy option的所有参数。

7.5.2 CopyManager 类简介

CopyManager是GaussDB(DWS) JDBC驱动中提供的一个API接口类，用于批量向GaussDB(DWS)集群中导入数据。

CopyManager 的继承关系

CopyManager类位于org.postgresql.copy Package中，继承自java.lang.Object类，该类的声明如下：

```
public class CopyManager
extends Object
```

构造方法

public CopyManager(BaseConnection connection)

throws SQLException

常用方法

表 7-9 CopyManager 常用方法

返回值	方法	描述	throws
CopyIn	copyIn(String sql)	-	SQLException
long	copyIn(String sql, InputStream from)	使用COPY FROM STDIN从InputStream中快速向数据库中的表导入数据。	SQLException,IOException
long	copyIn(String sql, InputStream from, int bufferSize)	使用COPY FROM STDIN从InputStream中快速向数据库中的表导入数据。	SQLException,IOException
long	copyIn(String sql, Reader from)	使用COPY FROM STDIN从Reader中快速向数据库中的表导入数据。	SQLException,IOException
long	copyIn(String sql, Reader from, int bufferSize)	使用COPY FROM STDIN从Reader中快速向数据库中的表导入数据。	SQLException,IOException
CopyOut	copyOut(String sql)	-	SQLException

返回值	方法	描述	throws
long	copyOut(String sql, OutputStream to)	将一个COPY TO STDOUT的结果集从数据库发送到OutputStream类中。	SQLException,IOException
long	copyOut(String sql, Writer to)	将一个COPY TO STDOUT的结果集从数据库发送到Writer类中。	SQLException,IOException

7.5.3 示例 1：通过本地文件导入导出数据

在使用JAVA语言基于GaussDB(DWS)进行二次开发时，可以使用CopyManager接口，通过流方式，将数据库中的数据导出到本地文件或者将本地文件导入数据库中，文件格式支持CSV、TEXT等格式。

样例程序如下，执行时需要加载GaussDB(DWS)的JDBC驱动。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.sql.SQLException;
import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Copy{

    public static void main(String[] args)
    {
     String urls = new String("jdbc:postgresql://localhost:8000/postgres"); //数据库URL
     String username = new String("username");            //用户名
     String password = new String("passwd");             //密码
     String tablename = new String("migration_table"); //定义表信息
     String tablename1 = new String("migration_table_1"); //定义表信息
     String driver = "org.postgresql.Driver";
     Connection conn = null;

     try {
           Class.forName(driver);
           conn = DriverManager.getConnection(urls, username, password);
         } catch (ClassNotFoundException e) {
             e.printStackTrace(System.out);
         } catch (SQLException e) {
             e.printStackTrace(System.out);
         }

     // 将表migration_table中数据导出到本地文件d:/data.txt
     try {
    copyToFile(conn, "d:/data.txt", "(SELECT * FROM migration_table)");
  } catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
  } catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
  }
     //将d:/data.txt中的数据导入到migration_table_1中。
```

```
    try {
    copyFromFile(conn, "d:/data.txt", tablename1);
 } catch (SQLException e) {
// TODO Auto-generated catch block
      e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

    // 将表migration_table_1中的数据导出到本地文件d:/data1.txt
    try {
    copyToFile(conn, "d:/data1.txt", tablename1);
 } catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
 } catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
   }

public static void copyFromFile(Connection connection, String filePath, String tableName)
      throws SQLException, IOException {

   FileInputStream fileInputStream = null;

   try {
      CopyManager copyManager = new CopyManager((BaseConnection)connection);
      fileInputStream = new FileInputStream(filePath);
      copyManager.copyIn("COPY " + tableName + " FROM STDIN with (" + "DELIMITER"+"'"+ delimiter +
"'" + "ENCODING " + "'" + encoding + "')", fileInputStream);
   } finally {
      if (fileInputStream != null) {
         try {
            fileInputStream.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
}
   public static void copyToFile(Connection connection, String filePath, String tableOrQuery)
       throws SQLException, IOException {

    FileOutputStream fileOutputStream = null;

    try {
       CopyManager copyManager = new CopyManager((BaseConnection)connection);
       fileOutputStream = new FileOutputStream(filePath);
       copyManager.copyOut("COPY " + tableOrQuery + " TO STDOUT", fileOutputStream);
    } finally {
       if (fileOutputStream != null) {
          try {
             fileOutputStream.close();
          } catch (IOException e) {
             e.printStackTrace();
          }
       }
    }
 }
}
```

7.5.4 示例 2：从 MySQL 向 GaussDB(DWS)进行数据迁移

下面示例演示如何通过CopyManager从MySQL向GaussDB(DWS)进行数据迁移的过程。

```
import java.io.StringReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Migration{

    public static void main(String[] args) {
        String url = new String("jdbc:postgresql://localhost:8000/postgres"); //数据库URL
        String user = new String("username");            //GaussDB(DWS)用户名
        String pass = new String("passwd");             //GaussDB(DWS)密码
        String tablename = new String("migration_table_1"); //定义表信息
        String delimiter = new String("|");              //定义分隔符
        String encoding = new String("UTF8");            //定义字符集
        String driver = "org.postgresql.Driver";
        StringBuffer buffer = new StringBuffer();       //定义存放格式化数据的缓存

        try {
            //获取源数据库查询结果集
            ResultSet rs = getDataSet();

            //遍历结果集，逐行获取记录
            //将每条记录中各字段值，按指定分隔符分割，由换行符结束，拼成一个字符串
            //把拼成的字符串，添加到缓存buffer
            while (rs.next()) {
                buffer.append(rs.getString(1) + delimiter
                        + rs.getString(2) + delimiter
                        + rs.getString(3) + delimiter
                        + rs.getString(4)
                        + "\n");
            }
            rs.close();

            try {
                //建立目标数据库连接
                Class.forName(driver);
                Connection conn = DriverManager.getConnection(url, user, pass);
                BaseConnection baseConn = (BaseConnection) conn;
                baseConn.setAutoCommit(false);

                //初始化表信息
                String sql = "Copy " + tablename + " from STDIN with (DELIMITER " + "'" + delimiter + "'" +","+ "
ENCODING " + "'" + encoding + "')";

                //提交缓存buffer中的数据
                CopyManager cp = new CopyManager(baseConn);
                StringReader reader = new StringReader(buffer.toString());
                cp.copyIn(sql, reader);
                baseConn.commit();
                reader.close();
                baseConn.close();
            } catch (ClassNotFoundException e) {
                e.printStackTrace(System.out);
            } catch (SQLException e) {
                e.printStackTrace(System.out);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //********************************
    // 从源数据库返回查询结果集
```

```
    //**********************************
    private static ResultSet getDataSet() {
        ResultSet rs = null;
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            Connection conn = DriverManager.getConnection("jdbc:mysql://10.119.179.227:3306/jack?
useSSL=false&allowPublicKeyRetrieval=true", "jack", "Gauss@123");
            Statement stmt = conn.createStatement();
            rs = stmt.executeQuery("select * from migration_table");
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return rs;
    }
}
```

7.6 使用 gsql 元命令导入数据

GaussDB(DWS)的gsql工具提供了元命令\copy进行数据导入。

\copy 命令

\copy命令格式以及说明参见**表 1 \copy元命令说明**。

表 7-10 \copy 元命令说明

语法	说明
\copy { table [(column_list)] | (query) } { from | to } { filename | stdin | stdout | pstdin | pstdout } [with] [binary] [oids] [delimiter [as] 'character'] [null [as] 'string'] [csv [header] [quote [as] 'character'] [escape [as] 'character'] [force quote column_list | *] [force not null column_list]]	在任何gsql客户端登录数据库成功后，可以使用该命令进行数据的导入/导出。但是与SQL的COPY命令不同，该命令读取/写入的文件是本地文件，而非数据库服务器端文件；所以，要操作的文件的可访问性、权限等，都是受限于本地用户的权限。 **说明** \COPY只适合小批量，格式良好的数据导入，容错能力较差。导入数据应优先选择GDS或COPY。

参数说明

- table
  表的名字（可以有模式修饰）。
  取值范围：已存在的表名。
- column_list
  可选的待拷贝字段列表。
  取值范围：任意字段。如果没有声明字段列表，将使用所有字段。
- query

其结果将被拷贝。

取值范围：一个必须用圆括弧包围的SELECT或VALUES命令。

- filename

  文件的绝对路径。执行copy命令的用户必须有此路径的写权限。

- stdin

  声明输入是来自标准输入。

- stdout

  声明输出打印到标准输出。

- pstdin

  声明输入是来自gsql的标准输入。

- pstout
- 声明输出打印到gsql的标准输出。
- binary

  使用二进制格式存储和读取，而不是以文本的方式。在二进制模式下，不能声明DELIMITER，NULL，CSV选项。指定binary类型后，不能再通过option或copy_option指定CSV、FIXED、TEXT等类型。

- oid

  为每行拷贝内部对象标识（oid）。

  说明

  若COPY FROM对象为query或者对于没有oid的表，指定oids标识报错。

  取值范围：true/on，false/off。

  默认值：false

- delimiter [as] 'character'

  指定数据文件行数据的字段分隔符。

  说明

  - 分隔符不能是\r和\n。
  - 分隔符不能和null参数相同，CSV格式数据的分隔符不能和quote参数相同。
  - TEXT格式数据的分隔符不能包含： \.abcdefghijklmnopqrstuvwxyz0123456789。
  - 数据文件中单行数据长度需<1GB，如果分隔符较长且数据列较多的情况下，会影响导出有效数据的长度。
  - 分隔符推荐使用多字符和不可见字符。多字符例如'$^&'；不可见字符例如0x07，0x08，0x1b等。

  取值范围：支持多字符分隔符，但分隔符不能超过10个字节。

  默认值：

  - TEXT格式的默认分隔符是水平制表符（tab）。
  - CSV格式的默认分隔符为“,”。
  - FIXED格式没有分隔符。

- null [as] 'string'

  用来指定数据文件中空值的表示。

  取值范围：

  - null值不能是\r和\n，最大为100个字符。
  - null值不能和分隔符、quote参数相同。

  默认值：

  - CSV格式下默认值是一个没有引号的空字符串。
  - 在TEXT格式下默认值是\N。

- header

  指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。header只能用于CSV，FIXED格式的文件中。

  在导入数据时，如果header选项为on，则数据文本第一行会被识别为标题行，会忽略此行。如果header为off，而数据文件中第一行会被识别为数据。

  在导出数据时，如果header选项为on，则需要指定fileheader。fileheader是指定导出数据包含标题行的定义文件。如果header为off，则导出数据文件不包含标题行。

  取值范围：true/on，false/off。

  默认值：false

- quote [as] 'character'

  CSV格式文件下的引号字符。

  默认值：双引号。

  说明

  - quote参数不能和分隔符、null参数相同。
  - quote参数只能是单字节的字符。
  - 推荐不可见字符作为quote，例如0x07，0x08，0x1b等。

- escape [as] 'character'

  CSV格式下，用来指定逃逸字符，逃逸字符只能指定为单字节字符。

  默认值：双引号。当与quote值相同时，会被替换为'\0'。

- force quote column_list | *

  在CSV COPY TO模式下，强制在每个声明的字段周围对所有非NULL值都使用引号包围。NULL输出不会被引号包围。

  取值范围：已存在的字段。

- force not null column_list

  在CSV COPY FROM模式下，指定的字段输入不能为空。

  取值范围：已存在的字段。

任务示例

1. 创建目标表a。

   ```
   CREATE TABLE a(a int);
   ```

2. 导入数据。

   a. 从stdin拷贝数据到目标表a。

   ```
   \copy a from stdin;
   ```

   出现>>符号提示时，输入数据，输入\.时结束。

   ```
   Enter data to be copied followed by a newline.
   End with a backslash and a period on a line by itself.
   >> 1
   ```

```
>> 2
>> \.
```

查询导入目标表a的数据。

```
SELECT * FROM a;
 a
---
 1
 2
(2 rows)
```

b. 从本地文件拷贝数据到目标表a。假设存在本地文件/home/omm/2.csv。

- 分隔符为‘，’。
- 在导入过程中，若数据源文件比外表定义的列数多，则忽略行尾多出来的列。

```
\copy a FROM '/home/omm/2.csv' WITH (delimiter',',IGNORE_EXTRA_DATA 'on');
```

7.7 从 MRS 导入数据到集群

7.7.1 从 MRS 导入数据概述

MapReduce服务（MapReduce Service，简称MRS）是一个基于开源Hadoop生态环境而运行的大数据集群，对外提供大容量数据的存储和分析能力，可解决用户的数据存储和处理需求。具体信息可参考**《MapReduce服务用户指南》**。

用户可以将海量业务数据，存储在MRS的分析集群，即使用Hive/Spark组件保存。Hive/Spark的数据文件则保存在HDFS中。GaussDB(DWS)支持在相同网络中，配置一个GaussDB(DWS)集群连接到一个MRS集群，然后将数据从HDFS中的文件读取到GaussDB(DWS)。

从 MRS 导入数据到集群的流程

1. **MRS集群上的数据准备**
2. （可选）**手动创建外部服务器**
3. **创建外表**
4. **执行数据导入**
5. **清除资源**

7.7.2 MRS 集群上的数据准备

从MRS导入数据到GaussDB(DWS)集群之前，假设您已经完成了以下准备工作：

1. 已创建MRS集群。
2. 在MRS集群上创建了Hive/Spark ORC表，且表数据已经存储到该表对应的HDFS路径上。

如果您已经完成上述准备，则可以跳过本章节。

为方便起见，我们将以在MRS集群上创建Hive ORC表作为示例，完成上述准备工作。在MRS集群上创建Spark ORC表的大致流程和SQL语法，同Hive类似，在本文中不再展开描述。

数据文件

假设有数据文件product_info.txt，示例数据如下所示：

```
100,XHDK-A-1293-#fJ3,2017-09-01,A,2017 Autumn New Shirt Women,red,M,328,2017-09-04,715,good
205,KDKE-B-9947-#kL5,2017-09-01,A,2017 Autumn New Knitwear Women,pink,L,584,2017-09-05,406,very good!
300,JODL-X-1937-#pV7,2017-09-01,A,2017 autumn new T-shirt men,red,XL,1245,2017-09-03,502,Bad.
310,QQPX-R-3956-#aD8,2017-09-02,B,2017 autumn new jacket women,red,L,411,2017-09-05,436,It's really super nice
150,ABEF-C-1820-#mC6,2017-09-03,B,2017 Autumn New Jeans Women,blue,M,1223,2017-09-06,1200,The seller's packaging is exquisite
200,BCQP-E-2365-#qE4,2017-09-04,B,2017 autumn new casual pants men,black,L,997,2017-09-10,301,The clothes are of good quality.
250,EABE-D-1476-#oB1,2017-09-10,A,2017 autumn new dress women,black,S,841,2017-09-15,299,Follow the store for a long time.
108,CDXK-F-1527-#pL2,2017-09-11,A,2017 autumn new dress women,red,M,85,2017-09-14,22,It's really amazing to buy
450,MMCE-H-4728-#nP9,2017-09-11,A,2017 autumn new jacket women,white,M,114,2017-09-14,22,Open the package and the clothes have no odor
260,OCDA-G-2817-#bD3,2017-09-12,B,2017 autumn new woolen coat women,red,L,
2004,2017-09-15,826,Very favorite clothes
980,ZKDS-J-5490-#cW4,2017-09-13,B,2017 Autumn New Women's Cotton Clothing,red,M,
112,2017-09-16,219,The clothes are small
98,FKQB-I-2564-#dA5,2017-09-15,B,2017 autumn new shoes men,green,M,4345,2017-09-18,5473,The clothes are thick and it's better this winter.
150,DMQY-K-6579-#eS6,2017-09-21,A,2017 autumn new underwear men,yellow,
37,2840,2017-09-25,5831,This price is very cost effective
200,GKLW-l-2897-#wQ7,2017-09-22,A,2017 Autumn New Jeans Men,blue,39,5879,2017-09-25,7200,The clothes are very comfortable to wear
300,HWEC-L-2531-#xP8,2017-09-23,A,2017 autumn new shoes women,brown,M,403,2017-09-26,607,good
100,IQPD-M-3214-#yQ1,2017-09-24,B,2017 Autumn New Wide Leg Pants Women,black,M,
3045,2017-09-27,5021,very good.
350,LPEC-N-4572-#zX2,2017-09-25,B,2017 Autumn New Underwear Women,red,M,239,2017-09-28,407,The seller's service is very good
110,NQAB-O-3768-#sM3,2017-09-26,B,2017 autumn new underwear women,red,S,
6089,2017-09-29,7021,The color is very good
210,HWNB-P-7879-#tN4,2017-09-27,B,2017 autumn new underwear women,red,L,3201,2017-09-30,4059,I like it very much and the quality is good.
230,JKHU-Q-8865-#uO5,2017-09-29,C,2017 Autumn New Clothes with Chiffon Shirt,black,M,
2056,2017-10-02,3842,very good
```

在 MRS 集群上创建 Hive ORC 表

1. 创建了MRS集群。
   具体操作请参见《 数据仓库服务管理指南 》的**创建MRS数据源连接**。
2. 登录MRS集群的Hive客户端。
   a. 登录Master节点。
      具体操作，请参见《 MapReduce服务用户指南 》中的**登录集群节点**章节。
   b. 执行以下命令切换用户。
      ```
      sudo su - omm
      ```
   c. 执行以下命令切换到客户端目录：
      ```
      cd /opt/client
      ```
   d. 执行以下命令配置环境变量：
      ```
      source bigdata_env
      ```
   e. 如果当前集群已启用Kerberos认证，执行以下命令认证当前用户，当前用户需要具有创建Hive表的权限，具体操作请参见《 MapReduce服务用户指南 》的**创建角色**。配置拥有对应权限的角色，具体操作请参见《 MapReduce服务用户指南 》的**创建用户**。为用户绑定对应角色。如果当前集群未启用Kerberos认证，则无需执行此命令。

```
kinit MRS集群用户
```

例如，kinit hiveuser。

f. 执行以下命令启动Hive客户端：

```
beeline
```

3. 在Hive中创建数据库demo。

执行以下命令创建数据库：

```
CREATE DATABASE demo;
```

4. 在数据库demo中新建了一个Hive TEXTFILE类型的表product_info，并将**数据文件**（product_info.txt）导入到该表对应的HDFS路径中。

执行以下命令切换到demo数据库：

```
USE demo;
```

执行以下命令，创建表product_info，表字段按照**数据文件**中的数据进行定义：

```
DROP TABLE product_info;

CREATE TABLE product_info
(
    product_price               int           not null,
    product_id                  char(30)      not null,
    product_time                date          ,
    product_level               char(10)      ,
    product_name                varchar(200)  ,
    product_type1               varchar(20)   ,
    product_type2               char(10)      ,
    product_monthly_sales_cnt   int           ,
    product_comment_time        date          ,
    product_comment_num         int           ,
    product_comment_content     varchar(200)
)
row format delimited fields terminated by ','
stored as TEXTFILE;
```

有关导入数据到MRS集群的操作，请参见《MapReduce服务用户指南》中的“**管理现有集群** > 管理数据文件”章节。

5. 在数据库demo中创建了一个Hive ORC表product_info_orc。

执行以下命令，创建Hive ORC表product_info_orc，表字段与上一步创建的表product_info完全一致：

```
DROP TABLE product_info_orc;

CREATE TABLE product_info_orc
(
    product_price               int           not null,
    product_id                  char(30)      not null,
    product_time                date          ,
    product_level               char(10)      ,
    product_name                varchar(200)  ,
    product_type1               varchar(20)   ,
    product_type2               char(10)      ,
    product_monthly_sales_cnt   int           ,
    product_comment_time        date          ,
    product_comment_num         int           ,
    product_comment_content     varchar(200)
)
row format delimited fields terminated by ','
stored as orc;
```

6. 将product_info表的数据插入到Hive ORC表product_info_orc中。

```
insert into product_info_orc select * from product_info;
```

查询表product_info_orc：

```
select * from product_info_orc;
```

如果查询到如**数据文件**所示的数据，表示已经成功将数据插入到ORC表。

7.7.3 手动创建外部服务器

创建外表语法（CREATE FOREIGN TABLE (SQL on Hadoop or OBS)）中，需指定一个与MRS数据源连接相关联的外部服务器。

当您通过GaussDB(DWS)管理控制台创建MRS数据源连接时，数据库管理员dbadmin会在默认数据库postgres中自动创建一个外部服务器。因此，如果您希望在默认数据库postgres中创建外表读取MRS数据，可以跳过本章节。

如果您希望使用普通用户在自定义数据库中创建外表读取MRS数据，必须先在自定义数据库中手动创建一个外部服务器。本章节将为您介绍，如何使用普通用户在自定义数据库中创建外部服务器。步骤如下：

1. 请确保GaussDB(DWS)集群已创建MRS数据源连接。
   具体操作请参见《数据仓库服务管理指南》的**创建MRS数据源连接**。
2. **创建用户和数据库并授予外表权限**
3. **手动创建外部服务器**

说明

需要注意的是，当您不再需要从该MRS数据源读取数据时，通过GaussDB(DWS)管理控制台删除MRS数据源，仅会删除在默认数据库postgres中自动创建的外部服务器，手动创建的外部服务器需要您手动删除，具体操作请参见**删除手动创建的外部服务器**中的描述。

创建用户和数据库并授予外表权限

以下示例，是新建一个普通用户dbuser并创建一个数据库mydatabase，然后使用管理员用户授予dbuser外表权限。

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres

例如，使用gsql客户端的用户通过如下语句连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 新建一个普通用户，并用它创建一个数据库。

新建一个具有创建数据库权限的用户dbuser：

```
CREATE USER dbuser WITH CREATEDB PASSWORD "Bigdata@123";
```

切换为新建的用户：

```
SET ROLE dbuser PASSWORD "Bigdata@123";
```

执行以下命令创建数据库：

```
CREATE DATABASE mydatabase;
```

查询数据库：

```
SELECT * FROM pg_database;
```

返回结果中有**mydatabase** 的信息表示创建成功：

```
datname   | datdba | encoding | datcollate | datctype | datistemplate | datallowconn | datconnlimit |
datlastsysoid | datfrozenxid | dattablespace | datcompatibility |                    datacl

------------+--------+----------+------------+----------+---------------+--------------+--------------+---------------
+--------------+---------------+-----------------+-----------------------------------
--------------
```

```
 template1  |     10 |        0 | C          | C        | t            | t           |          -1 |        14146 |        1351
|          1663 | ORA             | {=c/Ruby,Ruby=CTc/Ruby}
 template0  |     10 |        0 | C          | C        | t            | f           |          -1 |        14146 |        1350
|          1663 | ORA             | {=c/Ruby,Ruby=CTc/Ruby}
 postgres   |     10 |        0 | C          | C        | f            | t           |          -1 |        14146 |        1352 |
1663 | ORA             | {=Tc/Ruby,Ruby=CTc/Ruby,chaojun=C/Ruby,hu
obinru=C/Ruby}
 mydatabase |  17000 |        0 | C          | C        | f            | t           |          -1 |        14146 |        1351
|          1663 | ORA             |
(4 rows)
```

步骤3 使用管理员用户给普通用户赋予创建外部服务器的权限和使用外表的权限。

使用数据库管理员用户通过数据库客户端连接新建的数据库。

例如，使用gsql客户端的用户可以直接使用如下语句切换为管理员用户去连接新建的数据库：

```
\c mydatabase dbadmin;
```

根据提示输入用户密码。

说明

注意，必须先使用管理员用户连接到**将要创建外部服务器和使用外表的数据库**，再对普通用户进行授权。

默认只有系统管理员才可以创建外部服务器，普通用户需要授权才可以创建，执行以下命令授权：

```
GRANT ALL ON FOREIGN DATA WRAPPER hdfs_fdw TO dbuser;
```

其中FOREIGN DATA WRAPPER的名字只能是hdfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令赋予用户使用外表的权限。

```
ALTER USER dbuser USEFT;
```

查看用户：

```
SELECT r.rolname, r.rolsuper, r.rolinherit,
  r.rolcreaterole, r.rolcreatedb, r.rolcanlogin,
  r.rolconnlimit, r.rolvalidbegin, r.rolvaliduntil,
  ARRAY(SELECT b.rolname
        FROM pg_catalog.pg_auth_members m
        JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid)
        WHERE m.member = r.oid) as memberof
, r.rolreplication
, r.rolauditadmin
, r.rolsystemadmin
, r.roluseft
FROM pg_catalog.pg_roles r
ORDER BY 1;
```

返回结果中，dbuser的信息中包含了UseFT权限，表示授权成功：

```
 rolname  | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcanlogin | rolconnlimit | rolvalidbegin |
rolvaliduntil | memberof | rolreplication | rolauditadmin | rolsystemadmin | roluseft
----------+----------+------------+---------------+-------------+-------------+--------------+---------------
+---------------+----------+----------------+---------------+----------------+----------
 dbuser   | f        | t          | f             | t           | t           |           -1 |               |               | {}       | f
| f             | f              | t
 lily     | f        | t          | f             | f           | t           |           -1 |               |               | {}       | f             |
f             | f              | f
 Ruby     | t        | t          | t             | t           | t           |           -1 |               |               | {}       | t
| t             | t              | t
```

----结束

手动创建外部服务器

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

例如：通过gsql客户端登录数据库的用户可以使用以下两种方法中的一种进行连接：

可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令切换数据库和用户：
  ```
  \c postgres dbadmin;
  ```
  根据提示输入密码。
- 如果尚未登录gsql客户端，或者已经登录了gsql客户端执行**\q**退出gsql后，执行以下命令重新进行连接：
  ```
  gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
  ```
  根据提示输入密码。

步骤2 执行以下命令查询自动创建的外部服务器的信息。

```
SELECT * FROM pg_foreign_server;
```

返回结果如：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                        srvoptions
-----------------------------------------------+----------+--------+---------+------------+--------
+-----------------------------------------------------------------------------------------------------------------------------------------------------
 gsmpp_server                                 |       10 |  13673 |         |            |        |
 gsmpp_errorinfo_server                       |       10 |  13678 |         |            |        |
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |         |            |        |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(3 rows)
```

查询结果中，每一行代表一个外部服务器的信息。与MRS数据源连接相关联的外部服务器包含以下信息：

- srvname值包含“hdfs_server”字样以及MRS集群的ID，此ID与MRS管理控制台的集群列表MRS ID相同。
- srvoptions字段中的address参数为MRS集群的主备节点的IP地址及端口。

您可以根据上述信息找到您所要的外部服务器，并记录下它的srvname和srvoptions的值。

步骤3 切换为即将创建外部服务器的用户去连接其对应的数据库。

在本示例中，执行以下命令，使用**创建用户和数据库并授予外表权限**中创建的普通用户dbuser连接其创建的数据库mydatabase 。

```
\c mydatabase dbuser;
```

步骤4 创建外部服务器。

创建外部服务器的详细语法，请参见CREATE SERVER。示例如下：

```
CREATE SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca FOREIGN DATA WRAPPER
HDFS_FDW
OPTIONS
(
address '192.168.1.245:25000,192.168.1.218:25000',
hdfscfgpath '/MRS/8f79ada0-d998-4026-9020-80d6de2692ca',
type 'hdfs'
);
```

以下为必选参数的说明：

- 外部服务器名称

  允许用户自定义名字。

  在本例中，我们指定为前面的步骤**步骤2**中记录下来的srvname字段的值，如'*hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca*'。

  不同的数据库之间资源是隔离的，因此在不同的数据库中外部服务器名称可以相同。

- FOREIGN DATA WRAPPER

  只能指定为HDFS_FDW，它在数据库中已经存在。

- OPTIONS参数

  以下参数请分别指定为步骤**步骤2**中记录下来的srvoptions中的参数值。

  – address

    指定HDFS集群的主备节点所在的IP地址以及端口。

  – hdfscfgpath

    指定HDFS集群配置文件路径。该参数仅支持type为HDFS时设置。只能设置一个路径。

  – type

    取值为'hdfs'，表示HDFS_FDW连接的是HDFS。

步骤5 查看外部服务器。

```
SELECT * FROM pg_foreign_server WHERE
srvname='hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca';
```

返回结果如下所示，表示已经创建成功：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                srvoptions
----------------------------------------------------+----------+--------+---------+------------+--------
+-------------------------------------------------------------------------------------------------------------
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |         |            |      |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(1 row)
```

----结束

7.7.4 创建外表

在GaussDB(DWS)数据库中创建一个Hadoop外表，用来访问存储在MRS HDFS文件系统上的Hadoop结构化数据。Hadoop外表是只读的，只能用于查询操作，可直接使用SELECT查询其数据。

您可以按照以下步骤创建外表：

1. 请确保您已经完成**前提条件**
2. 根据创建外表（CREATE FOREIGN TABLE (SQL on Hadoop or OBS)）的语法描述，需要先获取以下信息：
   a. **获取MRS数据源的HDFS路径**
   b. **获取MRS数据源连接的外部服务器信息**
3. **创建外表**

   参考信息：**数据类型转换说明**

前提条件

- 已创建MRS集群，并将数据导入Hive/Spark数据库中的ORC表。
  请参见**MRS集群上的数据准备**。
- GaussDB(DWS)集群已创建MRS数据源连接。
  具体操作请参见《数据仓库服务管理指南》的**创建MRS数据源连接**。

获取 MRS 数据源的 HDFS 路径

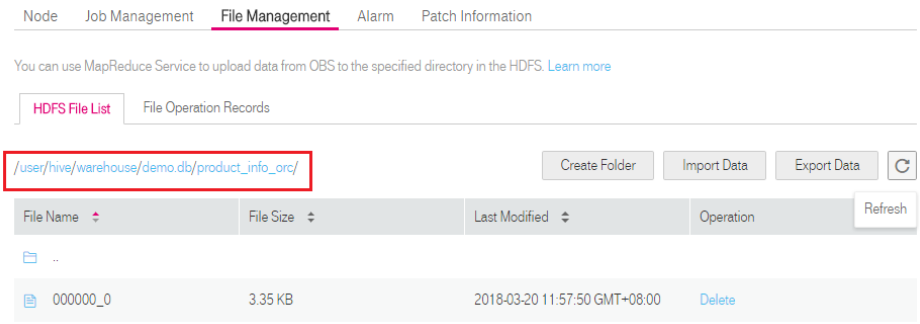
有两种方法可以查看：

- **方法一：**
  对于Hive数据，可以登录MRS的Hive客户端（参见2），执行以下命令查看表的详细信息，并记录下location参数中的数据存储路径。

  ```
  use <database_name>;
  desc formatted <table_name>;
  ```

  例如，返回结果中location参数值为“hdfs://hacluster/user/hive/warehouse/demo.db/product_info_orc/”，则记录HDFS路径为“/user/hive/warehouse/demo.db/product_info_orc/”。
- **方法二：**
  按以下步骤获取HDFS路径。
  a. 登录MRS管理控制台。
  b. 选择“集群列表 > 现有集群”，单击要查看的集群名称，进入集群基本信息页面。
  c. 单击“文件管理”，选择“HDFS文件列表”。
  d. 进入您要导入到GaussDB(DWS)集群的数据的存储目录，并记录其路径。

  图 7-7 在 MRS 上查看数据存储路径



获取 MRS 数据源连接的外部服务器信息

步骤1 使用创建外部服务器的用户去连接其对应的数据库。

是否使用普通用户在自定义数据库中创建外表，请根据需求进行选择：

- **是**
  a. 请先确保，您已按照**手动创建外部服务器**章节中的步骤，创建了普通用户dbuser和它的数据库mydatabase，并在mydatabase中手动创建了一个外部服务器。

b. 使用用户dbuser通过GaussDB(DWS)提供的数据库客户端连接数据库mydatabase。

如果已经使用gsql客户端连接至数据库，可以直接执行如下命令进行用户和数据库切换：

```
\c mydatabase dbuser;
```

根据界面提示输入密码。

- **否**

当您通过GaussDB(DWS)管理控制台创建MRS数据源连接时，数据库管理员dbadmin会在默认数据库postgres中自动创建一个外部服务器。因此，如果使用数据库管理员dbadmin在默认数据库postgres中创建外表，需要通过GaussDB(DWS)提供的数据库客户端工具连接数据库。例如，使用gsql客户端的用户通过如下命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 执行以下命令，查看已创建的MRS数据源连接的外部服务器信息。

```
SELECT * FROM pg_foreign_server;
```

说明

也可以执行**\desc+**命令查看外部服务器信息。

返回结果如：

```
                srvname                 | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                        srvoptions
-------------------------------------------------+----------+--------+---------+------------+--------
+-----------------------------------------------------------------------------------------------------------------------
 gsmpp_server                                   |       10 |  13673 |         |            |
 gsmpp_errorinfo_server                         |       10 |  13678 |         |            |
 hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca |    16476 |  13685 |         |            |
{"address=192.168.1.245:25000,192.168.1.218:25000",hdfscfgpath=/MRS/8f79ada0-
d998-4026-9020-80d6de2692ca,type=hdfs}
(3 rows)
```

查询结果中，每一行代表一个外部服务器的信息。与MRS数据源连接相关联的外部服务器包含以下信息：

- srvname值包含“hdfs_server”字样以及MRS集群的ID，此ID与MRS管理控制台的集群列表MRS ID相同。
- srvoptions字段中的address参数为MRS集群的主备节点的IP地址及端口。

您可以根据上述信息找到您所要的外部服务器，并记录下它的srvname和srvoptions的值。

----结束

创建外表

当完成**获取MRS数据源连接的外部服务器信息**和**获取MRS数据源的HDFS路径**后，就可以创建一个外表，用于读取MRS数据源数据。

创建外表的语法格式如下，详细的描述请参见CREATE FOREIGN TABLE (SQL on Hadoop or OBS)。

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( [ { column_name type_name
    [ { [CONSTRAINT constraint_name] NULL |
```

```
  [CONSTRAINT constraint_name] NOT NULL |
    column_constraint [...]} ] |
    table_constraint [, ...]} [, ...] ] )
  SERVER dfs_server
  OPTIONS ( { option_name ' value ' } [, ...] )
  DISTRIBUTE BY {ROUNDROBIN | REPLICATION}
  [ PARTITION BY ( column_name ) [ AUTOMAPPED ] ] ;
```

例如，创建一个名为"*foreign_product_info*"的外表，对语法中的参数按如下描述进行设置：

- **table_name**

  必选。外表的表名。

- **表字段定义**
  - **column_name**：外表中的字段名。
  - **type_name**：字段的数据类型。

  多个字段用“,”隔开。

  外表的字段个数和字段类型，需要与MRS上保存的数据完全一致。定义字段的数据类型之前，您必须先了解**数据类型转换说明**。

- **SERVER dfs_server**

  外表的外部服务器名称，这个server必须存在。外表通过设置外部服务器，从而关联MRS数据源连接并从MRS集群读取数据。

  此处应填写为通过**获取MRS数据源连接的外部服务器信息**查询到的“srvname”字段的值。

- **OPTIONS 参数**

  用于指定外表数据的各类参数，关键参数如下所示。
  - **format**：必选参数。取值只支持“orc”。表示数据源文件的格式，只支持Hive的ORC数据文件。
  - **foldername**：必选参数。表示数据在HDFS的存储目录或数据文件路径。

    如果是启用了Kerberos认证的MRS分析集群，请确保MRS数据源连接的MRS用户，拥有此目录的读取权限。

    请按照**获取MRS数据源的HDFS路径**中的步骤获取HDFS路径，该路径作为**foldername**的参数值。
  - **encoding**：可选参数。外表中数据源文件的编码格式名称，缺省为utf8。
  - **DISTRIBUTE BY**

    表示外表的数据读取方式。有以下两种方式供选择，在本例中我们选择ROUNDROBIN。
    - **ROUNDROBIN**：表示外表在从数据源读取数据时，GaussDB(DWS)集群每一个节点读取随机一部分数据，并组成完整数据。
    - **REPLICATION**：表示外表在从数据源读取数据时，GaussDB(DWS)集群每一个节点都读取一份完整数据。
  - **语法中的其他参数**

    其他参数均为可选参数，用户可以根据自己的需求进行设置，在本例中我们不需要设置。详细的描述请参见CREATE FOREIGN TABLE (SQL on Hadoop or OBS)。

根据以上信息，创建外表命令如下所示：

```
DROP FOREIGN TABLE IF EXISTS foreign_product_info;

CREATE FOREIGN TABLE foreign_product_info
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca
OPTIONS (
format 'orc',
encoding 'utf8',
foldername '/user/hive/warehouse/demo.db/product_info_orc/'
)
DISTRIBUTE BY ROUNDROBIN;
```

数据类型转换说明

当前用户导入到Hive/Spark的数据在HDFS存储为ORC文件格式，GaussDB(DWS)实际读取HDFS中的ORC文件，并对文件内的数据进行查询分析。

由于Hive/Spark支持的数据类型与GaussDB(DWS)自身支持的数据类型存在差异，在创建外表定义表字段时，您需要了解这两者之间数据类型的对应关系，具体如**表7-11**所示：

表 7-11 数据类型匹配表

类型名称	GaussDB(DWS)的HDFS/OBS外表支持的字段类型	Hive表字段类型	Spark表字段类型
2字节整数	SMALLINT	SMALLINT	SMALLINT
4字节整数	INTEGER	INT	INT
8字节整数	BIGINT	BIGINT	BIGINT
单精度浮点数	FLOAT4 (REAL)	FLOAT	FLOAT
双精度浮点型	FLOAT8(DOUBLE PRECISION)	DOUBLE	FLOAT
科学数据类型	DECIMAL[p (,s)] 最大支持38位精度	DECIMAL 最大支持38位（Hive 0.11）	DECIMAL
日期类型	DATE	DATE	DATE
时间类型	TIMESTAMP	TIMESTAMP	TIMESTAMP
Boolean类型	BOOLEAN	BOOLEAN	BOOLEAN
Char类型	CHAR(n)	CHAR (n)	STRING

类型名称	GaussDB(DWS)的HDFS/OBS外表支持的字段类型	Hive表字段类型	Spark表字段类型
VarChar类型	VARCHAR(n)	VARCHAR (n)	VARCHAR (n)
字符串	TEXT(CLOB)	STRING	STRING

7.7.5 执行数据导入

直接查询外表查看 MRS 数据源的数据

如果数据量较少，可直接使用SELECT查询外表，即可查看到MRS数据源的数据。

步骤1 执行以下命令，则可以从外表查询数据。

```
SELECT * FROM foreign_product_info;
```

查询结果显示如**数据文件**中所示的数据，表示导入成功。查询结果的结尾将显示以下信息：

```
(20 rows)
```

通过外表查询到数据后，用户可以将数据插入数据库的普通表。

----结束

导入数据后查询数据

也可以将MRS数据导入GaussDB(DWS)后，再查询数据。

步骤1 在GaussDB(DWS)数据库中，创建导入数据的目标表，用于存储导入的数据。

该表的表结构必须与**创建外表**中创建的外表的表结构保持一致，即字段个数、字段类型要完全一致。

例如，创建一个名为product_info的表，示例如下：

```
DROP TABLE IF EXISTS product_info;
CREATE TABLE product_info
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
with (
orientation = column,
compression=middle
)
DISTRIBUTE BY HASH (product_id);
```

步骤2 执行“INSERT INTO .. SELECT ..”命令从外表导入数据到目标表。

示例：

```
INSERT INTO product_info SELECT * FROM foreign_product_info;
```

若出现以下类似信息，说明数据导入成功。

```
INSERT 0 20
```

步骤3 执行SELECT命令，查看从MRS导入到GaussDB(DWS)中的数据。

```
SELECT * FROM product_info;
```

查询结果显示如**数据文件**中所示的数据，表示导入成功。查询结果的结尾将显示以下信息：

```
(20 rows)
```

----结束

7.7.6 清除资源

当完成本教程的示例后，如果您不再需要使用本示例中创建的资源，您可以删除这些资源，以免资源浪费或占用您的配额。

删除外表和目标表

步骤1 （可选）如果执行了**导入数据后查询数据**，请执行以下命令，删除目标表。

```
DROP TABLE product_info;
```

步骤2 执行以下命令，删除外表。

```
DROP FOREIGN TABLE foreign_product_info;
```

----结束

删除手动创建的外部服务器

如果执行了**手动创建外部服务器**，请按照以下步骤删除外部服务器、数据库和用户。

步骤1 使用创建外部服务器的用户通过GaussDB(DWS)提供的数据库客户端连接到外部服务器所在的数据库。

例如，使用gsql客户端的用户可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令进行切换：
  ```
  \c mydatabase dbuser;
  ```
  根据提示输入密码。
- 如果已经登录了gsql客户端，您也可以执行**\q**退出gsql后，再执行以下命令重新进行连接：
  ```
  gsql -d mydatabase -h 192.168.2.30 -U dbuser -p 8000 -r
  ```
  根据提示输入密码。

步骤2 删除手动创建的外部服务器。

执行以下命令进行删除，详细语法请参见DROP SERVER：

```
DROP SERVER hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca;
```

返回以下信息表示删除成功：

```
DROP SERVER
```

查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE
srvname='hdfs_server_8f79ada0_d998_4026_9020_80d6de2692ca';
```

返回结果如下所示，表示已经删除成功：

```
 srvname | srvowner | srvfdw | srvtype | srvversion | srvacl | srvoptions
---------+----------+--------+---------+------------+--------+------------
(0 rows)
```

步骤3 删除自定义数据库。

通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

如果已经登录了gsql客户端，可以直接执行如下命令进行切换：

```
\c postgres
```

根据界面提示输入密码。

执行以下命令，删除自定义数据库：

```
DROP DATABASE mydatabase;
```

返回以下信息表示删除成功：

```
DROP DATABASE
```

步骤4 使用管理员用户，删除本示例中创建的普通用户。

使用数据库管理员用户通过GaussDB(DWS)提供的数据库客户端连接数据库。

如果已经登录了gsql客户端，可以直接执行如下命令进行切换：

```
\c postgres dbadmin
```

执行以下命令回收创建外部服务器的权限：

```
REVOKE ALL ON FOREIGN DATA WRAPPER hdfs_fdw FROM dbuser;
```

其中FOREIGN DATA WRAPPER的名字只能是hdfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令删除用户：

```
DROP USER dbuser;
```

可使用**\du**命令查询用户，确认用户是否已经删除。

----结束

7.7.7 错误处理

如下错误信息，表示GaussDB(DWS)期望读取ORC数据文件，但实际却是*.txt类型的数据文件。请先创建Hive ORC类型的表，并将数据存储到该Hive ORC表中。

```
ERROR:  dn_6009_6010: Error occurs while creating an orc reader for file /user/hive/warehouse/
products_info.txt, detail can be found in dn log of dn_6009_6010.
```

7.8 使用 CDM 迁移数据到 Gauss(DWS)

使用云数据迁移服务（Cloud Data Migration，简称CDM），可以将其他数据源（例如MySQL）的数据迁移到GaussDB(DWS) 集群的数据库中。

使用CDM迁移数据到GaussDB(DWS) 的典型场景，请参见云数据迁移服务（简称CDM）的如下章节：

- 入门：该入门场景为使用CDM迁移本地MySQL数据库到GaussDB(DWS)

7.9 使用 DSC 工具迁移 SQL 脚本

DSC（Database Schema Convertor）是一款运行在Linux或Windows操作系统上的命令行工具，致力于向客户提供简单、快速、可靠的应用程序SQL脚本迁移服务，通过内置的语法迁移逻辑解析源数据库应用程序SQL脚本，并迁移为适用于GaussDB(DWS) 数据库的应用程序SQL脚本。DSC不需要连接数据库，可在离线模式下实现零停机迁移。在GaussDB(DWS) 中通过执行迁移后的SQL脚本即可恢复数据库，从而实现线下数据库轻松上云。

DSC支持迁移Teradata、Oracle、Netezza、MySQL和DB2数据库的SQL脚本。

下载 DSC SQL 语法迁移工具

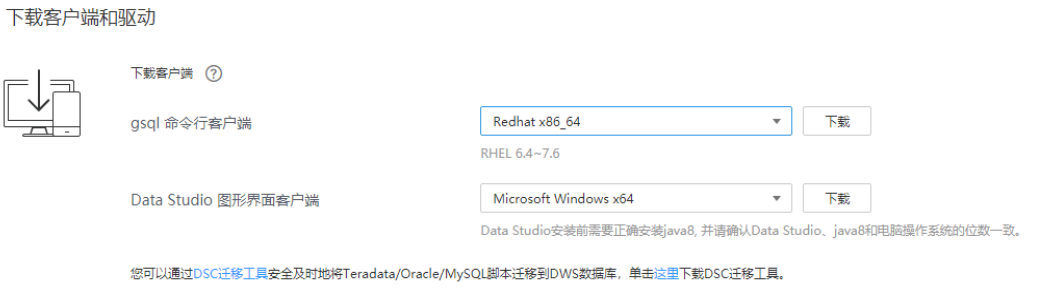
步骤1 登录GaussDB(DWS) 管理控制台。

步骤2 在左侧导航栏中，单击“连接管理”。

步骤3 在“下载客户端和驱动”区域，单击“这里”即可下载DSC迁移工具。

如果同时拥有不同版本的集群，系统会弹出对话框，提示您选择“集群版本”然后下载与集群版本相对应的客户端。在“集群管理”页面的集群列表中，单击指定集群的名称，再选择“基本信息”页签，可查看集群版本。

图 7-8 下载客户端



步骤4 下载到本机后，使用WinSCP工具，将DSC工具上传到一个需安装工具的Linux主机上。

执行上传操作的用户需要对Linux主机的目标存放目录有完全控制权限。

----结束

DSC SQL 语法迁移工具操作指导

详细指导请参见**DSC SQL语法迁移工具**。

7.10 查看数据倾斜状态

操作场景

数据倾斜会造成查询表性能下降。对于记录数超过千万条的表，建议在执行全量数据导入前，先导入部分数据，以进行数据倾斜检查和调整分布列，避免导入大量数据后发现数据倾斜，调整成本高。

背景信息

GaussDB(DWS)是采用Shared-nothing架构的MPP（Massive Parallel Processor，大规模并发处理）系统，采用水平分布的方式，将业务数据表的元组按合适的分布策略分散存储在所有的DN。

当前产品支持复制（Replication）和散列（Hash）两种用户表分布策略。

- Replication方式：在每一个DN上存储一份全量表数据。对于数据量比较小的表建议采取Replication分布策略。
- Hash方式：采用这种分布方式，需要为用户表指定一个分布列（distribute key）。当插入一条记录时，系统会根据分布列的值进行hash运算后，将数据存储在对应的DN中。对于数据量比较大的表建议采取Hash分布策略。

对于Hash分布策略，如果分布列选择不当，可能导致数据倾斜。因此在采用Hash分布策略之后会对用户表的数据进行数据倾斜性检查，以确保数据在各个DN上是均匀分布的。一般情况下分布列都是选择键值重复度小，数据分布比较均匀的列。

操作步骤

步骤1 分析数据源特征，选择若干个键值重复度小，数据分布比较均匀的备选分布列。

步骤2 从**步骤1**中选择一个备选分布列创建目标表。

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name
  ({ column_name data_type [ compress_mode ] [ COLLATE collation ] [ column_constraint [ ... ] ]
  | table_constraint   | LIKE source_table [ like_option [...] ] }
  [, ... ])   [ WITH ( {storage_parameter = value} [, ... ] ) ]
  [ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
  [ COMPRESS | NOCOMPRESS ]   [ TABLESPACE tablespace_name ]
  [ DISTRIBUTE BY { REPLICATION
                  | { HASH ( column_name [,...] ) } } ];
```

步骤3 参照前面章节中的办法向目标表中导入小批量数据。

对于单个数据源文件，在导入时，可通过均匀切割，导入部分切割后的数据源文件来验证数据倾斜性。

步骤4 检验数据倾斜性。命令中的table_name ，请填入实际的目标表名。

```
SELECT a.count,b.node_name FROM (SELECT count(*) AS count,xc_node_id FROM table_name GROUP BY xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER BY a.count desc;
```

步骤5 若各DN上数据分布差小于10%，表明数据分布均衡，选择的分布列合适。请清理已导入小批量数据，导入全量数据，以完成数据迁移。

若各DN上数据分布差大于等于10%，表明数据分布倾斜，请从**步骤1**的备选分布列中删除该列，删除目标表，并重复**步骤2** 、**步骤3** 、**步骤4** 和**步骤5**。

步骤6 （可选）如果上述步骤不能选出适合的分布列，需要从备选分布列选择多个列的组合作为分布列来完成数据迁移。

----结束

示例

对目标表staffs选择合适的分布列。

1. 分析表staffs的数据源特征，选择数据重复度低且分布均匀的备选分布列staff_ID、FIRST_NAME和LAST_NAME。
2. 先选择staff_ID作为分布列，创建目标表staffs。

```
CREATE TABLE staffs
(
  staff_ID       NUMBER(6) not null,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
  HIRE_DATE      DATE,
  employment_ID  VARCHAR2(10),
  SALARY         NUMBER(8,2),
  COMMISSION_PCT NUMBER(2,2),
  MANAGER_ID     NUMBER(6),
  section_ID     NUMBER(4)
)
DISTRIBUTE BY hash(staff_ID);
```

3. 向目标表staffs中导入部分数据。

   根据以下查询所得，集群环境中主DN数为8个，则建议导入的记录数为80000条。

```
SELECT count(*) FROM pgxc_node where node_type='D';
 count
-------
     8
(1 row)
```

4. 校验以staff_ID为分布列的目标表staffs的数据倾斜性。

```
SELECT a.count,b.node_name FROM (select count(*) as count,xc_node_id FROM staffs GROUP BY xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER BY a.count desc;
count | node_name
------+-----------
11010 | datanode4
10000 | datanode3
12001 | datanode2
 8995 | datanode1
10000 | datanode5
 7999 | datanode6
 9995 | datanode7
10000 | datanode8
(8 rows)
```

5. 根据上一步骤查询所得，各DN上数据分布差大于10%，数据分布倾斜。所以从步骤1的备选分布列中删除该列，并删除目标表staffs。

```
DROP TABLE staffs;
```

6. 尝试选择staff_ID、FIRST_NAME和LAST_NAME的组合作为分布列，创建目标表staffs。

```
CREATE TABLE staffs
(
  staff_ID       NUMBER(6) not null,
  FIRST_NAME     VARCHAR2(20),
  LAST_NAME      VARCHAR2(25),
  EMAIL          VARCHAR2(25),
  PHONE_NUMBER   VARCHAR2(20),
```

```
 HIRE_DATE       DATE,
 employment_ID  VARCHAR2(10),
 SALARY          NUMBER(8,2),
 COMMISSION_PCT NUMBER(2,2),
 MANAGER_ID      NUMBER(6),
 section_ID      NUMBER(4)
)
DISTRIBUTE BY hash(staff_ID,FIRST_NAME,LAST_NAME);
```

7. 校验以staff_ID、FIRST_NAME和LAST_NAME的组合为分布列的目标表staffs的数据倾斜性。

```
SELECT a.count,b.node_name FROM (select count(*) as count,xc_node_id FROM staffs GROUP BY
xc_node_id) a, pgxc_node b WHERE a.xc_node_id=b.node_id ORDER BY a.count desc;
count | node_name
------+-----------
10010 | datanode4
10000 | datanode3
10001 | datanode2
 9995 | datanode1
10000 | datanode5
 9999 | datanode6
 9995 | datanode7
10000 | datanode8
(8 rows)
```

8. 根据上一步骤查询所得，各DN上数据分布差小于10%，数据分布均衡，选择的分布列合适。
9. 清理已导入小批量数据。

```
TRUNCATE TABLE staffs;
```

10. 导入全量数据，以完成数据迁移。

7.11 分析表

执行计划生成器需要使用表的统计信息，以生成最有效的查询执行计划，提高查询性能。因此数据导入完成后，建议执行ANALYZE语句生成最新的表统计信息。统计结果存储在系统表PG_STATISTIC中。

分析表

ANALYZE支持的表类型有行/列存表、HDFS表、ORC/CARBONDATA格式的OBS外表。ANALYZE同时也支持对本地表的指定列进行信息统计。下面以表的ANALYZE为例，更多关于ANALYZE的信息，请参见ANALYZE | ANALYSE。

步骤1 更新表统计信息。

以表product_info为例，ANALYZE命令如下：

```
ANALYZE product_info;
ANALYZE
```

----结束

表自动分析

GaussDB(DWS)提供了GUC参数autovacuum用于控制数据库自动清理功能的启动。

autovacuum设置为on时，系统定时启动autovacuum线程来进行表自动分析，如果表中数据量发生较大变化达到阈值时，会触发表自动分析，即autoanalyze。

- 对于空表而言，当表中插入数据的行数大于50时，会触发表自动进行ANALYZE。

- 对于表中已有数据的情况，阈值设定为50+10%*reltuples，其中reltuples是表的总行数。

autovacuum自动清理功能的生效还依赖于下面两个GUC参数：

- **track_counts** 参数需要设置为on，表示开启收集收据库统计数据功能。
- autovacuum_max_workers参数需要大于0，该参数表示能同时运行的自动清理线程的最大数量。

须知

- autoanalyze只支持默认采样方式，不支持百分比采样方式。
- 多列统计信息仅支持百分比采样，因此autoanalyze不收集多列统计信息。
- autoanalyze支持行存表和列存表，不支持外表、HDFS表、OBS外表、临时表、unlogged表和toast表。

7.12 对表执行 VACUUM

如果导入过程中，进行了大量的更新或删除行时，应运行VACUUM FULL命令，然后运行ANALYZE命令。大量的更新和删除操作，会产生大量的磁盘页面碎片，从而逐渐降低查询的效率。VACUUM FULL可以将磁盘页面碎片恢复并交还操作系统。

步骤1 对表执行VACUUM FULL。

以表product_info为例，VACUUM FULL命令如下：

```
VACUUM FULL product_info
VACUUM
```

----结束

7.13 管理并发写入操作

7.13.1 事务隔离说明

GaussDB(DWS)基于MVCC（多版本并发控制）并结合两阶段锁的方式进行事务管理，其特点是读写之间不阻塞。SELECT是纯读操作，UPDATE和DELETE是读写操作。

- 读写操作和纯读操作之间并不会发生冲突，读写操作之间也不会发生冲突。每个并发事务在事务开始时创建事务快照，并发事务之间不能检测到对方的更改。
  - 读已提交隔离级别中，如果事务T1提交后，事务T2就可以看到事务T1更改的结果。
  - 可重复读级别中，如果事务T1提交事务前事务T2开始执行，则事务T1提交后，事务T2依旧看不到事务T1更改的结果，保证了一个事务开始后，查询的结果前后一致，不受其他事务的影响。
- 读写操作，支持的是行级锁，不同的事务可以并发更新同一个表，只有更新同一行时才需等待，后发生的事务会等待先发生的事务提交后，再执行更新操作。
  - READ COMMITTED：读已提交隔离级别，事务只能读到已提交的数据而不会读到未提交的数据，这是缺省值。

- REPEATABLE READ： 事务只能读到事务开始之前已提交的数据，不能读到未提交的数据以及事务执行期间其它并发事务提交的修改。

7.13.2 写入和读写操作

关于写入和读写操作的命令：

- INSERT，可向表中插入一行或多行数据。
- UPDATE，可修改表中现有数据。
- DELETE，可删除表中现有数据。
- COPY，导入数据。

INSERT和COPY是纯写入的操作。并发写入操作，需要等待，对同一个表的操作，当事务T1的INSERT或COPY未解除锁定时，事务T2的INSERT或COPY需等待，事务T1解除锁定时，事务T2正常继续。

UPDATE和DELETE是读写操作（先查询出要操作的行）。UPDATE和DELETE执行前需要先查询数据，由于并发事务彼此不可见，所以UPDATE和DELETE操作是读取事务发生前提交的数据的快照。写入操作，是行级锁，当事务T1和事务T2并发更新同一行时，后发生的事务T2会等待，根据设置的等待时长，若超时事务T1未提交则事务T2执行失败；当事务T1和事务T2并发更新的行不同时，事务T1和事务2都会执行成功。

7.13.3 并发写入事务的潜在死锁情况

只要事务涉及多个表的或者同一个表相同行的更新时，同时运行的事务就可能在同时尝试写入时变为死锁状态。事务会在提交或回滚时一次性解除其所有锁定，而不会逐一放弃锁定。 例如，假设事务T1和T2在大致相同的时间开始：

- 如果T1开始对表A进行写入且T2开始对表B进行写入，则两个事务均可继续而不会发生冲突；但是，如果T1完成了对表A的写入操作并需要开始对表B进行写入，此时操作的行数正好与T2一致，它将无法继续，因为T2仍保持对表B对应行的锁定，此时T2开始更新表A中与T1相同的行数，此时也将无法继续，产生死锁，在锁等待超时内，前面事务提交释放锁，后面的事务可以继续执行更新，等待时间超时后，事务抛错，有一个事务退出。
- 如果T1，T2都对表A进行写入，此时T1更新1-5行的数据，T2更新6-10行的数据，两个事务不会发生冲突，但是，如果T1完成后开始对表A的6-10行数据进行更新，T2完成后开始更新1-5行的数据，此时两个事务无法继续，在锁等待超时内，前面事务提交释放锁，后面的事务可以继续执行更新，等待时间超时后，事务抛错，有一个事务退出。

7.13.4 相同表的 INSERT 和 DELETE 并发

事务T1：

```
START TRANSACTION;
INSERT INTO test VALUES(1,'test1','test123');
COMMIT;
```

事务T2：

```
START TRANSACTION;
DELETE test WHERE NAME='test1';
COMMIT;
```

场景1：

开启事务T1，不提交的同时开启事务T2，事务T1执行INSERT完成后，执行事务T2的DELETE，此时显示DELETE 0，由于事务T1未提交，事务2看不到事务插入的数据；

场景2：

- READ COMMITTED级别

  开启事务T1，不提交的同时开启事务T2，事务T1执行INSERT完成后，提交事务T1，事务T2再执行DELETE语句时，此时显示DELETE 1，事务T1提交完成后，事务T2可以看到此条数据，可以删除成功。

- REPEATABLE READ级别

  开启事务T1，不提交的同时开启事务T2，事务T1执行INSERT完成后，提交事务T1，事务T2再执行DELETE语句时，此时显示DELETE 0，事务T1提交完成后，事务T2依旧看不到事务T1的数据，一个事务中前后查询到的数据是一致的。

7.13.5 相同表的并发 INSERT

事务T1：

```
START TRANSACTION;
INSERT INTO test VALUES(2,'test2','test123');
COMMIT;
```

事务T2：

```
START TRANSACTION;
INSERT INTO test VALUES(3,'test3','test123');
COMMIT;
```

场景1：

开启事务T1，不提交的同时开启事务T2，事务T1执行INSERT完成后，执行事务T2的INSERT语句，可以执行成功，读已提交和可重复读隔离级别下，此时在事务T1中执行SELECT语句，看不到事务T2中插入的数据，事务T2中执行查询语句看不到事务T1中插入的数据。

场景2：

- READ COMMITTED级别

  开启事务T1，不提交的同时开启事务T2，事务T1执行INSERT完成后直接提交，事务T2中执行INSERT语句后执行查询语句，可以看到事务T1中插入的数据。

- REPEATABLE READ级别

  开启事务T1，不提交的同时开启事务T2，事务T1执行INSERT完成后直接提交，事务T2中执行INSERT语句后执行查询语句，看不到事务T1中插入的数据。

7.13.6 相同表的并发 UPDATE

事务T1：

```
START TRANSACTION;
UPDATE test SET address='test1234' WHERE name='test1';
COMMIT;
```

事务T2：

```
START TRANSACTION;
UPDATE test SET address='test1234' WHERE name='test2';
COMMIT;
```

事务T3：

```
START TRANSACTION;
UPDATE test SET address='test1234' WHERE name='test1';
COMMIT;
```

场景1：

开启事务T1，不提交的同时开启事务T2，事务T1开始执行UPDATE，事务T2开始执行UPDATE，事务T1和事务T2都执行成功。更新不同行时，更新操作拿的是行级锁，不会发生冲突，两个事务都可以执行成功。

场景2：

开启事务T1，不提交的同时开启事务T3，事务T1开始执行UPDATE，事务T3开始执行UPDATE，事务T1执行成功，事务T3等待超时后会出错。更新相同行时，事务T1未提交时，未释放锁，导致事务T3执行不成功。

7.13.7 数据导入和查询的并发

事务T1：

```
START TRANSACTION;
COPY test FROM '...';
COMMIT;
```

事务T2：

```
START TRANSACTION;
SELECT * FROM test;
COMMIT;
```

场景1：

开启事务T1，不提交的同时开启事务T2，事务T1开始执行COPY，事务T2开始执行SELECT，事务T1和事务T2都执行成功。事务T2中查询看不到事务T1新COPY进来的数据。

场景2：

- READ COMMITTED级别

  开启事务T1，不提交的同时开启事务T2，事务T1开始执行COPY，然后提交，事务T2查询，可以看到事务T1中COPY的数据。

- REPEATABLE READ级别

  开启事务T1，不提交的同时开启事务T2，事务T1开始执行COPY，然后提交，事务T2 查询，看不到事务T1中COPY的数据。

8 导出数据

8.1 并行导出数据到 OBS

须知

- OBS导入导出数据时，不支持中文路径。
- OBS导入导出数据时，暂不支持跨Region进行OBS数据导入导出，必须确保OBS和DWS集群在同一个Region中。

8.1.1 关于 OBS 并行导出

概述

GaussDB(DWS)数据库支持通过OBS外表并行导出数据：通过OBS外表设置的导出模式、导出数据格式等信息来指定导出的数据文件，利用多DN并行的方式，将数据从GaussDB(DWS)数据库导出到外部，存放在OBS对象存储服务器上，从而提高整体导出性能。

- CN只负责任务的规划及下发，把数据导出的工作交给了DN，释放了CN的资源，使其有能力处理外部请求。
- 通过让各个DN都参与数据导出，充分利用各个设备的计算能力及网络带宽。
- 支持多个OBS服务并发导出，导出的桶和对象的路径必须不同并且为空。
- 选择OBS服务器与集群节点处于联网状态，导出速率会受网络带宽影响。
- 支持数据文件格式：TEXT、CSV。单行数据大小需<1GB。

相关概念

- **数据源文件**：存储有数据的TEXT、CSV文件。
- **OBS**：对象存储服务，是一种可存储文档、图片、影音视频等非结构化数据的云存储服务。从GaussDB(DWS)并行导出数据时，数据对象放置在OBS服务器上。
- **桶（Bucket）**：对OBS中的一个存储空间的形象称呼，是存储对象的容器。

  - 对象存储是一种非常扁平化的存储方式，桶中存储的对象都在同一个逻辑层级，去除了文件系统中的多层级树形目录结构。
  - 在OBS中，桶名必须是全局唯一的且不能修改，即用户创建的桶不能与自己已创建的其他桶名称相同，也不能与其他用户创建的桶名称相同。每个桶在创建时都会生成默认的桶ACL（Access Control List），桶ACL列表的每项包含了对被授权用户授予什么样的权限，如读取权限、写入权限、完全控制权限等。用户只有对桶有相应的权限，才可以对桶进行操作，如创建、删除、显示、设置桶ACL等。
  - 一个用户最多可创建100个桶，但每个桶中存放的总数据容量和对象/文件数量没有限制。
- **对象**：是存储在OBS中的基本数据单位。用户上传的数据以对象的形式存储在OBS的桶中。对象的属性包括名称Key，Metadata，Data。

  通常，我们将对象等同于文件来进行管理，但是由于OBS是一种对象存储服务，并没有文件系统中的文件和文件夹概念。为了使用户更方便进行管理数据，OBS提供了一种方式模拟文件夹。通过在对象的名称中增加“/”，如tpcds1000/stock.csv，tpcds1000可以等同于文件夹，stock.csv就可以等同于文件名，而对象名称（key）仍然是tpcds1000/stock.csv、对象的内容就是stock.csv数据文件的内容。
- **Key**：对象的名称（键），为经过UTF-8编码的长度大于0且不超过1024的字符序列，一个桶里的每个对象必须拥有唯一的对象键值。用户可使用桶名+对象名来存储和获取对应的对象。
- **Metadata**：对象元数据，用来描述对象的信息。元数据又可分为系统元数据和用户元数据。这些元数据以键值对（Key-value）的形式随http头域一起上传到OBS系统。
  - 系统元数据由OBS系统产生，在处理对象数据时使用。系统元数据包括：Date, Content-length, last-modify, Content-MD5等。
  - 用户元数据由用户上传对象时指定，是用户自己对对象的一些描述信息。
- **Data**：对象的数据内容，OBS对于数据的内容是无感知的，即认为对象内的数据为无状态的二进制数据。
- **外表**：用于识别数据源文件中的数据。外表中保存了数据源文件的位置、文件格式、存放位置、编码格式、数据间的分隔符等信息。

相关原理

下面分别从以下两类表介绍从集群导出数据到OBS的原理。

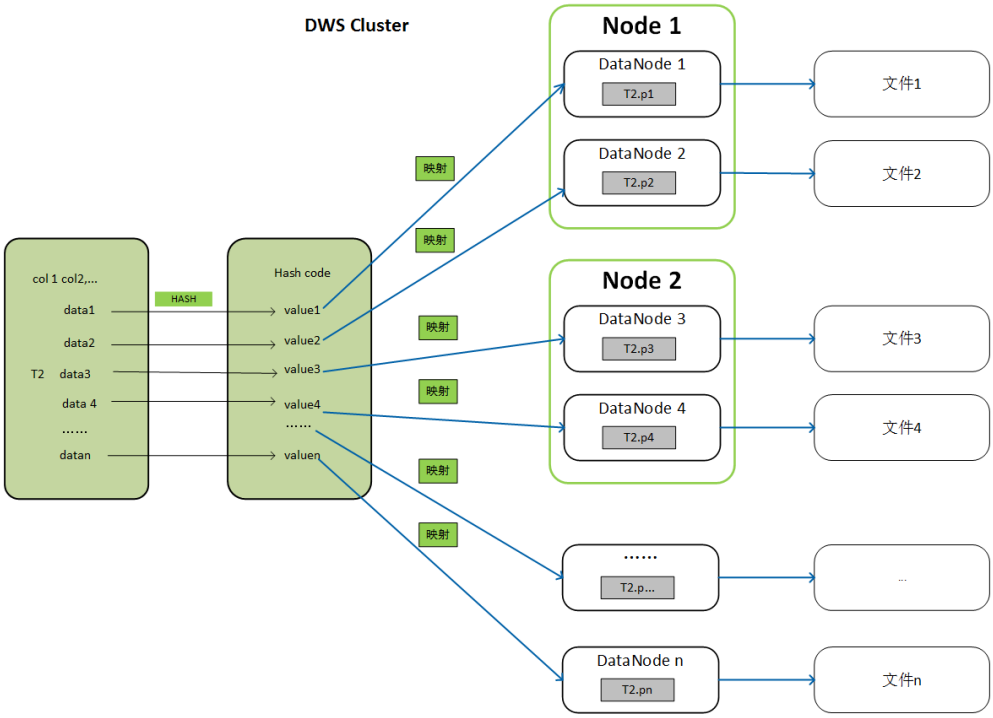
- Hash分布表：在建表语句中指定了DISTRIBUTE BY HASH (Column_Name)的表。

  对于Hash分布表而言，在存储表数据时，采用的是散列（Hash）方式的存储原理，如图8-1所示，图中以将表（T2）导出到OBS为例。

  在存储表数据时，将表（T2）中指定的Hash字段（col2）进行Hash运算后，生成相应的Hash值（value），根据DN与Hash值的映射关系，将该元组分发到相应的DN上进行存储。

  在导出数据到OBS时，每一个存储了导出表的（T2）数据的DN会直接向OBS导出属于自己的数据文件。多个节点将并行导出原始数据。

图 8-1 Hash 分布原理



- Replication表：在建表语句中指定了DISTRIBUTE BY REPLICATION的表。

  Replication表在GaussDB(DWS)集群的每个节点上都会存储一份完整的表数据。因此，在导出数据到OBS时，GaussDB(DWS)只会随机选择一个DN节点向OBS导出数据。

导出文件的命名规则

GaussDB(DWS)向OBS导出数据的文件命名规则如下：

- 从DN节点导出数据时，以segment的格式存储在OBS服务中，文件命名规则为“**表名称_节点名称_segment.n**”。这里的“n”是从0开始按照自然数0、1、2、3递增。

  例如，表t1在datanode3里面的数据导出成文件“t1_datanode3_segment.0”、“t1_datanode3_segment.1”等等，以此类推。

  对于来自不同集群或不同数据库的数据，建议用户可以将数据导出到不同的OBS桶或者同一个OBS桶的不同路径下。

- 每个segment可以存储的最大数据为1GB，并且不能切断元组。如果segment超过1GB，超过1GB的数据会作为第二个segment进行存储。

  例如：

  datanode3节点将表（t1）导出到OBS时，一个segment里面已经存储了100条元组，大小是1023MB，如果再插入一条5MB的元组，大小就变成1028MB了，此时会以1023MB生成一个“t1_datanode3_segment.0”保存到OBS服务中，新插入的第101条元组作为下一个“t1_datanode3_segment.1”保存到OBS服务中。

- 导出Hash分布表时，每个DataNode节点生成的segment数量和集群的DataNode节点数无关，而是取决于每个DataNode节点上存储的数据量。按照Hash方式存储在各个DataNode节点上的数据分布不一定均匀。

例如，一个有6个DataNode节点的集群，DataNode1到DataNode6分别有1.5GB、0.7GB、0.6GB、0.8GB、0.4GB、0.5GB的数据，则导出时会生成7个OBS segment文件，其中DataNode1会生成1GB和0.5GB两个segment文件。

导出流程

图 8-2 并行导出流程

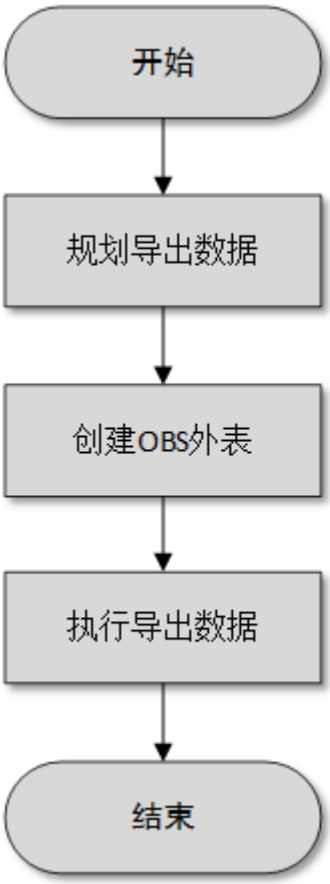


表 8-1 流程说明

流程	说明	子任务
规划导出数据	创建OBS桶，并在桶中创建导出后的数据文件的存放目录。 详细请参见**规划导出数据**。	-
创建OBS外表。	创建外表用于帮助OBS指定的待导出数据文件。外表中保存了数据源文件导出后的位置、文件格式、编码格式、数据间的分隔符等信息。 详细内容请参见**创建OBS外表**。	-

流程	说明	子任务
执行导出数据。	在创建好外表后，通过INSERT语句，将数据快速、高效地导出到数据文件中。 详细内容请参见**执行导出**。	-

8.1.2 规划导出数据

操作场景

在OBS上规划导出数据存放的位置。

规划 OBS 存储位置和文件

导出数据需要指定数据在OBS中的存储路径（需指定到目录），导出的数据可以按CSV解析格式保存到文件中。系统还支持TEXT类型的解析格式，将数据导出保存便于导入不同的应用程序。

导出路径的目标目录中不能存在任何文件。

规划 OBS 桶权限

在导出数据时，执行导出操作的用户需要具备以下条件：

- 已开通OBS服务。
- 具备数据导出路径所在的OBS桶的写入权限。
  通过配置桶的ACL权限，可以将写入权限授予指定的用户帐号。
  具体操作请参见**根据规划准备OBS存储位置和OBS桶的写权限**。

规划导出数据和外表

提前在数据库的表中准备好待导出的数据，且单行数据大小需要小于1GB。根据导出数据，规划匹配用户数据的外表，外表的字段、字段类型以及长度等属性需要能够对应用户数据。

根据规划准备 OBS 存储位置和 OBS 桶的写权限

步骤1 创建OBS桶，并在OBS桶中新建文件夹作为导出数据的存放目录。

1. 登录OBS管理控制台。
   单击“服务列表”，选择“对象存储服务”，打开OBS管理控制台页面。
2. 创建桶。
   如何创建OBS桶，具体请参见《对象存储服务控制台指南》中的**创建桶**章节。
   例如，创建桶：“mybucket”。
3. 新建文件夹。
   在OBS桶中，创建导出数据的存放目录。

例如，在已创建的OBS桶“mybucket”中新建一个文件夹“output_data”。

步骤2 获取新建文件夹的OBS路径。

在创建外表时需要指定导出数据文件的OBS存放目录的路径，用于创建外表时location参数设置。

location参数中OBS文件夹的路径由“obs://”、桶名和文件路径组成，即为：

obs://<bucket_name>/<file_path>/

例如，在本例中，location参数中OBS文件夹路径为：

```
obs://mybucket/output_data/
```

步骤3 为导出用户设置OBS桶的写权限。

在导出数据时，执行导出操作的用户需要具备数据导出路径所在的OBS桶的写入权限。通过配置桶的ACL权限，可以将写入权限授予指定的用户帐号。

----结束

8.1.3 创建 OBS 外表

操作步骤

步骤1 根据**规划导出数据**中规划的路径，由此确定创建外表时使用的参数**location**的值。

步骤2 用户获取OBS访问协议对应的AK值和SK值。获取访问密钥，请登录管理控制台，将鼠标移至右上角的用户名，单击“我的凭证”，然后在左侧导航树单击“访问密钥”。在访问密钥页面，可以查看已有的访问密钥ID（即AK），如果要同时获取AK和SK，可以单击“新增访问密钥”创建并下载访问密钥。

步骤3 梳理待导出数据的格式信息，确定创建外表时使用的数据格式参数的值。详细使用请参见数据格式参数。

步骤4 根据前面步骤确定的参数，**创建OBS外表**。外表的创建语法以及详细使用，请参考CREATE FOREIGN TABLE (OBS导入导出)。

----结束

示例

例如，在GaussDB(DWS)数据库中，创建一个外表。设置的参数信息如下所示：

- **location**

  在**规划导出数据**中，通过**步骤2. 获取数据源文件的OBS路径**，我们已经获取到数据源文件的OBS路径。

  因此，设置参数“location”为：

  ```
  location 'obs://mybucket/output_data/',
  ```

- **访问秘钥（AK和SK）**
  - 用户获取OBS访问协议对应的AK值（access_key）。
  - 用户获取OBS访问协议对应的SK值（secret_access_key）。

说明

用户在创建用户时已经获取了access_key和secret_access_key的密钥，请根据实际密钥替换示例中的斜体内容。

- **设置数据格式参数**
  - **数据源文件格式（format）**为CSV。
  - **编码格式（encoding）**为UTF-8。
  - **使用加密（encrypt）**为 'off'。
  - **字段分隔符（delimiter）**为','。
  - **header（指定导出数据文件是否包含标题行）**

    指定导出数据文件是否包含标题行，标题行一般用来描述表中每个字段的信息。

    OBS导出数据时不支持该参数为ture，使用缺省值false，不需要设置，表示导出的数据文件第一行不是标题行（即表头）。

根据以上信息，创建的外表如下所示：

```
DROP FOREIGN TABLE IF EXISTS product_info_output_ext;
CREATE FOREIGN TABLE product_info_output_ext
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
SERVER gsmpp_server
OPTIONS(
location 'obs://mybucket/output_data/',
FORMAT 'CSV' ,
DELIMITER ',',
encoding 'utf8',
header 'false',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
)
WRITE ONLY ;
```

返回如下信息表示创建成功：

```
CREATE FOREIGN TABLE
```

8.1.4 执行导出

操作步骤

步骤1 执行数据导出。

```
INSERT INTO  [foreign table 表名] SELECT * FROM [源表名];
```

说明

- 仅支持单个内表导出，不支持多表Join联合导出，不支持单表的聚集、排序、子查询、limit等操作结果导出。

----结束

执行导出数据示例

- **示例1**：将表product_info_output的数据通过外表product_info_output_ext导出到数据文件中。
  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output;
  ```
  若出现以下类似信息，说明数据导出成功。
  ```
  INSERT 0 10
  ```
- **示例2**：通过条件过滤（WHERE *product_price>500*），向数据文件中导出部分数据。
  ```
  INSERT INTO product_info_output_ext SELECT * FROM product_info_output WHERE product_price>500;
  ```

说明

对于特殊的数据类型如RAW类型，在导出之后是一个二进制文本，导入工具无法识别。需使用RAWTOHEX()函数将其转换为16进制文本导出。

8.1.5 示例

单表导出操作步骤

通过创建外表，将数据库中的单表导出至OBS的两个桶中。

步骤1 用户通过管理控制台登录到OBS数据服务器。在OBS数据服务器上，分别创建数据文件存放的两个桶“/input-data1”“/input-data2”，并创建每个桶下面的data目录“/input-data1/data”“/input-data2/data”。

步骤2 在GaussDB(DWS)数据库上，创建外表tpcds.customer_address_ext1和tpcds.customer_address_ext2用于OBS数据服务器接收数据库导出数据。

OBS与集群处于同一区域，需要导出的表为GaussDB(DWS)示例表tpcds.customer_address。

其中设置的**导出信息**如下所示：

- 由于OBS数据服务器上的数据源文件存放目录为“/input-data1/data/ ”和/input-data2/data/ ，所以设置tpcds.customer_address_ext1参数“location”为“obs://input-data1/data/ ”，设置tpcds.customer_address_ext2参数“location”为“ obs://input-data2/data/”。

设置的**数据格式信息**是根据表从数据库导出时需要的详细数据格式参数信息指定的，参数设置如下所示：

- 数据源文件格式（format）为CSV。
- 编码格式（encoding）为UTF-8。
- 字段分隔符（delimiter）为0E08。
- 引号字符（quote）为0x1b。
- 使用加密（encrypt）为 'off'。

- 用户获取OBS访问协议对应的AK值（access_key ）。（必选）
- 用户获取OBS访问协议对应的SK值（secret_access_key）。（必选）

说明

用户在创建用户时已经获取了access_key和secret_access_key的密钥，请根据实际密钥替换示例中的斜体内容。

根据以上信息，创建的外表如下所示：

```
CREATE FOREIGN TABLE tpcds.customer_address_ext1
(
ca_address_sk             integer                       ,
ca_address_id             char(16)                      ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                      ,
ca_suite_number           char(10)                      ,
ca_city                   varchar(60)                   ,
ca_county                 varchar(30)                   ,
ca_state                  char(2)                       ,
ca_zip                    char(10)                      ,
ca_country                varchar(20)                   ,
ca_gmt_offset             decimal(5,2)                  ,
ca_location_type          char(20)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input-data1/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
QUOTE E'\x1b',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
);
CREATE FOREIGN TABLE tpcds.customer_address_ext2
(
ca_address_sk             integer                       ,
ca_address_id             char(16)                      ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                      ,
ca_suite_number           char(10)                      ,
ca_city                   varchar(60)                   ,
ca_county                 varchar(30)                   ,
ca_state                  char(2)                       ,
ca_zip                    char(10)                      ,
ca_country                varchar(20)                   ,
ca_gmt_offset             decimal(5,2)                  ,
ca_location_type          char(20)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input-data2/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
QUOTE E'\x1b',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
);
```

步骤3 在GaussDB(DWS)数据库上，将数据表tpcds.customer_address并发导出到外表tpcds.customer_address_ext1和tpcds.customer_address_ext2中。

```
INSERT INTO tpcds.customer_address_ext1 SELECT * FROM tpcds.customer_address;
INSERT INTO tpcds.customer_address_ext2 SELECT * FROM tpcds.customer_address;
```

说明

OBS外表在设计上禁止往非空的路径下导出文件，但是在并发场景下会出现同一路径导出文件的情况，此时会发生异常。

异常场景：假如用户使用同一张表的数据并发导出到同一个OBS的外表，在一条SQL语句执行在OBS服务器上没有生成文件时，另一条SQL语句也执行导出，最终执行结果为两条SQL语句均执行成功，产生数据覆盖现象，建议用户在执行OBS外表导出任务时，不要往同一OBS外表并发导出。

----结束

多表并发导出操作步骤

通过创建的两个外表，将数据库中的两个表分别导出至OBS的桶中。

步骤1 用户通过管理控制台登录到OBS数据服务器。在OBS数据服务器上，分别创建数据文件存放的两个桶“/input-data1”“/input-data2”，并创建每个桶下面的data目录“/input-data1/data”“/input-data2/data”。

步骤2 在GaussDB(DWS)数据库上，创建外表tpcds.customer_address_ext1和tpcds.customer_address_ext2分别用于OBS服务器接收导出的数据。

规划OBS与集群处于同一区域，需要导出的表为已存在的表tpcds.customer_address1和tpcds.customer_address2。

其中设置的**导出信息**如下所示：

- 由于OBS服务器上的数据源文件存放目录为“/input-data1/data/ ”和/input-data2/data/ ，所以设置tpcds.customer_address_ext1参数“location”为“obs://input-data1/data/ ”，设置tpcds.customer_address_ext2参数“location”为“ obs://input-data2/data/”。

设置的**数据格式信息**是根据表从GaussDB(DWS)中导出时需要的详细数据格式参数信息指定的，参数设置如下所示：

- 数据源文件格式（ format ）为CSV。
- 编码格式（ encoding ）为UTF-8。
- 字段分隔符（ delimiter ）为0E08。
- 引号字符（ quote ）为0x1b。
- 使用加密（ encrypt ）为 'off'。
- 用户获取OBS访问协议对应的AK值（ access_key ）。（ 必选 ）
- 用户获取OBS访问协议对应的SK值（ secret_access_key ）。（ 必选 ）

说明

用户在创建用户是已经获取了access_key和secret_access_key的密钥，请根据实际密钥替换示例中的斜体内容。

根据以上信息，创建的外表如下所示：

```
CREATE FOREIGN TABLE tpcds.customer_address_ext1
(
ca_address_sk             integer               ,
ca_address_id             char(16)              ,
ca_street_number          char(10)                      ,
ca_street_name            varchar(60)                   ,
ca_street_type            char(15)                      ,
ca_suite_number           char(10)                      ,
```

```
ca_city                 varchar(60)               ,
ca_county               varchar(30)               ,
ca_state                char(2)                   ,
ca_zip                  char(10)                  ,
ca_country              varchar(20)               ,
ca_gmt_offset           decimal(5,2)              ,
ca_location_type        char(20)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input-data1/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
QUOTE E'\x1b',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
);
CREATE FOREIGN TABLE tpcds.customer_address_ext2
(
ca_address_sk           integer           ,
ca_address_id           char(16)          ,
ca_street_number        char(10)          ,
ca_street_name          varchar(60)       ,
ca_street_type          char(15)          ,
ca_suite_number         char(10)          ,
ca_city                 varchar(60)       ,
ca_county               varchar(30)       ,
ca_state                char(2)           ,
ca_zip                  char(10)          ,
ca_country              varchar(20)       ,
ca_gmt_offset           decimal(5,2)      ,
ca_location_type        char(20)
)
SERVER gsmpp_server
OPTIONS(LOCATION 'obs://input-data2/data/',
FORMAT 'CSV',
ENCODING 'utf8',
DELIMITER E'\x08',
QUOTE E'\x1b',
ENCRYPT 'off',
ACCESS_KEY 'access_key_value_to_be_replaced',
SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced'
);
```

步骤3 在GaussDB(DWS)数据库上，将数据表 tpcds.customer_address1和tpcds.customer_address2并发导出到外表tpcds.customer_address_ext1和tpcds.customer_address_ext2中。

```
INSERT INTO tpcds.customer_address_ext1 SELECT * FROM tpcds.customer_address1;
INSERT INTO tpcds.customer_address_ext2 SELECT * FROM tpcds.customer_address2;
```

----结束

8.2 使用 GDS 导出数据到远端服务器

8.2.1 关于 GDS 并行导出

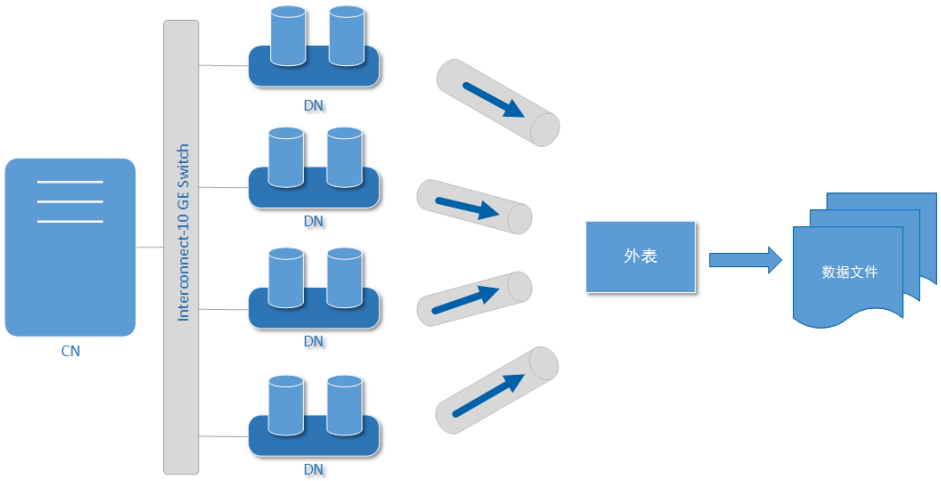
使用GDS工具将数据从数据库导出到普通文件系统中，适用于高并发、大量数据导出的场景。

概述

通过外表导出数据：通过GDS外表设置的导出模式、导出数据格式等信息来指定待导出的数据文件，利用多DN并行的方式，将数据从数据库导出到数据文件中，从而提高整体导出性能。不支持直接导出文件到HDFS文件系统。

- CN只负责任务的规划及下发，把数据导出的工作交给了DN，释放了CN的资源，使其有能力处理外部请求。
- 通过让各个DN都参与数据导出，充分利用各个设备的计算能力及网络带宽。

图 8-3 通过外表导出数据



相关概念

- **数据文件**：存储有数据的TEXT、CSV或FIXED文件。文件中保存的是从GaussDB(DWS)数据库导出的数据。
- **外表**：用于规划导出数据文件的数据文件格式、存放位置、编码格式等信息。
- **GDS**：数据服务工具。在导出数据时，需要将此工具部署到数据文件所在的服务器上，使DN可以通过该工具导出数据。
- **表**：数据库中的表，包括行存表和列存表。数据文件中的数据从这些表中导出。
- **Remote导出模式**：将集群中的业务数据导出到集群之外的主机上。

导出模式

GaussDB(DWS)支持的导出模式有Remote模式。

- **Remote模式：**将集群中的业务数据导出到集群之外的主机上。
  - 支持多个GDS服务并发导出，但1个GDS在同一时刻，只能为1个集群提供导出服务。
  - 配置与集群节点处于统一内网的GDS服务，导出速率受网络带宽影响，推荐的网络配置为10GE。
  - 支持数据文件格式：TEXT、CSV。单行数据大小需<1GB。

导出流程

图 8-4 GDS 并行导出流程

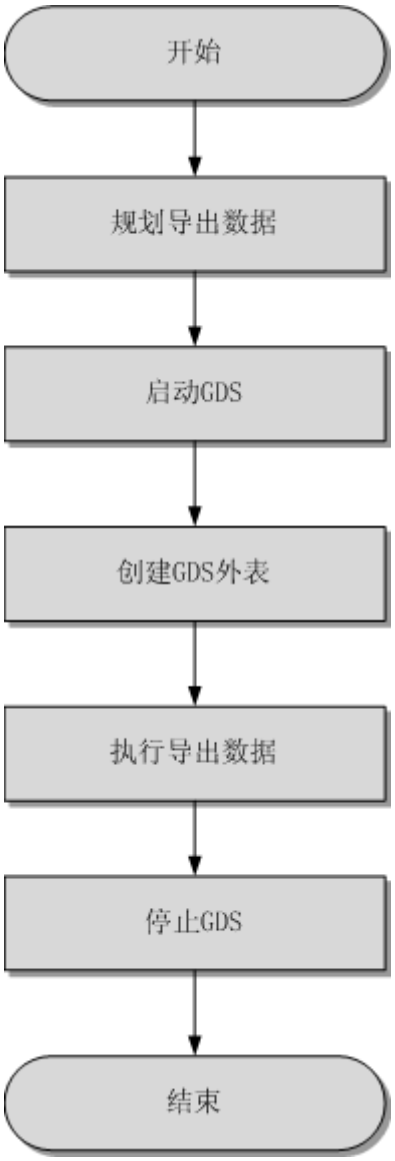


表 8-2 流程说明

流程	说明	子任务
规划导出数据。	根据所选模式，准备需要导出的数据并规划导出路径。 详细内容请参见**规划导出数据**	-
启动GDS。	若规划的导出模式为Remote模式，需在数据服务器上安装配置并启动GDS。 详细内容请参见**安装配置和启动GDS**。	-

流程	说明	子任务
创建外表。	创建外表用于帮助GDS指定导出的数据文件。外表中保存了导出数据文件的位置、文件格式、编码格式、数据间的分隔符等信息。 详细内容请参见**创建GDS外表**。	-
执行导出数据。	在创建好外表后，通过INSERT语句，将数据快速、高效地导出到数据文件中。 详细内容请参见**执行导出数据**。	-
停止GDS。	数据导出完成后，停止GDS。 详细请参见**停止GDS**。	-

8.2.2 规划导出数据

操作场景

使用GDS从集群导出到数据之前，要提前准备需要导出的数据，并规划导出的路径。

规划导出路径

- **Remote**模式

步骤1 以root用户登录GDS数据服务器，创建导出的数据文件存放目录“/output_data”。

```
mkdir -p /output_data
```

步骤2 （可选）创建用户及所属的用户组。此用户为启动GDS的用户，该用户需要拥有导出数据文件存放目录的写权限。

```
groupadd gdsgrp
useradd -g gdsgrp gdsuser
```

若出现以下提示，说明数据库用户及所属用户组已存在，可跳过本步骤。

```
useradd: Account 'gdsuser' already exists.
groupadd: Group 'gdsgrp' already exists.
```

步骤3 修改数据文件目录属主为gdsuser。

```
chown -R gdsuser:gdsgrp /output_data
```

----结束

8.2.3 安装配置和启动 GDS

GDS是GaussDB(DWS)提供的数据服务工具，通过和外表机制的配合，实现数据的高速导出。

详细内容请参见**安装配置和启动GDS**。

8.2.4 创建 GDS 外表

操作步骤

步骤1 根据**规划导出数据**中规划的路径确定外表参数**location**的值。

- **Remote**模式

  请通过URL方式设置参数“location”，用于指定导出的数据文件存放路径。

  - 不需要指定文件名。
  - 当有多个路径时，只有第一个路径有效。

  示例：

  GDS数据服务器IP为192.168.0.90，假定启动GDS时设置的监听端口为5000，设置的导出后文件存放目录为“/output_data/”。

  根据以上情况，在创建外表时，指定参数“location”为“gsfs://192.168.0.90:5000/”。

  说明

  location可以指定子目录如"gsfs://192.168.0.90:5000/2019/11/"实现同一张表根据日期导出到不同目录下。

  现有版本在执行**导出**任务的时候会判断"/output_data/2019/11" 目录是否存在，不存在则创建。导出时会将文件写入此目录下， 这样用户在创建或修改外表后就不需要再去手动执行“mkdir -p /output_data/2019/11”。

步骤2 梳理待导出数据的格式信息，确定创建外表时使用的数据格式参数的值。格式参数详细介绍，请参见CREATE FOREIGN TABLE (GDS导入导出)中的数据格式参数。

步骤3 根据前面步骤确定的参数，**创建GDS外表**。外表的创建语法以及详细使用，请参考CREATE FOREIGN TABLE (GDS导入导出)。

----结束

示例

- **示例：**创建GDS导出外表foreign_tpcds_reasons，待导出数据格式为CSV，用于接收数据服务器上的数据。

  其中设置的**导出模式**信息如下所示：

  规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，待导出的数据文件格式为CSV，选择并行导出模式为Remote模式。

  假定启动GDS时，规划导出的数据文件存放目录为“/output_data/”，GDS监听端口为5000，所以设置参数“location”为“gsfs://192.168.0.90:5000/”。

  设置导出的**数据格式信息**，参数设置如下所示：

  - 导出数据文件格式（format）为CSV。
  - 编码格式（encoding）为UTF-8。
  - 字段分隔符（delimiter）为E'\x08'。
  - 引号字符（quote）为0x1b。
  - 数据文件中空值（null）为没有引号的空字符串。
  - 逃逸字符（escape）为默认值双引号。
  - 数据文件是否包含标题行（header）为默认值false，即导出时数据文件第一行被识别为数据。

- 导出数据文件换行符样式（EOL）为0X0A。

创建的外表如下所示：

```
CREATE FOREIGN TABLE foreign_tpcds_reasons
(
  r_reason_sk    integer        not null,
  r_reason_id    char(16)       not null,
  r_reason_desc  char(100)
)
SERVER gsmpp_server
OPTIONS (LOCATION 'gsfs://192.168.0.90:5000/',
FORMAT 'CSV',
DELIMITER E'\x08',
QUOTE E'\x1b',
NULL '',
EOL '0x0a'
)
WRITE ONLY;
```

8.2.5 执行导出数据

前提条件

需要确保每一个CN和DN所在服务器到GDS服务器的IP和端口是互通的。

操作步骤

步骤1 执行数据导出。

```
INSERT INTO [foreign table 表名] SELECT * FROM [源表名];
```

说明

- 编写批处理任务脚本，实现并发批量导出数据。并发量视机器资源使用情况而定。可通过几个表测试，监控资源利用率，根据结果提高或减少并发量。常用资源监控命令有：内存和CPU监控top命令，IO监控命令iostat，网络监控命令sar等。相关案例请参见**示例：多线程导出**。
- 仅支持单个内表导出，不支持多表Join联合导出，不支持单表的聚集、排序、子查询、limit等操作结果导出。

----结束

任务示例

- **示例1**：将表reasons的数据通过外表foreign_tpcds_reasons导出到数据文件中。

```
INSERT INTO foreign_tpcds_reasons SELECT * FROM reasons;
```

- **示例2**：通过条件过滤（r_reason_sk =1），向数据文件中导出部分数据。

```
INSERT INTO foreign_tpcds_reasons SELECT * FROM reasons WHERE r_reason_sk=1;
```

- **示例3**：对于特殊的数据类型如RAW类型，在导出之后是一个二进制文本，导入工具无法识别。需使用RAWTOHEX()函数将其转换为16进制文本导出。

```
INSERT INTO foreign_blob_type_tab SELECT RAWTOHEX(c) FROM blob_type_tab;
```

8.2.6 停止 GDS

GDS是GaussDB(DWS)提供的数据服务工具，通过和外表机制的配合，实现数据的高速导出。

详细内容请参见**停止GDS**。

8.2.7 GDS 导出示例

示例：Remote 模式导出

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，导出数据文件格式为CSV，所以规划的并行导出模式为Remote模式。

Remote模式并行导出数据操作示例如下所示：

1. 以root用户登录GDS数据服务器，创建数据文件存放目录“/output_data”，启动gds_user用户及所属的用户组。
   ```
   mkdir -p /output_data
   ```
2. （可选）创建用户及其所属的用户组。此用户用于启动GDS。若该类用户及所属用户组已存在，可跳过此步骤。
   ```
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```
3. 修改数据服务器上数据文件目录“/output_data”的属主为gds_user。
   ```
   chown -R gds_user:gdsgrp /output_data
   ```
4. 以gds_user用户登录数据服务器上分别启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，导出数据文件存放在“/output_data/”目录下，数据服务器所在IP为192.168.0.90，GDS监听端口为5000，以后台方式运行。
   ```
   /opt/bin/dws/gds/bin/gds -d /output_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D
   ```
5. 在数据库中创建外表foreign_tpcds_reasons用于接收数据服务器上的数据。

   其中设置的**导出模式信息**如下所示：

   - 由于启动GDS时，设置的导出数据文件存放目录为“/output_data/”，GDS监听端口为5000。创建的导出数据文件存放目录为“/output_data/”。所以设置参数“location”为“gsfs://192.168.0.90:5000/”。

   设置导出的**数据文件格式**信息如下所示：

   - 数据文件格式（format）为CSV。
   - 编码格式（encoding）为UTF-8。
   - 字段分隔符（delimiter）为E'\x08'。
   - 引号字符（quote）为0x1b。
   - 数据文件中空值（null）为没有引号的空字符串。
   - 逃逸字符（escape）默认和quote相同。
   - 数据文件是否包含标题行（header）为默认值false，即导出时数据文件第一行被识别为数据。

   根据以上信息，创建的外表如下所示：
   ```
   CREATE FOREIGN TABLE foreign_tpcds_reasons
   (
     r_reason_sk    integer        not null,
     r_reason_id    char(16)       not null,
     r_reason_desc  char(100)
   ) SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://192.168.0.90:5000/', FORMAT 'CSV',ENCODING
   'utf8',DELIMITER E'\x08', QUOTE E'\x1b', NULL '') WRITE ONLY;
   ```
6. 在数据库上，通过外表foreign_tpcds_reasons，将数据导出到数据文件中。
   ```
   INSERT INTO foreign_tpcds_reasons SELECT * FROM reasons;
   ```
7. 待数据导出完成后，以gds_user用户登录数据服务器，停止GDS。

   其中GDS进程号为128954。
   ```
   ps -ef|grep gds
   gds_user 128954     1  0 15:03 ?        00:00:00 gds -d /output_data -p 192.168.0.90:5000 -D
   ```

```
gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
kill -9 128954
```

示例：多线程导出

规划数据服务器与集群处于同一内网，数据服务器IP为192.168.0.90，导出的数据文件格式为CSV，同时导出2个目标表，所以规划使用Remote模式进行多线程导出。

Remote模式多线程导出数据操作示例如下所示：

1. 以root用户登录GDS数据服务器，创建导出数据文件存放目录“/output_data”，数据库用户及所属的用户组。
   ```
   mkdir -p /output_data
   groupadd gdsgrp
   useradd -g gdsgrp gds_user
   ```
2. 修改数据服务器上数据文件目录“/output_data”的属主为gds_user。
   ```
   chown -R gds_user:gdsgrp /output_data
   ```
3. 以gds_user用户登录数据服务器上启动GDS。

   其中GDS安装路径为“/opt/bin/dws/gds”，导出数据文件存放在“/output_data/”目录下，数据服务器所在IP为192.168.0.90，GDS监听端口为5000，以后台方式运行，设定并发度为2。
   ```
   /opt/bin/dws/gds/bin/gds -d /output_data -p 192.168.0.90:5000 -H 10.10.0.1/24 -D -t 2
   ```
4. 在GaussDB(DWS)上，创建外表foreign_tpcds_reasons1和foreign_tpcds_reasons2用于接收数据服务器上的数据。
   - 其中设置的**导出模式信息**如下所示：
     - 由于启动GDS时，设置的导出数据文件存放目录为“/output_data/”，GDS监听端口为5000。创建的导出数据文件存放目录为“/output_data/”。所以设置参数“location”为“gsfs://192.168.0.90:5000/”。
   - 设置导出的数据**文件格式信息**如下所示：
     - 数据文件格式（format）为CSV。
     - 编码格式（encoding）为UTF-8。
     - 字段分隔符（delimiter）为E'\x08'。
     - 引号字符（quote）为0x1b。
     - 数据文件中空值（null）为没有引号的空字符串。
     - 逃逸字符（escape）为默认值双引号。
     - 数据文件是否包含标题行（header）为默认值false，即导出时数据文件第一行被识别为数据。

   根据以上信息，创建的外表foreign_tpcds_reasons1如下所示：
   ```
   CREATE FOREIGN TABLE foreign_tpcds_reasons1
   (
     r_reason_sk    integer     not null,
     r_reason_id    char(16)    not null,
     r_reason_desc  char(100)
   ) SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://192.168.0.90:5000/', FORMAT 'CSV',ENCODING
   'utf8', DELIMITER E'\x08', QUOTE E'\x1b', NULL '') WRITE ONLY;
   ```

   参考以上设置，创建的外表foreign_tpcds_reasons2如下所示：

```
CREATE FOREIGN TABLE foreign_tpcds_reasons2
(
  r_reason_sk    integer      not null,
  r_reason_id    char(16)     not null,
  r_reason_desc  char(100)
) SERVER gsmpp_server OPTIONS (LOCATION 'gsfs://192.168.0.90:5000/', FORMAT 'CSV', DELIMITER
E'\x08', QUOTE E'\x1b', NULL '') WRITE ONLY;
```

5. 在数据库中通过外表foreign_tpcds_reasons1和foreign_tpcds_reasons2，将表reasons1和reasons2中的数据导出到目录“/output_data”中。

```
INSERT INTO foreign_tpcds_reasons1 SELECT * FROM reasons1;
INSERT INTO foreign_tpcds_reasons2 SELECT * FROM reasons2;
```

6. 待数据导出完成后，以gds_user用户登录数据服务器，停止GDS。

   其中GDS进程号为128954。

```
ps -ef|grep gds
gds_user 128954      1  0 15:03 ?        00:00:00 gds -d /output_data -p 192.168.0.90:5000 -D -t 2
gds_user 129003 118723  0 15:04 pts/0    00:00:00 grep gds
kill -9 128954
```

8.3 使用 gs_dump 和 gs_dumpall 命令导出数据

8.3.1 概述

GaussDB(DWS)提供的gs_dump和gs_dumpall工具，能够帮助用户导出需要的数据库对象或其相关信息。通过导入工具将导出的数据信息导入至需要的数据库，可以完成数据库信息的迁移。gs_dump支持导出单个数据库或其内的对象，而gs_dumpall支持导出集群中所有数据库或各库的公共全局对象。详细的使用场景见表8-3。

表 8-3 适用场景

适用场景	支持的导出粒度	支持的导出格式	配套的导入方法
导出单个数据库	数据库级导出。 • 导出全量信息。 使用导出的全量信息可以创建一个与当前库相同的数据库，且库中数据也与当前库相同。 • 仅导出库中所有对象的定义，包含库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。 使用导出的对象定义，可以快速创建一个相同的数据库，但是库中并无原数据库的数据。 • 仅导出数据。	• 纯文本格式 • 自定义归档格式 • 目录归档格式 • tar归档格式	• 纯文本格式数据文件导入请参见使用gsql元命令导入数据。

适用场景	支持的导出粒度	支持的导出格式	配套的导入方法
	模式级导出。 ● 导出模式的全量信息。 ● 仅导出模式中数据。 ● 仅导出对象的定义，包含表定义、存储过程定义和索引定义等。		
	表级导出。 ● 导出表的全量信息。 ● 仅导出表中数据。 ● 仅导出表的定义。		
导出所有数据库	**数据库级导出**。 ● 导出全量信息。 使用导出的全量信息可以创建与当前集群相同的一个集群，拥有相同数据库和公共全局对象，且库中数据也与当前各库相同。 ● 仅导出各数据库中的对象定义，包含表空间、库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。使用导出的对象定义，可以快速创建与当前集群相同的一个集群，拥有相同的数据库和表空间，但是库中并无原数据库的数据。 ● 仅导出数据。	纯文本格式	数据文件导入请参见**使用gsql元命令导入数据**。
	各库公共全局对象导出 ● 仅导出表空间信息。 ● 仅导出角色信息。 ● 导出角色与表空间。		

gs_dump和gs_dumpall通过-U指定执行导出的用户帐户。如果当前使用的帐户不具备导出所要求的权限时，会无法导出数据。此时，可在导出命令中设置--role参数来指定具备权限的角色。在执行命令后，gs_dump和gs_dumpall会使用--role参数指定的角色，完成导出动作。可使用该功能的场景请参见**表8-3**，详细操作请参见**无权限角色导出数据**。

gs_dump和gs_dumpall通过对导出的数据文件加密，导入时对加密的数据文件进行解密，可以防止数据信息泄露，为数据库的安全提供保证。

gs_dump和gs_dumpall工具在进行数据导出时，其他用户可以访问集群数据库（读或写）。

gs_dump和gs_dumpall工具支持导出完整一致的数据。例如，T1时刻启动gs_dump导出A数据库，或者启动gs_dumpall导出整个集群数据库，那么导出数据结果将会是T1时刻A数据库或者该集群数据库的数据状态，T1时刻之后对A数据库或集群数据库的修改不会被导出。

gs_dump和gs_dumpall工具是通过“gsql命令行客户端”软件包解压缩获取。

注意事项

- 禁止修改导出的文件和内容，否则可能无法恢复成功。
- 为了保证数据一致性和完整性，导出工具会对需要转储的表设置共享锁。如果表在别的事务中设置了共享锁，gs_dump和gs_dumpall会等待锁释放后锁定表。如果无法在指定时间内锁定某个表，转储会失败。用户可以通过指定--lock-wait-timeout选项，自定义等待锁超时时间。
- 由于gs_dumpall读取所有数据库中的表，因此必须以数据库集群管理员身份进行连接，才能导出完整文件。在使用gsql执行脚本文件导入时，同样需要管理员权限，以便添加用户和组，以及创建数据库。

8.3.2 导出单个数据库

8.3.2.1 导出数据库

GaussDB(DWS)支持使用gs_dump工具导出某个数据库级的内容，包含数据库的数据和所有对象定义。可根据需要自定义导出如下信息：

- 导出数据库全量信息，包含数据和所有对象定义。

  使用导出的全量信息可以创建一个与当前库相同的数据库，且库中数据也与当前库相同。
- 仅导出所有对象定义，包括：库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。

  使用导出的对象定义，可以快速创建一个相同的数据库，但是库中并无原数据库的数据。
- 仅导出数据，不包含所有对象定义。

操作步骤

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 使用gs_dump导出postgres数据库。

```
gs_dump -W Bigdata@123 -U jack -f /home/dbadmin/backup/postgres_backup.tar -p 8000 postgres -F t
```

表 8-4 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名。	-U jack
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项； ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W Bigdata@123
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home/dbadmin/backup/*postgres*_backup.tar
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
dbname	需要导出的数据库名称	postgres
-F	选择导出文件格式。-F参数值如下： ● p：纯文本格式 ● c：自定义归档 ● d：目录归档格式 ● t：tar归档格式	-F t

其他参数说明请参见**gs_dump**。

----结束

示例

示例一：执行gs_dump，导出postgres数据库全量信息，并对导出文件进行压缩，导出文件格式为sql文本格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/postgres_backup.sql -p 8000 postgres -Z 8 -F p
gs_dump[port='8000'][postgres][2017-07-21 15:36:13]: dump database postgres successfully
gs_dump[port='8000'][postgres][2017-07-21 15:36:13]: total time: 3793  ms
```

示例二：执行gs_dump，仅导出postgres数据库中的数据，不包含数据库对象定义，导出文件格式为自定义归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/postgres_data_backup.dmp -p 8000 postgres -a -F c
gs_dump[port='8000'][postgres][2017-07-21 15:36:13]: dump database postgres successfully
gs_dump[port='8000'][postgres][2017-07-21 15:36:13]: total time: 3793  ms
```

示例三：执行gs_dump，仅导出postgres数据库所有对象的定义，导出文件格式为sql文本格式。

```
--导出前，表nation有数据
select n_nationkey,n_name,n_regionkey from nation limit 3;
```

```
 n_nationkey |          n_name           | n_regionkey
-------------+---------------------------+-------------
           0 | ALGERIA                   |           0
           3 | CANADA                    |           1
          11 | IRAQ                      |           4
(3 rows)

gs_dump -W Bigdata@123 -f /home/dbadmin/backup/postgres_def_backup.sql -p 8000 postgres -s -F p
gs_dump[port='8000'][postgres][2017-07-20 15:04:14]: dump database postgres successfully
gs_dump[port='8000'][postgres][2017-07-20 15:04:14]: total time: 472 ms
```

示例四：执行gs_dump，仅导出postgres数据库的所有对象的定义，导出文件格式为文本格式，并对导出文件进行加密。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/postgres_def_backup.sql -p 8000 postgres --with-encryption AES128 --with-key 1234567812345678 -s -F p
gs_dump[port='8000'][postgres][2018-11-14 11:25:18]: dump database postgres successfully
gs_dump[port='8000'][postgres][2018-11-14 11:25:18]: total time: 1161  ms
```

8.3.2.2 导出模式

GaussDB(DWS)目前支持使用gs_dump工具导出模式级的内容，包含模式的数据和定义。用户可通过灵活的自定义方式导出模式内容，不仅支持选定一个模式或多个模式的导出，还支持排除一个模式或者多个模式的导出。可根据需要自定义导出如下信息：

- 导出模式全量信息，包含数据和对象定义。
- 仅导出数据，即模式包含表中的数据，不包含对象定义。
- 仅导出模式对象定义，包括：表定义、存储过程定义和索引定义等。

操作步骤

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 使用gs_dump同时导出hr和public模式。

```
gs_dump -W Bigdata@123 -U jack -f /home/dbadmin/backup/MPPDB_schema_backup -p 8000 human_resource -n hr -F d
```

表 8-5 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名。	-U jack
-W	指定用户连接的密码。 • 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项。 • 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W Bigdata@123

参数	参数说明	举例
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home/dbadmin/backup/MPPDB_schema_backup
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
dbname	需要导出的数据库名称	human_resource
-n	只导出与模式名称匹配的模式，此选项包括模式本身和所有它包含的对象。 • 单个模式：-n *schemaname* • 多个模式：多次输入-n *schemaname*	• 单个模式：-n hr • 多个模式：-n hr -n public
-F	选择导出文件格式。-F参数值如下： • p：纯文本格式 • c：自定义归档 • d：目录归档格式 • t：tar归档格式	-F d

其他参数说明请参见**gs_dump**。

----结束

示例

示例一：执行gs_dump，导出hr模式全量信息，并对导出文件进行压缩，导出文件格式为文本格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_schema_backup.sql -p 8000 human_resource -n hr -Z 6 -F p
gs_dump[port='8000'][human_resource][2017-07-21 16:05:55]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:05:55]: total time: 2425  ms
```

示例二：执行gs_dump，仅导出hr模式的数据，导出文件格式为tar归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_schema_data_backup.tar -p 8000 human_resource -n hr -a -F t
gs_dump[port='8000'][human_resource][2018-11-14 15:07:16]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2018-11-14 15:07:16]: total time: 1865  ms
```

示例三：执行gs_dump，仅导出hr模式的定义，导出文件格式为目录归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_schema_def_backup -p 8000 human_resource -n hr -s -F d
gs_dump[port='8000'][human_resource][2018-11-14 15:11:34]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2018-11-14 15:11:34]: total time: 1652  ms
```

示例四：执行gs_dump，导出human_resource数据库时，排除hr模式，导出文件格式为自定义归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_schema_backup.dmp -p 8000
human_resource -N hr -F c
gs_dump[port='8000'][human_resource][2017-07-21 16:06:31]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:06:31]: total time: 2522  ms
```

示例五：执行gs_dump，同时导出hr和public模式，且仅导出模式定义，并对导出文件进行加密，导出文件格式为tar归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_schema_backup1.tar -p 8000
human_resource -n hr -n public -s --with-encryption AES128 --with-key 1234567812345678 -F t
gs_dump[port='8000'][human_resource][2017-07-21 16:07:16]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:07:16]: total time: 2132  ms
```

示例六：执行gs_dump，导出human_resource数据库时，排除hr和public模式，导出文件格式为自定义归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_schema_backup2.dmp -p 8000
human_resource -N hr -N public -F c
gs_dump[port='8000'][human_resource][2017-07-21 16:07:55]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:07:55]: total time: 2296  ms
```

示例七：执行gs_dump，导出public模式下所有表（视图、序列和外表）和hr模式中staffs表，包含数据和表定义，导出文件格式为自定义归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_backup3.dmp -p 8000 human_resource -t
public.* -t hr.staffs -F c
gs_dump[port='8000'][human_resource][2018-12-13 09:40:24]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2018-12-13 09:40:24]: total time: 896  ms
```

8.3.2.3 导出表

GaussDB(DWS)支持使用gs_dump工具导出表级的内容，包含表定义和表数据。视图、序列和外表属于特殊的表。用户可通过灵活的自定义方式导出表内容，不仅支持选定一个表或多个表的导出，还支持排除一个表或者多个表的导出。可根据需要自定义导出如下信息：

- 导出表全量信息，包含表数据和表定义。
- 仅导出数据，不包含表定义。
- 仅导出表定义。

操作步骤

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 使用gs_dump同时导出指定表hr.staffs和hr.employments。

```
gs_dump -W Bigdata@123 -U jack -f /home/dbadmin/backup/MPPDB_table_backup -p 8000
human_resource -t hr.staffs -F d
```

表 8-6 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名。	-U jack

参数	参数说明	举例
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项。 ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W Bigdata@123
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home/dbadmin/backup/MPPDB_table_backup
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
dbname	需要导出的数据库名称	human_resource
-t	指定导出的表（或视图、序列、外表），可以使用多个-t选项来选择多个表，也可以使用通配符指定多个表对象。当使用通配符指定多个表对象时，注意给pattern打引号，防止shell扩展通配符。 ● 单个表：-t *schema.table* ● 多个表：多次输入-t *schema.table*	● 单个表：-t hr.staffs ● 多个表：-t hr.staffs -t hr.employments
-F	选择导出文件格式。-F参数值如下： ● p：纯文本格式 ● c：自定义归档 ● d：目录归档格式 ● t：tar归档格式	-F d

其他参数说明请参见**gs_dump**。

----结束

示例

示例一：执行gs_dump，导出表hr.staffs的定义和数据，并对导出文件进行压缩，导出文件格式为文本格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_backup.sql -p 8000 human_resource -t hr.staffs -Z 6 -F p
gs_dump[port='8000'][human_resource][2017-07-21 17:05:10]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 17:05:10]: total time: 3116  ms
```

示例二：执行gs_dump，只导出表hr.staffs的数据，导出文件格式为tar归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_data_backup.tar -p 8000
human_resource -t hr.staffs -a -F t
gs_dump[port='8000'][human_resource][2017-07-21 17:04:26]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 17:04:26]: total time: 2570  ms
```

示例三：执行gs_dump，导出表hr.staffs的定义，导出文件格式为目录归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_def_backup -p 8000 human_resource -
t hr.staffs -s -F d
gs_dump[port='8000'][human_resource][2017-07-21 17:03:09]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 17:03:09]: total time: 2297  ms
```

示例四：执行gs_dump，不导出表hr.staffs，导出文件格式为自定义归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_backup4.dmp -p 8000 human_resource
-T hr.staffs -F c
gs_dump[port='8000'][human_resource][2017-07-21 17:14:11]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 17:14:11]: total time: 2450  ms
```

示例五：执行gs_dump，同时导出两个表hr.staffs和hr.employments，导出文件格式为文本格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_backup1.sql -p 8000 human_resource -
t hr.staffs -t hr.employments -F p
gs_dump[port='8000'][human_resource][2017-07-21 17:19:42]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 17:19:42]: total time: 2414  ms
```

示例六：执行gs_dump，导出时，排除两个表hr.staffs和hr.employments，导出文件格式为文本格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_backup2.sql -p 8000 human_resource -
T hr.staffs -T hr.employments -F p
gs_dump[port='8000'][human_resource][2017-07-21 17:21:02]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 17:21:02]: total time: 3165  ms
```

示例七：执行gs_dump，导出表hr.staffs的定义和数据，只导出表hr.employments的定义，导出文件格式为tar归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_backup3.tar -p 8000 human_resource -
t hr.staffs -t hr.employments --exclude-table-data hr.employments -F t
gs_dump[port='8000'][human_resource][2018-11-14 11:32:02]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2018-11-14 11:32:02]: total time: 1645  ms
```

示例八：执行gs_dump，导出表hr.staffs的定义和数据，并对导出文件进行加密，导出文件格式为文本格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_backup4.sql -p 8000 human_resource -
t hr.staffs --with-encryption AES128 --with-key 1212121212121212 -F p
gs_dump[port='8000'][human_resource][2018-11-14 11:35:30]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2018-11-14 11:35:30]: total time: 6708  ms
```

示例九：执行gs_dump，导出public模式下所有表（包括视图、序列和外表）和hr模式中staffs表，包含数据和表定义，导出文件格式为自定义归档格式。

```
gs_dump -W Bigdata@123 -f /home/dbadmin/backup/MPPDB_table_backup5.dmp -p 8000 human_resource
-t public.* -t hr.staffs -F c
gs_dump[port='8000'][human_resource][2018-12-13 09:40:24]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2018-12-13 09:40:24]: total time: 896  ms
```

示例十： 执行gs_dump，仅导出依赖于t1模式下的test1表对象的视图信息，导出文件格式为目录归档格式。

```
gs_dump -W Bigdata@123 -U jack -f /home/dbadmin/backup/MPPDB_view_backup6 -p 8000
human_resource -t t1.test1 --include-depend-objs --exclude-self -F d
gs_dump[port='8000'][jack][2018-11-14 17:21:18]: dump database human_resource successfully
gs_dump[port='8000'][jack][2018-11-14 17:21:23]: total time: 4239  ms
```

8.3.3 导出所有数据库

8.3.3.1 导出所有数据库

GaussDB(DWS)支持使用gs_dumpall工具导出所有数据库的全量信息，包含集群中每个数据库信息和公共的全局对象信息。可根据需要自定义导出如下信息：

- 导出所有数据库全量信息，包含集群中每个数据库信息和公共的全局对象信息（包含角色和表空间信息）。

  使用导出的全量信息可以创建与当前集群相同的一个集群，拥有相同数据库和公共全局对象，且库中数据也与当前各库相同。
- 仅导出数据，即导出每个数据库中的数据，且不包含所有对象定义和公共的全局对象信息。
- 仅导出所有对象定义，包括：表空间、库定义、函数定义、模式定义、表定义、索引定义和存储过程定义等。

  使用导出的对象定义，可以快速创建与当前集群相同的一个集群，拥有相同的数据库和表空间，但是库中并无原数据库的数据。

操作步骤

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 使用gs_dumpall一次导出所有数据库信息。

```
gs_dumpall -W Bigdata@123 -U dbadmin -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000
```

表 8-7 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名，需要是集群管理员用户。	-U dbadmin
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项； ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W Bigdata@123
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home/dbadmin/backup/MPPDB_backup.sql
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000

其他参数说明请参见**gs_dumpall**。

----结束

示例

示例一：执行gs_dumpall，导出所有数据库全量信息（dbadmin用户为管理员用户），导出文件为文本格式。执行命令后，会有很长的打印信息，最终出现total time即代表执行成功。示例中将不体现中间的打印信息。

```
gs_dumpall -W Bigdata@123 -U dbadmin -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000
gs_dumpall[port='8000'][2017-07-21 15:57:31]: dumpall operation successful
gs_dumpall[port='8000'][2017-07-21 15:57:31]: total time: 9627  ms
```

示例二：执行gs_dumpall，仅导出所有数据库定义（dbadmin用户为管理员用户），导出文件为文本格式。执行命令后，会有很长的打印信息，最终出现total time即代表执行成功。示例中将不体现中间的打印信息。

```
gs_dumpall -W Bigdata@123 -U dbadmin -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000 -s
gs_dumpall[port='8000'][2018-11-14 11:28:14]: dumpall operation successful
gs_dumpall[port='8000'][2018-11-14 11:28:14]: total time: 4147  ms
```

示例三：执行gs_dumpall，仅导出所有数据库中数据，并对导出文件进行加密，导出文件为文本格式。执行命令后，会有很长的打印信息，最终出现total time即代表执行成功。示例中将不体现中间的打印信息。

```
gs_dumpall -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000 -a --with-encryption AES128 --with-key
1234567812345678
gs_dumpall[port='8000'][2018-11-14 11:32:26]: dumpall operation successful
gs_dumpall[port='8000'][2018-11-14 11:23:26]: total time: 4147  ms
```

8.3.3.2 导出全局对象

GaussDB(DWS)支持使用gs_dumpall工具导出所有数据库公共的全局对象，包含数据库用户和组，表空间及属性（例如：适用于数据库整体的访问权限）信息。

操作步骤

步骤1 使用数据库管理员通过DWS提供的数据库客户端连接默认数据库postgres。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 使用gs_dumpall导出表空间对象信息。

```
gs_dumpall -W Bigdata@123 -U dbadmin -f /home/dbadmin/backup/MPPDB_tablespace.sql -p 8000 -t
```

表 8-8 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名，需要是集群管理员用户。	-U dbadmin

参数	参数说明	举例
-W	指定用户连接的密码。 • 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项； • 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W Bigdata@123
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home/dbadmin/backup/*MPPDB_tablespace*.sql
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
-t	或者--tablespaces-only，只转储表空间，不转储数据库或角色。	-

其他参数说明请参见**gs_dumpall**。

----结束

示例

示例一：执行gs_dumpall，导出所有数据库的公共全局表空间信息和用户信息（dbadmin用户为管理员用户），导出文件为文本格式。

```
gs_dumpall -W Bigdata@123 -U dbadmin -f /home/dbadmin/backup/MPPDB_globals.sql -p 8000 -g
gs_dumpall[port='8000'][2018-11-14 19:06:24]: dumpall operation successful
gs_dumpall[port='8000'][2018-11-14 19:06:24]: total time: 1150  ms
```

示例二： 执行gs_dumpall，导出所有数据库的公共全局表空间信息（dbadmin用户为管理员用户），并对导出文件进行加密，导出文件为文本格式。

```
gs_dumpall -W Bigdata@123 -U dbadmin -f /home/dbadmin/backup/MPPDB_tablespace.sql -p 8000 -t --with-encryption AES128 --with-key 1212121212121212
gs_dumpall[port='8000'][2018-11-14 19:00:58]: dumpall operation successful
gs_dumpall[port='8000'][2018-11-14 19:00:58]: total time: 186  ms
```

示例三：执行gs_dumpall，导出所有数据库的公共全局用户信息（dbadmin用户为管理员用户），导出文件为文本格式。

```
gs_dumpall -W Bigdata@123 -U dbadmin -f /home/dbadmin/backup/MPPDB_user.sql -p 8000 -r
gs_dumpall[port='8000'][2018-11-14 19:03:18]: dumpall operation successful
gs_dumpall[port='8000'][2018-11-14 19:03:18]: total time: 162  ms
```

8.3.4 无权限角色导出数据

gs_dump和gs_dumpall通过-U指定执行导出的用户帐户。如果当前使用的帐户不具备导出所要求的权限时，会无法导出数据。此时，可在导出命令中设置--role参数来指定具备权限的角色。在执行命令后，gs_dump和gs_dumpall会使用--role参数指定的角色，完成导出动作。

操作步骤

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 使用gs_dump导出human_resource数据库数据。

用户jack不具备导出数据库human_resource的权限，而角色role1具备该权限，要实现导出数据库human_resource，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为tar归档格式。

```
gs_dump -U jack -W Bigdata@234 -f /home/dbadmin/backup/MPPDB_backup.tar -p 8000 human_resource
--role role1 --rolepassword abc@1234 -F t
```

表 8-9 常用参数说明

参数	参数说明	举例
-U	连接数据库的用户名。	-U jack
-W	指定用户连接的密码。 ● 如果主机的认证策略是trust，则不会对数据库管理员进行密码验证，即无需输入-W选项。 ● 如果没有-W选项，并且不是数据库管理员，会提示用户输入密码。	-W Bigdata@123
-f	将导出文件发送至指定目录文件夹。如果这里省略，则使用标准输出。	-f /home/dbadmin/backup/MPPDB_backup.tar
-p	指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。	-p 8000
dbname	需要导出的数据库名称	human_resource
--role	指定导出使用的角色名。选择该选项，会使导出工具连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有导出工具要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。	-r role1
--rolepassword	指定具体角色用户的角色密码。	--rolepassword abc@1234

参数	参数说明	举例
-F	选择导出文件格式。-F参数值如下： • p：纯文本格式 • c：自定义归档 • d：目录归档格式 • t：tar归档格式	-F t

其他参数说明请参见**gs_dump**或**gs_dumpall**。

----结束

示例

示例一：执行gs_dump导出数据，用户jack不具备导出数据库human_resource的权限，而角色role1具备该权限，要实现导出数据库human_resource，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为tar归档格式。

```
human_resource=# CREATE USER jack IDENTIFIED BY "1234@abc";
CREATE USER

gs_dump -U jack -W 1234@abc -f /home/dbadmin/backup/MPPDB_backup11.tar -p 8000 human_resource --role role1 --rolepassword abc@1234 -F t
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: total time: 4239  ms
```

示例二：执行gs_dump导出数据，用户jack不具备导出模式public的权限，而角色role1具备该权限，要实现导出模式public，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为tar归档格式。

```
human_resource=# CREATE USER jack IDENTIFIED BY "1234@abc";
CREATE USER

gs_dump -U jack -W 1234@abc -f /home/dbadmin/backup/MPPDB_backup12.tar -p 8000 human_resource -n public --role role1 --rolepassword abc@1234 -F t
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: dump database human_resource successfully
gs_dump[port='8000'][human_resource][2017-07-21 16:21:10]: total time: 3278  ms
```

示例三：执行gs_dumpall导出数据，用户jack不具备导出所有数据库的权限，而角色role1（管理员）具备该权限，要实现导出所有数据库，可以在导出命令中设置--role角色为role1，使用role1的权限，完成导出目的。导出文件格式为文本归档格式。

```
human_resource=# CREATE USER jack IDENTIFIED BY "1234@abc";
CREATE USER

gs_dumpall -U jack -W 1234@abc -f /home/dbadmin/backup/MPPDB_backup.sql -p 8000 --role role1 --rolepassword abc@1234
gs_dumpall[port='8000'][human_resource][2018-11-14 17:26:18]: dumpall operation successful
gs_dumpall[port='8000'][human_resource][2018-11-14 17:26:18]: total time: 6437  ms
```

9 查询外部数据

9.1 查询 OBS 上的数据

9.1.1 查询 OBS 上的数据概述

在数据仓库服务中，提供了SQL on OBS特性，允许用户通过外表直接使用SELECT查询存储在OBS上的TEXT、CSV、ORC、CARBONDATA格式的数据，而不需要先将OBS数据导入到GaussDB(DWS)集群中再进行查询，从而大大节省了查询时间和集群资源。本章以查询存储在OBS上的ORC格式的数据为例。

说明

通过外表查询OBS上的数据，不支持由DLI生成的CARBONDATA格式的事务表，如果是非事务表则支持通过OBS外表查询。

9.1.2 OBS 上的数据准备

操作场景

使用SQL on OBS功能查询OBS数据之前：

1. 假设您已将ORC数据存储在OBS上。

   例如，在使用Hive或Spark等组件时创建了ORC表，其表数据已经存储在OBS上的场景。

   假设有2个ORC数据文件“product_info.0”和“product_info.1”，其原始数据如**原始数据**所示，都已经存储在OBS桶“mybucket”的“demo.db/product_info_orc/”目录中。
2. 如果您的数据文件已经在OBS上了，请执行**获取源数据的OBS路径并设置读取权限**中的步骤。

原始数据

假设您已将2个ORC数据文件存储在OBS上，其原始数据分别如下：

- 数据文件“product_info.0”

示例数据如下所示：

```
100,XHDK-A-1293-#fJ3,2017-09-01,A,2017 Autumn New Shirt Women,red,M,
328,2017-09-04,715,good!
205,KDKE-B-9947-#kL5,2017-09-01,A,2017 Autumn New Knitwear Women,pink,L,
584,2017-09-05,406,very good!
300,JODL-X-1937-#pV7,2017-09-01,A,2017 autumn new T-shirt men,red,XL,1245,2017-09-03,502,Bad.
310,QQPX-R-3956-#aD8,2017-09-02,B,2017 autumn new jacket women,red,L,411,2017-09-05,436,It's
really super nice.
150,ABEF-C-1820-#mC6,2017-09-03,B,2017 Autumn New Jeans Women,blue,M,
1223,2017-09-06,1200,The seller's packaging is exquisite.
```

- 数据文件“product_info.1”

  示例数据如下所示：

```
200,BCQP-E-2365-#qE4,2017-09-04,B,2017 autumn new casual pants men,black,L,
997,2017-09-10,301,The clothes are of good quality.
250,EABE-D-1476-#oB1,2017-09-10,A,2017 autumn new dress women,black,S,
841,2017-09-15,299,Follow the store for a long time.
108,CDXK-F-1527-#pL2,2017-09-11,A,2017 autumn new dress women,red,M,85,2017-09-14,22,It's
really amazing to buy.
450,MMCE-H-4728-#nP9,2017-09-11,A,2017 autumn new jacket women,white,M,
114,2017-09-14,22,Open the package and the clothes have no odor.
260,OCDA-G-2817-#bD3,2017-09-12,B,2017 autumn new woolen coat women,red,L,
2004,2017-09-15,826,Very favorite clothes.
```

获取源数据的 OBS 路径并设置读取权限

步骤1 登录OBS管理控制台。

单击“服务列表”，选择“对象存储服务”，打开OBS管理控制台页面。

步骤2 获取数据源文件的OBS路径。

数据源文件在上传到OBS桶之后，会生成全局唯一的访问路径。在创建外表时需要指定数据源文件的OBS路径。

例如，在本例中，查看到数据文件的OBS路径分别为：

```
https://obs.cn-north-1.myhuaweicloud.com/mybucket/demo.db/product_info_orc/product_info.0
https://obs.cn-north-1.myhuaweicloud.com/mybucket/demo.db/product_info_orc/product_info.1
```

步骤3 为用户设置OBS桶的读取权限。

在使用SQL on OBS功能时，执行该功能的用户需要取得数据源文件所在OBS桶的读取权限。通过配置桶的ACL权限，可以将读取权限授予指定的用户帐号。

----结束

9.1.3 创建外部服务器

创建外部服务器，用于定义OBS服务器的信息，供外表调用。创建外部服务器的详细语法，请参见CREATE SERVER。

（可选）新建用户及数据库并授予外表权限

如果您将使用普通用户在自定义数据库中创建外部服务器和外表，由于普通用户没有外表权限无法创建，所以，您必须参照以下步骤新建用户和数据库，并授予该用户外表权限。

以下示例，是新建一个普通用户dbuser并创建一个数据库mydatabase，然后使用管理员用户授予dbuser外表权限。

步骤1 使用数据库管理员通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

例如，使用gsql客户端的用户执行下面命令连接数据库：

```
gsql -d postgres -h 192.168.2.30 -U dbadmin -p 8000 -r
```

根据界面提示输入密码。

步骤2 新建一个普通用户，并用它创建一个数据库。

新建一个具有创建数据库权限的用户dbuser：

```
CREATE USER dbuser WITH CREATEDB PASSWORD "Bigdata@123";
```

切换为新建的用户：

```
SET ROLE dbuser PASSWORD "Bigdata@123";
```

执行以下命令创建数据库：

```
CREATE DATABASE mydatabase;
```

查询数据库：

```
SELECT * FROM pg_database;
```

返回结果中有**mydatabase** 的信息表示创建成功：

```
datname   | datdba | encoding | datcollate | datctype | datistemplate | datallowconn | datconnlimit |
datlastsysoid | datfrozenxid | dattablespace | datcompatibility |                    datacl

------------+--------+----------+------------+----------+---------------+--------------+--------------+---------------
+--------------+---------------+------------------+-------------------------------------
--------------
 template1  |     10 |        0 | C          | C        | t             | t            |           -1 |        14146 |        1351
|         1663 | ORA             | {=c/Ruby,omm=CTc/Ruby}
 template0  |     10 |        0 | C          | C        | t             | f            |           -1 |        14146 |        1350
|         1663 | ORA             | {=c/Ruby,omm=CTc/Ruby}
 postgres   |     10 |        0 | C          | C        | f             | t            |           -1 |        14146 |        1352 |
1663 | ORA             | {=Tc/Ruby,Ruby=CTc/Ruby,chaojun=C/Ruby,hu
obinru=C/Ruby}
 mydatabase |  17000 |        0 | C          | C        | f             | t            |           -1 |        14146 |        1351
|         1663 | ORA             |
(4 rows)
```

步骤3 使用管理员用户给普通用户赋予创建外部服务器的权限和使用外表的权限。

使用数据库管理员用户通过GaussDB(DWS)提供的数据库客户端连接新建的数据库。

例如，使用gsql客户端的用户可以直接使用如下语句切换为管理员用户去连接新建的数据库：

```
\c mydatabase dbadmin;
```

根据提示输入管理员用户密码。

说明

注意，必须先使用管理员用户连接到**将要创建外部服务器和使用外表的数据库**，再对普通用户进行授权。

默认只有系统管理员才可以创建外部服务器，普通用户需要授权才可以创建，执行以下命令授权：

```
GRANT ALL ON FOREIGN DATA WRAPPER dfs_fdw TO dbuser;
```

其中fdw_name的名字可以是hdfs_fdw或者dfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令赋予用户使用外表的权限。

```
ALTER USER dbuser USEFT;
```

查看用户：

```
SELECT r.rolname, r.rolsuper, r.rolinherit,
  r.rolcreaterole, r.rolcreatedb, r.rolcanlogin,
  r.rolconnlimit, r.rolvalidbegin, r.rolvaliduntil,
  ARRAY(SELECT b.rolname
        FROM pg_catalog.pg_auth_members m
        JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid)
        WHERE m.member = r.oid) as memberof
, r.rolreplication
, r.rolauditadmin
, r.rolsystemadmin
, r.roluseft
FROM pg_catalog.pg_roles r
ORDER BY 1;
```

返回结果中dbuser的信息中包含了UseFT权限，表示授权成功：

```
rolname  | rolsuper | rolinherit | rolcreaterole | rolcreatedb | rolcanlogin | rolconnlimit | rolvalidbegin |
rolvaliduntil | memberof | rolreplication | rolauditadmin | rolsystemadmin | roluseft
-----------+----------+------------+---------------+-------------+-------------+--------------+---------------
+---------------+----------+----------------+---------------+----------------+----------
 dbuser   | f      | t        | f          | t        | t        |        -1 |           |           | {}      | f
| f          | f          | t
 lily     | f      | t        | f          | f        | t        |        -1 |           |           | {}      | f          |
f          | f          | f
 Ruby     | t      | t        | t          | t        | t        |        -1 |           |           | {}      | t
| t          | t          | t
```

----结束

创建外部服务器

步骤1 使用即将创建外部服务器的用户去连接其对应的数据库。

在本示例中，将使用**（可选）新建用户及数据库并授予外表权限**中创建的普通用户dbuser连接其创建的数据库mydatabase 。用户需要通过GaussDB(DWS)提供的数据库客户端连接数据库。

例如，使用gsql客户端的用户可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令切换数据库和用户：
  ```
  \c mydatabase dbuser;
  ```
  根据提示输入密码。
- 如果尚未登录gsql客户端，或者已经登录了gsql客户端执行**\q**退出gsql后，执行以下命令重新进行连接：
  ```
  gsql -d mydatabase -h 192.168.2.30 -U dbuser -p 8000 -r
  ```
  根据提示输入密码。

步骤2 创建外部服务器。

创建外部服务器的详细语法，请参见CREATE SERVER。

例如，执行以下命令创建外部服务器'obs_server' ：

```
CREATE SERVER obs_server FOREIGN DATA WRAPPER dfs_fdw
OPTIONS (
  address 'obs.cn-north-1.myhuaweicloud.com' ,
  ACCESS_KEY 'access_key_value_to_be_replaced',
  SECRET_ACCESS_KEY 'secret_access_key_value_to_be_replaced',
```

```
type 'obs'
);
```

以下为必选参数的说明：

- **外部服务器名称**

  允许用户自定义名字。

  在本例中，我们指定为“obs_server”。

- **FOREIGN DATA WRAPPER**

  fdw_name的名字可以是hdfs_fdw或者dfs_fdw，它在数据库中已经存在。

- **OPTIONS参数**

  – **address**

    指定OBS服务的IP地址或域名。

    address的获取方法如下：

    i. 先通过**OBS上的数据准备**中的**2**获取OBS路径。

    ii. 在OBS上查看到的OBS路径，有以下两种形式，其中就包含了OBS服务的IP地址或域名：

       ○ https://*存储服务器IP地址或域名/桶名/文件夹目录层级/对象名*

       ○ https://*桶名.域名/文件夹目录层级/对象名*

  – **访问密钥（AK和SK）**（必选）

    GaussDB(DWS)需要通过访问密钥（AK和SK）访问OBS，因此，必须先获取访问密钥。

    ■ “access_key”（必选）：表示用户的AK信息。

    ■ “secret_access_key”（必选）：表示用户的SK信息。

    获取访问密钥，请登录管理控制台，将鼠标移至右上角的用户名，单击“我的凭证”，然后在左侧导航树单击“访问密钥”。在访问密钥页面，可以查看已有的访问密钥ID（即AK），如果要同时获取AK和SK，可以单击“新增访问密钥”创建并下载访问密钥。

  – **type**

    取值为'obs'，表示dfs_fdw连接的是OBS。

步骤3 查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE srvname='obs_server';
```

返回结果如下所示，表示已经创建成功：

```
 srvname   | srvowner | srvfdw | srvtype | srvversion | srvacl
|                                                srvoptions

------------+----------+--------+---------+------------+--------
+---------------------------------------------------------------------------------------------------------------------------
------------------------------------------------
-----------
 obs_server |    24661 |  13686 |         |            |        |
{address=xxx.xxx.x.xxx,access_key=xxxxxxxxxxxxxxxxxxxx,type=obs,secret_access_key=xxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxx}
(1 row)
```

----结束

9.1.4 创建外表

当完成**创建外部服务器**后，在GaussDB(DWS)数据库中创建一个OBS外表，用来访问存储在OBS上的数据。OBS外表是只读的，只能用于查询操作，可直接使用SELECT查询其数据。

创建外表

创建外表的语法格式如下，详细的描述请参见CREATE FOREIGN TABLE (SQL on Hadoop or OBS)。

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name
( [ { column_name type_name
   [ { [CONSTRAINT constraint_name] NULL |
   [CONSTRAINT constraint_name] NOT NULL |
     column_constraint [...]} ] |
     table_constraint [, ...]} [, ...] ] )
   SERVER dfs_server
   OPTIONS ( { option_name ' value ' } [, ...] )
   DISTRIBUTE BY {ROUNDROBIN | REPLICATION}
   [ PARTITION BY ( column_name ) [ AUTOMAPPED ] ] ;
```

例如，创建一个名为"*product_info_ext_obs*"的外表，对语法中的参数按如下描述进行设置：

- **table_name**

  外表的表名。

- **表字段定义**
  - **column_name**：外表中的字段名。
  - **type_name**：字段的数据类型。

  多个字段用“,”隔开。

  外表的字段个数和字段类型，需要与OBS上保存的数据完全一致。

- **SERVER dfs_server**

  外表的外部服务器名称，这个server必须存在。外表通过设置外部服务器连接OBS读取数据。

  此处应填写为参照**创建外部服务器**创建的外部服务器名称。

- **OPTIONS 参数**

  用于指定外表数据的各类参数，关键参数如下所示。
  - “format”：表示对应的OBS服务上的文件格式，支持“orc”、“carbondata”格式。
  - “foldername”：必选参数。数据源文件的OBS路径，此处仅需要填写“/*桶名/文件夹目录层级/*”，不包括OBS服务的IP地址或域名。

    可以先通过**OBS上的数据准备**中的**2**获取数据源文件的完整的OBS路径，该路径可能是以下两种格式中一种，其中就包含了桶名及文件夹目录层级：
    - https://*存储服务器IP地址或域名/桶名/文件夹目录层级/对象名*
    - https://*桶名.域名/文件夹目录层级/对象名*
  - “totalrows”：可选参数。该参数不是导入的总行数。由于OBS上文件可能很多，做analyze可能会很慢，通过“totalrows”参数，让用户来设置一个预估的值，使优化器能通过这个值做大小表的估计。一般预估值与实际值的数量级差不多时，查询效率较高。

  - “encoding”：外表中数据源文件的编码格式名称，缺省为utf8。对于OBS外表此参数为必选项。

- **DISTRIBUTE BY**：

  这个子句是必须的，对于OBS外表，当前只支持**ROUNDROBIN**分布方式。

  表示外表在从数据源读取数据时，GaussDB(DWS)集群每一个节点随机读取一部分数据，并组成完整数据。

- **语法中的其他参数**

  其他参数均为可选参数，用户可以根据自己的需求进行设置，在本例中我们不需要设置。详细的描述请参见CREATE FOREIGN TABLE (SQL on Hadoop or OBS)。

根据以上信息，创建外表命令如下所示：

```
DROP FOREIGN TABLE IF EXISTS product_info_ext_obs;

---建立不包含分区列的OBS外表，表关联的外部服务器为obs_server，表对应的OBS服务上的文件格式为‘orc’，OBS上的数据存储路径为'/mybucket/data/'。

CREATE FOREIGN TABLE product_info_ext_obs
(
    product_price                integer        not null,
    product_id                   char(30)       not null,
    product_time                 date           ,
    product_level                char(10)       ,
    product_name                 varchar(200)   ,
    product_type1                varchar(20)    ,
    product_type2                char(10)       ,
    product_monthly_sales_cnt    integer        ,
    product_comment_time         date           ,
    product_comment_num          integer        ,
    product_comment_content      varchar(200)
) SERVER obs_server
OPTIONS (
format 'orc',
foldername '/mybucket/demo.db/product_info_orc/',
encoding 'utf8',
totalrows '10'
)
DISTRIBUTE BY ROUNDROBIN;
```

9.1.5 通过外表查询 OBS 上的数据

直接查询外表查看 OBS 上的数据

如果数据量较少，可直接使用SELECT查询外表，即可查看到OBS上的数据。

步骤1 执行以下命令，则可以从外表查询数据。

```
SELECT * FROM product_info_ext_obs;
```

查询结果显示与**原始数据**显示相同，则表示导入成功。查询结果的结尾将显示以下信息：

```
(10 rows)
```

通过外表查询到数据后，用户可以将数据插入数据库的普通表。

----结束

导入数据后查询数据

步骤1 在GaussDB(DWS)数据库中，创建导入数据的目标表，用于存储导入的数据。

该表的表结构必须与**创建外表**中创建的外表的表结构保持一致，即字段个数、字段类型要完全一致。

例如，创建一个名为product_info的表，示例如下：

```
DROP TABLE IF EXISTS product_info;

CREATE TABLE product_info
(
    product_price               integer        not null,
    product_id                  char(30)       not null,
    product_time                date           ,
    product_level               char(10)       ,
    product_name                varchar(200)   ,
    product_type1               varchar(20)    ,
    product_type2               char(10)       ,
    product_monthly_sales_cnt   integer        ,
    product_comment_time        date           ,
    product_comment_num         integer        ,
    product_comment_content     varchar(200)
)
with (
orientation = column,
compression=middle
)
DISTRIBUTE BY HASH (product_id);
```

步骤2 执行“INSERT INTO .. SELECT ..”命令从外表导入数据到目标表。

示例：

```
INSERT INTO product_info SELECT * FROM product_info_ext_obs;
```

若出现以下类似信息，说明数据导入成功。

```
INSERT 0 10
```

步骤3 执行SELECT命令，查看从OBS导入到GaussDB(DWS)中的数据。

```
SELECT * FROM product_info;
```

查询结果显示如**原始数据**中所示的数据，表示导入成功。查询结果的结尾将显示以下信息：

```
(10 rows)
```

----结束

9.1.6 清除资源

当完成本教程的示例后，如果您不再需要使用本示例中创建的资源，您可以删除这些资源，以免资源浪费或占用您的配额。步骤如下：

1. **删除外表和目标表**
2. **删除创建的外部服务器**
3. **删除数据库及其所属的用户**

   如果您执行了**（可选）新建用户及数据库并授予外表权限**中的步骤，请删除数据库及其所属的用户。

删除外表和目标表

步骤1 （可选）如果执行了**导入数据后查询数据**，请执行以下命令，删除目标表。

```
DROP TABLE product_info;
```

当结果显示为如下信息，则表示删除成功。

```
DROP TABLE
```

步骤2 执行以下命令，删除外表。

```
DROP FOREIGN TABLE product_info_ext_obs;
```

当结果显示为如下信息，则表示删除成功。

```
DROP FOREIGN TABLE
```

----结束

删除创建的外部服务器

步骤1 使用创建外部服务器的用户连接到外部服务器所在的数据库。

在本示例中，使用的是普通用户dbuser在数据库mydatabase中创建了一个外部服务器 。用户需要通过GaussDB(DWS)提供的数据库客户端连接数据库。例如，使用gsql客户端的用户，可以通过以下两种方法中的一种进行连接：

- 如果已经登录了gsql客户端，可以执行以下命令进行切换：
  ```
  \c mydatabase dbuser;
  ```
  根据提示输入密码。
- 如果已经登录了gsql客户端，您也可以执行**\q**退出gsql后，再执行以下命令重新进行连接：
  ```
  gsql -d mydatabase -h 192.168.2.30 -U dbuser -p 8000 -r
  ```
  根据提示输入密码。

步骤2 删除创建的外部服务器。

执行以下命令进行删除，详细语法请参见DROP SERVER。

```
DROP SERVER obs_server;
```

返回以下信息表示删除成功：

```
DROP SERVER
```

查看外部服务器：

```
SELECT * FROM pg_foreign_server WHERE srvname='obs_server';
```

返回结果如下所示，表示已经删除成功：

```
 srvname | srvowner | srvfdw | srvtype | srvversion | srvacl | srvoptions
---------+----------+--------+---------+------------+--------+------------
(0 rows)
```

----结束

删除数据库及其所属的用户

如果您执行了**（可选）新建用户及数据库并授予外表权限**中的步骤，请参照以下步骤删除数据库及其所属的用户。

步骤1 删除自定义数据库。

通过GaussDB(DWS)提供的数据库客户端连接默认数据库postgres。

如果已经登录了gsql客户端，可以直接执行如下命令进行切换：

先切换到默认数据库postgres：

```
\c postgres
```

根据界面提示输入密码。

执行以下命令，删除自定义数据库：

```
DROP DATABASE mydatabase;
```

返回以下信息表示删除成功：

```
DROP DATABASE
```

步骤2 使用管理员用户，删除本示例中创建的普通用户。

使用数据库管理员用户通过GaussDB(DWS)提供的数据库客户端连接数据库。

如果已经登录了gsgl客户端，可以直接执行如下命令进行切换：

```
\c postgres dbadmin
```

执行以下命令回收创建外部服务器的权限：

```
REVOKE ALL ON FOREIGN DATA WRAPPER dfs_fdw FROM dbuser;
REVOKE
```

其中FOREIGN DATA WRAPPER的名字只能是dfs_fdw，dbuser为创建SERVER的用户名。

执行以下命令删除用户：

```
DROP USER dbuser;
DROP USER
```

可使用**\du**命令查询用户，确认用户是否已经删除。

----结束

9.2 查询 MRS 上的数据

GaussDB(DWS)支持远程读MRS的数据，及将MRS上的数据导入到GaussDB(DWS)进行查询。详细请参考**从MRS导入数据到集群**。

10 PostGIS Extension

10.1 PostGIS 概述

GaussDB(DWS)提供PostGIS Extension（版本为PostGIS-2.4.2）。PostGIS Extension是PostgreSQL的空间数据库扩展，提供如下空间信息服务功能：空间对象、空间索引、空间操作函数和空间操作符。PostGIS Extension完全遵循OpenGIS规范。

GaussDB(DWS)中PostGIS Extension依赖第三方开源软件如下。

- Geos 3.6.2
- Proj 4.9.2
- Json 0.12.1
- Libxml2 2.7.1
- Gdal 1.11.0

10.2 PostGIS 使用

创建 Extension

创建PostGIS Extension可直接使用CREATE EXTENSION命令进行创建：

```
CREATE EXTENSION postgis;
```

PostGIS Extension创建后，可根据需要执行如下命令使能PostGIS中的栅格功能。

```
CREATE EXTENSION postgis_raster;
```

使用 Extension

PostGIS Extension函数调用格式为：

```
SELECT GisFunction (Param1, Param2,.....);
```

其中GisFunction为函数名，Param1、Param2等为函数参数名。下列SQL语句展示PostGIS的简单使用，对于各函数的具体使用，请参考**《PostGIS-2.4.2用户手册》**。

示例1：几何表的创建。

```
CREATE TABLE cities ( id integer, city_name varchar(50) );
SELECT AddGeometryColumn('cities', 'position', 4326, 'POINT', 2);
```

示例2：几何数据的插入。

```
INSERT INTO cities (id, position, city_name) VALUES (1,ST_GeomFromText('POINT(-9.5 23)',4326),'CityA');
INSERT INTO cities (id, position, city_name) VALUES (2,ST_GeomFromText('POINT(-10.6 40.3)',4326),'CityB');
INSERT INTO cities (id, position, city_name) VALUES (3,ST_GeomFromText('POINT(20.8 30.3)',4326), 'CityC');
```

示例3：计算三个城市间任意两个城市距离。

```
SELECT p1.city_name,p2.city_name,ST_Distance(p1.position,p2.position) FROM cities AS p1, cities AS p2
WHERE p1.id > p2.id;
```

删除 Extension

在GaussDB(DWS)中删除PostGIS Extension的方法如下所示：

```
DROP EXTENSION postgis [CASCADE];
```

说明

如果Extension被其它对象依赖（如创建的几何表），需要加入CASCADE（级联）关键字，删除所有依赖对象。

10.3 PostGIS 支持和限制

支持数据类型

GaussDB(DWS)的PostGIS Extension支持如下数据类型:

- box2d
- box3d
- geometry_dump
- geometry
- geography
- raster

说明

- 栅格相关的三个GUC参数postgis.gdal_datapath、postgis.gdal_enabled_drivers及postgis.enable_outdb_rasters内部已经完全使能，使用时不用再手动设置。
- 创建Postgis和使用Postgis非同一用户时，请设置如下GUC参数：
  ```
  SET behavior_compat_options = 'bind_procedure_searchpath';
  ```

支持的操作符和函数列表

表 10-1 PostGIS 支持的操作符和函数列表

函数分类	包含函数
Management Functions	AddGeometryColumn、DropGeometryColumn、DropGeometryTable、PostGIS_Full_Version、PostGIS_GEOS_Version、PostGIS_Liblwgeom_Version、PostGIS_Lib_Build_Date、PostGIS_Lib_Version、PostGIS_PROJ_Version、PostGIS_Scripts_Build_Date、PostGIS_Scripts_Installed、PostGIS_Version、PostGIS_LibXML_Version、PostGIS_Scripts_Released、Populate_Geometry_Columns 、UpdateGeometrySRID
Geometry Constructors	ST_BdPolyFromText 、ST_BdMPolyFromText 、ST_Box2dFromGeoHash、ST_GeogFromText、ST_GeographyFromText、ST_GeogFromWKB、ST_GeomCollFromText、ST_GeomFromEWKB、ST_GeomFromEWKT、ST_GeometryFromText、ST_GeomFromGeoHash、ST_GeomFromGML、ST_GeomFromGeoJSON、ST_GeomFromKML、ST_GMLToSQL、ST_GeomFromText 、ST_GeomFromWKB、ST_LineFromMultiPoint、ST_LineFromText、ST_LineFromWKB、ST_LinestringFromWKB、ST_MakeBox2D、ST_3DMakeBox、ST_MakeEnvelope、ST_MakePolygon、ST_MakePoint、ST_MakePointM、ST_MLineFromText、ST_MPointFromText、ST_MPolyFromText、ST_Point、ST_PointFromGeoHash、ST_PointFromText、ST_PointFromWKB、ST_Polygon、ST_PolygonFromText、ST_WKBToSQL、ST_WKTToSQL
Geometry Accessors	GeometryType、ST_Boundary、ST_CoordDim、ST_Dimension、ST_EndPoint、ST_Envelope、ST_ExteriorRing、ST_GeometryN、ST_GeometryType、ST_InteriorRingN、ST_IsClosed、ST_IsCollection、ST_IsEmpty、ST_IsRing、ST_IsSimple、ST_IsValid、ST_IsValidReason、ST_IsValidDetail、ST_M、ST_NDims、ST_NPoints、ST_NRings、ST_NumGeometries、ST_NumInteriorRings、ST_NumInteriorRing、ST_NumPatches、ST_NumPoints、ST_PatchN、ST_PointN、ST_SRID、ST_StartPoint、ST_Summary、ST_X、ST_XMax、ST_XMin、ST_Y、ST_YMax、ST_YMin、ST_Z、ST_ZMax、ST_Zmflag、ST_ZMin
Geometry Editors	ST_AddPoint、ST_Affine、ST_Force2D、ST_Force3D、ST_Force3DZ、ST_Force3DM、ST_Force4D、ST_ForceCollection、ST_ForceSFS、ST_ForceRHR、ST_LineMerge、ST_CollectionExtract、ST_CollectionHomogenize、ST_Multi、ST_RemovePoint、ST_Reverse、ST_Rotate、ST_RotateX、ST_RotateY、ST_RotateZ、ST_Scale、ST_Segmentize、ST_SetPoint、ST_SetSRID、ST_SnapToGrid、ST_Snap、ST_Transform、ST_Translate、ST_TransScale

函数分类	包含函数
Geometry Outputs	ST_AsBinary、ST_AsEWKB、ST_AsEWKT、ST_AsGeoJSON、ST_AsGML、ST_AsHEXEWKB、ST_AsKML、ST_AsLatLonText 、ST_AsSVG、ST_AsText、ST_AsX3D、ST_GeoHash
Operators	&&、&&&、&<、&<|、&>、<<、<<|、=、>>、@ 、|&> 、|>>、~、~=、<->、<#>
Spatial Relationships and Measurements	ST_3DClosestPoint、ST_3DDistance、ST_3DDWithin、ST_3DDFullyWithin、ST_3DIntersects、ST_3DLongestLine、ST_3DMaxDistance、ST_3DShortestLine、ST_Area、ST_Azimuth、ST_Centroid、ST_ClosestPoint、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Crosses、ST_LineCrossingDirection、ST_Disjoint、ST_Distance、ST_HausdorffDistance、ST_MaxDistance、ST_DistanceSphere、ST_DistanceSpheroid、ST_DFullyWithin、ST_DWithin、ST_Equals、ST_HasArc、ST_Intersects、ST_Length、ST_Length2D、ST_3DLength、ST_Length_Spheroid、ST_Length2D_Spheroid、ST_3DLength_Spheroid、ST_LongestLine、ST_OrderingEquals、ST_Overlaps、ST_Perimeter、ST_Perimeter2D、ST_3DPerimeter、ST_PointOnSurface、ST_Project、ST_Relate、ST_RelateMatch、ST_ShortestLine、ST_Touches、ST_Within
Geometry Processing	ST_Buffer、ST_BuildArea、ST_Collect、ST_ConcaveHull、ST_ConvexHull、ST_CurveToLine、ST_DelaunayTriangles、ST_Difference、ST_Dump、ST_DumpPoints、ST_DumpRings、ST_FlipCoordinates、ST_Intersection、ST_LineToCurve、ST_MakeValid、ST_MemUnion、ST_MinimumBoundingCircle、ST_Polygonize、ST_Node、ST_OffsetCurve、ST_RemoveRepeatedPoints、ST_SharedPaths、ST_Shift_Longitude、ST_Simplify、ST_SimplifyPreserveTopology、ST_Split、ST_SymDifference、ST_Union、ST_UnaryUnion
Linear Referencing	ST_LineInterpolatePoint、ST_LineLocatePoint、ST_LineSubstring、ST_LocateAlong、ST_LocateBetween、ST_LocateBetweenElevations、ST_InterpolatePoint、ST_AddMeasure
Miscellaneous Functions	ST_Accum、Box2D、Box3D、ST_Expand、ST_Extent、ST_3Dextent、Find_SRID、ST_MemSize
Exceptional Functions	PostGIS_AddBBox、PostGIS_DropBBox、PostGIS_HasBBox
Raster Management Functions	AddRasterConstraints、DropRasterConstraints、AddOverviewConstraints、DropOverviewConstraints、PostGIS_GDAL_Version、PostGIS_Raster_Lib_Build_Date、PostGIS_Raster_Lib_Version、ST_GDALDrivers、UpdateRasterSRID

函数分类	包含函数
Raster Constructors	ST_AddBand、ST_AsRaster、ST_Band、ST_MakeEmptyRaster、ST_Tile、ST_FromGDALRaster
Raster Accessors	ST_GeoReference、ST_Height、ST_IsEmpty、ST_MetaData、ST_NumBands、ST_PixelHeight、ST_PixelWidth、ST_ScaleX、ST_ScaleY、ST_RasterToWorldCoord、ST_RasterToWorldCoordX、ST_RasterToWorldCoordY、ST_Rotation、ST_SkewX、ST_SkewY、ST_SRID、ST_Summary、ST_UpperLeftX、ST_UpperLeftY、ST_Width、ST_WorldToRasterCoord、ST_WorldToRasterCoordX、ST_WorldToRasterCoordY
Raster Band Accessors	ST_BandMetaData、ST_BandNoDataValue、ST_BandIsNoData、ST_BandPath、ST_BandPixelType、ST_HasNoBand
Raster Pixel Accessors and Setters	ST_PixelAsPolygon、ST_PixelAsPolygons、ST_PixelAsPoint、ST_PixelAsPoints、ST_PixelAsCentroid、ST_PixelAsCentroids、ST_Value、ST_NearestValue、ST_Neighborhood、ST_SetValue、ST_SetValues、ST_DumpValues、ST_PixelOfValue
Raster Editors	ST_SetGeoReference、ST_SetRotation、ST_SetScale、ST_SetSkew、ST_SetSRID、ST_SetUpperLeft、ST_Resample、ST_Rescale、ST_Reskew、ST_SnapToGrid、ST_Resize、ST_Transform
Raster Band Editors	ST_SetBandNoDataValue、ST_SetBandIsNoData
Raster Band Statistics and Analytics	ST_Count、ST_CountAgg、ST_Histogram、ST_Quantile、ST_SummaryStats、ST_SummaryStatsAgg、ST_ValueCount
Raster Outputs	ST_AsBinary、ST_AsGDALRaster、ST_AsJPEG、ST_AsPNG、ST_AsTIFF
Raster Processing	ST_Clip、ST_ColorMap、ST_Intersection、ST_MapAlgebra、ST_Reclass、ST_Union、ST_Distinct4ma、ST_InvDistWeight4ma、ST_Max4ma、ST_Mean4ma、ST_Min4ma、ST_MinDist4ma、ST_Range4ma、ST_StdDev4ma、ST_Sum4ma、ST_Aspect、ST_HillShade、ST_Roughness、ST_Slope、ST_TPI、ST_TRI、Box3D、ST_ConvexHull、ST_DumpAsPolygons、ST_Envelope、ST_MinConvexHull、ST_Polygon、ST_Contains、ST_ContainsProperly、ST_Covers、ST_CoveredBy、ST_Disjoint、ST_Intersects、ST_Overlaps、ST_Touches、ST_SameAlignment、ST_NotSameAlignmentReason、ST_Within、ST_DWithin、ST_DFullyWithin
Raster Operators	&&、&<、&>、=、@、~=、~

空间索引

GaussDB(DWS)数据库的PostGIS Extension支持GIST (Generalized Search Tree) 空间索引（分区表除外）。相比于B-tree索引，GIST索引适应于任意类型的非常规数据结构，可有效提高几何和地理数据信息的检索效率。

使用如下命令创建GIST索引：

```
CREATE INDEX indexname ON tablename USING GIST ( geometryfield );
```

扩展限制

- 只支持行存表。
- 只支持Oracle兼容格式数据库。
- 不支持拓扑对象管理模块Topology。
- 不支持BRIN索引。
- spatial_ref_sys表在扩容期间只支持查询操作。

10.4 OPEN SOURCE SOFTWARE NOTICE (For PostGIS)

This document contains open source software notice for the product. And this document is confidential information of copyright holder. Recipient shall protect it in due care and shall not disseminate it without permission.

