云数据库 GaussDB
V2.0-8.202
分布式版特性指南

文档版本	01
发布日期	2024-11-05

华为云计算技术有限公司

版权所有 © 华为云计算技术有限公司 2024。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

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云计算技术有限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，华为云计算技术有限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。

目 录

1 修订记录 1
2 物化视图 2
2.1 全量物化视图 2
2.1.1 概述 2
2.1.2 支持和约束 2
2.1.3 使用 3
2.2 增量物化视图 4
2.2.1 概述 4
2.2.2 支持和约束 4
2.2.3 使用 5
3 设置密态等值查询 7
3.1 密态等值查询概述 7
3.2 使用 gsql 操作密态数据库 9
3.3 使用 JDBC 操作密态数据库 10
3.4 前向兼容与安全增强 15
3.5 密态支持函数/存储过程 18
3.6 使用 third_kms 三方厂商加解密库 20
4 透明数据加密 26
5 设置账本数据库 35
5.1 账本数据库概述 35
5.2 查看账本历史操作记录 37
5.3 校验账本数据一致性 39
5.4 归档账本数据库 40
5.5 修复账本数据库 41
6 基于标签的强制访问控制 44
6.1 定义安全标签 44
6.2 应用安全标签 45
6.3 基于标签的强制访问控制检查 46
7 逻辑复制 47
7.1 逻辑解码 47
7.1.1 逻辑解码概述 47

 版权所有 © 华为云计算技术有限公司

7.1.2 逻辑解码选项 53
7.1.3 使用 SQL 函数接口进行逻辑解码 61
7.1.4 使用流式解码实现数据逻辑复制 62
7.1.5 逻辑解码支持 DDL 62
7.1.6 逻辑解码数据找回功能 70
8 分区表 72
8.1 大容量数据库 72
8.1.1 大容量数据库背景介绍 72
8.1.2 表分区技术 72
8.1.3 数据分区查找优化 73
8.1.4 数据分区运维管理 73
8.2 分区表介绍 74
8.2.1 基本概念 74
8.2.1.1 分区表（母表） 74
8.2.1.2 分区（分区子表、子分区） 75
8.2.1.3 分区键 76
8.2.2 分区策略 76
8.2.2.1 范围分区 76
8.2.2.2 哈希分区 78
8.2.2.3 列表分区 79
8.2.2.4 分区表对导入操作的性能影响 79
8.2.3 分区基本使用 80
8.2.3.1 创建分区表 81
8.2.3.2 使用和管理分区表 82
8.2.3.3 分区表 DQL/DML 83
8.3 分区表查询优化 84
8.3.1 分区剪枝 84
8.3.1.1 分区表静态剪枝 85
8.3.1.2 分区表动态剪枝 89
8.3.1.2.1 PBE 动态剪枝 89
8.3.1.2.2 参数化路径动态剪枝 92
8.3.2 分区索引 96
8.3.3 分区表统计信息 99
8.3.3.1 级联收集统计信息 99
8.3.3.2 分区级统计信息 102
8.3.4 Partition-wise Join 106
8.3.4.1 非 SMP 场景下的 Partition-wise Join 106
8.3.4.2 SMP 场景下的 Full Partition-wise Join 108
8.4 分区表运维管理 110
8.4.1 新增分区 110
8.4.1.1 向范围分区表新增分区 111
8.4.1.2 向列表分区表新增分区 111

 版权所有 © 华为云计算技术有限公司

8.4.2 删除分区 111
8.4.3 交换分区 112
8.4.4 清空分区 113
8.4.5 分割分区 113
8.4.5.1 对范围分区表分割分区 114
8.4.5.2 对列表分区表分割分区 114
8.4.6 合并分区 115
8.4.7 移动分区 115
8.4.8 重命名分区 116
8.4.8.1 对分区表重命名分区 116
8.4.8.2 对 Local 索引重命名索引分区 116
8.4.9 分区表行迁移 116
8.4.10 分区表索引重建/不可用 117
8.4.10.1 索引重建/不可用 117
8.4.10.2 Local 索引分区重建/不可用 117
8.5 分区表系统视图&DFX 117
8.5.1 分区表相关系统视图 118
8.5.2 分区表相关内置工具函数 118
9 存储引擎 122
9.1 存储引擎体系架构 122
9.1.1 存储引擎体系架构概述 122
9.1.1.1 静态编译架构 122
9.1.1.2 通用数据库服务层 123
9.1.2 设置存储引擎 123
9.1.3 存储引擎更新说明 124
9.1.3.1 GaussDB 内核 505 版本 124
9.1.3.2 GaussDB 内核 503 版本 124
9.1.3.3 GaussDB 内核 R2 版本 125
9.2 Astore 存储引擎 125
9.2.1 Astore 简介 125
9.3 Ustore 存储引擎 126
9.3.1 Ustore 简介 126
9.3.1.1 Ustore 特性与规格 126
9.3.1.1.1 特性约束 126
9.3.1.1.2 存储规格 126
9.3.1.2 使用 Ustore 进行测试 127
9.3.1.3 Ustore 的最佳实践 128
9.3.1.3.1 怎么配置 init_td 大小 128
9.3.1.3.2 怎么配置 fillfactor 大小 129
9.3.1.3.3 在线校验功能 130
9.3.1.3.4 怎么配置回滚段大小 130
9.3.2 存储格式 131

 版权所有 © 华为云计算技术有限公司

9.3.2.1 RCR Uheap 131
9.3.2.1.1 RCR Uheap 多版本管理 131
9.3.2.1.2 RCR Uheap 可见性机制 131
9.3.2.1.3 RCR Uheap 空闲空间管理 132
9.3.2.2 UBTree 132
9.3.2.2.1 RCR UBTree 132
9.3.2.2.2 PCR UBTree 135
9.3.2.3 Undo 137
9.3.2.3.1 回滚段管理 137
9.3.2.3.2 文件组织结构 137
9.3.2.3.3 空间管理 137
9.3.2.4 Enhanced Toast 138
9.3.2.4.1 概述 138
9.3.2.4.2 Enhanced Toast 存储结构 138
9.3.2.4.3 Enhanced Toast 使用 138
9.3.2.4.4 Enhanced Toast 增删改查 138
9.3.2.4.5 Enhanced Toast 相关 DDL 操作 139
9.3.2.4.6 Enhanced Toast 运维管理 141
9.3.3 Ustore 事务模型 142
9.3.3.1 事务提交 142
9.3.3.2 事务回滚 142
9.3.4 闪回恢复 143
9.3.4.1 闪回查询 143
9.3.4.2 闪回表 145
9.3.4.3 闪回 DROP/TRUNCATE 147
9.3.5 常用视图工具 155
9.3.6 常见问题及定位手段 158
9.3.6.1 snapshot too old 158
9.3.6.1.1 长事务阻塞 Undo 空间回收 158
9.3.6.1.2 大量回滚事务拖慢 Undo 空间回收 159
9.3.6.2 storage test error 160
9.3.6.3 备机读业务报错:"UBTreeSearch::read_page has conflict with recovery, please try again later" 160
9.3.6.4 长查询执行期间大量并发更新偶现写入性能下降 161
9.4 数据生命周期管理-OLTP 表压缩 162
9.4.1 特性简介 162
9.4.2 特性约束 162
9.4.3 特性规格 163
9.4.4 使用说明 163
9.4.5 维护窗口参数配置 168
9.4.6 运维 TIPS 169
10 内存管理 176
11 DBMind: 数据库自治运维 182

 版权所有 © 华为云计算技术有限公司

11.1 DBMind 模式说明 184
11.1.1 service 子命令 185
11.1.2 component 子命令 192
11.1.3 set 子命令 193
11.1.4 upgrade 子命令 194
11.2 DBMind 的支持组件 196
11.2.1 Prometheus Exporter 组件 204
11.2.1.1 概述 204
11.2.1.2 环境部署 205
11.2.1.3 使用指导 205
11.2.1.4 获取帮助 210
11.2.1.5 命令参考 211
11.2.1.6 常见问题处理 218
11.3 DBMind 的 AI 子功能 218
11.3.1 X-Tuner 281
11.3.1.1 概述 281
11.3.1.2 使用准备 282
11.3.1.3 使用示例 286
11.3.1.4 获取帮助 289
11.3.1.5 命令参考 290
11.3.1.6 常见问题处理 292
11.3.2 Index-advisor 292
11.3.2.1 单 query 索引推荐 292
11.3.2.2 虚拟索引 294
11.3.2.3 workload 级别索引推荐 297
11.3.3 Slow Query Diagnosis 301
11.3.3.1 概述 301
11.3.3.2 环境部署 301
11.3.3.3 使用指导 301
11.3.3.4 获取帮助 303
11.3.3.5 命令参考 303
11.3.3.6 注意事项说明 304
11.3.4 SQLdiag 305
11.3.4.1 概述 305
11.3.4.2 使用指导 306
11.3.4.3 获取帮助 307
11.3.4.4 命令参考 308
11.3.4.5 常见问题处理 308
11.3.5 SQL Rewriter 308
11.3.5.1 概述 308
11.3.5.2 使用指导 309
11.3.5.3 获取帮助 309

 版权所有 © 华为云计算技术有限公司

11.3.5.4 命令参考 310
11.3.5.5 常见问题处理 310
11.3.6 Anomaly detection 310
11.3.6.1 概述 310
11.3.6.2 使用指导 314
11.3.6.3 获取帮助 320
11.3.6.4 命令参考 321
11.3.6.5 常见问题处理 321
11.3.7 Distributed-key-advisor 321
11.3.7.1 外部工具 321
11.3.7.1.1 概述 322
11.3.7.1.2 使用指导 322
11.3.7.1.3 获取帮助 323
11.3.7.1.4 常见问题处理 324
11.3.7.1.5 命令参考 324
11.3.7.2 SQL 内置接口 325
11.3.7.2.1 概述 325
11.3.7.2.2 使用指导 325
11.3.7.2.3 命令参考 325
11.3.8 Cluster Diagnosis 326
11.3.8.1 概述 326
11.3.8.2 使用指导 327
11.3.8.3 获取帮助 328
11.3.8.4 命令参考 329
11.3.8.5 常见问题处理 329
11.3.9 智能巡检 329
11.3.9.1 概述 329
11.3.9.2 使用指导 337
11.3.9.3 命令参考 343
11.3.9.4 常见问题处理 343
12 ABO 优化器 345
12.1 智能基数估计 345
12.1.1 概述 345
12.1.2 前置条件 345
12.1.3 使用指导 345
12.1.4 最佳实践 345
12.1.5 常见问题处理 346
13 Foreign Data Wrapper 347
13.1 file_fdw 347
14 GTM 模式 349
15 敏感数据发现 351

 版权所有 © 华为云计算技术有限公司

16 在线 DDL 352

17 bucket 分布表 354

17.1 hashbucket 354

 版权所有 © 华为云计算技术有限公司

1 修订记录

介绍GaussDB《 特性指南 》文档相比于505.1.0.SPC0100版本的主要变化点。

发布日期	修改说明
2024-06-30	第一次正式发布，包含以下内容： • 分区表，新增“Partition-wise Join”。相关文档**Partition-wise Join**。

版权所有 © 华为云计算技术有限公司

2 物化视图

物化视图是一种特殊的物理表，物化视图是相对普通视图而言的。普通视图是虚拟表，应用的局限性较大，任何对视图的查询实际上都是转换为对SQL语句的查询，性能并没有实际提高。而物化视图实际上就是存储SQL所执行语句的结果，起到缓存的效果。物化视图常用的操作包括创建、查询、删除和刷新。

根据创建规则，物化视图分为全量物化视图和增量物化视图。全量物化视图只支持全量刷新；增量物化视图支持全量刷新和增量刷新两种方式。全量刷新会将基表中的数据全部重新刷入物化视图中，而增量刷新只会将两次刷新间隔期间的基表产生的增量数据刷入物化视图中。

目前Ustore引擎不支持创建、使用物化视图。

2.1 全量物化视图

2.1.1 概述

全量物化视图仅支持对创建好的物化视图做全量刷新，而不支持做增量刷新。创建全量物化视图语法和CREATE TABLE AS语法一致（详情请参见《开发指南》中的“SQL参考 > SQL语法 > CREATE TABLE AS”章节），不支持对全量物化视图指定NodeGroup创建。全量物化视图的创建继承GTM-Free的相关约束。

2.1.2 支持和约束

支持场景

- 通常全量物化视图所支持的查询范围与CREATE TABLE AS语句一致。
- 创建全量物化视图可以指定分布列。
- 可以在全量物化视图上创建索引。
- 支持analyze、explain。

不支持场景

- 全量物化视图不支持NodeGroup。
- 不可对物化视图做增删改操作，只支持查询语句。

版权所有 © 华为云计算技术有限公司

- Ustore引擎不支持全量物化视图的创建和使用。

约束

- 创建全量物化视图所使用的基表必须在所有DN上有定义，基表所属nodegroup必须为installation group。
- 全量物化视图的刷新、删除过程中会给基表加高级别锁，若物化视图的定义涉及多张表，需要注意业务逻辑，避免死锁产生。

2.1.3 使用

语法格式

- 创建全量物化视图
  ```
  CREATE MATERIALIZED VIEW view_name AS query;
  ```
- 刷新全量物化视图
  ```
  REFRESH MATERIALIZED VIEW view_name;
  ```
- 删除物化视图
  ```
  DROP MATERIALIZED VIEW view_name;
  ```
- 查询物化视图
  ```
  SELECT * FROM view_name;
  ```

参数说明

- **view_name**

  要创建的物化视图名。

  取值范围：字符串，要符合标识符的命名规范。

- **AS query**

  一个SELECT VALUES命令或者一个运行预备好的SELECT或VALUES查询的EXECUTE命令。

示例

```
-- 修改表的默认类型
gaussdb=#  set enable_default_ustore_table=off;

-- 准备数据
CREATE TABLE t1(c1 int, c2 int);
INSERT INTO t1 VALUES(1, 1);
INSERT INTO t1 VALUES(2, 2);

-- 创建全量物化视图
gaussdb=# CREATE MATERIALIZED VIEW mv AS select count(*) from t1;
CREATE MATERIALIZED VIEW

-- 查询物化视图结果
gaussdb=# SELECT * FROM mv;
 count
-------
     2
(1 row)

-- 再次向物化视图中基表插入数据
gaussdb=# INSERT INTO t1 VALUES(3, 3);

-- 对全量物化视图做全量刷新
gaussdb=# REFRESH MATERIALIZED VIEW mv;
REFRESH MATERIALIZED VIEW
```


版权所有 © 华为云计算技术有限公司

```
-- 查询物化视图结果
gaussdb=# SELECT * FROM mv;
 count
-------
     3
(1 row)

-- 删除物化视图，删除表
gaussdb=# DROP MATERIALIZED VIEW mv;
DROP MATERIALIZED VIEW
gaussdb=# DROP TABLE t1;
DROP TABLE
```

2.2 增量物化视图

2.2.1 概述

增量物化视图可以对物化视图增量刷新，需要用户手动执行语句，刷新物化视图在一段时间内的增量数据。与全量创建物化视图的不同在于目前增量物化视图所支持场景较小。目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

2.2.2 支持和约束

支持场景

- 单表查询语句。
- 多个单表查询的UNION ALL。
- 在物化视图上创建索引。
- 对物化视图进行ANALYZE操作。
- 增量物化视图会继承基表NodeGroup创建（检查各个基表是否在同一个NodeGroup，并基于这个NodeGroup进行创建）。

不支持场景

- 物化视图中不支持带Stream计划，多表join连接计划以及subquery计划。
- 不支持WITH子句、GROUP BY子句、ORDER BY子句、LIMIT子句、WINDOW子句、DISTINCT算子、AGG算子，不支持除UNION ALL外的子查询。
- 除少部分ALTER操作外，不支持对物化视图中基表做绝大多数DDL操作。
- 创建物化视图不可指定物化视图分布列。
- 不可对物化视图做增删改操作，只支持查询语句。
- 不支持用临时表/hashbucket/unlog/分区表创建物化视图，只支持hash分布表。
- 不支持物化视图嵌套创建（物化视图上创建物化视图）。
- 不支持UNLOGGED类型的物化视图，不支持WITH语法。
- Ustore引擎不支持增量物化视图的创建和使用。

约束

- 物化视图定义如果为UNION ALL，则其中每个子查询需使用不同的基表，且各基表分布列相同。物化视图的分布列会自动推导且与各基表相同。

版权所有 © 华为云计算技术有限公司

- 物化视图定义的列必须包含基表的所有分布列。
- 增量物化视图的创建、全量刷新、删除过程中会给基表加高级别锁，若物化视图的定义为UNION ALL，需要注意业务逻辑，避免死锁产生。

2.2.3 使用

语法格式

- 创建增量物化视图
  ```
  CREATE INCREMENTAL MATERIALIZED VIEW view_name AS query;
  ```
- 全量刷新物化视图
  ```
  REFRESH MATERIALIZED VIEW view_name;
  ```
- 增量刷新物化视图
  ```
  REFRESH INCREMENTAL MATERIALIZED VIEW view_name;
  ```
- 删除物化视图
  ```
  DROP MATERIALIZED VIEW view_name;
  ```
- 查询物化视图
  ```
  SELECT * FROM view_name;
  ```

参数说明

- **view_name**

  要创建的物化视图名。

  取值范围：字符串，要符合标识符的命名规范。

- **AS query**

  一个SELECT VALUES命令或者一个运行预备好的SELECT或VALUES查询的EXECUTE命令。

示例

```
-- 修改表的默认类型
gaussdb=#  SET enable_default_ustore_table=off;

-- 准备数据
CREATE TABLE t1(c1 int, c2 int);
INSERT INTO t1 VALUES(1, 1);
INSERT INTO t1 VALUES(2, 2);

-- 创建增量物化视图
gaussdb=# CREATE INCREMENTAL MATERIALIZED VIEW mv AS SELECT * FROM t1;
CREATE MATERIALIZED VIEW

-- 插入数据
gaussdb=# INSERT INTO t1 VALUES(3, 3);
INSERT 0 1

-- 增量刷新物化视图
gaussdb=# REFRESH INCREMENTAL MATERIALIZED VIEW mv;
REFRESH MATERIALIZED VIEW

-- 查询物化视图结果
gaussdb=# SELECT * FROM mv;
 c1 | c2
----+----
  1 |  1
  2 |  2
  3 |  3
(3 rows)
```


版权所有 © 华为云计算技术有限公司

```
-- 插入数据
gaussdb=# INSERT INTO t1 VALUES(4, 4);
INSERT 0 1

-- 全量刷新物化视图
gaussdb=# REFRESH MATERIALIZED VIEW mv;
REFRESH MATERIALIZED VIEW

-- 查询物化视图结果
gaussdb=# select * from mv;
 c1 | c2
----+----
  1 |  1
  2 |  2
  3 |  3
  4 |  4
(4 rows)

-- 删除物化视图，删除表
gaussdb=# DROP MATERIALIZED VIEW mv;
DROP MATERIALIZED VIEW
gaussdb=# DROP TABLE t1;
DROP TABLE
```


版权所有 © 华为云计算技术有限公司

3 设置密态等值查询

3.1 密态等值查询概述

随着企业数据上云，数据的安全隐私保护面临越来越严重的挑战。密态数据库将解决数据整个生命周期中的隐私保护问题，涵盖网络传输、数据存储以及数据运行状态；更进一步，密态数据库可以实现云化场景下的数据隐私权限分离，即实现数据拥有者和实际数据管理者的数据读取能力分离。密态等值查询将优先解决密文数据的等值类查询问题。

加密模型

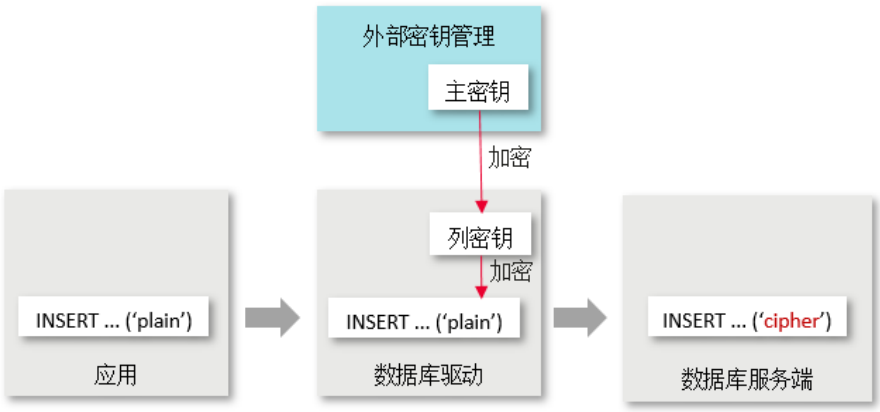
全密态数据库使用多级加密模型，加密模型中涉及3个对象：数据、列密钥和主密钥，以下是对3个对象的介绍：

- **数据**：
  a. SQL语法中包含的数据，比如INSERT... VALUES ('data')语法中包含'data'。
  b. 从数据库服务端返回的查询结果，如执行SELECT语法返回的查询结果。

  说明

  密态数据库会在驱动中，对SQL语法中属于加密列的数据进行加密，对数据库服务端返回的属于加密列的查询结果进行解密。
- **列密钥**：数据由列密钥进行加密，列密钥由数据库驱动生成或由用户手动导入，列密钥密文存储在数据库服务端。
- **主密钥**：列密钥由主密钥加密，主密钥由外部密钥管理者生成并存储。数据库驱动会自动访问外部密钥管理者，以实现对列密钥进行加解密。

 版权所有 © 华为云计算技术有限公司



整体流程

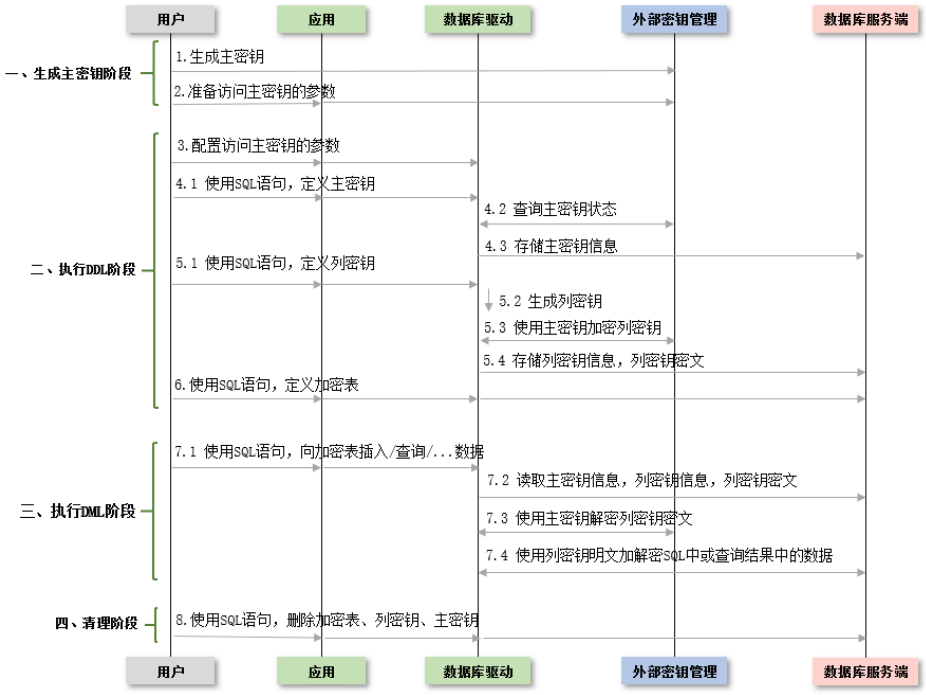
在使用全密态数据库的过程中，主要流程包括如下四个阶段，本节（2.1）介绍整体流程。**使用gsql操作密态数据库**、**使用JDBC操作密态数据库**章节介绍详细使用流程。

一、生成主密钥阶段：首先，用户需在华为云密钥服务或third_kms三方厂商密钥服务中生成主密钥。生成主密钥后，需准备访问主密钥的参数，以供数据库使用。

二、执行DDL阶段：在本阶段，用户可使用密态数据库的密钥语法依次定义主密钥和列密钥，然后定义表并指定表中某列为加密列。定义主密钥和列密钥的过程中，需访问上一阶段生成的主密钥。

三、执行DML阶段：在创建加密表后，用户可直接执行包含但不限于INSERT、SELECT、UPDATE、DELETE等语法。数据库驱动会自动根据上一阶段的加密定义自动对加密列中的数据进行加解密。

四、清理阶段：依次删除加密表、列密钥和主密钥。



版权所有 © 华为云计算技术有限公司

生成主密钥阶段

首次使用密态数据库时，需使用外部密钥管理服务生成至少一个主密钥，生成方式如下：

3.2 使用 gsql 操作密态数据库

执行 SQL 语句

执行本节的SQL语句前，请确保已提前生成主密钥，并确认访问主密钥的参数。

本节以完整的执行流程为例，介绍如何使用密态数据库语法，包括三个阶段：使用DDL阶段、使用DML阶段、清理阶段。

步骤1 连接数据库，并通过-C参数开启全密态开关

```
gsql -p PORT -d DATABASE -h HOST -U USER -W PASSWORD -r -C
```

步骤2 通过元命令设置访问主密钥的参数

注意：从keyType字符串开始，不要添加换行，不要添加空格，否则gsql工具无法识别完整参数。

步骤3 定义主密钥

在生成主密钥阶段，密钥服务已生成并存储主密钥，执行本语法只是将主密钥的相关信息存储在数据库中，方便以后访问。该语法详细格式参考：《开发指南》中“SQL参考 > SQL语法 > CREATE CLIENT MASTER KEY”章节。

```
CREATE CLIENT MASTER KEY
```

- 参数获取：生成主密钥阶段介绍了如何获取如下参数：KMS服务器地址、密钥ID。

步骤4 定义列密钥

列密钥由上一步定义的主密钥加密。详细语法参考：《开发指南》中“SQL参考 > SQL语法 > CREATE COLUMN ENCRYPTION KEY”章节。

```
gaussdb=# CREATE COLUMN ENCRYPTION KEY cek1 WITH VALUES (CLIENT_MASTER_KEY = cmk1,
ALGORITHM  = AES_256_GCM);
```

步骤5 定义加密表

本示例中，通过语法指定表中name和credit_card为加密列。

```
gaussdb=# CREATE TABLE creditcard_info (
  id_number int,
  name text encrypted with (column_encryption_key = cek1, encryption_type = DETERMINISTIC),
  credit_card varchar(19) encrypted with (column_encryption_key = cek1, encryption_type =
DETERMINISTIC));
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'id_number' as the distribution column by
default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
```

步骤6 对加密表进行其他操作

```
-- 向加密表写入数据
gaussdb=# INSERT INTO creditcard_info VALUES (1,'joe','6217986500001288393');
INSERT 0 1
gaussdb=# INSERT INTO creditcard_info VALUES (2, 'joy','6219985678349800033');
INSERT 0 1

-- 从加密表中查询数据
gaussdb=# select * from creditcard_info where name = 'joe';
```


版权所有 © 华为云计算技术有限公司

```
 id_number | name |      credit_card
-----------+------+--------------------
         1 | joe  | 6217986500001288393

-- 更新加密表中数据
gaussdb=# update creditcard_info set credit_card = '80000000011111111' where name = 'joy';
UPDATE 1

-- 向表中新增一列加密列
gaussdb=# ALTER TABLE creditcard_info ADD COLUMN age int ENCRYPTED WITH
(COLUMN_ENCRYPTION_KEY = cek1, ENCRYPTION_TYPE = DETERMINISTIC);
ALTER TABLE

-- 从表中删除一列加密列
gaussdb=# ALTER TABLE creditcard_info DROP COLUMN age;
ALTER TABLE

-- 从系统表中查询主密钥信息
gaussdb=# SELECT * FROM gs_client_global_keys;
 global_key_name | key_namespace | key_owner | key_acl |        create_date
-----------------+---------------+-----------+---------+----------------------------
 cmk1            |          2200 |        10 |         | 2021-04-21 11:04:00.656617
(1 rows)

-- 从系统表中查询列密钥信息
gaussdb=# SELECT column_key_name,column_key_distributed_id ,global_key_id,key_owner  FROM
gs_column_keys;
 column_key_name | column_key_distributed_id | global_key_id | key_owner
-----------------+---------------------------+---------------+-----------
 cek1            |                 760411027 |         16392 |        10
(1 rows)

-- 查看表中列的元信息
gaussdb=# \d creditcard_info
        Table "public.creditcard_info"
   Column    |       Type        | Modifiers
-------------+-------------------+------------
 id_number   | integer           |
 name        | text              |  encrypted
 credit_card | character varying |  encrypted
```

步骤7 清理阶段

```
-- 删除加密表
gaussdb=# DROP TABLE creditcard_info;
DROP TABLE

-- 删除列密钥
gaussdb=# DROP COLUMN ENCRYPTION KEY cek1;
DROP COLUMN ENCRYPTION KEY

-- 删除主密钥
gaussdb=# DROP CLIENT MASTER KEY cmk1;
DROP CLIENT MASTER KEY
```