Warranty Disclaimer

This document is provided “as is” without any warranty whatsoever, including the accuracy or comprehensiveness. Copyright holder of this document may change the contents of this document at any time without prior notice, and copyright holder disclaims any liability in relation to recipient’s use of this document.

Open source software is provided by the author “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of open source software, even if advised of the possibility of such damage.

Copyright Notice And License Texts

Software: postgis-2.4.2

Copyright notice:

"Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Copyright 2008 Kevin Neufeld

Copyright (c) 2009 Walter Bruce Sinclair

Copyright 2006-2013 Stephen Woodbridge.

Copyright (c) 2008 Walter Bruce Sinclair

Copyright (c) 2012 TJ Holowaychuk <tj@vision-media.ca>

Copyright (c) 2008, by Attractive Chaos <attractivechaos@aol.co.uk>

Copyright (c) 2001-2012 Walter Bruce Sinclair

Copyright (c) 2010 Walter Bruce Sinclair

Copyright 2006 Stephen Woodbridge

Copyright 2006-2010 Stephen Woodbridge.

Copyright (c) 2006-2014 Stephen Woodbridge.

Copyright (c) 2017, Even Rouault <even.rouault at spatialys.com>

Copyright (C) 2004-2015 Sandro Santilli <strk@kbt.io>

Copyright (C) 2008-2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2008 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright 2015 Nicklas Avén <nicklas.aven@jordogskog.no>

Copyright 2008 Paul Ramsey

Copyright (C) 2012 Sandro Santilli <strk@kbt.io>

Copyright 2012 Sandro Santilli <strk@kbt.io>

Copyright (C) 2014 Sandro Santilli <strk@kbt.io>

Copyright 2013 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2009 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2008 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2011 Sandro Santilli <strk@kbt.io>

Copyright 2015 Daniel Baston

Copyright 2009 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2014 Kashif Rasul <kashif.rasul@gmail.com> and

Shoaib Burq <saburq@gmail.com>

Copyright 2013 Sandro Santilli <strk@kbt.io>

Copyright 2010 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2015 Sandro Santilli <strk@kbt.io>

Copyright (C) 2009 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011 Sandro Santilli <strk@kbt.io>

Copyright 2010 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2014 Nicklas Avén

Copyright 2011-2016 Regina Obe

Copyright (C) 2008 Paul Ramsey

Copyright (C) 2011-2015 Sandro Santilli <strk@kbt.io>

Copyright 2010-2012 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright (C) 2015 Daniel Baston <dbaston@gmail.com>

Copyright (C) 2013 Nicklas Avén

Copyright (C) 2016 Sandro Santilli <strk@kbt.io>

Copyright 2017 Darafei Praliaskouski <me@komzpa.net>

Copyright (c) 2016, Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011-2012 Sandro Santilli <strk@kbt.io>

Copyright (C) 2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2007-2008 Mark Cave-Ayland

Copyright (C) 2001-2006 Refractions Research Inc.

Copyright 2015 Daniel Baston <dbaston@gmail.com>

Copyright 2009 David Skea <David.Skea@gov.bc.ca>

Copyright (C) 2012-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2012-2015 Sandro Santilli <strk@kbt.io>

Copyright 2001-2006 Refractions Research Inc.

Copyright (C) 2004 Refractions Research Inc.

Copyright 2011-2014 Sandro Santilli <strk@kbt.io>

Copyright 2009-2010 Sandro Santilli <strk@kbt.io>

Copyright 2015-2016 Daniel Baston <dbaston@gmail.com>

Copyright 2011-2015 Sandro Santilli <strk@kbt.io>

Copyright 2007-2008 Mark Cave-Ayland

Copyright 2012-2013 Oslandia <infos@oslandia.com>

Copyright (C) 2015-2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2001-2003 Refractions Research Inc.

Copyright 2016 Sandro Santilli <strk@kbt.io>

Copyright 2011 Kashif Rasul <kashif.rasul@gmail.com>

Copyright (C) 2014 Nicklas Avén

Copyright (C) 2010 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2011 Sandro Santilli <strk@kbt.io>

Copyright (C) 2011-2014 Sandro Santilli <strk@kbt.io>

Copyright (C) 1984, 1989-1990, 2000-2015 Free Software Foundation, Inc.

Copyright (C) 2011 Paul Ramsey

Copyright 2001-2003 Refractions Research Inc.

Copyright 2009-2010 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright 2010-2012 Oslandia

Copyright 2006 Corporacion Autonoma Regional de Santander

Copyright 2013 Nicklas Avén

Copyright 2011-2016 Arrival 3D, Regina Obe

Copyright (C) 2009 David Skea <David.Skea@gov.bc.ca>

Copyright (C) 2017 Sandro Santilli <strk@kbt.io>

Copyright (C) 2009-2012 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010 - Oslandia

Copyright (C) 2006 Mark Leslie <mark.leslie@lisasoft.com>

Copyright (C) 2008-2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (C) 2009-2015 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2010 Olivier Courtin <olivier.courtin@camptocamp.com>

Copyright 2010 Nicklas Avén

Copyright 2012 Paul Ramsey

Copyright 2011 Nicklas Avén

Copyright 2002 Thamer Alharbash

Copyright 2011 OSGeo

Copyright (C) 2009-2011 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2008 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (C) 2004-2007 Refractions Research Inc.

Copyright 2010 LISAsoft Pty Ltd

Copyright 2010 Mark Leslie

Copyright (c) 1999, Frank Warmerdam

Copyright 2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (c) 2007, Frank Warmerdam

Copyright 2008 OpenGeo.org

Copyright (C) 2008 OpenGeo.org

Copyright (C) 2009 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright 2010 LISAsoft

Copyright (C) 2010 Mark Cave-Ayland <mark.cave-ayland@siriusit.co.uk>

Copyright (c) 1999, 2001, Frank Warmerdam

Copyright (C) 2016-2017 Bj?rn Harrtell <bjorn@wololo.org>

Copyright (C) 2017 Danny G?tte <danny.goette@fem.tu-ilmenau.de>

Copyright 2009-2011 Paul Ramsey <pramsey@cleverelephant.ca>

^copyright^

Copyright 2012 (C) Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2006 Refractions Research Inc.

Copyright 2009 Paul Ramsey <pramsey@opengeo.org>

Copyright 2001-2009 Refractions Research Inc.

Copyright (C) 2010 Olivier Courtin <olivier.courtin@oslandia.com>

By Nathan Wagner, copyright disclaimed,

this entire file is in the public domain

Copyright 2009-2011 Olivier Courtin <olivier.courtin@oslandia.com>

Copyright (C) 2001-2005 Refractions Research Inc.

Copyright 2001-2011 Refractions Research Inc.

Copyright 2009-2014 Sandro Santilli <strk@kbt.io>

Copyright (C) 2008 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2010 Sandro Santilli <strk@kbt.io>

Copyright 2012 J Smith <dark.panda@gmail.com>

Copyright 2009 - 2010 Oslandia

Copyright 2009 Oslandia

Copyright 2001-2005 Refractions Research Inc.

Copyright 2016 Paul Ramsey <pramsey@cleverelephant.ca>

Copyright 2016 Daniel Baston <dbaston@gmail.com>

Copyright (C) 2011 OpenGeo.org

Copyright (c) 2003-2017, Troy D. Hanson http:troydhanson.github.com/uthash/

Copyright (C) 2011 Regents of the University of California

Copyright (C) 2011-2013 Regents of the University of California

Copyright (C) 2010-2011 Jorge Arevalo <jorge.arevalo@deimos-space.com>

Copyright (C) 2010-2011 David Zwarg <dzwarg@azavea.com>

Copyright (C) 2009-2011 Pierre Racine <pierre.racine@sbf.ulaval.ca>

Copyright (C) 2009-2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2008-2009 Sandro Santilli <strk@kbt.io>

Copyright (C) 2013 Nathaneil Hunter Clay <clay.nathaniel@gmail.com

Copyright (C) 2013 Nathaniel Hunter Clay <clay.nathaniel@gmail.com>

Copyright (C) 2013 Bborie Park <dustymugs@gmail.com>

Copyright (C) 2013 Nathaniel Hunter Clay <clay.nathaniel@gmail.com>

(C) 2009 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009-2010 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2009-2010 Jorge Arevalo <jorge.arevalo@deimos-space.com>

Copyright (C) 2012 Regents of the University of California

Copyright (C) 2013 Regents of the University of California

Copyright (C) 2012-2013 Regents of the University of California

Copyright (C) 2009 Sandro Santilli <strk@kbt.io>

"

License: The GPL v2 License.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.?

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents

constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy,

distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author

Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Software:Geos

Copyright notice：

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2013 Sandro Santilli <strk@keybit.net>

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright (C) 2005-2011 Refractions Research Inc.

Copyright (C) 2009 Ragi Y. Burhum <ragi@burhum.com>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2005 2006 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006-2011 Refractions Research Inc.

Copyright (C) 2011 Sandro Santilli <strk@keybit.net

Copyright (C) 2009-2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2016 Daniel Baston

Copyright (C) 2008 Sean Gillies

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Refractions Research Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Sandro Santilli <strk@keybit.net>

Copyright (C) 2008-2010 Safe Software Inc.

Copyright (C) 2006-2007 Refractions Research Inc.

Copyright (C) 2005-2007 Refractions Research Inc.

Copyright (C) 2007 Refractions Research Inc.

Copyright (C) 2014 Mika Heiskanen <mika.heiskanen@fmi.fi>

Copyright (C) 2009-2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 2011 Sandro Santilli <strk@keybit.net>

Copyright (C) 2010 Sandro Santilli <strk@keybit.net>

Copyright (C) 2009 Mateusz Loskot

Copyright (C) 2005-2009 Refractions Research Inc.

Copyright (C) 2001-2009 Vivid Solutions Inc.

Copyright (C) 2012 Sandro Santilli <strk@keybit.net>

Copyright (C) 2006 Wu Yongwei

Copyright (C) 2012 Excensus LLC.

Copyright (C) 1996-2015 Free Software Foundation, Inc.

Copyright (c) 1995 Olivier Devillers <Olivier.Devillers@sophia.inria.fr>

Copyright (C) 2007-2010 Safe Software Inc.

Copyright (C) 2010 Safe Software Inc.

Copyright (C) 2006 Refractions Research

Copyright 2004 Sean Gillies, sgillies@frii.com

Copyright (C) 2011 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2015 Nyall Dawson <nyall dot dawson at gmail dot com>

Original code (2.0 and earlier)copyright (c) 2000-2006 Lee Thomason (www.grinninglizard.com)

Original code (2.0 and earlier)copyright (c) 2000-2002 Lee Thomason (www.grinninglizard.com)

License：LGPL V2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public

Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and

is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in

non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the

freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The

former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under

copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an

appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the

Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1

above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of

its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in

themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or

collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany

it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to

distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a

work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License.

Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object

file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6,

whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work

under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system,

rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception,

the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on

which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot

use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions.

You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the

integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot

impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time.

Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does

not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is

copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status

of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING

RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990

Ty Coon, President of Vice

That's all there is to it!

Software: JSON-C

Copyright notice:

Copyright (c) 2004, 2005 Metaparadigm Pte. Ltd.

Copyright (c) 2009-2012 Eric Haszlakiewicz

Copyright (c) 2004, 2005 Metaparadigm Pte Ltd

Copyright (c) 2009 Hewlett-Packard Development Company, L.P.

Copyright 2011, John Resig

Copyright 2011, The Dojo Foundation

Copyright (c) 2012 Eric Haszlakiewicz

Copyright (c) 2009-2012 Hewlett-Packard Development Company, L.P.

Copyright (c) 2008-2009 Yahoo! Inc. All rights reserved.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006,

2007, 2008, 2009, 2010, 2011 Free Software Foundation, Inc.

Copyright (c) 2013 Metaparadigm Pte. Ltd.

License: MIT License

Copyright (c) 2009-2012 Eric Haszlakiewicz

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright (c) 2004, 2005 Metaparadigm Pte Ltd

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Software: proj

Copyright notice:

"Copyright (C) 2010 Mateusz Loskot <mateusz@loskot.net>

Copyright (C) 2007 Douglas Gregor <doug.gregor@gmail.com>

Copyright (C) 2007 Troy Straszheim

CMake, Copyright (C) 2009-2010 Mateusz Loskot <mateusz@loskot.net>)

Copyright (C) 2011 Nicolas David <nicolas.david@ign.fr>

Copyright (c) 2000, Frank Warmerdam

Copyright (c) 2011, Open Geospatial Consortium, Inc.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006,

2007, 2008, 2009, 2010, 2011 Free Software Foundation, Inc.

Copyright (c) Charles Karney (2012-2015) <charles@karney.com> and licensed

Copyright (c) 2005, Antonello Andrea

Copyright (c) 2010, Frank Warmerdam

Copyright (c) 1995, Gerald Evenden

Copyright (c) 2000, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2010, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2013, Frank Warmerdam

Copyright (c) 2003 Gerald I. Evenden

Copyright (c) 2012, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2002, Frank Warmerdam

Copyright (c) 2004 Gerald I. Evenden

Copyright (c) 2012 Martin Raspaud

Copyright (c) 2001, Thomas Flemming, tf@ttqv.com

Copyright (c) 2002, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2009, Frank Warmerdam

Copyright (c) 2003, 2006 Gerald I. Evenden

Copyright (c) 2011, 2012 Martin Lambers <marlam@marlam.de>

Copyright (c) 2006, Andrey Kiselev

Copyright (c) 2008-2012, Even Rouault <even dot rouault at mines-paris dot org>

Copyright (c) 2001, Frank Warmerdam

Copyright (c) 2001, Frank Warmerdam <warmerdam@pobox.com>

Copyright (c) 2008 Gerald I. Evenden

"

License: MIT License

Please see above

Software: libxml2

Copyright notice:

"See Copyright for the status of this software.

Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

Copyright (C) 2003 Daniel Veillard.

copy: see Copyright for the status of this software.

copy: see Copyright for the status of this software

copy: see Copyright for the status of this software.

Copyright (C) 2000 Bjorn Reese and Daniel Veillard.

Copy: See Copyright for the status of this software.

See COPYRIGHT for the status of this software

Copyright (C) 2000 Gary Pennington and Daniel Veillard.

Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004, 2005, 2006,

2007 Free Software Foundation, Inc.

Copyright (C) 1998 Bjorn Reese and Daniel Stenberg.

Copyright (C) 2001 Bjorn Reese <breese@users.sourceforge.net>

Copyright (C) 2000 Bjorn Reese and Daniel Stenberg.

Copyright (C) 2001 Bjorn Reese and Daniel Stenberg.

See Copyright for the status of this software

"

License: MIT License

Please see above

11 配置 GUC 参数

11.1 查看参数当前取值

GaussDB(DWS)安装后，有一套默认的运行参数，为了使GaussDB(DWS)与业务的配合度更高，用户需要根据业务场景和数据量的大小进行GUC参数调整。

操作步骤

步骤1 查看数据库运行参数当前取值。

- 方法一：使用SHOW命令。
  - 使用如下命令查看单个参数：
    ```
    SHOW server_version;
    ```
    server_version显示数据库版本信息的参数。
  - 使用如下命令查看所有参数：
    ```
    SHOW ALL;
    ```
- 方法二：使用pg_settings视图。
  - 使用如下命令查看单个参数：
    ```
    SELECT * FROM pg_settings WHERE NAME='server_version';
    ```
  - 使用如下命令查看所有参数：
    ```
    SELECT * FROM pg_settings;
    ```

----结束

示例

查看服务器的版本号。

```
SHOW server_version;
 server_version
----------------
 9.2.4
(1 row)
```

11.2 重设参数

背景信息

GaussDB(DWS)提供了多种修改GUC参数的方法，用户可以方便的针对数据库、用户、会话进行设置。

- 参数名称不区分大小写。
- 参数取值有整型、浮点型、字符串、布尔型和枚举型五类。
  - 布尔值可以是（on，off）、（true，false）、（yes，no）或者（1，0），且不区分大小写。
  - 枚举类型的取值是在系统表pg_settings的enumvals字段取值定义的。
- 对于有单位的参数，在设置时请指定单位，否则将使用默认的单位。
  - 参数的默认单位在系统表pg_settings的unit字段定义的。
  - 内存单位有：KB（千字节）、MB（兆字节）和GB（吉字节）。
  - 时间单位：ms（毫秒）、s（秒）、min（分钟）、h（小时）和d（天）。
- CN和DN参数可以同时进行设置，其他类型的参数不能同时进行设置。

具体参数说明请参见**GUC使用说明**中各参数的详细说明。

GUC 参数设置

GaussDB(DWS)提供了下述类型的GUC参数，具体分类和设置方式请参考**表11-1**：

表 11-1 GUC 参数分类

参数类型	说明	设置方式
SUSET	数据库管理员参数。可在数据库启动时、数据库启动后或者数据库管理员通过SQL进行设置。	支持**表11-2**中的方式设置。
USERSET	普通用户参数。可被任何用户在任何时刻设置。	支持**表11-2**中方式设置。

GaussDB(DWS)提供了下述方式来修改GUC参数，具体操作请参考**表11-2**：

表 11-2 GUC 参数设置方式

序号	设置方法
方式	修改指定数据库，用户，会话级别的参数。 • 设置数据库级别的参数 **ALTER DATABASE** *dbname* **SET** *paraname* **TO** *value;* 在下次会话中生效。 • 设置用户级别的参数 **ALTER USER** *username* **SET** *paraname* **TO** *value;* 在下次会话中生效。 • 设置会话级别的参数 **SET** *paraname* **TO** *value;* 修改本次会话中的取值。退出会话后，设置将失效。

操作步骤

设置参数，以设置explain_perf_mode参数为例。

步骤1 查看explain_perf_mode参数。

```
SHOW explain_perf_mode;
 explain_perf_mode
-------------------
 normal
(1 row)
```

步骤2 设置explain_perf_mode参数。

使用以下任意方式进行设置：

- 设置数据库级别的参数
  ALTER DATABASE postgres SET *explain_perf_mode* **TO** *pretty*;
  在下次会话中生效。
- 设置用户级别的参数
  ALTER USER dbadmin **SET** *explain_perf_mode* **TO** *pretty*;
  在下次会话中生效。
- 设置会话级别的参数
  SET *explain_perf_mode* **TO** *pretty*;

步骤3 检查参数设置的正确性。

```
SHOW explain_perf_mode;
 explain_perf_mode
--------------
 pretty
(1 row)
```

----结束

11.3 GUC 使用说明

数据库提供了许多运行参数，配置这些参数可以影响数据库系统的行为。在修改这些参数时请确保用户理解了这些参数对数据库的影响，否则可能会导致无法预料的结果。

注意事项

- 参数中如果取值范围为字符串，此字符串应遵循操作系统的路径和文件名命名规则。
- 取值范围最大值为INT_MAX的参数，此选项最大值跟所在的操作系统有关。
- 取值范围最大值为DBL_MAX的参数，此选项最大值跟所在的操作系统有关。

11.4 连接和认证

11.4.1 连接设置

介绍设置客户端和服务器连接方式相关的参数。

max_connections

参数说明：允许和数据库连接的最大并发连接数。此参数会影响集群的并发能力。

参数类型：POSTMASTER

取值范围：整型。CN最小值为1，最大值为16384；DN最小值为1， 最大值为262143，由于集群内部存在着各种连接，设置时通常达不到最大值，若日志中出现'invalid value for parameter "max_connections"'，需要调小DN的max_connections值。

默认值：CN节点为800，DN节点为5000，如果该默认值超过内核支持的最大值（在执行gs_initdb的时候判断），系统会提示错误。

设置建议：

CN中此参数建议保持默认值。DN中此参数建议设置为CN的个数乘以CN中此参数的值。

增大这个参数可能导致GaussDB(DWS)要求更多的SystemV共享内存或者信号量，可能超过操作系统缺省配置的最大值。这种情况下，请酌情对数值加以调整。

须知

max_connections取值的设置受max_prepared_transactions的影响，在设置max_connections之前，应确保max_prepared_transactions的值大于或等于max_connections的值，这样可确保每个会话都有一个等待中的预备事务。

sysadmin_reserved_connections

参数说明：为管理员用户预留的最少连接数。

参数类型：POSTMASTER

取值范围：整型，0～262143

默认值：3

application_name

参数说明：连接数据库的客户端程序名称。

参数类型：USERSET

取值范围：字符串。

默认值：gsql

connection_info

参数说明：连接数据库的驱动类型、驱动版本号、当前驱动的部署路径和进程属主用户。（运维类参数，不建议用户设置）

参数类型：USERSET

取值范围：字符串。

默认值：空字符串。

说明

- 空字符串，表示当前连接数据库的驱动不支持自动设置connection_info参数或应用程序未设置。
- 驱动连接数据库的时候自行拼接的connection_info参数格式如下：
  ```
  {"driver_name":"ODBC","driver_version": "(GaussDB A 8.0.0 build 62e7353e) compiled at 2019-06-26 14:56:09 commit 5361 last mr 9168 debug","driver_path":"/usr/local/lib/psqlodbcw.so","os_user":"omm"}
  ```
  默认显示driver_name和driver_version，driver_path和os_user的显示由用户控制。

11.4.2 安全和认证（postgresql.conf）

介绍设置客户端和服务器的安全认证方式的相关参数。

authentication_timeout

参数说明：完成客户端认证的最长时间。如果一个客户端没有在这段时间里完成与服务器端的认证，则服务器自动中断与客户端的连接，这样就避免了出问题的客户端无限制地占用连接数。

参数类型：SIGHUP

取值范围：整型，1～600，最小单位为秒（s）

默认值：1min

auth_iteration_count

参数说明：认证加密信息生成过程中使用的迭代次数。

参数类型：SIGHUP

取值范围：整型，2048～134217728

默认值：50000

须知

迭代次数设置过大会导致认证、用户创建等涉及口令加密的场景性能劣化，请根据实际硬件条件合理设置迭代次数。

session_timeout

参数说明：表明与服务器建立链接后，不进行任何操作的最长时间。

参数类型：USERSET

取值范围：整型，0-86400，最小单位为秒（s），0表示关闭超时设置。

默认值：10min

须知

- GaussDB(DWS) gsql客户端中有自动重连机制，所以针对初始化用户本地连接，超时后gsql表现的现象为断开后重连。
- pooler连接池到其它CN和DN的连接，不受session_timeout参数控制。

ssl

参数说明：启用SSL连接。

参数类型：POSTMASTER

取值范围：布尔型

- on表示启用SSL连接。
- off表示不启用SSL连接。

须知

GaussDB(DWS)目前支持SSL的场景为客户端连接CN场景，该参数目前建议只在CN中开启。

默认值：on

ssl_ciphers

参数说明：指定SSL支持的加密算法列表。

参数类型：POSTMASTER

取值范围：字符串，如果指定多个加密算法，加密算法之间需要以分号分割。

默认值：ALL

ssl_renegotiation_limit

参数说明：指定在会话密钥重新协商之前，通过SSL加密通道可以传输的流量。这个重新协商流量限制机制可以减少攻击者针对大量数据使用密码分析法破解密钥的几率，但是也带来较大的性能损失。流量是指发送和接受的流量总和。

参数类型：USERSET

说明

参数建议保持默认设置，即禁用重协商机制。不建议通过gs_guc工具或其他方式直接在postgresql.conf文件中设置ssl_renegotiation_limit参数，即使设置也不会生效。

取值范围：整型，0~INT_MAX，单位为KB。其中0表示禁用重新协商机制。

默认值：0

password_policy

参数说明：在使用CREATE ROLE/USER或者ALTER ROLE/USER命令创建或者修改GaussDB(DWS)帐户时，该参数决定是否进行密码复杂度检查。

参数类型：SIGHUP

须知

从安全性考虑，请勿关闭密码复杂度策略。

取值范围：整型，0、1

- 0表示不采用任何密码复杂度策略。
- 1表示采用默认密码复杂度校验策略。

默认值：1

password_reuse_time

参数说明：在使用ALTER USER或者ALTER ROLE修改用户密码时，该参数指定是否对新密码进行可重用天数检查。

参数类型：SIGHUP

须知

修改密码时会检查配置参数password_reuse_time和password_reuse_max。

- 当password_reuse_time和password_reuse_max都为正数时，只要满足其中任一个，即可认为密码可以重用。
- 当password_reuse_time为0时，表示不限制密码重用天数，仅限制密码重用次数。
- 当password_reuse_max为0时，表示不限制密码重用次数，仅限制密码重用天数。
- 当password_reuse_time和password_reuse_max都为0时，表示不对密码重用进行限制。

取值范围：浮点型，0~3650，单位为天。

- 0表示不检查密码可重用天数。
- 正数表示新密码不能为该值指定的天数内使用过的密码。

默认值：60

password_reuse_max

参数说明：在使用ALTER USER或者ALTER ROLE修改用户密码时，该参数指定是否对新密码进行可重用次数检查。

参数类型：SIGHUP

须知

修改密码时会检查配置参数password_reuse_time和password_reuse_max。

- 当password_reuse_time和password_reuse_max都为正数时，只要满足其中任一个，即可认为密码可以重用。
- 当password_reuse_time为0时，表示不限制密码重用天数，仅限制密码重用次数。
- 当password_reuse_max为0时，表示不限制密码重用次数，仅限制密码重用天数。
- 当password_reuse_time和password_reuse_max都为0时，表示不对密码重用进行限制。

取值范围：整型，最小值为0，最大值为1000。

- 0表示不检查密码可重用次数。
- 正整数表示新密码不能为该值指定的次数内使用过的密码。

默认值：0

password_lock_time

参数说明：该参数指定帐户被锁定后自动解锁的时间。

参数类型：SIGHUP

须知

password_lock_time和**failed_login_attempts**必须都为正数时锁定和解锁功能才能生效。

取值范围：浮点型，最小值为0，最大值为365，单位为天。

- 0表示密码验证失败时，自动锁定功能不生效。
- 正数表示帐户被锁定后，当锁定时间超过password_lock_time设定的值时，帐户将会被自行解锁。

默认值：1

failed_login_attempts

参数说明：在任意时候，如果输入密码错误的次数达到failed_login_attempts则当前帐户被锁定，password_lock_time秒后被自动解锁。 例如，登录时输入密码失败，ALTER USER时修改密码失败等。

参数类型：SIGHUP

取值范围：整型，最小值为0，最大值为1000。

- 0表示自动锁定功能不生效。
- 正整数表示当错误密码次数达到failed_login_attempts设定的值时，当前帐户将被锁定。

默认值：10

须知

- failed_login_attempts和**password_lock_time**必须都为正数时锁定和解锁功能才能生效。
- failed_login_attempts会与客户端SSL连接模式共同决定用户的密码错误次数。当PGSSLMODE取值是allow或prefer时，客户的一次密码连接请求会生成两次连接请求：一次是尝试SSL连接，另一次是尝试非SSL连接。此时，用户感知到的密码错误次数是failed_login_attempts除以2。

password_encryption_type

参数说明：该字段决定采用何种加密方式对用户密码进行加密存储。

参数类型：SIGHUP

取值范围：整型，0、1

- 0表示采用md5方式对密码加密。
- 1表示采用sha256方式对密码加密。

须知

md5为不安全的加密算法，不建议用户使用。

默认值：1

password_min_length

参数说明：该字段决定帐户密码的最小长度。

参数类型：SIGHUP

取值范围：整型，6~999

默认值：8

password_max_length

参数说明：该字段决定帐户密码的最大长度。

参数类型：SIGHUP

取值范围：整型，6~999

默认值：32

password_min_uppercase

参数说明：该字段决定帐户密码中至少需要包含大写字母个数。

参数类型：SIGHUP

取值范围：整型，0~999

- 0表示没有限制。
- 1~999表示创建账户所指定的密码中至少需要包含大写字母个数。

默认值：0

password_min_lowercase

参数说明：该字段决定帐户密码中至少需要包含小写字母的个数。

参数类型：SIGHUP

取值范围：整型，0~999

- 0表示没有限制。
- 1~999表示创建帐户所指定的密码中至少需要包含小写字母个数。

默认值：0

password_min_digital

参数说明：该字段决定帐户密码中至少需要包含数字的个数。

参数类型：SIGHUP

取值范围：整型，0~999

- 0表示没有限制。
- 1~999表示创建帐户所指定的密码中至少需要包含数字个数。

默认值：0

password_min_special

参数说明：该字段决定帐户密码中至少需要包含特殊字符个数。

参数类型：SIGHUP

取值范围：整型，0~999

- 0表示没有限制。
- 1~999表示创建帐户所指定的密码中至少需要包含特殊字符个数。

默认值：0

password_effect_time

参数说明：该字段决定帐户密码的有效时间。

参数类型：SIGHUP

取值范围：浮点型，0~999，单位为天。

- 0表示不开启有效期限制功能。
- 1~999表示创建帐户所指定的密码有效期，临近或超过有效期系统会提示用户修改密码。

默认值：90

password_notify_time

参数说明：该字段决定帐户密码到期前提醒的天数。

参数类型：SIGHUP

取值范围：整型，0~999，单位为天。

- 0表示不开启提醒功能。
- 1~999表示帐户密码到期前提醒的天数。

默认值：7

11.4.3 通信库参数

本节介绍通信库相关的参数设置及取值范围等内容。

tcp_keepalives_idle

参数说明：在支持TCP_KEEPIDLE套接字选项的系统上，设置发送活跃信号的间隔秒数。不设置发送保持活跃信号，连接就会处于闲置状态。

参数类型：USERSET

须知

- 如果操作系统不支持TCP_KEEPIDLE选项 ，这个参数的值必须为0。
- 在通过Unix域套接字进行的连接的操作系统上，这个参数将被忽略。

取值范围：整型，0~3600，单位为秒（s）。

默认值：0

tcp_keepalives_interval

参数说明：在支持TCP_KEEPINTVL套接字选项的操作系统上，以秒数声明在重新传输之间等待响应的时间。

参数类型：USERSET

取值范围：整型，0~180，单位为秒（s）。

默认值：0

须知

- 如果操作系统不支持TCP_KEEPINTVL选项，这个参数的值必须为0。
- 在通过Unix域套接字进行的连接的操作系统上，这个参数将被忽略。

tcp_keepalives_count

参数说明：在支持TCP_KEEPCNT套接字选项的操作系统上，设置GaussDB(DWS)服务端在断开与客户端连接之前可以等待的保持活跃信号个数。

参数类型：USERSET

须知

- 如果操作系统不支持TCP_KEEPCNT选项，这个参数的值必须为0。
- 在通过Unix域套接字进行连接的操作系统上，这个参数将被忽略。

取值范围：整型，0~100，其中0表示GaussDB(DWS)未收到客户端反馈的保持活跃信号则立即断开连接。

默认值：0

comm_tcp_mode

参数说明：通信库使用TCP或SCTP协议建立数据通道的切换开关，重启集群生效。

参数类型：POSTMASTER

取值范围：布尔型，CN设置为on表示使用TCP模式连接DN，DN设置为on表示DN间使用TCP代理通信。

默认值：on

comm_sctp_port

参数说明：TCP代理通信库或SCTP通信库使用的TCP或SCTP协议监听端口。

参数类型：POSTMASTER

须知

集群部署时会自动分配此端口号，请不要轻易修改此参数，如端口号配置不正确会导致数据库通信失败。

取值范围：整型，0~65535

默认值：port+本机主DN数*2+本DN在本机DN序号

comm_control_port

参数说明：TCP代理通信库或SCTP通信库使用的TCP协议监听端口。

参数类型：POSTMASTER

取值范围：整型，0~65535

默认值：port+本机主DN数*2+本DN在本机DN序号+1

须知

集群部署时会自动分配此端口号，请不要轻易修改此参数，如端口号配置不正确会导致数据库通信失败。

comm_max_datanode

参数说明：TCP代理通信库或SCTP通信库支持的最大DN数。

参数类型：USERSET

取值范围：整型，1~8192

默认值：DN数大于256时，默认值大于等于主DN个数2的N次方。DN数小于等于256时取256。

comm_max_receiver

参数说明：TCP代理通信库或SCTP通信库内部接收线程数量。

参数类型：POSTMASTER

取值范围：整型，1~50

默认值：4

comm_memory_pool_percent

参数说明：单个DN内TCP代理通信库或SCTP通信库可使用内存池资源的百分比，用于自适应负载预留通信库通信消耗的内存大小。

参数类型：POSTMASTER

取值范围：整型，0~100

默认值：0

须知

此参数需根据实际业务情况做调整，若通信库使用内存小，可设置该参数数值较小，反之设置数值较大。

comm_client_bind

参数说明：通信库客户端发起连接时是否使用bind绑定指定IP。

参数类型：USERSET

取值范围：布尔型

- on表示绑定指定IP。
- off表示不绑定指定IP。

须知

如果集群某一节点存在多个IP处于同一通信网段时，需设置为on。此时将绑定本地listen_addresses指定的IP发起通信，随机端口号不能重复使用，集群并发数量会受到可用随机端口号数量的限制。

默认值：off

comm_no_delay

参数说明：是否使用通信库连接的NO_DELAY属性，重启集群生效。

参数类型：USERSET

取值范围：布尔型

默认值：off

须知

如果集群出现因每秒接收数据包过多导致的丢包时，需设置为off，以便小包合并成大包发送，减少数据包总数。

comm_debug_mode

参数说明：TCP代理通信库或SCTP通信库debug模式开关，该参数设置是否打印通信层详细日志。

须知

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库详细debug日志。
- off表示不打印通信库详细debug日志。

默认值：off

comm_ackchk_time

参数说明：无数据包接收情况下，该参数设置通信库服务端主动ACK触发时长。

参数类型：USERSET

取值范围：整型，0～20000，单位为毫秒（ms）。取值为0表示关闭此功能。

默认值：2000

comm_timer_mode

参数说明：TCP代理通信库或SCTP通信库timer模式开关，该参数设置是否打印通信层各阶段时间桩。

须知

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库详细时间桩日志。
- off表示不打印通信库详细时间桩日志。

默认值：off

comm_stat_mode

参数说明：TCP代理通信库或SCTP通信库stat模式开关，该参数设置是否打印通信层的统计信息。

须知

设置为on时，打印日志量较大，会增加额外的overhead并降低数据库性能，仅在调试时打开。

参数类型：USERSET

取值范围：布尔型

- on表示打印通信库统计信息日志。
- off表示不打印通信库统计信息日志。

默认值：off

enable_stateless_pooler_reuse

参数说明：pooler复用切换开关，重启集群生效。

参数类型：POSTMASTER

取值范围：布尔型

- 设置为on/true表示使用pooler复用模式。
- 设置为off/false表示关闭pooler复用模式。

须知

CN和DN需要同步设置。如果CN设置enable_stateless_pooler_reuse为off，DN设置enable_stateless_pooler_reuse为on会导致集群不能正常通信，因此必须对该参数做CN和DN全局相同的配置，重启集群生效。

默认值：off

comm_cn_dn_logic_conn

参数说明：CN和DN间逻辑连接特性开关，重启集群生效。

参数类型：POSTMASTER

取值范围：布尔型

- 设置为on/true表示CN和DN之间连接为逻辑链接，使用libcomm组件。
- 设置为off/false表示CN和DN之间连接为物理连接，使用libpq组件。

须知

如果CN设置comm_cn_dn_logic_conn为off，DN设置comm_cn_dn_logic_conn为on会导致集群不能正常通信，因此必须对该参数做CN和DN全局相同的配置，重启集群生效。

默认值：off

11.5 资源消耗

11.5.1 内存

介绍与内存相关的参数设置。

须知

本节涉及的参数仅在数据库服务重新启动后生效。

enable_memory_limit

参数说明：启用逻辑内存管理模块。

参数类型：POSTMASTER

取值范围：布尔型

- on表示启用逻辑内存管理模块。
- off表示不启用逻辑内存管理模块。

默认值：on

须知

若max_process_memory-shared buffer-cstore buffers少于2G，GaussDB(DWS)强制把enable_memory_limit设置为off。

max_process_memory

参数说明：设置一个数据库节点可用的最大物理内存。

参数类型：POSTMASTER

取值范围：整型，2*1024*1024～INT_MAX/2，单位为KB。

默认值：非从备DN节点自动适配，公式为（物理内存大小）* 0.6 / (1+主DN个数)，当结果不足2GB时，默认取2GB。从备DN默认为12GB。

设置建议：

DN上该数值需要根据系统物理内存及单节点部署主DN个数决定的。计算公式如下：（物理内存大小 – vm.min_free_kbytes）* 0.7 / (n+主DN个数)。该参数目的是尽可能保证系统的可靠性，不会因数据库内存膨胀导致节点OOM。这个公式中提到vm.min_free_kbytes，其含义是预留操作系统内存供内核使用，通常用作操作系统内核中通信收发内存分配，至少为5%内存。即， max_process_memory=物理内存*0.665/(n+主DN个数)，其中，当集群规模小于256时， n=1；当集群规模大于256且小于512时， n=2；当集群规模超过512时，n=3。

CN上该数值内存可设置与DN数值一样。

RAM：集群规划时分配给集群的最大使用内存。

shared_buffers

参数说明：设置GaussDB(DWS)使用的共享内存大小。增加此参数的值会使GaussDB(DWS)比系统默认设置需要更多的System V共享内存。

参数类型：POSTMASTER

取值范围：整型，128～INT_MAX，单位为8KB。

改变BLCKSZ的值会改变最小值。

默认值：CN节点为512MB，DN节点为1GB。如果操作系统支持的共享内存小于32MB，则在初始化数据存储区时会自动调整为操作系统支持的最大值。

设置建议：

由于GaussDB(DWS)大部分查询下推，建议DN中此参数设置比CN大。

建议设置shared_buffers值为内存的40%以内。行存列存分开对待。行存设大，列存设小。列存：(单服务器内存/单服务器DN个数)*0.4*0.25。

如果设置较大的shared_buffers需要同时增加**checkpoint_segments**的值，因为写入大量新增、修改数据需要消耗更多的时间周期。

bulk_write_ring_size

参数说明：数据并行导入使用的环形缓冲区大小。

参数类型：USERSET

取值范围：整型，16384～INT_MAX，单位为KB。

默认值：2GB

设置建议：建议导入压力大的场景中增加DN中此参数配置。

temp_buffers

参数说明：设置每个数据库会话使用的LOCAL临时缓冲区的大小。

参数类型：USERSET

取值范围：整型，800~INT_MAX/2，单位为8KB。

默认值：8MB

说明

- 在每个会话的第一次使用临时表之前可以改变temp_buffers的值，之后的设置将是无效的。
- 一个会话将按照temp_buffers给出的限制，根据需要分配临时缓冲区。如果在一个并不需要大量临时缓冲区的会话里设置一个大的数值，其开销只是一个缓冲区描述符的大小。当缓冲区被使用，就会额外消耗8192字节。

max_prepared_transactions

参数说明：设置可以同时处于"预备"状态的事务的最大数目。增加此参数的值会使GaussDB(DWS)比系统默认设置需要更多的System V共享内存。

当GaussDB(DWS)部署为主备双机时，在备机上此参数的设置必须要高于或等于主机上的，否则无法在备机上进行查询操作。

参数类型：POSTMASTER

取值范围：整型，0~536870911，其中取值为800表示关闭预备事务的特性。

默认值：800

说明

为避免在准备步骤失败，此参数的值不能小于max_connections。

work_mem

参数说明：设置内部排序操作和Hash表在开始写入临时磁盘文件之前使用的内存大小。ORDER BY，DISTINCT和merge joins都要用到排序操作。Hash表在散列连接、散列为基础的聚集、散列为基础的IN子查询处理中都要用到。

对于复杂的查询，可能会同时并发运行好几个排序或者散列操作，每个都可以使用此参数所声明的内存量，不足时会使用临时文件。同样，好几个正在运行的会话可能会同时进行排序操作。因此使用的总内存可能是work_mem的好几倍。

参数类型：USERSET

取值范围：整型，64~2147483647，单位为KB。

默认值：64MB

设置建议：

依据查询特点和并发来确定，一旦work_mem限定的物理内存不够，算子运算数据将写入临时表空间，带来5-10倍的性能下降，查询响应时间从秒级下降到分钟级。

- 对于串行无并发的复杂查询场景，平均每个查询有5-10关联操作，建议work_mem=50%内存/10。
- 对于串行无并发的简单查询场景，平均每个查询有2-5个关联操作，建议work_mem=50%内存/5。
- 对于并发场景，建议work_mem=串行下的work_mem/物理并发数。

query_mem

参数说明：设置执行作业所使用的内存。如果设置的query_mem值大于0，在生成执行计划时，优化器会将作业的估算内存调整为该值。

参数类型：USERSET

取值范围：整型，0，或大于32M的整型，默认单位为KB。如果设置值为负数或小于32M，将设置为默认值0，此时优化器不会根据该值调整作业的估算内存。

默认值：0

query_max_mem

参数说明：设置执行作业所能够使用的最大内存。如果设置的query_max_mem值大于0，当作业执行时所使用内存超过该值时，将报错退出。

参数类型：USERSET

取值范围：整型，0，或大于32M的整型，单位为KB。如果设置值为负数或小于32M，将设置为默认值0，此时不会根据该值限制作业的内存使用。

默认值：0

maintenance_work_mem

参数说明：设置在维护性操作（比如VACUUM、CREATE INDEX、ALTER TABLE ADD FOREIGN KEY等）中可使用的最大的内存。该参数的设置会影响VACUUM、VACUUM FULL、CLUSTER、CREATE INDEX的执行效率。

参数类型：USERSET

取值范围：整型，1024~INT_MAX，单位为KB。

默认值：128MB

设置建议：

- 建议设置此参数的值大于**work_mem**，可以改进清理和恢复数据库转储的速度。因为在一个数据库会话里，任意时刻只有一个维护性操作可以执行，并且在执行维护性操作时不会有太多的会话。
- 当**自动清理**进程运行时，**autovacuum_max_workers**倍数的内存将会被分配，所以此时设置maintenance_work_mem的值应该不小于**work_mem**。
- 如果进行大数据量的cluster等，可以在session中调大该值。

psort_work_mem

参数说明：设置列存表在进行局部排序中在开始写入临时磁盘文件之前使用的内存大小。带partial cluster key的表、带索引的表插入，创建表索引，删除表和更新表都会用到。

参数类型：USERSET

须知

多个正在运行的会话可能会同时进行表的局部排序操作，因此使用的总内存可能是psort_work_mem的好几倍。

取值范围：整型，64~INT_MAX，单位为KB。

默认值：512MB

max_loaded_cudesc

参数说明：设置列存表在做扫描时，每列缓存cudesc信息的个数。增大设置会提高查询性能，但也会增加内存占用，特别是当列存表的列非常多时。

参数类型：USERSET

取值范围：整型，100~INT_MAX/2

默认值：1024

须知

max_loaded_cudesc设置过高时，有可能引起内存分配不足。

max_stack_depth

参数说明：设置GaussDB(DWS)执行堆栈的最大安全深度。需要这个安全界限是因为在服务器里，并非所有程序都检查了堆栈深度，只是在可能递规的过程，比如表达式计算这样的过程里面才进行检查。

参数类型：SUSET

设置原则：

- 此参数的最佳设置是等于操作系统内核允许的最大值（就是ulimit -s的设置）。
- 如果设置此参数的值大于实际的内核限制，则一个正在运行的递归函数可能会导致一个独立的服务器进程崩溃。在GaussDB(DWS)能够检测内核限制的操作系统上（SLES上），将自动限制设置为一个不安全的值。
- 因为并非所有的操作都能够检测，所以建议用户在此设置一个明确的值。

取值范围：整型，100~INT_MAX，单位为KB。

默认值：2MB

说明

默认值2MB，这个值相对比较小，不容易导致系统崩溃。但是可能会因为该值较小，导致无法执行复杂的函数。

cstore_buffers

参数说明：设置列存和HDFS所使用的共享缓冲区的大小。

参数类型：POSTMASTER

取值范围：整型，16384～INT_MAX，单位为KB。

默认值：32MB

设置建议：

列存表使用cstore_buffers设置的共享缓冲区，几乎不用shared_buffers。因此在列存表为主的场景中，应减少shared_buffers，增加cstore_buffers。

enable_orc_cache

参数说明：设置是否允许在初始化cstore_buffers时，将1/4的cstore_buffers空间预留，用于缓存orc元数据。

参数类型：POSTMASTER

取值范围：布尔型

默认值：on

- on表示开启缓存orc元数据，可提升hdfs表的查询性能，但是会占用列存buffer资源，导致列存性能下降。

- off表示关闭缓存orc元数据。

schedule_splits_threshold

参数说明：设置HDFS外表schedule阶段能够在内存中存储的最大文件个数，超过该限制时，将把文件列表下盘处理。

参数类型：USERSET

取值范围：整型，1~INT_MAX

默认值：60000

bulk_read_ring_size

参数说明：并行导出，使用的环形缓冲区大小。

参数类型：USERSET

取值范围：整型，256~INT_MAX，单位为KB。

默认值：16MB

check_cu_size_threshold

参数说明：列存表插入时，如果一个CU中已插入的数据量大于该参数时，开始进行行级大小校验，避免生成非压缩态大于1G的CU。（该参数仅8.0.0.9版本支持）

参数类型：USERSET

取值范围：整形，0~1024，单位为MB。

默认值：1024MB

11.5.2 磁盘空间

介绍与磁盘空间相关的参数，用于限制下盘文件所占用的磁盘空间。

sql_use_spacelimit

参数说明：限制单个SQL在单个DN上，触发落盘操作时，落盘文件的空间大小，管控的空间包括普通表、临时表以及中间结果集落盘占用的空间。

参数类型：USERSET

取值范围：整型，-1~INT_MAX，单位为KB。其中-1表示没有限制。

默认值：-1

temp_file_limit

参数说明：限制一个会话中，触发落盘操作时，单个落盘文件的空间大小。例如一次会话中，排序和哈希表使用的临时文件，或者游标占用的临时文件。

此设置为会话级别的落盘文件控制。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，单位为KB。其中-1表示没有限制。

默认值：-1

须知

SQL查询执行时使用的临时表空间不在此限制。

11.5.3 内核资源使用

介绍与操作系统内核相关的参数，这些参数是否生效依赖于操作系统的设置。

max_files_per_process

参数说明：设置每个服务器进程允许同时打开的最大文件数目。如果操作系统内核强制一个合理的数目，则不需要设置。

但是在一些平台上（特别是大多数BSD系统），内核允许独立进程打开比系统真正可以支持的数目大得多得文件数。如果用户发现有的“Too many open files”这样的失败现象，请尝试缩小这个设置。通常情况下需要满足，系统FD（file descriptor）数量>=最大并发数*当前物理机主DN个数*max_files_per_process*3 。

参数类型：POSTMASTER

取值范围：整型，25~INT_MAX

默认值：1000

11.5.4 基于开销的清理延迟

这个特性的目的是允许管理员减少VACUUM和ANALYZE语句在并发活动的数据库上的I/O影响。 比如，像VACUUM和ANALYZE这样的维护语句并不需要迅速完成， 并且不希望他们严重干扰系统执行其他的数据库操作。基于开销的清理延迟为管理员提供了一个实现这个目的手段。

须知

有些清理操作会持有关键的锁，这些操作应该尽快结束并释放锁。所以GaussDB(DWS)的机制是，在这类操作过程中，基于开销的清理延迟不会发生作用。为了避免在这种情况下的长延时，实际的开销限制取下面两者之间的较大值：

- vacuum_cost_delay*accumulated_balance/vacuum_cost_limit
- vacuum_cost_delay*4

背景信息

在ANALYZE | ANALYSE和VACUUM语句执行过程中，系统维护一个内部的记数器，跟踪所执行的各种I/O操作的近似开销。如果积累的开销达到了vacuum_cost_limit声明的限制，则执行这个操作的进程将睡眠vacuum_cost_delay指定的时间。然后它会重置记数器然后继续执行。

这个特性是缺省关闭的。要想打开它，把vacuum_cost_delay变量设置为一个非零值。

vacuum_cost_delay

参数说明：指定开销超过vacuum_cost_limit的值时，进程睡眠的时间。

参数类型：USERSET

取值范围：整型，0~100，单位为毫秒（ms）。正数值表示打开基于开销的清理延迟特性；0表示关闭基于开销的清理延迟特性。

默认值：0

须知

- 许多系统上，睡眠的有效分辨率是10毫秒。因此把vacuum_cost_delay设置为一个不是10的整数倍的数值与将它设置为下一个10的整数倍作用相同。
- 此参数一般设置较小，常见的设置是10或20毫秒。调整此特性资源占用率时，最好是调整其他参数，而不是该参数。

vacuum_cost_page_hit

参数说明：清理一个在共享缓存里找到的缓冲区的预计开销。他代表锁住缓冲池、查找共享的Hash表、扫描页面内容的开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ms）。

默认值：1

vacuum_cost_page_miss

参数说明：清理一个要从磁盘上读取的缓冲区的预计开销。他代表锁住缓冲池、查找共享Hash表、从磁盘读取需要的数据块、扫描它的内容的开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ms）。

默认值：10

vacuum_cost_page_dirty

参数说明：清理修改一个原先是干净的块的预计开销。他代表把一个脏的磁盘块再次刷新到磁盘上的额外开销。

参数类型：USERSET

取值范围：整型，0~10000，单位为毫秒（ms）。

默认值：20

vacuum_cost_limit

参数说明：导致清理进程休眠的开销限制。

参数类型：USERSET

取值范围：整型，1~10000，单位为毫秒（ms）。

默认值：200

11.5.5 异步 IO

enable_adio_debug

参数说明：允许维护人员输出一些与ADIO相关的日志，便于定位ADIO相关问题。开发人员专用，不建议普通用户使用。

参数类型：SUSET

取值范围：布尔型

- on/true表示开启此日志开关。
- off/false表示关闭此日志开关。

默认值：off

enable_fast_allocate

参数说明：磁盘空间快速分配开关。只有在XFS文件系统上才能开启该开关。

参数类型：SUSET

取值范围：布尔型

- on/true表示开启此功能。
- off/false表示关闭此功能。

默认值：off

prefetch_quantity

参数说明：描述行存储使用ADIO预读取IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为8KB。

默认值：32MB

backwrite_quantity

参数说明：描述行存储使用ADIO写入IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为8KB。

默认值：8MB

cstore_prefetch_quantity

参数说明：描述列存储使用ADIO预取IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为KB。

默认值：32MB

cstore_backwrite_quantity

参数说明：描述列存储使用ADIO写入IO量的大小。

参数类型：USERSET

取值范围：整型，1024～1048576，单位为KB。

默认值：8MB

cstore_backwrite_max_threshold

参数说明：描述列存储使用ADIO写入数据库可缓存最大的IO量。

参数类型：USERSET

取值范围：整型，4096～INT_MAX/2，单位为KB。

默认值：2GB

fast_extend_file_size

参数说明：描述列存储使用ADIO预扩展磁盘的大小。

参数类型：SUSET

取值范围：整型，1024～1048576，单位为KB。

默认值：8MB

effective_io_concurrency

参数说明：磁盘子系统可以同时有效处理的请求数。对于RAID阵列，此参数应该是阵列中驱动器主轴的数量。

参数类型：USERSET

取值范围：整型 ，0~1000

默认值：1

11.6 并行导入

GaussDB(DWS)提供了并行导入功能，以快速、高效地完成大量数据导入。介绍GaussDB(DWS)并行导入的相关参数。

raise_errors_if_no_files

参数说明：导入时是否区分“导入文件记录数为空”和“导入文件不存在”。raise_errors_if_no_files=TRUE，则“导入文件不存在”的时候，GaussDB(DWS)将抛出“文件不存在的”错误。

参数类型：SUSET

取值范围：布尔型

- on表示导入时区分“导入文件记录数为空”和“导入文件不存在”。
- off表示导入时不区分“导入文件记录数为空”和“导入文件不存在”。

默认值：off

partition_mem_batch

参数说明：为了优化对列存分区表的批量插入，在批量插入过程中会对数据进行缓存后再批量写盘。通过partition_mem_batch可指定缓存个数。该值设置过大，将消耗较多系统内存资源；设置过小，将降低系统列存分区表批量插入性能。

参数类型：USERSET

取值范围：1~ 65535

默认值：256

partition_max_cache_size

参数说明：为了优化对列存分区表的批量插入，在批量插入过程中会对数据进行缓存后再批量写盘。通过partition_max_cache_size可指定数据缓存区大小。该值设置过大，将消耗较多系统内存资源；设置过小，将降低列存分区表批量插入性能。

参数类型：USERSET

取值范围：

- 列存分区表：4096~ INT_MAX / 2，最小单位为KB。

默认值：2GB

gds_debug_mod

参数说明：为了增强对Gauss Data Service（以下简称GDS）相关问题的分析定位能力，可以通过此参数选择是否开启GDS的debug功能。参数开启后，将在集群节点对应的日志中输出GDS每次收发的包裹类型、命令交互的对端以及其他交互相关的细节信息，方便记录Gaussdb端状态机的状态跳转，以及目前所处的状态信息。此参数打开会输出额外日志，增加日志IO开销，进而影响性能和日志的信息有效性，因此请仅在定位GDS问题时开启。

参数类型：USERSET

取值范围：布尔型

- on表示开启GDS debug功能。
- off表示不开启GDS debug功能。

默认值：off

enable_delta_store

参数说明：为了增强列存单条数据导入的性能和解决磁盘冗余问题，可通过此参数选择是否开启支持列存delta表功能。该参数开启时，数据导入列存表，会根据表定义时指定的**DELTAROW_THRESHOLD**决定数据进入delta表存储还是主表CU存储，当数据量小于DELTAROW_THRESHOLD时，数据进入delta表。该参数影响的操作包括

insert，copy，vacuum，vacuum full，vacuum deltamerge重分布等所有涉及列存表数据移动的操作。

参数类型：POSTMASTER

取值范围：布尔型

- on表示开启列存delta表功能。
- off表示不开启列存delta表功能。

默认值：off

11.7 预写式日志

11.7.1 设置

wal_level

参数说明：设置写入WAL信息量的级别。

参数类型：POSTMASTER

取值范围：枚举类型

- minimal

  优点：一些重要操作（包括创建表、创建索引、簇操作和表的复制）都能安全的跳过，这样就可以使操作变得更快。

  缺点：WAL仅提供从数据库服务器崩溃或者紧急关闭状态恢复时所需要的基本信息，无法用WAL归档日志恢复数据。

- archive

  这个参数增加了WAL归档需要的日志信息，从而可以支持数据库的归档恢复。

- hot_standby
  - 这个参数进一步增加了在备机上运行的SQL查询的信息，这个参数只能在数据库服务重新启动后生效。
  - 为了在备机上开启只读查询，wal_level必须在主机上设置成hot_standby，并且备机必须打开hot_standby参数。hot_standby和archive级别之间的性能只有微小的差异，如果它们的设置对产品的性能影响有明显差异，欢迎反馈。

默认值：hot_standby

须知

- 如果需要启用WAL日志归档和主备机的数据流复制，必须将此参数设置为archive或者hot_standby。
- 如果此参数设置为archive，hot_standby必须设置为off，否则将导致数据库无法启动。

synchronous_commit

参数说明：设置当前事务的同步方式。

参数类型：USERSET

取值范围：枚举类型

- on表示将备机的同步日志刷新到磁盘。
- off表示异步提交。
- local表示为本地提交。
- remote_write表示要备机的同步日志写到磁盘。
- remote_receive表示要备机同步日志接收数据。

默认值：on

wal_buffers

参数说明：设置用于存放WAL数据的共享内存空间的XLOG_BLCKSZ数，XLOG_BLCKSZ的大小默认为8KB。

参数类型：POSTMASTER

取值范围：-1~2^{18}，单位为8KB。

- 如果设置为-1，表示wal_buffers的大小随着参数shared_buffers自动调整，为shared_buffers的1/32，最小值为8个XLOG_BLCKSZ，最大值为2048个XLOG_BLCKSZ。
- 如果设置为其他值，当小于8时，会被默认设置为8；当大于2048的时，会被强制设置为2048。

默认值：16MB

设置建议：每次事务提交时，WAL缓冲区的内容都写入到磁盘中，因此设置为很大的值不会带来明显的性能提升。如果将它设置成几百兆，就可以在有很多即时事务提交的服务器上提高写入磁盘的性能。根据经验来说，默认值可以满足大多数的情况。

commit_delay

参数说明：表示一个已经提交的数据在WAL缓冲区中存放的时间。

参数类型：USERSET

取值范围：整型， 0～100000（微秒），其中0表示无延迟。

默认值：0

须知

- 设置为非 0 值时事务执行commit后不会立即写入WAL中，而仍存放在WAL缓冲区中，等待WalWriter进程周期性写入磁盘。
- 如果系统负载很高，在延迟时间内，其他事务可能已经准备好提交。但如果没有事务准备提交，这个延迟就是在浪费时间。

commit_siblings

参数说明：当一个事务发出提交请求时，如果数据库中正在执行的事务数量大于此参数的值，则该事务将等待一段时间（**commit_delay**的值），否则该事务则直接写入WAL。

参数类型：USERSET

取值范围：整型， 0～1000

默认值：5

enable_xlog_group_insert

参数说明：控制WAL日志是否启动group的插入方式。仅华为鲲鹏架构支持。

参数类型：SIGHUP

取值范围：布尔型

默认值：on

11.7.2 检查点

checkpoint_segments

参数说明：设置**checkpoint_timeout**周期内所保留的最少WAL日志段文件数量。每个日志文件大小为16MB。

参数类型：SIGHUP

取值范围：整型，最小值1

默认值：64

须知

提升此参数可加快大数据的导入速度，但需要结合**checkpoint_timeout**、**shared_buffers**这两个参数统一考虑。这个参数同时影响WAL日志段文件复用数量，通常情况下pg_xlog文件夹下最大的复用文件个数为2倍的checkpoint_segments个，复用的文件被改名为后续即将使用的WAL日志段文件，不会被真正删除。

checkpoint_timeout

参数说明：设置自动WAL检查点之间的最长时间。

参数类型：SIGHUP

取值范围：整型， 30～3600（秒）

默认值：15min

须知

在提升**checkpoint_segments**以加快大数据导入的场景也需将此参数调大，同时这两个参数提升会加大**shared_buffers**的负担，需要综合考虑。

checkpoint_completion_target

参数说明：指定检查点完成的目标。

参数类型：SIGHUP

取值范围：0.0～1.0，其中默认值0.5表示每个checkpoint需要在checkpoints间隔时间的50%内完成。

默认值：0.5

checkpoint_warning

参数说明：如果由于填充检查点段文件导致检查点发生的时间间隔接近这个参数表示的秒数，就向服务器日志发送一个建议增加**checkpoint_segments**值的消息。

参数类型：SIGHUP

取值范围：整型（秒），其中0表示关闭警告。

默认值：5min

推荐值：5min

checkpoint_wait_timeout

参数说明：设置请求检查点等待checkpointer线程启动的最长时间。

参数类型：SIGHUP

取值范围：整型， 2～3600（秒）

默认值：1min

11.7.3 归档

archive_mode

参数说明：表示是否进行归档操作。

参数类型：SIGHUP

取值范围: 布尔值

- on表示进行归档。
- off表示不进行归档。

默认值：off

须知

当**wal_level**设置成minimal时，archive_mode参数无法使用。

archive_command

参数说明：由管理员设置的用于归档WAL日志的命令，建议归档路径为绝对路径。

参数类型：SIGHUP

取值范围：字符串

默认值：(disabled)

须知

- 字符串中任何%p都被要归档的文件的绝对路径代替，而任何%f都只被该文件名代替（相对路径都相对于数据目录的）。如果需要在命令里嵌入%字符就必须双写%。
- 这个命令当且仅当成功的时候才返回零。示例如下：
```
archive_command = 'cp --remove-destination %p /mnt/server/archivedir/%f'
archive_command = 'copy %p /mnt/server/archivedir/%f'  # Windows
```
- --remove-destination选项作用为：拷贝前如果目标文件已存在，会先删除已存在的目标文件，然后执行拷贝操作。

archive_timeout

参数说明：表示归档周期。

参数类型：SIGHUP

取值范围：整型，0 ~ INT_MAX，单位为秒。其中0表示禁用该功能。

默认值：0

须知

- 超过该参数设定的时间时强制切换WAL段。
- 由于强制切换而提早关闭的归档文件仍然与完整的归档文件长度相同。因此，将archive_timeout设为很小的值将导致占用巨大的归档存储空间，建议将archive_timeout设置为60秒。

11.8 双机复制

11.8.1 发送端服务器

wal_keep_segments

参数说明：Xlog日志文件段数量。设置“pg_xlog”目录下保留事务日志文件的最小数目，备机通过获取主机的日志进行流复制。

参数类型：SIGHUP

取值范围：整型，2~INT_MAX

默认值：65

设置建议：

- 当服务器开启日志归档或者从检查点恢复时，保留的日志文件数量可能大于wal_keep_segments设定的值。
- 如果此参数设置过小，则在备机请求事务日志时，此事务日志可能已经被产生的新事务日志覆盖，导致请求失败，主备关系断开。
- 当双机为异步传输时，以COPY方式连续导入4G以上数据需要增大wal_keep_segments配置。以T6000单板为例，如果导入数据量为50G，建议调整参数为1000。您可以在导入完成并且日志同步正常后，动态恢复此参数设置。

max_replication_slots

参数说明：设置主机端的日志复制slot个数。

参数类型：POSTMASTER

取值范围：整型，0~262143

默认值：8

物理流复制槽提供了一种自动化的方法来确保主DN在所有备DN或从备DN收到xlog之前，xlog不会被移除。也就是说物理流复制槽用于支撑集群HA。集群所需要的物理流复制槽数为：一组DN中，备加从备的和与主DN之间的比例。例如，假设集群的DN高可用方案为1主、1备、1从备，则所需物理流复制槽数为2。

关于逻辑复制槽数，请按如下规则考虑。

- 一个逻辑复制槽只能解码一个Database的修改，如果需要解码多个Database，则需要创建多个逻辑复制槽。
- 如果需要多路逻辑复制同步给多个目标数据库，在源端数据库需要创建多个逻辑复制槽，每个逻辑复制槽对应一条逻辑复制链路。

max_build_io_limit

参数说明：用于限制主机在提供备机重建（build）会话时，一秒时间内所允许磁盘读取的数据流量。

参数类型：SIGHUP

取值范围：整型，0～1048576，单位为KB。

默认值：0，表示主机对备机build无IO流控限制。

设置建议：可参考磁盘带宽和作业模型。无限制时或无作业干扰时，全量build在性能良好的磁盘（如SSD盘）下占磁盘带宽比例较小，磁盘IO未达到瓶颈，对业务性能影

响较小，不需要设置阈值限制。在普通10000RPM转速的SAS盘下，如果build过程中，发现业务性能明显下降，可对该参数进行设置，当前推荐设置为20MB。

此设置将直接对build的进行速度和完成时间产生影响，不建议设置过低（10MB以下不建议）。在业务低峰时，建议及时取消限制，恢复build的正常速度。

说明

- 该参数可在业务高峰期或主机磁盘IO压力较大场景时，通过限制备机build的流速阈值以减少对主机业务的影响。待业务高峰期过后，可取消限制或重新设置流速阈值。
- 具体业务场景以及磁盘性能状况，建议选择合适的阈值。

11.8.2 主服务器

enable_mix_replication

参数说明：控制主备、主从之间WAL日志及数据复制的方式。

参数类型：INTERNAL（固定参数，用户无法修改此参数，只能查看）

取值范围：布尔型

- on表示打开WAL日志、数据页混合复制模式。
- off表示关闭WAL日志、数据页混合复制模式。

默认值：off

须知

此参数属于内部参数，目前不允许正常业务场景下改变其值，即关闭WAL日志、数据页混合复制模式。

vacuum_defer_cleanup_age

参数说明：指定VACUUM使用的事务数，VACUUM会延迟清除无效的行存表记录，延迟的事务个数通过vacuum_defer_cleanup_age进行设置。即VACUUM和VACUUM FULL操作不会立即清理刚刚被删除元组。

参数类型：SIGHUP

取值范围：整型，0～1000000，值为0表示不延迟。

默认值：0

data_replicate_buffer_size

参数说明：发送端与接收端传递数据页时，队列占用内存的大小。此参数会影响主备之间复制的缓冲大小。

参数类型：POSTMASTER

取值范围：整型，4~1023，单位为MB。

默认值：128MB

enable_data_replicate

参数说明：当数据库在数据导入行存表时，主机与备机的数据同步方式可以进行选择。

参数类型：USERSET

取值范围：布尔型

- on表示导入数据行存表时主备数据采用数据页的方式进行同步。当replication_type参数为1时，不允许设置为on。
- off表示导入数据行存表时主备数据采用日志（Xlog）方式进行同步。

默认值：on

enable_incremental_catchup

参数说明：控制主备之间数据追赶（catchup）的方式。

参数类型：SIGHUP

取值范围：布尔类型

- on表示备机catchup时用增量catchup方式，即从从备本地数据文件扫描获得主备差异数据文件列表，进行主备之间的catchup。
- off表示备机catchup时用全量catchup方式，即从主机本地所有数据文件扫描获得主备差异数据文件列表，进行主备之间的catchup。

默认值：on

wait_dummy_time

参数说明：同时控制增量数据追赶（catchup）时，集群主备从按顺序启动时等待从备启动的最长时间以及等待从备发回扫描列表的最长时间。

参数类型：SIGHUP

取值范围：整型，范围1~INT_MAX，单位为秒。

默认值：300s

注意

单位只能设置为秒。

11.9 查询规划

11.9.1 优化器方法配置

这些配置参数提供了影响查询优化器选择查询规划的原始方法。如果优化器为特定的查询选择的缺省规划并不是最优的，可以通过使用这些配置参数强制优化器选择一个不同的规划来临时解决这个问题。更好的方法包括调节优化器开销常量、手动运行

ANALYZE、增加配置参数**default_statistics_target**的值、增加使用ALTER TABLE SET STATISTICS为指定列增加收集的统计信息。

enable_bitmapscan

参数说明：控制优化器对位图扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_hashagg

参数说明：控制优化器对Hash聚集规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_hashjoin

参数说明：控制优化器对Hash连接规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_indexscan

参数说明：控制优化器对索引扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_indexonlyscan

参数说明：控制优化器对仅索引扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_material

参数说明：控制优化器对实体化的使用。消除整个实体化是不可能的，但是可以关闭这个变量以防止优化器插入实体节点。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_mergejoin

参数说明：控制优化器对融合连接规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_nestloop

参数说明：控制优化器对内表全表扫描嵌套循环连接规划类型的使用。完全消除嵌套循环连接是不可能的，但是关闭这个变量就会让优化器在存在其他方法的时候优先选择其他方法。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_index_nestloop

参数说明：控制优化器对内表参数化索引扫描嵌套循环连接规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_seqscan

参数说明：控制优化器对顺序扫描规划类型的使用。完全消除顺序扫描是不可能的，但是关闭这个变量会让优化器在存在其他方法的时候优先选择其他方法。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_sort

参数说明：控制优化器使用的排序步骤。完全消除明确的排序是不可能的，但是关闭这个变量可以让优化器在存在其他方法的时候优先选择其他方法。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_tidscan

参数说明：控制优化器对TID扫描规划类型的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_kill_query

参数说明：CASCADE模式删除用户时，会删除此用户拥有的所有对象。此参数标识是否允许在删除用户的时候，取消锁定此用户所属对象的query。

参数类型：SUSET

取值范围：布尔型

- on表示允许取消锁定。
- off表示不允许取消锁定。

默认值：off

enforce_oracle_behavior

参数说明：控制正则表达式的规则匹配模式。

参数类型：USERSET

取值范围：布尔型

- on表示正则表达式采用ORACLE格式的匹配规则。
- off表示正则表达式采用POSIX格式的匹配规则。

默认值：on

enable_stream_concurrent_update

参数说明：控制优化器在并发更新场景下对stream的使用，该参数受限于**enable_stream_operator**参数。

参数类型：USERSET

取值范围：布尔型

- on表示允许优化器对update语句生成stream计划。
- off表示优化器对update语句仅能生成非stream计划。

默认值：on

enable_stream_operator

参数说明：控制优化器对stream的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_stream_recursive

参数说明：控制是否将with-recursive关联查询下推DN分布式执行。

参数类型：USERSET

取值范围：布尔型

- on表示支持使用with-recursive关联查询下推DN分布式执行。
- off表示不支持使用with_recursive下推。

默认值：on

max_recursive_times

参数说明：控制with recursive的最大迭代次数。

参数类型：USERSET

取值范围：整型，0～INT_MAX。

默认值：200

enable_vector_engine

参数说明：控制优化器对向量化执行引擎的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_broadcast

参数说明：控制优化器对stream代价估算时对broadcast分布方式的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

enable_change_hjcost

参数说明：控制优化器在Hash Join代价估算路径选择时，是否使用将内表运行时代价排除在Hash Join节点运行时代价外的估算方式。如果使用，则有利于选择条数少，但运行代价大的表做内表。

参数类型：SUSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：off

enable_fstream

参数说明：控制优化器下发语句时对stream的使用，该参数只限定于HDFS外表使用。

该参数现在已经废弃。为了保留前向兼容，可以设置成功，但是实际不起任何作用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。

- off表示不使用。

默认值：off

best_agg_plan

参数说明：对于stream下的Agg操作，优化器会生成三种计划：

1. hashagg+gather(redistribute)+hashagg。
2. redistribute+hashagg(+gather)。
3. hashagg+redistribute+hashagg(+gather)。

本参数用于控制优化器生成哪种hashagg的计划。

参数类型：USERSET

取值范围：0，1，2，3

- 取值为1时，强制生成第一种计划。
- 取值为2时，如果group by列可以重分布，强制生成第二种计划，否则生成第一种计划。
- 取值为3时，如果group by列可以重分布，强制生成第三种计划，否则生成第一种计划。
- 取值为0时，优化器会根据以上三种计划的估算cost选择最优的一种计划生成。

默认值：0

agg_redistribute_enhancement

参数说明：当进行Agg操作时，如果包含多个group by列且均不为分布列，进行重分布时会选择某一group by列进行重分布。本参数控制选择重分布列的策略。

参数类型：USERSET

取值范围：布尔型

- on表示会选择估算distinct值最多的一个可重分布列作为重分布列。
- off表示会选择第一个可重分布列为重分布列。

默认值：off

enable_valuepartition_pruning

参数说明：是否对DFS分区表进行静态/动态优化。

参数类型：USERSET

取值范围：布尔型

- on表示对DFS分区表进行静态/动态优化。
- off表示不对DFS分区表进行静态/动态优化。

默认值：on

expected_computing_nodegroup

参数说明：标识选定的计算Node Group模式或目标计算Node Group。Node Group目前为内部用机制，用户无需设置。

共4种计算Node Group模式，用于关联操作和聚集操作时选定计算Node Group。在每一种模式中，优化器有针对性地选定几个候选计算Node Group，然后根据代价，从中为当前算子挑选更合适的计算Node Group。

参数类型：USERSET

取值范围：字符串

- optimal：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前用户具有COMPUTE权限的所有Node Group包含的所有DN构成的Node Group。
- query：候选计算Node Group列表包含算子操作对象所在的Node Group和由当前查询涉及的所有基表所在Node Group包含的所有DN构成的Node Group。
- Node Group名（**enable_nodegroup_debug**被设置为off）：候选计算Node Group列表包含算子操作对象所在的Node Group和该指定的Node Group。
- Node Group名（**enable_nodegroup_debug**被设置为on）：候选计算Node Group为指定的Node Group。

默认值：query

enable_nodegroup_debug

参数说明：控制优化器在多Node Group环境下，是否使用强制弹性计算。Node Group目前为内部用机制，用户无需设置。

该参数只在**expected_computing_nodegroup**被设置为具体Node Group时生效。

参数类型：USERSET

取值范围：布尔型

- on表示强制将计算弹性到expected_computing_nodegroup所指定的Node Group进行计算。
- off表示不强制使用某个Node Group进行计算。

默认值：off

stream_multiple

参数说明：设置优化器计算Stream算子的开销时的加权。

在原代价模型的基础上，最终Stream代价将被乘以此加权参数。

参数类型：USERSET

取值范围：浮点型，0～DBL_MAX。

默认值：1

须知

此参数仅对Redistribute和Broadcast类型的Stream有效。

qrw_inlist2join_optmode

参数说明：控制是否使用inlist-to-join查询重写。

参数类型：USERSET

取值范围：字符串

- disable：关闭inlist2join查询重写。
- cost_base：基于代价的inlist2join查询重写。
- rule_base：基于规则的inlist2join查询重写，即强制使用inlist2join查询重写。
- 任意正整数：inlist2join查询重写阈值，即list内元素个数大于该阈值，进行inlist2join查询重写。

默认值：cost_base

skew_option

参数说明：控制是否使用优化策略。

参数类型：USERSET

取值范围：字符串

- off：关闭策略。
- normal：采用激进策略。对于不确定是否出现倾斜的场景，认为存在倾斜，并进行相应优化。
- lazy：采用保守策略。对于不确定是否出现倾斜场景，认为不存在倾斜，不进行优化。

默认值：normal

11.9.2 优化器开销常量

介绍优化器开销常量。这里描述的开销可以按照任意标准度量。只关心其相对值，因此以相同的系数缩放它们将不会对优化器的选择产生任何影响。缺省时，它们以抓取顺序页的开销为基本单位。也就是说将seq_page_cost设为1.0，同时其他开销参数以它为基准设置。也可以使用其他基准，比如以毫秒计的实际执行时间。

seq_page_cost

参数说明：设置优化器计算一次顺序磁盘页面抓取的开销。

参数类型：USERSET

取值范围：浮点型，0～DBL_MAX。

默认值：1

random_page_cost

参数说明：设置优化器计算一次非顺序抓取磁盘页面的开销。

参数类型：USERSET

取值范围：浮点型，0～DBL_MAX。

默认值：4

说明

- 虽然服务器允许将random_page_cost设置的比seq_page_cost小，但是物理上实际不受影响。如果所有数据库都位于随机访问内存中时，两者设置为相等很合理。因为在此种情况下，非顺序抓取页并没有副作用。同样，在缓冲率很高的数据库上，应该相对于CPU参数同时降低这两个值，因为获取内存中的页要比通常情况下开销小很多。
- 对于特别表空间中的表和索引，可以通过设置同名的表空间的参数来覆盖这个值。
- 相对于seq_page_cost，减少这个值将导致系统更倾向于使用索引扫描，而增加这个值使得索引扫描开销比较高。可以通过同时增加或减少这两个值来调整磁盘I/O相对于CPU的开销。

cpu_tuple_cost

参数说明：设置优化器计算在一次查询中处理每一行数据的开销。

参数类型：USERSET

取值范围：浮点型，0～DBL_MAX。

默认值：0.01

cpu_index_tuple_cost

参数说明：设置优化器计算在一次索引扫描中处理每条索引的开销。

参数类型：USERSET

取值范围：浮点型，0～DBL_MAX。

默认值：0.005

cpu_operator_cost

参数说明：设置优化器计算一次查询中执行一个操作符或函数的开销。

参数类型：USERSET

取值范围：浮点型，0～DBL_MAX。

默认值：0.0025

effective_cache_size

参数说明：设置优化器在一次单一的查询中可用的磁盘缓冲区的有效大小。

设置这个参数，还要考虑GaussDB(DWS)的共享缓冲区以及内核的磁盘缓冲区。另外，还要考虑预计的在不同表之间的并发查询数目，因为它们将共享可用的空间。

这个参数对GaussDB(DWS)分配的共享内存大小没有影响，它也不会使用内核磁盘缓冲，它只用于估算。数值是用磁盘页来计算的，通常每个页面是8192字节。

参数类型：USERSET

取值范围：整型，1～INT_MAX，单位为8KB。

比默认值高的数值可能会导致使用索引扫描，更低的数值可能会导致选择顺序扫描。

默认值：128MB

allocate_mem_cost

参数说明：设置优化器计算Hash Join创建Hash表开辟内存空间所需的开销，供Hash join估算不准时调优使用。

参数类型：USERSET

取值范围：浮点型，0～DBL_MAX。

默认值：0

11.9.3 基因查询优化器

介绍基因查询优化器相关的参数。基因查询优化器（GEQO）是一种启发式的查询规划算法。这个算法减少了对复杂查询规划的时间，而且生成规划的开销有时也小于正常的详尽的查询算法。

geqo

参数说明：控制基因查询优化的使用。

参数类型：USERSET

取值范围：布尔型

- on表示使用。
- off表示不使用。

默认值：on

须知

通常情况下在执行过程中不要关闭，geqo_threshold变量提供了更精细的控制GEQO的方法。

geqo_threshold

参数说明：如果执行语句的数量超过设计的FROM的项数，则会使用基因查询优化来执行查询。

参数类型：USERSET

取值范围：整型，2～INT_MAX。

默认值：12

须知

- 对于简单的查询，通常用详尽搜索方法，当涉及多个表的查询的时候，用GEQO可以更好的管理查询。
- 一个FULL OUTER JOIN构造仅作为一个FROM项。

geqo_effort

参数说明：控制GEQO在规划时间和规划质量之间的平衡。

参数类型：USERSET

取值范围：整型，1～10

默认值：5

须知

- 比默认值大的数值增加了查询规划的时间，但是也增加了选中有效查询的几率。
- geqo_effort实际上并没有直接作用，只是用于计算其他影响GEQO的变量的默认值。如有需求，可以手动设置其他相关参数。

geqo_pool_size

参数说明：控制GEQO使用池的大小，也就是基因全体中的个体数量。

参数类型：USERSET

取值范围：整型，0～INT_MAX

须知

至少是2，且有用的值一般在100到1000之间。设置为0，表示使用系统自适应方式，GaussDB(DWS)会基于geqo_effort和表的个数选取合适的值。

默认值：0

geqo_generations

参数说明：控制GEQO使用的算法的迭代次数。

参数类型：USERSET

取值范围：整型，0～INT_MAX

须知

必须至少是1，且有用的值介于100和1000之间。如果设置为0，则基于geqo_pool_size选取合适的值。

默认值：0

geqo_selection_bias

参数说明：控制GEQO的选择性偏好，即就是一个种群中的选择性压力。

参数类型：USERSET

取值范围：浮点型，1.5～2.0

默认值：2

geqo_seed

参数说明：控制GEQO使用的随机数生产器的初始化值，用来从顺序连接在一起的查询空间中查找随机路径。

参数类型：USERSET

取值范围：浮点型，0.0～1.0

须知

不同的值会改变搜索的连接路径，从而影响了所找路径的优劣。

默认值：0

11.9.4 其他优化器选项

default_statistics_target

参数说明：为没有用ALTER TABLE SET STATISTICS设置字段目标的表设置缺省统计目标。此参数设置为正数是代表统计信息的样本数量，为负数时，代表使用百分比的形式设置统计目标，负数转换为对应的百分比，即-5代表5%。采样时，会将default_statistics_target * 300作为随机抽样的大小，例如默认值为100时，会读取100* 300 个页面来完成随机抽样。

参数类型：USERSET

取值范围：整型，-100～10000。

须知

- 比默认值大的正数数值增加了ANALYZE所需的时间，但是可能会改善优化器的估计质量。
- 调整此参数可能存在性能劣化的风险，如果某个查询劣化，可以考虑
  1. 恢复默认的统计信息。
  2. 使用plan hint来调整到之前的查询计划。（详细参见**使用Plan Hint进行调优**）
- 当此guc参数设置为负数时，如果计算的采样样本数大于等于总数据量的2%，且用户表的数据量小于1600000时，ANALYZE所需时间相比guc参数为默认值的时间会有所增加。
- 当此guc参数设置为负数时，则autoanalyze不生效。

默认值：100

constraint_exclusion

参数说明：控制查询优化器使用表约束查询的优化。

参数类型：USERSET

取值范围：枚举类型

- on表示检查所有表的约束。
- off表示不检查约束。
- partition表示只检查继承的子表和UNION ALL子查询。

须知

当constraint_exclusion为on，优化器用查询条件和表的CHECK约束比较，并且在查询条件和约束冲突的时候忽略对表的扫描。

默认值：partition

说明

目前， constraint_exclusion缺省被打开，通常用来实现表分区。为所有的表打开它时，对于简单的查询强加了额外的规划，并且对简单查询没有什么好处。如果不用分区表，可以关掉它。

cursor_tuple_fraction

参数说明：优化器估计游标获取行数在总行数中的占比。

参数类型：USERSET

取值范围：浮点型，0.0～1.0。

须知

比默认值小的值与使用 “fast start” 为游标规划的值相偏离，从而使得前几行恢复的很快而抓取全部的行需要很长的时间。比默认值大的值加大了总的估计的时间。在最大的值1.0处，像正常的查询一样规划游标，只考虑总的估计时间和传送第一行的时间。

默认值：0.1

from_collapse_limit

参数说明：根据生成的FROM列表的项数来判断优化器是否将把子查询合并到上层查询，如果FROM列表项个数小于等于该参数值，优化器会将子查询合并到上层查询。

参数类型：USERSET

取值范围：整型，1～INT_MAX。

须知

比默认值小的数值将降低规划时间，但是可能生成差的执行计划。

默认值：8

join_collapse_limit

参数说明：根据得出的列表项数来判断优化器是否执行把除FULL JOINS之外的JOIN构造重写到FROM列表中。

参数类型：USERSET

取值范围：整型，1～INT_MAX。

须知

- 设置为1会避免任何JOIN重排。这样就使得查询中指定的连接顺序就是实际的连接顺序。查询优化器并不是总能选取最优的连接顺序，高级用户可以选择暂时把这个变量设置为1，然后指定它们需要的连接顺序。
- 比默认值小的数值减少规划时间但也降低了执行计划的质量。

默认值：8

plan_mode_seed

参数说明：该参数为调测参数，目前仅支持OPTIMIZE_PLAN和RANDOM_PLAN两种。其中：OPTIMIZE_PLAN表示通过动态规划算法进行代价估算的最优plan，参数值设置为0；RANDOM_PLAN表示随机生成的plan；如果设置为-1，表示用户不指定随机数的种子标识符seed值，由优化器随机生成[1, 2147483647]范围整型值的随机数，并根据随机数生成随机的执行计划；如果用户指定guc参数值为[1, 2147483647]范围的整型值，表示指定的生成随机数的种子标识符seed，优化器需要根据seed值生成随机的执行计划。

参数类型：USERSET

取值范围：整型，-1~ 2147483647

默认值：0

须知

- 当该参数设置为随机执行计划模式时，优化器会生成不同的随机执行计划，该执行计划可能不是最优计划。因此在随机计划模式下，会对查询性能产生影响，所以建议在升级、扩容、缩容等正常业务操作或运维过程中将该参数保持为默认值0。
- 当该参数不为0时，查询指定的plan hint不会生效。

enable_hdfs_predicate_pushdown

参数说明：标示是否启用谓词下推至原生数据层的功能。

参数类型：SUSET

取值范围：布尔型

- on表示启用谓词下推至原生数据层的功能。
- off表示不启用谓词下推至原生数据层的功能。

默认值：on

enable_random_datanode

参数说明：标示是否允许开启复制表DN随机查找功能，复制表在每个DN存放一份完整数据，随机选取可以缓解节点压力。

参数类型：USERSET

取值范围：布尔型

- on表示允许开启复制表DN随机查找功能。
- off表示不允许开启复制表DN随机查找功能。

默认值：on

hashagg_table_size

参数说明：用于设置执行HASH AGG操作时HASH表的大小。

参数类型：USERSET

取值范围：整型，0～INT_MAX/2。

默认值：0

enable_codegen

参数说明：标识是否允许开启代码生成优化，目前代码生成使用的是LLVM优化。

参数类型：USERSET

取值范围：布尔型

- on表示允许开启代码生成优化。
- off表示不允许开启代码生成优化。

须知

目前LLVM优化仅支持向量化执行引擎特性和SQL on Hadoop特性，在其他场景下建议关闭此参数。

默认值：on

codegen_strategy

参数说明：标识在表达式codegen化过程中所使用的代码生成优化策略。

参数类型：USERSET

取值范围：枚举类型

- partial表示当所计算表达式中即使包含部分未被codegen化的函数时，仍可借助表达式全codegen框架调用LLVM动态编译优化策略。
- pure表示当所计算表达式整体可被codegen化时，才考虑调用LLVM动态编译优化策略。

须知

在开启代码生成优化会导致查询性能下降的场景下可以设置此参数为pure，其他场景下建议不改变此参数的默认值partial。

默认值：partial

enable_codegen_print

参数说明：标识是否允许在log日志中打印所生成的LLVM IR函数。

参数类型：USERSET

取值范围：布尔型

- on表示允许在log日志中打印IR函数。
- off表示不允许在log日志中打印IR函数。

默认值：off

codegen_cost_threshold

参数说明：由于LLVM编译生成最终的可执行机器码需要一定时间，因此只有当实际执行的代价大于编译生成机器码所需要的代码和优化后的执行代价之和时，利用代码生成才有收益。codegen_cost_threshold标识代价的阈值，当执行估算代价大于该代价时，使用LLVM优化。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：10000

enable_constraint_optimization

参数说明: 标识是否允许HDFS外表使用信息约束（Informational Constraint）优化执行计划。

参数类型：SUSET

取值范围：布尔型

- on表示允许使用信息约束优化执行计划。
- off表示不允许使用信息约束优化执行计划。

默认值：on

enable_bloom_filter

参数说明: 标识是否允许使用BloomFilter优化。

参数类型：USERSET

取值范围：布尔型

- on表示允许使用BloomFilter优化。
- off表示不允许使用BloomFilter优化。

默认值：on

enable_extrapolation_stats

参数说明: 标识对于日期类型是否允许基于历史统计信息使用推理估算的逻辑。使用该逻辑对于未及时收集统计信息的表可以增大估算准确的可能性，但也存在错误推理导致估算过大的可能性，需要对于日期类型数据定期插入的场景开启此开关。

参数类型：SUSET

取值范围：布尔型

- on表示允许基于历史统计信息使用推理估算的逻辑。
- off表示不允许基于历史统计信息使用推理估算的逻辑。

默认值：off

autoanalyze

参数说明: 标识是否允许在生成计划的时候，对于没有统计信息的表进行统计信息自动收集。对于外表和临时表，不支持autoanalyze，如果需要收集统计信息，用户需手动执行analyze操作。如果在auto analyze某个表的过程中数据库发生异常，当数据库正常运行之后再执行语句有可能仍提示需要收集此表的统计信息。此时需要用户对该表手动执行一次analyze操作，以同步统计信息数据。

参数类型：SUSET

取值范围：布尔型

- on表示允许自动进行统计信息收集。
- off表示不允许自动进行统计信息收集。

默认值：off

query_dop

参数说明: 用户自定义的查询并行度。

参数类型：USERSET

取值范围：整型，-64-64

[1,64]：打开固定SMP功能，系统会使用固定并行度。

0：打开SMP自适应功能，系统会根据资源情况和计划特征动态为每个查询选取[1,8]之间（x86平台），[1,64]之间（鲲鹏平台）的最优的并行度。

[-64,-1]：打开SMP自适应功能，并限制自适应选取的最大并行度。

说明

- 在开启并行查询后，请保证系统CPU、内存、网络、I/O等资源充足，以达到良好效果。
- 为了避免用户设置不合理的过大值造成性能劣化，系统会计算出该DN可用最大CPU核数，并以此来作为query_dop的上限。如果用户设置query_dop超过4并且同时超过该上限，那么系统会重置query_dop为该上限值。

默认值：1

enable_analyze_check

参数说明: 标识是否允许在生成计划的时候，对于在pg_class中显示reltuples和relpages均为0的表，检查该表是否曾进行过统计信息收集。

参数类型：SUSET

取值范围：布尔型

- on表示允许检查。
- off表示不允许检查。

默认值：on

enable_sonic_hashagg

参数说明：标识是否依据规则约束使用基于面向列的hash表设计的Hash Agg算子。

参数类型：USERSET

取值范围：布尔型

- on表示在满足约束条件时使用基于面向列的hash表设计的Hash Agg算子。
- off表示不使用面向列的hash表设计的Hash Agg算子。

说明

- 在开启enable_sonic_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，查询对应的Hash Agg算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景(**enable_codegen**打开后获得较大性能提升)，对应的算子查询性能可能会出现劣化。
- 开启enable_sonic_hashagg，且查询达到约束条件使用基于面向列的hash表设计的Hash Agg算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Aggregation”，而未达到该约束条件时，算子名称将显示为“Hash Aggregation”，Explain详解请参见**SQL执行计划详解**。

默认值：on

enable_sonic_hashjoin

参数说明：标识是否依据规则约束使用基于面向列的hash表设计的Hash Join算子。

参数类型：USERSET

取值范围：布尔型

- on表示在满足约束条件时使用基于面向列的hash表设计的Hash Join算子。
- off表示不使用面向列的hash表设计的Hash Join算子。

说明

- 当前开关仅适用于Inner Join的场景。
- 在开启enable_sonic_hashjoin，查询对应的Hash Inner算子内存使用通常可获得精简。但对于代码生成技术可获得显著性能提升的场景，对应的算子查询性能可能会出现劣化。
- 开启enable_sonic_hashjoin，且查询达到约束条件使用基于面向列的hash表设计的Hash Join算子时，在Explain Analyze/Performance的执行计划和执行信息中，算子显示为“Sonic Hash Join”，而未达到该约束条件时，算子名称将显示为“Hash Join”，Explain详解请参见**SQL执行计划详解**。

默认值：on

enable_sonic_optspill

参数说明：标识是否优化sonic场景下HashJoin或者HashAgg的下盘文件个数。仅在enable_sonic_hashjoin或者 enable_sonic_hashagg开启情况下生效。

参数类型：USERSET

取值范围：布尔型

- on表示开启下盘文件数优化。
- off表示关闭下盘文件数优化。

说明

满足sonic条件下的HashJoin或者HashAgg算子，在关闭此参数（设置为off）时每列会产生1个下盘文件。开启此参数（设置为on）时如果不同列数据类型相似，只会有1个下盘文件（最多5个文件）。

默认值：on

plan_cache_mode

参数说明：标识在prepare语句中，选择生成执行计划的策略。

参数类型：USERSET

取值范围：枚举类型

- auto表示按照默认的方式选择custom plan或者generic plan。
- force_generic_plan表示强制走generic plan。
- force_custom_plan表示强制走custom plan。

说明

- 此参数只对prepare语句生效，一般用在prepare语句中参数化字段存在比较严重的数据倾斜的场景下。
- custom plan是指对于preapre语句，在执行execute的时候，把execute语句中的参数嵌套到语句之后生成的计划。custom plan会根据execute语句中具体的参数生成计划，这种方案的优点是每次都按照具体的参数生成优选计划，执行性能比较好；缺点是每次执行前都需要重新生成计划，存在大量的重复的优化器开销。
- generic plan是指对于preapre语句生成计划，该计划策略会在执行execute语句的时候把参数bind到plan中，然后执行计划。这种方案的优点是每次执行可以省去重复的优化器开销；缺点是当bind参数字段上数据存在倾斜时该计划可能不是最优的，部分bind参数场景下执行性能较差。

默认值：auto

hashjoin_spill_strategy

参数说明：选择hashjoin下盘策略。(该参数仅8.0.0.9版本支持)

参数类型：USERSET

取值范围：整型，范围0～4

- 0：当内表较大，并且多次下盘无法分开时，尝试外表是否可以放到数据库可用内存。如果内外表均很大，执行NestLoop。
- 1：当内表较大，并且多次下盘无法分开时，尝试外表是否可以放到数据库可用内存。如果内外表均很大，强制执行HashJoin。
- 2：当内表较大，并且多次下盘无法分开时，强制执行HashJoin。
- 3：当内表较大，并且多次下盘无法分开时，尝试外表是否可以放到数据库可用内存。如果内外表均很大，则报错。
- 4：当内表较大，并且多次下盘无法分开时，则报错。

说明

- 此参数只对向量化HashJoin生效。
- 对于数据distinct值很少且数据量很大场景，可能出现无法下盘导致使用内存过大产生内存不受控的问题。 取值0时通过尝试内外表交换或者Nestloop可以避免出现此类内存不受控问题。 执行Nestloop可能造成某些场景性能劣化。

默认值：0

11.10 错误报告和日志

11.10.1 记录日志的位置

log_truncate_on_rotation

参数说明：logging_collector设置为on时，log_truncate_on_rotation设置日志消息的写入方式。

参数类型：SIGHUP

取值范围： 布尔型

- on表示GaussDB(DWS)以覆盖写入的方式写服务器日志消息。
- off表示GaussDB(DWS)将日志消息附加到同名的现有日志文件上。

默认值：off

说明

示例：

假设日志需要保留7天，每天生成一个日志文件，日志文件名设置为server_log.Mon、server_log.Tue等。第二周的周二生成的日志消息会覆盖写入到server_log.Tue。设置方法：将log_filename设置为server_log.%a ，log_truncate_on_rotation设置为on，log_rotation_age设置为1440，即日志有效时间为1天。

log_rotation_age

参数说明：logging_collector设置为on时，log_rotation_age决定创建一个新日志文件的时间间隔。当现在的时间减去上次创建一个服务器日志的时间超过了log_rotation_age的值时，将生成一个新的日志文件。

参数类型：SIGHUP

取值范围：整型，0 ~ 24d，单位为min，h，d。其中0表示关闭基于时间的新日志文件的创建。

默认值：1d

log_rotation_size

参数说明：logging_collector设置为on时，log_rotation_size决定服务器日志文件的最大容量。当日志消息的总量超过日志文件容量时，服务器将生成一个新的日志文件。

参数类型：SIGHUP

取值范围：整型，INT_MAX / 1024，单位为KB。

0表示关闭基于容量的新日志文件的创建。

默认值：20MB

event_source

参数说明：log_destination设置为eventlog时，event_source设置在日志中GaussDB(DWS)日志消息的标识。

参数类型：POSTMASTER

取值范围：字符串

默认值：PostgreSQL

11.10.2 记录日志的时间

client_min_messages

参数说明：控制发送到客户端的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，发送给客户端的消息就越少。

参数类型：USERSET

须知

当client_min_messages和**log_min_messages**取相同值时，其值所代表的级别不同。

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error。参数的详细信息请参见**表11-3**。

默认值：notice

log_min_messages

参数说明：控制写到服务器日志文件中的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，服务器运行日志中记录的消息就越少。

参数类型：SUSET

须知

当**client_min_messages**和log_min_messages取相同值log时所代表的消息级别不同。

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表11-3**。

默认值：warning

log_min_error_statement

参数说明：控制在服务器日志中记录错误的SQL语句。

参数类型：SUSET

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表11-3**。

说明

- 设置为error ，表示导致错误、日志消息、致命错误、panic的语句都将被记录。
- 设置为panic，表示关闭此特性。

默认值：error

log_min_duration_statement

参数说明：当某条语句的持续时间大于或者等于特定的毫秒数时，log_min_duration_statement参数用于控制记录每条完成语句的持续时间。

设置log_min_duration_statement可以很方便地跟踪需要优化的查询语句。对于使用扩展查询协议的客户端，语法分析、绑定、执行每一步所花时间被独立记录。

参数类型：SUSET

须知

当此选项与log_statement同时使用时，已经被log_statement记录的语句文本不会被重复记录。在没有使用syslog情况下，推荐使用log_line_prefix记录PID或会话ID，方便将当前语句消息连接到最后的持续时间消息。

取值范围：整型，-1 ~ INT_MAX，单位为毫秒。

- 设置为250，所有运行时间不短于250ms的SQL语句都会被记录。
- 设置为0，输出所有语句的持续时间。
- 设置为-1，关闭此功能。

默认值：30min

backtrace_min_messages

参数说明：控制当产生该设置参数级别相等或更高级别的信息时，会打印函数的堆栈信息到服务器日志文件中。

参数类型：SUSET

须知

该参数作为客户现场问题定位手段使用，且由于频繁的打印函数栈会对系统的开销及稳定性有一定的影响，因此如果需要进行问题定位时，建议避免将backtrace_min_messages的值设置为fatal及panic以外的级别。

取值范围：枚举类型

有效值有debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见**表11-3**。

默认值：panic

表11-3解释GaussDB(DWS)中使用的消息安全级别。当日志输出到syslog或者eventlog时，GaussDB(DWS)进行如表中的转换。

表 11-3 信息严重程度分类

信息严重程度类型	详细说明	系统日志	事件日志
debug[1-5]	报告详细调试信息。	DEBUG	INFORMATION
log	报告对数据库管理员有用的信息， 比如检查点操作统计信息。	INFO	INFORMATION
info	报告用户可能需求的信息， 比如在VACUUM VERBOSE过程中的信息。	INFO	INFORMATION
notice	报告可能对用户有帮助的信息， 比如， 长标识符的截断， 作为主键一部分创建的索引等。	NOTICE	INFORMATION
warning	报告警告信息， 比如在事务块范围之外的COMMIT。	NOTICE	WARNING
error	报告导致当前命令退出的错误。	WARNING	ERROR
fatal	报告导致当前会话终止的原因。	ERR	ERROR
panic	报告导致整个数据库被关闭的原因。	CRIT	ERROR

plog_merge_age

参数说明：该参数用于控制性能日志数据输出的周期。

参数类型：SUSET

须知

该参数以毫秒为单位的，建议在使用过程中设置值为1000的整数倍，即设置值以秒为最小单位。该参数所控制的性能日志文件以prf为扩展名，文件放置在$GAUSSLOG/gs_profile/<node_name> 目录下面，其中node_name是由postgres.conf文件中的pgxc_node_name的值，不建议外部使用该参数。

取值范围：0~INT_MAX，单位为毫秒（ ms ）。

- 当设置为0时，当前会话不再输出性能日志数据。
- 当设置为非0时，当前会话按照指定的时间周期进行输出性能日志数据。该参数设置得越小，输出的日志数据越多，对性能的负面影响越大。

默认值：3s

11.10.3 记录日志的内容

debug_print_parse

参数说明：用于控制打印解析树结果。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启打印结果的功能。
- off表示关闭打印结果的功能。

默认值：off

debug_print_rewritten

参数说明：用于控制打印查询重写结果。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启打印结果的功能。
- off表示关闭打印结果的功能。

默认值：off

debug_print_plan

参数说明：用于控制打印查询执行结果。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启打印结果的功能。
- off表示关闭打印结果的功能。

默认值：off

须知

- 只有当日志的级别为log及以上时，debug_print_parse、debug_print_rewritten和debug_print_plan的调试信息才会输出。当这些选项打开时，调试信息只会记录在服务器的日志中，而不会输出到客户端的日志中。通过设置**client_min_messages**和**log_min_messages**参数可以改变日志级别。
- 在打开debug_print_plan开关的情况下需尽量避免调用gs_encrypt_aes128及gs_decrypt_aes128函数，避免敏感参数信息在日志中泄露的风险。同时建议用户在打开debug_print_plan开关生成的日志中对gs_encrypt_aes128及gs_decrypt_aes128函数的参数信息进行过滤后再提供给外部维护人员定位，日志使用完成后请及时删除。

debug_pretty_print

参数说明：设置此选项对debug_print_parse、debug_print_rewritten和debug_print_plan产生的日志进行缩进，会生成易读但比设置为off时更长的输出格式。

参数类型：USERSET

取值范围：布尔型

- on表示进行缩进。
- off表示不进行缩进。

默认值：on

log_checkpoints

参数说明：控制在服务器日志中记录检查点和重启点的信息。打开此参数时，服务器日志消息包含涉及检查点和重启点的统计量，其中包含需要写的缓存区的数量及写入所花费的时间等。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开此参数时，服务器日志消息包含涉及检查点和重启点的统计量。
- off表示关闭此参数时，服务器日志消息包含不涉及检查点和重启点的统计量。

默认值：off

log_connections

参数说明：控制记录客户端的连接请求信息。

参数类型：BACKEND

须知

- 会话连接参数，不建议用户设置。
- 有些客户端程序（例如gsql），在判断是否需要口令的时候会尝试连接两次，因此日志消息中重复的“connection receive”（收到连接请求）不一定是问题。

取值范围：布尔型

- on表示记录信息。
- off表示不记录信息。

默认值：off

log_disconnections

参数说明：控制记录客户端结束连接信息。

参数类型：BACKEND

取值范围：布尔型

- on表示记录信息。
- off表示不记录信息。

默认值：off

说明

会话连接参数，不建议用户设置。

log_duration

参数说明：控制记录每个已完成SQL语句的执行时间。对使用扩展查询协议的客户端、会记录语法分析、绑定和执行每一步所花费的时间。

参数类型：SUSET

取值范围：布尔型

- 设置为off ，该选项与**log_min_duration_statement**的不同之处在于log_min_duration_statement强制记录查询文本。
- 设置为on并且log_min_duration_statement大于零，记录所有持续时间，但是仅记录超过阈值的语句。这可用于在高负载情况下搜集统计信息。

默认值：on

log_error_verbosity

参数说明：控制服务器日志中每条记录的消息写入的详细度。

参数类型：SUSET

取值范围：枚举类型

- terse输出不包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录。
- verbose输出包括SQLSTATE错误代码 、源代码文件名、函数名及产生错误所在的行号。
- default输出包括DETAIL、HINT、QUERY及CONTEXT错误信息的记录，不包括SQLSTATE错误代码 、源代码文件名、函数名及产生错误所在的行号。

默认值：default

log_hostname

参数说明：默认状态下，连接消息日志只显示正在连接主机的IP地址。打开此选项同时可以记录主机名。由于解析主机名可能需要一定的时间，可能影响数据库的性能。

参数类型：SIGHUP

取值范围：布尔型

- on表示可以同时记录主机名。
- off表示不可以同时记录主机名。

默认值：off

log_lock_waits

参数说明：当一个会话的等待获得一个锁的时间超过**deadlock_timeout**的值时，此选项控制在数据库日志中记录此消息。这对于决定锁等待是否会产生一个坏的行为是非常有用的。

参数类型：SUSET

取值范围：布尔型

- on表示记录此信息。
- off表示不记录此信息。

默认值：off

log_statement

参数说明：控制记录SQL语句。对于使用扩展查询协议的客户端，记录接收到执行消息的事件和绑定参数的值（内置单引号要双写）。

参数类型：SUSET

须知

即使log_statement设置为all，包含简单语法错误的语句也不会被记录，因为仅在完成基本的语法分析并确定了语句类型之后才记录日志。在使用扩展查询协议的情况下，在执行阶段之前（语法分析或规划阶段）同样不会记录。将log_min_error_statement设为ERROR或更低才能记录这些语句。

取值范围：枚举型

- none表示不记录语句。
- ddl表示记录所有的数据定义语句，比如CREATE、ALTER和DROP语句。
- mod表示记录所有DDL语句，还包括数据修改语句INSERT、UPDATE、DELETE、TRUNCATE和COPY FROM 。
- all表示记录所有语句，PREPARE、EXECUTE和EXPLAIN ANALYZE语句也同样被记录。

默认值：none

log_temp_files

参数说明：控制记录临时文件的删除信息。临时文件可以用来排序、哈希及临时查询结果。当一个临时文件被删除时，将会产生一条日志消息。

参数类型：SUSET

取值范围：整型，-1~INT_MAX，单位为KB

- 正整数表示只记录比log_temp_files设定值大的临时文件的删除信息。
- 值0 表示记录所有的临时文件的删除信息。
- 值-1 表示不记录任何临时文件的删除信息。

默认值：-1

log_timezone

参数说明：设置服务器写日志文件时使用的时区。与TimeZone不同，这个值是数据库范围的，针对所有连接到本数据库的会话生效。

参数类型：SIGHUP

取值范围：字符串

默认值：PRC

说明

gs_initdb进行相应系统环境设置时会对默认值进行修改。

logging_module

参数说明：用于设置或者显示模块日志在服务端的可输出性。该参数属于会话级参数，不建议通过gs_guc工具来设置。

参数类型：USERSET

取值范围：字符串

默认值：所有模块日志在服务端是不输出的，可由SHOW logging_module查看。

设置方法：首先，可以通过SHOW logging_module来查看哪些模块是支持可控制的。例如，查询输出结果为：

```
postgres=# show logging_module;
logging_module
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------
ALL,on(),off(DFS,GUC,HDFS,ORC,SLRU,MEM_CTL,AUTOVAC,CACHE,ADIO,SSL,GDS,TBLSPC,WLM,OBS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,OPT_SETOP,OPT_SKEW,UDF,COOP_ANALYZE,WLMCP,ACCELERATE,PLANHINT,PARQUET,CARBONDATA,SNAPSHOT,XACT,HANDLE,CLOG,EC,REMOTE,CN_RETRY,PLSQL,TEXTSEARCH,SEQ,INSTR,COMM_IPC,COMM_PARAM)
(1 row)
```

支持可控制的模块使用大写来标识，特殊标识ALL用于对所有模块日志进行设置。可以使用on/off来控制模块日志的输出。设置SSL模块日志为可输出，使用如下命令：

```
postgres=# set logging_module='on(SSL)';
SET
postgres=# show
logging_module;
      logging_module
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------
ALL,on(SSL),off(DFS,GUC,HDFS,ORC,CARBONDATA,SLRU,MEM_CTL,AUTOVAC,CACHE,ADIO,GDS,TBLSPC,WLM,OBS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,OPT_SETOP,OPT_SKEW,UDF,COOP_ANALYZE,WLMCP,ACCELERATE,PLANHINT,SNAPSHOT,XACT,HANDLE,CLOG,EC,REMOTE,CN_RETRY,PLSQL,TEXTSEARCH,SEQ)
(1 row)
```

可以看到模块SSL的日志输出被打开。

ALL标识是相当于一个快捷操作，即对所有模块的日志可输出进行开启或关闭。

```
postgres=# set logging_module='off(ALL)';
SET
postgres=# show
logging_module;
```

```
          logging_module
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------
ALL,on(),off(DFS,GUC,HDFS,ORC,CARBONDATA,SLRU,MEM_CTL,AUTOVAC,CACHE,ADIO,SSL,GDS,TBLSPC,WL
M,OBS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,O
PT_SETOP,OPT_SKEW,UDF,COOP_ANALYZE,WLMCP,ACCELERATE,PLANHINT,SNAPSHOT,XACT,HANDLE,CLO
G,EC,REMOTE,CN_RETRY,PLSQL,TEXTSEARCH,SEQ)
(1 row)

postgres=# set logging_module='on(ALL)';
SET
postgres=# show
logging_module;
         logging_module
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------
ALL,on(DFS,GUC,HDFS,ORC,CARBONDATA,SLRU,MEM_CTL,AUTOVAC,CACHE,ADIO,SSL,GDS,TBLSPC,WLM,O
BS,EXECUTOR,VEC_EXECUTOR,STREAM,LLVM,OPT,OPT_REWRITE,OPT_JOIN,OPT_AGG,OPT_SUBPLAN,OPT_S
ETOP,OPT_SKEW,UDF,COOP_ANALYZE,WLMCP,ACCELERATE,PLANHINT,SNAPSHOT,XACT,HANDLE,CLOG,EC,
REMOTE,CN_RETRY,PLSQL,TEXTSEARCH,SEQ),off()
(1 row)
```

依赖关系：该参数依赖于**log_min_messages**参数的设置。

enable_unshipping_log

参数说明：用于控制是否打印语句不下推的日志，主要用于帮助用户定位不下推语句可能导致的性能问题。

参数类型：SUSET

取值范围：布尔型

- on表示打印日志。
- off表示不打印日志。

默认值：on

11.11 告警检测

在集群运行的过程中，会对数据库中的错误场景进行检测，便于用户及早感知到数据库集群的错误。

enable_alarm

参数说明：允许打开告警检测线程，检测数据库中可能的错误场景。

参数类型：POSTMASTER

取值范围：布尔型

- on表示允许打开告警检测线程。
- off表示不允许打开告警检测线程。

默认值：on

connection_alarm_rate

参数说明：允许和数据库连接的最大并发连接数的比率限制。数据库连接的最大并发连接数为max_connections* connection_alarm_rate。

参数类型：SIGHUP

取值范围：浮点型，0.0~1.0

默认值：0.9

alarm_report_interval

参数说明：指定告警上报的时间间隔。

参数类型：SIGHUP

取值范围：非负整型，单位为秒。

默认值：10

11.12 运行时统计

11.12.1 查询和索引统计收集器

查询和索引统计收集器负责收集数据库系统运行中的统计数据，如在一个表和索引上进行了多少次插入与更新操作、磁盘块的数量和元组的数量、每个表上最近一次执行清理和分析操作的时间等。可以通过查询系统视图pg_stats和pg_statistic查看统计数据。下面的参数设置服务器范围内的统计收集特性。

track_activities

参数说明：控制收集每个会话中当前正在执行命令的统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

默认值：on

track_counts

参数说明：控制收集数据库活动的统计数据。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

说明

在AutoVacuum自动清理进程中选择清理的数据库时，需要数据库的统计数据，故默认值设为on。

默认值：on

track_io_timing

参数说明：控制收集数据库I/O调用时序的统计数据。I/O时序统计数据可以在pg_stat_database中查询。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能，开启时，收集器会在重复地去查询当前时间的操作系统，这可能会引起某些平台的重大开销，故默认值设置为off。
- off表示关闭收集功能。

默认值：off

track_functions

参数说明：控制收集函数的调用次数和调用耗时的统计数据。

参数类型：SUSET

须知

当SQL语言函数设置为调用查询的“内联”函数时，不管是否设置此选项，这些SQL语言函数无法被追踪到。

取值范围：枚举类型

- pl表示只追踪过程语言函数。
- all表示追踪SQL和C语言函数。
- none表示关闭函数追踪功能。

默认值：none

track_activity_query_size

参数说明：设置用于跟踪每一个活动会话的当前正在执行命令的字节数。

参数类型：POSTMASTER

取值范围：整型，100～102400

默认值：1024

update_process_title

参数说明：控制收集因每次服务器接收到一个新的SQL语句时而产生的进程名称更新的统计数据。

进程名称可以通过ps命令进行查看，在Windows下通过任务管理器查看。

参数类型：SUSET

取值范围：布尔型

- on表示开启收集功能。
- off表示关闭收集功能。

默认值：off

track_thread_wait_status_interval

参数说明：用来定期收集thread状态信息的时间间隔。

参数类型：SUSET

取值范围：整型，0～1440，单位为min。

默认值：30min

enable_save_datachanged_timestamp

参数说明：确定是否收集insert/update/delete, exchange/truncate/drop partition操作对表数据改动的时间。

参数类型：USERSET

取值范围：布尔型

- on表示允许收集相关操作对表数据改动的时间。
- off表示禁止收集相关操作对表数据改动的时间。

默认值：on

instr_unique_sql_count

参数说明：控制是否收集Unique SQL，以及收集数量限制。

参数类型：SIGHUP

取值范围：整型，0～ INT_MAX

- 值为0时，表示不收集Unique SQL统计信息；
- 值大于0时，在CN节点上，将会控制收集的Unique SQL数量不超过该设置值。当收集数量达到限制时，不再收集新的Unique SQL，此时可通过reload调大设置值，继续收集新的Unique SQL。

默认值：0

⚠ 注意

通过reload加载新的设置值时，如果新设置值小于原设置值，将会清空对应CN节点已收集的Unique SQL统计信息。需特别注意该清理操作将由资源管理后台线程完成，若GUC参数**use_workload_manager**为off时清理操作可能失败，可直接使用函数reset_instr_unique_sql进行清理。

track_sql_count

参数说明：控制对每个会话中当前正在执行的SELECT、INSERT、UPDATE、DELETE、MERGE INTO语句是否进行计数统计，对SELECT、INSERT、UPDATE、DELETE语句进行响应时间的统计，以及对DDL、DML、DCL语句进行计数的统计。

参数类型：SUSET

取值范围：布尔型

- on表示开启统计功能。
- off表示关闭统计功能。

默认值：off

说明

- track_sql_count参数受track_activities约束：
  - track_activities开启而track_sql_count关闭时，如果查询了gs_sql_count、pgxc_sql_count、gs_workload_sql_count、pgxc_workload_sql_count、global_workload_sql_count、gs_workload_sql_elapse_time、pgxc_workload_sql_elapse_time、或global_workload_sql_elapse_time视图，将会有LOG提示track_sql_count是关闭的；
  - track_activities和track_sql_count同时关闭，那么此时将会有两条LOG，分别提示track_activities是关闭的和track_sql_count是关闭的；
  - track_activities关闭而track_sql_count开启，此时将仅有LOG提示track_activities是关闭。
- 当参数关闭时，查询视图的结果为0行。

11.12.2 性能统计

在数据库在运行过程中，会涉及到锁的访问、磁盘IO操作、无效消息的处理，这些操作都可能是数据库的性能瓶颈，通过GaussDB(DWS)提供的性能统计方法，可以方便定位性能问题。

输出性能统计日志

参数说明：对每条查询，以下4个选项控制在服务器日志里记录相应模块的性能统计数据，具体含义如下：

- log_parser_stats控制在服务器日志里记录解析器的性能统计数据。
- log_planner_stats控制在服务器日志里记录查询优化器的性能统计数据。
- log_executor_stats控制在服务器日志里记录执行器的性能统计数据。
- log_statement_stats控制在服务器日志里记录整个语句的性能统计数据。

这些参数只能辅助管理员进行粗略分析，类似Linux中的操作系统工具getrusage()。

参数类型：SUSET

须知

- log_statement_stats记录总的语句统计数据，而其他的只记录针对每个模块的统计数据。
- log_statement_stats不能和其他任何针对每个模块统计的选项一起打开。

取值范围：布尔型

- on表示开启记录性能统计数据的功能。
- off表示关闭记录性能统计数据的功能。

默认值：off

11.13 负载管理

未对数据库资源做控制时，容易出现并发任务抢占资源导致操作系统过载甚至最终崩溃。操作系统过载时，其响应用户任务的速度会变慢甚至无响应；操作系统崩溃时，整个系统将无法对用户提供任何服务。GaussDB(DWS)的负载管理功能能够基于可用资源的多少均衡数据库的负载，以避免数据库系统过载。

use_workload_manager

参数说明：是否开启资源管理功能。此参数需在CN和DN同时应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开资源管理。
- off表示关闭资源管理。

  说明

  - 当使用表11-2中的方式二来修改参数值时，新参数值只能对更改操作执行后启动的线程生效。此外，对于后台线程以及线程复用执行的新作业，该参数值的改动不会生效。如果希望这类线程即时识别参数变化，可以使用kill session或重启节点的方式来实现。
  - use_workload_manager参数由off变为on状态后，资源管理视图变为可用，并且可以查询off状态下统计的存储资源使用情况。若存在些许误差的情况下，需要矫正用户使用的存储资源，可数据库中执行如下命令，在执行该命令的过程中，如果对表中插入数据，可能会出现统计不够准确的情况：

    ```
    select gs_wlm_readjust_user_space(0);
    ```

默认值：on

enable_control_group

参数说明：是否开启Cgroups功能。此参数需在CN和DN同时应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开Cgroups管理。
- off表示关闭Cgroups管理。

默认值：on

说明

当使用表11-2中的方式二来修改参数值时，新参数值只能对更改操作执行后启动的线程生效。此外，对于后台线程以及线程复用执行的新作业，该值的改动不会生效。如果希望这类线程即时识别参数变化，可以使用kill session或重启节点的方式来实现。

enable_backend_control

参数说明：是否控制数据库常驻线程到DefaultBackend控制组。此参数需在CN和DN同时应用。

参数类型：POSTMASTER

取值范围：布尔型

- on表示控制常驻线程到DefaultBackend控制组。
- off表示不控制常驻线程到DefaultBackend控制组。

默认值：on

enable_vacuum_control

参数说明：是否控制数据库常驻线程autoVacuumWorker到Vacuum控制组。此参数需在CN和DN同时应用。

参数类型：POSTMASTER

取值范围：布尔型

- on表示控制数据库常驻线程autoVacuumWorker到Vacuum控制组。
- off表示不控制数据库常驻线程autoVacuumWorker到Vacuum控制组。

默认值：on

enable_perm_space

参数说明：是否开启perm space功能。此参数需在CN和DN同时应用。

参数类型：POSTMASTER

取值范围：布尔型

- on表示打开perm space管理。
- off表示关闭perm space管理。

默认值：on

enable_verify_active_statements

参数说明：在静态自适应负载场景下，是否开启后台校准功能。此参数需在CN上应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开后台校准功能。
- off表示关闭后台校准功能。

默认值：on

max_active_statements

参数说明：设置全局的最大并发数量。此参数只应用到CN，且针对一个CN上的执行作业。

数据库管理员需根据系统资源（如CPU资源、IO资源和内存资源）情况，调整此数值大小，使得系统支持最大限度的并发作业，且防止并发执行作业过多，引起系统崩溃。

参数类型：SIGHUP

取值范围：整型，-1 ~ INT_MAX。设置为-1和0表示对最大并发数不做限制。

默认值：60

parctl_min_cost

参数说明：设置语句受到资源池并发控制的最小执行代价。

参数类型：SIGHUP

取值范围：整型，-1 ~ INT_MAX

- 值为-1时或者执行语句的代价小于10时，不受资源池并发控制。
- 值大于等于0时，当**enable_dynamic_workload**为off时，如果执行语句的代价大于或等于10并且超过这个参数值就会受到资源池并发控制。

默认值：100000

cgroup_name

参数说明：设置当前使用的Cgroups的名字或者调整当前group下排队的优先级。

即如果先设置cgroup_name，再设置session_respool，那么session_respool关联的控制组起作用，如果再切换cgroup_name，那么新切换的cgroup_name起作用。

切换cgroup_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

参数类型：USERSET

建议尽量不要混合使用cgroup_name和session_respool。

取值范围：字符串

默认值：DefaultClass:Medium

说明

DefaultClass:Medium表示DefaultClass下Timeshare控制组中的Medium控制组。

cpu_collect_timer

参数说明：设置语句执行时在DN上收集CPU时间的周期。

数据库管理员需根据系统资源（如CPU资源、IO资源和内存资源）情况，调整此数值大小，使得系统支持较合适的收集周期，太小会影响执行效率，太大会影响异常处理的精确度。

参数类型：SIGHUP

取值范围：整型，1 ~ INT_MAX， 单位为秒。

默认值：30

enable_cgroup_switch

参数说明：是否控制数据库执行语句时根据类型自动切换到TopWD组。

参数类型：USERSET

取值范围：布尔型

- on表示控制数据库执行语句时根据类型自动切换到TopWD组。
- off表示控制数据库执行语句时根据类型不自动切换到TopWD组。

默认值：off

memory_tracking_mode

参数说明：设置记录内存信息的模式。

参数类型：USERSET

取值范围：

- none，不启动内存统计功能。
- normal，仅做内存实时统计，不生成文件。
- executor，生成统计文件，包含执行层使用过的所有已分配内存的上下文信息。
- fullexec，生成文件包含执行层申请过的所有内存上下文信息。

默认值：none

memory_detail_tracking

参数说明：设置需要的线程内分配内存上下文的顺序号以及当前线程所在query的plannodeid。

参数类型：USERSET

取值范围：字符型

默认值：空

须知

该参数不允许用户进行设置，建议保持默认值。

enable_resource_track

参数说明：设置是否开启资源实时监控功能。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开资源监控。
- off表示关闭资源监控。

默认值：on

enable_resource_record

参数说明：设置是否开启资源监控记录归档功能。开启时，对于history视图（GS_WLM_SESSION_HISTORY和GS_WLM_OPERATOR_HISTORY）中的记录，每隔3分钟会分别被归档到相应的info视图（GS_WLM_SESSION_INFO和GS_WLM_OPERATOR_INFO），归档后history视图中的记录会被清除。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启资源监控记录归档功能。
- off表示关闭资源监控记录归档功能。

默认值：off

enable_user_metric_persistent

参数说明：设置是否开启用户历史资源监控转存功能。开启时，对于**PG_TOTAL_USER_RESOURCE_INFO**视图中数据，会定期采样保存到**GS_WLM_USER_RESOURCE_HISTORY**系统表中。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启用户历史资源监控转存功能。
- off表示关闭用户历史资源监控转存功能。

默认值：on

user_metric_retention_time

参数说明：设置用户历史资源监控数据的保存天数。该参数仅在enable_user_metric_persistent为on时有效。

参数类型：SIGHUP

取值范围：整型，0～3650，单位为天。

- 值等于0时，用户历史资源监控数据将永久保存。
- 值大于0时，用户历史资源监控数据将保存对应天数。

默认值：7

enable_instance_metric_persistent

参数说明：设置是否开启实例资源监控转存功能。开启时，对实例的监控数据会保存到**GS_WLM_INSTANCE_HISTORY**系统表中。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启实例资源监控转存功能。
- off表示关闭实例资源监控转存功能。

默认值：on

instance_metric_retention_time

参数说明：设置实例历史资源监控数据的保存天数。该参数仅在enable_instance_metric_persistent为on时有效。

参数类型：SIGHUP

取值范围：整型，0～3650，单位为天。

- 值等于0时，实例历史资源监控数据将永久保存。
- 值大于0时，实例历史资源监控数据将保存对应设置天数。

默认值：7

resource_track_level

参数说明：设置当前会话的资源监控的等级。该参数只有当参数enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：枚举型

- none，不开启资源监控功能。
- query，开启query级别资源监控功能。
- operator，开启query级别和算子级别资源监控功能。

默认值：query

resource_track_cost

参数说明：设置对当前会话的语句进行资源监控的最小执行代价。该参数只有当参数enable_resource_track为on时才有效。

参数类型：USERSET

取值范围：整型，-1～INT_MAX

- 值为-1时，不进行资源监控。
- 值大于或等于0且小于等于9时，对执行代价大于等于10的语句进行资源监控。
- 值大于或等于10时，对执行代价超过该参数值的语句进行资源监控。

默认值：100000

resource_track_duration

参数说明：设置资源监控实时视图（参见**表12-4**）中记录的语句执行结束后进行历史信息转存的最小执行时间。当执行完成的作业，其执行时间不小于此参数值时，作业信息会从实时视图（以statistics为后缀的视图）转存到相应的历史视图（以history为后缀的视图）中。

参数类型：USERSET

取值范围：整型，0～INT_MAX，单位为秒。

- 值为0时，资源监控实时视图（**表12-4**）中记录的所有语句都进行历史信息归档。
- 值大于0时，资源监控实时视图（**表12-4**）中记录的语句的执行时间超过这个值就会进行历史信息归档。

默认值：1min

dynamic_memory_quota

参数说明：自适应负载场景下，设置内存控制的比重，即可以使用系统最大可用内存的比例。

参数类型：SIGHUP

取值范围：整型，1～100

默认值：80

disable_memory_protect

参数说明：禁止内存保护功能。当系统内存不足时如果需要查询系统视图，可以先将此参数置为on，禁止内存保护功能，保证视图可以正常查询。该参数只适用于在系统内存不足时进行系统诊断和调试，正常运行时请保持该参数配置为off。

参数类型：USERSET

取值范围：布尔型

- on表示禁止内存保护功能。
- off表示启动内存保护功能。

默认值：off

query_band

参数说明：用于标示当前会话的作业类型，由用户自定义。

参数类型：USERSET

取值范围：字符型

默认值：空

enable_bbox_dump

参数说明：是否开启黑匣子功能，在系统不配置core机制的时候仍可产生core文件。此功能需要在CN和DN同时应用。

参数类型：SIGHUP

取值范围：布尔型

- on表示打开黑匣子功能。
- off表示关闭黑匣子功能。

默认值：off

enable_dynamic_workload

参数说明：是否开启动态负载管理功能。

参数类型：POSTMASTER

取值范围：布尔型

- on表示打开动态负载管理功能。
- off表示关闭动态负载管理功能。

默认值：on

须知

- 开启内存自适应后，不再需要使用work_mem进行算子内存使用调优，由系统根据当前负载情况，为每个语句生成计划，并估算每个算子的内存使用量和整个语句的内存使用量。系统根据负载情况和整个语句内存使用量进行队列调度，所以多并发场景会出现语句排队的情况。
- 由于优化器行数估算不准现象的存在，会出现语句内存使用量低估或高估的情况。低估时，执行时内存会自动扩展。高估时，会导致系统内存利用不足，排队语句增多，可能导致性能非最优。此时需要识别语句估算内存远大于实际DN峰值内存的语句，通过设置query_mem进行调优。
- 列存分区表导入会消耗较多内存资源，为性能敏感场景，故不推荐启用动态负载管理。

bbox_dump_count

参数说明：在bbox_dump_path定义的路径下，允许存储的GaussDB(DWS)所产生core文件最大数。超过此数量，旧的core文件会被删除。此参数只有当enable_bbox_dump为on时才生效。

参数类型：USERSET

取值范围：整型，1～20

默认值：8

说明

在并发产生core文件时，core文件的产生个数可能大于bbox_dump_count。

io_limits

参数说明：每秒触发IO的上限。

参数类型：USERSET

取值范围：整型，0～1073741823

默认值：0

io_priority

参数说明：IO利用率高达90%时，重消耗IO作业进行IO资源管控时关联的优先级等级。

参数类型：USERSET

取值范围：枚举型

- None: 表示不受控。
- Low: 表示限制iops为该作业原始触发数值的20%。
- Medium: 表示限制iops为该作业原始触发数值的50%。
- High: 表示限制iops为该作业原始触发数值的80%。

默认值：None

session_respool

参数说明：当前的session关联的resource pool。

参数类型：USERSET

即如果先设置cgroup_name，再设置session_respool，那么session_respool关联的控制组起作用，如果再切换cgroup_name，那么新切换的cgroup_name起作用。

切换cgroup_name的过程中如果指定到Workload控制组级别，数据库不对级别进行验证。级别的范围只要在1-10范围内都可以。

建议尽量不要混合使用cgroup_name和session_respool。

取值范围：string类型，通过create resource pool所设置的资源池。

默认值：invalid_pool

enable_transaction_parctl

参数说明：是否管控事务块语句和存储过程语句。

参数类型：USERSET

取值范围：布尔型

- on表示对事务块及存储过程语句进行管控。
- off表示对事务块及存储过程语句不进行管控。

默认值：on

session_statistics_memory

参数说明：设置实时查询视图的内存大小。

参数类型：SIGHUP

取值范围：整型，5MB ~ max_process_memory的50%。

默认值：5MB

session_history_memory

参数说明：设置历史查询视图的内存大小。

参数类型：SIGHUP

取值范围：整型，10MB ~ max_process_memory的50%。

默认值：10MB

topsql_retention_time

参数说明：设置历史TopSQL中gs_wlm_session_info和gs_wlm_operator_info表中数据的保存时间。

参数类型：SIGHUP

取值范围：整型，0~3650，单位为天。

- 值为0时，表示数据永久保存。
- 值大于0时，表示数据能够保存的对应天数。

默认值：0

注意

设置此GUC参数启用数据保存功能前，请先清理gs_wlm_session_info和gs_wlm_operator_info表中的数据。

transaction_pending_time

参数说明：当enable_transaction_parctl为on时，事务块语句和存储过程语句排队的最大时间。

参数类型：USERSET

取值范围：整型，-1～INT_MAX，单位为秒。

- 值为-1或0：事务块语句和存储过程语句无超时判断，排队至资源满足可执行条件。
- 值大于0：事务块语句和存储过程语句排队超过所设数值的时间后，无视当前资源情况强制执行。

默认值：0

须知

此参数仅对存储过程及事务块的内部语句有效，即PG_SESSION_WLMSTAT中enqueue字段显示为Transaction或StoredProc的语句才会生效。

11.14 自动清理

系统自动清理进程（autovacuum）自动执行VACUUM和ANALYZE命令，回收被标识为删除状态的记录空间，并更新表的统计数据。

autovacuum

参数说明：控制数据库自动清理进程（autovacuum）的启动。自动清理进程运行的前提是将**track_counts**设置为on。

参数类型：SIGHUP

说明

- 如果希望系统在故障恢复后，具备自动清理两阶段事务的功能，请将autovacuum设置为on;
- 当设置autovacuum为on，**autovacuum_max_workers**为0时，表示系统不会自动进行autovacuum，只会在故障恢复后，自动清理两阶段事务；
- 当设置autovacuum为on，**autovacuum_max_workers**大于0时，表示系统不仅在故障恢复后，自动清理两阶段事务，并且还可以自动清理进程。

须知

即使此参数设置为off，当事务ID回绕即将发生时，数据库也会自动启动自动清理进程。对于create/drop database发生异常时，可能有的节点提交或回滚，有的节点未提交（prepared状态），此时系统不能自动修复，需要手动修复，修复步骤：

1. 使用gs_clean工具（-N参数）查询出异常两阶段事务的xid以及处于prepared的节点；
2. 登录事务处于prepared状态的节点，系统管理员连接一个可用的数据库（如postgres），执行语句set xc_maintenance_mode = on；
3. 根据事务全局状态提交或者回滚此两阶段事务（如提交语句；回滚语句）。

取值范围：布尔型

- on表示开启数据库自动清理进程。
- off表示关闭数据库自动清理进程。

默认值：off

autovacuum_mode

参数说明：该参数仅在autovacuum设置为on的场景下生效，它控制autoanalyze或autovacuum的打开情况。

参数类型：SIGHUP

取值范围：枚举类型

- analyze表示只做autoanalyze。
- vacuum表示只做autovacuum。
- mix表示autoanalyze和autovacuum都做。

- none表示二者都不做。

默认值：mix

autoanalyze_timeout

参数说明：设置autoanalyze的超时时间。在对某张表做autoanalyze时，如果该表的analyze时长超过了autoanalyze_timeout，则自动取消该表此次analyze。

参数类型：SIGHUP

取值范围：整型，0～2147483，单位为秒（s）。

默认值：5min

autovacuum_io_limits

参数说明：控制autovacuum进程每秒触发IO的上限。

参数类型：SIGHUP

取值范围：整型，-1～1073741823。其中-1表示不控制，而是使用系统默认控制组。

默认值：-1

log_autovacuum_min_duration

参数说明：当自动清理的执行时间大于或者等于某个特定的值时，向服务器日志中记录自动清理执行的每一步操作。设置此选项有助于追踪自动清理的行为。

参数类型：SIGHUP

例如：将log_autovacuum_min_duration设置为250ms，记录所有运行大于或者等于250ms的自动清理命令的相关信息。

取值范围：整型，-1~INT_MAX，单位为毫秒（ms）。

- 当参数设置为0时，表示所有的自动清理操作都记录到日志中。
- 当参数设置为-1时，表示所有的自动清理操作都不记录到日志中。
- 当参数设置为非-1时，当由于锁冲突的存在导致一个自动清理操作被跳过，记录一条消息。

默认值：-1

autovacuum_max_workers

参数说明：设置能同时运行的自动清理线程的最大数量。

参数类型：POSTMASTER

取值范围：整型，0～262143 。其中0表示不会自动进行autovacuum。

默认值：3

autovacuum_naptime

参数说明：设置两次自动清理操作的时间间隔。

参数类型：SIGHUP

取值范围：整型，1～2147483 ，单位为秒（s）。

默认值：10min

autovacuum_vacuum_threshold

参数说明：设置触发VACUUM的阈值。当表上被删除或更新的记录数超过设定的阈值时才会对这个表执行VACUUM操作。

参数类型：SIGHUP

取值范围：整型，0～INT_MAX

默认值：50

autovacuum_analyze_threshold

参数说明：设置触发ANALYZE操作的阈值。当表上被删除、插入或更新的记录数超过设定的阈值时才会对这个表执行ANALYZE操作。

参数类型：SIGHUP

取值范围：整型，0～INT_MAX

默认值：50

autovacuum_vacuum_scale_factor

参数说明：设置触发一个VACUUM时增加到autovacuum_vacuum_threshold的表大小的缩放系数。

参数类型：SIGHUP

取值范围：浮点型，0.0～100.0

默认值：0.2

autovacuum_analyze_scale_factor

参数说明：设置触发一个ANALYZE时增加到autovacuum_analyze_threshold的表大小的缩放系数。

参数类型：SIGHUP

取值范围：浮点型，0.0～100.0

默认值：0.1

autovacuum_freeze_max_age

参数说明：设置事务内的最大时间，使得表的pg_class.relfrozenxid字段在VACUUM操作执行之前被写入。

VACUUM也可以删除pg_clog/子目录中的旧文件；即使自动清理进程被禁止，系统也会调用自动清理进程来防止循环重复。

参数类型：POSTMASTER

取值范围：整型，100 000～576 460 752 303 423 487

默认值：20000000000

autovacuum_vacuum_cost_delay

参数说明：设置在自动VACUUM操作里使用的开销延迟数值。

参数类型：SIGHUP

取值范围：整型，-1～100，单位为毫秒（ ms ）。其中-1表示使用常规的vacuum_cost_delay。

默认值：20ms

autovacuum_vacuum_cost_limit

参数说明：设置在自动VACUUM操作里使用的开销限制数值。

参数类型：SIGHUP

取值范围：整型，-1～10000。其中-1表示使用常规的vacuum_cost_limit。

默认值：-1

11.15 客户端连接缺省设置

11.15.1 语句行为

介绍SQL语句执行过程的相关默认参数。

search_path

参数说明：当一个被引用对象没有指定模式时，此参数设置模式搜索顺序。它的值由一个或多个模式名构成，不同的模式名用逗号隔开。

参数类型：USERSET

- 当前会话如果存放临时表的模式时，可以使用别名pg_temp将它列在搜索路径中，如'pg_temp，public' 。 存放临时表的模式始终会作为第一个被搜索的对象，排在pg_catalog和search_path中所有模式的前面，即具有第一搜索优先级。建议用户不要在search_path中显示设置pg_temp。如果在search_path中指定了pg_temp，但不是在最前面，系统会提示设置无效，pg_temp仍被优先搜索。通过使用别名pg_temp，系统只会在存放临时表的模式中搜索表、视图和数据类型这样的数据库对象，不会在里面搜索函数或运算符这样的数据库对象。
- 系统表所在的模式pg_catalog，总是排在search_path中指定的所有模式前面被搜索，即具有第二搜索优先级（ pg_temp具有第一搜索优先级 ）。建议用户不要在search_path中显式设置pg_catalog。如果在search_path中指定了pg_catalog，但不是在最前面，系统会提示设置无效，pg_catalog仍被第二优先搜索。
- 当没有指定一个特定模式而创建一个对象时，它们被放置到以search_path为命名的第一个有效模式中。当搜索路径为空时，会报错误。
- 通过SQL函数current_schema可以检测当前搜索路径的有效值。这和检测search_path的值不尽相同，因为current_schema显示search_path中首位有效的模式名称。

取值范围：字符串

说明

- 设置为"$user"，public时，支持共享数据库（没有用户具有私有模式和所有共享使用public），用户私有模式和这些功能的组合使用。可以通过改变默认搜索路径来获得其他效果，无论是全局化的还是私有化的。
- 设置为空串（''）的时候，系统会自动转换成一对双引号。
- 设置的内容中包含双引号，系统会认为是不安全字符，会将每个双引号转换成一对双引号。

默认值："$user",public

说明

$user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。

current_schema

参数说明：设置当前的模式。

参数类型：USERSET

取值范围：字符串

默认值："$user",public

说明

$user表示与当前会话用户名同名的模式名，如果这样的模式不存在，$user将被忽略。

default_tablespace

参数说明：当CREATE命令没有明确声明表空间时，所创建对象(表和索引等)的缺省表空间。

- 值是一个表空间的名字或者一个表示使用当前数据库缺省表空间的空字符串。若指定的是一个非默认表空间，用户必须具有它的CREATE权限，否则尝试创建会失败。
- 临时表不使用此参数，可以用**temp_tablespaces**代替。
- 创建数据库时不使用此参数。默认情况下，一个新的数据库从模板数据库继承表空间配置。

参数类型：USERSET

取值范围：字符串，其中空表示使用默认表空间。

默认值：空

default_storage_nodegroup

参数说明：设置当前的默认建表所在的Node Group，目前只适用普通表。

- 值为“installation”表示建表会默认建在安装的Node Group上。
- 值为其他字符串表示建表会默认建在设置的Node Group上。

参数类型：USERSET

取值范围：字符串

默认值：installation

temp_tablespaces

参数说明：当一个CREATE命令没有明确指定一个表空间时，temp_tablespaces指定了创建临时对象（临时表和临时表的索引）所在的表空间。在这些表空间中创建临时文件用来做大型数据的排序工作。

其值是一系列表空间名的列表。如果列表中有多个表空间时，每次临时对象的创建，GaussDB(DWS)会在列表中随机选择一个表空间；如果在事务中，连续创建的临时对象被放置在列表里连续的表空间中。如果选择的列表中的元素是一个空串，GaussDB(DWS)将自动将当前的数据库设为默认的表空间。

参数类型：USERSET

取值范围：字符串。空字符串表示所有的临时对象仅在当前数据库默认的表空间中创建，请参见**default_tablespace**。

默认值：空

check_function_bodies

参数说明：设置是否在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。为了避免产生问题（比如避免从转储中恢复函数定义时向前引用的问题），偶尔会禁用验证。

参数类型：USERSET

取值范围：布尔型

- on表示在CREATE FUNCTION执行过程中进行函数体字符串的合法性验证。
- off表示在CREATE FUNCTION执行过程中不进行函数体字符串的合法性验证。