----结束

3.3 使用 JDBC 操作密态数据库

配置 JDBC 驱动

1. 获取JDBC驱动包，JDBC驱动获取及使用可参考《开发指南》中“应用程序开发教程 > 基于JDBC开发”及”应用程序开发教程 > 兼容性参考 > JDBC兼容性包”章节。

   密态数据库支持的JDBC驱动包为gsjdbc4.jar、opengaussjdbc.jar、gscejdbc.jar。

 版权所有 © 华为云计算技术有限公司

- gscejdbc.jar（目前仅支持EulerOS操作系统）：主类名为“com.huawei.gaussdb.jdbc.Driver”，数据库连接的url前缀为“jdbc:gaussdb”，密态场景推荐使用此驱动包。本章的Java代码示例默认使用gscejdbc.jar包。
- gaussdbjdbc.jar：主类名为“com.huawei.gaussdb.jdbc.Driver”，数据库连接的url前缀为“jdbc:gaussdb”，此驱动包没有打包密态数据库需要加载的加解密相关的依赖库， 需要手动配置LD_LIBRARY_PATH环境变量。
- gaussdbjdbc-JRE7.jar：主类名为“com.huawei.gaussdb.jdbc.Driver”，数据库连接的url前缀为“jdbc:gaussdb”，在JDK1.7环境使用gaussdbjdbc-JRE7.jar包，此驱动包没有打包密态数据库需要加载的加解密相关的依赖库，需要手动配置LD_LIBRARY_PATH环境变量。

说明

其他兼容性：密态数据库支持的JDBC驱动包还支持其他兼容性驱动包gsjdbc4.jar、opengaussjdbc.jar。

- gsjdbc4.jar：主类名为“org.postgresql.Driver”，数据库连接的url前缀为“jdbc:postgresql”。
- opengaussjdbc.jar：主类名为“com.huawei.opengauss.jdbc.Driver”，数据库连接的url前缀为“jdbc:opengauss”。

2. 配置LD_LIBRARY_PATH。

   密态场景使用JDBC驱动包时，需要先设置环境变量LD_LIBRARY_PATH。

   - 使用gscejdbc.jar驱动包时，gscejdbc.jar驱动包中密态数据库需要的依赖库会自动复制到该路径下，并在开启密态功能连接数据库的时候加载。
   - 使用gaussdbjdbc.jar、gaussdbjdbc-JRE7.jar、opengaussjdbc.jar或gsjdbc4.jar时，需要同时解压包名为GaussDB-Kernel_数据库版本号_操作系统版本号_64bit_libpq.tar.gz的压缩包解压到指定目录，并将lib文件夹所在目录路径，添加至LD_LIBRARY_PATH环境变量中。

注意

全密态场景使用JDBC驱动包时需要有System.loadLibrary权限，以及环境变量LD_LIBRARY_PATH中第一优先路径的文件读写权限，建议使用独立目录作为全密态依赖库的存放路径。若在执行的时候指定java.library.path，需要与LD_LIBRARY_PATH的第一优先路径保持一致。

使用gscejdbc.jar时，jvm加载class文件需要依赖系统的libstdc++库，若开启密态则gscejdbc.jar会自动复制密态数据库依赖的动态库（包括libstdc++库）到用户设定的LD_LIBRARY_PATH路径下。如果依赖库与现有系统库版本不匹配，则首次运行仅部署依赖库，再次调用后即可正常使用。

执行 SQL 语句

执行本节的SQL语句前，请确保已提前生成主密钥，并确认访问主密钥的参数。

本节以完整的执行流程为例，介绍如何使用密态数据库语法，包括三个阶段：使用DDL阶段、使用DML阶段、清理阶段。

JDBC开发中与非密态场景操作一致的部分请参考《 开发指南 》中“应用程序开发教程 > 基于JDBC开发”章节。

版权所有 © 华为云计算技术有限公司

- 密态数据库连接参数

  enable_ce：String类型。其中enable_ce=0表示不开启全密态开关，enable_ce=1表示支持密态等值查询基本能力，enable_ce=3表示在密态等值查询能力的基础上支持内存解密逃生通道。

```
// 以下用例以gscejdbc.jar驱动为例，如果使用其他驱动包，仅需修改驱动类名和数据库连接的url前缀。
// gsjdbc4.jar: 主类名为“org.postgresql.Driver”，数据库连接的url前缀为“jdbc:postgresql”。
// opengaussjdbc.jar：主类名为“com.huawei.opengauss.jdbc.Driver”，数据库连接的url前缀为“jdbc:opengauss”。
// gscejdbc.jar：主类名为“com.huawei.gaussdb.jdbc.Driver”，数据库连接的url前缀为“jdbc:gaussdb”
// gaussdbjdbc.jar：主类名为“com.huawei.gaussdb.jdbc.Driver”，数据库连接的url前缀为“jdbc:gaussdb”。
// gaussdbjdbc-JRE7.jar：主类名为“com.huawei.gaussdb.jdbc.Driver”，数据库连接的url前缀为“jdbc:gaussdb”。

public static void main(String[] args) {
    // 驱动类。
    String driver = "com.huawei.gaussdb.jdbc.Driver";
    // 数据库连接描述符。enable_ce=1表示支持密态等值查询基本能力。
    String sourceURL = "jdbc:gaussdb://127.0.0.1:8000/postgres?enable_ce=1";
    // 在环境变量USER、PASSWORD分别配置用户名密码。
    String username = System.getenv("USER");
    String passwd = System.getenv("PASSWORD");
    Connection conn = null;
    try {
        // 加载驱动
        Class.forName(driver);
        // 创建连接
        conn = DriverManager.getConnection(sourceURL, username, passwd);
        System.out.println("Connection succeed!");
        // 创建语句对象
        Statement stmt = conn.createStatement();

        // 设置访问主密钥的参数
        // 此处介绍2种方式，选择其中1种方式即可：
        // 认证方式一 aksk认证（生成主密钥阶段介绍了如何获取相关参数：项目ID、AK、SK）


        /* 示例:
        */
        // 认证方式二 账号密码认证（生成主密钥阶段介绍了如何获取相关参数：IAM服务器地址、IAM用户名、IAM用户密码、账号名、项目）
            "iamUrl={IAM服务器地址}," +
            "iamUser={IAM用户名}," +
            "iamPassword={IAM用户密码}," +
            "iamDomain={账号名}," +
            "kmsProject={项目}");
        /* 示例：
                "iamUrl=https://iam.example.com/v3/auth/tokens," +
                "iamUser=test," +
                "iamPassword=*********," +
                "iamDomain=test_account," +
                "kmsProject=xxx");
        */
        // 定义主密钥：cmk1为主密钥名字，可自行取名
        // 生成主密钥阶段介绍了如何获取如下参数：KMS服务器地址、密钥ID


        /*
            解释：在生成主密钥阶段，密钥服务已生成并存储主密钥，执行本语法只是将主密钥的相关信息存储在数据库中，方便以后访问
            提示：KEY_PATH格式请参考：《开发指南》中“SQL参考 > SQL语法 > CREATE CLIENT MASTER KEY”章节
        */
        // 定义列加密密钥：列密钥由上一步创建的主密钥加密。详细语法参考：《开发指南》中“SQL参考 > SQL语法 > CREATE COLUMN ENCRYPTION KEY”章节
        int rc2 = stmt.executeUpdate("CREATE COLUMN ENCRYPTION KEY ImgCEK1 WITH VALUES (CLIENT_MASTER_KEY = ImgCMK1, ALGORITHM  = AES_256_GCM);");
        // 定义加密表
```

 版权所有 © 华为云计算技术有限公司

```
        int rc3 = stmt.executeUpdate("CREATE TABLE creditcard_info (id_number int, name varchar(50) 
encrypted with (column_encryption_key = ImgCEK1, encryption_type = DETERMINISTIC),credit_card 
varchar(19) encrypted with (column_encryption_key = ImgCEK1, encryption_type = 
DETERMINISTIC));");
        // 插入数据
        int rc4 = stmt.executeUpdate("INSERT INTO creditcard_info VALUES 
(1,'joe','6217986500001288393');");
        // 查询加密表
        ResultSet rs = null;
        rs = stmt.executeQuery("select * from creditcard_info where name = 'joe';");
        // 删除加密表
        int rc5 = stmt.executeUpdate("DROP TABLE IF EXISTS creditcard_info;");
        // 删除列加密密钥
        int rc6 = stmt.executeUpdate("DROP COLUMN ENCRYPTION KEY IF EXISTS ImgCEK1;");
        // 删除客户端主密钥
        int rc7 = stmt.executeUpdate("DROP CLIENT MASTER KEY IF EXISTS ImgCMK1;");
        // 关闭语句对象
        stmt.close();
        // 关闭连接
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
        return;
    }
}
```

说明

- 使用JDBC操作密态数据库时，一个数据库连接对象对应一个线程，否则，不同线程变更可能导致冲突。
- 使用JDBC操作密态数据库时，不同connection对密态配置数据有变更，由客户端调用isvalid方法保证connection能够持有变更后的密态配置数据，此时需要保证参数refreshClientEncryption为1(默认值为1)，在单客户端操作密态数据场景下，refreshClientEncryption参数可以设置为0。

- 调用isValid方法刷新缓存示例

```
// 创建连接conn1
Connection conn1 = DriverManager.getConnection("url","user","password");
// 在另外一个连接conn2中创建客户端主密钥
...
// conn1通过调用isValid刷新缓存，刷新conn1密钥缓存
try {
 if (!conn1.isValid(60)) {
  System.out.println("isValid Failed for connection 1");
 }
} catch (SQLException e) {
 e.printStackTrace();
        return null;
}
```

执行密态等值密文解密

数据库连接接口PgConnection类型新增解密接口，可以对全密态数据库的密态等值密文进行解密。解密后返回其明文值，通过schema.table.column找到解文对应的密文列并返回其原始数据类型。

 版权所有 © 华为云计算技术有限公司

表 3-1 新增 com.huawei.gaussdb.jdbc.jdbc.PgConnection 函数接口

方法名	返回值类型	支持JDBC 4
decryptData(String ciphertext, Integer len, String schema, String table, String column)	ClientLogicDecryptResult	Yes

参数说明：

- **ciphertext**
  需要解密的密文。
- **len**
  密文长度。当取值小于实际密文长度时，解密失败。
- **schema**
  加密列所属schema名称。
- **table**
  加密列所属table名称。
- **column**
  加密列所属column名称。

  说明

  下列场景可以解密成功，但不推荐：
  - 密文长度入参比实际密文长。
  - schema.table.column指向其他加密列，此时将返回被指向的加密列的原始数据类型。

表 3-2 新增 com.huawei.gaussdb.jdbc.jdbc.clientlogic.ClientLogicDecryptResult 函数接口

方法名	返回值类型	描述	支持JDBC4
isFailed()	Boolean	解密是否失败，若失败返回True，否则返回False。	Yes
getErrMsg()	String	获取错误信息。	Yes
getPlaintext()	String	获取解密后的明文。	Yes
getPlaintextSize()	Integer	获取解密后的明文长度。	Yes
getOriginalType()	String	获取加密列的原始数据类型。	Yes

```
// 通过非密态连接、逻辑解码等其他方式获得密文后，可使用该接口对密文进行解密
import com.huawei.gaussdb.jdbc.jdbc.PgConnection;
```

 版权所有 © 华为云计算技术有限公司

```
import com.huawei.gaussdb.jdbc.jdbc.clientlogic.ClientLogicDecryptResult;

// conn为密态连接
// 调用密态PgConnection的decryptData方法对密文进行解密，通过列名称定位到该密文的所属加密列，并返回其原始数据类型
ClientLogicDecryptResult decrypt_res = null;
decrypt_res = ((PgConnection)conn).decryptData(ciphertext, ciphertext.length(), schemaname_str,
        tablename_str, colname_str);
// 检查返回结果类解密成功与否，失败可获取报错信息，成功可获得明文及长度和原始数据类型
if (decrypt_res.isFailed()) {
    System.out.println(String.format("%s\n", decrypt_res.getErrMsg()));
} else {
    System.out.println(String.format("decrypted plaintext: %s size: %d type: %s\n", decrypt_res.getPlaintext(),
        decrypt_res.getPlaintextSize(), decrypt_res.getOriginalType()));
}
```

执行加密表的预编译 SQL 语句

```
// 调用Connection的prepareStatement方法创建预编译语句对象。
PreparedStatement pstmt = conn.prepareStatement("INSERT INTO creditcard_info VALUES (?, ?, ?);");
// 调用PreparedStatement的setShort设置参数。
pstmt.setInt(1, 2);
pstmt.setString(2, "joy");
pstmt.setString(3, "6219985678349800033");
// 调用PreparedStatement的executeUpdate方法执行预编译SQL语句。
int rowcount = pstmt.executeUpdate();
// 调用PreparedStatement的close方法关闭预编译语句对象。
pstmt.close();
```

执行加密表的批处理操作

```
// 调用Connection的prepareStatement方法创建预编译语句对象。
Connection conn = DriverManager.getConnection("url","user","password");
PreparedStatement pstmt = conn.prepareStatement("INSERT INTO creditcard_info (id_number, name,
credit_card) VALUES (?,?,?)");
// 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。
int loopCount = 20;
 for (int i = 1; i < loopCount + 1; ++i) {
      pstmt.setInt(1, i);
      pstmt.setString(2, "Name " + i);
      pstmt.setString(3, "CreditCard " + i);
      // Add row to the batch.
      pstmt.addBatch();
}
// 调用PreparedStatement的executeBatch方法执行批处理。
int[] rowcount = pstmt.executeBatch();
// 调用PreparedStatement的close方法关闭预编译语句对象。
pstmt.close();
```

3.4 前向兼容与安全增强

前向兼容

在上文中，支持通过key_info设置访问外部密钥管理的参数：

1. 使用gsql时，通过元命令\key_info xxx设置。
2. 使用JDBC时，通过连接参数conn.setProperty(“key_info”, “xxx”)设置。

为保持前向兼容，还支持通过环境变量等方式设置访问主密钥的参数。

 版权所有 © 华为云计算技术有限公司

⚠ 注意

第一次配置使用密态数据库时，可忽略下述方法。如果以前使用下述方法配置密态数据库，建议改用’key_info’配置。

使用系统级环境变量配置的方式如下：

```
export HUAWEI_KMS_INFO='iamUrl=https://iam.{项目}.myhuaweicloud.com/v3/auth/tokens,iamUser={IAM用户名},iamPassword={IAM用户密码},iamDomain={账号名},kmsProject={项目}'
# 该方法中操作系统日志可能会记录环境变量中的敏感信息，使用过程中注意及时清理。
```

还可通过标准库接口设置进程级环境变量，不同语言设置方法如下：

1. C/C++

   ```
   setenv("HIS_KMS_INFO", "xxx");
   ```

2. GO

   ```
   os.Setenv("HIS_KMS_INFO", "xxx");
   ```

外部密钥服务的身份验证

当数据库驱动访问华为云密钥管理服务时，为避免攻击者伪装为密钥服务，在数据库驱动与密钥服务建立https连接的过程中，可通过CA证书验证密钥服务器的合法性。为此，需提前配置CA证书，如果未配置，将不会验证密钥服务的身份。本节介绍如何下载与配置CA证书。

配置方法

在key_info参数的中，增加证书相关参数即可。

- 使用gsql时

  ```
  gaussdb=# \key_info keyType=,iamUrl=https://iam.example.com/v3/auth/tokens,iamUser={IAM用户名},iamPassword={IAM用户密码},iamDomain={账号名},kmsProject={项目},iamCaCert=/路径/IAM的CA证书文件,kmsCaCert=/路径/KMS的CA证书文件

  gaussdb=# \key_info keyType=,kmsProjectId={项目ID},ak={AK},sk={SK},kmsCaCert=/路径/KMS的CA证书文件
  ```

- 使用JDBC时

  ```
  "iamUrl=https://iam.{example.com/v3/auth/tokens," +
  "iamUser={IAM用户名}," +
  "iamPassword={IAM用户密码}," +
  "iamDomain={账号名}," +
  "kmsProject={项目}," +
  "iamCaCert=/路径/IAM的CA证书文件," +
  "kmsCaCert=/路径/KMS的CA证书文件");
  ```

获取证书

大部分浏览器均会自动下载网站对应的CA证书，并提供证书导出功能。虽然，诸如**https://www.ssleye.com/ssltool/certs_down.html**等很多网站也提供自动下载CA证书的功能，但可能因本地环境中存在代理或网关，导致CA证书无法正常使用。所以，建议借助浏览器下载CA证书。下载方式如下：

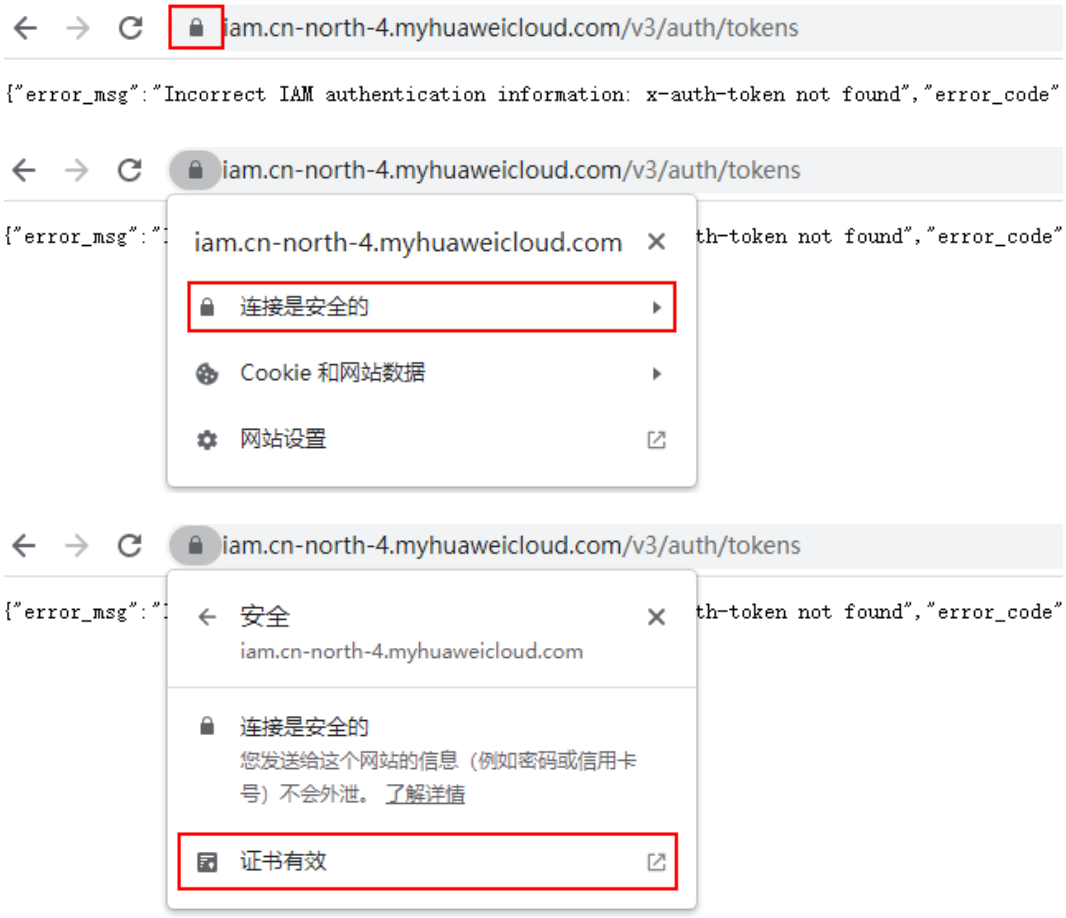
⚠ 注意

由于使用restful接口访问密钥服务，当在浏览器输入接口对应的url时，可忽略下述**步骤2**中的失败页面，因为即使在失败的情况下，浏览器也早已提前自动下载CA证书。

版权所有 © 华为云计算技术有限公司

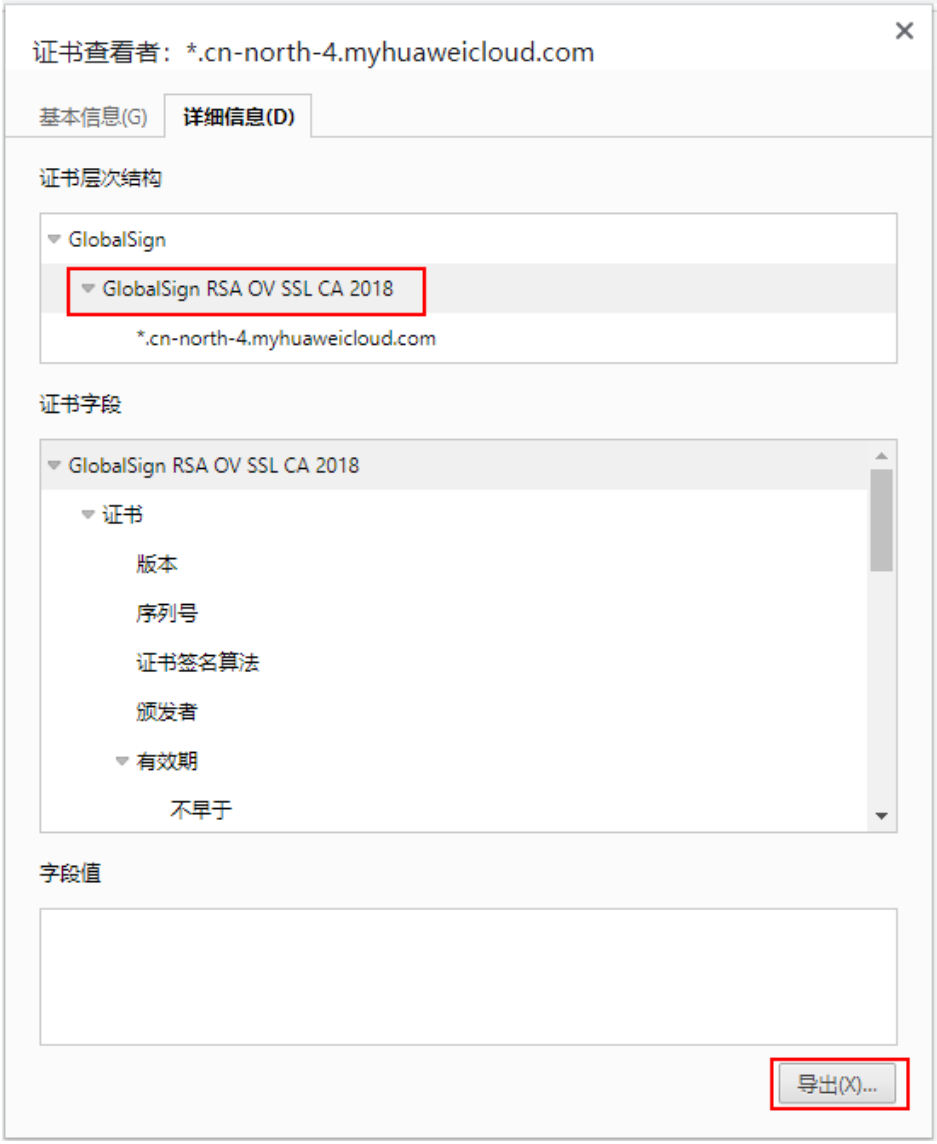
步骤1 输入域名：打开浏览器，在华为云场景中，分别输入IAM服务器地址和KMS服务器地址，地址获取方式参见：**生成主密钥阶段**。

步骤2 查找证书：在每次输入域名后，找到SSL连接相关信息，单击后会发现证书，继续单击可查看证书内容。



步骤3 导出证书：在证书查看页面，可能会看到证书分为很多级，仅需要域名的上一级证书即可，选择该证书并单击导出，便可直接生成证书文件，即需要的证书文件。

 版权所有 © 华为云计算技术有限公司



步骤4 上传证书：将导出的证书上传至应用端，并配置到上述参数中即可。

----结束

3.5 密态支持函数/存储过程

密态支持函数/存储过程，当前版本只支持sql和plpgsql两种语言。由于密态支持存储过程中创建和执行函数/存储过程对用户是无感知的，因此语法和非密态无区别。

函数/存储过程语法参考《开发指南》中“用户自定义函数”章节和“存储过程”章节。

密态等值查询支持函数存储过程新增系统表gs_encrypted_proc，用于存储参数返回的原始数据类型。

系统表具体字段含义可参考《开发指南》中“系统表和系统视图 > 系统表 > GS_ENCRYPTED_PROC”章节。

 版权所有 © 华为云计算技术有限公司

创建并执行涉及加密列的函数/存储过程

步骤1 创建密钥，详细步骤请参见**使用gsql操作密态数据库**。

步骤2 创建加密表。

```
gaussdb=# CREATE TABLE creditcard_info (
  id_number int,
  name  text,
  credit_card varchar(19) encrypted with (column_encryption_key = cek1, encryption_type = DETERMINISTIC)
 ) with (orientation=row) distribute by hash(id_number);
CREATE TABLE
```

步骤3 插入数据。

```
gaussdb=# insert into creditcard_info values(1, 'Avi', '1234567890123456');
INSERT 0 1
gaussdb=# insert into creditcard_info values(2, 'Eli', '2345678901234567');
INSERT 0 1
```

步骤4 创建函数支持密态等值查询。

```
gaussdb=# CREATE FUNCTION f_encrypt_in_sql(val1 text, val2 varchar(19)) RETURNS text AS 'SELECT name from creditcard_info where name=$1 or credit_card=$2 LIMIT 1' LANGUAGE SQL;
CREATE FUNCTION
gaussdb=# CREATE FUNCTION f_encrypt_in_plpgsql (val1 text, val2 varchar(19), OUT c text) AS $$
 BEGIN
 SELECT into c name from creditcard_info where name=$1 or credit_card =$2 LIMIT 1;
 END; $$
 LANGUAGE plpgsql;
CREATE FUNCTION
```

步骤5 执行函数。

```
gaussdb=# SELECT f_encrypt_in_sql('Avi','1234567890123456');
 f_encrypt_in_sql
------------------
 Avi
(1 row)

gaussdb=# SELECT f_encrypt_in_plpgsql('Avi', val2=>'1234567890123456');
 f_encrypt_in_plpgsql
----------------------
 Avi
(1 row)
```

----结束

 版权所有 © 华为云计算技术有限公司

说明

1. 函数/存储过程中的“执行动态查询语句”所包含的查询是在执行过程编译，因此函数/存储过程中的表名、列名不能在创建阶段未知，输入参数不能用于表名、列名或以任何方式连接。
2. 函数/存储过程中的“执行动态查询语句”不支持EXECUTE 'query'中带有需要加密的数据值。
3. 在RETURNS、IN和OUT的参数中，不支持混合使用加密和非加密类型参数。虽然参数类型都是原始数据类型，但实际类型不同。
4. 在高级包接口中，如dbe_output.print_line()等在服务端打印输出的接口不会做解密操作，由于加密数据类型在强转成明文原始数据类型时会打印出该数据类型的默认值。
5. 当前版本函数/存储过程的LANGUAGE只支持SQL和plpgsql，不支持C和JAVA等其他过程语言。
6. 不支持在函数/存储过程中执行其他查询加密列的函数/存储过程。
7. 当前版本不支持default、DECLARE中为变量授予默认值，且不支持对DECLARE中的返回值进行解密，用户可以在执行函数时用输入参数、输出参数来代替使用。
8. 不支持gs_dump对涉及加密列的function进行备份。
9. 不支持在函数/存储过程中创建密钥。
10. 该版本密态函数/存储过程不支持触发器。
11. 密态等值查询函数/存储过程不支持对plpgsql语言对语法进行转义，对于语法主体带有引号的CREATE FUNCTION AS ‘语法主体'，可以用CREATE FUNCTION AS $$语法主体$$代替。
12. 不支持在密态等值查询函数/存储过程中执行修改加密列定义的操作，包括对创建加密表，添加加密列，由于执行函数是在服务端，客户端没法判断是否需要刷新缓存，需断开连接后或触发刷新客户端加密列缓存才可以对该列做加密操作。
13. 不支持使用密态数据类型（byteawithoutorderwithequalcol、byteawithoutordercol、_byteawithoutorderwithequalcol、_byteawithoutordercol）创建函数和存储过程。
14. 密态函数若返回值有加密类型，不支持返回不确定的行类型结果，如RETURN [SETOF] RECORD，可以使用返回可确定的行类型结果替代，如RETURN TABLE(columnname typename[,...])。
15. 密态支持函数在创建加密函数时会在系统表gs_encrypted_proc中添加参数对应的加密列的OID，因此删除表后重建同名表可能会使密态函数失效，需要重新创建密态函数。

3.6 使用 third_kms 三方厂商加解密库

全密态等值查询与透明加密功能选择密钥管理工具时，可以选择third_kms作为密钥管理工具及加解密库。在该功能打开后，数据库进程会把动态库文件载入到内存。加解密接口的正确性和安全性由第三方厂商保证，上线前需经过充分测试，避免影响数据库正常业务。