默认值：on

default_transaction_isolation

参数说明：设置默认的事务隔离级别。

参数类型：USERSET

取值范围：枚举型

- read committed：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。这是缺省值。
- read uncommitted：读未提交隔离级别，GaussDB(DWS)不支持read uncommitted，如果设置了read uncommitted，实际上使用的是read committed。
- repeatable read：可重复读隔离级别，仅仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其它并发事务提交的修改。
- serializable：事务可序列化，GaussDB(DWS)不支持SERIALIZABLE，如果设置了serializable，实际上使用的是repeatable read。

默认值：read committed

default_transaction_read_only

参数说明：设置每个新创建事务是否是只读状态。

参数类型：SIGHUP

取值范围：布尔型

- on表示只读状态。
- off表示非只读状态。

默认值：off

default_transaction_deferrable

参数说明：控制每个新事务的默认延迟状态。只读事务或者那些比序列化更加低的隔离级别的事务除外。

GaussDB(DWS)不支持可串行化的隔离级别，因此，该参数无实际意义。

参数类型：USERSET

取值范围：布尔型

- on表示默认延迟。
- off表示默认不延迟。

默认值：off

session_replication_role

参数说明：控制当前会话与复制相关的触发器和规则的行为。

参数类型：USERSET

须知

设置此参数会丢弃之前任何缓存的执行计划。

取值范围：枚举型

- origin表示从当前会话中复制插入、删除、更新等操作。
- replica表示从其他地方复制插入、删除、更新等操作到当前会话。
- local表示函数执行复制时会检测当前登录数据库的角色并采取相应的操作。

默认值：origin

statement_timeout

参数说明：当语句执行时间超过该参数设置的时间（从服务器收到命令时开始计时）时，该语句将会报错并退出执行。

参数类型：USERSET

取值范围：整型，0～2147483647，单位为毫秒（ms）。

默认值：0

vacuum_freeze_min_age

参数说明：指定VACUUM在扫描一个表时用于判断是否用FrozenXID替换事务ID的中断寿命(在同一个事务中)。

参数类型：USERSET

取值范围：整型，0～576 460 752 303 423 487

说明

尽管随时可以将此参数设为0到10亿之间的任意值，但是，VACUUM将默认其有效值范围限制在autovacuum_freeze_max_age的50%以内。

默认值：5000000000

vacuum_freeze_table_age

参数说明：指定VACUUM对全表的扫描冻结元组的时间。如果表的pg_class.relfrozenxid字段的值已经达到了参数指定的时间，VACUUM对全表进行扫描。

参数类型：USERSET

取值范围：整型，0～576 460 752 303 423 487

说明

尽管随时可以将此参数设为零到20亿之间的值，但是，VACUUM将默认其有效值范围限制在autovacuum_freeze_max_age的95%以内。定期的手动VACUUM可以在对此表的反重叠自动清理启动之前运行。

默认值：15000000000

bytea_output

参数说明：设置bytea类型值的输出格式。

参数类型：USERSET

取值范围：枚举型

- hex：将二进制数据编码为每字节2位十六进制数字。
- escape：传统化的PostgreSQL格式。采用以ASCII字符序列表示二进制串的方法，同时将那些无法表示成ASCII字符的二进制串转换成特殊的转义序列。

默认值：hex

xmlbinary

参数说明：设置二进制值是如何在XML中进行编码的。

参数类型：USERSET

取值范围：枚举型

- base64
- hex

默认值：base64

xmloption

参数说明：当XML和字符串值之间进行转换时，设置document或content是否是隐含的。

参数类型：USERSET

取值范围：枚举型

- document：表示HTML格式的文档。
- content：普通的字符串。

默认值：content

max_compile_functions

参数说明：设置服务器存储的函数编译结果的最大数量。存储过多的函数和存储过程的编译结果可能占用很大内存。将此参数设置为一个合理的值，有助于减少内存占用，提升系统性能。

参数类型：POSTMASTER

取值范围：整型，1～INT_MAX。

默认值：1000

gin_pending_list_limit

参数说明：设置当GIN索引启用fastupdate时，pending list容量的最大值。当pending list的容量大于设置值时，会把pending list中数据批量移动到GIN索引数据结构中以进行清理。单个GIN索引可通过更改索引存储参数覆盖此设置值。

参数类型：USERSET

取值范围：整型，64～INT_MAX，单位为KB。

默认值：4MB

11.15.2 区域和格式化

介绍时间格式设置的相关参数。

DateStyle

参数说明：设置日期和时间值的显示格式，以及有歧义的输入值的解析规则。

这个变量包含两个独立的部分：输出格式声明（ISO、Postgres、SQL、German）和输入输出的年/月/日顺序（DMY、MDY、YMD）。这两个可以独立设置或者一起设置。关键字Euro和European等价于DMY；关键字US、NonEuro、NonEuropean等价于MDY 。

参数类型：USERSET

取值范围：字符串

默认值：ISO, MDY

说明

gs_initdb会将这个参数初始化成与**lc_time**一致的值。

设置建议：优先推荐使用ISO格式。Postgres、SQL和German均采用字母缩写的形式来表示时区，例如“EST、WST、CST”等。这些缩写可同时指代不同的时区，比如CST可同时代表美国中部时间(Central Standard Time (USA) UT-6:00)、澳大利亚中部时间(Central Standard Time (Australia) UT+9:30)等。这种情况下在时区转化时可能会得不到正确的结果，从而引发其他问题。

IntervalStyle

参数说明：设置区间值的显示格式。

参数类型：USERSET

取值范围：枚举型

- sql_standard表示产生与SQL标准规定匹配的输出。
- postgres表示产生与PostgreSQL 8.4版本相匹配的输出，当**DateStyle**参数被设为ISO时。
- postgres_verbose表示产生与PostgreSQL 8.4版本相匹配的输出，当**DateStyle**参数被设为non_ISO时。
- iso_8601表示产生与在ISO 8601中定义的“格式与代号”相匹配的输出。
- oracle表示产生于Oracle中与numtodsinterval函数相匹配的输出结果，详细请参考numtodsinterval。

须知

IntervalStyle参数也会影响不明确的间隔输入的说明。

默认值：postgres

timezone_abbreviations

参数说明：设置服务器接受的时区缩写值。

参数类型：USERSET

取值范围：字符串，可查询视图pg_timezone_names获得。

默认值：Default

说明

Default表示通用时区的缩写。但也有其他诸如 'Australia' 和 'India' 等用来定义特定的安装。

extra_float_digits

参数说明：调整浮点值显示的数据位数，浮点类型包括float4、float8 以及几何数据类型。参数值加在标准的数据位数上（ FLT_DIG或DBL_DIG中合适的 ）。

参数类型：USERSET

取值范围：整型，-15～3

说明

- 设置为3，表示包括部分有效的数据位。对转储需要精确恢复的浮点数据尤其有用。
- 设置为负数，表示摒弃不需要的数据位。

默认值：0

client_encoding

参数说明：设置客户端的字符编码类型。

请根据前端业务的情况确定。尽量客户端编码和服务器端编码一致，提高效率。

参数类型：USERSET

取值范围：兼容PostgreSQL所有的字符编码类型。其中UTF8表示使用数据库的字符编码类型。

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。
- 参数建议保持默认值，不建议通过gs_guc工具或其他方式直接在postgresql.conf文件中设置client_encoding参数，即使设置也不会生效，以保证集群内部通信编码格式一致。

默认值：UTF8

推荐值：SQL_ASCII/UTF8

lc_messages

参数说明：设置信息显示的语言。

可接受的值是与系统相关的；在一些系统上，这个区域范畴并不存在，不过仍然允许设置这个变量，只是不会有任何效果。同样，也有可能是所期望的语言的翻译信息不存在。在这种情况下，用户仍然能看到英文信息。

参数类型：SUSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_monetary

参数说明：设置货币值的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_numeric

参数说明：设置数值的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

lc_time

参数说明：设置时间和区域的显示格式，影响to_char之类的函数的输出。可接受的值是系统相关的。

参数类型：USERSET

取值范围：字符串

说明

- 使用命令locale -a查看当前系统支持的区域和相应的编码格式，并可以选择进行设置。
- 默认情况下，gs_initdb会根据当前的系统环境初始化此参数，通过locale命令可以查看当前的配置环境。

默认值：C

default_text_search_config

参数说明：设置全文检索的配置信息。

如果设置为不存在的文本搜索配置时将会报错。如果default_text_search_config对应的文本搜索配置被删除，需要重新设置default_text_search_config，否则会报设置错误。

- 其被文本搜索函数使用，这些函数并没有一个明确指定的配置。
- 当与环境相匹配的配置文件确定时，gs_initdb会选择一个与环境相对应的设置来初始化配置文件。

参数类型：USERSET

取值范围：字符串

说明

GaussDB(DWS)支持pg_catalog.english，pg_catalog.simple两种配置。

默认值：pg_catalog.english

11.15.3 其他缺省

主要介绍数据库系统默认的库加载参数。

dynamic_library_path

参数说明：设置数据查找动态加载的共享库文件的路径。当需要打开一个可以动态装载的模块并且在CREATE FUNCTION或LOAD命令里面声明的名字没有目录部分时，系统将搜索这个目录以查找声明的文件。

用于dynamic_library_path的数值必须是一个冒号分隔（Windows下是分号分隔）的绝对路径列表。当一个路径名字以特殊变量$libdir为开头时，会替换为GaussDB(DWS)发布提供的模块安装路径。例如：

```
dynamic_library_path = '/usr/local/lib/postgresql:/opt/testgs/lib:$libdir'
```

参数类型：SUSET

取值范围：字符串

说明

设置为空字符串，表示关闭自动路径搜索。

默认值：$libdir

gin_fuzzy_search_limit

参数说明：设置GIN索引返回的集合大小的上限。

参数类型：USERSET

取值范围：整型，0～INT_MAX

默认值：0

11.16 锁管理

在GaussDB(DWS)中，并发执行的事务由于竞争资源会导致死锁。本节介绍的参数主要管理事务锁的机制。

deadlock_timeout

参数说明：设置死锁超时检测时间，以毫秒为单位。当申请的锁超过设定值时，系统会检查是否产生了死锁。

- 死锁的检查代价是比较高的，服务器不会在每次等待锁的时候都运行这个过程。在系统运行过程中死锁是不经常出现的，因此在检查死锁前只需等待一个相对较短的时间。增加这个值就减少了无用的死锁检查浪费的时间，但是会减慢真正的死锁错误报告的速度。在一个负载过重的服务器上，用户可能需要增大它。这个值的设置应该超过事务持续时间，这样就可以减少在锁释放之前就开始死锁检查的问题。
- 设置**log_lock_waits**时，这个选项也决定了在一个日志消息发出关于锁等待以前要等待的时间。当需要调查锁延迟时，请设置比正常deadlock_timeout更小的值。

参数类型：SUSET

取值范围：整型，1~INT_MAX，单位为毫秒（ ms ）。

默认值：1s

lockwait_timeout

参数说明：控制单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ ms ）。

默认值：20min

update_lockwait_timeout

参数说明：允许并发更新参数开启情况下，该参数控制并发更新同一行时单个锁的最长等待时间。当申请的锁等待时间超过设定值时，系统会报错。

参数类型：SUSET

取值范围：整型，0 ~ INT_MAX，单位为毫秒（ ms ）。

默认值：2min

max_locks_per_transaction

参数说明：控制每个事务能够得到的平均的对象锁的数量。

- 共享的锁表的大小是以假设任意时刻最多只有max_locks_per_transaction*(max_connections+max_prepared_transactions) 个独立的对象需要被锁住为基础进行计算的。不超过设定数量的多个对象可以在任一时刻同时被锁定。当在一个事务里面修改很多不同的表时，可能需要提高这个默认数值。只能在数据库启动的时候设置。
- 增大此参数可能导致GaussDB(DWS)请求更多的System V共享内存，有可能超过操作系统的缺省配置。
- 当运行备机时，请将此参数设置不小于主机上的值，否则，在备机上查询操作不会被允许。

参数类型：POSTMASTER

取值范围：整型，10 ~ INT_MAX

默认值：256

max_pred_locks_per_transaction

参数说明：控制每个事务允许断定锁的最大数量，是一个平均值。

- 共享的断定锁表的大小是以假设任意时刻最多只有max_pred_locks_per_transaction*(max_connections +max_prepared_transactions) 个独立的对象需要被锁住为基础进行计算的。不超过设定数量的多个对象可以在任一时刻同时被锁定。当在一个事务里面修改很多不同的表时，可能需要提高这个默认数值。只能在服务器启动的时候设置。
- 增大这个参数可能导致GaussDB(DWS)请求更多的System V共享内存，有可能超过操作系统的缺省配置。

参数类型：POSTMASTER

取值范围：整型，10 ~ INT_MAX

默认值：64

partition_lock_upgrade_timeout

参数说明：分区上的锁级别由允许读的ExclusiveLock升级到读写阻塞的AccessExclusiveLock时，会进行尝试性的锁升级，partition_lock_upgrade_timeout指示了尝试锁升级的超时时间。

- 在分区表上进行MERGE PARTITION和CLUSTER PARTITION操作时，都利用了临时表进行数据重排和文件交换，为了最大程度提高分区上的操作并发度，在数据重排阶段给相关分区加锁ExclusiveLock，在文件交换阶段加锁AccessExclusiveLock。
- 常规加锁方式是等待加锁，直到加锁成功，或者等待时间超过**lockwait_timeout**发生超时失败。
- 在分区表上进行MERGE PARTITION或CLUSTER PARTITION操作时，进入文件交换阶段需要申请加锁AccessExclusiveLock，加锁方式是尝试性加锁，加锁成功了则立即返回，不成功则等待50ms后继续下次尝试，加锁超时时间使用会话级设置参数partition_lock_upgrade_timeout。
- 特殊值：若partition_lock_upgrade_timeout取值-1，表示无限等待，即不停的尝试锁升级，直到加锁成功。

参数类型：USERSET

取值范围：整型，-1~3000，单位为秒（s）。

默认值：1800

enable_online_ddl_waitlock

参数说明：控制DDL是否会阻塞等待pg_advisory_lock/pgxc_lock_for_backup等集群锁。主要用于OM在线操作场景，不建议用户设置。

参数类型：SIGHUP

取值范围：布尔型

- on表示开启。
- off表示关闭。

默认值：off

11.17 版本和平台兼容性

11.17.1 历史版本兼容性

GaussDB(DWS)介绍数据库的向下兼容性和对外兼容性特性的参数控制。数据库系统的向后兼容性能够为对旧版本的数据库应用提供支持。本节介绍的参数主要控制数据库的向后兼容性。

array_nulls

参数说明：控制数组输入解析器是否将未用引用的NULL识别为数组的一个NULL元素。

参数类型：USERSET

取值范围：布尔型

- on表示允许向数组中输入空元素。
- off表示向下兼容旧式模式。仍然能够创建包含NULL值的数组。

默认值：on

backslash_quote

参数说明：控制字符串文本中的单引号是否能够用\'表示。

参数类型：USERSET

须知

在字符串文本符合SQL标准的情况下，\没有任何其他含义。这个参数影响的是如何处理不符合标准的字符串文本，包括明确的字符串转义语法是（ E'...' ）。

取值范围：枚举类型

- on表示一直允许使用\'表示。
- off表示拒绝使用\'表示。
- safe_encoding表示仅在客户端字符集编码不会在多字节字符末尾包含\的ASCII值时允许。

默认值：safe_encoding

default_with_oids

参数说明：在没有声明WITH OIDS和WITHOUT OIDS的情况下，这个选项控制在新创建的表中CREATE TABLE和CREATE TABLE AS是否包含一个OID字段。它还决定SELECT INTO创建的表里面是否包含OID 。

不推荐在用户表中使用OID，故默认设置为off。需要带有OID字段的表应该在创建时声明WITH OIDS 。

参数类型：USERSET

取值范围：布尔型

- on表示在新创建的表中CREATE TABLE和CREATE TABLE AS可以包含一个OID字段。
- off表示在新创建的表中CREATE TABLE和CREATE TABLE AS不可以包含一个OID字段。

默认值：off

escape_string_warning

参数说明：警告在普通字符串中直接使用反斜杠转义。

- 如果需要使用反斜杠作为转义，可以调整为使用转义字符串语法(E'...')来做转义，因为在每个SQL标准中，普通字符串的默认行为现在将反斜杠作为一个普通字符。
- 这个变量可以帮助定位需要改变的代码。

参数类型：USERSET

取值范围：布尔型

默认值：on

lo_compat_privileges

参数说明：控制是否启动对大对象权限检查的向后兼容模式。

参数类型：SUSET

取值范围：布尔型

on表示当读取或修改大对象时禁用权限检查，与PostgreSQL 9.0以前的版本兼容。

默认值：off

quote_all_identifiers

参数说明：当数据库生成SQL时，此选项强制引用所有的标识符（包括非关键字）。这将影响到EXPLAIN的输出及函数的结果，例如pg_get_viewdef。详细说明请参见gs_dump的--quote-all-identifiers选项。

参数类型：USERSET

取值范围：布尔型

- on表示打开强制引用。

- off表示关闭强制引用。

默认值：off

sql_inheritance

参数说明：控制继承语义。

参数类型：USERSET

取值范围：布尔型

off表示各种命令不能访问子表，即默认使用ONLY关键字。这是为了兼容7.1之前版本而设置的。

默认值：on

standard_conforming_strings

参数说明：控制普通字符串文本（'...'）中是否按照SQL标准把反斜扛当普通文本。

- 应用程序通过检查这个参数可以判断字符串文本的处理方式。
- 建议明确使用转义字符串语法（E'...'）来转义字符。

参数类型：USERSET

取值范围：布尔型

- on表示打开控制功能。
- off表示关闭控制功能。

默认值：on

synchronize_seqscans

参数说明：控制启动同步的顺序扫描。在大约相同的时间内并行扫描读取相同的数据块，共享I/O负载。

参数类型：USERSET

取值范围：布尔型

- on表示扫描可能从表的中间开始，然后选择"环绕"方式来覆盖所有的行，为了与已经在进行中的扫描活动同步。这可能会造成没有用ORDER BY子句的查询得到行排序造成不可预测的后果。
- off表示确保顺序扫描是从表头开始的。

默认值：on

enable_beta_features

参数说明：控制开启某些功能受限的特性，例如GDS表关联操作。这些特性在历史版本未明确禁止，但某些场景功能上存在问题，不建议用户使用。

参数类型：USERSET

取值范围：布尔型

- on表示开启这些功能受限的特性，保持前向兼容。但某些场景可能存在功能上的问题。
- off表示禁止使用这些特性。

默认值：off

11.17.2 平台和客户端兼容性

很多平台都使用数据库系统，数据库系统的对外兼容性给平台提供了很大的方便。

transform_null_equals

参数说明：控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。如果expr得出NULL值则返回真，否则返回假。

- 正确的SQL标准兼容的expr = NULL总是返回NULL（未知）。
- Microsoft Access里的过滤表单生成的查询使用expr = NULL来测试空值。打开这个选项，可以使用该接口来访问数据库。

参数类型：USERSET

取值范围：布尔型

- on表示控制表达式expr = NULL（或NULL = expr）当做expr IS NULL处理。
- off表示不控制，即expr = NULL总是返回NULL（未知）。

默认值：off

说明

新用户经常在涉及NULL的表达式上语义混淆，故默认值设为off。

lastval_supported

参数说明：控制是否可以使用lastval函数。

参数类型：POSTMASTER

取值范围：布尔型

- on表示支持lastval函数，同时nextval函数不支持下推。
- off表示不支持lastval函数，同时nextval函数可以下推。

默认值：off

td_compatible_truncation

参数说明：控制是否开启与Teradata数据库相应兼容的特征。该参数在用户连接上与TD兼容的数据库时，可以将参数设置成为on（即超长字符串自动截断功能启用），该功能启用后，在后续的insert语句中，对目标表中char和varchar类型的列插入超长字符串时，会按照目标表中相应列定义的最大长度对超长字符串进行自动截断。保证数据都能插入目标表中，而不是报错。

说明

- 超长字符串自动截断功能不适用于insert语句包含外表的场景。
- 如果向字符集为字节类型编码（SQL_ASCII，LATIN1等）的数据库中插入多字节字符数据（如汉字等），且字符数据跨越截断位置，这种情况下，按照字节长度自动截断，自动截断后会在尾部产生非预期结果。如果用户有对于截断结果正确性的要求，建议用户采用UTF8等能够按照字符截断的输入字符集作为数据库的编码集。

参数类型：USERSET

取值范围：布尔型

- on表示启动超长字符串自动截断功能。
- off表示停止超长字符串自动截断功能。

默认值：off

11.18 容错性

当数据库系统发生错误时，以下参数控制服务器处理错误的方式。

exit_on_error

参数说明：控制终止会话。

参数类型：SUSET

取值范围：布尔型

- on表示任何错误都会终止当前的会话。
- off表示只有FATAL级别的错误才会终止会话。

默认值：off

omit_encoding_error

参数说明：设置为on，数据库的客户端字符集编码为UTF-8时，出现的字符编码转换错误将打印在日志中，有转换错误的被转换字符会被忽略，以"?"代替。

参数类型：USERSET

取值范围：布尔型

- on表示有转换错误的字符将被忽略，以"?"代替，打印错误信息到日志中。
- off表示有转换错误的字符不能被转换，打印错误信息到终端。

默认值：off

max_query_retry_times

参数说明：指定SQL语句出错自动重试功能的最大重跑次数（目前支持重跑的错误类型为“Connection reset by peer”、“Lock wait timeout”和“Connection timed out”等。），设定为0时关闭重跑功能。

参数类型：USERSET

取值范围：整型，0~20。

默认值：6

cn_send_buffer_size

参数说明：指定CN端数据发送数据缓存区的大小。

参数类型：POSTMASTER

取值范围：整型，8~128， 单位为KB。

默认值：8KB

max_cn_temp_file_size

参数说明：指定SQL语句出错自动重试功能中CN端使用临时文件的最大值，设定为0表示不使用临时文件。

参数类型：SIGHUP

取值范围：整型，0~10485760， 单位为KB。

默认值：5GB

retry_ecode_list

参数说明：指定SQL语句出错自动重试功能支持的错误类型列表。

参数类型：USERSET

取值范围：字符串。

默认值：YY001 YY002 YY003 YY004 YY005 YY006 YY007 YY008 YY009 YY010 YY011 YY012 YY013 YY014 YY015 53200 08006 08000 57P01 XX003 XX009 YY016 CG003 CG004

data_sync_retry

参数说明：控制当fsync到磁盘失败后是否继续运行数据库。由于在某些操作系统的场景下，fsync失败后重试阶段即使再次fsync失败也不会报错，从而导致数据丢失。

参数类型：POSTMASTER

取值范围：布尔型

- on表示当fsync同步到磁盘失败后采取重试机制，数据库继续运行。
- off表示当fsync同步到磁盘失败后直接报panic，停止数据库。

默认值：off

11.19 连接池参数

当使用连接池访问数据库时，在系统运行过程中，数据库连接是被当作对象存储在内存中的，当用户需要访问数据库时，并非建立一个新的连接，而是从连接池中取出一个已建立的空闲连接来使用。用户使用完毕后，数据库并非将连接关闭，而是将连接放回连接池中，以供下一个请求访问使用。

min_pool_size

参数说明：CN的连接池与其它某个CN/DN的最小连接数。

参数类型：POSTMASTER

取值范围：整型，1~65535

默认值：1

max_pool_size

参数说明：CN的连接池与其它某个CN/DN的最大连接数。

参数类型：POSTMASTER

取值范围：整型，1~65535

默认值：800

max_coordinators

参数说明：集群中CN的最大数目。

参数类型：POSTMASTER

取值范围：整型，2~16

默认值：10

max_datanodes

参数说明：集群中DN的最大数目。

参数类型：POSTMASTER

取值范围：整型，2~65535

默认值：4096

cache_connection

参数说明：是否回收连接池的连接。

参数类型：SIGHUP

取值范围：布尔型

- on表示回收连接池的连接。
- off表示不回收连接池的连接。

默认值：on

enable_force_reuse_connections

参数说明：会话是否强制重用新的连接。

参数类型：BACKEND

取值范围：布尔型

- on表示强制使用新连接。
- off表示使用现有连接。

默认值：off

说明

会话连接参数，不建议用户设置。

enable_pooler_parallel

参数说明： CN的连接池是否可以在并行的模式下进行连接。

参数类型：SIGHUP

取值范围：布尔型

- on表示CN的连接池可以在并行的模式下进行连接。
- off表示CN的连接池不可以在并行的模式下进行连接。

默认值：on

11.20 集群事务

介绍集群事务隔离、事务只读、最大prepared事务数、集群维护模式目的参数设置及取值范围等内容。

transaction_isolation

参数说明：设置当前事务的隔离级别。

参数类型：USERSET

取值范围：

- read committed：读已提交隔离级别，只能读到已经提交的数据，而不会读到未提交的数据。这是缺省值。
- read uncommitted：读未提交隔离级别，GaussDB(DWS)不支持read uncommitted，如果设置了read uncommitted，实际上使用的是read committed。
- repeatable read：可重复读隔离级别，仅仅能看到事务开始之前提交的数据，不能看到未提交的数据，以及在事务执行期间由其它并发事务提交的修改。
- serializable：事务可序列化，GaussDB(DWS)不支持SERIALIZABLE，如果设置了serializable，实际上使用的是repeatable read。

默认值：read committed

transaction_read_only

参数说明：设置当前事务是只读事务。

参数类型：USERSET

取值范围：布尔型

- on表示设置当前事务为只读事务。
- off表示该事务可以是非只读事务。

默认值：off

xc_maintenance_mode

参数说明：设置系统进入维护模式。

该参数属于SUSET类型参数，仅支持表11-2中的方式三进行设置。

取值范围：布尔型

- on表示该功能启用。
- off表示该功能被禁用。

须知

谨慎打开这个开关，避免引起集群数据不一致。

默认值：off

allow_concurrent_tuple_update

参数说明：设置是否允许并发更新。

参数类型：USERSET

取值范围：布尔型

- on表示该功能启用。
- off表示该功能被禁用。

默认值：on

gtm_backup_barrier

参数说明：指定是否为GTM启动点创建还原点。

参数类型：SUSET

取值范围：布尔型

- on表示创建还原点。
- off表示不创建还原点。

默认值：off

gtm_conn_check_interval

参数说明：设置CN检查本地线程与主GTM连接是否正常时间。

参数类型：SUSET

取值范围：整型， 0 ~ INT_MAX / 1000，单位为秒。

默认值：10s

transaction_deferrable

参数说明：指定是否允许一个只读串行事务延迟执行，使其不会执行失败。该参数设置为on时，当一个只读事务发现读取的元组正在被其他事务修改，则延迟该只读事务直到其他事务修改完成。目前，GaussDB(DWS)暂时未用到这个参数。与该参数类似的还有一个**default_transaction_deferrable**，设置它来指定一个事务是否允许延迟。

参数类型：USERSET

取值范围：布尔型

- on表示允许执行。
- off表示不允许执行。

默认值：off

enforce_two_phase_commit

参数说明：为了兼容历史版本功能保留该参数，当前版本设置无效。

enable_show_any_tuples

参数说明：该参数只有在只读事务中可用，用于分析。当这个参数被置为on/true时，表中元组的所有版本都会可见。

参数类型：USERSET

取值范围：布尔型

- on/true表示表中元组的所有版本都会可见。
- off/false表示表中元组的所有版本都不可见。

默认值：off

gtm_connect_retries

参数说明：控制GTM连接重试的次数。

参数类型：SIGHUP

取值范围：int，最小值为1，最大值为2147483647。

默认值：30

enable_redistribute

参数说明：节点不匹配时是否重新分配。

参数类型：SUSET

取值范围：布尔型

- on表示节点不匹配时重新分配。
- off表示节点不匹配时不重新分配。

默认值：off

enable_gtm_free

参数说明：大并发场景下同一时刻存在活跃事务较多，GTM下发的快照变大且快照请求变多的情况下，瓶颈卡在GTM与CN通讯的网络上。为消除该瓶颈，引入GTM-FREE模式。取消CN和GTM的交互，取消CN下发GTM获取的事务信息给DN。CN只向各个DN发送query，各个DN由本地产生快照及xid等信息，开启该参数支持分布式事务读最终一致性，即分布式事务只有写外部一致性，不具有读外部一致性。

对于要求强一致性读的OLTP场景或OLAP场景，建议不要开启该参数。
GaussDB(DWS)不支持该特性，设置后无法生效。

参数类型：SUSET

取值范围：布尔型

- on表示开启GTM-FREE模式，集群状态为读最终一致性。
- off表示非GTM-FREE模式。

默认值：off

11.21 双集群复制参数

max_changes_in_memory

参数说明：逻辑解码时单条事务在内存中缓存的大小上限，单位字节。

参数类型：POSTMASTER

取值范围：整型，1~INT_MAX

默认值：4096

max_cached_tuplebufs

参数说明：逻辑解码时总元组信息在内存中缓存的大小上限，单位字节。建议设置为**max_changes_in_memory**的两倍以上。

参数类型：POSTMASTER

取值范围：整型，1~INT_MAX

默认值：8192

11.22 开发人员选项

enable_fast_query_shipping

参数说明：控制查询优化器是否使用分布式框架。

参数类型：USERSET

取值范围：布尔型

- on表示执行计划在CN和DN上各自生成。
- off表示使用分布式框架，即执行计划在CN上生成，然后发送到DN中执行。

默认值：on

enable_trigger_shipping

参数说明：控制触发器场景是否允许将触发器下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许将触发器下推到DN执行。
- off表示不允许将触发器下推到DN执行，在CN执行。

默认值：on

enable_remotejoin

参数说明：设置是否允许连接操作计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许连接操作计划下推到DN执行。
- off表示不允许连接操作计划下推到DN执行。

默认值：on

enable_remotegroup

参数说明：设置是否允许group by与aggregates执行计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许group by与aggregates执行计划下推到DN执行。
- off表示不允许group by与aggregates执行计划下推到DN执行。

默认值：on

enable_remotelimit

参数说明：设置是否允许LIMIT子句执行计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许LIMIT子句执行计划下推到DN执行。
- off表示不允许LIMIT子句执行计划下推到DN执行。

默认值：on

enable_remotesort

参数说明：设置是否允许ORDER BY子句操作计划下推到DN执行。

参数类型：USERSET

取值范围：布尔型

- on表示允许ORDER BY子句操作计划下推到DN执行。
- off表示不允许ORDER BY子句操作计划下推到DN执行。

默认值：on

enable_join_pseudoconst

参数说明：设置是否允许与伪常数进行join。伪常数是指join两侧的变量都等于同一个常量。

参数类型：USERSET

取值范围：布尔型

- on表示允许与伪常数进行join。
- off表示不允许与伪常数进行join。

默认值：off

debug_assertions

参数说明：控制打开各种断言检查。能够协助调试，当遇到奇怪的问题或者崩溃，请把此参数打开，因为它能暴露编程的错误。要使用这个参数，必须在编译GaussDB(DWS)的时候定义宏USE_ASSERT_CHECKING（通过configure选项 --enable-cassert完成）。

参数类型：USERSET

取值范围：布尔型

- on表示打开断言检查。
- off表示不打开断言检查。

说明

当启用断言选项编译GaussDB(DWS)时，debug_assertions缺省值为on 。

默认值：off

distribute_test_param

参数说明：控制分布式测试框架打桩点是否生效。通常开发人员在进行故障注入测试时会在代码中预埋一些打桩点，使用唯一的名称进行标识，使用此参数可以控制代码中预埋的打桩点是否生效。参数采用逗号分隔的三元组形式，分别指定线程级别、测试桩名称和注入故障的错误级别。

参数类型：USERSET

取值范围：字符串，任一个已预埋的测试桩名称。

默认值：-1, default, default

ignore_checksum_failure

参数说明： 设置读取数据时是否忽略校验信息检查失败（但仍然会告警），继续执行。该参数仅在enable_crc_check = on时有效。继续执行可能导致崩溃，传播或隐藏损坏数据，无法从远程节点恢复数据及其他严重问题。不建议用户修改设置。

参数类型：SUSET

取值范围：布尔型

- on表示忽略数据校验错误。
- off表示数据校验错误正常报错。

默认值：off

enable_colstore

参数说明：创建表时，当不指定存储方式时，默认创建为列存表。使用时，各节点值需要保持一致。属于测试参数，禁止用户启用。

参数类型：SUSET

取值范围：布尔型

默认值：off

enable_force_vector_engine

参数说明：对于支持向量化的执行器算子，如果其子节点是非向量化的算子，通过设置此参数为on，强制生成向量化的执行计划。

参数类型：USERSET

取值范围：布尔型

默认值：off

enable_csqual_pushdown

参数说明：进行查询时，是否要将过滤条件下推，进行Rough Check。

参数类型：SUSET

取值范围：布尔型

- on表示进行查询时，要将过滤条件下推，进行Rough Check。
- off表示进行查询时，不要将过滤条件下推，进行Rough Check。

默认值：on

explain_dna_file

参数说明：指定**explain_perf_mode**为run，导出的csv信息的目标文件。

参数类型：USERSET

须知

这个参数的取值必须是绝对路径加上.csv格式的文件名。

取值范围：字符串

默认值：NULL

explain_perf_mode

参数说明：此参数用来指定explain的显示格式。

参数类型：USERSET

取值范围：normal、pretty、summary、run

- normal：代表使用默认的打印格式。
- pretty：代表使用GaussDB(DWS)改进后的新显示格式。新的格式层次清晰，计划包含了plan node id，性能分析简单直接。
- summary：是在pretty的基础上增加了对打印信息的分析。
- run：在summary的基础上，将统计的信息输出到csv格式的文件中，以便于进一步分析。

默认值：normal

trace_notify

参数说明：为LISTEN和NOTIFY命令生成大量调试输出。**client_min_messages**或**log_min_messages**级别必须是DEBUG1或者更低时，才能把这些输出分别发送到客户端或者服务器日志。

参数类型：USERSET

取值范围：布尔型

- on表示打开输出功能。
- off表示关闭输出功能。

默认值：off

trace_recovery_messages

参数说明：启用恢复相关调试输出的日志录，否则将不会被记录。该参数允许覆盖正常设置的**log_min_messages**，但是仅限于特定的消息，这是为了在调试备机中使用。

参数类型：SIGHUP

取值范围：枚举类型，有效值有debug5、debug4、debug3、debug2、debug1、log，取值的详细信息请参见**log_min_messages**。

默认值：log

说明

- 默认值log表示不影响记录决策。
- 除默认值外，其他值会导致优先级更高的恢复相关调试信息被记录，因为它们有log优先权。对于常见的log_min_messages设置，这会导致无条件地将它们记录到服务器日志上。

trace_sort

参数说明：控制是否在日志中打印排序操作中的资源使用相关信息。这个选项只有在编译GaussDB(DWS)的时候定义了TRACE_SORT宏的时候才可用，不过目前TRACE_SORT是由缺省定义的。

参数类型：USERSET

取值范围：布尔型

- on表示打开控制功能。
- off表示关闭控制功能。

默认值：off

zero_damaged_pages

参数说明：控制检测导致GaussDB(DWS)报告错误的损坏的页头，终止当前事务。

参数类型：SUSET

取值范围：布尔型

默认值：off

说明

- 设置为on时，系统报告一个警告，把损坏的页面填充为零然后继续处理。该行为会破坏数据，即被损坏页面上的所有行。但是它允许绕开损坏页面然后从表中存在的未损坏页面上继续检索数据行。因此该参数在硬件或者软件错误导致的数据损坏中进行恢复是有作用的。通常不建议该参数设置为on，除非不需要从损坏的页面中恢复数据。
- 对于列存表，会将整个CU跳过然后继续处理。支持的场景包括crc校验失败、magic校验失败以及读取的CU长度错误。

string_hash_compatible

参数说明：该参数用来说明char类型和varchar/text类型的hash值计算方式是否相同，以此来判断进行分布列从char类型到相同值的varchar/text类型转换，数据分布变化时，是否需要进行重分布。

参数类型：POSTMASTER

取值范围：布尔型

- on表示计算方式相同，不需要进行重分布。
- off表示计算方式不同，需要进行重分布。

说明

计算方式的不同主要体现在字符串计算hash值时传入的字节长度上。（如果为char，则会忽略字符串后面空格的长度，如果为text或varchar，则会保留字符串后面空格的长度。）hash值的计算会影响到查询的计算结果，因此此参数一旦设置后，在整个数据库使用过程中不能再对其进行修改，以避免查询错误。

默认值：off

replication_test

参数说明：此参数用于数据复制的内部功能测试。

参数类型：USERSET

取值范围：布尔型

- on表示启用功能测试。
- off表示关闭功能测试。

默认值：off

cost_param

参数说明：该参数用于控制在特定的客户场景中，使用不同的估算方法使得估算值与真实值更接近。此参数可以同时控制多种方法，与某一方法对应的位做与操作，不为0表示该方法被选择。

当cost_param & 1 不为0，表示对于求不等值连接选择率时选择一种改良机制，此方法在自连接（两个相同的表之间连接）的估算中更加准确，V300R002C00版本开始，已弃用cost_param & 1 不为0时的路径，默认选择更优的估算公式；

当cost_param & 2 不为0，表示求多个过滤条件（Filter）的选择率时，选择最小的作为总的选择率，而非两者乘积，此方法在过滤条件的列之间关联性较强时估算更加准确；

当cost_param & 4 不为0，表示在进行stream节点估算时，选用调试模型，该模型不推荐用户使用。

参数类型：USERSET

取值范围：整型，1～INT_MAX

默认值：0

convert_string_to_digit

参数说明：设置隐式转换优先级，是否优先将字符串转为数字。

参数类型：USERSET

取值范围：布尔型

- on表示优先将字符串转为数字。
- off表示不优先将字符串转为数字。

默认值：on

须知

请谨慎调整该参数，调整该参数会修改内部数据类型转换规则并可能导致不可预期的行为。

nls_timestamp_format

参数说明：设置时间戳默认格式。

参数类型：USERSET

取值范围：字符串

默认值：DD-Mon-YYYY HH:MI:SS.FF AM

enable_partitionwise

参数说明：分区表连接操作是否选择智能算法。

参数类型：USERSET

取值范围：布尔型

- on表示选择智能算法。
- off表示不选择智能算法。

默认值：off

max_user_defined_exception

参数说明：异常最大个数，默认值不可更改。

参数类型：USERSET

取值范围：整型

默认值：1000

datanode_strong_sync

参数说明：目前该参数已废弃，无实际意义。

参数类型：USERSET

取值范围：布尔型

- on表示启用stream节点间强同步。
- off表示不启用stream节点间强同步。

默认值：off

enable_debug_vacuum

参数说明：允许输出一些与VACUUM相关的日志，便于定位VACUUM相关问题。开发人员专用，不建议普通用户使用。

参数类型：SIGHUP

取值范围：布尔型

- on/true表示开启此日志开关。
- off/false表示关闭此日志开关。

默认值：off

enable_global_stats

参数说明：标识当前统计信息模式，为便于进行全局统计信息生成计划和单DN统计信息计划对比使用，默认创建为全局统计信息模式。属于测试参数，禁止用户启用。

参数类型：SUSET

取值范围：布尔型

- on/true表示全局统计信息。
- off/false表示单DN统计信息。

默认值：on

enable_fast_numeric

参数说明：标识是否开启Numeric类型数据运算优化。Numeric数据运算是较为耗时的操作之一，通过将Numeric转化为int64/int128类型，提高Numeric运算的性能。

参数类型：USERSET

取值范围：布尔型

- on/true表示开启Numeric优化。
- off/false表示关闭Numeric优化。

默认值：on

rewrite_rule

参数说明：标识开启的可选查询重写规则。有部分查询重写规则是可选的，开启它们并不能总是对查询效率有提升效果。在特定的客户场景中，通过此GUC参数对查询重写规则进行设置，使得查询效率最优。

此参数可以控制查询重写规则的组合，比如有多个重写规则：rule1、rule2、rule3、rule4。可以设置：

```
set rewrite_rule=rule1;          --启用查询重写规则rule1
set rewrite_rule=rule2,rule3;    --启用查询重写规则rule2和rule3
set rewrite_rule=none;           --关闭所有可选查询重写规则
```

参数类型：USERSET

取值范围：字符串

- none：不使用任何可选查询重写规则
- lazyagg：使用Lazy Agg查询重写规则（消除子查询中的聚集运算）
- magicset：使用Magic Set查询重写规则（从主查询中下推条件到子查询）

- uniquecheck : 使用Unique Check重写规则（允许目标列不含聚集函数的表达式子链接场景提升，需在子链接按关联列聚集后目标列值唯一才能开启，建议专业调优人员使用）。
- disablerep : 使用禁止复制表的子链接提升规则（针对复制表禁止子链接提升）。

默认值：magicset

enable_compress_spill

参数说明：标识是否开启下盘压缩功能。

参数类型：USERSET

取值范围：布尔型

- on/true表示开启下盘优化。
- off/false表示关闭下盘优化。

默认值：on

analysis_options

参数说明：通过开启对应选项中所对应的功能选项使用相应的定位功能，包括数据校验，性能统计等，参见取值范围中的选项说明。

参数类型：USERSET

取值范围：字符串

- LLVM_COMPILE表示在explain performance显示界面中显示每个线程的codegen编译时间。
- HASH_CONFLICT表示在DN进程的pg_log目录中的log日志中显示hash表的统计信息，包括hash表大小，hash链长，hash冲突情况。
- STREAM_DATA_CHECK表示对网络传输前后的数据进行CRC校验。

默认值：off(ALL)，不开启任何定位功能。

resource_track_log

参数说明：控制自诊断的日志级别。目前仅对多列统计信息进行控制。

参数类型：USERSET

取值范围：字符串

- summary：显示简略的诊断信息。
- detail：显示详细的诊断信息。

目前这两个参数值只在显示多列统计信息未收集的告警的情况下有差别，summary不显示未收集多列统计信息的告警，detail会显示这类告警。

默认值：summary

udf_memory_limit

参数说明：控制每个CN、DN执行UDF时可用的最大物理内存量。

参数类型：POSTMASTER

取值范围：整型，200*1024～max_process_memory，单位为KB。

默认值：200MB

UDFWorkerMemHardLimit

参数说明：控制fencedUDFMemoryLimit的最大值。

参数类型：POSTMASTER

设置建议：不建议设置此参数，可用**udf_memory_limit**代替。详细内容请参见**PL/pgSQL语言函数**。

取值范围：整数，可带单位（KB，MB，GB）。

默认值：1GB

pljava_vmoptions

参数说明：用户自定义设置PL/Java函数所使用的JVM虚拟机的启动参数。

参数类型：SUSET

取值范围：字符串， 支持：

- JDK8 JVM启动参数
- JDK8 JVM系统属性参数（以-D开头如-Djava.ext.dirs）
- 用户自定义参数（以-D开头，如-Duser.defined.option）

须知

如果用户在pljava_vmoptions中设置参数不满足上述取值范围，会在使用PL/Java语言函数时报错。此参数的详细说明参见**PL/pgSQL语言函数**。

默认值：空

enable_pbe_optimization

参数说明：设置优化器是否对以PBE（Parse Bind Execute）形式执行的语句进行查询计划的优化。

参数类型：SUSET

取值范围：布尔型。

- on表示优化器将优化PBE语句的查询计划。
- off表示不使用优化。

默认值：on

enable_light_proxy

参数说明：设置优化器是否对简单查询在CN上优化执行。

参数类型：SUSET

取值范围：布尔型。

- on表示优化器将优化CN上简单查询的执行。
- off表示不使用优化。

默认值：on

checkpoint_flush_after

参数说明：设置checkpointer线程在连续写多少个磁盘页后会进行异步刷盘操作。GaussDB(DWS)中，磁盘页大小为8KB。

参数类型：SIGHUP

取值范围：整型，0~256（0表示关闭异步刷盘功能）。例如，取值32，表示checkpointer线程连续写32个磁盘页，即32*8=256KB磁盘空间后会进行异步刷盘。

默认值：32

enable_parallel_ddl

参数说明：控制多CN对同一数据库对象是否能安全的并发执行DDL操作。

参数类型：USERSET

取值范围：布尔型

- on表示可以安全的并发执行DDL操作，不会出现分布式死锁。
- off表示不能安全的并发执行DDL操作，可能会出现分布式死锁。

默认值：on

show_acce_estimate_detail

参数说明：在GaussDB(DWS)集群使用加速集群场景下（即**acceleration_with_compute_pool**设置为on），控制explain命令是否显示用于评估执行计划下推到加速集群的评估信息。评估信息一般用于运维人员在维护工作中使用，因此该参数默认关闭，此外为了避免这些信息干扰正常的explain信息显示，只有在explain命令的verbose选项打开的情况下才显示评估信息。

参数类型：USERSET

取值范围：布尔型

- on表示可以在explain命令的输出中显示评估信息。
- off表示不在explain命令的输出中显示评估信息。

默认值：off

support_batch_bind

参数说明：控制是否允许通过JDBC、ODBC、Libpq等接口批量绑定和执行PBE形式的语句。

参数类型：SIGHUP

取值范围：布尔型

- on表示使用批量绑定和执行。
- off表示不使用批量绑定和执行。

默认值：on

11.23 审计

11.23.1 审计开关

audit_enabled

参数说明：控制审计进程的开启和关闭。审计进程开启后，将从管道读取后台进程写入的审计信息，并写入审计文件。

参数类型：SIGHUP

取值范围：布尔型

- on表示启动审计功能。
- off表示关闭审计功能。

默认值：on

audit_data_format

参数说明：审计日志文件的格式。当前仅支持二进制格式。

参数类型：POSTMASTER

取值范围：字符串

默认值：binary

audit_rotation_interval

参数说明：指定创建一个新审计日志文件的时间间隔。当现在的时间减去上次创建一个审计日志的时间超过了此参数值时，服务器将生成一个新的审计日志文件。

参数类型：SIGHUP

取值范围：整型，1~INT_MAX/60，单位为min。

默认值：1d

须知

请不要随意调整此参数，否侧可能会导致audit_resource_policy无法生效，如果需要控制审计日志的存储空间和时间，请使用**audit_resource_policy**、**audit_space_limit**和**audit_file_remain_time**参数进行控制。

audit_rotation_size

参数说明：指定审计日志文件的最大容量。当审计日志消息的总量超过此参数值时，服务器将生成一个新的审计日志文件。

参数类型：SIGHUP

取值范围：整型，1~1024，单位为MB。

默认值：10MB

须知

请不要随意调整此参数，否侧可能会导致audit_resource_policy无法生效，如果需要控制审计日志的存储空间和时间，请使用audit_resource_policy、audit_space_limit和audit_file_remain_time参数进行控制。

audit_resource_policy

参数说明：控制审计日志的保存策略，以空间还是时间限制为优先策略。

参数类型：SIGHUP

取值范围：布尔型

- on表示采用空间优先策略，最多存储**audit_space_limit**大小的日志。
- off表示采用时间优先策略，最少存储**audit_file_remain_time**长度时间的日志。

默认值：on

audit_file_remain_time

参数说明：表示需记录审计日志的最短时间要求，该参数在**audit_resource_policy**为off时生效。

参数类型：SIGHUP

取值范围：整型，0~730，单位为day，0表示无时间限制。

默认值：90

audit_space_limit

参数说明：审计文件占用的磁盘空间总量。

参数类型：SIGHUP

取值范围：整型，1024KB~1024GB，单位为KB。

默认值：1GB

audit_file_remain_threshold

参数说明：审计目录下审计文件个数的最大值。

参数类型：SIGHUP

取值范围：整型，1~1048576

默认值：1048576

须知

请尽量保证此参数为1048576，并不要随意调整此参数，否则可能会导致audit_resource_policy无法生效，如果需要控制审计日志的存储空间和时间，请使用audit_resource_policy、audit_space_limit和audit_file_remain_time参数进行控制。

11.23.2 操作审计

audit_system_object

参数说明：该参数决定是否对GaussDB(DWS)数据库对象的CREATE、DROP、ALTER操作进行审计。GaussDB(DWS)数据库对象包括DATABASE、USER、schema、TABLE等。通过修改该配置参数的值，可以只审计需要的数据库对象的操作。

参数类型：SIGHUP

取值范围：整型，0～524287

- 0代表关闭GaussDB(DWS)数据库对象的CREATE、DROP、ALTER操作审计功能。
- 非0代表只审计GaussDB(DWS)的某类或者某些数据库对象的CREATE、DROP、ALTER操作。

取值说明：

该参数的值由19个二进制位的组合求出，这19个二进制位分别代表GaussDB(DWS)的19类数据库对象。如果对应的二进制位取值为0，表示不审计对应的数据库对象的CREATE、DROP、ALTER操作；取值为1，表示审计对应的数据库对象的CREATE、DROP、ALTER操作。这19个二进制位代表的具体审计内容请参见**表11-4**。

默认值：12295

表 11-4 audit_system_object 取值含义说明

二进制位	含义	取值说明
第0位	是否审计DATABASE对象的CREATE、DROP、ALTER操作。	• 0表示不审计该对象的CREATE、DROP、ALTER操作； • 1表示审计该对象的CREATE、DROP、ALTER操作。
第1位	是否审计SCHEMA对象的CREATE、DROP、ALTER操作。	• 0表示不审计该对象的CREATE、DROP、ALTER操作； • 1表示审计该对象的CREATE、DROP、ALTER操作。

二进制位	含义	取值说明
第2位	是否审计USER对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作。
第3位	是否审计TABLE对象的CREATE、DROP、ALTER、TRUNCATE操作。	● 0表示不审计该对象的CREATE、DROP、ALTER、TRUNCATE操作； ● 1表示审计该对象的CREATE、DROP、ALTER、TRUNCATE操作。
第4位	是否审计INDEX对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作。
第5位	是否审计VIEW对象的CREATE、DROP操作。	● 0表示不审计该对象的CREATE、DROP操作； ● 1表示审计该对象的CREATE、DROP操作。
第6位	是否审计TRIGGER对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作。
第7位	是否审计PROCEDURE/FUNCTION对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作。
第8位	是否审计TABLESPACE对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作。
第9位	是否审计RESOURCE POOL对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作
第10位	是否审计WORKLOAD对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作

二进制位	含义	取值说明
第11位	是否审计SERVER FOR HADOOP对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作
第12位	是否审计DATA SOURCE对象的CRAETE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作。
第13位	是否审计NODE GROUP对象的CREATE、DROP操作。	● 0表示不审计该对象的CREATE、DROP操作； ● 1表示审计该对象的CREATE、DROP操作。
第14位	是否审计ROW LEVEL SECURITY对象的CREATE、DROP、ALTER操作。	● 0表示不审计该对象的CREATE、DROP、ALTER操作； ● 1表示审计该对象的CREATE、DROP、ALTER操作。
第15位	是否审计TYPE对象的CREATE、DROP、ALTER操作。	● 0表示不审计TYPE对象的CREATE、DROP、ALTER操作； ● 1表示审计TYPE对象的CREATE、DROP、ALTER操作。
第16位	是否审计TEXT SEARCH对象（CONFIGURATION和DICTIONARY）的CREATE、DROP、ALTER操作。	● 0表示不审计TEXT SEARCH对象的CREATE、DROP、ALTER操作； ● 1表示审计TEXT SEARCH对象的CREATE、DROP、ALTER操作。
第17位	是否审计DIRECTORY对象的CREATE、DROP、ALTER操作。	● 0表示不审计DIRECTORY对象的CREATE、DROP、ALTER操作； ● 1表示审计DIRECTORY对象的CREATE、DROP、ALTER操作。
第18位	是否审计SYNONYM对象的CREATE、DROP、ALTER操作。	● 0表示不审计SYNONYM对象的CREATE、DROP、ALTER操作； ● 1表示审计SYNONYM对象的CREATE、DROP、ALTER操作。

enableSeparationOfDuty

参数说明：是否开启三权分立选项。

参数类型：POSTMASTER

取值范围：布尔型

- on表示开启三权分立。
- off表示不开启三权分立。

默认值：off

enable_copy_server_files

参数说明：是否开启copy服务器端文件的权限。

参数类型：POSTMASTER

取值范围：布尔型

- on表示开启copy服务端文件的权限。
- off表示不开启copy服务端文件的权限。

默认值：true

须知

copy from/to file要求具有系统管理员权限的用户才能使用，但是，在三权分立开启的状态下，系统管理员与初始用户的权限不同，可以通过使用enable_copy_server_file控制系统管理员的copy权限，避免系统管理员权限升级。

11.24 事务监控

通过设置事务超时预警，可以监控自动回滚的事务并定位其中的语句问题，并且也可以监控执行时间过长的语句。

transaction_sync_naptime

参数说明：为保证数据一致性，当本地事务与GTM上snapshot中状态不一样时会阻塞其他事务的运行，需要等待本地节点上事务状态与GTM状态一致后再运行。当CN上等待时长超过transaction_sync_naptime时会主动触发gs_clean进行清理，缩短不一致时的阻塞时长。

参数类型：USERSET

取值范围：整型，最小值为0，单位为秒（s）。

默认值：30s

说明

若该值设为0，则不会在阻塞达到时长时主动调用gs_clean进行清理，而是靠gs_clean_timeout间隔来调用gs_clean，默认是5分钟。

transaction_sync_timeout

参数说明：为保证数据一致性，当本地事务与GTM上snapshot中状态不一样时会阻塞其他事务的运行，需要等待本地节点上事务状态与GTM状态一致后再运行。当CN上等待时长超过transaction_sync_timeout时会报错，回滚事务，避免由于sync lock等其他情况长时间进程停止响应造成对系统的阻塞。

参数类型：USERSET

取值范围：整型，最小值为0，单位为秒（s）。

默认值：10min

说明

- 若该值设为0，则不会在阻塞超时时报错，回滚事务。
- 该值必须大于gs_clean_timeout，避免DN上由于还未被gs_clean清理的残留事务阻塞超时引起的不必要的事务回滚。

11.25 GTM 相关参数

config_file

参数说明：GTM配置文件名。

取值范围：字符串

默认值：gtm.conf

data_dir

参数说明：GTM数据文件目录。

取值范围：字符串

默认值：NULL

listen_addresses

参数说明：声明服务器监听客户端的TCP/IP地址。

参数类型：POSTMASTER

取值范围：

- 主机名或IP地址，多个值之间用英文逗号分隔。
- 星号（*）表示所有IP地址。
- 置空则服务器不会监听任何IP地址，这种情况下，只有Unix域套接字可以用于连接数据库。

默认值：localhost，只允许进行本地“回环”连接。

log_directory

参数说明：当logging_collector设置为on时，log_directory决定存放服务器日志文件的目录。 它可以是绝对路径，或者是相对路径（相对于数据目录的路径）。

参数类型：SIGHUP

须知

- 当配置文件中log_directory的值为非法路径（即用户对此路径无读写权限）时，会导致集群无法重新启动。
- 修改log_directory时，当指定路径为合法路径（即用户对此路径有读写权限）时，日志输出到新的路径下。当指定路径为非法路径时，日志输出到上一次的合法日志输出路径下而不影响数据库正常运行。此时即使指定的log_directory的值非法，也会写入到配置文件中。

取值范围：字符串

默认值：“pg_log”，表示在数据目录下的“pg_log/”目录下生成服务器日志。

log_min_messages

参数说明：控制写到服务器日志文件中的消息级别。每个级别都包含排在它后面的所有级别中的信息。级别越低，服务器运行日志中记录的消息就越少。

须知

当client_min_messages和log_min_messages取值相同时，其值所代表的级别不同。

参数类型：SUSET

取值范围：枚举类型，有效值有debug、debug5、debug4、debug3、debug2、debug1、info、log、notice、warning、error、fatal、panic。参数的详细信息请参见表11-3。

默认值：warning

alarm_component

参数说明：在对告警做上报时，会进行告警抑制，即同一个实例的同一个告警项在alarm_report_interval（默认值为10s）内不做重复上报。在这种情况下设置用于处理告警内容的告警组件的位置。

参数类型：POSTMASTER

取值范围：字符串

默认值：/opt/huawei/snas/bin/snas_cm_cmd

alarm_report_interval

参数说明：指定告警上报的时间间隔。

参数类型：SIGHUP

取值范围：非负整型，单位为秒。

默认值：10

enable_alarm

参数说明：是否允许打开告警检测线程，检测数据库中可能的错误场景。

参数类型：POSTMASTER

取值范围：布尔型

- on表示允许打开告警检测线程。
- off表示不允许打开告警检测线程。

默认值：on

standby_only

参数说明：是否强制同步信息到备机。

取值范围：整型，0和1

- 0表示不强制同步信息到备机。
- 1表示强制同步信息到备机。

默认值：0

gtm_max_trans

参数说明：设置gtm最大可接收连接数， 不建议用户修改该参数。如果需要改动，此参数不能小于最大连接数加100。

取值范围：整型，256~16384

默认值：8192

enable_connect_control

参数说明：设置gtm开启连接控制，检测连接IP是否来自集群内部。

取值范围：布尔值

- on：检测连接IP是否来自集群内部，非集群内部的IP连接则拒绝访问。
- off：不检测连接IP是否来自集群内部。

默认值：on

11.26 其它选项

请参考**表11-1**中对应设置方法设置以下参数。

gin_fuzzy_search_limit

参数说明：开发GIN索引的主要目的是让GaussDB(DWS)支持高度可升级的全文索引，并且通常会遇见全文索引返回海量结果的情形。此外，在查询高频词的时候，这样做所得到的结果集没什么用处。因为从磁盘读取大量记录并对其进行排序会消耗大量资源，这在产品环境下是不能接受的。为了易于控制这种情况，GIN有一个可配置的结果集大小上限配置参数gin_fuzzy_search_limit。缺省值0表示没有限制。如果设置了非零值，那么返回的结果就是从完整结果集中随机选择的一部分。

该参数属于USERSET类型参数。

取值范围：0到int最大值

默认值：0

enable_cluster_resize

参数说明：对于sql语句中涉及多个表，并且属于不同group，打开此开关可以支持此语句执行计划下推来提高性能。

该参数属于SUSET类型参数。

取值范围：布尔型

- on表示支持此语句执行计划下推来提高性能。
- off表示不支持此语句执行计划下推来提高性能。

默认值：off

说明

此参数用于内部运维场景，请勿随意开启。

cstore_insert_mode

参数说明：向HDFS表导入数据时，控制数据的存储位置，涉及导入数据功能的操作都受此参数控制，比如INSERT，UPDATE，COPY，VACUUM FULL等。

该参数属于USERSET类型参数。

取值范围：枚举型

- AUTO：数据导入的主体部分保存在HDFS存储上，少量数据保存在delta table中。
- DELTA：所有导入的数据都保存在delta table中。
- MAIN：所有导入的数据都保存在HDFS存储上。

默认值：auto

说明

该参数也可以在配置文件中指定默认值。

enable_upgrade_merge_lock_mode

参数说明：当该参数设置为on时，通过提升deltamerge内部实现的锁级别，避免和update/delete并发操作时的报错。

该参数属于USERSET类型参数。

取值范围： 布尔型

- on，提升deltamerge内部实现的锁级别，并发执行deltamerge和update/delete操作时，一个操作先执行，另一个操作被阻塞，在前一个操作完成后，后一个操作再执行。
- off，在对HDFS表的delta table的同一行并发执行deltamerge和update/delete操作时，后一个对同一行数据更新的操作会报错退出。

默认值：off

job_queue_processes

参数说明：表示系统可以并发执行的job数目。该参数为postmaster级别，通过gs_guc设置，需要重启gaussdb才能生效。

参数类型：POSTMASTER

取值范围：0～1000

功能：

- 当job_queue_processes设置为0值，表示不启用定时任务功能，任何job都不会被执行（因为开启定时任务的功能会对系统的性能有影响，有些局点可能不需要定时任务的功能，可以通过设置为0不启用定时任务功能）。
- 当job_queue_processes为大于0时，表示启用定时任务功能且系统能够并发处理的最大任务数。

启用定时任务功能后，job_scheduler线程会在定时时间间隔轮询pg_job系统表，系统设置定时任务检查周期默认为1s。

由于并行运行的任务数太多会消耗更多的系统资源，因此需要设置系统并发处理的任务数，当前并发的任务数达到job_queue_processes时，且此时又有任务到期，那么这些任务本次得不到执行而延期到下一轮询周期。因此，建议用户需要根据每个任务的执行时长合理的设置任务的时间间隔（即submit接口中的interval参数），来避免由于任务执行时间太长而导致下个轮询周期无法正常执行。

注：如果同一时间内并行的job数很多，过小的参数值会导致job等待。而过大的参数值则消耗更多的系统资源，建议设置此参数为100，用户可以根据系统资源情况合理调整。

默认值：10

ngram_gram_size

参数说明：ngram解析器分词的长度。

该参数属于USERSET类型参数。

取值范围：整型 ，1～4

默认值：2

ngram_grapsymbol_ignore

参数说明：ngram解析器是否忽略图形化字符。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示忽略图形化字符。
- off表示不忽略图形化字符。

默认值：off

ngram_punctuation_ignore

参数说明：ngram解析器是否忽略标点符号。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示忽略标点符号。
- off表示不忽略标点符号。

默认值：on

zhparser_multi_duality

参数说明：Zhparser解析器设定是否将长词内的文字自动以二字分词法聚合。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示将长词内的文字自动以二字分词法聚合。
- off表示不将长词内的文字自动以二字分词法聚合。

默认值：off

zhparser_multi_short

参数说明：Zhparser解析器分词执行时是否执行针对长词复合切分。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示执行针对长词复合切分。
- off表示不执行针对长词复合切分。

默认值：on

zhparser_multi_zall

参数说明：Zhparser解析器是否将全部单字单独显示。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示将全部单字单独显示。
- off表示不将全部单字单独显示。

默认值：off

zhparser_multi_zmain

参数说明：Zhparser解析器是否将重要单字单独显示。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示将重要单字单独显示。
- off表示不将重要单字单独显示。

默认值：off

zhparser_punctuation_ignore

参数说明：Zhparser解析器分词结果是否忽略所有的标点等特殊符号（不会忽略\r和\n）。

该参数属于USERSET类型参数。

取值范围：布尔型

- on：忽略所有的标点等特殊符号。
- off：不忽略所有的标点等特殊符号。

默认值：on

zhparser_seg_with_duality

参数说明：Zhparser解析器是否将闲散文字自动以二字分词法聚合。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示将闲散文字自动以二字分词法聚合。
- off表示不将闲散文字自动以二字分词法聚合。

默认值：off

acceleration_with_compute_pool

参数说明：在查询包含OBS或HDFS外表时，通过该参数决定查询是否通过计算资源池进行加速。

该参数属于USERSET类型参数。

取值范围：布尔型

- on表示包含有OBS或HDFS外表的查询在计算资源池可用时，会根据代价评估决定是否通过计算资源池对查询加速。
- off表示任何查询都不会通过计算资源池进行加速。

默认值：off

behavior_compat_options

参数说明：数据库兼容性行为配置项，该参数的值由若干个配置项用逗号隔开构成。

该参数属于USERSET类型参数。

取值范围：字符串

默认值：空

说明

- 当前只支持**表11-5**。
- 配置多个兼容性配置项时，相邻配置项用逗号隔开，例如：set behavior_compat_options='end_month_calculate,display_leading_zero'。

表 11-5 兼容性配置项

兼容性配置项	兼容性行为控制
display_leading_zero	浮点数显示配置项。 ● 不设置此配置项时，对于-1~0和0~1之间的小数，不显示小数点前的0。比如，0.25显示为.25。 ● 设置此配置项时，对于-1~0和0~1之间的小数，显示小数点前的0。比如，0.25显示为0.25。

end_month_calculate	add_months函数计算逻辑配置项。
	假定函数add_months的两个参数分别为param1和param2，param1的月份和param2的和为月份result。
	● 不设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期小，计算结果中的日期字段（Day字段）和param1的日期字段保持一致。比如，
	`select add_months('2018-02-28',3) from dual;`
	`add_months`
	`---------------------`
	`2018-05-28 00:00:00`
	`(1 row)`
	● 设置此配置项时，如果param1的日期（Day字段）为月末，并且param1的日期（Day字段）比result月份的月末日期比小，计算结果中的日期字段（Day字段）和result的月末日期保持一致。比如，
	`select add_months('2018-02-28',3) from dual;`
	`add_months`
	`---------------------`
	`2018-05-31 00:00:00`
	`(1 row)`

compat_analyze_sample	analyze采样行为配置项。 设置此配置项时，会优化analyze的采样行为，主要体现在analyze时全局采样会更精确的控制在3万条左右，更好的控制analyze时Coordinator端的内存消耗，保证analyze性能的稳定性。
bind_schema_tablespace	绑定模式与同名表空间配置项。 如果存在与模式名sche_name相同的表空间名，那么如果设置search_path为sche_name， default_tablespace也会同步切换到sche_name。

兼容性配置项	兼容性行为控制
bind_procedure_searchpath	未指定模式名的数据库对象的搜索路径配置项。 在存储过程中如果不显示指定模式名，会优先在存储过程所属的模式下搜索。 如果找不到，则有两种情况： ● 若不设置此参数，报错退出。 ● 若设置此参数，按照search_path中指定的顺序继续搜索。如果还是找不到，报错退出。
correct_to_number	控制to_number()结果兼容性的配置项。 若设置此配置项，则to_number()函数结果与PG11保持一致，否则默认与Oracle保持一致。
unbind_divide_bound	控制对整数除法的结果进行范围校验。 若设置此配置项，则不需要对除法结果做范围校验，例如，INT_MIN/(-1)可以得到输出结果为INT_MAX+1，反之，则会因为超过结果大于INT_MAX而报越界错误。
merge_update_multi	控制merge into匹配多行时是否进行update操作。 若设置此配置项，匹配多行时update不报错，否则默认与oracle保持一致，报错。

return_null_string	控制函数lpad()、rpad()、repeat()、regexp_split_to_table()和split_part()的结果为空字符串''的显示配置项。
	Teradata兼容模式下，此参数无影响。
	oracle兼容模式下，原来输出的空字符串变为输出NULL。
	● 不设置此配置项时，空字符串显示为NULL。
	`select length(lpad('123',0,'*')) from dual;`
	`length`
	`--------`
	`(1 row)`
	● 设置此配置项时，空字符串显示为''。
	`select length(lpad('123',0,'*')) from dual;`
	`length`
	`--------`
	`0`
	`(1 row)`

compat_concat_variadic	控制函数concat()和concat_ws()对variadic类型结果兼容性的配置项。 若设置此配置项，当concat函数参数为variadic类型时，保留oracle和Teradata兼容模式下不同的结果形式；否则默认oracle和Teradata兼容模式下结果相同，且与oracle保持一致。

兼容性配置项	兼容性行为控制
convert_string_digit_to_numeric	控制CHAR类型和INT类型进行二元BOOL运算时类型转换优先级的配置项。 ● 不设置此配置项时，类型转换优先级与PG9.6一致。 ● 设置此配置项时，所有CHAR类型和INT类型的二元BOOL运算均强制转换为NUMERIC类型进行计算。 设置此配置项后会被影响的CHAR类型包括BPCHAR、VARCHAR、NVARCHAR2、TEXT四种类型，会被影响的INT类型包括INT1、INT2、INT4、INT8四种类型。 **注意** 此配置项只对二元BOOL运算生效，例如，INT2>TEXT、INT4=BPCHAR，非BOOL运算不会受到影响，该配置项暂不支持INT>'1.1'这类UNKNOWN类型运算的转换。由于该配置项开启后，CHAR类型与INT类型的BOOL运算会优先转换为NUMERIC类型进行计算，因此会影响数据库计算性能，当JOIN列为受影响的类型组合时，还会影响执行计划。
disable_select_truncate_parallel	控制分区表的truncate等ddl的锁等级。（8.0.0.5及以上版本支持该配置项） ● 不设置此配置项时，分区表上不同分区的select与truncate可以并发进行，同时关闭分区表的FQS（快速下发）来避免可能的不一致问题。 ● 设置此配置项时，提升truncate获取主表锁为8级，支持分区表的FQS。（当出现以下场景的需求时，建议设置：1. 在分区表上以shippable建触发器；2. 关闭stream后在分区表上执行select for update/share；3. 提升分区表的简单查询以及单点delete/update的性能。）
disable_case_specific	控制字符类型匹配时是否忽略大小写。仅在Teradata兼容模式下生效。（仅8.0.0.7及以上版本支持该配置项） ● 不设置此配置项时，字符类型匹配时，字符的大小写敏感。 ● 设置此配置项时，字符类型匹配时，字符的大小写不敏感。 设置此配置项后会影响的字符类型包括CHAR、TEXT、BPCHAR、VARCHAR、NVARCHAR五种类型，会被影响的场景包括<、>、=、>=、<=、!=、<>、!=、like、not like、in、not in 共12种操作符以及case when、decode表达式。 **注意** 由于该配置项开启后，字符类型前会增加UPPER函数进而会影响估算逻辑，需要使用增强的估算模型。（建议设置：cost_param=16、cost_model_version = 1、join_num_distinct=-20、qual_num_distinct=200）

兼容性配置项	兼容性行为控制
enable_interval_to_text	控制interval到text类型的隐式转换功能。（仅8.0.0.9及以上版本支持该配置项） ● 设置此选项时，支持interval类型到text类型的隐式转换。 SELECT TO_DATE('20200923', 'yyyymmdd') - TO_DATE('20200920', 'yyyymmdd') = '3'::text; ?column? ---------- f (1 row) ● 不设置此选项时，不支持interval类型到text类型的隐式转换。 SELECT TO_DATE('20200923', 'yyyymmdd') - TO_DATE('20200920', 'yyyymmdd') = '3'::text; ?column? ---------- t (1 row)

table_skewness_warning_threshold

参数说明：设置用于表倾斜告警的阈值。

该参数属于SUSET类型参数。

取值范围：浮点型，0～1

默认值：1

table_skewness_warning_rows

参数说明：设置用于表倾斜告警的行数。

该参数属于SUSET类型参数。

取值范围：整型，0～INT_MAX

默认值：100000

enable_prevent_job_task_startup

参数说明：设置用于阻止job线程的启动。该参数属于系统内部参数，不建议用户修改设置。

参数类型：SIGHUP

取值范围：布尔型

- on表示阻止启动job线程。当job周期到来时，不会启动job执行线程。
- off表示允许启动job线程。当job周期到来时，会启动job执行线程，完成job中规定的操作。

默认值： off

说明

该参数只需在CN上设置。

12 资源负载管理

12.1 资源负载管理概述

影响集群性能的资源包括内存、CPU和磁盘I/O 和存储空间等。

在业务运行中，系统压力可能集中在集群中的一部分节点或者系统资源中的某项资源，导致系统资源不能充分利用，集群性能不能充分发挥。GaussDB(DWS)提供了一系列资源负载管理手段，来均衡任务对系统资源的利用。

CPU、内存是服务器上的计算资源，通过对资源的集中管控，可以有效避免“作业”占用资源的冲突，实现所有作业和谐共处，高优先级（例如关键报表的生成）的作业优先执行，以及用户间的资源隔离。

存储空间管理是多租户场景下的重要特性，用于限定不同用户可以使用的空间配额。GaussDB(DWS)提供简易的语法接口，在创建用户时指定存储空间的大小，通过内部逻辑实现不同用户存储空间的统计和控制。

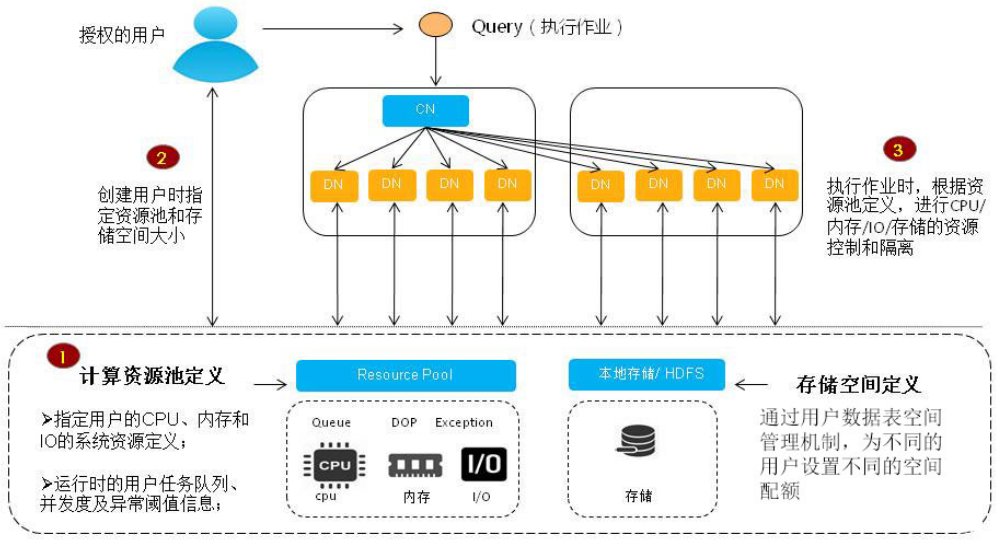
如图12-1所示。其中：

- CPU，I/O等计算资源：通过资源池进行管理。
- 内存管理可以实现节点级别的控制和作业级别的控制。内存管理通过数据库系统参数、GUC参数，和资源池等进行控制。
- 数据存储空间：通过创建用户时指定。

  说明

  资源池是GaussDB(DWS)进行负载管理的基本单元，负责管理业务运行所需的系统资源（包括CPU、I/O和内存），并提供SQL的并发控制功能。

图 12-1 资源负载管理概览



12.2 内存管理

12.2.1 内存管理概述

概述

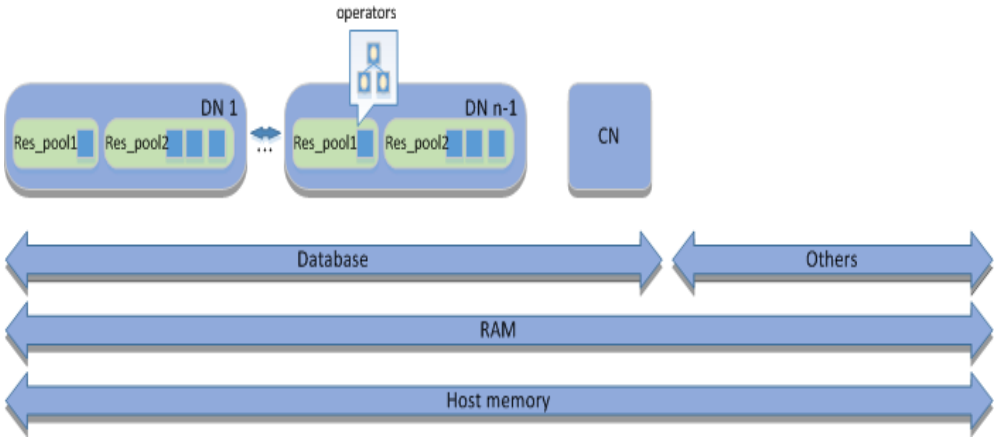
内存是GaussDB(DWS)运行的一个关键资源，如果能够有效利用，可以提升业务查询性能。

而资源池是GaussDB(DWS)管理系统资源的集合，负责分配和管理运行在资源池中任务资源。

主机内存

在一台GaussDB(DWS)节点上，主机可用内存资源被所有正在运行的进程共享，包括操作系统进程、GaussDB(DWS)实例进程、其他应用程序进程。管理员必须监控GaussDB(DWS)和非GaussDB(DWS)进程之间的内存共享情况。

图 12-2 GaussDB(DWS)主机内存使用示意图



- Host memory表示服务器可用的所有内存，这里仅包括服务器的物理内存RAM（在LINUX操作系统中，Host memory还包括交换分区Swap，因为GaussDB(DWS)使用的是精确内存管理方式，所以这里Host memory仅包括物理内存RAM）。物理内存RAM被GaussDB(DWS)进程和其他所有运行在这台主机上的进程共享。有些程序会占用大量资源，可以通过调整每个节点上的DN数量或者增加物理内存等方式解决。
- 如果该节点上部署有CN的话，CN和DN共享GaussDB(DWS)所占用内存。
- 在每一个DN中，资源池共享DN所占用的内存。
- 资源池内运行的并发SQL语句（图中标识为方块）共享资源池所占用的内存资源。
- 查询优化器生成一个执行计划，包含一系列任务（又称为算子，图中标识为operators）。任务算子例如表扫描、join或处理一些数据记录产生一个中间结果集。一个查询中的所有算子共享该查询语句占用的内存。

配置主机内存

主机内存是一台服务器上被所有程序共享的所有内存，可以通过添加更多RAM到服务器节点上以增加物理内存。

物理内存和操作系统配置通常由系统管理员来管理。

12.2.2 作业级别内存控制

背景信息

GaussDB(DWS)支持作业级别的内存控制。相比于单个算子的内存控制work_mem，更能有效利用内存资源，有效得控制内存资源的使用。

前提条件

集群安装完成且状态正常。

内存相关 GUC 参数

通过设置session级别的GUC参数query_mem，可以对作业可用内存进行控制。

操作步骤

数据库管理员可以通过下列步骤实现数据库逻辑内存管理。

步骤1 设置query_mem为500MB

```
set query_mem='500MB';
SET
```

步骤2 执行作业。

----结束

须知

如果query_mem超过资源池可用内存的上限或者低于256MB时，query_mem将不起作用，作业依然使用work_mem

12.3 资源负载管理基础框架

12.3.1 资源池

背景信息

GaussDB(DWS)资源负载管理的核心是资源池，资源池提供多种属性可以控制内存、IO和CPU资源的控制，基于优先级调度机制实现资源管理和分配，对用户业务提供资源负载管理服务。

基于资源池的资源负载管理的范围包括：并发管理、优先级调度。

资源池概述（Resource Pools）

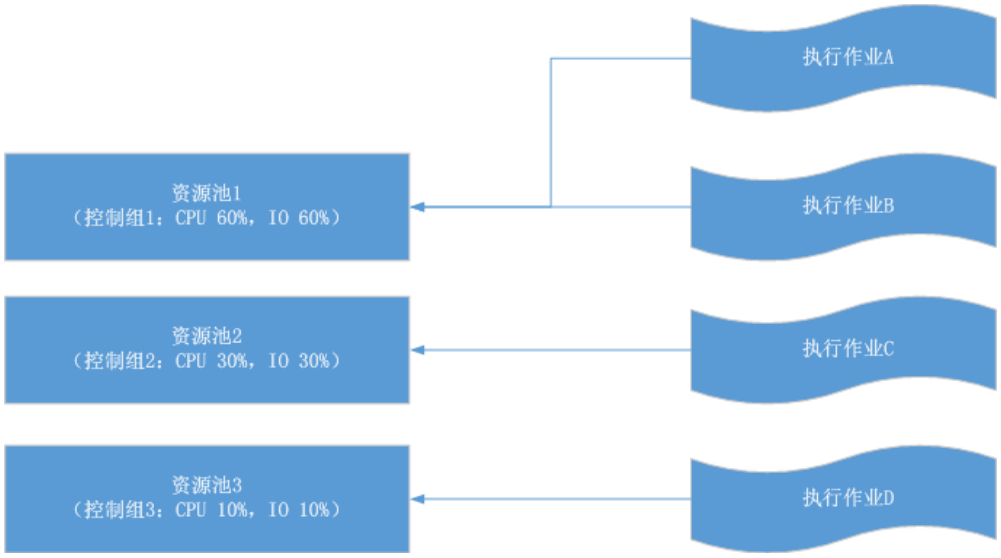
资源池是一种配置机制，用于对主机资源进行划分。资源池指定资源和并发队列属性，业务作业通过绑定资源池，来设置业务作业的优先级及能够利用到的资源。

数据库管理员需要根据不同的业务类型，将资源池绑定至对应的控制组。

基于资源池的负载管理

资源负载管理的工作原理如图12-3所示。

图 12-3 负载管理示意图



资源池通过绑定控制组进行实现资源的分配，作业的优先级和其关联的资源池的资源数量有关。一般情况下，我们认为作业关联到的资源池拥有的资源数量越多，则其优先级越高，因为该作业能够拥有更多的资源去执行。

在示例中：

- 作业A的优先级高于作业C和D，因为作业A关联的资源池拥有的资源最多。
- 作业A和作业B在同一资源池中，两者优先级相同，发生资源竞争时，平分资源池1的系统资源。

如果要调整作业的优先级，只要切换其绑定的资源池就可以了。

任务流程

资源负载管理的操作流程请参见**图12-4**和**表12-1**。

图 12-4 负载管理流程图

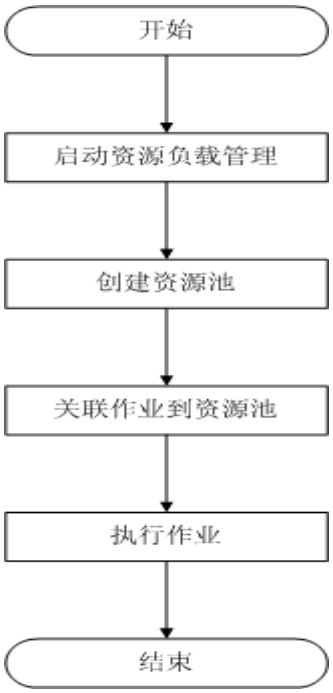


表 12-1 负载管理任务说明

任务项	说明
资源池	可以使用GaussDB(DWS)提供的对应DDL语句来创建资源池，也可以通过相应的DDL语句对其进行的修改和删除操作。
关联作业	执行作业前，将用户关联到某一个资源池，即可将作业关联到指定的资源。
执行作业	-

default_pool

在开启了资源负载管理功能之后，default_pool是由系统自动创建的，当一个会话或者用户没有指定关联的资源池时，都会被默认关联到default_pool。default_pool默认绑定DefaultClass:Medium控制组，并且不限制所关联的业务的并发数。

default_pool的详细属性参见表12-2：

表 12-2 default_pool 属性

属性	属性值	说明
respool_name	default_pool	资源池名称。
mem_percent	100	最大占用内存百分比。
cpu_affinity	-1	CPU亲和性，保留参数。

属性	属性值	说明
control_group	DefaultClass:Medium	资源池关联的控制组。
active_statements	-1	资源池允许的最大并发数。-1为不限制并发数量。
max_dop	1	开启SMP后，算子执行的并发度，保留参数。
memory_limit	8GB	内存使用上限，保留参数。
parentid	0	父资源池OID。
io_limits	0	每秒触发IO的次数上限。行存单位是万次/s,列存是次/s。0表示不控制。
io_priority	None	IO利用率高达90%时，重消耗IO作业进行IO资源管控时关联的优先级等级，包括三档可选：Low、Medium和High，分别对应限制iops为该作业原始触发数值的25%、50%及80%。None表示不控制。

须知

- GaussDB(DWS)不允许对default_pool参数进行修改。
- default_pool资源池关联的业务并发量会受到全局并发参数max_active_statements的影响。max_active_statements是指允许在某一CN上执行的最大查询

操作过程

创建资源池

开启资源负载管理之后，仅使用默认资源池并不能满足业务对资源负载管理的诉求，必须根据需要创建新的资源池，对系统资源进行重分配，来满足实际业务对系统资源精细管理的需要。

使用dbadmin或具有DBA权限的用户连接数据库后，就可以使用相关SQL语句创建和管理资源池。详细语法信息请参考CREATE RESOURCE POOL、ALTER RESOURCE POOL和DROP RESOURCE POOL。

- 创建一个关联默认控制组的资源池。如果在创建资源池的时候不指定所关联的控制组，则该资源池会被关联到默认控制组（ DefaultClass控制组下的"Medium" Timeshare控制组 ）。
  ```
  CREATE RESOURCE POOL respool1;
  ```
  当结果显示为如下信息，则表示创建成功。
  ```
  CREATE RESOURCE POOL
  ```
- 创建一个关联到"Rush" Timeshare控制组的资源池。
  ```
  CREATE RESOURCE POOL respool2 WITH (control_group='Rush');
  ```
  当结果显示为如下信息，则表示创建成功。

```
CREATE RESOURCE POOL
```

说明

- control_group取值区分大小写，指定时要使用单引号''。
- 只可以指定Timeshare控制组代表的字符串，即"Rush"、"High"、"Medium"或"Low"其中一种，如control_group的字符串为"High"；代表资源池指定到DefaultClass控制组下的"High" Timeshare控制组。

- 创建一个具有并发限制的资源池。在创建资源池的时候可以通过制定ACTIVE_STATEMENTS限制关联在其上的并发任务数，当并发数量超过所设定的ACTIVE_STATEMENTS上限时，会启用排队机制，从而实现任务并发数控制。

  ```
  CREATE RESOURCE POOL respool3 WITH (active_statements=5);
  ```

  当结果显示为如下信息，则表示创建成功。

  ```
  CREATE RESOURCE POOL
  ```

  说明

  1. active_statements字段的默认值为 10，代表该资源池上的语句最多可以有10个并发。
  2. 其取值范围为 -1 ~ INT_MAX， 当为-1时，代表不受限制。

- 创建一个具有内存限制的资源池。在创建资源池时，可以通过MEM_PERCENT参数来限制该资源池可以使用的最大内存。MEM_PERCENT的取值范围为0-100的整数。

  ```
  CREATE RESOURCE POOL respool4 WITH (MEM_PERCENT=20);
  ```

  当结果显示为如下信息，则表示创建成功。

  ```
  CREATE RESOURCE POOL
  ```

  说明

  设置资源池使用的内存大小为可用内存大小的20%。

管理资源池

在完成创建资源池后，管理员经常会根据需要对资源池进行调整配置以及删除已经废弃的资源池。

管理资源池，主要包括以下方面：

- 修改资源池的属性。
  - 修改资源池关联的控制组。

    ```
    ALTER RESOURCE POOL respool1 WITH (control_group="Rush");
    ```

    当结果显示为如下信息，则表示修改成功。

    ```
    ALTER RESOURCE POOL
    ```

  - 修改资源池的并发量。

    ```
    ALTER RESOURCE POOL respool1 WITH (ACTIVE_STATEMENTS=15);
    ```

    当结果显示为如下信息，则表示修改成功。

    ```
    ALTER RESOURCE POOL
    ```

  - 修改资源池的内存限制。

    ```
    ALTER RESOURCE POOL respool1 WITH (MEM_PERCENT=20);
    ```

    当结果显示为如下信息，则表示修改成功。

    ```
    ALTER RESOURCE POOL
    ```

  更多使用方式请参考ALTER RESOURCE POOL。

- 删除一个资源池。

  ```
  DROP RESOURCE POOL respool1;
  DROP RESOURCE POOL
  ```

局部并发管理

局部并发量，又称资源池并发量，是指各个资源池上允许运行的最大作业并发量。资源池并发量是有资源池参数“ACTIVE_STATEMENTS”来限定的。

一般而言，全局并发量应该大于局部并发量之和。如果全局并发量小于局部并发量，则实际业务执行时的并发量不大于设置的全局并发量。

资源池并发量可以在创建资源池时指定，也可以在资源池创建成功后调整。

- 资源池创建时指定：
  ```
  CREATE RESOURCE POOL pool1 WITH (ACTIVE_STATEMENTS=5);
  ```
- 资源池创建成功后调整：
  ```
  ALTER RESOURCE POOL pool1 WITH (ACTIVE_STATEMENTS=3);
  ```
  资源池使用并发点数的计数方式来计算可执行的并发数量，并发点数计算公式为

  作业使用点数：active_points = (query_mem/respool_mem) * active_statements * 100

  资源池总点数：total_points = active_statements * 100

  单位点数： 100

  说明

  1.作业不使用query_mem时会使用单位点数。

  2. 当资源池总点数耗尽后，会触发排队操作，队列满足先进先出。

查看当前资源池的信息

须知

- 不允许使用insert、update、delete、truncate操作资源负载管理的系统表pg_resource_pool，否则会导致通过不同CN查到的pg_resource_pool表的内容不一致。
- 不允许修改资源池的memory_limit、max_dop和cpu_affinity属性。

- 查看某一用户所绑定的资源池。
  ```
  SELECT rolrespool FROM PG_AUTHID WHERE rolname = 'rolename';
    rolrespool
  --------------
   default_pool
  (1 row)
  ```
- 查看当前集群中所有的资源池信息。
  ```
  SELECT * FROM PG_RESOURCE_POOL;
   respool_name | mem_percent | cpu_affinity |    control_group    | active_statements | max_dop | memory_limit | parentid | io_limits | io_priority
  --------------+-------------+--------------+---------------------+-------------------+---------+--------------+----------+-----------+-------------
   default_pool |         100 |           -1 | DefaultClass:Medium |                -1 |       1 | 8GB          |        0 |         0 | None
  (1 row)
  ```

查看资源池的控制组信息

执行如下命令查看某个资源池关联的控制组信息。命令中“resource_pool_a1”为资源池名称。

```
SELECT * FROM gs_control_group_info('resource_pool_a1');
        name         | class |  workload   | type  | gid | shares | limits | rate | cpucores
---------------------+-------+-------------+-------+-----+--------+--------+------+----------
 class_a:workload_a1 | class_a | workload_a1 | DEFWD |  87 |     30 |      0 |    0 | 0-3
(1 row)
```

表 12-3 gs_control_group_info 属性

属性	属性值	说明
name	class_a:workload_a1	class和workload名称
class	class_a	Class控制组名称
workload	workload_a1	Workload控制组名称
type	DEFWD	控制组类型（Top、CLASS、BAKWD、DEFWD、TSWD）
gid	87	控制组id
shares	30	占父节点CPU资源的百分比
limits	0	占父节点CPU核数的百分比
rate	0	Timeshare中的分配比例
cpucores	0-3	CPU核心数

12.3.2 关联作业

背景信息

在使用资源池对系统资源进行分配后，需要将作业关联到某个资源池上，才能实现对业务的负载管理。

把资源池与用户进行关联后，该用户下执行的所有作业都会自动关联到该资源池下。

操作步骤

步骤1 创建一个资源池respool，默认关联到默认控制组。

```
CREATE RESOURCE POOL respool;
CREATE RESOURCE POOL
```

说明

可以根据需要，将资源池关联至自定义控制组。请参见**资源池**。

步骤2 创建一个用户user1，将其关联到respool资源池。

```
CREATE USER user1 WITH RESOURCE POOL 'respool' password 'Bigdata@123';
CREATE USER
```

步骤3 使用user1执行作业job.sql。

```
gsql -U user1 -W Bigdata@123 -d postgres -p 8000 -f job.sql
```

或者使用user1连接到相应database，然后执行作业。

```
gsql -U user1 -W Bigdata@123 -d postgres -p 8000
```

----结束

12.4 优先级调度

12.4.1 CPU 优先级调度

优先级调度包括CPU优先级调度和IO优先级调度。CPU优先级调度根据关联控制组的优先级，将新的任务加入资源池等待队列实现调度，而IO优先级调度通过挂起数据库的IO请求实现。本节主要介绍CPU优先级调度。

前提条件

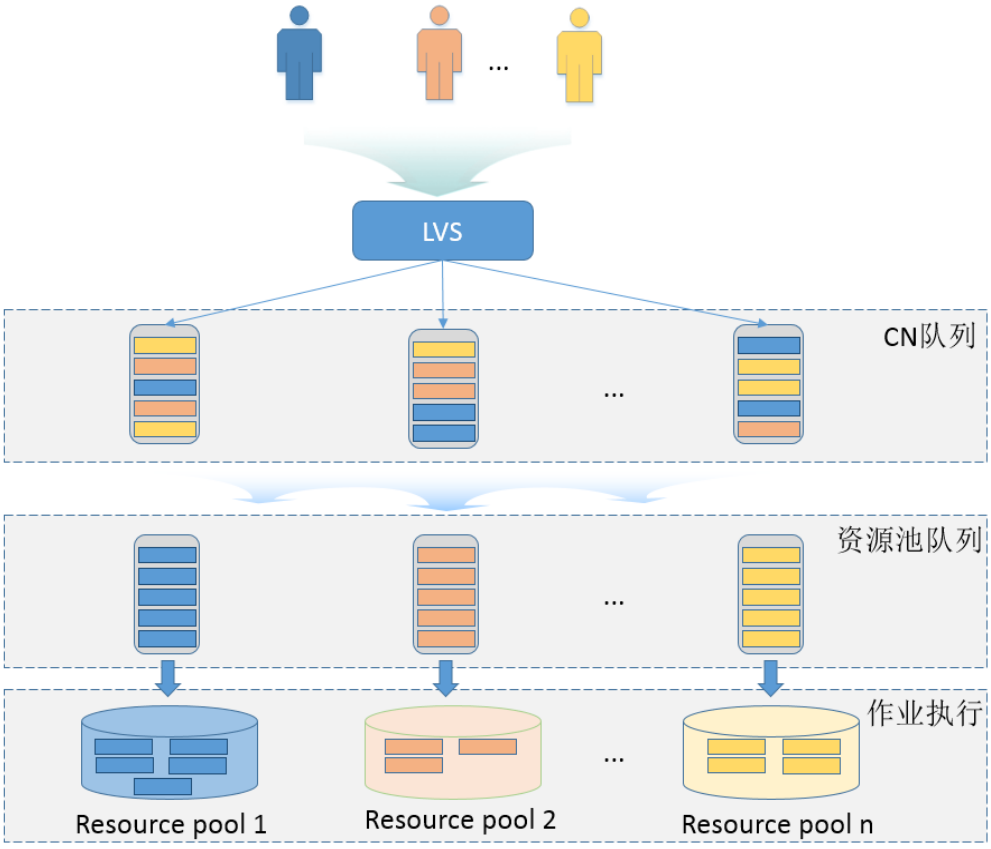
资源负载管理功能开启。

背景信息

当资源池执行的并发任务达到资源池上限后，新进入的任务会进入资源池等待队列中。

资源负载管理开启模式下等待队列的机制如**图1 负载管理等待队列的机制**所示：

图 12-5 负载管理等待队列的机制



其中，用户任务会经由CN队列和资源池队列进入到资源池中进行执行，过程如下：

1. 各用户执行的作业任务通过LVS分发至各CN队列。用户也可以直接连接至CN实例执行任务，即用户任务可以直接进入连接的CN队列。其中，在任意CN上，任务按照先进先出的原则，进入该CN的等待队列，先进入的该队列的任务会优先进入下一步。
2. 在任意CN队列中，任务会被调度至其对应的资源池队列中。规则是：
   - CN队列中的任务按照先进先出的方式被调度执行。
   - 任务进入的资源池队列是由执行任务的用户所绑定的资源池决定的，如果该用户没有绑定资源池，则任务进入默认资源池default_pool的等待队列。
3. 资源池队列中的任务按照先进先出的原则被执行。当资源池中的空闲资源满足资源池队列中最下端任务的需求时，则该任务进入资源池中开始执行。

操作步骤

- 调整资源池的优先级。

  如果一个资源池拥有的资源比例发生了变化，则其对应的优先级也会发生变化。可以通过调整资源池中属性来修改优先级。

  ```
  ALTER RESOURCE POOL respool2 WITH(CONTROL_GROUP = "Rush");
  ALTER RESOURCE POOL
  ```

  说明

  修改资源池的控制组属性为Rush。

- 调整当前用户的优先级。调整用户的优先级可以通过调整其绑定的资源池来修改。例如，

  ```
  ALTER USER user1 WITH RESOURCE POOL 'respool2';
  ALTER USER
  ```

  其中user1为需要调整优先级的用户，respool2为调整后的资源池。调整用户优先级需要有sysadmin权限。

- 调整当前会话的优先级。

  我们可以通过临时切换当前会话绑定的控制组来实现会话中任务优先级的调整。例如，

  ```
  SET CGROUP_NAME="Rush";
  SET
  ```

  查看当前会话绑定的控制组：

  ```
  SHOW CGROUP_NAME;
  ```

- 手动调整任务在队列中的位置。

  如果语句优先级较低或者长时间被阻塞需要尽快执行，可以使用插队操作，该操作仅管理员可以执行。

  a. 查询CN队列中的阻塞任务，获取需要调整的阻塞任务的pid。
     ```
     SELECT * FROM pg_session_wlmstat where status = 'pending';
     ```
  b. 调整任务在CN队列中的位置，将该任务调整到CN队列的最前端。
     ```
     SELECT pg_wlm_jump_queue(threadid);
     ```
  c. 作业执行中，把作业的优先级调整到High。通过视图pg_session_wlmstat查询到作业的threadid。
     ```
     SELECT gs_wlm_switch_cgroup(threadid, 'High');
     ```

  说明

  只有管理员才有权限对正在执行的任务调整优先级。

12.4.2 I/O 优先级调度

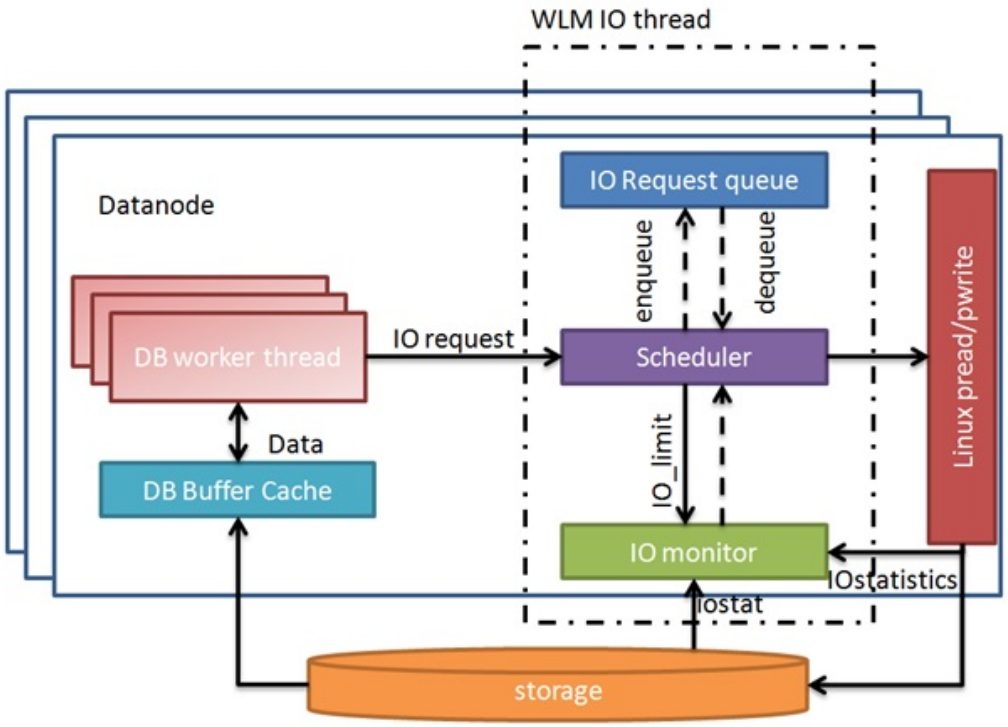
背景信息

在导入或者查询场景下，进行导入作业或者复杂查询作业的IO资源控制。

IO资源管理提供两种模式的控制方式，即优先级模式和上限数值模式，它们均可用于Session级别或者用户级别的作业IO资源控制。

IO资源管理的设计如图12-6所示：

图 12-6 IO 资源管理设计图示



IO资源管理通过以下几个维度来进行分析：

- 只考虑特定语句的作业的IO管控。对于写IO的控制，可以对INSERT INTO SELECT，COPY FROM, CREATE TABLE AS 这几种语句进行控制。对于读IO操作来说，需要根据优化器的评估计算出要扫描的数据量，通常单DN数据量大约超过500MB时就会触发IO控制。VACUUM FULL，“CREATE INDEX”，“CLUSTER TABLE USING INDEX”，“ANALYZE”，“MERGE INTO”作为读写混合的场景，也可以在一定条件下触发IO的控制。
- 需要用户手动设置GUC参数或者调整资源池来引入IO控制。
- GUC参数可对session级IO进行控制，包括优先级模式io_priority，和上限数值模式io_limits两种。
- 可以通过用户关联资源池来实现用户级别的IO控制，需要设置资源池属性io_limits, 或者io_priority.
- io_limits作为上限模式，控制每秒可以触发的IO请求数量。默认为0，表示不控制。对于行存，以万次每秒为单位，对于列存，以次每秒为单位。

- 在磁盘使用率达到90以上时，会触发IO优先级模式，优先级通过io_priority 指定。io_priority提供三种优先级，分别是High, Medium, Low，相当于控制作业触发IO比例的80%、50%及20%。默认为None，表示不控制。

  说明

  - 对于行存，由于缓冲区策略，读写IO请求并不在真正读写磁盘过程中控制，而是在读写缓冲区过程中控制。io_limits以万次每秒为单位，可设置100左右。对于行存建议使用io_priority进行控制，防止在IO比较宽松的环境下仍然做控制造成IO资源不能得到充分利用。
  - 列存查询和导入的机制跟行存不同，列存的io_limits可以设置100左右，如果设置过大，起不到控制IO的作用。

IO 资源管理相关视图

- pg_session_iostat提供session级IO资源监控
  ```
  select * from pg_session_iostat;
  ```
- pg_user_iostat提供user级IO资源监控
  ```
  select * from pg_user_iostat('username');
  ```

12.5 存储空间管理

操作场景

为了防止用户使用存储空间过大导致业务执行受阻，因此增加存储空间的控制。

存储空间管理

- 用户的空间限制。

操作步骤

假设我们要设置用户user1的最大空间为100GB。

步骤1 创建用户user1。

```
CREATE USER user1 PASSWORD 'Bigdata@123' PERM SPACE '100G';
CREATE USER
```

步骤2 通过该用户执行插入数据的作业。

----结束

说明

1. 空间信息CN会定时从DN获取并执行检查，间隔周期为10s。

2. CN检查发现空间超过限制时候，会对所执行的插入作业执行cancel操作，并会在pg_log目录下的日志文件中记录如下信息："cancel thread <threadid> for space limit exceed."

12.6 资源监控

12.6.1 用户资源查询

背景信息

在多租户管理的框架下，用户可以实时查询所有用户资源（包括内存，CPU核数，存储空间、临时空间、算子落盘空间和IO）实时使用情况，也可以查询用户资源的历史使用情况。

说明

- 用户实时资源相关视图/函数为：**PG_TOTAL_USER_RESOURCE_INFO**、GS_WLM_USER_RESOURCE_INFO；用户历史资源相关表为：**GS_WLM_USER_RESOURCE_HISTORY**。
- 上述视图、表中，used_cpu数值表示该用户对应资源池的CPU使用情况，且资源池仅记录复杂作业的CPU资源消耗。
- 上述视图、表中，IO相关资源统计值仅会统计用户所执行复杂作业的IO读写数据。
- 当用户数量较多，集群规模较大时，查询此类实时视图，因CN/DN间实时通信开销，会有一定的网络延时。
- 用户内存、CPU监控不监控简单作业，不监控管理员用户作业。

操作步骤

- 查询所有用户的资源限额和资源实时使用情况。

```
SELECT * FROM PG_TOTAL_USER_RESOURCE_INFO;
```

得到的结果视图如下：

```
username        | used_memory | total_memory | used_cpu | total_cpu | used_space | total_space |
used_temp_space | total_temp_space | used_spill_space | total_spill_space | read_kbytes | write_kbytes |
read_counts | write_counts | read_speed | write_speed
-----------------------+-------------+--------------+----------+-----------+------------+-------------
+-----------------+------------------+------------------+-------------------+-------------+--------------
+-------------+--------------+------------+-------------
perfadm             |          0 |       17250 |        0 |         0 |          0 |          -1 |             0 |
-1 |              0 |             -1 |          0 |           0 |           0 |           0 |           0 |          0
usern               |          0 |       17250 |        0 |        48 |          0 |          -1 |             0 |             -1
|              0 |             -1 |          0 |           0 |           0 |           0 |           0 |          0
userg               |         34 |       15525 |    23.53 |        48 |          0 |          -1 |             0 |
-1 |       814955731 |             -1 |     6111952 |     1145864 |     763994 |      143233 |     42678
|       8001
userg1              |         34 |       13972 |    23.53 |        48 |          0 |          -1 |             0 |
-1 |       814972419 |             -1 |     6111952 |     1145864 |     763994 |      143233 |     42710
|       8007
(4 rows)
```

其中，IO资源监控字段(read_kbytes、write_kbytes、read_counts、write_counts、read_speed和write_speed)需要在GUC参数enable_user_metric_persistent开启时才有监控数据。

所查各字段说明详见**PG_TOTAL_USER_RESOURCE_INFO** 。

- 查询具体某个用户的资源限额和资源实时使用情况。

```
SELECT * FROM GS_WLM_USER_RESOURCE_INFO('username');
```

查询结果如下：

```
userid | used_memory | total_memory | used_cpu | total_cpu | used_space | total_space |
used_temp_space | total_temp_space | used_spill_space | total_spill_space | read_kbytes | write_kbytes |
read_counts | write_counts | read_speed | write_speed
--------+-------------+--------------+----------+-----------+------------+-------------+-----------------
+-----------------+------------------+------------------+-------------------+-------------+--------------
+-------------+--------------+------------+-------------
16407 |          18 |       1655 |        6 |        19 |         13787176 |          -1 |             0 |             -1
|              0 |             -1 |          0 |           0 |           0 |           0 |           0 |          0
(1 row)
```

- 查询所有用户的资源限额和资源历史使用情况。

```
SELECT * FROM GS_WLM_USER_RESOURCE_HISTORY;
```

查询结果如下：

```
username        |           timestamp           | used_memory | total_memory | used_cpu | total_cpu |
used_space | total_space | used_temp_space | total_temp_space | used_spill_space | total_spill_space |
read_kbytes | write_kbytes | read_counts | write_counts | read_speed  | write_speed
---------------------+-------------------------------+-------------+--------------+----------+-----------
+------------+-------------+-----------------+------------------+------------------+-------------------
+-------------+--------------+-------------+--------------+-------------+-------------
usern               | 2020-01-08 22:56:06.456855+08 |           0 |        17250 |        0 |        48 |         0
|          -1 |             0 |              -1 |         88349078 |              -1 |       45680 |          34 |        5710
|           8 |         320 |           0
userg               | 2020-01-08 22:56:06.458659+08 |           0 |        15525 |    33.48 |        48 |         0
|          -1 |             0 |              -1 |        110169581 |              -1 |       17648 |          23 |
2206 |           5 |         123 |           0
userg1              | 2020-01-08 22:56:06.460252+08 |           0 |        13972 |    33.48 |        48 |         0
|          -1 |             0 |              -1 |        136106277 |              -1 |       17648 |          23 |
2206 |           5 |         123 |           0
```

对于系统表**GS_WLM_USER_RESOURCE_HISTORY**，仅当GUC参数enable_user_metric_persistent开启时，才会定期将视图**PG_TOTAL_USER_RESOURCE_INFO**中的数据保存到历史表中。

所查各字段说明详见**GS_WLM_USER_RESOURCE_HISTORY**。

- 查询具体某个用户的IO管控资源实时使用情况（视图中的数值并非表示IO读写实际值，而是代表相应的IO管控资源使用情况）。

```
SELECT * FROM pg_user_iostat('username');
```

查询结果如下：

```
 userid | min_curr_iops | max_curr_iops | min_peak_iops | max_peak_iops | io_limits | io_priority
 -------+---------------+---------------+---------------+---------------+-----------+-------------
     10 |             0 |             0 |             0 |             0 |         0 | None
(1 row)
```

12.6.2 内存资源监控

内存监控

GaussDB(DWS)提供了监控整个集群内存使用状态的视图：

查询pgxc_total_memory_detail视图，必须具有sysadmin权限。

```
SELECT * FROM pgxc_total_memory_detail;
```

如果查询该视图时出现以下错误，请开启内存管理功能。

```
SELECT * FROM pgxc_total_memory_detail;
ERROR:  unsupported view for memory protection feature is disabled.
CONTEXT:  PL/pgSQL function pgxc_total_memory_detail() line 12 at FOR over EXECUTE statement
```

共享内存监控

用户可以通过视图pg_shared_memory_detail查询共享内存上下文信息：

```
SELECT * FROM pg_shared_memory_detail;
          contextname          | level |            parent             | totalsize | freesize | usedsize
-------------------------------+-------+-------------------------------+-----------+----------+----------
 ProcessMemory                 |     0 |                               |     24576 |     9840 |    14736
 Workload manager memory context |     1 | ProcessMemory                 |   2105400 |     7304 |  2098096
 wlm collector hash table      |     2 | Workload manager memory context |      8192 |     3736 |     4456
 Resource pool hash table      |     2 | Workload manager memory context |     24576 |    15968 |     8608
 wlm cgroup hash table         |     2 | Workload manager memory context |     24576 |    15968 |     8608
(5 rows)
```

该视图列举了内存使用的上下文名称、级别、上级内存上下文、共享内存总大小等。

另外，在数据库中，GUC参数“memory_tracking_mode”用来设置内存信息统计的模式，共支持四种模式：

- none，不启动内存统计功能。
- normal，仅做内存实时统计，不生成文件。
- executor，生成统计文件，包含执行层使用过的所有已分配内存的上下文信息。

  当为executor模式时，将在DN进程的pg_log目录下生成cvs格式文件，命名方式为：memory_track_<DN名称>_query_<queryid>.csv。作业执行时，执行器postgres线程和所有stream线程执行的算子信息，都将输入该文件。

  文件内容根据以下面内容组成实例如下：

  ```
  0, 0, ExecutorState, 0, PortalHeapMemory, 0, 40K, 602K, 23
  1, 3, CStoreScan_29360131_25, 0, ExecutorState, 1, 265K, 554K, 23
  2, 128, cstore scan per scan memory context, 1, CStoreScan_29360131_25, 2, 24K, 24K, 23
  3, 127, cstore scan memory context, 1, CStoreScan_29360131_25, 2, 264K, 264K, 23
  4, 7, InitPartitionMapTmpMemoryContext, 1, CStoreScan_29360131_25, 2, 31K, 31K, 23
  5, 2, VecPartIterator_29360131_24, 0, ExecutorState, 1, 16K, 16K, 23
  0, 0, ExecutorState, 0, PortalHeapMemory, 0, 24K, 1163K, 20
  1, 3, CStoreScan_29360131_22, 0, ExecutorState, 1, 390K, 1122K, 20
  2, 20, cstore scan per scan memory context, 1, CStoreScan_29360131_22, 2, 476K, 476K, 20
  3, 19, cstore scan memory context, 1, CStoreScan_29360131_22, 2, 264K, 264K, 20
  4, 7, InitPartitionMapTmpMemoryContext, 1, CStoreScan_29360131_22, 2, 23K, 23K, 20
  5, 2, VecPartIterator_29360131_21, 0, ExecutorState, 1, 16K, 16K, 20
  ```

  其中各字段分别为：输出顺序号、线程内分配内存上下文的顺序号、当前内存上下文的名称、父内存上下文的输出顺序号、父内存上下文的名称、内存上下文树形层次级别号、当前内存上下文使用的内存峰值、当前内存上下文及其所有子内存上下文使用的内存峰值、当前线程所在query的plannodeid。

  在本例中，记录“1, 3, CStoreScan_29360131_22, 0, ExecutorState, 1, 390K, 1122K, 20”和Explain Analyze的对应关系如下：

  - CstoreScan_29360131_22代表CstoreScan算子。
  - 1122K代表CstoreScan算子的PeakMemory。
- fullexec，生成文件包含执行层申请过的所有内存上下文信息。

  当设置为fullexec模式时，输出信息和executor模式相同，但可能增加部分内存上下文分配信息，因为所有申请内存（无论是否申请成功）相关的信息都会被打印出来。由于仅记录内存申请信息，故记录中内存上下文使用的峰值均为0。

12.6.3 实例资源监控

GaussDB(DWS)提供了监控CN、DN实例资源使用状态（包括内存，CPU，磁盘IO，进程物理IO和进程逻辑IO）的系统表及监控整个集群资源使用状态的系统表。

关于系统表GS_WLM_INSTANCE_HISTORY的详细介绍，请参考GS_WLM_INSTANCE_HISTORY。

说明

系统表GS_WLM_INSTANCE_HISTORY中的数据分布在对应的实例中，CN实例监控数据保存在CN实例中，DN实例监控数据保存在DN实例中；DN实例由于有备机，当主DN实例异常时，该DN实例的监控数据能够从备机恢复；但CN实例无备机，当某CN实例异常再恢复时，该CN实例的监控数据会丢失。

操作步骤

- 查询当前实例最近的资源使用情况。

  ```
  SELECT * FROM GS_WLM_INSTANCE_HISTORY ORDER BY TIMESTAMP DESC;
  ```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
--------------+-------------------------------+----------+----------+----------+----------+----------
+------------+--------------+---------------+---------------+--------------+---------------+-------------+--------------
dn_6015_6016 | 2020-01-10 17:29:17.329495+08 |        0 |    14570 |     8982 |  662.923 |  99.9601 |
697666 |    93655.5 |       183104 |         30082 |       285659 |         30079 |      357717 |        37667
dn_6015_6016 | 2020-01-10 17:29:07.312049+08 |        0 |    14578 |     8974 |  883.102 |  99.9801 |
756228 |    81417.4 |       189722 |         30786 |       285681 |         30780 |      358103 |        38584
dn_6015_6016 | 2020-01-10 17:28:57.284472+08 |        0 |    14583 |     8969 |  727.135 |  99.9801 |
648581 |    88799.6 |       177120 |         31176 |       252161 |         31175 |      316085 |        39079
dn_6015_6016 | 2020-01-10 17:28:47.256613+08 |        0 |    14591 |     8961 |  679.534 |   100.08 |
655360 |     169962 |       179404 |         30424 |       242002 |         30422 |      303351 |        38136
```

- 查询当前实例某一段时间内的资源使用情况。

```
SELECT * FROM GS_WLM_INSTANCE_HISTORY WHERE TIMESTAMP > '2020-01-10' AND TIMESTAMP
< '2020-01-11' ORDER BY TIMESTAMP DESC;
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
--------------+-------------------------------+----------+----------+----------+----------+----------
+------------+--------------+---------------+---------------+--------------+---------------+-------------+--------------
dn_6015_6016 | 2020-01-10 17:29:17.329495+08 |        0 |    14570 |     8982 |  662.923 |  99.9601 |
697666 |    93655.5 |       183104 |         30082 |       285659 |         30079 |      357717 |        37667
dn_6015_6016 | 2020-01-10 17:29:07.312049+08 |        0 |    14578 |     8974 |  883.102 |  99.9801 |
756228 |    81417.4 |       189722 |         30786 |       285681 |         30780 |      358103 |        38584
dn_6015_6016 | 2020-01-10 17:28:57.284472+08 |        0 |    14583 |     8969 |  727.135 |  99.9801 |
648581 |    88799.6 |       177120 |         31176 |       252161 |         31175 |      316085 |        39079
dn_6015_6016 | 2020-01-10 17:28:47.256613+08 |        0 |    14591 |     8961 |  679.534 |   100.08 |
655360 |     169962 |       179404 |         30424 |       242002 |         30422 |      303351 |        38136
```

- 查询集群最近的资源使用情况，可以在CN节点上调用pgxc_get_wlm_current_instance_info存储过程函数。

```
SELECT * FROM pgxc_get_wlm_current_instance_info('ALL');
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await | io_util |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
write_counts
--------------+-------------------------------+----------+----------+----------+----------+---------+-----------
+------------+--------------+---------------+---------------+--------------+---------------+-------------+--------------
coordinator2 | 2020-01-14 21:58:29.290894+08 |        0 |    12010 |      278 |  16.0445 | 7.19561 |
184.431 |    27959.3 |            0 |            10 |            0 |             0 |           0 |            0
coordinator3 | 2020-01-14 21:58:27.567655+08 |        0 |    12000 |      288 |  .964557 | 3.40659 |
332.468 |    3375.02 |           26 |            13 |            0 |             0 |           0 |            0
datanode1    | 2020-01-14 21:58:23.900321+08 |        0 |    11899 |      389 |  1.17296 |    3.25 |
329.6 |     2870.4 |           28 |             8 |           13 |             3 |          18 |            6
datanode2    | 2020-01-14 21:58:32.832989+08 |        0 |    11904 |      384 |   17.948 | 8.52148 |
214.186 |    25894.1 |           28 |            10 |           13 |             3 |          18 |            6
datanode3    | 2020-01-14 21:58:24.826694+08 |        0 |    11894 |      394 |  1.16088 |    3.15 |       328
|     2868.8 |           25 |            10 |           13 |             3 |          18 |           6
coordinator1 | 2020-01-14 21:58:33.367649+08 |        0 |    11988 |      300 |  9.53286 |   10.05 |
43.2 |       55232 |            0 |             0 |            0 |             0 |           0 |            0
coordinator1 | 2020-01-14 21:58:23.216645+08 |        0 |    11988 |      300 |  1.17085 | 3.21182 |
324.729 |    2831.13 |            8 |            13 |            0 |             0 |           0 |            0
(7 rows)
```

- 查询集群历史的资源使用情况，可以在CN节点上调用pgxc_get_wlm_current_instance_info存储过程函数。

```
SELECT * FROM pgxc_get_wlm_history_instance_info('ALL', '2020-01-14 21:00:00', '2020-01-14
22:00:00', 3);
```

查询结果如下：

```
instancename |           timestamp           | used_cpu | free_mem | used_mem | io_await |  io_util  |
disk_read | disk_write | process_read | process_write | logical_read | logical_write | read_counts |
```

```
write_counts
-------------+-------------------------------+----------+----------+----------+----------+-----------+-----------
+------------+-------------+--------------+-------------+--------------+------------+-------------
coordinator2 | 2020-01-14 21:50:49.778902+08 |       0 |    12020 |      268 |  .127371 |   .789211 |
15.984 |    3994.41 |           0 |           0 |           0 |           0 |           0 |           0
coordinator2 | 2020-01-14 21:53:49.043646+08 |       0 |    12018 |      270 |  30.2902 |   8.65404 |
276.77 |    16741.8 |           3 |           1 |           0 |           0 |           0 |           0
coordinator2 | 2020-01-14 21:57:09.202654+08 |       0 |    12018 |      270 |   .16051 |   .979021 |
59.9401 |       5596 |           0 |           0 |           0 |           0 |           0 |           0
coordinator3 | 2020-01-14 21:38:48.948646+08 |       0 |    12012 |      276 | .0769231 | .00999001
|         0 |    35.1648 |           0 |           1 |           0 |           0 |           0 |           0
coordinator3 | 2020-01-14 21:40:29.061178+08 |       0 |    12012 |      276 |  .118421 |   .0199601
|         0 |    970.858 |           0 |           0 |           0 |           0 |           0 |           0
coordinator3 | 2020-01-14 21:50:19.612777+08 |       0 |    12010 |      278 |   24.411 |   11.7665 |
8.78244 |    44641.1 |           0 |           0 |           0 |           0 |           0 |           0
datanode1    | 2020-01-14 21:49:42.758649+08 |       0 |    11909 |      379 |  .798776 |      8.02 |
51.2 |    20924.8 |           0 |           0 |           0 |           0 |           0 |           0
datanode1    | 2020-01-14 21:49:52.760188+08 |       0 |    11909 |      379 |  23.8972 |      14.1 |
0 |      74760 |           0 |           0 |           0 |           0 |           0 |           0
datanode1    | 2020-01-14 21:50:22.769226+08 |       0 |    11909 |      379 |  39.5868 |       7.4 |         0
|    19760.8 |           0 |           0 |           0 |           0 |           0 |           0
datanode2    | 2020-01-14 21:58:02.826185+08 |       0 |    11905 |      383 |  .351648 |       .32 |
20.8 |      504.8 |           0 |           0 |           0 |           0 |           0 |           0
datanode2    | 2020-01-14 21:56:42.80793+08  |       0 |    11906 |      382 |  .559748 |       .04 |         0
|      326.4 |           0 |           0 |           0 |           0 |           0 |           0
datanode2    | 2020-01-14 21:45:21.632407+08 |       0 |    11901 |      387 |  12.1313 |   4.55544 |
3.1968 |    45177.2 |           0 |           0 |           0 |           0 |           0 |           0
datanode3    | 2020-01-14 21:58:14.823317+08 |       0 |    11898 |      390 |  .378205 |       .99 |
48 |    23353.6 |           0 |           0 |           0 |           0 |           0 |           0
datanode3    | 2020-01-14 21:47:50.665028+08 |       0 |    11901 |      387 |  1.07494 |       1.19 |
0 |    15506.4 |           0 |           0 |           0 |           0 |           0 |           0
datanode3    | 2020-01-14 21:51:21.720117+08 |       0 |    11903 |      385 |  10.2795 |       3.11 |
0 |    11031.2 |           0 |           0 |           0 |           0 |           0 |           0
coordinator1 | 2020-01-14 21:42:59.121945+08 |       0 |    12020 |      268 | .0857143 | .0699301
|         0 |    6579.02 |           0 |           0 |           0 |           0 |           0 |           0
coordinator1 | 2020-01-14 21:41:49.042646+08 |       0 |    12020 |      268 |  20.9039 |   11.3786 |
6042.76 |    57903.7 |           0 |           0 |           0 |           0 |           0 |           0
coordinator1 | 2020-01-14 21:41:09.007652+08 |       0 |    12020 |      268 | .0446429 |    .03996 |
0 |    1109.29 |           0 |           0 |           0 |           0 |           0 |           0
(18 rows)
```

12.6.4 实时 TopSQL

数据库用户执行查询作业时，对处于Active状态的SQL查询，系统提供了Query级别和算子级别的资源监控实时视图用来查询执行代价大于**resource_track_cost**的实时TopSQL。资源监控实时视图记录了作业运行时的资源使用情况(包括内存、下盘、CPU时间、IO等)以及性能告警信息。您可以根据这些信息评估查询是否存在性能瓶颈或对集群的整体性能造成影响。

默认开启query级别的资源监控功能，如果要开启query级别和算子级别资源监控功能，需将**resource_track_level**设置为operator。支持实时资源监控的作业类型为：

- 优化器估算的执行代价大于等于**resource_track_cost**的作业。使用explain语句可以查询语句的执行代价。
- SELECT, INSERT, UPDATE, DELETE和CREATE TABLE AS语句，包括使用explain analyze和explain performance来执行的场景。query级别视图还额外支持fetch语法。

资源监控实时视图如下表所示，您可以像查询数据库表一样查询这些视图，普通用户仅支持查询以gs为前缀的视图：

表 12-4 资源监控实时视图

监控级别	节点范围	查询视图	视图说明
Query级别	当前CN	GS_WLM_SESSION_STATISTICS	查询当前CN的实时资源。
	所有CN	PGXC_WLM_SESSION_STATISTICS	查询所有CN的实时资源。
算子级别	当前CN	GS_WLM_OPERATOR_STATISTICS	查询当前CN作业算子执行实时资源信息。
	所有CN	PGXC_WLM_OPERATOR_STATISTICS	查询所有CN作业算子执行实时资源信息。

说明

- 对外接口通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持登录到集群的某个CN查询以gs为前缀的视图。
- 查询此类实时视图时，因需要获取作业运行实时资源使用情况，会有一定的网络延时。
- 实例故障时，实时TopSQL视图有可能记录不全。
- 实时TopSQL中能够记录的SQL语句的规格是：
  - 不记录数据定义语句，例如CREATE、ALTER、DROP、GRANT、REVOKE和VACUUM；
  - 记录数据操作语句，例如SELECT、INSERT、UPDATE和DELETE；
  - 记录函数与存储过程的调用入口语句与内部的语句，但是当函数与存储过程中含有循环体时不记录其循环体内部的语句，记录循环体外部的语句；
  - 不记录匿名块中的语句；
  - 记录事务块中的语句，但是当事务块中含有循环体时，不记录其循环体内部的语句，记录循环体外部的语句；
  - 记录游标语句。

典型的实时 TopSQL 查询

此处以查询在当前CN上执行的SQL语句为例，列举一些典型的查询实时TopSQL资源使用信息及性能运行信息的SQL语句。如果需要查询所有CN上执行的SQL语句，请将查询语句中视图名的前缀gs替换为pgxc。

- Query级别查询。此处提供的每条查询语句，默认第一列为query（即当前查询的SQL语句）。
  - 查询SQL语句执行的开始时间，结束时间，实际执行时间，执行状态信息。时间单位为：ms。

    ```
    SELECT query,start_time,finish_time,duration,status FROM gs_wlm_session_history ORDER BY start_time DESC;
    ```

  - 查询SQL语句在所有DN上的最大内存峰值，语句执行过程中的内存使用平均值，语句在各DN间的内存使用倾斜率。内存单位为：MB。

    ```
    SELECT query,max_peak_memory,average_peak_memory,memory_skew_percent FROM gs_wlm_session_statistics ORDER BY start_time DESC;
    ```

  - 查询SQL语句在所有DN上的下盘信息，所有DN上下盘的最大数据量，所有DN上下盘的平均数据量，DN间下盘倾斜率。数据量单位为：MB。

```
SELECT query,spill_info,max_spill_size,average_spill_size,spill_skew_percent FROM
gs_wlm_session_statistics ORDER BY start_time DESC;
```

  - 查询SQL语句在所有DN上的最大执行时间，在各DN上的执行时间倾斜率。时间单位为：ms。
    ```
    SELECT query,max_dn_time,dntime_skew_percent FROM gs_wlm_session_statistics ORDER BY
    start_time DESC;
    ```
  - 查询SQL语句在所有DN上的最大CPU时间，在所有DN上的CPU总时间，在DN间的CPU时间倾斜率。时间单位：ms。
    ```
    SELECT query,max_cpu_time,total_cpu_time,cpu_skew_percent FROM gs_wlm_session_statistics
    ORDER BY start_time DESC;
    ```
  - 查询SQL语句在所有DN上的每秒最大IO峰值，在所有DN上的每秒平均IO峰值，在DN间的IO倾斜率。IO单位：列存单位是次/s，行存单位是万次/s。
    ```
    SELECT query,max_peak_iops,average_peak_iops,iops_skew_percent FROM
    gs_wlm_session_statistics ORDER BY start_time DESC;
    ```

- 算子级别查询。此处提供的每条查询语句，默认第一列为plan_node_name（即对应于plan_node_id的算子的名称）。
  - 当前算子处理第一条数据的开始时间，到结束时候的总执行时间(单位为：ms)，以及执行状态。
    ```
    SELECT plan_node_name,start_time,duration,status FROM gs_wlm_operator_statistics ORDER BY
    start_time DESC;
    ```
  - 当前算子在所有DN上的最大内存峰值，在所有DN上的平均内存峰值，在各DN内存使用倾斜率。内存单位为：MB。
    ```
    SELECT plan_node_name,max_peak_memory,average_peak_memory,memory_skew_percent
    FROM gs_wlm_operator_statistics ORDER BY start_time DESC;
    ```
  - 当前算子在所有DN上下盘最大数据量，所有DN上下盘平均数据量，DN间下盘倾斜率。数据量单位为：MB
    ```
    SELECT plan_node_name,max_spill_size,average_spill_size,spill_skew_percent FROM
    gs_wlm_operator_statistics ORDER BY start_time DESC;
    ```
  - 该算子在所有DN上的最大执行时间，在所有DN上的总执行时间，各DN间执行时间的倾斜率。时间单位为：ms。
    ```
    SELECT plan_node_name,max_cpu_time,total_cpu_time,cpu_skew_percent FROM
    gs_wlm_operator_statistics ORDER BY start_time DESC;
    ```

注意事项

- 重分布过程中的作业不统计。
- 资源监控实时视图通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持查询以gs为前缀的视图。
- 查询实时视图时，因需要获取作业运行实时资源使用情况，会有一定的网络延时
- 查询如**表12-4**所示的实时视图时，因需要获取作业运行实时资源使用情况，会有一定的网络延时。
- 单独查询实时CPU信息，可以使用视图**GS_SESSION_CPU_STATISTICS**；单独查询内存信息可以使用视图**GS_SESSION_MEMORY_STATISTICS**；视图**GS_WLM_SESSION_STATISTICS**字段包含GS_SESSION_CPU_STATISTICS和GS_SESSION_MEMORY_STATISTICS查询字段结果。

12.6.5 历史 TopSQL

当作业运行结束时，能够对历史的作业进行信息追溯，展现过去态作业的资源使用情况(包括内存、下盘、CPU时间、IO等)和运行状态信息(包括报错、终止、异常等)以及性能告警信息。系统提供了query级别和算子级别的资源监控历史视图用于查询执行代价大于**resource_track_cost**的历史TopSQL。

默认开启query级别的资源监控功能，如果要开启query级别和算子级别资源监控功能，需将**resource_track_level**设置为operator。支持历史资源监控的作业类型为：

- 优化器估算的执行代价大于等于**resource_track_cost**的作业。使用explain语句可以查询语句的执行代价。
- 资源监控实时视图中记录的作业结束时的执行时间大于或等于**resource_track_duration**的作业。

资源监控历史视图如下表所示，您可以像查询数据库表一样查询这些视图，普通用户仅支持查询以gs为前缀的视图：

表 12-5 资源监控历史视图

监控级别	节点范围	查询视图	视图功能
Query级别	当前CN	GS_WLM_SESSION_HISTORY	查询执行结束作业的负载记录。
	所有CN	PGXC_WLM_SESSION_HISTORY	查询执行结束作业的负载记录。
算子级别	当前CN	GS_WLM_OPERATOR_HISTORY	查询执行结束的作业算子的资源信息。
	所有CN	PGXC_WLM_OPERATOR_HISTORY	查询执行结束的作业算子的资源信息。

说明

- 对外接口通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持登录到集群的某个CN查询以gs为前缀的视图。
- 对于gs_wlm_session_info，gs_wlm_operator_info，pgxc_wlm_session_info以及pgxc_wlm_operator_info视图只支持在连接postgres数据库时查询。
- 实例故障时，历史TopSQL视图有可能记录不全。
- 历史TopSQL能够记录的SQL语句的规格与实时TopSQL能够记录的SQL语句的规格一致。请参考**实时TopSQL中能够记录的SQL语句的规格**。

存储在history视图（如**表12-5**）中的资源监控信息，每隔3分钟会被归档到相应的info视图（如**表12-6**）中，归档后history视图中的记录会被清除。info视图只支持在连接postgres数据库时查询。

表 12-6 资源监控历史信息的归档视图

监控级别	节点范围	查询视图	视图功能
Query级别	当前CN	GS_WLM_SESSION_INFO	查询执行结束作业的负载记录。
	所有CN	PGXC_WLM_SESSION_INFO	查询执行结束作业的负载记录。

监控级别	节点范围	查询视图	视图功能
算子级别	当前CN	GS_WLM_OPERATOR_INFO	查询执行结束的作业算子的资源信息。
	所有CN	PGXC_WLM_OPERATOR_INFO	查询执行结束的作业算子的资源信息。

说明

enable_resource_record开关打开后，会引起存储空间膨胀及轻微性能影响，不用时请关闭。

典型的历史 TopSQL 查询

此处以查询在当前CN上三分钟以内执行结束的SQL语句为例。如果需要查询所有CN上执行的SQL语句，请将查询语句中视图名称的前缀gs替换为pgxc。如果需要查询三分钟及以上执行结束的SQL语句，请将查询语句中视图名称的后缀history替换为info。

- Query级别查询。此处提供的每条查询语句，默认第一列为query（即当前查询的SQL语句）。
  - 查询SQL语句执行的开始时间，结束时间，实际执行时间,执行状态信息。时间单位为：ms。
    ```
    SELECT query,start_time,finish_time,duration,status FROM gs_wlm_session_history ORDER BY start_time DESC;
    ```
  - 查询SQL语句在所有DN上的最大内存峰值，语句执行过程中的内存使用平均值，语句在各DN间的内存使用倾斜率。内存单位为：MB。
    ```
    SELECT query,max_peak_memory,average_peak_memory,memory_skew_percent FROM gs_wlm_session_history ORDER BY start_time DESC;
    ```
  - 查询SQL语句在所有DN上的下盘信息，所有DN上下盘的最大数据量，所有DN上下盘的平均数据量，DN间下盘倾斜率。数据量单位为：MB。
    ```
    SELECT query,spill_info,max_spill_size,average_spill_size,spill_skew_percent FROM gs_wlm_session_history ORDER BY start_time DESC;
    ```
  - 查询SQL语句在所有DN上的最大执行时间，平均执行时间，在各DN上的执行时间倾斜率。时间单位为：ms。
    ```
    SELECT query,max_dn_time,average_dn_time,dntime_skew_percent FROM gs_wlm_session_history ORDER BY start_time DESC;
    ```
  - 查询SQL语句在所有DN上的最大CPU时间，在所有DN上的CPU总时间，在DN间的CPU时间倾斜率。时间单位：ms。
    ```
    SELECT query,max_cpu_time,total_cpu_time,cpu_skew_percent FROM gs_wlm_session_history ORDER BY start_time DESC;
    ```
  - 查询SQL语句在所有DN上的每秒最大IO峰值，在所有DN上的每秒平均IO峰值，在DN间的IO倾斜率。IO单位：列存单位是次/s，行存单位是万次/s。
    ```
    SELECT query,max_peak_iops,average_peak_iops,iops_skew_percent FROM gs_wlm_session_history ORDER BY start_time DESC;
    ```
- 算子级别查询。提供的每条查询语句，默认第一列为plan_node_name（即对应于plan_node_id的算子的名称）。
  - 当前算子处理第一条数据的开始时间，到结束时候的总执行时间，以及执行状态。时间单位为：ms。
    ```
    SELECT plan_node_name,start_time,duration FROM gs_wlm_operator_history ORDER BY start_time DESC;
    ```
  - 当前算子在所有DN上的最大内存峰值，在所有DN上的平均内存峰值，在各DN内存使用倾斜率。内存单位为：MB。

```
SELECT plan_node_name,max_peak_memory,average_peak_memory,memory_skew_percent
FROM gs_wlm_operator_history ORDER BY start_time DESC;
```

- 当前算子在所有DN上下盘最大数据量，所有DN上下盘平均数据量，DN间下盘倾斜率。数据量单位为：MB

```
SELECT plan_node_name,max_spill_size,average_spill_size,spill_skew_percent FROM
gs_wlm_operator_history ORDER BY start_time DESC;
```

- 该算子在所有DN上的最大执行时间，在所有DN上的总执行时间，各DN间执行时间的倾斜率。时间单位为：ms。

```
SELECT plan_node_name,max_cpu_time,total_cpu_time,cpu_skew_percent FROM
gs_wlm_operator_history ORDER BY start_time DESC;
```

注意事项

- 对于由于FATAL、PANIC错误导致查询异常结束时，状态信息列只显示aborted，无法记录详细异常信息。
- 对于查询解析，优化阶段的状态信息则无法监控。
- 资源监控历史视图通过不同的前缀(gs与pgxc)来区分单CN查询视图以及集群级别查询视图。普通用户仅支持查询以gs为前缀的视图。
- 对于资源监控历史视图信息，由于预设内存的限制，内存中能够保留的数据记录数量是有限的。实时的查询在结束后会导入到历史相关的视图中，历史的视图信息当其在内存hash表中记录的时间超过3分钟时进行自动清理。关于记录上限，对于query级别视图，当新的需要记录的查询超过内存约束记录数上限时，则当前查询无法记录，下条查询重新进行规则判断；在每个CN上，记query级别实时查询视图在内存中可记录的最大数为max_session_realt_num(当前系统值为15240)，历史视图在内存中可记录的最大数为max_session_hist_num(当前系统值为18995)；业务系统单条查询的平均执行时间为run_time(s)以上，则在每个CN上，实时视图允许客户执行作业的最大并发数：num_realt_active = max_session_realt_num；历史视图允许客户执行作业的最大并发数：num_hist_active = max_session_hist_num/(180/run_time)。
- 对于算子级别视图，当需要记录的查询的plan_node数量加上当前内存中已有的记录数量超过内存约束记录数上限时，则当前查询的所有算子节点不记录，下条查询重新按照算子规则判定。在每个CN上，记算子级别视图在内存中可记录的最大实时和历史记录数分别为max_oper_realt_num(当前系统值为93622), max_oper_hist_num(187245)；记当前用户业务系统的平均每个查询的节点数为num_plan_node, 则在每个CN上，实时视图允许客户执行作业的最大并发数: num_realt_active = max_oper_realt_num/num_plan_node；历史视图允许客户执行作业的最大并发数：num_hist_active = max_oper_hist_num/(180/run_time)/num_plan_node。

12.6.6 TopSQL 查询示例

本章节以查询TPC-DS样例数据的作业为例，演示如何查看**实时TopSQL**和**历史TopSQL**。

配置集群参数

查询TopSQL资源监控信息之前，需要先配置相关的GUC参数，以便能查询到作业的资源监控历史信息或归档信息。步骤如下：

1. 登录GaussDB(DWS)管理控制台。
2. 在“集群管理”页面，找到所需要的集群，单击集群名称，进入集群详情页面。
3. 单击“参数修改”标签页，可以看到当前集群的参数值。

4. 修改参数**resource_track_duration**值为合适的值，单击“保存”按钮进行保存。

   说明

   enable_resource_record开关打开后，会引起存储空间膨胀及轻微性能影响，不用时请关闭。

5. 返回集群管理页面，单击右上角的刷新按钮，等待集群参数配置完成。

TopSQL 查询示例

本示例以TPC-DS样例数据为例。

步骤1 打开SQL客户端工具，连接到您的数据库。

步骤2 使用explain语句查询所要执行的SQL语句的预估代价，从而可以确定该SQL语句是否会进行资源监控。

默认执行代价大于**resource_track_cost**（默认值为100000）的查询才会进行资源监控，用户才可以查询到相关的资源监控信息。

例如，执行如下语句查询该SQL语句的预估执行代价：

```
SET CURRENT_SCHEMA = tpcds;
EXPLAIN WITH customer_total_return AS
( SELECT sr_customer_sk as ctr_customer_sk,
sr_store_sk as ctr_store_sk,
sum(SR_FEE) as ctr_total_return
FROM store_returns, date_dim
WHERE sr_returned_date_sk = d_date_sk AND d_year =2000
GROUP BY sr_customer_sk, sr_store_sk )
SELECT  c_customer_id
FROM customer_total_return ctr1, store, customer
WHERE ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
FROM customer_total_return ctr2
WHERE ctr1.ctr_store_sk = ctr2.ctr_store_sk)
AND s_store_sk = ctr1.ctr_store_sk
AND s_state = 'TN'
AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY c_customer_id
limit 100;
```

查询结果如下所示，第一行的E-costs列的值即为当前语句的预估代价。

图 12-7 explain 查询结果

```
id |                                operation                                | E-rows | E-width | E-costs
---+-------------------------------------------------------------------------+--------+---------+---------
 1 | ->  Row Adapter                                                         |      6 |      20 | 153.06
 2 |    ->  Vector Limit                                                     |      6 |      20 | 153.06
 3 |       ->  Vector Streaming (type: GATHER)                               |      6 |      20 | 153.06
 4 |          ->  Vector Limit                                               |      6 |      20 | 152.84
 5 |             ->  Vector Sort                                             |      6 |      20 | 152.84
 6 |                ->  Vector Hash Join (7,26)                              |      6 |      20 | 152.83
 7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |      6 |       4 | 134.57
 8 |                      ->  Vector Hash Join (9,18)                        |      6 |       4 | 134.46
 9 |                         ->  Vector Hash Join (10,11)                    |      1 |      44 | 97.33
10 |                            ->  CStore Scan on store                     |      1 |       4 | 60.23
11 |                            ->  Vector Subquery Scan on ctr1             |      6 |      40 | 37.07
12 |                               ->  Vector Hash Aggregate                 |      6 |      54 | 37.06
13 |                                  ->  Vector Streaming(type: REDISTRIBUTE)|      6 |      22 | 37.04
14 |                                     ->  Vector Hash Join (15,17)        |      6 |      22 | 37.00
15 |                                        ->  Vector Streaming(type: BROADCAST)|      6 |       4 | 18.74
16 |                                           ->  CStore Scan on date_dim   |      1 |       4 | 18.06
17 |                                        ->  CStore Scan on store_returns |     60 |      26 | 18.02
18 |                         ->  Vector Hash Aggregate                       |      6 |      68 | 37.09
19 |                            ->  Vector Subquery Scan on ctr2             |      6 |      36 | 37.07
20 |                               ->  Vector Hash Aggregate                 |      6 |      54 | 37.06
21 |                                  ->  Vector Streaming(type: REDISTRIBUTE)|      6 |      22 | 37.04
22 |                                     ->  Vector Hash Join (23,25)        |      6 |      22 | 37.00
23 |                                        ->  Vector Streaming(type: BROADCAST)|      6 |       4 | 18.74
24 |                                           ->  CStore Scan on date_dim   |      1 |       4 | 18.06
25 |                                        ->  CStore Scan on store_returns |     60 |      26 | 18.02
26 |                   ->  CStore Scan on customer                           |     60 |      24 | 18.02
(26 rows)
```

本示例为了演示TopSQL的资源监控功能，需要将resource_track_cost参数设置为比explain查询结果中的预估代价小的一个值，例如100，设置方法请参见**resource_track_cost**。

说明

在完成本示例后，仍然要将**resource_track_cost**设置为原始的默认值100000或者一个比较合理的值，否则参数值太小会影响数据库性能。

步骤3 执行SQL语句。

```
SET CURRENT_SCHEMA = tpcds;
WITH customer_total_return AS
(SELECT sr_customer_sk as ctr_customer_sk,
sr_store_sk as ctr_store_sk,
sum(SR_FEE) as ctr_total_return
FROM store_returns,date_dim
WHERE sr_returned_date_sk = d_date_sk
AND d_year =2000
GROUP BY sr_customer_sk ,sr_store_sk)
SELECT  c_customer_id
FROM customer_total_return ctr1, store, customer
WHERE ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
FROM customer_total_return ctr2
WHERE ctr1.ctr_store_sk = ctr2.ctr_store_sk)
AND s_store_sk = ctr1.ctr_store_sk
AND s_state = 'TN'
AND ctr1.ctr_customer_sk = c_customer_sk
ORDER BY c_customer_id
limit 100;
```

步骤4 在SQL语句执行期间，查询该条SQL语句在当前CN上的Memory峰值的实时信息。

```
SELECT query,max_peak_memory,average_peak_memory,memory_skew_percent FROM
gs_wlm_session_statistics ORDER BY start_time DESC;
```

含义：查询**query级别**的SQL语句的Memory峰值**实时信息**（语句在所有DN上的每秒最大Memory峰值，在所有DN上的每秒平均Memory峰值，在DN间的Memory倾斜率）

实时TopSQL资源监控信息的更多查询示例，请参见**典型的实时TopSQL查询**。

步骤5 等待**步骤3**中的SQL执行完成，然后查询该语句执行期间的资源监控历史信息。

```
select query,start_time,finish_time,duration,status from gs_wlm_session_history order by start_time desc;
```

含义：查询**query级别**的SQL语句执行期间的**历史信息**（语句执行的开始时间，结束时间，实际执行时间，执行状态），时间单位为ms。

历史TopSQL资源监控信息的更多查询示例，请参见**典型的历史TopSQL查询**。

步骤6 等待**步骤3**中的SQL执行结束的三分钟后，在info视图中查询该语句的资源监控历史信息。

如果设置参数enable_resource_record为“on”，且**步骤3**中SQL的执行时间不小于resource_track_duration所设置的值，该条语句的历史信息将会在三分钟后被归档到gs_wlm_session_info视图中。

对于info视图，只支持在连接postgres数据库时查询。因此，请切换为连接postgres数据库后，再执行以下语句进行查询：

```
select query,start_time,finish_time,duration,status from gs_wlm_session_info order by start_time desc;
```

----结束

13 用户自定义函数

13.1 PL/Java 语言函数

使用GaussDB(DWS)数据库的PL/Java函数，用户可以使用自己喜欢的Java IDE编写Java方法，并将包含这些方法的jar文件安装到GaussDB(DWS)数据库中，然后使用该方法。GaussDB(DWS) PL/Java基于开源Greenplum PL/Java 1.4.0开发，PL/Java所使用的JDK版本为1.8.0_201。

使用限制

Java UDF可以实现一些java逻辑计算，强烈建议不要在Java UDF中封装业务

- 强烈建议不要在Java函数中使用任何方式连接数据库，包括但不限于JDBC。
- 暂不支持的数据类型：除**表13-1**内容之外的数据类型，包括自定义类型，复杂数据类型（Java Array类及派生类）。
- 暂不支持UDAF，UDTF。

示例

使用PL/Java函数时，需要首先将Java方法的实现打包为jar包并且部署到数据库中，然后使用数据库管理员账号创建函数，考虑兼容性问题，请使用1.8.0_201版本的JDK进行编译。

步骤1 编译jar包。

Java方法的实现和出包可以借助IDE来实现，以下是一个通过命令行来进行编译和出包的简单的示例，通过这个简单示例可以创建出一个包含单个方法的jar包文件。

首先，编写一个Example.java文件，在此文件中实现子字符串大写转换的方法，本例中类名为Example，方法名为upperString，内容如下：

```
public class Example
{
    public static String upperString (String text, int beginIndex, int endIndex)
    {
        return text.substring(beginIndex, endIndex).toUpperCase();
    }
}
```

然后，创建manifest.txt清单文件，文件内容如下：

```
Manifest-Version: 1.0
Main-Class: Example
Specification-Title: "Example"
Specification-Version: "1.0"
Created-By: 1.6.0_35-b10-428-11M3811
Build-Date: 08/14/2018 10:09 AM
```

其中，Manifest-Version定义了manifest文件的版本，Main-Class定义了jar文件的入口类，Specification-Title和Specification-Version属于包的扩展属性，Specification-Title定义了扩展规范的标题，Specification-Version定义了扩展规范的版本，Created-By声明了该文件的生成者，Build-Date声明了该文件构建日期。

最后，编译java文件并打包得到javaudf-example.jar

```
javac Example.java
jar cfm javaudf-example.jar manifest.txt Example.class
```

须知

jar包的命名规则应符合JDK命名要求，如果含有非法字符，在部署或者使用函数时将出错。

步骤2 部署jar包。

Jar包首先需要放置到OBS服务器中，放置方法具体请参见*《对象存储服务控制台指南》*的上传文件章节。接着创建访问密钥AK/SK，获取访问密钥的具体步骤，请参见《数据仓库服务用户指南》中的“创建访问密钥（AK和SK）”章节。登录数据库运行gs_extend_library函数，将文件导入到GaussDB(DWS)中：

```
SELECT gs_extend_library('addjar', 'obs://bucket/path/javaudf-example.jar
accesskey=access_key_value_to_be_replaced secretkey=secret_access_key_value_to_be_replaced
region=region_name libraryname=example');
```

gs_extend_library函数如何使用请参见 **管理jar包和文件**。函数中的AK/SK值，请用户根据实际获取值替换。region_name请用户根据实际所在的区域名称替换。

步骤3 使用PL/Java函数。

首先，使用拥有sysadmin权限的数据库用户（例如：dbadmin）登录数据库并创建java_upperstring函数如下：

```
CREATE FUNCTION java_upperstring(VARCHAR, INTEGER, INTEGER)
   RETURNS VARCHAR
   AS 'Example.upperString'
LANGUAGE JAVA;
```

说明

- 函数java_upperstring中定义的数据类型为GaussDB(DWS)的数据类型。该数据类型需要和**步骤1**中java定义的方法upperString中数据类型一一对应。GaussDB(DWS)与Java数据类型的对应关系，请参见**表13-1**。
- AS子句用于指定该函数所调用的Java方法的类名和static方法名，格式为“类名.方法名”。该字段需要和**步骤1**中java定义的类名和方法名一致。当前示例中未指定package，若使用中有指定package，在使用CREATE FUNCTION时需要指定完整类名。
- 使用PL/Java函数时，LANGUAGE字段应指定为JAVA。
- CREATE FUNCTION更多说明，请参见**创建函数**。

然后，执行java_upperstring函数：

```
SELECT java_upperstring('test', 0, 1);
```

得到预期结果为：

```
 java_upperstring
---------------------
 T
(1 row)
```

步骤4 授权普通用户使用PL/Java函数。

创建普通用户，名称为udf_user，密码为'GAUSS@123'。

```
CREATE USER udf_user PASSWORD 'GAUSS@123';
```

授权普通用户udf_user对java_upperstring函数的使用权限。注意，此处需要把函数所在模式和函数的使用权限同时赋予给用户，用户才可以使用此函数。

```
GRANT ALL PRIVILEGES ON SCHEMA public TO udf_user;
GRANT ALL PRIVILEGES ON FUNCTION java_upperstring(VARCHAR, INTEGER, INTEGER) TO udf_user;
```

以普通用户udf_user登录数据库。

```
SET SESSION SESSION AUTHORIZATION udf_user PASSWORD 'GAUSS@123';
```

执行java_upperstring函数：

```
SELECT public.java_upperstring('test', 0, 1);
```

得到预期结果为：

```
 java_upperstring
---------------------
 T
(1 row)
```

步骤5 删除函数。

如果不再使用该函数可以进行删除：

```
DROP FUNCTION java_upperstring;
```

步骤6 卸载jar包。

使用gs_extend_library函数卸载jar包：

```
SELECT gs_extend_library('rmjar', 'libraryname=example');
```

----结束

SQL 定义与使用

- **管理jar包和文件**

  拥有sysadmin权限的数据库用户可以使用gs_extend_library函数来部署、查看和删除数据库中的jar包，函数语法如下：

  ```
  SELECT gs_extend_library('[action]', '[operation]');
  ```

说明

- **action**表明操作动作，值可以为：
  - ls表示查看数据库中jar包，会对各节点的文件进行MD5值一致性检查。
  - addjar表示将OBS服务器中的jar包部署到数据库中。
  - rmjar表示将数据库中的jar包删除。
- **operation**表明操作字符串，格式为：

  obs://[bucket]/[source_filepath] accesskey=[accesskey] secretkey=[secretkey] region=[region] libraryname=[libraryname]
  - bucket: obs文件所属桶名，不可缺省。
  - source_filepath: obs服务器上的文件路径，仅支持jar文件。
  - accesskey: obs服务获得的accesskey，不可缺省。
  - secret_key: obs服务获得的secretkey，不可缺省。
  - region: 自定义函数jar包所存放的OBS桶所属的区域，不可缺省。
  - libraryname: 自定义库名，此自定义命名用于GaussDB(DWS)内对jar文件的调用。当action为addjar和rmjar时，该参数不可缺省，当action为ls时，该参数可以缺省。注意，自定义库名不允许含有/|;&$<>\'{}"()[]~*?!等字符。

- **创建函数**

  PL/Java函数通过CREATE FUNCTION语法创建，并且定义为LANGUAGE JAVA，且包含RETURNS和AS子句。
  - CREATE FUNCTION时指定所创建函数的名称，以及参数类型；
  - RETURNS子句用于指定该函数的返回类型；
  - AS子句与用于指定该函数所调用的Java方法的类名和static方法名，如果需要向Java方法传递NULL值作为入参，还需要指定该参数类型所对应的Java封装类名（详见**NULL值处理**）。
  - 更多语法说明，请参见CREATE FUNCTION。

```
CREATE [ OR REPLACE ] FUNCTION function_name
( [ { argname [ argmode ] argtype [ { DEFAULT | := | = } expression ]} [, ···] ])
[ RETURNS rettype [ DETERMINISTIC ] ]
LANGAUGE JAVA
[
   { IMMUTABLE | STATBLE | VOLATILE }
   | [ NOT ] LEAKPROOF
   | WINDOW
   | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT |STRICT }
   | {[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER | AUTHID DEFINER |
AUTHID CURRENT_USER}
   | { FENCED }
   | COST execution_cost
   | ROWS result_rows
   | SET configuration_parameter { {TO |=} value | FROM CURRENT}
] [···]
{
   AS 'class_name.method_name' ( { argtype } [, ···] )
}
```

- **使用函数**

  执行时，PL/Java会根据jar包名的字母序列，在所有部署的jar包中寻找函数指定的Java类，并调用首次找到的类中函数所指定的Java方法，并返回调用结果。

- **删除函数**

  PL/Java函数通过DROP FUNCTION语法删除函数。更多语法说明，请参见DROP FUNCTION。

```
DROP FUNCTION [ IF EXISTS ] function_name [ ( [ {[ argmode ] [ argname ] argtype} [, ...] ] )
[ CASCADE | RESTRICT ] ];
```

需要注意的是，如果所删除的函数为重载函数（详见**重载函数**），则删除时需要指明该函数的参数类型，如为非重载函数，则可直接指定函数名进行删除。

- **函数授权**

  非sysadmin户无法创建PL/Java函数，sysadmin用户可以赋予其他类型用户使用函数的权限。更多语法说明，请参见GRANT。

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION {function_name ( [ {[ argmode ] [ arg_name ] arg_type} [, ...] ] )} [, ...]
        | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
    [ WITH GRANT OPTION ];
```

基本数据类型映射关系

表 13-1 PL/Java 默认数据类型映射关系

GaussDB(DWS)	Java
BOOLEAN	boolean
"char"	byte
bytea	byte[]
SMALLINT	short
INTEGER	int
BIGINT	long
FLOAT4	float
FLOAT8	double
CHAR	java.lang.String
VARCHAR	java.lang.String
TEXT	java.lang.String
name	java.lang.String
DATE	java.sql.Timestamp
TIME	java.sql.Time (stored value treated as local time)
TIMETZ	java.sql.Time
TIMESTAMP	java.sql.Timestamp
TIMESTAMPTZ	java.sql.Timestamp

数组类型处理

GaussDB(DWS)支持基础数组类型的转换，只需要在创建函数时在数据类型后追加 [] 即可，例如：

```
CREATE FUNCTION java_arrayLength(INTEGER[])
    RETURNS INTEGER
    AS 'Example.getArrayLength'
LANGUAGE JAVA;
```

Java代码类似于：

```
public class Example
{
    public static int getArrayLength(Integer[] intArray)
    {
        return intArray.length;
    }
}
```

那么下面的调用的语句后：

```
SELECT java_arrayLength(ARRAY[1, 2, 3]);
```

得到预期结果应该如下所示：

```
java_arrayLength
---------------------
3
(1 row)
```

NULL 值处理

对于默认与Java的简单类型进行映射转换的那些GaussDB(DWS)数据类型，是无法处理NULL值的，如果希望在Java方法里能够获得并处理从GaussDB(DWS)中传入的NULL值，可以使用Java的封装类，并通过以下方式在AS子句中指定该Java封装类：

```
CREATE FUNCTION java_countnulls(INTEGER[])
    RETURNS INTEGER
    AS 'Example.countNulls(java.lang.Integer[])'
LANGUAGE JAVA;
```

Java代码类似于：

```
public class Example
{
    public static int countNulls(Integer[] intArray)
    {
        int nullCount = 0;
        for (int idx = 0; idx < intArray.length; ++idx)
        {
            if (intArray[idx] == null)
            nullCount++;
        }
        return nullCount;
    }
}
```

那么下面的调用的语句后：

```
SELECT java_countNulls(ARRAY[null, 1, null, 2, null]);
```

得到的预期结果应该如下所示：

```
java_countNulls
--------------------
3
(1 row)
```

重载函数

PL/Java支持重载函数，因此可以创建同名函数，或者调用Java代码中的重载方法。步骤如下：

步骤1 创建重载函数

例如，在Java中可以实现两个方法名相同，输入参数类型不同的方法dummy(int) 和dummy(String)

```
public class Example
{
    public static int dummy(int value)
    {
        return value*2;
    }
    public static String dummy(String value)
    {
        return value;
    }
}
```

并在GaussDB(DWS)中创建两个同名函数分别指定为上述两个方法：

```
CREATE FUNCTION java_dummy(INTEGER)
    RETURNS INTEGER
    AS 'Example.dummy'
LANGUAGE JAVA;

CREATE FUNCTION java_dummy(VARCHAR)
    RETURNS VARCHAR
    AS 'Example.dummy'
LANGUAGE JAVA;
```

步骤2 调用重载函数

在调用重载函数时，GaussDB(DWS)会根据输入的参数类型去调用匹配该类型的Java方法。因此上述两个函数的调用结果如下所示：

```
SELECT java_dummy(5);
 java_dummy
------------------
           10
(1 row)

SELECT java_dummy('5');
 java_dummy
---------------
5
(1 row)
```

需要注意的是，由于GaussDB(DWS)对数据类型存在隐式转换的情况，因此建议在调用重载函数时，指定输入参数的类型，例如：

```
SELECT java_dummy(5::varchar);
 java_dummy
---------------
5
(1 row)
```

此时会优先匹配所指定的参数类型，如果不存在指定参数类型的Java方法，则会对参数进行隐式转换匹配转换后的参数类型对应的Java方法。

```
SELECT java_dummy(5::INTEGER);
 java_dummy
-----------------
10
```

```
(1 row)

DROP FUNCTION java_dummy(INTEGER);

SELECT java_dummy(5::INTEGER);
 java_dummy
----------------
5
(1 row)
```

须知

隐式转换的数据类型包括：

- 可以默认转换为INTEGER类型的包括：SMALLINT
- 可以默认转换为BIGINT类型的包括：SMALLINT, INTEGER
- 可以默认转换为BOOL类型的包括：TINYINT, SMALLINT, INTEGER, BIGINT
- 可以默认转换为TEXT类型的包括：CHAR, NAME, BIGINT, INTEGER, SMALLINT, TINYINT, RAW, FLOAT4, FLOAT8, BPCHAR, VARCHAR, NVARCHAR2, DATE, TIMESTAMP, TIMESTAMPTZ, NUMERIC, SMALLDATETIME
- 可以默认转换为VARCHAR类型的包括：TEXT, CHAR, BIGINT, INTEGER, SMALLINT, TINYINT, RAW, FLOAT4, FLOAT8, BPCHAR, DATE, NVARCHAR2, TIMESTAMP, NUMERIC, SMALLDATETIME

步骤3 删除重载函数

对于重载函数，删除时需要指定函数的参数类型，否则无法删除。

```
DROP FUNCTION java_dummy(INTEGER);
```

----结束

相关 GUC 参数

- **pljava_vmoptions**

  会话级别的GUC参数，该参数用于设置JVM的启动参数，例如：

  ```
  SET pljava_vmoptions='-Xmx64m –Xms2m –XX:MaxMetaspaceSize=8m';
  ```

  pljava_vmoptions可接受的参数包括：

  – JDK8 JVM启动参数

  – JDK8 JVM系统属性参数（以–D开头，如：–Djava.ext.dirs）。

  须知

  不建议用户设置任何包含目录的参数，可能会导致不可预期的行为。

  – 用户自定义参数（以–D开头，如：–Duser.defined.option）

须知

如果所设置的参数不在上述范围内，视为非法参数，会在调用函数时报错。

```
SET pljava_vmoptions=' illegal.option';
SET
SELECT java_dummy(5::int);
ERROR:  UDF Error:cannot use PL/Java before successfully completing its setup.Please check if your pljava_vmoption is set correctly,since we do not ignore illegal parameters.Or check the log for more messages.
```

- **udf_memory_limit**

  系统级别的GUC参数，用于限制每个CN、DN执行UDF可以使用的物理内存量，默认为200MB。可通过修改postgresql.conf文件进行配置，配置后需要重启数据库服务后才可生效。

  须知

  - udf_memory_limit是max_process_memory的一部分。每个CN、DN启动时，会预留(udf_memory_limit - 200MB)内存供UDF Worker进程使用。CN、DN和UDF Worker是不同的进程，但CN、DN自动少用一部分内存，把这部分内存节省下来供UDF Worker进程使用。

    例如：在某DN上把max_process_memory设置为10GB，udf_memory_limit设置为4GB，则此DN最多使用10GB - (4GB - 200MB)=6.2GB内存。即使用户没有执行任何UDF，则此DN也最多只能使用6.2GB内存。默认情况下，udf_memory_limit为200MB。查询pv_total_memory_detail视图时可以发现，process_used_memory永远不会超过max_process_memory - (udf_memory_limit - 200MB)。
  - 一个CN执行最简单的Java UDF函数，使用的物理内存量大约为50MB，用户可以根据自己Java函数的内存使用量和并发度设置此参数。新增此参数后，不再建议用户设置UDFWorkerMemHardLimit和FencedUDFMemoryLimit。
  - 当UDF进程并发度过大，内存超出udf_memory_limit设置值时会导致进程退出等非预期情况，该场景下执行结果可能不可靠，强烈建议根据实际情况进行参数设置，保留足够内存余量。内存严重不足时，甚至可能导致UDF master进程退出，可以查看UDF master日志进行分析，默认的UDF master日志路径在$GAUSSLOG/cm/cm_agent/pg_log下。例如，出现以下日志时就说明内存资源严重不足，导致了UDF master进程退出，需要检查udf_memory_limit参数设置。

    0 [BACKEND] FATAL: poll() failed: Bad address, please check the parameter:udf_memory_limit to make sure there is enough memory.

- **FencedUDFMemoryLimit**

  会话级别的GUC参数，用户限制会话发起的单个Fenced UDF Worker进程的最大虚拟内存使用量，设置方法如下：

  ```
  SET FencedUDFMemoryLimit='512MB';
  ```

  该参数的取值范围为 (150MB, 1G]，当设置大于1G时会立即报错，当设置小于等于150MB时，则会在调用函数时报错。

须知

- FencedUDFMemoryLimit设置为0，表示不控制Fenced UDF Worker的虚拟内存使用量。
- 建议通过设置udf_memory_limit控制Fenced UDF Worker使用的物理内存量。不建议用户使用FencedUDFMemoryLimit，尤其在使用Java UDF时不建议用户设置此参数。但是如果用户非常清楚设置该参数带来的影响，可以参考下列信息进行设置：
  - C UDF worker启动之后，占用的虚拟内存约为200MB，占用的物理内存约为16MB。
  - Java UDF worker启动之后，占用的虚拟内存约为2.5GB，占用的物理内存约为50MB。

异常处理

如果在JVM中发生异常，PL/Java的异常处理机制会将异常时JVM的堆栈信息输出到客户端。

日志

PL/Java使用标准的Java Logger。 因此，用户可以通过如下方式记录日志：

```
Logger.getAnonymousLogger().config( "Time is " + new
Date(System.currentTimeMillis()));
```

初始化的Java Logger类会默认设置为CONFIG级别，对应为GaussDB(DWS)的LOG级别。Java Logger类输出的日志消息都会重定向到GaussDB(DWS)后端，并写入到服务器日志或显示在用户界面上。MPPDB服务器日志将记录LOG、WARNING、ERROR级别的信息，而SQL用户界面将显示WARNING和ERROR级别的日志消息。Java Logger级别与GaussDB(DWS)的日志级别对应关系见下表。

表 13-2 PL/Java 日志级别

java.util.logging.Level	GaussDB(DWS) 日志级别
SERVER	ERROR
WARINING	WARNING
CONFIG	LOG
INFO	INFO
FINE	DEBUG1
FINER	DEBUG2
FINEST	DEBUG3

用户可以通过以下方式更改Java Logger的记录级别。例如通过下面的Java代码修改Java Logger级别为SEVERE，此时再记录WARNING级别的日志时，日志消息（msg）就不会再写入到GaussDB(DWS)日志中。

```
Logger log = Logger.getAnonymousLogger();
Log.setLevel(Level.SEVERE);
log.log(Level.WARNING, msg);
```

安全问题

在GaussDB(DWS)中，PL/Java是一种untrusted语言，PL/Java函数只能由数据库sysdamin用户进行创建，通过GRANT方式赋予其他用户使用权限（详见**函数授权**）。

同时PL/Java控制用户对文件系统的访问权限，不允许用户在Java方法中对大部分的系统文件进行读操作，不允许所有的写、删除和执行操作。

13.2 PL/pgSQL 语言函数

PL/pgSQL类似于Oracle的PL/SQL，是一种可载入的过程语言。

用PL/pgSQL创建的函数可以被用在任何可以使用内建函数的地方。例如，可以创建复杂条件的计算函数并且后面用它们来定义操作符或把它们用于索引表达式。

SQL被大多数数据库用作查询语言。它是可移植的并且容易学习。但是每一个SQL语句必须由数据库服务器单独执行。

这意味着客户端应用必须发送每一个查询到数据库服务器、等待它被处理、接收并处理结果、做一些计算，然后发送更多查询给服务器。如果客户端和数据库服务器不在同一台机器上，所有这些会引起进程间通信并且将带来网络负担。

通过PL/pgSQL，可以将一整块计算和一系列查询分组在数据库服务器内部，这样就有了一种过程语言的能力并且使SQL更易用，同时能节省的客户端/服务器通信开销。

- 客户端和服务器之间的额外往返通信被消除。
- 客户端不需要的中间结果不必被整理或者在服务器和客户端之间传送。
- 多轮的查询解析可以被避免。

PL/pgSQL可以使用SQL中所有的数据类型、操作符和函数。

应用PL/pgSQL创建函数的语法为CREATE FUNCTION。正如前面所说，PL/pgSQL类似于Oracle的PL/SQL，是一种可载入的过程语言。其应用方法与存储过程相似，只是存储过程无返回值，函数有返回值。

14 优化查询性能

14.1 优化查询性能概述

SQL调优的唯一目的是“资源利用最大化”，即CPU、内存、磁盘IO、网络IO四种资源利用最大化。所有调优手段都是围绕资源使用开展的。所谓资源利用最大化是指SQL语句尽量高效，节省资源开销，以最小的代价实现最大的效益。比如做典型点查询的时候，可以用seqscan+filter(即读取每一条元组和点查询条件进行匹配)实现，也可以通过indexscan实现，显然indexscan可以以更小的代价实现相同的效果。

本章主要讲述了查询的分析和改进方法，并且为用户提供了一些常见案例以及错误处理办法。

14.2 分析查询

14.2.1 Query 执行流程

SQL引擎从接受SQL语句到执行SQL语句需要经历的步骤如**图14-1**和**表14-1**所示。其中，红色字体部分为DBA可以介入实施调优的环节。

图 14-1 SQL 引擎执行查询类 SQL 语句的流程

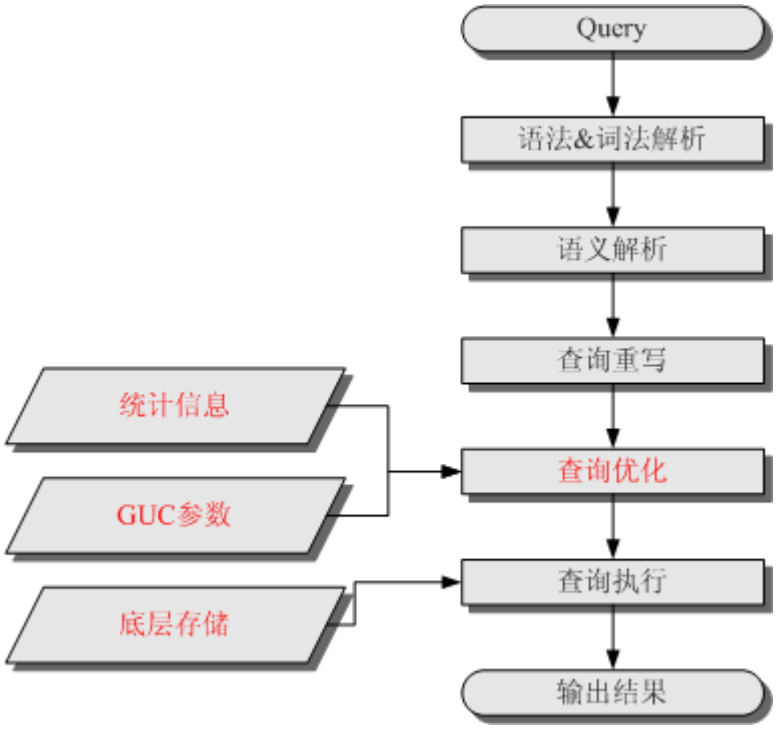


表 14-1 SQL 引擎执行查询类 SQL 语句的步骤说明

步骤	说明
1、语法&词法解析	按照约定的SQL语句规则，把输入的SQL语句从字符串转化为格式化结构(Stmt)。
2、语义解析	将“语法&词法解析”输出的格式化结构转化为数据库可以识别的对象。
3、查询重写	根据规则把“语义解析”的输出等价转化为执行上更为优化的结构。
4、查询优化	根据“查询重写”的输出和数据库内部的统计信息规划SQL语句具体的执行方式，也就是执行计划。统计信息和GUC参数对查询优化（执行计划）的影响，请参见**调优手段之统计信息**和**调优手段之GUC参数**。
5、查询执行	根据“查询优化”规划的执行路径执行SQL查询语句。底层存储方式的选择合理性，将影响查询执行效率。详见**调优手段之底层存储**。

调优手段之统计信息

GaussDB(DWS)优化器是典型的基于代价的优化 (Cost-Based Optimization，简称CBO)。在这种优化器模型下，数据库根据表的元组数、字段宽度、NULL记录比率、distinct值、MCV值、HB值等表的特征值，以及一定的代价计算模型，计算出每一个执行步骤的不同执行方式的输出元组数和执行代价(cost)，进而选出整体执行代价最小/首元组返回代价最小的执行方式进行执行。这些特征值就是统计信息。从上面描述

可以看出统计信息是查询优化的核心输入，准确的统计信息将帮助规划器选择最合适的查询规划，一般来说我们通过analyze语法收集整个表或者表的若干个字段的统计信息，周期性地运行ANALYZE，或者在对表的大部分内容做了更改之后马上运行它是个好习惯。

调优手段之 GUC 参数

查询优化的主要目的是为查询语句选择高效的执行方式。

如下SQL语句:

```
select count(1)
from customer inner join store_sales on (ss_customer_sk = c_customer_sk);
```

在执行customer inner join store_sales的时候，GaussDB(DWS)支持Nested Loop、Merge Join和Hash Join三种不同的Join方式。优化器会根据表customer和表store_sales的统计信息估算结果集的大小以及每种join方式的执行代价，然后对比选出执行代价最小的执行计划。

正如前面所说，执行代价计算都是基于一定的模型和统计信息进行估算，当因为某些原因代价估算不能反映真实的cost的时候，我们就需要通过guc参数设置的方式让执行计划倾向更优规划。

调优手段之底层存储

GaussDB(DWS)的表支持行存表、列存表，底层存储方式的选择严格依赖于客户的具体业务场景。一般来说计算型业务查询场景(以关联、聚合操作为主)建议使用列存表；点查询、大批量UPDATE/DELETE业务场景适合行存表。

对于每种存储方式还有对应的存储层优化手段，这部分会在后续的调优章节深入介绍。

调优手段之 SQL 重写

除了上述干预SQL引擎所生成执行计划的执行性能外，根据数据库的SQL执行机制以及大量的实践发现，有些场景下，在保证客户业务SQL逻辑的前提下，通过一定规则由DBA重写SQL语句，可以大幅度的提升SQL语句的性能。

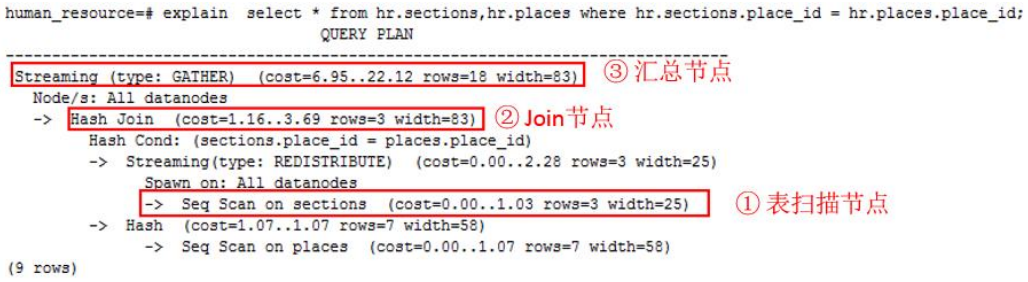
这种调优场景对DBA的要求比较高，需要对客户业务有足够的了解，同时也需要扎实的SQL语句基本功，后续会介绍几个常见的SQL改写场景。

14.2.2 SQL 执行计划概述

SQL执行计划是一个节点树，显示DWS执行一条SQL语句时执行的详细步骤。每一个步骤为一个数据库运算符。

使用EXPLAIN命令可以查看优化器为每个查询生成的具体执行计划。EXPLAIN给每个执行节点都输出一行，显示基本的节点类型和优化器为执行这个节点预计的开销值。如图14-2所示。

图 14-2 SQL 执行计划示例



- 最底层节点是表扫描节点，它扫描表并返回原始数据行。不同的表访问模式有不同的扫描节点类型：顺序扫描、索引扫描等。最底层节点的扫描对象也可能是非表行数据（不是直接从表中读取的数据），如VALUES子句和返回行集的函数，它们有自己的扫描节点类型。
- 如果查询需要连接、聚集、排序、或者对原始行做其它操作，那么就会在扫描节点之上添加其它节点。 并且这些操作通常都有多种方法，因此在这些位置也有可能出现不同的执行节点类型。
- 第一行(最上层节点)是执行计划总执行开销的预计。这个数值就是优化器试图最小化的数值。

执行计划显示格式

DWS对执行计划提供了normal、pretty、summary、run四种显示格式：

- normal：代表使用默认的打印格式。图14-2中即为此显示格式。
- pretty：代表使用DWS改进后的新显示格式。新的格式层次清晰，计划包含了plan node id，性能分析简单直接。如图14-3。
- summary：是在pretty的基础上增加了对打印信息的分析。
- run：在summary的基础上，将统计的信息输出到csv格式的文件中，以便于进一步分析。

图 14-3 pretty 格式执行计划示例

```
postgres=# explain select cjxh, count(1) from dwcjk group by cjxh;
 id |               operation                | E-rows | E-memory | E-width | E-costs
----+----------------------------------------+--------+----------+---------+---------
  1 | ->  Row Adapter                        |      1 |          |      52 | 58.42
  2 |    ->  Vector Streaming (type: GATHER) |      1 |          |      52 | 58.42
  3 |       ->  Vector Hash Aggregate        |      1 | 16MB     |      52 | 58.02
  4 |          ->  CStore Scan on dwcjk      |      1 | 1MB      |      44 | 58.00
(4 rows)
```

通过设置GUC参数explain_perf_mode，可以显示不同格式的执行计划。下文的用例默认显示pretty格式。

执行计划显示信息

除了设置不同的执行计划显示格式外，还可以通过不同的EXPLAIN用法，显示不同详细程度的执行计划信息。常见有如下几种，关于更多用法请参见EXPLAIN语法说明。

- EXPLAIN *statement*： 只生成执行计划，不实际执行。其中statement代表SQL语句。

- EXPLAIN ANALYZE *statement*：生成执行计划，进行执行，并显示执行的概要信息。显示中加入了实际的运行时间统计，包括在每个规划节点内部花掉的总时间(以毫秒计)和它实际返回的行数。
- EXPLAIN PERFORMANCE *statement*：生成执行计划，进行执行，并显示执行期间的全部信息。

为了测量运行时在执行计划中每个节点的开销，EXPLAIN ANALYZE或EXPLAIN PERFORMANCE会在当前查询执行上增加性能分析的开销。在一个查询上运行EXPLAIN ANALYZE或EXPLAIN PERFORMANCE有时会比普通查询明显的花费更多的时间。超支的数量依赖于查询的本质和使用的平台。

因此，当定位SQL运行慢问题时，如果SQL长时间运行未结束，建议通过EXPLAIN命令查看执行计划，进行初步定位。如果SQL可以运行出来，则推荐使用EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看执行计划及其实际的运行信息，以便更精准地定位问题原因。

EXPLAIN PERFORMANCE轻量化执行方式与EXPLAIN PERFORMANCE保持一致，在原来的基础上减少了性能分析的时间，执行时间与SQL执行时间的差异显著减少。

14.2.3 SQL 执行计划详解

如**SQL执行计划概述**节中所说，EXPLAIN会显示执行计划，但并不会实际执行SQL语句。EXPLAIN ANALYZE和EXPLAIN PERFORMANCE两者都会实际执行SQL语句并返回执行信息。在这一节将详细解释执行计划及执行信息。

执行计划

以如下SQL语句为例：

```
select
   cjxh,
   count(1)
from dwcjk
group by cjxh;
```

执行EXPLAIN的输出为：

```
postgres=# explain select cjxh, count(1) from dwcjk group by cjxh;
 id |               operation                | E-rows | E-memory | E-width | E-costs
----+----------------------------------------+--------+----------+---------+---------
  1 | ->  Row Adapter                        |      1 |          |      52 | 58.42
  2 |    ->  Vector Streaming (type: GATHER) |      1 |          |      52 | 58.42
  3 |       ->  Vector Hash Aggregate        |      1 | 16MB     |      52 | 58.02
  4 |          ->  CStore Scan on dwcjk      |      1 | 1MB      |      44 | 58.00
(4 rows)
```

执行计划字段解读（横向）：

- id：执行算子节点编号。
- operation：具体的执行节点算子名称。

  Vector前缀的算子是指向量化执行引擎算子，一般出现含有列存表的Query中。

  Streaming是一个特殊的算子，它实现了分布式架构的核心数据shuffle功能，Streaming共有三种形态，分别对应了分布式结构下不同的数据shuffle功能：

  - Streaming (type: GATHER)：作用是coordinator从DN收集数据。
  - Streaming(type: REDISTRIBUTE)：作用是DN根据选定的列把数据重分布到所有的DN。

  - Streaming(type: BROADCAST)：作用是把当前DN的数据广播给其他所有的DN
- E-rows：每个算子估算的输出行数。
- E-memory：DN上每个算子估算的内存使用量，只有DN上执行的算子会显示。某些场景会在估算的内存使用量后使用括号显示该算子在内存资源充足下可以自动扩展的内存上限。
- E-width：每个算子输出元组的估算宽度。
- E-costs：每个算子估算的执行代价。
  - E-costs是优化器根据成本参数定义的单位来衡量的，习惯上以磁盘页面抓取为1个单位， 其它开销参数将参照它来设置。
  - 每个节点的开销（E-costs值）包括它的所有子节点的开销。
  - 开销只反映了优化器关心的东西，并没有把结果行传递给客户端的时间考虑进去。虽然这个时间可能在实际的总时间里占据相当重要的分量，但是被优化器忽略了，因为它无法通过修改规划来改变。

执行计划层级解读（纵向）：

1. 第一层：CStore Scan on dwcjk

   表扫描算子，用CStore Scan的方式扫描表dwcjk。这一层的作用是把表dwcjk的数据从buffer或者磁盘上读上来输送给上层节点参与计算。
2. 第二层：Vector Hash Aggregate

   聚合算子，作用是把下层计算输送上来的算子做聚合操作(group by)。
3. 第三层：Vector Streaming (type: GATHER)

   Shuffle算子，此处GATHER类型的Shuffle算子作用是把数据从DN汇聚到CN。
4. 第四层：Row Adapter

   存储格式转化算子，主要作用是把内存中列式格式数据转为行式数据，以便客户端展示。

需要注意的是最顶层算子为Data Node Scan时，需要设置enable_fast_query_shipping为off才能看到具体的执行计划，如下面这个计划：

```
explain select cjxh, count(1) from dwcjk group by cjxh;
                QUERY PLAN
--------------------------------------------------
 Data Node Scan  (cost=0.00..0.00 rows=0 width=0)
   Node/s: All datanodes
(2 rows)
```

设置enable_fast_query_shipping参数之后，执行计划显示如下：

```
postgres=# set enable_fast_query_shipping=off;
SET
postgres=# explain select cjxh, count(1) from dwcjk group by cjxh;
 id |               operation                | E-rows | E-memory | E-width | E-costs
----+----------------------------------------+--------+----------+---------+---------
  1 | ->  Row Adapter                        |      1 |          |      52 | 58.42
  2 |    ->  Vector Streaming (type: GATHER) |      1 |          |      52 | 58.42
  3 |       ->  Vector Hash Aggregate        |      1 | 16MB     |      52 | 58.02
  4 |          ->  CStore Scan on dwcjk      |      1 | 1MB      |      44 | 58.00
(4 rows)
```

执行计划中的关键字说明：

1. 表访问方式
   - Seq Scan

     全表顺序扫描。
   - Index Scan

     优化器决定使用两步的规划：最底层的规划节点访问一个索引，找出匹配索引条件的行的位置，然后上层规划节点真实地从表中抓取出那些行。独立地抓取数据行比顺序地读取它们的开销高很多，但是因为并非所有表的页面都被访问了，这么做实际上仍然比一次顺序扫描开销要少。使用两层规划的原因是，上层规划节点在读取索引标识出来的行位置之前，会先将它们按照物理位置排序，这样可以最小化独立抓取的开销。

     如果在WHERE里面使用的好几个字段上都有索引，那么优化器可能会使用索引的AND或OR的组合。但是这么做要求访问两个索引，因此与只使用一个索引，而把另外一个条件只当作过滤器相比，这个方法未必是更优。

     索引扫描可以分为以下几类，他们之间的差异在于索引的排序机制。

     - Bitmap Index Scan

       使用位图索引抓取数据页。

     - Index Scan using index_name

       使用简单索引搜索，该方式表的数据行是以索引顺序抓取的，这样就令读取它们的开销更大，但是这里的行少得可怜，因此对行位置的额外排序并不值得。最常见的就是看到这种规划类型只抓取一行，以及那些要求ORDER BY条件匹配索引顺序的查询。因为那时候没有多余的排序步骤是必要的以满足ORDER BY。
2. 表连接方式
   - Nested Loop

     嵌套循环，适用于被连接的数据子集较小的查询。在嵌套循环中，外表驱动内表，外表返回的每一行都要在内表中检索找到它匹配的行，因此整个查询返回的结果集不能太大（不能大于10000），要把返回子集较小的表作为外表，而且在内表的连接字段上建议要有索引。
   - (Sonic) Hash Join

     哈希连接，适用于数据量大的表的连接方式。优化器使用两个表中较小的表，利用连接键在内存中建立hash表，然后扫描较大的表并探测散列，找到与散列匹配的行。Sonic和非Sonic的Hash Join的区别在于所使用hash表结构不同，不影响执行的结果集。
   - Merge Join

     归并连接，通常情况下执行性能差于哈希连接。如果源数据已经被排序过，在执行融合连接时，并不需要再排序，此时融合连接的性能优于哈希连接。
3. 运算符
   - sort

     对结果集进行排序。
   - filter

     EXPLAIN输出显示WHERE子句当作一个"filter"条件附属于顺序扫描计划节点。这意味着规划节点为它扫描的每一行检查该条件，并且只输出符合条件的行。预计的输出行数降低了，因为有WHERE子句。不过，扫描仍将必须访问所有 10000 行，因此开销没有降低；实际上它还增加了一些（确切的说，通过10000 * cpu_operator_cost）以反映检查WHERE条件的额外CPU时间。

- LIMIT

  LIMIT限定了执行结果的输出记录数。如果增加了LIMIT，那么不是所有的行都会被检索到。

执行信息

在SQL调优过程中经常需要执行EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看SQL语句实际执行信息，通过对比实际执行与优化器的估算之间的差别来为优化提供依据。EXPLAIN PERFORMANCE相对于EXPLAIN ANALYZE增加了每个DN上的执行信息。

以如下SQL语句为例：

```
select count(1) from tb1;
```

执行EXPLAIN PERFORMANCE输出为：

```
postgres=# explain performance select count(1) from tb1;
 id |             operation              |     A-time      | A-rows | E-rows | E-distinct | Peak Memory  | E-memory | A-width | E-width | E-costs
----+------------------------------------+-----------------+--------+--------+------------+--------------+----------+---------+---------+---------
  1 | ->  Aggregate                      | 19.269          |      1 |      1 |            | 10KB         |          |         |       8 | 2.19
  2 |    ->  Streaming (type: GATHER)    | 19.191          |      4 |      4 |            | 144KB        |          |         |       8 | 2.19
  3 |       ->  Aggregate                | [0.004,0.069]   |      4 |      4 |            | [10KB, 10KB] | 1MB      |         |       8 | 2.03
  4 |          ->  Seq Scan on public.tb1 | [0.001,0.060]   |      5 |      5 |            | [12KB, 12KB] | 1MB      |         |       0 | 2.02
(4 rows)

                Memory Information (identified by plan id)
-----------------------------------------------------------------
 Coordinator Query Peak Memory:
        Query Peak Memory: 0MB
 DataNode Query Peak Memory
        datanode1 Query Peak Memory: 0MB
        datanode2 Query Peak Memory: 0MB
        datanode3 Query Peak Memory: 0MB
        datanode4 Query Peak Memory: 0MB
   1 --Aggregate
        Peak Memory: 10KB, Estimate Memory: 64MB
   2 --Streaming (type: GATHER)
        Peak Memory: 144KB, Estimate Memory: 64MB
   3 --Aggregate
        datanode1 Peak Memory: 10KB, Estimate Memory: 1024KB
        datanode2 Peak Memory: 10KB, Estimate Memory: 1024KB
        datanode3 Peak Memory: 10KB, Estimate Memory: 1024KB
        datanode4 Peak Memory: 10KB, Estimate Memory: 1024KB
        datanode1 Stream Send time: 0.000; Data Serialize time: 0.006
        datanode2 Stream Send time: 0.000; Data Serialize time: 0.004
        datanode3 Stream Send time: 0.000; Data Serialize time: 0.005
        datanode4 Stream Send time: 0.000; Data Serialize time: 0.003
   4 --Seq Scan on public.tb1
        datanode1 Peak Memory: 12KB, Estimate Memory: 1024KB
        datanode2 Peak Memory: 12KB, Estimate Memory: 1024KB
        datanode3 Peak Memory: 12KB, Estimate Memory: 1024KB
        datanode4 Peak Memory: 12KB, Estimate Memory: 1024KB
(25 rows)
```

```
 Targetlist Information (identified by plan id)
-------------------------------------------------
   1 --Aggregate
        Output: count((count(1)))
   2 --Streaming (type: GATHER)
        Output: (count(1))
        Node/s: All datanodes
   3 --Aggregate
        Output: count(1)
   4 --Seq Scan on public.tb1
        Output: a, b
        Distribute Key: a
(10 rows)
```

```
                    Datanode Information (identified by plan id)
-------------------------------------------------------------------------------
  1 --Aggregate
       (actual time=19.269..19.269 rows=1 loops=1)
       (Buffers: 0)
       (CPU: ex c/r=50593, ex row=4, ex cyc=202372, inc cyc=50094480)
  2 --Streaming (type: GATHER)
       (actual time=12.371..19.191 rows=4 loops=1)
       (Buffers: 0)
       (CPU: ex c/r=12473027, ex row=4, ex cyc=49892108, inc cyc=49892108)
  3 --Aggregate
       datanode1 (actual time=0.041..0.041 rows=1 loops=1)
       datanode2 (actual time=0.063..0.063 rows=1 loops=1)
       datanode3 (actual time=0.069..0.069 rows=1 loops=1)
       datanode4 (actual time=0.004..0.004 rows=1 loops=1)
       datanode1 (Buffers: shared hit=1)
       datanode2 (Buffers: shared hit=1)
       datanode3 (Buffers: shared hit=1)
       datanode4 (Buffers: 0)
       datanode1 (CPU: ex c/r=32888, ex row=1, ex cyc=32888, inc cyc=107740)
       datanode2 (CPU: ex c/r=9992, ex row=2, ex cyc=19984, inc cyc=163264)
       datanode3 (CPU: ex c/r=12272, ex row=2, ex cyc=24544, inc cyc=180008)
       datanode4 (CPU: ex c/r=0, ex row=0, ex cyc=8100, inc cyc=10768)
  4 --Seq Scan on public.tb1
       datanode1 (actual time=0.027..0.029 rows=1 loops=1)
       datanode2 (actual time=0.054..0.055 rows=2 loops=1)
       datanode3 (actual time=0.059..0.060 rows=2 loops=1)
       datanode4 (actual time=0.001..0.001 rows=0 loops=1)
       datanode1 (Buffers: shared hit=1)
       datanode2 (Buffers: shared hit=1)
       datanode3 (Buffers: shared hit=1)
       datanode4 (Buffers: 0)
       datanode1 (CPU: ex c/r=74852, ex row=1, ex cyc=74852, inc cyc=74852)
       datanode2 (CPU: ex c/r=71640, ex row=2, ex cyc=143280, inc cyc=143280)
       datanode3 (CPU: ex c/r=77732, ex row=2, ex cyc=155464, inc cyc=155464)
       datanode4 (CPU: ex c/r=0, ex row=0, ex cyc=2668, inc cyc=2668)
(34 rows)

                      User Define Profiling
---------------------------------------------------------------------
 Plan Node id: 2  Track name: coordinator get datanode connection
       coordinator1: (time=0.054 total_calls=1 loops=1)
(2 rows)

                        ====== Query Summary =====
-------------------------------------------------------------------------------
 Datanode executor start time [datanode1, datanode2]: [1.352 ms,2.276 ms]
 Datanode executor end time [datanode4, datanode3]: [0.121 ms,0.187 ms]
 Remote query poll time: 10.992 ms, Deserialze time: 0.000 ms
 System available mem: 9188966KB
 Query Max mem: 9473228KB
 Query estimated mem: 2048KB
 Coordinator executor start time: 0.505 ms
 Coordinator executor run time: 19.277 ms
 Coordinator executor end time: 0.380 ms
 Planner runtime: 2.260 ms
 Query Id: 79375943434081342
 Total runtime: 20.803 ms
(12 rows)
```

图中显示执行信息分为以下7个部分

1. 以表格的形式将计划显示出来，包含有11个字段，分别是：id、operation、A-time、A-rows、E-rows、E-distinct、Peak Memory、E-memory、A-width、E-width和E-costs。其中计划类字段（id、operation以及部分E开头字段）的含义与执行EXPLAIN时的含义一致，详见**执行计划**小节中的说明。A-time、A-rows、E-distinct、Peak Memory、A-width的含义说明如下：
   - A-time：当前算子执行完成时间，一般DN上执行的算子的A-time是由[]括起来的两个值，分别表示此算子在所有DN上完成的最短时间和最长时间。
   - A-rows：表示当前算子的实际输出元组数。

- E-distinct: 表示hashjoin算子的distinct估计值。
- Peak Memory：此算子在每个DN上执行时使用的内存峰值。
- A-width：表示当前算子每行元组的实际宽度，仅对于重内存使用算子会显示，包括：(Vec)HashJoin、(Vec)HashAgg、(Vec) HashSetOp、(Vec)Sort、(Vec)Materialize算子等，其中(Vec)HashJoin计算的宽度是其右子树算子的宽度，会显示在其右子树上。

2. Predicate Information (identified by plan id)：

   这一部分主要显示的是静态信息，即在整个计划执行过程中不会变的信息，主要是一些join条件和一些filter信息。

3. Memory Information (identified by plan id)：

   这一部分显示的是整个计划中会将内存的使用情况打印出来的算子的内存使用信息，主要是Hash、Sort算子，包括算子峰值内存（peak memory），控制内存（control memory），估算内存使用（operator memory），执行时实际宽度（width），内存使用自动扩展次数（auto spread num），是否提前下盘（early spilled），以及下盘信息，包括重复下盘次数（spill Time(s)），内外表下盘分区数（inner/outer partition spill num），下盘文件数（temp file num），下盘数据量及最小和最大分区的下盘数据量（written disk IO [min, max] ）。

4. Targetlist Information (identified by plan id)

   这一部分显示的是每一个算子输出的目标列。

5. DataNode Information (identified by plan id)：

   这一部分会将各个算子的执行时间、CPU、buffer的使用情况全部打印出来。

6. User Define Profiling

   这一部分显示的是CN和DN、DN和DN建连的时间，以及存储层的一些执行信息。

7. ====== Query Summary =====：

   这一部分主要打印总的执行时间和网络流量，包括了各个DN上初始化和结束阶段的最大最小执行时间、CN上的初始化、执行、结束阶段的时间，以及当前语句执行时系统可用内存、语句估算内存等信息。

须知

- A-rows和E-rows的差异体现了优化器估算和实际执行的偏差度。一般来说，他们偏差越大，我们越可以认为优化器生成的计划的越不可信，人工干预调优的必要性越大。
- A-time中的两个值偏差越大，表明此算子的计算偏斜(在不同DN上执行时间差异)越大，人工干预调优的必要性越大。
- Max Query Peak Memory经常用来估算SQL语句耗费内存，也被用来作为SQL语句调优时运行态内存参数设置的重要依据。一般会以EXPLAIN ANALYZE或EXPLAIN PERFORMANCE的输出作为进一步调优的输入。
- Query Summary中的Total runtime大于id=1的算子的A-time，因为后者是单纯执行SQL语句的时间，而前者包括了收集所有实例的执行信息，并按照输出格式加工执行信息的过程。一般集群规模越大，执行计划越复杂，二者的差值越大。

14.2.4 查询最耗性能的 SQL

系统中有些SQL语句运行了很长时间还没有结束，这些语句会消耗很多的系统性能，请根据本章内容查询长时间运行的SQL语句。

操作步骤

步骤1 查询系统中长时间运行的查询语句。

```
SELECT current_timestamp - query_start AS runtime, datname, usename, query FROM pg_stat_activity
where state != 'idle' ORDER BY 1 desc;
```

查询后会按执行时间从长到短顺序返回查询语句列表，第一条结果就是当前系统中执行时间最长的查询语句。返回结果中包含了系统调用的SQL语句和用户执行SQL语句，请根据实际找到用户执行时间长的语句。

若当前系统较为繁忙，可以通过限制current_timestamp - query_start大于某一阈值来查看执行时间超过此阈值的查询语句。

```
SELECT query FROM pg_stat_activity WHERE current_timestamp - query_start > interval '1 days';
```

步骤2 设置参数track_activities为on。

```
SET track_activities = on;
```

当此参数为on时，数据库系统才会收集当前活动查询的运行信息。

步骤3 查看正在运行的查询语句。

以查看视图pg_stat_activity为例：

```
SELECT datname, usename, state FROM pg_stat_activity;
 datname  | usename | state  |
----------+---------+--------+
 postgres |   omm   | idle   |
 postgres |   omm   | active |
(2 rows)
```

如果state字段显示为idle，则表明此连接处于空闲，等待用户输入命令。

如果仅需要查看非空闲的查询语句，则使用如下命令查看：

```
SELECT datname, usename, state FROM pg_stat_activity WHERE state != 'idle';
```

步骤4 分析长时间运行的查询语句状态。

- 若查询语句处于正常状态，则等待其执行完毕。
- 若查询语句阻塞，则通过如下命令查看当前处于阻塞状态的查询语句：

  ```
  SELECT datname, usename, state, query FROM pg_stat_activity WHERE waiting = true;
  ```

  查询结果中包含了当前被阻塞的查询语句，该查询语句所请求的锁资源可能被其他会话持有，正在等待持有会话释放锁资源。

  说明

  只有当查询阻塞在系统内部锁资源时，waiting字段才显示为true。尽管等待锁资源是数据库系统最常见的阻塞行为，但是在某些场景下查询也会阻塞在等待其他系统资源上，例如写文件、定时器等。但是这种情况的查询阻塞，不会在视图pg_stat_activity中体现。

----结束

14.2.5 分析作业是否被阻塞

数据库系统运行时，在某些业务场景下查询语句会被阻塞，导致语句运行时间过长，可以强制结束有问题的会话。

操作步骤

步骤1 查看阻塞的查询语句及阻塞查询的表、模式信息。

```
SELECT w.query as waiting_query,
w.pid as w_pid,
w.usename as w_user,
l.query as locking_query,
l.pid as l_pid,
l.usename as l_user,
t.schemaname || '.' || t.relname as tablename
from pg_stat_activity w join pg_locks l1 on w.pid = l1.pid
and not l1.granted join pg_locks l2 on l1.relation = l2.relation
and l2.granted join pg_stat_activity l on l2.pid = l.pid join pg_stat_user_tables t on l1.relation = t.relid
where w.waiting;
```

该查询返回线程ID、用户信息、查询状态，以及导致阻塞的表、模式信息。

步骤2 使用如下命令结束相应的会话。其中，139834762094352为线程ID。

```
SELECT PG_TERMINATE_BACKEND(139834762094352);
```

显示类似如下信息，表示结束会话成功。

```
 PG_TERMINATE_BACKEND
----------------------
 t
(1 row)
```

显示类似如下信息，表示用户正在尝试结束当前会话，此时仅会重连会话，而不是结束会话。

```
FATAL:  terminating connection due to administrator command
FATAL:  terminating connection due to administrator command
The connection to the server was lost. Attempting reset: Succeeded.
```

说明

gsql客户端使用PG_TERMINATE_BACKEND函数终止本会话后台线程时，客户端不会退出而是自动重连。

----结束

14.3 改进查询

14.3.1 调优流程

对慢SQL语句进行分析，通常包括以下步骤：

操作步骤

步骤1 收集SQL中涉及到的所有表的统计信息。在数据库中，统计信息是规划器生成计划的源数据。没有收集统计信息或者统计信息陈旧往往会造成执行计划严重劣化，从而导致性能问题。从经验数据来看，10%左右性能问题是因为没有收集统计信息。具体请参见**更新统计信息**。

步骤2 通过查看执行计划来查找原因。如果SQL长时间运行未结束，通过EXPLAIN命令查看执行计划，进行初步定位。如果SQL可以运行出来，则推荐使用EXPLAIN ANALYZE或EXPLAIN PERFORMANCE查看执行计划及实际运行情况，以便更精准地定位问题原因。有关执行计划的详细介绍请参见**SQL执行计划概述**。

步骤3 **审视和修改表定义**。

步骤4 针对EXPLAIN或EXPLAIN PERFORMANCE信息，定位SQL慢的具体原因以及改进措施，具体参见**典型SQL调优点**。

步骤5 通常情况下，有些SQL语句可以通过查询重写转换成等价的，或特定场景下等价的语句。重写后的语句比原语句更简单，且可以简化某些执行步骤达到提升性能的目的。查询重写方法在各个数据库中基本是通用的。

步骤6 用户可以通过指定join顺序，join、stream、scan方法，指定结果行数，指定重分布过程中的倾斜信息等多个手段来进行执行计划的调优，以提升查询的性能。详细请参见**使用Plan Hint进行调优**。

步骤7 为了保证数据库性能的持续优质，建议**例行维护表**和**例行重建索引**。

步骤8 （可选）GaussDB(DWS)支持在资源富足的情况下，通过算子并行来提升性能。详细请参见**SMP使用建议**。

----结束

14.3.2 更新统计信息

在数据库中，统计信息是规划器生成计划的源数据。没有收集统计信息或者统计信息陈旧往往会造成执行计划严重劣化，从而导致性能问题。

背景信息

ANALYZE语句可收集与数据库中表内容相关的统计信息，统计结果存储在系统表PG_STATISTIC中。查询优化器会使用这些统计数据，以生成最有效的执行计划。

建议在执行了大批量插入/删除操作后，例行对表或全库执行ANALYZE语句更新统计信息。目前默认收集统计信息的采样比例是30000行（即：guc参数default_statistics_target默认设置为100），如果表的总行数超过一定行数（大于1600000），建议设置guc参数default_statistics_target为-2，即按2%收集样本估算统计信息。

对于在批处理脚本或者存储过程中生成的中间表，也需要在完成数据生成之后显式的调用ANALYZE。

对于表中多个列有相关性且查询中有同时基于这些列的条件或分组操作的情况，可尝试收集多列统计信息，以便查询优化器可以更准确地估算行数，并生成更有效的执行计划。

操作步骤

使用以下命令更新某个表或者整个database的统计信息。

```
ANALYZE tablename;                    --更新单个表的统计信息
ANALYZE;                              --更新全库的统计信息
```

使用以下命令进行多列统计信息相关操作。

```
ANALYZE tablename ((column_1, column_2));                    --收集tablename表的column_1、column_2列的多列统计信息

ALTER TABLE tablename ADD STATISTICS ((column_1, column_2));    --添加tablename表的column_1、column_2列的多列统计信息声明
ANALYZE tablename;                                           --收集单列统计信息，并收集已声明的多列统计信息
```

```
ALTER TABLE tablename DELETE STATISTICS ((column_1, column_2)); --删除tablename表的column_1、column_2列的多列统计信息或其声明
```

须知

在使用ALTER TABLE tablename ADD STATISTICS语句添加了多列统计信息声明后，系统并不会立刻收集多列统计信息，而是在下次对该表或全库进行ANALYZE时，进行多列统计信息的收集。

如果想直接收集多列统计信息，请使用ANALYZE命令进行收集。

说明

使用EXPLAIN查看各SQL的执行计划时，如果发现某个表SEQ SCAN的输出中rows=10，rows=10是系统给的默认值，有可能该表没有进行ANALYZE，需要对该表执行ANALYZE。

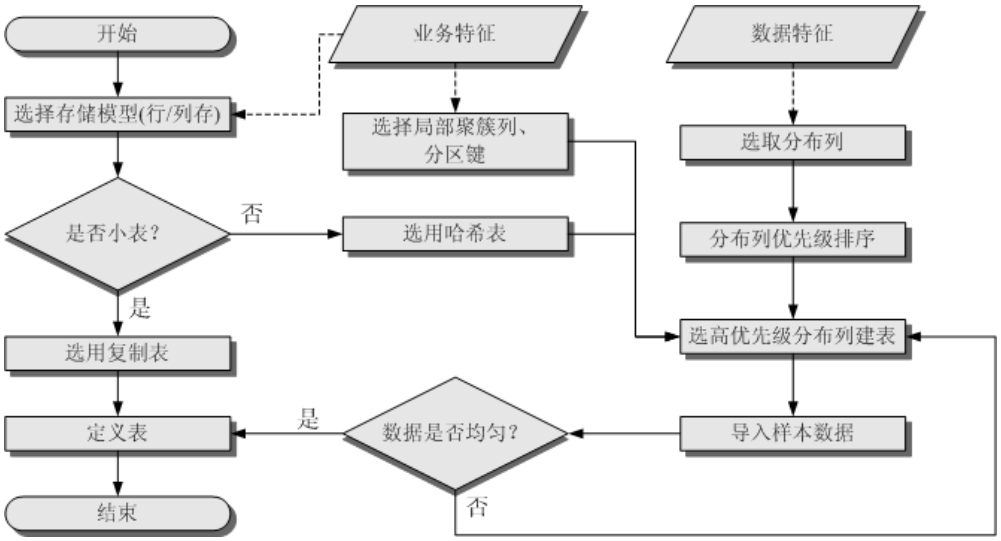
14.3.3 审视和修改表定义

在分布式框架下，数据分布在各个DN上。一个或者几个DN的数据存在一块物理存储设备上，好的表定义至少需要达到以下几个目标：

1. **表数据均匀分布在各个DN上**，以防止单个DN对应的存储设备空间不足造成集群有效容量下降。选择合适分布列，避免数据分布倾斜可以实现该点。
2. **表Scan压力均匀分散在各个DN上**，以避免单DN的Scan压力过大，形成Scan的单节点瓶颈。分布列不选择基表上等值filter中的列可以实现该点。
3. **减少扫描数据量**。通过分区的剪枝机制可以实现该点。
4. **尽量减少随机IO**。通过聚簇/局部聚簇可以实现该点。
5. **尽量避免数据shuffle**，减小网络压力。通过选择join-condition或者group by列为分布列可以最大程度的实现这点。

从上述描述来看表定义中最重要的一点是分布列的选择。创建表定义一般遵循图14-4所示流程。表定义在数据库设计阶段创建，在SQL调优过程中进行审视和修改。

图 14-4 表定义流程



14.3.4 典型 SQL 调优点

14.3.4.1 典型 SQL 调优点概述

SQL调优是一个不断分析与尝试的过程：试跑Query，判断性能是否满足要求；如果不满足要求，则通过查看执行计划分析原因并进行针对性优化；然后重新试跑和优化，直到满足性能目标。本章主要介绍常见的SQL调优方法。

14.3.4.2 SQL 自诊断

用户在执行查询或者执行INSERT/DELETE/UPDATE/CREATE TABLE AS语句时，可能会遇到性能问题。这种情况下，通过查询**GS_WLM_SESSION_STATISTICS**，**GS_WLM_SESSION_HISTORY**，**GS_WLM_SESSION_INFO**视图的warning字段可以获得对应查询可能导致性能问题的告警信息，为性能调优提供参考。

SQL自诊断的告警类型与**resource_track_level**的设置有关系。如果resource_track_level设置为query，则可以诊断多列/单列统计信息未收集和SQL不下推的告警。如果resource_track_level设置为operator，则可以诊断所有的告警场景。

SQL自诊断的诊断范围与**resource_track_cost**的设置有关系。当SQL的代价大于resource_track_cost时，SQL才会被诊断。SQL的代价可以通过explain来确认。

告警场景

目前支持对以下8种导致性能问题的场景上报告警。

- 多列/单列统计信息未收集

如果存在单列或者多列统计信息未收集，则上报相关告警。调优方法可以参考**更新统计信息**和**统计信息调优**。

需要特别注意的是，对于基于OBS外表和HDFS外表的查询，如果未收集统计信息也会上报统计信息未收集的告警，但是由于OBS外表和HDFS外表的analyze的性能比较差，因此，需要用户对这种场景下告警是否通过analyze收集统计信息，以获取更优的性能，和查询本身的复杂度做权衡。

告警信息示例：

整表的统计信息未收集：

```
Statistic Not Collect:
    schema_test.t1
```

单列统计信息未收集：

```
Statistic Not Collect:
    schema_test.t2(c1,c2)
```

多列统计信息未收集:

```
Statistic Not Collect:
    schema_test.t3((c1,c2))
```

单列和多列统计信息未收集：

```
Statistic Not Collect:
    schema_test.t4(c1,c2)    schema_test.t4((c1,c2))
```

- SQL不下推

对于不下推的SQL，尽可能详细上报导致不下推的原因。调优方法可以参考案例**语句下推调优**。

- 对于函数导致的不下推，告警导致不下推的函数名信息；
- 对于不支持下推的语法，会告警对应语法不支持下推，例如：含有With Recursive，Distinct On，row表达式，返回值为record类型的，会告警相应语法不支持下推等等。

告警信息示例：

```
SQL is not plan-shipping, reason : ""enable_stream_operator" is off"
SQL is not plan-shipping, reason : ""Distinct On" can not be shipped"
SQL is not plan-shipping, reason : ""v_test_unshipping_log" is VIEW that will be treated as Record type can't be shipped"
```

- HashJoin中大表做内表

如果在表连接过程中使用了Hashjoin(可以在**GS_WLM_SESSION_HISTORY**的query_plan字段中查看到)，且连接的内表行数是外表行数的10倍或以上；同时内表在每个DN上的平均行数大于10万行，且发生了下盘，则上报相关告警。调优方法可以参考**Plan Hint调优概述**。

告警信息示例：

```
PlanNode[7] Large Table is INNER in HashJoin “Vector Hash Aggregate”
```

- 大表等值连接使用Nestloop

如果在表连接过程中使用了nestloop(可以在**GS_WLM_SESSION_HISTORY**的query_plan字段中查看到)，并且两个表中较大表的行数平均每个DN上的行数大于10万行、表的连接中存在等值连接，则上报相关告警。调优方法可以参考**Plan Hint调优概述**。

告警信息示例：

```
PlanNode[5] Large Table with Equal-Condition use Nestloop"Nested Loop"
```

- 大表Broadcast

如果在Broadcast算子中，平均每DN的行数大于10万行，则告警大表broadcast。调优方法可以参考**Plan Hint调优概述**。

告警信息示例：

```
PlanNode[5] Large Table in Broadcast "Streaming(type: BROADCAST dop: 1/2)"
```

- 数据倾斜

某表在各DN上的分布，存在某DN上的行数是另一DN上行数的10倍或以上，且有DN中的行数大于10万行，则上报相关告警。调优方法可以参考案例**案例：选择合适的分布列**和**数据倾斜调优**。

告警信息示例：

```
PlanNode[6] DataSkew:"Seq Scan", min_dn_tuples:0, max_dn_tuples:524288
```

- 索引不合理

  在基表扫描时，满足下述条件则上报相关告警：

  - 对于行存表：
    - 使用索引扫描，输出行数/扫描行数>1/1000且输出行数>1万行
    - 使用顺序扫描，输出行数/扫描行数<1/1000且输出行数<=1万行、扫描行数>1万行

- 对于列存表：
  - 使用索引扫描，输出行数/扫描行数>1/10000且输出行数>100
  - 使用顺序扫描，输出行数/扫描行数<1/10000且输出行数<=100、扫描行数>1万行

调优方法可以参考**算子级调优**。

告警信息示例：

```
PlanNode[4] Indexscan is not properly used:"Index Only Scan", output:524288, filtered:0, rate:1.00000
PlanNode[5] Indexscan is ought to be used:"Seq Scan", output:1, filtered:524288, rate:0.00000
```

- 估算不准

如果优化器的估算行数和实际行数中的较大值平均每DN行数大于10万行，并且估算行数和实际行数中较大值是较小值的10倍或以上，则上报相关告警。调优方法可以参考**Plan Hint调优概述**。

告警信息示例：

```
PlanNode[5] Inaccurate Estimation-Rows: "Hash Join" A-Rows:0, E-Rows:52488
```

规格约束

1. 告警字符串长度上限为2048。如果告警信息超过这个长度（例如存在大量未收集统计信息的超长表名，列名等信息）则不告警，只上报warning：
   ```
   WARNING, "Planner issue report is truncated, the rest of planner issues will be skipped"
   ```
2. 如果query存在limit节点（即查询语句中包含limit），则不会上报limit节点以下的Operator级别的告警。
3. 对于“数据倾斜”和“估算不准”两种类型告警，在某一个plan树结构下，只上报下层节点的告警，上层节点不再重复告警。这主要是因为这两种类型的告警可能是因为底层触发上层的。例如，如果在scan节点已经存在数据倾斜，那么在上层的hashagg等其他算子很可能也出现数据倾斜。

14.3.4.3 语句下推调优

语句下推介绍

目前，GaussDB(DWS)优化器在分布式框架下制定语句的执行策略时，有三种执行计划方式：生成下推语句计划、生成分布式执行计划、生成发送语句的分布式执行计划。

- 下推语句计划：指直接将查询语句从CN发送到DN进行执行，然后将执行结果返回给CN。
- 分布式执行计划：指CN对查询语句进行编译和优化，生成计划树，再将计划树发送给DN进行执行，并在执行完毕后返回结果到CN。
- 发送语句的分布式执行计划：上述两种方式都不可行时，将可下推的查询部分组成查询语句（多为基表扫描语句）下推到DN进行执行，获取中间结果到CN，然后在CN执行剩下的部分。

在第3种策略中，要将大量中间结果从DN发送到CN，并且要在CN运行不能下推的部分语句，会导致CN成为性能瓶颈（带宽、存储、计算等）。在进行性能调优的时候，应尽量避免只能选择第3种策略的查询语句。

执行语句不能下推是因为语句中含有**不支持下推的函数**或者**不支持下推的语法**。一般都可以通过等价改写规避执行计划不能下推的问题。

查看执行计划是否下推

执行计划是否下推可以依靠如下方法快速判断：

步骤1 将GUC参数“**enable_fast_query_shipping**”设置为off，使查询优化器使用分布式框架策略。

```
SET enable_fast_query_shipping = off;
```

步骤2 查看执行计划。

如果执行计划中有Data Node Scan节点，那么此执行计划为不可下推的执行计划；如果执行计划中有Streaming节点，那么计划是可以下推的。

例如如下业务SQL：

```
select
count(ss.ss_sold_date_sk order by ss.ss_sold_date_sk)c1
from store_sales ss, store_returns sr
where
sr.sr_customer_sk = ss.ss_customer_sk;
```

执行计划如下，可以看出此SQL语句不能下推。

```
                            QUERY PLAN
------------------------------------------------------------------------------
Aggregate
->  Hash Join
Hash Cond: (ss.ss_customer_sk = sr.sr_customer_sk)
->  Data Node Scan on store_sales "_REMOTE_TABLE_QUERY_"
Node/s: All datanodes
->  Hash
->  Data Node Scan on store_returns "_REMOTE_TABLE_QUERY_"
Node/s: All datanodes
(8 rows)
```

----结束

不支持下推的语法

以如下三个表定义说明不支持下推的SQL语法。

```
CREATE TABLE CUSTOMER1
(
    C_CUSTKEY     BIGINT NOT NULL
  , C_NAME        VARCHAR(25) NOT NULL
  , C_ADDRESS     VARCHAR(40) NOT NULL
  , C_NATIONKEY   INT NOT NULL
  , C_PHONE       CHAR(15) NOT NULL
  , C_ACCTBAL     DECIMAL(15,2)   NOT NULL
  , C_MKTSEGMENT  CHAR(10) NOT NULL
  , C_COMMENT     VARCHAR(117) NOT NULL
)
DISTRIBUTE BY hash(C_CUSTKEY);
CREATE TABLE test_stream(a int,b float); --float不支持重分布
CREATE TABLE sal_emp ( c1 integer[] ) DISTRIBUTE BY replication;
```

- 不支持returning语句下推

  ```
  explain update customer1 set C_NAME = 'a' returning c_name;
                                QUERY PLAN
  ------------------------------------------------------------------
   Update on customer1  (cost=0.00..0.00 rows=30 width=187)
     Node/s: All datanodes
  ```

```
  Node expr: c_custkey
  ->  Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=187)
        Node/s: All datanodes
(5 rows)
```

- 不支持聚集函数中使用order by语句的下推

```
explain verbose select count ( c_custkey order by c_custkey) from customer1;

                          QUERY PLAN
------------------------------------------------------------------------ Aggregate  (cost=2.50..2.51 rows=1 width=8)
  Output: count(customer1.c_custkey ORDER BY customer1.c_custkey)
  ->  Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=8)
        Output: customer1.c_custkey
        Node/s: All datanodes
        Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
(6 rows)
```

- count(distinct expr)中的字段不支持重分布，则不支持下推

```
explain verbose select count(distinct b) from test_stream;
                                    QUERY PLAN
------------------------------------------------------------------------ Aggregate  (cost=2.50..2.51 rows=1 width=8)
  Output: count(DISTINCT test_stream.b)
  ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=8)
        Output: test_stream.b
        Node/s: All datanodes
        Remote query: SELECT b FROM ONLY public.test_stream WHERE true
(6 rows)
```

- 不支持distinct on用法下推

```
explain verbose select distinct on (c_custkey) c_custkey from customer1 order by c_custkey;
                                      QUERY PLAN
------------------------------------------------------------------------ Unique  (cost=49.83..54.83 rows=30 width=8)
  Output: customer1.c_custkey
  ->  Sort  (cost=49.83..52.33 rows=30 width=8)
        Output: customer1.c_custkey
        Sort Key: customer1.c_custkey
        ->  Data Node Scan on customer1 "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30
width=8)
              Output: customer1.c_custkey
              Node/s: All datanodes
              Remote query: SELECT c_custkey FROM ONLY public.customer1 WHERE true
(9 rows)
```

- Fulljoin的join列如果不支持重分布，则不支持下推

```
explain select * from test_stream t1 full join test_stream t2 on t1.a=t2.b;
                                      QUERY PLAN
------------------------------------------------------------------------ Hash Full Join  (cost=0.38..0.82 rows=30
width=24)
  Hash Cond: ((t1.a)::double precision = t2.b)
  ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30 width=12)
        Node/s: All datanodes
  ->  Hash  (cost=0.00..0.00 rows=30 width=12)
        ->  Data Node Scan on test_stream "_REMOTE_TABLE_QUERY_"  (cost=0.00..0.00 rows=30
width=12)
              Node/s: All datanodes
(7 rows)
```

- 不支持数组表达式下推

```
explain verbose select array[c_custkey,1] from customer1 order by c_custkey;

                          QUERY PLAN
------------------------------------------------------------------------ Sort  (cost=49.83..52.33 rows=30 width=8)
  Output: (ARRAY[customer1.c_custkey, 1::bigint]), customer1.c_custkey
  Sort Key: customer1.c_custkey
  ->  Data Node Scan on "__REMOTE_SORT_QUERY__"  (cost=0.00..0.00 rows=30 width=8)
        Output: (ARRAY[customer1.c_custkey, 1::bigint]), customer1.c_custkey
        Node/s: All datanodes
        Remote query: SELECT ARRAY[c_custkey, 1::bigint], c_custkey FROM ONLY public.customer1
WHERE true ORDER BY 2
(7 rows)
```

- With Recursive当前版本不支持下推的场景和原因如下：

序号	场景	不下推原因
1	包含外表、HDFS表的查询场景	LOG: SQL can't be shipped, reason: RecursiveUnion contains HDFS Table or ForeignScan is not shippable（LOG为CN日志中打印的不下推原因，下同） 外表、HDFS表，当前版本暂不支持下推。
2	多nodegroup场景	LOG: SQL can't be shipped, reason: With-Recursive under multi-nodegroup scenario is not shippable 基表存储nodegroup不相同，或者计算nodegroup与基表不相同，当前版本暂不支持下推。
3	WITH recursive t_result AS (SELECT dm,sj_dm,name,1 as level FROM test_rec_part WHERE sj_dm > 10 UNION SELECT t2.dm,t2.sj_dm,t2.name||' > '|| t1.name,t1.level+1 FROM t_result t1 JOIN test_rec_part t2 ON t2.sj_dm = t1.dm) SELECT * FROM t_result t;	LOG: SQL can't be shipped, reason: With-Recursive does not contain "ALL" to bind recursive & none-recursive branches UNION不带ALL，需要去重。
4	WITH RECURSIVE x(id) AS (select count(1) from pg_class where oid=1247 UNION ALL SELECT id+1 FROM x WHERE id < 5), y(id) AS (select count(1) from pg_class where oid=1247 UNION ALL SELECT id+1 FROM x WHERE id < 10) SELECT y.*, x.* FROM y LEFT JOIN x USING (id) ORDER BY 1;	LOG: SQL can't be shipped, reason: With-Recursive contains system table is not shippable 基表中有系统表。
5	WITH RECURSIVE t(n) AS (VALUES (1) UNION ALL SELECT n+1 FROM t WHERE n < 100) SELECT sum(n) FROM t;	LOG: SQL can't be shipped, reason: With-Recursive contains only values rte is not shippable 基表扫描只有VALUES子句，仅在CN上即可完成执行。

序号	场景	不下推原因
6	select a.ID,a.Name, (with recursive cte as (select ID, PID, NAME from b where b.ID = 1 union all select parent.ID,parent.PID,parent.NAME from cte as child join b as parent on child.pid=parent.id where child.ID = a.ID) select NAME from cte limit 1) cName from (select id, name, count(*) as cnt from a group by id,name) a order by 1,2;	LOG: SQL can't be shipped, reason: With-Recursive recursive term correlated only is not shippable 相关子查询的关联条件仅在递归部分，非递归部分无关联条件。
7	WITH recursive t_result AS (select * from(SELECT dm,sj_dm,name,1 as level FROM test_rec_part WHERE sj_dm < 10 order by dm limit 6 offset 2) UNION all SELECT t2.dm,t2.sj_dm,t2.name||' > '||t1.name,t1.level+1 FROM t_result t1 JOIN test_rec_part t2 ON t2.sj_dm = t1.dm) SELECT * FROM t_result t;	LOG: SQL can't be shipped, reason: With-Recursive contains conflict distribution in none-recursive(Replicate) recursive(Hash) 非递归部分带limit为Replicate计划，递归部分为Hash计划，计划存在冲突。
8	with recursive cte as (select * from rec_tb4 where id<4 union all select h.id,h.parentID,h.name from (with recursive cte as (select * from rec_tb4 where id<4 union all select h.id,h.parentID,h.name from rec_tb4 h inner join cte c on h.id=c.parentID) SELECT id ,parentID,name from cte order by parentID) h inner join cte c on h.id=c.parentID) SELECT id ,parentID,name from cte order by parentID,1,2,3;	LOG: SQL can't be shipped, reason: Recursive CTE references recursive CTE "cte" 多层Recursive嵌套，即recursive的递归部分又嵌套另一个recursive查询。

不支持下推的函数

首先介绍函数的易变性。在GaussDB(DWS)中共分三种形态：

- **IMMUTABLE**
  表示该函数在给出同样的参数值时总是返回同样的结果。
- **STABLE**

表示该函数不能修改数据库，对相同参数值，在同一次表扫描里，该函数的返回值不变，但是返回值可能在不同SQL语句之间变化。

- **VOLATILE**

  表示该函数值可以在一次表扫描内改变，因此不会做任何优化。

函数易变性可以查询pg_proc的provolatile字段获得，i代表IMMUTABLE，s代表STABLE，v代表VOLATILE。另外，在pg_proc中的proshippable字段，取值范围为t/f/NULL，这个字段与provolatile字段一起用于描述函数是否下推。

- 如果函数的provolatile属性为i，则无论proshippable的值是否为t，则函数始终可以下推。
- 如果函数的provolatile属性为s或v，则仅当proshippable的值为t时，函数可以下推。
- random如果出现CTE中，也不下推。因为这种场景下下推可能出现结果错误。

对于用户自定义函数，可以在创建函数的时候指定provolatile和proshippable属性的值，详细请参考CREATE FUNCTION。

对于函数不能下推的场景：

- 如果是系统函数，建议根据业务等价替换这个函数。
- 如果是自定义函数，建议分析客户业务场景，看函数的provolatile和proshippable属性定义是否正确。

实例分析：自定义函数

对于自定义函数，如果对于确定的输入，有确定的输出，则应将函数定义为immutable类型。

利用TPCDS的销售信息举个例子，比如我们要写一个函数，获取商品的打折情况，需要一个计算折扣的函数，我们可以将这个函数定义为：

```
CREATE FUNCTION func_percent_2 (NUMERIC, NUMERIC) RETURNS NUMERIC
AS 'SELECT $1 / $2 WHERE $2 > 0.01'
LANGUAGE SQL
VOLATILE;
```

执行下列语句：

```
SELECT func_percent_2(ss_sales_price, ss_list_price)
FROM store_sales;
```

其执行计划为：

```
 Data Node Scan on store_sales "_REMOTE_TABLE_QUERY_"
   Output: func_percent_2(store_sales.ss_sales_price, store_sales.ss_list_price)
   Remote query: SELECT ss_sales_price, ss_list_price FROM ONLY store_sales WHERE true
(3 rows)
```

可见，func_percent_2并没有被下推，而是将ss_sales_price和ss_list_price收到CN上，再进行计算，消耗大量CN的资源，而且计算缓慢。

由于该自定义函数对确定的输入有确定的输出，如果将该自定义函数改为：

```
CREATE FUNCTION func_percent_1 (NUMERIC, NUMERIC) RETURNS NUMERIC
AS 'SELECT $1 / $2 WHERE $2 > 0.01'
LANGUAGE SQL
IMMUTABLE;
```

执行语句：

```
SELECT func_percent_1(ss_sales_price, ss_list_price)
FROM store_sales;
```

其执行计划为：

```
 Data Node Scan
   Output: (func_percent_1(store_sales.ss_sales_price, store_sales.ss_list_price))
   Remote query: SELECT func_percent_1(ss_sales_price, ss_list_price) AS func_percent_1 FROM store_sales
(3 rows)
```

可见函数func_percent_1被下推到DN执行，提升了执行效率（TPCDS 1000X，3CN18DN，查询效率提升100倍以上）。

实例分析 2：使排序下推

请参考**案例：使排序下推**。

14.3.4.4 子查询调优

子查询介绍

一个“SELECT...FROM...WHERE...”语句称为一个查询块，将一个查询块嵌套到另一个查询的FROM子句，WHERE子句，HAVING等子句中的查询称为嵌套查询。其中被嵌入其它查询块中的查询称为嵌套子查询，简称**子查询**。

子查询可以分为相关性子查询和非相关性子查询。相关性子查询是指该子查询的执行依赖于外层父查询的某些属性值，它依赖于父查询传给它的参数，当参数改变时，需要重新执行一遍子查询得到新的结果。非相关子查询是完全的独立的，它只需要执行一次即可。

优化示例

从上面对子查询的介绍可以看出，非相关子查询不会影响性能，因为它只需要执行一次即可。而相关子查询是需要迭代的执行多次，对性能有很大的影响。下面介绍两个常见示例：

示例1：修改基表为replicate表，并且在过滤列上创建索引。

```
create table master_table (a int);
create table sub_table(a int, b int);
select a from master_table group by a having a in (select a from sub_table);
```

上述事例中存在一个相关性子查询，为了提升查询的性能，可以将sub_table修改为一个relication表，并且在字段a上创建一个index。

示例2：修改select语句，将子查询修改为和主表的join，或者修改为可以提升的subquery，但是在修改前后需要保证语义的正确性。

```
explain (costs off)select * from master_table as t1 where t1.a in (select t2.a from sub_table as t2 where t1.a
= t2.b);
                    QUERY PLAN
-----------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Seq Scan on master_table t1
         Filter: (SubPlan 1)
         SubPlan 1
           ->  Result
                 Filter: (t1.a = t2.b)
                 ->  Materialize
                       ->  Streaming(type: BROADCAST)
                             Spawn on: All datanodes
```

```
                        ->  Seq Scan on sub_table t2
(11 rows)
```

上面事例计划中存在一个subPlan，为了消除这个subPlan可以修改语句为：

```
explain(costs off) select * from master_table as t1 where exists (select t2.a from sub_table as t2 where t1.a
= t2.b and t1.a = t2.a);
                 QUERY PLAN
-----------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Hash Semi Join
         Hash Cond: (t1.a = t2.b)
         ->  Seq Scan on master_table t1
         ->  Hash
               ->  Streaming(type: REDISTRIBUTE)
                     Spawn on: All datanodes
                     ->  Seq Scan on sub_table t2
(9 rows)
```

从计划可以看出，subPlan消除了，计划变成了两个表的semi join，这样会大大提高执行效率。

14.3.4.5 统计信息调优

统计信息调优介绍

GaussDB(DWS)是基于代价估算生成的最优执行计划。优化器需要根据analyze收集的统计信息行数估算和代价估算，因此统计信息对优化器行数估算和代价估算起着至关重要的作用。通过analyze收集全局统计信息，主要包括：pg_class表中的relpages和reltuples；pg_statistic表中的stadistinct、stanullfrac、stanumbersN、stavaluesN、histogram_bounds等。

实例分析 1：未收集统计信息导致查询性能差

在很多场景下，由于查询中涉及到的表或列没有收集统计信息，会对查询性能有很大的影响。

表结构如下所示：

```
CREATE TABLE LINEITEM
(
L_ORDERKEY        BIGINT        NOT NULL
, L_PARTKEY       BIGINT        NOT NULL
, L_SUPPKEY       BIGINT        NOT NULL3
, L_LINENUMBER    BIGINT        NOT NULL
, L_QUANTITY      DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT      DECIMAL(15,2) NOT NULL
, L_TAX           DECIMAL(15,2) NOT NULL
, L_RETURNFLAG    CHAR(1)       NOT NULL
, L_LINESTATUS    CHAR(1)       NOT NULL
, L_SHIPDATE      DATE          NOT NULL
, L_COMMITDATE    DATE          NOT NULL
, L_RECEIPTDATE   DATE          NOT NULL
, L_SHIPINSTRUCT  CHAR(25)      NOT NULL
, L_SHIPMODE      CHAR(10)      NOT NULL
, L_COMMENT       VARCHAR(44)   NOT NULL
) with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(L_ORDERKEY);

CREATE TABLE ORDERS
(
O_ORDERKEY       BIGINT        NOT NULL
, O_CUSTKEY      BIGINT        NOT NULL
```

```
, O_ORDERSTATUS    CHAR(1)       NOT NULL
, O_TOTALPRICE     DECIMAL(15,2) NOT NULL
, O_ORDERDATE      DATE NOT NULL
, O_ORDERPRIORITY CHAR(15)       NOT NULL
, O_CLERK          CHAR(15)      NOT NULL
, O_SHIPPRIORITY  BIGINT         NOT NULL
, O_COMMENT        VARCHAR(79)   NOT NULL
)with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(O_ORDERKEY);
```

查询语句如下所示：

```
explain verbose select
count(*) as numwait
from
lineitem l1,
orders
where
o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and not exists (
select
*
from
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
order by
numwait desc;
```

当出现该问题时，可以通过如下方法确认查询中涉及到的表或列有没有做过analyze收集统计信息。

1. 通过explain verbose执行query分析执行计划时会提示WARNING信息，如下所示：
   ```
   WARNING:Statistics in some tables or columns(public.lineitem.l_receiptdate,
   public.lineitem.l_commitdate, public.lineitem.l_orderkey, public.lineitem.l_suppkey,
   public.orders.o_orderstatus, public.orders.o_orderkey) are not collected.
   HINT:Do analyze for them in order to generate optimized plan.
   ```
2. 可以通过在pg_log目录下的日志文件中查找以下信息来确认是当前执行的query是否由于没有收集统计信息导致查询性能变差。
   ```
   2017-06-14 17:28:30.336 CST 140644024579856 20971684 [BACKEND] LOG:Statistics in some tables
   or columns(public.lineitem.l_receiptdate, public.lineitem.l_commitdate, public.lineitem.l_orderkey,
   public.linei
   tem.l_suppkey, public.orders.o_orderstatus, public.orders.o_orderkey) are not collected.
   2017-06-14 17:28:30.336 CST 140644024579856 20971684 [BACKEND] HINT:Do analyze for them in
   order to generate optimized plan.
   ```

当通过以上方法查看到哪些表或列没有做analyze，可以通过对WARNING或日志中上报的表或列做analyze可以解决由于为收集统计信息导致查询变慢的问题。

实例分析 2：设置 cost_param 对查询性能优化

请参考**案例：设置cost_param对查询性能优化**。

实例分析 3：多表 join 的复杂查询存在中间结果不准调优

现象描述：查询与指定人在前后15分钟内、同一网吧登记上网的人员信息：

```
SELECT
C.WBM,
C.DZQH,
```

```
C.DZ,
B.ZJHM,
B.SWKSSJ,
B.XWSJ
FROM
b_zyk_wbswxx A,
b_zyk_wbswxx B,
b_zyk_wbcs C
WHERE
A.ZJHM = '522522******3824'
AND A.WBDM = B.WBDM
AND A.WBDM = C.WBDM
AND abs(to_date(A.SWKSSJ,'yyyymmddHH24MISS') - to_date(B.SWKSSJ,'yyyymmddHH24MISS')) <
INTERVAL '15 MINUTES'
ORDER BY
B.SWKSSJ,
B.ZJHM
limit 10 offset 0
;
```

执行计划如图14-5所示。该查询实际耗时约12秒。

图 14-5 应用 unlogged table 案例（一）

```
                                QUERY PLAN
--------------------------------------------------------------------------------
Limit  (cost=221021.41..221021.43 rows=10 width=120)
  ->  Sort  (cost=221021.41..221022.01 rows=240 width=120)
        Sort Key: b.swkssj, b.zjhm
        ->  Streaming (type: GATHER)  (cost=221015.62..221016.22 rows=240 width=120)
              Node/s: All datanodes
              ->  Limit  (cost=9208.98..9209.01 rows=10 width=120)
                    ->  Sort  (cost=9208.98..9211.60 rows=1048 width=120)
                          Sort Key: b.swkssj, b.zjhm
                          ->  Nested Loop  (cost=23.27..9186.34 rows=1048 width=120)
                                Join Filter: (((a.zjhm)::text <> (b.zjhm)::text) AND ((a.wbdm)::text = (b.wbdm)::text)
                                AND (abs(((to_date((a.swkssj)::text, 'yyyymmddHH24MISS'::text)
                                  - to_date((b.swkssj)::text, 'yyyymmddHH24MISS'::text)))::numeric) < .0104166666666667))
                                ->  Streaming(type: BROADCAST)  (cost=0.00..6.33 rows=24 width=135)
                                      Spawn on: All datanodes
                                      ->  Nested Loop  (cost=0.00..106.80 rows=1 width=135)
                                            ->  Streaming(type: BROADCAST)  (cost=0.00..24.75 rows=264 width=48)
                                                  Spawn on: All datanodes
                                                  ->  Partition Iterator  (cost=0.00..48.44 rows=11 width=48)
                                                        Iterations: 25
                                                        ->  Partitioned Index Scan using idx_b_zyk_wbswxx_zjhm on b_zyk_wbswxx a  (cost=0.00..48.44 rows=11 width=48)
                                                              Index Cond: ((zjhm)::text = '522522******3824'::text)
                                                              Selected Partitions:  1..25
                                            ->  Index Scan using idx_b_zyk_wbcs_wbdm on b_zyk_wbcs c  (cost=0.00..2.82 rows=1 width=87)
                                                  Index Cond: ((wbdm)::text = (a.wbdm)::text)
                                ->  Partition Iterator  (cost=23.27..7306.33 rows=2454 width=63)
                                      Iterations: 25
                                      ->  Partitioned Bitmap Heap Scan on b_zyk_wbswxx b  (cost=23.27..7306.33 rows=2454 width=63)
                                            Recheck Cond: ((wbdm)::text = (c.wbdm)::text)
                                            Filter: ('522522198405243824'::text <> (zjhm)::text)
                                            Selected Partitions:  1..25
                                            ->  Partitioned Bitmap Index Scan on idx_b_zyk_wbswxx_wbdm  (cost=0.00..22.65 rows=2454 width=0)
                                                  Index Cond: ((wbdm)::text = (c.wbdm)::text)
(30 rows)
```

优化分析：分析过程如下：

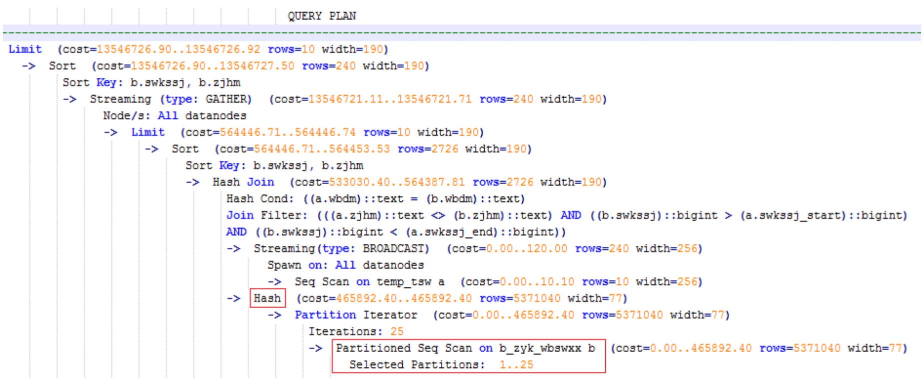
1. 分析该执行计划发现，扫描节点已使用Index Scan，耗时主要在最外层Nest Loop Join的Join Filter计算中，且该计算执行了字符串的加减法和不等值比较。
2. 考虑使用unlogged table保存目标人的上网信息，且在插入时处理上网开始时间和终止时间，以避免后续进行时间加减。

```
//创建临时unlogged table
CREATE UNLOGGED TABLE temp_tsw
(
ZJHM         NVARCHAR2(18),
WBDM         NVARCHAR2(14),
SWKSSJ_START NVARCHAR2(14),
SWKSSJ_END   NVARCHAR2(14),
WBM          NVARCHAR2(70),
DZQH         NVARCHAR2(6),
DZ           NVARCHAR2(70),
IPDZ         NVARCHAR2(39)
)
;
//插入目标人的上网记录，并处理上网开始和结束时间。
```

```
INSERT INTO
temp_tsw
SELECT
A.ZJHM,
A.WBDM,
to_char((to_date(A.SWKSSJ,'yyyymmddHH24MISS') - INTERVAL '15
MINUTES'),'yyyymmddHH24MISS'),
to_char((to_date(A.SWKSSJ,'yyyymmddHH24MISS') + INTERVAL '15
MINUTES'),'yyyymmddHH24MISS'),
B.WBM,B.DZQH,B.DZ,B.IPDZ
FROM
b_zyk_wbswxx A,
b_zyk_wbcs B
WHERE
A.ZJHM='522522******3824' AND A.WBDM = B.WBDM
;

//查询和目标人在前后十五分钟内在同一网吧上网的人员信息，比较大小时强制转换为int8。
SELECT
A.WBM,
A.DZQH,
A.DZ,
A.IPDZ,
B.ZJHM,
B.XM,
to_date(B.SWKSSJ,'yyyymmddHH24MISS') as SWKSSJ,
to_date(B.XWSJ,'yyyymmddHH24MISS') as XWSJ,
B.SWZDH
FROM temp_tsw A,
b_zyk_wbswxx B
WHERE
A.ZJHM <> B.ZJHM
AND A.WBDM = B.WBDM
AND (B.SWKSSJ)::int8 > (A.swkssj_start)::int8
AND (B.SWKSSJ)::int8 < (A.swkssj_end)::int8
order by
B.SWKSSJ,
B.ZJHM
limit 10 offset 0
;
```

上述查询耗时约7秒，执行计划如图14-6所示。

图 14-6 应用 unlogged table 案例（二）



3. 分析上述执行计划，发现执行了Hash Join，对大表b_zyk_wbswxx建立了Hash Table。由于该表数据量大，创建过程耗时较长。

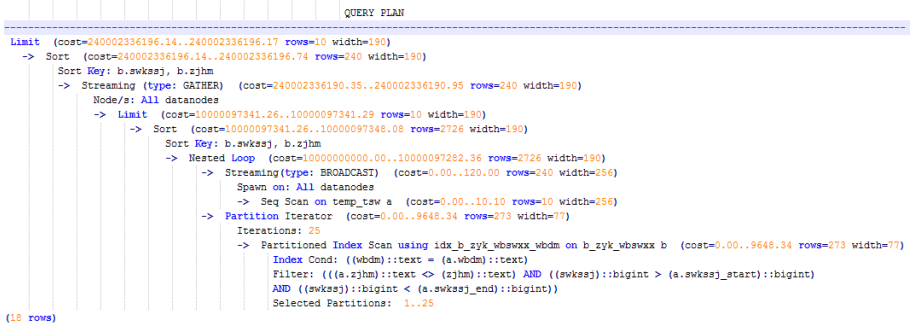
   由于temp_tsw中仅包含几百条记录，且temp_tsw和b_zyk_wbswxx均通过wbdm（网吧代码）执行等值连接。因此，如果Join方式改为Nest Loop Join，则扫描节点可以实现Index Scan，性能预计将会提升。

4. 执行如下语句，将Join方式改为Nest Loop Join。

```
SET enable_hashjoin = off;
```

执行计划如图14-7所示。查询耗时约3秒。

图 14-7 应用 unlogged table 案例（三）



5. 使用unlogged table保存结果集并用于分页显示。

   如果需要在上层应用页面实现分页显示，需要修改offset值确定显示目标页的结果集。按此实现，每次翻页时均执行上面查询语句，耗时较长。

   为解决上述问题，建议使用unlogged table保存结果集。

```
//创建保存结果集的unlogged table
CREATE UNLOGGED TABLE temp_result
(
WBM      NVARCHAR2(70),
DZQH     NVARCHAR2(6),
DZ       NVARCHAR2(70),
IPDZ     NVARCHAR2(39),
ZJHM     NVARCHAR2(18),
XM       NVARCHAR2(30),
SWKSSJ   date,
XWSJ     date,
SWZDH    NVARCHAR2(32)
);

//将结果集插入unlogged table，插入耗时约3秒。
INSERT INTO
temp_result
SELECT
A.WBM,
A.DZQH,
A.DZ,
A.IPDZ,
B.ZJHM,
B.XM,
to_date(B.SWKSSJ,'yyyymmddHH24MISS') as SWKSSJ,
to_date(B.XWSJ,'yyyymmddHH24MISS') as XWSJ,
B.SWZDH
FROM temp_tsw A,
b_zyk_wbswxx B
WHERE
A.ZJHM <> B.ZJHM
AND A.WBDM = B.WBDM
AND (B.SWKSSJ)::int8 > (A.swkssj_start)::int8
AND (B.SWKSSJ)::int8 < (A.swkssj_end)::int8
;

//查询结果集表进行分页显示，分页查询耗时约10ms。
SELECT
*
FROM
temp_result
ORDER BY
SWKSSJ,
ZJHM
LIMIT 10 OFFSET 0;
```

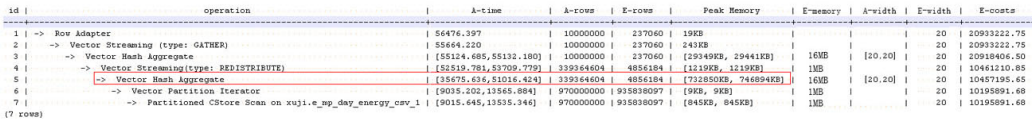
> **注意**
>
> 通过analyze收集全局统计信息，通常会改善查询性能。
>
> 如果遇到性能问题：可以使用plan hint来调整到之前的查询计划，详情请参见**Plan Hint调优概述**。

14.3.4.6 算子级调优

算子级调优介绍

一个查询语句要经过多个算子步骤才会输出最终的结果。由于个别算子耗时过长导致整体查询性能下降的情况比较常见。这些算子是整个查询的瓶颈算子。通用的优化手段是EXPLAIN ANALYZE/PERFORMANCE命令查看执行过程的瓶颈算子，然后进行针对性优化。

如下面的执行过程信息中，Hashagg算子的执行时间占总时间的：(51016-13535)/56476 ≈66%，此处Hashagg算子就是这个查询的瓶颈算子，在进行性能优化时应当优先考虑此算子的优化。



算子级调优示例

示例1：基表扫描时，对于点查或者范围扫描等过滤大量数据的查询，如果使用SeqScan全表扫描会比较耗时，可以在条件列上建立索引选择IndexScan进行索引扫描提升扫描效率。

```
 explain (analyze on, costs off) select * from store_sales where ss_sold_date_sk = 2450944;
 id |             operation          |       A-time        | A-rows | Peak Memory  | A-width
----+--------------------------------+---------------------+--------+--------------+---------
  1 | ->  Streaming (type: GATHER)   | 3666.020            |   3360 | 195KB        |
  2 |    ->  Seq Scan on store_sales | [3594.611,3594.611] |   3360 | [34KB, 34KB] |
(2 rows)

 Predicate Information (identified by plan id)
-----------------------------------------------
   2 --Seq Scan on store_sales
         Filter: (ss_sold_date_sk = 2450944)
         Rows Removed by Filter: 4968936
 create index idx on store_sales_row(ss_sold_date_sk);
CREATE INDEX
 explain (analyze on, costs off) select * from store_sales_row where ss_sold_date_sk = 2450944;
 id |                   operation                    |     A-time      | A-rows | Peak Memory  | A-width
----+------------------------------------------------+-----------------+--------+--------------+---------
  1 | ->  Streaming (type: GATHER)                   | 81.524          |   3360 | 195KB        |
  2 |    ->  Index Scan using idx on store_sales_row | [13.352,13.352] |   3360 | [34KB, 34KB] |
(2 rows)
```

上述例子中，全表扫描返回3360条数据，过滤掉大量数据，在ss_sold_date_sk列上建立索引后，使用IndexScan扫描效率显著提高，从3.6秒提升到13毫秒。

示例2：如果从执行计划中看，两表join选择了NestLoop，而实际行数比较大时，NestLoop Join可能执行比较慢。如下的例子中NestLoop耗时181秒，如果设置参数enable_mergejoin=off关掉Merge Join，同时设置参数enable_nestloop=off关掉NestLoop，让优化器选择HashJoin，则Join耗时提升至200多毫秒。

```
postgres=#  explain analyze select count(*) from store_sales ss, item i where ss.ss_item_sk = i.i_item_sk;
 id |                    operation                    |        A-time         |  A-rows   | E-rows  |  Peak Memory   | E-memory | A-width | E-width |   E-costs
----+-------------------------------------------------+-----------------------+-----------+---------+----------------+----------+---------+---------+-------------
  1 | ->  Row Adapter                                 | [184300.301           |         1 |       1 | [11KB          |          |         |       0 | 48629179.77
  2 |    ->  Vector Aggregate                         | [184300.280           |         1 |       1 | [181KB         |          |         |       0 | 48629179.77
  3 |       ->  Vector Streaming (type: GATHER)       | [184300.186           |         4 |       4 | [188KB         |          |         |       0 | 48629179.77
  4 |          ->  Vector Aggregate                   | [165575.384,184252.368] |         4 |       4 | [140KB, 140KB] | 1MB      |         |       0 | 48629179.61
  5 |             ->  Vector Nest Loop (6,7)          | [162918.848,181438.162] |   2880404 | 2880404 | [74KB, 74KB]   | 1MB      |         |       0 | 48627379.35
  6 |                ->  CStore Scan on store_sales ss | [15.660,16.229]       |   2880404 | 2880404 | [490KB, 490KB] | 1MB      |         |       4 | 16683.10
  7 |                ->  Vector Materialize           | [118314.521,132478.454] | 12968211302 |   18000 | [869KB, 900KB] | 16MB     | [8,8]   |       4 | 3890.00
  8 |                   ->  CStore Scan on item i     | [0.234,0.243]         |     18000 |   18000 | [476KB, 476KB] | 1MB      |         |       4 | 3867.50
(8 rows)
```

```
postgres=#  set enable_nestloop=off;
SET
postgres=#  set enable_mergejoin=off;
SET
ppostgres=# explain analyze select count(*) fpostgres=# ales ss, item i where ss.ss_item_sk = i.i_item_sk;
 id |                    operation                    |        A-time         | A-rows  | E-rows  |  Peak Memory   | E-memory | A-width | E-width | E-costs
----+-------------------------------------------------+-----------------------+---------+---------+----------------+----------+---------+---------+----------
  1 | ->  Row Adapter                                 | [291.066              |       1 |       1 | [11KB          |          |         |       0 | 32308.66
  2 |    ->  Vector Aggregate                         | [291.052              |       1 |       1 | [181KB         |          |         |       0 | 32308.66
  3 |       ->  Vector Streaming (type: GATHER)       | [290.973              |       4 |       4 | [188KB         |          |         |       0 | 32308.66
  4 |          ->  Vector Aggregate                   | [220.792,234.532]     |       4 |       4 | [140KB, 140KB] | 1MB      |         |       0 | 32308.50
  5 |             ->  Vector Hash Join (6,7)          | [209.987,223.345]     | 2880404 | 2880404 | [236KB, 241KB] | 16MB     | [8,8]   |       0 | 30508.24
  6 |                ->  CStore Scan on store_sales ss | [13.132,13.717]     | 2880404 | 2880404 | [490KB, 490KB] | 1MB      |         |       4 | 16683.10
  7 |                ->  CStore Scan on item i        | [0.214,0.246]         |   18000 |   18000 | [477KB, 477KB] | 1MB      |         |       4 | 3867.50
(7 rows)
```

示例3：通常情况下Agg选择HashAgg性能较好，如果大结果集选择了Sort +GroupAgg，则需要设置enable_sort=off，HashAgg耗时明显优于Sort+GroupAgg。

```
postgres=#  explain analyze select count(*) from store_sales group by ss_item_sk;
 id |                    operation                    |        A-time         | A-rows  | E-rows  |      Peak Memory      | E-memory | A-width | E-width | E-costs
----+-------------------------------------------------+-----------------------+---------+---------+-----------------------+----------+---------+---------+----------
  1 | ->  Row Adapter                                 | 1977.385              |   18000 |   17644 | 20KB                  |          |         |       4 | 92875.24
  2 |    ->  Vector Streaming (type: GATHER)          | 1973.617              |   18000 |   17644 | 1946KB                |          |         |       4 | 92875.24
  3 |       ->  Vector Sort Aggregate                 | [1784.800,1883.243]   |   18000 |   17644 | [273KB, 273KB]        | 1MB      |         |       4 | 92186.02
  4 |          ->  Vector Sort                        | [1752.270,1848.357]   | 2880404 | 2880404 | [128466KB, 135135KB]  | 16MB     | [8,8]   |       4 | 88541.40
  5 |             ->  CStore Scan on store_sales      | [12.483,13.548]       | 2880404 | 2880404 | [490KB, 490KB]        | 1MB      |         |       4 | 16683.10
(5 rows)
```

```
postgres=#  set enable_sort=off;
SET
postgres=#  explain analyze select count(*) from store_sales group by ss_item_sk;
 id |                    operation                    |        A-time         | A-rows  | E-rows  |      Peak Memory      | E-memory | A-width | E-width | E-costs
----+-------------------------------------------------+-----------------------+---------+---------+-----------------------+----------+---------+---------+----------
  1 | ->  Row Adapter                                 | 838.218               |   18000 |   17644 | 20KB                  |          |         |       4 | 21016.93
  2 |    ->  Vector Streaming (type: GATHER)          | 834.264               |   18000 |   17644 | 228KB                 |          |         |       4 | 21016.93
  3 |       ->  Vector Hash Aggregate                 | [585.017,758.204]     |   18000 |   17644 | [262552KB, 262564KB]  | 16MB     | [8,8]   |       4 | 20327.72
  4 |          ->  CStore Scan on store_sales         | [12.540,13.941]       | 2880404 | 2880404 | [490KB, 490KB]        | 1MB      |         |       4 | 16683.10
(4 rows)
```

14.3.4.7 数据倾斜调优

数据倾斜问题是分布式架构的重要难题，它破坏了MPP架构中各个节点对等的要求，导致单节点（倾斜节点）所存储或者计算的数据量远大于其他节点，所以会造成以下危害：

- 存储上的倾斜会严重限制系统容量，在系统容量不饱和的情况下，由于单节点倾斜的限制，使得整个系统容量无法继续增长。
- 计算上的倾斜会严重影响系统性能，由于倾斜节点所需要运算的数据量远大于其它节点，导致倾斜节点降低系统整体性能。
- 数据倾斜还严重影响了MPP架构的扩展性。由于在存储或者计算时，往往会将相同值的数据放到同一节点，因此当倾斜数据（大量数据的值相同）出现之后，即使我们增加节点，系统瓶颈仍然受限于倾斜节点的容量或者性能。

GaussDB(DWS)数据库针对数据倾斜问题给出了完整的解决方案，包括存储倾斜和计算倾斜两大问题，下面分别进行介绍。

存储层数据倾斜

GaussDB(DWS)数据库中，数据分布存储在各个DN上，通过分布式执行提高查询的效率。但是，如果数据分布存在倾斜，则会导致分布式执行某些DN成为瓶颈，影响查询性能。这种情况通常是由于分布列选择不合理，可以通过调整分布列的方式解决。

例如下例：

```
explain performance select count(*) from inventory;
5 --CStore Scan on lmz.inventory
        dn_6001_6002 (actual time=0.444..83.127 rows=42000000 loops=1)
        dn_6003_6004 (actual time=0.512..63.554 rows=27000000 loops=1)
        dn_6005_6006 (actual time=0.722..99.033 rows=45000000 loops=1)
        dn_6007_6008 (actual time=0.529..100.379 rows=51000000 loops=1)
```

```
      dn_6009_6010 (actual time=0.382..71.341 rows=36000000 loops=1)
      dn_6011_6012 (actual time=0.547..100.274 rows=51000000 loops=1)
      dn_6013_6014 (actual time=0.596..118.289 rows=60000000 loops=1)
      dn_6015_6016 (actual time=1.057..132.346 rows=63000000 loops=1)
      dn_6017_6018 (actual time=0.940..110.310 rows=54000000 loops=1)
      dn_6019_6020 (actual time=0.231..41.198 rows=21000000 loops=1)
      dn_6021_6022 (actual time=0.927..114.538 rows=54000000 loops=1)
      dn_6023_6024 (actual time=0.637..118.385 rows=60000000 loops=1)
      dn_6025_6026 (actual time=0.288..32.240 rows=15000000 loops=1)
      dn_6027_6028 (actual time=0.566..118.096 rows=60000000 loops=1)
      dn_6029_6030 (actual time=0.423..82.913 rows=42000000 loops=1)
      dn_6031_6032 (actual time=0.395..78.103 rows=39000000 loops=1)
      dn_6033_6034 (actual time=0.376..51.052 rows=24000000 loops=1)
      dn_6035_6036 (actual time=0.569..79.463 rows=39000000 loops=1)
```

在performance信息中，可以看到inventory表各DN的scan行数，发现各DN的行数差距较大，最大的为63000000，最小的只有15000000，差了4倍。这个差距对于数据扫描的性能影响还可以接受，但如果上层有join算子，则影响较大。

通常，数据表在各DN上是hash分布的，因此分布列的选择很重要。通过table_skewness()来查看上述inventory表在各DN的数据分布倾斜，查询结果如下：

```
select table_skewness('inventory');
              table_skewness
-------------------------------------------
 ("dn_6015_6016       ",63000000,8.046%)
 ("dn_6013_6014       ",60000000,7.663%)
 ("dn_6023_6024       ",60000000,7.663%)
 ("dn_6027_6028       ",60000000,7.663%)
 ("dn_6017_6018       ",54000000,6.897%)
 ("dn_6021_6022       ",54000000,6.897%)
 ("dn_6007_6008       ",51000000,6.513%)
 ("dn_6011_6012       ",51000000,6.513%)
 ("dn_6005_6006       ",45000000,5.747%)
 ("dn_6001_6002       ",42000000,5.364%)
 ("dn_6029_6030       ",42000000,5.364%)
 ("dn_6031_6032       ",39000000,4.981%)
 ("dn_6035_6036       ",39000000,4.981%)
 ("dn_6009_6010       ",36000000,4.598%)
 ("dn_6003_6004       ",27000000,3.448%)
 ("dn_6033_6034       ",24000000,3.065%)
 ("dn_6019_6020       ",21000000,2.682%)
 ("dn_6025_6026       ",15000000,1.916%)
(18 rows)
```

通过查询建表定义，可以发现，目前该表是以inv_date_sk作为分布列的，导致存在倾斜。通过查看各列的数据分布情况，改为inv_item_sk作为分布列，则倾斜情况分布如下：

```
select table_skewness('inventory');
              table_skewness
-------------------------------------------
 ("dn_6001_6002       ",43934200,5.611%)
 ("dn_6007_6008       ",43829420,5.598%)
 ("dn_6003_6004       ",43781960,5.592%)
 ("dn_6031_6032       ",43773880,5.591%)
 ("dn_6033_6034       ",43763280,5.589%)
 ("dn_6011_6012       ",43683600,5.579%)
 ("dn_6013_6014       ",43551660,5.562%)
 ("dn_6027_6028       ",43546340,5.561%)
 ("dn_6009_6010       ",43508700,5.557%)
 ("dn_6023_6024       ",43484540,5.554%)
 ("dn_6019_6020       ",43466800,5.551%)
 ("dn_6021_6022       ",43458500,5.550%)
 ("dn_6017_6018       ",43448040,5.549%)
 ("dn_6015_6016       ",43247700,5.523%)
 ("dn_6005_6006       ",43200240,5.517%)
 ("dn_6029_6030       ",43181360,5.515%)
```

```
 ("dn_6025_6026        ",43179700,5.515%)
 ("dn_6035_6036        ",42960080,5.487%)
(18 rows)
```

数据分布倾斜的问题得到解决。

除了table_skewness()视图外，当前版本还提供了table_distribution函数和**PGXC_GET_TABLE_SKEWNESS**视图，可以更加高效的查询各表的数据倾斜情况。

计算层数据倾斜

即使通过修改表的分布键，使得数据存储在各个节点上是均衡的，但是在执行查询的过程中，仍然可能出现数据倾斜的问题。在运算过程中某个算子在DN上输出的结果集出现倾斜，从而导致此算子上层的运算出现计算倾斜。一般来说，这是由于在执行过程中，数据重分布导致的。

在查询执行的过程中，join key、group by key等往往不是表的分布列，因此需要按照join key、group by key上数据的hash值，让数据在各个DN之间进行重新分布，这个过程对应于计划中的Redistribute算子。当重分布列上的数据存在倾斜时，就会导致运行时的数据倾斜，即重分布后部分节点的数据远远大于其他。倾斜节点需要处理更多的数据，导致倾斜节点的计算性能远远低于其他节点。

如下例中，s表和t表join，join条件中的s.x和t.x均不是表的分布列，因此需要重分布（REDISTRIBUTE算子）。其中s.x列上存在倾斜值，t.x上不存在倾斜。id=6的stream算子在datanode2节点输出的结果集是其他DN的3倍，从而导致了计算倾斜。

```
select * from skew s,test t where s.x = t.x order by s.a limit 1;
 id |                    operation                    |        A-time
----+-------------------------------------------------+----------------------
  1 | ->  Limit                                       | 52622.382
  2 |    ->  Streaming (type: GATHER)                 | 52622.374
  3 |       ->  Limit                                 | [30138.494,52598.994]
  4 |         ->  Sort                                | [30138.486,52598.986]
  5 |           ->  Hash Join (6,8)                   | [30127.013,41483.275]
  6 |             ->  Streaming(type: REDISTRIBUTE)   | [11365.110,22024.845]
  7 |               ->  Seq Scan on public.skew s     | [2019.168,2175.369]
  8 |             ->  Hash                            | [2460.108,2499.850]
  9 |               ->  Streaming(type: REDISTRIBUTE) | [1056.214,1121.887]
 10 |                 ->  Seq Scan on public.test t   | [310.848,325.569]
(10 rows)
 6 --Streaming(type: REDISTRIBUTE)
        datanode1 (rows=5050368)
        datanode2 (rows=15276032)
        datanode3 (rows=5174272)
        datanode4 (rows=5219328)
```

和存储倾斜相比，计算倾斜更难以提前识别，因此GaussDB提出了RLBT(Runtime Load Balance Technology)方案，用以解决运行时的计算倾斜问题，该特性由参数**skew_option**控制。RLBT方案主要分为两个层面，第一步是计算倾斜识别，第二步是计算倾斜解决。下面分别进行介绍。

1. 倾斜识别

   计算倾斜的识别，即预先识别计算过程中的重分布列是否存在倾斜数据。RLBT方案中给出了三个解决手段，统计信息识别，hint方式指定以及规则识别：

   - 统计信息识别

     需要用户先执行analyze收集各表的统计信息，然后优化器能够自动利用统计信息对重分布键上的倾斜数据进行提前识别，对于存在倾斜的查询，生成相应的优化计划。在重分布键有多列的情况，只有所有列都属于同一个基表才能利用统计信息进行识别。

统计信息只能给出基表的倾斜情况，当基表某一列存在倾斜，其他列上带有过滤条件，或者经过和其他表的join之后，我们无法准确判断倾斜列上倾斜数据是否依旧存在。当**skew_option**为normal时，这里认为倾斜数据依旧存在，仍然会对基表中识别到的倾斜进行优化；当**skew_option**为lazy时，这里认为倾斜数据已经不再存在，也就不会进行相应的优化。

- hint方式指定

  统计信息有着一定的局限性，对于较为复杂的查询，其中间结果难以通过统计信息进行估算和识别倾斜数据。对于这种情况，我们设计了hint手段，通过用户手动指定的方式，给定倾斜信息。优化器根据用户给定的倾斜信息，来对查询进行优化。详细hint使用语法参见**运行倾斜的hint**。

- 规则识别

  现在BI系统往往会产生大量带有outer join（left join、right join、full join）的SQL，outer join在匹配失败的情况下会补空产生大量NULL值，如果接下来在补空列上进行join或者group by操作，就会导致NULL值倾斜。当前RLBT技术会自动识别这种场景，并生成相应的NULL值倾斜优化计划。

2. 计算倾斜解决

   在解决倾斜时，目前针对最常见的join和agg算子进行了优化。

   - join优化

   基本思路是将倾斜数据和非倾斜数据进行隔离处理。主要分为以下三种情况：

   a. join两侧都需要做重分布：

      对倾斜侧做PART_REDISTRIBUTE_PART_ROUNDROBIN，其中对倾斜数据做roundrobin，非倾斜数据做redistribute；

      对非倾斜侧做PART_REDISTRIBUTE_PART_BROADCAST，其中对倾斜数据做broadcast，非倾斜数据做redistribute；

   b. join一侧需要重分布，另一侧不需要重分布：

      对需要重分布的一侧做PART_REDISTRIBUTE_PART_ROUNDROBIN；

      对不需要重分布的一侧做PART_LOCAL_PART_BROADCAST，其中对等于倾斜值的部分做broadcast，其余数据保留在本地。

   c. 对于有补NULL值的表：

      对该表做PART_REDISTERIBUTE_PART_LOCAL，其中将NULL值保留在本地，其余数据做redistribute。

   以前面的查询为例，s.x列上存在倾斜数据，倾斜数据的值为0。优化器通过统计信息，识别到了该倾斜数据，生成了倾斜优化计划如下：

```
 id |                                operation                                 |        A-time
----+--------------------------------------------------------------------------+-----------------------
  1 | ->  Limit                                                                | 23642.049
  2 |    ->  Streaming (type: GATHER)                                          | 23642.041
  3 |       ->  Limit                                                          | [23310.768,23618.021]
  4 |          ->  Sort                                                        | [23310.761,23618.012]
  5 |             ->  Hash Join (6,8)                                          | [20898.341,21115.272]
  6 |                ->  Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)    |
[7125.834,7472.111]
  7 |                   ->  Seq Scan on public.skew s                          | [1837.079,1911.025]
  8 |                ->  Hash                                                  | [2612.484,2640.572]
  9 |                   ->  Streaming(type: PART REDISTRIBUTE PART BROADCAST)  | [1193.548,1297.894]
 10 |                      ->  Seq Scan on public.test t                       | [314.343,328.707]
(10 rows)
  5 --Vector Hash Join (6,8)
      Hash Cond: s.x = t.x
      Skew Join Optimizated by Statistic
  6 --Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)
      datanode1  (rows=7635968)
```

```
       datanode2 (rows=7517184)
       datanode3 (rows=7748608)
       datanode4 (rows=7818240)
```

上述执行计划中，可以看到Skew Join Optimizated by Statistic的字样，代表该计划为倾斜优化计划，其中Statistic关键字代表该倾斜优化来自于统计信息，除此之外还有Hint和Rule，分别代表倾斜优化来自于hint语句和规则。对比前面的计划可以看到，这里对于非倾斜数据和倾斜数据做了分别处理。对于s表中的非倾斜数据，依旧按照原有的方案，根据数据的hash值进行重分布；而对于倾斜数据（即等于0的数据），则通过轮询发送的方式，均衡地发送到所有节点。通过这样的方式，解决了倾斜数据分布不均衡的问题。

同时，为了保证结果的正确性，需要对t表做相应的处理。对于t表中等于0（s.x表中的倾斜值）的数据做广播，对于其他数据，依旧根据数据的hash值进行重分布。

通过这样的方式，就解决了join操作中，数据倾斜的问题。从上面的结果来看，id=6的stream算子各个DN的输出结果已经非常均衡，同时查询端到端性能提升了1倍。

- agg优化

对于agg操作，解决倾斜的思路与join操作不同，这里是通过首先在本DN内按照group by key进行去重操作，然后再进行重分布。因为经过DN内部去重之后，从全局来看，每个值的数量都不会超过DN数，因此不会出现严重的数据倾斜问题。以如下query为例：

```
select c1, c2, c3, c4, c5, c6, c7, c8, c9, count(*) from t group by c1, c2, c3, c4, c5, c6, c7, c8, c9 limit 10;
```

原执行结果如下：

```
 id |                operation                 |        A-time         | A-rows
----+------------------------------------------+-----------------------+----------
  1 | ->  Streaming (type: GATHER)             | 130621.783            |      12
  2 |    ->  GroupAggregate                    | [85499.711,130432.341] |      12
  3 |       ->  Sort                           | [85499.509,103145.632] | 36679237
  4 |         ->  Streaming(type: REDISTRIBUTE) | [25668.897,85499.050]  | 36679237
  5 |           ->  Seq Scan on public.t       | [9835.069,10416.388]   | 36679237
(5 rows)

   4 --Streaming(type: REDISTRIBUTE)
       datanode1 (rows=36678837)
       datanode2 (rows=100)
       datanode3 (rows=100)
       datanode4 (rows=200)
```

其中存在大量倾斜数据，导致数据按照group by key进行重分布之后，datanode1的数据量是其他节点的数十万倍。在倾斜优化之后，首先在本DN进行一次group by操作，达到数据去重的效果，然后再进行重分布，可以发现基本没有数据倾斜的问题出现。

```
 id |                operation                 |        A-time
----+------------------------------------------+-----------------------
  1 | ->  Streaming (type: GATHER)             | 10961.337
  2 |    ->  HashAggregate                     | [10953.014,10953.705]
  3 |       ->  HashAggregate                  | [10952.957,10953.632]
  4 |         ->  Streaming(type: REDISTRIBUTE) | [10952.859,10953.502]
  5 |           ->  HashAggregate              | [10084.280,10947.139]
  6 |             ->  Seq Scan on public.t     | [4757.031,5201.168]
(6 rows)

 Predicate Information (identified by plan id)
-----------------------------------------------
   3 --HashAggregate
       Skew Agg Optimized by Statistic
(2 rows)

   4 --Streaming(type: REDISTRIBUTE)
       datanode1 (rows=17)
```

```
    datanode2 (rows=8)
    datanode3 (rows=8)
    datanode4 (rows=14)
```

适用范围

- join算子
  - 支持nest loop，merge join，hash join等join方式；
  - 当倾斜数据处于join的left侧时，支持inner join，left join，semi join，anti join；当倾斜属于位于join的right侧时，支持inner join，right join，right semi join，right anti join。
  - 通过统计信息得到的倾斜优化计划，优化器会根据代价判断该计划是否为最优计划。通过hint和规则会强制生成倾斜优化计划。
- agg算子
  - array_agg、string_agg、subplan in agg qual这几种场景不支持优化；
  - 通过统计信息识别到的倾斜优化计划会受到代价、plan_mode_seed参数、best_agg_plan参数影响，而hint、规则识别到的不会。

14.3.5 SQL 调优关键参数调整

本节将介绍影响GaussDB(DWS) SQL调优性能的关键CN配置参数，配置方法参见数据库参数一节。

表 14-2 CN 配置参数

参数/参考值	描述
enable_nestloop=off	控制查询优化器对嵌套循环连接（Nest Loop Join）类型的使用。当设置为“on”后，优化器优先使用Nest Loop Join；当设置为“off”后，优化器在存在其他方法时将优先选择其他方法。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行如下命令： `SET enable_nestloop to on;` 此参数默认设置为“off”，但实际调优中应根据情况选择是否打开。一般情况下，当Nest Loop的内表可以使用索引时，可以选择打开。
enable_bitmapscan=on	控制查询优化器对位图扫描规划类型的使用。设置为“on”，表示使用；设置为“off”，表示不使用。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行命令如下命令： `SET enable_bitmapscan to off;` bitmapscan扫描方式适用于“where a > 1 and b > 1”且a列和b列都有索引这种查询条件，但有时其性能不如indexscan。因此，现场调优如发现查询性能较差且计划中有bitmapscan算子，可以关闭bitmapscan，看性能是否有提升。

参数/参考值	描述
enable_fast_query_shipping=on	控制查询优化器是否使用分布式框架，执行快速执行计划。设置为“on”，表示执行计划在CN和DN上各自生成；设置为“off”，表示使用分布式框架，即执行计划在CN上生成，然后发送到DN中执行。 **说明** 如果只需要在当前数据库连接（即当前Session）中临时更改该参数值，则只需要在SQL语句中执行如下命令： `SET enable_fast_query_shipping to off;`
enable_hashagg=on	控制优化器对Hash聚集规划类型的使用。
enable_hashjoin=on	控制优化器对Hash连接规划类型的使用。
enable_mergejoin=on	控制优化器对融合连接规划类型的使用。
enable_indexscan=on	控制优化器对索引扫描规划类型的使用。
enable_indexonlyscan=on	控制优化器对仅索引扫描规划类型的使用。
enable_seqscan=on	控制优化器对顺序扫描规划类型的使用。完全消除顺序扫描是不可能的，但是关闭这个变量会让优化器在存在其他方法的时候优先选择其他方法。
enable_sort=on	控制优化器使用的排序步骤。该设置不可能完全消除明确的排序，但是关闭这个变量可以让优化器在存在其他方法的时候优先选择其他方法。
enable_broadcast=on	控制查询优化器对于broadcast广播模式数据传输的使用。此方式网络传输数据量较大，因此当网络传输节点（Stream）实际数据量较大而估算不准时，可以将该参数设置为off，看性能是否有提升。
rewrite_rule	控制优化器是否启用LAZY_AGG和MAGIC_SET重写规则。

14.3.6 使用 Plan Hint 进行调优

14.3.6.1 Plan Hint 调优概述

Plan Hint为用户提供了直接影响执行计划生成的手段，用户可以通过指定join顺序，join、stream、scan方法，指定结果行数，指定重分布过程中的倾斜信息，指定配置参数的值等多个手段来进行执行计划的调优，以提升查询的性能。

功能描述

Plan Hint仅支持在SELECT关键字后通过如下形式指定：

```
/*+ <plan hint>*/
```

可以同时指定多个hint，之间使用空格分隔。hint只能hint当前层的计划，对于子查询计划的hint，需要在子查询的select关键字后指定hint。

例如：

```
select /*+ <plan_hint1> <plan_hint2> */ * from t1, (select /*+ <plan_hint3> */ from t2) where 1=1;
```

其中<plan_hint1>，<plan_hint2>为外层查询的hint，<plan_hint3>为内层子查询的hint。

须知

如果在视图定义（CREATE VIEW）时指定hint，则在该视图每次被应用时会使用该hint。

当使用random plan功能（参数plan_mode_seed不为0）时，查询指定的plan hint不会被使用。

支持范围

当前版本Plan Hint支持的范围如下，后续版本会进行增强。

- 指定Join顺序的Hint - leading hint
- 指定Join方式的Hint，仅支持除semi/anti join，unique plan之外的常用hint。
- 指定结果集行数的Hint
- 指定Stream方式的Hint
- 指定Scan方式的Hint，仅支持常用的tablescan，indexscan和indexonlyscan的hint。
- 指定子链接块名的Hint
- 指定倾斜信息的Hint，仅支持Join与HashAgg的重分布过程倾斜。
- 指定配置参数值的Hint，仅支持best_agg_plan、enable_nodegroup_debug、expected_computing_nodegroup和query_dop的hint。（仅8.0.0.5及以上版本支持）

注意事项

- 不支持Agg、Sort、Setop和Subplan的hint。
- 不支持SMP和Node Group场景下的Hint。

示例

本章节使用同一个语句进行示例，便于Plan Hint支持的各方法作对比，示例语句及不带hint的原计划如下所示：

```
explain
select i_product_name product_name
,i_item_sk item_sk
,s_store_name store_name
,s_zip store_zip
,ad2.ca_street_number c_street_number
,ad2.ca_street_name c_street_name
,ad2.ca_city c_city
,ad2.ca_zip c_zip
```

```
,count(*) cnt
,sum(ss_wholesale_cost) s1
,sum(ss_list_price) s2
,sum(ss_coupon_amt) s3
FROM   store_sales
,store_returns
,store
,customer
,promotion
,customer_address ad2
,item
WHERE  ss_store_sk = s_store_sk AND
ss_customer_sk = c_customer_sk AND
ss_item_sk = i_item_sk and
ss_item_sk = sr_item_sk and
ss_ticket_number = sr_ticket_number and
c_current_addr_sk = ad2.ca_address_sk and
ss_promo_sk = p_promo_sk and
i_color in ('maroon','burnished','dim','steel','navajo','chocolate') and
i_current_price between 35 and 35 + 10 and
i_current_price between 35 + 1 and 35 + 15
group by i_product_name
,i_item_sk
,s_store_name
,s_zip
,ad2.ca_street_number
,ad2.ca_street_name
,ad2.ca_city
,ad2.ca_zip
;
```

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |  E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 3401632.49
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 3401632.49
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 3401630.82
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 3401630.78
  5 |             ->  Vector Hash Join (6,21)                                  |          6 | 16MB     |     169 | 3401630.42
  6 |                ->  Vector Hash Join (7,20)                               |          7 | 43MB     |     173 | 3400343.15
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)               |          7 | 1MB      |     123 | 3395775.64
  8 |                      ->  Vector Hash Join (9,19)                         |          7 | 27MB     |     123 | 3395775.48
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 3386294.72
 10 |                            ->  Vector Hash Join (11,18)                  |          7 | 16MB     |     123 | 3386294.56
 11 |                               ->  Vector Hash Join (12,14)               |          7 | 19MB     |     112 | 3384018.02
 12 |                                  ->  Vector Partition Iterator           |  287999764 | 1MB      |      12 | 227383.99
 13 |                                     ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      12 | 227383.99
 14 |                                  ->  Vector Hash Join (15,17)            |    1516824 | 16MB     |     124 | 3065686.08
 15 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 16 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |      66 | 2756066.50
 17 |                                     ->  CStore Scan on item              |        158 | 1MB      |      58 | 4051.25
 18 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 20 |                      ->  CStore Scan on customer_address ad2             |    6000000 | 1MB      |      58 | 5770.00
 21 |                   ->  CStore Scan on promotion                           |      36000 | 1MB      |       4 | 1268.50
(21 rows)
```

14.3.6.2 Join 顺序的 Hint

功能描述

指明join的顺序，包括内外表顺序。

语法格式

- 仅指定join顺序，不指定内外表顺序。

```
leading(join_table_list)
```

- 同时指定join顺序和内外表顺序，内外表顺序仅在最外层生效。

```
leading((join_table_list))
```

参数说明

join_table_list为表示表join顺序的hint字符串，可以包含当前层的任意个表（别名），或对于子查询提升的场景，也可以包含子查询的hint别名，同时任意表可以使用括号指定优先级，表之间使用空格分隔。

须知

表只能用单个字符串表示，不能带schema。

表如果存在别名，需要优先使用别名来表示该表。

join table list中指定的表需要满足以下要求，否则会报语义错误。

- list中的表必须在当前层或提升的子查询中存在。
- list中的表在当前层或提升的子查询中必须是唯一的。如果不唯一，需要使用不同的别名进行区分。
- 同一个表只能在list里出现一次。
- 如果表存在别名，则list中的表需要使用别名。

例如：

leading(t1 t2 t3 t4 t5)表示：t1, t2, t3, t4, t5先join，五表join顺序及内外表不限。

leading((t1 t2 t3 t4 t5))表示：t1和t2先join，t2做内表；再和t3 join，t3做内表；再和t4 join，t4做内表；再和t5 join，t5做内表。

leading(t1 (t2 t3 t4) t5)表示：t2, t3, t4先join，内外表不限；再和t1, t5 join，内外表不限。

leading((t1 (t2 t3 t4) t5))表示：t2, t3, t4先join，内外表不限；在最外层，t1再和t2, t3, t4的join表join，t1为外表，再和t5 join，t5为内表。

leading((t1 (t2 t3) t4 t5)) leading((t3 t2))表示：t2, t3先join，t2做内表；然后再和t1 join，t2, t3的join表做内表；然后再依次跟t4, t5做join，t4, t5做内表。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ leading((((((store_sales store) promotion) item) customer) ad2) store_returns) leading((store store_sales))*/ i_product_name product_name ...
```

该hint表示：表之间的join关系是：store_sales和store先join， store_sales做内表，然后依次跟promotion, item, customer, ad2, store_returns做join。生成计划如下所示：

```
WARNING:  Duplicated or conflict hint: Leading(store_sales store), will be discarded.
 id |                                  operation                                  |   E-rows   | E-memory | E-width |   E-costs
----+-----------------------------------------------------------------------------+------------+----------+---------+-------------
  1 | ->  Row Adapter                                                             |          6 |          |     273 | 16308094.34
  2 |    ->  Vector Streaming (type: GATHER)                                      |          6 |          |     273 | 16308094.34
  3 |       ->  Vector Hash Aggregate                                             |          6 | 16MB     |     273 | 16308092.67
  4 |          ->  Vector Hash Join (5,20)                                        |          6 | 585MB    |     169 | 16308092.63
  5 |             ->  Vector Streaming(type: REDISTRIBUTE)                        |    1320811 | 1MB      |     181 | 16069870.93
  6 |                ->  Vector Hash Join (7,19)                                  |    1320811 | 43MB     |     181 | 16061891.00
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)                  |    1320811 | 1MB      |     131 | 16056566.78
  8 |                      ->  Vector Hash Join (9,18)                            |    1320811 | 27MB     |     131 | 16048586.85
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)            |    1383248 | 1MB      |     131 | 16038321.62
 10 |                            ->  Vector Hash Join (11,17)                     |    1383248 | 16MB     |     131 | 16029664.50
 11 |                               ->  Vector Hash Join (12,16)                  | 2626366951 | 16MB     |      73 | 15751384.88
 12 |                                  ->  Vector Hash Join (13,14)               | 2750085660 | 2156MB   |      77 | 14226077.19
 13 |                                     ->  CStore Scan on store                |      24048 | 1MB      |      19 | 2264.00
 14 |                                     ->  Vector Partition Iterator           | 2879987999 | 1MB      |      66 | 2756066.50
 15 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB      |      66 | 2756066.50
 16 |                                  ->  CStore Scan on promotion               |      36000 | 1MB      |       4 | 1268.50
 17 |                               ->  CStore Scan on item                       |        158 | 1MB      |      58 | 4051.25
 18 |                         ->  CStore Scan on customer                         |   12000000 | 1MB      |       8 | 12923.00
 19 |                      ->  CStore Scan on customer_address ad2                |    6000000 | 1MB      |      58 | 5770.00
 20 |               ->  Vector Partition Iterator                                 |  287999764 | 1MB      |      12 | 227383.99
 21 |                  ->  Partitioned CStore Scan on store_returns               |  287999764 | 1MB      |      12 | 227383.99
(21 rows)
```

图中计划顶端warning的提示详见**Hint的错误、冲突及告警**的说明。

14.3.6.3 Join 方式的 Hint

功能描述

指明Join使用的方法，可以为Nested Loop，Hash Join和Merge Join。

语法格式

```
[no] nestloop|hashjoin|mergejoin(table_list)
```

参数说明

- **no**表示hint的join方式不使用。
- **table_list**为表示hint表集合的字符串，该字符串中的表与**join_table_list**相同，只是中间不允许出现括号指定join的优先级。

例如：

no nestloop(t1 t2 t3)表示：生成t1，t2，t3三表连接计划时，不使用nestloop。三表连接计划可能是t2 t3先join，再跟t1 join，或t1 t2先join，再跟t3 join。此hint只hint最后一次join的join方式，对于两表连接的方法不hint。如果需要，可以单独指定，例如：任意表均不允许nestloop连接，且希望t2 t3先join，则增加hint：no nestloop(t2 t3)。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ nestloop(store_sales store_returns item) */ i_product_name product_name ...
```

该hint表示：生成store_sales，store_returns和item三表的结果集时，最后的两表关联使用nestloop。生成计划如下所示：

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |     E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 100061693161.06
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 100061693161.06
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 100061693159.40
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 100061693159.36
  5 |             ->  Vector Hash Join (6,22)                                  |          6 | 43MB     |     169 | 100061693158.99
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                  |          6 | 1MB      |     119 | 100061688591.48
  7 |                   ->  Vector Hash Join (8,21)                            |          6 | 16MB     |     119 | 100061688591.30
  8 |                      ->  Vector Hash Join (9,20)                         |          7 | 27MB     |     123 | 100061687304.04
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 100061677823.27
 10 |                            ->  Vector Hash Join (11,19)                  |          7 | 16MB     |     123 | 100061677823.12
 11 |                               ->  Vector Nest Loop (12,17)               |          7 | 1MB      |     112 | 100061675546.57
 12 |                                  ->  Vector Hash Join (13,15)            |      13670 | 585MB    |      62 | 6163443.54
 13 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 14 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |  66 | 2756066.50
 15 |                                     ->  Vector Partition Iterator        |  287999764 | 1MB      |      12 | 227383.99
 16 |                                        ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB | 12 | 227383.99
 17 |                                  ->  Vector Materialize                  |        158 | 16MB     |      58 | 4051.28
 18 |                                     ->  CStore Scan on item              |        158 | 1MB      |      58 | 4051.25
 19 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 20 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 21 |                         ->  CStore Scan on promotion                     |      36000 | 1MB      |       4 | 1268.50
 22 |                      ->  CStore Scan on customer_address ad2             |    6000000 | 1MB      |      58 | 5770.00
(22 rows)
```

14.3.6.4 行数的 Hint

功能描述

指明中间结果集的大小，支持绝对值和相对值的hint。

语法格式

```
rows(table_list #|+|-|* const)
```

参数说明

- **#,+,-,***，进行行数估算hint的四种操作符号。#表示直接使用后面的行数进行hint。+,-,*表示对原来估算的行数进行加、减、乘操作，运算后的行数最小值为1行。table_list为hint对应的单表或多表join结果集，与**Join方式的Hint**中**table_list**相同。
- **const**可以是任意非负数，支持科学计数法。

例如：

rows(t1 #5)表示：指定t1表的结果集为5行。

rows(t1 t2 t3 *1000)表示：指定t1, t2, t3 join完的结果集的行数乘以1000。

建议

- 推荐使用两个表*的hint。对于两个表的采用*操作符的hint，只要两个表出现在join的两端，都会触发hint。例如：设置hint为rows(t1 t2 * 3)，对于(t1 t3 t4)和(t2 t5 t6)join时，由于t1和t2出现在join的两端，所以其join的结果集也会应用该hint规则乘以3。
- rows hint支持在单表、多表、function table及subquery scan table的结果集上指定hint。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ rows(store_sales store_returns *50) */ i_product_name product_name ...
```

该hint表示：store_sales，store_returns关联的结果集估算行数在原估算行数基础上乘以50。生成计划如下所示：

```
 id |                                operation                                |   E-rows   | E-memory | E-width |  E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                         |        312 |          |     273 | 3401656.58
  2 |    ->  Vector Streaming (type: GATHER)                                  |        312 |          |     273 | 3401656.58
  3 |       ->  Vector Hash Aggregate                                         |        312 | 16MB     |     273 | 3401634.91
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                       |        313 | 1MB      |     169 | 3401634.39
  5 |             ->  Vector Hash Join (6,21)                                 |        313 | 43MB     |     169 | 3401633.06
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                 |        313 | 1MB      |     119 | 3397065.38
  7 |                   ->  Vector Hash Join (8,20)                           |        313 | 27MB     |     119 | 3397064.31
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)           |        328 | 1MB      |     119 | 3387583.37
  9 |                         ->  Vector Hash Join (10,19)                    |        328 | 16MB     |     119 | 3387582.18
 10 |                            ->  Vector Hash Join (11,18)                 |        344 | 16MB     |     123 | 3386294.74
 11 |                               ->  Vector Hash Join (12,14)              |        360 | 19MB     |     112 | 3384018.02
 12 |                                  ->  Vector Partition Iterator          |  287999764 | 1MB      |      12 | 227383.99
 13 |                                     ->  Partitioned CStore Scan on store_returns |  287999764 | 1MB      |      12 | 227383.99
 14 |                                  ->  Vector Hash Join (15,17)           |    1516824 | 16MB     |     124 | 3065686.08
 15 |                                     ->  Vector Partition Iterator       | 2879987999 | 1MB      |      66 | 2756066.50
 16 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB      |      66 | 2756066.50
 17 |                                     ->  CStore Scan on item             |        158 | 1MB      |      58 | 4051.25
 18 |                                ->  CStore Scan on store                 |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on promotion                 |      36000 | 1MB      |       4 | 1268.50
 20 |                      ->  CStore Scan on customer                        |   12000000 | 1MB      |       8 | 12923.00
 21 |                ->  CStore Scan on customer_address ad2                  |    6000000 | 1MB      |      58 | 5770.00
(21 rows)
```

第11行算子的估算行数修正为360行，原估算行数为7行（四舍五入后取值）。

14.3.6.5 Stream 方式的 Hint

功能描述

指明stream使用的方法，可以为broadcast和redistribute。

语法格式

```
[no] broadcast|redistribute(table_list)
```

参数说明

- **no**表示hint的stream方式不使用。
- **table_list**为进行stream操作的单表或多表join结果集，见**参数说明**。

示例

对**示例**中原语句使用如下hint:

```
explain
select /*+ no redistribute(store_sales store_returns item store) leading(((store_sales store_returns item store) customer)) */ i_product_name product_name ...
```

原计划中，(store_sales store_returns item store)和customer做join时，前者做了重分布，此hint表示禁止前者混合表做重分布，但仍然保持join顺序，则生成计划如下所示：

```
 id |                                operation                                |   E-rows   | E-memory | E-width |  E-costs
----+-------------------------------------------------------------------------+------------+----------+---------+------------
  1 | ->  Row Adapter                                                         |          6 |          |     273 | 5718448.94
  2 |    ->  Vector Streaming (type: GATHER)                                  |          6 |          |     273 | 5718448.94
  3 |       ->  Vector Hash Aggregate                                         |          6 | 16MB     |     273 | 5718447.27
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                       |          6 | 1MB      |     169 | 5718447.23
  5 |             ->  Vector Hash Join (6,21)                                 |          6 | 16MB     |     169 | 5718446.86
  6 |                ->  Vector Hash Join (7,20)                              |          7 | 43MB     |     173 | 5717159.60
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)              |          7 | 1MB      |     123 | 5712592.09
  8 |                      ->  Vector Hash Join (9,18)                        |          7 | 585MB    |     123 | 5712591.93
  9 |                         ->  Vector Hash Join (10,17)                    |          7 | 16MB     |     123 | 3386294.56
 10 |                            ->  Vector Hash Join (11,13)                 |          7 | 19MB     |     112 | 3384018.02
 11 |                               ->  Vector Partition Iterator             |  287999764 | 1MB      |      12 | 227383.99
 12 |                                  ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      12 | 227383.99
 13 |                               ->  Vector Hash Join (14,16)              |    1516824 | 16MB     |     124 | 3065686.08
 14 |                                  ->  Vector Partition Iterator          | 2879987999 | 1MB      |      66 | 2756066.50
 15 |                                     ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |      66 | 2756066.50
 16 |                                  ->  CStore Scan on item                |        158 | 1MB      |      58 | 4051.25
 17 |                            ->  CStore Scan on store                     |      24048 | 1MB      |      19 | 2264.00
 18 |                         ->  Vector Streaming(type: BROADCAST)           |  288000000 | 1MB      |       8 | 2176297.36
 19 |                            ->  CStore Scan on customer                  |   12000000 | 1MB      |       8 | 12923.00
 20 |                      ->  CStore Scan on customer_address ad2            |    6000000 | 1MB      |      58 | 5770.00
 21 |                   ->  CStore Scan on promotion                          |      36000 | 1MB      |       4 | 1268.50
(21 rows)
```

14.3.6.6 Scan 方式的 Hint

功能描述

指明scan使用的方法，可以是tablescan、indexscan和indexonlyscan。

语法格式

```
[no] tablescan|indexscan|indexonlyscan(table [index])
```

参数说明

- **no**表示hint的scan方式不使用。
- **table**表示hint指定的表，只能指定一个表，如果表存在别名应优先使用别名进行hint。
- **index**表示使用indexscan或indexonlyscan的hint时，指定的索引名称，当前只能指定一个。

说明

对于indexscan或indexonlyscan，只有hint的索引属于hint的表时，才能使用该hint。

scan hint支持在行列存表、hdfs内外表、obs表、子查询表上指定。对于hdfs内表，由主表和delta表组成，delta表对用户不可见，故hint仅作用在主表上。

示例

为了hint使用索引扫描，需要首先在表item的i_item_sk列上创建索引，名称为i。

```
create index i on item(i_item_sk);
```

对**示例**中原语句使用如下hint:

```
explain
select /*+ indexscan(item i) */ i_product_name product_name ...
```

该hint表示：item表使用索引i进行扫描。生成计划如下所示：

```
 id |                                operation                                 |   E-rows   | E-memory | E-width |     E-costs
----+--------------------------------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                                          |          6 |          |     273 | 100061674938.26
  2 |    ->  Vector Streaming (type: GATHER)                                   |          6 |          |     273 | 100061674938.26
  3 |       ->  Vector Hash Aggregate                                          |          6 | 16MB     |     273 | 100061674936.59
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                        |          6 | 1MB      |     169 | 100061674936.55
  5 |             ->  Vector Hash Join (6,21)                                  |          6 | 43MB     |     169 | 100061674936.19
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                  |          6 | 1MB      |     119 | 100061670368.67
  7 |                   ->  Vector Hash Join (8,20)                            |          6 | 16MB     |     119 | 100061670368.50
  8 |                      ->  Vector Hash Join (9,19)                         |          7 | 27MB     |     123 | 100061669081.23
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)         |          7 | 1MB      |     123 | 100061659600.47
 10 |                            ->  Vector Hash Join (11,18)                  |          7 | 16MB     |     123 | 100061659600.31
 11 |                               ->  Vector Nest Loop (12,17)               |          7 | 1MB      |     112 | 100061657323.77
 12 |                                  ->  Vector Hash Join (13,15)            |      13670 | 585MB    |      62 | 6163443.54
 13 |                                     ->  Vector Partition Iterator        | 2879987999 | 1MB      |      66 | 2756066.50
 14 |                                        ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |  66 | 2756066.50
 15 |                                     ->  Vector Partition Iterator        |  287999764 | 1MB      |      12 | 227383.99
 16 |                                        ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |  12 | 227383.99
 17 |                                  ->  CStore Index Scan using i on item   |          1 | 1MB      |      58 | 4.01
 18 |                               ->  CStore Scan on store                   |      24048 | 1MB      |      19 | 2264.00
 19 |                            ->  CStore Scan on customer                   |   12000000 | 1MB      |       8 | 12923.00
 20 |                         ->  CStore Scan on promotion                     |      36000 | 1MB      |       4 | 1268.50
 21 |                   ->  CStore Scan on customer_address ad2                |    6000000 | 1MB      |      58 | 5770.00
(21 rows)
```

14.3.6.7 子链接块名的 hint

功能描述

指明子链接块的名称。

语法格式

```
blockname (table)
```

参数说明

- **table**表示为该子链接块hint的别名的名称。

说明

- **blockname hint**仅在对应的子链接块提升时才会被上层查询使用。目前支持的子链接提升包括IN子链接提升、EXISTS子链接提升和包含Agg等值相关子链接提升。该hint通常会和前面章节提到的hint联合使用。
- 对于FROM关键字后的子查询，则需要使用子查询的别名进行hint，blockname hint不会被用到。
- 如果子链接中含有多个表，则提升后这些表可与外层表以任意优化顺序连接，hint也不会被用到。

示例

```
explain select /*+nestloop(store_sales tt) */ * from store_sales where ss_item_sk in (select /*
+blockname(tt)*/ i_item_sk from item group by 1);
```

该hint表示：子链接的别名为tt，提升后与上层的store_sales表关联时使用nestloop。生成计划如下所示：

```
 id |                    operation                    |   E-rows   | E-memory | E-width |     E-costs
----+-------------------------------------------------+------------+----------+---------+-----------------
  1 | ->  Row Adapter                                 | 1439994000 |          |     216 | 325105765847.91
  2 |    ->  Vector Streaming (type: GATHER)          | 1439994000 |          |     216 | 325105765847.91
  3 |       ->  Vector Nest Loop Semi Join (4, 6)     | 1439994000 | 1MB      |     216 | 325026664615.00
  4 |          ->  Vector Partition Iterator          | 2879987999 | 1MB      |     216 | 2756066.50
  5 |             ->  Partitioned CStore Scan on store_sales | 2879987999 | 1MB |  216 | 2756066.50
  6 |          ->  Vector Materialize                 |     300000 | 16MB     |       4 | 4176.25
  7 |             ->  Vector Hash Aggregate           |     300000 | 16MB     |       4 | 3988.75
  8 |                ->  CStore Scan on item          |     300000 | 1MB      |       4 | 3832.50
(8 rows)
```

14.3.6.8 运行倾斜的 hint

功能描述

指明查询运行时重分布过程中存在倾斜的重分布键和倾斜值，针对Join和HashAgg运算中的重分布进行优化。

语法格式

- 指定单表倾斜：
  ```
  skew(table (column) [(value)])
  ```
- 指定中间结果倾斜：
  ```
  skew((join_rel) (column) [(value)])
  ```

参数说明

- **table**表示存在倾斜的单个表名。
- **join_rel**表示参与join的两个或多个表，如（t1 t2）表示t1和t2join后的结果存在倾斜。
- **column**表示倾斜表中存在倾斜的一个或多个列。
- **value**表示倾斜的列中存在倾斜的一个或多个值。

说明

- skew hint仅在需要重分布且指定的倾斜信息与查询执行过程中的重分布信息相匹配时才会被使用。
- skew hint受GUC参数**skew_option**限制，如果参数处于关闭状态，则无法进行skew hint倾斜调优。
- skew hint目前仅处理普通表和子查询类型的表关系，支持基表hint、子查询hint、with as子句hint。对于子查询，无论提升与否都支持在skew hint中使用，这点与其它hint不一样。
- 对于倾斜表，如果定义了别名，则在hint中必须使用别名。
- 对于倾斜列，在不产生歧义的情况下，可以使用原名也可以使用别名。skew hint的column不支持表达式，如果需要指定采用分布键为表达式的重分布存在倾斜，需要将重分布键指定为新的列，以新的列进行hint。
- 对于倾斜值，个数需为列数的整数倍并按列的顺序进行组合，组合的个数不能超过10个。如果各倾斜列的倾斜值的个数不一样，为了满足按列组合，值可以重复指定。如，表t1的c1和c2存在倾斜，c1列的倾斜值只有a1，而c2列的倾斜有b1和b2，则skew hint如下：skew(t1 (c1 c2) ((a1 b1)(a1 b2)))。例中(a1 b1)为一个值组合，NULL可以作为倾斜值出现，每个hint中的值组合不超过十个， 且需为列的整数倍。
- 在Join的重分布优化中，skew hint中的value不可缺省，在HashAgg中可以缺省。
- 对于表、列、值中若指定多个，则同类间需以空格分离。
- 对于倾斜值，不支持在hint中进行类型强转；对于string类型，需要使用单引号。

例如：

- 指定单表倾斜

  每一个skew hint用来表示一个表关系存在的倾斜信息，如果想要指定在查询中的多个表关系存在的倾斜信息，则通过指定多个skew hint实现。

  在指定skew时，包括以下四个场景的用法：

  – 单列单值： skew(t (c1) (v1))

    说明：表关系t的c1列中的v1值在查询执行中存在倾斜。

- 单列多值：skew(t (c1) (v1 v2 v3 ...))

  说明：表关系t的c1列中的v1、v2、v3…等值在查询执行中存在倾斜。
- 多列单值：skew(t (c1 c2) (v1 v2))

  说明：表关系t的c1列的v1值和c2列的v2值在查询执行中存在倾斜。
- 多列多值：skew(t (c1 c2) ((v1 v2) (v3 v4) (v5 v6) ...))

  说明：表关系t的c1列的v1、v3、v5…值和c2列的v2、v4、v6…值在查询执行中存在倾斜。

须知

多列多值时，各组倾斜值间也可以不使用括号，如：skew(t (c1 c2) (v1 v2 v3 v4 v5 v6 ...))。是否使用括号必须统一，不可混合，

如：skew(t (c1 c2) (v1 v2 v3 v4 （ v5 v6 ） ...)) 将会产生语法报错。

- 指定中间结果倾斜

  如果基表不存在倾斜，而是查询执行中的中间结果出现倾斜，则需要通过指定中间结果倾斜的skew hint来进行倾斜的调优。skew((t1 t2) (c1) (v1))

  说明：表关系t1和t2 Join后的结果存在倾斜，倾斜的是t1表的c1列，c1列的倾斜值是v1。

  为了避免产生歧义，“c1”只能存在于join_rel的一个表关系中，如果存在同名列则通过别名进行规避。

建议

- 如果查询具有多层，则哪一层出现倾斜，则将hint写在哪一层中。
- 对于提升的子查询，skew hint支持直接使用子查询名进行hint。如果明确子查询提升后的哪一个基表存在倾斜，则直接使用基表进行hint的可用性更高。
- 无论对于表或列，若存在别名，则优先使用别名进行hint。

示例

指定单表倾斜

- 原query中进行hint。

  采用如下查询进行skew hint倾斜调优的举例，查询语句及不带hint的原计划如下所示：

```
explain
with customer_total_return as
(select sr_customer_sk as ctr_customer_sk
,sr_store_sk as ctr_store_sk
,sum(SR_FEE) as ctr_total_return
from store_returns
,date_dim
where sr_returned_date_sk = d_date_sk
and d_year =2000
group by sr_customer_sk
,sr_store_sk)
 select  c_customer_id
from customer_total_return ctr1
,store
,customer
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
```

```
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = 'NM'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id
limit 100;
```