规格约束

当使用third_kms时，不支持密钥轮转语句轮转密钥。

操作步骤

在使用third_kms三方厂商加解密库前需要按照以下操作步骤进行配置：

步骤1 向第三方厂商获取名字为libthird_crypto.so的加解密文件及其使用说明。

步骤2 将加解密库文件放置在LD_LIBRARY_PATH下的目录中。

建议与数据库客户端其他库放在同一目录下，比如libpq_ce.so。

版权所有 © 华为云计算技术有限公司

步骤3 使用crypto_module_info设置动态库加载信息以及设置key_info，不同客户端的设置内容一致。

其中module_third_msg为加密库加载时的必要信息，keyThirdMsg为加密库创建会话时的登录信息，这两项信息的格式和内容由第三方厂商定义，数据库仅通过crypto_module_init接口对该参数的合法性进行校验，建议第三方厂商通过该参数校验对应版本号的前后向兼容性。

- gsql：
```
gaussdb=#  \crypto_module_info enable_crypto_module=on,module_third_msg=aaa
load crypto module success
gaussdb=#  \key_info keyType=third_kms,keyThirdMsg=aaa
```
- JDBC：JDBC客户端可以通过连接参数或Connection类中的setClientInfo方法设置。
  - 通过连接参数设置
```
String sourceURL = "jdbc:gaussdb://127.0.0.1:8000/postgres?
enable_ce=1&crypto_module_info='enable_crypto_module=on,module_third_msg=aaa'&key_info=
'keyType=third_kms, keyThirdMsg=aaa'";
// 获取数据库连接池句柄
Connection conn = DriverManager.getConnection(sourceURL, username, passwd);
```
  - 通过Connection类设置
```
//此示例前已成功创建连接获得Connection类的实例conn
conn.setClientInfo("crypto_module_info", "enable_crypto_module=on,module_third_msg=aaa" );
conn.setClientInfo("key_info", "keyType=third_kms,keyThirdMsg=aaa" );
```
- go：go客户端通过连接参数的方式设置。
```
str := "host=xxx.xxx.xxx.x port=xxx user=xxx password=******** dbname=xxx enable_ce=1
crypto_module_info='enable_crypto_module=on,module_third_msg=aaa' key_info='keyType=third_kms,
keyThirdMsg=aaa'"
// 获取数据库连接池句柄
db, err := sql.Open("gaussdb", str)
```

步骤4 创建主密钥、列密钥以及创建加密表。

```
gaussdb=# CREATE CLIENT MASTER KEY cmk1 WITH (KEY_STORE = third_kms);
CREATE CLIENT MASTER KEY
-- CREATE COLUMN ENCRYPTION KEY语法中ENCRYPTED_VALUE为可选入参，可以用以传入列密钥ID（hex格式）。不填则通过三方库密钥生成接口自动生成随机密钥。
gaussdb=# CREATE COLUMN ENCRYPTION KEY cek1 WITH VALUES (CLIENT_MASTER_KEY = cmk1,
ENCRYPTED_VALUE='13481754638');
CREATE COLUMN ENCRYPTION KEY
gaussdb=# CREATE TABLE t16 (c1 INT, c2 TEXT ENCRYPTED WITH (COLUMN_ENCRYPTION_KEY = cek1 ,
ENCRYPTION_TYPE = DETERMINISTIC));
CREATE TABLE
gaussdb=# INSERT INTO t16 VALUES(1, 'Gauss');
INSERT 0 1
```

----结束

第三方厂商实现加解密接口注意事项

- 三方厂商需按照**第三方厂商实现加解密接口头文件**对接口进行实现并编译成以libthird_crypto.so命名的文件。
- 加密库链接时，仅可使用与数据库进程中相同版本的依赖库，不允许引入外部依赖库。
- 厂商提供加密库给客户时需保证编译环境与客户环境一致（例如：x86 or ARM，os-release版本等）。
- 在**第三方厂商实现加解密接口头文件**中，crypto_module_init的返回值SupportedFeature需要支持除MODULE_DETERMINISTIC_KEY外的所有算法，需要实现除my_cipher_deterministic_enc_dec外所有接口。（不实现确定性算法及其接口仅导致全密态等值查询不可使用该库，仍可供透明加密功能使用）

 版权所有 © 华为云计算技术有限公司

- 为保证兼容性，若涉及接口头文件升级，厂商需要基于升级后的头文件重新编译C函数共享库。
- 如果实现语言为C++，则需要为实现的函数加上extern "C"，以保证数据库加载时能正确找到函数。
- 编译选项需要带上安全编译选项和优化选项，如trapv、Wl、relro、z、now、noexecstack、pie、PIC、stack-protector、O3等。

⚠ 注意

厂商应充分了解要实现的接口功能及入参范围，在调用前应检查参数合法性，避免出现空指针等可能导致程序crash的问题。厂商应避免接口调用过程中产生内存泄漏。

第三方厂商实现加解密接口头文件

```
#define CRYPT_MOD_OK 1
#define CRYPT_MOD_ERR 0
typedef enum {
    MODULE_AES_128_CBC = 0,
    MODULE_AES_128_CTR,
    MODULE_AES_128_GCM,
    MODULE_AES_256_CBC,
    MODULE_AES_256_CTR,
    MODULE_AES_256_GCM,
    MODULE_SM4_CBC,
    MODULE_SM4_CTR,
    MODULE_HMAC_SHA256,
    MODULE_HMAC_SM3,
    MODULE_DETERMINISTIC_KEY,
    MODULE_ALGO_MAX = 1024
} ModuleSymmKeyAlgo;
typedef enum {
    MODULE_SHA256 = 0,
    MODULE_SM3,
    MODULE_DIGEST_MAX = 1024
} ModuleDigestAlgo;
typedef enum {
    KEY_TYPE_INVALID,
    KEY_TYPE_PLAINTEXT,
    KEY_TYPE_CIPHERTEXT,
    KEY_TYPE_NAMEORIDX,
    KEY_TYPE_MAX
} KeyType;
typedef struct {
    KeyType key_type;
    int supported_symm[MODULE_ALGO_MAX]; // 不支持算法填入0或者支持算法填入1
    int supported_digest[MODULE_DIGEST_MAX]; // 不支持算法填入0或者支持算法填入1
} SupportedFeature;
/** 初始化密码模块
 *
 * @param[in]
 *    load_info            密码模块相关信息（硬件设备密码，硬件设备、硬件库路径等），通过kv方式传入
 *
 * @param[out]
 *    supported_feature    返回当前密码模块支持的加密方式，参考上述结构体
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_module_init(char *load_info, SupportedFeature *supported_feature);
/** 会话中连接密码模块
 *
 * @param[in]
 *    key_info        密码相关信息（用户密码等信息），通过kv方式传入
```


版权所有 © 华为云计算技术有限公司

```
 *
 * @param[out]
 *    sess            会话信息
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_module_sess_init(char *key_info, void **sess);
/** 会话中断开连接密码模块
 *
 * @param[in]
 *    sess            会话信息
 *
 * @param[out]
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
void crypto_module_sess_exit(void *sess);
/** 创建密钥
 *
 * @param[in]
 *    sess            会话信息
 *    algo            密钥使用场景的算法
 *
 * @param[out]
 *    key_id          返回生成密钥/密钥ID/密钥密文
 *    key_id_size     返回生成内容长度
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_create_symm_key(void *sess, ModuleSymmKeyAlgo algo, unsigned char *key_id, size_t
*key_id_size);
/** 密钥上下文初始化，后续进行加解密可直接使用上下文
 *
 * @param[in]
 *    sess            会话信息
 *    algo            加密算法
 *    enc             加密1、解密0
 *    key_id          密码信息
 *    key_id_size     密码信息长度
 * @param[out]
 *    ctx             返回使用密钥信息
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_ctx_init(void *sess, void **ctx, ModuleSymmKeyAlgo algo, int enc, unsigned char *key_id, size_t
key_id_size);
/** 获取数据加解密后的数据长度
 *
 * @param[in]
 *    ctx             加解密上下文信息
 *    enc             加密1、解密0
 * @param[out]
 *    data_size       返回加解密结果长度
 * @return    成功返回数据长度，失败返回-1
 *
 */
int crypto_result_size(void *ctx, int enc, size_t data_size);
/** 密钥上下文清理
 *
 * @param[in]
 *    ctx             加解密上下文信息
 * @param[out]
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
void crypto_ctx_clean(void *ctx);
/** 执行加解密
```

 版权所有 © 华为云计算技术有限公司

```
 *
 * @param[in]
 *    ctx             加解密上下文信息
 *    enc             加密1、解密0
 *    data            原数据信息
 *    data_size       原数据长度
 *    iv              iv信息
 *    iv_size         iv信息长度
 *    tag             GCM模式的校验值
 * @param[out]
 *    result          返回结果信息
 *    result_size     返回结果信息长度
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_encrypt_decrypt(void *ctx, int enc, unsigned char *data, size_t data_size, unsigned char *iv, size_t
iv_size, unsigned char *result, size_t result_size, unsigned char *tag);
/** 计算摘要
 *
 * @param[in]
 *    sess            会话信息
 *    algo            摘要算法
 *    data            原数据信息
 *    data_size       原数据长度
 * @param[out]
 *    result          返回结果信息
 *    result_size     返回结果信息长度
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_digest(void *sess, ModuleDigestAlgo algo, unsigned char * data, size_t data_size,unsigned char
*result, size_t *result_size);
/** hmac初始化
 *
 * @param[in]
 *    sess            会话信息
 *    algo            摘要算法
 *    key_id          密码信息
 *    key_id_size     密码信息长度
 * @param[out]
 *    ctx             返回密钥上下文
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_hmac_init(void *sess, void **ctx, ModuleDigestAlgo algo, unsigned char *key_id, size_t key_id_size);
/** hmac清理
 *
 * @param[in]
 *    ctx             密钥上下文信息
 *
 * @param[out]
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
void crypto_hmac_clean(void *ctx);
/** 执行hmac计算
 *
 * @param[in]
 *    ctx             密钥上下文信息
 *    data            原数据信息
 *    data_size       原数据长度
 * @param[out]
 *    result          返回结果信息
 *    result_size     返回结果信息长度
 *
```

 版权所有 © 华为云计算技术有限公司

```
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_hmac(void *ctx, unsigned char * data, size_t data_size, unsigned char *result, size_t *result_size);
/** 生成随机数
 *
 * @param[in]
 *    sess            会话信息
 *    size            申请的随机信息长度
 *
 * @param[out]
 *    buffer          返回随机信息
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_gen_random(void *sess, char *buffer, size_t size);
/** 执行确定性加解密
 *
 * @param[in]
 *    sess            会话信息
 *    enc             加密1、解密0
 *    data            原数据信息
 *    data_size       原数据长度
 *    key_id          密钥信息
 *    key_id_size     密钥信息长度
 * @param[out]
 *    result          返回结果信息
 *    result_size     返回结果信息长度
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_deterministic_enc_dec(void *sess, int enc, unsigned char *data, unsigned char *key_id,
    size_t key_id_size, size_t data_size, unsigned char *result, size_t *result_size);
/** 获取报错信息
 *
 * @param[in]
 *    sess            会话信息
 * @param[out]
 *    errmsg          返回结果信息,最长256字节
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_get_errmsg(void *sess, char *errmsg);
```

 版权所有 © 华为云计算技术有限公司

4 透明数据加密

透明加密提供表级数据加密存储功能。当用户使用本特性提供的语法创建加密表后，数据库向磁盘写入加密表数据前，会自动将其加密；同时，数据库从磁盘读取加密表数据后，会自动将其解密。向加密表中进行数据插入、更新、查询和删除等语法与非加密表一致。

说明

- 透明加密基本原理请参见《特性描述》的“数据库安全 > 透明数据加密”章节。
- 使用透明加密前需联系管理员配置开启该功能。

查看透明加密基本配置

步骤1 查看透明加密功能是否已开启

enable_tde取值为on时表示开启，取值为off是表示关闭。该参数由管理员设置。

```
gaussdb=# SHOW enable_tde;
 enable_tde
------------
 on
(1 row)
```

步骤2 查看是否已设置访问密钥管理服务的参数

tde_key_info参数为空时表示未设置，tde_key_info不为空时表示已设置。该参数由管理员设置。

```
gaussdb=# show tde_key_info;
      tde_key_info
-------------------------
 keyType=...
```

----结束

操作加密表

步骤1 创建加密表。

创建表时，通过在WITH子句中设置enable_tde=on参数，即可设置该表为加密表。

数据库默认使用'AES_128_CTR'算法对加密表进行加密，如需使用其他算法，可通过encrypt_algo参数设置。

当透明加密基本配置的tde_key_info中的keyType为third_kms时，可以设置dek_token作为数据密钥的ID，并会在透明加密时传给第三方加解密库。

 版权所有 © 华为云计算技术有限公司

```
gaussdb=# CREATE TABLE t1 (c1 INT, c2 TEXT) WITH (enable_tde = on);
CREATE TABLE
gaussdb=# CREATE TABLE t2 (c1 INT, c2 TEXT) WITH (enable_tde = on, encrypt_algo = 'SM4_CTR');
CREATE TABLE
```

步骤2 查看加密表基本信息。

加密表基本信息存储在pg_class系统表中的reloptions字段中。其中，dek_cipher为数据密钥密文，由数据库自动生成，并由密钥管理服务加密。每个加密表都有1个独立的数据密钥。

```
gaussdb=# SELECT relname,reloptions FROM pg_class WHERE relname = 't1';
 relname | reloptions
---------
+-----------------------------------------------------------------------------------------------------------------------------
 t1      |
{orientation=row,enable_tde=on,encrypt_algo=AES_128_CTR,compression=no,storage_type=USTORE,key_type
=...,dek_cipher=...
```

步骤3 向加密表写入数据。

操作加密表与非加密表的语法一致。数据库将表中数据写入磁盘前，才会自动对加密表的数据进行加密。

```
gaussdb=# INSERT INTO t1 VALUES (1, 'tde plain 123');
INSERT 0 1
```

步骤4 从加密表查询数据。

对于合法用户而言，查询加密表与非加密表的语法一致，加解密操作由数据库自动实现。如果攻击者绕过数据库，直接读取磁盘上加密表对应的数据文件，会发现文件中的数据均已被加密。

```
gaussdb=# SELECT * FROM t1;
 c1 |      c2
----+---------------
  1 | tde plain 123
(1 row)
```

步骤5 轮转加密表的密钥。

为提高安全性，建议定期使用以下语法轮转加密表的数据密钥，即使用新的密钥对数据进行加密。

```
gaussdb=# ALTER TABLE t1 ENCRYPTION KEY ROTATION;
ALTER TABLE
```

轮转密钥后，数据库仍可以正常解密由旧密钥加密的数据。

步骤6 加密表与非加密表转换。

透明加密支持将加密表转换为非加密表，以及将非加密表转换为加密表。建议在每次转换后，手动执行VACUUM FULL tablename命令，以强制同步转换表中所有数据。

```
gaussdb=# CREATE TABLE t3 (c1 INT, c2 TEXT);
CREATE TABLE
gaussdb=# ALTER TABLE t3 SET (enable_tde = on);
ALTER TABLE
gaussdb=# VACUUM FULL t3;
VACUUM
gaussdb=# ALTER TABLE t3 SET (enable_tde = off);
ALTER TABLE
gaussdb=# VACUUM FULL t3;
VACUUM
```

步骤7 删除加密表。

```
gaussdb=# DROP TABLE IF EXISTS t1, t2, t3;
DROP TABLE
```

----结束

 版权所有 © 华为云计算技术有限公司

操作加密索引

步骤1 创建加密表。

创建索引的基表，需确保基表也是加密表。

```
gaussdb=# CREATE TABLE t1 (c1 INT, c2 TEXT) WITH (enable_tde = on);
CREATE TABLE
```

步骤2 创建加密索引。

与创建加密表的方式相同，通过在WITH子句中设置enable_tde=on参数，即将索引设置为加密索引。

索引与基表使用相同的加密算法和密钥，对基表进行密钥轮转时，索引也会使用新密钥。

```
gaussdb=# CREATE INDEX i1 ON t1(c2) WITH (enable_tde = on);
CREATE INDEX
```

步骤3 查看加密索引基本信息。

与加密表一样，索引基本信息也存储在pg_class系统表中的reloptions字段中，索引的dek_cipher、encrypt_algo等参数与基表保持一致。

```
gaussdb=# SELECT relname,reloptions FROM pg_class WHERE relname = 'i1';
 relname | reloptions
---------
+-----------------------------------------------------------------------------------------------------------------------
 i1      |
{orientation=row,enable_tde=on,encrypt_algo=AES_128_CTR,compression=no,storage_type=USTORE,key_type
=...,dek_cipher=...
```

步骤4 加密索引与非加密索引转换。

透明加密支持将非加密索引转换为加密索引，将加密索引转换为非加密索引。

```
gaussdb=# CREATE TABLE t2 (c1 INT, c2 TEXT) WITH (enable_tde = on);
ALTER TABLE
gaussdb=# CREATE INDEX i2 ON t2(c2);
CREATE INDEX
gaussdb=# ALTER INDEX i2 SET (enable_tde = on);
ALTER INDEX
gaussdb=# ALTER INDEX i2 SET (enable_tde = off);
ALTER INDEX
```

步骤5 自动对索引进行加密。

默认情况下，主动设置enable_tde参数才可创建加密索引。当设置GUC参数tde_index_default_encrypt=on，且以加密表为基表创建索引时，数据库会自动将索引转换为加密索引。示例如下：

```
gaussdb=# CREATE TABLE t3 (c1 INT, c2 TEXT) WITH (enable_tde = on);
ALTER TABLE
gaussdb=# CREATE INDEX i3 ON t3(c2);
CREATE INDEX
gaussdb=# SELECT relname,reloptions FROM pg_class WHERE relname = 'i3';
 relname | reloptions
---------
+-----------------------------------------------------------------------------------------------------------------------
 i1      |
{orientation=row,enable_tde=on,encrypt_algo=AES_128_CTR,compression=no,storage_type=USTORE,key_type
=...,dek_cipher=...

-- 解释：虽然未指定i3为加密索引，但是开启了tde_index_default_encrypt=on，且基表t3是加密表，数据库自动将i3转换为加密索引
```

 版权所有 © 华为云计算技术有限公司

步骤6 删除加密表和索引。

```
gaussdb=# DROP TABLE IF EXISTS t1, t2, t3;
DROP TABLE
```

----结束

支持加密 xlog 和 undo-log

步骤1 创建TDE属性的表。

```
gaussdb=# CREATE TABLE t1 (c1 INT, c2 TEXT) WITH (enable_tde = on);
CREATE TABLE
```

步骤2 记录xlog的写入位置。

```
gaussdb=# SELECT pg_xlogfile_name_offset((SELECT pg_current_xlog_location()));
     pg_xlogfile_name_offset
-----------------------------------
 (00000001000000000000000D,3121400)
(1 row)
```

步骤3 操作含TDE属性的表，生成携带数据的xlog。

```
INSERT INTO t1 VALUES (1, 'sssssssssssssss');
```

步骤4 验证数据在xlog中已被加密。

```
gaussdb=#  \! hexdump -C $DATADIR/pg_xlog/00000001000000000000000D -s 3121400 -n 2048
002fa0f8  5e 00 00 80 01 00 00 00  00 00 00 00 00 00 00 00  |^...............|
002fa108  d8 9c 2f 0d 00 00 00 00  e0 16 00 00 00 00 00 00  |../.............|
002fa118  26 74 d4 9e ff 38 b8 17  00 00 00 00 00 00 d8 0e  |&t...8..........|
002fa128  00 00 00 00 00 00 52 31  01 00 00 00 00 00 6b 26  |......R1......k&|
002fa138  06 00 00 00 00 00 7a 33  01 00 00 00 00 00 7b 33  |......z3......{3|
002fa148  01 00 00 00 00 00 f0 3d  2f 0d 00 00 00 00 00 00  |.......=/.......|
002fa158  3e 00 00 80 01 00 00 00  ab 33 01 00 00 00 00 00  |>........3......|
002fa168  f8 a0 2f 0d 00 00 00 00  80 16 00 00 00 00 00 00  |../.............|
002fa178  14 79 79 89 ff 18 00 00  00 00 00 10 00 00 00 00  |.yy.............|
002fa188  10 00 00 10 00 00 00 00  00 00 00 00 00 00 00 00  |................|
002fa198  3e 00 00 80 01 00 00 00  ab 33 01 00 00 00 00 00  |>........3......|
002fa1a8  58 a1 2f 0d 00 00 00 00  b0 16 00 00 00 00 00 00  |X./.............|
002fa1b8  6d 09 43 cc ff 18 00 00  00 00 00 10 00 00 00 80  |m.C.............|
002fa1c8  00 00 00 10 00 00 00 00  00 00 00 00 00 00 00 00  |................|
002fa1d8  13 02 00 80 01 00 00 00  ab 33 01 00 00 00 00 00  |.........3......|
002fa1e8  98 a1 2f 0d 00 00 00 00  80 14 00 00 00 00 00 80  |../.............|
002fa1f8  12 61 3e 44 40 60 1d 00  7f 06 00 00 13 32 00 00  |.a>D@`.......2..|
002fa208  57 41 00 00 25 94 24 f9  f6 8d 46 50 8a f7 3f 92  |WA..%.$...FP..?.|
002fa218  98 08 2c 57 a2 fc 16 3d  18 dc 6f 67 9e 8a c8 ba  |..,W...=..og....|
002fa228  a3 7b 58 3f ba 50 d9 6c  52 d2 91 01 75 a7 a4 d9  |.{X?.P.lR...u...|
002fa238  05 86 2d b1 10 aa 1c 46  14 7f 00 00 d3 3a 4a 22  |..-....F.....:J"|
002fa248  e0 55 00 00 7f 06 00 00  09 00 00 01 00 35 a8 44  |.U...........5.D|
002fa258  00 7f 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

步骤5 删除t1表数据，生成undo-log，判断undo文件已被加密。

预期：无法搜索到数据明文

```
gaussdb=# DELETE FROM t1;
DELETE 1
grep -c 'sssssssssssssssssssssss' $DATADIR/undo/permanent/*
```

步骤6 删除加密表。

```
gaussdb=# DROP TABLE IF EXISTS t1;
DROP TABLE
```

----结束

 版权所有 © 华为云计算技术有限公司

使用 third_kms 三方厂商加解密库

透明加密特性需要使用third_kms三方厂商加解密库前需要按照以下操作步骤进行配置：

步骤1 向第三方厂商获取名字为libthird_crypto.so的加解密文件及其使用说明。

步骤2 将加解密文件放置在$GAUSSHOME下的lib目录中，给予数据库进程可读权限。

步骤3 管理员设置GUC参数crypto_module_info和tde_key_info。

其中module_third_msg为加密库加载时的必要信息，keyThirdMsg为加密库创建会话时的登录信息，这两项信息的格式和内容由第三方厂商定义，数据库仅通过crypto_module_init接口对该参数的合法性进行校验，建议第三方厂商通过该参数校验对应版本号的前后向兼容性。

```
gs_guc set -D $DATA_PATH -c "enable_tde=on" # 透明加密开关
gs_guc set -D $DATA_PATH -c "crypto_module_info='enable_crypto_module=on, module_third_msg=aaa'"
gs_guc set -D $DATA_PATH -c "tde_key_info='keyType=third_kms, keyThirdMsg =aaa'"
```

步骤4 重启数据库。

步骤5 创建透明加密表及其列索引。

```
-- CREATE TABLE/INDEX语法中dek_token 为可选入参，可以用以传入数据密钥ID（hex格式）。不填则通过三方库密钥生成接口自动生成随机密钥。
gaussdb=# CREATE TABLE t1 (c1 INT, c2 TEXT) WITH (enable_tde = on, dek_token
='29a6b6ab63efa4687b4c62acb3746b90');
CREATE TABLE
-- 可以通过ALTER SET语句设置新的密钥ID。
gaussdb=# ALTER TABLE t1 SET(dek_token = '29a6b6ab63efa4687b4c62acb3746b95');
ALTER TABLE
-- 可以通过ALTER ENCRYPTION KEY ROTATION语句调取三方库密钥生成接口，自动生成新的随机密钥。
gaussdb=# ALTER TABLE t1 ENCRYPTION KEY ROTATION;
ALTER TABLE
gaussdb=# CREATE INDEX idx1 ON t1 USING ubtree(c2) WITH (enable_tde=on);
CREATE INDEX
```

----结束

第三方厂商实现加解密接口注意事项

- 三方厂商需按照**第三方厂商实现加解密接口头文件**对接口进行实现并编译成以libthird_crypto.so命名的文件。
- 加密库链接时，仅可使用与数据库进程中相同版本的依赖库，不允许引入外部依赖库。
- 厂商提供加密库给客户时需保证编译环境与客户环境一致（例如：x86 or ARM，os-release版本等）。
- 在**第三方厂商实现加解密接口头文件**中，crypto_module_init的返回值SupportedFeature需要支持除MODULE_DETERMINISTIC_KEY外的所有算法，需要实现除my_cipher_deterministic_enc_dec外所有接口。（不实现确定性算法及其接口仅导致全密态等值查询不可使用该库，仍可供透明加密功能使用）
- 为保证兼容性，若涉及接口头文件升级，厂商需要基于升级后的头文件重新编译C函数共享库。
- 如果实现语言为C++，则需要为实现的函数加上extern "C"，以保证数据库加载时能正确找到函数。
- 编译选项需要带上安全编译选项和优化选项，如trapv、Wl、relro、z、now、noexecstack、pie、PIC、stack-protector、O3等。

版权所有 © 华为云计算技术有限公司

⚠ 注意

厂商应充分了解要实现的接口功能及入参范围，在调用前应检查参数合法性，避免出现空指针等可能导致程序crash的问题。厂商应避免接口调用过程中产生内存泄漏。

第三方厂商实现加解密接口头文件

```
#define CRYPT_MOD_OK 1
#define CRYPT_MOD_ERR 0
typedef enum {
    MODULE_AES_128_CBC = 0,
    MODULE_AES_128_CTR,
    MODULE_AES_128_GCM,
    MODULE_AES_256_CBC,
    MODULE_AES_256_CTR,
    MODULE_AES_256_GCM,
    MODULE_SM4_CBC,
    MODULE_SM4_CTR,
    MODULE_HMAC_SHA256,
    MODULE_HMAC_SM3,
    MODULE_DETERMINISTIC_KEY,
    MODULE_ALGO_MAX = 1024
} ModuleSymmKeyAlgo;
typedef enum {
    MODULE_SHA256 = 0,
    MODULE_SM3,
    MODULE_DIGEST_MAX = 1024
} ModuleDigestAlgo;
typedef enum {
    KEY_TYPE_INVALID,
    KEY_TYPE_PLAINTEXT,
    KEY_TYPE_CIPHERTEXT,
    KEY_TYPE_NAMEORIDX,
    KEY_TYPE_MAX
} KeyType;
typedef struct {
    KeyType key_type;
    int supported_symm[MODULE_ALGO_MAX]; // 不支持算法填入0或者支持算法填入1
    int supported_digest[MODULE_DIGEST_MAX]; // 不支持算法填入0或者支持算法填入1
} SupportedFeature;
/** 初始化密码模块
 *
 * @param[in]
 *    load_info            密码模块相关信息（硬件设备密码，硬件设备、硬件库路径等），通过kv方式传入
 *
 * @param[out]
 *    supported_feature    返回当前密码模块支持的加密方式，参考上述结构体
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_module_init(char *load_info, SupportedFeature *supported_feature);
/** 会话中连接密码模块
 *
 * @param[in]
 *    key_info        密码相关信息（用户密码等信息），通过kv方式传入
 *
 * @param[out]
 *    sess            会话信息
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_module_sess_init(char *key_info, void **sess);
/** 会话中断开连接密码模块
 *
 * @param[in]
 *    sess            会话信息
```


版权所有 © 华为云计算技术有限公司

```
 *
 * @param[out]
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
void crypto_module_sess_exit(void *sess);
/** 创建密钥
 *
 * @param[in]
 *    sess            会话信息
 *    algo            密钥使用场景的算法
 *
 * @param[out]
 *    key_id          返回生成密钥/密钥ID/密钥密文
 *    key_id_size     返回生成内容长度
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_create_symm_key(void *sess, ModuleSymmKeyAlgo algo, unsigned char *key_id, size_t
*key_id_size);
/** 密钥上下文初始化，后续进行加解密可直接使用上下文
 *
 * @param[in]
 *    sess            会话信息
 *    algo            加密算法
 *    enc             加密1、解密0
 *    key_id          密码信息
 *    key_id_size     密码信息长度
 * @param[out]
 *    ctx             返回使用密钥信息
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_ctx_init(void *sess, void **ctx, ModuleSymmKeyAlgo algo, int enc, unsigned char *key_id, size_t
key_id_size);
/** 获取数据加解密后的数据长度
 *
 * @param[in]
 *    ctx             加解密上下文信息
 *    enc             加密1、解密0
 * @param[out]
 *    data_size       返回加解密结果长度
 * @return    成功返回数据长度，失败返回-1
 *
 */
int crypto_result_size(void *ctx, int enc, size_t data_size);
/** 密钥上下文清理
 *
 * @param[in]
 *    ctx             加解密上下文信息
 * @param[out]
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
void crypto_ctx_clean(void *ctx);
/** 执行加解密
 *
 * @param[in]
 *    ctx             加解密上下文信息
 *    enc             加密1、解密0
 *    data            原数据信息
 *    data_size       原数据长度
 *    iv              iv信息
 *    iv_size         iv信息长度
 *    tag             GCM模式的校验值
 * @param[out]
 *    result          返回结果信息
```

 版权所有 © 华为云计算技术有限公司

```
 *    result_size     返回结果信息长度
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_encrypt_decrypt(void *ctx, int enc, unsigned char *data, size_t data_size, unsigned char *iv, size_t
iv_size, unsigned char *result, size_t result_size, unsigned char *tag);
/** 计算摘要
 *
 * @param[in]
 *    sess            会话信息
 *    algo            摘要算法
 *    data            原数据信息
 *    data_size       原数据长度
 * @param[out]
 *    result          返回结果信息
 *    result_size     返回结果信息长度
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_digest(void *sess, ModuleDigestAlgo algo, unsigned char * data, size_t data_size,unsigned char
*result, size_t *result_size);
/** hmac初始化
 *
 * @param[in]
 *    sess            会话信息
 *    algo            摘要算法
 *    key_id          密码信息
 *    key_id_size     密码信息长度
 * @param[out]
 *    ctx             返回密钥上下文
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_hmac_init(void *sess, void **ctx, ModuleDigestAlgo algo, unsigned char *key_id, size_t key_id_size);
/** hmac清理
 *
 * @param[in]
 *    ctx             密钥上下文信息
 *
 * @param[out]
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
void crypto_hmac_clean(void *ctx);
/** 执行hmac计算
 *
 * @param[in]
 *    ctx             密钥上下文信息
 *    data            原数据信息
 *    data_size       原数据长度
 * @param[out]
 *    result          返回结果信息
 *    result_size     返回结果信息长度
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_hmac(void *ctx, unsigned char * data, size_t data_size, unsigned char *result, size_t *result_size);
/** 生成随机数
 *
 * @param[in]
 *    sess            会话信息
 *    size            申请的随机信息长度
 *
 * @param[out]
```

 版权所有 © 华为云计算技术有限公司

```
 *    buffer          返回随机信息
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_gen_random(void *sess, char *buffer, size_t size);
/** 执行确定性加解密
 *
 * @param[in]
 *    sess            会话信息
 *    enc             加密1、解密0
 *    data            原数据信息
 *    data_size       原数据长度
 *    key_id          密钥信息
 *    key_id_size     密钥信息长度
 * @param[out]
 *    result          返回结果信息
 *    result_size     返回结果信息长度
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_deterministic_enc_dec(void *sess, int enc, unsigned char *data, unsigned char *key_id,
    size_t key_id_size, size_t data_size, unsigned char *result, size_t *result_size);
/** 获取报错信息
 *
 * @param[in]
 *    sess            会话信息
 * @param[out]
 *    errmsg          返回结果信息,最长256字节
 *
 * @return    成功返回CRYPT_MOD_OK，失败返回错误码
 *
 */
int crypto_get_errmsg(void *sess, char *errmsg);
```

 版权所有 © 华为云计算技术有限公司

5 设置账本数据库

5.1 账本数据库概述

背景信息

账本数据库融合了区块链思想，将用户操作记录至两种历史表：用户历史表和全局区块表中。当用户创建防篡改用户表时，系统将自动为该表添加一个hash列来保存每行数据的hash摘要信息，同时在blockchain模式下会创建一张用户历史表来记录对应用户表中每条数据的变更行为；而用户对防篡改用户表的每一次修改行为将被记录到全局区块表中。由于历史表具有只可追加不可修改的特点，因此历史表记录串联起来便形成了用户对防篡改用户表的修改历史。

用户历史表命名和结构如下：

表 5-1 用户历史表 blockchain.<schemaname>_<tablename>_hist 所包含的字段

字段名	类型	描述
rec_num	bigint	行级修改操作在历史表中的执行序号。
hash_ins	hash16	INSERT或UPDATE操作插入的数据行的hash值。
hash_del	hash16	DELETE或UPDATE操作删除数据行的hash值。
pre_hash	hash32	当前用户历史表的数据整体摘要。

表 5-2 hash_ins 与 hash_del 场景对应关系

-	hash_ins	hash_del
INSERT	(√) 插入行的hash值。	空

版权所有 © 华为云计算技术有限公司

-	hash_ins	hash_del
DELETE	空	(√) 删除行的hash值。
UPDATE	(√) 新插入数据的hash值。	(√) 删除前该行的hash值。

操作步骤

步骤1 创建防篡改模式。

例如，创建防篡改模式ledgernsp。

```
gaussdb=# CREATE SCHEMA ledgernsp WITH BLOCKCHAIN;
```

说明

如果需要创建防篡改模式或更改普通模式为防篡改模式，则需设置enable_ledger参数为on。enable_ledger默认参数为off。

步骤2 在防篡改模式下创建防篡改用户表。

例如，创建防篡改用户表ledgernsp.usertable。

```
gaussdb=# CREATE TABLE ledgernsp.usertable(id int, name text);
```

查看防篡改用户表结构及其对应的用户历史表结构。

```
gaussdb=# \d+ ledgernsp.usertable;
gaussdb=# \d+ blockchain.ledgernsp_usertable_hist;
```

执行结果如下：

```
gaussdb=# \d+ ledgernsp.usertable;
                    Table "ledgernsp.usertable"
 Column     |  Type   | Modifiers | Storage  | Stats target | Description
--------+---------+-----------+----------+--------------+-------------
 id         | integer |           | plain    |              |
 name       | text    |           | extended |              |
 hash_69dd43   | hash16  |           | plain    |              |
Has OIDs: no
Distribute By: HASH(id)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no
History table name: ledgernsp_usertable_hist

gaussdb=# \d+ blockchain.ledgernsp_usertable_hist;
             Table "blockchain.ledgernsp_usertable_hist"
  Column  |  Type  | Modifiers | Storage | Stats target | Description
----------+--------+-----------+---------+--------------+-------------
 rec_num  | bigint |           | plain   |              |
 hash_ins | hash16 |           | plain   |              |
 hash_del | hash16 |           | plain   |              |
 pre_hash | hash32 |           | plain   |              |
Indexes:
    "gs_hist_69dd43_index" PRIMARY KEY, btree (rec_num int4_ops) TABLESPACE pg_default
Has OIDs: no
Distribute By: HASH(rec_num)
Location Nodes: ALL DATANODES
Options: internal_mask=263
```


版权所有 © 华为云计算技术有限公司

说明

1. 防篡改模式下仅行存表为防篡改表，临时表、外表、unlog表及非行存表均无防篡改属性。
2. 防篡改表在创建时会自动增加一个用于校验的系统列，所以防篡改表单表最大列数为1599。

步骤3 修改防篡改用户表数据。

例如，对防篡改用户表执行INSERT、UPDATE、DELETE操作。

```
gaussdb=# INSERT INTO ledgernsp.usertable VALUES(1, 'alex'), (2, 'bob'), (3, 'peter');
INSERT 0 3
gaussdb=# SELECT *, hash_69dd43 FROM ledgernsp.usertable ORDER BY id;
 id | name  |        hash_69dd43
----+-------+------------------
  1 | alex  | 1f2e543c580cb8c5
  2 | bob   | 8fcd74a8a6a4b484
  3 | peter | f51b4b1b12d0354b
(3 rows)

gaussdb=# UPDATE ledgernsp.usertable SET name = 'bob2' WHERE id = 2;
UPDATE 1
gaussdb=# SELECT *, hash_69dd43 FROM ledgernsp.usertable ORDER BY id;
 id | name  |        hash_69dd43
----+-------+------------------
  1 | alex  | 1f2e543c580cb8c5
  2 | bob2  | 437761affbb7c605
  3 | peter | f51b4b1b12d0354b
(3 rows)

gaussdb=# DELETE FROM ledgernsp.usertable WHERE id = 3;
DELETE 1
gaussdb=# SELECT *, hash_69dd43 FROM ledgernsp.usertable ORDER BY id;
 id | name |        hash_69dd43
----+------+------------------
  1 | alex | 1f2e543c580cb8c5
  2 | bob2 | 437761affbb7c605
(2 rows)
```

步骤4 删除表和模式。

若要执行其他账本数据库章节的示例，请在执行完之后再执行当前步骤，否则请直接执行当前步骤。

```
gaussdb=# DROP TABLE ledgernsp.usertable;
DROP TABLE
gaussdb=# DROP SCHEMA ledgernsp;
DROP SCHEMA
```

----结束

5.2 查看账本历史操作记录

前提条件

- 系统中需要有审计管理员或者具有审计管理员权限的角色。
- 数据库正常运行，并且对防篡改数据库执行了一系列增、删、改等操作，保证在查询时段内有账本操作记录结果产生。
- 数据库各个CN节点全局区块表记录单独记录，全局区块表只能记录连接到当前CN执行的SQL操作。

版权所有 © 华为云计算技术有限公司

背景信息

- 只有拥有AUDITADMIN属性的用户才可以查看账本历史操作记录。有关数据库用户及创建用户的办法请参见《开发指南》中“数据库安全 > 用户及权限 > 用户”章节。
- 查询全局区块表命令是直接查询gs_global_chain表，操作为：

  ```
  SELECT * FROM gs_global_chain;
  ```

  该表有10个字段，每个字段的含义请参见《开发指南》中“系统表和系统视图 > 系统表 > GS_GLOBAL_CHAIN”章节。
- 查询用户历史表的命令是直接查询BLOCKCHAIN模式下的用户历史表，操作为：

  例如用户表所在的模式为ledgernsp，表名为usertable，则对应的用户历史表名为blockchain.ledgernsp_usertable_hist：

  ```
  SELECT * FROM blockchain.ledgernsp_usertable_hist;
  ```

  用户历史表有4个字段，每个字段的含义请参见**表5-1**。

  说明

  用户历史表的表名一般为blockchain.<schemaname>_<tablename>_hist形式。当防篡改用户表模式名或者表名过长导致前述方式生成的表名超出表名长度限制，则会采用blockchain.<schema_oid>_<table_oid>_hist的方式命名。

操作步骤

步骤1 查询全局区块表记录。

```
gaussdb=# SELECT * FROM gs_global_chain;
```

查询结果如下：

```
 blocknum |  dbname  | username |          starttime          | relid |  relnsp   |  relname  |     relhash
|           globalhash           |
              txcommand
----------+----------+----------+-----------------------------+-------+-----------+-----------+-----------------
+----------------------------------+------------------
-------------------------------------------------------------
        0 | testdb | omm      | 2021-04-14 07:00:46.32757+08  | 16393 | ledgernsp | usertable |
a41714001181a294 | 6b5624e039e8aee36bff3e8295c75b40 | insert into ledge
rnsp.usertable values(1, 'alex'), (2, 'bob'), (3, 'peter');
        1 | testdb | omm      | 2021-04-14 07:01:19.767799+08 | 16393 | ledgernsp | usertable |
b3a9ed0755131181 | 328b48c4370faed930937869783c23e0 | update ledgernsp.
usertable set name = 'bob2' where id = 2;
        2 | testdb | omm      | 2021-04-14 07:01:29.896148+08 | 16393 | ledgernsp | usertable |
0ae4b4e4ed2fcab5 | aa8f0a236357cac4e5bc1648a739f2ef | delete from ledge
rnsp.usertable where id = 3;
```

该结果表明，用户omm连续执行了三条DML命令，包括INSERT、UPDATE和DELETE操作。

步骤2 查询历史表记录。

```
gaussdb=# SELECT * FROM blockchain.ledgernsp_usertable_hist;
```

查询结果如下：

```
 rec_num |     hash_ins     |     hash_del     |            pre_hash
---------+------------------+------------------+----------------------------------
       0 | 1f2e543c580cb8c5 |                  | e1b664970d925d09caa295abd38d9b35
       1 | 8fcd74a8a6a4b484 |                  | dad3ed8939a141bf3682043891776b67
       2 | f51b4b1b12d0354b |                  | 53eb887fc7c4302402343c8914e43c69
       3 | 437761affbb7c605 | 8fcd74a8a6a4b484 | c2868c5b49550801d0dbbbaa77a83a10
       4 |                  | f51b4b1b12d0354b | 9c512619f6ffef38c098477933499fe3
(5 rows)
```


版权所有 © 华为云计算技术有限公司

查询结果显示，用户omm对ledgernsp.usertable表插入了3条数据，更新了1条数据，随后删除了1行数据，最后剩余2行数据，hash值分别为1f2e543c580cb8c5和437761affbb7c605。

步骤3 查询用户表数据及校验列。

```
gaussdb=# SELECT *, hash_69dd43 FROM ledgernsp.usertable;
```

查询结果如下：

```
 id | name |       hash_69dd43
----+------+------------------
  1 | alex | 1f2e543c580cb8c5
  2 | bob2 | 437761affbb7c605
(2 rows)
```

查询结果显示，用户表中剩余2条数据，与**2**中的记录一致。

----结束

5.3 校验账本数据一致性

前提条件

数据库正常运行，并且对防篡改数据库执行了一系列增、删、改等操作，保证在查询时段内有账本操作记录结果产生。

背景信息

- 账本数据库校验功能目前提供两种校验接口，分别为：ledger_hist_check(text, text)和ledger_gchain_check(text, text)。普通用户调用校验接口，仅能校验自己有权限访问的表。
- 校验防篡改用户表和用户历史表的接口为pg_catalog.ledger_hist_check，操作为：

  ```
  SELECT pg_catalog.ledger_hist_check(schema_name text,table_name text);
  ```

  如果校验通过，函数返回t，反之则提示失败原因并返回f。
- 校验防篡改用户表、用户历史表和全局区块表三者是否一致的接口为pg_catalog.ledger_gchain_check，操作为：

  ```
  SELECT pg_catalog.ledger_gchain_check(schema_name text, table_name text);
  ```

  如果校验通过，函数返回t，反之则提示失败原因并返回f。

操作步骤

步骤1 校验防篡改用户表ledgernsp.usertable与其对应的历史表是否一致。

```
gaussdb=# SELECT pg_catalog.ledger_hist_check('ledgernsp', 'usertable');
```

查询结果如下：

```
 ledger_hist_check
-------------------
 t
(1 row)
```

该结果表明防篡改用户表和用户历史表中记录的结果能够一一对应，保持一致。

步骤2 查询防篡改用户表ledgernsp.usertable与其对应的历史表以及全局区块表中关于该表的记录是否一致。

```
gaussdb=# SELECT pg_catalog.ledger_gchain_check('ledgernsp', 'usertable');
```


版权所有 © 华为云计算技术有限公司

查询结果如下：

```
 ledger_gchain_check
---------------------
 t
(1 row)
```

查询结果显示，上述三表中关于ledgernsp.usertable的记录保持一致，未发生篡改行为。

----结束

5.4 归档账本数据库

前提条件

- 系统中需要有审计管理员或者具有审计管理员权限的角色。
- 数据库正常运行，并且对防篡改数据库执行了一系列增、删、改等操作，保证在查询时段内有账本操作记录结果产生。
- 数据库已经正确配置审计文件的存储路径audit_directory。

背景信息

- 账本数据库归档功能目前提供两种校验接口，分别为：ledger_hist_archive(text, text)和ledger_gchain_archive(text, text)。账本数据库接口仅审计管理员可以调用。
- 归档用户历史表的接口为pg_catalog.ledger_hist_archive，表示归档当前DN的用户历史表数据。执行操作为：

  ```
  SELECT pg_catalog.ledger_hist_archive(schema_name text,table_name text);
  ```

  如果归档成功，函数返回t，反之则提示失败原因并返回f。
- 归档全局区块表的接口为pg_catalog.ledger_gchain_archive，表示归档当前CN的全局历史表数据。执行操作为：

  ```
  SELECT pg_catalog.ledger_gchain_archive();
  ```

  如果归档成功，函数返回t，反之则提示失败原因并返回f。

操作步骤

步骤1 使用EXECUTE DIRECT对某个DN节点进行归档操作。

```
gaussdb=# EXECUTE DIRECT ON (datanode1) 'select pg_catalog.ledger_hist_archive(''ledgernsp'', ''usertable'');';
```

查询结果如下：

```
 ledger_hist_archive
---------------------
 t
(1 row)
```

用户历史表将归档为一条数据：

```
gaussdb=# EXECUTE DIRECT ON (datanode1) 'SELECT * FROM blockchain.ledgernsp_usertable_hist;';
 rec_num |     hash_ins     |     hash_del     |             pre_hash
---------+------------------+------------------+----------------------------------
       3 | e78e75b00d396899 | 8fcd74a8a6a4b484 | fd61cb772033da297d10c4e658e898d7
(1 row)
```

该结果表明datanode1节点用户历史表导出成功。

版权所有 © 华为云计算技术有限公司

步骤2 连接CN执行全局区块表导出操作。

```
gaussdb=# SELECT pg_catalog.ledger_gchain_archive();
```

查询结果如下：

```
 ledger_gchain_archive
-----------------------
 t
(1 row)
```

全局历史表将以用户表为单位归档为N（用户表数量）条数据：

```
gaussdb=# SELECT * FROM gs_global_chain;
 blocknum |  dbname  | username |           starttime           | relid |  relnsp   |  relname  |     relhash
|            globalhash            | txcommand
----------+----------+----------+-------------------------------+-------+-----------+-----------+------------------
+----------------------------------+-----------
        1 | testdb | libc     | 2021-05-10 19:59:38.619472+08 | 16388 | ledgernsp | usertable |
57c101076694b415 | be82f98ee68b2bc4e375f69209345406 | Archived.
(1 row)
```

该结果表明，当前coordinator节点全局区块表导出成功。

----结束

5.5 修复账本数据库

前提条件

- 系统中需要有审计管理员或者具有审计管理员权限的角色。
- 数据库正常运行，并且对防篡改数据库执行了一系列增、删、改等操作，保证在查询时段内有账本操作记录结果产生。

背景信息

- 当前的账本数据库机制为：全局区块表存储在CN端，各个CN数据独立。用户历史表存储在DN端，历史表记录的数据为所在DN防篡改表的数据变化。因此，在触发数据重分布时，可能导致防篡改表和用户历史表数据不一致，此时需要使用ledger_hist_repair(text, text)接口对指定DN节点的用户历史表进行修复，修复后当前DN节点调用历史表校验接口结果为true。在CN剔除、修复的场景下，可能导致全局区块表数据丢失或者与用户历史表不一致，此时需要使用ledger_gchain_repair(text, text)接口对整个集群范围内的全局区块表进行修复，修复后调用全局区块表校验接口结果为true。
- 修复用户历史表的接口为pg_catalog.ledger_hist_repair，操作为：

  ```
  SELECT pg_catalog.ledger_hist_repair(schema_name text,table_name text);
  ```

  如果修复成功，函数返回修复过程中用户历史表hash的增量。

  注：对用户表执行闪回DROP时，可使用该函数恢复用户表和用户历史表名称，请参见**恢复用户表和用户历史表名称**。
- 修复全局区块表的接口为pg_catalog.ledger_gchain_repair，操作为：

  ```
  SELECT pg_catalog.ledger_gchain_repair(schema_name text,table_name text);
  ```

  如果修复成功，函数返回修复过程中全局区块表中指定表的hash总和。

恢复用户表数据和全局区块表数据

以omm用户为例进行操作，步骤如下。

版权所有 © 华为云计算技术有限公司

步骤1 以操作系统用户omm登录数据库主节点。

步骤2 使用EXECUTE DIRECT对某个DN节点进行历史表修复操作。

```
gaussdb=# EXECUTE DIRECT ON (datanode1) 'select pg_catalog.ledger_hist_repair(''ledgernsp'', ''usertable'');';
```

查询结果如下：

```
 ledger_hist_repair
--------------------
 84e8bfc3b974e9cf
(1 row)
```

该结果表明datanode1节点用户历史表修复成功，修复造成的用户历史表hash增量为84e8bfc3b974e9cf。

步骤3 连接CN执行全局区块表修复操作。

```
gaussdb=# SELECT pg_catalog.ledger_gchain_repair('ledgernsp', 'usertable');
```

查询结果如下：

```
 ledger_gchain_repair
----------------------
 a41714001181a294
(1 row)
```

该结果表明，当前集群全局区块表修复成功，且向当前CN节点插入一条修复数据，其hash值为a41714001181a294。

----结束

恢复用户表和用户历史表名称

已通过enable_recyclebin参数和recyclebin_retention_time参数开启闪回DROP功能，恢复用户表和用户历史表名称。示例如下：

- DROP用户表，对用户表执行闪回DROP。使用ledger_hist_repair对用户表、用户历史表进行表名恢复。

```
-- 对用户表执行闪回drop，使用ledger_hist_repair对用户历史表进行表名恢复。
gaussdb=# CREATE TABLE ledgernsp.tab2(a int, b text);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'a' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'rec_num' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
gaussdb=# DROP TABLE ledgernsp.tab2;
DROP TABLE
gaussdb=# SELECT rcyrelid, rcyname, rcyoriginname FROM gs_recyclebin;
 rcyrelid |           rcyname            |    rcyoriginname
----------+------------------------------+---------------------
    32838 | BIN$39B523388046$55C8400==$0 | tab2
    32846 | BIN$39B52338804E$55C90E8==$0 | gs_hist_tab2_index
    32843 | BIN$39B52338804B$55C96A0==$0 | ledgernsp_tab2_hist
    32841 | BIN$39B523388049$55C9EE0==$0 | pg_toast_32838
(4 rows)
-- 对用户表执行闪回drop。
gaussdb=# TIMECAPSULE TABLE ledgernsp.tab2 TO BEFORE DROP;
TimeCapsule Table
-- 使用ledger_hist_repair恢复用户历史表表名。
gaussdb=# SELECT ledger_hist_repair('ledgernsp', 'tab2');
 ledger_hist_repair
--------------------
 0000000000000000
(1 row)
```

 版权所有 © 华为云计算技术有限公司

```
gaussdb=# \d+ ledgernsp.tab2;
                        Table "ledgernsp.tab2"
   Column    |  Type   | Modifiers | Storage  | Stats target | Description
-------------+---------+-----------+----------+--------------+-------------
 a           | integer |           | plain    |              |
 b           | text    |           | extended |              |
 hash_1d2d14 | hash16  |           | plain    |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=enhanced_toast
History table name: ledgernsp_tab2_hist

-- 对用户表执行闪回drop，使用ledger_hist_repair对用户表进行表名恢复。
gaussdb=# CREATE TABLE ledgernsp.tab3(a int, b text);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'a' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'rec_num' as the distribution column by
default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
gaussdb=# DROP TABLE ledgernsp.tab3;
DROP TABLE
gaussdb=# SELECT rcyrelid, rcyname, rcyoriginname FROM gs_recyclebin;
 rcyrelid |            rcyname             |    rcyoriginname
----------+--------------------------------+---------------------
    32952 | BIN$80B6233880B8$FECFF98==$0 | tab3
    32960 | BIN$80B6233880C0$FED0C98==$0 | gs_hist_tab3_index
    32957 | BIN$80B6233880BD$FED1250==$0 | ledgernsp_tab3_hist
    32955 | BIN$80B6233880BB$FED1A00==$0 | pg_toast_32952
(4 rows)
-- 对用户历史表执行闪回drop。
gaussdb=# TIMECAPSULE TABLE blockchain.ledgernsp_tab3_hist TO BEFORE DROP;
TimeCapsule Table
-- 拿到回收站中用户表对应的rcyname，使用ledger_hist_repair恢复用户表表名。
gaussdb=# SELECT ledger_hist_repair('ledgernsp', 'BIN$80B6233880B8$FECFF98==$0');
 ledger_hist_repair
--------------------
 0000000000000000
(1 row)

gaussdb=# \d+ ledgernsp.tab3;
                        Table "ledgernsp.tab3"
   Column    |  Type   | Modifiers | Storage  | Stats target | Description
-------------+---------+-----------+----------+--------------+-------------
 a           | integer |           | plain    |              |
 b           | text    |           | extended |              |
 hash_7a0c87 | hash16  |           | plain    |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=enhanced_toast
History table name: ledgernsp_tab3_hist

-- 删除表。
gaussdb=# DROP TABLE ledgernsp.tab2 PURGE;
DROP TABLE
gaussdb=# DROP TABLE ledgernsp.tab3 PURGE;
DROP TABLE
```

 版权所有 © 华为云计算技术有限公司

6 基于标签的强制访问控制

6.1 定义安全标签

前提条件

数据库正常运行，执行操作的用户需要具有SYSADMIN权限或者继承了内置角色gs_role_seclabel的权限。

背景信息

提供系统表gs_security_label来查看系统中已创建好的安全标签：

```
SELECT * FROM gs_security_label;
```

操作步骤

以某个实际业务应用举例说明，比如某公司的信息等级为公开、秘密和绝密，那么就可分别对应成等级L1、L2和L3。信息存储范围为一层到五层各层员工的信息，那么可分别对应成范围G1、G2、G3、G4、G5，以此为基础定义安全标签。

步骤1 创建一个安全标签，标志信息等级为公开，范围为二层和四层。

```
gaussdb=# CREATE SECURITY LABEL label1 'L1:G2,G4';
```

步骤2 创建一个安全标签，标志信息等级为秘密，范围为二层到四层。

```
gaussdb=# CREATE SECURITY LABEL label2 'L2:G2-G4';
```

步骤3 创建一个安全标签，标志信息等级为绝密，范围为一层到五层。

```
gaussdb=# CREATE SECURITY LABEL label3 'L3:G1-G5';
```

步骤4 查询系统表gs_security_label。

```
gaussdb=# SELECT * FROM gs_security_label;
```

查询结果如下：

```
gaussdb=# SELECT * FROM gs_security_label;
 label_name | label_content
------------+---------------
 label1     | L1:G2,G4
 label2     | L2:G2-G4
 label3     | L3:G1-G5
(3 rows)
```

 版权所有 © 华为云计算技术有限公司

步骤5 删除安全标签。

若要执行其他安全标签章节的示例，请在执行完之后再执行当前步骤，否则请直接执行当前步骤。

```
gaussdb=# DROP SECURITY LABEL label1;
gaussdb=# DROP SECURITY LABEL label2;
gaussdb=# DROP SECURITY LABEL label3;
```

----结束

6.2 应用安全标签

前提条件

数据库正常运行，执行操作的用户需要具有SYSADMIN权限或者继承了内置角色gs_role_seclabel的权限。

背景信息

- 提供系统表pg_seclabel来查看表或表列上的安全标签：
  ```
  SELECT * FROM pg_seclabel;
  ```
- 提供系统表pg_shseclabel来查看用户/角色上的安全标签：
  ```
  SELECT * FROM pg_shseclabel;
  ```
- 提供系统视图pg_seclabels来查看系统中所有的安全标签记录：
  ```
  SELECT * FROM pg_seclabels;
  ```

操作步骤

数据库中已有用户user1、user2和数据表tbl。

步骤1 对用户user1应用安全标签label1，表示用户user1对应的信息等级为公开，范围为二层和四层。

```
gaussdb=# SECURITY LABEL ON USER user1 is 'label1';
```

步骤2 对用户user2应用安全标签label3，表示用户user2对应的信息等级为绝密，范围为一层到五层。

```
gaussdb=# SECURITY LABEL ON USER user2 is 'label3';
```

步骤3 对表tbl应用安全标签label2，表示表tbl对应的信息等级为秘密，范围为二层到四层。

```
gaussdb=# SECURITY LABEL ON TABLE tbl is 'label2';
```

步骤4 查询系统视图pg_seclabels。

```
gaussdb=# SELECT * FROM pg_seclabels;
```

查询结果如下：

```
gaussdb=# SELECT * FROM pg_seclabels;
 objoid | classoid | objsubid | objtype | objnamespace | objname | provider | label
--------+----------+----------+---------+--------------+---------+----------+--------
  16399 |     1259 |        0 | table   |         2200 | tbl     | maclabel | label2
  16391 |     1260 |        0 | role    |              | user1   | maclabel | label1
  16395 |     1260 |        0 | role    |              | user2   | maclabel | label3
(3 rows)
```

----结束

版权所有 © 华为云计算技术有限公司

6.3 基于标签的强制访问控制检查

前提条件

- 数据库正常运行，强制访问控制检查开关（enable_mac_check=on）打开。
- 执行操作的用户已具有所需要访问表的ACL权限。

背景信息

基于安全标签的强制访问控制策略规则由系统内置设定，用户不能更改：

- 插入（INSERT）策略：只有主体安全标记等级小于等于客体安全标记等级且主体安全标记范围是客体安全标记范围的子集时才允许插入数据。
- 查询（SELECT）策略：只有主体安全标记等级大于等于客体安全标记等级且主体安全标记范围是客体安全标记范围的超集时才允许查询数据。
- 修改（UPDATE）和删除（DELETE）策略：只有主体安全标记等级等于客体安全标记等级且主体安全标记范围等于客体安全标记范围时才允许修改和删除数据。
- 若客体未标记，则强制访问控制策略对该客体不生效。
- 若主体未标记，则不能访问任意带有标记的客体。

操作步骤

步骤1 授予用户user1和user2数据表tbl的SELECT和INSERT操作权限。