```
 id |                                operation                                 |  E-rows   |    E-memory    | E-width |  E-costs
----+--------------------------------------------------------------------------+-----------+----------------+---------+-----------
  1 | ->  Row Adapter                                                          |       100 |                |      20 | 911254.47
  2 |    ->  Vector Limit                                                      |       100 |                |      20 | 911254.47
  3 |       ->  Vector Streaming (type: GATHER)                                |      2400 |                |      20 | 911325.75
  4 |          ->  Vector Limit                                                |      2400 | 1MB            |      20 | 911247.62
  5 |             ->  Vector Sort                                              |   3684816 | 16MB           |      20 | 911631.21
  6 |                ->  Vector Hash Join (7,29)                               |   3684817 | 41MB(12374MB)  |      20 | 905379.41
  7 |                   ->  Vector Streaming(type: REDISTRIBUTE)               |   3684817 | 384KB          |       4 | 883010.31
  8 |                      ->  Vector Hash Join (9,19)                         |   3684817 | 16MB           |       4 | 861302.05
  9 |                         ->  Vector Hash Join (10,18)                     |  11054450 | 16MB           |      44 | 427109.71
 10 |                            ->  Vector Hash Aggregate                     |  50247501 | 397MB(12671MB) |      54 | 395302.57
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)   |  50247501 | 384KB          |      22 | 358663.76
 12 |                                  ->  Vector Hash Join (13,15)            |  50247501 | 16MB           |      22 | 294300.51
 13 |                                     ->  Vector Partition Iterator        | 287999764 | 1MB            |      26 | 227383.99
 14 |                                        ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |      26 | 227383.99
 15 |                                     ->  Vector Streaming(type: BROADCAST)|      8712 | 384KB          |       4 | 975.56
 16 |                                        ->  Vector Partition Iterator     |       363 | 1MB            |       4 | 910.65
 17 |                                           ->  Partitioned CStore Scan on date_dim |  363 | 1MB |       4 | 910.65
 18 |                            ->  CStore Scan on store                      |        44 | 1MB            |       4 | 1006.39
 19 |                         ->  Vector Hash Aggregate                        |       192 | 16MB           |      68 | 426707.38
 20 |                            ->  Vector Subquery Scan on ctr2              |  50247501 | 1MB            |      36 | 416239.03
 21 |                               ->  Vector Hash Aggregate                  |  50247501 | 397MB(12671MB) |      54 | 395302.57
 22 |                                  ->  Vector Streaming(type: REDISTRIBUTE)|  50247501 | 384KB          |      22 | 358663.76
 23 |                                     ->  Vector Hash Join (24,26)         |  50247501 | 16MB           |      22 | 294300.51
 24 |                                        ->  Vector Partition Iterator     | 287999764 | 1MB            |      26 | 227383.99
 25 |                                           ->  Partitioned CStore Scan on store_returns | 287999764 | 1MB |  26 | 227383.99
 26 |                                        ->  Vector Streaming(type: BROADCAST)|      8712 | 384KB          |       4 | 975.56
 27 |                                           ->  Vector Partition Iterator  |       363 | 1MB            |       4 | 910.65
 28 |                                              ->  Partitioned CStore Scan on date_dim |  363 | 1MB |       4 | 910.65
 29 |                   ->  CStore Scan on customer                            |  12000000 | 1MB            |      24 | 12923.00
(29 rows)
```

对内层with子句中的HashAgg和外层的Hash Join进行hint指定，带hint的查询如下：

```
explain
with customer_total_return as
(select /*+ skew(store_returns(sr_store_sk sr_customer_sk)) */sr_customer_sk as ctr_customer_sk
,sr_store_sk as ctr_store_sk
,sum(SR_FEE) as ctr_total_return
from store_returns
,date_dim
where sr_returned_date_sk = d_date_sk
and d_year =2000
group by sr_customer_sk
,sr_store_sk)
 select  /*+ skew(ctr1(ctr_customer_sk)(11))*/  c_customer_id
from customer_total_return ctr1
,store
,customer
where ctr1.ctr_total_return > (select avg(ctr_total_return)*1.2
from customer_total_return ctr2
where ctr1.ctr_store_sk = ctr2.ctr_store_sk)
and s_store_sk = ctr1.ctr_store_sk
and s_state = 'NM'
and ctr1.ctr_customer_sk = c_customer_sk
order by c_customer_id
limit 100;
```

该hint表示：内层with子句中的group by在做HashAgg中进行重分布时存在倾斜，对应原计划的10和21号Hash Agg算子；外层ctr1表的ctr_customer_sk列在做Hash Join中进行重分布时存在倾斜，对应原计划的6号算子。生成计划如下所示：

```
id |                              operation                                  | E-rows   |   E-memory      | E-width | E-costs
----+-------------------------------------------------------------------------+----------+-----------------+---------+-----------
  1 | ->  Row Adapter                                                          |      100 |                 |      20 | 1061778.14
  2 |   ->  Vector Limit                                                       |      100 |                 |      20 | 1061778.14
  3 |     ->  Vector Streaming (type: GATHER)                                  |     2400 |                 |      20 | 1061849.41
  4 |       ->  Vector Limit                                                   |     2400 | 1MB             |      20 | 1061771.29
  5 |         ->  Vector Sort                                                  |  3684816 | 16MB            |      20 | 1062154.87
  6 |           ->  Vector Hash Join (7,31)                                    |  3684817 | 41MB(12344MB)   |      20 | 1055903.08
  7 |             ->  Vector Streaming(type: PART REDISTRIBUTE PART ROUNDROBIN)|  3684817 | 384KB           |       4 | 1013056.49
  8 |               ->  Vector Hash Join (9,20)                                |  3684817 | 16MB            |       4 | 1000006.10
  9 |                 ->  Vector Hash Join (10,19)                             | 11054450 | 16MB            |      44 | 496461.73
 10 |                   ->  Vector Hash Aggregate                              | 50247501 | 397MB(12010MB)  |      54 | 464654.59
 11 |                     ->  Vector Streaming(type: REDISTRIBUTE)             | 50247501 | 384KB           |      54 | 428015.79
 12 |                       ->  Vector Hash Aggregate                          | 50247501 | 397MB(12010MB)  |      54 | 330939.31
 13 |                         ->  Vector Hash Join (14,16)                     | 50247501 | 16MB            |      22 | 294300.51
 14 |                           ->  Vector Partition Iterator                  |287999764 | 1MB             |      26 | 227383.99
 15 |                             ->  Partitioned CStore Scan on store_returns |287999764 | 1MB             |      26 | 227383.99
 16 |                           ->  Vector Streaming(type: BROADCAST)          |     8712 | 384KB           |       4 | 975.56
 17 |                             ->  Vector Partition Iterator                |      363 | 1MB             |       4 | 910.65
 18 |                               ->  Partitioned CStore Scan on date_dim    |      363 | 1MB             |       4 | 910.65
 19 |                   ->  CStore Scan on store                               |       44 | 1MB             |       4 | 1006.39
 20 |                 ->  Vector Hash Aggregate                                |      192 | 16MB            |      68 | 496059.40
 21 |                   ->  Vector Subquery Scan on ctr2                       | 50247501 | 1MB             |      36 | 485591.05
 22 |                     ->  Vector Hash Aggregate                            | 50247501 | 397MB(12010MB)  |      54 | 464654.59
 23 |                       ->  Vector Streaming(type: REDISTRIBUTE)           | 50247501 | 384KB           |      54 | 428015.79
 24 |                         ->  Vector Hash Aggregate                        | 50247501 | 397MB(12010MB)  |      54 | 330939.31
 25 |                           ->  Vector Hash Join (26,28)                   | 50247501 | 16MB            |      22 | 294300.51
 26 |                             ->  Vector Partition Iterator                |287999764 | 1MB             |      26 | 227383.99
 27 |                               ->  Partitioned CStore Scan on store_returns |287999764 | 1MB           |      26 | 227383.99
 28 |                             ->  Vector Streaming(type: BROADCAST)        |     8712 | 384KB           |       4 | 975.56
 29 |                               ->  Vector Partition Iterator              |      363 | 1MB             |       4 | 910.65
 30 |                                 ->  Partitioned CStore Scan on date_dim  |      363 | 1MB             |       4 | 910.65
 31 |             ->  Vector Streaming(type: PART LOCAL PART BROADCAST)        | 12000000 | 384KB           |      24 | 34485.50
 32 |               ->  CStore Scan on customer                                | 12000000 | 1MB             |      24 | 12923.00
(32 rows)
```

从优化后的计划可以看出：①对于Hash Agg，由于其重分布存在倾斜，所以优化为双层Agg；②对于Hash Join，同样由于其重分布存在倾斜，所以优化为采用新的重分布算子。

- 需要改写query后进行hint

  不带hint的查询和计划如下：

```
explain select count(*) from store_sales_1 group by round(ss_list_price);
```

```
----+-------------------------------------------------------------+---------+----------+---------+----------
  1 | ->  Row Adapter                                             |   16672 |          |      14 | 62261.28
  2 |   ->  Vector Streaming (type: GATHER)                       |   16672 |          |      14 | 62261.28
  3 |     ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)       |   16672 | 32KB     |      14 | 61479.78
  4 |       ->  Vector Hash Aggregate                             |   16672 | 16MB     |      14 | 61452.00
  5 |         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) | 3112836 | 128KB |       6 | 57498.43
  6 |           ->  CStore Scan on store_sales_1                  | 3112836 | 1MB      |       6 | 21810.25
(6 rows)
```

由于hint中列不支持表达式，在进行倾斜优化时需要借助subquery改写查询，改写后的查询和计划如下：

```
explain
select count(*)
from (select round(ss_list_price),ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

```
----+-------------------------------------------------------------+---------+----------+---------+----------
  1 | ->  Row Adapter                                             |   16672 |          |      14 | 62261.28
  2 |   ->  Vector Streaming (type: GATHER)                       |   16672 |          |      14 | 62261.28
  3 |     ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)       |   16672 | 32KB     |      14 | 61479.78
  4 |       ->  Vector Hash Aggregate                             |   16672 | 16MB     |      14 | 61452.00
  5 |         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2) | 3112836 | 128KB |       6 | 57498.43
  6 |           ->  CStore Scan on store_sales_1                  | 3112836 | 1MB      |       6 | 21810.25
(6 rows)
```

改写注意不要影响到业务逻辑。

采用改写后的查询进行hint，带hint的查询和计划如下：

```
explain
select /*+ skew(tmp(a)) */ count(*)
from (select round(ss_list_price),ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

```
 id |                       operation                               | E-rows  | E-memory | E-width | E-costs
----+---------------------------------------------------------------+---------+----------+---------+----------
  1 | ->  Row Adapter                                               |   16672 |          |      14 | 27771.82
  2 |   ->  Vector Streaming (type: GATHER)                         |   16672 |          |      14 | 27771.82
  3 |     ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)         |   16672 | 32KB     |      14 | 26990.32
  4 |       ->  Vector Hash Aggregate                               |   16671 | 16MB     |      14 | 26962.54
  5 |         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2)|   66216 | 128KB    |      14 | 26838.09
  6 |           ->  Vector Hash Aggregate                           |   66216 | 16MB     |      14 | 25949.61
  7 |             ->  CStore Scan on store_sales_1                  | 3112836 | 1MB      |       6 | 21810.25
(7 rows)
```

从计划可以看出，对Hash Agg进行倾斜优化后，采用了双层agg实现，大大过滤了进行重分布时的数据量，减少了重分布时间。

此外，需要说明的是，对于子查询，支持使用查询内部的列进行hint，如：

```
explain
select /*+ skew(tmp(b)) */ count(*)
```

```
from (select round(ss_list_price) b,ss_hdemo_sk
from store_sales_1)tmp(a,ss_hdemo_sk)
group by a;
```

14.3.6.9 Hint 的错误、冲突及告警

Plan Hint的结果会体现在计划的变化上，可以通过explain来查看变化。

Hint中的错误不会影响语句的执行，只是不能生效，该错误会根据语句类型以不同方式提示用户。对于explain语句，hint的错误会以warning形式显示在界面上，对于非explain语句，会以debug1级别日志显示在日志中，关键字为PLANHINT。

hint 的错误类型

- 语法错误

  语法规则树归约失败，会报错，指出出错的位置。

  例如：hint关键字错误，leading hint或join hint指定2个表以下，其它hint未指定表等。一旦发现语法错误，则立即终止hint的解析，所以此时只有错误前面的解析完的hint有效。

  例如：

  ```
  leading((t1 t2)) nestloop(t1) rows(t1 t2 #10)
  ```

  nestloop(t1)存在语法错误，则终止解析，可用hint只有之前解析的leading((t1 t2))。

- 语义错误
  - 表不存在，存在多个，或在leading或join中出现多次，均会报语义错误。
  - scanhint中的index不存在，会报语义错误。
  - 另外，如果子查询提升后，同一层出现多个名称相同的表，且其中某个表需要被hint，hint会存在歧义，无法使用，需要为相同表增加别名规避。
- hint重复或冲突

  如果存在hint重复或冲突，只有第一个hint生效，其它hint均会失效，会给出提示。
  - hint重复是指，hint的方法及表名均相同。例如：nestloop(t1 t2) nestloop(t1 t2)。
  - hint冲突是指，table list一样的hint，存在不一样的hint，hint的冲突仅对于每一类hint方法检测冲突。

    例如：nestloop (t1 t2) hashjoin (t1 t2)，则后面与前面冲突，此时hashjoin的hint失效。注意：nestloop(t1 t2)和no mergejoin(t1 t2)不冲突。

    须知

    leading hint中的多个表会进行拆解。例如：leading ((t1 t2 t3))会拆解成：leading((t1 t2)) leading(((t1 t2) t3))，此时如果存在leading((t2 t1))，则两者冲突，后面的会被丢弃。（例外：指定内外表的hint若与不指定内外表的hint重复，则始终丢弃不指定内外表的hint。）

- 子链接提升后hint失效

  子链接提升后的hint失效，会给出提示。通常出现在子链接中存在多个表连接的场景。提升后，子链接中的多个表不再作为一个整体出现在join中。

- 列类型不支持重分布
  - 对于skew hint来说，目的是为了进行重分布时的调优，所以当hint列的类型不支持重分布时，hint将无效。
- hint未被使用
  - 非等值join使用hashjoin hint或mergejoin hint
  - 不包含索引的表使用indexscan hint或indexonlyscan hint
  - 通常只有在索引列上使用过滤条件才会生成相应的索引路径，全表扫描将不会使用索引，因此使用indexscan hint或indexonlyscan hint将不会使用
  - indexonlyscan只有输出列仅包含索引列才会使用，否则指定时hint不会被使用
  - 多个表存在等值连接时，仅尝试有等值连接条件的表的连接，此时没有关联条件的表之间的路径将不会生成，所以指定相应的leading，join，rows hint将不使用，例如：t1 t2 t3表join，t1和t2, t2和t3有等值连接条件，则t1和t3不会优先连接，leading(t1 t3)不会被使用。
  - 生成stream计划时，如果表的分布列与join列相同，则不会生成redistribute的计划；如果不同，且另一表分布列与join列相同，只能生成redistribute的计划，不会生成broadcast的计划，指定相应的hint则不会被使用。
  - 如果子链接未被提升，则blockname hint不会被使用。
  - 对于skew hint，hint未被使用可能由于：
    - 计划中不需要进行重分布。
    - hint指定的列为包含分布键。
    - hint指定倾斜信息有误或不完整，如对于join优化未指定值。
    - 倾斜优化的GUC参数处于关闭状态。

14.3.6.10 Plan Hint 实际调优案例

本节以TPC-DS标准测试的Q24的部分语句为例，在1000X，24DN环境上，说明使用plan hint进行实际调优的过程。示例如下：

```
select avg(netpaid) from
(select c_last_name
,c_first_name
,s_store_name
,ca_state
,s_state
,i_color
,i_current_price
,i_manager_id
,i_units
,i_size
,sum(ss_sales_price) netpaid
from store_sales
,store_returns
,store
,item
,customer
,customer_address
where ss_ticket_number = sr_ticket_number
and ss_item_sk = sr_item_sk
and ss_customer_sk = c_customer_sk
and ss_item_sk = i_item_sk
and ss_store_sk = s_store_sk
and c_birth_country = upper(ca_country)
```

```
and s_zip = ca_zip
and s_market_id=7
group by c_last_name
,c_first_name
,s_store_name
,ca_state
,s_state
,i_color
,i_current_price
,i_manager_id
,i_units
,i_size);
```

1. 该语句的初始计划如下，运行时间110s：

```
 id |                                                                operation                                                                |        A-time         |  A-rows   |  E-rows
----+-----------------------------------------------------------------------------------------------------------------------------------------+-----------------------+-----------+-----------+
  1 | ->  Row Adapter                                                                                                                         | 110324.107            |         1 |         1 |
  2 |    ->  Vector Aggregate                                                                                                                 | 110324.093            |         1 |         1 |
  3 |       ->  Vector Streaming (type: GATHER)                                                                                               | 110323.958            |        24 |        24 |
  4 |          ->  Vector Aggregate                                                                                                           | [110179.302,110309.653] |        24 |        24 |
  5 |             ->  Vector Hash Aggregate                                                                                                   | [110178.388,110308.515] |    647824 |     16656 |
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                                                                                 | [77616.177,96478.771] |    666834733 |     16664 |
  7 |                   ->  Vector Hash Join (8,22)                                                                                           | [81727.257,84728.519] |    666834733 |     16664 |
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)                                                                           | [78770.520,82021.087] |    666834733 |     16664 |
  9 |                         ->  Vector Hash Join (10,21)                                                                                    | [88066.755,90701.860] |    666834733 |     16664 |
 10 |                            ->  Vector Streaming(type: BROADCAST)                                                                        | [7940.962,21430.725]  |    591882336 |     51360 |
 11 |                               ->  Vector Hash Join (12,20)                                                                              | [2419.995,5319.606]   |     24661764 |      2140 |
 12 |                                  ->  Vector Streaming(type: REDISTRIBUTE)                                                               | [1750.448,4659.581]   |     25258268 |      2241 |
 13 |                                     ->  Vector Hash Join (14,18)                                                                        | [15240.666,17159.616] |     25258268 |      2241 |
 14 |                                        ->  Vector Hash Join (15,17)                                                                     | [12112.913,13563.366] |    252564412 |    472070592 |
 15 |                                           ->  Vector Partition Iterator                                                                 | [11148.731,12473.230] |   2879987999 |   2879987999 |
 16 |                                              ->  Partitioned CStore Scan on public.store_sales                                          | [11097.921,12412.596] |   2879987999 |   2879987999 |
 17 |                                           ->  CStore Scan on public.store                                                               | [0.447,0.689]         |         2064 |         2064 |
 18 |                                        ->  Vector Partition Iterator                                                                    | [296.805,319.014]     |    287999764 |    287999764 |
 19 |                                           ->  Partitioned CStore Scan on public.store_returns                                           | [292.938,314.787]     |    287999764 |    287999764 |
 20 |                                     ->  CStore Scan on public.customer                                                                  | [114.358,144.462]     |     12000000 |     12000000 |
 21 |                                  ->  CStore Scan on public.customer_address                                                             | [38.426,56.753]       |      6000000 |      6000000 |
 22 |                               ->  CStore Scan on public.item                                                                            | [3.160,5.026]         |       300000 |       300000 |
(22 rows)
```

该计划中，第10层算子使用broadcast性能较差，由于第11层算子估算行数为2140，比实际行数严重低估。错误行数估算主要来源于第13层算子的行数低估，根因是第13层hashjoin中，使用store_sales的(ss_ticket_number, ss_item_sk)列和store_returns的(sr_ticket_number, sr_item_sk)列进行关联，由于缺少多列相关性的估算导致行数严重低估。

2. 使用如下的rows hint进行调优后，计划如下，运行时间318s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns * 11270)*/ c_last_name ...
```

```
 id |                                                                operation                                                                |        A-time         |  A-rows   |  E-rows
----+-----------------------------------------------------------------------------------------------------------------------------------------+-----------------------+-----------+-----------+
  1 | ->  Row Adapter                                                                                                                         | 318585.246            |         1 |         1 |
  2 |    ->  Vector Aggregate                                                                                                                 | 318585.232            |         1 |         1 |
  3 |       ->  Vector Streaming (type: GATHER)                                                                                               | 318585.082            |        24 |        24 |
  4 |          ->  Vector Aggregate                                                                                                           | [318323.324,318499.290] |        24 |        24 |
  5 |             ->  Vector Hash Aggregate                                                                                                   | [318320.813,318497.054] |    647824 |  187770504 |
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                                                                                 | [288074.860,305601.698] |    666834733 |  187770507 |
  7 |                   ->  Vector Hash Join (8,22)                                                                                           | [253642.468,315808.664] |    666834733 |  187770507 |
  8 |                      ->  Vector Hash Join (9,18)                                                                                        | [250904.317,315684.018] |    666834733 |  187770507 |
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)                                                                        | [4552.500,310602.307] |    275042158 |  147106999 |
 10 |                            ->  Vector Hash Join (11,17)                                                                                 | [7658.951,14053.823]  |    275042158 |  147106999 |
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)                                                                  | [3953.255,10264.943]  |    287999764 |  154060900 |
 12 |                                  ->  Vector Hash Join (13,15)                                                                           | [28196.188,32838.794] |    287999764 |  154060900 |
 13 |                                     ->  Vector Partition Iterator                                                                       | [11477.673,12324.583] |   2879987999 |   2879987999 |
 14 |                                        ->  Partitioned CStore Scan on public.store_sales                                                | [11411.382,12250.209] |   2879987999 |   2879987999 |
 15 |                                     ->  Vector Partition Iterator                                                                       | [304.188,403.205]     |    287999764 |    287999764 |
 16 |                                        ->  Partitioned CStore Scan on public.store_returns                                              | [299.838,398.255]     |    287999764 |    287999764 |
 17 |                                  ->  CStore Scan on public.customer                                                                     | [122.246,170.128]     |     12000000 |     12000000 |
 18 |                               ->  Vector Streaming(type: REDISTRIBUTE)                                                                  | [57.558,117.461]      |       492915 |       146467 |
 19 |                                  ->  Vector Hash Join (20,21)                                                                           | [45.554,96.238]       |       492915 |       146467 |
 20 |                                     ->  CStore Scan on public.customer_address                                                          | [39.738,89.412]       |      6000000 |      6000000 |
 21 |                                     ->  CStore Scan on public.store                                                                     | [0.361,1.095]         |         2064 |         2064 |
 22 |                            ->  Vector Streaming(type: BROADCAST)                                                                        | [48.986,91.170]       |      7200000 |      7200000 |
 23 |                               ->  CStore Scan on public.item                                                                            | [4.506,6.602]         |       300000 |       300000 |
(23 rows)
```

时间反而劣化了，原因是第8层hashjoin过慢引起第9层redistribute时间过慢导致，其中第9层redistribute并没有数据倾斜，hashjoin慢的原因是由于第18层redistribute后数据倾斜导致。

3. 经过实际数据查证，customer_address的两个join列的不同值数目较少，使用其进行join容易出现数据倾斜，故把customer_address放到最后进行join。使用如下的hint进行调优后，计划如下，运行时间116s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((store_sales store_returns store item customer) customer_address)*/
c_last_name ...
```

```
 id |                                operation                                |        A-time         |  A-rows   |  E-rows   |
----+-------------------------------------------------------------------------+-----------------------+-----------+-----------+
  1 | ->  Row Adapter                                                         | 116326.597            |         1 |         1 |
  2 |    ->  Vector Aggregate                                                 | 116326.590            |         1 |         1 |
  3 |       ->  Vector Streaming (type: GATHER)                               | 116326.473            |        24 |        24 |
  4 |          ->  Vector Aggregate                                           | [116157.161,116236.494] |        24 |        24 |
  5 |             ->  Vector Hash Aggregate                                   | [116155.328,116233.946] |    647824 | 187770504 |
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                 | [84103.951,102052.326] |  666834733 | 187770507 |
  7 |                   ->  Vector Hash Join (8,10)                           | [23229.469,47484.697] |  666834733 | 187770507 |
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)           | [38.367,74.930]       |   6000000 |   6000000 |
  9 |                         ->  CStore Scan on public.customer_address      | [69.877,121.460]      |   6000000 |   6000000 |
 10 |                      ->  Vector Streaming(type: REDISTRIBUTE)           | [17404.744,17567.550] |  24661764 |  24112909 |
 11 |                         ->  Vector Hash Join (12,22)                    | [16123.627,16397.246] |  24661764 |  24112909 |
 12 |                            ->  Vector Streaming(type: REDISTRIBUTE)     | [15320.663,15741.646] |  25258268 |  25252751 |
 13 |                               ->  Vector Hash Join (14,21)              | [14962.342,16375.458] |  25258268 |  25252751 |
 14 |                                  ->  Vector Hash Join (15,19)           | [14449.031,15825.949] |  25258268 |  25252751 |
 15 |                                     ->  Vector Hash Join (16,18)        | [11439.959,12510.065] |  252564412 |  472070592 |
 16 |                                        ->  Vector Partition Iterator    | [10531.986,11536.213] | 2879987999 | 2879987999 |
 17 |                                           ->  Partitioned CStore Scan on public.store_sales | [10483.634,11474.944] | 2879987999 | 2879987999 |
 18 |                                        ->  CStore Scan on public.store  | [0.347,0.463]         |      2064 |      2064 |
 19 |                                     ->  Vector Partition Iterator       | [293.977,365.021]     |  287999764 |  287999764 |
 20 |                                        ->  Partitioned CStore Scan on public.store_returns | [289.936,360.808] |  287999764 |  287999764 |
 21 |                                  ->  CStore Scan on public.item         | [3.109,5.245]         |    300000 |    300000 |
 22 |                               ->  CStore Scan on public.customer        | [113.871,141.791]     |  12000000 |  12000000 |
(22 rows)
```

发现时间基本花在了第6层redistribute算子上，需要进一步优化。

4. 由于最后一层redistribute包含倾斜，所以时间较长。为了避免倾斜，需要将item表放在最后join，由于item表的join并不能使行数减少。修改hint如下并执行，计划如下，运行时间120s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((customer_address (store_sales store_returns store customer) item))
c_last_name ...
```

```
 id |                                operation                                |        A-time         |  A-rows   |  E-rows   |
----+-------------------------------------------------------------------------+-----------------------+-----------+-----------+
  1 | ->  Row Adapter                                                         | 120377.258            |         1 |         1 |
  2 |    ->  Vector Aggregate                                                 | 120377.245            |         1 |         1 |
  3 |       ->  Vector Streaming (type: GATHER)                               | 120377.091            |        24 |        24 |
  4 |          ->  Vector Aggregate                                           | [120184.884,120301.704] |        24 |        24 |
  5 |             ->  Vector Hash Aggregate                                   | [120183.119,120297.845] |    647824 | 187770504 |
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                 | [87775.682,106070.878] |  666834733 | 187770507 |
  7 |                   ->  Vector Hash Join (8,22)                           | [22323.764,49878.523] |  666834733 | 187770507 |
  8 |                      ->  Vector Hash Join (9,11)                        | [21129.236,45208.255] |  666834733 | 187770507 |
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)        | [37.859,75.412]       |   6000000 |   6000000 |
 10 |                            ->  CStore Scan on public.customer_address   | [74.798,114.449]      |   6000000 |   6000000 |
 11 |                         ->  Vector Streaming(type: REDISTRIBUTE)        | [15714.458,15824.928] |  24661764 |  24112909 |
 12 |                            ->  Vector Hash Join (13,21)                 | [14637.516,14955.464] |  24661764 |  24112909 |
 13 |                               ->  Vector Streaming(type: REDISTRIBUTE)  | [13898.593,14333.200] |  25258268 |  25252751 |
 14 |                                  ->  Vector Hash Join (15,19)           | [14166.917,15378.244] |  25258268 |  25252751 |
 15 |                                     ->  Vector Hash Join (16,18)        | [11272.239,12052.532] |  252564412 |  472070592 |
 16 |                                        ->  Vector Partition Iterator    | [10409.566,11127.981] | 2879987999 | 2879987999 |
 17 |                                           ->  Partitioned CStore Scan on public.store_sales | [10365.838,11077.601] | 2879987999 | 2879987999 |
 18 |                                        ->  CStore Scan on public.store  | [0.431,0.609]         |      2064 |      2064 |
 19 |                                     ->  Vector Partition Iterator       | [343.780,408.254]     |  287999764 |  287999764 |
 20 |                                        ->  Partitioned CStore Scan on public.store_returns | [339.844,403.923] |  287999764 |  287999764 |
 21 |                               ->  CStore Scan on public.customer        | [117.234,163.598]     |  12000000 |  12000000 |
 22 |                            ->  Vector Streaming(type: BROADCAST)        | [44.571,130.129]      |   7200000 |   7200000 |
 23 |                               ->  CStore Scan on public.item            | [4.169,6.347]         |    300000 |    300000 |
(23 rows)
```

该计划中的redistribute问题并没有解决，因为第22层item表做了broadcast，导致与customer_address表join后的倾斜并没有被消除掉。

5. 增加如下禁止item表做broadcast的hint，使与customer_address join的表做redistribute（也可以进行join表redistribute的hint），计划如下，运行时间105s：

```
select avg(netpaid) from
(select /*+rows(store_sales store_returns *11270)
leading((customer_address (store_sales store_returns store customer) item))
no broadcast(item)*/
c_last_name ...
```

```
 id |                                                operation                                                |          A-time          | A-rows  |  E-rows
----+---------------------------------------------------------------------------------------------------------+--------------------------+---------+-----------
  1 | ->  Row Adapter                                                                                         | 105854.957               |       1 |         1
  2 |    ->  Vector Aggregate                                                                                 | 105854.948               |       1 |         1
  3 |       ->  Vector Streaming (type: GATHER)                                                               | 105854.825               |      24 |        24
  4 |          ->  Vector Aggregate                                                                           | [105706.709,105776.135]  |      24 |        24
  5 |             ->  Vector Hash Aggregate                                                                   | [105705.061,105773.013]  |  647824 | 187770504
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                                                 | [70701.966,89973.672]    |  666834733 | 187770507
  7 |                   ->  Vector Hash Join (8,23)                                                           | [71759.500,79018.433]    |  666834733 | 187770507
  8 |                      ->  Vector Streaming(type: REDISTRIBUTE)                                           | [69794.307,77269.178]    |  666834733 | 187770507
  9 |                         ->  Vector Hash Join (10,12)                                                    | [21443.307,46714.378]    |  666834733 | 187770507
 10 |                            ->  Vector Streaming(type: REDISTRIBUTE)                                     | [41.295,83.419]          |    6000000 |   6000000
 11 |                               ->  CStore Scan on public.customer_address                                | [70.405,166.072]         |    6000000 |   6000000
 12 |                            ->  Vector Streaming(type: REDISTRIBUTE)                                     | [15689.053,15788.475]    |   24661764 |  24112909
 13 |                               ->  Vector Hash Join (14,22)                                              | [14517.847,14712.929]    |   24661764 |  24112909
 14 |                                  ->  Vector Streaming(type: REDISTRIBUTE)                               | [13806.733,14089.770]    |   25258268 |  25252751
 15 |                                     ->  Vector Hash Join (16,20)                                        | [13709.384,15095.449]    |   25258268 |  25252751
 16 |                                        ->  Vector Hash Join (17,19)                                     | [10944.796,11827.285]    |  252564412 | 472070592
 17 |                                           ->  Vector Partition Iterator                                 | [10070.316,10884.728]    | 2879987999 | 2879987999
 18 |                                              ->  Partitioned CStore Scan on public.store_sales          | [10018.966,10828.990]    | 2879987999 | 2879987999
 19 |                                           ->  CStore Scan on public.store                               | [0.447,0.568]            |       2064 |        2064
 20 |                                        ->  Vector Partition Iterator                                    | [293.042,329.056]        |  287999764 | 287999764
 21 |                                           ->  Partitioned CStore Scan on public.store_returns           | [288.631,324.782]        |  287999764 | 287999764
 22 |                                  ->  CStore Scan on public.customer                                     | [113.735,138.235]        |   12000000 |  12000000
 23 |                      ->  CStore Scan on public.item                                                     | [3.127,5.357]            |     300000 |    300000
(23 rows)
```

6. 发现最后一层使用单层Agg，但行数缩减较多。使用相同的hint，同时结合参数best_agg_plan=3进行双层Agg调优，最终计划如下图所示，运行时间94s，完成调优。

```
 id |                                                operation                                                |          A-time          | A-rows  |  E-rows
----+---------------------------------------------------------------------------------------------------------+--------------------------+---------+-----------
  1 | ->  Row Adapter                                                                                         | 94004.670                |       1 |         1
  2 |    ->  Vector Aggregate                                                                                 | 94004.655                |       1 |         1
  3 |       ->  Vector Streaming (type: GATHER)                                                               | 94004.504                |      24 |        24
  4 |          ->  Vector Aggregate                                                                           | [93833.832,93928.052]    |      24 |        24
  5 |             ->  Vector Hash Aggregate                                                                   | [93832.460,93926.412]    |  647824 | 187770507
  6 |                ->  Vector Streaming(type: REDISTRIBUTE)                                                 | [93640.866,93787.939]    |  647824 | 183912384
  7 |                   ->  Vector Hash Aggregate                                                             | [93687.544,93791.242]    |  647824 | 183912384
  8 |                      ->  Vector Hash Join (9,24)                                                        | [70025.469,72773.161]    |  666834733 | 187770507
  9 |                         ->  Vector Streaming(type: REDISTRIBUTE)                                        | [68242.223,71275.972]    |  666834733 | 187770507
 10 |                            ->  Vector Hash Join (11,13)                                                 | [21421.136,44830.306]    |  666834733 | 187770507
 11 |                               ->  Vector Streaming(type: REDISTRIBUTE)                                  | [35.444,71.328]          |    6000000 |   6000000
 12 |                                  ->  CStore Scan on public.customer_address                             | [67.246,119.224]         |    6000000 |   6000000
 13 |                               ->  Vector Streaming(type: REDISTRIBUTE)                                  | [16089.853,16212.570]    |   24661764 |  24112909
 14 |                                  ->  Vector Hash Join (15,23)                                           | [14822.972,15188.942]    |   24661764 |  24112909
 15 |                                     ->  Vector Streaming(type: REDISTRIBUTE)                            | [14061.867,14604.162]    |   25258268 |  25252751
 16 |                                        ->  Vector Hash Join (17,21)                                     | [13949.756,15492.311]    |   25258268 |  25252751
 17 |                                           ->  Vector Hash Join (18,20)                                  | [10935.742,12160.719]    |  252564412 | 472070592
 18 |                                              ->  Vector Partition Iterator                              | [10052.958,11194.962]    | 2879987999 | 2879987999
 19 |                                                 ->  Partitioned CStore Scan on public.store_sales       | [10008.415,11143.984]    | 2879987999 | 2879987999
 20 |                                              ->  CStore Scan on public.store                            | [0.452,0.839]            |       2064 |        2064
 21 |                                           ->  Vector Partition Iterator                                 | [298.235,332.736]        |  287999764 | 287999764
 22 |                                              ->  Partitioned CStore Scan on public.store_returns        | [294.067,327.629]        |  287999764 | 287999764
 23 |                                        ->  CStore Scan on public.customer                               | [114.377,145.156]        |   12000000 |  12000000
 24 |                      ->  CStore Scan on public.item                                                     | [3.150,3.530]            |     300000 |    300000
(24 rows)
```

如果有统计信息变更引起的查询劣化，可以考虑用plan hint来调整到之前的查询计划。这里以TPCH-Q17为例，在收集default_statistics_target设置为–2的统计信息之后，计划相比于默认统计信息发生劣化。

1. 默认统计信息（default_statistics_target设置为100）的计划如下：

```
 id |                                operation                                |          A-time
----+-------------------------------------------------------------------------+--------------------------
  1 | ->  Row Adapter                                                         | 265006.779
  2 |    ->  Vector Aggregate                                                 | 265006.764
  3 |       ->  Vector Streaming (type: GATHER)                               | 265006.071
  4 |          ->  Vector Aggregate                                           | [263699.512,264503.084]
  5 |             ->  Vector Hash Join (6,17)                                 | [263676.665,264477.932]
  6 |                ->  Vector Streaming(type: LOCAL GATHER dop: 1/4)        | [1.998,7.594]
  7 |                   ->  Vector Hash Aggregate                             | [201775.393,202432.672]
  8 |                      ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 4/4) | [201567.130,202231.524]
  9 |                         ->  Vector Hash Join (10,12)                    | [170675.231,199908.410]
 10 |                            ->  Vector Partition Iterator                | [34847.797,51968.266]
 11 |                               ->  Partitioned CStore Scan on tpch10wx_col.lineitem | [33805.013,51137.657]
 12 |                            ->  Vector Hash Aggregate                    | [23283.387,25359.493]
 13 |                               ->  Vector Streaming(type: SPLIT BROADCAST dop: 4/4) | [12850.624,14608.515]
 14 |                                  ->  Vector Hash Aggregate             | [2690.439,3616.623]
 15 |                                     ->  Vector Partition Iterator      | [2659.700,3579.390]
 16 |                                        ->  Partitioned CStore Scan on tpch10wx_col.part | [2642.213,3559.093]
 17 |                ->  Vector Streaming(type: REDISTRIBUTE dop: 1/4)        | [262300.732,262961.078]
 18 |                   ->  Vector Hash Join (19,21)                          | [225749.727,260990.322]
 19 |                      ->  Vector Partition Iterator                      | [40046.078,56220.694]
 20 |                         ->  Partitioned CStore Scan on tpch10wx_col.lineitem | [39204.414,55328.448]
 21 |                      ->  Vector Streaming(type: SPLIT BROADCAST dop: 4/4) | [55748.177,61987.136]
 22 |                         ->  Vector Partition Iterator                   | [3042.864,3873.942]
 23 |                            ->  Partitioned CStore Scan on tpch10wx_col.part | [3027.023,3848.159]
(23 rows)
```

2. 统计信息变更（default_statistics_target设置为–2）的计划如下：

```
id |                                                operation                                                |          A-time
---+---------------------------------------------------------------------------------------------------------+---------------------------
 1 | ->  Row Adapter                                                                                         | 1440492.994
 2 |    ->  Vector Aggregate                                                                                 | 1440492.982
 3 |       ->  Vector Streaming  (type: GATHER)                                                              | 1440491.021
 4 |          ->  Vector Streaming(type: LOCAL GATHER dop: 1/6)                                              | [1439737.284,1440008.568]
 5 |             ->  Vector Aggregate                                                                        | [1439008.369,1439854.148]
 6 |                ->  Vector Hash Join (7,18)                                                              | [1439006.016,1439851.619]
 7 |                   ->  Vector Streaming(type: LOCAL BROADCAST dop: 6/6)                                  | [2.932,139.405]
 8 |                      ->  Vector Hash Aggregate                                                          | [190452.312,195910.748]
 9 |                         ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 6/6)                         | [190171.929,195653.119]
10 |                            ->  Vector Hash Join (11,13)                                                 | [161076.195,178831.123]
11 |                               ->  Vector Partition Iterator                                             | [27306.318,45564.565]
12 |                                  ->  Partitioned CStore Scan on tpch10wx_col.lineitem                   | [26752.444,44912.020]
13 |                               ->  Vector Hash Aggregate                                                 | [35601.624,39812.058]
14 |                                  ->  Vector Streaming(type: SPLIT BROADCAST dop: 6/6)                   | [23096.460,27057.137]
15 |                                     ->  Vector Hash Aggregate                                           | [2372.587,3052.445]
16 |                                        ->  Vector Partition Iterator                                    | [2345.381,3012.732]
17 |                                           ->  Partitioned CStore Scan on tpch10wx_col.part              | [2329.874,2989.393]
18 |                   ->  Vector Hash Join (19,22)                                                          | [1437388.414,1438470.781]
19 |                      ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 6/6)                            | [1392693.529,1408571.859]
20 |                         ->  Vector Partition Iterator                                                   | [29065.204,41264.514]
21 |                            ->  Partitioned CStore Scan on tpch10wx_col.lineitem                         | [28212.219,40133.491]
22 |                      ->  Vector Streaming(type: LOCAL REDISTRIBUTE dop: 6/6)                            | [2570.841,3438.567]
23 |                         ->  Vector Partition Iterator                                                   | [2447.569,3276.369]
24 |                            ->  Partitioned CStore Scan on tpch10wx_col.part                             | [2432.124,3263.641]
(24 rows)
```

3. 经过对比，劣化的原因主要为lineitem和part表join时stream类型由BroadCast变更为Redistribute导致。可以对语句进行stream方式的hint来调整到之前的计划，例如：

```
select /*+ no redistribute(part lineitem) */
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#23'
    and p_container = 'MED BOX'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
```

14.3.7 例行维护表

为了保证数据库的有效运行，数据库必须在插入/删除操作后，基于客户场景，定期做vacuum full和analyze，更新统计信息，以便获得更优的性能。

相关概念

使用VACUUM、VACUUM FULL和ANALYZE命令定期对每个表进行维护，主要有以下原因：

- VACUUM FULL可回收已更新或已删除的数据所占据的磁盘空间，同时将小数据文件合并。
- ANALYZE可收集与数据库中表内容相关的统计信息。统计结果存储在系统表PG_STATISTIC中。查询优化器会使用这些统计数据，生成最有效的执行计划。

操作步骤

步骤1 使用VACUUM或VACUUM FULL命令，进行磁盘空间回收。

- **VACUUM FULL**：
  ```
  VACUUM FULL customer;
  VACUUM
  ```
  需要向正在执行的表增加排他锁，且需要停止其他所有数据库操作。

步骤2 使用ANALYZE语句更新统计信息。

```
ANALYZE customer;
ANALYZE
```

使用ANALYZE VERBOSE语句更新统计信息，并输出表的相关信息。

```
ANALYZE VERBOSE customer;
ANALYZE
```

也可以同时执行VACUUM ANALYZE命令进行查询优化。

```
VACUUM ANALYZE customer;
VACUUM
```

说明

VACUUM和ANALYZE会导致I/O流量的大幅增加，这可能会影响其他活动会话的性能。

----结束

维护建议

- 定期对部分大表做VACUUM FULL，在性能下降后为全库做VACUUM FULL，目前暂定每月做一次VACUUM FULL。
- 定期对系统表做VACUUM FULL，主要是PG_ATTRIBUTE。
- 启用系统自动清理进程（AUTOVACUUM）自动执行VACUUM和ANALYZE命令，回收被标识为删除状态的记录空间，并更新表的统计数据。

14.3.8 例行重建索引

背景信息

数据库经过多次删除操作后，索引页面上的索引键将被删除，造成索引膨胀。例行重建索引，可有效的提高查询效率。

数据库支持的索引类型包含B-tree索引、GIN索引和PSORT索引。

- 对于B-tree索引，例行重建索引可有效的提高查询效率。
  - 如果数据发生大量删除后，索引页面上的索引键将被删除，导致索引页面数量的减少，造成索引膨胀。重建索引可回收浪费的空间。
  - 新建的索引中逻辑结构相邻的页面，通常在物理结构中也是相邻的，所以一个新建的索引比更新了多次的索引访问速度要快。
- 对于非B-tree索引，不建议例行重建。

重建索引

重建索引有以下两种方式：

- 先删除索引（DROP INDEX），再创建索引（CREATE INDEX）。

  在删除索引过程中，会在父表上增加一个短暂的排他锁，阻止相关读写操作。在创建索引过程中，会锁住写操作但是不会锁住读操作，此时读操作只能使用顺序扫描。

- 使用REINDEX语句重建索引。
  - 使用REINDEX TABLE语句重建索引，会在重建过程中增加排他锁，阻止相关读写操作。
  - 使用REINDEX INTERNAL TABLE语句重建desc表（包括）的索引，会在重建过程中增加排他锁，阻止相关读写操作。

操作步骤

假定在导入表“areaS”上的“area_id”字段上存在普通索引“areaS_idx”。重建索引有以下两种方式：

- 先删除索引（DROP INDEX），再创建索引（CREATE INDEX）
  a. 删除索引。
  ```
  DROP INDEX areaS_idx;
  DROP INDEX
  ```
  b. 创建索引。
  ```
  CREATE INDEX areaS_idx ON areaS (area_id);
  CREATE INDEX
  ```
- 使用REINDEX重建索引。
  - 使用REINDEX TABLE语句重建索引。
  ```
  REINDEX TABLE areaS;
  REINDEX
  ```
  - 使用REINDEX INTERNAL TABLE重建desc表（包括）的索引。
  ```
  REINDEX INTERNAL TABLE areaS;
  REINDEX
  ```

14.3.9 配置 SMP

14.3.9.1 SMP 适用场景与限制

SMP特性通过算子并行来提升性能，同时会占用更多的系统资源，包括CPU、内存、网络、I/O等等。本质上SMP是一种以资源换取时间的方式，在合适的场景以及资源充足的情况下，能够起到较好的性能提升效果；但是如果在不合适的场景下，或者资源不足的情况下，反而可能引起性能的劣化。1.3.x新增了SMP自适应特性，该特性会根据当前资源和查询特征，动态选取最优的并行度。下面对各种资源对SMP性能的影响情况分别进行说明：

适用场景

- 支持并行的算子

  计划中存在以下算子支持并行：

  a. Scan：支持行存普通表和行存分区表顺序扫描、列存普通表和列存分区表顺序扫描、OBS外表（ORC格式）顺序扫描；支持GDS数据导入的外表扫描并行。以上均不支持复制表。

  b. Join：HashJoin、NestLoop

  c. Agg：HashAgg、SortAgg、PlainAgg、WindowAgg(只支持partition by，不支持order by)。

d. Stream：Redistribute、Broadcast

e. 其他：Result、Subqueryscan、Unique、Material、Setop、Append、VectoRow、RowToVec

- SMP特有算子

  为了实现并行，新增了并行线程间的数据交换Stream算子供SMP特性使用。这些新增的算子可以看做Stream算子的子类。

  a. Local Gather：实现DN内部并行线程的数据汇总

  b. Local Redistribute：在DN内部各线程之间，按照分布键进行数据重分布

  c. Local Broadcast：将数据广播到DN内部的每个线程

  d. Local RoundRobin：在DN内部各线程之间实现数据轮询分发

  e. Split Redistribute：在集群跨DN的并行线程之间实现数据重分布

  f. Split Broadcast：将数据广播到集群所有DN的并行线程

  上述新增算子可以分为Local与非Local两类，Local类算子实现了DN内部并行线程间的数据交换，而非Local类算子实现了跨DN的并行线程间的数据交换。

- 示例说明

  以TPCH Q1的并行计划为例：

  图 14-8 TPCH Q1 并行执行计划

```
 id |                                operation                                 |
----+--------------------------------------------------------------------------+
  1 | ->  Row Adapter                                                          |
  2 |    ->  Vector Streaming (type: GATHER)                                   |
  3 |       ->  Vector Sort                                                    |
  4 |          ->  Vector Streaming(type: LOCAL GATHER dop: 1/2)               |
  5 |             ->  Vector Hash Aggregate                                    |
  6 |                ->  Vector Streaming(type: SPLIT REDISTRIBUTE dop: 2/2)   |
  7 |                   ->  Vector Hash Aggregate                              |
  8 |                      ->  Vector Partition Iterator                       |
  9 |                         ->  Partitioned CStore Scan on public.lineitem   |
(9 rows)
```

  在这个计划中，实现了Partitioned CStore Scan以及HashAgg算子的并行，并且新增了Local Gather和Split Redistribute数据交换算子。

  其中6号算子为Split Redistribute算子，上面标有的“dop: 2/2”表明Split Redistribute的发送端和接收端线程的并行度均为2。4号算子为Local Gather，上面标有“dop: 1/2”，该算子的发送端线程并行度为2，而接收端线程并行度为1，即下层的5号Hash Aggregate算子按照2并行度执行，而上层的1~3号算子按照串行执行，4号算子实现了DN内并行线程的数据汇总。

  通过计划Stream算子上标明的dop信息即可看出各个算子的并行情况。

非适用场景

1. 不支持CN上的算子并行。
2. 不支持不能下推的查询并行执行。
3. 不支持子查询subplan和initplan的并行，以及包含子查询的算子并行。

14.3.9.2 资源对 SMP 性能的影响

SMP架构是一种利用富余资源来换取时间的方案，计划并行之后必定会引起资源消耗的增加，包括CPU、内存、I/O和网络带宽等资源的消耗都会出现明显的增长，而且随着并行度的增大，资源消耗也随之增大。当上述资源成为瓶颈的情况下，SMP无法提

升性能，反而可能导致集群整体性能的劣化。下面对各种资源对SMP性能的影响情况分别进行说明：

- **CPU资源**

  在一般客户场景中，系统CPU利用率不高的情况下，利用SMP并行架构能够更充分地利用系统CPU资源，提升系统性能。但当数据库服务器的CPU核数较少，CPU利用率已经比较高的情况下，如果打开SMP并行，不仅性能提升不明显，反而可能因为多线程间的资源竞争而导致性能劣化。

- **内存资源**

  查询并行后会导致内存使用量的增长，但每个算子使用内存上限仍受到资源管理模块的限制；在系统内存不足时，并行后可能出现数据下盘，导致查询性能劣化的问题。

- **网络带宽资源**

  为了实现查询并行执行，会新增并行线程间的数据交换算子。对于Local类Stream算子，所需要进行数据交换的线程在同一个DN内，通过内存交换，不会增加网络负担。而非Local类算子，需要通过网络进行数据交换，因此会加重网络负担。当网络资源成为瓶颈的情况下，并行可能会导致一定程度的劣化。

- **I/O资源**

  要实现并行扫描必定会增加I/O的资源消耗，因此只有在I/O资源充足的情况下，并行扫描才能够提高扫描性能。

14.3.9.3 其他因素对 SMP 性能的影响

除了资源因素外，还有一些因素也会对SMP并行性能造成影响。例如分区表中分区数据不均，以及系统并发度等因素。

- **数据倾斜对SMP性能的影响**

  当数据中存在严重数据倾斜时，并行效果较差。例如某表join列上某个值的数据量远远大于其他值，开启并行后，根据join列的值对该表数据做hash重分布，使得某个并行线程的数据量远多于其他线程，造成长尾问题，导致并行后效果差。

- **系统并发度对SMP性能的影响**

  SMP特性会增加资源的使用，而在高并发场景下资源剩余较少。所以，如果在高并发场景下，开启SMP并行，会导致各查询之间严重的资源竞争问题。一旦出现了资源竞争的现象，无论是CPU、I/O、内存或者网络资源，都会导致整体性能的下降。因此在高并发场景下，开启SMP往往不能达到性能提升的效果，甚至可能引起性能劣化。

14.3.9.4 SMP 使用建议

根据上面的分析，总结如下：

使用限制

想要利用SMP提升查询性能需要满足以下条件：

系统的CPU、内存、I/O和网络带宽等资源充足。SMP架构是一种利用富余资源来换取时间的方案，计划并行之后必定会引起资源消耗的增加，当上述资源成为瓶颈的情况下，SMP无法提升性能，反而可能导致性能的劣化。在出现资源瓶颈的情况下，建议关闭SMP。

配置步骤

1. 设置query_dop=1（默认值），利用explain打出执行计划，观察计划是否符合SMP适用场景与限制中的限制条件。如果符合，进入下一步。
2. 设置query_dop=0（自适应），系统会根据资源情况和计划特征，动态为每个查询选取[1,8]之间的最优的并行度，最大化提升查询性能。
3. 设置query_dop=-value，在考虑资源情况和计划特征基础上，限制dop选取的范围为[1,value]。
4. 设置query_dop=value，不考虑资源情况和计划特征，强制选取dop为1或value。

14.4 实际调优案例

14.4.1 案例：选择合适的分布列

现象描述

表定义如下：

```
CREATE TABLE t1 (a int, b int);
CREATE TABLE t2 (a int, b int);
```

执行如下查询：

```
SELECT * FROM t1, t2 WHERE t1.a = t2.b;
```

优化分析

如果将a作为t1和t2的分布列：

```
CREATE TABLE t1 (a int, b int) DISTRIBUTE BY HASH (a);
CREATE TABLE t2 (a int, b int) DISTRIBUTE BY HASH (a);
```

则执行计划将存在“Streaming”，导致DN之间存在较大通信数据量，如图14-9所示。

图 14-9 选择合适的分布列案例（一）

```
postgres=> explain select * from t1, t2 where t1.a = t2.b;
                                  QUERY PLAN
------------------------------------------------------------------------------
 Streaming (type: GATHER)  (cost=245.40..582.15 rows=240 width=16)
   Node/s: All datanodes
   ->  Hash Join  (cost=10.22..24.26 rows=10 width=16)
         Hash Cond: (t1.a = t2.b)
         ->  Seq Scan on t1  (cost=0.00..10.10 rows=10 width=8)
         ->  Hash  (cost=3.79..3.79 rows=10 width=8)
               ->  Streaming(type: REDISTRIBUTE)  (cost=0.00..3.79 rows=10 width=8)
                     Spawn on: All datanodes
                     ->  Seq Scan on t2  (cost=0.00..10.10 rows=10 width=8)
(9 rows)
```

如果将a作为t1的分布列，将b作为t2的分布列：

```
CREATE TABLE t1 (a int, b int) DISTRIBUTE BY HASH (a);
CREATE TABLE t2 (a int, b int) DISTRIBUTE BY HASH (b);
```

则执行计划将不包含“Streaming”，减少DN之间存在的通信数据量，从而提升查询性能，如图14-10所示。

图 14-10 选择合适的分布列案例（二）

```
postgres=> explain select * from t1, t2 where t1.a = t2.b;
                                  QUERY PLAN
------------------------------------------------------------------------------
 Streaming (type: GATHER)  (cost=245.40..491.10 rows=240 width=16)
   Node/s: All datanodes
   ->  Hash Join  (cost=10.22..20.46 rows=10 width=16)
         Hash Cond: (t1.a = t2.b)
         ->  Seq Scan on t1  (cost=0.00..10.10 rows=10 width=8)
         ->  Hash  (cost=10.10..10.10 rows=10 width=8)
               ->  Seq Scan on t2  (cost=0.00..10.10 rows=10 width=8)
(7 rows)
```

14.4.2 案例：建立合适的索引

现象描述

查询与销售部所有员工的信息：

```
SELECT staff_id,first_name,last_name,employment_id,state_name,city
FROM staffs,sections,states,places
WHERE sections.section_name='Sales'
AND staffs.section_id = sections.section_id
AND sections.place_id = places.place_id
AND places.state_id = states.state_id
ORDER BY staff_id;
```

优化分析

在优化前，没有创建places.place_id和states.state_id索引，执行计划如下：

```
 id |                          operation                          |     A-time      | A-rows | E-rows | Peak Memory  | E-memory | A-width | E-width | E-costs
----+-------------------------------------------------------------+-----------------+--------+--------+--------------+----------+---------+---------+---------
  1 | ->  Streaming (type: GATHER)                                | 69.801          |     34 |      2 |212KB         |          |         |     354 | 53.67
  2 |    ->  Sort                                                 | [21.508,23.283] |     34 |      2 |[30KB, 36KB]  | 16MB     | [380,380] |     354 | 53.32
  3 |       ->  Hash Join (4,5)                                   | [21.483,23.153] |     34 |      2 |[7KB, 7KB]    | 1MB      |         |     354 | 53.31
  4 |          ->  Seq Scan on hr.states                          | [0.007,0.022]   |     17 |     20 |[12KB, 12KB]  | 1MB      |         |     110 | 13.13
  5 |          ->  Hash                                           | [21.026,22.663] |     34 |      2 |[262KB, 294KB]| 16MB     | [284,284] |     268 | 40.08
  6 |             ->  Streaming(type: REDISTRIBUTE)               | [21.024,22.458] |     34 |      2 |[85KB, 86KB]  | 1MB      |         |     268 | 40.08
  7 |                ->  Hash Join (8,9)                          | [13.814,14.827] |     34 |      2 |[8KB, 8KB]    | 1MB      |         |     268 | 39.80
  8 |                   ->  Seq Scan on hr.staffs                 | [0.035,0.043]   |    107 |     20 |[19KB, 19KB]  | 1MB      |         |     190 | 13.13
  9 |                   ->  Hash                                  | [13.361,14.348] |      2 |      4 |[292KB, 292KB]| 16MB     | [124,124] |     102 | 26.57
 10 |                      ->  Streaming(type: BROADCAST)         | [13.291,14.279] |      2 |      4 |[85KB, 85KB]  | 1MB      |         |     102 | 26.57
 11 |                         ->  Hash Join (12,13)               | [6.359,7.446]   |      1 |      2 |[6KB, 6KB]    | 1MB      |         |     102 | 26.48
 12 |                            ->  Seq Scan on hr.places        | [0.008,0.018]   |     15 |     20 |[14KB, 14KB]  | 1MB      |         |     102 | 13.13
 13 |                            ->  Hash                         | [5.999,7.077]   |      1 |      1 |[259KB, 291KB]| 16MB     | [32,32] |      24 | 13.28
 14 |                               ->  Streaming(type: REDISTRIBUTE) | [5.999,6.958]   |      1 |      1 |[84KB, 85KB]  | 1MB      |         |      24 | 13.28
 15 |                                  ->  Seq Scan on hr.sections | [0.021,0.022]   |      1 |      1 |[14KB, 14KB]  | 1MB      |         |      24 | 13.16
(15 rows)

         Predicate Information (identified by plan id)
-------------------------------------------------------------
  3 --Hash Join (4,5)
        Hash Cond: (states.state_id = places.state_id)
  7 --Hash Join (8,9)
        Hash Cond: (staffs.section_id = sections.section_id)
 11 --Hash Join (12,13)
        Hash Cond: (places.place_id = sections.place_id)
 15 --Seq Scan on hr.sections
        Filter: ((sections.section_name)::text = 'Sales'::text)
        Rows Removed by Filter: 26
(9 rows)
```

建议在places.place_id和states.state_id列上建立2个索引，执行计划如下：

```
 id |                                operation                                |     A-time      | A-rows | E-rows | Peak Memory | E-memory | A-width | E-width | E-costs
----+-------------------------------------------------------------------------+-----------------+--------+--------+-------------+----------+---------+---------+---------
  1 | ->  Streaming (type: GATHER)                                            | 50.411          |     34 |      2 | 212KB       |          |         |     354 |   46.84
  2 |    ->  Sort                                                             | [20.889,22.621] |     34 |      2 | [30KB, 36KB] |  16MB    | [380,380] |     354 |   46.49
  3 |       ->  Nested Loop (4,13)                                            | [20.870,22.490] |     34 |      2 | [5KB, 5KB]  |  1MB     |         |     354 |   46.48
  4 |          ->  Streaming(type: REDISTRIBUTE)                              | [20.869,21.852] |     34 |      2 | [85KB, 86KB] |  1MB     |         |     268 |   35.46
  5 |             ->  Nested Loop (6,7)                                       | [13.433,14.452] |     34 |      2 | [6KB, 6KB]  |  1MB     |         |     268 |   35.18
  6 |                ->  Seq Scan on hr.staffs                                | [0.027,0.032]   |    107 |     20 | [19KB, 19KB] |  1MB     |         |     190 |   13.13
  7 |                ->  Materialize                                          | [13.237,14.230] |    109 |      4 | [10KB, 10KB] |  16MB    | [124,124] |     102 |   21.66
  8 |                   ->  Streaming(type: BROADCAST)                        | [13.150,14.137] |      2 |      4 | [85KB, 85KB] |  1MB     |         |     102 |   21.65
  9 |                      ->  Nested Loop (10,12)                            | [5.784,6.822]   |      1 |      2 | [3KB, 3KB]  |  1MB     |         |     102 |   21.56
 10 |                         ->  Streaming(type: REDISTRIBUTE)               | [5.782,6.675]   |      1 |      1 | [84KB, 85KB] |  1MB     |         |      24 |   13.28
 11 |                            ->  Seq Scan on hr.sections                  | [0.019,0.020]   |      1 |      1 | [14KB, 14KB] |  1MB     |         |      24 |   13.16
 12 |                         ->  Index Scan using loc_id_pk on hr.places     | [0.091,0.091]   |      1 |      2 | [24KB, 24KB] |  1MB     |         |     102 |    8.27
 13 |          ->  Index Scan using state_c_id_pk on hr.states                | [0.352,0.352]   |     34 |      2 | [23KB, 23KB] |  1MB     |         |     110 |    5.50
(13 rows)

          Predicate Information (identified by plan id)
-----------------------------------------------------------------
   5 --Nested Loop (6,7)
         Join Filter: (sections.section_id = staffs.section_id)
         Rows Removed by Join Filter: 73
  11 --Seq Scan on hr.sections
         Filter: ((sections.section_name)::text = 'Sales'::text)
         Rows Removed by Filter: 26
  12 --Index Scan using loc_id_pk on hr.places
         Index Cond: (places.place_id = sections.place_id)
  13 --Index Scan using state_c_id_pk on hr.states
         Index Cond: (states.state_id = places.state_id)
(10 rows)
```

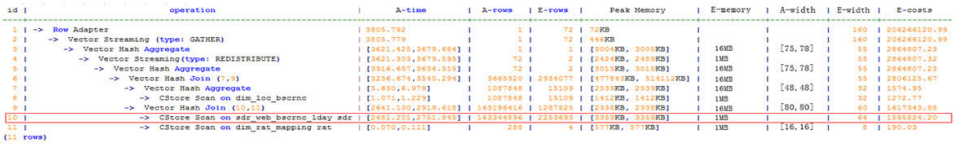
14.4.3 案例：增加 JOIN 列非空条件

现象描述

```
SELECT
 *
FROM
( ( SELECT
  STARTTIME STTIME,
  SUM(NVL(PAGE_DELAY_MSEL,0)) PAGE_DELAY_MSEL,
  SUM(NVL(PAGE_SUCCEED_TIMES,0)) PAGE_SUCCEED_TIMES,
  SUM(NVL(FST_PAGE_REQ_NUM,0)) FST_PAGE_REQ_NUM,
  SUM(NVL(PAGE_AVG_SIZE,0)) PAGE_AVG_SIZE,
  SUM(NVL(FST_PAGE_ACK_NUM,0)) FST_PAGE_ACK_NUM,
  SUM(NVL(DATATRANS_DW_DURATION,0)) DATATRANS_DW_DURATION,
  SUM(NVL(PAGE_SR_DELAY_MSEL,0)) PAGE_SR_DELAY_MSEL
 FROM
  PS.SDR_WEB_BSCRNC_1DAY SDR
  INNER JOIN (SELECT
      BSCRNC_ID,
      BSCRNC_NAME,
      ACCESS_TYPE,
      ACCESS_TYPE_ID
     FROM
      nethouse.DIM_LOC_BSCRNC
     GROUP BY
      BSCRNC_ID,
      BSCRNC_NAME,
      ACCESS_TYPE,
      ACCESS_TYPE_ID) DIM
  ON SDR.BSCRNC_ID = DIM.BSCRNC_ID
  AND DIM.ACCESS_TYPE_ID IN (0,1,2)
  INNER JOIN nethouse.DIM_RAT_MAPPING RAT
  ON (RAT.RAT = SDR.RAT)
 WHERE
  ( (STARTTIME >= 1461340800
  AND STARTTIME < 1461427200) )
  AND RAT.ACCESS_TYPE_ID IN (0,1,2)
  --and SDR.BSCRNC_ID is not null
 GROUP BY
  STTIME ) ) ;
```

执行计划如图14-11所示。

图 14-11 增加 JOIN 列非空条件（一）



优化分析

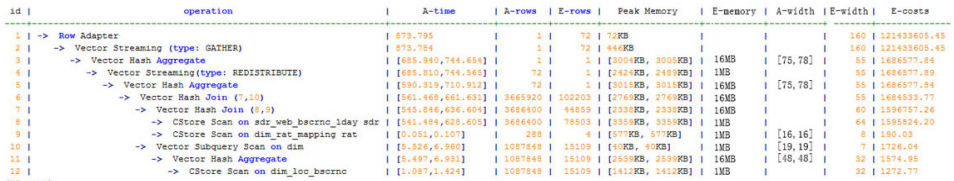
1. 分析执行计划图14-11可知，在顺序扫描阶段耗时较多。
2. 多表JOIN中，由于表PS.SDR_WEB_BSCRNC_1DAY的JOIN列“BSCRNC_ID”存在大量空值，JOIN性能差。

   建议在语句中手动添加JOIN列的非空判断，修改后的语句如下所示。

```
SELECT
 *
FROM
( ( SELECT
  STARTTIME STTIME,
  SUM(NVL(PAGE_DELAY_MSEL,0)) PAGE_DELAY_MSEL,
  SUM(NVL(PAGE_SUCCEED_TIMES,0)) PAGE_SUCCEED_TIMES,
  SUM(NVL(FST_PAGE_REQ_NUM,0)) FST_PAGE_REQ_NUM,
  SUM(NVL(PAGE_AVG_SIZE,0)) PAGE_AVG_SIZE,
  SUM(NVL(FST_PAGE_ACK_NUM,0)) FST_PAGE_ACK_NUM,
  SUM(NVL(DATATRANS_DW_DURATION,0)) DATATRANS_DW_DURATION,
  SUM(NVL(PAGE_SR_DELAY_MSEL,0)) PAGE_SR_DELAY_MSEL
 FROM
  PS.SDR_WEB_BSCRNC_1DAY SDR
  INNER JOIN (SELECT
      BSCRNC_ID,
      BSCRNC_NAME,
      ACCESS_TYPE,
      ACCESS_TYPE_ID
     FROM
      nethouse.DIM_LOC_BSCRNC
     GROUP BY
      BSCRNC_ID,
      BSCRNC_NAME,
      ACCESS_TYPE,
      ACCESS_TYPE_ID) DIM
  ON SDR.BSCRNC_ID = DIM.BSCRNC_ID
  AND DIM.ACCESS_TYPE_ID IN (0,1,2)
  INNER JOIN nethouse.DIM_RAT_MAPPING RAT
  ON (RAT.RAT = SDR.RAT)
 WHERE
  ( (STARTTIME >= 1461340800
  AND STARTTIME < 1461427200) )
  AND RAT.ACCESS_TYPE_ID IN (0,1,2)
  and SDR.BSCRNC_ID is not null
 GROUP BY
  STTIME ) ) A;
```

执行计划如图14-12所示。

图 14-12 增加 JOIN 列非空条件（二）



14.4.4 案例：使排序下推

现象描述

在做场景性能测试时，发现某场景大部分时间是CN端在做window agg，占到总执行时间95%以上，系统资源不能充分利用。研究发现该场景的特点是：将两列分别求sum作为一个子查询，外层对两列的和再求和后做trunc，然后排序。

表结构如下所示：

```
CREATE TABLE public.test(imsi int,L4_DW_THROUGHPUT int,L4_UL_THROUGHPUT int)
with (orientation = column) DISTRIBUTE BY hash(imsi);
```

查询语句如下所示：

```
SELECT COUNT(1) over() AS DATACNT,
IMSI AS IMSI_IMSI,
CAST(TRUNC(((SUM(L4_UL_THROUGHPUT) + SUM(L4_DW_THROUGHPUT))), 0) AS
DECIMAL(20)) AS TOTAL_VOLOME_KPIID
FROM public.test AS test
GROUP BY IMSI
order by TOTAL_VOLOME_KPIID DESC;
```

执行计划如下：

```
Row Adapter  (cost=10.70..10.70 rows=10 width=12)
  ->  Vector Sort  (cost=10.68..10.70 rows=10 width=12)
        Sort Key: ((trunc((((sum(l4_ul_throughput)) + (sum(l4_dw_throughput))))::numeric,
0))::numeric(20,0))
        ->  Vector WindowAgg  (cost=10.09..10.51 rows=10 width=12)
              ->  Vector Streaming (type: GATHER)  (cost=242.04..246.84 rows=240 width=12)
                    Node/s: All datanodes
                    ->  Vector Hash Aggregate  (cost=10.09..10.29 rows=10 width=12)
                          Group By Key: imsi
                          ->  CStore Scan on test  (cost=0.00..10.01 rows=10 width=12)
```

可以看到window agg和sort全部在CN端执行，耗时非常严重。

优化分析

尝试将语句改写为子查询。

```
SELECT COUNT(1) over() AS DATACNT, IMSI_IMSI, TOTAL_VOLOME_KPIID
FROM (SELECT IMSI AS IMSI_IMSI,
CAST(TRUNC(((SUM(L4_UL_THROUGHPUT) + SUM(L4_DW_THROUGHPUT))),
0) AS DECIMAL(20)) AS TOTAL_VOLOME_KPIID
FROM public.test AS test
GROUP BY IMSI
ORDER BY TOTAL_VOLOME_KPIID DESC);
```

将trunc两列的和作为一个子查询，然后在子查询的外面做window agg，这样排序就可以下推了，执行计划如下：

```
Row Adapter  (cost=10.70..10.70 rows=10 width=24)
  ->  Vector WindowAgg  (cost=10.45..10.70 rows=10 width=24)
        ->  Vector Streaming (type: GATHER)  (cost=250.83..253.83 rows=240 width=24)
              Node/s: All datanodes
              ->  Vector Sort  (cost=10.45..10.48 rows=10 width=12)
                    Sort Key: ((trunc(((sum(test.l4_ul_throughput) + sum(test.l4_dw_throughput)))::numeric,
0))::numeric(20,0))
                    ->  Vector Hash Aggregate  (cost=10.09..10.29 rows=10 width=12)
                          Group By Key: test.imsi
                          ->  CStore Scan on test  (cost=0.00..10.01 rows=10 width=12)
```

经过SQL改写，性能由120s提升7s，优化效果明显。

14.4.5 案例：设置 cost_param 对查询性能优化

现象描述 1

cost_param的bit0(set cost_param=1)值为1时，表示对于求!=连接的选择率时选择一种改良机制，此方法在自连接（两个相同的表之间连接）的估算中更加准确。下面查询的例子是cost_param的bit0为1时的优化场景。V300R002C00版本开始已弃用cost_param & 1不为0时的路径，默认选择已优化的估算公式。

注：选择率是两表join时，满足join条件的行数在join结果集中所占的比率。

表结构如下所示：

```
CREATE TABLE LINEITEM
(
L_ORDERKEY BIGINT NOT NULL
, L_PARTKEY BIGINT NOT NULL
, L_SUPPKEY BIGINT NOT NULL
, L_LINENUMBER BIGINT NOT NULL
, L_QUANTITY DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL
, L_DISCOUNT DECIMAL(15,2) NOT NULL
, L_TAX DECIMAL(15,2) NOT NULL
, L_RETURNFLAG CHAR(1) NOT NULL
, L_LINESTATUS CHAR(1) NOT NULL
, L_SHIPDATE DATE NOT NULL
, L_COMMITDATE DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE CHAR(10) NOT NULL
, L_COMMENT VARCHAR(44) NOT NULL
) with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(L_ORDERKEY);

CREATE TABLE ORDERS
(
O_ORDERKEY BIGINT NOT NULL
, O_CUSTKEY BIGINT NOT NULL
, O_ORDERSTATUS CHAR(1) NOT NULL
, O_TOTALPRICE DECIMAL(15,2) NOT NULL
, O_ORDERDATE DATE NOT NULL
, O_ORDERPRIORITY CHAR(15) NOT NULL
, O_CLERK CHAR(15) NOT NULL
, O_SHIPPRIORITY BIGINT NOT NULL
, O_COMMENT VARCHAR(79) NOT NULL
)with (orientation = column, COMPRESSION = MIDDLE) distribute by hash(O_ORDERKEY);
```

查询语句如下所示：

```
explain verbose select
count(*) as numwait
from
lineitem l1,
orders
where
o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and not exists (
select
*
from
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
```

```
order by
numwait desc;
```

执行计划如下图所示：（verbose条件下，新增distinct列，受cost off/on控制，hashjoin行显示内外表的distinct估值，其他行为空）

```
 id |                          operation                          | E-rows | E-distinct | E-width | E-costs
----+-------------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Row Adapter                                             |      1 |            |       8 | 39.36
  2 |    ->  Vector Sort                                          |      1 |            |       8 | 39.36
  3 |       ->  Vector Aggregate                                  |      1 |            |       8 | 39.34
  4 |          ->  Vector Streaming (type: GATHER)                |      2 |            |       8 | 39.34
  5 |             ->  Vector Aggregate                            |      2 |            |       8 | 39.25
  6 |                ->  Vector Hash Anti Join (7, 10)            |      2 | 4, 5       |       0 | 39.24
  7 |                   ->  Vector Hash Join (8,9)                |      2 | 200, 1     |      16 | 26.12
  8 |                      ->  CStore Scan on public.lineitem l1  |      7 |            |      16 | 13.05
  9 |                      ->  CStore Scan on public.orders       |      1 |            |       8 | 13.05
 10 |                   ->  CStore Scan on public.lineitem l3     |      7 |            |      16 | 13.05
(10 rows)
```

优化分析 1

以上查询为lineitem表自连接的Anti Join，当使用cost_param的bit0为0时，估算Anti Join的行数与实际行数相差很大，导致查询性能下降。可以通过设置cost_param的bit0为1时，使Anti Join的行数估算更准确，从而提高查询性能。优化后的执行计划如下：

```
id |                          operation                          |  E-rows   | E-memory | E-width |  E-costs
---+-------------------------------------------------------------+-----------+----------+---------+-----------
 1 |  -> Row Adapter                                             |         1 |          |       0 | 9104892.37 9
 2 |    -> Vector Sort                                           |         1 |          |       0 | 9104892.37 9
 3 |      -> Vector Aggregate                                    |         1 |          |       0 | 9104892.35 8
 4 |        -> Vector Streaming (type: GATHER)                   |        48 |          |       0 | 9104892.35 8
 5 |          -> Vector Aggregate                                |        48 | 1MB      |       0 | 9104890.82 5
 6 |            -> Vector Hash Join (7.12)                       | 2526630903 | 929MB    |       0 | 8973295.45 4
 7 |              -> Vector Hash Anti Join (8. 10)               | 1999996587 | 3178MB   |       8 | 7198231.14
 8 |                -> Vector Partition Iterator                 | 1999996587 | 1MB      |      16 | 3000158.25
 9 |                  -> Partitioned CStore Scan on public.lineitem 11 | 1999996587 | 1MB | 16 | 3000158.25 1
10 |                -> Vector Partition Iterator                 | 1999996587 | 1MB      |      16 | 3000158.25
11 |                  -> Partitioned CStore Scan on public.lineitem 13 | 1999996587 | 1MB | 16 | 3000158.25
12 |              -> Vector Partition Iterator                   |  730839014 | 1MB      |       8 | 589611.00
13 |                -> Partitioned CStore Scan on public.orders  |  730839014 | 1MB      |       8 | 589611.00
(13 rows)
```

现象描述 2

当cost_param的bit1(set cost_param=2)为1时，表示求多个过滤条件（Filter）的选择率时，选择最小的作为总的选择率，而非两者乘积，此方法在过滤条件的列之间关联性较强时估算更加准确。下面查询的例子是cost_param的bit1为1时的优化场景。

表结构如下所示：

```
CREATE TABLE NATION
(
N_NATIONKEYINT NOT NULL
, N_NAMECHAR(25) NOT NULL
, N_REGIONKEYINT NOT NULL
, N_COMMENTVARCHAR(152)
) distribute by replication;
CREATE TABLE SUPPLIER
(
S_SUPPKEYBIGINT NOT NULL
, S_NAMECHAR(25) NOT NULL
, S_ADDRESSVARCHAR(40) NOT NULL
, S_NATIONKEYINT NOT NULL
```

```
, S_PHONECHAR(15) NOT NULL
, S_ACCTBALDECIMAL(15,2) NOT NULL
, S_COMMENTVARCHAR(101) NOT NULL
) distribute by hash(S_SUPPKEY);
CREATE TABLE PARTSUPP
(
PS_PARTKEYBIGINT NOT NULL
, PS_SUPPKEYBIGINT NOT NULL
, PS_AVAILQTYBIGINT NOT NULL
, PS_SUPPLYCOSTDECIMAL(15,2)NOT NULL
, PS_COMMENTVARCHAR(199) NOT NULL
)distribute by hash(PS_PARTKEY);
```

查询语句如下所示：

```
set cost_param=2;
explain verbose select
nation,
sum(amount) as sum_profit
from
(
select
n_name as nation,
l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
from
supplier,
lineitem,
partsupp,
nation
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and s_nationkey = n_nationkey
) as profit
group by nation
order by nation;
```

当cost_param的bit1为0时，执行计划如下图所示：

```
 id |                           operation                           | E-rows | E-distinct | E-width | E-costs
----+---------------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Sort                                                      |      1 |            |     208 | 61.52
  2 |    ->  HashAggregate                                          |      1 |            |     208 | 61.51
  3 |       ->  Streaming (type: GATHER)                            |      2 |            |     208 | 61.51
  4 |          ->  HashAggregate                                    |      2 |            |     208 | 61.36
  5 |             ->  Hash Join (6,7)                               |      2 | 20, 15     |     176 | 61.33
  6 |                ->  Seq Scan on public.nation                  |     40 |            |     108 | 20.20
  7 |                ->  Hash                                       |      2 |            |      76 | 41.04
  8 |                   ->  Hash Join (9,16)                        |      2 | 10, 13     |      76 | 41.04
  9 |                      ->  Streaming(type: REDISTRIBUTE)        |      2 |            |      88 | 27.73
 10 |                         ->  Hash Join (11,14)                 |      2 | 10, 13     |      88 | 27.62
 11 |                            ->  Streaming(type: REDISTRIBUTE)  |     20 |            |      70 | 14.19
 12 |                               ->  Row Adapter                 |     21 |            |      70 | 13.01
 13 |                                  ->  CStore Scan on public.lineitem |     20 |            |      70 | 13.01
 14 |                            ->  Hash                           |     21 |            |      34 | 13.13
 15 |                               ->  Seq Scan on public.partsupp |     20 |            |      34 | 13.13
 16 |                      ->  Hash                                 |     21 |            |      12 | 13.13
 17 |                         ->  Seq Scan on public.supplier       |     20 |            |      12 | 13.13
(17 rows)
```

优化分析 2

在以上查询中，supplier、lineitem、partsupp三表做hashjoin的条件为(lineitem.l_suppkey = supplier.s_suppkey) AND (lineitem.l_partkey = partsupp.ps_partkey)，此hashjoin条件中存在两个过滤条件，这前一个过滤条件中的lineitem.l_suppkey和后一个过滤条件中的lineitem.l_partkey同为lineitem表的两列，这两列存在强相关的关联关系。在这种情况，估算hashjoin条件的选择率时，如果使用cost_param的bit1为0时，实际是将AND的两个过滤条件分别计算的2个选择率的值

相乘来得到hashjoin条件的选择率，导致行数估算不准确，查询性能较差。所以需要将cost_param的bit1为1时，选择最小的选择率作为总的选择率估算行数比较准确，查询性能较好，优化后的计划如下图所示：

```
 id |                           operation                           | E-rows | E-distinct | E-width | E-costs
----+---------------------------------------------------------------+--------+------------+---------+---------
  1 | ->  Sort                                                      |     10 |            |     208 | 64.42
  2 |    ->  HashAggregate                                          |     10 |            |     208 | 64.23
  3 |       ->  Streaming (type: GATHER)                            |     20 |            |     208 | 64.23
  4 |          ->  HashAggregate                                    |     20 |            |     208 | 62.71
  5 |             ->  Hash Join (6,7)                               |     20 | 20, 10     |     176 | 62.46
  6 |                ->  Seq Scan on public.nation                  |     40 |            |     108 | 20.20
  7 |                ->  Hash                                       |     20 |            |      76 | 41.97
  8 |                   ->  Hash Join (9,16)                        |     20 | 10, 13     |      76 | 41.97
  9 |                      ->  Streaming(type: REDISTRIBUTE)        |     20 |            |      82 | 28.54
 10 |                         ->  Hash Join (11,14)                 |     20 | 10, 13     |      82 | 27.63
 11 |                            ->  Streaming(type: REDISTRIBUTE)  |     20 |            |      70 | 14.19
 12 |                               ->  Row Adapter                 |     21 |            |      70 | 13.01
 13 |                                  ->  CStore Scan on public.lineitem | 20 |            |      70 | 13.01
 14 |                            ->  Hash                           |     21 |            |      12 | 13.13
 15 |                               ->  Seq Scan on public.supplier |     20 |            |      12 | 13.13
 16 |                      ->  Hash                                 |     21 |            |      34 | 13.13
 17 |                         ->  Seq Scan on public.partsupp       |     20 |            |      34 | 13.13
(17 rows)
```

14.4.6 案例：调整分布键

现象描述

某局点测试过程中EXPLAIN ANALYZE后有如下情况：

```
 id |                                operation                                |          A-time          | A-rows | E-rows |  Peak Memory  | E-memory | A-width | E-width |   E-costs
----+-------------------------------------------------------------------------+--------------------------+--------+--------+---------------+----------+---------+---------+-------------
  1 | ->  Streaming (type: GATHER)                                            | 94138.404                |      0 | 670912 | 292KB         |          |         |      73 | 102576573.63
  2 |    ->  Insert on temp_calc_emprate0101_t3                               | [93259.538,93430.438]    |    310 | 670912 | [1108KB, 1108KB] | 1MB      |         |      73 | 102534641.63
  3 |       ->  Streaming(type: REDISTRIBUTE)                                 | [93259.507,93430.400]    |    310 | 670912 | [2091KB, 2093KB] | 1MB      |         |      73 | 102534641.63
  4 |          ->  Subquery Scan on "*SELECT*"                                | [93212.430,93419.986]    |    310 | 670912 | [7KB, 7KB]    | 1MB      |         |      73 | 102533776.78
  5 |             ->  HashAggregate                                           | [93212.425,93419.980]    |    310 | 670912 | [145KB, 197KB] | 16MB     | [65,65] |      45 | 102533645.74
  6 |                ->  Streaming(type: REDISTRIBUTE)                        | [93212.374,93419.924]    |   5586 | 670934 | [2091KB, 2093KB] | 1MB      |         |      45 | 102533305.05
  7 |                   ->  Hash Join (8,12)                                  | [2657.406,93339.924]     |   5586 | 670934 | [20KB, 20KB]  | 1MB      |         |      45 | 102532655.39
  8 |                      ->  Seq Scan on s_riskrate_setting a               | [38.885,2940.983]        | 77252027 | 78594218 | [812KB, 903KB] | 1MB      |         |      56 | 275264.71
  9 |                      ->  Hash                                           | [2241.418,2713.381]      |   8536241 |   8536241 | [1031KB, 97803KB] | 16MB     | [48,48] |      46 | 50870.88
 10 |                         ->  Streaming(type: REDISTRIBUTE)               | [210.226,2617.195]       |   8536241 |   8536241 | [2091KB, 2093KB] | 1MB      |         |      46 | 50870.88
 11 |                            ->  Seq Scan on temp_calc_emprate0101 b      | [86.790,141.293]         |   8536241 |   8536241 | [16KB, 16KB]  | 1MB      |         |      46 | 11564.79
(11 rows)
```

从执行信息上比较明确的可以看出HashJoin是整个计划的性能瓶颈点，并且从HashJoin的执行时间信息[2657.406,93339.924](数值的具体含义请参见**SQL执行计划详解**)，上可以看出HashJoin在不同的DN上存在严重的计算偏斜。

同时在Memory Information(如下图)中可以看出各个节点的内存资源消耗也存在极为严重的偏斜。

```
                Memory Information (identified by plan id)
--------------------------------------------------------------------------------
Coordinator:
  Query Peak Memory: 4MB
Datanode:
  Max Query Peak Memory: 118MB
  Min Query Peak Memory: 24MB
 12 --Hash
        Max Buckets: 131072  Max Batches: 1  Max Memory Usage: 91857kB
        Min Buckets: 131072  Min Batches: 1  Min Memory Usage: 0kB
(8 rows)
```

优化分析

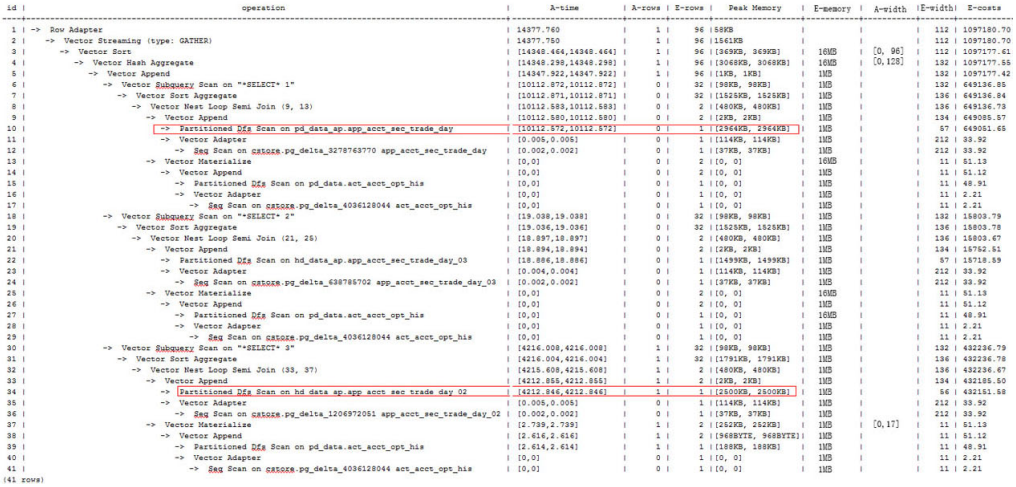
上述两个特征表明了此SQL语句存在极为严重的计算倾斜。进一步向HashJoin算子的下层分析发现Seq Scan on s_riskrate_setting也存在极为严重的计算倾斜[38.885,2940.983]。根据Scan的含义推测此计划性能问题的根源在于表

s_riskrate_setting数据的分布倾斜。实际分析之后确实发现表s_riskrate_setting存在严重的数据倾斜。整改之后性能从94s提升为50s。

14.4.7 案例：调整局部聚簇键

现象描述

某局点EXPLAIN PERFORMANCE信息如下。分析发现如图红框标识的两个性能瓶颈点均为表Scan动作。

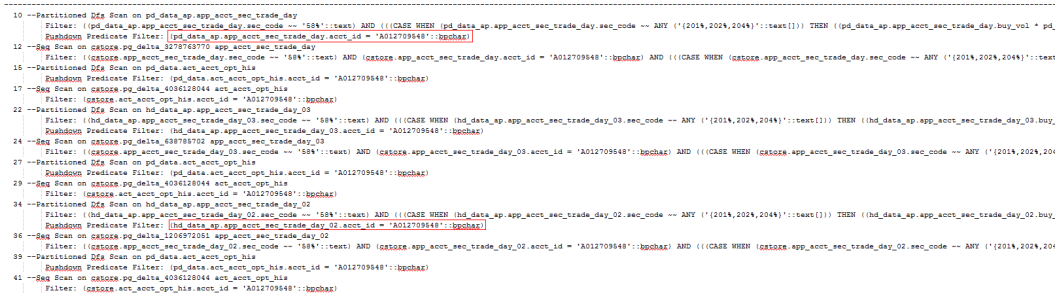


id	operation	A-time	A-rows	E-rows	Peak Memory	E-memory	A-width	E-width	E-costs
1	-> Row Adapter	14377.760	1	96	58KB				112	1097180.70
2	-> Vector Streaming (type: GATHER)	14377.750	1	96	1561KB				112	1097180.70
3	-> Vector Sort	[14348.464,14348.464]	1	96	[369KB, 369KB]	16MB	[0, 96]		112	1097177.61
4	-> Vector Hash Aggregate	[14348.298,14348.298]	1	96	[3068KB, 3068KB]	16MB	[0,128]		132	1097177.55
5	-> Vector Append	[14347.922,14347.922]	1	96	[1KB, 1KB]	1MB			132	1097177.42
6	-> Vector Subquery Scan on "*SELECT* 1"	[10112.872,10112.872]	0	32	[98KB, 98KB]	1MB			132	649136.85
7	-> Vector Sort Aggregate	[10112.871,10112.871]	0	32	[1525KB, 1525KB]	1MB			136	649136.84
8	-> Vector Nest Loop Semi Join (9, 13)	[10112.583,10112.583]	0	2	[480KB, 480KB]	1MB			136	649136.73
9	-> Vector Append	[10112.580,10112.580]	0	2	[2KB, 2KB]	1MB			134	649085.57
10	-> Partitioned Dfs Scan on pd_data_ap.app_acct_sec_trade_day	[10112.572,10112.572]	0	1	[2964KB, 2964KB]	1MB			67	649061.65
11	-> Vector Adapter	[0.005,0.005]	0	1	[114KB, 114KB]	1MB			212	33.92
12	-> Seq Scan on cstore.pg_delta_3278763770 app_acct_sec_trade_day	[0.002,0.002]	0	1	[37KB, 37KB]	1MB			212	33.92
13	-> Vector Materialize	[0,0]	0	2	[0, 0]	16MB			11	51.13
14	-> Vector Append	[0,0]	0	2	[0, 0]	1MB			11	51.12
15	-> Partitioned Dfs Scan on pd_data.act_acct_opt_his	[0,0]	0	1	[0, 0]	1MB			11	48.91
16	-> Vector Adapter	[0,0]	0	1	[0, 0]	1MB			11	2.21
17	-> Seq Scan on cstore.pg_delta_4036128044 act_acct_opt_his	[0,0]	0	1	[0, 0]	1MB			11	2.21
18	-> Vector Subquery Scan on "*SELECT* 2"	[19.038,19.038]	0	32	[98KB, 98KB]	1MB			132	15803.79
19	-> Vector Sort Aggregate	[19.036,19.036]	0	32	[1525KB, 1525KB]	1MB			136	15803.78
20	-> Vector Nest Loop Semi Join (21, 25)	[18.897,18.897]	0	2	[480KB, 480KB]	1MB			136	15803.67
21	-> Vector Append	[18.894,18.894]	0	2	[2KB, 2KB]	1MB			134	15752.51
22	-> Partitioned Dfs Scan on hd_data_ap.app_acct_sec_trade_day_03	[18.886,18.886]	0	1	[1499KB, 1499KB]	1MB			67	15718.59
23	-> Vector Adapter	[0.004,0.004]	0	1	[114KB, 114KB]	1MB			212	33.92
24	-> Seq Scan on cstore.pg_delta_638785702 app_acct_sec_trade_day_03	[0.002,0.002]	0	1	[37KB, 37KB]	1MB			212	33.92
25	-> Vector Materialize	[0,0]	0	2	[0, 0]	16MB			11	51.13
26	-> Vector Append	[0,0]	0	2	[0, 0]	1MB			11	51.12
27	-> Partitioned Dfs Scan on pd_data.act_acct_opt_his	[0,0]	0	1	[0, 0]	16MB			11	48.91
28	-> Vector Adapter	[0,0]	0	1	[0, 0]	1MB			11	2.21
29	-> Seq Scan on cstore.pg_delta_4036128044 act_acct_opt_his	[0,0]	0	1	[0, 0]	1MB			11	2.21
30	-> Vector Subquery Scan on "*SELECT* 3"	[4216.008,4216.008]	1	32	[98KB, 98KB]	1MB			132	432236.79
31	-> Vector Sort Aggregate	[4216.004,4216.004]	1	32	[1791KB, 1791KB]	1MB			136	432236.78
32	-> Vector Nest Loop Semi Join (33, 37)	[4215.608,4215.608]	1	2	[480KB, 480KB]	1MB			136	432236.67
33	-> Vector Append	[4212.855,4212.855]	1	2	[2KB, 2KB]	1MB			134	432185.50
34	-> Partitioned Dfs Scan on hd_data_ap.app_acct_sec_trade_day_02	[4212.846,4212.846]	1	1	[2500KB, 2500KB]	1MB			56	432151.58
35	-> Vector Adapter	[0.005,0.005]	0	1	[114KB, 114KB]	1MB			212	33.92
36	-> Seq Scan on cstore.pg_delta_1206972051 app_acct_sec_trade_day_02	[0.002,0.002]	0	1	[37KB, 37KB]	1MB			212	33.92
37	-> Vector Materialize	[2.739,2.739]	1	2	[252KB, 252KB]	1MB	[0,17]		11	51.13
38	-> Vector Append	[2.616,2.616]	1	2	[968BYTE, 968BYTE]	1MB			11	51.12
39	-> Partitioned Dfs Scan on pd_data.act_acct_opt_his	[2.614,2.614]	1	1	[188KB, 188KB]	1MB			11	48.91
40	-> Vector Adapter	[0,0]	0	1	[0, 0]	1MB			11	2.21
41	-> Seq Scan on cstore.pg_delta_4036128044 act_acct_opt_his	[0,0]	0	1	[0, 0]	1MB			11	2.21

(41 rows)

优化分析

进一步分析表Scan的filter条件发现两个表存在acct_id = 'A012709548'::bpchar这样的filter条件。



```
10 --Partitioned Dfs Scan on pd_data_ap.app_acct_sec_trade_day
     Filter: ((pd_data_ap.app_acct_sec_trade_day.sec_code ~~ '58%'::text) AND (((CASE WHEN (pd_data_ap.app_acct_sec_trade_day.sec_code ~~ ANY ('{201%,202%,204%}'::text[])) THEN ((pd_data_ap.app_acct_sec_trade_day.buy_vol * pd_
     Pushdown Predicate Filter: (pd_data_ap.app_acct_sec_trade_day.acct_id = 'A012709548'::bpchar)
12 --Seq Scan on cstore.pg_delta_3278763770 app_acct_sec_trade_day
     Filter: ((cstore.app_acct_sec_trade_day.sec_code ~~ '58%'::text) AND (cstore.app_acct_sec_trade_day.acct_id = 'A012709548'::bpchar) AND (((CASE WHEN (cstore.app_acct_sec_trade_day.sec_code ~~ ANY ('{201%,202%,204%}'::text
15 --Partitioned Dfs Scan on pd_data.act_acct_opt_his
     Pushdown Predicate Filter: (pd_data.act_acct_opt_his.acct_id = 'A012709548'::bpchar)
17 --Seq Scan on cstore.pg_delta_4036128044 act_acct_opt_his
     Filter: (cstore.act_acct_opt_his.acct_id = 'A012709548'::bpchar)
22 --Partitioned Dfs Scan on hd_data_ap.app_acct_sec_trade_day_03
     Filter: ((hd_data_ap.app_acct_sec_trade_day_03.sec_code ~~ '58%'::text) AND (((CASE WHEN (hd_data_ap.app_acct_sec_trade_day_03.sec_code ~~ ANY ('{201%,202%,204%}'::text[])) THEN ((hd_data_ap.app_acct_sec_trade_day_03.buy_
     Pushdown Predicate Filter: (hd_data_ap.app_acct_sec_trade_day_03.acct_id = 'A012709548'::bpchar)
24 --Seq Scan on cstore.pg_delta_638785702 app_acct_sec_trade_day_03
     Filter: ((cstore.app_acct_sec_trade_day_03.sec_code ~~ '58%'::text) AND (cstore.app_acct_sec_trade_day_03.acct_id = 'A012709548'::bpchar) AND (((CASE WHEN (cstore.app_acct_sec_trade_day_03.sec_code ~~ ANY ('{201%,202%,20
27 --Partitioned Dfs Scan on pd_data.act_acct_opt_his
     Pushdown Predicate Filter: (pd_data.act_acct_opt_his.acct_id = 'A012709548'::bpchar)
29 --Seq Scan on cstore.pg_delta_4036128044 act_acct_opt_his
     Filter: (cstore.act_acct_opt_his.acct_id = 'A012709548'::bpchar)
34 --Partitioned Dfs Scan on hd_data_ap.app_acct_sec_trade_day_02
     Filter: ((hd_data_ap.app_acct_sec_trade_day_02.sec_code ~~ '58%'::text) AND (((CASE WHEN (hd_data_ap.app_acct_sec_trade_day_02.sec_code ~~ ANY ('{201%,202%,204%}'::text[])) THEN ((hd_data_ap.app_acct_sec_trade_day_02.buy_
     Pushdown Predicate Filter: (hd_data_ap.app_acct_sec_trade_day_02.acct_id = 'A012709548'::bpchar)
36 --Seq Scan on cstore.pg_delta_1206972051 app_acct_sec_trade_day_02
     Filter: ((cstore.app_acct_sec_trade_day_02.sec_code ~~ '58%'::text) AND (cstore.app_acct_sec_trade_day_02.acct_id = 'A012709548'::bpchar) AND (((CASE WHEN (cstore.app_acct_sec_trade_day_02.sec_code ~~ ANY ('{201%,202%,20
39 --Partitioned Dfs Scan on pd_data.act_acct_opt_his
     Pushdown Predicate Filter: (pd_data.act_acct_opt_his.acct_id = 'A012709548'::bpchar)
41 --Seq Scan on cstore.pg_delta_4036128044 act_acct_opt_his
     Filter: (cstore.act_acct_opt_his.acct_id = 'A012709548'::bpchar)
```

试着给两个表的acct_id列增加局部聚簇键，然后对两张表执行VACUUM FULL，使局部聚簇生效。调整后性能得到提升。

14.4.8 案例：调整中间表存储方式

现象描述

在GaussDB(DWS)中行存表天然的使用行执行引擎，列存表天然的使用列执行引擎。如果一个SQL语句涉及的表既有行存表又有列存表，系统会自动选择行执行引擎。由于列执行引擎的性能(除indexscan相关的算子)比行执行引擎性能要好很多，因此一般建议使用列存表。特别是对一些中间结果集转储的表，一定要分析清楚，使用合适的表存储类型。

某局点测试过程遇到如下的执行计划，客户希望将性能提升至3s内返回结果。

```
 id |                                  operation                                    |        A-time         |  A-rows  |  E-rows   |   Peak Memory     | E-memory | A-width | E-width | E-costs
----+-------------------------------------------------------------------------------+-----------------------+----------+-----------+-------------------+----------+---------+---------+-----------
  1 | ->  Streaming (type: GATHER)                                                  | 4651.039              |        7 |        17 | 304KB             | 1MB      |         |      41 | 101740.13
  2 |    ->  Hash Join (3,7)                                                        | [4.801,4629.889]      |        7 |        17 | [8KB, 8KB]        | 1MB      |         |      41 | 101739.10
  3 |       ->  Append                                                              | [3474.514,3840.836]   | 34752433 | 108109436 | [1KB, 1KB]        | 1MB      |         |      49 | 88969.75
  4 |          ->  Row Adapter                                                      | [2727.901,3040.924]   | 33011417 |  99098596 | [49KB, 49KB]      | 1MB      |         |      49 | 70615.19
  5 |             ->  Partitioned Dfs Scan on sd_data.act_account_his ta            | [2288.421,2558.707]   | 33011417 |  99098596 | [1002KB, 1012KB]  | 1MB      |         |      49 | 70615.19
  6 |          ->  Seq Scan on cstore.pg_delta_2425217623 ta                        | [163.977,169.707]     |  1741016 |   9010840 | [15KB, 15KB]      | 1MB      |         |      50 | 18354.56
  7 |       ->  Hash                                                                | [4.385,7.999]         |        9 |        32 | [260KB, 292KB]    | 16MB     | [0, 36] |      30 | 100.17
  8 |          ->  Streaming(type: REDISTRIBUTE)                                    | [4.384,7.997]         |        9 |        32 | [1054KB, 1055KB]  | 1MB      |         |      30 | 100.17
  9 |             ->  Hash Join (10,11)                                             | [0.162,1.043]         |        9 |        32 | [5KB, 5KB]        | 1MB      |         |      30 | 100.06
 10 |                ->  Seq Scan on pg_temp_cn_5001_140148717123328.input_acct_id_tbl tb | [0.005,0.176]   |     1030 |     31968 | [11KB, 11KB]      | 1MB      |         |      11 | 15.99
 11 |                ->  Hash                                                       | [0.001,0.849]         |        9 |        32 | [258KB, 290KB]    | 16MB     | [0, 37] |      19 | 80.31
 12 |                   ->  HashAggregate                                           | [0.001,0.849]         |        9 |        32 | [10KB, 13KB]      | 1MB      |         |      19 | 80.30
 13 |                      ->  Seq Scan on public.row_unlogged_table                | [0.000,0.847]         |      449 |       449 | [13KB, 13KB]      | 1MB      |         |      19 | 78.70
(13 rows)
```

优化分析

经过分析发现计划走了行引擎。根本原因是：临时计划表input_acct_id_tbl和中间结果转储表row_unlogged_table使用了行存表。

修改这两个表为列存表之后，性能提升至1.6s。

```
 id |                                  operation                                    |        A-time         |  A-rows  |  E-rows   |   Peak Memory     | E-memory | A-width | E-width | E-costs
----+-------------------------------------------------------------------------------+-----------------------+----------+-----------+-------------------+----------+---------+---------+-----------
  1 | ->  Row Adapter                                                               | 1567.367              |        7 |        17 | 38KB              |          |         |      41 | 101758.52
  2 |    ->  Vector Streaming (type: GATHER)                                        | 1567.349              |        7 |        17 | 393KB             |          |         |      41 | 101758.52
  3 |       ->  Vector Hash Join (4,8)                                              | [8.130,1529.101]      |        7 |        17 | [2362KB, 2446KB]  | 16MB     |         |      41 | 101757.48
  4 |          ->  Vector Append                                                    | [542.823,1452.479]    |  5681770 | 108109436 | [1KB, 1KB]        | 1MB      |         |      49 | 88969.75
  5 |             ->  Partitioned Dfs Scan on sd_data.act_account_his ta            | [295.796,1195.830]    |  3940754 |  99098596 | [861KB, 1012KB]   | 1MB      |         |      49 | 70615.19
  6 |             ->  Vector Adapter                                                | [236.065,260.284]     |  1741016 |   9010840 | [129KB, 129KB]    | 1MB      |         |      50 | 18354.56
  7 |                ->  Seq Scan on cstore.pg_delta_2425217623 ta                  | [152.595,168.048]     |  1741016 |   9010840 | [15KB, 15KB]      | 1MB      |         |      50 | 18354.56
  8 |          ->  Vector Streaming(type: REDISTRIBUTE)                             | [7.727,12.981]        |        9 |        32 | [1092KB, 1141KB]  | 1MB      | [0, 40] |      30 | 118.56
  9 |             ->  Vector Hash Join (10,11)                                      | [0.132,4.955]         |        9 |        32 | [2217KB, 2217KB]  | 16MB     |         |      30 | 118.45
 10 |                ->  CStore Scan on pg_temp_cn_5001_140148155066112.input_acct_id_tbl tb | [4.372,4.372] |      999 |     31968 | [207KB, 207KB]    | 1MB      |         |      11 | 81.00
 11 |                ->  Vector Hash Aggregate                                      | [0.062,0.209]         |        9 |        32 | [2225KB, 2225KB]  | 16MB     | [0, 35] |      19 | 33.67
 12 |                   ->  CStore Scan on public.col_unlogged_table                | [0.011,0.107]         |      449 |       449 | [541KB, 598KB]    | 1MB      |         |      19 | 32.08
(12 rows)
```

14.4.9 案例：调整局部聚簇列

现象描述

某局点测试过程中出现如下计划，客户要求将性能提升至3s内返回。

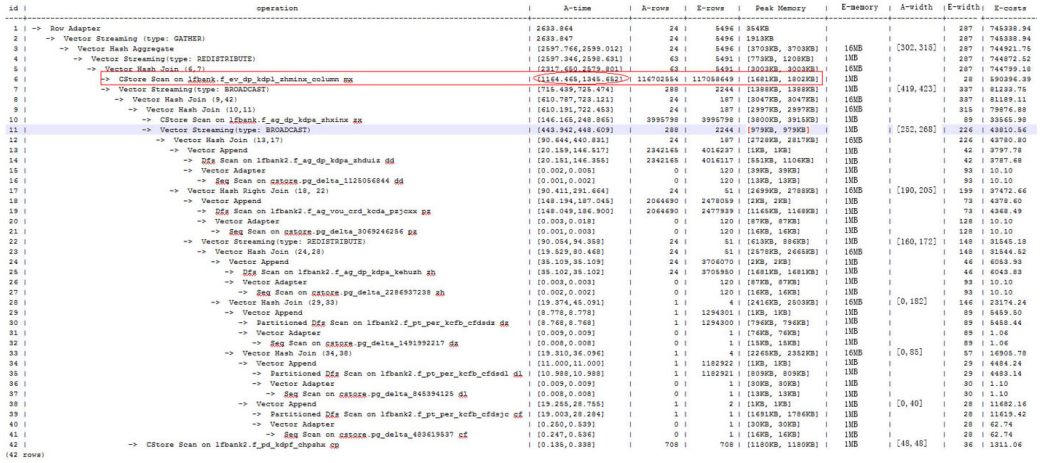
```
 id |                                  operation                                    |         A-time          |  A-rows   |  E-rows   |   Peak Memory     | E-memory |  A-width   | width | E-costs
----+-------------------------------------------------------------------------------+-------------------------+-----------+-----------+-------------------+----------+------------+-------+-----------
  1 | ->  Row Adapter                                                               | 11626.124               |        24 |      5472 | 354KB             |          |            |   287 | 592214.49
  2 |    ->  Vector Streaming (type: GATHER)                                        | 11626.091               |        24 |      5472 | 1913KB            |          |            |   287 | 592214.49
  3 |       ->  Vector Hash Aggregate                                               | [11586.971,11588.002]   |        24 |      5472 | [3703KB, 3703KB]  | 16MB     | [302,315]  |   287 | 591799.12
  4 |          ->  Vector Streaming(type: REDISTRIBUTE)                             | [11586.355,11587.436]   |        63 |      5476 | [773KB, 1208KB]   | 1MB      |            |   287 | 591750.10
  5 |             ->  Vector Hash Join (6,10)                                       | [11046.690,11554.757]   |        63 |      5476 | [3091KB, 3091KB]  | 16MB     |            |   287 | 591677.08
  6 |                ->  Vector Append                                              | [9629.968,10086.082]    | 108402955 | 116745164 | [1KB, 1KB]        | 1MB      |            |    27 | 448209.06
  7 |                   ->  Partitioned Dfs Scan on lfbank.f_ev_dp_kdpl_zhminx mx   | [8513.733,8948.960]     |  89159360 |  97499543 | [20473KB, 20587KB]| 1MB      |            |    27 | 381046.53
  8 |                   ->  Vector Adapter                                          | [1005.755,1165.251]     |  19243595 |  19245621 | [122KB, 123KB]    | 1MB      |            |    27 | 67162.52
  9 |                      ->  Seq Scan on cstore.pg_delta_2186520081 mx            | [515.208,617.631]       |  19243595 |  19245621 | [22KB, 22KB]      | 1MB      |            |    27 | 67162.52
 10 |                ->  Vector Streaming(type: BROADCAST)                          | [961.659,978.946]       |       288 |      2244 | [1388KB, 1388KB]  | 1MB      | [302,315]  |   335 | 70494.93
 11 |                   ->  Vector Hash Join (12,48)                                | [784.660,964.821]       |        24 |       187 | [3047KB, 3047KB]  | 16MB     |            |   335 | 70450.28
 12 |                      ->  Vector Hash Join (13,17)                             | [784.245,964.340]       |        24 |       187 | [3086KB, 3086KB]  | 16MB     |            |   313 | 69138.06
 13 |                         ->  Vector Append                                     | [357.334,531.923]       |   3995798 |   3995918 | [2KB, 2KB]        | 1MB      |            |    87 | 22827.08
 14 |                            ->  Partitioned Dfs Scan on lfbank2.f_ag_dp_kdpa_zhxinx zx | [357.094,531.672] | 3995798 |   3995798 | [2185KB, 2190KB]  | 1MB      |            |    87 | 22816.98
 15 |                            ->  Vector Adapter                                 | [0.003,0.006]           |         0 |       120 | [132KB, 132KB]    | 1MB      |            |   168 | 10.10
 16 |                               ->  Seq Scan on cstore.pg_delta_950882137 zx    | [0.001,0.004]           |         0 |       120 | [18KB, 18KB]      | 1MB      |            |   168 | 10.10
 17 |                         ->  Vector Streaming(type: BROADCAST)                 | [398.300,416.767]       |       288 |      2244 | [979KB, 979KB]    | 1MB      | [252,268]  |   226 | 43810.56
 18 |                            ->  Vector Hash Join (19,23)                       | [89.939,410.302]        |        24 |       187 | [2728KB, 2817KB]  | 16MB     |            |   226 | 43780.80
 19 |                               ->  Vector Append                               | [18.727,130.776]        |   2342165 |   4016237 | [1KB, 1KB]        | 1MB      |            |    42 | 3797.78
 20 |                                  ->  Dfs Scan on lfbank2.f_ag_dp_kdpa_zhduiz dd | [18.720,130.588]      |   2342165 |   4016117 | [551KB, 1106KB]   | 1MB      |            |    42 | 3787.68
 21 |                                  ->  Vector Adapter                           | [0.002,0.022]           |         0 |       120 | [39KB, 39KB]      | 1MB      |            |    93 | 10.10
 22 |                                     ->  Seq Scan on cstore.pg_delta_1125056844 dd | [0.001,0.020]       |         0 |       120 | [13KB, 13KB]      | 1MB      |            |    93 | 10.10
 23 |                               ->  Vector Hash Right Join (24, 28)             | [89.699,268.539]        |        24 |        51 | [2699KB, 2788KB]  | 16MB     | [190,205]  |   199 | 37472.66
 24 |                                  ->  Vector Append                            | [140.984,164.540]       |   2064690 |   2478059 | [2KB, 2KB]        | 1MB      |            |    73 | 4378.60
 25 |                                     ->  Dfs Scan on lfbank2.f_ag_vou_crd_kcda_pzjcxx pz | [140.833,164.367] | 2064690 |   2477939 | [1165KB, 1168KB]  | 1MB      |            |    73 | 4368.49
 26 |                                     ->  Vector Adapter                        | [0.003,0.005]           |         0 |       120 | [87KB, 87KB]      | 1MB      |            |   128 | 10.10
 27 |                                        ->  Seq Scan on cstore.pg_delta_3069246256 pz | [0.001,0.002]    |         0 |       120 | [16KB, 16KB]      | 1MB      |            |   128 | 10.10
 28 |                                  ->  Vector Streaming(type: REDISTRIBUTE)     | [75.277,94.315]         |        24 |        51 | [613KB, 886KB]    | 1MB      | [160,172]  |   148 | 31545.18
 29 |                                     ->  Vector Hash Join (30,34)              | [19.793,76.929]         |        24 |        51 | [2578KB, 2665KB]  | 16MB     |            |   148 | 31544.52
 30 |                                        ->  Vector Append                      | [33.191,33.191]         |        24 |   3706070 | [2KB, 2KB]        | 1MB      |            |    46 | 6053.93
 31 |                                           ->  Dfs Scan on lfbank2.f_ag_dp_kdpa_kehuzh zh | [33.184,33.184] |    24 |   3705950 | [1681KB, 1681KB]  | 1MB      |            |    46 | 6043.83
 32 |                                           ->  Vector Adapter                  | [0.003,0.003]           |         0 |       120 | [87KB, 87KB]      | 1MB      |            |    93 | 10.10
 33 |                                              ->  Seq Scan on cstore.pg_delta_2286937238 zh | [0.001,0.001] |     0 |       120 | [16KB, 16KB]      | 1MB      |            |    93 | 10.10
 34 |                                        ->  Vector Hash Join (35,39)           | [19.617,43.471]         |         1 |         4 | [2416KB, 2503KB]  | 16MB     | [0,182]    |   146 | 23174.24
 35 |                                           ->  Vector Append                   | [9.048,9.048]           |         1 |   1294301 | [1KB, 1KB]        | 1MB      |            |    89 | 5459.50
 36 |                                              ->  Partitioned Dfs Scan on lfbank2.f_pt_per_kcfb_cfdsdz dz | [9.035,9.035] | 1 | 1294300 | [796KB, 796KB] | 1MB      |            |    89 | 5458.44
 37 |                                              ->  Vector Adapter               | [0.011,0.011]           |         0 |         1 | [76KB, 76KB]      | 1MB      |            |    89 | 1.06
 38 |                                                 ->  Seq Scan on cstore.pg_delta_1491992217 dz | [0.009,0.009] |   0 |         1 | [15KB, 15KB]      | 1MB      |            |    89 | 1.06
 39 |                                           ->  Vector Hash Join (40,44)        | [19.509,34.169]         |         1 |         4 | [2265KB, 2352KB]  | 16MB     | [0,85]     |    57 | 16905.78
 40 |                                              ->  Vector Append                | [10.964,10.964]         |         1 |   1182922 | [1KB, 1KB]        | 1MB      |            |    29 | 4484.24
 41 |                                                 ->  Partitioned Dfs Scan on lfbank2.f_pt_per_kcfb_cfdsdl dl | [10.950,10.950] | 1 | 1182921 | [809KB, 809KB] | 1MB    |            |    29 | 4483.14
 42 |                                                 ->  Vector Adapter            | [0.011,0.011]           |         0 |         1 | [30KB, 30KB]      | 1MB      |            |    30 | 1.10
 43 |                                                    ->  Seq Scan on cstore.pg_delta_845394125 dl | [0.010,0.010] |  0 |         1 | [13KB, 13KB]      | 1MB      |            |    30 | 1.10
 44 |                                              ->  Vector Append                | [19.458,26.149]         |         1 |         2 | [1KB, 1KB]        | 1MB      | [0,40]     |    28 | 11682.16
 45 |                                                 ->  Partitioned Dfs Scan on lfbank2.f_pt_per_kcfb_cfdsjc cf | [19.155,25.863] | 1 | 1 | [1691KB, 1786KB] | 1MB      |            |    28 | 11619.42
 46 |                                                 ->  Vector Adapter            | [0.217,0.390]           |         0 |         1 | [30KB, 30KB]      | 1MB      |            |    28 | 62.74
 47 |                                                    ->  Seq Scan on cstore.pg_delta_483619537 cf | [0.215,0.388] |  0 |         1 | [16KB, 16KB]      | 1MB      |            |    28 | 62.74
 48 |                   ->  CStore Scan on lfbank2.f_pd_kdpf_chpshx cp              | [0.072,0.195]           |       708 |       708 | [1180KB, 1180KB]  | 1MB      | [48,48]    |    36 | 1311.06
(48 rows)
```

优化分析

分析发现上述计划的性能瓶颈点为lfbank.f_ev_dp_kdpl_zhminx的scan。进一步分析此表的Scan条件如下：

```
                              Predicate Information (identified by plan id)
------------------------------------------------------------------------------------------------------------
  5 --Vector Hash Join (6,10)
        Hash Cond: (((mx.zhanghao)::text = (dd.zhanghao)::text) AND ((mx.farendma)::text = (cf.farendma)::text))
  7 --Partitioned Dfs Scan on lfbank.f_ev_dp_kdpl_zhminx mx
        Pushdown Predicate Filter: ((mx.yezdminc)::text = 'DPLDGBAL'::text)
  9 --Seq Scan on cstore.pg_delta_2186520081 mx
        Filter: ((mx.yezdminc)::text = 'DPLDGBAL'::text)
        Rows Removed by Filter: 489
```


尝试把lfbank.f_ev_dp_kdpl_zhmin表修改为列存表，然后在yezdminc列上建PCK（局部聚簇），并设置PARTIAL_CLUSTER_ROWS=100000000。执行计划优化为：



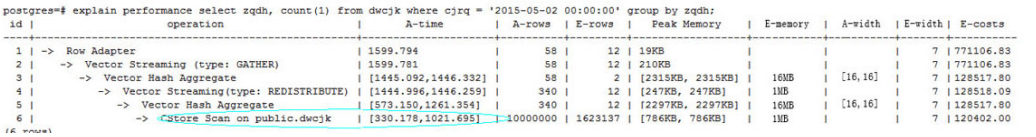
说明

- 此方法实际是靠牺牲数据导入时的性能来提升业务查询性能。
- 此方法导致局部排序的元组数增加，需要增大psort_work_mem来提高排序效率。

14.4.10 案例：改建分区表

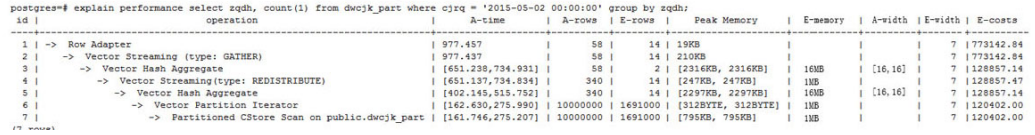
现象描述

如下简单SQL语句查询， 性能瓶颈点在dwcjk的Scan上。



优化分析

从业务层确认表数据(在cjrq字段上)有明显的日期特征，符合分区表的特征。重新规划dwcjk表的表定义：字段cjrq为分区键、天为间隔单位定义分区表dwcjk_part。修改后结果如下，性能提升近1倍。



14.4.11 案例：调整 GUC 参数 best_agg_plan

现象描述

t1的表定义为：

```
create table t1(a int, b int, c int) distribute by hash(a);
```

假设agg下层算子所输出结果集的分布列为setA，agg操作的group by列为setB，则在Stream框架下，Agg操作可以分为两个场景。

1. setA是setB的一个子集。

   对于这种场景，直接对下层结果集进行汇聚的结果就是正确的汇聚结果，上层算子直接使用即可。如下图所示：

```
explain select a, count(1) from t1 group by a;
 id |          operation           | E-rows | E-width | E-costs
----+------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER) |     30 |       4 | 15.56
  2 |    ->  HashAggregate         |     30 |       4 | 14.31
  3 |       ->  Seq Scan on t1     |     30 |       4 | 14.14
(3 rows)
```

2. setA不是setB的一个子集。

   对于这种场景，Stream执行框架分为如下三种计划形态：

   hashagg+gather(redistribute)+hashagg

   redistribute+hashagg(+gather)

   hashagg+redistribute+hashagg(+gather)

   GaussDB(DWS)提供了guc参数best_agg_plan来干预执行计划，强制其生成上述对应的执行计划，此参数取值范围为0，1，2，3

   - 取值为1时，强制生成第一种计划。
   - 取值为2时，如果group by列可以重分布，强制生成第二种计划，否则生成第一种计划。
   - 取值为3时，如果group by列可以重分布，强制生成第三种计划，否则生成第一种计划。
   - 取值为0时，优化器会根据以上三种计划的估算代价选择最优的一种计划生成。

   具体影响请看下述图片

```
set best_agg_plan to 1;
SET
explain select b,count(1) from t1 group by b;
 id |           operation            | E-rows | E-width | E-costs
----+--------------------------------+--------+---------+---------
  1 | ->  HashAggregate              |      8 |       4 | 15.83
  2 |    ->  Streaming (type: GATHER) |     25 |       4 | 15.83
  3 |       ->  HashAggregate        |     25 |       4 | 14.33
  4 |          ->  Seq Scan on t1    |     30 |       4 | 14.14
(4 rows)
set best_agg_plan to 2;
SET
explain select b,count(1) from t1 group by b;
 id |               operation               | E-rows | E-width | E-costs
----+---------------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER)          |     30 |       4 | 15.85
  2 |    ->  HashAggregate                  |     30 |       4 | 14.60
  3 |       ->  Streaming(type: REDISTRIBUTE) |     30 |       4 | 14.45
  4 |          ->  Seq Scan on t1           |     30 |       4 | 14.14
(4 rows)
set best_agg_plan to 3;
SET
explain select b,count(1) from t1 group by b;
 id |               operation               | E-rows | E-width | E-costs
----+---------------------------------------+--------+---------+---------
  1 | ->  Streaming (type: GATHER)          |     30 |       4 | 15.84
  2 |    ->  HashAggregate                  |     30 |       4 | 14.59
  3 |       ->  Streaming(type: REDISTRIBUTE) |     25 |       4 | 14.59
  4 |          ->  HashAggregate            |     25 |       4 | 14.33
  5 |             ->  Seq Scan on t1        |     30 |       4 | 14.14
(5 rows)
```

优化说明

通常优化器总会选择最优的执行计划，但是众所周知代价估算，尤其是中间结果集的代价估算一般会有比较大的偏差，这种比较大的偏差就可能会导致agg的计算方式出现比较大的偏差，这时候就需要通过best_agg_plan进行agg计算模型的干预。

一般来说，当agg汇聚的收敛度很小时，即结果集的个数在agg之后并没有明显变少时（经验上以5倍为临界点），选择redistribute+hashagg执行方式，否则选择hashagg+redistribute+hashagg执行方式。

14.4.12 案例：改写 SQL 消除子查询（案例 1）

现象描述

```
select
   1,
   (select count(*) from customer_address_001 a4 where a4.ca_address_sk = a.ca_address_sk) as GZCS
from customer_address_001 a;
```

此SQL性能较差，查看发现执行计划中存在SubPlan，具体如下：

```
postgres=# explain select 1,(select count(*)
postgres(#           from customer_address_001 a4
postgres(#          where a4.ca_address_sk = a.ca_address_sk
postgres(#         ) as GZCS from customer_address_001 a;
 id |                          operation                          | E-rows | E-width | E-costs
----+-------------------------------------------------------------+--------+ -------+---------
  1 | ->  Streaming (type: GATHER)                                |    320 |       4 | 4529.27
  2 |    ->  Seq Scan on customer_address_001 a                   |    320 |       4 | 4496.27
  3 |       ->  Aggregate  [2, SubPlan 1]                         |     32 |       4 | 139.50
  4 |          ->  Result                                         |  10240 |       4 | 138.69
  5 |             ->  Materialize                                 |  10240 |       4 | 138.69
  6 |                ->  Streaming(type: BROADCAST)               |  10240 |       4 | 137.09
  7 |                   ->  Seq Scan on customer_address_001 a4   |    320 |       4 | 32.32
(7 rows)
```

优化说明

此优化的核心就是消除子查询。分析业务场景发现*a.ca_address_sk*不为null，那么从SQL语义出发，可以等价改写SQL为：

```
select
count(*)
from customer_address_001 a4, customer_address_001 a
where a4.ca_address_sk = a.ca_address_sk
group by  a.ca_address_sk;
```

说明

为了保证改写的等效性，在*customer_address_001. ca_address_sk*加了*not null*约束。

14.4.13 案例：改写 SQL 消除子查询（案例 2）

现象描述

某局点客户反馈如下SQL语句的执行时间超过1天未结束：

```
UPDATE calc_empfyc_c_cusr1 t1
SET ln_rec_count =
(
   SELECT CASE WHEN current_date - ln_process_date + 1 <= 12 THEN 0 ELSE t2.ln_rec_count END
   FROM calc_empfyc_c1_policysend_tmp t2
```

```
    WHERE t1.ln_branch = t2.ln_branch AND t1.ls_policyno_cusr1 = t2.ls_policyno_cusr1
)
WHERE dsign = '1'
AND flag = '1'
AND EXISTS
    (SELECT 1
    FROM calc_empfyc_c1_policysend_tmp t2
    WHERE t1.ln_branch = t2.ln_branch AND t1.ls_policyno_cusr1 = t2.ls_policyno_cusr1
    );
```

对应的执行计划如下：

```
Streaming (type: GATHER)  (cost=44693.26..19548819558.34 rows=4058158 width=1061)
  Node/s: All datanodes
  ->  Update on channel.calc_empfyc_c_cusr1 t1  (cost=44689.26..19546717163.01 rows=4058158 width=1061)
        ->  Hash Join  (cost=44689.26..19546717163.01 rows=4058158 width=1061)
              Hash Cond: (((t1.ln_branch)::text = ((t2.ln_branch)::text)) AND ((t1.ls_policyno_cusr1)::text = ((t2.ls_policyno_cusr1)::text)))
              ->  Seq Scan on channel.calc_empfyc_c_cusr1 t1  (cost=0.00..28692.39 rows=7105667 width=1055)
                    Filter: ((t1.dsign = '1'::bpchar) AND (t1.flag = '1'::bpchar))
              ->  Hash  (cost=2112.16..2112.16 rows=108998016 width=37)
                    ->  Unique  (cost=2112.06..2112.16 rows=108998016 width=37)
                          ->  Sort  (cost=2112.06..2112.09 rows=775 width=37)
                                Sort Key: ((t2.ln_branch)::text), ((t2.ls_policyno_cusr1)::text)
                                ->  Streaming(type: BROADCAST)  (cost=2109.81..2111.85 rows=775 width=37)
                                      Spawn on: All datanodes
                                      ->  HashAggregate  (cost=2109.81..2109.82 rows=12 width=37)
                                            Group By Key: (t2.ln_branch)::text, (t2.ls_policyno_cusr1)::text
                                            ->  Seq Scan on channel.calc_empfyc_c1_policysend_tmp t2  (cost=0.00..1406.87 rows=1703094 width=37)
              SubPlan 1
                ->  Result  (cost=0.00..308262.89 rows=108998016 width=44)
                      Filter: (((t1.ln_branch)::text = (t2.ln_branch)::text) AND ((t1.ls_policyno_cusr1)::text = (t2.ls_policyno_cusr1)::text))
                      ->  Materialize  (cost=0.00..295489.68 rows=108998016 width=44)
                            ->  Streaming(type: BROADCAST)  (cost=0.00..286974.21 rows=108998016 width=44)
                                  Spawn on: All datanodes
                                  ->  Seq Scan on channel.calc_empfyc_c1_policysend_tmp t2  (cost=0.00..1406.87 rows=1703094 width=44)
```

优化说明

很明显，执行计划中存在SubPlan，并且SubPlan中的运算相当重，即此SubPlan是一个明确的性能瓶颈点。

根据SQL语意等价改写SQL消除SubPlan如下：

```
UPDATE calc_empfyc_c_cusr1 t1
SET ln_rec_count = CASE WHEN current_date - ln_process_date + 1 <= 12 THEN 0 ELSE t2.ln_rec_count END
FROM calc_empfyc_c1_policysend_tmp t2
WHERE
t1.dsign = '1' AND t1.flag = '1'
AND t1.ln_branch = t2.ln_branch AND t1.ls_policyno_cusr1 = t2.ls_policyno_cusr1;
```

改写之后SQL语句在50S内执行完成

14.4.14 案例：改写 SQL 排除剪枝干扰

现象描述

某局点测试中：ddw_f10_op_cust_asset_mon为分区表，分区键为year_mth，此字段是由年月两个值拼接而成的字符串。

测试SQL如下：

```
select
    count(1)
from t_ddw_f10_op_cust_asset_mon b1
where b1.year_mth between to_char(add_months(to_date("20170222",'yyyymmdd'), -11),'yyyymm') and
substr("20170222",1 ,6 );
```

测试结果显示此SQL的表Scan耗时长达135s。初步猜测可能是性能瓶颈点。

说明

add_months为本地适配函数:

```
CREATE OR REPLACE FUNCTION ADD_MONTHS(date, integer) RETURNS date
   AS $$
   SELECT
   CASE
   WHEN (EXTRACT(day FROM $1) = EXTRACT(day FROM (date_trunc('month', $1) + INTERVAL '1
month - 1 day'))) THEN
       date_trunc('month', $1) + CAST($2 + 1 || ' month - 1 day' as interval)
   ELSE
       $1 + CAST($2 || ' month' as interval)
   END
   $$
   LANGUAGE SQL
   IMMUTABLE;
```

优化说明

分析语句的执行计划，发现执行计划中显示的基表filter如下：

```
Filter: (((year_mth)::text <= '201702'::text) AND ((year_mth)::text >=
to_char(add_months(to_date('20170222'::text, 'YYYYMMDD'::text), (-11)), 'YYYYMM'::text)))
```

Filter条件中存在表达式to_char(add_months(to_date(''20170222'','yyyymmdd'), -11),'yyyymm')，这种**非常量的表达式是不能用来剪枝的**，因而会导致查询语句扫描分区表所有数据。

查询pg_proc发现此处的to_date和to_char均为stable类型的函数，根据Postgresql中对函数行为的约定，此类函数不能在预处理阶段转化为Const值，这也是不能导致分区剪枝的根本原因。

根据以上分析，优化表达式使其可以进行分区剪枝是性能优化的关键。根据语意将原SQL等价改写为：

```
select
    count(1)
from t_ddw_f10_op_cust_asset_mon b1
where b1.year_mth between(substr(ADD_MONTHS('20170222'::date, -11), 1, 4)||
substr(ADD_MONTHS('20170222'::date, -11), 6, 2)) and substr(''20170222'',1 ,6 );
```

改写之后，SQL执行时间从135s提升至18s。

14.4.15 案例：改写 SQL 消除 in-clause

现象描述

in-clause/any-clause是常见的SQL语句约束条件，有时in或any后面的clause都是常量，类似于：

```
select
count(1)
from calc_empfyc_c1_result_tmp_t1
where ls_pid_cusr1 in (‘20120405’, ‘20130405’);
```

或者

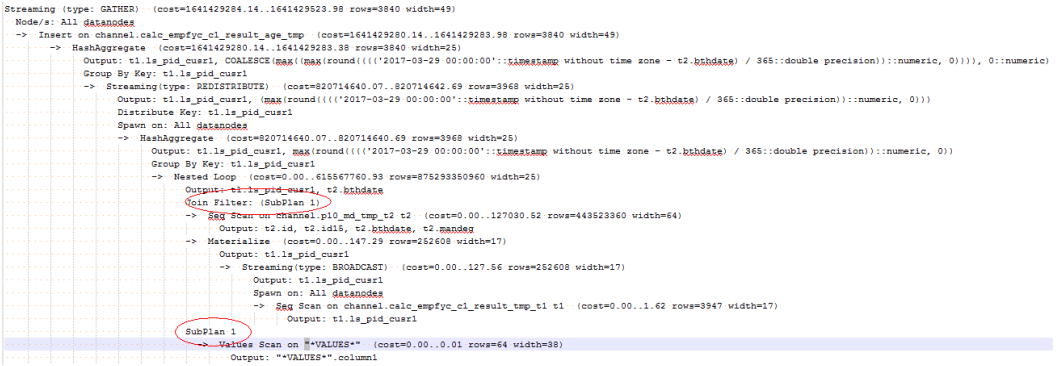
```
select
count(1)
from calc_empfyc_c1_result_tmp_t1
where ls_pid_cusr1 in any(‘20120405’, ‘20130405’);
```

但是也有一些如下的特殊用法：

```
SELECT
ls_pid_cusr1,COALESCE(max(round((current_date-bthdate)/365)),0)
FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
WHERE t1.ls_pid_cusr1 = any(values(id),(id15))
GROUP BY ls_pid_cusr1;
```

其中，id、id15为p10_md_tmp_t2中的两列，“t1.ls_pid_cusr1 = any(values(id),(id15))”等价于“t1.ls_pid_cusr1 = id or t1.ls_pid_cusr1 = id15”。

因此join-condition实质上是一个不等式，这种不等值的join操作必须走nestloop，对应执行计划如下：

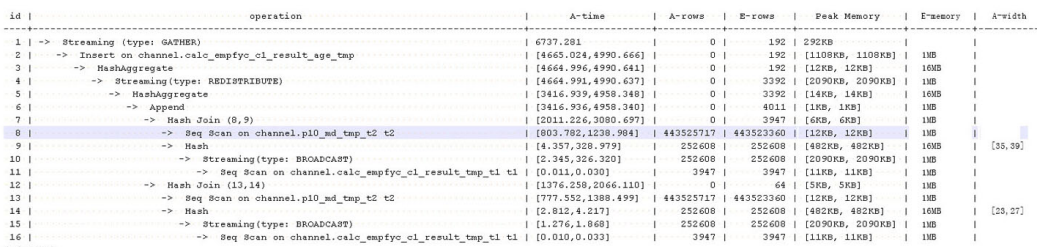


优化说明

测试发现由于两表结果集过大，导致nestloop耗时过长，超过一小时未返回结果，因此性能优化的关键是消除nestloop，让join走更高效的hashjoin。从语义等价的角度消除any-clause，SQL改写如下：

```
select
ls_pid_cusr1,COALESCE(max(round(ym/365)),0)
from
(
	(
		SELECT
			ls_pid_cusr1,(current_date-bthdate) as ym
		FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
		WHERE t1.ls_pid_cusr1 = t2.id and t1.ls_pid_cusr1 != t2.id15
	)
	union all
	(
		SELECT
			ls_pid_cusr1,(current_date-bthdate) as ym
		FROM calc_empfyc_c1_result_tmp_t1 t1,p10_md_tmp_t2 t2
		WHERE t1.ls_pid_cusr1 = id15
	)
)
GROUP BY ls_pid_cusr1;
```

优化后的SQL查询由两个等值join的子查询构成，而每个子查询都可以走更适合此场景的hashjoin。优化后的执行计划如下



id	operation	A-time	A-rows	E-rows	Peak Memory	E-memory	A-width
1	-> Streaming (type: GATHER)	6737.281	0	192	292KB		
2	-> Insert on channel.calc_empfyc_c1_result_age_tmp	[4665.024,4990.666]	0	192	[1108KB, 1108KB]	1MB	
3	-> HashAggregate	[4664.996,4990.641]	0	192	[12KB, 12KB]	16MB	
4	-> Streaming(type: REDISTRIBUTE)	[4664.991,4990.637]	0	3392	[2090KB, 2090KB]	1MB	
5	-> HashAggregate	[3416.939,4958.348]	0	3392	[14KB, 14KB]	16MB	
6	-> Append	[3416.936,4958.340]	0	4011	[1KB, 1KB]	1MB	
7	-> Hash Join (8,9)	[2011.226,3080.697]	0	3947	[6KB, 6KB]	1MB	
8	-> Seq Scan on channel.p10_md_tmp_t2 t2	[803.782,1238.984]	443525717	443523360	[12KB, 12KB]	1MB	
9	-> Hash	[4.357,328.979]	252608	252608	[482KB, 482KB]	16MB	[35,39]
10	-> Streaming(type: BROADCAST)	[2.345,326.320]	252608	252608	[2090KB, 2090KB]	1MB	
11	-> Seq Scan on channel.calc_empfyc_c1_result_tmp_t1 t1	[0.011,0.030]	3947	3947	[11KB, 11KB]	1MB	
12	-> Hash Join (13,14)	[1376.258,2066.110]	0	64	[5KB, 5KB]	1MB	
13	-> Seq Scan on channel.p10_md_tmp_t2 t2	[777.552,1388.499]	443525717	443523360	[12KB, 12KB]	1MB	
14	-> Hash	[2.812,4.217]	252608	252608	[482KB, 482KB]	16MB	[23,27]
15	-> Streaming(type: BROADCAST)	[1.276,1.868]	252608	252608	[2090KB, 2090KB]	1MB	
16	-> Seq Scan on channel.calc_empfyc_c1_result_tmp_t1 t1	[0.010,0.033]	3947	3947	[11KB, 11KB]	1MB	

(16 rows)

优化后，从超过1个小时未返回结果优化到7s返回结果。

14.4.16 案例：使用 partial cluster key

列存表可以选取某一列或几列设置为partial cluster key(column_name[, ...])。在导入数据时，按设置的列进行局部排序（默认每70个CU即420万行排序一次），生成的CU会聚集在一起，即CU的min,max会在一个较小的区间内。当查询时，where条件含有这些列时，可产生良好的过滤效果。

1. 使用partial cluster key。

```
CREATE TABLE lineitem
(
L_ORDERKEY    BIGINT NOT NULL
, L_PARTKEY     BIGINT NOT NULL
, L_SUPPKEY     BIGINT NOT NULL
, L_LINENUMBER  BIGINT NOT NULL
, L_QUANTITY    DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT    DECIMAL(15,2) NOT NULL
, L_TAX         DECIMAL(15,2) NOT NULL
, L_RETURNFLAG  CHAR(1) NOT NULL
, L_LINESTATUS  CHAR(1) NOT NULL
, L_SHIPDATE    DATE NOT NULL
, L_COMMITDATE  DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE     CHAR(10) NOT NULL
, L_COMMENT      VARCHAR(44) NOT NULL
)
with (orientation = column)
distribute by hash(L_ORDERKEY);

select
sum(l_extendedprice * l_discount) as revenue
from
lineitem
where
l_shipdate >= '1994-01-01'::date
and l_shipdate < '1994-01-01'::date + interval '1 year'
and l_discount between 0.06 - 0.01 and 0.06 + 0.01
and l_quantity < 24;
```

where条件中l_shipdate和l_quantity的distinct值数量较少且可以做min max过滤，修改表定义如下：

```
CREATE TABLE lineitem
(
L_ORDERKEY    BIGINT NOT NULL
, L_PARTKEY     BIGINT NOT NULL
, L_SUPPKEY     BIGINT NOT NULL
, L_LINENUMBER  BIGINT NOT NULL
, L_QUANTITY    DECIMAL(15,2) NOT NULL
, L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL
, L_DISCOUNT    DECIMAL(15,2) NOT NULL
, L_TAX         DECIMAL(15,2) NOT NULL
, L_RETURNFLAG  CHAR(1) NOT NULL
, L_LINESTATUS  CHAR(1) NOT NULL
, L_SHIPDATE    DATE NOT NULL
, L_COMMITDATE  DATE NOT NULL
, L_RECEIPTDATE DATE NOT NULL
, L_SHIPINSTRUCT CHAR(25) NOT NULL
, L_SHIPMODE     CHAR(10) NOT NULL
, L_COMMENT      VARCHAR(44) NOT NULL
, partial cluster key(l_shipdate, l_quantity)
)
with (orientation = column)
distribute by hash(L_ORDERKEY);
```

重新导入数据后执行查询，对比使用前后执行时间：

图 14-13 未使用 partial cluster key

```
 id |                  operation                  |      A-time       | A-rows | E-rows |  Peak Memory   | A-width | E-width |  E-costs
----+---------------------------------------------+-------------------+--------+--------+----------------+---------+---------+-----------
  1 | ->  Row Adapter                             | 1653.156          |      1 |      1 | 12KB           |         |      44 | 205803.90
  2 |    ->  Vector Aggregate                     | 1653.146          |      1 |      1 | 184KB          |         |      44 | 205803.90
  3 |       ->  Vector Streaming (type: GATHER)   | 1653.070          |      1 |      1 | 174KB          |         |      44 | 205803.90
  4 |          ->  Vector Aggregate               | [1481.497,1481.497] |      1 |      1 | [225KB, 225KB] |         |      44 | 205803.84
  5 |             ->  CStore Scan on public.lineitem | [1405.004,1405.004] | 114160 | 111485 | [792KB, 792KB] |         |      12 | 205246.40
(5 rows)
```

图 14-14 未使用 partial cluster key 后 CU 加载情况

```
  5 --CStore Scan on public.lineitem
        datanode1 (actual time=40.623..1405.004 rows=114160 loops=1)
        datanode1 (RoughCheck CU: CUNone: 0, CUSome: 101)
        datanode1 (LLVM Optimized)
        datanode1 (Buffers: shared hit=18385 read=23)
        datanode1 (CPU: ex c/r=31917, ex cyc=3643646206, inc cyc=3643646206)
```

图 14-15 使用 partial cluster key

```
 id |                  operation                  |      A-time       | A-rows | E-rows |  Peak Memory   | A-width | E-width |  E-costs
----+---------------------------------------------+-------------------+--------+--------+----------------+---------+---------+-----------
  1 | ->  Row Adapter                             | 459.539           |      1 |      1 | 12KB           |         |      44 | 205693.85
  2 |    ->  Vector Aggregate                     | 459.528           |      1 |      1 | 184KB          |         |      44 | 205693.85
  3 |       ->  Vector Streaming (type: GATHER)   | 459.452           |      1 |      1 | 174KB          |         |      44 | 205693.85
  4 |          ->  Vector Aggregate               | [285.177,285.177] |      1 |      1 | [225KB, 225KB] |         |      44 | 205693.79
  5 |             ->  CStore Scan on public.lineitem | [249.757,249.757] | 114160 |  89475 | [792KB, 792KB] |         |      12 | 205246.40
(5 rows)
```

图 14-16 使用 partial cluster key 后 CU 加载情况

```
  5 --CStore Scan on public.lineitem
        datanode1 (actual time=23.017..249.757 rows=114160 loops=1)
        datanode1 (RoughCheck CU: CUNone: 84, CUSome: 17)
        datanode1 (LLVM Optimized)
        datanode1 (Buffers: shared hit=2853 read=23)
        datanode1 (CPU: ex c/r=5673, ex cyc=647656146, inc cyc=647656146)
```

使用partial cluster key后，5-- CStore Scan on public.lineitem的时间减少了1.2s , 得益于有84个CU被过滤掉了。

2. 选取partial cluster key列。
   - 列存表支持创建partial cluster key的类型 character varying(n), varchar(n), character(n), char(n), text, nvarchar2, timestamp with time zone, timestamp without time zone, date, time without time zone, time with time zone。
   - 数据的distinct值数量较少，这样能产生较好的过滤效果。
   - 出现在查询where条件中，优先选取能过滤大量数据的列。
   - partial cluster key中设置多个列时，是先按第一个列排序，当第一个列值相同时，使用第二列比较，后续列依次类推。推荐不要超出3个列。
3. 添加partial cluster key后，优化导入性能。

   由于添加了partial cluster key，在导入时会增加排序计算，会对导入性能产生影响。当排序可以完全在内存中进行时影响较小，如果无法在内存中完成排序时，会下盘写临时文件，这时就会产生较大的影响。

   排序使用的内存通过GUC参数psort_work_mem来设置，可以设置较大的值来使用更大的内存进行排序。

   排序的数据量是通过表的存储参数PARTIAL_CLUSTER_ROWS来设置，降低这个数值，可减少一次排序的数据量。这个参数通常与存储参数MAX_BATCHROW配置使用。PARTIAL_CLUSTER_ROWS设置值必须是MAX_BATCHROW的整数倍，MAX_BATCHROW是设置单个CU中数据的最大行数。

14.5 SQL 执行 troubleshooting

14.5.1 分析查询效率异常降低的问题

通常在几十毫秒内完成的查询，有时会突然需要几秒的时间完成；而通常需要几秒完成的查询，有时需要半小时才能完成。如何分析这种查询效率异常降低的问题呢?