```
gaussdb=# GRANT SELECT,INSERT ON tbl TO user1,user2;
```

步骤2 用户user1登录数据库执行SELECT和INSERT操作，SELECT失败，INSERT成功，符合强制访问控制策略规则。

```
gaussdb=> SELECT current_user;
 current_user
--------------
 user1
(1 row)
gaussdb=> SELECT * FROM tbl;
ERROR:  permission denied for relation tbl
DETAIL:  N/A
gaussdb=> INSERT INTO tbl VALUES (1);
INSERT 0 1
```

步骤3 用户user2登录数据库执行SELECT和INSERT操作，SELECT成功，INSERT失败，符合强制访问控制策略规则。

```
gaussdb=> SELECT current_user;
 current_user
--------------
 user2
(1 row)
gaussdb=> SELECT * FROM tbl;
 fir
-----
   1
(1 row)
gaussdb=> INSERT INTO tbl VALUES (1);
ERROR:  permission denied for relation tbl
DETAIL:  N/A
```

----结束

 版权所有 © 华为云计算技术有限公司

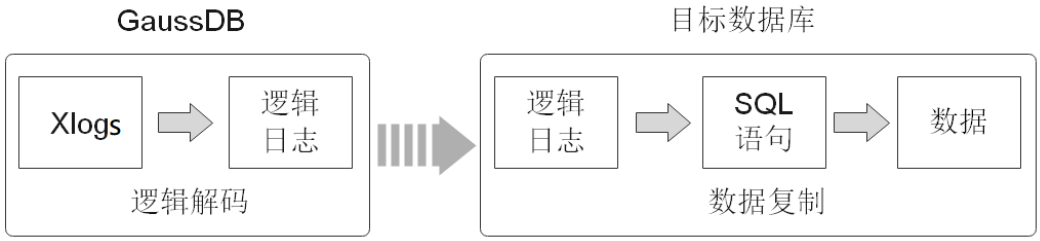
7 逻辑复制

GaussDB对数据复制能力的支持情况为：

支持通过数据迁移工具定期向异构数据库（如Oracle数据库等）进行数据同步，不具备实时数据复制能力，因此不足以支撑与异构数据库间并网运行实时数据同步的诉求。

基于上述情况，GaussDB提供了逻辑解码功能，通过反解xlog的方式生成逻辑日志。目标数据库解析逻辑日志以实时进行数据复制。具体如图7-1所示。逻辑复制降低了对目标数据库的形态限制，支持异构数据库、同构异形数据库对数据的同步，支持目标库进行数据同步期间的数据可读写，数据同步时延低。

图 7-1 逻辑复制



逻辑复制由两部分组成：逻辑解码和数据复制。逻辑解码会输出以事务为单位组织的逻辑日志。业务或数据库中间件将会对逻辑日志进行解析并最终实现数据复制。GaussDB当前只提供逻辑解码功能，因此本章节只涉及逻辑解码的说明。

7.1 逻辑解码

7.1.1 逻辑解码概述

功能描述

逻辑解码为逻辑复制提供事务解码的基础能力，GaussDB可以使用SQL函数接口进行逻辑解码。此方法调用方便，不需使用工具，对接外部工具接口也比较清晰，不需要额外适配。

版权所有 © 华为云计算技术有限公司

由于逻辑日志是以事务为单位的，在事务提交后才能输出，且逻辑解码是由用户驱动的。因此，为了防止事务开始时的xlog被系统回收，或所需的事务信息被VACUUM回收，GaussDB新增了逻辑复制槽，用于阻塞xlog的回收。

一个逻辑复制槽表示一个更改流，这些更改可以在其他数据库上以它们在原数据库上产生的顺序重新执行。每个逻辑复制槽都由其对应逻辑日志的获取者维护。如果处于流式解码中的逻辑复制槽所在库不存在业务，则该复制槽会依照其他库的日志位置来推进。活跃状态的LSN序逻辑复制槽在处理到活跃事务快照日志时可以根据当前日志的LSN推进复制槽；活跃状态的CSN序逻辑复制槽在处理到虚拟事务日志时可以根据当前日志的CSN推进复制槽。

前提条件

- 逻辑日志目前从CN或DN中抽取，如果进行逻辑复制，应使用SSL连接，因此需要保证相应节点上的GUC参数SSL设置为on。

  说明

  为避免安全风险，请保证启用SSL连接。

- 设置GUC参数wal_level为logical。
- 设置GUC参数max_replication_slots>=每个节点所需的（物理流复制槽数+备份槽数+逻辑复制槽数）。

  说明

  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。
  - 同一个CN节点上最多支持同时开启10个分布式CN强一致解码任务，同一个DN节点上最多支持同时开启20个解码任务。

- 用户需要通过DN端口连接数据库，才可以直接使用SQL函数接口进行逻辑解码操作，相关操作请参见**使用SQL函数接口进行逻辑解码**。如果使用CN端口连接数据库，则需要通过EXECUTE DIRECT ON (datanode_name) 'statement'语句来执行SQL函数。
- 仅限初始用户或拥有REPLICATION权限的用户进行操作。三权分立关闭时数据库管理员可进行逻辑复制操作，三权分立开启时不允许数据库管理员进行逻辑复制操作。

注意事项

- 不支持数据页复制的DML解码。
- 逻辑复制不支持集群在线扩容。在线扩容前，需要删除已存在的逻辑复制槽，扩容完成后重新创建。
- 单条元组大小不超过1GB，考虑解码结果可能大于插入数据，因此建议单条元组大小不超过500MB。
- GaussDB支持解码的数据类型为：INTEGER、BIGINT、SMALLINT、TINYINT、SERIAL、SMALLSERIAL、BIGSERIAL、FLOAT、DOUBLE PRECISION、BOOLEAN、BIT(n)、BIT VARYING(n)、DATE、TIME[WITHOUT TIME ZONE]、TIMESTAMP[WITHOUT TIME ZONE]、CHAR(n)、VARCHAR(n)、TEXT、CLOB（解码成TEXT格式）。

版权所有 © 华为云计算技术有限公司

- 非M兼容库的浮点型数据、M兼容库的不带标度的float类型，解码结果显示精度extra_float_digits配置为3（该参数的含义参见GUC参数float_shortest_precision的具体信息）。
- 不支持解码PUBLIC SCHEMA的DDL操作。
- 支持M-Compatibility模式数据库的DML、DDL逻辑解码，解码数据类型如下：
  - 整型：TINYINT(M)、SMALLINT(M)、MEDIUMINT(M)、BIGINT(M)、INT(M)/INTEGER、BOOL/BOOLEAN。
  - 浮点型：FLOAT(M,D)、DOUBLE(M,D)。
  - 定点型：DECIMAL(M,D)、NUMERIC(M,N)。
  - BIT类型：BIT。
  - 字符串二进制类型：CHAR(N)、VARCHAR(N)、BINARY、VARBINARY。
  - 文本类型：TINYTEXT、TEXT、MEDIUMTEXT、LONGTEXT。
  - 日期类型：DATE、TIME、DATETIME、TIMESTAMP、YEAR。
  - 大对象类型：TINYBLOB、BLOB、MEDIUMBLOB、LONGBLOB。
  - 数据类型属性：UNSIGNED、ZEROFILL。
- M-Compatibility模式数据库，逻辑解码特殊约束如下：
  - CREATE TABLE、ALTER TABLE、DROP TABLE等语法中的[partition_options]、ENGINE、ROW_FORMAT、algorithm_option、lock_option等选项在语法中无实际作用，此类语法不进行解码输出。
  - 对于ALTER SCHEMA、CREATE SCHEMA、DROP SCHEMA、ALTER DATABASE、CREATE DATABASE、DROP DATABASE语法中，因为M-Compatibility模式数据库下，DATABASE与Schema等价，所以ALTER DATABASE、CREATE DATABASE、DROP DATABASE会被解码为ALTER SCHEMA、CREATE SCHEMA、DROP SCHEMA。
  - 对于写法不同的字符设置语法，比如CHARACTER SET、CHAR SET、CHARSET都会被解码为CHARACTER SET。
  - 对于ALTER TABLE tbl_name DROP {INDEX | KEY} index_name语法，逻辑解码结果为DROP INDEX；对于删除主键 ALTER TABLE tbl_name DROP {primary key | {index | key} index_name} 语法会被解码为 ALTER TABLE tbl_name DROP CONSTRAINT index_name。
  - 对于ALTER TABLE tbl_name ADD INDEX语法，逻辑解码结果为CREATE INDEX。
  - 在解码DML语句时，涉及插入时间类型带精度的（time、timestamp、datetime），按照最大精度6解码。
  - 对于支持二进制输入的数据类型（BINARY、VARBINARY、TINYBLOB、BLOB、MEDIUMBLOB、LONGBLOB），如果数据中包括'\0'会发生截断，仅支持并行解码BINARY格式输出（即解码任务参数decode-style='b'），串行解码、函数解码以及并行json、text格式输出均会发生截断。
- REPLACE语法为DML语法，REPLACE会被解码为对数据实际的操作，比如INSERT或DELETE+INSERT（存在主键或唯一键冲突）。
- COPY语法/LOAD语法为DML语法，会被解码为对数据实际的操作，比如多条INSERT。
- 逻辑复制槽名称必须小于64个字符，使用SQL函数创建或使用复制槽时，复制槽名称仅支持小写字母、数字以及_、?、-和.字符，且不支持.或..单独作为复制槽名称。调用JDBC API创建或使用复制槽时，复制槽名称仅支持小写字母、数字以及_

 版权所有 © 华为云计算技术有限公司

字符（可以输入大写字母，但在内部会将其被转成小写字母进行处理，例如输入名称为MSLOT，实际名称是mslot）。

- 对多库的解码需要分别在库内创建流复制槽并开始解码，每个库的解码都需要单独扫描一遍日志。
- 不支持强切，强切后需要重新全量导出数据。
- 备机解码时，switchover和failover可能出现解码数据变多，需用户手动过滤。Quorum协议下，switchover和failover选择升主的备机，需要与当前主机日志同步。
- 备机解码时，如果需要删除数据库，需要业务保证数据完成迁移后，主机再删除逻辑复制槽对应的数据库。当前没有强制要求删除数据库之前一定要先删除库上的逻辑复制槽，可能出现当前库备机解码还未完成，主机删除数据库的情况。如果出现这种情况，当备机回放到删除数据库的动作，备机即使未完成当前库的所有解码任务，也会退出解码，如果再次重连，备机解码会因为已经删除数据库而无法启动，造成解码任务受损。
- 只支持CN和主DN创建删除复制槽。当删除的复制槽为最后一个复制槽时，删除完成后会产生告警"replicationSlotMinLSN is INVALID_WAL_REC_PTR!!!"和"replicationSlotMaxLSN is INVALID_WAL_REC_PTR!!!"。
- 不允许主备DN、多个备DN同时使用同一个复制槽解码，否则会产生数据不一致或者其他异常错误的情况。
- 数据库故障重启或逻辑复制进程重启后，解码数据可能存在重复，用户需手动过滤。
- 计算机内核故障后，解码可能存在乱码，需手动或自动过滤。
- 请确保在创建逻辑复制槽过程中未启动长事务，启动长事务会阻塞逻辑复制槽的创建。
- 不支持全局临时表的DML解码。
- 不支持本地临时表的DML解码。
- SELECT INTO语句会解码创建目标表的DDL操作，数据插入的DML操作不解码。
- 为解析某个astore表的UPDATE和DELETE语句，需为此表配置REPLICA IDENTITY属性，在此表无主键时需要配置为FULL，具体配置方式请参考《开发指南》中“SQL参考 > SQL语法 > A > ALTER TABLE”章节中“REPLICA IDENTITY { DEFAULT | USING INDEX index_name | FULL | NOTHING }”字段。
- 禁止在使用逻辑复制槽时在其他节点对该复制槽进行操作，删除复制槽的操作需在该复制槽停止解码后执行。
- 基于目标库可能需要源库的系统状态信息考虑，逻辑解码仅自动过滤模式'pg_catalog'和'pg_toast'下OID小于16384的系统表的逻辑日志。若目标库不需要复制其他相关系统表的内容，逻辑日志回放过程中需要对相关系统表进行过滤。
- 在开启逻辑复制的场景下，如需创建包含系统列的主键索引，必须将该表的REPLICA IDENTITY属性设置为FULL或是使用USING INDEX指定不包含系统列的、唯一的、非局部的、不可延迟的、仅包括标记为NOT NULL列的索引。
- 对于缩容或升级前已存在的复制表场景，需要对复制表手动配置logical_repl_node属性或RESET为默认值，配置方式请参考《开发指南》中“SQL参考 > SQL语法 > A > ALTER TABLE”章节中“storage_parameter”参数的使用说明，以及“logical_repl_node”属性相关说明。
- 若一个事务的子事务过多导致落盘文件过多，退出解码时需执行SQL函数pg_terminate_backend(*逻辑解码的walsender线程id*)来手动停止解码，而且退出时延增加约为1分钟/30万个子事务。因此在开启逻辑解码时，若一个事务的子事务数量达到5万时，会打印一条WARNING日志。

 版权所有 © 华为云计算技术有限公司

- 当逻辑复制槽处于非活跃状态，且设置GUC参数enable_xlog_prune=on、enable_logicalrepl_xlog_prune=on、max_size_for_xlog_retention为非零值，且备份槽或逻辑复制槽导致保留日志段数已超过GUC参数wal_keep_segments，同时其他复制槽并未导致更多的保留日志段数时，如果max_size_for_xlog_retention大于0且当前逻辑复制槽导致保留日志的段数（每段日志大小为16MB）超过max_size_for_xlog_retention，或者max_size_for_xlog_retention小于0且磁盘使用率达到(-max_size_for_xlog_retention)/100，当前逻辑复制槽会强制失效，其restart_lsn将被设置为7FFFFFFF/FFFFFFFF。该状态的逻辑复制槽不参与阻塞日志回收或系统表历史版本的回收，但仍占用复制槽的限制数量，需要手动删除。
- 备机解码启动后，向主机发送复制槽推进指令后会占用主机上对应的逻辑复制槽（即标识为活跃状态）。在此之前主机上对应逻辑复制槽为非活跃状态，此状态下如果满足逻辑复制槽强制失效条件则会被标记为失效（即restart_lsn将被设置为7FFFFFFF/FFFFFFFF），备机将无法推进主机复制槽，且备机回放完成复制槽失效日志后当前复制槽的备机解码断开后将无法重连。
- 不活跃的逻辑复制槽将阻塞WAL日志回收和系统表元组历史版本清理，导致磁盘日志堆积和系统表扫描性能下降，因此不再使用的逻辑复制槽请及时清理。需要特别注意，在升级提交之前观察期内使用DN扩展IP连接DN创建的逻辑复制槽，在升级回滚之前务必手动清理，否则随着DN扩展IP特性回滚无法直连DN清理。
- 分布式强一致逻辑解码（连接CN解码）仅支持GTM-Lite分布式部署及流式解码，不支持CN连接备DN进行解码、SQL逻辑解码函数、在线扩容、全局索引。
- 针对分布式强一致逻辑解码（连接CN解码）功能，CN高可用由业务负责切换。
- CN上的CSN序逻辑复制槽仅起到占位作用，不随着逻辑解码的进行而推进，同时也不会阻塞日志回收。
- 通过协议连接CN创建逻辑复制槽仅支持CSN序复制槽，通过协议连接DN创建逻辑复制槽仅支持LSN序复制槽。
- 针对分布式解码，对于故障报错或者手动停止解码客户端等场景，需等待15秒再次重试解码，如有复制槽占用则需通过执行SQL函数pg_terminate_backend(*占用该复制槽线程id*)来手动解除复制槽占用。
- 在CN上创建复制槽失败报错后，需要在CN上进行复制槽删除操作，然后在CN上重新创建复制槽。
- 在CN上删除逻辑复制槽时，若为LSN序逻辑复制槽，则仅删除当前节点复制槽，其他节点同名复制槽不受影响；否则只要其他节点有残留同名CSN序逻辑复制槽，执行删除时不会因为某些节点不存在复制槽而报错，同时所有节点的同名复制槽会被成功删除；如果任何节点均不存在该复制槽，则报错。
- 在CN上创建CSN序逻辑复制槽时，某些节点如残留同名LSN序逻辑复制槽，需在这些节点执行删除残留复制槽的操作。否则会在除当前CN节点外，其他不存在同名复制槽的CN和主DN节点上创建CSN序逻辑复制槽。
- 如果当前CN节点残留LSN序逻辑复制槽，同时其他某些节点上残留同名CSN序逻辑复制槽，则在当前CN节点上执行删除复制槽操作仅会删除本地LSN序逻辑复制槽，待删除完成再次执行删除操作方可删除其他节点的同名复制槽。
- 解码使用JSON格式输出时不支持数据列包含特殊字符（如'\0'空字符），解码输出列内容将出现被截断现象。
- 不支持无日志表的DML解码。
- 执行备份恢复操作时，实例恢复完成后会清理所有的逻辑复制槽，如有需要须重新建槽。
- 删除复制槽需要在删除数据库之前执行，当逻辑复制槽所在数据库被删除后，这些复制槽变为不可用状态，需要用户手动删除，否则会阻塞wal日志回收。

 版权所有 © 华为云计算技术有限公司

- 当同一事务产生大量需要落盘的子事务时，同时打开的文件句柄可能会超限，需将GUC参数max_files_per_process配置成大于子事务数量上限的两倍。
- 不支持全局二级索引，不支持分布列修改的DML解码。
- 容灾集群的备集群不支持通过SQL系统函数或工具进行逻辑解码。
- 不支持账本数据库功能，当前版本如果开启解码任务的数据库中有关于账本数据库的DML操作，则解码结果中会包含hash列，从而导致回放失败。
- 扩容场景下，若集群中创建有逻辑复制槽，会导致扩容失败，因此扩容前需要删除当前集群中已存在的逻辑复制槽。
- 不支持主键、外键、唯一约束中的信息约束（如not enforced等）选项，如果出现信息约束，则涉及信息约束的约束不解码，其他DDL照常解码。例如：“create table test(a int primary key not enforced);”语句会被解码为“create table test(a int);”。
- 使用CREATE TABLE语法创建表时不支持MURMURHASH选项的解码，若有MURMURHASH选项，将不解码该DDL语句。
- 仅支持wal_level=logical的WAL日志解码，对于非logical的日志，串行解码（包括函数解码）输出结果中没有对应的值和类型，并行解码不输出其逻辑日志。
- 逻辑解码支持如下Xlog类型的DML：
  - Astore: insert、delete、update、multi-insert。
  - Ustore: insert、delete、update、multi-insert。
- wal_level值为logical的情况下，Xlog日志数据量相比hot_standby级别有一定膨胀。例如：在TPCC场景下，Astore的Xlog膨胀率约为11%，Ustore的Xlog膨胀率约为110%。
- 大小写不敏感数据库解码时，无论创建表名或用户名时使用的是大写还是小写，输入解码选项白名单（white-table-list）或黑名单（exclude-users）中的表名或用户名时都须使用小写。
- 逻辑复制槽会保护未解析XLOG对应版本的系统表记录不被过早清理。当逻辑复制槽处于不活跃状态或者客户端消费过慢时，可能会对业务造成以下影响：（1）如果业务中有大量的DDL，则会导致系统表历史版本过多，影响SQL命令执行效率；（2）由于GaussDB数据页面管理的约束：同一页面中无法存放xid差值超过2^32的两条记录。当系统表历史版本过多时，可能会阻塞DDL和DML等业务操作无法正常执行（错误码：GAUSS-21297）。

  处理措施：（1）及时清理不再使用的逻辑复制槽；（2）如果逻辑复制槽推进过慢，请参考《故障处理》文档“常见故障界定与处理 -> 逻辑解码相关故障 -> 逻辑复制槽异常/逻辑解码性能慢”章节进行处理；（3）如果措施(2)未解决推进慢问题，则需要删除复制槽，重建逻辑解码任务。

SQL 函数解码性能

1. 在Benchmarksql-5.0的100warehouse场景下，采用pg_logical_slot_get_changes时：
   - 单次解码数据量4K行（对应约5MB～10MB日志），解码性能0.3MB/s～0.5MB/s。
   - 单次解码数据量32K行（对应约40MB～80MB日志），解码性能3MB/s～5MB/s。
   - 单次解码数据量256K行（对应约320MB～640MB日志），解码性能3MB/s～5MB/s。

 版权所有 © 华为云计算技术有限公司

- 单次解码数据量再增大，解码性能无明显提升。

如果采用pg_logical_slot_peek_changes + pg_replication_slot_advance方式，解码性能相比采用pg_logical_slot_get_changes时要下降30%～50%。

2. 在Benchmarksql-5.0的100warehouse场景下，采用pg_logical_get_area_changes时：
   - 单次解码数据量4K行（对应约5MB～10MB日志），解码性能0.3MB/s～0.5MB/s。
   - 单次解码数据量32K行（对应约40MB～80MB日志），解码性能3MB/s～5MB/s。
   - 单次解码数据量256K行（对应约320MB～640MB日志），解码性能3MB/s～5MB/s。
   - 单次解码数据量再增大，解码性能无明显提升。
3. 在Benchmarksql-5.0的100warehouse场景下，采用pg_logical_slot_get_binary_changes时：
   - 单次解码数据量4K行（对应约5MB～10MB日志），解码性能0.3MB/s～0.5MB/s。
   - 单次解码数据量32K行（对应约40MB～80MB日志），解码性能2MB/s～3MB/s。
   - 单次解码数据量256K行（对应约320MB～640MB日志），解码性能2MB/s～3MB/s。
   - 单次解码数据量再增大，解码性能无明显提升。

   如果采用pg_logical_slot_peek_binary_changes + pg_replication_slot_advance方式，解码性能相比采用pg_logical_slot_get_binary_changes时要下降30%～50%。

流式解码性能

在并行解码的标准场景下（16核CPU、内存128GB、网络带宽 > 200MBps、表的列数为10~100、单行数据量0.1KB~1KB、DML操作以insert为主、不涉及落盘事务即单个事务中语句数量小于4096、parallel-decode-num为8、解码格式为't'且开启批量发送功能），连接DN解码性能（以Xlog消耗量为标准）不低于100MBps，连接CN解码性能不低于80MBps。为保证解码性能达标以及尽量降低对业务的影响，一台备机上应尽量仅建立一个并行解码连接，保证CPU、内存、带宽资源充足。

7.1.2 逻辑解码选项

逻辑解码选项可以用来为本次逻辑解码提供限制或额外功能，如“解码结果是否包含事务号”、“解码时是否忽略空事务”等。对于具体配置方法，SQL函数解码请参考《开发指南》中“SQL参考 > 函数和操作符 > 系统管理函数 > 逻辑复制函数”章节中函数pg_logical_slot_peek_changes的可选入参''options_name'和'options_value'，JDBC流式解码请参考《开发指南》中“应用程序开发教程 > 基于JDBC开发 > 示例：逻辑复制代码示例”章节示例代码中函数withSlotOption的使用方法。

通用选项（串行解码和并行解码均可配置，但可能无效，请参考相关选项详细说明）

- include-xids：

  解码出的data列是否包含xid信息。

  取值范围：boolean型，默认值为true。

 版权所有 © 华为云计算技术有限公司

  - false：设为false时，解码出的data列不包含xid信息。
  - true：设为true时，解码出的data列包含xid信息。
- skip-empty-xacts：

  解码时是否忽略空事务信息。

  取值范围：boolean型，默认值为false。
  - false：设为false时，解码时不忽略空事务信息。
  - true：设为true时，解码时会忽略空事务信息。
- include-timestamp：

  解码信息是否包含commit时间戳。

  取值范围：boolean型，针对并行解码场景默认值为false，针对SQL函数解码和串行解码场景默认值为true。
  - false：设为false时，解码信息不包含commit时间戳。
  - true：设为true时，解码信息包含commit时间戳。
- only-local：

  是否仅解码本地日志。

  取值范围：boolean型，默认值为true。
  - false：设为false时，解码非本地日志和本地日志。
  - true：设为true时，仅解码本地日志。
- white-table-list：

  白名单参数，包含需要进行解码的schema和表名。

  取值范围：包含白名单中表名的字符串，不同的表以','为分隔符进行隔离；使用'*'来模糊匹配所有情况；schema名和表名间以'.'分隔，不允许存在任意空白符。例如：