处理步骤

通过下列的操作步骤，可以分析出查询效率异常降低的原因。

步骤1 使用analyze命令分析数据库。

analyze命令更新所有表中数据大小以及属性等相关统计信息，该命令较为轻量级，可以经常执行。如果此命令执行后性能恢复或者有所提升，则表明autovacuum未能很好的完成它的工作，有待进一步分析。

步骤2 检查查询语句是否返回了多余的数据信息。

例如，如果查询语句先查询一个表中所有的记录，而只用到结果中的前10条记录。对于包含50条记录的表，查询起来是很快的；但是，当表中包含的记录达到50000，查询效率将会有所下降。

若业务应用中存在只需要部分数据信息，但是查询语句却是返回所有信息的情况，建议修改查询语句，增加LIMIT子句来限制返回的记录数。这样至少使数据库优化器有了一定的优化空间，一定程度上会提升查询效率。

步骤3 检查查询语句单独运行时是否仍然较慢。

尝试在数据库没有其他查询或查询较少的时候运行查询语句，并观察运行效率。如果效率较高，则说明可能是由于之前运行数据库系统的主机负载过大导致查询低效。此外，还可能是由于执行计划比较低效，但是由于主机硬件较快使得查询效率较高。

步骤4 检查重复相同查询语句的执行效率。

查询效率低的一个重要原因是查询所需信息没有缓存在内存中，这可能是由于内存资源紧张，缓存信息被其他查询处理覆盖。

重复执行相同的查询语句，如果后续执行的查询语句效率提升，则可能是由于上述原因导致。

----结束

14.5.2 执行 SELECT 查询时，提示 failed to find conversion function from unknown to text

问题描述

执行select语句时，报错

failed to find conversion function from unknown to text

```
ERROR:  failed to find conversion function from unknown to text
```

原因分析

在select的语句的子查询中包含常量，但是没有指定常量类型。

处理方法

将子查询中的常量类型强转为一个固定的类型。

例：

```
create table t1(col1 int, col2 int);

SELECT distinct Q.*
   FROM (SELECT col1,
         '2015-03-01' AS start_time,
         '2015-03-01' AS last_time
        FROM t1)Q;
```

可以修改为：

```
SELECT distinct Q.*
   FROM (SELECT col1,
         cast('2015-03-01' as text) AS start_time,
         cast('2015-03-01' as text) AS last_time
        FROM t1)Q;
```

14.5.3 DROP TABLE 失败

问题现象

DROP TABLE失败的两种现象：

- 在gsql客户端下使用\dt+命令查看数据库中无表table_name；CREATE TABLE table_name时报table_name已经存在的错误，使用DROP TABLE table_name失败，报不存在该表的错误，导致无法再次创建table_name表。
- 在gsql客户端下使用\dt+命令查看数据库中有表table_name；使用DROP TABLE table_name失败，报不存在该表的错误，导致无法再次创建table_name表。

原因分析

导致该错误的原因有的节点上有该张表，有的节点上无该张表。

处理方法

当遇到上面两种现象时，使用DROP TABLE table_name失败，请再次使用DROP TABLE if exists table_name命令，使得DROP表可以成功。

14.5.4 不同用户查询同表显示数据不同

问题现象

2个用户登录相同数据库human_resource，分别执行的查询语句如下：select count(*) from areas，查询同一张表areas时，查询结果却不一致。

原因分析

请先判断同名的表是否确实是同一张表。在关系型数据库中，确定一张表通常需要3个因素：database，schema，table。从问题现象描述看，database，table已经确定，

分别是human_resource、areas。接着，需要检查schema。使用dbadmin，user01分别登录发现，search_path依次是public和"$user"。dbadmin作为集群管理员，默认不会创建dbadmin同名的schema，即不指定schema的情况下所有表都会建在public下。而对于普通用户如user01，则会在创建用户时，默认创建同名的schema，即不指定schema时表都会创建在user01的schema下。最终确定该案例发生时，确实因为2个用户之间交错对表进行操作，导致了同名不同表的情况。

处理方法

在操作表时加上schema引用，格式：schema.table。

14.5.5 业务运行时整数转换错误

问题现象

在转换整数时报出以下错误：

```
Invalid input syntax for integer: "13."
```

原因分析

有些数据类型不能转换成目标数据类型。

处理办法

逐步缩小SQL范围来确定。

14.5.6 SQL 语句出错自动重试

GaussDB(DWS)支持在SQL语句执行出错时的自动重试功能（下文简称CN Retry）。对于来自gsql客户端、JDBC、ODBC驱动的SQL语句，在SQL语句执行失败时，CN端能够自动识别语句执行过程中的报错，并重新下发任务进行自动重试。

该功能的限制和约束如下：

- 功能范围限制：
  - 仅能提高故障发生时SQL语句执行成功率，不能保证100%的执行成功。
  - CN Retry默认开启，开启后temp表会记录日志，关闭CN Retry后，temp表即不会记录日志，因此不能在使用temp表时反复打开/关闭CN Retry开关，否则主备切换后再CN Retry会造成数据不一致。
  - CN Retry默认开启，开启后新创建的unlogged表会忽视unlogged关键字，建成普通表。关闭CN Retry后，unlogged表不会记录日志，因此不能在使用unlogged表时反复打开/关闭CN Retry开关，否则主备切换后再CN Retry会造成数据不一致。
  - 在使用gds进行数据导出时，支持CN Retry。现有机制导出时会对重复文件进行检测并删除相同的文件，因此建议不要对相同的外表重复导出数据，除非确定数据目录中相同文件名的文件需要删除。
- 错误类型约束：

  SQL语句出错时能够被识别和重试的错误，仅限在错误类型列表（请参考**表14-3**）中定义的错误。
- 语句类型约束：

  支持单语句CN Retry、存储过程、函数、匿名块。不支持事务块中的语句。

- 存储过程语句约束：
  - 包含EXCEPTION的存储过程，如果在执行过程中（包含语句块执行和EXCEPTION中的语句执行）错误被抛出，可以retry，如果报错被EXCEPTION捕获则不能retry。
  - 不支持使用全局变量的package。
  - 不支持DBMS_JOB。
  - 不支持UTL_FILE。
- 集群状态约束：
  - 仅支持DN、GTM实例故障。
  - CN Retry有次数限制，如果在CN Retry达到最大尝试次数（最大次数由**max_query_retry_times**控制）之前，集群状态无法从故障状态恢复到正常状态，CN Retry不能保证执行成功。
  - 扩容时不支持CN Retry。
- 数据导入约束：
  - 不支持COPY FROM STDIN语句。
  - 不支持gsql \copy from元命令。
  - 不支持JDBC CopyManager copyIn导入数据。

CN Retry支持的错误类型列表和对应的错误码信息见**表14-3**，可以通过GUC参数**retry_ecode_list**设置CN Retry支持的错误类型列表，但不建议用户直接修改该参数，如有修改需求请联系技术工程师协助处理。

表 14-3 CN Retry 支持的错误类型列表

错误类型	错误码	备注
对端连接重置（CONNECTION_RESET_BY_PEER）	YY001	TCP通信错误：Connection reset by peer（CN和DN间通信）
对端流重置（STREAM_CONNECTION_RESET_BY_PEER）	YY002	TCP通信错误：Stream connection reset by peer（DN和DN间通信）
锁等待超时（LOCK_WAIT_TIMEOUT）	YY003	锁超时，Lock wait timeout
连接超时（CONNECTION_TIMED_OUT）	YY004	TCP通信错误，Connection timed out
查询设置错误（SET_QUERY_ERROR）	YY005	SET命令发送失败，Set query
超出逻辑内存（OUT_OF_LOGICAL_MEMORY）	YY006	内存申请失败，Out of logical memory
通信库内存分配（SCTP_MEMORY_ALLOC）	YY007	SCTP通信错误，Memory allocate error
无通信库缓存数据（SCTP_NO_DATA_IN_BUFFER）	YY008	SCTP通信错误，SCTP no data in buffer

错误类型	错误码	备注
通信库释放内存关闭（SCTP_RELEASE_MEMORY_CLOSE）	YY009	SCTP通信错误，Release memory close
SCTP、TCP断开（SCTP_TCP_DISCONNECT）	YY010	SCTP通信错误，TCP disconnect
通信库断开（SCTP_DISCONNECT）	YY011	SCTP通信错误，SCTP disconnect
通信库远程关闭（SCTP_REMOTE_CLOSE）	YY012	SCTP通信错误，Stream closed by remote
等待未知通信库通信（SCTP_WAIT_POLL_UNKNOW）	YY013	等待未知通信库通信，SCTP wait poll unknow
无效快照（SNAPSHOT_INVALID）	YY014	快照非法，Snapshot invalid
通讯接收信息错误（ERRCODE_CONNECTION_RECEIVE_WRONG）	YY015	连接获取错误，Connection receive wrong
内存耗尽（OUT_OF_MEMORY）	53200	内存耗尽，Out of memory
连接失败（CONNECTION_FAILURE）	08006	GTM出错，Connection failure
连接异常（CONNECTION_EXCEPTION）	08000	连接出现错误，和DN的通讯失败，Connection exception
管理员关闭系统（ADMIN_SHUTDOWN）	57P01	管理员关闭系统，Admin shutdown
关闭远程流接口（STREAM_REMOTE_CLOSE_SOCKET）	XX003	关闭远程套接字，Stream remote close socket
重复查询编号（ERRCODE_STREAM_DUPLICATE_QUERY_ID）	XX009	重复查询，Duplicate query id
stream查询并发更新同一行（ERRCODE_STREAM_CONCURRENT_UPDATE）	YY016	stream查询并发更新同一行，Stream concurrent update
LLVM内存分配错误（ERRCODE_LLVM_BAD_ALLOC_ERROR ）	CG003	内存分配错误， Allocate error
LLVM致命错误（ERRCODE_LLVM_FATAL_ERROR）	CG004	致命错误，Fatal error

开启CN Retry功能需要设置如下GUC参数：

- 必选的GUC参数（CN和DN都需设置）

  max_query_retry_times，具体的参数信息请参考**max_query_retry_times**。

> **⚠ 注意**
>
> CN Retry功能开启时会为临时表数据记录日志，为保证数据一致性，在使用临时表时不能切换CN Retry开关状态（参考**max_query_retry_times**），保持使用临时表的会话中CN Retry开关始终处于打开状态或者关闭状态。

15 存储过程

15.1 数据类型

数据类型是一组值的集合以及定义在这个值集上的一组操作。GaussDB(DWS)数据库是由表的集合组成的，而各表中的列定义了该表，每一列都属于一种数据类型，GaussDB(DWS)根据数据类型有相应函数对其内容进行操作，例如GaussDB(DWS)可对数值型数据进行加、减、乘、除操作。

15.2 数据类型转换

数据库中允许有些数据类型进行隐式类型转换（赋值、函数调用的参数等），有些数据类型间不允许进行隐式数据类型转换，可尝试使用GaussDB(DWS)提供的类型转换函数，例如CAST进行数据类型强转。

GaussDB(DWS)数据库常见的隐式类型转换，请参见表15-1。

须知

GaussDB(DWS)支持的DATE的效限范围是：公元前4713年到公元294276年。

表 15-1 隐式类型转换表

原始数据类型	目标数据类型	备注
CHAR	VARCHAR2	-
CHAR	NUMBER	原数据必须由数字组成。
CHAR	DATE	原数据不能超出合法日期范围。
CHAR	RAW	-
CHAR	CLOB	-

原始数据类型	目标数据类型	备注
VARCHAR2	CHAR	-
VARCHAR2	NUMBER	原数据必须由数字组成。
VARCHAR2	DATE	原数据不能超出合法日期范围。
VARCHAR2	CLOB	-
NUMBER	CHAR	-
NUMBER	VARCHAR2	-
DATE	CHAR	-
DATE	VARCHAR2	-
RAW	CHAR	-
RAW	VARCHAR2	-
CLOB	CHAR	-
CLOB	VARCHAR2	-
CLOB	NUMBER	原数据必须由数字组成。
INT4	CHAR	-

15.3 数组

数组类型的使用

在使用数组之前，需要自定义一个数组类型。

在存储过程中紧跟AS关键字后面定义数组类型。定义方法为：

```
TYPE array_type IS VARRAY(size) OF data_type;
```

其中：

- array_type：要定义的数组类型名。
- VARRAY：表示要定义的数组类型。
- size：取值为正整数，表示可以容纳的成员的最大数量。
- data_type：要创建的数组中成员的类型。

说明

- 在GaussDB(DWS)中，数组会自动增长，访问越界会返回一个NULL，不会报错。
- 在存储过程中定义的数组类型，其作用域仅在该存储过程中。
- 建议选择上述定义方法的一种来自定义数组类型，当同时使用两种方法定义同名的数组类型时，GaussDB(DWS)会优先选择存储过程中定义的数组类型来声明数组变量。

GaussDB(DWS)支持使用圆括号来访问数组元素，且还支持一些特有的函数，如extend，count，first，last来访问数组的内容。

说明

存储过程中如果有DML语句（SELECT、UPDATE、INSERT、DELETE），DML语句只能使用中括号来访问数组元素，这样可以和函数表达式区分开。

示例

```
--演示在存储过程中对数组进行操作。
CREATE OR REPLACE PROCEDURE array_proc
AS
    TYPE ARRAY_INTEGER IS VARRAY(1024) OF INTEGER;--定义数组类型
    ARRINT ARRAY_INTEGER := ARRAY_INTEGER();  --声明数组类型的变量
BEGIN
    ARRINT.extend(10);
    FOR I IN 1..10 LOOP
        ARRINT(I) := I;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(ARRINT.COUNT);
    DBMS_OUTPUT.PUT_LINE(ARRINT(1));
    DBMS_OUTPUT.PUT_LINE(ARRINT(10));
    DBMS_OUTPUT.PUT_LINE(ARRINT(ARRINT.FIRST));
    DBMS_OUTPUT.PUT_LINE(ARRINT(ARRINT.last));
END;
/

--调用该存储过程。
CALL array_proc();

--删除存储过程。
DROP PROCEDURE array_proc;
```

15.4 record

record 类型的变量

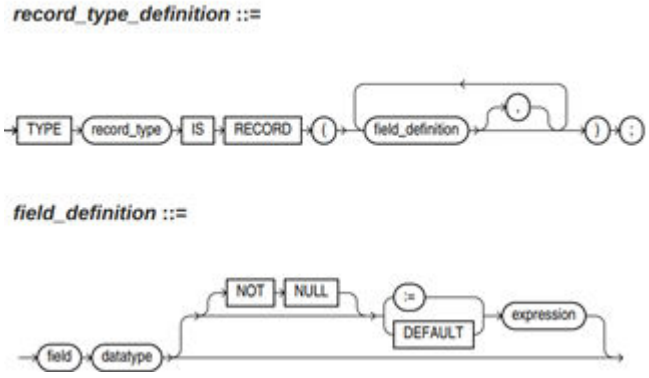
创建一个record变量的方式：

定义一个record类型 ，然后使用该类型来声明一个变量。

语法

record类型的语法参见图15-1。

图 15-1 record 类型的语法



对以上语法格式的解释如下：

- record_type：声明的类型名称。
- field：record类型中的成员名称。
- datatype：record类型中成员的类型。
- expression：设置默认值的表达式。

说明

在GaussDB(DWS)中：

- record类型的变量的赋值支持，
  - 在函数或存储过程的声明阶段，声明一个record类型，并且可以在该类型中定义成员变量。
  - 一个record变量到另一个record变量的赋值。
  - SELECT INTO和FETCH向一个record类型的变量中赋值。
  - 将一个NULL值赋值给一个record变量。
- 不支持INSERT和UPDATE语句使用record变量进行插入数据和更新数据。
- 如果成员有复合类型，在声明阶段不支持指定默认值，该行为同声明阶段的变量一样。

示例

```
下面存储过程中用到的表定义如下：
\d emp_rec
                Table "public.emp_rec"
  Column  |              Type              | Modifiers
----------+--------------------------------+-----------
 empno    | numeric(4,0)                   | not null
 ename    | character varying(10)          |
 job      | character varying(9)           |
 mgr      | numeric(4,0)                   |
 hiredate | timestamp(0) without time zone |
 sal      | numeric(7,2)                   |
 comm     | numeric(7,2)                   |
 deptno   | numeric(2,0)                   |

--演示在存储过程中对数组进行操作。
CREATE OR REPLACE FUNCTION regress_record(p_w VARCHAR2)
RETURNS
VARCHAR2  AS $$
DECLARE

  --声明一个record类型.
  type rec_type is record (name  varchar2(100), epno int);
  employer rec_type;

  --使用%type声明record类型
  type rec_type1 is record (name  emp_rec.ename%type, epno int not null :=10);
  employer1 rec_type1;

  --声明带有默认值的record类型
  type rec_type2 is record (
        name varchar2 not null := 'SCOTT',
        epno int not null :=10);
   employer2 rec_type2;
   CURSOR C1 IS  select ename,empno from emp_rec order by 1 limit 1;

BEGIN
     --对一个record类型的变量的成员赋值。
    employer.name := 'WARD';
    employer.epno = 18;
    raise info 'employer name: % , epno:%', employer.name, employer.epno;

    --将一个record类型的变量赋值给另一个变量。
```

```
    employer1 := employer;
    raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;

     --将一个record类型变量赋值为NULL。
    employer1 := NULL;
    raise info 'employer1 name: % , epno: %',employer1.name, employer1.epno;

     --获取record变量的默认值。
    raise info 'employer2 name: % ,epno: %', employer2.name, employer2.epno;

     --在for循环中使用record变量
     for employer in select ename,empno from emp_rec order by 1  limit 1
         loop
             raise info 'employer name: % , epno: %', employer.name, employer.epno;
         end loop;

     --在select into 中使用record变量。
     select ename,empno  into employer2 from emp_rec order by 1 limit 1;
     raise info 'employer name: % , epno: %', employer2.name, employer2.epno;

     --在cursor中使用record变量。
     OPEN C1;
     FETCH C1 INTO employer2;
     raise info 'employer name: % , epno: %', employer2.name, employer2.epno;
     CLOSE C1;
     RETURN employer.name;
END;
$$
LANGUAGE plpgsql;

--调用该存储过程。
CALL regress_record('abc');

--删除存储过程。
DROP PROCEDURE regress_record;
```

15.5 声明语法

15.5.1 基本结构

结构

PL/SQL块中可以包含子块，子块可以位于PL/SQL中任何部分。PL/SQL块的结构如下：

- 声明部分：声明PL/SQL用到的变量，类型及游标，以及局部的存储过程和函数。
  ```
  DECLARE
  ```

  说明

  不涉及变量声明时声明部分可以没有。

  - 对匿名块来说，没有变量声明部分时，可以省去DECLARE关键字。
  - 对存储过程来说，没有DECLARE， AS相当于DECLARE。即便没有变量声明的部分，关键字AS也必须保留。
- 执行部分：过程及SQL语句，程序的主要部分。必选。
  ```
  BEGIN
  ```
- 执行异常部分：错误处理。可选。
  ```
  EXCEPTION
  ```
- 结束

```
END;
/
```

须知

禁止在PL/SQL块中使用连续的Tab，连续的Tab可能会造成在使用gsql工具带“-r”参数执行PL/SQL块时出现异常。

分类

PL/SQL块可以分为以下几类：

- 匿名块：动态构造，只能执行一次。语法请参考图15-2。
- 子程序：存储在数据库中的存储过程、函数和操作符及高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

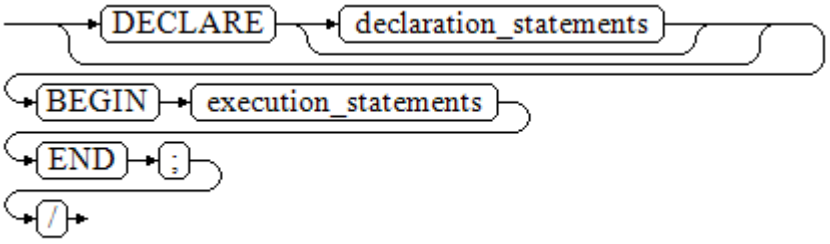
15.5.2 匿名块

匿名块（Anonymous Block）一般用于不频繁执行的脚本或不重复进行的活动。它们在一个会话中执行，并不被存储。

语法

匿名块的语法参见图15-2。

图 15-2 anonymous_block::=



对以上语法图的解释如下：

- 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。输入“/”按回车执行它。

须知

最后的结束符“/”必须独占一行，不能直接跟在END后面。

- 声明部分包括变量定义、类型、游标定义等。
- 最简单的匿名块不执行任何命令。但一定要在任意实施块里至少有一个语句，甚至是一个NULL语句。

示例

下面列举了基本的匿名块程序：

```
--空语句块
BEGIN
    NULL;
END;
/

--将信息打印到控制台：
BEGIN
    dbms_output.put_line('hello world!');
END;
/

--将变量内容打印到控制台：
DECLARE
    my_var VARCHAR2(30);
BEGIN
    my_var :='world';
    dbms_output.put_line('hello'||my_var);
END;
/
```

15.5.3 子程序

存储在数据库中的存储过程、函数和操作符及高级包等。当在数据库上建立好后，可以在其他程序中调用它们。

15.6 基本语句

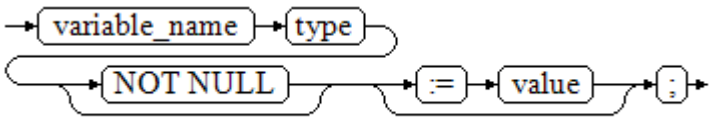
15.6.1 定义变量

介绍PL/SQL中变量的声明，以及该变量在代码中的作用域。

变量声明

变量声明语法请参见图15-3。

图 15-3 declare_variable::=



对以上语法格式的解释如下：

- variable_name，为变量名。
- type，为变量类型。
- value，是该变量的初始值（如果不给定初始值，则初始为NULL）。value也可以是表达式。

示例

```
DECLARE
    emp_id  INTEGER := 7788;  --定义变量并赋值
BEGIN
    emp_id := 5*7784;  --变量赋值
END;
/
```

变量类型除了支持基本类型，还可是使用%TYPE和%ROWTYPE去声明一些与其他表字段或表结构本身相关的变量。

%TYPE 属性

%TYPE主要用于声明某个与其他变量类型（例如，表中某列的类型）相同的变量。假如我们想定义一个my_name变量，它的变量类型与employee的firstname类型相同，我们可以通过如下定义：

```
my_name employee.firstname%TYPE
```

这样定义可以带来两个好处，首先，我们不用预先知道employee 表的firstname类型具体是什么。其次，即使之后firstname类型有了变化，我们也不需要再次修改my_name的类型。

%ROWTYPE 属性

%ROWTYPE属性主要用于对一组数据的类型声明，用于存储表中的一行数据，或从游标匹配的结果。假如，我们需要一组数据，该组数据的字段名称与字段类型都与employee表相同。我们可以通过如下定义：

```
my_employee employee%ROWTYPE
```

须知

多个CN的环境下，存储过程中无法声明临时表的%ROWTYPE及%TYPE属性。因为临时表仅在当前session有效，在编译阶段其他CN无法看到当前CN的临时表。故多个CN的环境下，会提示该临时表不存在。

变量作用域

变量的作用域表示变量在代码块中的可访问性和可用性。只有在它的作用域内，变量才有效。

- 变量必须在declare部分声明，即必须建立BEGIN-END块。块结构也强制变量必须先声明后使用，即变量在过程内有不同作用域、不同的生存期。
- 同一变量可以在不同的作用域内定义多次，内层的定义会覆盖外层的定义。
- 在外部块定义的变量，可以在嵌套块中使用。但外部块不能访问嵌套块中的变量。

示例

```
DECLARE
    emp_id  INTEGER :=7788;  --定义变量并赋值
    outer_var  INTEGER :=6688;  --定义变量并赋值
BEGIN
    DECLARE
        emp_id INTEGER :=7799;  --定义变量并赋值
        inner_var  INTEGER :=6688;  --定义变量并赋值
```

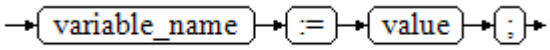
```
    BEGIN
        dbms_output.put_line('inner emp_id ='||emp_id);  --显示值为7799
        dbms_output.put_line('outer_var ='||outer_var);  --引用外部块的变量
    END;
    dbms_output.put_line('outer emp_id ='||emp_id);  --显示值为7788
END;
/
```

15.6.2 赋值语句

语法

给变量赋值的语法请参见图15-4。

图 15-4 assignment_value::=



对以上语法格式的解释如下：

- variable_name，为变量名。
- value，可以是值或表达式。值value的类型需要和变量variable_name的类型兼容才能正确赋值。

示例

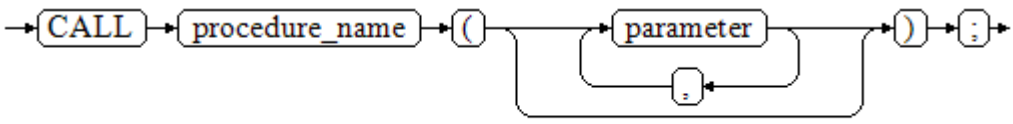
```
DECLARE
    emp_id  INTEGER := 7788;--赋值
BEGIN
    emp_id := 5;--赋值
    emp_id := 5*7784;
END;
/
```

15.6.3 调用语句

语法

调用一个语句的语法请参见图15-5。

图 15-5 call_clause::=



对以上语法格式的解释如下：

- procedure_name，为存储过程名。
- parameter，为存储过程的参数，可以没有或者有多个参数。

示例

```
--创建存储过程proc_staffs
CREATE OR REPLACE PROCEDURE proc_staffs
(
section     NUMBER(6),
salary_sum out NUMBER(8,2),
staffs_count out INTEGER
)
IS
BEGIN
SELECT sum(salary), count(*) INTO salary_sum, staffs_count FROM hr.staffs where section_id = section;
END;
/

--创建存储过程proc_return.
CREATE OR REPLACE PROCEDURE proc_return
AS
v_num NUMBER(8,2);
v_sum INTEGER;
BEGIN
proc_staffs(30, v_sum, v_num);  --调用语句
dbms_output.put_line(v_sum||'#'||v_num);
RETURN;  --返回语句
END;
/

--调用存储过程proc_return.
CALL proc_return();

--清除存储过程
DROP PROCEDURE proc_staffs;
DROP PROCEDURE proc_return;

--创建函数func_return.
CREATE OR REPLACE FUNCTION func_return returns void
language plpgsql
AS $$
DECLARE
v_num INTEGER := 1;
BEGIN
dbms_output.put_line(v_num);
RETURN;  --返回语句
END $$;


-- 调用函数func_return
 CALL func_return();

-- 清除函数
 DROP FUNCTION func_return;
```

15.6.4 返回语句

语法

返回语句的语法请参见图15-6。

图 15-6 return_clause::=

对以上语法的解释如下：

用于将控制从存储过程返回给调用者。

示例

请参见调用语句的**示例**。

存储函数的语法

RETURN NEXT和RETURN QUERY适用于函数，不适用存储过程。

在创建函数时需要指定返回值SETOF datatype。

return_next_clause::=

return_query_clause::=

对以上语法的解释如下：

当需要返回值是一个集合时，使用RETURN NEXT或者RETURN QUERY向结果集追加结果。然后继续执行函数的下一条语句。随着后续的RETURN NEXT或RETURN QUERY命令的执行，结果集中会有多个结果。函数执行完成后会一起返回。

RETURN NEXT 可以用用于标量和符合数据类型。

RETURN QUREY将执行一个查询，并将查询结果追加到函数的结果集中。RETURN QUERY 有一种变体 RETURN QUERY EXECUTE。 后面还可以增加动态查询。通过USING向查询插入参数。

存数函数的示例

```
CREATE TABLE t1(a int);
INSERT INTO t1 VALUES(1),(10);

--RETURN NEXT
CREATE OR REPLACE FUNCTION fun_for_return_next() RETURNS SETOF t1 AS $$
DECLARE
  r t1%ROWTYPE;
BEGIN
  FOR r IN select * from t1
  LOOP
    RETURN NEXT r;
  END LOOP;
  RETURN;
END;
$$ LANGUAGE PLPGSQL;
call fun_for_return_next();
 a
---
 1
 10
(2 rows)

-- RETURN QUERY
CREATE OR REPLACE FUNCTION fun_for_return_query() RETURNS SETOF t1 AS $$
DECLARE
```

```
  r t1%ROWTYPE;
BEGIN
  RETURN QUERY select * from t1;
END;
$$
language plpgsql;
call fun_for_return_next();
 a
---
 1
 10
(2 rows)
```

15.7 动态语句

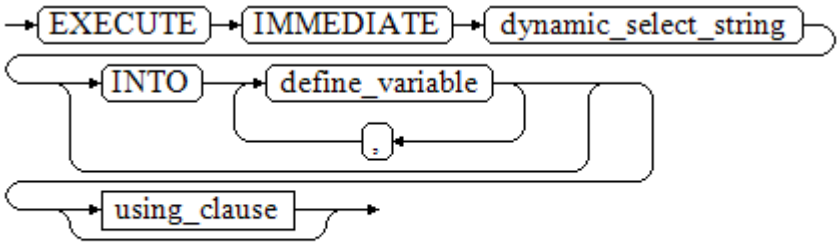
15.7.1 执行动态查询语句

介绍执行动态查询语句。GaussDB(DWS)提供两种方式：使用EXECUTE IMMEDIATE、OPEN FOR实现动态查询。前者通过动态执行SELECT语句，后者结合了游标的使用。当需要将查询的结果保存在一个数据集用于提取时，可使用OPEN FOR实现动态查询。

EXECUTE IMMEDIATE

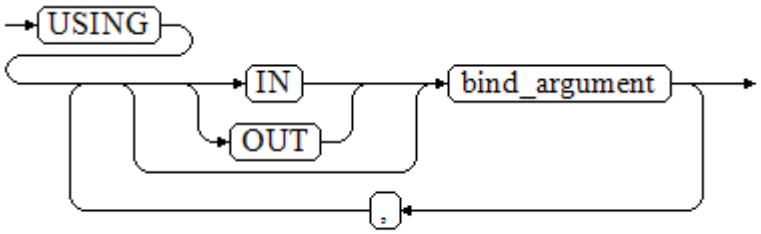
语法图请参见图15-7。

图 15-7 EXECUTE IMMEDIATE dynamic_select_clause::=



using_clause子句的语法图参见图15-8。

图 15-8 using_clause-1



对以上语法格式的解释如下：

- define_variable，用于指定存放单行查询结果的变量。
- USING IN bind_argument，用于指定存放传递给动态SQL值的变量，即在dynamic_select_string中存在占位符时使用。
- USING OUT bind_argument，用于指定存放动态SQL返回值的变量。

须知

- 查询语句中，into和out不能同时存在；
- 占位符命名以“:”开始，后面可跟数字、字符或字符串，与USING子句的bind_argument一一对应；
- bind_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象，即不支持使用bind_argument为动态SQL语句传递模式对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic_select_clause；
- 动态PL/SQL块允许出现重复的占位符，即相同占位符只能与USING子句的一个bind_argument按位置对应。

示例

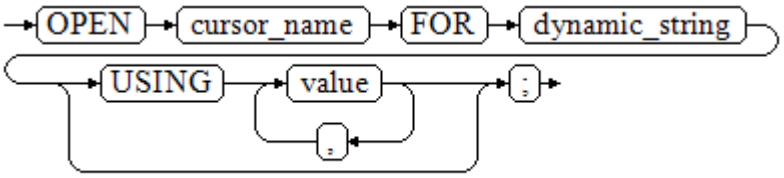
```
--从动态语句检索值（INTO 子句）：
DECLARE
   staff_count  VARCHAR2(20);
BEGIN
   EXECUTE IMMEDIATE 'select count(*) from hr.staffs'
      INTO staff_count;
   dbms_output.put_line(staff_count);
END;
/

--传递并检索值（INTO子句用在USING子句前）：
CREATE OR REPLACE PROCEDURE dynamic_proc
AS
   staff_id     NUMBER(6) := 200;
   first_name   VARCHAR2(20);
   salary       NUMBER(8,2);
BEGIN
   EXECUTE IMMEDIATE 'select first_name, salary from hr.staffs where staff_id = :1'
      INTO first_name, salary
      USING IN staff_id;
   dbms_output.put_line(first_name || ' ' || salary);
END;
/

--调用存储过程
CALL dynamic_proc();

--删除存储过程
DROP PROCEDURE dynamic_proc;
```

OPEN FOR

动态查询语句还可以使用OPEN FOR打开动态游标来执行。

语法参见图15-9。

图 15-9 open_for::=



参数说明：

- cursor_name：要打开的游标名。
- dynamic_string：动态查询语句。
- USING value：在dynamic_string中存在占位符时使用。

游标的使用请参考**游标概述**。

示例

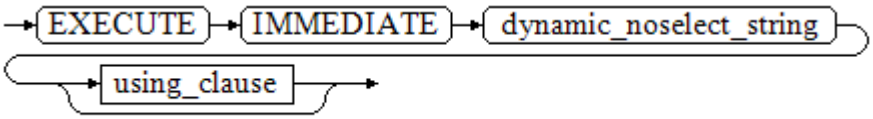
```
DECLARE
   name          VARCHAR2(20);
   phone_number  VARCHAR2(20);
   salary        NUMBER(8,2);
   sqlstr        VARCHAR2(1024);

   TYPE app_ref_cur_type IS REF CURSOR;  --定义游标类型
   my_cur app_ref_cur_type;  --定义游标变量

BEGIN
   sqlstr := 'select first_name,phone_number,salary from hr.staffs
       where section_id = :1';
   OPEN my_cur FOR sqlstr USING '30';  --打开游标, using是可选的
   FETCH my_cur INTO name, phone_number, salary; --获取数据
   WHILE my_cur%FOUND LOOP
         dbms_output.put_line(name||'#'||phone_number||'#'||salary);
         FETCH my_cur INTO name, phone_number, salary;
   END LOOP;
   CLOSE my_cur;   --关闭游标
END;
/
```

15.7.2 执行动态非查询语句

语法

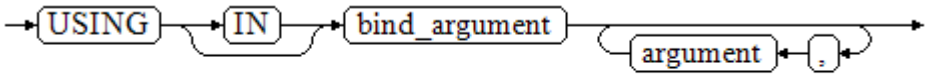
语法请参见**图15-10**。

图 15-10 noselect::=



using_clause子句的语法参见**图15-11**。

图 15-11 using_clause-2



对以上语法格式的解释如下：

USING IN bind_argument用于指定存放传递给动态SQL值的变量，在dynamic_noselect_string中存在占位符时使用，即动态SQL语句执行时，bind_argument将替换相对应的占位符。要注意的是，bind_argument只能是值、变量或表达式，不能是表名、列名、数据类型等数据库对象。如果存储过程需要通过声明参数传递数据库对象来构造动态SQL语句（常见于执行DDL语句时），建议采用连接运算符“||”拼接dynamic_select_clause。另外，动态语句允许出现重复的占位符，相同占位符只能与唯一一个bind_argument按位置一一对应。

示例

```
--创建表
CREATE TABLE sections_t1
(
   section       NUMBER(4) ,
   section_name  VARCHAR2(30),
   manager_id    NUMBER(6),
   place_id      NUMBER(4)
)
DISTRIBUTE BY hash(manager_id);

--声明变量
DECLARE
   section       NUMBER(4) := 280;
   section_name  VARCHAR2(30) := 'Info support';
   manager_id    NUMBER(6) := 103;
   place_id      NUMBER(4) := 1400;
   new_colname   VARCHAR2(10) := 'sec_name';
BEGIN
--执行查询
    EXECUTE IMMEDIATE 'insert into sections_t1 values(:1, :2, :3, :4)'
       USING section, section_name, manager_id,place_id;
--执行查询（重复占位符）
    EXECUTE IMMEDIATE 'insert into sections_t1 values(:1, :2, :3, :1)'
       USING section, section_name, manager_id;
--执行ALTER语句（建议采用“||”拼接数据库对象构造DDL语句）
    EXECUTE IMMEDIATE 'alter table sections_t1 rename section_name to ' || new_colname;
END;
/

--查询数据
SELECT * FROM sections_t1;

--删除表
DROP TABLE sections_t1;
```

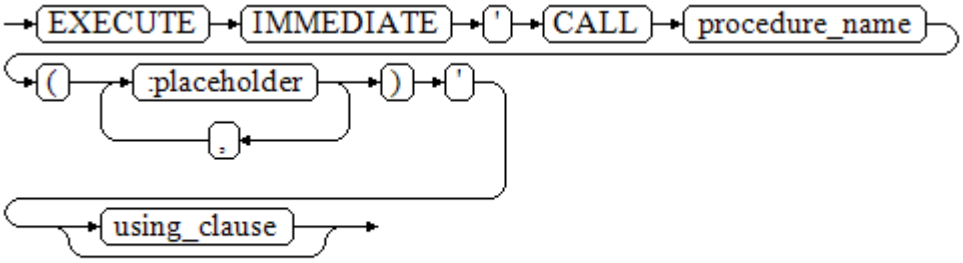
15.7.3 动态调用存储过程

动态调用存储过程必须使用匿名的语句块将存储过程或语句块包在里面，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

语法

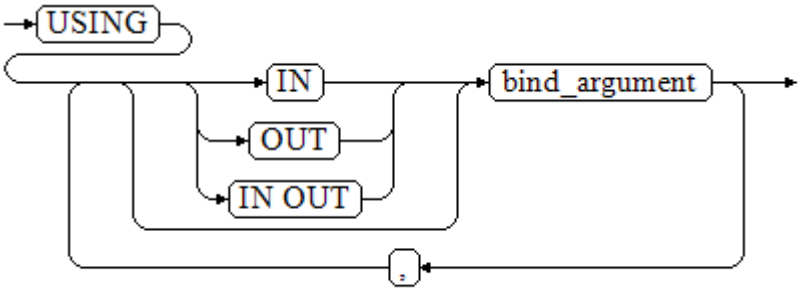
语法请参见图15-12。

图 15-12 call_procedure::=



using_clause子句的语法参见图15-13。

图 15-13 using_clause-3



对以上语法格式的解释如下：

- CALL procedure_name，调用存储过程。
- [:placeholder1，:placeholder2，…]，存储过程参数占位符列表。占位符个数与参数个数相同。
- USING [IN|OUT|IN OUT] bind_argument，用于指定存放传递给存储过程参数值的变量。bind_argument前的修饰符与对应参数的修饰符一致。

示例

```
--创建存储过程proc_add。
CREATE OR REPLACE PROCEDURE proc_add
(
    param1    in   INTEGER,
    param2    out  INTEGER,
    param3    in   INTEGER
)
AS
BEGIN
   param2:= param1 + param3;
END;
/

DECLARE
    input1 INTEGER:=1;
    input2 INTEGER:=2;
    statement  VARCHAR2(200);
    param2     INTEGER;
BEGIN
  --声明调用语句
```

```
    statement := 'call proc_add(:col_1, :col_2, :col_3)';
    --执行语句
    EXECUTE IMMEDIATE statement
        USING IN input1, OUT param2, IN input2;
    dbms_output.put_line('result is: '||to_char(param2));
END;
/

--删除存储过程
DROP PROCEDURE proc_add;
```

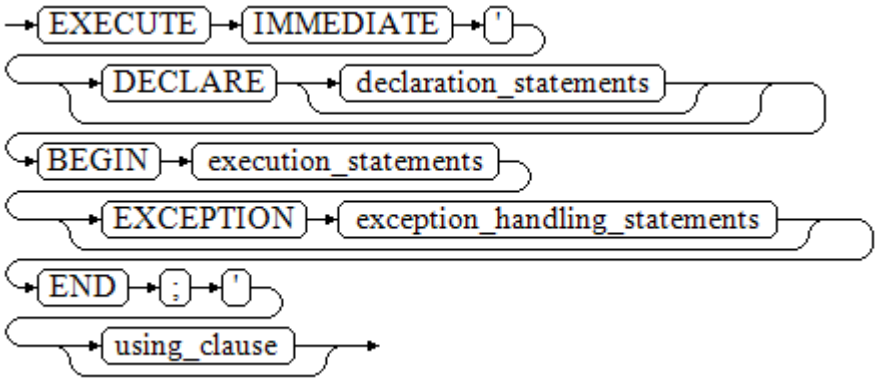
15.7.4 动态调用匿名块

动态调用匿名块是指在动态语句中执行匿名块，使用EXECUTE IMMEDIATE…USING语句后面带IN、OUT来输入、输出参数。

语法

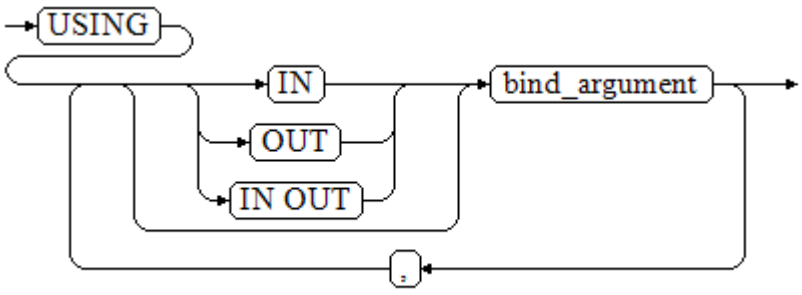
语法请参见图15-14。

图 15-14 call_anonymous_block::=



using_clause子句的语法参见图15-15。

图 15-15 using_clause-4



对以上语法格式的解释如下：

- 匿名块程序实施部分，以BEGIN语句开始，以END语句停顿，以一个分号结束。
- USING [IN|OUT|IN OUT] bind_argument，用于指定存放传递给存储过程参数值的变量。bind_argument前的修饰符与对应参数的修饰符一致。

- 匿名块中间的输入输出参数使用占位符来指明，要求占位符个数与参数个数相同，并且占位符所对应参数的顺序和USING中参数的顺序一致。
- 目前GaussDB(DWS)在动态语句调用匿名块时，EXCEPTION语句中暂不支持使用占位符进行输入输出参数的传递。

示例

```
--创建存储过程dynamic_proc
CREATE OR REPLACE PROCEDURE dynamic_proc
AS
   staff_id     NUMBER(6) := 200;
   first_name   VARCHAR2(20);
   salary       NUMBER(8,2);
BEGIN
    --执行匿名块
    EXECUTE IMMEDIATE 'begin select first_name, salary into :first_name, :salary from hr.staffs where
staff_id= :dno; end;'
       USING OUT first_name, OUT salary, IN staff_id;
   dbms_output.put_line(first_name|| ' ' || salary);
END;
/

--调用存储过程
CALL dynamic_proc();

--删除存储过程
DROP PROCEDURE dynamic_proc;
```

15.8 控制语句

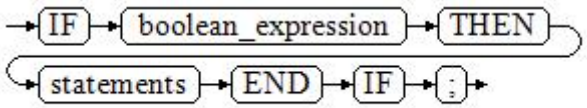
15.8.1 条件语句

条件语句的主要作用判断参数或者语句是否满足已给定的条件，根据判定结果执行相应的操作。

GaussDB(DWS)有五种形式的IF：

- IF_THEN

  图 15-16 IF_THEN::=

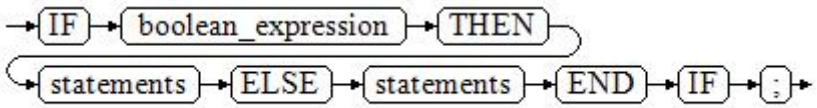
  IF_THEN语句是IF的最简单形式。如果条件为真，statements将被执行。否则，将忽略它们的结果使该IF_THEN语句执行结束。

  示例

  ```
  IF v_user_id <> 0 THEN
     UPDATE users SET email = v_email WHERE user_id = v_user_id;
  END IF;
  ```

- IF_THEN_ELSE

图 15-17 IF_THEN_ELSE::=



IF_THEN_ELSE语句增加了ELSE的分支，可以声明在条件为假的时候执行的语句。

示例

```
IF parentid IS NULL OR parentid = ''
THEN
    RETURN;
ELSE
    hp_true_filename(parentid);--表示调用存储过程
END IF;
```

- IF_THEN_ELSE IF

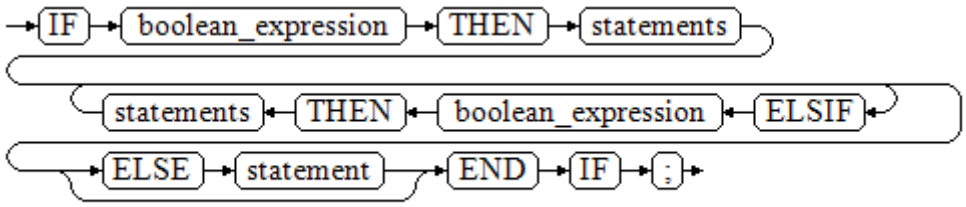
  IF语句可以嵌套，嵌套方式如下：

```
IF sex = 'm' THEN
    pretty_sex := 'man';
ELSE
    IF sex = 'f' THEN
        pretty_sex := 'woman';
    END IF;
END IF;
```

  这种形式实际上就是在一个IF语句的ELSE部分嵌套了另一个IF语句。因此需要一个END IF语句给每个嵌套的IF，另外还需要一个END IF语句结束父IF-ELSE。如果有多个选项，可使用下面的形式。

- IF_THEN_ELSIF_ELSE

  图 15-18 IF_THEN_ELSIF_ELSE::=



  示例

```
IF number_tmp = 0 THEN
    result := 'zero';
ELSIF number_tmp > 0 THEN
    result := 'positive';
ELSIF number_tmp < 0 THEN
    result := 'negative';
ELSE
    result := 'NULL';
END IF;
```

- IF_THEN_ELSEIF_ELSE

  ELSEIF是ELSIF的别名。

  综合示例

```
CREATE OR REPLACE PROCEDURE proc_control_structure(i in integer)
AS
    BEGIN
```

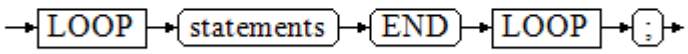
```
        IF i > 0 THEN
            raise info 'i:% is greater than 0. ',i;
        ELSIF i < 0 THEN
            raise info 'i:% is smaller than 0. ',i;
        ELSE
            raise info 'i:% is equal to 0. ',i;
        END IF;
        RETURN;
    END;
/

CALL proc_control_structure(3);

--删除存储过程
DROP PROCEDURE proc_control_structure;
```

15.8.2 循环语句

简单 LOOP 语句

语法图

图 15-19 loop::=



示例

```
CREATE OR REPLACE PROCEDURE proc_loop(i in integer, count out integer)
AS
    BEGIN
        count:=0;
        LOOP
        IF count > i THEN
            raise info 'count is %. ', count;
            EXIT;
        ELSE
            count:=count+1;
        END IF;
        END LOOP;
    END;
/

CALL proc_loop(10,5);
```

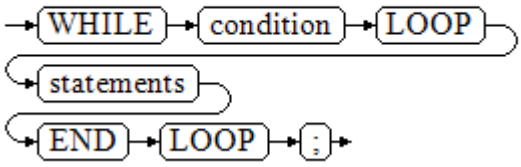
须知

该循环必须要结合EXIT使用，否则将陷入死循环。

WHILE_LOOP 语句

语法图

图 15-20 while_loop::=



只要条件表达式为真，WHILE语句就会不停的在一系列语句上进行循环，在每次进入循环体的时候进行条件判断。

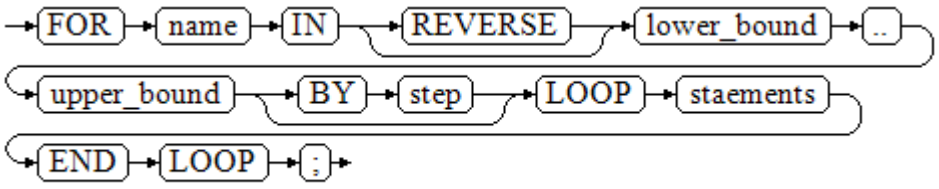
示例

```
CREATE TABLE integertable(c1 integer) DISTRIBUTE BY hash(c1);
CREATE OR REPLACE PROCEDURE proc_while_loop(maxval in integer)
AS
   DECLARE
   i int :=1;
   BEGIN
      WHILE i < maxval LOOP
         INSERT INTO integertable VALUES(i);
         i:=i+1;
      END LOOP;
   END;
/

--调用函数
CALL proc_while_loop(10);

--删除存储过程和表
DROP PROCEDURE proc_while_loop;
DROP TABLE integertable;
```

FOR_LOOP（integer 变量）语句

语法图

图 15-21 for_loop::=



说明

- 变量name会自动定义为integer类型并且只在此循环里存在。变量name介于lower_bound和upper_bound之间。
- 当使用REVERSE关键字时，lower_bound必须大于等于upper_bound，否则循环体不会被执行。

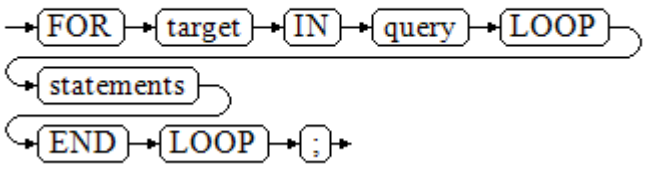
示例

```
--从0到5进行循环
CREATE OR REPLACE PROCEDURE proc_for_loop()
AS
    BEGIN
    FOR I IN 0..5 LOOP
        DBMS_OUTPUT.PUT_LINE('It is '||to_char(I) || ' time;') ;
    END LOOP;
END;
/

--调用函数
CALL proc_for_loop();

--删除存储过程
DROP PROCEDURE proc_for_loop;
```

FOR_LOOP 查询语句

语法图

图 15-22 for_loop_query::=



说明

变量target会自动定义，类型和query的查询结果的类型一致，并且只在此循环中有效。target的取值就是query的查询结果。

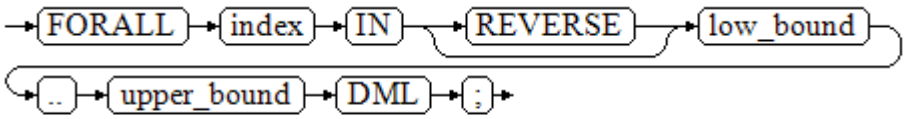
示例

```
--循环输出查询结果。
CREATE OR REPLACE PROCEDURE proc_for_loop_query()
AS
    record VARCHAR2(50);
BEGIN
    FOR record IN SELECT spcname FROM pg_tablespace LOOP
    dbms_output.put_line(record);
    END LOOP;
END;
/

--调用函数
CALL proc_for_loop_query();

--删除存储过程
DROP PROCEDURE proc_for_loop_query;
```

FORALL 批量查询语句

语法图

图 15-23 forall::=



说明

变量index会自动定义为integer类型并且只在此循环里存在。index的取值介于low_bound和upper_bound之间。

示例

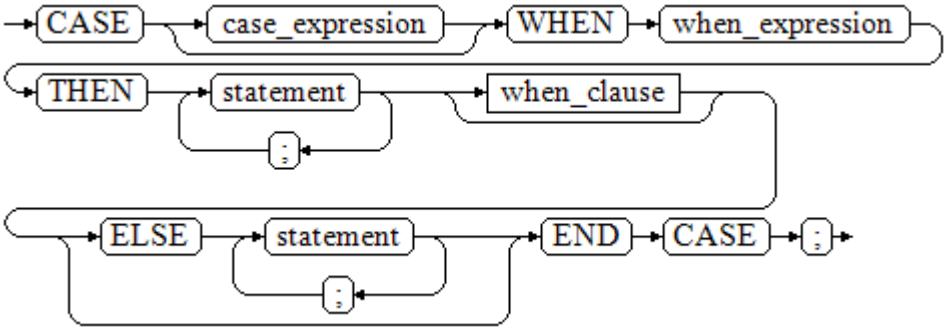
```
CREATE TABLE hdfs_t1 (
  title NUMBER(6),
  did VARCHAR2(20),
  data_period VARCHAR2(25),
  kind VARCHAR2(25),
  interval VARCHAR2(20),
  time DATE,
  isModified VARCHAR2(10)
)
DISTRIBUTE BY hash(did);

INSERT INTO hdfs_t1 VALUES( 8, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', to_date('21-06-1999',
'dd-mm-yyyy'), 'SH_CLERK' );

CREATE OR REPLACE PROCEDURE proc_forall()
AS
BEGIN
    FORALL i IN 100..120
        insert into hdfs_t1(title) values(i);
END;
/

--调用函数
CALL proc_forall();

--查询存储过程调用结果
SELECT * FROM hdfs_t1 WHERE title BETWEEN 100 AND 120;

--删除存储过程和表
DROP PROCEDURE proc_forall;
DROP TABLE hdfs_t1;
```

15.8.3 分支语句

语法

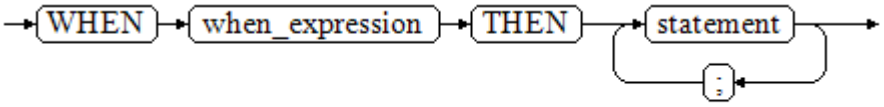
分支语句的语法请参见图15-24。

图 15-24 case_when::=



when_clause子句的语法图参见图15-25。

图 15-25 when_clause::=



参数说明：

- case_expression：变量或表达式。
- when_expression：常量或者条件表达式。
- statement：执行语句。

示例

```
CREATE OR REPLACE PROCEDURE proc_case_branch(pi_result in integer, pi_return out integer)
AS
   BEGIN
      CASE pi_result
         WHEN 1 THEN
            pi_return := 111;
         WHEN 2 THEN
            pi_return := 222;
         WHEN 3 THEN
            pi_return := 333;
         WHEN 6 THEN
            pi_return := 444;
         WHEN 7 THEN
            pi_return := 555;
         WHEN 8 THEN
            pi_return := 666;
         WHEN 9 THEN
            pi_return := 777;
         WHEN 10 THEN
            pi_return := 888;
         ELSE
            pi_return := 999;
      END CASE;
      raise info 'pi_return : %',pi_return ;
END;
/

CALL proc_case_branch(3,0);
```

```
--删除存储过程
DROP PROCEDURE proc_case_branch;
```

15.8.4 空语句

在PL/SQL程序中，可以用NULL语句来说明“不用做任何事情”，相当于一个占位符，可以使某些语句变得有意义，提高程序的可读性。

语法

空语句的用法如下：

```
DECLARE
   ...
BEGIN
   ...
   IF v_num IS NULL THEN
      NULL; -- 不需要处理任何数据。
   END IF;
END;
/
```

15.8.5 错误捕获语句

缺省时，当PL/SQL函数执行过程中发生错误时退出函数执行，并且周围的事务也会回滚。可以用一个带有EXCEPTION子句的BEGIN块捕获错误并且从中恢复。其语法是正常的BEGIN块语法的一个扩展：

```
[<<label>>]
[DECLARE
   declarations]
BEGIN
   statements
EXCEPTION
   WHEN condition [OR condition ...] THEN
      handler_statements
   [WHEN condition [OR condition ...] THEN
      handler_statements
   ...]
END;
```

如果没有发生错误，这种形式的块儿只是简单地执行所有语句，然后转到END之后的下一个语句。但是如果在执行的语句内部发生了一个错误，则这个语句将会回滚，然后转到EXCEPTION列表。寻找匹配错误的第一个条件。若找到匹配，则执行对应的handler_statements，然后转到END之后的下一个语句。如果没有找到匹配，则会向事务的外层报告错误，和没有EXCEPTION子句一样。

也就是说该错误可以被一个包围块用EXCEPTION捕获，如果没有包围块，则进行退出函数处理。

condition的名字可以是《 错误码参考 》的SQL标准错误码编号说明的任意值。特殊的条件名OTHERS匹配除了QUERY_CANCELED之外的所有错误类型。

如果在选中的handler_statements里发生了新错误，则不能被这个EXCEPTION子句捕获，而是向事务的外层报告错误。一个外层的EXCEPTION子句可以捕获它。

如果一个错误被EXCEPTION 捕获，PL/SQL函数的局部变量保持错误发生时的原值，但是所有该块中想写入数据库中的状态都回滚。

示例：

```
CREATE TABLE mytab(id INT,firstname VARCHAR(20),lastname VARCHAR(20))DISTRIBUTE BY hash(id);
```

```
INSERT INTO mytab(firstname, lastname) VALUES('Tom', 'Jones');

CREATE FUNCTION fun_exp() RETURNS INT
AS $$
DECLARE
    x INT :=0;
    y INT;
BEGIN
    UPDATE mytab SET firstname = 'Joe' WHERE lastname = 'Jones';
    x := x + 1;
    y := x / 0;
EXCEPTION
    WHEN division_by_zero THEN
        RAISE NOTICE 'caught division_by_zero';
        RETURN x;
END;$$
LANGUAGE plpgsql;

call fun_exp();
NOTICE:  caught division_by_zero
 fun_exp
---------
       1
(1 row)

select * from mytab;
 id | firstname | lastname
----+-----------+----------
    | Tom       | Jones
(1 row)

DROP FUNCTION fun_exp();
DROP TABLE mytab;
```

当控制到达给y赋值的地方时，会有一个division_by_zero错误失败。这个错误将被EXCEPTION子句捕获。而在RETURN语句里返回的数值将是x的增量值。

说明

进入和退出一个包含EXCEPTION子句的块要比不包含的块开销大的多。因此，不必要的时候不要使用EXCEPTION。

在下列场景中，无法捕获处理异常，整个存储过程回滚：节点故障、网络故障引起的存储过程参与节点线程退出。

示例： UPDATE/INSERT异常

这个例子根据使用异常处理器执行恰当的UPDATE或INSERT 。

```
CREATE TABLE db (a INT, b TEXT);

CREATE FUNCTION merge_db(key INT, data TEXT) RETURNS VOID AS
$$
BEGIN
    LOOP

--第一次尝试更新key
        UPDATE db SET b = data WHERE a = key;
        IF found THEN
            RETURN;
        END IF;
--不存在，所以尝试插入key，如果其他人同时插入相同的key，我们可能得到唯一key失败。
        BEGIN
            INSERT INTO db(a,b) VALUES (key, data);
            RETURN;
        EXCEPTION WHEN unique_violation THEN
        --什么也不做，并且循环尝试再次更新。
        END;
    END LOOP;
```

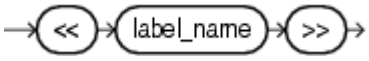
```
END;
$$
LANGUAGE plpgsql;

SELECT merge_db(1, 'david');
SELECT merge_db(1, 'dennis');

--删除FUNCTION和TABLE
DROP FUNCTION merge_db;
DROP TABLE db ;
```

15.8.6 GOTO 语句

GOTO语句可以实现从GOTO位置到目标语句的无条件跳转。GOTO语句会改变原本的执行逻辑，因此应该慎重使用，或者也可以使用EXCEPTION处理特殊场景。当执行GOTO语句时，目标Label必须是唯一的。

语法

label declaration ::=



goto statement ::=

示例

```
CREATE OR REPLACE PROCEDURE GOTO_test()
AS
DECLARE
    v1  int;
BEGIN
    v1  := 0;
        LOOP
        EXIT WHEN v1 > 100;
                v1 := v1 + 2;
                if v1 > 25 THEN
                        GOTO pos1;
                END IF;
        END LOOP;
<<pos1>>
v1 := v1 + 10;
raise info 'v1 is %. ', v1;
END;
/

call GOTO_test();
```

限制场景

GOTO使用有以下限制场景

- 不支持有多个相同的GOTO labels目标场景，无论是否在同一个block中。

  ```
  BEGIN
    GOTO pos1;
    <<pos1>>
    SELECT * FROM ...
    <<pos1>>
    UPDATE t1 SET ...
  END;
  ```

- 不支持GOTO跳转到IF语句，CASE语句，LOOP语句中。

```
BEGIN
  GOTO pos1;
  IF valid THEN
    <<pos1>>
    SELECT * FROM ...
  END IF;
END;
```

- 不支持GOTO语句从一个IF子句跳转到另一个IF子句，或从一个CASE语句的WHEN子句跳转到另一个WHEN子句。

```
BEGIN
  IF valid THEN
    GOTO pos1;
    SELECT * FROM ...
  ELSE
    <<pos1>>
    UPDATE t1 SET ...
  END IF;
END;
```

- 不支持从外部块跳转到内部的BEGIN-END块。

```
BEGIN
  GOTO pos1;
  BEGIN
    <<pos1>>
    UPDATE t1 SET ...
  END;
END;
```

- 不支持从异常处理部分跳转到当前的BEGIN-END块。但可以跳转到上层BEGIN-END块。

```
BEGIN
  <<pos1>>
  UPDATE t1 SET ...
  EXCEPTION
    WHEN condition THEN
      GOTO pos1;
END;
```

- 如果从GOTO到一个不包含执行语句的位置，需要添加NULL语句。

```
DECLARE
  done  BOOLEAN;
BEGIN
  FOR i IN 1..50 LOOP
    IF done THEN
      GOTO end_loop;
    END IF;
    <<end_loop>>  -- not allowed unless an executable statement follows
    NULL; -- add NULL statement to avoid error
  END LOOP;  -- raises an error without the previous NULL
END;
/
```

15.9 锁操作

GaussDB(DWS)提供了多种锁模式用于控制对表中数据的并发访问。这些模式可以用在MVCC（多版本并发控制）无法给出期望行为的场合。同样，大多数GaussDB(DWS)命令自动施加恰当的锁，以保证被引用的表在命令的执行过程中不会以一种不兼容的方式被删除或者修改。比如，在存在其他并发操作的时候，ALTER TABLE是不能在同一个表上执行的。

15.10 游标

15.10.1 游标概述

为了处理SQL语句，存储过程进程分配一段内存区域来保存上下文联系。游标是指向上下文区域的句柄或指针。借助游标，存储过程可以控制上下文区域的变化。

须知

当游标作为存储过程的返回值时，如果使用JDBC调用该存储过程，返回的游标将不可用。

游标的使用分为显式游标和隐式游标。对于不同的SQL语句，游标的使用情况不同，详细信息请参见表15-2。

表 15-2 游标使用情况

SQL语句	游标
非查询语句	隐式的
结果是单行的查询语句	隐式的或显式的
结果是多行的查询语句	显式的

15.10.2 显式游标

显式游标主要用于对查询语句的处理，尤其是在查询结果为多条记录的情况下。

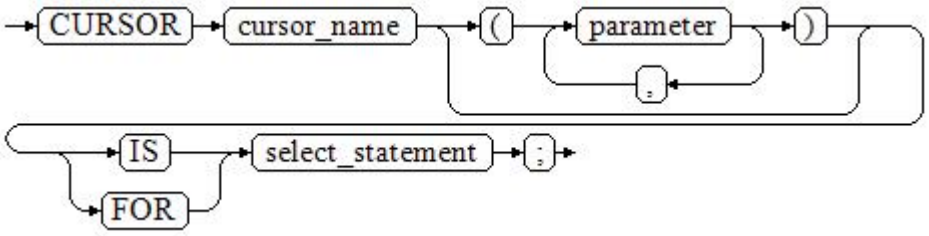
处理步骤

显式游标处理需六个PL/SQL步骤：

步骤1 **定义静态游标：**就是定义一个游标名，以及与其相对应的SELECT语句。

定义静态游标的语法图，请参见图15-26。

图 15-26 static_cursor_define::=



参数说明：

- cursor_name：定义的游标名。
- parameter：游标参数，只能为输入参数，其格式为：
  ```
  parameter_name datatype
  ```

- select_statement：查询语句。

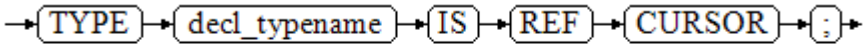
说明

根据执行计划的不同，系统会自动判断该游标是否可以用于以倒序的方式检索数据行。

定义动态游标：指ref游标，可以通过一组静态的SQL语句动态的打开游标。首先定义ref游标类型，然后定义该游标类型的游标变量，在打开游标时通过OPEN FOR动态绑定SELECT语句。

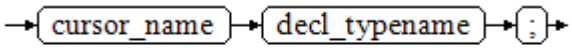
定义动态游标的语法图，请参见图15-27和图15-28。

图 15-27 cursor_typename::=



GaussDB(DWS)支持sys_refcursor动态游标类型，函数或存储过程可以通过sys_refcursor参数传入或传出游标结果集合，函数也可以通过返回sys_refcursor来返回游标结果集合。

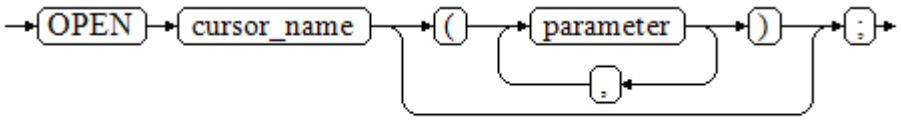
图 15-28 dynamic_cursor_define::=



步骤2 **打开静态游标：**就是执行游标所对应的SELECT语句，将其查询结果放入工作区，并且指针指向工作区的首部，标识游标结果集合。如果游标查询语句中带有FOR UPDATE选项，OPEN语句还将锁定数据库表中游标结果集合对应的数据行。

打开静态游标的语法图，请参见图15-29。

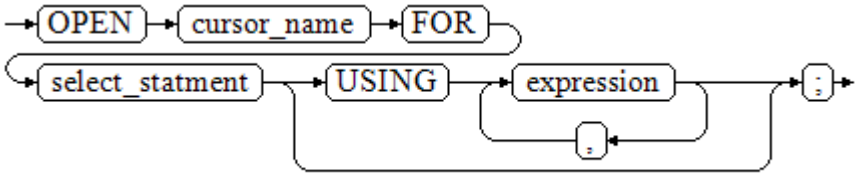
图 15-29 open_static_cursor::=



打开动态游标：可以通过OPEN FOR语句打开动态游标，动态绑定SQL语句。

打开动态游标的语法图，请参见图15-30。

图 15-30 open_dynamic_cursor::=

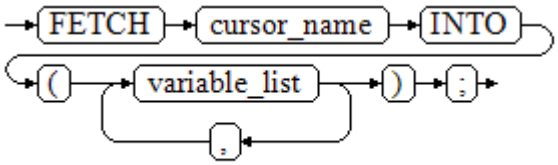


PL/SQL程序不能用OPEN语句重复打开一个游标。

步骤3 提取游标数据：检索结果集合中的数据行，放入指定的输出变量中。

提取游标数据的语法图，请参见图15-31。

图 15-31 fetch_cursor::=



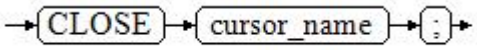
步骤4 对该记录进行处理。

步骤5 继续处理，直到活动集合中没有记录。

步骤6 关闭游标：当提取和处理完游标结果集合数据后，应及时关闭游标，以释放该游标所占用的系统资源，并使该游标的工作区变成无效，不能再使用FETCH语句获取其中数据。关闭后的游标可以使用OPEN语句重新打开。

关闭游标的语法图，请参见图15-32。

图 15-32 close_cursor::=



----结束

属性

游标的属性用于控制程序流程或者了解程序的状态。当运行DML语句时，PL/SQL打开一个内建游标并处理结果，游标是维护查询结果的内存中的一个区域，游标在运行DML语句时打开，完成后关闭。显式游标的属性为：

- %FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
- %NOTFOUND布尔型属性：与%FOUND相反。
- %ISOPEN布尔型属性：当游标已打开时返回TRUE。
- %ROWCOUNT数值型属性：返回已从游标中读取的记录数。

示例

```
--游标参数的传递方法。
CREATE OR REPLACE PROCEDURE cursor_proc1()
AS
DECLARE
   DEPT_NAME VARCHAR(100);
   DEPT_LOC NUMBER(4);
   --定义游标
   CURSOR C1 IS
      SELECT section_name, place_id FROM hr.sections WHERE section_id <= 50;
   CURSOR C2(sect_id INTEGER) IS
      SELECT section_name, place_id FROM hr.sections WHERE section_id <= sect_id;
   TYPE CURSOR_TYPE IS REF CURSOR;
   C3 CURSOR_TYPE;
```

```
    SQL_STR VARCHAR(100);
BEGIN
    OPEN C1;--打开游标
    LOOP
        --通过游标取值
        FETCH C1 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C1;--关闭游标

    OPEN C2(10);
    LOOP
        FETCH C2 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C2%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C2;

    SQL_STR := 'SELECT section_name, place_id FROM hr.sections WHERE section_id <= :DEPT_NO;';
    OPEN C3 FOR SQL_STR USING 50;
    LOOP
        FETCH C3 INTO DEPT_NAME, DEPT_LOC;
        EXIT WHEN C3%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(DEPT_NAME||'---'||DEPT_LOC);
    END LOOP;
    CLOSE C3;
END;
/

CALL cursor_proc1();

DROP PROCEDURE cursor_proc1;
--给工资低于3000的员工增加工资500。
CREATE TABLE hr.staffs_t1 AS TABLE hr.staffs;

CREATE OR REPLACE PROCEDURE cursor_proc2()
AS
DECLARE
   V_EMPNO  NUMBER(6);
   V_SAL    NUMBER(8,2);
   CURSOR C IS SELECT staff_id, salary FROM hr.staffs_t1;
BEGIN
   OPEN C;
   LOOP
      FETCH C INTO V_EMPNO, V_SAL;
      EXIT WHEN C%NOTFOUND;
      IF V_SAL<=3000 THEN
           UPDATE hr.staffs_t1 SET salary =salary + 500 WHERE staff_id = V_EMPNO;
      END IF;
   END LOOP;
   CLOSE C;
END;
/

CALL cursor_proc2();

--删除存储过程
DROP PROCEDURE cursor_proc2;
DROP TABLE hr.staffs_t1;
--SYS_REFCURSOR类型做为函数参数
CREATE OR REPLACE PROCEDURE proc_sys_ref(O OUT SYS_REFCURSOR)
IS
C1 SYS_REFCURSOR;
BEGIN
OPEN C1 FOR SELECT section_ID FROM HR.sections ORDER BY section_ID;
O := C1;
END;
/
```

```
DECLARE
C1 SYS_REFCURSOR;
TEMP NUMBER(4);
BEGIN
proc_sys_ref(C1);
LOOP
  FETCH C1 INTO TEMP;
  DBMS_OUTPUT.PUT_LINE(C1%ROWCOUNT);
  EXIT WHEN C1%NOTFOUND;
END LOOP;
END;
/

--删除存储过程
DROP PROCEDURE proc_sys_ref;
```

15.10.3 隐式游标

对于非查询语句，如修改、删除操作，则由系统自动地为这些操作设置游标并创建其工作区，这些由系统隐含创建的游标称为隐式游标，隐式游标的名字为SQL，这是由系统定义的。

简介

对于隐式游标的操作，如定义、打开、取值及关闭操作，都由系统自动地完成，无需用户进行处理。用户只能通过隐式游标的相关属性，来完成相应的操作。在隐式游标的工作区中，所存放的数据是最新处理的一条SQL语句所包含的数据，与用户自定义的显式游标无关。

格式调用为： SQL%

说明

INSERT，UPDATE，DELETE，SELECT语句中不必明确定义游标。

属性

隐式游标属性为：

- SQL%FOUND布尔型属性：当最近一次读记录时成功返回，则值为TRUE。
- SQL%NOTFOUND布尔型属性：与%FOUND相反。
- SQL%ROWCOUNT数值型属性：返回已从游标中读取得记录数。
- SQL%ISOPEN布尔型属性：取值总是FALSE。SQL语句执行完毕立即关闭隐式游标。

示例

```
--删除EMP表中某部门的所有员工，如果该部门中已没有员工，则在DEPT表中删除该部门。
CREATE TABLE hr.staffs_t1AS TABLE hr.staffs;
CREATE TABLE hr.sections_t1AS TABLE hr.sections;

CREATE OR REPLACE PROCEDURE proc_cursor3()
AS
   DECLARE
   V_DEPTNO NUMBER(4) := 100;
   BEGIN
      DELETE FROM hr.staffs WHERE section_ID = V_DEPTNO;
      --根据游标状态做进一步处理
      IF SQL%NOTFOUND THEN
```

```
        DELETE FROM hr.sections_t1 WHERE section_ID = V_DEPTNO;
        END IF;
    END;
/

CALL proc_cursor3();

--删除存储过程和临时表
DROP PROCEDURE proc_cursor3;
DROP TABLE hr.staffs_t1;
DROP TABLE hr.sections_t1;
```

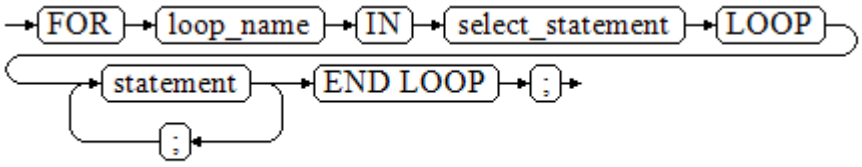
15.10.4 游标循环

游标在WHILE语句、LOOP语句中的使用称为游标循环，一般这种循环都需要使用OPEN、FETCH和CLOSE语句。下面要介绍的一种循环不需要这些操作，可以简化游标循环的操作，这种循环方式适用于静态游标的循环，不用执行静态游标的四个步骤。

语法

FOR AS循环的语法请参见图15-33。

图 15-33 FOR_AS_loop::=



注意事项

- 不能在该循环语句中对查询的表进行更新操作。
- 变量loop_name会自动定义且只在此循环中有效，类型和select_statement的查询结果类型一致。loop_name的取值就是select_statement的查询结果。
- 游标的属性中%FOUND、%NOTFOUND、%ROWCOUNT在GaussDB(DWS)数据库中都是访问同一个内部变量，事务和匿名块不支持多个游标同时访问。

示例

```
BEGIN
FOR ROW_TRANS IN
        SELECT first_name FROM hr.staffs
    LOOP
        DBMS_OUTPUT.PUT_LINE (ROW_TRANS.first_name );
    END LOOP;
END;
/

--创建表
CREATE TABLE integerTable1( A INTEGER)DISTRIBUTE BY hash(A);
CREATE TABLE integerTable2( B INTEGER) DISTRIBUTE BY hash(B);
INSERT INTO integerTable2 VALUES(2);

--多游标共享游标属性的标量
DECLARE
    CURSOR C1 IS SELECT A FROM integerTable1;--声明游标
    CURSOR C2 IS SELECT B FROM integerTable2;
    PI_A INTEGER;
```

```
  PI_B INTEGER;
BEGIN
  OPEN C1;--打开游标
  OPEN C2;
  FETCH C1 INTO PI_A; ----  C1%FOUND 和 C2%FOUND 值为 FALSE
  FETCH C2 INTO PI_B; ----  C1%FOUND 和 C2%FOUND 的值都为 TRUE
  --判断游标状态
  IF C1%FOUND THEN
     IF C2%FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Dual cursor share paremeter.');
     END IF;
  END IF;
  CLOSE C1;--关闭游标
  CLOSE C2;
END;
/

--删除临时表
DROP TABLE integerTable1;
DROP TABLE integerTable2;
```

15.11 高级包

15.11.1 DBMS_LOB

接口介绍

高级功能包DBMS_LOB支持的所有接口请参见**表15-3**。

表 15-3 DBMS_LOB

接口名称	描述
DBMS_LOB.GETLENGTH	获取并返回指定的LOB类型对象的长度。
DBMS_LOB.OPEN	打开一个LOB返回一个LOB的描述符。
DBMS_LOB.READ	根据指定的长度及起始位置偏移读取LOB内容的一部分到BUFFER缓冲区。
DBMS_LOB.WRITE	根据指定长度及起始位置偏移将BUFFER中内容写入到LOB中。
DBMS_LOB.WRITEAPPEND	根据指定长度将BUFFER中内容写入到LOB的尾部。
DBMS_LOB.COPY	根据指定长度及起始位置偏移将BLOB内容写入到另一个BLOB中。
DBMS_LOB.ERASE	根据指定长度及起始位置偏移删除BLOB中的内容。
DBMS_LOB.CLOSE	关闭已经打开的LOB描述符。
DBMS_LOB.INSTR	返回一个字符串在LOB中第N次出现的位置。
DBMS_LOB.COMPARE	比较两个LOB或者两个LOB的某一部分。

接口名称	描述
DBMS_LOB.SUBSTR	用于读取一个LOB的子串，返回读取的字节个数或者字符个数。
DBMS_LOB.TRIM	用于截断指定长度的LOB，执行完会将LOB的长度设置为newlen参数指定的长度。
DBMS_LOB.CREATETEMPORARY	创建一个临时的BLOB或者CLOB。
DBMS_LOB.APPEND	将原LOB的内容拼接到目的LOB中。

- DBMS_LOB.GETLENGTH

  存储过程GETLENGTH获取并返回指定的LOB类型对象的长度。

  DBMS_LOB.GETLENGTH函数原型为：

```
DBMS_LOB.GETLENGTH (
lob_loc    IN   BLOB)
RETURN INTEGER;

DBMS_LOB.GETLENGTH (
lob_loc    IN   CLOB)
RETURN INTEGER;
```

  表 15-4 DBMS_LOB.GETLENGTH 接口参数说明

参数	描述
lob_loc	待获取长度的指定的LOB类型的对象。

- DBMS_LOB.OPEN

  存储过程打开一个LOB，并返回一个LOB描述符，该过程无实际意义，仅用于兼容。

  DBMS_LOB.OPEN函数原型为：

```
DBMS_LOB.LOB (
lob_loc    INOUT   BLOB,
open_mode IN BINARY_INTEGER);

DBMS_LOB.LOB (
lob_loc  INOUT    CLOB,
open_mode IN BINARY_INTEGER);
```

  表 15-5 DBMS_LOB.OPEN 接口参数说明

参数	描述
lob_loc	被打开的一个BLOB或者CLOB描述符。
open_mode IN BINARY_INTEGER	打开的模式，目前支持DBMS_LOB.LOB_READWRITE

- DBMS_LOB.READ

  存储过程READ根据指定长度及起始位置偏移读取LOB内容的一部分到BUFFER缓冲区。

  DBMS_LOB.READ函数原型为：

```
DBMS_LOB.READ (
lob_loc     IN          BLOB,
amount      IN          INTEGER,
offset      IN          INTEGER,
buffer      OUT         RAW);

DBMS_LOB.READ (
lob_loc    IN           CLOB,
amount     IN OUT       INTEGER,
offset     IN           INTEGER,
buffer     OUT          VARCHAR2);
```

  表 15-6 DBMS_LOB.READ 接口参数说明

参数	说明
lob_loc	待读入的指定的LOB类型的对象。
amount	读入长度。 **说明** 如果读入长度为负，会收到错误提示“ERROR: argument 2 is null, invalid, or out of range.”
offset	指定从LOB内容的哪个位置开始读取的偏移（即相对LOB内容起始位置的字节数）。
buffer	读取LOB内容后存放的目标缓冲区。

- DBMS_LOB.WRITE

  存储过程WRITE根据指定长度及起始位置偏移将BUFFER中内容写入到LOB变量中。

  DBMS_LOB.WRITE函数原型为：

```
DBMS_LOB.WRITE (
lob_loc    IN OUT     BLOB,
amount     IN         INTEGER,
offset     IN         INTEGER,
buffer     IN         RAW);

DBMS_LOB.WRITE (
lob_loc   IN OUT      CLOB,
amount    IN          INTEGER,
offset    IN          INTEGER,
buffer    IN          VARCHAR2);
```

  表 15-7 DBMS_LOB.WRITE 接口参数说明

参数	说明
lob_loc	待写入的指定的LOB类型的对象。
amount	写入长度。 **说明** 如果写入长度小于1或写入长度大于待写入的内容长度，则报错。

参数	说明
offset	指定从LOB内容的哪个位置开始写入的偏移（即相对LOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于LOB类型最大长度，则报错。
buffer	待写入的内容。

- DBMS_LOB.WRITEAPPEND

  存储过程WRITEAPPEND根据指定长度将BUFFER中内容写入到LOB的尾部。

  DBMS_LOB.WRITEAPPEND函数原型为：

```
DBMS_LOB.WRITEAPPEND (
lob_loc     IN OUT     BLOB,
amount      IN         INTEGER,
buffer      IN         RAW);

DBMS_LOB.WRITEAPPEND (
lob_loc     IN OUT     CLOB,
amount      IN         INTEGER,
buffer      IN         VARCHAR2);
```

表 15-8 DBMS_LOB.WRITEAPPEND 接口参数说明

参数	说明
lob_loc	待写入的指定的LOB类型的对象。
amount	写入长度。 **说明** 如果写入长度小于1或写入长度大于待写入的内容长度，则报错。
buffer	待写入的内容。

- DBMS_LOB.COPY

  存储过程COPY根据指定长度及起始位置偏移将BLOB内容拷贝到另一个BLOB中。

  DBMS_LOB.COPY函数原型为：

```
DBMS_LOB.COPY (
dest_lob      IN OUT     BLOB,
src_lob       IN         BLOB,
amount        IN         INTEGER,
dest_offset   IN         INTEGER  DEFAULT 1,
src_offset    IN         INTEGER  DEFAULT 1);
```

表 15-9 DBMS_LOB.COPY 接口参数说明

参数	说明
dest_lob	待拷入的BLOB类型对象。
src_lob	待拷出的BLOB类型对象。

参数	说明
amount	拷贝长度。 **说明** 如果拷入长度小于1或拷入长度大于BLOB类型最大长度，则报错。
dest_offset	指定从BLOB内容的哪个位置开始拷入的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于BLOB类型最大长度，则报错。
src_offset	指定从BLOB内容的哪个位置开始拷出的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于拷贝来源BLOB的长度，则报错。

- DBMS_LOB.ERASE

  存储过程ERASE根据指定长度及起始位置偏移删除BLOB中的内容。

  DBMS_LOB.ERASE函数原型为：

```
DBMS_LOB.ERASE (
lob_loc          IN OUT   BLOB,
amount           IN OUT   INTEGER,
offset           IN       INTEGER DEFAULT 1);
```

  表 15-10 DBMS_LOB.ERASE 接口参数说明

参数	说明
lob_loc	待删除内容的BLOB类型对象。
amount	待删除的长度。 **说明** 如果删除长度小于1或删除长度大于BLOB类型最大长度，则报错。
offset	指定从BLOB内容的哪个位置开始删除的偏移（即相对BLOB内容起始位置的字节数）。 **说明** 如果偏移量小于1或偏移量大于BLOB类型最大长度，则报错。

- DBMS_LOB.CLOSE

  存储过程CLOSE根据指定长度及起始位置偏移关闭已经打开的LOB内容。

  DBMS_LOB.CLOSE函数原型为：

```
DBMS_LOB.CLOSE(
src_lob      IN             BLOB);

DBMS_LOB.CLOSE (
src_lob      IN             CLOB);
```

表 15-11 DBMS_LOB.CLOSE 接口参数说明

参数	说明
src_loc	待关闭的LOB类型对象。

- DBMS_LOB.INSTR

  该函数返回在LOB中第N次出现的位置，如果输入的是一些无效值会返回NULL值。offset < 1 or offset > LOBMAXSIZE，nth < 1，nth > LOBMAXSIZE。

  DBMS_LOB.INSTR函数原型为：

```
DBMS_LOB.INSTR (
lob_loc     IN      BLOB,
pattern     IN      RAW,
offset      IN      INTEGER := 1,
nth         IN      INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.INSTR (
lob_loc    IN      CLOB,
pattern    IN      VARCHAR2 ,
offset     IN      INTEGER := 1,
nth        IN      INTEGER := 1)
RETURN INTEGER;
```

表 15-12 DBMS_LOB.INSTR 接口参数说明

参数	说明
lob_loc	要查找的LOB描述符。
pattern	要匹配的模式，对于BLOB是由一组RAW类型的数据组成，对于CLOB是由一组text类型的数据组成。
offset	对于BLOB是以字节为单位的绝对偏移量，对于CLOB是以字符为单位的偏移量，模式匹配的起始位置是1。
nth	模式匹配的次数，最小值为1。

- DBMS_LOB.COMPARE

  这个函数比较两个LOB或者两个LOB的一部分。

  - 如果比较的结果相等返回0，否则返回非零的值。
  - 如果第一个CLOB比第二个小，返回-1；如果第一个CLOB比第二个大，返回1。
  - 如果amount，offset_1，offset_2这几个参数有无效参数返回NULL，有效的偏移量范围是1~LOBMAXSIZE。

  DBMS_LOB.READ函数原型为：

```
DBMS_LOB.COMPARE (
lob_1     IN      BLOB,
lob_2     IN      BLOB,
amount    IN      INTEGER := DBMS_LOB.LOBMAXSIZE,
offset_1  IN      INTEGER := 1,
offset_2  IN      INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.COMPARE (
lob_1     IN      CLOB,
```

```
lob_2     IN     CLOB,
amount    IN     INTEGER := DBMS_LOB.LOBMAXSIZE,
offset_1  IN     INTEGER := 1,
offset_2  IN     INTEGER := 1)
RETURN INTEGER;
```

表 15-13 DBMS_LOB.COMPARE 接口参数说明

参数	说明
lob_1	第一个要比较的LOB描述符。
lob_2	第二个要比较的LOB描述符。
amount	要比较的字符数或者字节数， 最大值为DBMS_LOB.LOBMAXSIZE。
offset_1	第一个LOB描述符的偏移量，初始位置是1。
offset_2	第二个LOB描述符的偏移量，初始位置是1。

- DBMS_LOB.SUBSTR

  用于读取一个LOB的子串，返回读取的字节个数或者字符个数，当amount > 1或者amount < 32767，offset < 1或者offset > LOBMAXSIZE的时候返回值是NULL。

  DBMS_LOB.SUBSTR函数原型为：

  ```
  DBMS_LOB.SUBSTR (
  lob_loc     IN     BLOB,
  amount      IN     INTEGER := 32767,
  offset      IN     INTEGER := 1)
  RETURN RAW;

  DBMS_LOB.SUBSTR (
  lob_loc     IN     CLOB,
  amount      IN     INTEGER := 32767,
  offset      IN     INTEGER := 1)
  RETURN VARCHAR2;
  ```

  表 15-14 DBMS_LOB.SUBSTR 接口参数说明

参数	说明
lob_loc	将要读取子串的LOB描述符，对于BLOB类型的返回值是读取的字节个数，对于CLOB类型的返回值是字符个数。
offset	要读取的字节数或者字符数量。
buffer	从开始位置偏移的字符数或者字节数量。

- DBMS_LOB.TRIM

  这个存储过程用于截断指定长度的LOB，执行完这个存储过程会将LOB的长度设置为newlen参数指定的长度。如果对一个空的LOB执行截断操作，不会有任何执行结果；如果指定的长度比LOB的长度长，会产生一个异常。

  DBMS_LOB.TRIM函数原型为：

  ```
  DBMS_LOB.TRIM (
  lob_loc     IN OUT     BLOB,
  ```

```
newlen      IN          INTEGER);

DBMS_LOB.TRIM (
lob_loc    IN          OUT CLOB,
newlen     IN          INTEGER);
```

表 15-15 DBMS_LOB.TRIM 接口参数说明

参数	说明
lob_loc	待读入的指定BLOB类型的对象。
newlen	截断后LOB的新长度对于BLOB是字节数，对于CLOB是字符数。

- DBMS_LOB.CREATETEMPORARY

  这个存储过程创建一个临时的BLOB或者CLOB，这个存储过程仅用于语法上的兼容，并无实际意义。

  DBMS_LOB.CREATETEMPORARY函数原型为：

```
DBMS_LOB.CREATETEMPORARY (
lob_loc    IN OUT      BLOB,
cache      IN          BOOLEAN,
dur        IN          INTEGER);

DBMS_LOB.CREATETEMPORARY (
lob_loc    IN OUT     CLOB,
cache      IN          BOOLEAN,
dur        IN         INTEGER);
```

表 15-16 DBMS_LOB.CREATETEMPORARY 接口参数说明

参数	说明
lob_loc	LOB描述符。
cache	仅用于语法上的兼容。
dur	仅用于语法上的兼容。

- DBMS_LOB.APPEND

  存储过程READ根据指定长度及起始位置偏移读取BLOB内容的一部分到BUFFER缓冲区。

  DBMS_LOB.APPEND函数原型为：

```
DBMS_LOB.APPEND (
dest_lob    IN OUT      BLOB,
src_lob     IN          BLOB);

DBMS_LOB.APPEND (
dest_lob    IN OUT      CLOB,
src_lob     IN          CLOB);
```

表 15-17 DBMS_LOB.APPEND 接口参数说明

参数	说明
dest_lob	要写入的LOB描述符。

参数	说明
src_lob	读取的LOB描述符。

示例

```
--获取字符串的长度
SELECT DBMS_LOB.GETLENGTH('12345678');

DECLARE
myraw  RAW(100);
amount INTEGER :=2;
buffer INTEGER :=1;
begin
DBMS_LOB.READ('123456789012345',amount,buffer,myraw);
dbms_output.put_line(myraw);
end;
/

CREATE TABLE blob_Table (t1 blob) DISTRIBUTE BY REPLICATION;
CREATE TABLE blob_Table_bak (t2 blob) DISTRIBUTE BY REPLICATION;
INSERT INTO blob_Table VALUES('abcdef');
INSERT INTO blob_Table_bak VALUES('22222');

DECLARE
str varchar2(100) := 'abcdef';
source raw(100);
dest blob;
copyto blob;
amount int;
PSV_SQL varchar2(100);
PSV_SQL1 varchar2(100);
a int :=1;
len int;
BEGIN
source := utl_raw.cast_to_raw(str);
amount := utl_raw.length(source);

PSV_SQL :='select * from blob_Table for update';
PSV_SQL1 := 'select * from blob_Table_bak for update';

EXECUTE IMMEDIATE PSV_SQL into dest;
EXECUTE IMMEDIATE PSV_SQL1 into copyto;

DBMS_LOB.WRITE(dest, amount, 1, source);
DBMS_LOB.WRITEAPPEND(dest, amount, source);

DBMS_LOB.ERASE(dest, a, 1);
DBMS_OUTPUT.PUT_LINE(a);
DBMS_LOB.COPY(copyto, dest, amount, 10, 1);
DBMS_LOB.CLOSE(dest);
RETURN;
END;
/

--删除表
DROP TABLE blob_Table;
DROP TABLE blob_Table_bak;
```

15.11.2 DBMS_RANDOM

接口介绍

高级功能包DBMS_RANDOM支持的所有接口请参见**表15-18**。

表 15-18 DBMS_RANDOM 接口参数说明

接口名称	描述
DBMS_RANDOM.SEED	设置一个随机数的种子。
DBMS_RANDOM.VALUE	生成一个大小介于指定的low及high之间的随机数。

- DBMS_RANDOM.SEED

  存储过程SEED用于设置一个随机数的种子。DBMS_RANDOM.SEED函数原型为：

  ```
  DBMS_RANDOM.SEED (seed  IN  INTEGER);
  ```

  表 15-19 DBMS_RANDOM.SEED 接口参数说明

参数	描述
seed	用于产生一个随机数的种子。

- DBMS_RANDOM.VALUE

  存储过程VALUE生成一个大小介于指定的low及high之间的随机数。
  DBMS_RANDOM.VALUE函数原型为：

  ```
  DBMS_RANDOM.VALUE(
  low  IN  NUMBER,
  high IN  NUMBER)
  RETURN NUMBER;
  ```

  表 15-20 DBMS_RANDOM.VALUE 接口参数说明

参数	描述
low	指定随机数大小的下边界，生成的随机数大于或等于low。
high	指定随机数大小的上边界，生成的随机数小于high。

说明

实际上，只要求这里的参数类型是NUMERIC即可，对于左右边界的大小并没有要求。

示例

```
--产生0到1之间的随机数：
SELECT DBMS_RANDOM.VALUE(0,1);

--对于指定范围内的整数，要加入参数low和high，并从结果中截取较小的数（最大值不能被作为可能的值）。所以对于0到99之间的整数，使用下面的代码：
SELECT TRUNC(DBMS_RANDOM.VALUE(0,100));
```

15.11.3 DBMS_OUTPUT

接口介绍

高级功能包DBMS_OUTPUT支持的所有接口请参见表15-21。

表 15-21 DBMS_OUTPUT

接口名称	描述
DBMS_OUTPUT.PUT_LINE	输出指定的文本，文本长度不能超过32767字节。
DBMS_OUTPUT.PUT	将指定的文本输出到指定文本的前面，不添加换行符，文本长度不能超过32767字节。
DBMS_OUTPUT.ENABLE	设置输出缓冲区的大小。若不指定，缓冲区最大只能容纳20000字节，缓冲区最小可设置为2000字节，若设置小于2000字节将按2000字节处理。

- DBMS_OUTPUT.PUT_LINE

存储过程PUT_LINE向消息缓冲区写入一行带有行结束符的文本。
DBMS_OUTPUT.PUT_LINE函数原型为：

```
DBMS_OUTPUT.PUT_LINE (
item IN VARCHAR2);
```

表 15-22 DBMS_OUTPUT.PUT_LINE 接口参数说明

参数	描述
item	写入消息缓冲区的文本。

- DBMS_OUTPUT.PUT

存储过程PUT将指定的文本输出到指定文本的前面，不添加换行符。
DBMS_OUTPUT.PUT函数原型为：

```
DBMS_OUTPUT.PUT (
item IN VARCHAR2);
```

表 15-23 DBMS_OUTPUT.PUT 接口参数说明

参数	描述
item	写入指定文本前的文本。

- DBMS_OUTPUT.ENABLE

存储过程ENABLE设置输出缓冲区的大小，如果不指定的话缓冲区最大只能容纳20000字节。DBMS_OUTPUT.ENABLE函数原型为：

```
DBMS_OUTPUT.ENABLE (
buf IN INTEGER);
```

表 15-24 DBMS_OUTPUT.ENABLE 接口参数说明

参数	描述
buf	设置输出缓冲区的大小。

示例

```
BEGIN
    DBMS_OUTPUT.ENABLE(50);
    DBMS_OUTPUT.PUT ('hello, ');
    DBMS_OUTPUT.PUT_LINE('database!');--输出hello, database!
END;
/
```

15.11.4 UTL_RAW

接口介绍

高级功能包UTL_RAW支持的所有接口请参见**表15-25**。

表 15-25 UTL_RAW

接口名称	描述
UTL_RAW.CAST_FROM_BINARY_INTEGER	将INTEGER类型值转换为二进制表示形式（即RAW类型）。
UTL_RAW.CAST_TO_BINARY_INTEGER	将二进制表示形式的整型值（即RAW类型）转换为INTEGER类型。
UTL_RAW.LENGTH	获取RAW类型对象的长度。
UTL_RAW.CAST_TO_RAW	将VARCHAR2类型值转化为二进制表示形式（即RAW类型）。

须知

RAW类型的外部表现形式是十六进制，内部存储形式是二进制。例如一个RAW类型的数据11001011的表现形式为‘CB’，即在实际的类型转换中输入的是‘CB’。

- UTL_RAW.CAST_FROM_BINARY_INTEGER

  存储过程CAST_FROM_BINARY_INTEGER将INTEGER类型值转换为二进制表示形式（即RAW类型）。

  UTL_RAW.CAST_FROM_BINARY_INTEGER函数原型为：

```
UTL_RAW.CAST_FROM_BINARY_INTEGER (
n           IN  INTEGER,
endianess   IN  INTEGER)
RETURN RAW;
```

表 15-26 UTL_RAW.CAST_FROM_BINARY_INTEGER 接口参数说明

参数	描述
n	待转成RAW类型的整型数值。
endianess	表示字节序的整型值1或2（1代表BIG_ENDIAN，2代表LITTLE-ENDIAN）。

- UTL_RAW.CAST_TO_BINARY_INTEGER

  存储过程CAST_TO_BINARY_INTEGER将二进制表示形式的整型值（即RAW类型）转换为INTEGER类型。

  UTL_RAW.CAST_TO_BINARY_INTEGER函数原型为：

  ```
  UTL_RAW.CAST_TO_BINARY_INTEGER (
  r           IN  RAW,
  endianess  IN  INTEGER)
  RETURN BINARY_INTEGER;
  ```

  表 15-27 UTL_RAW.CAST_TO_BINARY_INTEGER 接口参数说明

参数	描述
r	二进制表示形式的整型值（即RAW类型）。
endianess	表示字节序的整型值1或2（1代表BIG_ENDIAN，2代表LITTLE-ENDIAN）。

- UTL_RAW.LENGTH

  存储过程LENGTH返回RAW类型对象的长度。

  UTL_RAW.LENGTH函数原型为：

  ```
  UTL_RAW.LENGTH(
  r      IN RAW)
  RETURN INTEGER;
  ```

  表 15-28 UTL_RAW.LENGTH 接口参数说明

参数	描述
r	RAW类型对象

- UTL_RAW.CAST_TO_RAW

  存储过程CAST_TO_RAW将VARCHAR2类型的对象转换成RAW类型。

  UTL_RAW.CAST_TO_RAW函数原型为：

  ```
  UTL_RAW.CAST_TO_RAW(
  c      IN VARCHAR2)
  RETURN RAW;
  ```

表 15-29 UTL_RAW.CAST_TO_RAW 接口参数说明

参数	描述
c	待转换的VARCHAR2类型对象

示例

```
--在存储过程中操作RAW数据
CREATE OR REPLACE PROCEDURE proc_raw
AS
str varchar2(100) := 'abcdef';
source raw(100);
amount integer;
BEGIN
source := utl_raw.cast_to_raw(str);--类型转换
amount := utl_raw.length(source);--获取长度
dbms_output.put_line(amount);
END;
/

--调用存储过程
CALL proc_raw();

--删除存储过程
DROP PROCEDURE proc_raw;
```

15.11.5 DBMS_JOB

接口介绍

高级功能包DBMS_JOB支持的所有接口请参见**表15-30**。

表 15-30 DBMS_JOB

接口名称	描述
DBMS_JOB.SUBMIT	提交一个定时任务。作业号由系统自动生成。
DBMS_JOB.ISUBMIT	提交一个定时任务。作业号由用户指定。
DBMS_JOB.REMOVE	通过作业号来删除定时任务。
DBMS_JOB.BROKEN	禁用或者启用定时任务。
DBMS_JOB.CHANGE	修改定时任务的属性，包括任务内容、下次执行时间、执行间隔。
DBMS_JOB.WHAT	修改定时任务的任务内容属性。
DBMS_JOB.NEXT_DATE	修改定时任务的下次执行时间属性。
DBMS_JOB.INTERVAL	修改定时任务的执行间隔属性。

- DBMS_JOB.SUBMIT

  存储过程SUBMIT提交一个系统提供的定时任务。

  DBMS_JOB.SUBMIT函数原型为：

```
DMBS_JOB.SUBMIT(
what         IN   TEXT,
next_date    IN   TIMESTAMP DEFAULT sysdate,
job_interval IN   TEXT  DEFAULT 'null',
job          OUT  INTEGER);
```

  说明

  当创建一个定时任务（DBMS_JOB）时，系统默认将当前数据库和用户名与当前创建的定时任务（DBMS_JOB）绑定起来。该接口函数可以通过call或select调用，如果通过select调用，可以不填写出参。如果在存储过程中则需要用通过perform调用该接口函数。

  表 15-31 DBMS_JOB.SUBMIT 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
what	text	IN	否	要执行的SQL语句。支持一个或多个‘DML’，‘匿名块’，‘调用存储过程的语句’或3种混合的场景。
next_date	timestamp	IN	否	下次作业运行时间。默认值为当前系统时间（sysdate）。如果是过去时间，在提交作业时表示立即执行。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式，可以是interval表达式，也可以是sysdate加上一个numeric值（例如：sysdate+1.0/24）。如果为空值或字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。
job	integer	OUT	否	作业号。范围为1～32767。当使用select调用dbms.submit时，该参数可以省略。

  示例：

```
select DBMS_JOB.SUBMIT('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1');

select DBMS_JOB.SUBMIT('call pro_xxx();', to_date('20180101','yyyymmdd'),'sysdate+1.0/24');

CALL DBMS_JOB.SUBMIT('INSERT INTO T_JOB  VALUES(1);  call pro_1(); call pro_2();',
add_months(to_date('201701','yyyymm'),1), 'date_trunc(''day'',SYSDATE) + 1 +(8*60+30.0)/
(24*60)' ,:jobid);
```

- DBMS_JOB.ISUBMIT

  ISUBMIT与SUBMIT语法功能相同，但其第一个参数是入参，即指定的作业号，SUBMIT最后一个参数是出参，表示系统自动生成的作业号。

  示例：

```
CALL dbms_job.isubmit(101, 'insert_msg_statistic1;', sysdate, 'sysdate+3.0/24');
```

- DBMS_JOB.REMOVE

  存储过程REMOVE删除指定的定时任务。

  DBMS_JOB.REMOVE函数原型为：

  ```
  REMOVE(job  IN  INTEGER);
  ```

  表 15-32 DBMS_JOB.REMOVE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。

  示例：

  ```
  CALL dbms_job.remove(101);
  ```

- DBMS_JOB.BROKEN

  存储过程BROKEN禁用或者启用定时任务。

  DBMS_JOB.BROKEN函数原型为：

  ```
  DMBS_JOB.BROKEN(
  job          IN   INTEGER,
  broken       IN   BOOLEAN,
  next_date    IN   TIMESTAMP  DEFAULT  sysdate);
  ```

  表 15-33 DBMS_JOB.BROKEN 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
broken	boolean	IN	否	状态标志位，true代表禁用，false代表启用。具体true或false值更新当前job；如果为空值，则不改变原有job的状态。
next_date	timestamp	IN	是	下次运行时间，默认为当前系统时间。如果参数broken状态为true，则更新该参数为'4000-1-1'；如果参数broken状态为false，且如果参数next_date不为空值，则更新指定job的next_date值，如果next_date为空值，则不更新next_date值。该参数可以省略，为默认值。

  示例：

```
CALL dbms_job.broken(101, true);
CALL dbms_job.broken(101, false, sysdate);
```

- DBMS_JOB.CHANGE

  存储过程CHANGE修改定时任务的属性，包括任务内容、下次执行时间、执行间隔。

  DBMS_JOB.CHANGE函数原型为：

```
DMBS_JOB.CHANGE(
job          IN   INTEGER,
what         IN   TEXT,
next_date    IN   TIMESTAMP,
interval     IN   TEXT);
```

  表 15-34 DBMS_JOB.CHANGE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
what	text	IN	是	执行的存储过程名或者sql语句块。如果该参数为空值，则不更新指定job的what值，否则更新指定job的what值。
next_date	timestamp	IN	是	下次运行时间。如果该参数为空值，则不更新指定job的next_date值，否则更新指定job的next_date值。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式。如果该参数为空值，则不更新指定job的interval值；如果该参数不为空值，会校验interval是否为有效的时间类型或interval类型，则更新指定job的interval值。如果为字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。

  示例：

```
CALL dbms_job.change(101, 'call userproc();', sysdate, 'sysdate + 1.0/1440');
CALL dbms_job.change(101, 'insert into tbl_a values(sysdate);', sysdate, 'sysdate + 1.0/1440');
```

- DBMS_JOB.WHAT

  存储过程WHAT修改定时任务的任务内容属性。

  DBMS_JOB.WHAT函数原型为：

```
DMBS_JOB.WHAT(
job            IN     INTEGER,
what           IN     TEXT);
```

表 15-35 DBMS_JOB.WHAT 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
what	text	IN	否	执行的存储过程调用或者sql语句块。

说明

- 当what参数是一个或多个可以执行成功的sql语句/程序块/调用存储过程时，该接口函数才能被执行成功，否则会执行失败。
- 若what参数为一个简单的insert、update等语句，需要在表前加模式名。

示例：

```
CALL dbms_job.what(101, 'call userproc();');
CALL dbms_job.what(101, 'insert into tbl_a values(sysdate);');
```

- DBMS_JOB.NEXT_DATE

  存储过程NEXT_DATE修改定时任务的下次执行时间属性。

  DBMS_JOB.NEXT_DATE函数原型为：

```
DMBS_JOB.NEXT_DATE(
job         IN    INTEGER,
next_date   IN    TIMESTAMP);
```

表 15-36 DBMS_JOB.NEXT_DATE 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
next_date	timestamp	IN	否	下次运行时间。

说明

如果输入的next_date的值小于当前日期值，该job会立即执行一次。

示例：

```
CALL dbms_job.next_date(101, sysdate);
```

- DBMS_JOB.INTERVAL

  存储过程INTERVAL修改定时任务的执行间隔属性。

  DBMS_JOB.INTERVAL函数原型为：

```
DMBS_JOB.INTERVAL(
job             IN   INTEGER,
interval        IN   TEXT);
```

表 15-37 DBMS_JOB.INTERVAL 接口参数说明

参数	类型	入参/出参	是否可以为空	描述
job	integer	IN	否	指定的作业号。
interval	text	IN	是	用来计算下次作业运行时间的时间表达式。如果为空值或字符串"null"表示只执行一次，执行后JOB状态STATUS变成'd' 不再执行。interval是否为有效的时间类型或interval类型。

示例：

```
CALL dbms_job.interval(101, 'sysdate + 1.0/1440');
```

说明

对于指定job正在运行状态（即job_status为'r'）时，不允许通过remove、change、next_date、what、interval等接口删除或修改job的参数信息。

约束说明

1. 创建一个新job后，该job从属于当前coordinator（即：该job仅在当前coordinator上调度和执行），其他coordinator不会调度和执行该job。所有coordinator都可以查看、修改、删除其他CN创建的job。
2. job只能通过dbms_job高级包提供的接口进行创建、更新、删除操作，因为高级包的接口中会考虑所有CN间job信息的同步和pg_job与pg_job_proc表主键的关联操作，如果通过DML语句对pg_job表进行增删改，会导致job信息在CN间不一致和系统表无法关联变更的混乱问题，会严重影响job内部的管理。
3. 由于用户创建的每个任务和CN绑定，当任务运行过程中，该CN故障，则该任务的状态无法实时刷新，仍为’r’状态，需要等CN启动正常后才能刷新为’s’状态。如果在任务未执行时CN故障，则该CN上的任务都得不到正常的调度和执行，需要人为干预让该CN恢复正常，或进行节点删除/替换、job才能正常的调度和执行。
4. job在定时执行过程中，需要在当前job所属的CN上实时更新该job的运行状态、最近执行开始时间、最近执行结束时间、下次开始时间、失败次数（如果job执行失败）等相关参数信息到pg_job系统表中，并同步到其他CN，保证job信息的一致性。如果其他CN存在节点故障，那么job所属CN会同步超时重发的处理，导致job执行时间变长，但CN间同步超时失败后，原CN上pg_job表中job的相关信息仍然能正常更新，且job能正常执行成功。当故障CN恢复正常后，可能出现该CN上pg_job表中当前job的执行时间、运行状态等参数与原CN上不一致的情况，需要原CN上再次执行该job后才能保证job信息的同步。
5. 对于并发同时有多个job到达执行时间的场景，由于会为每个job创建一个线程来执行job，由于系统内部启动每个线程的时间会有延迟，因此会导致同时并发执行的job的开始时间有延迟，每个job的延迟时间在0.1ms左右。

15.11.6 DBMS_SQL

接口介绍

高级功能包DBMS_SQL支持的接口请参见**表1 DBMS_SQL**。

表 15-38 DBMS_SQL

接口名称	描述
DBMS_SQL.OPEN_CURSOR	打开一个游标。
DBMS_SQL.CLOSE_CURSOR	关闭一个已打开的游标。
DBMS_SQL.PARSE	向游标传递一组SQL语句，目前只支持SELECT。
DBMS_SQL.EXECUTE	在游标上执行一组动态定义操作。
DBMS_SQL.FETCHE_ROWS	读取游标一行数据。
DBMS_SQL.DEFINE_COLUMN	动态定义一个列。
DBMS_SQL.DEFINE_COLUMN_CHAR	动态定义一个char类型的列。
DBMS_SQL.DEFINE_COLUMN_INT	动态定义一个int类型的列。
DBMS_SQL.DEFINE_COLUMN_LONG	动态定义一个long类型的列。
DBMS_SQL.DEFINE_COLUMN_RAW	动态定义一个raw类型的列。
DBMS_SQL.DEFINE_COLUMN_TEXT	动态定义一个text类型的列。
DBMS_SQL.DEFINE_COLUMN_UNKNOWN	动态定义一个未知列（类型不识别时入此接口）。
DBMS_SQL.COLUMN_VALUE	读取一个已动态定义的列值。
DBMS_SQL.COLUMN_VALUE_CHAR	读取一个已动态定义的列值（指定char类型）。
DBMS_SQL.COLUMN_VALUE_INT	读取一个已动态定义的列值（指定int类型）。
DBMS_SQL.COLUMN_VALUE_LONG	读取一个已动态定义的列值（指定long类型）。
DBMS_SQL.COLUMN_VALUE_RAW	读取一个已动态定义的列值（指定raw类型）。
DBMS_SQL.COLUMN_VALUE_TEXT	读取一个已动态定义的列值（指定text类型）。
DBMS_SQL.COLUMN_VALUE_UNKNOWN	读取一个已动态定义的列值（类型不识别时入此接口）。
DBMS_SQL.IS_OPEN	检查游标是否已打开。

说明

- 建议使用dbms_sql.define_column及dbms_sql.column_value定义参数列。
- 当结果集大于work_mem设定值时会触发结果集临时下盘，但最大阈值不超过512MB。

- DBMS_SQL.OPEN_CURSOR

  该函数用来打开一个游标，是后续dbms_sql各项操作的前提。该函数不传入任何参数，内部自动递增生成游标ID，并作为返回值返回给integer定义的变量。

  DBMS_SQL.OPEN_CURSOR函数原型为：

  ```
  DBMS_SQL.OPEN_CURSOR (
  )
  RETURN INTEGER;
  ```

- DBMS_SQL.CLOSE_CURSOR

  该函数用来关闭一个游标，是dbms_sql各项操作的结束。如果在存储过程结束时没有调用该函数，则该游标占用的内存仍然会保存，因此关闭游标非常重要。由于异常情况的发生会中途退出存储过程，导致游标未能关闭，因此建议存储过程中有异常处理，将该接口包含在内。

  DBMS_SQL.CLOSE_CURSOR函数原型为：

  ```
  DBMS_SQL.CLOSE_CURSOR (
  cursorid     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 15-39 DBMS_SQL.CLOSE_CURSOR 接口说明

参数名称	描述
cursorid	打算关闭的游标ID号

- DBMS_SQL.PARSE

  该函数用来解析给定游标的查询语句，被传入的查询语句会立即执行。目前仅支持SELECT查询语句的解析，且语句参数仅可通过text类型传递，长度不大于1G。

  DBMS_SQL.PARSE函数的原型为：

  ```
  DBMS_SQL.PARSE (
  cursorid     IN INTEGER,
  query_string IN TEXT,
  label        IN INTEGER
  )
  RETURN BOOLEAN;
  ```

  表 15-40 DBMS_SQL.PARSE 接口说明

参数名称	描述
cursorid	执行查询语句解析的游标ID
query_string	执行的查询语句
language_flag	版本语言号，目前只支持1

- DBMS_SQL.EXECUTE

  该函数用来执行一个给定的游标。该函数接收一个游标ID，运行后获得的数据用于后续操作。目前仅支持SELECT查询语句的执行。

DBMS_SQL.EXECUTE函数的原型为：

```
DBMS_SQL.EXECUTE(
cursorid     IN INTEGER,
)
RETURN INTEGER;
```

表 15-41 DBMS_SQL.EXECUTE 接口说明

参数名称	描述
cursorid	执行查询语句解析的游标ID

- DBMS_SQL.FETCHE_ROWS

  该函数返回符合查询条件的数据行数，每一次运行该接口都会获取到新的行数的集合，直到数据读取完毕获取不到新行为止。

  DBMS_SQL.FETCHE_ROWS函数的原型为：

  ```
  DBMS_SQL.FETCHE_ROWS(
  cursorid     IN INTEGER,
  )
  RETURN INTEGER;
  ```

  表 15-42 DBMS_SQL.FETCH_ROWS 接口说明

参数名称	描述
curosorid	执行的游标ID

- DBMS_SQL.DEFINE_COLUMN

  该函数用来定义从给定游标返回的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN(
  cursorid     IN INTEGER,
  position     IN INTEGER,
  column_ref   IN ANYELEMENT,
  column_size      IN INTEGER default 1024
  )
  RETURN INTEGER;
  ```

  表 15-43 DBMS_SQL.DEFINE_COLUMN 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_ref	任意类型的变量，可根据变量类型选择适当的接口动态定义列
column_size	定义的列的长度

- DBMS_SQL.DEFINE_COLUMN_CHAR

该函数用来定义从给定游标返回的CHAR类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

DBMS_SQL.DEFINE_COLUMN_CHAR函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_CHAR(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN TEXT,
column_size     IN INTEGER
)
RETURN INTEGER;
```

表 15-44 DBMS_SQL.DEFINE_COLUMN_CHAR 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column	需要定义的某类型的参数变量
column_size	动态定义列长度

- DBMS_SQL.DEFINE_COLUMN_INT

  该函数用来定义从给定游标返回的INT类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_INT函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN_INT(
  cursorid     IN INTEGER,
  position     IN INTEGER
  )
  RETURN INTEGER;
  ```

  表 15-45 DBMS_SQL.DEFINE_COLUMN_INT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

- DBMS_SQL.DEFINE_COLUMN_LONG

  该函数用来定义从给定游标返回的长列类型（非数据类型long）的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。长列的大小限制为1G。

  DBMS_SQL.DEFINE_COLUMN_LONG函数的原型为：

  ```
  DBMS_SQL.DEFINE_COLUMN_LONG(
  cursorid     IN INTEGER,
  position     IN INTEGER
  )
  RETURN INTEGER;
  ```

表 15-46 DBMS_SQL.DEFINE_COLUMN_LONG 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

- DBMS_SQL.DEFINE_COLUMN_RAW

  该函数用来定义从给定游标返回的RAW类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_RAW函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_RAW(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN BYTEA,
column_size      IN INTEGER
)
RETURN INTEGER;
```

表 15-47 DBMS_SQL.DEFINE_COLUMN_RAW 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column	RAW类型的参数变量
column_size	列的长度

- DBMS_SQL.DEFINE_COLUMN_TEXT

  该函数用来定义从给定游标返回的TEXT类型的列，该接口只能应用于SELECT定义的游标中。定义的列通过查询列表的相对位置来标识，传入变量的数据类型决定了该列被定义的类型。

  DBMS_SQL.DEFINE_COLUMN_TEXT函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_CHAR(
cursorid     IN INTEGER,
position     IN INTEGER,
max_size     IN INTEGER
)
RETURN INTEGER;
```

表 15-48 DBMS_SQL.DEFINE_COLUMN_TEXT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
max_size	定义的TEXT类型的最大长度

- DBMS_SQL.DEFINE_COLUMN_UNKNOWN

  该函数用来处理从给定游标返回的未知数据类型的列，该接口仅用于类型不识别时的报错退出。

  DBMS_SQL.DEFINE_COLUMN_UNKNOWN函数的原型为：

```
DBMS_SQL.DEFINE_COLUMN_CHAR(
cursorid     IN INTEGER,
position     IN INTEGER,
column       IN TEXT
)
RETURN INTEGER;
```

  表 15-49 DBMS_SQL.DEFINE_COLUMN_UNKNOWN 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column	动态定义的参数

- DBMS_SQL.COLUMN_VALUE

  该函数用来返回给定游标给定位置的游标元素值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE函数的原型为：

```
DBMS_SQL.COLUMN_VALUE(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT ANYELEMENT
)
RETURN ANYELEMENT;
```

  表 15-50 DBMS_SQL.COLUMN_VALUE 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_value	定义的列的返回值

- DBMS_SQL.COLUMN_VALUE_CHAR

  该函数用来返回给定游标给定位置的游标CHAR类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_CHAR函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_CHAR(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT CHARACTER,
err_num               INOUT NUMERIC default 0,
actual_length           INOUT INTEGER default 1024
)
RETURN RECORD;
```

表 15-51 DBMS_SQL.COLUMN_VALUE_CHAR 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_value	返回值
err_num	错误号。传出参数，须传入变量做参数。目前未实现，固定传出-1。
actual_length	返回值的实际长度

- DBMS_SQL.COLUMN_VALUE_INT

  该函数用来返回给定游标给定位置的游标INT类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。DBMS_SQL.COLUMN_VALUE_INT函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_INT(
cursorid              IN    INTEGER,
position              IN    INTEGER
)
RETURN INTEGER;
```

表 15-52 DBMS_SQL.COLUMN_VALUE_INT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

- DBMS_SQL.COLUMN_VALUE_LONG

  该函数用来返回给定游标给定位置的游标长列（非long/bigint整型）类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_LONG函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_LONG(
cursorid              IN    INTEGER,
position              IN    INTEGER,
length                IN    INTEGER,
off_set               IN    INTEGER,
column_value          INOUT TEXT,
actual_length         INOUT INTEGER default 1024
)
RETURN RECORD;
```

表 15-53 DBMS_SQL.COLUMN_VALUE_LONG 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
length	返回值的长度

参数名称	描述
off_set	返回值的起始位置
column_value	返回值
actual_length	实际返回值的长度

- DBMS_SQL.COLUMN_VALUE_RAW

  该函数用来返回给定游标给定位置的游标RAW类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_RAW函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_RAW(
cursorid              IN    INTEGER,
position              IN    INTEGER,
column_value            INOUT BYTEA,
err_num               INOUT NUMERIC default 0,
actual_length           INOUT INTEGER default 1024
)
RETURN RECORD;
```

  表 15-54 DBMS_SQL.COLUMN_VALUE_RAW 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_value	返回的列值
err_num	错误号。传出参数，须传入变量做参数。目前未实现，固定传出-1。
actual_length	返回值的实际长度，不能长于此值，否则截断

- DBMS_SQL.COLUMN_VALUE_TEXT

  该函数用来返回给定游标给定位置的游标TEXT类型的值，该接口访问的是DBMS_SQL.FETCH_ROWS获取的数据。

  DBMS_SQL.COLUMN_VALUE_TEXT函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_TEXT(
cursorid              IN    INTEGER,
position              IN    INTEGER
)
RETURN TEXT;
```

  表 15-55 DBMS_SQL.COLUMN_VALUE_TEXT 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置

- DBMS_SQL.COLUMN_VALUE_UNKNOWN

  该函数用来返回给定游标给定位置的游标未知类型的值，该接口为类型不支持时的报错处理接口。

  DBMS_SQL.COLUMN_VALUE_UNKNOWN函数的原型为：

```
DBMS_SQL.COLUMN_VALUE_UNKNOWN(
cursorid              IN    INTEGER,
position              IN    INTEGER,
COLUMN_TYPE           IN    TEXT
)
RETURN TEXT;
```

  表 15-56 DBMS_SQL.COLUMN_VALUE_UNKNOWN 接口说明

参数名称	描述
cursorid	执行的游标ID
position	动态定义列在查询中的位置
column_type	返回的参数类型

- DBMS_SQL.IS_OPEN

该函数用来返回游标的当前状态：打开、解析、执行、定义。取值是为TRUE，关闭后为FALSE，未知时报错，其余默认为关闭。

DBMS_SQL.IS_OPEN函数的原型为：

```
DBMS_SQL.IS_OPEN(
cursorid              IN    INTEGER
)
RETURN BOOLEAN;
```

表 15-57 DBMS_SQL.IS_OPEN 接口说明

参数名称	描述
cursorid	被查询的游标ID

示例

```
--在存储过程中操作raw数据
create or replace procedure pro_dbms_sql_all_02(in_raw raw,v_in int,v_offset int)
as
cursorid int;
v_id int;
v_info bytea :=1;
query varchar(2000);
execute_ret int;
define_column_ret_raw bytea :='1';
define_column_ret int;
begin
drop table if exists pro_dbms_sql_all_tb1_02 ;
create table pro_dbms_sql_all_tb1_02(a int ,b blob);
insert into pro_dbms_sql_all_tb1_02 values(1,HEXTORAW('DEADBEEE'));
insert into pro_dbms_sql_all_tb1_02 values(2,in_raw);
query := 'select * from pro_dbms_sql_all_tb1_02 order by 1';
--打开游标
cursorid := dbms_sql.open_cursor();
--编译游标
```

```
dbms_sql.parse(cursorid, query, 1);
--定义列
define_column_ret:= dbms_sql.define_column(cursorid,1,v_id);
define_column_ret_raw:= dbms_sql.define_column_raw(cursorid,2,v_info,10);
--执行
execute_ret := dbms_sql.execute(cursorid);
loop
exit when (dbms_sql.fetch_rows(cursorid) <= 0);
--获取值
dbms_sql.column_value(cursorid,1,v_id);
dbms_sql.column_value_raw(cursorid,2,v_info,v_in,v_offset);
--输出结果
dbms_output.put_line('id:'|| v_id || ' info:' || v_info);
end loop;
--关闭游标
dbms_sql.close_cursor(cursorid);
end;
/
--调用存储过程
call pro_dbms_sql_all_02(HEXTORAW('DEADBEEF'),0,1);

--删除存储过程
DROP PROCEDURE pro_dbms_sql_all_02;
```

15.12 调试

语法

RAISE有以下五种语法格式：

图 15-34 raise_format::=

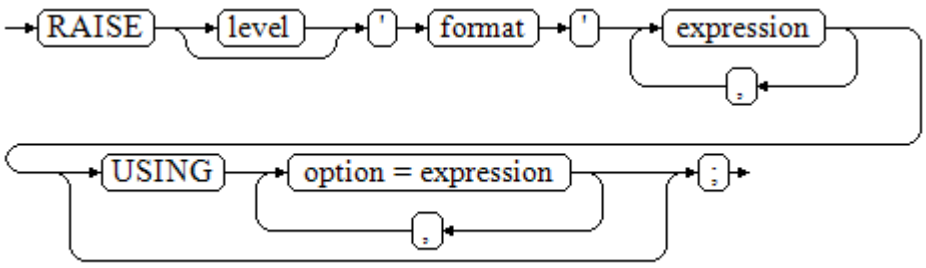


图 15-35 raise_condition::=

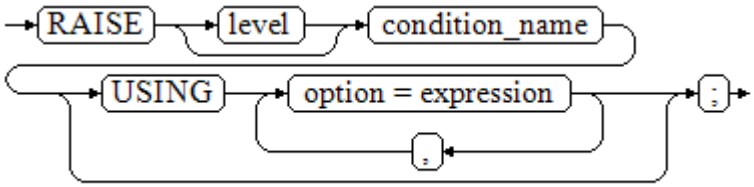


图 15-36 raise_sqlstate::=

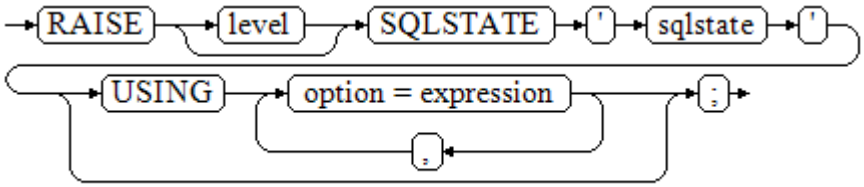


图 15-37 raise_option::=

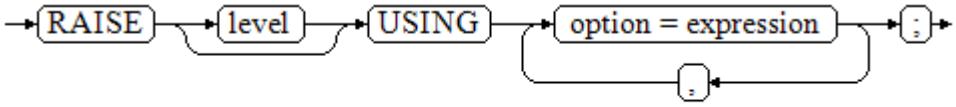
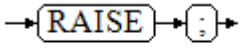


图 15-38 raise::=



参数说明：

- level选项用于指定错误级别，有DEBUG，LOG，INFO，NOTICE，WARNING以及EXCEPTION（默认值）。EXCEPTION抛出一个正常终止当前事务的异常，其他的仅产生不同异常级别的信息。特殊级别的错误信息是否报告到客户端、写到服务器日志由**client_min_messages**和**log_min_messages**这两个配置参数控制。
- format：格式字符串，指定要报告的错误消息文本。格式字符串后可跟表达式，用于向消息文本中插入。在格式字符串中，%由format后面跟着的参数的值替换，%%用于打印出%。例如：

  ```
  --v_job_id 将替换字符串中的 %：
  RAISE NOTICE 'Calling cs_create_job(%)',v_job_id;
  ```

- option = expression：向错误报告中添加另外的信息。关键字option可以是MESSAGE、DETAIL、HINT以及ERRCODE，并且每一个expression可以是任意的字符串。
  - MESSAGE，指定错误消息文本，这个选项不能用于在USING前包含一个格式字符串的RAISE语句中。
  - DETAIL，说明错误的详细信息。
  - HINT，用于打印出提示信息。
  - ERRCODE，向报告中指定错误码（SQLSTATE）。可以使用条件名称或者直接用五位字符的SQLSTATE错误码。
- condition_name：错误码对应的条件名。
- sqlstate：错误码。

如果在RAISE EXCEPTION命令中既没有指定条件名也没有指定SQLSTATE，默认用RAISE EXCEPTION (P0001)。如果没有指定消息文本，默认用条件名或者SQLSTATE作为消息文本。

须知

当由SQLSTATE指定了错误码，则不局限于已定义的错误码，可以选择任意包含五个数字或者大写的ASCII字母的错误码，而不是00000。建议避免使用以三个0结尾的错误码，因为这种错误码是类别码，会被整个种类捕获。

说明

图15-38所示的语法不接任何参数。这种形式仅用于一个BEGIN块中的EXCEPTION语句，它使得错误重新被处理。

示例

终止事务时，给出错误和提示信息：

```
CREATE OR REPLACE PROCEDURE proc_raise1(user_id in integer)
AS
BEGIN
RAISE EXCEPTION 'Noexistence ID --> %',user_id USING HINT = 'Please check your user ID';
END;
/

call proc_raise1(300011);

--执行结果
ERROR:  Noexistence ID --> 300011
HINT:  Please check your user ID
```

两种设置SQLSTATE的方式：

```
CREATE OR REPLACE PROCEDURE proc_raise2(user_id in integer)
AS
BEGIN
RAISE 'Duplicate user ID: %',user_id USING ERRCODE = 'unique_violation';
END;
/

\set VERBOSITY verbose
call proc_raise2(300011);

--执行结果
ERROR:  Duplicate user ID: 300011
SQLSTATE: 23505
LOCATION:  exec_stmt_raise, pl_exec.cpp:3482
```

如果主要的参数是条件名或者是SQLSTATE，可以使用：

RAISE division_by_zero;

RAISE SQLSTATE '22012';

例如：

```
CREATE OR REPLACE PROCEDURE division(div in integer, dividend in integer)
AS
DECLARE
res int;
    BEGIN
    IF dividend=0 THEN
        RAISE division_by_zero;
        RETURN;
    ELSE
        res := div/dividend;
        RAISE INFO 'division result: %', res;
        RETURN;
    END IF;
```

```
    END;
/
call division(3,0);