  ```
  select * from pg_logical_slot_peek_changes('slot1', NULL, 4096, 'white-table-list',
  'public.t1,public.t2,*.t3,my_schema.*');
  ```
- max-txn-in-memory：

  内存管控参数，单位为MB，单个事务占用内存大于该值即进行落盘。

  串行解码-取值范围：0~100的整型，默认值为0，即不开启此种管控。

  并行解码-取值范围：0~max_process_memory总量的25%，默认值为max_process_memory/4/1024，其中1024为kB到MB的单位转换，0表示不开启此条内存管控项。
- max-reorderbuffer-in-memory

  内存管控参数，单位为GB，拼接-发送线程中正在拼接的事务总内存（包含缓存）大于该值则对当前解码事务进行落盘。

  串行解码-取值范围：0~100的整型，默认值为0，即不开启此种管控。

  并行解码-取值范围：0~max_process_memory总量的50%，默认值为max_process_memory/2/1048576，其中1048576为kB到GB的单位转换，0表示不开启此条内存管控项。
- include-user：

  事务的BEGIN逻辑日志是否输出事务的用户名。事务的用户名特指授权用户——执行事务对应会话的登录用户，它在事务的整个执行过程中不会发生变化。

  取值范围：boolean型，默认值为false。
  - false：设为false时，事务的BEGIN逻辑日志不输出事务的用户名。

 版权所有 © 华为云计算技术有限公司

  - true：设为true时，事务的BEGIN逻辑日志输出事务的用户名。

- exclude-userids：

  黑名单用户的OID参数，该参数只支持直连DN解码任务配置，分布式CN强一致解码不支持该参数。

  取值范围：字符串类型，指定黑名单用户的OID，多个OID通过','分隔，不校验用户OID是否存在。

- exclude-users：

  黑名单用户的名称列表。

  取值范围：字符串类型，指定黑名单用户名，通过','分隔，不校验用户名是否存在。

- dynamic-resolution：

  是否动态解析黑名单用户名。如果解码某条XLOG，且XLOG写入时，用户未创建，则认为用户不存在。

  取值范围：boolean型，默认值为true。

  - false：设为false时，当解码观测到黑名单exclude-users中用户不存在时将会报错并退出逻辑解码；当用户存在，黑名单功能正常过滤用户的操作。
  - true：设为true时，当解码观测到黑名单exclude-users中用户不存在时不报错，并正常解码；当用户存在，黑名单功能正常过滤用户的操作。

- standby-connection：

  仅流式解码设置，是否仅限制备机解码，因为CN没有备机，所以该参数在仅连接DN时支持。

  取值范围：boolean型，默认值为false。

  - true：设为true时，仅允许连接备机解码，连接主机解码时会报错退出。
  - false：设为false时，不做限制，允许连接主机或备机解码。

  说明

  如果主机资源使用率较大且业务对增量数据同步的实时性不敏感，建议进行备机解码；如果业务对增量数据同步的实时性要求高并且主机业务压力较小，建议使用主机解码。

- sender-timeout：

  仅流式解码设置，内核与客户端的心跳超时阈值。如果该时间段内没有收到客户端任何消息，逻辑解码将主动停止，并断开和客户端的连接。单位为毫秒（ms）。

  取值范围：0~2147483647的int型，默认值取决于GUC参数logical_sender_timeout的配置值。设置为0，表示逻辑解码不会主动断开与客户端的连接，如果设置过小，例如1ms，则可能存在解码任务中断风险。

- change-log-max-len:

  逻辑日志缓存长度上限参数，单位为字节。仅连接DN的并行解码有效，分布式强一致解码、串行解码及SQL函数解码无效。如果单条解码结果长度超过上限，则会销毁重新分配大小为1024字节的内存并缓存。过长会增加内存占用，过短会频繁触发内存申请和释放的操作，不建议设置成小于1024的值。

  取值范围：1~65535，默认值为4096。

- max-decode-to-sender-cache-num:

  并行解码日志的缓存条数阈值。仅连接DN的并行解码有效，分布式强一致解码、串行解码及SQL函数解码无效。缓存中的日志条数未超过这个阈值时，使用完毕的解码日志将置入缓存，否则直接释放。

 版权所有 © 华为云计算技术有限公司

取值范围：1~65535，默认值为4096。

- enable-heartbeat：

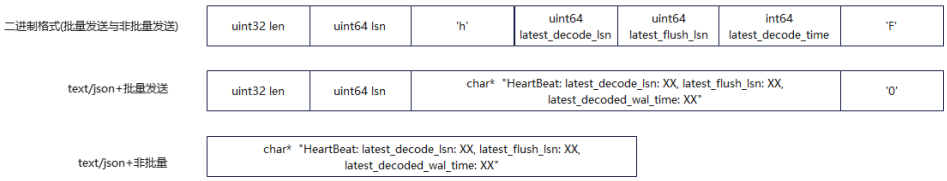
  仅流式解码时设置，代表是否输出心跳日志。

  取值范围：boolean型，默认值为false。

  - true：设为true时，输出心跳日志。
  - false：设为false时，不输出心跳日志。

  说明

  若开启心跳日志选项，此处说明并行解码场景心跳日志如何解析：二进制格式首先是字符'h'表示消息是心跳日志，之后是心跳日志内容，分别是8字节uint64，直连DN解码场景代表LSN，表示发送心跳逻辑日志时读取的WAL日志结束位置，而在分布式强一致解码场景为CSN，表示发送心跳逻辑日志时已发送的解码日志事务CSN；8字节uint64，直连DN解码场景代表LSN，表示发送心跳逻辑日志时刻已经落盘的WAL日志的位置，而在分布式强一致解码场景为CSN，表示集群下一个提交事务将获得的CSN；8字节int64代表时间戳（从1970年1月1日开始），表示最新解码到的事务日志或检查点日志的产生时间戳。关于消息结束符：如果是二进制格式则为字符'F'，如果格式为text或者json且为批量发送则结束符为0，否则没有结束符。消息内容采用大端字节序进行数据传输。具体格式见下图（考虑到前向兼容性，相关部分仍保留着LSN的命名方式，实际含义依具体场景而定）：

- parallel-decode-num：

  仅流式解码设置有效，并行解码的Decoder线程数量；系统函数调用场景下此选项无效，仅校验取值范围。

  取值范围：1~20的int型，取1表示按照原有的串行逻辑进行解码，取其余值即为开启并行解码，默认值为1。

须知

当parallel-decode-num不配置（即为默认值1）或显式配置为1时，下述“并行解码”中的选项不可配置。

- output-order：

  仅流式解码设置有效，代表是否使用CSN顺序输出解码结果；系统函数调用场景下此选项无效，仅校验取值范围。

  取值范围：0或1的int型，默认值为0。

  - 0：设为0时，解码结果按照事务的COMMIT LSN排序，当且仅当解码复制槽的confirmed_csn列值为0（即不显示）时可使用该方式，否则报错。
  - 1：设为1时，解码结果按照事务的CSN排序，当且仅当解码复制槽的confirmed_csn列值为非零时可使用该方式，否则报错。

版权所有 © 华为云计算技术有限公司

须知

- 当output-order不配置（即为默认值0，按照COMMIT LSN排序）或显式配置为0时，下述“分布式强一致解码”中的选项不可配置。
- 在流式解码场景，DN收到来自CN的逻辑解码连接时，output-order选项失效，默认采用CSN序解码。

- auto-advance：

  仅流式解码设置有效，代表是否允许自主推进逻辑复制槽。

  取值范围：boolean型，默认值为false。

  - true：设为true时，在已发送日志都被确认推进且没有待发送事务时，推进逻辑复制槽到当前解码位置。
  - false：设为false时，完全交由复制业务调用日志确认接口推进逻辑复制槽。

- skip-generated-columns：

  逻辑解码控制参数，用于跳过存储生成列的输出。对UPDATE和DELETE的旧元组无效，相应元组始终会输出存储生成列。分布式版本暂不支持存储生成列，此配置选项暂无实际影响。

  取值范围：boolean型，默认值为false/off。

  - true/on：值为true/on时，不输出存储生成列的解码结果。
  - false/off：设为false/off时，输出存储生成列的解码结果。

  虚拟生成列不受此参数控制，DML的解码结果始终不会输出虚拟生成列。

- enable-ddl-decoding：

  逻辑解码控制参数，用于控制是否开启DDL语句的逻辑解码。

  取值范围：boolean型，默认值为false。

  - true：值为true时，开启DDL语句的逻辑解码。
  - false：值为false时，不开启DDL语句的逻辑解码。

- enable-ddl-json-format：

  逻辑解码控制参数，用于控制DDL的反解析流程以及输出形式。

  取值范围：boolean型，默认值为false。

  - true：值为true时，传送JSON格式的DDL反解析结果。
  - false：设为false时，传送decode-style指定格式的DDL反解析结果。

- timezone-is-utc：

  逻辑解码控制参数，用于控制时间类型（携带时区）数据的输出。如ORA、MYSQL兼容下的timestamptz类型，M兼容的timestamp类型都表示带时区的时间类型，默认会输出当前数据库时区的时间，设置为true时，在解码时会强制输出为0时区的时间。该参数仅对流式解码有效，函数解码使用该参数会忽略不生效。

  取值范围：boolean型，默认值为false。

  - true：值为true时，解码时间类型数据输出0时区的时间。
  - false：值为false时，解码时间类型数据输出当前数据库时区的时间。

- decode-sequence：

  逻辑解码控制参数，用来指定是否输出sequence值的变更日志的解码结果。

  取值范围：boolean型，默认值为false。

 版权所有 © 华为云计算技术有限公司

- true：暂不支持设置。
- false：设为false时，不输出sequence值的变更日志的解码结果。

须知

解码选项decode-sequence当前仅允许设置为false，设置为true会在启动解码时报错退出。

分布式强一致解码

- logical-receiver-num：

  仅流式解码设置有效，分布式解码启动的logical_receiver数量，系统函数调用场景下此选项无效，仅校验取值范围。

  取值范围：1~20的int型，默认值为1。当该值被设置为比当前集群分片数更大时，将被修改为分片数。

- slice-id：

  仅连接DN解码时设置，指定当前DN所在的分片号，用于复制表解码。

  取值范围：0~8192的int型，默认值为-1，即不指定分片号，但在解码到复制表时会报错。

⚠ 注意

该配置选项在尝试连接DN使用CSN序逻辑复制槽（confirmed_csn为非0值的复制槽）进行解码时使用，用来表示自己的分片号（即第几个分片，第一个分片则输入0），如果不设置该参数（即使用默认值-1）在解码到复制表时将会报错。此参数用于使用连接CN的分布式解码时，CN从DN收集解码结果时使用，CN只会收集指定DN分片的复制表解码结果。不建议在此场景下手动连接DN解码。

- start-position：

  仅连接DN设置，主要功能为过滤掉小于指定CSN对应的事务，以及针对指定的CSN对应的事务，过滤掉小于指定LSN的日志，且指定CSN对应事务的BEGIN日志一定被过滤掉。

  取值范围：字符串类型，可以解析为以'/'分隔，左右两侧分别为代表CSN和LSN的两个uint64类型。

⚠ 注意

该配置选项用于CN解码时，CN建立与DN的连接后发送解码请求时使用此配置选项过滤可能已经被接收过的日志。不建议在此场景下手动连接DN解码使用此参数。

串行解码

- force-binary：

  是否以二进制格式输出解码结果，针对不同场景呈现不同行为。

版权所有 © 华为云计算技术有限公司

- 针对系统函数pg_logical_slot_get_binary_changes和pg_logical_slot_peek_binary_changes：

  取值范围：boolean型，默认值为false。此值无实际意义，均以二进制格式输出解码结果。

- 针对系统函数pg_logical_slot_get_changes、pg_logical_slot_peek_changes和pg_logical_get_area_changes：

  取值范围：仅取false值的boolean型。以文本格式输出解码结果。

- 针对流式解码（仅连接DN时支持）：

  取值范围：boolean型，默认值为false。此值无实际意义，均以文本格式输出解码结果。

并行解码

以下配置选项仅限流式解码设置。

- decode-style：

  当enable-ddl-json-format参数值为true时，DDL的格式由enable-ddl-json-format控制，decode-style仅指定DML语句的解码格式；当enable-ddl-json-format参数值为false时，decode-style指定DML和DDL语句的解码格式。

  取值范围：char型的字符'j'、't'或'b'，分别代表json格式、text格式及二进制格式。

  默认值：

  - 没有指定decode-style：

    针对复制槽插件类型为mppdb_decoding、sql_decoding，decode-style默认值为'b'即二进制格式解码。针对复制槽插件类型为parallel_binary_decoding、parallel_json_decoding、parallel_text_decoding，decode-style默认值分别为'b'、'j'、't'，解码格式分别为二进制格式、json格式、text格式。

  - 指定decode-style：

    按照指定的decode-style进行解码。

  对于json格式和text格式解码，开启批量发送选项时的解码结果中，每条解码语句的前4字节组成的uint32代表该条语句总字节数（不包含该uint32类型占用的4字节，0代表本批次解码结束），8字节uint64代表相应lsn（begin对应first_lsn，commit对应end_lsn，其他场景对应该条语句的lsn）。

  例如：以mppdb_decoding插件为例，当decode-style为b类型时，以二进制格式解码，结果如下：

  ```
  current_lsn: 0/CFE5C80 BEGIN CSN: 2357 first_lsn: 0/CFE5C80
  current_lsn: 0/CFE5D40 INSERT INTO public.test1 new_tuple: {a[typid = 23]: "1", b[typid = 23]: "2"}
  current_lsn: 0/CFE5E68 COMMIT xid: 78108
  ```

  当decode-style为j类型时，以json格式解码，结果如下：

  ```
  BEGIN CSN: 2358 first_lsn: 0/CFE6220
  {"table_name":"public.test1","op_type":"INSERT","columns_name":["a","b"],"columns_type":["integer","integer"],"columns_val":["3","3"],"old_keys_name":[],"old_keys_type":[],"old_keys_val":[]}
  COMMIT XID: 78109
  ```

  当decode-style为t类型时，以text格式解码，结果如下：

  ```
  BEGIN CSN: 2359 first_lsn: 0/CFE64D0
  table public test1 INSERT: a[integer]:3 b[integer]:4
  COMMIT XID: 78110
  ```

 版权所有 © 华为云计算技术有限公司

说明

二进制格式编码规则如下所示：

1. 前4字节代表接下来到语句级别分隔符字母P（不含）或者该批次结束符F（不含）的解码结果的总字节数，该值如果为0代表本批次解码结束。
2. 接下来8字节uint64代表相应lsn（begin对应first_lsn，commit对应end_lsn，其他场景对应该条语句的lsn）。
3. 接下来1字节的字母有5种B/C/I/U/D，分别代表begin/commit/insert/update/delete。
4. 第3步字母为B时：
    1. 接下来的8字节uint64代表CSN。
    2. 接下来的8字节uint64代表first_lsn。
    3. 【该部分为可选项】接下来的1字节字母如果为T，则代表后面4字节uint32表示该事务commit时间戳长度，再后面等同于该长度的字符为时间戳字符串。
    4. 【该部分为可选项】接下来的1字节字母如果为N，则代表后面4字节uint32表示该事务用户名的长度，再后面等同于该长度的字符为事务的用户名字。
    5. 因为之后仍可能有解码语句，接下来会有1字节字母P或F作为语句间的分隔符，P代表本批次仍有解码的语句，F代表本批次解码完成。
5. 第3步字母为C时：
    1. 【该部分为可选项】接下来1字节字母如果为X，则代表后面的8字节uint64表示xid。
    2. 【该部分为可选项】接下来的1字节字母如果为T，则代表后面4字节uint32表示时间戳长度，再后面等同于该长度的字符为时间戳字符串。
    3. 因为批量发送日志时，一个COMMIT日志解码之后可能仍有其他事务的解码结果，接下来的1字节字母如果为P则表示该批次仍需解码，如果为F则表示该批次解码结束。
6. 第3步字母为I/U/D时：
    1. 接下来的2字节uint16代表schema名的长度。
    2. 按照上述长度读取schema名。
    3. 接下来的2字节uint16代表table名的长度。
    4. 按照上述长度读取table名。
    5. 【该部分为可选项】接下来1字节字母如果为N代表为新元组，如果为O代表为旧元组，这里先发送新元组。
        1. 接下来的2字节uint16代表该元组需要解码的列数，记为attrnum。
        2. 以下流程重复attrnum次。
            1. 接下来2字节uint16代表列名的长度。
            2. 按照上述长度读取列名。
            3. 接下来4字节uint32代表当前列类型的OID。
            4. 接下来4字节uint32代表当前列值（以字符串格式存储）的长度，如果为0xFFFFFFFF则表示NULL，如果为0则表示长度为0的字符串。
            5. 按照上述长度读取列值。
    6. 因为之后仍可能有解码语句，接下来的1字节字母如果为P则表示该批次仍需解码，如果为F则表示该批次解码结束。

- sending-batch：

  指定是否批量发送。

  取值范围：0或1的int型，默认值为0。

  - 0：设为0时，表示逐条发送解码结果。
  - 1：设为1时，表示解码结果累积到达1MB则批量发送解码结果。

  开启批量发送的场景中，当解码格式为'j'或't'时，在原来的每条解码语句之前会附加一个uint32类型，表示本条解码结果长度（长度不包含当前的uint32类型），以及一个uint64类型，表示当前解码结果对应的lsn。

 版权所有 © 华为云计算技术有限公司

须知

在CSN序解码（即output-order设置为1）场景下，批量发送仅限于单个事务内（即如果一个事务有多条较小的语句会采用批量发送），即不会使用批量发送功能在同一批次里发送多个事务，且BEGIN和COMMIT语句不会批量发送。

- parallel-queue-size：
  指定并行逻辑解码线程间进行交互的队列长度。
  取值范围：2~1024的int型，且必须为2的整数幂，默认值为128。
  队列长度和解码过程的内存使用量正相关。

7.1.3 使用 SQL 函数接口进行逻辑解码

GaussDB可以通过调用SQL函数，进行创建、删除、推进逻辑复制槽，获取解码后的事务日志。

操作步骤

步骤1 以具有REPLICATION权限的用户登录GaussDB集群任一主DN。

步骤2 使用如下命令通过DN端口连接数据库。

```
gsql -U user1 -W password -d gaussdb -p 40000 -r
```

其中，user1为用户名，password为密码，gaussdb为需要连接的数据库名称，40000为数据库DN端口号，用户可根据实际情况替换。复制槽是建立在DN上的，因此需要通过DN端口连接数据库。

步骤3 创建名称为slot1的逻辑复制槽。

```
gaussdb=> SELECT * FROM pg_create_logical_replication_slot('slot1', 'mppdb_decoding');
slotname | xlog_position
----------+---------------
slot1    | 0/601C150
(1 row)
```

步骤4 通过CN端口连接数据库，在数据库中创建表t，并向表t中插入数据。

```
gaussdb=> CREATE TABLE t(a int PRIMARY KEY, b int);
gaussdb=> INSERT INTO t VALUES(3,3);
```

步骤5 参考**步骤2**连接DN，读取复制槽slot1解码结果，解码条数为4096。

说明

逻辑解码选项请参见**逻辑解码选项**。

```
gaussdb=> SELECT * FROM pg_logical_slot_peek_changes('slot1', NULL, 4096);
location  |  xid  | data
-----------+-------
+-----------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------
-----------------------------------------------
 0/601C188 | 1010023 | BEGIN 1010023
 0/601ED60 | 1010023 | COMMIT 1010023 (at 2023-09-14 16:03:51.394287+08) CSN 1010022
 0/601ED60 | 1010024 | BEGIN 1010024
 0/601ED60 | 1010024 | {"table_name":"public.t","op_type":"INSERT","columns_name":
["a","b"],"columns_type":["integer","integer"],"columns_val":["3","3"],"old_keys_name":[],"old_keys_type":
[],"old_keys_val":[]}
 0/601EED8 | 1010024 | COMMIT 1010024 (at 2023-09-14 16:03:57.239821+08) CSN 1010023
(5 rows)
```


版权所有 © 华为云计算技术有限公司

步骤6 删除逻辑复制槽slot1。

```
gaussdb=> SELECT * FROM pg_drop_replication_slot('slot1');
 pg_drop_replication_slot
--------------------------

(1 row)
```

----结束

7.1.4 使用流式解码实现数据逻辑复制

第三方复制工具通过流式逻辑解码从GaussDB抽取逻辑日志后到对端数据库回放。对于使用JDBC连接数据库的复制工具，具体代码请参考《开发指南》中“应用程序开发教程 > 基于JDBC开发 > 示例：逻辑复制代码示例”章节。

7.1.5 逻辑解码支持 DDL

GaussDB主机上正常执行DDL语句，通过逻辑解码工具可以获取到DDL语句。

表 7-1 具体支持的 DDL 类型

表	索引	自定义函数	自定义存储过程	触发器	Sequence	Schema	Comment on
CREATE TABLE [PARTITION | AS | SUBPARTITION ALTER TABLE [PARTITION | SUBPARTITION DROP TABLE RENAME TABLE TRUNCATE	CREATE INDEX ALTER INDEX DROP INDEX REINDEX	CREATE FUNCTION ALTER FUNCTION DROP FUNCTION	CREATE PROCEDURE ALTER PROCEDURE DROP PROCEDURE	CREATE TRIGGER ALTER TRIGGER DROP TRIGGER	CREATE SEQUENCE ALTER SEQUENCE DROP SEQUENCE	CREATE SCHEMA ALTER SCHEMA DROP SCHEMA	COMMENT ON

版权所有 © 华为云计算技术有限公司

表 7-2 M-Compatibility 模式数据库支持的 DDL 类型

表	索引	Sequence	Schema	Comment on
ALTER TABLE CREATE TABLE DROP TABLE ALTER TABLE PARTITION CREATE TABLE PARTITION TRUNCATE	ALTER INDEX CREATE INDEX DROP INDEX REINDEX ALTER TABLE DROP INDEX ALTER TABLE ADD INDEX	ALTER SEQUENCE CREATE SEQUENCE DROP SEQUENCE	ALTER SCHEMA CREATE SCHEMA DROP SCHEMA ALTER DATABASE CREATE DATABASE DROP DATABASE	COMMENT ON

功能描述

数据库在执行DML的时候，存储引擎会生成对应的DML日志，用于进行恢复，对这些DML日志进行解码，即可还原对应的DML语句，生成逻辑日志。而对于DDL语句，数据库并不记录DDL原语句的日志，而是记录DDL语句涉及的系统表的DML日志。DDL种类多样、语法复杂，逻辑复制要支持DDL语句，通过这些系统表的DML日志来解码原DDL语句是非常困难的。新增DDL日志记录原DDL信息，并在解码时通过DDL日志可以得到DDL原语句。

在DDL语句执行过程中，SQL引擎解析器会对原语句进行语法、词法解析，并生成解析树（不同的DDL语法会生成不同类型的解析树，解析树中包含DDL语句的全部信息）。随后，执行器通过这些信息执行对应操作，生成、修改对应元信息。

本节通过新增DDL日志的方式，来支持逻辑解码DDL，其内容由解析器结果（解析树）以及执行器结果生成，并在执行器执行完成后生成该日志。

从语法树反解析出DDL，DDL反解析能够将DDL命令转换为JSON格式的语句，并提供必要的信息在目标位置重建DDL命令。与原始DDL命令字符串相比，使用DDL反解析的好处包括：

1. 解析出来的每个数据库对象都带有Schema，因此如果使用不同的search_path，也不会有歧义。
2. 结构化的JSON和格式化的输出能支持异构数据库。如果用户使用的是不同的数据库版本，并且存在某些DDL语法差异，需要在应用之前解决这些差异。

反解析输出的结果是规范化后的形式，结果与用户输入等价，不保证完全相同，例如：

示例1：在函数体中没有单引号'时，函数体的分隔符$$会被解析为单引号'。

原始SQL语句：

```
CREATE FUNCTION func(a INT) RETURNS INT AS
$$
BEGIN
a:= a+1;
CREATE TABLE test(col1 INT);
```

 版权所有 © 华为云计算技术有限公司

```
INSERT INTO test VALUES(1);
DROP TABLE test;
RETURN a;
END;
$$
LANGUAGE plpgsql;
```

反解析结果：

```
CREATE  FUNCTION public.func ( IN a pg_catalog.int4 ) RETURNS pg_catalog.int4 LANGUAGE plpgsql
VOLATILE CALLED ON NULL INPUT SECURITY INVOKER COST 100 AS '
BEGIN
a:= a+1;
CREATE TABLE test(col1 INT);
INSERT INTO test VALUES(1);
DROP TABLE test;
RETURN a;
END;
';
```

示例3：“ALTER INDEX "Alter_Index_Index" REBUILD PARTITION "CA_ADDRESS_SK_index2"”会被反解析为“REINDEX INDEX public."Alter_Index_Index" PARTITION "CA_ADDRESS_SK_index2"”。

示例4：创建/修改范围分区表，START END语法格式均解码转化为LESS THAN语句：

```
gaussdb=# CREATE TABLE test_create_table_partition2 (c1 INT, c2 INT)
PARTITION BY RANGE (c2) (
    PARTITION p1 START(1) END(1000) EVERY(200) ,
    PARTITION p2 END(2000),
    PARTITION p3 START(2000) END(2500),
    PARTITION p4 START(2500),
    PARTITION p5 START(3000) END(5000) EVERY(1000)
);
```

会被反解析为：

```
gaussdb=# CREATE TABLE test_create_table_partition2 (c1 INT, c2 INT)
PARTITION BY RANGE (c2) (
    PARTITION p1_0 VALUES LESS THAN ('1'), PARTITION p1_1 VALUES LESS THAN ('201'), PARTITION p1_2
VALUES LESS THAN ('401'), PARTITION p1_3 VALUES LESS THAN ('601'), PARTITION p1_4 VALUES LESS
THAN ('801'), PARTITION p1_5 VALUES LESS    THAN ('1000'),
    PARTITION p2 VALUES LESS THAN ('2000'),
    PARTITION p3 VALUES LESS THAN ('2500'),
    PARTITION p4 VALUES LESS THAN ('3000'),
    PARTITION p5_1 VALUES LESS THAN ('4000'),
    PARTITION p5_2 VALUES LESS THAN ('5000')
);
```

示例5：新增表的列字段时，使用IF NOT EXISTS判断。

- 原始SQL语句：

  ```
  gaussdb=# CREATE TABLE IF NOT EXISTS tb5 (c1 int,c2 int) with (ORIENTATION=ROW,
  STORAGE_TYPE=USTORE);
  gaussdb=# ALTER TABLE IF EXISTS tb5 * ADD COLUMN IF NOT EXISTS c2 char(5) after c1; -- 可解码。
  TABLE中已有int型的列c2，语句执行跳过，反解析结果中c2列的类型仍为原来的类型。
  ```

  反解析结果：

  ```
  gaussdb=# ALTER TABLE IF EXISTS public.tb5 ADD COLUMN IF NOT EXISTS c2 pg_catalog.int4 AFTER
  c1;
  ```

- 原始SQL语句：

  ```
  gaussdb=# ALTER TABLE IF EXISTS tb5 * ADD COLUMN IF NOT EXISTS c2 char(5) after c1，ADD
  COLUMN IF NOT EXISTS c3 char(5) after c1; -- 解码。新增列c3的反解析结果类型正确。
  ```

  反解析结果：

  ```
  gaussdb=# ALTER TABLE IF EXISTS public.tb5 ADD COLUMN IF NOT EXISTS c2 pg_catalog.int4 AFTER
  c1, ADD COLUMN IF NOT EXISTS c3 pg_catalog.bpchar(5) AFTER c1;
  ```

- 原始SQL语句：

  ```
  gaussdb=# ALTER TABLE IF EXISTS tb5 * ADD COLUMN c2 char(5) after c1，ADD COLUMN IF NOT
  EXISTS c4 int after c1; --不解码，语句执行错误。
  ```

 版权所有 © 华为云计算技术有限公司

示例6：定义数据库对象（DROP TABLE/INDEX/SEQUENCE）时，解码结果会添加IFEXISTS语法选项。

- 原始SQL语句：

```
gaussdb=# CREATE TABLE IF NOT EXISTS tb6 (c1 int,c2 int) with (ORIENTATION=ROW,
STORAGE_TYPE=USTORE);
gaussdb=#    DROP TABLE tb6;
```

  反解析结果：

```
gaussdb=# DROP TABLE IF EXISTS public.tb6 RESTRICT;
```

规格约束

- 逻辑解码支持DDL规格：
  - 纯DDL逻辑解码性能标准环境下约为100MB/S，DDL/DML混合事务逻辑解码性能标准环境下约为100MB/S。
  - 开启此功能后（设置wal_level=logical且enable_logical_replication_ddl=on），对DDL语句影响性能下降小于15%。
- 解码通用约束（串行和并行）：
  - 不支持解码本地临时对象的DDL操作。
  - 不支持FOREIGN TABLE场景的DDL解码。
  - alter table add column的default值不支持stable类型和volatile类型的函数；create table和alter table的column的check表达式不支持stable类型和volatile类型的函数；alter table如果有多条子语句，只要其中一条子语句存在上述两种情况，则该条alter table整条语句不反解析。

    ```
    gaussdb=# ALTER TABLE tbl_28 ADD COLUMN b1 TIMESTAMP DEFAULT NOW(); -- 's' NOT
    DEPARSE
    gaussdb=# ALTER TABLE tbl_28 ADD COLUMN b2 INT DEFAULT RANDOM(); -- 'v' NOT DEPARSE
    gaussdb=# ALTER TABLE tbl_28 ADD COLUMN b3 INT DEFAULT ABS(1); -- 'i' DEPARSE
    ```

  - 不支持分布式CREATE MATERIALIZED VIEW的DDL解码。
  - 不支持CREATE/ALTER/DROP VIEW、COMMENT ON VIEW的DDL解码。
  - 不支持REINDEX DATABASE/SYSTEM的DDL解码。
  - 不支持视图上触发器相关的DDL解码。
  - 不支持CONCURRENTLY相关语句的DDL解码。
  - 创建对象时语句中存在IF NOT EXISTS时，如果对象已存在，则不进行解码。删除对象时语句中存在IF EXISTS时，如果对象不存在，则不进行解码。
  - 不对ALTER PACKAGE COMPILE语句进行解码，但会解码实例化内容中包含的DDL/DML语句。如果PACKAGE里没有DDL或DML部分的实例化内容，则alter package compile会被逻辑解码忽略。
  - 仅支持本版本之前版本的商用DDL语法，以下SQL语句不支持逻辑解码。
    - 创建行存表，设置ILM策略。

      原始SQL语句：

      ```
      gaussdb=# CREATE TABLE IF NOT EXISTS tb3 (c1 int) with
      (storage_type=USTORE,ORIENTATION=ROW) ILM ADD POLICY ROW STORE COMPRESS
      ADVANCED ROW AFTER 7 day OF NO MODIFICATION;
      ```

      反解析结果：

      ```
      gaussdb=# CREATE TABLE IF NOT EXISTS public.tb3 (c1 pg_catalog.int4) WITH
      (storage_type = 'ustore', orientation = 'row', compression = 'no') NOCOMPRESS;
      ```

    - B兼容模式下创建或修改表时，给表或列添加注释。

      原始SQL语句：

 版权所有 © 华为云计算技术有限公司

```
gaussdb=# CREATE TABLE IF NOT EXISTS tb6 (c1 integer comment 'Mysql兼容注释语法')
with (ORIENTATION=ROW, STORAGE_TYPE=USTORE);
```

反解析结果：

```
gaussdb=# CREATE TABLE IF NOT EXISTS public.tb6 (c1 pg_catalog.int4) WITH
(storage_type = 'ustore', orientation = 'row', compression = 'no') NOCOMPRESS;
```

- 逻辑解码不支持DDL（DCL）/DML混合事务，混合事务中DDL之后的DML解码不支持。

```
-- 均不反解析，DCL为不支持语句故不解析，DML处于DCL之后也不反解析
gaussdb=# BEGIN;
gaussdb=# GAINT ALL PRIVILEGES to u01;
gaussdb=# INSERT INTO test1(col1) values(1);
gaussdb=# COMMIT;

-- 只反解析第一句和第三句SQL语句
gaussdb=# BEGIN;
gaussdb=# CREATE TABLE mix_tran_t4(id int);
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# CREATE TABLE mix_tran_t5(id int);
gaussdb=# COMMIT;

-- 只反解析第一句和第二句SQL语句
gaussdb=# BEGIN;
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# CREATE TABLE mix_tran_t6(id int);
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# COMMIT;

-- 全反解析
gaussdb=# BEGIN;
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# CREATE TABLE mix_tran_t7(id int);
gaussdb=# CREATE TABLE mix_tran_t8(id int);
gaussdb=# COMMIT;

-- 只反解析第一句和第三句SQL语句
gaussdb=# BEGIN;
gaussdb=# CREATE TABLE mix_tran_t7(id int);
gaussdb=# CREATE TYPE compfoo AS (f1 int, f2 text);
gaussdb=# CREATE TABLE mix_tran_t8(id int);
gaussdb=# COMMIT;

-- 全反解析
gaussdb=# BEGIN;
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# COMMIT;

-- 只反解析第一句SQL语句
gaussdb=# BEGIN;
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# CREATE TYPE compfoo AS (f1 int, f2 text);
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# COMMIT;

-- 只反解析第一句和第三句SQL语句
gaussdb=# BEGIN;
gaussdb=# INSERT INTO mix_tran_t4 VALUES(111);
gaussdb=# CREATE TYPE compfoo AS (f1 int, f2 text);
gaussdb=# CREATE TABLE mix_tran_t9(id int);
gaussdb=# COMMIT;
```

- 逻辑解码语句CREATE TABLE AS SELECT、SELECT INTO和CREATE TABLE AS仅能解码出CREATE TABLE语句，暂不支持解码INSERT语句。

  说明

  对于CTAS创建的表，仍会解码其ALTER和DROP语句。

 版权所有 © 华为云计算技术有限公司

示例：

原始SQL语句：