--执行结果
ERROR:  division_by_zero
```

或者另一种方式：

```
RAISE unique_violation USING MESSAGE = 'Duplicate user ID: ' || user_id;
```

16 系统表和系统视图

16.1 系统表和系统视图概述

系统表是GaussDB(DWS)存放结构元数据的地方，它是GaussDB(DWS)数据库系统运行控制信息的来源，是数据库系统的核心组成部分。

系统视图提供了查询系统表和访问数据库内部状态的方法。

三权分立下，非管理员无权查看系统表和视图。非三权分立下，系统表和系统视图要么只对管理员可见，要么对所有用户可见。下面的系统表和视图有些标识了需要管理员权限，这些系统表和视图只有管理员可以查询。

用户可以删除后重新创建这些表、增加列、插入和更新数值，但是用户修改系统表会导致系统信息的不一致，从而导致系统控制紊乱。正常情况下不应该由用户手工修改系统表或系统视图，或者手工重命名系统表或系统视图所在的模式，而是由SQL语句关联的系统表操作自动维护系统表信息。

须知

用户应该禁止对系统表进行增删改等操作，人为对系统表的修改或破坏可能会导致系统各种异常情况甚至集群不可用

16.2 系统表

16.2.1 GS_OBSSCANINFO

GS_OBSSCANINFO系统表定义了在云上加速场景中，使用加速集群时扫描OBS数据的运行时信息，每条记录对应一个query中单个OBS外表的运行时信息。

表 16-1 GS_OBSSCANINFO 字段

名字	类型	引用	描述
query_id	bigint	-	查询标识。

名字	类型	引用	描述
user_id	text	-	执行该查询的数据库用户。
table_name	text	-	OBS外表的表名。
file_type	text	-	底层数据保存的文件格式。
time_stamp	time_stam	-	扫描操作开始的时间。
actual_time	double	-	扫描操作执行时间，单位为秒。
file_scanned	bigint	-	扫描的文件数量。
data_size	double	-	扫描的数据量，单位为字节。
billing_info	text	-	保留字段。

16.2.2 GS_WLM_INSTANCE_HISTORY

GS_WLM_INSTANCE_HISTORY系统表存储与实例(CN或DN)相关的资源使用相关信息。该系统表里每条记录都是对应时间点某实例资源使用情况，包括：内存、CPU核数、磁盘IO、进程物理IO和进程逻辑IO信息。

表 16-2 GS_WLM_INSTANCE_HISTORY 字段

名称	类型	描述
instancename	text	实例名称。
timestamp	timestamp with time zone	时间戳。
used_cpu	int	实例使用CPU所占用的百分比。
free_mem	int	实例未使用的内存大小，单位MB。
used_mem	int	实例已使用的内存大小，单位MB。
io_await	real	实例所使用磁盘的io_wait值（10秒均值）。
io_util	real	实例所使用磁盘的io_util值（10秒均值）。
disk_read	real	实例所使用磁盘的读速率（10秒均值），单位KB/s。
disk_write	real	实例所使用磁盘的写速率（10秒均值），单位KB/s。
process_read	bigint	实例对应进程从磁盘读数据的读速率(不包括从磁盘pagecache中读取的字节数，10秒均值)，单位KB/s。

名称	类型	描述
process_write	bigint	实例对应进程向磁盘写数据的写速率(不包括向磁盘pagecache中写入的字节数，10秒均值)，单位KB/s。
logical_read	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑读字节速率，单位KB/s。
logical_write	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑写字节速率，单位KB/s。
read_counts	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑读操作次数之和，单位次。
write_counts	bigint	CN实例：不统计。 DN实例：该实例在本次统计间隙（10秒）内逻辑写操作次数之和，单位次。

16.2.3 GS_WLM_OPERATOR_INFO

GS_WLM_OPERATOR_INFO系统表显示执行作业结束后的算子相关的记录。此数据是从内核中转储到系统表中的数据。

表 16-3 GS_WLM_OPERATOR_INFO 的字段

名称	类型	描述
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程id。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。

名称	类型	描述
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. Spill file size large than 256MB 3. Broadcast size large than 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict

16.2.4 GS_WLM_SESSION_INFO

GS_WLM_SESSION_INFO系统表显示当前CN执行作业结束后的负载管理记录。此数据是从内核中转储到系统表中的数据。

16.2.5 GS_WLM_USER_RESOURCE_HISTORY

GS_WLM_USER_RESOURCE_HISTORY系统表存储与用户使用资源相关的信息，仅在CN上有效。该系统表的每条记录都是对应时间点某用户的资源使用情况，包括：内

存、CPU核数、存储空间、临时空间、算子落盘空间、逻辑IO流量、逻辑IO次数和逻辑IO速率信息。其中，内存、CPU、IO相关监控项仅记录用户复杂作业的资源使用情况。

GS_WLM_USER_RESOURCE_HISTORY系统表的数据来源于**PG_TOTAL_USER_RESOURCE_INFO**视图。

表 16-4 GS_WLM_USER_RESOURCE_HISTORY 字段

名称	类型	描述
username	text	用户名。
timestamp	timestamp with time zone	时间戳。
used_memory	int	正在使用的内存大小，单位MB。
total_memory	int	可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。
used_cpu	real	正在使用的CPU核数。
total_cpu	int	该机器节点上，用户关联控制组的CPU核数总和。
used_space	bigint	已使用的存储空间大小，单位KB。
total_space	bigint	可使用的存储空间大小，单位KB，值为-1表示未限制最大存储空间。
used_temp_space	bigint	已使用的临时存储空间大小，单位KB。
total_temp_space	bigint	可使用的临时存储空间大小，单位KB，值为-1表示未限制最大临时存储空间。
used_spill_space	bigint	已使用的算子落盘存储空间大小，单位KB。
total_spill_space	bigint	可使用的算子落盘存储空间大小，单位KB，值为-1表示未限制最大算子落盘存储空间。
read_kbytes	bigint	监控周期内，读操作的字节流量，单位KB。
write_kbytes	bigint	监控周期内，写操作的字节流量，单位KB。
read_counts	bigint	监控周期内，读操作的次数，单位次。
write_counts	bigint	监控周期内，写操作的次数，单位次。

名称	类型	描述
read_speed	real	监控周期内，读操作的字节速率，单位KB/s。
write_speed	real	监控周期内，写操作的字节速率，单位KB/s。

16.2.6 PG_AGGREGATE

PG_AGGREGATE系统表存储与聚集函数有关的信息。PG_AGGREGATE里的每条记录都是一条pg_proc里面的记录的扩展。PG_PROC记录承载该聚集的名字、输入和输出数据类型，以及其它一些和普通函数类似的信息。

表 16-5 PG_AGGREGATE 字段

名字	类型	引用	描述
aggfnoid	regproc	**PG_PROC**.oid	此聚集函数的**PG_PROC** OID。
aggtransfn	regproc	**PG_PROC**.oid	转换函数。
aggcollectfn	regproc	**PG_PROC**.oid	收集函数。
aggfinalfn	regproc	**PG_PROC**.oid	最终处理函数（如果没有则为0）。
aggsortop	oid	**PG_OPERATOR**.oid	关联排序操作符（如果没有则为0）。
aggtranstype	oid	**PG_TYPE**.oid	此聚集函数的内部转换（状态）数据的数据类型。
agginitval	text	-	转换状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则转换状态值从null开始。
agginitcollect	text	-	收集状态的初始值。这是一个文本数据域，它包含初始值的外部字符串表现形式。如果数据域是null，则收集状态值从null开始。

16.2.7 PG_AM

PG_AM系统表存储有关索引访问方法的信息。系统支持的每种索引访问方法都有一行。

表 16-6 PG_AM 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amname	name	-	访问方法的名称。
amstrategies	smallint	-	访问方法的操作符策略个数，或者如果访问方法没有一个固定的操作符策略集则为0。
amsupport	smallint	-	访问方法的支持过程个数。
amcanorder	boolean	-	这种访问方式是否支持通过索引字段值的命令扫描排序。
amcanorderbyop	boolean	-	这种访问方式是否支持通过索引字段上操作符的结果的命令扫描排序。
amcanbackward	boolean	-	访问方式是否支持向后扫描。
amcanunique	boolean	-	访问方式是否支持唯一索引。
amcanmulticol	boolean	-	访问方式是否支持多字段索引。
amoptionalkey	boolean	-	访问方式是否支持第一个索引字段上没有任何约束的扫描。
amsearcharray	boolean	-	访问方式是否支持ScalarArrayOpExpr搜索。
amsearchnulls	boolean	-	访问方式是否支持IS NULL/NOT NULL搜索。
amstorage	boolean	-	允许索引存储的数据类型与列的数据类型是否不同。
amclusterable	boolean	-	是否允许在一个这种类型的索引上集群。
ampredlocks	boolean	-	是否允许这种类型的一个索引管理细粒度的谓词锁定。
amkeytype	oid	PG_TYPE.oid	存储在索引里数据的类型，如果不是一个固定的类型则为0。
aminsert	regproc	PG_PROC.oid	“插入此行”函数。
ambeginscan	regproc	PG_PROC.oid	“准备索引扫描” 函数。
amgettuple	regproc	PG_PROC.oid	“下一个有效行”函数，如果没有则为0。
amgetbitmap	regproc	PG_PROC.oid	“抓取所有的有效行” 函数，如果没有则为0。

名字	类型	引用	描述
amrescan	regproc	PG_PROC.oid	“（重新）开始索引扫描”函数。
amendscan	regproc	PG_PROC.oid	“索引扫描后清理” 函数。
ammarkpos	regproc	PG_PROC.oid	“标记当前扫描位置”函数。
amrestrpos	regproc	PG_PROC.oid	“恢复已标记的扫描位置”函数。
ammerge	regproc	PG_PROC.oid	“归并多个索引对象”函数。
ambuild	regproc	PG_PROC.oid	“建立新索引”函数。
ambuildempty	regproc	PG_PROC.oid	“建立空索引”函数。
ambulkdelete	regproc	PG_PROC.oid	批量删除函数。
amvacuumcleanup	regproc	PG_PROC.oid	VACUUM后的清理函数。
amcanreturn	regproc	PG_PROC.oid	检查是否索引支持唯一索引扫描的函数，如果没有则为0。
amcostestimate	regproc	PG_PROC.oid	估计一个索引扫描开销的函数。
amoptions	regproc	PG_PROC.oid	用于分析和验证索引的reloptions函数。

16.2.8 PG_AMOP

PG_AMOP系统表存储有关和访问方法操作符族关联的信息。如果一个操作符是一个操作符族中的成员，则在这个表中会占据一行。一个族成员是一个search操作符或一个ordering操作符。一个操作符可以在多个族中出现，但是不能在一个族中的多个搜索位置或多个排序位置中出现。

表 16-7 PG_AMOP 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
amopfamily	oid	PG_OPFAMILY.oid	该项的操作符族。
amoplefttype	oid	PG_TYPE.oid	操作符的左输入类型。
amoprighttype	oid	PG_TYPE.oid	操作符的右输入类型。
amopstrategy	smallint	-	操作符策略数。
amoppurpose	"char"	-	操作符目的，s为搜索或o为排序。
amopopr	oid	PG_OPERATOR.oid	该操作符的OID。

名字	类型	引用	描述
amopmethod	oid	PG_AM.oid	使用此操作符族的索引访问方法。
amopsortfamily	oid	PG_OPFAMILY.oid	如果是一个排序操作符，则该项会按照btree操作符族排序；如果是一个搜索操作符，则为0。

search操作符表明这个操作符族的一个索引可以被搜索，找到所有满足WHERE indexed_column operator constant的行。显然，这样的操作符必须返回布尔值，并且它的左输入类型必须匹配索引的字段数据类型。

ordering操作符表明这个操作符族的一个索引可以被扫描，返回以ORDER BY indexed_column operator constant顺序表示的行。这样的操作符可以返回任意可排序的数据类型，它的左输入类型也必须匹配索引的字段数据类型。ORDER BY的确切语义是由amopsortfamily字段指定的，该字段必须为操作符的返回类型引用一个btree操作符族。

16.2.9 PG_AMPROC

PG_AMPROC系统表存储有关与访问方法操作符族相关联的支持过程的信息。每个属于某个操作符族的支持过程都占有一行。

表 16-8 PG_AMPROC 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
amprocfamily	oid	PG_OPFAMILY.oid	该项的操作符族
amproclefttype	oid	PG_TYPE.oid	相关操作符的左输入数据类型
amprocrighttype	oid	PG_TYPE.oid	相关操作符的右输入数据类型
amprocnum	smallint	-	支持过程编号
amproc	regproc	PG_PROC.oid	过程的OID

amproclefttype和amprocrighttype字段的习惯解释，标识一个特定支持过程所支持的操作符的左右输入类型。对于某些访问方式，匹配支持过程本身的输入数据类型，对其他的则不会匹配。有一个对索引的“缺省”支持过程的概念，amproclefttype和amprocrighttype都等于索引操作符类的opcintype。

16.2.10 PG_APP_WORKLOADGROUP_MAPPING

PG_APP_WORKLOADGROUP_MAPPING系统表提供了数据库负载映射组的信息。

表 16-9 PG_APP_WORKLOADGROUP_MAPPING 字段

名称	类型	描述
appname	name	应用名称
workload_gpname	name	映射到的负载组名称

16.2.11 PG_ATTRDEF

PG_ATTRDEF系统表存储字段的默认值。

表 16-10 PG_ATTRDEF 字段

名称	类型	描述
adrelid	oid	该字段所属的表
adnum	smallint	字段编号
adbin	pg_node_tree	字段缺省值的内部表现形式
adsrc	text	人类可读的缺省值的内部表现形式

16.2.12 PG_ATTRIBUTE

PG_ATTRIBUTE系统表存储关于表字段的信息。

表 16-11 PG_ATTRIBUTE 字段

名称	类型	描述
attrelid	oid	该字段所属的表。
attname	name	字段名。
atttypid	oid	字段类型。
attstattarget	integer	控制ANALYZE为该字段设置的统计细节的级别。 ● 零值表示不收集统计信息。 ● 负数表示使用系统缺省的统计对象。 ● 正数值的确切信息是和数据类型相关的。 对于标量数据类型，ATTSTATTARGET既是要收集的“最常用数值”的目标数目，也是要创建的柱状图的目标数量。
attlen	smallint	是本字段类型pg_type.typlen的拷贝。
attnum	smallint	字段编号。

名称	类型	描述
attndims	integer	如果该字段是数组，该值表示数组的维数，否则是0 。
attcacheoff	integer	在磁盘上总是-1 ，但是如果加载入内存中的行描述器中，它可能会被更新为缓冲在行中字段的偏移量。
atttypmod	integer	记录创建新表时支持的类型特定的数据（比如，varchar字段的最大长度）。它传递给类型相关的输入和长度转换函数当做第三个参数。其值对那些不需要ATTTYPMOD的类型通常为-1。
attbyval	boolean	pg_type.typbyval字段值的拷贝。
attstorage	"char"	pg_type.typstorage字段值的拷贝。
attalign	"char"	pg_type.typalign字段值的拷贝。
attnotnull	boolean	代表一个非空约束。可以改变这个字段来打开或者关闭该约束。
atthasdef	boolean	该字段是否存在缺省值，此时它对应pg_attrdef表里实际定义此值的记录。
attisdropped	boolean	该字段是否已经被删除，不再有效。如果被删除，该字段物理上仍然存在表中，但会被分析器忽略，因此不能再通过SQL访问。
attislocal	boolean	该字段是否局部定义在对象中。一个字段可以同时是局部定义和继承的。
attcmprmode	tinyint	对某一列指定压缩方式。压缩方式包括： ● ATT_CMPR_NOCOMPRESS ● ATT_CMPR_DELTA ● ATT_CMPR_DICTIONARY ● ATT_CMPR_PREFIX ● ATT_CMPR_NUMSTR
attinhcount	integer	该字段所拥有的直接父表的个数。如果一个字段的父表个数非零，则它就不能被删除或重命名。
attcollation	oid	对此列定义的校对列。
attacl	aclitem[]	列级访问权限控制。
attoptions	text[]	属性级可选项。
attfdwoptions	text[]	属性级外数据选项。
attinitdefval	bytea	存储了此列默认的值表达式。行存表的ADD COLUMN需要使用此字段。

16.2.13 PG_AUTHID

PG_AUTHID系统表存储有关数据库认证标识符（角色）的信息。角色把“用户”的概念包含在内。一个用户实际上就是一个rolcanlogin标志被设置的角色。任何角色（不管rolcanlogin设置与否）都能够把其他角色作为成员。

在一个集群中只有一份pg_authid，不是每个数据库有一份。需要有系统管理员权限才可以访问此系统表。

表 16-12 PG_AUTHID 字段

名称	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。
rolname	name	角色名称。
rolsuper	boolean	角色是否是拥有最高权限的初始系统管理员。
rolinherit	boolean	角色是否自动继承其所属角色的权限。
rolcreaterole	boolean	角色是否可以创建更多角色。
rolcreatedb	boolean	角色是否可以创建数据库。
rolcatupdate	boolean	角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
rolcanlogin	boolean	角色是否可以登录，即该角色是否能够作为初始会话授权标识符。
rolreplication	boolean	角色是一个复制的角色（适配作用，没有实际的功能）。
rolauditadmin	boolean	审计用户。
rolsystemadmin	boolean	管理员用户。
rolconnlimit	integer	对于可以登录的角色，限制其最大并发连接数量。-1 表示没有限制。
rolpassword	text	口令(可能是加密的)，如果没有口令，则为NULL。
rolvalidbegin	timestamp with time zone	帐户的有效开始时间，如果没有开始时间，则为NULL。
rolvaliduntil	timestamp with time zone	帐户的有效结束时间，如果没有结束时间，则为NULL。
rolrespool	name	用户所能够使用的resource pool。

名称	类型	描述
roluseft	boolean	角色是否可以操作外表。
rolparentid	oid	用户所在组用户的OID。
roltabspace	text	用户永久表存储空间限额。
rolkind	char	特殊用户种类，包括私有用户、逻辑集群管理员、普通用户。
rolnodegroup	oid	用户所关联的Node Group OID，该Node Group必须是逻辑集群。
roltempspace	text	用户临时表存储空间限额。
rolspillspace	text	用户算子落盘空间限额。
rolexcpdata	text	保留字段未使用。
rolauthinfo	text	用户采用LDAP认证时的额外信息。如果是其他认证模式，则为NULL。

16.2.14 PG_AUTH_HISTORY

PG_AUTH_HISTORY系统表记录了角色的认证历史。需要有系统管理员权限才可以访问此系统表。

表 16-13 PG_AUTH_HISTORY 字段

名称	类型	描述
roloid	oid	角色标识
passwordtime	timestamp with time zone	创建和修改密码的时间
rolpassword	text	角色密码，使用MD5、SHA256加密或者不加密

16.2.15 PG_AUTH_MEMBERS

PG_AUTH_MEMBERS系统表存储显示角色之间的成员关系。

表 16-14 PG_AUTH_MEMBERS 字段

名称	类型	描述
roleid	oid	拥有成员的角色ID
member	oid	属于ROLEID角色的成员角色ID

名称	类型	描述
grantor	oid	赋予此成员关系的角色ID
admin_option	boolean	如果有权限可以把ROLEID角色的成员关系赋予其他角色，则为真

16.2.16 PG_CAST

PG_CAST系统表存储数据类型之间的转化关系。

表 16-15 PG_CAST 字段

名称	类型	描述
castsource	oid	源数据类型的OID。
casttarget	oid	目标数据类型的OID。
castfunc	oid	转化函数的OID。0表示不需要转化函数。
castcontext	"char"	源数据类型和目标数据类型间的转化方式： ● e表示只能进行显式转化（使用CAST或::语法）。 ● i表示只能进行隐式转化。 ● a表示类型间同时支持隐式和显式转化。
castmethod	"char"	转化方法： ● f表示使用castfunc字段中指定的函数进行转化。 ● b表示类型间是二进制强制转化，不使用castfunc。

16.2.17 PG_CLASS

PG_CLASS系统表存储数据库对象信息及其之间的关系。

表 16-16 PG_CLASS 字段

名称	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）。
relname	name	表、索引、视图等对象的名称。
relnamespace	oid	包含该关系的命名空间的OID。
reltype	oid	对应该表的行类型的数据类型（索引为零，因为索引没有pg_type记录）。
reloftype	oid	复合类型的OID，0表示其他类型。

名称	类型	描述
relowner	oid	关系所有者。
relam	oid	如果行是索引，则就是所用的访问模式（B-tree，hash等）。
relfilenode	oid	该关系在磁盘上的文件的名称，如果没有则为0。
reltablespace	oid	该关系存储所在的表空间。如果为0，则使用该数据库的缺省表空间。如果关系无磁盘文件，该字段无意义。
relpages	double precision	以页(大小为BLCKSZ)为单位的此表在磁盘上的大小，只是优化器使用的一个近似值。
reltuples	double precision	表中行的数目，只是优化器使用的一个估计值。
relallvisible	integer	被标识为全可见的表中的页数。此字段是优化器用来做SQL执行优化使用的。VACUUM、ANALYZE和一些DDL语句（例如，CREATE INDEX）会引起此字段更新。
reltoastrelid	oid	与此表关联的TOAST表的OID ，如果没有则为0。 TOAST表在一个从属表里“离线”存储大字段。
reltoastidxid	oid	对于TOAST表是它的索引的OID，如果不是TOAST表则为0。
reldeltarelid	oid	Delta表的OID。 Delta表附属于列存表。用于存储数据导入过程中的甩尾数据。
reldeltaidx	oid	Delta表的索引表OID。
relcudescrelid	oid	CU描述表的OID。 CU描述表（Desc表）附属于列存表。用于控制表目录中存储数据的可见性。
relcudescidx	oid	CU描述表的索引表OID。
relhasindex	boolean	如果对象是一个表且至少有（或者最近建有）一个索引，则为真。 由CREATE INDEX设置，但DROP INDEX不会立即将它清除。如果VACUUM进程检测一个表没有索引，会清理relhasindex字段，将relhasindex值设置为假。
relisshared	boolean	如果该表在整个集群中由所有数据库共享则为真。只有某些系统表（比如pg_database）是共享的。

名称	类型	描述
relpersistence	"char"	• p表示永久表。 • u表示非日志表。 • t表示临时表。
relkind	"char"	• r表示普通表。 • i表示索引。 • S表示序列。 • v表示视图。 • c表示复合类型。 • t表示TOAST表。 • f表示外表。
relnatts	smallint	关系中用户字段数目（除了系统字段以外）。在pg_attribute里肯定有相同数目对应行。
relchecks	smallint	表上检查约束的数目，参见**PG_CONSTRAINT**。
relhasoids	boolean	如果为关系中每行都生成一个OID，则为真。
relhaspkey	boolean	如果该表有一个（或曾有）主键，则为真。
relhasrules	boolean	如果表有规则，则为真。是否有规则可参考系统表**PG_REWRITE**。
relhastriggers	boolean	如果表有（或曾有）触发器，则为真。参见**PG_TRIGGER**。
relhassubclass	boolean	如果表有（或曾有）任何继承的子表，则为真。
relcmprs	tinyint	表示是否启用表的压缩特性。需要特别注意，当且仅当批量插入才会触发压缩，普通的CRUD并不能够触发压缩。 • 0表示其他不支持压缩的表（主要是指系统表，不支持压缩属性的修改操作）。 • 1表示表数据的压缩特性为NOCOMPRESS或者无指定关键字。 • 2表示表数据的压缩特性为COMPRESS。
relhasclusterkey	boolean	是否有局部聚簇存储。
relrowmovement	boolean	针对分区表进行update操作时，是否允许行迁移。 • true：表示允许行迁移。 • false：表示不允许行迁移。
parttype	"char"	表或者索引是否具有分区表的性质。 • p表示带有分区表性质。 • n表示没有分区表特性。 • v表示该表为HDFS的Value分区表。

名称	类型	描述
relfrozenxid	xid32	该表中所有在此之间的事务ID已经被替换为一个固定的（"frozen"）事务ID。该字段用于跟踪表是否需要为了防止事务ID重叠（或者允许收缩pg_clog）而进行清理。如果该关系不是表则为0（InvalidTransactionId）。 为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。
relacl	aclitem[]	访问权限。 查询的回显结果为以下形式： rolename=xxxx/yyyy --赋予一个角色的权限 =xxxx/yyyy --赋予public的权限 xxxx表示赋予的权限，yyyy表示授予该权限的角色。权限的参数说明请参见表16-17。
reloptions	text[]	索引的访问方法，使用"keyword=value"格式的字符串。
relfrozenxid64	xid	该表中所有在此之前的事务ID已经被替换为一个固定的（"frozen"）事务ID。该字段用于跟踪表是否需要为了防止事务ID重叠（或者允许收缩pg_clog）而进行清理。如果该关系不是表则为0（InvalidTransactionId）。

表 16-17 权限的参数说明

参数	参数说明
r	SELECT（读）
w	UPDATE（写）
a	INSERT（插入）
d	DELETE
D	TRUNCATE
x	REFERENCES
t	TRIGGER
X	EXECUTE
U	USAGE
C	CREATE
c	CONNECT
T	TEMPORARY
arwdDxt	ALL PRIVILEGES（用于表）

参数	参数说明
*	给前面权限的授权选项

16.2.18 PG_COLLATION

PG_COLLATION系统表描述可用的排序规则，本质上从一个SQL名字映射到操作系统本地类别。

表 16-18 PG_COLLATION 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
collname	name	-	排序规则名（每个命名空间和编码唯一）
collnamespace	oid	PG_NAMESPACE.oid	包含该排序规则的命名空间的OID
collowner	oid	PG_AUTHID.oid	排序规则的所有者
collencoding	integer	-	排序规则可用的编码，如果适用于任意编码为-1
collcollate	name	-	排序规则对象的LC_COLLATE
collctype	name	-	排序规则对象的LC_CTYPE

16.2.19 PG_CONSTRAINT

PG_CONSTRAINT系统表存储表上的检查约束、主键、唯一约束和外键约束。

表 16-19 PG_CONSTRAINT 字段

名称	类型	描述
conname	name	约束名称（不一定是唯一的）。
connamespace	oid	包含约束的命名空间的OID。
contype	"char"	• c = 检查约束 • f = 外键约束 • p = 主键约束 • u = 唯一约束 • t = 触发器约束
condeferrable	boolean	该约束是否可以推迟。

名称	类型	描述
condeferred	boolean	缺省时该约束是否可以推迟。
convalidated	boolean	约束是否有效。目前，只有外键和CHECK约束可将其设置为FALSE。
conrelid	oid	该约束所在的表；如果不是表约束则为0。
contypid	oid	该约束所在的域；如果不是一个域约束则为0。
conindid	oid	与约束关联的索引ID。
confrelid	oid	如果是外键，则为参考的表；否则为0。
confupdtype	"char"	外键更新动作代码。 ● a = 没动作 ● r = 限制 ● c = 级联 ● n =设置为null ● d =设置为缺省
confdeltype	"char"	外键删除动作代码。 ● a = 没动作 ● r = 限制 ● c = 级联 ● n =设置为null ● d =设置为缺省
confmatchtype	"char"	外键匹配类型。 ● f = 全部 ● p = 部分 ● u = 简单（未指定）
conislocal	boolean	是否是为关系创建的本地约束。
coninhcount	integer	约束直接继承父表的数目。继承父表数非零时，不能删除或重命名该约束。
connoinherit	boolean	是否可以被继承。
consoft	boolean	是否为信息约束(Informational Constraint)。
conopt	boolean	是否使用信息约束优化执行计划。
conkey	smallint[]	如果是表约束，则是约束控制的字段列表。
confkey	smallint[]	如果是一个外键，则是参考的字段的列表。
conpfeqop	oid[]	如果是一个外键，是做PK=FK比较的相等操作符ID的列表。

名称	类型	描述
conppeqop	oid[]	如果是一个外键，是做PK=PK比较的相等操作符ID的列表。
conffeqop	oid[]	如果是一个外键，是做FK=FK比较的相等操作符ID的列表。
conexclop	oid[]	如果是一个排他约束，是列的排他操作符ID列表。
conbin	pg_node_tree	如果是检查约束，则是其表达式的内部形式。
consrc	text	如果是检查约束，则是表达式的人类可读形式。

须知

- 当被引用的对象改变时，consrc不能被更新。例如，它不会跟踪字段的重命名。最好还是使用pg_get_constraintdef()来抽取一个检查约束的定义，而不是依赖这个字段。
- pg_class.relchecks需要和每个关系在此目录中的检查约束数量保持一致。

16.2.20 PG_CONVERSION

PG_CONVERSION系统表描述编码转换信息。

表 16-20 PG_CONVERSION 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
conname	name	-	转换名（在一个命名空间里唯一）
connamespace	oid	PG_NAMESPACE.oid	包含此转换的命名空间的OID
conowner	oid	PG_AUTHID.oid	编码转换的属主
conforencoding	integer	-	源编码ID
contoencoding	integer	-	目的编码ID
conproc	regproc	PG_PROC.oid	转换过程
condefault	boolean	-	如果为缺省转换则为真

16.2.21 PG_DATABASE

PG_DATABASE系统表存储关于可用数据库的信息。

表 16-21 PG_DATABASE 字段

名称	类型	描述
datname	name	数据库名称。
datdba	oid	数据库所有者，通常为其创建者。
encoding	integer	数据库的字符编码方式。
datcollate	name	数据库使用的排序顺序。
datctype	name	数据库使用的字符分类。
datistemplate	boolean	是否允许作为模板数据库。
datallowconn	boolean	如果为假，则没有用户可以连接到这个数据库。这个字段用于保护template0数据库不被更改。
datconnlimit	integer	该数据库上允许的最大并发连接数，-1表示无限制。
datlastsysoid	oid	数据库里最后一个系统OID 。
datfrozenxid	xid32	用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。 为保持前向兼容，保留此字段，新增datfrozenxid64用于记录此信息。
dattablespace	oid	数据库的缺省表空间。
datcompatibility	name	数据库兼容模式。
datacl	aclitem[]	访问权限。
datfrozenxid64	xid	用于跟踪该数据库是否需要为了防止事务ID重叠而进行清理。

16.2.22 PG_DB_ROLE_SETTING

PG_DB_ROLE_SETTING系统表存储数据库运行时每个角色与数据绑定的配置项的默认值 。

表 16-22 PG_DB_ROLE_SETTING 字段

名称	类型	描述
setdatabase	oid	配置项所对应的数据库，如果未指定数据库，则为0。
setrole	oid	配置项所对应的角色，如果未指定角色，则为0。

名称	类型	描述
setconfig	text[]	运行时配置项的默认值。

16.2.23 PG_DEFAULT_ACL

PG_DEFAULT_ACL系统表存储为新建对象设置的初始权限。

表 16-23 PG_DEFAULT_ACL 字段

名称	类型	描述
defaclrole	oid	与此权限相关的角色ID。
defaclnamespace	oid	与此权限相关的命名空间，如果没有，则为0。
defaclobjtype	"char"	此权限的对象类型。 • r表示表或视图。 • S表示序列。 • f表示函数。 • T表示类型。
defaclacl	aclitem[]	创建该类型时所拥有的访问权限。

16.2.24 PG_DEPEND

PG_DEPEND系统表记录数据库对象之间的依赖关系。这些信息允许DROP命令找出哪些其它对象必须由DROP CASCADE删除，或者是在DROP RESTRICT的情况下避免删除。

另请参考**PG_SHDEPEND**，对于记录那些在数据库集群之间共享的对象之间的依赖性关系提供了相似的功能。

表 16-24 PG_DEPEND 字段

名字	类型	引用	描述
classid	oid	**PG_CLASS**.oid	依赖对象所在系统表的OID。
objid	oid	任意OID属性	指定依赖对象的OID。
objsubid	integer	-	对于表字段，是该属性的字段数（objid和classid引用表本身）。对于所有其它对象类型，此字段是0。
refclassid	oid	**PG_CLASS**.oid	被引用对象所在的系统表的OID。
refobjid	oid	任意OID属性	指定的被引用对象的OID。

名字	类型	引用	描述
refobjsubid	integer	-	对于表字段，是该字段的字段号（refobjid和refclassid引用表本身）。对于所有其它对象类型，此字段是零。
deptype	"char"	-	定义此依赖关系特定语义的代码。

在所有情况下，一个PG_DEPEND记录表示被引用对象不能在没有删除依赖对象的情况下被删除。但是其中也有几种由deptype定义的情况：

- DEPENDENCY_NORMAL (n)：独立创建的对象之间的一般关系。依赖对象可以在不影响被引用对象的情况下删除。被引用对象只能通过指定CASCADE被删除，这种情况下依赖对象也会被删除。例如：一个表字段对其数据类型有一般依赖关系。
- DEPENDENCY_AUTO (a)：依赖对象可以和被引用对象分别删除，且在被引用对象被删除时应自动被删除（不管是RESTRICT或CASCADE模式）。例如：一个表上的命名约束是该表上的自动依赖关系，因此如果删除了表，它也会被删除。
- DEPENDENCY_INTERNAL (i)：依赖对象作为被引用对象过程的一部分创建，且是其内部实现的一部分。 DROP依赖对象是不会直接允许的（会给户发出一个针对被引用对象的DROP）。不管是否指定CASCADE，一个被引用对象的DROP将被传播来删除其依赖对象。例如：一个用于强制外键约束的触发器将被设置为内部依赖于其约束的**PG_CONSTRAINT**项。
- DEPENDENCY_EXTENSION (e)：依赖对象作为被依赖对象extension的一个成员（请参见**PG_EXTENSION**）。依赖对象可以通过在被依赖对象上DROP EXTENSION删除。在功能上，这种依赖类型和内部依赖的作用相同，其存在只是为了清晰和简化gs_dump。
- DEPENDENCY_PIN (p)：没有依赖对象。这种类型的记录标志着系统本身依赖于被引用对象，因此这个对象决不能被删除。这种类型的记录只有在initdb的时候创建。有依赖对象的字段都为0。

16.2.25 PG_DESCRIPTION

PG_DESCRIPTION系统表可以给每个数据库对象存储一个可选的描述（注释）。许多内置的系统对象的描述提供了PG_DESCRIPTION的初始内容。

这个表的功能类似**PG_SHDESCRIPTION**，用于记录整个集群范围内共享对象的注释。

表 16-25 PG_DESCRIPTION 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	描述所属对象的OID。
classoid	oid	**PG_CLASS**oid	对象显示的系统表的OID。
objsubid	integer	-	对于一个表字段的注释，为字段号（objoid和classoid指向表自身）。对于其它对象类型，为0。
description	text	-	对该对象描述的任意文本。

16.2.26 PG_DIRECTORY

PG_DIRECTORY系统表用于保存用户添加的directory对象可以通过CREATE DIRECTORY语句向该表中添加记录，目前只有系统管理员用户可以向该表中添加记录。

表 16-26 PG_DIRECTORY 字段

名字	类型	描述
oid	oid	行标识符（隐藏属性，必须明确选择才会显示）
dirname	name	目录对象的名称
owner	oid	目录对象的所有者
dirpath	text	目录路径
diracl	aclitem[]	访问权限

16.2.27 PG_ENUM

PG_ENUM系统表包含显示每个枚举类型值和标签的记录。给定枚举类型的内部表示实际上是PG_ENUM里面相关行的OID。

表 16-27 PG_ENUM 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
enumtypid	oid	PG_TYPE.oid	包含此枚举值的pg_type项的OID
enumsortorder	real	-	此枚举值在其枚举类型中的排序位置
enumlabel	name	-	此枚举值的文本标签

PG_ENUM行的OID值遵循一种特殊规则：OID的数值被保证按照其枚举类型一样的排序顺序排序。即如果两个偶数OID属于同一枚举类型，那么较小的OID必然具有较小enumsortorder值。奇数OID不需要遵循排序顺序。这种规则使得枚举比较例程在很多常见情况下可以避免系统目录查找。创建和修改枚举类型的例程尝试尽可能地为枚举值分配偶数OID。

当一个枚举类型被创建后，其成员会被分配排序顺序位置1到n。但是后面增加的成员可能会分配负值或者分数值的enumsortorder。对于这些值的唯一要求是它们必须被正确的排序且在每个枚举类型中保持唯一。

16.2.28 PG_EXTENSION

PG_EXTENSION系统表存储关于所安装扩展的信息。GaussDB(DWS)默认有十二个扩展，即PLPGSQL、DIST_FDW、FILE_FDW、HDFS_FDW 、HSTORE、PLDBGAPI、DIMSEARCH、PACKAGES、GC_FDW、UUID-OSSP、LOG_FDW和ROACH_API。

表 16-28 PG_EXTENSION

名称	类型	描述
extname	name	扩展名
extowner	oid	扩展的所有者
extnamespace	oid	扩展导出对象的命名空间
extrelocatable	boolean	如果扩展能够重定位到其他schema，则为true
extversion	text	扩展的版本号
extconfig	oid[]	扩展的配置信息
extcondition	text[]	扩展配置信息的过滤条件

16.2.29 PG_EXTENSION_DATA_SOURCE

PG_EXTENSION_DATA_SOURCE系统表存储外部数据源对象的信息。一个外部数据源对象（Data Source）包含了外部数据库的一些口令编码等信息，主要配合Extension Connector使用。

表 16-29 PG_EXTENSION_DATA_SOURCE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
srcname	name	-	外部数据源对象的名称。
srcowner	oid	PG_AUTHID.oid	外部数据源对象的所有者。
srctype	text	-	外部数据源对象的类型，缺省为空。
srcversion	text	-	外部数据源对象的版本，缺省为空。
srcacl	aclitem[]	-	访问权限。
srcoptions	text[]	-	外部数据源对象的指定选项，使用“keyword=value”格式的字符串。

16.2.30 PG_FOREIGN_DATA_WRAPPER

PG_FOREIGN_DATA_WRAPPER系统表存储外部数据封装器定义。一个外部数据封装器是在外部服务器上驻留外部数据的机制，是可以访问的。

表 16-30 PG_FOREIGN_DATA_WRAPPER 字段

名字	类型	引用	描述
oid	oid	-	行标识符(隐藏属性，必须明确选择才会显示)。
fdwname	name	-	外部数据封装器名。
fdwowner	oid	PG_AUTHID.oid	外部数据封装器的所有者。
fdwhandler	oid	PG_PROC.oid	引用一个负责为外部数据封装器提供扩展例程的处理函数。如果没有提供处理函数则为0。
fdwvalidator	oid	PG_PROC.oid	引用一个验证器函数，这个验证器函数负责验证给予外部数据封装器的选项、外部服务器选项和使用外部数据封装器的用户映射的有效性。如果没有提供验证器函数则为0。
fdwacl	aclitem[]	-	访问权限。
fdwoptions	text[]	-	外部数据封装器指定选项，使用“keyword=value”格式的字符串。

16.2.31 PG_FOREIGN_SERVER

PG_FOREIGN_SERVER系统表存储外部服务器定义。一个外部服务器描述了一个外部数据源，例如一个远程服务器。外部服务器通过外部数据封装器访问。

表 16-31 PG_FOREIGN_SERVER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
srvname	name	-	外部服务器名
srvowner	oid	PG_AUTHID.oid	外部服务器的所有者
srvfdw	oid	PG_FOREIGN_DATA_WRAPPER.oid	此外部服务器的外部数据封装器的OID
srvtype	text	-	服务器的类型（可选）
srvversion	text	-	服务器的版本（可选）

名字	类型	引用	描述
srvacl	aclitem[]	-	访问权限
srvoptions	text[]	-	外部服务器指定选项，使用“keyword=value”格式的字符串

16.2.32 PG_FOREIGN_TABLE

PG_FOREIGN_TABLE系统表存储外部表的辅助信息。

表 16-32 PG_FOREIGN_TABLE 字段

名称	类型	描述
ftrelid	oid	外部表的OID
ftserver	oid	外部表的所在服务器的OID
ftwriteonly	boolean	外部表是否可写
ftoptions	text[]	外部表的可选项

16.2.33 PG_INDEX

PG_INDEX系统表存储索引的一部分信息，其他的信息大多数在PG_CLASS中。

表 16-33 PG_INDEX 字段

名称	类型	描述
indexrelid	oid	此索引的pg_class项的OID。
indrelid	oid	使用该索引的表在pg_class项的OID。
indnatts	smallint	索引中的字段数目。
indisunique	boolean	如果为真，为唯一索引。
indisprimary	boolean	如果为真，该索引为该表的主键。该字段为真时，indisunique也总是为真。
indisexclusion	boolean	如果为真，该索引支持排他约束。
indimmediate	boolean	如果为真，在插入数据时会立即执行唯一性检查。
indisclustered	boolean	如果为真，则该表最后以此索引进行了聚簇。
indisusable	boolean	如果为真，此索引对INSERT/SELECT可用。

名称	类型	描述
indisvalid	boolean	如果为真，则此索引可以用于查询。如果为假，则该索引可能不完整，仍然必须在INSERT/UPDATE操作时进行更新，但不能安全的被用于查询。如果是唯一索引，则唯一属性也不为真。
indcheckxmin	boolean	如果为真，查询不能使用此索引，直到pg_index此行的xmin低于其快照的TransactionXmin，因为该表可能包含它们可见的不兼容行断开的HOT链。
indisready	boolean	如果为真，表示此索引对插入数据可用。如果为假，在插入或修改数据时忽略此索引。
indkey	int2vector	这是一个包含indnatts值的数组，这些数组值表示此索引所建立的表字段。比如一个值为1 3的意思是第一个字段和第三个字段组成这个索引键。数组中的0表示对应的索引属性是一个表字段上的表达式，而不是一个简单的字段引用。
indcollation	oidvector	索引用到的各列的ID。
indclass	oidvector	对于索引键中的每个字段，该字段都包含要使用的操作符类的OID，详见**PG_OPCLASS**。
indoption	int2vector	存储列前标识位，该标识位是由索引的访问方法定义。
indexprs	pg_node_tree	表达式树（以nodeToString()形式表现）用于那些非简单字段引用的索引属性。它是一个列表，个数与INDKEY中的零值个数相同。如果所有索引属性都是简单的引用，则为空。
indpred	pg_node_tree	部分索引谓词的表达式树（以nodeToString()的形式表现）。如果不是部分索引，则为空。

16.2.34 PG_INHERITS

PG_INHERITS系统表记录关于表继承层次的信息。数据库里每个直接的子系表都有一条记录。间接的继承可以通过追溯记录链来判断。

表 16-34 PG_INHERITS 字段

名字	类型	引用	描述
inhrelid	oid	**PG_CLASS**.oid	子表的OID。
inhparent	oid	**PG_CLASS**.oid	父表的OID。

名字	类型	引用	描述
inhseqno	integer	-	如果一个子表存在多个直系父表（多重继承），这个数字表示此继承字段的排列顺序。计数从1开始。

16.2.35 PG_JOB

PG_JOB系统表存储用户创建的定时任务的任务详细信息，定时任务线程定时轮询pg_job系统表中的时间，当任务到期会触发任务的执行，并更新pg_job表中的任务状态。该系统表属于Shared Relation，所有创建的job记录对所有数据库可见。

表 16-35 PG_JOB 字段

名字	类型	描述
job_id	bigint	作业ID，主键，是唯一的（有唯一索引）。
current_postgres_pid	bigint	如果当前任务已被执行，那么此处记录运行此任务的postgres线程ID。默认为 -1，表示此任务未被执行过。
log_user	name	创建者的UserName。
priv_user	name	作业执行者的UserName。
dbname	name	标识作业要在哪个数据库执行的数据库名字。
node_name	name	标识当前作业是在哪个CN上创建和执行。
job_status	"char"	当前任务的执行状态，取值范围：('r', 's', 'f', 'd')，默认为's'，取值含义： Status of job step: r=running, s=successfully finished, f=job failed, d=disable 当job连续执行失败16次，会将job_status自动设置为失效状态'd'，后续不再执行该job。 注：当用户将定时任务关闭（即：guc参数job_queue_processes为0时），由于监控job执行的线程不会启动，所以该状态不会根据job的实时状态进行设置，用户不需要关注此状态。只有当开启定时任务功能（即：guc参数job_queue_processes为非0时），系统才会根据当前job的实时状态刷新该字段值。
start_date	timestamp without time zone	作业第一次开始执行时间，时间精确到毫秒。
next_run_date	timestamp without time zone	下次定时执行任务的时间，时间精确到毫秒。

名字	类型	描述
failure_count	smallint	失败计数，作业连续执行失败16次，不再继续执行。
interval	text	作业执行的重复时间间隔。
last_start_date	timestamp without time zone	上次运行开始时间，时间精确到毫秒。
last_end_date	timestamp without time zone	上次运行的结束时间，时间精确到毫秒。
last_suc_date	timestamp without time zone	上次成功运行的开始时间，时间精确到毫秒。
this_run_date	timestamp without time zone	正在运行任务的开始时间，时间精确到毫秒。

16.2.36 PG_JOB_PROC

PG_JOB_PROC系统表对应PG_JOB表中每个任务的作业内容（包括：PL/SQL代码块、匿名块）。将存储过程信息独立出来，是因为Oracle中这个字段是varchar(4000)的，如果放到PG_JOB中，被加载到共享内存的时候，会占用不必要的空间，所以在使用的时候再进行查询获取。

表 16-36 PG_JOB_PROC 字段

名字	类型	描述
job_oid	integer	外键，关联pg_job表中的job_id
what	text	作业内容

16.2.37 PG_LANGUAGE

PG_LANGUAGE系统表登记编程语言，用户可以用这些语言或接口写函数或者存储过程。

表 16-37 PG_LANGUAGE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。

名字	类型	引用	描述
lanname	name	-	语言的名称。
lanowner	oid	PG_AUTHID.oid	语言的所有者。
lanispl	boolean	-	内部语言为假（比如SQL），用户定义语言为真。目前，gs_dump仍然使用该字段判断哪些语言需要转储，但可能在将来会被其它机制所取代。
lanpltrusted	boolean	-	如果是可信语言则为真，即系统相信它不会被授予任何正常SQL执行环境之外的权限。只有初始用户可以在用非可信的语言创建函数。
lanplcallfoid	oid	PG_PROC.oid	对于非内部语言，这是指该语言处理器的引用，语言处理器是一个特殊函数，负责执行以某种语言写的所有函数。
laninline	oid	PG_PROC.oid	此字段引用一个负责执行“inline”匿名代码块的函数（DO块）。如果不支持内联块则为0。
lanvalidator	oid	PG_PROC.oid	此字段引用一个语言校验器函数，它负责检查新创建函数的语法和有效性。如果没有提供校验器，则为0。
lanacl	aclitem[]	-	访问权限。

16.2.38 PG_LARGEOBJECT

PG_LARGEOBJECT系统表保存那些标记着“大对象”的数据。一个大对象是使用其创建时分配的OID标识的。每个大对象都分解成足够小的小段或者“页面”以便以行的形式存储在PG_LARGEOBJECT里。每页的数据定义为LOBLKSIZE。

需要有系统管理员权限才可以访问此系统表。

表 16-38 PG_LARGEOBJECT 字段

名字	类型	引用	描述
loid	oid	PG_LARGEOBJECT_METADATA.oid	包含本页的大对象的标识符。
pageno	integer	-	本页在其大对象数据中的页码，从0开始计算。
data	bytea	-	实际存储在大对象中的数据。这些数据不会超过LOBLKSIZE字节，且可能更小。

PG_LARGEOBJECT的每一行保存一个大对象的一个页面，从该对象内部的字节偏移（pageno * LOBLKSIZE）开始。这种实现允许稀疏存储：页面可能丢失，并且可以比LOBLKSIZE字节少（即使它们不是对象的最后一页）。大对象中丢失的区域会被读为0。

16.2.39 PG_LARGEOBJECT_METADATA

PG_LARGEOBJECT_METADATA系统表存储与大数据相关的元数据。实际的大对象数据存储在PG_LARGEOBJECT里。

表 16-39 PG_LARGEOBJECT_METADATA 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
lomowner	oid	PG_AUTHID.oid	大对象的所有者
lomacl	aclitem[]	-	访问权限

16.2.40 PG_NAMESPACE

PG_NAMESPACE系统表存储命名空间，即存储schema相关的信息。

表 16-40 PG_NAMESPACE 字段

名称	类型	描述
nspname	name	命名空间的名称。
nspowner	oid	命名空间的所有者。
nsptimeline	bigint	在DN上创建此命名空间时的时间线。此字段为内部使用，仅在DN上有效。
nspacl	aclitem[]	访问权限。具体请参见GRANT和REVOKE。

16.2.41 PG_OBJECT

PG_OBJECT系统表存储限定类型对象(object_type中存在的类型)的创建用户、创建时间、最后修改时间和最后analyze时间。

表 16-41 PG_OBJECT 字段

名称	类型	描述
object_oid	oid	对象标识符。

名称	类型	描述
object_type	"char"	对象类型： • r表示表，包括普通表和临时表 • i表示索引 • s表示序列 • v表示视图 • p表示存储过程和函数
creator	oid	创建用户的标识符。
ctime	timestamp with time zone	对象创建时间。
mtime	timestamp with time zone	对象最后修改时间，修改行为包括ALTER操作和GRANT、REVOKE操作。
last_analyze_time	timestamp with time zone	对象进行最后一次analyze的时间。

须知

- 仅针对用户正常操作行为进行记录，无法记录对象升级以前和initdb过程中的行为。
- ctime和mtime的时间记录为本次操作的事务起始时间。
- 由扩容引起的对象修改时间也会被记录。

16.2.42 PG_OBSSCANINFO

PG_OBSSCANINFO系统表定义了在云上加速场景中，使用加速集群时扫描OBS数据的运行时信息，每条记录对应一个query中单个OBS外表的运行时信息。

表 16-42 PG_OBSSCANINFO 字段

名字	类型	引用	描述
query_id	bigint	-	查询标识
user_id	text	-	执行该查询的数据库用户
table_name	text	-	OBS外表的表名
file_type	text	-	底层数据保存的文件格式
time_stamp	time_stam	-	扫描操作开始的时间

名字	类型	引用	描述
actual_time	double	-	扫描操作执行时间，单位为秒
file_scanned	bigint	-	扫描的文件数量
data_size	double	-	扫描的数据量，单位为字节
billing_info	text	-	保留字段

16.2.43 PG_OPCLASS

PG_OPCLASS系统表定义索引访问方法操作符类。

每个操作符类为一种特定数据类型和一种特定索引访问方法定义索引字段的语义。一个操作符类本质上指定一个特定的操作符族适用于一个特定的可索引的字段数据类型。索引的字段实际可用的族中的操作符集是接受字段的数据类型作为它们的左边的输入的那个。

表 16-43 PG_OPCLASS 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
opcmethod	oid	PG_AM.oid	操作符类所属的索引访问方法。
opcname	name	-	操作符类的名称。
opcnamespace	oid	PG_NAMESPACE.oid	操作符类所属的命名空间。
opcowner	oid	PG_AUTHID.oid	操作符类所有者。
opcfamily	oid	PG_OPFAMILY.oid	包含此操作符类的操作符族。
opcintype	oid	PG_TYPE.oid	操作符类索引的数据类型。
opcdefault	boolean	-	如果操作符类是opcintype的缺省，则为真。
opckeytype	oid	PG_TYPE.oid	索引数据的类型，如果和opcintype相同则为0。

一个操作符类的opcmethod必须匹配包含它的操作符族的opfmethod。同样，对于任意给定的opcmethod和opcintype的组合，不能有超过一个PG_OPCLASS行有opcdefault为真。

16.2.44 PG_OPERATOR

PG_OPERATOR系统表存储有关操作符的信息。

表 16-44 PG_OPERATOR 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
oprname	name	-	操作符的名称
oprnamespace	oid	PG_NAMESPACE.oid	包含此操作符的命名空间的OID
oprowner	oid	PG_AUTHID.oid	操作符所有者
oprkind	"char"	-	• b=infix =中缀（两边） • l=前缀（左边） • r=后缀（右边）
oprcanmerge	boolean	-	该操作符是否支持合并连接
oprcanhash	boolean	-	该操作符是否支持Hash连接
oprleft	oid	PG_TYPE.oid	左操作数的类型
oprright	oid	PG_TYPE.oid	右操作数的类型
oprresult	oid	PG_TYPE.oid	结果类型
oprcom	oid	PG_OPERATOR.oid	此操作符的交换符（如果存在）
oprnegate	oid	PG_OPERATOR.oid	此操作符的反转器（如果存在）
oprcode	regproc	PG_PROC.oid	实现该操作符的函数
oprrest	regproc	PG_PROC.oid	此操作符的约束选择性计算函数
oprjoin	regproc	PG_PROC.oid	此操作符的连接选择性计算函数

16.2.45 PG_OPFAMILY

PG_OPFAMILY系统表定义操作符族。

每个操作符族是操作符和相关支持例程的集合，这些例程实现了为特定索引访问方法指定的语义。此外，按照访问方法指定的某种方式，一个族内的操作符都是“兼容的”。操作符族允许跨数据类型操作符与索引一起使用，并可以推理使用访问方法语义相关内容。

表 16-45 PG_OPFAMILY 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
opfmethod	oid	PG_AM.oid	操作符族使用的索引方法
opfname	name	-	操作符族的名称
opfnamespace	oid	PG_NAMESPACE.oid	操作符族的命名空间
opfowner	oid	PG_AUTHID.oid	操作符族的所有者

定义一个操作符族的大多数信息不在PG_OPFAMILY，而是在相关的PG_AMOP，PG_AMPROC和PG_OPCLASS中。

16.2.46 PG_PARTITION

PG_PARTITION系统表存储数据库内所有分区表(partitioned table)、分区(table partition)、分区上toast表和分区索引(index partition)四类对象的信息。分区表索引(partitioned index)的信息不在PG_PARTITION系统表中保存。

表 16-46 PG_PARTITION 字段

名称	类型	描述
relname	name	分区表、分区、分区上toast表和分区索引的名称。
parttype	"char"	对象类型： • 'r'：partitioned table • 'p'：table partition • 'x'：index partition • 't'：toast table
parentid	oid	当对象为分区表或分区时，此字段表示分区表在PG_CLASS中的OID。 当对象为index partition时，此字段表示所属分区表索引(partitioned index)的OID。
rangenum	integer	保留字段。
intervalnum	integer	保留字段。
partstrategy	"char"	分区表分区策略，现在仅支持： 'r'：范围分区。 'v'：数值分区。

名称	类型	描述
relfilenode	oid	table partition、index partition、分区上toast表的物理存储位置。
reltablespace	oid	table partition、index partition、分区上toast表所属表空间的OID。
relpages	double precision	统计信息：table partition、index partition的数据页数量。
reltuples	double precision	统计信息：table partition、index partition的元组数。
relallvisible	integer	统计信息：table partition、index partition的可见数据页数。
reltoastrelid	oid	table partition所对应toast表的OID。
reltoastidxid	oid	table partition所对应toast表的索引的OID。
indextblid	oid	index partition对应table partition的OID。
indisusable	boolean	分区索引是否可用。
reldeltarelid	oid	Delta表的OID。
reldeltaidx	oid	Delta表的索引表的OID。
relcudescrelid	oid	CU描述表的OID。
relcudescidx	oid	CU描述表的索引表的OID。
relfrozenxid	xid32	冻结事务ID号。 为保持前向兼容，保留此字段，新增relfrozenxid64用于记录此信息。
intspnum	integer	间隔分区所属表空间的个数。
partkey	int2vector	分区键的列号。
intervaltablespace	oidvector	间隔分区所属的表空间，间隔分区以round-robin方式落在这些表空间内。
interval	text[]	间隔分区的间隔值。
boundaries	text[]	范围分区和间隔分区的上边界。
transit	text[]	间隔分区的跳转点。
reloptions	text[]	设置partition的存储属性，与pg_class.reloptions的形态一样，用"keyword=value"格式的字符串来表示，目前用于在线扩容的信息搜集。
relfrozenxid64	xid	冻结事务ID号。

16.2.47 PG_PLTEMPLATE

PG_PLTEMPLATE系统表存储过程语言的“模板”信息。

表 16-47 PG_PLTEMPLATE 字段

名称	类型	描述
tmplname	name	该模板所应用的语言的名称。
tmpltrusted	boolean	如果语言被认为是可信的，则为真。
tmpldbacreate	boolean	如果语言是由数据库所有者创建的，则为真。
tmplhandler	text	调用处理器函数的名称。
tmplinline	text	匿名块处理器的名称，如果没有则为NULL。
tmplvalidator	text	校验函数的名称，如果没有则为NULL。
tmpllibrary	text	实现语言的共享库的路径。
tmplacl	aclitem[]	模板的访问权限（未使用）。

16.2.48 PG_PROC

PG_PROC系统表存储函数或过程的信息。

表 16-48 PG_PROC 字段

名称	类型	描述
proname	name	函数名。
pronamespace	oid	此函数所在命名空间的OID。
proowner	oid	函数的所有者。
prolang	oid	实现语言或函数的调用接口。
procost	real	估计执行成本。
prorows	real	结果行估计数。
provariadic	oid	参数元素的数据类型。
protransform	regproc	此函数的简化调用方式。
proisagg	boolean	函数是否为聚集函数。
proiswindow	boolean	函数是否为窗口函数。
prosecdef	boolean	函数是否为一个安全定义器（例如，一个“setuid”函数）。

名称	类型	描述
proleakproof	boolean	函数有无其他影响。如果函数没有对参数进行防泄露处理，则会抛出错误。
proisstrict	boolean	如果任意调用参数为空，函数是否返回空值。这种情况下函数实际上根本不会被调用。非“strict”的函数必须准备处理空值输入。
proretset	boolean	函数是否返回一个集合（即，指定数据类型的多个数值）。
provolatile	"char"	说明该函数的结果是只倚赖于它的输入参数，或者还会被外接因素影响。 ● i表示“不可变的”（immutable）函数，对于相同的输入总是输出相同的结果。 ● s表示稳定的”（stable）函数，对于固定输入其结果在一次扫描里不变。 ● v表示“易变”（volatile）函数，其结果可能在任何时候都变化。
pronargs	smallint	参数个数。
pronargdefaults	smallint	有默认值的参数个数。
prorettype	oid	返回参数类型的OID。
proargtypes	oidvector	函数参数的数据类型的数组。数组里只包括输入参数（包括INOUT参数），因此也表现了函数的调用特征。
proallargtypes	oid[]	函数参数的数据类型的数组。数组里包括所有参数的类型（包括OUT和INOUT参数），如果所有参数都是IN参数，则这个字段就会为空。注意数组下标是以1为起点的，而因为历史原因，proargtypes的下标起点为0。
proargmodes	"char"[]	函数参数模式的数组。 ● i表示IN参数 ● o表示OUT参数 ● b表示INOUT参数 如果所有参数都是IN参数，则这个字段为空。注意此数组下标对应的是proallargtypes的位置，而不是proargtypes。
proargnames	text[]	函数参数的名字的数组。没有名字的参数在数组里设置为空字符串。如果没有一个参数有名字，这个字段为空。注意此数组的下标对应proallargtypes而不是proargtypes。
proargdefaults	pg_node_tree	默认值的表达式树。是PRONARGDEFAULTS元素的列表。

名称	类型	描述
prosrc	text	描述函数或存储过程的定义。例如，对于解释型语言来说就是函数的源程序，或者一个链接符号，一个文件名，或者函数和存储过程创建时指定的其他任何函数体内容，具体取决于语言/调用习惯的实现。
probin	text	关于如何调用该函数的附加信息。同样，其含义也是和语言相关的。
proconfig	text[]	函数针对运行时配置变量的本地设置。
proacl	aclitem[]	访问权限。具体请参见GRANT和REVOKE。
prodefaultargpos	int2vector	函数默认值的位置，不局限于能最后几个参数才可有默认值。
fencedmode	boolean	函数的执行模式，表示函数是在fence还是not fence模式下执行。如果是fence执行模式，函数的执行会在重新fork的进程中执行。默认值是fence。
proshippable	boolean	函数是否可以下推到DN上执行，默认值是false。 • 对于IMMUTABLE类型的函数，函数始终可以下推到DN上执行。 • 对于STABLE/VOLATILE类型的函数，仅当函数的属性是SHIPPABLE的时候，函数可以下推到DN执行。
propackage	boolean	该函数是否支持重载，主要针对Oracle风格的函数，默认值是false。

16.2.49 PG_RANGE

PG_RANGE系统表存储关于范围类型的信息。

除了**PG_TYPE**里类型的记录。

表 16-49 PG_RANGE 字段

名字	类型	引用	描述
rngtypid	oid	**PG_TYPE**.oid	范围类型的OID。
rngsubtype	oid	**PG_TYPE**.oid	该范围类型的元素类型（子类型）的OID。
rngcollation	oid	**PG_COLLATION**.oid	用于范围比较的排序规则的OID，如果没有则为0。

名字	类型	引用	描述
rngsubopc	oid	PG_OPCLASS.oid	用于范围比较的子类型的操作符类的OID。
rngcanonical	regproc	PG_PROC.oid	转换范围类型为规范格式的函数的OID，如果没有则为0。
rngsubdiff	regproc	PG_PROC.oid	返回两个double precision元素值的不同的函数的OID，如果没有则为0。

rngsubopc（如果元素类型是可排序的，则加上rngcollation）决定用于范围类型的排序顺序。当元素类型是时使用rngcanonical用于离散类型的元素类型。

16.2.50 PG_REDACTION_COLUMN

PG_REDACTION_COLUMN系统表存储脱敏列的信息。

表 16-50 PG_REDACTION_COLUMN 字段

名称	类型	描述
object_oid	oid	脱敏对象OID
column_attrno	smallint	脱敏列attrno
function_type	integer	脱敏类型
function_parameters	text	脱敏类型为partial类型时的参数
regexp_pattern	text	脱敏类型为regexp时，格式化字符串
regexp_replace_string	text	脱敏类型为regexp时，替换串
regexp_position	integer	脱敏类型为regexp时，起始替换位置
regexp_occurrence	integer	脱敏类型为regexp时，替换次数
regexp_match_parameter	text	脱敏类型为regexp时，正则控制参数
column_description	text	脱敏列描述信息

16.2.51 PG_REDACTION_POLICY

PG_REDACTION_POLICY系统表脱敏对象的信息。

表 16-51 PG_REDACTION_POLICY 字段

名称	类型	描述
object_oid	oid	脱敏对象OID
policy_name	name	脱敏策略名称
enable	boolean	策略状态（开启、关闭） **说明** 函数enable的取值： • true表示开启 • false表示关闭
expression	pg_node_tree	策略生效表达式（针对用户）
policy_description	text	策略描述信息

16.2.52 PG_RESOURCE_POOL

PG_RESOURCE_POOL系统表提供了数据库资源池的信息。

表 16-52 PG_RESOURCE_POOL 字段

名称	类型	描述
respool_name	name	资源池名称。
mem_percent	integer	内存配置的百分比。
cpu_affinity	bigint	CPU绑定core的数值。
control_group	name	资源池所在的control group名字。
active_statements	integer	资源池上最大的并发数。
max_dop	integer	最大并发度，保留字段。
memory_limit	name	资源池最大的内存。
parentid	oid	父资源池OID。
io_limits	integer	每秒触发IO的次数上限。行存单位是万次/s，列存是次/s。
io_priority	text	IO利用率高达90%时，重消耗IO作业进行IO资源管控时关联的优先级等级。
is_foreign	boolean	表示资源池是否用于逻辑集群之外的用户。如果为true，表示资源池用来控制不属于当前资源池的普通用户的资源。

16.2.53 PG_REWRITE

PG_REWRITE系统表存储为表和视图定义的重写规则。

表 16-53 PG_REWRITE 字段

名称	类型	描述
rulename	name	规则名称
ev_class	oid	使用该规则的表名
ev_attr	smallint	该规则适用的字段（目前总是为0，表示整个表）
ev_type	"char"	规则适用的事件类型： • 1 = SELECT • 2 = UPDATE • 3 = INSERT • 4 = DELETE
ev_enabled	"char"	用于控制复制的触发 • O =“origin”和“local”模式时触发 • D =禁用触发 • R =“replica”时触发 • A =任何模式都会触发
is_instead	boolean	如果是INSTEAD规则，则为真
ev_qual	pg_node_tree	规则条件的表达式树（以nodeToString() 形式存在）
ev_action	pg_node_tree	规则动作的查询树（以nodeToString() 形式存在）

16.2.54 PG_SECLABEL

PG_SECLABEL系统表存储数据对象上的安全标签。

PG_SHSECLABEL的作用类似，只是用于在一个数据库集群内共享的数据库对象的安全标签上。

表 16-54 PG_SECLABEL 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此安全标签所属的对象的OID
classoid	oid	PG_CLASS.oid	此对象的系统目录的OID
objsubid	integer	-	出现在此对象中的列的序号

名字	类型	引用	描述
provider	text	-	与此标签相关的标签提供者
label	text	-	应用于此对象的安全标签

16.2.55 PG_SHDEPEND

PG_SHDEPEND系统表记录数据库对象和共享对象（比如角色）之间的依赖关系。这些信息使得GaussDB(DWS)可以确保对象在被删除时没有被其他对象引用。

PG_DEPEND的作用类似，只是它是针对单个数据库中对象之间的依赖。

和大多数其他系统表不同，PG_SHDEPEND在集群的所有数据库之间共享：每个数据库集群只有一个PG_SHDEPEND，并非每个数据库一个。

表 16-55 PG_SHDEPEND 字段

名字	类型	引用	描述
dbid	oid	PG_DATABASE.oid	依赖对象所在的数据库的OID，如果是共享对象，则为0。
classid	oid	PG_CLASS.oid	依赖对象所在的系统表的OID。
objid	oid	任意OID属性	指定的依赖对象的OID。
objsubid	integer	-	对于一个表字段，为字段号（objid和classid参考表本身）。对于所有其他对象类型，该字段为0。
refclassid	oid	PG_CLASS.oid	被引用对象所在的系统表的OID(必须是一个共享表)。
refobjid	oid	任意OID属性	指定的被引用对象的OID。
deptype	"char"	-	定义该依赖关系的特定语义的代码见表后说明。
objfile	text	-	用户定义C函数库文件路径。

在任何情况下，一条PG_SHDEPEND记录就表明被引用的对象不能在未删除依赖对象的前提下被删除。但是其中也有几种依赖类型由deptype定义的情况：

- SHARED_DEPENDENCY_OWNER (o)

  被引用的对象（必须是一个角色）是依赖对象的所有者。
- SHARED_DEPENDENCY_ACL (a)

  在依赖对象的ACL（访问控制列表，也就是权限列表）中提到被引用的对象（必须是一个角色）。不会为对象的所有者创建SHARED_DEPENDENCY_ACL，因为所有者将具有SHARED_DEPENDENCY_OWNER记录。
- SHARED_DEPENDENCY_PIN (p)

没有依赖对象。这类记录标识系统自身依赖于被依赖对象，因此这种对象绝对不能被删除。此类型的记录只能被initdb创建，依赖对象的字段都为0。

16.2.56 PG_SHDESCRIPTION

PG_SHDESCRIPTION系统表存储共享数据库对象的可选注释。可以使用COMMENT命令操作注释的内容，使用psql的\d命令查看注释内容。

PG_DESCRIPTION提供了类似的功能，它记录了单个数据库中对象的注释。

不同于大多数系统表，PG_SHDESCRIPTION在集群中所有数据库之间共享：每个数据库集群只有一个PG_SHDESCRIPTION，而不是每个数据库一个。

表 16-56 PG_SHDESCRIPTION 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此描述所属的对象的OID
classoid	oid	PG_CLASS.oid	此对象所在的系统目录的OID
description	text	-	作为对该对象描述的任意文本

16.2.57 PG_SHSECLABEL

PG_SHSECLABEL系统表存储在共享数据库对象上的安全标签。安全标签可以用SECURITY LABEL命令操作。

查看安全标签的简单点的方法，请参阅PG_SECLABELS。

PG_SECLABEL的作用类似，只是它是用于在单个数据库内部的对象的安全标签的。

不同于大多数的系统表，PG_SHSECLABEL在一个集群中的所有数据库中共享：每个数据库集群只有一个PG_SHSECLABEL，而不是每个数据库一个。

表 16-57 PG_SHSECLABEL 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	此安全标签所属对象的OID
classoid	oid	PG_CLASS.oid	对象所属系统目录的OID
provider	text	-	与此标签关联的标签提供者
label	text	-	应用于该对象的安全标签

16.2.58 PG_STATISTIC

PG_STATISTIC系统表存储有关该数据库中表和索引列的统计数据。需要有系统管理员权限才可以访问此系统表。

表 16-58 PG_STATISTIC 字段

名称	类型	描述
starelid	oid	所描述的字段所属的表或者索引。
starelkind	"char"	所属对象的类型。
staattnum	smallint	所描述的字段在表中的编号，从1开始。
stainherit	boolean	是否统计有继承关系的对象。
stanullfrac	real	该字段中为NULL的记录的比率。
stawidth	integer	非NULL记录的平均存储宽度，以字节计。
stadistinct	real	标识全局统计信息中所有DN上字段里唯一的非NULL数据值的数目。 ● 大于0的值是独立数值的实际数目。 ● 小于0的值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
stakindN	smallint	一个编码，表示这种类型的统计存储在pg_statistic行的第n个“槽位”。 n的取值范围：1～5
staopN	oid	用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。 n的取值范围：1～5
stanumbersN	real[]	第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则就是NULL。 n的取值范围：1～5
stavaluesN	anyarray	第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则就是NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好的办法。 n的取值范围：1～5
stadndistinct	real	标识dn1上字段里唯一的非NULL数据值的数目。 ● 大于0的值是独立数值的实际数目。 ● 小于0的值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
staextinfo	text	统计信息的扩展信息。预留字段。

16.2.59 PG_STATISTIC_EXT

PG_STATISTIC_EXT系统表存储有关该数据库中表的扩展统计数据，包括多列统计数据和表达式统计数据（后续支持）。收集哪些扩展统计数据是由用户指定的。需要有系统管理员权限才可以访问此系统表。

表 16-59 PG_STATISTIC_EXT 字段

名称	类型	描述
starelid	oid	所描述的字段所属的表或者索引。
starelkind	"char"	所属对象的类型。
stainherit	boolean	是否统计有继承关系的对象。
stanullfrac	real	该字段中为NULL的记录的比率。
stawidth	integer	非NULL记录的平均存储宽度，以字节计。
stadistinct	real	标识全局统计信息中所有DN上字段里唯一的非NULL数据值的数目。 ● 一个大于零的数值是独立数值的实际数目。 ● 一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
stadndistinct	real	标识dn1上字段里唯一的非NULL数据值的数目。 ● 一个大于零的数值是独立数值的实际数目。 ● 一个小于零的数值是表中行数的分数的负数（比如，一个字段的数值平均出现概率为两次，则可以表示为stadistinct=-0.5）。 ● 0表示独立数值的数目未知。
stakindN	smallint	一个编码，表示这种类型的统计存储在pg_statistic行的第n个“槽位”。 n的取值范围：1～5
staopN	oid	一个用于生成这些存储在第n个“槽位”的统计信息的操作符。比如，一个柱面图槽位会显示<操作符，该操作符定义了该数据的排序顺序。 n的取值范围：1～5
stakey	int2vector	所描述的字段编号的数组。
stanumbersN	real[]	第n个“槽位”的相关类型的数值类型统计，如果该槽位和数值类型没有关系，则就是NULL。 n的取值范围：1～5

名称	类型	描述
stavaluesN	anyarray	第n个“槽位”类型的字段数据值，如果该槽位类型不存储任何数据值，则为NULL。每个数组的元素值实际上都是指定字段的数据类型，因此，除了把这些字段的类型定义成anyarray之外，没有更好的办法。 n的取值范围：1～5
staexprs	pg_node_tree	扩展统计信息对应的表达式。

16.2.60 PG_SYNONYM

PG_SYNONYM系统表存储同义词对象名与其他数据库对象名间的映射信息。

表 16-60 PG_SYNONYM 字段

名称	类型	描述
synname	name	同义词名称。
synnamespace	oid	该同义词所在的命名空间的OID。
synowner	oid	同义词的所有者，通常是其创建者OID。
synobjschema	name	关联对象指定的模式名。
synobjname	name	关联对象名。

16.2.61 PG_TABLESPACE

PG_TABLESPACE系统表存储表空间信息。

表 16-61 PG_TABLESPACE 字段

名称	类型	描述
spcname	name	表空间名。
spcowner	oid	表空间的所有者，通常是其创建者。
spcacl	aclitem[]	访问权限。具体请参见GRANT和REVOKE。
spcoptions	text[]	表空间的选项。
spcmaxsize	text	可使用的最大磁盘空间大小，单位Byte。

16.2.62 PG_TRIGGER

PG_TRIGGER系统表存储触发器信息。

名称	类型	描述
tgrelid	oid	触发器所在表的OID。
tgname	name	触发器名。
tgfoid	oid	触发器OID。
tgtype	smallint	触发器类型。
tgenabled	"char"	O表示触发器在“origin”和“local”模式下触发。 D表示触发器被禁用。 R表示触发器在“replica”模式下触发。 A表示触发器始终触发。
tgisinternal	boolean	内部触发器标识，如果为true表示内部触发器。
tgconstrrelid	oid	完整性约束引用的表。
tgconstrindid	oid	完整性约束的索引。
tgconstraint	oid	约束触发器在pg_constraint中的OID。
tgdeferrable	boolean	约束触发器是为DEFERRABLE类型。
tginitdeferred	boolean	约束触发器是否为INITIALLY DEFERRED类型。
tgnargs	smallint	触发器函数入参个数。
tgattr	int2vector	当触发器指定列时的列号，未指定则为空数组。
tgargs	bytea	传递给触发器的参数。
tgqual	pg_node_tree	表示触发器的WHEN条件，如果没有则为null。

16.2.63 PG_TS_CONFIG

PG_TS_CONFIG系统表包含表示文本搜索配置的选项。一个配置指定一个特定的文本搜索解析器和一个用于解析器输出类型的字典列表。

解析器在PG_TS_CONFIG记录中显示，但是字典映射的标记是由PG_TS_CONFIG_MAP中的辅助记录定义的。

表 16-62 PG_TS_CONFIG 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
cfgname	name	-	文本搜索配置名
cfgnamespace	oid	PG_NAMESPACE.oid	此配置所在的命名空间的OID

名字	类型	引用	描述
cfgowner	oid	PG_AUTHID.oid	配置的所有者
cfgparser	oid	PG_TS_PARSER.oid	此配置的文本搜索解析器的OID
cfoptions	text[]	-	分词相关配置选项

16.2.64 PG_TS_CONFIG_MAP

PG_TS_CONFIG_MAP系统表包含为每个文本搜索配置的解析器的每种输出符号类型，显示有哪些文本搜索字典可供查询以及以哪种顺序搜索。

表 16-63 PG_TS_CONFIG_MAP 字段

名字	类型	引用	描述
mapcfg	oid	PG_TS_CONFIG.oid	拥有此映射记录的PG_TS_CONFIG的OID
maptokentype	integer	-	由配置的解析器发出的一个符号类型
mapseqno	integer	-	查询该项的顺序
mapdict	oid	PG_TS_DICT.oid	查询的文本搜索字典的OID

16.2.65 PG_TS_DICT

PG_TS_DICT系统表包含定义文本搜索字典的项。字典取决于文本搜索模板，该模板显示所有需要实现的功能。字典本身提供了用户可设置参数的模板。

即允许字典通过非权限用户创建。参数由文本字符串dictinitoption指定，参数的格式和意义取决于模板。

表 16-64 PG_TS_DICT 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
dictname	name	-	文本搜索字典名
dictnamespace	oid	PG_NAMESPACE.oid	此字典所在的命名空间的OID
dictowner	oid	PG_AUTHID.oid	字典的所有者

名字	类型	引用	描述
dicttemplate	oid	PG_TS_TEMPLATE.oid	此字典的文本搜索模板的OID
dictinitoption	text	-	模板的初始化选项字符串

16.2.66 PG_TS_PARSER

PG_TS_PARSER系统表包含定义文本解析器的项。解析器负责分割输入文本为词位，并且为每个词位分配标记类型。因为解析器必须通过C语言级别的函数实现，所以新解析器必须由数据库系统管理员创建。

表 16-65 PG_TS_PARSER 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
prsname	name	-	文本搜索解析器名称
prsnamespace	oid	PG_NAMESPACE.oid	解析器所在的命名空间的OID
prsstart	regproc	PG_PROC.oid	解析器的启动函数的OID
prstoken	regproc	PG_PROC.oid	解析器的下一个标记函数的OID
prsend	regproc	PG_PROC.oid	解析器的关闭函数的OID
prsheadline	regproc	PG_PROC.oid	解析器的标题函数的OID
prslextype	regproc	PG_PROC.oid	解析器的lextype函数的OID

16.2.67 PG_TS_TEMPLATE

PG_TS_TEMPLATE系统表包含定义文本搜索模板的项。模板是文本搜索字典的类的实现框架。因为模板必须通过C语言级别的函数实现，索引新模板的创建必须由数据库系统管理员创建。

表 16-66 PG_TS_TEMPLATE 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）
tmplname	name	-	文本搜索模板名
tmplnamespace	oid	PG_NAMESPACE.oid	模板所属的命名空间的OID
tmplinit	regproc	PG_PROC.oid	模板的初始化函数的OID
tmpllexize	regproc	PG_PROC.oid	模板的lexize函数的OID

16.2.68 PG_TYPE

PG_TYPE系统表存储数据类型的相关信息。

表 16-67 PG_TYPE 字段

名称	类型	描述
typname	name	数据类型名称。
typnamespace	oid	此类型所在的命名空间的OID。
typowner	oid	此类型的所有者。
typlen	smallint	对于定长类型是该类型内部表现形式的字节数。对于变长类型typlen为负值。 ● -1表示一种“变长”（有长度字属性的数据）。 ● -2表示一个以NULL结尾的C字符串。
typbyval	boolean	指定内部传递这个类型的数值时是传值还是传引用。如果该类型的TYPLEN不是1，2，4，8， TYPBYVAL最好为假。变长类型通常是传引用。即使TYPLEN允许传值，TYPBYVAL也可以为假。
typtype	"char"	● b表示基础类型。 ● c表示复合类型（比如，一个表的行类型）。 ● e表示枚举类型。 ● p表示伪类型。 参见typrelid和typbasetype。
typcategory	"char"	数据类型的模糊分类，可用于解析器使用的数据转换依据。

名称	类型	描述
typispreferred	boolean	如果为真，则数据符合TYPCATEGORY所指定的转换规则时进行转换。
typisdefined	boolean	如果定义了类型，则为真。如果是一种尚未定义的类型的占位符，则为假。如果为假，则除了该类型名称，名名空间和OID之外没有可依赖的对象。
typdelim	"char"	分析数组输入时，分隔两个此类型数值的字符。请注意，分隔符是与数组元素数据类型相关联，而不是与数组数据类型相关联。
typrelid	oid	如果是复合类型（请参见typtype），则此字段指向pg_class中定义该表的行。对于独立的复合类型，pg_class记录并不表示一个表，但是总需要它来查找该类型连接的pg_attribute记录。非复合类型为0。
typelem	oid	如果不为0，则它标识pg_type中的另一行。当前类型可以像一个产生类型为typelem的数组来描述。“true”数组类型是变长的（typlen= -1），但是某些定长（typlen > 0）类型也有非零的typelem（比如name和point）。如果一个定长类型有typelem，则其内部形式必须是typelem数据类型的某个数目的个数值，不能有其他数据。变长数组类型有一个该数组子过程定义的头（文件）。
typarray	oid	如果不为0，则表示在pg_type中有对应的类型记录。
typinput	regproc	输入转换函数（文本格式）。
typoutput	regproc	输出转换函数（文本格式）。
typreceive	regproc	输入转换函数（二进制格式），如果没有则为0。
typsend	regproc	输出转换函数（二进制格式），如果没有则为0。
typmodin	regproc	输入类型修改符函数，如果为0，则不支持。
typmodout	regproc	输出类型修改符函数，如果为0，则不支持。
typanalyze	regproc	自定义的ANALYZE函数，如果使用标准函数，则为0。

名称	类型	描述
typalign	"char"	当存储此类型的数值时要求的对齐方式。适用于磁盘存储以及该值在数据库中的大多数形式。如果数值是连续存储的，比如在磁盘上以完全的裸数据的形式存放时，则先在此类型的数据前填充空白，这样它就可以按照要求的边界存储。对齐引用是该序列中第一个数据的开头。可能的值包含： ● c = char对齐，即不需要对齐。 ● s = short对齐（在大多数机器上是2字节） ● i = int对齐（在大多数机器上是4字节） ● d = double对齐（在大多数机器上是8字节，但不一定是全部） **须知** 对于系统表里使用的类型，在pg_type里定义的尺寸和对齐方式要和编译器在表示表行的结构中布局方式保持一致。
typstorage	"char"	指明一个变长类型（那些有typlen = -1）是否准备好应付非常规值，以及对这种属性的类型的缺省策略是什么。可能的值包含： ● p：数值总是以简单方式存储。 ● e：数值可以存储在一个"次要"关系中（如果有该关系，请参见pg_class.reltoastrelid）。 ● m：数值可以以内联压缩方式存储。 ● x：数值可以以内联压缩方式或者在"次要"表里存储。 **须知** m域也可以移到从属表里存储，但只是最后的解决方法（首先移动e和x域）。
typenotnull	boolean	表示在某类型上的一个NOTNULL约束。目前只用于域。
typbasetype	oid	如果这是一个衍生类型（请参见typtype），则该标识作为这个类型的基础的类型。如果不是衍生类型则为零。
typtypmod	integer	域使用typtypmod记录要应用于其基础类型上的typmod（如果基础类型不使用typmod，则为-1）。如果此类型不是域，则为-1。
typndims	integer	如果一个域是数组，则typndims是数组维数的数值（即typbasetype是一个数组类型；域的typelem将匹配基本类型的typelem）。除了数组类型的域以外的类型为0。
typcollation	oid	指定类型的排序规则。如果为0，则表示不支持排序规则。
typdefaultbin	pg_node_tree	如果不为NULL，则为该类型缺省表达式的nodeToString() 表现形式。目前这个字段只用于域。

名称	类型	描述
typdefault	text	如果某类型没有相关缺省值，则为NULL。如果typdefaultbin不为NULL，则typdefault必须包含一个typdefaultbin代表的缺省表达式的人类可读版本。如果typdefaultbin为NULL但typdefault不为NULL，typdefault则是该类型缺省值的外部表现形式，可以将其输入到类型的转换器生成一个常量。
typacl	aclitem[]	访问权限。

16.2.69 PG_USER_MAPPING

PG_USER_MAPPING系统表存储从本地用户到远程的映射。

需要有系统管理员权限才可以访问此系统表。普通用户可以使用视图PG_USER_MAPPINGS进行查询。

表 16-68 PG_USER_MAPPING 字段

名字	类型	引用	描述
oid	oid	-	行标识符（隐藏属性，必须明确选择才会显示）。
umuser	oid	PG_AUTHID.oid	被映射的本地用户的OID，如果用户映射是公共的则为0。
umserver	oid	PG_FOREIGN_SERVER.oid	包含此映射的外部服务器的OID。
umoptions	text[]	-	用户映射指定选项，使用“keyword=value”格式的字符串。

16.2.70 PG_USER_STATUS

PG_USER_STATUS系统表提供了访问数据库用户的状态。需要有系统管理员权限才可以访问此系统表

表 16-69 PG_USER_STATUS 字段

名称	类型	描述
roloid	oid	角色的标识。
failcount	integer	尝试失败次数。
locktime	timestamp with time zone	角色被锁定的时间点。

名称	类型	描述
rolstatus	smallint	角色的状态。 ● 0：正常状态。 ● 1：由于登录失败次数超过阈值被锁定了一定的时间。 ● 2：被管理员锁定。
permspace	bigint	角色已经使用的永久表存储空间大小。
tempspace	bigint	角色已经使用的临时表存储空间大小。

16.2.71 PG_WORKLOAD_GROUP

PG_WORKLOAD_GROUP系统表提供了数据库负载组的信息。

表 16-70 PG_WORKLOAD_GROUP 字段

名称	类型	描述
workload_gpname	name	负载组名称
respool_oid	oid	绑定的资源池的OID
act_statements	integer	负载组内最大的活跃语句数

16.2.72 PGXC_CLASS

PGXC_CLASS系统表存储每张表的复制或分布信息。

表 16-71 PGXC_CLASS 字段

名称	类型	描述
pcrelid	oid	表的OID
pclocatortype	"char"	定位器类型 ● H：hash ● M：Modulo ● N：Round Robin ● R：Replicate
pchashalgorithm	smallint	使用哈希算法分布元组
pchashbuckets	smallint	哈希容器的值

名称	类型	描述
pgroup	name	节点组名称
redistributed	"char"	表已经完成重分布
redis_order	integer	重分布的顺序
pcattnum	int2vector	用作分布键的列标号
nodeoids	oidvector_extend	表分布的节点OID列表
options	text	系统内部保留字段，存储扩展状态信息

16.2.73 PGXC_GROUP

PGXC_GROUP系统表存储节点组信息。

表 16-72 PGXC_GROUP 字段

名称	类型	描述
group_name	name	节点组名称
in_redistribution	"char"	是否需要重分布 ● n表示NodeGroup没有再进行重分布 ● y表示NodeGroup是重分布过程中的源节点组 ● t表示NodeGroup是重分布过程中的目的节点组
group_members	oidvector_extend	节点组的节点OID列表
group_buckets	text	分布数据桶的集合
is_installation	boolean	是否安装子集群
group_acl	aclitem[]	访问权限
group_kind	"char"	node group类型 ● i表示installation node group ● n表示普通非逻辑集群node group ● v表示逻辑集群node group ● e表示弹性集群

16.2.74 PGXC_NODE

PGXC_NODE系统表存储集群节点信息。

表 16-73 PGXC_NODE 字段

名称	类型	描述
node_name	name	节点名称。
node_type	"char"	节点类型。 C：协调节点。 D：数据节点。
node_port	integer	节点的端口号。
node_host	name	节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。
node_port1	integer	复制节点的端口号。
node_host1	name	复制节点的主机名称或者IP（如配置为虚拟IP，则为虚拟IP）。
hostis_primary	boolean	表明当前节点是否发生主备切换。
nodeis_primary	boolean	在replication表下，是否优选当前节点作为优先执行的节点进行非查询操作。
nodeis_preferred	boolean	在replication表下，是否优选当前节点作为首选的节点进行查询。
node_id	integer	节点标识符。
sctp_port	integer	主节点使用TCP代理通信库或SCTP通信库的数据通道监听端口。
control_port	integer	主节点使用TCP代理通信库或SCTP通信库的控制通道监听端口。
sctp_port1	integer	备节点使用TCP代理通信库或SCTP通信库的数据通道监听端口。
control_port1	integer	备节点使用TCP代理通信库或SCTP通信库的控制通道监听端口。
nodeis_central	boolean	当前节点为中心控制节点。

16.3 系统视图

16.3.1 ALL_ALL_TABLES

ALL_ALL_TABLES视图存储当前用户所能访问的表或视图。

表 16-74 ALL_ALL_TABLES 字段

名称	类型	描述
owner	name	表或视图的所有者
table_name	name	表或视图的名称
tablespace_name	name	表或视图所在的表空间

16.3.2 ALL_CONSTRAINTS

ALL_CONSTRAINTS视图存储当前用户可访问的约束的信息。

表 16-75 ALL_CONSTRAINTS 字段

名称	类型	描述
constraint_name	vcharacter varying(64)	约束名
constraint_type	text	约束类型 • c表示检查约束 • f表示外键约束 • p表示主键约束 • u表示唯一约束
table_name	character varying(64)	约束相关的表名
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）

16.3.3 ALL_CONS_COLUMNS

ALL_CONS_COLUMNS视图存储当前用户可访问的约束字段的信息。

表 16-76 ALL_CONS_COLUMNS 字段

名称	类型	描述
table_name	character varying(64)	约束相关的表名
column_name	character varying(64)	约束相关的列名

名称	类型	描述
constraint_name	character varying(64)	约束名
position	smallint	表中列的位置

16.3.4 ALL_COL_COMMENTS

ALL_COL_COMMENTS视图存储当前用户可访问的表中字段的注释信息。

表 16-77 ALL_COL_COMMENTS 字段

名称	类型	描述
column_name	character varying(64)	列名
table_name	character varying(64)	表名
owner	character varying(64)	表的所有者
comments	text	注释

16.3.5 ALL_DEPENDENCIES

ALL_DEPENDENCIES视图存储了当前用户可访问的函数、高级包之间的依赖关系。

须知

因为相关信息的限制，目前GaussDB(DWS)中，此表为空表，表内没有任何记录。

表 16-78 ALL_DEPENDENCIES 字段

名称	类型	描述
owner	character varying(30)	对象的所有者
name	character varying(30)	对象的名称
type	character varying(17)	对象的类型
referenced_owner	character varying(30)	引用对象的所有者
referenced_name	character varying(64)	引用对象的名称
referenced_type	character varying(17)	引用对象的类型
referenced_link_name	character varying(128)	引用对象的链接的名称
schemaid	numeric	当前schema的ID

名称	类型	描述
dependency_type	character varying(4)	依赖类型（REF或HARD）

16.3.6 ALL_IND_COLUMNS

ALL_IND_COLUMNS视图存储了当前用户可访问的所有索引的字段信息。

表 16-79 ALL_IND_COLUMNS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
column_name	name	列名
column_position	smallint	索引中列的位置

16.3.7 ALL_IND_EXPRESSIONS

ALL_IND_EXPRESSIONS视图存储了当前用户可访问的表达式索引的信息。

表 16-80 ALL_IND_EXPRESSIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
column_expression	text	定义列的基于函数的索引表达式
column_position	smallint	索引中列的位置

16.3.8 ALL_INDEXES

ALL_INDEXES视图存储了当前用户可访问的索引信息。

表 16-81 ALL_INDEXES 字段

名称	类型	描述
owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_name	character varying(64)	索引对应的表名
uniqueness	text	表示索引是否为唯一索引
generated	character varying(1)	表示索引的名称是否为系统生成
partitioned	character(3)	表示索引是否具有分区表的性质

16.3.9 ALL_OBJECTS

ALL_OBJECTS视图记录了当前用户可访问的数据库对象。

表 16-82 ALL_OBJECTS 字段

名称	类型	描述
owner	name	对象的所有者
object_name	name	对象的名称
object_id	oid	对象的OID
object_type	name	对象的类型
namespace	oid	对象所在的命名空间
created	timestamp with time zone	对象的创建时间
last_ddl_time	timestamp with time zone	对象的最后修改时间

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

16.3.10 ALL_PROCEDURES

ALL_PROCEDURES视图存储了当前用户可访问的所有存储过程或函数信息。

表 16-83 ALL_PROCEDURES 字段

名称	类型	描述
owner	name	对象的所有者
object_name	name	对象名称

16.3.11 ALL_SEQUENCES

ALL_SEQUENCES视图存储当前用户能够访问的所有序列。

表 16-84 ALL_SEQUENCES 字段

名称	类型	描述
sequence_owner	name	序列所有者
sequence_name	name	序列的名称
min_value	bigint	序列最小值
max_value	bigint	序列最大值
increment_by	bigint	序列的增量
cycle_flag	character(1)	表示序列是否是循环序列，取值为Y或N ● Y表示是循环序列 ● N表示不是循环序列

16.3.12 ALL_SOURCE

ALL_SOURCE视图存储当前用户可访问的存储过程或函数信息，且提供存储过程或函数定义的字段。

表 16-85 ALL_SOURCE 字段

名称	类型	描述
owner	name	对象的所有者
name	name	对象名称
type	name	对象的类型
text	text	对象的定义

16.3.13 ALL_SYNONYMS

ALL_SYNONYMS视图存储了当前用户可访问的所有同义词信息。

表 16-86 ALL_SYNONYMS 字段

名称	类型	描述
owner	text	同义词的所有者
schema_name	text	同义词所属模式名
synonym_name	text	同义词的名称
table_owner	text	关联对象的所有者
table_schema_name	text	关联对象所属模式名
table_name	text	关联对象名

16.3.14 ALL_TAB_COLUMNS

ALL_TAB_COLUMNS视图存储了当前用户可访问的表的列的描述信息。

表 16-87 ALL_TAB_COLUMNS 字段

名称	类型	描述
owner	character varying(64)	表的所有者。
table_name	character varying(64)	表名。
column_name	character varying(64)	列名。
data_type	character varying(128)	列的数据类型。
column_id	integer	对象创建或增加列时列的序号。
data_length	integer	列的字节长度。
avg_col_len	numeric	列的平均长度（单位字节）。
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。

名称	类型	描述
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2，bpchar，char类型有效。

16.3.15 ALL_TAB_COMMENTS

ALL_TAB_COMMENTS视图存储当前用户可访问的所有表和视图的注释信息。

表 16-88 ALL_TAB_COMMENTS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者
table_name	character varying(64)	表或视图的名称
comments	text	注释

16.3.16 ALL_TABLES

ALL_TABLES视图存储当前用户可访问的所有表。

表 16-89 ALL_TABLES 字段

名称	类型	描述
owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
tablespace_name	character varying(64)	表所在的表空间名称
status	character varying(8)	当前记录是否有效
temporary	character(1)	表是否为临时表 • Y表示是临时表 • N表示不是临时表
dropped	character varying	当前记录是否已删除 • YES表示已删除 • NO表示未删除
num_rows	numeric	表的估计行数

16.3.17 ALL_USERS

ALL_USERS视图存储记录数据库中所有用户，但不对用户信息进行详细的描述。

表 16-90 ALL_USERS 字段

名称	类型	描述
username	name	用户名
user_id	oid	用户的OID

16.3.18 ALL_VIEWS

ALL_VIEWS视图存储了当前用户可访问的所有视图描述信息。

表 16-91 ALL_VIEWS 字段

名称	类型	描述
owner	name	视图的所有者
view_name	name	视图的名称
text_length	integer	视图文本长度
text	text	视图文本

16.3.19 DBA_DATA_FILES

DBA_DATA_FILES视图存储关于数据库文件的描述。需要有系统管理员权限才可以访问。

表 16-92 DBA_DATA_FILES 字段

名称	类型	描述
tablespace_name	name	文件所属的表空间的名称
bytes	double precision	文件的字节长度

16.3.20 DBA_USERS

DBA_USERS视图存储关于数据库所有用户名信息。需要有系统管理员权限才可以访问。

表 16-93 DBA_USERS 字段

名称	类型	描述
username	character varying(64)	用户名

16.3.21 DBA_COL_COMMENTS

DBA_COL_COMMENTS视图存储关于数据库中表中字段的注释信息。需要有系统管理员权限才可以访问。

表 16-94 DBA_COL_COMMENTS 字段

名称	类型	描述
column_name	character varying(64)	列名
table_name	character varying(64)	表名
owner	character varying(64)	表的所有者
comments	text	注释

16.3.22 DBA_CONSTRAINTS

DBA_CONSTRAINTS视图存储关于数据库表中约束的信息。需要有系统管理员权限才可以访问。

表 16-95 DBA_CONSTRAINTS 字段

名称	类型	描述
constraint_name	vcharacter varying(64)	约束名
constraint_type	text	约束类型 • c表示检查约束 • f表示外键约束 • p表示主键约束 • u表示唯一约束
table_name	character varying(64)	约束相关的表名
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）

16.3.23 DBA_CONS_COLUMNS

DBA_CONS_COLUMNS视图存储关于数据库表中约束字段的信息。需要有系统管理员权限才可以访问。

表 16-96 DBA_CONS_COLUMNS 字段

名称	类型	描述
table_name	character varying(64)	约束相关的表名
column_name	character varying(64)	约束相关的列名
constraint_name	character varying(64)	约束名
position	smallint	表中列的位置

16.3.24 DBA_IND_COLUMNS

DBA_IND_COLUMNS视图存储关于数据库中所有索引的字段信息。需要有系统管理员权限才可以访问。

表 16-97 DBA_IND_COLUMNS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
column_name	name	列名
column_position	smallint	索引中列的位置

16.3.25 DBA_IND_EXPRESSIONS

DBA_IND_EXPRESSIONS视图存储了数据库中的表达式索引的信息。需要有系统管理员权限才可以访问。

表 16-98 DBA_IND_EXPRESSIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_owner	character varying(64)	表的所有者
table_name	character varying(64)	表名

名称	类型	描述
column_expression	text	定义列的基于函数的索引表达式
column_position	smallint	索引中列的位置

16.3.26 DBA_IND_PARTITIONS

DBA_IND_PARTITIONS视图存储数据库中所有索引分区的信息。数据库中每个分区表的每个索引分区（如果存在的话）在DBA_IND_PARTITIONS里都会有一行记录。需要有系统管理员权限才可以访问。

表 16-99 DBA_IND_PARTITIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引分区所属分区表索引的所有者的名称
schema	character varying(64)	索引分区所属分区表索引的模式
index_name	character varying(64)	索引分区所属分区表索引的名称
partition_name	character varying(64)	索引分区的名称
index_partition_usable	boolean	索引分区是否可用
high_value	text	索引分区所对应分区的上边界
def_tablespace_name	name	索引分区的表空间名称

16.3.27 DBA_INDEXES

DBA_INDEXES视图存储关于数据库下的所有索引信息。需要有系统管理员权限才可以访问。

表 16-100 DBA_INDEXES 字段

名称	类型	描述
owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_name	character varying(64)	索引对应的表名
uniqueness	text	表示索引是否为唯一索引

名称	类型	描述
generated	character varying(1)	表示索引名称是否为系统生成
partitioned	character(3)	表示索引是否具有分区表的性质

16.3.28 DBA_OBJECTS

DBA_OBJECTS视图存储了数据库中所有数据库对象。需要有系统管理员权限才可以访问。

表 16-101 DBA_OBJECTS 字段

名称	类型	描述
owner	name	对象的所有者
object_name	name	对象的名称
object_id	oid	对象的OID
object_type	name	对象的类型
namespace	oid	对象所在的命名空间
created	timestamp with time zone	对象的创建时间
last_ddl_time	timestamp with time zone	对象的最后修改时间

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

16.3.29 DBA_PART_INDEXES

DBA_PART_INDEXES视图存储数据库中所有分区表索引的信息。需要有系统管理员权限才可以访问。

表 16-102 DBA_PART_INDEXES 字段

名称	类型	描述
index_owner	character varying(64)	分区表索引的所有者名称

名称	类型	描述
schema	character varying(64)	分区表索引的模式
index_name	character varying(64)	分区表索引的名称
table_name	character varying(64)	分区表索引所属的分区表名称
partitioning_type	text	分区表的分区策略 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）。
partition_count	bigint	分区表索引的索引分区的个数
def_tablespace_name	name	分区表索引的表空间名称
partitioning_key_count	integer	分区表的分区键个数

16.3.30 DBA_PART_TABLES

DBA_PART_TABLES视图存储数据中所有分区表的信息。需要有系统管理员权限才可以访问。

表 16-103 DBA_PART_TABLES 字段

名称	类型	描述
table_owner	character varying(64)	分区表的所有者名称
schema	character varying(64)	分区表的模式
table_name	character varying(64)	分区表的名称
partitioning_type	text	分区表的分区策略 **说明** 当前分区表策略仅支持范围分区（Range Partitioning）。
partition_count	bigint	分区表的分区个数
def_tablespace_name	name	分区表的表空间名称
partitioning_key_count	integer	分区表的分区键个数

16.3.31 DBA_PROCEDURES

DBA_PROCEDURES视图存储关于数据库下的所有存储过程或函数信息。需要有系统管理员权限才可以访问。

表 16-104 DBA_PROCEDURES 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者
object_name	character varying(64)	存储过程或函数名称
argument_number	smallint	存储过程入参个数

16.3.32 DBA_SEQUENCES

DBA_SEQUENCES视图存储关于数据库下的所有序列信息。需要有系统管理员权限才可以访问。

表 16-105 DBA_SEQUENCES 字段

名称	类型	描述
sequence_owner	character varying(64)	序列的所有者
sequence_name	character varying(64)	序列的名称

16.3.33 DBA_SOURCE

DBA_SOURCE视图存储关于数据库下的所有存储过程或函数信息，且提供存储过程或函数定义的字段。需要有系统管理员权限才可以访问。

表 16-106 DBA_SOURCE 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者
name	character varying(64)	存储过程或函数的名称
text	text	存储过程或函数的定义

16.3.34 DBA_SYNONYMS

DBA_SYNONYMS视图存储关于数据库下的所有同义词信息。需要有系统管理员权限才可以访问。

表 16-107 DBA_SYNONYMS 字段

名称	类型	描述
owner	text	同义词的所有者
schema_name	text	同义词所属模式名
synonym_name	text	同义词的名称
table_owner	text	关联对象的所有者
table_schema_name	text	关联对象所属模式名
table_name	text	关联对象名

16.3.35 DBA_TAB_COLUMNS

DBA_TAB_COLUMNS视图存储关于表的字段的信息。数据库里每个表的每个字段都在DBA_TAB_COLUMNS里有一行。需要有系统管理员权限才可以访问。

表 16-108 DBA_TAB_COLUMNS 字段

名称	类型	描述
owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
column_name	character varying(64)	列名
data_type	character varying(128)	列的数据类型
column_id	integer	创建表时列的序号
data_length	integer	列的字节长度
comments	text	注释
avg_col_len	numeric	列的平均长度（单位字节）
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0

名称	类型	描述
char_length	numeric	列的长度（以字符计），只对varchar，nvarchar2， bpchar，char类型有效

16.3.36 DBA_TAB_COMMENTS

DBA_TAB_COMMENTS视图存储关于数据库下的所有表和视图的注释信息。需要有系统管理员权限才可以访问。

表 16-109 DBA_TAB_COMMENTS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者
table_name	character varying(64)	表或视图的名称
comments	text	注释

16.3.37 DBA_TAB_PARTITIONS

DBA_TAB_PARTITIONS视图提供数据库中所有分区的信息。

表 16-110 DBA_TAB_PARTITIONS 字段

名称	类型	描述
table_owner	character varying(64)	分区所在表的所有者
schema	character varying(64)	分区表模式
table_name	character varying(64)	表名
partition_name	character varying(64)	分区的名称
high_value	text	范围分区和间隔分区的上边界
tablespace_name	name	分区所在表空间的名称

16.3.38 DBA_TABLES

DBA_TABLES视图存储关于数据库下的所有表信息。需要有系统管理员权限才可以访问。

表 16-111 DBA_TABLES 字段

名称	类型	描述
owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
tablespace_name	character varying(64)	表所在的表空间的名称
status	character varying(8)	当前记录是否有效
temporary	character(1)	是否为临时表 ● Y表示是临时表 ● N表示不是临时表
dropped	character varying	当前记录是否已删除 ● YES表示已删除 ● NO表示未删除
num_rows	numeric	表的估计行数

16.3.39 DBA_TABLESPACES

DBA_TABLESPACES视图存储有关可用的表空间的信息。需要有系统管理员权限才可以访问。

表 16-112 DBA_TABLESPACES 字段

名称	类型	描述
tablespace_name	character varying(64)	表空间的名称

16.3.40 DBA_TRIGGERS

DBA_TRIGGERS视图存储关于数据库内的触发器信息。需要有系统管理员权限才可以访问。

表 16-113 DBA_TRIGGERS 字段

名称	类型	描述
trigger_name	character varying(64)	触发器名称
table_name	character varying(64)	定义触发器的表的名称
table_owner	character varying(64)	定义触发器的表的所有者

16.3.41 DBA_VIEWS

DBA_VIEWS视图存储关于数据库内的视图信息。需要有系统管理员权限才可以访问。

表 16-114 DBA_VIEWS 字段

名称	类型	描述
owner	character varying(64)	视图的所有者
view_name	character varying(64)	视图的名称

16.3.42 DUAL

DUAL视图是数据库根据数据字典自动创建的，它只有一个文本字段，且只有一行，用于保存表达式计算结果。任何用户都可以访问它。

表 16-115 DUAL 字段

名称	类型	描述
dummy	text	表达式计算结果

16.3.43 GLOBAL_WORKLOAD_SQL_COUNT

GLOBAL_WORKLOAD_SQL_COUNT视图显示集群中所有Workload控制组内SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。

表 16-116 GLOBAL_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
workload	name	Workload控制组名称
select_count	bigint	SELECT数量
update_count	bigint	UPDATE数量
insert_count	bigint	INSERT数量
delete_count	bigint	DELETE数量
ddl_count	bigint	DDL数量
dml_count	bigint	DML数量
dcl_count	bigint	DCL数量

16.3.44 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME

GLOBAL_WORKLOAD_SQL_ELAPSE_TIME视图显示集群中所有Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。

表 16-117 GLOBAL_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
workload	name	Workload控制组名称
total_select_elapse	bigint	SELECT总响应时间
max_select_elapse	bigint	SELECT最大响应时间
min_select_elapse	bigint	SELECT最小响应时间
avg_select_elapse	bigint	SELECT平均响应时间
total_update_elapse	bigint	UPDATE总响应时间
max_update_elapse	bigint	UPDATE最大响应时间
min_update_elapse	bigint	UPDATE最小响应时间
avg_update_elapse	bigint	UPDATE平均响应时间
total_insert_elapse	bigint	INSERT总响应时间
max_insert_elapse	bigint	INSERT最大响应时间
min_insert_elapse	bigint	INSERT最小响应时间
avg_insert_elapse	bigint	INSERT平均响应时间
total_delete_elapse	bigint	DELETE总响应时间
max_delete_elapse	bigint	DELETE最大响应时间
min_delete_elapse	bigint	DELETE最小响应时间
avg_delete_elapse	bigint	DELETE平均响应时间

16.3.45 GS_ALL_CONTROL_GROUP_INFO

GS_ALL_CONTROL_GROUP_INFO视图显示数据库内所有的控制组信息。

表 16-118 GS_ALL_CONTROL_GROUP_INFO 字段

名称	类型	描述
name	text	控制组的名称
type	text	控制组的类型

名称	类型	描述
gid	bigint	控制组ID
classgid	bigint	Workload所属Class的控制组ID
class	text	Class控制组
workload	text	Workload控制组
shares	bigint	控制组分配的CPU资源配额
limits	bigint	控制组分配的CPU资源限额
wdlevel	bigint	Workload控制组层级
cpucores	text	控制组使用的CPU核的信息

16.3.46 GS_CLUSTER_RESOURCE_INFO

GS_CLUSTER_RESOURCE_INFO视图显示的是所有DN资源的汇总信息。

表 16-119 GS_CLUSTER_RESOURCE_INFO 字段

名称	类型	描述
min_mem_util	integer	DN最小内存使用率
max_mem_util	integer	DN最大内存使用率
min_cpu_util	integer	DN最小CPU使用率
max_cpu_util	integer	DN最大CPU使用率
min_io_util	integer	DN最小IO使用率
max_io_util	integer	DN最大IO使用率
used_mem_rate	integer	物理节点最大内存使用率

16.3.47 GS_INSTR_UNIQUE_SQL

Unique SQL 定义

数据库将接收到的每个SQL的文本字符串，都进行解析并生成内部解析树，遍历解析树并忽略其中的常数值，以一定的算法计算出来一个整数值作为Unique SQL ID，用来唯一标识这一类SQL，Unique SQL ID相同的一类SQL就叫做Unique SQL。

示例

假如，用户先后输入SQL:

```
select * from t1 where id = 1;
select * from t1 where id = 2;
```

那么，这两条SQL的统计信息会汇聚到同一个Unique SQL上：

```
select * from t1 where id = ?;
```

GS_INSTR_UNIQUE_SQL 视图

GS_INSTR_UNIQUE_SQL视图显示当前节点收集的Unique SQL的执行信息，主要包括以下内容：

- Unique SQL ID以及归一化后的SQL文本字符串，归一化后的SQL文本如**示例**中所示。
- 执行次数（成功执行的次数），响应时间（数据库内部的SQL执行时间，包括最大、最小和总时间）。
- Cache/IO信息，包含block的物理读、逻辑读次数，仅统计执行成功的SQL在各DN节点上的相关信息。该统计值与查询执行当时所处理的数据量、所使用的内存、是否多次执行、内存管理策略、是否有其他并发查询等因素相关，反映整个查询执行过程中的buffer块物理读和逻辑读次数，不同时间执行可能统计值不同。
- 行活动，包含SELECT语句的结果集返回行数、更新行、插入行、删除行、顺序扫描行、随机扫描行等信息。除结果集返回行数与该SELECT语句的结果集行数一致、且仅在CN上记录外，其他行活动信息均在DN上记录，且统计数值反应的是整个查询执行过程中的行活动，包括对相关系统表、元数据表、数据表等做必要的扫描和修改，与对应数据量以及相关参数设置相关，即统计数值将会大于等于对实际数据表的扫描和修改。
- 时间分布，包含：DB_TIME/CPU_TIME/EXECUTION_TIME/PARSE_TIME/PLAN_TIME/REWRITE_TIME/PL_EXECUTION_TIME/PL_COMPILATION_TIME/NET_SEND_TIME/DATA_IO_TIME，相关定义见**表1**。该信息在CN和DN节点均有统计，视图查询时将汇总展示。
- 软硬解析次数，包含软解析（缓存计划）、硬解析（生成计划）的次数，即如果本次执行的是之前缓存的计划，软解析次数+1，如果本次执行的计划是重新生成的，则硬解析次数+1。该次数在CN和DN节点上都会统计，视图查询时将汇总展示。

Unique SQL收集功能存在以下约束：

- 只有执行成功的SQL才会显示其详细的统计信息，否则可能只记录query、node、user等信息。
- 根据当前的Unique SQL ID生成算法，相同的SQL在不同节点执行可能生成不同的ID。
- 如果开启Unique SQL收集功能，CN节点将对所有接受到的查询进行统计收集，包括工具和用户的查询等。
- 若一条SQL语句内部包含执行多条SQL语句、类似存储过程执行等场景，仅会对最外层SQL生成一条Unique SQL，所有子SQL的统计信息都会汇总到该Unique SQL记录上。
- Unique SQL的响应时间统计中不完全包含NET_SEND_TIME阶段的时间，所以EXECUTION_TIME和elapse_time等时间之间不存在大小比较关系。

普通用户访问GS_INSTR_UNIQUE_SQL视图，只能看到该用户相关的Unique SQL信息，管理员用户可以看到当前节点所有的Unique SQL信息。CN和DN上均可查询GS_INSTR_UNIQUE_SQL视图，DN上显示的是本节点内的Unique SQL统计信息，CN上显示的是本节点Unique SQL完整统计信息，即该CN节点会收集其他CN和DN上对应

该CN的Unique SQL的执行信息，进行汇总展示。通过查询GS_INSTR_UNIQUE_SQL视图，能够定位由于消耗不同资源导致的Top SQL，为集群性能调优和维护提供依据。

表 16-120 GS_INSTR_UNIQUE_SQL 字段

名称	类型	描述
node_name	name	接收SQL的CN节点名称
node_id	integer	节点ID，等同于pgxc_node表中node_id
user_name	name	用户名称
user_id	oid	用户ID
unique_sql_id	bigint	归一化的UNIQUE SQL ID
query	text	归一化的SQL文本
n_calls	bigint	成功执行次数
min_elapse_time	bigint	SQL在数据库内的最小运行时间（单位：微秒）
max_elapse_time	bigint	SQL在数据库内的最大运行时间（单位：微秒）
total_elapse_time	bigint	SQL在数据库内的总运行时间（单位：微秒）
n_returned_rows	bigint	行活动-SELECT语句返回的结果集行数
n_tuples_fetched	bigint	行活动-随机扫描行（列存表/外表不统计）
n_tuples_returned	bigint	行活动-顺序扫描行（列存表/外表不统计）
n_tuples_inserted	bigint	行活动-插入行数
n_tuples_updated	bigint	行活动-更新行数
n_tuples_deleted	bigint	行活动-删除行数
n_blocks_fetched	bigint	buffer的块访问次数，即物理读/IO
n_blocks_hit	bigint	buffer的块命中次数，即逻辑读/Cache
n_soft_parse	bigint	软解析次数（缓存计划）
n_hard_parse	bigint	硬解析次数（生成计划）

名称	类型	描述
db_time	bigint	有效的DB执行时间，包含等待时间、网络发送时间等，若查询执行涉及到多线程，DB_TIME是多个线程的DB_TIME之和（单位：微秒）
cpu_time	bigint	CPU的执行时间，不包含sleep时间（单位：微秒）
execution_time	bigint	查询执行器内的SQL执行时间，DDL语句、以及某些不经过执行器执行的语句（例如Copy语句）不计数（单位：微秒）
parse_time	bigint	SQL解析时间（单位：微秒）
plan_time	bigint	SQL生成计划时间（单位：微秒）
rewrite_time	bigint	SQL重写时间（单位：微秒）
pl_execution_time	bigint	plpgsql过程化语言函数上的执行时间（单位：微秒）
pl_compilation_time	bigint	plpgsql过程化语言函数上的编译时间（单位：微秒）
net_send_time	bigint	网络时间，包含CN向客户端发送数据、DN向CN发送数据等时间（单位：微秒）
data_io_time	bigint	IO时间，文件IO耗时（单位：微秒）

16.3.48 GS_SESSION_CPU_STATISTICS

GS_SESSION_CPU_STATISTICS视图显示和当前用户执行复杂作业正在运行时的负载管理CPU使用的信息。

表 16-121 GS_SESSION_CPU_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据库OID

名称	类型	描述
usename	name	登录到该后端的用户名
pid	bigint	后端线程ID
start_time	timestamp with time zone	语句执行的开始时间
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位为ms
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位为ms
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位为ms
query	text	正在执行的语句
node_group	text	语句所属用户对应的逻辑集群

16.3.49 GS_SESSION_MEMORY_STATISTICS

GS_SESSION_MEMORY_STATISTICS视图显示和当前用户执行复杂作业正在运行时的负载管理内存使用的信息。

表 16-122 GS_SESSION_MEMORY_STATISTICS 字段

名称	类型	描述
datid	oid	连接后端的数据库OID
usename	name	登录到该后端的用户名
pid	bigint	后端线程ID
start_time	timestamp with time zone	语句执行的开始时间
min_peak_memory	integer	语句在所有DN上的最小内存峰值大小，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值大小，单位MB。
spill_info	text	语句在所有DN上的下盘信息 None：所有DN均未下盘 All: 所有DN均下盘 [a:b]: 数量为b个DN中有a个DN下盘
query	text	正在执行的语句

名称	类型	描述
node_group	text	语句所属用户对应的逻辑集群

16.3.50 GS_SQL_COUNT

GS_SQL_COUNT视图显示数据库当前节点当前时刻执行的五类语句（SELECT、INSERT、UPDATE、DELETE、MERGE INTO）统计信息，包括执行次数和响应时间（除MERGE INTO语句外，统计其他四类语句的最大、最小、平均和总响应时间，单位为微秒），以及DDL、DML、DCL类型语句的执行次数。

GS_SQL_COUNT视图中对DDL、DML、DCL类型语句分类与SQL语法中略有不同，具体如下：

- CREATE/ALTER/DROP USER，CREATE/ALTER/DROP ROLE等用户相关语句属于DCL类型。
- BEGIN/COMMIT/SET CONSTRAINTS/ROLLBACK/SAVEPOINT/START等事务相关语句属于DCL类型。
- ALTER SYSTEM KILL SESSION等价于SELECT pg_terminate_backend()语句，属于DML类型。

其余语句的分类与SQL语法中定义类似。

普通用户查询GS_SQL_COUNT视图仅能看到该用户当前节点的统计信息。管理员权限用户查询GS_SQL_COUNT视图则能看到所有用户当前节点的统计信息。当集群或该节点重启时，计数会清零，并重新开始计数 。计数以节点收到的查询数为准，包括集群内部进行的查询。例如，CN收到一条查询，若下发多条查询DN，将在DN上进行相应次数的计数。

表 16-123 GS_SQL_COUNT 字段

名称	类型	描述
node_name	name	节点名称
user_name	name	用户名
select_count	bigint	SELECT数量
update_count	bigint	UPDATE数量
insert_count	bigint	INSERT数量
delete_count	bigint	DELETE数量
mergeinto_count	bigint	MERGE INTO数量
ddl_count	bigint	DDL数量
dml_count	bigint	DML数量
dcl_count	bigint	DCL数量

名称	类型	描述
total_select_elapse	bigint	SELECT总响应时间
avg_select_elapse	bigint	SELECT平均响应时间
max_select_elapse	bigint	SELECT最大响应时间
min_select_elapse	bigint	SELECT最小响应时间
total_update_elapse	bigint	UPDATE总响应时间
avg_update_elapse	bigint	UPDATE平均响应时间
max_update_elapse	bigint	UPDATE最大响应时间
min_update_elapse	bigint	UPDATE最小响应时间
total_delete_elapse	bigint	DELETE总响应时间
avg_delete_elapse	bigint	DELETE平均响应时间
max_delete_elapse	bigint	DELETE最大响应时间
min_delete_elapse	bigint	DELETE最小响应时间
total_insert_elapse	bigint	INSERT总响应时间
avg_insert_elapse	bigint	INSERT平均响应时间
max_insert_elapse	bigint	INSERT最大响应时间
min_insert_elapse	bigint	INSERT最小响应时间

16.3.51 GS_WLM_CGROUP_INFO

GS_WLM_CGROUP_INFO视图显示当前执行作业的控制组的信息。

表 16-124 GS_WLM_CGROUP_INFO 字段

名称	类型	描述
cgoup_name	text	控制组的名称
priority	interger	作业的优先级
usage_pecent	interger	控制组占用的百分比
shares	bigint	控制组分配的CPU资源配额
cpuacct	bigint	CPU配额分配
cpuset	text	CPU限额分配
relpath	text	控制组的相对路径
valid	text	该控制组是否有效
node_group	text	逻辑集群名称

16.3.52 GS_WLM_OPERATOR_HISTORY

GS_WLM_OPERATOR_HISTORY视图显示的是当前用户当前CN上执行作业结束后的算子的相关记录。

此视图用于Database Manager从内核中查询数据，内核中的数据会定时被清理，清理周期为3分钟。

16.3.53 GS_WLM_OPERATOR_STATISTICS

GS_WLM_OPERATOR_STATISTICS视图显示当前用户正在执行的作业的算子相关信息。

表 16-125 GS_WLM_OPERATOR_STATISTICS 的字段

名称	类型	描述
queryid	bigint	语句执行使用的内部query_id。
pid	bigint	后端线程id。
plan_node_id	integer	查询对应的执行计划的plan node id。
plan_node_name	text	对应于plan_node_id的算子的名称。
start_time	timestamp with time zone	该算子处理第一条数据的开始时间。
duration	bigint	该算子到结束时候总的执行时间(ms)。
status	text	当前算子的执行状态，包括finished和running。

名称	类型	描述
query_dop	integer	当前算子执行时的并行度。
estimated_rows	bigint	优化器估算的行数信息。
tuple_processed	bigint	当前算子返回的元素个数。
min_peak_memory	integer	当前算子在所有DN上的最小内存峰值(MB)。
max_peak_memory	integer	当前算子在所有DN上的最大内存峰值(MB)。
average_peak_memory	integer	当前算子在所有DN上的平均内存峰值(MB)。
memory_skew_percent	integer	当前算子在各DN间的内存使用倾斜率。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_cpu_time	bigint	该算子在所有DN上的最小执行时间(ms)。
max_cpu_time	bigint	该算子在所有DN上的最大执行时间(ms)。
total_cpu_time	bigint	该算子在所有DN上的总执行时间(ms)。
cpu_skew_percent	integer	DN间执行时间的倾斜率。
warning	text	主要显示如下几类告警信息： 1. Sort/SetOp/HashAgg/HashJoin spill 2. Spill file size large than 256MB 3. Broadcast size large than 100MB 4. Early spill 5. Spill times is greater than 3 6. Spill on memory adaptive 7. Hash table conflict

16.3.54 GS_WLM_SESSION_HISTORY

GS_WLM_SESSION_HISTORY视图显示当前用户在当前CN上执行作业结束后的负载管理记录。此视图用于Database Manager从GaussDB(DWS)中查询数据，GaussDB(DWS)中的数据会定时被清理，清理周期为3分钟。

表 16-126 GS_WLM_SESSION_HISTORY 的字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
dbname	text	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN名称。
username	text	连接到后端的用户名。
application_name	text	连接到后端的应用名。
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标示作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
block_time	bigint	语句执行前的阻塞时间，包含语句解析和优化时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
finish_time	timestamp with time zone	语句执行的结束时间。
duration	bigint	语句实际执行的时间，单位ms。
estimate_total_time	bigint	语句预估执行时间，单位ms。
status	text	语句执行结束状态：正常为finished，异常为aborted。
abort_info	text	语句执行结束状态为aborted时显示异常信息。

名称	类型	描述
resource_pool	text	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All: 所有DN均下盘。 [a:b]: 数量为b个DN中有a个DN下盘。
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	integer	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。

名称	类型	描述
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
warning	text	主要显示如下几类告警信息以及**SQL自诊断调优相关告警**: 1. Spill file size large than 256MB 2. Broadcast size large than 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict
queryid	bigint	语句执行使用的内部query id。
query	text	执行的语句。
query_plan	text	语句的执行计划。
node_group	text	语句所属用户对应的逻辑集群。

16.3.55 GS_WLM_SESSION_STATISTICS

GS_WLM_SESSION_STATISTICS视图显示当前用户在当前CN上正在执行的作业的负载管理记录。

表 16-127 GS_WLM_SESSION_STATISTICS 的字段

名称	类型	描述
datid	oid	连接后端的数据OID。
dbname	name	连接后端的数据库名称。
schemaname	text	模式名。
nodename	text	语句执行的CN节点名称。
username	name	连接到后端的用户名。
application_name	text	连接到后端的应用名。

名称	类型	描述
client_addr	inet	连接到后端的客户端的IP地址。 如果此字段是null，它表明通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，这个字段是通过client_addr的反向DNS查找得到。这个字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
query_band	text	用于标示作业类型，可通过GUC参数query_band进行设置，默认为空字符串。
pid	bigint	后端线程ID。
block_time	bigint	语句执行前的阻塞时间，单位ms。
start_time	timestamp with time zone	语句执行的开始时间。
duration	bigint	语句已经执行的时间，单位ms。
estimate_total_time	bigint	语句执行预估总时间，单位ms。
estimate_left_time	bigint	语句执行预估剩余时间，单位ms。
enqueue	text	工作负载管理资源状态。
resource_pool	name	用户使用的资源池。
control_group	text	语句所使用的Cgroup。
estimate_memory	integer	语句预估使用内存，单位MB。
min_peak_memory	integer	语句在所有DN上的最小内存峰值，单位MB。
max_peak_memory	integer	语句在所有DN上的最大内存峰值，单位MB。
average_peak_memory	integer	语句执行过程中的内存使用平均值，单位MB。
memory_skew_percent	integer	语句在各DN间的内存使用倾斜率。
spill_info	text	语句在所有DN上的下盘信息： None：所有DN均未下盘。 All: 所有DN均下盘。 [a:b]: 数量为b个DN中有a个DN下盘。

名称	类型	描述
min_spill_size	integer	若发生下盘，所有下盘DN的最小下盘数据量(MB)，默认为0。
max_spill_size	integer	若发生下盘，所有下盘DN的最大下盘数据量(MB)，默认为0。
average_spill_size	integer	若发生下盘，所有下盘DN的平均下盘数据量(MB)，默认为0。
spill_skew_percent	integer	若发生下盘，DN间下盘倾斜率。
min_dn_time	bigint	语句在所有DN上的最小执行时间，单位ms。
max_dn_time	bigint	语句在所有DN上的最大执行时间，单位ms。
average_dn_time	bigint	语句在所有DN上的平均执行时间，单位ms。
dntime_skew_percent	bigint	语句在各DN间的执行时间倾斜率。
min_cpu_time	bigint	语句在所有DN上的最小CPU时间，单位ms。
max_cpu_time	bigint	语句在所有DN上的最大CPU时间，单位ms。
total_cpu_time	bigint	语句在所有DN上的CPU总时间，单位ms。
cpu_skew_percent	integer	语句在各DN间的CPU时间倾斜率。
min_peak_iops	integer	语句在所有DN上的每秒最小IO峰值（列存单位是次/s，行存单位是万次/s）。
max_peak_iops	integer	语句在所有DN上的每秒最大IO峰值（列存单位是次/s，行存单位是万次/s）。
average_peak_iops	integer	语句在所有DN上的每秒平均IO峰值（列存单位是次/s，行存单位是万次/s）。
iops_skew_percent	integer	语句在DN间的IO倾斜率。
warning	text	主要显示如下几类告警信息以及**SQL自诊断调优相关告警**: 1. Spill file size large than 256MB 2. Broadcast size large than 100MB 3. Early spill 4. Spill times is greater than 3 5. Spill on memory adaptive 6. Hash table conflict
queryid	bigint	语句执行使用的内部query id。

名称	类型	描述
query	text	正在执行的语句。
query_plan	text	语句的执行计划。
node_group	text	语句所属用户对应的逻辑集群。

16.3.56 GS_WLM_WORKLOAD_RECORDS

GS_WLM_WORKLOAD_RECORDS视图显示当前用户在每个CN上执行作业时在CN上的状态信息。

表 16-128 GS_WLM_WORKLOAD_RECORDS 字段

名称	类型	描述
node_name	text	作业执行所在CN的名称
thread_id	bigint	后端线程ID
processid	integer	后端线程的pid
time_stamp	name	语句执行的开始时间
username	interger	登录到该后端的用户名
memory	interger	语句所需的内存大小
active_points	interger	语句在资源池上消耗的资源点数
priority	interger	作业的优先级
resource_pool	text	作业所在资源池
status	text	作业执行的状态，包括： ● pending： 阻塞状态 ● running: 执行状态 ● finished：结束状态 ● aborted：终止状态 ● unknown：未知状态
control_group	text	作业所使用的Cgroups
enqueue	text	作业的排队信息，包括： ● GLOBAL：全局排队 ● RESPOOL: 资源池排队 ● ACTIVE：不排队
query	text	正在执行的语句
node_group	text	逻辑集群名称

16.3.57 GS_WORKLOAD_SQL_COUNT

GS_WORKLOAD_SQL_COUNT视图显示当前节点上Workload控制组内的SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。

表 16-129 GS_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
workload	name	Workload控制组名称
select_count	bigint	SELECT数量
update_count	bigint	UPDATE数量
insert_count	bigint	INSERT数量
delete_count	bigint	DELETE数量
ddl_count	bigint	DDL数量
dml_count	bigint	DML数量
dcl_count	bigint	DCL数量

16.3.58 GS_WORKLOAD_SQL_ELAPSE_TIME

GS_WORKLOAD_SQL_ELAPSE_TIME视图显示当前节点上Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。

表 16-130 GS_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
workload	name	Workload控制组名称
total_select_elapse	bigint	SELECT总响应时间
max_select_elapse	bigint	SELECT最大响应时间
min_select_elapse	bigint	SELECT最小响应时间
avg_select_elapse	bigint	SELECT平均响应时间
total_update_elapse	bigint	UPDATE总响应时间
max_update_elapse	bigint	UPDATE最大响应时间
min_update_elapse	bigint	UPDATE最小响应时间
avg_update_elapse	bigint	UPDATE平均响应时间
total_insert_elapse	bigint	INSERT总响应时间

名称	类型	描述
max_insert_elapse	bigint	INSERT最大响应时间
min_insert_elapse	bigint	INSERT最小响应时间
avg_insert_elapse	bigint	INSERT平均响应时间
total_delete_elapse	bigint	DELETE总响应时间
max_delete_elapse	bigint	DELETE最大响应时间
min_delete_elapse	bigint	DELETE最小响应时间
avg_delete_elapse	bigint	DELETE平均响应时间

16.3.59 GS_WORKLOAD_TRANSACTION

GS_WORKLOAD_TRANSACTION视图提供查询单CN上Workload控制组相关的事务信息。数据库记录每个Workload控制组事务提交和回滚的次数及事务提交和回滚的响应时间，单位是微秒。

表 16-131 GS_WORKLOAD_TRANSACTION 字段

名称	类型	描述
workload	name	Workload控制组名称
commit_counter	bigint	提交次数
rollback_counter	bigint	回滚次数
resp_min	bigint	最小响应时间
resp_max	bigint	最大响应时间
resp_avg	bigint	平均响应时间
resp_total	bigint	响应时间总和

16.3.60 GS_STAT_DB_CU

GS_STAT_DB_CU视图查询集群各个节点，每个数据库的CU命中情况。可以通过gs_stat_reset()进行清零。

表 16-132 GS_STAT_DB_CU 字段

名称	类型	描述
node_name1	text	节点名称
db_name	text	数据库名称

名称	类型	描述
mem_hit	integer	内存命中次数
hdd_sync_read	integer	硬盘同步读次数
hdd_asyn_read	integer	硬盘异步读次数

16.3.61 GS_STAT_SESSION_CU

GS_STAT_SESSION_CU视图查询当前集群各个节点，当前运行session的CU命中情况。session退出相应的统计数据会清零。集群重启后，统计数据也会清零。

表 16-133 GS_STAT_SESSION_CU 字段

名称	类型	描述
node_name1	text	节点名称
mem_hit	integer	内存命中次数
hdd_sync_read	integer	硬盘同步读次数
hdd_asyn_read	integer	硬盘异步读次数

16.3.62 GS_TOTAL_NODEGROUP_MEMORY_DETAIL

GS_TOTAL_NODEGROUP_MEMORY_DETAIL视图统计当前数据库逻辑集群使用内存的信息，单位为MB。

表 16-134 GS_TOTAL_NODEGROUP_MEMORY_DETAIL 字段

名称	类型	描述
ngname	text	逻辑集群名称

名称	类型	描述
memorytype	text	内存类型，包括以下几种： • ng_total_memory：该逻辑集群的总内存大小 • ng_used_memory：该逻辑集群的实际使用内存大小 • ng_estimate_memory：该逻辑集群的估算使用内存大小 • ng_foreignrp_memsize：该逻辑集群的外部资源池的总内存大小 • ng_foreignrp_usedsize：该逻辑集群的外部资源池实际使用内存大小 • ng_foreignrp_peaksize：该逻辑集群的外部资源池使用内存的峰值 • ng_foreignrp_mempct：该逻辑集群的外部资源池占该逻辑集群总内存大小的百分比 • ng_foreignrp_estmsize：该逻辑集群的外部资源池估算使用内存大小
memorymbytes	integer	内存类型分配内存的大小

16.3.63 GS_USER_TRANSACTION

GS_USER_TRANSACTION视图提供查询单CN上用户相关的事务信息。数据库记录每个用户事务提交和回滚的次数及事务提交和回滚的响应时间，单位是微秒。

表 16-135 GS_USER_TRANSACTION 字段

名称	类型	描述
usename	name	用户名称
commit_counter	bigint	提交次数
rollback_counter	bigint	回滚次数
resp_min	bigint	最小响应时间
resp_max	bigint	最大响应时间
resp_avg	bigint	平均响应时间
resp_total	bigint	响应时间总和

16.3.64 PG_AVAILABLE_EXTENSION_VERSIONS

PG_AVAILABLE_EXTENSION_VERSIONS视图显示数据库中某些特性的扩展版本信息。

表 16-136 PG_AVAILABLE_EXTENSION_VERSIONS 字段

名称	类型	描述
name	name	扩展名
version	text	版本名
installed	boolean	如果此扩展的版本当前已经安装，则为真
superuser	boolean	如果只允许系统管理员安装此扩展，则为真
relocatable	boolean	如果扩展可以重新加载到另一个模式，则为真
schema	name	扩展必须安装到的模式名，如果部分或全部可重新定位，则为NULL
requires	name[]	必备扩展的名称，如果没有则为NULL
comment	text	扩展的控制文件的注释字符串

16.3.65 PG_AVAILABLE_EXTENSIONS

PG_AVAILABLE_EXTENSIONS视图显示数据库中某些特性的扩展信息。

表 16-137 PG_AVAILABLE_EXTENSIONS 字段

名称	类型	描述
name	name	扩展名
default_version	text	缺省版本名，如果没有指定则为NULL
installed_version	text	当前安装的扩展版本，如果未安装则为NULL
comment	text	扩展控制文件的注释字符串

16.3.66 PG_COMM_CLIENT_INFO

PG_COMM_CLIENT_INFO视图存储单个节点客户端连接信息（DN上查询该视图显示CN连接DN的信息）。

表 16-138 PG_COMM_CLIENT_INFO 字段

名称	类型	描述
node_name	text	当前节点的名称。
app	text	客户端应用名。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。

名称	类型	描述
query_id	bigint	查询ID，对应debug_query_id。
socket	integer	如果是物理连接，显示socket。
remote_ip	text	对端节点IP。
remote_port	text	对端节点port。
logic_id	integer	如果是逻辑连接，显示sid，显示-1时表示当前连接是物理连接。

16.3.67 PG_COMM_DELAY

PG_COMM_DELAY视图展示单个DN的通信库时延状态。

表 16-139 PG_COMM_DELAY 字段

名称	类型	描述
node_name	text	节点名称
remote_name	text	连接对端节点名称
remote_host	text	连接对端IP地址
stream_num	integer	当前物理连接使用的stream逻辑连接数量
min_delay	integer	当前物理连接一分钟内探测到的最小时延，单位微秒 **说明** 负数结果无效，请重新等待时延状态更新后再执行。
average	integer	当前物理连接一分钟内探测时延的平均值，单位微秒
max_delay	integer	当前物理连接一分钟内探测到的最大时延，单位微秒

16.3.68 PG_COMM_STATUS

PG_COMM_STATUS视图展示单个DN的通信库状态。

表 16-140 PG_COMM_STATUS 字段

名称	类型	描述
node_name	text	节点名称。
rxpck/s	integer	节点通信库接收速率，单位Byte/s。

名称	类型	描述
txpck/s	integer	节点通信库发送速率，单位Byte/s。
rxkB/s	bigint	节点通信库接收速率，单位KByte/s。
txkB/s	bigint	节点通信库发送速率，单位KByte/s。
buffer	bigint	cmailbox的buffer大小。
memKB(libcomm)	bigint	libcomm进程通信内存大小，单位Byte。
memKB(libpq)	bigint	libpq进程通信内存大小，单位Byte。
%USED(PM)	integer	postmaster线程实时使用率。
%USED (sflow)	integer	gs_sender_flow_controller线程实时使用率。
%USED (rflow)	integer	gs_receiver_flow_controller线程实时使用率。
%USED (rloop)	integer	多个gs_receivers_loop线程中最高的实时使用率。
stream	integer	当前使用的逻辑连接总数。

16.3.69 PG_COMM_RECV_STREAM

PG_COMM_RECV_STREAM视图展示单个DN上所有的通信库接收流状态。

表 16-141 PG_COMM_RECV_STREAM 字段

名称	类型	描述
node_name	text	节点名称
local_tid	bigint	使用此通信流的线程ID
remote_name	text	连接对端节点名称
remote_tid	bigint	连接对端线程ID
idx	integer	通信对端DN在本DN内的标识编号
sid	integer	通信流在物理连接中的标识编号
tcp_sock	integer	通信流所使用的tcp通信socket
state	text	通信流当前的状态
query_id	bigint	通信流对应的debug_query_id编号
pn_id	integer	通信流所执行查询的plan_node_id编号
send_smp	integer	通信流所执行查询send端的smpid编号
recv_smp	integer	通信流所执行查询recv端的smpid编号

名称	类型	描述
recv_bytes	bigint	通信流接收的数据总量，单位Byte
time	bigint	通信流当前生命周期使用时长，单位ms
speed	bigint	通信流的平均接收速率，单位Byte/s
quota	bigint	通信流当前的通信配额值，单位Byte
buff_usize	bigint	通信流当前缓存的数据大小，单位Byte

16.3.70 PG_COMM_SEND_STREAM

PG_COMM_SEND_STREAM视图展示单个DN上所有的通信库发送流状态。

表 16-142 PG_COMM_SEND_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。
state	text	通信流当前的状态。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
send_bytes	bigint	通信流发送的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均发送速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
wait_quota	bigint	通信流等待quota值产生的额外时间开销，单位ms。

16.3.71 PG_CONTROL_GROUP_CONFIG

PG_CONTROL_GROUP_CONFIG视图存储系统的控制组配置信息。

表 16-143 PG_CONTROL_GROUP_CONFIG 字段

名称	类型	描述
pg_control_group_config	text	控制组的配置信息

16.3.72 PG_CURSORS

PG_CURSORS视图列出了当前可用的游标。

表 16-144 PG_CURSORS 字段

名称	类型	描述
name	text	游标名。
statement	text	声明改游标时的查询语句。
is_holdable	boolean	如果该游标是持久的（就是在声明该游标的事务结束后仍然可以访问该游标）则为TRUE，否则为FALSE。
is_binary	boolean	如果该游标被声明为BINARY则为TRUE，否则为FALSE。
is_scrollable	boolean	如果该游标可以滚动（就是允许以不连续的方式检索）则为TRUE，否则为FALSE。
creation_time	timestamp with time zone	声明该游标的时间戳。

16.3.73 PG_EXT_STATS

PG_EXT_STATS视图提供对存储在**PG_STATISTIC_EXT**表里面的扩展统计信息的访问。扩展统计信息目前包括多列统计信息。

表 16-145 PG_EXT_STATS 字段

名称	类型	引用	描述
schemaname	name	**PG_NAMESPACE**.nspname	包含表的模式名。
tablename	name	**PG_CLASS**.relname	表名。

名称	类型	引用	描述
attname	int2vector	PG_STATISTIC_EXT.stakey	统计信息扩展的多列信息。
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段组合为空的百分比。
avg_width	integer	-	字段组合记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段组合中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个字段组合中独立数值的个数和行数相同。 • 如果等于0，表示独立数值的数目未知。
n_dndistinct	real	-	标识dn1上字段组合中非NULL数据的唯一值的数目。 • 如果大于0，表示独立数值的实际数目。 • 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段组合的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 • 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值均不为NULL。
most_common_freqs	real[]	-	一个最常用数值组合的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。

名称	类型	引用	描述
most_common_vals_null	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。本列保存的多列常用数值中至少有一个值为NULL。
most_common_freqs_null	real[]	-	一个最常用数值组合的频率的列表，即每个出现的次数除以行数。如果most_common_vals_null是NULL，则为NULL。

16.3.74 PG_GET_INVALID_BACKENDS

PG_GET_INVALID_BACKENDS视图提供显示CN上连接到当前DN备机的后端线程信息。

表 16-146 PG_GET_INVALID_BACKENDS 字段

名称	类型	描述
pid	bigint	线程ID
node_name	text	后端线程中连接的节点信息
dbname	name	当前连接的数据库
backend_start	timestamp with time zone	后端线程启动的时间
query	text	后端线程正在执行的查询语句

16.3.75 PG_GET_SENDERS_CATCHUP_TIME

PG_GET_SENDERS_CATCHUP_TIME视图显示单个DN上当前活跃的主备发送线程的追赶信息。

表 16-147 PG_GET_SENDERS_CATCHUP_TIME 字段

名称	类型	描述
pid	bigint	当前sender的线程ID
lwpid	integer	当前sender的lwpid
local_role	text	本地的角色
peer_role	text	对端的角色
state	text	当前sender的复制状态

名称	类型	描述
type	text	当前sender的类型
catchup_start	timestamp with time zone	catchup启动的时间
catchup_end	timestamp with time zone	catchup结束的时间

16.3.76 PG_GROUP

PG_GROUP视图查看数据库认证角色及角色之间的成员关系。

表 16-148 PG_GROUP 字段

名称	类型	描述
groname	name	组的名称
grosysid	oid	组的ID
grolist	oid[]	一个数组，包含这个组里面所有角色的ID

16.3.77 PG_INDEXES

PG_INDEXES视图提供对数据库中每个索引的有用信息的访问。

表 16-149 PG_INDEXES 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表和索引的模式名
tablename	name	PG_CLASS.relname	此索引所服务的表名
indexname	name	PG_CLASS.relname	索引名
tablespace	name	PG_TABLESPACE.spcname	包含索引的表空间名称
indexdef	text	-	索引定义（一个重建的CREATE INDEX命令）

16.3.78 PG_LOCKS

PG_LOCKS视图存储各打开事务所持有的锁信息。

表 16-150 PG_LOCKS 字段

名称	类型	引用	描述
locktype	text	-	被锁定对象的类型：relation，extend，page，tuple，transactionid，virtualxid，object，userlock，advisory。
database	oid	PG_DATABASE.oid	被锁定对象所在数据库的OID。 • 如果被锁定的对象是共享对象，则OID为0。 • 如果是一个事务ID，则为NULL。
relation	oid	PG_CLASS.oid	被锁定对象关系的OID，如果锁定的对象不是关系，也不是关系的一部分，则为NULL。
page	integer	-	关系内部的页面编号，如果对象不是关系页或者不是行页，则为NULL。
tuple	smallint	-	页面里边的行编号，如果对象不是行，则为NULL。
virtualxid	text	-	事务的虚拟ID，如果对象不是一个虚拟事务ID，则为NULL。
transactionid	xid	-	事务的ID，如果对象不是一个事务ID，则为NULL。
classid	oid	PG_CLASS.oid	包含该对象的系统表的OID，如果对象不是普通的数据库对象，则为NULL。
objid	oid	-	对象在其系统表内的OID，如果对象不是普通数据库对象，则为NULL。
objsubid	smallint	-	对于表的某个字段对应为字段编号；对于其他对象类型，该字段为0；如果该对象不是普通数据库对象，则为NULL。
virtualtransaction	text	-	持有此锁或者在等待此锁的事务的虚拟ID。
pid	bigint	-	持有此锁或者等待此锁的服务器线程的逻辑ID。如果锁被一个预备事务持有，则为NULL。
mode	text	-	此线程持有的或者是期望持有的锁模式。

名称	类型	引用	描述
granted	boolean	-	• 如果锁是持有锁，则为TRUE。 • 如果锁是等待锁，则为FALSE。
fastpath	boolean	-	如果通过fast-path获得锁，则为TRUE；如果通过主锁表获得，则为FALSE。

16.3.79 PG_NODE_ENV

PG_NODE_ENV视图提供获取当前节点的环境变量信息。

表 16-151 PG_NODE_ENV 字段

名称	类型	描述
node_name	text	当前节点名称
host	text	当前节点的主机名称
process	integer	当前节点的进程号
port	integer	当前节点的端口号
installpath	text	当前节点的安装目录
datapath	text	当前节点的数据目录
log_directory	text	当前节点的日志目录

16.3.80 PG_OS_THREADS

PG_OS_THREADS视图提供当前节点下所有线程的状态信息。

表 16-152 PG_OS_THREADS 字段

名称	类型	描述
node_name	text	当前节点名称
pid	bigint	当前节点进程中正在运行的线程号
lwpid	integer	与pid对应的轻量级线程号
thread_name	text	与pid对应的线程名称
creation_time	timestamp with time zone	与pid对应的线程创建的时间

16.3.81 PG_POOLER_STATUS

PG_POOLER_STATUS视图查询pooler中的缓存连接状态。该视图只能在CN上执行查询，显示本地CN的pooler模块的连接缓存信息。

表 16-153 PG_POOLER_STATUS 字段

名称	类型	描述
database	text	数据库名称
user_name	text	用户名
tid	bigint	连接CN的线程ID
node_oid	bigint	连接的实例节点OID
node_name	name	连接的实例节点名称
in_use	boolean	连接是否正被使用 • t（true）：表示连接正在使用 • f（false）：表示连接没有使用
fdsock	bigint	对端socket
remote_pid	bigint	对端线程号
session_params	text	由此连接下发的GUC session参数

16.3.82 PG_PREPARED_STATEMENTS

PG_PREPARED_STATEMENTS视图显示当前会话所有可用的预备语句。

表 16-154 PG_PREPARED_STATEMENTS 字段

名称	类型	描述
name	text	预备语句的标识符。
statement	text	创建该预备语句的查询字符串。对于从SQL创建的预备语句而言是客户端提交的PREPARE语句；对于通过前/后端协议创建的预备语句而言是预备语句自身的文本。
prepare_time	timestamp with time zone	创建该预备语句的时间戳。
parameter_types	regtype[]	该预备语句期望的参数类型，以regtype类型的数组格式出现。与该数组元素相对应的OID可以通过把regtype转换为oid值得到。

名称	类型	描述
from_sql	boolean	• 如果该预备语句是通过PREPARE语句创建的则为true。 • 如果是通过前/后端协议创建的则为false。

16.3.83 PG_PREPARED_XACTS

PG_PREPARED_XACTS视图显示当前准备好进行两阶段提交的事务的信息。

表 16-155 PG_PREPARED_XACTS 字段

名称	类型	引用	描述
transaction	xid	-	预备事务的数字事务标识
gid	text	-	赋予该事务的全局事务标识
prepared	timestamp with time zone	-	事务准备好提交的时间
owner	name	PG_AUTHID.rolname	执行该事务的用户名
database	name	PG_DATABASE.datname	执行该事务所在的数据库名

16.3.84 PG_REPLICATION_SLOTS

PG_REPLICATION_SLOTS视图查看复制节点的信息。

表 16-156 PG_REPLICATION_SLOTS 字段

名称	类型	描述
slot_name	text	复制节点的名称
plugin	name	逻辑复制槽对应的输出插件名
slot_type	text	复制节点的类型
datoid	oid	复制节点的数据库OID
database	name	复制节点的数据库名称
active	boolean	复制节点是否为激活状态
xmin	xid	复制节点事务标识
catalog_xmin	text	逻辑复制槽对应的最早解码事务标识

名称	类型	描述
restart_lsn	text	复制节点的Xlog文件信息
dummy_standby	boolean	复制节点是否为假备

16.3.85 PG_ROLES

PG_ROLES视图提供访问数据库角色的相关信息。

表 16-157 PG_ROLES 字段

名称	类型	引用	描述
rolname	name	-	角色名。
rolsuper	boolean	-	该角色是否是拥有最高权限的初始系统管理员。
rolinherit	boolean	-	该角色是否继承角色的权限。
rolcreaterole	boolean	-	该角色是否可以创建其他的角色。
rolcreatedb	boolean	-	该角色是否可以创建数据库。
rolcatupdate	boolean	-	该角色是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
rolcanlogin	boolean	-	该角色是否可以登录数据库。
rolreplication	boolean	-	该角色是否可以复制。
rolauditadmin	boolean	-	该角色是否为审计管理员。
rolsystemadmin	boolean	-	该角色是否为系统管理员。
rolconnlimit	integer	-	对于可以登录的角色，rolconnlimit限制了该角色允许发起的最大并发连接数。 -1表示无限制。
rolpassword	text	-	不是口令，总是********。
rolvalidbegin	timestamp with time zone	-	帐户的有效开始时间；如果没有设置有效开始时间，则为NULL。
rolvaliduntil	timestamp with time zone	-	帐户的有效结束时间；如果没有设置有效结束时间，则为NULL。

名称	类型	引用	描述
rolrespool	name	-	用户所能够使用的resource pool。
rolparentid	oid	PG_AUTHID.rolparentid	用户所在组用户的OID。
roltabspace	text	-	用户永久表存储空间限额。
roltempspace	text	-	用户临时表存储空间限额。
rolspillspace	text	-	用户算子落盘空间限额。
rolconfig	text[]	-	运行时配置变量的会话缺省。
oid	oid	PG_AUTHID.oid	角色的ID。
roluseft	boolean	PG_AUTHID.roluseft	角色是否可以操作外表。
nodegroup	name	-	角色所关联的逻辑集群名字，如果没有关联逻辑集群，该值为空。

16.3.86 PG_RULES

PG_RULES视图提供对查询重写规则的有用信息访问的接口。

表 16-158 PG_RULES 字段

名称	类型	描述
schemaname	name	包含表的模式名
tablename	name	规则作用的表名
rulename	name	规则的名称
definition	text	规则定义（一个重新构造的创建命令）

16.3.87 PG_RUNNING_XACTS

PG_RUNNING_XACTS视图主要功能是显示当前节点运行事务的信息。

表 16-159 PG_RUNNING_XACTS 字段

名称	类型	描述
handle	integer	事务在GTM对应的句柄
gxid	xid	事务ID号

名称	类型	描述
state	tinyint	事务状态（3：prepared或者0：starting）
node	text	节点名称
xmin	xid	节点上当前数据涉及的最小事务号xmin
vacuum	boolean	标志当前事务是否是lazy vacuum事务
timeline	bigint	标志数据库重启次数
prepare_xid	xid	处于prepared状态的事务的ID号，若不在prepared状态，值为0
pid	bigint	事务对应的线程id
next_xid	xid	CN传给DN的事务id号

16.3.88 PG_SECLABELS

PG_SECLABELS视图提供关于安全标签的信息。

表 16-160 PG_SECLABELS 字段

名字	类型	引用	描述
objoid	oid	任意OID属性	安全标签所属的对象的OID。
classoid	oid	PG_CLASS.oid	此对象的系统表的OID。
objsubid	integer	-	对于某个在表字段上的安全标签，为字段编号（引用表本身的objoid和classoid）。对于所有其他对象类型，该字段为0。
objtype	text	-	标签出现的对象的类型。
objnamespace	oid	PG_NAMESPACE.oid	对象的命名空间的OID，如果适用；否则为NULL。
objname	text	-	标签适用的对象名。
provider	text	PG_SECLABEL.provider	与标签相关的标签提供者。
label	text	PG_SECLABEL.label	应用于此对象的安全标签。

16.3.89 PG_SESSION_WLMSTAT

PG_SESSION_WLMSTAT视图显示和当前用户执行作业正在运行时的负载管理相关信息。

表 16-161 PG_SESSION_WLMSTAT 字段

名称	类型	描述
datid	oid	连接后端的数据库OID。
datname	name	连接后端的数据库名称。
threadid	bigint	后端线程ID。
processid	integer	后端线程的pid。
usesysid	oid	登录后端的用户OID。
appname	text	连接到后端的应用名。
usename	name	登录到该后端的用户名。
priority	bigint	语句所在Cgroups的优先级。
attribute	text	语句的属性： ● Ordinary：语句发送到数据库后被解析前的默认属性。 ● Simple：简单语句。 ● Complicated：复杂语句。 ● Internal：数据库内部语句。
block_time	bigint	语句当前为止的pending的时间，单位s。
elapsed_time	bigint	语句当前为止的实际执行时间，单位s。
total_cpu_time	bigint	语句在上一时间周期内的DN上CPU使用的总时间，单位s。
cpu_skew_percent	integer	语句在上一时间周期内的DN上CPU使用的倾斜率。
statement_mem	integer	语句执行使用的statement_mem，预留字段。
active_points	integer	语句占用的资源池并发点数。
dop_value	integer	语句的从资源池中获取的dop值。
control_group	text	语句当前所使用的Cgroups。
status	text	语句当前的状态，包括： ● pending：执行前状态。 ● running：执行进行状态。 ● finished：执行正常结束。（当enqueue字段为StoredProc或Transaction时，仅代表语句中的部分作业已经执行完毕，该状态会持续到该语句完全执行完毕。） ● aborted：执行异常终止。 ● active: 非以上四种状态外的正常状态。 ● unknown: 未知状态。

名称	类型	描述
enqueue	text	语句当前的排队情况，包括： ● Global:：在全局队列中排队。 ● Respool：在资源池队列中排队。 ● CentralQueue：在中心协调节点(CCN)中排队。 ● Transaction：语句处于一个事务块中。 ● StoredProc : 语句处于一个存储过程中。 ● None：未在排队。 ● Forced None : 事务块语句或存储过程语句由于超出设定的等待时间而强制执行。
resource_pool	name	语句当前所在的资源池。
query	text	该后端的最新查询。如果state状态是active（活的），此字段显示当前正在执行的查询。所有其他情况表示上一个查询。
isplana	bool	逻辑集群模式下，语句当前是否占用其他逻辑集群的资源执行。该值默认为f（否）。
node_group	text	语句所属用户对应的逻辑集群。

16.3.90 PG_SESSION_IOSTAT

PG_SESSION_IOSTAT视图显示当前用户执行作业正在运行时的IO负载管理相关信息。

此视图中涉及到iops，对于行存，均以万次/s为单位；对于列存，均以次/s为单位。

表 16-162 PG_SESSION_IOSTAT 字段

名称	类型	描述
query_id	bigint	作业ID。
mincurriops	integer	该作业当前io在各DN中的最小值。
maxcurriops	integer	该作业当前io在各DN中的最大值。
minpeakiops	integer	在作业运行时，作业io峰值中，各DN的最小值。
maxpeakiops	integer	在作业运行时，作业io峰值中，各DN的最大值。
io_limits	integer	该作业所设GUC参数io_limits。
io_priority	text	该作业所设GUC参数io_priority。
query	text	作业。
node_group	text	作业所属用户对应的逻辑集群。

16.3.91 PG_SETTINGS

PG_SETTINGS视图显示数据库运行时参数的相关信息。

表 16-163 PG_SETTINGS 字段

名称	类型	描述
name	text	参数名称。
setting	text	参数当前值。
unit	text	参数的隐式结构。
category	text	参数的逻辑组。
short_desc	text	参数的简单描述。
extra_desc	text	参数的详细描述。
context	text	设置参数值的上下文，包括internal，backend，superuser，user。
vartype	text	参数类型，包括bool，enum，integer，real，string。
source	text	参数的赋值方式。
min_val	text	参数最小值。如果参数类型不是数值型，那么该字段值为null。
max_val	text	参数最大值。如果参数类型不是数值型，那么该字段值为null。
enumvals	text[]	enum类型参数合法值。如果参数类型不是enum型，那么该字段值为null。
boot_val	text	数据库启动时参数默认值。
reset_val	text	数据库重置时参数默认值。
sourcefile	text	设置参数值的配置文件。如果参数不是通过配置文件赋值，那么该字段值为null。
sourceline	integer	设置参数值的配置文件的行号。如果参数不是通过配置文件赋值，那么该字段值为null。

16.3.92 PG_SHADOW

PG_SHADOW视图显示了所有在PG_AUTHID中标记了rolcanlogin的角色的属性。

这个系统表的名字来自于该表是不可读的，因为它包含口令。**PG_USER**是一个在PG_SHADOW上公开可读的视图，只是把口令域填成了空白。

表 16-164 PG_SHADOW 字段

名字	类型	引用	描述
usename	name	PG_AUTHID.rolname	用户名。
usesysid	oid	PG_AUTHID.oid	用户的ID。
usecreatedb	boolean	-	用户可以创建数据库。
usesuper	boolean	-	用户是系统管理员。
usecatupd	boolean	-	用户可以更新系统表。即使是系统管理员，如果此字段不为真，也不能更新系统表。
userepl	boolean	-	用户可以初始化流复制和使系统处于或不处于备份模式。
passwd	text	-	口令（可能是加密的）；如果没有则为null。参阅PG_AUTHID获取加密的口令是如何存储的信息。
valbegin	timestamp with time zone	-	帐户的有效开始时间；如果没有设置有效开始时间，则为NULL。
valuntil	timestamp with time zone	-	帐户的有效结束时间；如果没有设置有效结束时间，则为NULL。
respool	name	-	用户使用的资源池。
parent	oid	-	父资源池。
spacelimit	text	-	永久表存储空间限额。
tempspacelimit	text	-	临时表存储空间限额。
spillspacelimit	text	-	算子落盘空间限额。
useconfig	text[ ]	-	运行时配置变量的会话缺省。

16.3.93 PG_SHARED_MEMORY_DETAIL

PG_SHARED_MEMORY_DETAIL视图查询所有已产生的共享内存上下文的使用信息。

表 16-165 PG_SHARED_MEMORY_DETAIL 字段

名字	类型	描述
contextname	text	内存上下文的名字
level	smallint	当前上下文在整体内存上下文中的层级
parent	text	上级内存上下文
totalsize	bigint	共享内存总大小，单位Byte
freesize	bigint	共享内存剩余大小，单位Byte
usedsize	bigint	共享内存使用大小，单位Byte

16.3.94 PG_STATS

PG_STATS视图提供对存储在pg_statistic表里面的单列统计信息的访问。

表 16-166 PG_STATS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
attname	name	PG_ATTRIBUTE.attname	字段名。
inherited	boolean	-	如果为真，则包含继承的子列，否则只是指定表的字段。
null_frac	real	-	记录中字段为空的百分比。
avg_width	integer	-	字段记录以字节记的平均宽度。
n_distinct	real	-	• 如果大于0，表示字段中独立数值的估计数目。 • 如果小于0，表示独立数值的数目被行数除的负数。 用负数形式是因为ANALYZE认为独立数值的数目是随着表增长而增长； 正数的形式用于在字段看上去好像有固定的可能值数目的情况下。比如，-1表示一个唯一字段，独立数值的个数和行数相同。

名称	类型	引用	描述
n_dndistinct	real	-	标识dn1上字段中非NULL数据的唯一值的数目。 ● 如果大于0，表示独立数值的实际数目。 ● 如果小于0，表示独立数值的数目被行数除的负数。（比如，一个字段的数值平均出现概率为两次，则可以表示为n_dndistinct=-0.5）。 ● 如果等于0，表示独立数值的数目未知。
most_common_vals	anyarray	-	一个字段组合里最常用数值的列表。如果该字段组合不存在最常用数值，则为NULL。
most_common_freqs	real[]	-	一个最常用数值的频率的列表，即每个出现的次数除以行数。如果most_common_vals是NULL，则为NULL。
histogram_bounds	anyarray	-	一个数值的列表，它把字段的数值分成几组大致相同的组。如果在most_common_vals里有数值，则在此饼图的计算中省略。如果字段数据类型没有<操作符或者most_common_vals列表代表了整个分布性，则此字段为NULL。
correlation	real	-	统计与字段值的物理行序和逻辑行序有关。它的范围从-1到+1。在数值接近-1或者+1的时候，在字段上的索引扫描将被认为比它接近零的时候开销更少，因为减少了对磁盘的随机访问。如果字段数据类型没有<操作符，则这个字段为NULL。
most_common_elems	anyarray	-	一个最常用的非空元素的列表。
most_common_elem_freqs	real[]	-	一个最常用元素的频率的列表。
elem_count_histogram	real[]	-	对于独立的非空元素的统计直方图。

16.3.95 PG_STAT_ACTIVITY

PG_STAT_ACTIVITY视图显示和当前用户查询相关的信息。

表 16-167 PG_STAT_ACTIVITY 字段

名称	类型	描述
datid	oid	用户会话在后端连接到的数据库OID。
datname	name	用户会话在后端连接到的数据库名称。
pid	bigint	后端线程ID。
usesysid	oid	登录该后端的用户OID。
usename	name	登录该后端的用户名。
application_name	text	连接到该后端的应用名。
client_addr	inet	连接到该后端的客户端的IP地址。 如果此字段是null，则表示通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，此字段是通过client_addr的反向DNS查找得到。此字段只有在启动log_hostname且使用IP连接时才非空。
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query_start列。
query_start	timestamp with time zone	开始当前活跃查询的时间， 如果state的值不是active，则这个值是上一个查询的开始时间。
state_change	timestamp with time zone	状态最后一次改变的时间。
waiting	boolean	如果后端当前正等待锁则为true。

名称	类型	描述
enqueue	text	语句当前排队状态。可能值是： ● waiting in queue：表示语句在排队中。 ● waiting in global queue：表示语句在全局排队中。 ● waiting in respool queue：表示语句在资源池排队中。 ● waiting in ccn queue：表示作业在CCN排队中。 ● 空：表示语句正在运行。
state	text	后端当前总体状态。可能值是： ● active：后台正在执行查询。 ● idle：后台正在等待新的客户端命令。 ● idle in transaction：后端在事务中，但事务中没有语句在执行。 ● idle in transaction (aborted)：后端在事务中，但事务中有语句执行失败。 ● fastpath function call：后端正在执行一个fast-path函数。 ● disabled：如果后端禁用track_activities，则报告此状态。 **说明** 普通用户只能查看到自己帐户所对应的会话状态。即其他帐户的state信息为空。例如以judy用户连接数据库后，在pg_stat_activity中查看到的普通用户joe及初始用户dbadmin的state信息为空： SELECT datname, usename, usesysid, state,pid FROM pg_stat_activity; datname | usename | usesysid | state | pid ----------+---------+----------+--------+----------------- postgres | dbadmin | 10 | | 139968752121616 postgres | dbadmin | 10 | | 139968903116560 db_tpcds | judy | 16398 | active | 139968391403280 postgres | dbadmin | 10 | | 139968643069712 postgres | dbadmin | 10 | | 139968680818448 postgres | joe | 16390 | | 139968563377936 (6 rows)
resource_pool	name	用户使用的资源池。

名称	类型	描述
query_id	bigint	查询语句的ID。
query	text	此后端的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。其他情况表示上一个查询。
connection_info	text	json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息（参见**connection_info**）。

16.3.96 PG_STAT_ALL_INDEXES

PG_STAT_ALL_INDEXES视图将包含当前数据库中的每个索引行，显示访问特定索引的统计。

索引可以通过简单的索引扫描或"位图"索引扫描进行使用。位图扫描中几个索引的输出可以通过AND或者OR规则进行组合， 因此当使用位图扫描的时候，很难将独立堆行抓取与特定索引进行组合， 因此，一个位图扫描增加
pg_stat_all_indexes.idx_tup_read使用索引计数，并且增加
pg_stat_all_tables.idx_tup_fetch表计数，但不影响
pg_stat_all_indexes.idx_tup_fetch。

表 16-168 PG_STAT_ALL_INDEXES 字段

名称	类型	描述
relid	oid	索引的表的OID
indexrelid	oid	索引的OID
schemaname	name	索引中模式名
relname	name	索引的表名
indexrelname	name	索引名
idx_scan	bigint	索引上开始的索引扫描数
idx_tup_read	bigint	通过索引上扫描返回的索引项数
idx_tup_fetch	bigint	通过使用索引的简单索引扫描抓取的活表行数

16.3.97 PG_STAT_ALL_TABLES

PG_STAT_ALL_TABLES视图包含当前数据库中每个表每行的信息（包括TOAST表），显示访问特定表的统计信息。

表 16-169 PG_STAT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID。
schemaname	name	此表的模式名。
relname	name	表名。
seq_scan	bigint	在此表上启动的顺序扫描数。
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目。
idx_scan	bigint	索引扫描的次数。
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目。
n_tup_ins	bigint	插入的行数。
n_tup_upd	bigint	更新的行数。
n_tup_del	bigint	删除的行数。
n_tup_hot_upd	bigint	HOT更新的行数（即不需要单独的索引更新）。
n_live_tup	bigint	live行估计数。
n_dead_tup	bigint	dead行估计数。
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）。
last_autovacuum	timestamp with time zone	最后一次autovacuum时间。
last_analyze	timestamp with time zone	最后一次analyze时间。
last_autoanalyze	timestamp with time zone	最后一次autovacuum时间。
vacuum_count	bigint	vacuum次数（不计算VACUUM FULL）。
autovacuum_count	bigint	autovacuum次数。
analyze_count	bigint	analyze次数。
autoanalyze_count	bigint	autoanalyze次数。

名称	类型	描述
last_data_changed	timestamp with time zone	记录该表最后一次数据发生变化的时间（引起数据变化的操作包括INSERT/UPDATE/DELETE、EXCHANGE/TRUNCATE/DROP partition），该列数据仅在本地CN记录。

16.3.98 PG_STAT_BAD_BLOCK

PG_STAT_BAD_BLOCK视图显示自节点启动后，读取数据时出现Page/CU校验失败的统计信息。

表 16-170 PG_STAT_BAD_BLOCK 字段

名字	类型	描述
nodename	text	节点名称
databaseid	integer	数据库OID
tablespaceid	integer	表空间OID
relfilenode	integer	文件对象ID
forknum	integer	文件类型
error_count	integer	出现校验失败的次数
first_time	timestamp with time zone	第一次出现时间
last_time	timestamp with time zone	最近一次出现时间

16.3.99 PG_STAT_BGWRITER

PG_STAT_BGWRITER视图显示关于后端写进程活动的统计信息。

表 16-171 PG_STAT_BGWRITER 字段

名称	类型	描述
checkpoints_timed	bigint	定期执行的检查点数量。
checkpoints_req	bigint	请求执行的检查点数量。

名称	类型	描述
checkpoint_write_time	double precision	检查点期间将文件写入磁盘花费的时间，以毫秒为单位。
checkpoint_sync_time	double precision	检查点期间数据同步到磁盘花费的时间，以毫秒为单位。
buffers_checkpoint	bigint	检查点期间写入缓冲区的数量。
buffers_clean	bigint	后端写进程写的缓冲区数量。
maxwritten_clean	bigint	由于写入缓冲区太多，后端写进程停止清理扫描的次数。
buffers_backend	bigint	后端直接写入的缓冲区数量。
buffers_backend_fsync	bigint	后端需要fsync的次数。
buffers_alloc	bigint	分配的缓冲区数量。
stats_reset	timestamp with time zone	统计重置的时间。

16.3.100 PG_STAT_DATABASE

PG_STAT_DATABASE视图将包含集群中每个数据库的每一行， 显示数据库统计。

表 16-172 PG_STAT_DATABASE 字段

名称	类型	描述
datid	oid	数据库OID。
datname	name	数据库名。
numbackends	integer	当前连接到该数据库的后端数。 这是在返回一个反映目前状态值的视图中唯一的列；自上次重置所有其他列返回累积值。
xact_commit	bigint	该数据库中已经提交的事务数。
xact_rollback	bigint	该数据库中已经回滚的事务数。
blks_read	bigint	该数据库中读取的磁盘块的数量。
blks_hit	bigint	高速缓存中已经发现的磁盘块的次数， 这样读取是不必要的（这只包括PostgreSQL缓冲区高速缓存，没有操作系统的文件系统缓存）。
tup_returned	bigint	通过数据库查询返回的行数。

名称	类型	描述
tup_fetched	bigint	通过数据库查询抓取的行数。
tup_inserted	bigint	通过数据库查询插入的行数。
tup_updated	bigint	通过数据库查询更新的行数。
tup_deleted	bigint	通过数据库查询删除的行数。
conflicts	bigint	由于数据库恢复冲突取消的查询数量。（只在备用服务器上发生）；请参见PG_STAT_DATABASE_CONFLICTS获取更多信息。
temp_files	bigint	通过数据库查询创建的临时文件数量。计算所有临时文件， 不论为什么创建临时文件（比如排序或者哈希）， 而且不考虑log_temp_files设置。
temp_bytes	bigint	通过数据库查询写入临时文件的大小。计算所有临时文件，不论为什么创建临时文件，而且不考虑log_temp_files设置。
deadlocks	bigint	在该数据库中检索的死锁数量。
blk_read_time	double precision	通过数据库后端读取数据文件块花费的时间，以毫秒计算。
blk_write_time	double precision	通过数据库后端写入数据文件块花费的时间，以毫秒计算。
stats_reset	timestamp with time zone	统计重置的时间。

16.3.101 PG_STAT_DATABASE_CONFLICTS

PG_STAT_DATABASE_CONFLICTS视图显示数据库冲突状态的统计信息。

表 16-173 PG_STAT_DATABASE_CONFLICTS 字段

名称	类型	描述
datid	oid	数据库OID
datname	name	数据库名
confl_tablespace	bigint	冲突的表空间的数目
confl_lock	bigint	冲突的锁数目
confl_snapshot	bigint	冲突的快照数目

名称	类型	描述
confl_bufferpin	bigint	冲突的缓冲区数目
confl_deadlock	bigint	冲突的死锁数目

16.3.102 PG_STAT_GET_MEM_MBYTES_RESERVED

PG_STAT_GET_MEM_MBYTES_RESERVED视图显示线程在内存中保存的当前活动信息。该函数在调用时需要指定线程ID，线程ID的选取请参考**PG_STAT_ACTIVITY**中的pid，线程ID为0时表示选取当前线程ID，例如：