```
CREATE TABLE IF NOT EXISTS tb35_2 (c1 int) with (storage_type=USTORE,ORIENTATION=ROW);
INSERT INTO tb35_2 VALUES (6);
CREATE TABLE tb35_1 with (storage_type=USTORE,ORIENTATION=ROW) AS SELECT * FROM tb35_2;
```

最后一句SQL语句反解析结果：

```
 CREATE  TABLE  public.tb35_1 (c1 pg_catalog.int4) WITH (storage_type = 'ustore', orientation = 'row', compression = 'no') NOCOMPRESS;
```

- 执行存储过程/函数/高级包时，若其本身包含DDL/DML混合事务或者其本身与同事务内其他语句组成DDL/DML混合事务，则按照混合事务原则执行解码。
- 逻辑解码不支持账本数据库功能，创建账本数据库的DDL语句解码结果中会包含hash列。
  - 原始语句：
    ```
    CREATE SCHEMA blockchain_schema WITH BLOCKCHAIN;
    CREATE TABLE blockchain_schema.blockchain_table(mes int);
    ```
  - 解码结果：
    ```
    CREATE SCHEMA blockchain_schema WITH BLOCKCHAIN;
    CREATE TABLE blockchain_schema.blockchain_table (mes pg_catalog.int4, hash_a1d895 pg_catalog.hash16);  -- 此语句无法在目标端回放。
    ```
    需要在目标端手动关闭blockchain_schema的防篡改属性后，才可以正常回放，此时目标端的blockchain_table等同于一张普通表，再次执行DML命令可以正常回放。

    SQL命令：

    ```
    ALTER SCHEMA blockchain_schema WITHOUT BLOCKCHAIN;
    CREATE TABLE blockchain_schema.blockchain_table (mes pg_catalog.int4, hash_a1d895 pg_catalog.hash16);
    ```

- 串行逻辑解码支持DDL特有约束：
  - sql_decoding插件不支持json格式的DDL。

解码格式

- JSON格式

  对于输入的DDL语句，SQL引擎解析器会通过语法、词法分析将其分解为解析树，解析树节点中包含了DDL的全部信息，并且执行器会根据解析树内容，执行系统元信息的修改。在执行器执行完成之后，便可以获取到DDL操作数据对象的search_path。本特性在执行器执行成功之后，对解析树信息以及执行器结果进行反解析，以还原出DDL原语句的全部信息。反解析的方式可以分解整个DDL语句，以方便输出JSON格式的DDL，用以适配异构数据库场景。

  CREATE TABLE语句在经过词法、语法分析之后，得到对应的CreateStmt解析树节点，节点中包含了表信息、列信息、分布式信息（DistributeBy结构体）、分区信息（PartitionState结构）等。通过反解析后，可输出的JSON格式如下：

  ```
  {"JDDL":{"fmt":"CREATE %{persistence}s TABLE %{if_not_exists}s %{identity}D %{table_elements}s %{with_clause}s %{compression}s","identity":{"object_name":"test_create_table_a","schema_name":"public"},"compression":"NOCOMPRESS","persistence":"","with_clause":{"fmt":"WITH (%{with:, }s)","with":[{"fmt":"%{label}s = %{value}L","label":{"fmt":"%{label}I","label":"orientation"},"value":"row"},{"fmt":"%{label}s = %{value}L","label":{"fmt":"%{label}I","label":"compression"},"value":"no"}]},"if_not_exists":"","table_elements":{"fmt":"(%{elements:, }s)","elements":[{"fmt":"%{name}I %{column_type}T","name":"a","column_type":{"typmod":"","typarray":false,"type_name":"int4","schema_name":"pg_catalog"}}]}}}
  ```


版权所有 © 华为云计算技术有限公司

可以看到，JSON格式中包含对象的search_path，其中的identity键标识schema为public，表名为test_create_table_a，其中%{persistence}s对应的字段如下，此SQL语句不含此字段所以为空。

```
[ [ GLOBAL | LOCAL ] [ TEMPORARY | TEMP ] | UNLOGGED ]
```

%{if_not_exists}s对应SQL语句中的字段，不含此字段所以为空：

```
[ IF NOT EXISTS ]
```

%{identity}D对应SQL语句中的字段：

```
table_name
```

%{table_elements}s对应SQL语句中的字段：

```
( column_name data_type )
```

%{with_clause}s对应SQL语句中的字段：

```
[ WITH ( {storage_parameter = value} [, ... ] ) ]
```

%{compression}s对应SQL语句中的字段：

```
[ COMPRESS | NOCOMPRESS ]
```

- decode-style指定格式

  输出的格式由decode-style参数控制，如当decode-style='j'时，输出格式如下：

  ```
  {"TDDL":"CREATE  TABLE  public.test_create_table_a (a pg_catalog.int4) WITH (orientation = 'row', compression = 'no') NOCOMPRESS"}
  ```

  其中语句中也包含Schema名称。

接口设计

- 新增控制参数

  a. 新增逻辑解码控制参数，用于控制DDL的反解析流程以及输出形式。可通过JDBC接口或者pg_logical_slot_peek_changes开启。

     - enable-ddl-decoding：默认false，不开启DDL语句的逻辑解码；值为true时，开启DDL语句的逻辑解码。
     - enable-ddl-json-format：默认false，传送TEXT格式的DDL反解析结果；值为true时，传送JSON格式的DDL反解析结果。

  b. 新增GUC参数

     - enable_logical_replication_ddl：默认为ON，ON状态下，逻辑复制可支持DDL，否则，不支持DDL。只有当ON状态下，才会对DDL执行结果进行反解析，并生成DDL的WAL日志。否则，不反解析也不生成WAL日志。

       enable_logical_replication_ddl的开关日志，以证明是否是用户修改了该参数导致逻辑解码不支持DDL。

- 新增日志

  新增DDL日志xl_logical_ddl_message，其类型为RM_LOGICALDDLMSG_ID。其定义如下：

名称	类型	意义
db_id	OID	数据库ID
rel_id	OID	表ID
csn	CommitSeqNo	CSN快照

 版权所有 © 华为云计算技术有限公司

名称	类型	意义
cid	CommandId	Command ID
tag_type	NodeTag	DDL类型
message_size	Size	日志内容长度
filter_message_size	Size	日志中白名单过滤信息长度
message	char *	DDL内容

使用步骤

步骤1 逻辑解码特性需提前设置GUC参数wal_level为logical，该参数需要重启生效。

```
gs_guc set -Z datanode -D $node_dir -c "wal_level = logical"
```

其中，$node_dir为数据库节点路径，用户可根据实际情况替换。

步骤2 以具有REPLICATION权限的用户登录GaussDB数据库主节点，使用如下命令连接数据库。

```
gsql -U user1 -W password -d db1 -p 16000 -r
```

其中，user1为用户名，password为密码，db1为需要连接的数据库名称，16000为数据库端口号，用户可根据实际情况替换。

步骤3 创建名称为slot1的逻辑复制槽。

```
gaussdb=# SELECT * FROM pg_create_logical_replication_slot('slot1', 'mppdb_decoding');
 slotname | xlog_position
----------+---------------
 slot1    | 0/3764C788
(1 row)
```

步骤4 在数据库中创建Package。

```
gaussdb=# CREATE OR REPLACE PACKAGE ldp_pkg1 IS
    var1 int:=1;  --公有变量
    var2 int:=2;
    PROCEDURE testpro1(var3 int);  --公有存储过程，可以被外部调用
END ldp_pkg1;
/
```

步骤5 读取复制槽slot1解码结果，可通过JDBC接口或者pg_logical_slot_peek_changes推进复制槽。

说明

- 逻辑解码选项请参见**逻辑解码选项**和新增控制参数。
- 并行解码中，在JDBC接口中改变参数decode_style可以决定解码格式：

  通过配置选项decode-style，指定解码格式。其取值为char型的字符'j'、't'或'b'，分别代表json格式、text格式及二进制格式。

```
gaussdb=# SELECT data FROM pg_logical_slot_peek_changes('slot1', NULL, NULL, 'enable-ddl-decoding',
'true', 'enable-ddl-json-format', 'false') WHERE data not like 'BEGIN%' AND data not like 'COMMIT%' AND
data not like '%dbe_pldeveloper.gs_source%';

                                                
```

```
CURRENT_USER IS  var1 int:=1;  --公有变量\n    var2 int:=2;\n
PROCEDURE testpro1(var3 int);  --公有存储过程，可以被外部调用\nEND  ldp_pkg1; \n /"}
(1 row)
```

步骤6 删除逻辑复制槽slot1，删除package ldp_pkg1。

```
gaussdb=# SELECT * FROM pg_drop_replication_slot('slot1');
 pg_drop_replication_slot
--------------------------

(1 row)

gaussdb=# DROP PACKAGE ldp_pkg1;
NOTICE:  drop cascades to function public.testpro1(integer)
DROP PACKAGE
```

----结束

7.1.6 逻辑解码数据找回功能

逻辑解码对外提供单节点DML数据找回能力，从WAL日志是否归档的角度分为在线日志找回和归档日志找回：

- 在线数据找回：使用pg_logical_get_area_changes函数，可以在线找回相关DML数据，具体使用参考pg_logical_get_area_changes函数的使用方法。
- 归档数据找回：OM_Agent通过pg_logical_get_area_changes函数提供OBS上归档日志的找回能力，OM_Agent从OBS下载已经归档的日志到对应节点上，并创建与WAL日志文件前8位同名的文件夹。解码完成后，将找回的数据存入文件并返回路径。具体参考《OM服务化参考》中“OM Agent接口 > 备份恢复相关接口 > 数据找回”章节。

约束

1. 解析的WAL日志级别为logical。
2. 数据表的复制标识必须为FULL，否则UPDATE和DELETE操作涉及到的被修改行不是全字段。
3. WAL日志记录的数据修改操作所对应的业务表，从找回起始位置到目前不能执行VACUUM FULL操作，否则该表VACUUM FULL之前的DML操作不会被数据找回。
4. WAL日志记录的数据修改操作所对应的业务表，从找回起始位置到目前，结构只能发生以下变化：
   a. 业务表结构不发生变化，即无DDL操作。
   b. 业务表在最后一列增加字段，且字段类型不能为带有默认值超过124字符长度的varchar类型。
   c. 业务表删除某一列字段。
   d. varchar字段增加原有长度。

   其余表结构变更导致该表relfilenode发生变化的场景不支持，表结构变更前的DML操作不会被数据找回。
5. 每条WAL日志不能超过500MB。
6. 不支持扩容前的xlog日志数据找回。
7. 集群的每个分片都会生成一个结果文件，多个文件不会进行合并。
8. 仅支持归档数据找回，且需要开启归档，若数据尚未归档，则无法通过本接口找回。

 版权所有 © 华为云计算技术有限公司

9. OM_Agent在下载之前会验证本地已用空间是否大于总空间的80%，如果大于则会会报错（需要额外空间用于存放解码文件），报错信息为："no enough space left on device, available space must be greater than 20%"。
10. 下载失败或解码失败后，都会将下载的WAL日志文件进行清理，如果清理不成功，不会强制结束程序，只会把错误信息记录到DN的日志中。
11. 由用户传入的时间，起始时间不能超出系统表gs_txn_lsn_time的最大时间，终止时间不能超过系统表gs_txn_lsn_time的最小时间，否则将会报错。
12. 每次下载的xlog日志量不超过3GB。
13. 每次解码数据量不超过max_process_memory 的10%。
14. 不支持同一节点并发调用数据找回接口。

 版权所有 © 华为云计算技术有限公司

8 分区表

本章节围绕分区表在大数据量场景下如何对保存的数据进行“查询优化”和“运维管理”出发，分六个章节对分区表使用进行系统性说明，包含语义、原理、约束限制等方面。

8.1 大容量数据库

8.1.1 大容量数据库背景介绍

随着处理数据量的日益增长和使用场景的多样化，数据库越来越多地面对容量大、数据多样化的场景。在过去数据库业界发展的20多年时间里，数据量从最初的MB、GB级数据量逐渐发展到现在的TB级数据量，在如此数据大规模、数据多样化的客观背景下，数据库管理系统（DBMS）在数据查询、数据管理方面提出了更高的要求，客观上要求数据库能够支持多种优化查找策略和管理运维方式。

在计算机科学经典的算法中，人们通常使用分治法（Divide and Conquer）解决场景和规模较大的问题。其基本思想就是把一个复杂的问题分成两个或更多的相同或相似的子问题，再把子问题分成更小的子问题直到最后子问题可以简单的直接求解，原问题的解可看成子问题的解的合并。对于大容量数据场景，数据库提供对数据进行“分治处理”的方式即分区，将逻辑数据库或其组成元素划分为不同的独立部分，每一个分区维护逻辑上存在相类似属性的数据，这样就把庞大的数据整体进行了切分，有利于数据的管理、查找和维护。

8.1.2 表分区技术

表分区技术（Table-Partitioning）通过将非常大的表或者索引从逻辑上切分为更小、更易管理的逻辑单元（分区），能够让用户对表查询、变更等语句操作具备更小的影响范围，能够让用户通过分区键（Partition Key）快速定位到数据所在的分区，从而避免在数据库中对大表的全量扫描，能够在不同的分区上并发进行DDL、DML操作。从用户使用的角度来看，表分区技术主要有以下三个方面能力：

1. 提升大容量数据场景查询效率：由于表内数据按照分区键进行逻辑分区，查询结果可以通过访问分区的子集而不是整个表来实现。这种分区剪枝技术可以提供数量级的性能增益。
2. 降低运维与查询的并发操作影响：降低DML语句、DDL语句并发场景的相互影响，在对一些大数据量以时间维度进行分区的场景下会明显受益。例如，新数据分区进行入库、实时点查操作，老数据分区进行数据清洗、分区合并等运维性质操作。

 版权所有 © 华为云计算技术有限公司

3. 提供大容量场景下灵活的数据运维管理方式：由于分区表从物理上对不同分区的数据做了表文件层面的隔离，每个分区可以具有单独的物理属性，如启用或禁用压缩、物理存储设置和表空间。同时它支持数据管理操作，如数据加载、索引创建和重建，以及分区级别的备份和恢复，而不是对整个表进行操作，从而减少了操作时间。

8.1.3 数据分区查找优化

分区表对数据查找方面的帮助主要体现在对分区键进行谓词查询场景，例如一张以月份Month作为分区键的表，如图8-1所示。如果以普通表的方式设计表结构则需要访问表全量的数据（Full Table Scan），如果以日期为分区键重新设计该表，那么原有的全表扫描会被优化成为分区扫描。当表内的数据量很大同时具有很长的历史周期时，由于扫描数据缩减所带来的性能提升会有明显的效果，如图8-2所示。

图 8-1 分区表示例图

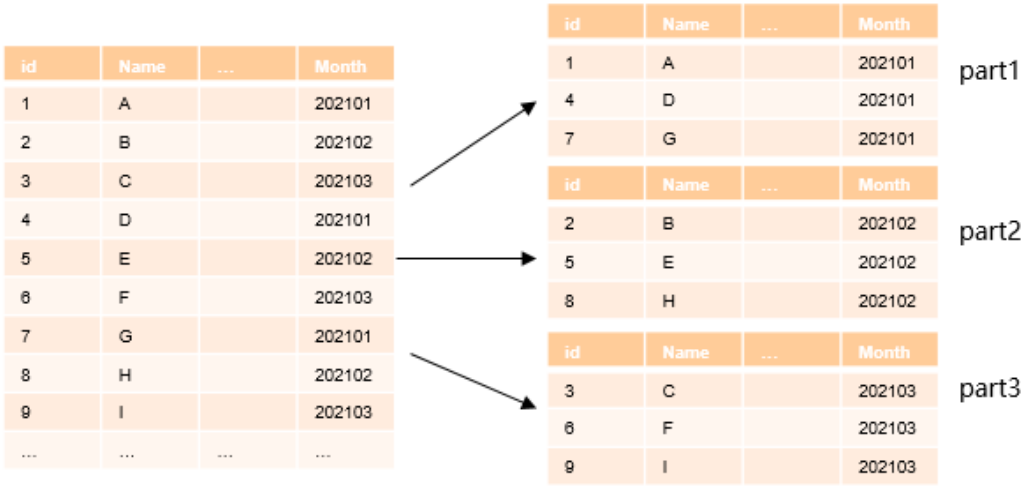
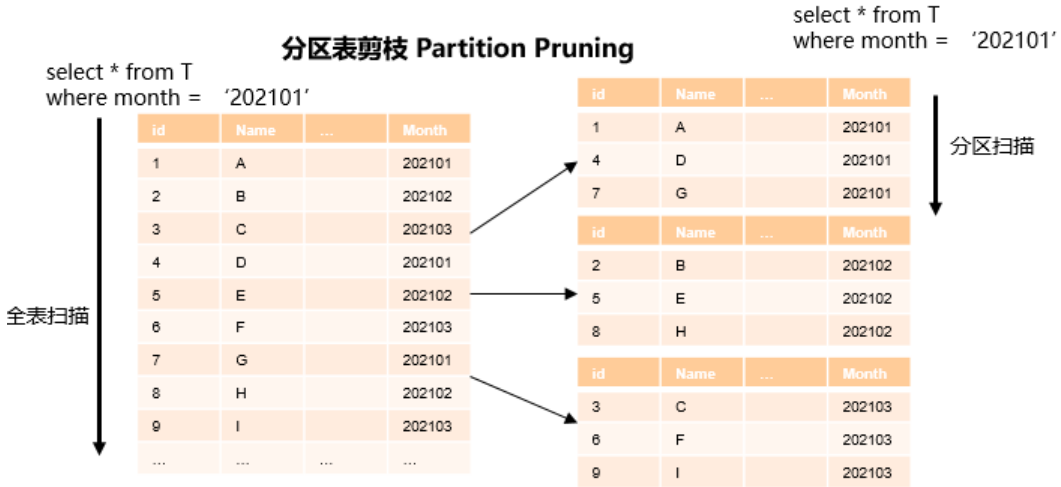


图 8-2 分区表剪枝示例图



8.1.4 数据分区运维管理

分区表技术为数据生命周期管理（Data Life Cycle Management，DLM）提供了灵活性的支持，数据生命周期管理是一组用于在数据的整个使用寿命中管理数据的过程和

版权所有 © 华为云计算技术有限公司

策略。其中一个重要组成部分是确定在数据生命周期的任何时间点存储数据的最合适和最经济高效的介质：日常操作中使用的较新数据存储在最快、可用性最高的存储层上，而不经常访问的较旧数据可能存储在成本较低、效率较低的存储层。较旧的数据也可能更新的频率较低，因此将数据压缩并存储为只读是有意义的。

分区表为实施DLM解决方案提供了理想的环境，通过不同分区使用不同表空间，最大限度在确保易用性的同时，实现了有效的数据生命周期的成本优化。这部分的设置由数据库运维人员在服务端设置操作完成，实际用户并不感知这一层面的优化设置，对用户而言逻辑上仍然是对同一张表的查询操作。此外不同分区可以分别实施备份、恢复、索引重建等运维性质的操作，能够对单个数据集不同子类进行分治操作，满足用户业务场景的差异化需求。

8.2 分区表介绍

分区表（Partitioned Table）指在单节点内对表数据内容按照分区键以及围绕分区键的分区策略对表进行逻辑切分。从数据分区的角度来看是一种水平分区（horizontal partition）策略方式。分区表增强了数据库应用程序的性能、可管理性和可用性，并有助于降低存储大量数据的总体拥有成本。分区允许将表、索引和索引组织的表细分为更小的部分，使这些数据库对象能够在更精细的粒度级别上进行管理和访问。GaussDB提供了丰富的分区策略和扩展，以满足不同业务场景的需求。由于分区策略的实现完全由数据库内部实现，对用户是完全透明的，因此它几乎可以在实施分区表优化策略以后做平滑迁移，无需潜在耗费人力物力的应用程序更改。本章围绕GaussDB分区表的基本概念从以下几个方面展开介绍：

1. 分区表基本概念：从表分区的基本概念出发，介绍分区表的catalog存储方式以及内部对应原理。
2. 分区策略：从分区表所支持的基本类型出发，介绍各种分区模式下对应的特性以及能够达到的优化特点和效果。

8.2.1 基本概念

8.2.1.1 分区表（母表）

实际对用户体现的表，用户对该表进行常规DML语句的增、删、查、改操作。通常使用在建表DDL语句显式的使用PARTITION BY语句进行定义，创建成功以后在pg_class表中新增一个entry，并且parttype列内容为'p'，表明该entry为分区表的母表。分区母表通常是一个逻辑形态，对应的表文件并不存放数据。

示例：t1_hash为一个分区表，分区类型为hash：

```
gaussdb=# CREATE TABLE t1_hash (c1 INT, c2 INT, c3 INT)
PARTITION BY HASH(c1)
(
    PARTITION p0,
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4,
    PARTITION p5,
    PARTITION p6,
    PARTITION p7,
    PARTITION p8,
    PARTITION p9
);

gaussdb=# \d+ t1_hash
                    Table "public.t1_hash"
```

 版权所有 © 华为云计算技术有限公司

```
Column |  Type    | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
c1     | integer |           | plain   |              |
c2     | integer |           | plain   |              |
c3     | integer |           | plain   |              |
Partition By HASH(c1)
Number of partitions: 10 (View pg_partition to check each partition range.)
Distribute By: HASH(c1)
Location Nodes: ALL DATANODES
Has OIDs: no
Options: orientation=row, compression=no

--查询t1_hash分区类型。
gaussdb=#  SELECT relname, parttype FROM pg_class WHERE relname = 't1_hash';
relname | parttype
---------+----------
t1_hash | p
(1 row)

--删除t1_hash。
gaussdb=# DROP TABLE t1_hash;
```

8.2.1.2 分区（分区子表、子分区）

分区表中实际保存数据的表，对应的entry通常保存在pg_partition中，各个子分区的parentid作为外键关联其分区母表在pg_class表中的OID列。

示例：t1_hash为一个分区表：

```
gaussdb=# CREATE TABLE t1_hash (c1 INT, c2 INT, c3 INT)
PARTITION BY HASH(c1)
(
    PARTITION p0,
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4,
    PARTITION p5,
    PARTITION p6,
    PARTITION p7,
    PARTITION p8,
    PARTITION p9
);

--查询t1_hash分区类型。
gaussdb=# SELECT oid, relname, parttype FROM pg_class WHERE relname = 't1_hash';
oid  | relname | parttype
-------+---------+----------
16685 | t1_hash | p
(1 row)

--查询t1_hash的分区信息。
gaussdb=# SELECT oid, relname, parttype, parentid FROM pg_partition WHERE parentid = 16685;
oid  | relname | parttype | parentid
-------+---------+----------+----------
16688 | t1_hash | r        |    16685
16689 | p0      | p        |    16685
16690 | p1      | p        |    16685
16691 | p2      | p        |    16685
16692 | p3      | p        |    16685
16693 | p4      | p        |    16685
16694 | p5      | p        |    16685
16695 | p6      | p        |    16685
16696 | p7      | p        |    16685
16697 | p8      | p        |    16685
16698 | p9      | p        |    16685
(11 rows)
```

 版权所有 © 华为云计算技术有限公司

```
--删除t1_hash
gaussdb=# DROP TABLE t1_hash;
```

8.2.1.3 分区键

分区键由一个或多个列组成，分区键值结合对应分区方法能够唯一确定某一元组所在的分区，通常在建表时通过PARTITION BY语句指定：

```
CREATE TABLE table_name (…) PARTITION BY part_strategy (partition_key) (…)
```

须知

范围分区表和列表分区表支持最多16列分区键，其他分区表只支持1列分区键。

8.2.2 分区策略

分区策略在使用DDL语句建表语句时通过PARTITION BY语句的语法指定，分区策略描述了在分区表中数据和分区路由映射规则。常见的分区类型有基于条件的Range分区、基于哈希散列函数的Hash分区、基于数据枚举的List列表分区：

```
CREATE TABLE table_name (…) PARTITION BY partition_strategy (partition_key) (…)
```

8.2.2.1 范围分区

范围分区（Range Partition）根据为每个分区建立分区键的值范围将数据映射到分区。范围分区是生产系统中最常见的分区类型，通常在以时间维度（Date、Time Stamp）描述数据场景中使用。范围分区有两种语法格式，示例如下：

1. VALUES LESS THAN的语法格式

   对于从句是VALUE LESS THAN的语法格式，范围分区策略的分区键最多支持16列。

   - 单列分区键示例如下：

     ```
     gaussdb=# CREATE TABLE range_sales
     (
        product_id      INT4 NOT NULL,
        customer_id     INT4 NOT NULL,
        time            DATE,
        channel_id      CHAR(1),
        type_id         INT4,
        quantity_sold   NUMERIC(3),
        amount_sold     NUMERIC(10,2)
     )
     PARTITION BY RANGE (time)
     (
        PARTITION date_202001 VALUES LESS THAN ('2020-02-01'),
        PARTITION date_202002 VALUES LESS THAN ('2020-03-01'),
        PARTITION date_202003 VALUES LESS THAN ('2020-04-01'),
        PARTITION date_202004 VALUES LESS THAN ('2020-05-01')
     );

     gaussdb=# DROP TABLE range_sales;
     ```

     其中date_202002表示2020年2月的分区，将包含分区键值从2020年2月1日到2020年2月29日的数据。

     每个分区都有一个VALUES LESS子句，用于指定分区的非包含上限。大于或等于该分区键的任何值都将添加到下一个分区。除第一个分区外，所有分区都具有由前一个分区的VALUES LESS子句指定的隐式下限。可以为最高分区定义MAXVALUE关键字，MAXVALUE表示一个虚拟无限值，其排序高于分区键的任何其他可能值，包括空值。

版权所有 © 华为云计算技术有限公司

- 多列分区键示例如下：

```
gaussdb=# CREATE TABLE range_sales_with_multiple_keys
(
    c1      INT4 NOT NULL,
    c2      INT4 NOT NULL,
    c3      CHAR(1)
)
PARTITION BY RANGE (c1,c2)
(
    PARTITION p1 VALUES LESS THAN (10,10),
    PARTITION p2 VALUES LESS THAN (10,20),
    PARTITION p3 VALUES LESS THAN (20,10)
);
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(9,5,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(9,20,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(9,21,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(10,5,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(10,15,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(10,20,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(10,21,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(11,5,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(11,20,'a');
gaussdb=# INSERT INTO range_sales_with_multiple_keys VALUES(11,21,'a');

gaussdb=# SELECT * FROM range_sales_with_multiple_keys PARTITION (p1);
 c1 | c2 | c3
----+----+----
  9 |  5 | a
  9 | 20 | a
  9 | 21 | a
 10 |  5 | a
(4 rows)

gaussdb=# SELECT * FROM range_sales_with_multiple_keys PARTITION (p2);
 c1 | c2 | c3
----+----+----
 10 | 15 | a
(1 row)

gaussdb=# SELECT * FROM range_sales_with_multiple_keys PARTITION (p3);
 c1 | c2 | c3
----+----+----
 10 | 20 | a
 10 | 21 | a
 11 |  5 | a
 11 | 20 | a
 11 | 21 | a
(5 rows)

gaussdb=# DROP TABLE range_sales_with_multiple_keys;
```

说明

多列分区的分区规则如下：

1. 从第一列开始比较。
2. 如果插入的当前列小于分区当前列边界值，则直接插入。
3. 如果插入的当前列等于分区当前列的边界值，则比较插入值的下一列与分区下一列边界值的大小。
4. 如果插入的当前列大于分区当前列的边界值，则换下一个分区进行比较。

2. START END语法格式

   对于从句是START END语法格式，范围分区策略的分区键最多支持1列。

   示例如下：

```
-- 创建表空间。
gaussdb=# CREATE TABLESPACE startend_tbs1 LOCATION '/home/omm/startend_tbs1';
gaussdb=# CREATE TABLESPACE startend_tbs2 LOCATION '/home/omm/startend_tbs2';
```

 版权所有 © 华为云计算技术有限公司

```
gaussdb=# CREATE TABLESPACE startend_tbs3 LOCATION '/home/omm/startend_tbs3';
gaussdb=# CREATE TABLESPACE startend_tbs4 LOCATION '/home/omm/startend_tbs4';
-- 创建临时schema。
gaussdb=# CREATE SCHEMA tpcds;
gaussdb=# SET CURRENT_SCHEMA TO tpcds;
-- 创建分区表，分区键是integer类型。
gaussdb=# CREATE TABLE tpcds.startend_pt (c1 INT, c2 INT)
TABLESPACE startend_tbs1
PARTITION BY RANGE (c2) (
    PARTITION p1 START(1) END(1000) EVERY(200) TABLESPACE startend_tbs2,
    PARTITION p2 END(2000),
    PARTITION p3 START(2000) END(2500) TABLESPACE startend_tbs3,
    PARTITION p4 START(2500),
    PARTITION p5 START(3000) END(5000) EVERY(1000) TABLESPACE startend_tbs4
)
ENABLE ROW MOVEMENT;