```
SELECT pg_stat_get_mem_mbytes_reserved(0);
```

表 16-174 PG_STAT_GET_MEM_MBYTES_RESERVED 信息

名称	描述
ConnectInfo	连接信息
ParctlManager	并发管理信息
GeneralParams	基本参数信息
GeneralParams RPDATA	基本资源池信息
ExceptionManager	异常管理信息
CollectInfo	收集信息
GeneralInfo	基本信息
ParctlState	并发状态信息
CPU INFO	CPU信息
ControlGroup	控制组信息
IOSTATE	IO状态信息

16.3.103 PG_STAT_USER_FUNCTIONS

PG_STAT_USER_FUNCTIONS视图显示命名空间中用户自定义函数（函数语言为非内部语言）的状态信息。

表 16-175 PG_STAT_USER_FUNCTIONS 字段

名称	类型	描述
funcid	oid	函数OID

名称	类型	描述
schemaname	name	模式名
funcname	name	函数名
calls	bigint	函数被调用的次数
total_time	double precision	函数的总执行时长
self_time	double precision	当前线程调用函数的总的时长

16.3.104 PG_STAT_USER_INDEXES

PG_STAT_USER_INDEXES视图显示数据库中用户自定义普通表和toast表的索引状态信息。

表 16-176 PG_STAT_USER_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID
indexrelid	oid	索引的OID
schemaname	name	索引中模式名
relname	name	索引的表名
indexrelname	name	索引名
idx_scan	bigint	通过索引扫描的次数
idx_tup_read	bigint	通过索引上扫描返回的索引条目数量
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目

16.3.105 PG_STAT_USER_TABLES

PG_STAT_USER_TABLES视图显示所有命名空间中用户自定义普通表和toast表的状态信息。

表 16-177 PG_STAT_USER_TABLES 字段

名称	类型	描述
relid	oid	表的OID
schemaname	name	表的模式名
relname	name	表名
seq_scan	bigint	在此表上表启动的顺序扫描的次数

名称	类型	描述
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目
idx_scan	bigint	索引扫描的次数
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目
n_tup_ins	bigint	插入的行数
n_tup_upd	bigint	更新的行数
n_tup_del	bigint	删除的行数
n_tup_hot_upd	bigint	HOT更新的行数（即不需要单独的索引更新）
n_live_tup	bigint	live行估计数
n_dead_tup	bigint	dead行估计数
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）
last_autovacuum	timestamp with time zone	最后一次autovacuum时间
last_analyze	timestamp with time zone	最后一次analyze时间
last_autoanalyze	timestamp with time zone	最后一次autoanalyze时间
vacuum_count	bigint	vacuum的次数（不计算VACUUM FULL）
autovacuum_count	bigint	autovacuum的次数
analyze_count	bigint	analyze的次数
autoanalyze_count	bigint	autoanalyze的次数

16.3.106 PG_STAT_REPLICATION

PG_STAT_REPLICATION视图用于描述日志同步状态信息，如发起端发送日志位置，收端接收日志位置等。

表 16-178 PG_STAT_REPLICATION 字段

名称	类型	描述
pid	bigint	线程的PID
usesysid	oid	用户系统ID
usename	name	用户名
application_name	text	程序名称
client_addr	inet	客户端地址
client_hostname	text	客户端名
client_port	integer	客户端端口号
backend_start	timestamp with time zone	程序启动时间
state	text	日志复制的状态（追赶状态，还是一致的流状态）
sender_sent_location	text	发送端发送日志位置
receiver_write_location	text	接收端write日志位置
receiver_flush_location	text	接收端flush日志位置
receiver_replay_location	text	接收端replay日志位置
sync_priority	integer	同步复制的优先级（0表示异步）
sync_state	text	同步状态（异步复制，同步复制，还是潜在同步）

16.3.107 PG_STAT_SYS_INDEXES

PG_STAT_SYS_INDEXES视图显示pg_catalog、information_schema模式中所有系统表的索引状态信息。

表 16-179 PG_STAT_SYS_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID
indexrelid	oid	索引的OID

名称	类型	描述
schemaname	name	索引中模式名
relname	name	索引的表名
indexrelname	name	索引名
idx_scan	bigint	通过索引扫描的次数
idx_tup_read	bigint	通过索引上扫描返回的索引条目数
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目

16.3.108 PG_STAT_SYS_TABLES

PG_STAT_SYS_TABLES视图显示pg_catalog、information_schema模式的所有命名空间中系统表的统计信息。

表 16-180 PG_STAT_SYS_TABLES 字段

名称	类型	描述
relid	oid	表的OID
schemaname	name	表的模式名
relname	name	表名
seq_scan	bigint	在此表上表启动的顺序扫描的次数
seq_tup_read	bigint	顺序扫描抓取的有live数据行的数目
idx_scan	bigint	索引扫描的次数
idx_tup_fetch	bigint	索引扫描抓取的有live数据行的数目
n_tup_ins	bigint	插入的行数
n_tup_upd	bigint	更新的行数
n_tup_del	bigint	删除的行数
n_tup_hot_upd	bigint	HOT更新的行数（比如没有更新所需的单独索引）
n_live_tup	bigint	live行估计数
n_dead_tup	bigint	dead行估计数
last_vacuum	timestamp with time zone	最后一次手动vacuum时间（不计算VACUUM FULL）

名称	类型	描述
last_autovacuum	timestamp with time zone	最后一次autovacuum时间
last_analyze	timestamp with time zone	最后一次analyze时间
last_autoanalyze	timestamp with time zone	最后一次autoanalyze时间
vacuum_count	bigint	vacuum的次数（不计算VACUUM FULL）
autovacuum_count	bigint	autovacuum的次数
analyze_count	bigint	analyze的次数
autoanalyze_count	bigint	autoanalyze的次数

16.3.109 PG_STAT_XACT_ALL_TABLES

PG_STAT_XACT_ALL_TABLES视图显示命名空间中所有普通表和toast表的事务状态信息。

表 16-181 PG_STAT_XACT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID
schemaname	name	此表的模式名
relname	name	表名
seq_scan	bigint	在此表上启动的顺序扫描数
seq_tup_read	bigint	顺序扫描抓取的活跃行数
idx_scan	bigint	在此表上启动的索引扫描数
idx_tup_fetch	bigint	索引扫描抓取的活跃行数
n_tup_ins	bigint	插入的行数
n_tup_upd	bigint	更新的行数
n_tup_del	bigint	删除的行数

名称	类型	描述
n_tup_hot_upd	bigint	HOT更新行数（比如没有更新所需的单独索引）

16.3.110 PG_STAT_XACT_SYS_TABLES

PG_STAT_XACT_SYS_TABLES视图显示命名空间中系统表的事务状态信息。

表 16-182 PG_STAT_XACT_SYS_TABLES 字段

名称	类型	描述
relid	oid	表的OID
schemaname	name	此表的模式名
relname	name	表名
seq_scan	bigint	在此表上启动的顺序扫描数
seq_tup_read	bigint	顺序扫描抓取的活跃行数
idx_scan	bigint	在此表启动的的索引扫描数
idx_tup_fetch	bigint	索引扫描抓取的活跃行数
n_tup_ins	bigint	插入行数
n_tup_upd	bigint	更新行数
n_tup_del	bigint	删除行数
n_tup_hot_upd	bigint	HOT更新行数（比如没有更新所需的单独索引）

16.3.111 PG_STAT_XACT_USER_FUNCTIONS

PG_STAT_XACT_USER_FUNCTIONS视图包含每个跟踪函数的行，显示关于函数执行的统计。

表 16-183 PG_STAT_XACT_USER_FUNCTIONS 字段

名称	类型	描述
funcid	oid	函数OID
schemaname	name	模式名
funcname	name	函数名
calls	bigint	函数被调用的次数

名称	类型	描述
total_time	double precision	函数的总执行时长
self_time	double precision	当前线程调用函数的总的时长

16.3.112 PG_STAT_XACT_USER_TABLES

PG_STAT_XACT_USER_TABLES视图显示命名空间中用户表的事务状态信息。

表 16-184 PG_STAT_XACT_USER_TABLES 字段

名称	类型	描述
relid	oid	表的OID
schemaname	name	此表的模式名
relname	name	表名
seq_scan	bigint	在该表上启动的顺序扫描数
seq_tup_read	bigint	顺序扫描抓取的活跃行数
idx_scan	bigint	在该表上启动的索引扫描数
idx_tup_fetch	bigint	索引扫描抓取的活跃行数
n_tup_ins	bigint	插入行数
n_tup_upd	bigint	更新行数
n_tup_del	bigint	删除行数
n_tup_hot_upd	bigint	HOT更新行数（比如没有更新所需的单独索引）

16.3.113 PG_STATIO_ALL_INDEXES

PG_STATIO_ALL_INDEXES视图将包含当前数据库中的每个索引行， 显示特定索引的I/O的统计。

表 16-185 PG_STATIO_ALL_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID
indexrelid	oid	索引的OID

名称	类型	描述
schemaname	name	索引中模式名
relname	name	索引的表名
indexrelname	name	索引名
idx_blks_read	bigint	从索引中读取的磁盘块数
idx_blks_hit	bigint	索引缓冲区命中数量

16.3.114 PG_STATIO_ALL_SEQUENCES

PG_STATIO_ALL_SEQUENCES视图包含当前数据库中每个序列的每一行， 显示特定序列关于I/O的统计。

表 16-186 PG_STATIO_ALL_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID
schemaname	name	序列中模式名
relname	name	序列名
blks_read	bigint	从序列中读取的磁盘块数
blks_hit	bigint	序列缓冲区命中数量

16.3.115 PG_STATIO_ALL_TABLES

PG_STATIO_ALL_TABLES视图将包含当前数据库中每个表的一行（包括TOAST表），显示出特定表I/O的统计。

表 16-187 PG_STATIO_ALL_TABLES 字段

名称	类型	描述
relid	oid	表OID
schemaname	name	此表模式名
relname	name	表名
heap_blks_read	bigint	从该表中读取的磁盘块数
heap_blks_hit	bigint	此表缓冲区命中数
idx_blks_read	bigint	从表中所有索引读取的磁盘块数

名称	类型	描述
idx_blks_hit	bigint	表中所有索引命中缓冲区数
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）

16.3.116 PG_STATIO_SYS_INDEXES

PG_STATIO_SYS_INDEXES视图显示命名空间中所有系统表索引的IO状态信息。

表 16-188 PG_STATIO_SYS_INDEXES 字段

名称	类型	描述
relid	oid	此索引表的OID
indexrelid	oid	索引的OID
schemaname	name	索引的模式名
relname	name	索引的表名
indexrelname	name	索引名
idx_blks_read	bigint	从索引中读取的磁盘块数
idx_blks_hit	bigint	索引缓冲区命中数量

16.3.117 PG_STATIO_SYS_SEQUENCES

PG_STATIO_SYS_SEQUENCES视图显示命名空间中所有系统表为序列的IO状态信息。

表 16-189 PG_STATIO_SYS_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID
schemaname	name	序列中模式名
relname	name	序列名
blks_read	bigint	从序列中读取的磁盘块数
blks_hit	bigint	序列缓冲区命中数量

16.3.118 PG_STATIO_SYS_TABLES

PG_STATIO_SYS_TABLES视图显示命名空间中所有系统表的IO状态信息。

表 16-190 PG_STATIO_SYS_TABLES 字段

名称	类型	描述
relid	oid	表OID
schemaname	name	表模式名
relname	name	表名
heap_blks_read	bigint	从表中读取的磁盘块数
heap_blks_hit	bigint	此表缓冲区命中数
idx_blks_read	bigint	从表中所有索引读取的磁盘块数
idx_blks_hit	bigint	表中所有索引命中缓冲区数
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）

16.3.119 PG_STATIO_USER_INDEXES

PG_STATIO_USER_INDEXES视图显示命名空间中所有用户关系表索引的IO状态信息。

表 16-191 PG_STATIO_USER_INDEXES 字段

名称	类型	描述
relid	oid	索引的表的OID
indexrelid	oid	该索引的OID
schemaname	name	该索引的模式名
relname	name	该索引的表名
indexrelname	name	索引名称
idx_blks_read	bigint	从索引中读取的磁盘块数

名称	类型	描述
idx_blks_hit	bigint	索引命中缓冲区数

16.3.120 PG_STATIO_USER_SEQUENCES

PG_STATIO_USER_SEQUENCES视图显示命名空间中所有用户关系表类型为序列的IO状态信息。

表 16-192 PG_STATIO_USER_SEQUENCES 字段

名称	类型	描述
relid	oid	序列OID
schemaname	name	序列中模式名
relname	name	序列名
blks_read	bigint	从序列中读取的磁盘块数
blks_hit	bigint	序列中缓存命中数

16.3.121 PG_STATIO_USER_TABLES

PG_STATIO_USER_TABLES视图显示命名空间中所有用户关系表的IO状态信息。

表 16-193 PG_STATIO_USER_TABLES 字段

名称	类型	描述
relid	oid	表OID
schemaname	name	该表模式名
relname	name	表名
heap_blks_read	bigint	从该表中读取的磁盘块数
heap_blks_hit	bigint	此表缓冲区命中数
idx_blks_read	bigint	从表中所有索引读取的磁盘块数
idx_blks_hit	bigint	表中所有索引缓冲区命中数
toast_blks_read	bigint	此表的TOAST表读取的磁盘块数（如果存在）
toast_blks_hit	bigint	此表的TOAST表命中缓冲区数（如果存在）
tidx_blks_read	bigint	此表的TOAST表索引读取的磁盘块数（如果存在）

名称	类型	描述
tidx_blks_hit	bigint	此表的TOAST表索引命中缓冲区数（如果存在）

16.3.122 PG_THREAD_WAIT_STATUS

通过PG_THREAD_WAIT_STATUS视图可以检测当前实例中工作线程（backend thread）以及辅助线程（auxiliary thread）的阻塞等待情况。

表 16-194 PG_THREAD_WAIT_STATUS 字段

名称	类型	描述
node_name	text	当前节点的名称。
db_name	text	数据库名称。
thread_name	text	线程名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
ptid	integer	streaming线程的父线程。
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。等待状态的详细信息请参见**表 16-195**。
wait_event	text	如果wait_status是acquire lock、acquire lwlock、wait io三种类型，此列描述具体的锁、轻量级锁、IO的信息。否则为空。

wait_status列的等待状态有以下状态。

表 16-195 等待状态列表

wait_status值	含义
none	未等任意事件。
acquire lock	等待加锁，要么加锁成功，要么加锁等待超时。
acquire lwlock	等待获取轻量级锁。
wait io	等待IO完成。

wait_status值	含义
wait cmd	等待完成读取网络通信包。
wait pooler get conn	等待pooler完成获取连接。
wait pooler abort conn	等待pooler完成终止连接。
wait pooler clean conn	等待pooler完成清理连接。
pooler create conn: [nodename], total N	等待pooler建立连接，当前正在与nodename指定节点建立连接，且仍有N个连接等待建立。
get conn	获取到其他节点的连接。
set cmd: [nodename]	在连接上执行SET/RESET/TRANSACTION BLOCK LEVEL PARA SET/SESSION LEVEL PARA SET，当前正在nodename指定节点上执行。
cancel query	取消某连接上正在执行的SQL语句。
stop query	停止某连接上正在执行的查询。
wait node: [nodename](plevel), total N, [phase]	等待接收与某节点的连接上的数据，当前正在等待nodename节点plevel线程的数据，且仍有N个连接的数据待返回。如果状态包含phase信息，则可能的阶段状态有： ● begin：表示处于事务开始阶段。 ● commit：表示处于事务提交阶段。 ● rollback：表示处于事务回滚阶段。
wait transaction sync: xid	等待xid指定事务同步。
wait wal sync	等待特定LSN的wal log完成到备机的同步。
wait data sync	等待完成数据页到备机的同步。
wait data sync queue	等待把行存的数据页或列存的CU放入同步队列。
flush data: [nodename](plevel), [phase]	等待向网络中nodename指定节点的plevel对应线程发送数据。如果状态包含phase信息，则可能的阶段状态为wait quota，即当前通信流正在等待quota值。
stream get conn: [nodename], total N	初始化stream flow时，等待与nodename节点的consumer对象建立连接，且当前有N个待建连对象。
wait producer ready: [nodename](plevel), total N	初始化stream flow时，等待每个producer都准备好，当前正在等待nodename节点plevel对应线程的producer对象准备好，且仍有N个producer对象处于等待状态。

wait_status值	含义
synchronize quit	steam plan结束时，等待stream线程组内的线程统一退出。
nodegroup destroy	steam plan结束时，等待销毁stream node group。
wait active statement	等待作业执行，正在资源负载管控中。
wait global queue	等待作业执行，正在全局队列排队。
wait respool queue	等待作业执行，正在资源池上排队。
wait ccn queue	等待作业执行，正在中心协调节点(CCN)中排队。
gtm connect	等待与GTM建连。
gtm get gxid	等待从GTM获取事务xid。
gtm get snapshot	等待从GTM获取事务快照snapshot。
gtm begin trans	等待GTM开始事务。
gtm commit trans	等待GTM提交事务。
gtm rollback trans	等待GTM执行事务回滚。
gtm start prepare trans	等待GTM开始两阶段事务的prepare阶段。
gtm prepare trans	等待GTM完成两阶段事务的prepare阶段。
gtm open sequence	等待GTM打开sequence。
gtm close sequence	等待GTM关闭sequence。
gtm create sequence	等待GTM创建sequence。
gtm alter sequence	等待GTM修改sequence。
gtm get sequence val	等待从GTM获取sequence的下一个值。
gtm set sequence val	等待GTM设置sequence的值。
gtm drop sequence	等待GTM删除sequence。
gtm rename sequece	等待GTM重命名sequence。
analyze: [relname], [phase]	当前正在对表relname执行analyze。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的analyze分析操作。
vacuum: [relname], [phase]	当前正在对表relname执行vacuum。如果状态包含phase信息，则为autovacuum，表示是数据库自动开启AutoVacuum线程执行的vacuum清理操作。
vacuum full: [relname]	当前正在对表relname执行vacuum full清理。

wait_status值	含义
create index	当前正在创建索引。
HashJoin - [build hash | write file]	当前是HashJoin算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashJoin算子正在建立哈希表。 ● write file：表示当前HashJoin算子正在将数据写入磁盘。
HashAgg - [build hash | write file]	当前是HashAgg算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashAgg算子正在建立哈希表。 ● write file：表示当前HashAgg算子正在将数据写入磁盘。
HashSetop - [build hash | write file]	当前是HashSetop算子，主要关注耗时的执行阶段。 ● build hash：表示当前HashSetop算子正在建立哈希表。 ● write file：表示当前HashSetop算子正在将数据写入磁盘。
Sort | Sort - write file	当前是Sort算子做排序，write file表示Sort算子正在将数据写入磁盘。
Material | Material - write file	当前是Material算子，write file表示Material算子正在将数据写入磁盘。
wait sync consumer next step	Consumer接收端同步等待下一轮迭代。
wait sync producer next step	Producer发送端同步等待下一轮迭代。

当wait_status为acquire lwlock、acquire lock或者wait io时，表示有等待事件。正在等待获取wait_event列对应类型的轻量级锁、事务锁，或者正在进行IO。

其中，wait_status值为acquire lwlock（轻量级锁）时对应的wait_event等待事件类型与描述信息如下。（wait_event为extension时，表示此时的轻量级锁是动态分配的锁，未被监控。）

表 16-196 轻量级锁等待事件列表

wait_event类型	类型描述
ShmemIndexLock	用于保护共享内存中的主索引哈希表。
OidGenLock	用于避免不同线程产生相同的OID。
XidGenLock	用于避免两个事务获得相同的xid。

wait_event类型	类型描述
ProcArrayLock	用于避免并发访问或修改ProcArray共享数组。
SInvalReadLock	用于避免与清理失效消息并发执行。
SInvalWriteLock	用于避免与其它写失效消息、清理失效消息并发执行。
WALInsertLock	用于避免与其它WAL插入操作并发执行。
WALWriteLock	用于避免并发WAL写盘。
ControlFileLock	用于避免pg_control文件的读写并发、写写并发。
CheckpointLock	用于避免多个checkpoint并发执行。
CLogControlLock	用于避免并发访问或者修改Clog控制数据结构。
SubtransControlLock	用于避免并发访问或者修改子事务控制数据结构。
MultiXactGenLock	用于串行分配唯一MultiXact id。
MultiXactOffsetControlLock	用于避免对pg_multixact/offset的写写并发和读写并发。
MultiXactMemberControlLock	用于避免对pg_multixact/members的写写并发和读写并发。
RelCacheInitLock	用于失效消息场景对init文件进行操作时加锁。
CheckpointerCommLock	用于向checkpointer发起文件刷盘请求场景，需要串行的向请求队列插入请求结构。
TwoPhaseStateLock	用于避免并发访问或者修改两阶段信息共享数组。
TablespaceCreateLock	用于确定tablespace是否已经存在。
BtreeVacuumLock	用于防止vacuum清理B-tree中还在使用的页面。
AutovacuumLock	用于串行化访问autovacuum worker数组。
AutovacuumScheduleLock	用于串行化分配需要vacuum的table。
AutoanalyzeLock	用于获取和释放允许执行Autoanalyze的任务资源。
SyncScanLock	用于确定heap扫描时某个relfilenode的起始位置。
NodeTableLock	用于保护存放CN和DN节点信息的共享结构。
PoolerLock	用于保证两个线程不会同时从连接池里取到相同的连接。
RelationMappingLock	用于等待更新系统表到存储位置之间映射的文件。
AsyncCtlLock	用于避免并发访问或者修改共享通知状态。
AsyncQueueLock	用于避免并发访问或者修改共享通知信息队列。

wait_event类型	类型描述
SerializableXactHashLock	用于避免对于可串行事务共享结构的写写并发和读写并发。
SerializableFinishedListLock	用于避免对于已完成可串行事务共享链表的写写并发和读写并发。
SerializablePredicateLockListLock	用于保护对于可串行事务持有的锁链表。
OldSerXidLock	用于保护记录冲突可串行事务的结构。
FileStatLock	用于保护存储统计文件信息的数据结构。
SyncRepLock	用于在主备复制时保护xlog同步信息。
DataSyncRepLock	用于在主备复制时保护数据页同步信息。
CStoreColspaceCacheLock	用于保护列存表的CU空间分配。
CStoreCUCacheSweepLock	用于列存CU Cache循环淘汰。
MetaCacheSweepLock	用于元数据循环淘汰。
DfsConnectorCacheLock	用于保护缓存HDFS连接的句柄的全局哈希表。
dummyServerInfoCacheLock	用于保护缓存加速集群连接信息的全局哈希表。
ExtensionConnectorLibLock	用于初始化ODBC连接场景，在加载与卸载特定动态库时进行加锁。
SearchServerLibLock	用于GPU加速场景初始化加载特定动态库时，对读文件操作进行加锁。
DfsUserLoginLock	用于保护HDFS用户信息的全局链表。
DfsSpaceCacheLock	用于控制HDFS表导入时文件ID单调递增。
LsnXlogChkFileLock	用于串行更新特定结构中记录的主备机的xlog flush位置点。
GTMHostInfoLock	用于避免并发访问或者修改GTM主机信息。
ReplicationSlotAllocationLock	用于主备复制时保护主机端的流复制槽的分配。
ReplicationSlotControlLock	用于主备复制时避免并发更新流复制槽状态。
ResourcePoolHashLock	用于避免并发访问或者修改资源池哈希表。
WorkloadStatHashLock	用于避免并发访问或者修改包含CN侧的SQL请求构成的哈希表。
WorkloadIoStatHashLock	用于避免并发访问或者修改用于统计当前DN的IO信息的哈希表。

wait_event类型	类型描述
WorkloadCGroupHashLock	用于避免并发访问或者修改Cgroup信息构成的哈希表。
OBSGetPathLock	用于避免对obs路径的写写并发和读写并发。
WorkloadUserInfoLock	用于避免并发访问或修改负载管理的用户信息哈希表。
WorkloadRecordLock	用于避免并发访问或修改在内存自适应管理时对CN收到请求构成的哈希表。
WorkloadIOUtilLock	用于保护记录iostat，CPU等负载信息的结构。
WorkloadNodeGroupLock	用于避免并发访问或者修改内存中的nodegroup信息构成的哈希表。
JobShmemLock	用于MPP兼容ORACLE定时任务功能中保护定时读取的全局变量。
OBSRuntimeLock	用于获取环境变量，如GASSHOME。
LLVMDumpIRLock	用于导出动态生成函数所对应的汇编语言。
LLVMParseIRLock	用于在查询开始处从IR文件中编译并解析已写好的IR函数。
RPNumberLock	用于加速集群的DN对正在执行计划的任务线程的计数。
ClusterRPLock	用于加速集群的CCN中维护的集群负载数据的并发存取控制。
CriticalCacheBuildLock	用于从共享或者本地缓存初始化文件中加载cache的场景。
WaitCountHashLock	用于保护用户语句计数功能场景中的共享结构。
BufMappingLock	用于保护对共享缓冲映射表的操作。
LockMgrLock	用于保护常规锁结构信息。
PredicateLockMgrLock	用于保护可串行事务锁结构信息。
OperatorRealTLock	用于避免并发访问或者修改记录算子级实时数据的全局结构。
OperatorHistLock	用于避免并发访问或者修改记录算子级历史数据的全局结构。
SessionRealTLock	用于避免并发访问或者修改记录query级实时数据的全局结构。
SessionHistLock	用于避免并发访问或者修改记录query级历史数据的全局结构。
CacheSlotMappingLock	用于保护CU Cache全局信息。

wait_event类型	类型描述
BarrierLock	用于保证当前只有一个线程在创建Barrier。

当wait_status值为wait io时对应的wait_event等待事件类型与描述信息如下。

表 16-197 IO 等待事件列表

wait_event类型	类型描述
BufFileRead	从临时文件中读取数据到指定buffer。
BufFileWrite	向临时文件中写入指定buffer中的内容。
ControlFileRead	读取pg_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。
ControlFileSync	将pg_control文件持久化到磁盘。数据库初始化时发生。
ControlFileSyncUpdate	将pg_control文件持久化到磁盘。主要在数据库启动、执行checkpoint和主备校验过程中发生。
ControlFileWrite	写入pg_control文件。数据库初始化时发生。
ControlFileWriteUpdate	更新pg_control文件。主要在数据库启动、执行checkpoint和主备校验过程中发生。
CopyFileRead	copy文件时读取文件内容。
CopyFileWrite	copy文件时写入文件内容。
DataFileExtend	扩展文件时向文件写入内容。
DataFileFlush	将表数据文件持久化到磁盘
DataFileImmediateSync	将表数据文件立即持久化到磁盘。
DataFilePrefetch	异步读取表数据文件。
DataFileRead	同步读取表数据文件。
DataFileSync	将表数据文件的修改持久化到磁盘。
DataFileTruncate	表数据文件truncate。
DataFileWrite	向表数据文件写入内容。
LockFileAddToDataDirRead	读取"postmaster.pid"文件。
LockFileAddToDataDirSync	将"postmaster.pid"内容持久化到磁盘。
LockFileAddToDataDirWrite	将pid信息写到"postmaster.pid"文件。

wait_event类型	类型描述
LockFileCreateRead	读取LockFile文件"%s.lock"。
LockFileCreateSync	将LockFile文件"%s.lock"内容持久化到磁盘。
LockFileCreateWRITE	将pid信息写到LockFile文件"%s.lock"。
RelationMapRead	读取系统表到存储位置之间的映射文件
RelationMapSync	将系统表到存储位置之间的映射文件持久化到磁盘。
RelationMapWrite	写入系统表到存储位置之间的映射文件。
ReplicationSlotRead	读取流复制槽文件。重新启动时发生。
ReplicationSlotRestoreSync	将流复制槽文件持久化到磁盘。重新启动时发生。
ReplicationSlotSync	checkpoint时将流复制槽临时文件持久化到磁盘。
ReplicationSlotWrite	checkpoint时写流复制槽临时文件。
SLRUFlushSync	将pg_clog、pg_subtrans和pg_multixact文件持久化到磁盘。主要在执行checkpoint和数据库停机时发生。
SLRURead	读取pg_clog、pg_subtrans和pg_multixact文件。
SLRUSync	将脏页写入文件pg_clog、pg_subtrans和pg_multixact并持久化到磁盘。主要在执行checkpoint和数据库停机时发生。
SLRUWrite	写入pg_clog、pg_subtrans和pg_multixact文件。
TimelineHistoryRead	读取timeline history文件。在数据库启动时发生。
TimelineHistorySync	将timeline history文件持久化到磁盘。在数据库启动时发生。
TimelineHistoryWrite	写入timeline history文件。在数据库启动时发生。
TwophaseFileRead	读取pg_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。
TwophaseFileSync	将pg_twophase文件持久化到磁盘。在两阶段事务提交、两阶段事务恢复时发生。
TwophaseFileWrite	写入pg_twophase文件。在两阶段事务提交、两阶段事务恢复时发生。
WALBootstrapSync	将初始化的WAL文件持久化到磁盘。在数据库初始化发生。
WALBootstrapWrite	写入初始化的WAL文件。在数据库初始化发生。
WALCopyRead	读取已存在的WAL文件并进行复制时产生的读操作。在执行归档恢复完后发生。

wait_event类型	类型描述
WALCopySync	将复制的WAL文件持久化到磁盘。在执行归档恢复完后发生。
WALCopyWrite	读取已存在WAL文件并进行复制时产生的写操作。在执行归档恢复完后发生。
WALInitSync	将新初始化的WAL文件持久化磁盘。在日志回收或写日志时发生。
WALInitWrite	将新创建的WAL文件初始化为0。在日志回收或写日志时发生。
WALRead	从xlog日志读取数据。两阶段文件redo相关的操作产生。
WALSyncMethodAssign	将当前打开的所有WAL文件持久化到磁盘。
WALWrite	写入WAL文件。

当wait_status值为acquire lock（事务锁）时对应的wait_event等待事件类型与描述信息如下。

表 16-198 事务锁等待事件列表

wait_event类型	类型描述
relation	对表加锁
extend	对表扩展空间时加锁
partition	对分区表加锁
partition_seq	对分区表的分区加锁
page	对表页面加锁
tuple	对页面上的tuple加锁
transactionid	对事务ID加锁
virtualxid	对虚拟事务ID加锁
object	加对象锁
cstore_freespace	对列存空闲空间加锁
userlock	加用户锁
advisory	加advisory锁

16.3.123 PG_TABLES

PG_TABLES视图提供了对数据库中每个表访问的有用信息。

表 16-199 PG_TABLES 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含表的模式名。
tablename	name	PG_CLASS.relname	表名。
tableowner	name	pg_get_userbyid(PG_CLASS.relowner)	表的所有者。
tablespace	name	PG_TABLESPACE.spcname	包含表的表空间，默认为NULL。
hasindexes	boolean	PG_CLASS.relhasindex	如果表上有索引（或者最近拥有）则为TRUE，否则为FALSE。
hasrules	boolean	PG_CLASS.relhasruls	如果表上有规则，则为TRUE，否则为FALSE。
hastriggers	boolean	PG_CLASS.RELHASTRIGGERS	如果表上有触发器，则为TRUE，否则为FALSE。
tablecreator	name	pg_get_userbyid(PG_OBJECT.creator)	表创建用户，如创建用户已删除，则返回空。
created	timestamp with time zone	PG_OBJECT.ctime	表创建时间。
last_ddl_time	timestamp with time zone	PG_OBJECT.mtime	表最后修改时间。

16.3.124 PG_TIMEZONE_ABBREVS

PG_TIMEZONE_ABBREVS视图提供了输入例程能够识别的所有时区缩写。

表 16-200 PG_TIMEZONE_ABBREVS 字段

名称	类型	描述
abbrev	text	时区缩写
utc_offset	interval	相对于UTC的偏移量
is_dst	boolean	如果这是一个夏时制时区缩写则为TRUE，否则为FALSE

16.3.125 PG_TIMEZONE_NAMES

PG_TIMEZONE_NAMES视图提供了显示了所有能够被SET TIMEZONE识别的时区名及其缩写、UTC偏移量、是否夏时制。

表 16-201 PG_TIMEZONE_NAMES 字段

名称	类型	描述
name	text	时区名
abbrev	text	时区名缩写
utc_offset	interval	相对于UTC的偏移量
is_dst	boolean	如果当前正处于夏令时范围则为TRUE，否则为FALSE

16.3.126 PG_TOTAL_MEMORY_DETAIL

PG_TOTAL_MEMORY_DETAIL视图显示某个数据库节点内存使用情况。

表 16-202 PG_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称
memorytype	text	内存的名称
memorymbytes	integer	内存使用的大小，单位MB

16.3.127 PG_TOTAL_USER_RESOURCE_INFO

PG_TOTAL_USER_RESOURCE_INFO视图显示所有用户资源使用情况，需要使用管理员用户进行查询。此视图在参数use_workload_manager为on时才有效。

表 16-203 PG_TOTAL_USER_RESOURCE_INFO 字段

名称	类型	描述
username	name	用户名。
used_memory	integer	正在使用的内存大小，单位MB。
total_memory	integer	可以使用的内存大小，单位MB。值为0表示未限制最大可用内存，其限制取决于数据库最大可用内存。

名称	类型	描述
used_cpu	double precision	正在使用的CPU核数（仅统计复杂作业CPU使用情况，且该值为相关控制组的CPU使用统计值）。
total_cpu	integer	在该机器节点上，用户关联控制组的CPU核数总和。
used_space	bigint	已使用的永久表存储空间大小，单位KB。
total_space	bigint	可使用的永久表存储空间大小，单位KB，值为-1表示未限制永久表存储空间。
used_temp_space	bigint	已使用的临时表存储空间大小，单位KB。
total_temp_space	bigint	可使用的临时表存储空间大小，单位KB，值为-1表示未限制临时表存储空间。
used_spill_space	bigint	已使用的算子落盘空间大小，单位KB。
total_spill_space	bigint	可使用的算子落盘空间大小，单位KB，值为-1表示未限制算子落盘空间。
read_kbytes	bigint	CN：过去5秒内，该用户在所有DN上复杂作业read的字节总数。(单位KB) DN：实例启动至当前时间为止，该用户复杂作业read的字节总数。(单位KB)
write_kbytes	bigint	CN：过去5秒内，该用户在所有DN上复杂作业write的字节总数。(单位KB) DN：实例启动至当前时间为止，该用户复杂作业write的字节总数。(单位KB)
read_counts	bigint	CN：过去5秒内，该用户在所有DN上复杂作业read的次数之和。(单位次) DN：实例启动至当前时间为止，该用户复杂作业read的次数之和。(单位次)
write_counts	bigint	CN：过去5秒内，该用户在所有DN上复杂作业write的次数之和。(单位次) DN：实例启动至当前时间为止，该用户复杂作业write的次数之和。(单位次)
read_speed	double precision	CN：过去5秒内，该用户在单个DN上复杂作业read平均速率。(单位KB/s) DN：过去5秒内，该用户在该DN上复杂作业read平均速率。(单位KB/s)

名称	类型	描述
write_speed	double precision	CN：过去5秒内，该用户在单个DN上复杂作业write平均速率。(单位KB/s) DN：过去5秒内，该用户在该DN上复杂作业write平均速率。(单位KB/s)

16.3.128 PG_USER

PG_USER视图提供了访问数据库用户的信息。

表 16-204 PG_USER 字段

名称	类型	描述
usename	name	用户名。
usesysid	oid	此用户的ID。
usecreatedb	boolean	用户是否可以创建数据库。
usesuper	boolean	用户是否是拥有最高权限的初始系统管理员。
usecatupd	boolean	用户是否可以直接更新系统表。只有usesysid=10的初始系统管理员拥有此权限。其他用户无法获得此权限。
userepl	boolean	用户是否可以复制数据流。
passwd	text	密文存储后的用户口令，始终为********。
valbegin	timestamp with time zone	帐户的有效开始时间；如果没有设置有效开始时间，则为NULL。
valuntil	timestamp with time zone	帐户的有效结束时间；如果没有设置有效结束时间，则为NULL。
respool	name	用户所在的资源池。
parentid	oid	父用户OID
spacelimit	text	永久表存储空间限额。
tempspacelimit	text	临时表存储空间限额。
spillspacelimit	text	算子落盘空间限额。
useconfig	text[]	运行时配置参数的会话缺省。
nodegroup	name	用户关联的逻辑集群名字，如果该用户没有管理逻辑集群，则该字段为空。

16.3.129 PG_USER_MAPPINGS

PG_USER_MAPPINGS视图提供访问关于用户映射的信息的接口。

这个视图只是一个PG_USER_MAPPING的可读部分的视图化表现，如果用户无权使用它则查询此表时，有些选项字段会显示为空。

表 16-205 PG_USER_MAPPINGS 字段

名字	类型	引用	描述
umid	oid	PG_USER_MAPPING.oid	用户映射的OID。
srvid	oid	PG_FOREIGN_SERVER.oid	包含这个映射的外部服务器的OID。
srvname	name	PG_FOREIGN_SERVER.srvname	外部服务器的名字。
umuser	oid	PG_AUTHID.oid	被映射的本地角色的OID，如果用户映射是公共的则为0。
usename	name	-	被映射的本地用户的名字。
umoptions	text[]	-	如果当前用户是外部服务器的所有者，则为用户映射指定选项， 使用“keyword=value”字符串，否则为null。

16.3.130 PG_VIEWS

PG_VIEWS视图提供访问数据库中每个视图的有用信息。

表 16-206 PG_VIEWS 字段

名称	类型	引用	描述
schemaname	name	PG_NAMESPACE.nspname	包含视图的模式名
viewname	name	PG_CLASS.relname	视图的名称
viewowner	name	PG_AUTHID.Erolname	视图的所有者
definition	text	-	视图的定义

16.3.131 PG_WLM_STATISTICS

PG_WLM_STATISTICS视图显示作业结束后或已被处理异常后的负载管理相关信息。

表 16-207 PG_WLM_STATISTICS 字段

名称	类型	描述
statement	text	执行了异常处理的语句
block_time	bigint	语句执行前的阻塞时间
elapsed_time	bigint	语句的实际执行时间
total_cpu_time	bigint	语句执行异常处理时DN上CPU使用的总时间
qualification_time	bigint	语句检查倾斜率的时间周期
cpu_skew_percent	integer	语句在执行异常处理时DN上CPU使用的倾斜率
control_group	text	语句执行异常处理时所使用的Cgroups
status	text	语句执行异常处理后的状态，包括： ● pending：执行前预备状态 ● running：执行进行状态 ● finished：执行正常结束 ● abort：执行异常终止
action	text	语句执行的异常处理动作，包括： ● abort：执行终止操作 ● adjust：执行Cgroups调整操作，目前只有降级操作 ● finish：正常结束

16.3.132 PGXC_COMM_CLIENT_INFO

PGXC_COMM_CLIENT_INFO视图存储所有节点客户端连接信息（DN上查询该视图显示CN连接DN的信息）。

表 16-208 PGXC_COMM_CLIENT_INFO 字段

名称	类型	描述
node_name	text	当前节点的名称。
app	text	客户端应用名。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
query_id	bigint	查询ID，对应debug_query_id。

名称	类型	描述
socket	integer	如果是物理连接，展示socket。
remote_ip	text	对端节点IP。
remote_port	text	对端节点port。
logic_id	integer	如果是逻辑连接，展示sid，显示-1时表示当前连接是物理连接。

16.3.133 PGXC_COMM_DELAY

PGXC_COMM_DELAY视图展示所有DN的通信库时延状态。

表 16-209 PGXC_COMM_DELAY 字段

名称	类型	描述
node_name	text	节点名称
remote_name	text	连接对端节点的对端节点名称
remote_host	text	连接对端IP的对端地址
stream_num	integer	当前物理连接使用的stream逻辑连接数量
min_delay	integer	当前物理连接一分钟内探测到的最小时延，单位微秒 **说明** 负数结果无效，请重新等待时延状态更新后再执行查询。
average	integer	当前物理连接一分钟内探测时延的平均值，单位微秒
max_delay	integer	当前物理连接一分钟内探测到的最大时延，单位微秒

16.3.134 PGXC_COMM_RECV_STREAM

PGXC_COMM_RECV_STREAM视图展示所有DN上的通信库接收流状态。

表 16-210 PGXC_COMM_RECV_STREAM 字段

名称	类型	描述
node_name	text	节点名称
local_tid	bigint	使用此通信流的线程ID
remote_name	text	连接对端节点名称

名称	类型	描述
remote_tid	bigint	连接对端线程ID
idx	integer	通信对端DN在本DN内的标识编号
sid	integer	通信流在物理连接中的标识编号
tcp_sock	integer	通信流所使用的tcp通信socket
state	text	通信流当前的状态
query_id	bigint	通信流对应的debug_query_id编号
pn_id	integer	通信流所执行查询的plan_node_id编号
send_smp	integer	通信流所执行查询send端的smpid编号
recv_smp	integer	通信流所执行查询recv端的smpid编号
recv_bytes	bigint	通信流接收的数据总量，单位Byte
time	bigint	通信流当前生命周期使用时长，单位ms
speed	bigint	通信流的平均接收速率，单位Byte/s
quota	bigint	通信流当前的通信配额值，单位Byte
buff_usize	bigint	通信流当前缓存的数据大小，单位Byte

16.3.135 PGXC_COMM_SEND_STREAM

PGXC_COMM_SEND_STREAM视图展示所有DN上的通信库发送流状态。

表 16-211 PGXC_COMM_SEND_STREAM 字段

名称	类型	描述
node_name	text	节点名称。
local_tid	bigint	使用此通信流的线程ID。
remote_name	text	连接对端节点名称。
remote_tid	bigint	连接对端线程ID。
idx	integer	通信对端DN在本DN内的标识编号。
sid	integer	通信流在物理连接中的标识编号。
tcp_sock	integer	通信流所使用的tcp通信socket。
state	text	通信流当前的状态。
query_id	bigint	通信流对应的debug_query_id编号。
pn_id	integer	通信流所执行查询的plan_node_id编号。

名称	类型	描述
send_smp	integer	通信流所执行查询send端的smpid编号。
recv_smp	integer	通信流所执行查询recv端的smpid编号。
send_bytes	bigint	通信流发送的数据总量，单位Byte。
time	bigint	通信流当前生命周期使用时长，单位ms。
speed	bigint	通信流的平均发送速率，单位Byte/s。
quota	bigint	通信流当前的通信配额值，单位Byte。
wait_quota	bigint	通信流等待quota值产生的额外时间开销，单位ms。

16.3.136 PGXC_COMM_STATUS

PGXC_COMM_STATUS视图展示所有DN的通信库状态。

表 16-212 PGXC_COMM_STATUS 字段

名称	类型	描述
node_name	text	节点名称。
rxpck/s	integer	节点通信库接收速率，单位Byte/s。
txpck/s	integer	节点通信库发送速率，单位Byte/s。
rxkB/s	bigint	节点通信库接收速率，单位KByte/s。
txkB/s	bigint	节点通信库发送速率，单位KByte/s。
buffer	bigint	cmailbox的buffer大小。
memKB(libcomm)	bigint	libcomm进程通信内存大小，单位Byte。
memKB(libpq)	bigint	libpq进程通信内存大小，单位Byte。
%USED(PM)	integer	postmaster线程实时使用率。
%USED (sflow)	integer	gs_sender_flow_controller线程实时使用率。
%USED (rflow)	integer	gs_receiver_flow_controller线程实时使用率。
%USED (rloop)	integer	多个gs_receivers_loop线程中最高的实时使用率。
stream	integer	当前使用的逻辑连接总数。

16.3.137 PGXC_GET_STAT_ALL_TABLES

PGXC_GET_STAT_ALL_TABLES视图获取各表的插入、更新、删除以及脏页率信息。

对于高脏页率的系统表，建议在确认当前没有人操作该系统表时，再执行VACUUM FULL。

建议对脏页率超过80%的非系统表执行VACUUM FULL，用户也可根据业务场景自行选择是否执行VACUUM FULL。

表 16-213 PGXC_GET_STAT_ALL_TABLES 字段

名称	类型	描述
relid	oid	表的OID
relname	name	表名
schemaname	name	表的模式名
n_tup_ins	numeric	插入的元组条数
n_tup_upd	numeric	更新的元组条数
n_tup_del	numeric	删除的元组条数
n_live_tup	numeric	live元组的条数
n_dead_tup	numeric	dead元组的条数
page_dirty_rate	numeric(5,2)	表的脏页率信息(%)

16.3.138 PGXC_GET_TABLE_SKEWNESS

PGXC_GET_TABLE_SKEWNESS视图展示当前库中表的数据分布倾斜情况。

表 16-214 PGXC_GET_TABLE_SKEWNESS 字段

名称	类型	描述
schemaname	name	表所在的模式名。
tablename	name	表名。
totalsize	numeric	表的总大小，单位Byte。
avgsize	numeric(1000,0)	表大小平均值(totalsize/DN个数，该值为平均分布的理想情况下，表在各DN占用空间大小)。
maxratio	numeric(4,3)	单DN表大小最大值占比（表在各DN占用空间的最大值/totalsize）。
minratio	numeric(4,3)	单DN表大小最小值占比（表在各DN占用空间的最小值/totalsize）。
skewsize	bigint	表分布倾斜值（单DN表大小最大值 - 单DN表大小最小值）。

名称	类型	描述
skewratio	numeric(4,3)	表分布倾斜率（skewsize/totalsize）。
skewstddev	numeric(1000,0)	表分布标准方差（在表大小一定的情况下，该值越大表明表的整体分布情况越倾斜）。

16.3.139 PGXC_GTM_SNAPSHOT_STATUS

PGXC_GTM_SNAPSHOT_STATUS视图用于查看当前GTM上事务信息。

表 16-215 PGXC_GTM_SNAPSHOT_STATUS 字段

名称	类型	描述
xmin	xid	仍在运行的最小事务号。
xmax	xid	已完成的事务号最大的事务的下一个事务号。
csn	integer	待提交事务的序列号。
oldestxmin	xid	当前最早的活跃事务在其取快照时，所有运行事务号最小的事务。
xcnt	integer	当前活跃的事务个数。
running_xids	text	当前活跃的事务号。

16.3.140 PGXC_INSTR_UNIQUE_SQL

PGXC_INSTR_UNIQUE_SQL视图展示集群中所有CN节点的Unique SQL的完整统计信息。

需要有系统管理员权限才可以访问此视图，具体的字段请参考GS_INSTR_UNIQUE_SQL。

16.3.141 PGXC_NODE_ENV

PGXC_NODE_ENV视图提供获取集群中所有节点的环境变量信息。

表 16-216 PGXC_NODE_ENV 字段

名称	类型	描述
node_name	text	集群中所有节点的名称
host	text	集群中所有节点的主机名称

名称	类型	描述
process	integer	集群中所有节点的进程号
port	integer	集群中所有节点的端口号
installpath	text	集群中所有节点的安装目录
datapath	text	集群中所有节点的数据目录
log_directory	text	集群中所有节点的日志目录

16.3.142 PGXC_OS_THREADS

PGXC_OS_THREADS视图提供当前集群中所有正常节点下的线程状态信息。

表 16-217 PGXC_OS_THREADS 字段

名称	类型	描述
node_name	text	当前集群中所有正常节点的名称
pid	bigint	当前集群中所有正常节点进程中正在运行的线程号
lwpid	integer	与pid对应的轻量级线程号
thread_name	text	与pid对应的线程名称
creation_time	timestamp with time zone	与pid对应的线程创建的时间

16.3.143 PGXC_PREPARED_XACTS

PGXC_PREPARED_XACTS视图显示当前处于prepared阶段的两阶段事务。

表 16-218 PGXC_PREPARED_XACTS 字段

名字	类型	描述
pgxc_prepared_xact	text	查看当前处于prepared阶段的两阶段事务。

16.3.144 PGXC_RUNNING_XACTS

PGXC_RUNNING_XACTS视图主要功能是显示集群中各个节点运行事务的信息，字段内容和PG_RUNNING_XACTS相同。

表 16-219 PGXC_RUNNING_XACTS 字段

名称	类型	描述
handle	integer	事务在GTM对应的句柄。
gxid	xid	事务ID号。
state	tinyint	事务状态（3：prepared或者0：starting）。
node	text	节点名称。
xmin	xid	节点上当前数据涉及的最小事务号xmin。
vacuum	boolean	标志当前事务是否是lazy vacuum事务。
timeline	bigint	标识数据库重启次数。
prepare_xid	xid	处于prepared状态事务的id号，若不在prepared状态，值为0。
pid	bigint	事务对应的线程ID。
next_xid	xid	CN传给DN的事务ID号。

16.3.145 PGXC_STAT_ACTIVITY

PGXC_STAT_ACTIVITY视图显示当前集群下所有CN的当前用户查询相关的信息。

表 16-220 PGXC_STAT_ACTIVITY 字段

名称	类型	描述
coorname	text	当前集群下的CN名称。
datid	oid	用户会话在后端连接到的数据库OID。
datname	name	用户会话在后端连接到的数据库名称。
pid	bigint	后端线程ID。
usesysid	oid	登录该后端的用户OID。
usename	name	登录该后端的用户名。
application_name	text	连接到该后端的应用名。
client_addr	inet	连接到此后端的客户端的IP地址。 如果此字段是null，则表示通过服务器机器上UNIX套接字连接客户端或者这是内部进程，如autovacuum。
client_hostname	text	客户端的主机名，该字段是通过client_addr的反向DNS查找得到。仅在启动log_hostname且使用IP连接时才非空。

名称	类型	描述
client_port	integer	客户端用于与后端通讯的TCP端口号，如果使用Unix套接字，则为-1。
backend_start	timestamp with time zone	后端进程启动时间，即客户端连接服务器的时间。
xact_start	timestamp with time zone	当前事务的启动时间，如果没有事务是活跃的，则为null。如果当前查询是首个事务，则这列等同于query_start列。
query_start	timestamp with time zone	开始当前活跃查询的时间， 如果state的值不是active，则这个值是上一个查询的开始时间。
state_change	timestamp with time zone	状态最后一次改变的时间。
waiting	boolean	如果后端当前正等待锁则为true。
enqueue	text	语句当前排队状态。可能值是： ● waiting in queue：表示语句在排队中。 ● 空：表示语句正在运行。
state	text	该后端当前总体状态。可能值是： ● active：后端正在执行一个查询。 ● idle：后端正在等待一个新的客户端命令。 ● idle in transaction：后端在事务中，但事务中没有语句在执行。 ● idle in transaction (aborted)：后端在事务中，但事务中有语句执行失败。 ● fastpath function call：后端正在执行一个fast-path函数。 ● disabled：如果后端禁用track_activities，则报告此状态。 **说明** 只有系统管理员能查看到自己帐户所对应的会话状态。其他帐户的state信息为空。例如以judy用户连接数据库后，在pgxc_stat_activity中查看到的普通用户joe及初始用户dbadmin的state信息为空： SELECT datname, usename, usesysid, state,pid FROM pgxc_stat_activity; datname | usename | usesysid | state | pid ----------+---------+----------+--------+----------------- postgres | dbadmin | 10 | | 139968752121616 postgres | dbadmin | 10 | | 139968903116560 db_tpcds | judy | 16398 | active | 139968391403280 postgres | dbadmin | 10 | | 139968643069712 postgres | dbadmin | 10 | | 139968680818448 postgres | joe | 16390 | | 139968563377936 (6 rows)

名称	类型	描述
resource_pool	name	用户使用的资源池。
query_id	bigint	查询语句的ID。
query	text	此后端的最新查询。如果state状态是active（活跃的），此字段显示当前正在执行的查询。其他情况表示上一个查询。
connection_info	text	json格式字符串，记录当前连接数据库的驱动类型、驱动版本号、当前驱动的部署路径、进程属主用户等信息（参见**connection_info**）。

16.3.146 PGXC_STAT_BAD_BLOCK

PGXC_STAT_BAD_BLOCK视图显示集群所有节点从启动后，在读取数据时出现Page/CU校验失败的统计信息。

表 16-221 PGXC_STAT_BAD_BLOCK 字段

名字	类型	描述
nodename	text	节点名称
databaseid	integer	数据库OID
tablespaceid	integer	表空间OID
relfilenode	integer	文件对象ID
forknum	integer	文件类型
error_count	integer	出现校验失败的次数
first_time	timestamp with time zone	第一次出现的时间
last_time	timestamp with time zone	最近一次出现的时间

16.3.147 PGXC_SQL_COUNT

通过PGXC_SQL_COUNT视图，可以实时显示SELECT、INSERT、UPDATE、DELETE、MERGE INTO五种SQL、以及DDL、DML、DCL语句的节点级和用户级统计结果，识别当前业务负载较重的query类型，衡量整个集群和单个节点执行某种类型查询的能力。通过对以上几类SQL查询进行计数和响应时间统计，获得指定时刻的统计结果，经计算可以得到指定QPS等统计信息。例如，T1时刻，USER1的SELECT计数结果为X1，T2时刻为X2，则可计算得到该用户SELECT查询的QPS值为(X2-X1)/(T2-T1)。由此，可获得集群用户级QPS曲线图和集群吞吐情况，监测每个用户的业务负载是否发生剧烈变化。如果有剧烈变化，可以定位具体的语句类型(SELECT/INSERT/UPDATE/

DELETE/MERGE INTO)。同时观测QPS曲线可以获知问题发生时间点，结合其它工具，定位问题点。能够为集群性能调优、问题定位等提供依据。

PGXC_SQL_COUNT视图的字段与GS_SQL_COUNT一致，详见**表16-123**。

说明

当执行用户的MERGE INTO语句时，若能下推，在DN上收到的是MERGE INTO语句，将在DN节点上进行MERGE INTO类型计数，相应mergeinto_count列计数增加；若不能下推，在DN上收到的是UPDATE或INSERT语句，将在DN节点上进行UPDATE或INSERT类型计数，相应的update_count列或insert_count列计数增加。

16.3.148 PGXC_THREAD_WAIT_STATUS

通过CN节点查看PGXC_THREAD_WAIT_STATUS视图，可以查看集群全局各个节点上所有SQL语句产生的线程之间的调用层次关系，以及各个线程的阻塞等待状态，从而更容易定位进程停止响应问题以及类似现象的原因。

PGXC_THREAD_WAIT_STATUS视图和PG_THREAD_WAIT_STATUS视图列定义完全相同，这是由于PGXC_THREAD_WAIT_STATUS视图本质是到集群中各个节点上查询PG_THREAD_WAIT_STATUS视图汇总的结果。

表 16-222 PGXC_THREAD_WAIT_STATUS 字段

名称	类型	描述
node_name	text	当前节点的名称。
db_name	text	数据库名称。
thread_name	text	线程名称。
query_id	bigint	查询ID，对应debug_query_id。
tid	bigint	当前线程的线程号。
lwtid	integer	当前线程的轻量级线程号。
ptid	integer	streaming线程的父线程。
tlevel	integer	streaming线程的层级。
smpid	integer	并行线程的ID。
wait_status	text	当前线程的等待状态。等待状态的详细信息请参见**表16-195**。
wait_event	text	如果wait_status是acquire lock、acquire lwlock、wait io三种类型，此列描述具体的锁、轻量级锁、IO的信息。否则是空。

例如：

在coordinator1执行一条语句之后长时间没有响应。可以创建另外一个连接到coordinator1上，查询coordinator1上的线程状态。

```
select * from pg_thread_wait_status where query_id > 0;
 node_name   | db_name  | thread_name  | query_id |       tid       | lwtid | ptid  | tlevel | smpid |
```

```
wait_status   |   wait_event
--------------+----------+--------------+----------+-----------------+-------+-------+---------+-------
+----------------------
 coordinator1 | postgres | gsql         | 20971544 | 140274089064208 | 22579 |       |      0 |     0 | wait node:
datanode4 |
(1 rows)
```

此外，可以查看该语句在全局范围内各个节点上的工作情况。如下所示，每个DN上都没有在等待的阻塞资源，因为读取的数据太多而执行较慢。

```
select * from pgxc_thread_wait_status where query_id=20971544;
  node_name   | db_name  | thread_name  | query_id |       tid        | lwtid | ptid  | tlevel | smpid |
wait_status   |   wait_event
--------------+----------+--------------+----------+-----------------+-------+-------+---------+-------
+----------------------
 datanode1    | postgres | coordinator1 | 20971544 | 139902867994384 | 22735 |       |      0 |     0 | wait
node: datanode3 |
 datanode1    | postgres | coordinator1 | 20971544 | 139902838634256 | 22970 | 22735 |      5 |     0 |
synchronize quit     |
 datanode1    | postgres | coordinator1 | 20971544 | 139902607947536 | 22972 | 22735 |      5 |     1 |
synchronize quit     |
 datanode2    | postgres | coordinator1 | 20971544 | 140632156796688 | 22736 |       |      0 |     0 | wait
node: datanode3 |
 datanode2    | postgres | coordinator1 | 20971544 | 140632030967568 | 22974 | 22736 |      5 |     0 |
synchronize quit     |
 datanode2    | postgres | coordinator1 | 20971544 | 140632081299216 | 22975 | 22736 |      5 |     1 |
synchronize quit     |
 datanode3    | postgres | coordinator1 | 20971544 | 140323627988752 | 22737 |       |      0 |     0 | wait
node: datanode3 |
 datanode3    | postgres | coordinator1 | 20971544 | 140323523131152 | 22976 | 22737 |      5 |     0 | net
flush data       |
 datanode3    | postgres | coordinator1 | 20971544 | 140323548296976 | 22978 | 22737 |      5 |     1 | net
flush data
 datanode4    | postgres | coordinator1 | 20971544 | 140103024375568 | 22738 |       |      0 |     0 | wait
node: datanode3
 datanode4    | postgres | coordinator1 | 20971544 | 140102919517968 | 22979 | 22738 |      5 |     0 |
synchronize quit     |
 datanode4    | postgres | coordinator1 | 20971544 | 140102969849616 | 22980 | 22738 |      5 |     1 |
synchronize quit     |
 coordinator1 | postgres | gsql         | 20971544 | 140274089064208 | 22579 |       |      0 |     0 | wait node:
datanode4  |
(13 rows)
```

16.3.149 PGXC_TOTAL_MEMORY_DETAIL

PGXC_TOTAL_MEMORY_DETAIL视图显示集群内存使用情况。

表 16-223 PGXC_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。

名称	类型	描述
memorytype	text	内存使用的名称。 ● max_process_memory：GaussDB(DWS)集群实例所占用的内存大小。 ● process_used_memory：GaussDB(DWS)进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_shared_memory：最大共享内存。 ● shared_used_memory：已使用的共享内存。 ● max_cstore_memory：列存所允许使用的最大内存。 ● cstore_used_memory：列存已使用的内存大小。 ● max_sctpcomm_memory：通信库所允许使用的最大内存。 ● sctpcomm_used_memory：通信库已使用的内存大小。 ● sctpcomm_peak_memory：通信库的内存峰值。 ● other_used_memory：其他已使用的内存大小。
memorymbytes	integer	内存使用的大小，单位为MB。

16.3.150 PGXC_USER_TRANSACTION

PGXC_USER_TRANSACTION视图提供查询所有CN上用户相关的事务信息。需要有系统管理员权限才可以访问此视图。该视图仅在资源实时监控功能开启，即**enable_resource_track**为on时有效。

表 16-224 PGXC_USER_TRANSACTION 字段

名称	类型	描述
node_name	name	节点名称
usename	name	用户名称
commit_counter	bigint	提交次数
rollback_counter	bigint	回滚次数
resp_min	bigint	最小响应时间
resp_max	bigint	最大响应时间
resp_avg	bigint	平均响应时间
resp_total	bigint	响应时间总和

16.3.151 PGXC_VARIABLE_INFO

PGXC_VARIABLE_INFO视图用于查询集群中所有节点的xid、oid的状态。

表 16-225 PGXC_VARIABLE_INFO 字段

名称	类型	描述
node_name	text	节点名称
nextOid	oid	该节点下一次生成的OID
nextXid	xid	该节点下一次生成的事务号
oldestXid	xid	该节点最早的事务号。
xidVacLimit	xid	强制autovacuum的临界点
oldestXidDB	oid	该节点datafrozenxid最小的数据库OID
lastExtendCSNLogpage	integer	最后一次扩展csnlog的页面号
startExtendCSNLogpage	integer	csnlog扩展的起始页面号
nextCommitSeqNo	integer	该节点下次生成的csn号
latestCompletedXid	xid	该节点提交或者回滚后节点上的最新事务号
startupMaxXid	xid	该节点关机前的最后一个事务号

16.3.152 PGXC_WLM_OPERATOR_HISTORY

PGXC_WLM_OPERATOR_HISTORY视图显示在所有CN上执行作业结束时的算子信息。此视图用于Database Manager从数据库中查询数据，数据库中的数据会被定时清理，清理周期为3分钟。

需要有系统管理员权限才可以访问此视图。具体的字段请参考**表16-3**。

16.3.153 PGXC_WLM_OPERATOR_INFO

PGXC_WLM_OPERATOR_INFO视图显示在所有CN上执行作业结束时的算子信息。此视图的数据直接从系统表**GS_WLM_OPERATOR_INFO**获取。

需要有系统管理员权限才可以访问此视图。具体的字段请参考**表16-3**。

16.3.154 PGXC_WLM_OPERATOR_STATISTICS

PGXC_WLM_OPERATOR_STATISTICS视图显示在所有CN上正在执行作业的算子信息。

需要有系统管理员权限才可以访问此视图。具体的字段请参考**表16-125**。

16.3.155 PGXC_WLM_SESSION_INFO

PGXC_WLM_SESSION_INFO视图显示在所有CN上执行作业结束后的负载管理记录。此视图的数据直接从系统表**GS_WLM_SESSION_INFO**获取。

需要有系统管理员权限才可以访问此视图。具体的字段请参考**表16-126**。

16.3.156 PGXC_WLM_SESSION_HISTORY

PGXC_WLM_SESSION_HISTORY视图显示在所有CN上执行作业结束后的负载管理记录。此视图用于Database Manager从数据库中查询数据，数据库中的数据会被定时清理，清理周期为3分钟，详见**GS_WLM_SESSION_HISTORY**视图介绍。

需要有系统管理员权限才可以访问此视图。具体的字段请参考**表16-126**

16.3.157 PGXC_WLM_SESSION_STATISTICS

PGXC_WLM_SESSION_STATISTICS视图显示在所有CN上正在执行的作业的负载管理信息。

需要有系统管理员权限才可以访问此视图。具体的字段请参考**表16-127**。

16.3.158 PGXC_WLM_WORKLOAD_RECORDS

PGXC_WLM_WORKLOAD_RECORDS视图显示当前用户在每个CN上执行作业时在CN上的状态信息。需要有系统管理员权限才可以访问此视图。该视图仅在动态负载功能开启，即enable_dynamic_workload为on时有效。

表 16-226 PGXC_WLM_WORKLOAD_RECORDS 字段

名称	类型	描述
node_name	text	作业执行所在的CN的name

名称	类型	描述
thread_id	bigint	后端线程ID
processid	integer	线程的lwpid
timestamp	bigint	语句执行的开始时间
username	name	登录到该后端的用户名
memory	integer	语句所需的内存大小
active_points	integer	语句在资源池上消耗的资源点数
max_points	integer	资源在资源池上的最大资源数
priority	integer	作业的优先级
resource_pool	text	作业所在资源池
status	text	作业执行的状态，包括： pending： 阻塞状态 running: 执行状态 finished：结束状态 aborted：终止状态 unkown：未知状态
control_group	text	作业所使用的Cgroups
enqueue	text	作业的排队信息，包括： GLOBAL：全局排队 RESPOOL: 资源池排队 ACTIVE：不排队
query	text	正在执行的语句

16.3.159 PGXC_WORKLOAD_SQL_COUNT

PGXC_WORKLOAD_SQL_COUNT视图显示集群中所有CN节点上的Workload控制组内的SQL语句执行次数的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的执行次数统计，以及DDL、DML、DCL类型语句的执行次数统计。需要有系统管理员权限才可以访问此视图。

表 16-227 PGXC_WORKLOAD_SQL_COUNT 字段

名称	类型	描述
node_name	name	节点名称
workload	name	Workload控制组名称
select_count	bigint	SELECT数量

名称	类型	描述
update_count	bigint	UPDATE数量
insert_count	bigint	INSERT数量
delete_count	bigint	DELETE数量
ddl_count	bigint	DDL数量
dml_count	bigint	DML数量
dcl_count	bigint	DCL数量

16.3.160 PGXC_WORKLOAD_SQL_ELAPSE_TIME

PGXC_WORKLOAD_SQL_ELAPSE_TIME视图显示集群中所有CN节点上Workload控制组内SQL语句执行的响应时间的统计信息，包括SELECT、UPDATE、INSERT、DELETE语句的最大、最小、平均、以及总响应时间，单位为微秒。需要有系统管理员权限才可以访问此视图。

表 16-228 PGXC_WORKLOAD_SQL_ELAPSE_TIME 字段

名称	类型	描述
node_name	name	节点名称
workload	name	Workload控制组名称
total_select_elapse	bigint	SELECT总响应时间
max_select_elapse	bigint	SELECT最大响应时间
min_select_elapse	bigint	SELECT最小响应时间
avg_select_elapse	bigint	SELECT平均响应时间
total_update_elapse	bigint	UPDATE总响应时间
max_update_elapse	bigint	UPDATE最大响应时间
min_update_elapse	bigint	UPDATE最小响应时间
avg_update_elapse	bigint	UPDATE平均响应时间
total_insert_elapse	bigint	INSERT总响应时间
max_insert_elapse	bigint	INSERT最大响应时间
min_insert_elapse	bigint	INSERT最小响应时间
avg_insert_elapse	bigint	INSERT平均响应时间
total_delete_elapse	bigint	DELETE总响应时间
max_delete_elapse	bigint	DELETE最大响应时间

名称	类型	描述
min_delete_elapse	bigint	DELETE最小响应时间
avg_delete_elapse	bigint	DELETE平均响应时间

16.3.161 PGXC_WORKLOAD_TRANSACTION

PGXC_WORKLOAD_TRANSACTION视图提供查询所有CN上Workload控制组相关的事务信息。需要有系统管理员权限才可以访问此视图。该视图仅在资源实时监控功能开启，即**enable_resource_track**为on时有效。

表 16-229 PGXC_WORKLOAD_TRANSACTION 字段

名称	类型	描述
node_name	name	节点名称
workload	name	Workload控制组名称
commit_counter	bigint	提交次数
rollback_counter	bigint	回滚次数
resp_min	bigint	最小响应时间，单位微秒
resp_max	bigint	最大响应时间，单位微秒
resp_avg	bigint	平均响应时间，单位微秒
resp_total	bigint	响应时间总和，单位微秒

16.3.162 PLAN_TABLE

PLAN_TABLE显示用户通过执行EXPLAIN PLAN收集到的计划信息。计划信息的生命周期是session级别，session退出后相应的数据将被清除。同时不同session和不同user间的数据是相互隔离的。

表 16-230 PLAN_TABLE 字段

名称	类型	描述
statement_id	varchar2(30)	用户输入的查询标签。
plan_id	bigint	查询标识。
id	int	查询生成的计划中的每一个执行算子的编号。
operation	varchar2(30)	计划中算子的操作描述。
options	varchar2(255)	操作选项。

名称	类型	描述
object_name	name	操作对应的对象名，非查询中使用到的对象别名。来自于用户定义。
object_type	varchar2(30)	对象类型。
object_owner	name	对象所属schema，来自于用户定义。
projection	varchar2(4000)	操作输出的列信息。

说明

- object_type取值范围为PG_CLASS中定义的relkind类型（TABLE普通表，INDEX索引，SEQUENCE序列，VIEW视图，FOREIGN TABLE外表，COMPOSITE TYPE复合类型，TOASTVALUE TOAST表）和计划使用到的rtekind(SUBQUERY, JOIN, FUNCTION, VALUES, CTE, REMOTE_QUERY)。
- object_owner对于RTE来说是计划中使用的对象描述，非用户定义的类型不存在object_owner。
- statement_id、object_name、object_owner、projection字段内容遵循用户定义的大小写存储，其它字段内容采用大写存储。
- 支持用户对PLAN_TABLE进行SELECT和DELETE操作，不支持其它DML操作。

16.3.163 PLAN_TABLE_DATA

PLAN_TABLE_DATA存储了用户通过执行EXPLAIN PLAN收集到的计划信息。与PLAN_TABLE视图不同的是PLAN_TABLE_DATA表存储了所有session和user执行EXPLAIN PLAN收集的计划信息。

表 16-231 PLAN_TABLE 字段

名称	类型	描述
session_id	text	表示插入该条数据的会话，由服务线程启动时间戳和服务线程ID组成。受非空约束限制。
user_id	oid	用户ID，用于标识触发插入该条数据的用户。受非空约束限制。
statement_id	varchar2(30)	用户输入的查询标签。
plan_id	bigint	查询标识。
id	int	计划中的节点编号。
operation	varchar2(30)	操作描述。
options	varchar2(255)	操作选项。
object_name	name	操作对应的对象名，来自于用户定义。
object_type	varchar2(30)	对象类型。

名称	类型	描述
object_owner	name	对象所属schema，来自于用户定义。
projection	varchar2(4000)	操作输出的列信息。

说明

- PLAN_TABLE_DATA中包含了当前节点所有用户、所有会话的数据，仅管理员有访问权限。普通用户可以通过**PLAN_TABLE**视图查看属于自己的数据。
- 对于不活跃（已退出）的会话，其在PLAN_TABLE_DATA中的数据会在一定时间（默认5min）后被gs_clean清理。用户也可以手动执行gs_clean -C选项对表中不活跃的会话数据进行清理。
- PLAN_TABLE_DATA中的数据是用户通过执行EXPLAIN PLAN命令后由系统自动插入表中，因此禁止用户手动对数据进行插入或更新，否则会引起表中的数据混乱。需要对表中数据删除时，建议通过**PLAN_TABLE**视图。
- statement_id、object_name、object_owner和projection字段内容遵循用户定义的大小写存储，其它字段内容采用大写存储。

16.3.164 PV_FILE_STAT

PV_FILE_STAT视图通过对数据文件IO的统计，反映数据的IO性能，用以发现IO操作异常等性能问题。

表 16-232 PV_FILE_STAT 字段

名称	类型	描述
filenum	oid	文件标识
dbid	oid	数据库标识
spcid	oid	表空间标识
phyrds	bigint	读物理文件的数目
phywrts	bigint	写物理文件的数目
phyblkrd	bigint	读物理文件块的数目
phyblkwrt	bigint	写物理文件块的数目
readtim	bigint	读文件的总时长，单位微秒
writetim	bigint	写文件的总时长，单位微秒
avgiotim	bigint	读写文件的平均时长，单位微秒
lstiotim	bigint	最后一次读文件时长，单位微秒
miniotim	bigint	读写文件的最小时长，单位微秒
maxiowtm	bigint	读写文件的最大时长，单位微秒

16.3.165 PV_INSTANCE_TIME

PV_INSTANCE_TIME视图用于统计进程的运行时间信息及各执行阶段所消耗时间，单位为微秒。

提供当前节点下的各种时间消耗信息，主要分为以下类型：

- DB_TIME: 作业在多核下的有效时间花费。
- CPU_TIME：CPU时间的消耗。
- EXECUTION_TIME：执行器内花费的时间。
- PARSE_TIME：SQL解析的时间花费。
- PLAN_TIME：生成Plan的时间花费。
- REWRITE_TIME：SQL重写的时间消耗。
- PL_EXECUTION_TIME ： plpgsql（存储过程）的执行时间.
- PL_COMPILATION_TIME：plpgsql（存储过程）编译时间。
- NET_SEND_TIME：网络上的时间花销。
- DATA_IO_TIME：IO时间上的花销。

表 16-233 PV_INSTANCE_TIME 字段

名称	类型	描述
stat_id	integer	类型编号
stat_name	text	运行时间类型名称
value	bigint	运行时间值

16.3.166 PV_OS_RUN_INFO

PV_OS_RUN_INFO视图显示当前操作系统运行的状态信息。

表 16-234 PV_OS_RUN_INFO 字段

名称	类型	描述
id	integer	编号
name	text	操作系统运行状态名称
value	numeric	操作系统运行状态值
comments	text	操作系统运行状态注释
cumulative	boolean	操作系统运行状态的值是否为累加值

16.3.167 PV_SESSION_MEMORY

PV_SESSION_MEMORY视图统计Session级别的内存使用情况，包含执行作业在数据节点上Postgres线程和Stream线程分配的所有内存。

表 16-235 PV_SESSION_MEMORY 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识。
init_mem	integer	当前正在执行作业进入执行器前已分配的内存，单位MB。
used_mem	integer	当前正在执行作业已分配的内存，单位MB。
peak_mem	integer	当前正在执行作业已分配的内存峰值，单位MB。

16.3.168 PV_SESSION_MEMORY_DETAIL

PV_SESSION_MEMORY_DETAIL统计线程的内存使用情况，以MemoryContext节点来统计。

其中内存上下文“TempSmallContextGroup”，记录当前线程中所有内存上下文字段“totalsize”小于8192字节的信息汇总，并且内存上下文统计计数记录到“usedsize”字段中。所以在视图中，“TempSmallContextGroup”内存上下文中的“totalsize”和“freesize”是该线程中所有内存上下文“totalsize”小于8192字节的汇总总和，usedsize字段表示统计的内存上下文个数。

可通过"select * from pv_session_memctx_detail(threadid, '');"将某个线程所有内存上下文信息记录到“/tmp/dumpmem”目录下的“threadid_timestamp.log”文件中。其中threadid可通过下表sessid中获得。

表 16-236 PV_SESSION_MEMORY_DETAIL 字段

名称	类型	描述
sessid	text	线程启动时间+线程标识（字符串信息为timestamp.threadid）。
sesstype	text	线程名称。
contextname	text	内存上下文名称。
level	smallint	当前上下文在整体内存上下文中的层级。
parent	text	父内存上下文名称。
totalsize	bigint	当前内存上下文的内存总数，单位Byte。
freesize	bigint	当前内存上下文中已释放的内存总数，单位Byte。

名称	类型	描述
usedsize	bigint	当前内存上下文中已使用的内存总数，单位Byte；“TempSmallContextGroup”内存上下文中该字段含义为统计计数。

16.3.169 PV_SESSION_STAT

PV_SESSION_STAT视图以会话线程或AutoVacuum线程为单位，统计会话状态信息。

表 16-237 PV_SESSION_STAT 字段

名称	类型	描述
sessid	text	线程标识+线程启动时间
statid	integer	统计编号
statname	text	统计会话名称
statunit	text	统计会话单位
value	bigint	统计会话值

16.3.170 PV_SESSION_TIME

PV_SESSION_TIME视图用于统计会话线程的运行时间信息及各执行阶段所消耗时间，单位为微秒。

表 16-238 PV_SESSION_TIME 字段

名称	类型	描述
sessid	text	线程标识+线程启动时间
stat_id	integer	统计编号
stat_name	text	运行时间类型名称
value	bigint	运行时间值

16.3.171 PV_TOTAL_MEMORY_DETAIL

PV_TOTAL_MEMORY_DETAIL视图统计当前数据库节点使用内存的信息，单位为MB。

表 16-239 PV_TOTAL_MEMORY_DETAIL 字段

名称	类型	描述
nodename	text	节点名称。
memorytype	text	内存类型，包括以下几种： ● max_process_memory：GaussDB(DWS)集群实例所占用的内存大小。 ● process_used_memory：GaussDB(DWS)进程所使用的内存大小。 ● max_dynamic_memory：最大动态内存。 ● dynamic_used_memory：已使用的动态内存。 ● dynamic_peak_memory：内存的动态峰值。 ● dynamic_used_shrctx：最大动态共享内存上下文。 ● dynamic_peak_shrctx：共享内存上下文的动态峰值。 ● max_shared_memory：最大共享内存。 ● shared_used_memory：已使用的共享内存。 ● max_cstore_memory：列存所允许使用的最大内存。 ● cstore_used_memory：列存已使用的内存大小。 ● max_sctpcomm_memory：通信库所允许使用的最大内存。 ● sctpcomm_used_memory：通信库已使用的内存大小。 ● sctpcomm_peak_memory：通信库的内存峰值。 ● other_used_memory：其他已使用的内存大小。
memorymbytes	integer	内存类型分配内存的大小。

16.3.172 REDACTION_COLUMNS

REDACTION_COLUMNS视图展示当前数据库内所有脱敏列信息。

表 16-240 REDACTION_COLUMNS 字段

名称	类型	描述
object_owner	name	脱敏对象owner
object_name	name	脱敏对象名称
column_name	name	脱敏列名称
function_type	integer	脱敏类型

名称	类型	描述
function_parameters	text	脱敏类型为partial类型时的参数
regexp_pattern	text	脱敏类型为regexp时，pattern串
regexp_replace_string	text	脱敏类型为regexp时，替换串
regexp_position	integer	脱敏类型为regexp时，起始替换位置
regexp_occurrence	integer	脱敏类型为regexp时，替换次数
regexp_match_parameter	text	脱敏类型为regexp时，正则控制参数
column_description	text	脱敏列描述信息

16.3.173 REDACTION_POLICIES

REDACTION_POLICIES视图展示当前数据库内所有脱敏对象信息。

表 16-241 REDACTION_POLICIES 字段

名称	类型	描述
object_owner	name	脱敏对象owner
object_name	name	脱敏对象名称
policy_name	name	脱敏策略名称
expression	text	策略生效表达式（针对用户）
enable	boolean	策略状态（开启、关闭）
policy_description	text	策略描述信息

16.3.174 USER_COL_COMMENTS

USER_COL_COMMENTS视图存储当前用户下表的列注释信息。

表 16-242 USER_COL_COMMENTS 字段

名称	类型	描述
column_name	character varying(64)	列名

名称	类型	描述
table_name	character varying(64)	表名
owner	character varying(64)	表的所有者
comments	text	注释

16.3.175 USER_CONSTRAINTS

USER_CONSTRAINTS视图存储当前用户下表中约束的信息。

表 16-243 USER_CONSTRAINTS 字段

名称	类型	描述
constraint_name	vcharacter varying(64)	约束名
constraint_type	text	约束类型 • c表示检查约束 • f表示外键约束 • p表示主键约束 • u表示唯一约束
table_name	character varying(64)	约束相关的表名
index_owner	character varying(64)	约束相关的索引的所有者（只针对唯一约束和主键约束）
index_name	character varying(64)	约束相关的索引名（只针对唯一约束和主键约束）

16.3.176 USER_CONS_COLUMNS

USER_CONS_COLUMNS视图存储当前用户下表中约束列的信息。

表 16-244 USER_CONS_COLUMNS 字段

名称	类型	描述
table_name	character varying(64)	约束相关的表名
column_name	character varying(64)	约束相关的列名

名称	类型	描述
constraint_name	character varying(64)	约束名
position	smallint	表中列的位置

16.3.177 USER_INDEXES

USER_INDEXES视图存储关于本模式下的索引信息。

表 16-245 USER_INDEXES 字段

名称	类型	描述
owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_name	character varying(64)	索引对应的表名
uniqueness	text	表示此索引是否为唯一索引
generated	character varying(1)	表示此索引的名字是否为系统生成
partitioned	character(3)	表示此索引是否具有分区表的性质

16.3.178 USER_IND_COLUMNS

USER_IND_COLUMNS视图存储当前用户下所有索引的字段信息。

表 16-246 USER_IND_COLUMNS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
column_name	name	列名
column_position	smallint	索引中列的位置

16.3.179 USER_IND_EXPRESSIONS

USER_IND_EXPRESSIONS视图存储当前用户下基于函数的表达式索引的信息。

表 16-247 USER_IND_EXPRESSIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引的所有者
index_name	character varying(64)	索引名
table_owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
column_expression	text	定义列的基于函数的索引表达式
column_position	smallint	索引中列的位置

16.3.180 USER_IND_PARTITIONS

USER_IND_PARTITIONS视图存储当前用户下的索引分区信息。

表 16-248 USER_IND_PARTITIONS 字段

名称	类型	描述
index_owner	character varying(64)	索引分区所属分区表索引的所有者的名称
schema	character varying(64)	索引分区所属分区表索引的模式
index_name	character varying(64)	索引分区所属分区表索引的名称
partition_name	character varying(64)	索引分区的名称
index_partition_usable	boolean	索引分区是否可用
high_value	text	索引分区所对应分区的上边界
def_tablespace_name	name	索引分区的表空间名称

16.3.181 USER_JOBS

USER_JOBS视图为当前用户所属定时任务的详细信息。

表 16-249 USER_JOBS 字段

名字	类型	描述
job	int4	作业ID。
log_user	name not null	创建者的UserName。
priv_user	name not null	作业执行者的UserName。
dbname	name not null	作业创建数据库名字。
start_date	timestamp without time zone	作业的开始时间。
start_suc	text	作业成功执行的开始时间。
last_date	timestamp without time zone	上次运行开始时间。
last_suc	text	上次成功运行的开始时间。
this_date	timestamp without time zone	正在运行任务的开始时间。
this suc	text	正在运行任务成功的开始时间。
next_date	timestamp without time zone	任务下次执行时间。
next suc	text	任务下次成功执行时间。
broken	text	任务状态 如果为Y，不尝试运行此任务。 如果为N，将尝试执行此任务。
status	char	当前任务的执行状态，取值范围：('r', 's', 'f', 'd')，默认为'r'，取值含义： ● r=running ● s=successfully finished ● f= job failed ● d=aborted
interval	text	用来计算下次运行时间的时间表达式，如果为nul，l则表示定时任务只执行一次。
failures	smallint	失败计数，作业连续执行失败16次，不再继续执行。

名字	类型	描述
what	text	可执行的作业。

16.3.182 USER_OBJECTS

USER_OBJECTS视图描述了当前用户拥有的数据库对象。

表 16-250 USER_OBJECTS 字段

名称	类型	描述
object_name	name	对象的名称
object_id	oid	对象的OID
object_type	name	对象的类型，包括TABLE、INDEX、SEQUENCE、VIEW
namespace	oid	对象所属的命名空间
created	timestamp with time zone	对象的创建时间
last_ddl_time	timestamp with time zone	对象的最后修改时间

须知

created和last_ddl_time支持的范围参见**PG_OBJECT**中的记录范围。

16.3.183 USER_PART_INDEXES

USER_PART_INDEXES视图存储当前用户下分区表索引的信息。

表 16-251 USER_PART_INDEXES 字段

名称	类型	描述
index_owner	character varying(64)	分区表索引的所有者名称
schema	character varying(64)	分区表索引的模式
index_name	character varying(64)	分区表索引的名称
table_name	character varying(64)	分区表索引所属的分区表名称

名称	类型	描述
partitioning_type	text	分区表的分区策略
partition_count	bigint	分区表索引的索引分区的个数
def_tablespace_name	name	分区表索引的表空间名称
partitioning_key_count	integer	分区表的分区键个数

16.3.184 USER_PART_TABLES

USER_PART_TABLES视图存储当前用户下分区表的信息。

表 16-252 USER_PART_TABLES 字段

名称	类型	描述
table_owner	character varying(64)	分区表的所有者名称
schema	character varying(64)	分区表的模式
table_name	character varying(64)	分区表的名称
partitioning_type	text	分区表的分区策略
partition_count	bigint	分区表的分区个数
def_tablespace_name	name	分区表的表空间名称
partitioning_key_count	integer	分区表的分区键个数

16.3.185 USER_PROCEDURES

USER_PROCEDURES视图存储关于本模式下的存储过程或函数信息。

表 16-253 USER_PROCEDURES 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者
object_name	character varying(64)	存储过程或函数名称
argument_number	smallint	存储过程入参个数

16.3.186 USER_SEQUENCES

USER_SEQUENCES视图存储关于本模式下的序列信息。

表 16-254 USER_SEQUENCES 字段

名称	类型	描述
sequence_owner	character varying(64)	序列的所有者
sequence_name	character varying(64)	序列的名称

16.3.187 USER_SOURCE

USER_SOURCE视图存储关于本模式下的存储过程或函数信息，且提供存储过程或函数定义的字段。

表 16-255 USER_SOURCE 字段

名称	类型	描述
owner	character varying(64)	存储过程或函数的所有者
name	character varying(64)	存储过程或函数的名称
text	text	存储过程或函数的定义

16.3.188 USER_SYNONYMS

USER_SYNONYMS视图存储当前用户可访问的同义词信息。

表 16-256 USER_SYNONYMS 字段

名称	类型	描述
schema_name	text	同义词所属模式名
synonym_name	text	同义词的名称
table_owner	text	关联对象的所有者
table_schema_name	text	关联对象所属模式名
table_name	text	关联对象名

16.3.189 USER_TAB_COLUMNS

USER_TAB_COLUMNS视图存储当前用户可访问的表字段信息。

表 16-257 USER_TAB_COLUMNS 字段

名称	类型	描述
owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
column_name	character varying(64)	列名
data_type	character varying(128)	列的数据类型
column_id	integer	创建表时列的序号
data_length	integer	列的字节长度
comments	text	注释
avg_col_len	numeric	列的平均长度（单位字节）
nullable	bpchar	该列是否允许为空，对于主键约束和非空约束，该值为n。
data_precision	integer	数据类型的精度，对于numeric数据类型有效，其他类型为NULL。
data_scale	integer	小数点右边的位数，对于numeric数据类型有效，其他类型为0。
char_length	numeric	列的长度（单位字符），只对varchar，nvarchar2，bpchar，char类型有效。

16.3.190 USER_TAB_COMMENTS

USER_TAB_COMMENTS视图存储当前用户所有表和视图的注释信息。

表 16-258 USER_TAB_COMMENTS 字段

名称	类型	描述
owner	character varying(64)	表或视图的所有者
table_name	character varying(64)	表或视图的名称
comments	text	注释

16.3.191 USER_TAB_PARTITIONS

USER_TAB_PARTITIONS视图存储当前用户下所有分区的信息。当前用户下每个分区表的每个分区在USER_TAB_PARTITIONS中都会有一条记录。

表 16-259 USER_TAB_PARTITIONS 字段

名称	类型	描述
table_owner	character varying(64)	分区表的所有者名称
schema	character varying(64)	分区表的模式
table_name	character varying(64)	分区表的名称
partition_name	character varying(64)	分区的名称
high_value	text	分区的上边界
tablespace_name	name	分区的表空间名称

16.3.192 USER_TABLES

USER_TABLES视图存储关于当前模式下的表信息。

表 16-260 USER_TABLES 字段

名称	类型	描述
owner	character varying(64)	表的所有者
table_name	character varying(64)	表名
tablespace_name	character varying(64)	表所在的表空间名称
status	character varying(8)	当前记录是否有效
temporary	character(1)	是否为临时表 • Y表示是临时表 • N表示不是临时表
dropped	character varying	当前记录是否已删除 • YES表示已删除 • NO表示未删除
num_rows	numeric	表的估计行数

16.3.193 USER_TRIGGERS

USER_TRIGGERS视图存储关于当前用户下的触发器信息。

表 16-261 USER_TRIGGERS 字段

名称	类型	描述
trigger_name	character varying(64)	触发器名称

名称	类型	描述
table_name	character varying(64)	关系表名称
table_owner	character varying(64)	角色名称

16.3.194 USER_VIEWS

USER_VIEWS视图存储关于当前模式下的所有视图信息。

表 16-262 USER_VIEWS 字段

名称	类型	描述
owner	character varying(64)	视图的所有者
view_name	character varying(64)	视图名称

16.3.195 V$SESSION

V$SESSION视图存储当前会话的所有会话信息。

表 16-263 V$SESSION 字段

名称	类型	描述
sid	bigint	当前活动的后台进程的OID。
serial#	integer	当前活动的后台进程的序号，在GaussDB(DWS)中为0。
user#	oid	登录此后台进程的用户的OID。
username	name	登录此后台进程的用户名。

16.3.196 V$SESSION_LONGOPS

V$SESSION_LONGOPS视图存储当前正在执行的操作的进度。

表 16-264 V$SESSION_LONGOPS 字段

名称	类型	描述
sid	bigint	当前正在执行的后台进程的OID。
serial#	integer	当前正在执行的后台进程的序号，在GaussDB(DWS)中为0。

名称	类型	描述
sofar	integer	目前完成的工作量，在GaussDB(DWS)中为空。
totalwork	integer	工作总量，在GaussDB(DWS)中为空。

17 Information Schema

信息模式本身是一个名为information_schema的模式。这个模式自动存在于所有数据库中。信息模式由一组视图构成，它们包含定义在当前数据库中对象的信息。这个模式的拥有者是初始数据库用户，并且该用户自然地拥有这个模式上的所有特权，包括删除它的能力。

18 工具参考

在使用GaussDB(DWS)过程中，经常需要对集群进行安装、升级、扩容、卸载以及集群健康管理。为了简单、方便的维护集群，GaussDB(DWS)提供了一系列的集群管理工具。

18.1 gs_dump

背景信息

gs_dump是GaussDB(DWS)用于导出数据库相关信息的工具，用户可以自定义导出一个数据库或其中的对象（模式、表、视图等）。支持导出的数据库可以是默认数据库postgres，也可以是自定义数据库。

gs_dump工具在进行数据导出时，其他用户可以访问集群数据库（读或写）。

gs_dump工具支持导出完整一致的数据。例如，T1时刻启动gs_dump导出A数据库，那么导出数据结果将会是T1时刻A数据库的数据状态，T1时刻之后对A数据库的修改不会被导出。

gs_dump支持将数据库信息导出至纯文本格式的SQL脚本文件或其他归档文件中。

- 纯文本格式的SQL脚本文件：包含将数据库恢复为其保存时的状态所需的SQL语句。通过gsql运行该SQL脚本文件，可以恢复数据库。即使在其他主机和其他数据库产品上，只要对SQL脚本文件稍作修改，也可以用来重建数据库。
- 归档格式文件：包含将数据库恢复为其保存时的状态所需的数据，可以是tar格式、目录归档格式或自定义归档格式，详见**表18-1**。该导出结果必须与gs_restore配合使用来恢复数据库，gs_restore工具在导入时，系统允许用户选择需要导入的内容，甚至可以在导入之前对等待导入的内容进行排序。

主要功能

gs_dump可以创建四种不同的导出文件格式，通过**[**-F或者--format=**]**选项指定，具体如**表18-1**所示。

表 18-1 导出文件格式

格式名称	-F的参数值	说明	建议	对应导入工具
纯文本格式	p	纯文本脚本文件包含SQL语句和命令。命令可以由gsql命令行终端程序执行，用于重新创建数据库对象并加载表数据。	小型数据库，一般推荐纯文本格式。	使用gsql工具恢复数据库对象前，可根据需要使用文本编辑器编辑纯文本导出文件。
自定义归档格式	c	一种二进制文件。支持从导出文件中恢复所有或所选数据库对象。	中型或大型数据库，推荐自定义归档格式。	使用gs_restore可以选择要从自定义归档导出文件中导入相应的数据库对象。
目录归档格式	d	该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和blob对象对应的数据文件。	-	
tar归档格式	t	tar归档文件支持从导出文件中恢复所有或所选数据库对象。tar归档格式不支持压缩且对于单独表大小应小于8GB。	-	

说明

可以使用gs_dump程序将文件压缩为纯文本或自定义归档导出文件，减少导出文件的大小。生成纯文本导出文件时，默认不压缩。生成自定义归档导出文件时，默认进行中等级别的压缩。gs_dump程序无法压缩已归档导出文件。

注意事项

禁止修改导出的文件和内容，否则可能无法恢复成功。

为了保证数据一致性和完整性，gs_dump会对需要转储的表设置共享锁。如果表在别的事务中设置了共享锁，gs_dump会等待锁释放后锁定表。如果无法在指定时间内锁定某个表，转储会失败。用户可以通过指定--lock-wait-timeout选项，自定义等待锁超时时间。

语法

```
gs_dump [OPTION]... [DBNAME]
```

说明

“dbname”前面不需要加短或长选项。“dbname”指定要连接的数据库。

例如：

不需要-d，直接指定“dbname”。

```
gs_dump -p port_number  postgres -f dump1.sql
```

或者

```
export PGDATABASE=postgres
 gs_dump -p port_number -f dump1.sql
```

环境变量： PGDATABASE

参数说明

通用参数：

- -f, --file=FILENAME

  将输出发送至指定文件或目录。如果省略该参数，则使用标准输出。如果输出格式为(-F c/-F d/-F t)时，必须指定-f参数。如果-f的参数值含有目录，要求目录对当前用户具有读写权限。

- -F, --format=c|d|t|p

  选择输出格式。格式如下：

  - p|plain：输出一个文本SQL脚本文件（默认）。
  - c|custom：输出一个自定义格式的归档，并且以目录形式输出，作为gs_restore输入信息。该格式是最灵活的输出格式，因为能手动选择，而且能在恢复过程中将归档项重新排序。该格式默认状态下会被压缩。
  - d|directory：该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和blob对象对应的数据文件。
  - t|tar：输出一个tar格式的归档形式，作为gs_restore输入信息。tar格式与目录格式兼容；tar格式归档形式在提取过程中会生成一个有效的目录格式归档形式。但是，tar格式不支持压缩且对于单独表有8GB的大小限制。此外，表数据项的相应排序在恢复过程中不能更改。

    输出一个tar格式的归档形式，也可以作为gsql输入信息。

- -v, --verbose

  指定verbose模式。该选项将导致gs_dump向转储文件输出详细的对象注解和启动/停止次数，向标准错误流输出处理信息。

- -V, --version

  打印gs_dump版本，然后退出。

- -Z, --compress=0-9

  指定使用的压缩比级别。

  取值范围：0~9

  - 0表示无压缩。
  - 1表示压缩比最小，处理速度最快。
  - 9表示压缩比最大，处理速度最慢。

  针对自定义归档格式，该选项指定单个表数据片段的压缩，默认方式是以中等级别进行压缩。对于文本输出，设置非零压缩级别将会导致整个输出文件被压缩（类似通过gzip进行压缩），默认不压缩。tar归档格式目前不支持压缩。

- --lock-wait-timeout=TIMEOUT

  请勿在转储刚开始时一直等待以获取共享表锁。如果无法在指定时间内锁定某个表，就选择失败。可以以任何符合SET statement_timeout的格式指定超时时间。

- -?, --help

  显示gs_dump命令行参数帮助，然后退出。

转储参数：

- -a, --data-only

  只输出数据，不输出模式(数据定义)。转储表数据、大对象和序列值。

- -b, --blobs

  该参数为扩展预留接口，不建议使用。

- -c, --clean

  在将创建数据库对象的指令输出到备份文件之前，先将清理（删除）数据库对象的指令输出到备份文件中。（如果目标数据库中没有任何对象，gs_restore工具可能会输出一些提示性的错误信息）

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- -C, --create

  备份文件以创建数据库和连接到创建的数据库的命令开始。（如果命令脚本是这种方式执行，无所谓在运行脚本之前连接的是哪个数据库。）

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- -E, --encoding=ENCODING

  以指定的字符集编码创建转储。默认情况下，以数据库编码创建转储。（得到相同结果的另一个办法是将环境变量“PGCLIENTENCODING”设置为所需的转储编码。）

- -n, --schema=SCHEMA

  只转储与模式名称匹配的模式，此选项包括模式本身和所有它包含的对象。如果该选项没有指定，所有在目标数据库中的非系统模式将会被转储。写入多个-n选项来选择多个模式。此外，根据gsql的\d命令所使用的相同规则，模式参数可被理解成一个pattern，所以多个模式也可以通过在该pattern中写入通配符来选择。使用通配符时，注意给pattern打引号，防止shell扩展通配符。

  说明

  - 当-n已指定时，gs_dump不会转储已选模式所附着的任何其他数据库对象。因此，无法保证某个指定模式的转储结果能够自行成功地储存到一个空数据库中。
  - 当-n指定时，非模式对象不会被转储。

  转储支持多个模式的转储。多次输入-n schemaname转储多个模式。

  例如：

  ```
  gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -n sch1 -n sch2
  ```

  在上面这个例子中，sch1和sch2会被转储。

- -N, --exclude-schema=SCHEMA

  不转储任何与模式pattern匹配的模式。Pattern将参照针对-n的相同规则来理解。可以通过输入多次-N，不转储与任何pattern匹配的模式。

  当同时输入-n和-N时，会转储与至少一个-n选项匹配、与-N选项不匹配的模式。如果有-N没有-n，则不转储常规转储中与-N匹配的模式。

  转储过程支持排除多个模式。

在转储过程中，输入-N exclude schema name排除多个模式。

例如：

```
gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -N sch1 -N sch2
```

在上面这个例子中，sch1和sch2在转储过程中会被排除。

- -o, --oids

  转储每个表的对象标识符（OIDs），作为表的一部分数据。该选项用于应用以某种方式（例如：外键约束方式）参照了OID列的情况。如果不是以上这种情况，请勿使用该选项。

- -O, --no-owner

  不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_dump会发出ALTER OWNER或SET SESSION AUTHORIZATION语句设置所创建的数据库对象的归属。如果脚本正在运行，该语句不会执行成功，除非是由系统管理员触发（或是拥有脚本中所有对象的同一个用户）。通过指定-O，编写一个任何用户都能存储的脚本，且该脚本会授予该用户拥有所有对象的权限。

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- -s, --schema-only

  只转储对象定义（模式），而非数据。

- -S, --sysadmin=NAME

  该参数为扩展预留接口，不建议使用。

- -t, --table=TABLE

  指定转储的表（或视图、或序列、或外表）对象列表，可以使用多个-t选项来选择多个表，也可以使用通配符指定多个表对象。

  当使用通配符指定多个表对象时，注意给pattern打引号，防止shell扩展通配符。

  当使用-t时，-n和-N没有任何效应，这是因为由-t选择的表的转储不受那些选项的影响。

  说明

  -t参数选项个数必须小于等于100。

  如果-t参数选项个数大于100，建议使用参数--include-table-file来替换。

  当-t已指定时，gs_dump不会转储已选表所附着的任何其他数据库对象。因此，无法保证某个指定表的转储结果能够自行成功地储存到一个空数据库中。

  -t tablename只转储在默认搜索路径中可见的表。-t '*.tablename'转储数据库下所有模式下的tablename表。-t schema.table转储特定模式中的表。

  -t tablename不会导出表上的触发器信息。

  例如：

  ```
  gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -t schema1.table1 -t schema2.table2
  ```

  在上面这个例子中，schema1.table1和schema2.table2会被转储。

- --include-table-file=FILENAME

  指定需要dump的表文件。

- -T, --exclude-table=TABLE

  不转储的表（或视图、或序列、或外表）对象列表，可以使用多个-t选项来选择多个表，也可以使用通配符指定多个表对象。

  当同时输入-t和-T时，会转储在-t列表中，而不在-T列表中的表对象。

  例如：

```
gs_dump -h host_name -p port_number postgres -f backup/bkp_shl2.sql -T table1 -T table2
```

在上面这个例子中，table1和table2在转储过程中会被排除。

- --exclude-table-file=FILENAME

  指定不需要dump的表文件。

  说明

  同--include-table-file，其内容格式如下：

  schema1.table1

  schema2.table2

  ......

- -x, --no-privileges|--no-acl

  防止转储访问权限（授权/撤销命令）。

- --binary-upgrade

  该参数为扩展预留接口，不建议使用。

- --binary-upgrade-usermap="USER1=USER2"

  该参数为扩展预留接口，不建议使用。

- --column-inserts|--attribute-inserts

  以INSERT命令带列名（INSERT INTO表（列、…）值…）方式导出数据。这会导致恢复缓慢。但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。

- --disable-dollar-quoting

  该选项将禁止在函数体前使用美元符号$，并强制使用SQL标准字符串语法对其进行引用。

- --disable-triggers

  该参数为扩展预留接口，不建议使用。

- --exclude-table-data=TABLE

  指定不转储任何匹配表pattern的表这方面的数据。依照针对-t的相同规则理解该pattern。

  可多次输入--exclude-table-data来排除匹配任何pattern的表。当用户需要特定表的定义但不需要其中的数据时，这个选项很有帮助。

  排除数据库中所有表的数据，参见**--schema-only**。

- --inserts

  发出INSERT命令（而非COPY命令）时转储数据。这会导致恢复缓慢。

  但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。注意如果重排列顺序，可能会导致恢复整个失败。列顺序改变时，--column-inserts选项不受影响，虽然会更慢。

- --no-security-labels

  该参数为扩展预留接口，不建议使用。

- --no-tablespaces

  不输出选择表空间的命令。使用该选项，无论默认表空间是哪个，在恢复过程中所有对象都会被创建。

  该选项只对文本格式有意义。针对归档格式，可以在调用gs_restore时指定选项。

- --no-unlogged-table-data

该参数为扩展预留接口，不建议使用。

- --non-lock-table

  该参数为扩展预留接口，不建议使用。

- --quote-all-identifiers

  强制对所有标识符加引号。为了向后续版本迁移，且其中可能涉及引入额外关键词，在转储相应数据库时该选项会有帮助。

- --section=SECTION

  指定已转储的名称区段（ pre-data、data、和post-data ）。

- --serializable-deferrable

  转储过程中使用可串行化事务，以确保所使用的快照与之后的数据库状态一致；要实现该操作需要在无异常状况的事务流中等待某个点，因为这样才能保证转储成功，避免引起其他事务出现serialization_failure要重新再做。

  但是该选项对于灾难恢复没有益处。对于在原始数据库进行升级的时候，加载一个数据库的拷贝作为报告或其他只读加载共享的转储是有帮助的。没有这个选项，转储会反映一个与任何事务最终提交的序列化执行不一致的状态。

  如果当gs_dump启动时，读写事务仍处于非活动状态，即便使用该选项也不会对其产生影响。如果读写事务处于活动状态，转储的开始时间可能会延迟一段不确定的时间。

- --use-set-session-authorization

  输出符合SQL标准的SET SESSION AUTHORIZATION命令而不是ALTER OWNER命令来确定对象所有权。这样令转储更加符合标准，但是如果转储文件中的对象的历史有些问题，那么可能不能正确恢复。并且，使用SET SESSION AUTHORIZATION的转储需要数据库系统管理员的权限才能转储成功，而ALTER OWNER需要的权限则低得多。

- --with-encryption=AES128

  指定转储数据需用AES128进行加密。

- --with-key=KEY

  AES128密钥长度必须是16字节。

- --include-nodes

  将TO NODE/TO GROUP语句包含在已转储的CREATE TABLE/CREATE FOREIGN TABLE语句中。该参数只对HDFS表和外表生效。

- --include-extensions

  在转储中包含扩展。

- --include-depend-objs

  备份结果包含依赖于指定对象的对象信息。该参数需要同-t/--include-table-file参数关联使用才会生效。

- --exclude-self

  备份结果不包含指定对象自身的信息。该参数需要同-t/--include-table-file参数关联使用才会生效。

- --dont-overwrite-file

  文本、tar、以及自定义格式情况下会重写现有文件。这对目录格式不适用。

  例如：

  设想这样一种情景，即当前目录下backup.sql已存在。如果在输入命令中输入-f backup.sql选项时，当前目录恰好也生成backup.sql，文件就会被重写。

如果备份文件已存在，且输入--dont-overwrite-file选项，则会报告附带‘转储文件已经存在’信息的错误。

```
gs_dump -p port_number postgres -f backup.sql -F plain --dont-overwrite-file
```

说明

- -s/--schema-only和-a/--data-only不能同时使用。
- -c/--clean和-a/--data-only不能同时使用。
- --inserts/--column-inserts和-o/--oids不能同时使用，因为INSERT命令不能设置OIDS。
- --role和--rolepassword必须一起使用。
- --binary-upgrade-usermap和--binary-upgrade必须一起使用。
- --include-depend-objs/--exclude-self需要同-t/--include-table-file参数关联使用才会生效
- --exclude-self必须同--include-depend-objs一起使用。

连接参数：

- -h, --host=HOSTNAME

  指定主机名称。如果数值以斜杠开头，则被用作到Unix域套接字的路径。缺省从PGHOST环境变量中获取（如果已设置），否则，尝试一个Unix域套接字连接。

  该参数只针对集群外，对集群内本机只能用127.0.0.1。

  例如：主机名

  环境变量：PGHOST

- -p, --port=PORT

  指定主机端口。

  环境变量：PGPORT

- -U, --username=NAME

  指定所连接主机的用户名。

  环境变量：PGUSER

- -w, --no-password

  不出现输入密码提示。如果主机要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。

- -W, --password=PASSWORD

  指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W选项；如果没有-W选项，并且不是系统管理员，“Dump Restore工具”会提示用户输入密码。

- --role=ROLENAME

  指定创建转储使用的角色名。选择该选项，会使gs_dump连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有gs_dump要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以超系统管理员身份登录，而使用该选项能够在不违反该规定的情况下完成转储。

- --rolepassword=ROLEPASSWORD

  指定角色名的密码。

说明

场景1

如果某数据库集群有任何本地数据要添加到template1数据库，请谨慎将gs_dump的输出恢复到一个真正的空数据库中，否则可能会因为被添加对象的定义被复制，出现错误。要创建一个无本地添加的空数据库，需从template0而非template1复制，例如：

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

tar归档形式的文件大小不得超过8GB（tar文件格式的固有限制）。tar文档整体大小和任何其他输出格式没有限制，操作系统可能对此有要求。

由gs_dump生成的转储文件不包含优化程序用来做执行计划决定的统计数据。因此，最好从某转储文件恢复之后运行ANALYZE以确保最佳效果。转储文件不包含任何ALTER DATABASE…SET命令，这些设置由gs_dumpall转储，还有数据库用户和其他完成安装设置。

场景2

当SEQUENCE已经到达最大或最小值时，通过gs_dump来备份SEQUENCE值会因执行报错退出。可参考如下说明处理：

1. SEQUENCE已经到达最大值，但最大值小于2^63-2

报错示例：

sequence对象定义

```
CREATE SEQUENCE seq INCREMENT 1 MINVALUE 1 MAXVALUE 3 START WITH 1;
```

执行gs_dump备份

```
gs_dump -U dbadmin -W Bigdata@123 -p 37300 postgres -t PUBLIC.seq -f backup/MPPDB_backup.sql
gs_dump[port='37300'][postgres][2019-12-27 15:09:49]: The total objects number is 337.
gs_dump[port='37300'][postgres][2019-12-27 15:09:49]: WARNING:  get invalid xid from GTM because connection is not established
gs_dump[port='37300'][postgres][2019-12-27 15:09:49]: WARNING:  Failed to receive GTM rollback transaction response  for aborting prepared (null).
gs_dump: [port='37300'] [postgres] [archiver (db)] [2019-12-27 15:09:49] query failed: ERROR:  Can not connect to gtm when getting gxid, there is a connection error.
gs_dump: [port='37300'] [postgres] [archiver (db)] [2019-12-27 15:09:49] query was: RELEASE bfnextval
```

处理方法：

通过SQL语句连接postgres数据库，执行如下语句，修改sequence seq1的最大值。

```
gsql -p 37300 postgres -r -c "ALTER SEQUENCE PUBLIC.seq MAXVALUE 10;"
```

执行dump工具进行备份。

```
gs_dump -U dbadmin -W Bigdata@123 -p 37300 postgres -t PUBLIC.seq -f backup/MPPDB_backup.sql
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: The total objects number is 337.
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: [100.00%] 337 objects have been dumped.
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: dump database postgres successfully
gs_dump[port='37300'][postgres][2019-12-27 15:10:53]: total time: 230  ms
```

2. SEQUENCE已经到达最小值或最大值2^63-2

gs_dump不支持该场景下的SEQUENCE数值备份。

说明

SQL端不支持SEQUENCE到达最大值2^63-2后的MAXVALUE修改，不支持SEQUENCE到达最小值后的MINVALUE修改。

示例

使用gs_dump转储数据库为SQL文本文件或其它格式的操作，如下所示。

示例中“Bigdata@123”表示数据库用户密码；“backup/MPPDB_backup.sql”表示导出的文件，其中backup表示相对于当前目录的相对目录；“37300”表示数据库服务器端口；“postgres”表示要访问的数据库名。

说明

导出操作时，请确保该目录存在并且当前的操作系统用户对其具有读写权限。

示例1：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup.sql文件格式为纯文本格式。

```
gs_dump -U dbadmin -W Bigdata@123 -f backup/MPPDB_backup.sql -p 37300 postgres -F p
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: The total objects number is 356.
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: [100.00%] 356 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 09:49:17]: total time: 1274  ms
```

使用gsql程序从纯文本导出文件中导入数据。

示例2：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup.tar文件格式为tar格式。

```
gs_dump -U dbadmin -W Bigdata@123 -f backup/MPPDB_backup.tar -p 37300 postgres -F t
gs_dump[port='37300'][postgres][2018-06-27 10:02:24]: The total objects number is 1369.
gs_dump[port='37300'][postgres][2018-06-27 10:02:53]: [100.00%] 1369 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:02:53]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:02:53]: total time: 50086  ms
```

示例3：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup.dmp文件格式为自定义归档格式。

```
gs_dump -U dbadmin -W Bigdata@123 -f backup/MPPDB_backup.dmp -p 37300 postgres -F c
gs_dump[port='37300'][postgres][2018-06-27 10:05:40]: The total objects number is 1369.
gs_dump[port='37300'][postgres][2018-06-27 10:06:03]: [100.00%] 1369 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:06:03]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:06:03]: total time: 36620  ms
```

示例4：执行gs_dump，导出postgres数据库全量信息，导出的MPPDB_backup文件格式为目录格式。

```
gs_dump -U dbadmin -W Bigdata@123 -f backup/MPPDB_backup -p 37300  postgres -F d
gs_dump[port='37300'][postgres][2018-06-27 10:16:04]: The total objects number is 1369.
gs_dump[port='37300'][postgres][2018-06-27 10:16:23]: [100.00%] 1369 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:16:23]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:16:23]: total time: 33977  ms
```

示例5：执行gs_dump，导出postgres数据库信息，但不导出/home/MPPDB_temp.sql中指定的表信息。导出的MPPDB_backup.sql文件格式为纯文本格式。

```
gs_dump -U dbadmin -W Bigdata@123 -p 37300 postgres --exclude-table-file=/home/MPPDB_temp.sql -f
backup/MPPDB_backup.sql
gs_dump[port='37300'][postgres][2018-06-27 10:37:01]: The total objects number is 1367.
gs_dump[port='37300'][postgres][2018-06-27 10:37:22]: [100.00%] 1367 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-27 10:37:22]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-27 10:37:22]: total time: 37017  ms
```

示例6：执行gs_dump，仅导出依赖于指定表testtable的视图信息。然后创建新的testtable表，再恢复依赖其上的视图。

备份仅依赖于testtable的视图

```
gs_dump -s -p 37300 postgres -t PUBLIC.testtable --include-depend-objs --exclude-self -f backup/
MPPDB_backup.sql -F p
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: The total objects number is 331.
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: [100.00%] 331 objects have been dumped.
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: dump database postgres successfully
gs_dump[port='37300'][postgres][2018-06-15 14:12:54]: total time: 327  ms
```

修改testtable名称

```
gsql -p 37300 postgres -r -c "ALTER TABLE PUBLIC.testtable RENAME TO testtable_bak;"
```

创建新的testtable表

```
CREATE TABLE PUBLIC.testtable(a int, b int, c int);
```

还原依赖于testtable的视图

```
gsql -p 37300 postgres -r -f backup/MPPDB_backup.sql
```

相关命令

gs_dumpall

18.2 gs_dumpall

背景信息

gs_dumpall是GaussDB(DWS)用于导出所有数据库相关信息工具，它可以导出集群数据库的所有数据，包括默认数据库postgres的数据、自定义数据库的数据、以及集群所有数据库公共的全局对象。

gs_dumpall工具在进行数据导出时，其他用户可以访问集群数据库（读或写）。

gs_dumpall工具支持导出完整一致的数据。例如，T1时刻启动gs_dumpall导出整个集群数据库，那么导出数据结果将会是T1时刻该集群数据库的数据状态，T1时刻之后对集群数据库的修改不会被导出。

gs_dumpall在导出整个集群所有数据库时分为两部分：

- gs_dumpall自身对所有数据库公共的全局对象进行导出，包括有关数据库用户和组，表空间以及属性（例如，适用于数据库整体的访问权限）信息。
- gs_dumpall通过调用gs_dump来完成集群中各数据库的SQL脚本文件导出，该脚本文件包含将数据库恢复为其保存时的状态所需要的全部SQL语句。

以上两部分导出的结果为纯文本格式的SQL脚本文件，使用gsql运行该脚本文件可以恢复集群数据库。

注意事项

- 禁止修改导出的文件和内容，否则可能无法恢复成功。
- 为了保证数据一致性和完整性，gs_dumpall会对需要转储的表设置共享锁。如果某张表在别的事务中设置了共享锁，gs_dumpall会等待此表的锁释放后锁定此表。如果无法在指定时间内锁定某张表，转储会失败。用户可以通过指定--lock-wait-timeout选项，自定义等待锁超时时间。
- 由于gs_dumpall读取所有数据库中的表，因此必须以数据库集群管理员身份进行连接，才能导出完整文件。在使用gsql执行脚本文件导入时，同样需要管理员权限，以便添加用户和组，以及创建数据库。

语法

gs_dumpall [OPTION]...

参数说明

通用参数：

- -f, --filename=FILENAME

  将输出发送至指定文件。如果这里省略，则使用标准输出。

- -v, --verbose

  指定verbose模式。该选项将导致gs_dumpall向转储文件输出详细的对象注解和启动/停止次数，向标准错误流输出处理信息。

- -V, --version

  打印gs_dumpall版本，然后退出。

- --lock-wait-timeout=TIMEOUT

  请勿在转储刚开始时一直等待以获取共享表锁。如果无法在指定时间内锁定某个表，就选择失败。可以以任何符合SET statement_timeout的格式指定超时时间。

- -?, --help

  显示gs_dumpall命令行参数帮助，然后退出。

转储参数：

- -a, --data-only

  只转储数据，不转储模式（数据定义）。

- -c, --clean

  在重新创建数据库之前，执行SQL语句清理（删除）这些数据库。针对角色和表空间的转储命令已添加。

- -g, --globals-only

  只转储全局对象（角色和表空间），无数据库。

- -o, --oids

  转储每个表的对象标识符（OIDs），作为表的一部分数据。该选项用于应用以某种方式（例如：外键约束方式）参照了OID列的情况。如果不是以上这种情况，请勿使用该选项。

- -O, --no-owner

  不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_dumpall会发出ALTER OWNER或SET SESSION AUTHORIZATION语句设置所创建的模式元素的所属。如果脚本正在运行，该语句不会执行成功，除非是由系统管理员触发（或是拥有脚本中所有对象的同一个用户）。通过指定-O，编写一个任何用户都能存储的脚本，且该脚本会授予该用户拥有所有对象的权限。

- -r, --roles-only

  只转储角色，不转储数据库或表空间。

- -s, --schema-only

  只转储对象定义（模式），而非数据。

- -S, --sysadmin=NAME

  在转储过程中使用的系统管理员名称。

- -t, --tablespaces-only

  只转储表空间，不转储数据库或角色。

- -x, --no-privileges

防止转储访问权限（授权/撤销命令）。

- --column-inserts|--attribute-inserts

  以INSERT命令带列名（INSERT INTO表（列、…）值…）方式导出数据。这会导致恢复缓慢。但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。

- --disable-dollar-quoting

  该选项将禁止在函数体前使用美元符号$，并强制使用SQL标准字符串语法对其进行引用。

- --disable-triggers

  该参数为扩展预留接口，不建议使用。

- --inserts

  发出INSERT命令（而非COPY命令）时转储数据。这会导致恢复缓慢。注意如果重排列顺序，可能会导致恢复整个失败。--column-inserts选项更加安全，虽然可能更慢些。

- --no-security-labels

  该参数为扩展预留接口，不建议使用。

- --no-tablespaces

  请勿输出创建表空间的命令，也请勿针对对象选择表空间。使用该选项，无论默认表空间是哪个，在恢复过程中所有对象都会被创建。

- --no-unlogged-table-data

  该参数为扩展预留接口，不建议使用。

- --quote-all-identifiers

  强制对所有标识符加引号。为了向后续版本迁移，且其中可能涉及引入额外关键词，在转储相应数据库时该选项会有帮助。

- --dont-overwrite-file

  不重写当前文件。

- --use-set-session-authorization

  输出符合SQL标准的SET SESSION AUTHORIZATION命令而不是ALTER OWNER命令来确定对象所有权。这样令转储更加符合标准，但是如果转储文件中的对象的历史有些问题，那么可能不能正确恢复。并且，使用SET SESSION AUTHORIZATION的转储需要数据库系统管理员的权限才能转储成功，而ALTER OWNER需要的权限则低得多。

- --with-encryption=AES128

  指定转储数据需用AES128进行加密。

- --with-key=KEY

  AES128密钥长度必须是16字节。

- --include-extensions

  如果include-extensions参数被设置，将备份所有的CREATE EXTENSION语句。

- --include-templatedb

  转储过程中包含模板库。

- --dump-nodes

  转储过程中包含节点和Node Group。

- --include-nodes
  将TO NODE语句包含在已转储的CREATE TABLE命令中。
- --include-buckets
  该参数为扩展预留接口，不建议使用。
- --dump-wrm
  存储过程中包含负载资源管理器，具体包括资源池、负载组以及负载组映射。
- --binary-upgrade
  该参数为扩展预留接口，不建议使用。
- --binary-upgrade-usermap="USER1=USER2"
  该参数为扩展预留接口，不建议使用。
- --tablespaces-postfix
  该参数为扩展预留接口，不建议使用。
- --parallel-jobs
  指定备份进程并发数，取值范围为1~1000。

说明

- -g/--globals-only和-r/--roles-only不能同时使用。
- -g/--globals-only和-t/--tablespaces-only不能同时使用。
- -r/--roles-only和-t/--tablespaces-only不能同时使用。
- -s/--schema-only和-a/--data-only不能同时使用。
- -r/--roles-only和-a/--data-only不能同时使用。
- -t/--tablespaces-only和-a/--data-only不能同时使用。
- -g/--globals-only和-a/--data-only不能同时使用。
- --tablespaces-postfix和--binary-upgrade必须一起使用。
- --binary-upgrade-usermap和--binary-upgrade必须一起使用。
- --parallel-jobs和-f/--file必须一起使用。

连接参数：

- -h, --host
  指定主机的名称。如果取值是以斜线开头，它将用作Unix域套接字的目录。默认值取自PGHOST环境变量；如果没有设置，将启动某个Unix域套接字建立连接。
  该参数只针对集群外，对集群内本机只能用127.0.0.1。
  环境变量：PGHOST
- -l, --database
  指定所连接的转储全局对象的数据库名称，并去寻找还有其他哪些数据库需要被转储。如果没有指定，会使用postgres数据库，如果postgres数据库不存在，会使用template1。
- -p, --port
  指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。默认值设置为PGPORT环境变量。
  环境变量：PGPORT
- -U, --username
  所连接的用户名。

环境变量：PGUSER

- -w, --no-password

  不出现输入密码提示。如果服务器要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。

- -W, --password

  指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W选项；如果没有-W选项，并且不是系统管理员，“Dump Restore工具”会提示用户输入密码。

- --role

  指定创建转储使用的角色名。选择该选项，会使gs_dumpall连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有gs_dumpall要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以系统管理员身份登录，而使用该选项能够在不违反该规定的情况下完成转储。

- --rolepassword

  指定具体角色用户的角色密码。

说明

由于gs_dumpall内部调用**gs_dump**，所以一些诊断信息参见**gs_dump**。

一旦恢复，最好在每个数据库上运行ANALYZE，优化程序提供有用的统计数据。

gs_dumpall恢复前需要所有必要的表空间目录才能退出；否则，对于处在非默认位置的数据库，数据库创建会失败。

示例

使用gs_dumpall一次导出集群的所有数据库。

说明

gs_dumpall仅支持纯文本格式导出。所以只能使用gsql恢复gs_dumpall导出的转储内容。

```
gs_dumpall -f backup/bkp2.sql -p 37300
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:09]: The total objects number is 2371.
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:35]: [100.00%] 2371 objects have been dumped.
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:46]: dump database dbname='postgres' successfully
gs_dump[port='37300'][dbname='postgres'][2018-06-27 09:55:46]: total time: 55567  ms
gs_dumpall[port='37300'][2018-06-27 09:55:46]: dumpall operation successful
gs_dumpall[port='37300'][2018-06-27 09:55:46]: total time: 56088  ms
```

相关命令

gs_dump

18.3 gs_restore

背景信息

gs_restore是GaussDB(DWS)提供的针对gs_dump导出数据的导入工具。通过此工具可由gs_dump生成的导出文件进行导入。

主要功能包含：

- 导入到数据库

  如果连接参数中指定了数据库，则数据将被导入到指定的数据库中。其中，并行导入必须指定连接的密码。
- 导入到脚本文件

  如果未指定导入数据库，则创建包含重建数据库所必须的SQL语句脚本并写入到文件或者标准输出。等效于直接使用gs_dump导出为纯文本格式。

命令格式

```
gs_restore [OPTION]... FILE
```

说明

- FILE没有短选项或长选项。用来指定归档文件所处的位置。
- 作为前提条件，需输入dbname或-l选项。不允许用户同时输入dbname和-l选项。
- gs_restore默认是以追加的方式进行数据导入。为避免多次导入造成数据异常，在进行导入时，建议使用"-e"和"-c" 参数，即导入前删除已存在于待导入数据库中的数据库对象，同时当出现导入错误时，忽略当前错误，继续执行导入任务，并在导入后会显示相应的错误信息。

参数说明

通用参数：

- -d, --dbname=NAME

  连接数据库dbname并直接导入到该数据库中。
- -f, --file=FILENAME

  指定生成脚本的输出文件，或使用-l时列表的输出文件。

  默认是标准输出。

  说明

  -f不能同-d一起使用。
- -F, --format=c|d|t

  指定归档格式。由于gs_restore会自动决定格式，因此不需要指定格式。

  取值范围：
  - c/custom：该归档形式为4.21-gs_dump的自定义格式。
  - d/directory：该归档形式是一个目录归档形式。
  - t/tar：该归档形式是一个tar归档形式。
- -l, --list

列出归档形式内容。这一操作的输出可用作-L选项的输入。注意如果像-n或-t的过滤选项与-l使用，过滤选项将会限制列举的项目（即归档形式内容）。

- -v, --verbose

  指定verbose模式。

- -V, --version

  打印gs_restore版本，然后退出。

- -?, --help

  显示gs_restore命令行参数帮助，然后退出。

导入参数：

- -a, -data-only

  只导入数据，不导入模式（数据定义）。gs_restore的导入是以追加方式进行的。

- -c, --clean

  在重新创建数据库对象前，清理（删除）已存在于将要还原的数据库中的数据库对象

- -C, --create

  导入到数据库之前请创建数据库。（选择该选项后，以-d打头的数据库将被用作发布首个CREATE DATABASE命令。所有数据将被导入到出现在归档文件的数据库中。）

- -e, --exit-on-error

  当发送SQL语句到数据库时如果出现错误，请退出。默认状态下会继续，且在导入后会显示一系列错误信息。

- -I, --index=NAME

  只导入已列举的index的定义。允许导入多个index。如果多次输入-I index导入多个index。

  例如：

  ```
  gs_restore -h host_name -p port_number -d postgres -I Index1 -I Index2 backup/MPPDB_backup.tar
  ```

  在上面这个例子中，Index1和Index2会被导入。

- -j, --jobs=NUM

  运行gs_restore最耗时的部分（如加载数据、创建index、或创建约束）使用并发任务。该选项能大幅缩短导入时间，即将一个大型数据库导入到某一多处理器的服务器上。

  每个任务可能是一个进程或一个线程，这由操作系统决定；每个任务与服务器进行单独连接。

  该选项的最优值取决于服务器的硬件设置、客户端、以及网络。还包括这些因素，如CPU核数量、硬盘设置。建议是从增加服务器上的CPU核数量入手，更大的值（服务器上CPU核数量）在很多情况下也能导致数据文件更快的被导入。当然，过高的值会由于超负荷反而导致性能降低。

  该选项只支持自定义归档格式。输入文件必须是常规文件（不能是像pipe的文件）。如果是通过脚本文件，而非直接连接数据库服务器，该选项可忽略。而且，多任务不能与--single-transaction选项一起使用。

- -L, --use-list=FILENAME

  只导入列举在list-file中的那些归档形式元素，导入顺序以它们在文件中的顺序为准。注意如果像-n或-t的过滤选项与-L使用，它们将会进一步限制导入的项目。

一般情况下，list-file是通过编辑前面提到的某个-l参数的输出创建的。文件行的位置可更改或直接删除行，也可使用分号（;）在行的开始注出。见下文的举例。

- -n, --schema=NAME

  只导入已列举的模式中的对象。

  该选项可与-t选项一起用以导入某个指定的表。

  多次输入-n *schemaname*可以导入多个模式。

  例如：

  ```
  gs_restore -h host_name -p port_number -d postgres -n sch1 -n sch2 backup/MPPDB_backup.tar
  ```

  在上面这个例子中，sch1和sch2会被导入。

- -O, --no-owner

  不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_restore会发出ALTER OWNER或SET SESSION AUTHORIZATION语句设置所创建的模式元素的所属。除非是由系统管理员（或是拥有脚本中所有对象的同一个用户）进行数据库首次连接的操作，否则语句会失败。使用-O选项，任何用户名都可用于首次连接，且该用户拥有所有已创建的对象。

- -P, --function=NAME(args)

  只导入已列举的函数。请按照函数所在转储文件中的目录，准确拼写函数名称和参数。

  当-P单独使用时，表示导入文件中所有'function-name(args)'函数；当-P同-n一起使用时，表示导入指定模式下的'function-name(args)'函数；多次输入-P，而仅指定一次-n，表示所有导入的函数默认都是位于-n模式下的。

  可以多次输入-n schema-name -P 'function-name(args)'同时导入多个指定模式下的函数。

  例如：

  ```
  ./gs_restore -h host_name -p port_number -d postgres -n test1 -P 'Func1(integer)' -n test2 -P
  'Func2(integer)' backup/MPPDB_backup.tar
  ```

  在上面这个例子中，test1模式下的函数Func1(i integer)和test2模式下的函数Func2(j integer)会被一起导入。

- -s, --schema-only

  只导入模式（数据定义），不导入数据（表内容）。当前的序列值也不会导入。

- -S, --sysadmin=NAME

  该参数为扩展预留接口，不建议使用。

- -t, --table=NAME

  只导入已列举的表定义、数据或定义和数据。该选项与-n选项同时使用时，用来指定某个模式下的表对象。-n参数不输入时，默认为PUBLIC模式。多次输入-n <schemaname> -t <tablename>可以导入指定模式下的多个表。

  例如：

  导入PUBLIC模式下的table1

  ```
  gs_restore -h host_name -p port_number -d postgres -t table1 backup/MPPDB_backup.tar
  ```

  导入test1模式下的test1和test2模式下test2

  ```
  gs_restore -h host_name -p port_number -d postgres -n test1 -t test1 -n test2 -t test2 backup/
  MPPDB_backup.tar
  ```

  导入PUBLIC模式下的table1和test1 模式下test1

  ```
  gs_restore -h host_name -p port_number -d postgres -n PUBLIC -t table1 -n test1 -t table1 backup/
  MPPDB_backup.tar
  ```

须知

-t不支持schema_name.table_name的输入格式。

- -T, --trigger=NAME

  该参数为扩展预留接口。

- -x, --no-privileges/--no-acl

  防止导入访问权限（grant/revoke命令）。

- -1, --single-transaction

  执行导入作为一个单独事务（即把命令包围在BEGIN/COMMIT中）。

  该选项确保要么所有命令成功完成，要么没有改变应用。该选项意为--exit-on-error。

- --disable-triggers

  该参数为扩展预留接口，不建议使用。

- --no-data-for-failed-tables

  默认状态下，即使创建表的命令失败（如表已经存在），表数据仍会被导入。使用该选项，像这种表的数据会被跳过。如果目标数据库已包含想要的表内容，这种行为会有帮助。

  该选项只有在直接导入到某数据库中时有效，不针对生成SQL脚本文件输出。

- --no-security-labels

  该参数为扩展预留接口，不建议使用。

- --no-tablespaces

  不输出选择表空间的命令。使用该选项，无论默认表空间是哪个，在导入过程中所有对象都会被创建。

- --section=SECTION

  导入已列举的区段（如pre-data、data、或post-data）。

- --use-set-session-authorization

  该选项用来进行文本格式的备份。

  输出SET SESSION AUTHORIZATION命令，而非ALTER OWNER命令，用以决定对象归属。该选项使转储更加兼容标准，但通过参考转储中对象的记录，导入过程可能会有问题。使用SET SESSION AUTHORIZATION的转储要求必须是系统管理员，同时在导入前还需参考"SET SESSION AUTHORIZATION"，手工对导出文件的密码进行修改验证，只有这样才能进行正确的导入操作，相比之下，ALTER OWNER对权限要求较低。

- --with-key=KEY

  AES128密钥长度必须是16字节。

  说明

  如果转储被加密，则必须在gs_restore命令中输入--with-key <keyname>选项。如果未输入，用户会收到错误信息。

  应该输入转储时所输入的相同的key。

须知

- 如果安装过程中有任何本地数据要添加到template1数据库，请谨慎将gs_restore的输出载入到一个真正的空数据库中；否则可能会因为被添加对象的定义被复制，而出现错误。要创建一个无本地添加的空数据库，需从template0而非template1复制，例如：

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

- gs_restore不能选择性地导入大对象；例如只能导入那些指定表的对象。如果某个归档形式包含大对象，那所有大对象都会被导入，或一个都不会被导入，如果它们通过-L、-t或其他选项被排除。

说明

1.-d/--dbname 和 -f/--file 不能同时使用；

2. -s/--schema-only 和 -a/--data-only不能同时使用；

3. -c/--clean 和 -a/--data-only不能同时使用；

4. 使用--single-transaction时，-j/--jobs必须为单任务；

5. --role 和 --rolepassword必须一起使用。

连接参数：

- -h, --host=HOSTNAME

  指定的主机名称。如果取值是以斜线开头，他将用作Unix域套接字的目录。默认值取自PGHOST环境变量；如果没有设置，将启动某个Unix域套接字建立连接。

  该参数只针对集群外，对集群内本机只能用127.0.0.1。

- -p, --port=PORT

  指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。默认值设置为PGPORT环境变量。

- -U, --username=NAME

  所连接的用户名。

- -w, --no-password

  不出现输入密码提示。如果服务器要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。

- -W, --password=PASSWORD

  指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W参数；如果没有-W参数，并且不是系统管理员，“gs_restore”会提示用户输入密码。

- --role=ROLENAME

  指定导入操作使用的角色名。选择该参数，会使gs_restore连接数据库后，发起一个SET ROLE角色名命令。当所授权用户（由-U指定）没有gs_restore要求的权限时，该参数会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以初始用户身份登录，而使用该参数能够在不违反该规定的情况下完成导入。

- --rolepassword=ROLEPASSWORD

  指定具体角色用户的角色密码。

示例

特例：执行gsql程序，使用如下选项导入由gs_dump/gs_dumpall生成导出文件夹（纯文本格式）的MPPDB_backup.sql文件到postgres数据库。

```
gsql -d postgres -p 8000 -W Bigdata@123 -f /home/omm/test/MPPDB_backup.sql
SET
SET
SET
SET
SET
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
CREATE INDEX
CREATE INDEX
CREATE INDEX
SET
CREATE INDEX
REVOKE
REVOKE
GRANT
GRANT
total time: 30476  ms
```

gs_restore用来导入由gs_dump生成的导出文件。

示例1：执行gs_restore，将导出的MPPDB_backup.dmp文件（自定义归档格式）导入到postgres数据库。

```
gs_restore -W Bigdata@123 backup/MPPDB_backup.dmp -p 8000 -d postgres
gs_restore: restore operation successful
gs_restore: total time: 13053  ms
```

示例2：执行gs_restore，将导出的MPPDB_backup.tar文件（tar格式）导入到postgres数据库。

```
gs_restore backup/MPPDB_backup.tar -p 8000 -d postgres
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21203  ms
```

示例3：执行gs_restore，将导出的MPPDB_backup文件（目录格式）导入到postgres数据库。

```
gs_restore backup/MPPDB_backup -p 8000 -d postgres
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21003  ms
```

示例4：执行gs_restore，使用自定义归档格式的MPPDB_backup.dmp文件来进行如下导入操作。 导入PUBLIC模式下所有对象的定义和数据。在导入时会先删除已经存在的对象，如果原对象存在跨模式的依赖则需手工强制干预。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d postgres -e -c -n PUBLIC
gs_restore: [archiver (db)] Error while PROCESSING TOC:
gs_restore: [archiver (db)] Error from TOC entry 313; 1259 337399 TABLE table1 gaussdba
gs_restore: [archiver (db)] could not execute query: ERROR:  cannot drop table table1 because other objects depend on it
DETAIL:  view t1.v1 depends on table table1
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
   Command was: DROP TABLE public.table1;
```

手工删除依赖，导入完成后再重新创建。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d postgres -e -c -n PUBLIC
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 2203  ms
```

示例5：执行gs_restore，使用自定义归档格式的MPPDB_backup.dmp文件来进行如下导入操作。只导入PUBLIC模式下表table1的定义。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d postgres -e -c -s -n PUBLIC -t table1
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21000  ms
```

示例6：执行gs_restore，使用自定义归档格式的MPPDB_backup.dmp文件来进行如下导入操作。只导入PUBLIC模式下表table1的数据。

```
gs_restore backup/MPPDB_backup.dmp -p 8000 -d postgres -e -a -n PUBLIC -t table1
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 20203  ms
```

相关命令

gs_dump，**gs_dumpall**

19 术语表

术语	解释
A – E	
ACID	在可靠数据库管理系统（DBMS）中，事务（transaction）所应该具有的四个特性：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、和持久性（Durability）。
安全环	每一个环都由若干物理机组成，环内的DN形成主、备、从备关系，不向环外延伸。也就是说，环内的任何一个节点的主，或者备，或从备，依然在环内。扩容与缩容时，是以环为最小单位进行的。
Bgwriter	数据库启动时创建的一个后台写线程，此线程用于将数据库中脏页面写入到持久性设备（例如磁盘）中。
bit	比特。计算机处理的最小的信息单位。比特用来表示二进制数字1或0，或者一种逻辑条件真或假。在物理上，比特表示一个电路上高或低的电压点或者磁盘上的磁化单程或其它。一个单独的比特位所传达的信息很少有意义的。然而，一个8位组却构成了一个字节，可用于表示如一个英文字母，十进制数字，或其它字符等多种类型的信息。
Bloom Filter	布隆过滤器。由Howard Bloom在1970年提出的二进制向量数据结构，它具有很好的空间和时间效率，被用来检测一个元素是不是集合中的一个成员，这种检测只会对在集合内的数据错判，而不会对不是集合内的数据进行错判，这样每个检测请求返回有“在集合内（可能错误）”和“不在集合内（绝对不在集合内）”两种情况，可见Bloom filter是牺牲了正确率换取时间和空间。
CCN	Control Coordinator Node，GaussDB(DWS)动态负载管理中心控制点。负责进行各CN中复杂作业是否可以执行的中心判断、排队和调度，以实现动态负载管理。

术语	解释
CIDR	Classless Inter-Domain Routing，无类域间路由IP编址方案。CIDR摒弃传统的基于类（A类：8，B类：16，C类：24）的地址分配方式，允许使用任意长度的地址前缀，有效提高地址空间的利用率。CIDR表示方法：IP地址/网络ID的位数。比如192.168.23.35/21，其中“21”表示前面地址中的前21位代表网络部分，其余位代表主机部分。
Cgroups	Control Groups，控制组（GaussDB(DWS)中也称之为优先级组）。SUSE Linux和RedHat内核提供的一种可以限制、记录、隔离进程组所使用的物理资源的机制。
CLI	Command-line Interface，命令行界面。应用程序和用户交互的一种方式，完全基于文本输入和输出。命令通过键盘或类似装置输入，由程序编译并执行。结果是以文本或图形的方式呈现在终端界面。
CM	Cluster Manager，集群管理模块。管理和监控分布式系统中各个功能单元和物理资源的运行情况，确保整个系统的稳定运行。
CMS	Cluster Management Service，集群管理服务。是用于管理集群状态的部件。
CN	Coordinator Node，负责数据库系统元数据存储、查询任务的分解和部分执行，以及将DN中查询结果汇聚在一起。
CU	Compression Unit，压缩单元。列存表的最小存储单位。
core文件	当程序出现内存越界、断言失败或者访问非法内存时，操作系统会中止进程，并将当前内存状态导出到core文件中，以便进一步分析。 core文件包含内存转储，支持全二进制和指定端口格式。core文件名称由字符串core以及操作系统进程ID组成。 core文件不依赖于任何平台。
Core Dump	通常在程序异常终止时，核心转储（Core Dump）、内存转储或系统转储用于记录特定时间计算机程序工作内存的状态。实际上，其它关键程序的状态经常在同一时间进行转储，例如处理器寄存器，包括程序指标和栈指针、内存管理信息、其它处理器和操作系统标记及信息。Core Dump经常用于辅助诊断和纠错计算机程序问题。
DBA	Database Administrator，数据库管理员。指导或执行所有和维护数据库环境相关的操作。
DBLINK	DBLINK是定义一个数据库到另一个数据库路径的对象，通过它可以查询远程数据库对象。
DBMS	Database Management System，数据库管理系统。数据库管理系统是为了访问数据库中的信息而使用的一个管理系统软件。它包含一组程序使用户可以进入、管理、查询数据库中数据。基于真实数据的位置，可以分为内存数据库管理系统和磁盘数据库管理系统。
DCL	Data Control Language，数据控制语言。
DDL	Data Definition Language，数据定义语言。

术语	解释
DML	Data Manipulation Language，数据操纵语言。
DN	Data Node，和CN对应的概念。负责实际执行表数据的存储、查询操作。
ETCD	Editable Text Configuration Daemon，分布式键值存储系统，用于共享配置和服务发现（服务注册和查找）。
ETL	Extract-Transform-Load，描述将数据从来源端经过抽取（extract）、转换（transform）、加载（load）至目的端的过程。
备份	备份件或者备份过程。指复制并归档计算机数据，当发生数据丢失事件时，可以用该复制并归档的数据来恢复原始数据。
备份和恢复	保护数据库防止由于媒介失效或人为错误造成的数据丢失过程中涉及的一组概念、过程及策略。
备机	GaussDB(DWS)双机方案中的一个节点，用于作为主机的备份，在主机异常时，备机会切换到主机状态，以确保能正常提供数据服务。
崩溃	崩溃（或系统崩溃）指计算机或程序（例如软件应用程序或操作系统）异常终止的事件。出现错误后，通常会自动退出。有时出现恶意程序冻结或挂起直到崩溃上报服务记录崩溃的详细信息。对于操作系统内核关键部分的程序，整个计算机可能瘫痪（可能造成致命的系统错误）。
编码	编码是指用代码来表示各组数据资料，使其成为可利用计算机进行处理和分析的信息。用预先规定的方法将文字、数字或其它对象编成数码，或将信息、数据转换成规定的电脉冲信号。
编码技术	呈现计算机软硬件识别的特定字符集数据的技术。
表	表是由行与列组合成的。每一列被当作是一个字段。每个字段中的值代表一种类型的数据。例如，一个表可能有3个字段：姓名、城市和国家。这个表就会有3列：一列代表姓名，一列代表城市，一列代表国家。表中的每一行包含3个字段的内容，姓名字段包含姓名，城市字段包含城市，国家字段包含国家。
表空间	包含表、索引、大对象、长数据等数据的逻辑存储结构。表空间在物理数据和逻辑数据间提供了抽象的一层，为所有的数据库对象分配存储空间。表空间创建好后，创建数据库对象时可以指定该对象所属的表空间。
并发控制	在多用户环境下同时执行多个事务并保证数据完整性的一个DBMS服务。并发控制是GaussDB(DWS)提供的一种多线程管理机制，用来保证多线程环境下在数据库中执行的操作是安全的和一致的。
查询	向数据库发出的信息请求，包含更新、修改、查询或删除信息的请求。

术语	解释
查询操作符	Query Operator，也称为查询迭代算子（Iterator）或查询节点（Query Tree Node）。一个查询的执行可以分解为一个或多个查询操作符，是构成一个查询执行的最基本单位。常见的查询操作符包括表扫描（Scan），表关联（Join），表聚集（Aggregation）等。
查询片段	每一个查询任务都可以分解成为一个或者多个查询片段。每个查询片段由一个或多个查询操作符构成，可独立在节点上运行。通过数据流操作符与其它查询片段块交换数据。
持久性	数据库事务的ACID特性之一。在事务完成以后，该事务对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。
存储过程	存储过程（StoredProcedure）是在大型数据库系统中，一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名称并设置参数（如果该存储过程带有参数）来执行它。
操作系统	操作系统OS（operating system）由引导程序加载到计算中，对计算机中其它程序进行管理。其它程序叫做应用或应用程序。
从备	Secondary，为了保证集群的高可靠性，主、备间无法正常同步数据时，主节点会将日志同步到从备。如果主节点突然故障不可用，备节点会升主，并且升主成功后从从备节点上同步之前异常期间的日志。
大对象	大对象（Blob）在数据库中指使用二进制方式存储的数据。它通常可以用于存储视频、音频和图像等多媒体数据。
段	数据库中，一段指包含一个或多个区域的数据库中的一部分。区域是数据库的最小范围，由单元调用块组成。一个或多个段组成一个表空间。
F – J	
Failover	指当某个节点出现故障时，自动切换到备节点上的过程。反之，从备节点上切换回来的过程称为Failback。
FDW	Foreign Data Wrapper，外部数据封装器。是Postgres提供的一个SQL接口，用于访问远程数据存储中的大数据对象，使DBA可以整合来自不相关数据源的数据，将它们存入数据库中的一个公共模型。
Freeze	在事务ID耗尽时由AutoVacuum Worker进程自动执行的操作。GaussDB(DWS)会把事务ID记在行头，在一个事务取得一行时，通过比较行头的事务ID和事务本身的ID判断这行是否可见，而事务ID是一个无符号整数，如果事务ID耗尽，事务ID会跨过整数的界限重新计算，此时原先可见的行就会变成不可见的行，为了避免这个问题，Freeze操作会将行头的事务标记为一个特殊的事务ID，标记了这个特殊的事务ID的行将对所有事务可见，以此避免事务ID耗尽产生的问题。

术语	解释
GDB	GNU工程调试器，可以监控其它程序运行时的内部情况，或者其它程序要崩溃时发生了什么。GDB支持如下四种主要操作（使PDK功能更加强大），辅助查找缺陷。 ● 启动程序，指定可能影响行为的任何因素。 ● 特定条件下，停止程序。 ● 程序停止时，检查发生了什么。 ● 修改程序内容，尝试纠正一个缺陷并继续下一个。
GDS	General Data Service，数据并行加载工具。向GaussDB(DWS)导入数据时，需要将此工具部署到源数据所在的服务器上，使DN可以通过该工具获取数据。
GNU	GNU计划，又称革奴计划，是由RichardStallman在1983年9月27日公开发起的。它的目标是创建一套完全自由的操作系统。GNU是“GNU's NotUnix”的递归缩写。Stallman宣布GNU应当发音为Guh-NOO以避免与new这个单词混淆（注：Gnu在英文中原意为非洲牛羚，发音与new相同）。Unix是一种广泛使用的商业操作系统的名称。技术上讲，GNU类似Unix。但是GNU却给了用户自由。
gsql	GaussDB(DWS)交互终端。通过gsql能够以交互的方式输入查询，下发查询到GaussDB(DWS)，然后查看查询结果。或者，也可以从文件中输入。此外，gsql还提供许多元命令和各种类似shell命令，协助脚本编写及自动化各种任务。
GTM	Global Transaction Manager，全局事务管理器。用于管理事务状态的部件。
GUC	Grand Unified Configuration，数据库运行参数。配置这些参数可以影响数据库系统的行为。
HA	高可用性（HighAvailability），通过尽量缩短因日常维护操作（计划）和突发的系统崩溃（非计划）所导致的停机时间，以提高系统和应用的可用性。
HBA	host-based authentication，主机认证。主机鉴权允许主机鉴权部分或全部系统用户。适用于系统所有用户或者使用Match指令的子集。该类型鉴权对于管理计算集群以及其它完全同质设备非常有用。总之，服务器上的三个文件以及客户端上的一个文件必须修改，为主机鉴权做准备。
HDFS	Hadoop Distributed File System，Apache Hadoop项目的一个子项目。一个高度容错的分布式文件系统，设计用于在低成本硬件上运行。HDFS提供高吞吐量应用程序数据访问功能，适合带有大型数据集的应用程序。
服务器	为客户端提供服务的软硬件的组合。单独使用时，指运行服务器操作系统的计算机，也可以指提供服务的软件或者专用硬件。
高级包	GaussDB(DWS)提供的具有一定逻辑和功能的存储过程、函数，这些具备功能的存储过程、函数统称为高级包。

术语	解释
隔离性	数据库事务的ACID特性之一。它是指一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务之间不能互相干扰。
关系型数据库	创建在关系模型基础上的数据库。关系型数据库借助于集合代数等数学概念和方法来处理数据库中的数据。
归档线程	数据库打开归档功能时启动的一个线程，此线程用于将数据库日志归档到指定的路径。
故障接管	功能对等的系统部件对于故障部件的自动替换过程。系统部件包含处理器、服务器、网络、数据库等。
环境变量	定义进程操作环境某一方面的变量。例如，环境变量可以为主目录，命令搜索路径，使用终端或当前时区。
检查点	将数据库内存中某一时刻的数据存到磁盘的机制。GaussDB(DWS)定期将已提交的事务数据和未提交的事务数据存到磁盘，这些数据用来和Redo日志一起在数据库重启和崩溃时恢复数据库。
加密	用于传输数据的功能。通过该功能，可以隐藏信息内容，防止非法使用。
节点	将构成GaussDB(DWS)集群环境的各台服务器（物理机或虚拟机）称为集群节点，简称节点。
纠错	系统自动识别软件和数据流上的错误并自动修正错误的能力，提升系统的稳定性和可靠性。
进程	在单个计算机上执行程序的实例。一个进程由一个或多个线程组成。其它进程不能接入某个进程已占用的线程。
基于时间点恢复	PITR（Point-In-TIme Recovery），基于时间点恢复是GaussDB(DWS)备份恢复的一个特性，是指在备份数据和WAL日志正常的情况下，数据可以恢复到指定时间点。
记录	在关系型数据库中，每一条记录对应表中的每一行数据。
集群	集群是由一组服务器和其它资源组成的一个单独的系统，可以实现高可用性。有的情况下，可以实现负载均衡及并行处理。
K – O	
LLVM	LLVM命名最早源自于底层虚拟机（Low Level Virtual Machine）的缩写。LLVM是构架编译器（compiler）的框架系统，以C++编写而成，用于优化以任意程序语言编写的程序的编译时间（compile-time）、链接时间（link-time）、运行时间（run-time）以及空闲时间（idle-time），对开发者保持开放，并兼容已有脚本。
LVS	Linux Virtual Server，虚拟服务器集群系统，用于负责集群的负载均衡。
MPP	Massive Parallel Processing，大规模并行处理。指利用多个机器构成集群的架构方式，也称为集群（Cluster）系统。

术语	解释
MVCC	Multi-Version Concurrency Control，多版本并发控制。数据库并发控制协议的一种，它的基本算法是一个元组可以有多个版本，不同的查询可以工作在不同的版本上。一个基本的好处是读和写可以不冲突。
NameNode	NameNode是Hadoop系统中的一个中心服务器，负责管理文件系统的名字空间(namespace)以及客户端对文件的访问。
OBS	Object Storage Service，对象存储服务。
OLAP	Online Analytical Processing，联机分析处理。是数据仓库系统最主要的应用，专门设计用于支持复杂的分析操作，侧重对决策人员和高层管理人员的决策支持，可以根据分析人员的要求快速、灵活地进行大数据量的复杂查询处理，并且以一种直观而易懂的形式将查询结果提供给决策人员，以便准确掌握企业（公司）的经营状况，了解对象的需求，制定正确的方案。
OM	Operations Management，运维管理模块。提供集群日常运维、配置管理的管理接口、工具。
ORC	Optimized Row Columnar，一种广泛使用的Hadoop系统结构化数据的文件格式，由Hadoop HIVE项目引入。
客户端	连接或请求其它计算机或程序服务的计算机或程序。
空闲空间管理	管理表内空闲空间的机制，通过记录每个表内空闲空间信息，并建立易于查找的数据结构，可以加速对空闲空间进行的操作（例如INSERT）。
垃圾元组	是指使用DELETE和UPDATE语句删除的元组，GaussDB(DWS)在删除元组时只是打个删除标记，由Vacuum线程清理这种垃圾元组。
列	字段的等效概念。在数据库中，表由一列或多列组成。
逻辑节点	一个物理节点上可以安装多个逻辑节点。一个逻辑节点是一个数据库实例。
模式	数据库对象集，包括逻辑结构，例如表、视图、序、存储过程、同义名、索引、集群及数据库链接。
模式文件	用于决定数据库结构的SQL文件。
P – T	
Page	GaussDB(DWS)数据库关系对象结构中行存的最小存储单元。一个Page大小为默认为8KB。
PostgreSQL	PostgreSQL是一个开源的关系数据库管理系统(DBMS)，由全球志愿者团队开发。PostgreSQL不受任何公司或个体所控制，源代码免费使用。
Postgres-XC	一款多节点同步，读写可扩展的PostgreSQL集群数据库。

术语	解释
Postmaster	数据库服务启动时启动的一个线程。用于监听来自集群其它节点或客户端的连接请求。 主机上监听到备机连接请求，并接受后，就会创建一个WAL Sender线程，用于处理与备机的交互。
RHEL	Red Hat Enterprise Linux，红帽企业Linux。
REDO日志	记录对数据库进行操作的日志，这些日志包含重新执行这些操作所需要的信息。当数据库故障时，可以利用REDO日志将数据库恢复到故障前的状态。
SCTP	Stream Control Transmission Protocol，流控制传输协议。是IETF于2000年新定义的一个传输层协议。是提供基于不可靠传输业务的协议之上的可靠的数据报传输协议。SCTP的设计用于通过IP网传输SCN窄带信令消息。
Session	数据库系统在接收到应用程序的连接请求时，为该连接创建的一个任务。它被Session Manager管理，完成一些初始化任务，执行用户的所有操作。
Shared-nothing architecture	无共享架构是一种分布式计算架构，这种架构中不存在集中共享CPU、存储的状态，这种架构具有非常强的扩展性。
SLES	SUSE Linux Enterprise Server，由SUSE提供的企业级Linux操作系统。
SMP	Symmetric Multi-Processing，对称多处理技术，是指在一台计算机上汇集了一组处理器（多CPU），各CPU之间共享内存子系统以及总线结构。操作系统必须支持多任务和多线程处理，以使得SMP系统发挥高效的性能。数据库领域的SMP并行技术，一般指利用多线程技术实现查询的并行执行，以充分利用CPU资源，从而提升查询性能。
SQL	Structure Query Language，结构化查询语言。数据库的标准查询语言。它可以分为数据定义语言（DDL），数据操纵语言（DML）和数据控制语言（DCL）。
SSL	Secure Socket Layer，安全套接层。SSL是Netscape公司率先采用的网络安全协议。它是在传输通信协议（TCP/IP）上实现的一种安全协议，采用公开密钥技术。SSL广泛支持各种类型的网络，同时提供三种基本的安全服务，它们都使用公开密钥技术。SSL支持服务通过网络进行通信而不损害安全性。它在客户端和服务器之间创建一个安全连接。然后通过该连接安全地发送任意数据量。
收敛比	交换机下行带宽与上行带宽的比值。收敛比越高，流量收敛程度越大，丢包越严重。
trace	一种特殊的日志记录方法，用来记录程序执行的信息。程序员使用该信息进行纠错。另外，根据trace日志中信息的类型和内容，有经验的系统管理员或技术支持人员以及软件监控工具诊断软件常见问题。
全备份	备份整个数据库集群。

术语	解释
全量同步	GaussDB(DWS)双机方案中的一种数据同步机制，是指把主机中的所有数据同步给备机。
日志文件	计算机记录自身活动的记录。
事务	数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成，事务必须满足ACID原则。
数据	事实或指令的一种表达形式，适用于人为或自动的通信、解释或处理。数据包含常量、变量、阵列和字符串。
数据重分布	表数据在分布式环境中的分布方式（Distribution），即数据表以何种方式打散存储到各个数据库实例上去。具体的分布方式可以有：散列（Hash）方式，复制方式（Replication）和随机方式（Random）。散列方式根据元组中指定字段的取值算得哈希值，根据节点与哈希值的映射关系获得该元组的目标存储位置。复制方式将元组复制到所有节点上。随机方式将数据随机分布到各节点。
数据分布	表数据在分布式环境中的分布方式（Distribution），即数据表以何种方式打散存储到各个数据库实例上去。具体的分布方式可以有：散列（Hash）方式，复制方式（Replication）和随机方式（Random）。散列方式根据元组中指定字段的取值算得哈希值，根据节点与哈希值的映射关系获得该元组的目标存储位置。复制方式将元组复制到所有节点上。随机方式将数据随机分布到各节点。
数据分区	数据分区是指在一个数据库实例内部，将表按照划分为多个数据互不重叠的部分（Partition）。具体的分区方式可以有：范围分区（Range），它根据元组中指定字段的取值所处的范围映射到目标存储位置。
数据库	数据库是存储在一起的相关数据的集合，这些数据可以被访问，管理以及更新。同一视图中，数据库可以根据存储内容类型分为以下几类：数目类、全文本类、数字类及图像类。
数据库实例	一个数据库实例是一个GaussDB(DWS)进程以及它控制的数据库文件。GaussDB(DWS)在一个物理节点上安装多个数据库实例，集群各节点上所安装的GTM、CM、CN、DN统称为实例。一个数据库实例也被称为一个逻辑节点。
数据库双机	GaussDB(DWS)提供的高可靠性双机方案。在此方案中，每个GaussDB(DWS)逻辑节点标识为主机或备机。在同一时间内，只有一个GaussDB(DWS)被标识为主机。双机初次建立时，主机会对每个备机数据做全量同步，然后做增量同步。双机建立之后的运行过程中，主机能接受数据读和写的操作请求，备机只做日志同步。
数据库文件	保存用户数据和数据库系统内部数据的二进制文件。
数据流操作符	负责查询片段间交换数据的操作符。根据数据流的输入、输出关系，可以细分为聚合流（Gather）、广播流（Broadcast）和重分布流（Redistribution）。聚合流将数据从多个查询片段聚合到一个。广播流将数据从一个查询片段向多个传输。重分布流则将多个查询片段的数据，按照一定规则重组后向多个传输。

术语	解释
数据字典	数据字典是一系列只读的表，用来提供数据库的信息。这些信息包括：数据库设计信息、存储过程信息、用户权限、用户统计数据、数据库进程信息、数据库增长统计数据和数据库性能统计数据。
死锁	为使用同一资源而产生的无法解决的争用状态。
索引	数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。
统计信息	数据库使用统计信息估算查询代价，以查找代价最小的执行计划，统计信息一般是数据库自动收集的，包括表级信息（元组数、页面数等）和列级信息（列的值域分布直方图）。
投影(Projection)	投影是单目运算。该运算从表中选出**指定的属性值**组成一个新表，记为：ΠA(R)。 其中A是**属性名（即列名）**表，R是表名。
停用词	在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些字或词，这些字或词即被称为Stop Words（停用词）。
U – Z	
Vacuum	数据库定期启动的清理垃圾元组的线程，根据配置参数可以同时启动多个。
verbose	verbose选项指定显示在屏幕上的处理信息。
WAL	Write-Ahead Logging，预写日志系统。实现事务日志的标准方法，是指对数据文件（表和索引的载体）持久化修改之前必须先持久化相应的日志。
WAL Receiver	数据库复制时备机创建的一个线程的名称。此线程用于从主机接收数据、命令，并反馈确认信息至主机。一个备机只有一个WAL Receiver线程。
WAL Sender	数据库复制过程中，主机接受到备机的连接请求后创建的一个线程的名称。此线程用于发送命令、数据到备机，并从备机接收信息。一个主机可能会有多个WAL Sender线程，每一个WAL Sender线程对应一个备机的一个连接请求。
WAL Writer	数据库启动时创建的一个写Redo日志的线程，用于将内存中的日志写入到持久性设备（如：磁盘）。
WLM	WorkLoad Manager，负载管理。GaussDB(DWS)中系统资源控制和分配的模块。
Xlog	表示事务日志，一个逻辑节点中只有一个，不允许创建多个Xlog文件。
xDR	详单。用户面和信令面详单的统称，包括CDR和UFDR、TDR和SDR。
网络备份	网络备份为各种平台提供一套完整的、灵活的数据保护方案。平台包含Microsoft Windows、UNIX及Linux。网络备份支持备份、归档、恢复计算机上的文件、文件夹或目录、卷或分区。

术语	解释
物理节点	一个物理机器称为一个物理节点。
系统表	存储数据库元信息的表，元信息包括数据库中的用户表、索引、列、函数和数据类型等。
选择(Selection)	选择是单目运算，其运算对象是一个表。该运算按给定的条件，从表中选出**满足条件的行**形成一个新表作为运算结果。 选择运算的记号为 σF(R)。 其中σ是选择运算符，下标F是一个条件表达式，R是被操作的表。
压缩	数据压缩，信源编码，或比特率降低涉及使用相比原来较少比特的编码信息。压缩可以是有损或无损。无损压缩通过识别和消除统计冗余降低比特位。无损压缩中没有信息丢失。有损压缩识别并删除次要信息，减少了比特位。减少数据文件大小的方法被普遍称为数据压缩，尽管其正式名称为源编码（数据源的编码，然后将其存储或传输）。
一致性	数据库事务的ACID特性之一。在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。
元数据	用来定义数据的数据。主要是描述数据自身信息，包含源、大小、格式或其它数据特征。数据库字段中，元数据用于理解以及诠释数据仓库的内容。
原子性	数据库事务的ACID特性之一。整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚到事务开始前的状态，就像这个事务从来没有执行过一样。
脏页面	已经被修改且未写入持久性设备的页面。
增量备份	基于上次有效备份之后对文件修改的备份。
增量同步	GaussDB(DWS)双机方案中的一种数据同步机制，是指把主机中数据增量同步给备机，即只同步主备间有差异的数据。
主机	GaussDB(DWS)数据库双机系统中接受数据读写操作的节点，和所有备机一起协同工作。在同一时间内，双机系统中只有一个节点被标识为主机。
主题词	在标引和检索中用以表达文献主题的规范化的词或词组。
转储文件	转储文件是一种特定类型的trace文件。转储文件为响应事件过程中一次性输出的诊断数据，trace文件指诊断数据的连续输出。
资源池	资源池是GaussDB(DWS)提供的一种资源划分的配置机制。通过将用户绑定到资源池，来限定其所执行作业的优先级及能够利用到的资源。
最小恢复点	最小恢复点是GaussDB(DWS)提供的数据一致性保障手段之一。最小恢复点特性可以在GaussDB(DWS)启动时检查出WAL日志和持久化到磁盘的数据的不一致性，并提示用户进行处理。