-- 查看分区表信息。
gaussdb=# SELECT relname, boundaries, spcname FROM pg_partition p JOIN pg_tablespace t ON
    p.reltablespace=t.oid and p.parentid='tpcds.startend_pt'::regclass ORDER BY 1;
relname | boundaries | spcname
-------------+------------+---------------
p1_0 | {1} | startend_tbs2
p1_1 | {201} | startend_tbs2
p1_2 | {401} | startend_tbs2
p1_3 | {601} | startend_tbs2
p1_4 | {801} | startend_tbs2
p1_5 | {1000} | startend_tbs2
p2 | {2000} | startend_tbs1
p3 | {2500} | startend_tbs3
p4 | {3000} | startend_tbs1
p5_1 | {4000} | startend_tbs4
p5_2 | {5000} | startend_tbs4
startend_pt | | startend_tbs1
(12 rows)

--清理示例
gaussdb=#  DROP TABLE tpcds.startend_pt;
DROP TABLE
gaussdb=# DROP SCHEMA tpcds;
DROP SCHEMA
```

8.2.2.2 哈希分区

哈希分区（Hash Partition）基于对分区键使用哈希算法将数据映射到分区。使用的哈希算法为GaussDB内置哈希算法，在分区键取值范围不倾斜（no data skew）的场景下，哈希算法在分区之间均匀分布行，使分区大小大致相同。因此哈希分区是实现分区间均匀分布数据的理想方法。哈希分区也是范围分区的一种易于使用的替代方法，尤其是当要分区的数据不是历史数据或没有明显的分区键时，示例如下：

```
gaussdb=# CREATE TABLE bmsql_order_line (
    ol_w_id         INTEGER   NOT NULL,
    ol_d_id         INTEGER   NOT NULL,
    ol_o_id         INTEGER   NOT NULL,
    ol_number       INTEGER   NOT NULL,
    ol_i_id         INTEGER   NOT NULL,
    ol_delivery_d   TIMESTAMP,
    ol_amount       DECIMAL(6,2),
    ol_supply_w_id  INTEGER,
    ol_quantity     INTEGER,
    ol_dist_info    CHAR(24)
)
--预先定义100个分区。
PARTITION BY HASH(ol_d_id)
(
    PARTITION p0,
    PARTITION p1,
    PARTITION p2,
```

 版权所有 © 华为云计算技术有限公司

```
    ...
    PARTITION p99
);
```

上述例子中，bmsql_order_line表的ol_d_id进行了分区，ol_d_id列是一个identifier性质的属性列，本身并不带有时间或者某一个特定维度上的区分。使用哈希分区策略来对其进行分表处理则是一个较为理想的选择。相比其他分区类型，除了预先确保分区键没有过多数据倾斜（某一、某几个值重复度高），只需要指定分区键和分区数即可创建分区，同时还能够确保每个分区的数据均匀，提升了分区表的易用性。

8.2.2.3 列表分区

列表分区（List Partition）能够通过在每个分区的描述中为分区键指定离散值列表来显式控制行如何映射到分区。列表分区的优势在于可以以枚举分区值方式对数据进行分区，可以对无序和不相关的数据集进行分组和组织。对于未定义在列表中的分区键值，可以使用默认分区（DEFAULT）来进行数据的保存，这样所有未映射到任何其他分区的行都不会生成错误。示例如下：

```
gaussdb=# CREATE TABLE bmsql_order_line (
    ol_w_id         INTEGER   NOT NULL,
    ol_d_id         INTEGER   NOT NULL,
    ol_o_id         INTEGER   NOT NULL,
    ol_number       INTEGER   NOT NULL,
    ol_i_id         INTEGER   NOT NULL,
    ol_delivery_d   TIMESTAMP,
    ol_amount       DECIMAL(6,2),
    ol_supply_w_id  INTEGER,
    ol_quantity     INTEGER,
    ol_dist_info    CHAR(24)
)
PARTITION BY LIST(ol_d_id)
(
    PARTITION p0 VALUES (1,4,7),
    PARTITION p1 VALUES (2,5,8),
    PARTITION p2 VALUES (3,6,9),
    PARTITION p3 VALUES (DEFAULT)
);

--清理示例
gaussdb=# DROP TABLE bmsql_order_line;
```

上述例子和之前给出的哈希分区的例子类似，同样通过ol_d_id列进行分区，但是在List分区中直接通过对ol_d_id的可能取值范围进行限定，不在列表中的数据会进入p3分区（DEFAULT）。相比哈希分区，List列表分区对分区键的可控性更好，往往能够准确的将目标数据保存在预想的分区中，但是如果列表值较多在分区定义时变得麻烦，该情况下推荐使用Hash分区。List、Hash分区往往都是处理无序、不相关的数据集进行分组和组织。

⚠ 注意

列表分区的分区键最多支持16列。如果分区键定义为1列，子分区定义时List列表中的枚举值不允许为NULL值；如果分区键定义为多列，子分区定义时List列表中的枚举值允许有NULL值。

8.2.2.4 分区表对导入操作的性能影响

在GaussDB内核实现中，分区表数据插入的处理过程相比非分区表增加分区路由部分的开销，因从整体上分区表场景的数据插入开销主要看成：（1）heap-insert基表插

版权所有 © 华为云计算技术有限公司

入；（2）partition-routing分区路由两个部分。其中heap基表插入解决tuple入库对应heap表的问题并且该部分普通表和分区表共用，而分区路由部分解决分区路由即tuple元组插入到对应partRel的问题。

因此对数据插入优化的侧重点如下：

1. 分区表基表Heap表插入：
   a. 算子底噪优化
   b. heap数据插入
   c. 索引插入build优化（带索引）
2. 分区表分区路由：
   a. 路由查找算法逻辑优化
   b. 路由底噪优化，包括分区表partRel句柄开启、新增的函数调用逻辑开销

说明

分区路由的性能主要通过大数据量的单条INSERT语句体现，UPDATE场景内部包含了查找对应要更新的元组进行DELETE操作然后再进行INSERT，因此不如单条INSERT语句场景直接。

不同分区类型的路由算法逻辑如表8-1所示：

表 8-1 路由算法逻辑

分区方式	路由算法复杂度	实现概述说明
范围分区（Range Partition）	O(logN)	基于二分binary-search实现
哈希分区（Hash-Partition）	O(1)	基于key-partOid哈希表实现
列表分区（List-Partition）	O(1)	基于key-partOid哈希表实现

注意

分区路由的主要处理逻辑根据导入数据元组的分区键计算其所在分区的过程，相比非分区表这部分为额外增加的开销，这部分开销在最终数据导入上的具体性能损失和服务器CPU处理能力、表宽度、磁盘/内存的实际容量相关，通常可以粗略认为：

- x86服务器场景下分区表相比普通表的导入性能会略低10%以内。
- ARM服务器场景下为20%，造成x86和ARM指向性能略微差异的主要原因是分区路由为in-memory计算强化场景，主流x86体系CPU在单核指令处理能力上略优于ARM。

8.2.3 分区基本使用

版权所有 © 华为云计算技术有限公司

8.2.3.1 创建分区表

创建分区表

由于SQL语言功能强大和灵活多样性，SQL语法树通常比复杂，分区表同样如此，分区表的创建可以理解成在原有非分区表的基础上新增表分区属性，因此分区表的语法接口可以看成是对原有非分区表CREATE TABLE语句进行扩展PARTITION BY语句部分，同时指定分区相关的三个核元素：

1. 分区类型（ partType ）： 描述分区表的分区策略，分别有RANGE/LIST/HASH。
2. 分区键（ partKey ）： 描述分区表的分区列，目前RANGE/LIST分区支持多列（ 不超过16列 ）分区键，HASH分区只支持单列分区。
3. 分区表达式（ partExpr ）： 描述分区表的具体分区表方式，即键值与分区的对应映射关系。

这三部分重要元素在建表语句的Partition By Clause子句中体现，*PARTITION BY partType* (*partKey*) (*partExpr[,partExpr]*…) 。示例如下：

```
CREATE TABLE [ IF NOT EXISTS ] partition_table_name
(
   [ /* 该部分继承于普通表的Create Table */
   { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
   | table_constraint
   | LIKE source_table [ like_option [...] ] }[, ... ]
   ]
)
[ WITH ( {storage_parameter = value} [, ... ] ) ]
[ COMPRESS | NOCOMPRESS ]
[ TABLESPACE tablespace_name ]
/* 范围分区场景 */
PARTITION BY RANGE (partKey) (
   { partition_start_end_item [, ... ] | partition_less_then_item [, ... ] }
)
/* 列表分区场景 */
PARTITION BY LIST (partKey)
(
   PARTITION partition_name VALUES (list_values_clause) [ TABLESPACE tablespace_name [, ... ] ]
...
)
/* 哈希分区场景 */
PARTITION BY HASH (partKey) (
   PARTITION partition_name [ TABLESPACE tablespace_name [, ... ] ]
...
)
/* 开启/关闭分区表行迁移 */
[ { ENABLE | DISABLE } ROW MOVEMENT ];
```

规格约束：

1. Range/List分区最大支持16个分区键，Hash分区只支持1个分区键。
2. 除哈希分区外，分区键不能插入空值，否则DML语句会进行报错处理。唯一例外：Range分区表定义有MAXVALUE分区/List分区表定义有DEFAULT分区。
3. 分区数最大值为1048575个，可以满足大部分业务场景的诉求。但分区数增加会导致系统中文件数增加，影响系统的性能，一般对于单个表而言不建议分区数超过200。

修改分区属性

分区表和分区相关的部分属性可以使用类似非分区表的ALTER-TABLE命令进行分区属性修改，常用的分区属性修改语句包括：

 版权所有 © 华为云计算技术有限公司

1. 增加分区
2. 删除分区
3. 删除/清空分区数据
4. 切割分区
5. 合并分区
6. 移动分区
7. 交换分区
8. 重命名分区

以上常用的分区属性变更语句基于对普通表ALTER TABLE语句进行扩展，在使用方式上大部分使用方式类似，分区表属性变更的基本语法框架示例如下：

```
/* 基本alter table语法 */
ALTER TABLE [ IF EXISTS ] { table_name [*] | ONLY table_name | ONLY ( table_name )}
action [, ... ];
```

分区表ALTER TABLE语句使用方法请参见**分区表运维管理**、《开发指南》中“SQL参考 > SQL语法 > ALTER TABLE PARTITION”章节。

8.2.3.2 使用和管理分区表

分区表支持大部分非分区表的相关功能，具体可以参考《开发指南》中常规表的各类操作语法相关资料。

除此之外，分区表还支持大量的分区级操作命令，包括分区级DQL/DML（如SELECT、INSERT、UPDATE、DELETE、UPSERT、MERGE INTO）、分区级DDL（如ADD、DROP、TRUNCATE、EXCHANGE、SPLIT、MERGE、MOVE、RENAME）、分区VACUUM/ANALYZE、分类分区索引等。相关命令使用方法请参见**分区表DQL/DML**、**分区索引**、**分区表运维管理**、以及《开发指南》中各个语法命令对应的章节。

分区级操作命令一般通过指定分区名或者分区值的方式进行，比如语法命令可能是如下情形：

```
sql_action [ t_name ] { PARTITION | SUBPARTITION } { p_name | (p_name) };
sql_action [ t_name ] { PARTITION | SUBPARTITION } FOR (p_value);
```

通过指定分区名p_name或指定分区值p_value来定向操作某个特定分区，此时业务只会作用于对象分区，而不会影响其他任何分区。如果通过指定分区名p_name来执行业务，数据库会匹配p_name对应的分区，该分区不存在则业务抛出异常；如果通过指定分区值p_value来执行业务，数据库会匹配p_value值所属分区。

比如定义有如下的分区表：

```
gaussdb=# CREATE TABLE list_01
(
   id   INT,
   role VARCHAR(100),
   data VARCHAR(100)
)
PARTITION BY LIST (id)
(
   PARTITION p_list_1 VALUES(0,1,2,3,4),
   PARTITION p_list_2 VALUES(5,6,7,8,9),
   PARTITION p_list_3 VALUES(DEFAULT)
);

-- 清理示例
gaussdb=# DROP TABLE list_01;
```

指定分区业务中，PARTITION p_list_1与PARTITION FOR (4) 等价，为同一个分区；PARTITION p_list_3与PARTITION FOR (12) 等价，为同一个分区。

 版权所有 © 华为云计算技术有限公司

8.2.3.3 分区表 DQL/DML

由于分区的实现完全体现在数据库内核中，用户对分区表的DQL/DML与非分区表相比，在语法上没有任何区别。

出于分区表的易用性考虑，GaussDB支持指定分区的DQL/DML操作，指定分区可以通过PARTITION (partname)或者PARTITION FOR (partvalue)来进行。对于二级分区，可以通过SUBPARTITION(subpartname)或者SUBPARTITION FOR (subpartvalue)指定具体的二级分区。指定分区执行DQL/DML时，若插入的数据不属于目标分区，则业务报错；若查询的数据不属于目标分区，则跳过该数据的处理。

指定分区DQL/DML支持以下几类语法：

1. 查询（SELECT）
2. 插入（INSERT）
3. 更新（UPDATE）
4. 删除（DELETE）
5. 插入或更新（UPSERT）
6. 合并（MERGE INTO）

指定分区做DQL/DML的示例如下：

```
--创建分区表list_02。
gaussdb=# CREATE TABLE IF NOT EXISTS list_02
(
    id   INT,
    role VARCHAR(100),
    data VARCHAR(100)
)
PARTITION BY LIST (id)
(
    PARTITION p_list_2 VALUES(0,1,2,3,4,5,6,7,8,9),
    PARTITION p_list_3 VALUES(10,11,12,13,14,15,16,17,18,19),
    PARTITION p_list_4 VALUES( DEFAULT ),
    PARTITION p_list_5 VALUES(20,21,22,23,24,25,26,27,28,29),
    PARTITION p_list_6 VALUES(30,31,32,33,34,35,36,37,38,39),
    PARTITION p_list_7 VALUES(40,41,42,43,44,45,46,47,48,49)
) ENABLE ROW MOVEMENT;
--导入数据。
INSERT INTO list_02 VALUES(null, 'alice', 'alice data');
INSERT INTO list_02 VALUES(2, null, 'bob data');
INSERT INTO list_02 VALUES(null, null, 'peter data');

--对指定分区进行查询。
-- 查询分区表全部数据。
gaussdb=# SELECT * FROM list_02 ORDER BY data;
 id | role  |    data
----+-------+------------
    | alice | alice data
  2 |       | bob data
    |       | peter data
(3 rows)
--查询分区p_list_2数据。
gaussdb=# SELECT * FROM list_02 PARTITION (p_list_2) ORDER BY data;
 id | role |   data
----+------+----------
  2 |      | bob data
(1 row)
--查询(100)所对应的分区的数据，即分区p_list_4。
gaussdb=# SELECT * FROM list_02 PARTITION FOR (100) ORDER BY data;
 id | role  |    data
----+-------+------------
    | alice | alice data
    |       | peter data
```

 版权所有 © 华为云计算技术有限公司

```
(2 rows)

--对指定分区做IUD。
-- 删除分区p_list_5中的全部数据。
gaussdb=# DELETE FROM list_02 PARTITION (p_list_5);
--指定分区p_list_7插入数据，由于数据不符合该分区约束，插入报错。
gaussdb=# INSERT INTO list_02 PARTITION (p_list_7) VALUES(null, 'cherry', 'cherry data');
ERROR:  inserted partition key does not map to the table partition
--将分区值100所属分区，即分区p_list_4的数据进行更新。
gaussdb=# UPDATE list_02 PARTITION FOR (100) SET data = '';

--UPSERT。
gaussdb=# INSERT INTO list_02 (id, role, data) VALUES (1, 'test', 'testdata') ON DUPLICATE KEY UPDATE
role = VALUES(role), data = VALUES(data);

--MERGE INTO。
gaussdb=# CREATE TABLE IF NOT EXISTS list_tmp
(
    id   INT,
    role VARCHAR(100),
    data VARCHAR(100)
)
PARTITION BY LIST (id)
(
    PARTITION p_list_2 VALUES(0,1,2,3,4,5,6,7,8,9),
    PARTITION p_list_3 VALUES(10,11,12,13,14,15,16,17,18,19),
    PARTITION p_list_4 VALUES( DEFAULT ),
    PARTITION p_list_5 VALUES(20,21,22,23,24,25,26,27,28,29),
    PARTITION p_list_6 VALUES(30,31,32,33,34,35,36,37,38,39),
    PARTITION p_list_7 VALUES(40,41,42,43,44,45,46,47,48,49)) ENABLE ROW MOVEMENT;

gaussdb=# MERGE INTO list_tmp target
USING list_02 source
ON (target.id = source.id)
WHEN MATCHED THEN
  UPDATE SET target.data = source.data,
             target.role = source.role
WHEN NOT MATCHED THEN
  INSERT (id, role, data)
  VALUES (source.id, source.role, source.data);

--删除表。
gaussdb=#
DROP TABLE list_02;
DROP TABLE list_tmp;
```

8.3 分区表查询优化

说明

本节示例对应explain_perf_mode参数值为normal。

8.3.1 分区剪枝

分区剪枝是GaussDB提供的一种分区表查询优化技术，数据库SQL引擎会根据查询条件，只扫描特定的部分分区。分区剪枝是自动触发的，当分区表查询条件符合剪枝场景时，会自动触发分区剪枝。根据剪枝阶段的不同，分区剪枝分为静态剪枝和动态剪枝，静态剪枝在优化器阶段进行，在生成计划之前，数据库已经知道需要访问的分区信息；动态剪枝在执行器阶段进行（执行开始/执行过程中），在生成计划时，数据库并不知道需要访问的分区信息，只是判断“可以进行分区剪枝”，具体的剪枝信息由执行器决定。

只有分区表页面扫描和Local索引扫描才会触发分区剪枝，Global索引没有分区的概念，不需要进行剪枝。

 版权所有 © 华为云计算技术有限公司

8.3.1.1 分区表静态剪枝

对于检索条件中分区键上带有常数的分区表查询语句，在优化器阶段将对indexscan、bitmap indexscan、indexonlyscan等算子中包含的检索条件作为剪枝条件，完成分区的筛选。算子包含的检索条件中需要至少包含一个分区键字段，对于含有多个分区键的分区表，包含任意分区键子集即可。

静态剪枝支持范围如下所示：

1. 支持分区类型：范围分区、哈希分区、列表分区。
2. 支持表达式类型：比较表达式（<，<=，=，>=，>）、逻辑表达式、数组表达式。

⚠ 注意

目前静态剪枝不支持子查询表达式。

为了支持分区表剪枝，在计划生成时会将分区键上的过滤条件强制转换为分区键类型，该操作与隐式类型转换规则存在差异，可能导致相同条件在分区键上转换报错，非分区键无报错的情况。

- 静态剪枝支持的典型场景具体示例如下：
  a. 比较表达式

```
--创建分区表。
gaussdb=# CREATE TABLE t1 (c1 int, c2 int)
PARTITION BY RANGE (c1)
(
    PARTITION p1 VALUES LESS THAN(10),
    PARTITION p2 VALUES LESS THAN(20),
    PARTITION p3 VALUES LESS THAN(MAXVALUE)
);
gaussdb=# SET max_datanode_for_plan = 1;

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 = 1;
                      QUERY PLAN
-------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: datanode1
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = 1

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = 1
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 = 1)
           Selected Partitions:  1

(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 < 1;
                      QUERY PLAN
-------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 < 1

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 < 1
```


版权所有 © 华为云计算技术有限公司

```
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 < 1)
           Selected Partitions:  1

(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 > 11;
                        QUERY PLAN
-----------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 > 11

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 > 11
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 2
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 > 11)
           Selected Partitions:  2..3

(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 is NULL;
                         QUERY PLAN
-----------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: datanode1
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 IS NULL

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 IS NULL
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 IS NULL)
           Selected Partitions:  3

(15 rows)
```

b. 逻辑表达式

```
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 = 1 AND c2 = 2;
                            QUERY PLAN
-------------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: datanode1
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = 1 AND c2 = 2

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = 1 AND c2 = 2
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: ((t1.c1 = 1) AND (t1.c2 = 2))
           Selected Partitions:  1
```

 版权所有 © 华为云计算技术有限公司

```
(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 = 1 OR c1 = 2;
                          QUERY PLAN
------------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = 1 OR c1 = 2

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = 1 OR c1 = 2
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: ((t1.c1 = 1) OR (t1.c1 = 2))
           Selected Partitions:  1

(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE NOT c1 = 1;
                        QUERY PLAN
-----------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE NOT c1 = 1

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE NOT c1 = 1
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 3
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 <> 1)
           Selected Partitions:  1..3

(15 rows)
```

c. 数组表达式

```
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 IN (1, 2, 3);
                            QUERY PLAN
-------------------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = ANY (ARRAY[1, 2, 3])

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = ANY (ARRAY[1, 2, 3])
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 = ANY ('{1,2,3}'::integer[]))
           Selected Partitions:  1

(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 = ALL(ARRAY[1,
2, 3]);
                              QUERY PLAN
-------------------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
```

 版权所有 © 华为云计算技术有限公司

```
  Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = ALL (ARRAY[1, 2, 3])

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = ALL (ARRAY[1, 2, 3])
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 0
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 = ALL ('{1,2,3}'::integer[]))
           Selected Partitions:  NONE

(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 = ANY(ARRAY[1,
2, 3]);
                              QUERY PLAN
-----------------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = ANY (ARRAY[1, 2, 3])

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = ANY (ARRAY[1, 2, 3])
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 = ANY ('{1,2,3}'::integer[]))
           Selected Partitions:  1

(15 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 =
SOME(ARRAY[1, 2, 3]);
                              QUERY PLAN
-----------------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = ANY (ARRAY[1, 2, 3])

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = ANY (ARRAY[1, 2, 3])
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 1
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 = ANY ('{1,2,3}'::integer[]))
           Selected Partitions:  1

(15 rows)
```

- 静态剪枝不支持的典型场景具体示例如下：

  子查询表达式

```
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 WHERE c1 = ALL(SELECT c2
FROM t1 WHERE c1 > 10);
                          QUERY PLAN
-----------------------------------------------------------------------------
 Streaming (type: GATHER)
   Output: public.t1.c1, public.t1.c2
   Node/s: All datanodes
   ->  Partition Iterator
         Output: public.t1.c1, public.t1.c2
         Iterations: 3
         ->  Partitioned Seq Scan on public.t1
```

 版权所有 © 华为云计算技术有限公司

```
            Output: public.t1.c1, public.t1.c2
            Distribute Key: public.t1.c1
            Filter: (SubPlan 1)
            Selected Partitions:  1..3
            SubPlan 1
              ->  Materialize
                    Output: public.t1.c2
                    ->  Streaming(type: BROADCAST)
                          Output: public.t1.c2
                          Spawn on: All datanodes
                          Consumer Nodes: All datanodes
                          ->  Partition Iterator
                                Output: public.t1.c2
                                Iterations: 2
                                ->  Partitioned Seq Scan on public.t1
                                      Output: public.t1.c2
                                      Distribute Key: public.t1.c1
                                      Filter: (public.t1.c1 > 10)
                                      Selected Partitions:  2..3
(26 rows)

--清理示例环境。
gaussdb=# DROP TABLE t1;
```

8.3.1.2 分区表动态剪枝

对于检索条件中存在带有变量的分区表查询语句，由于优化器阶段无法获取用户的绑定参数，因此优化器阶段仅能完成indexscan、bitmapindexscan、indexonlyscan等算子检索条件的解析，后续会在执行器阶段获得绑定参数后，完成分区筛选。算子包含的检索条件中需要至少包含一个分区键字段，对于含有多个分区键的分区表，包含任意分区键子集即可。目前分区表动态剪枝仅支持PBE（Prepare/Bind/Execute）场景和参数化路径场景。

8.3.1.2.1 PBE 动态剪枝

PBE动态剪枝支持范围如下所示：

1. 支持分区类型：范围分区、哈希分区、列表分区。
2. 支持表达式类型：比较表达式（<，<=，=，>=，>）、逻辑表达式、数组表达式。
3. 支持隐式类型转换和函数：对于类型可以相互转换的场景和immutable函数可以支持PBE动态剪枝

⚠ 注意

- PBE动态剪枝支持表达式、隐式转换、immutable函数和stable函数，不支持子查询表达式和volatile函数。对于stable函数，如to_timestamp等类型转换函数，可能会受GUC参数变化，影响剪枝结果。为了保持性能优化，此情况可以通过analyze表重新生成gplan解决。
- 由于PBE动态剪枝是基于generic plan的剪枝，所以判断语句是否能PBE动态剪枝时，需要设置参数plan_cache_mode = 'force_generic_plan'，排除custom plan的干扰。

- PBE动态剪枝支持的典型场景具体示例如下：

  a. 比较表达式。

  ```
  gaussdb=#
  --创建分区表
  ```


版权所有 © 华为云计算技术有限公司

```
CREATE TABLE t1 (c1 int, c2 int)
PARTITION BY RANGE (c1)
(
    PARTITION p1 VALUES LESS THAN(10),
    PARTITION p2 VALUES LESS THAN(20),
    PARTITION p3 VALUES LESS THAN(MAXVALUE)
);

gaussdb=# PREPARE p1(int) AS SELECT * FROM t1 WHERE c1 = $1;
PREPARE
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) EXECUTE p1(1);
                        QUERY PLAN
------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: datanode1
   Node expr: $1
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = $1

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = $1
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: PART
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: (t1.c1 = $1)
           Selected Partitions:  1 (pbe-pruning)

(16 rows)
```

b. 逻辑表达式。

```
gaussdb=# PREPARE p2(INT, INT) AS SELECT * FROM t1 WHERE c1 = $1 AND c2 = $2;
PREPARE
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) EXECUTE p2(1, 2);
                              QUERY PLAN
-----------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: datanode1
   Node expr: $1
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = $1 AND c2 = $2

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = $1 AND c2 = $2
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: PART
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: ((t1.c1 = $1) AND (t1.c2 = $2))
           Selected Partitions:  1 (pbe-pruning)

(16 rows)
```

c. 类型转换触发隐式转换。

```
gaussdb=# set plan_cache_mode = 'force_generic_plan';
gaussdb=# PREPARE p3(TEXT) AS SELECT * FROM t1 WHERE c1 = $1;
PREPARE
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) EXECUTE p3('12');
                            QUERY PLAN
------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: datanode1
   Node expr: $1
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1 = $1::bigint

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1 = $1::bigint
 Datanode Name: datanode1
```

 版权所有 © 华为云计算技术有限公司

```
  Partition Iterator
    Output: c1, c2
    Iterations: PART
    ->  Partitioned Seq Scan on public.t1
          Output: c1, c2
          Filter: (t1.c1 = ($1)::bigint)
          Selected Partitions:  2 (pbe-pruning)

(16 rows)
```

- PBE动态剪枝不支持的典型场景具体示例如下：

  a. 子查询表达式。

```
gaussdb=# PREPARE p4(INT) AS SELECT * FROM t1 WHERE c1 = ALL(SELECT c2 FROM t1
WHERE c1 > $1);
PREPARE
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) EXECUTE p4(1);
                          QUERY PLAN
------------------------------------------------------------------------------
 Streaming (type: GATHER)
   Output: public.t1.c1, public.t1.c2
   Node/s: All datanodes
   ->  Partition Iterator
         Output: public.t1.c1, public.t1.c2
         Iterations: 3
         ->  Partitioned Seq Scan on public.t1
               Output: public.t1.c1, public.t1.c2
               Distribute Key: public.t1.c1
               Filter: (SubPlan 1)
               Selected Partitions:  1..3
               SubPlan 1
                 ->  Materialize
                       Output: public.t1.c2
                       ->  Streaming(type: BROADCAST)
                             Output: public.t1.c2
                             Spawn on: All datanodes
                             Consumer Nodes: All datanodes
                             ->  Partition Iterator
                                   Output: public.t1.c2
                                   Iterations: 3
                                   ->  Partitioned Seq Scan on public.t1
                                         Output: public.t1.c2
                                         Distribute Key: public.t1.c1
                                         Filter: (public.t1.c1 > 1)
                                         Selected Partitions:  1..3
(26 rows)
```

  b. 类型转换无法直接触发隐式转换。

```
gaussdb=# PREPARE p5(name) AS SELECT * FROM t1 WHERE c1 = $1;
PREPARE
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) EXECUTE p5('12');
                          QUERY PLAN
------------------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM public.t1 WHERE c1::text = '12'::text

 Remote SQL: SELECT c1, c2 FROM public.t1 WHERE c1::text = '12'::text
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 3
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Filter: ((t1.c1)::text = '12'::text)
           Selected Partitions:  1..3

(15 rows)
```

 版权所有 © 华为云计算技术有限公司

c. stable/volatile函数。

```
gaussdb=# create sequence seq;
gaussdb=# PREPARE p6(TEXT) AS SELECT * FROM t1 WHERE c1 = currval($1);--volatile函数不支持剪枝。
PREPARE
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) EXECUTE p6('seq');
                              QUERY PLAN
-----------------------------------------------------------------------
 Data Node Scan
   Output: t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT c1, c2 FROM ONLY public.t1 WHERE true
   Coordinator quals: ((t1.c1)::numeric = currval(('seq'::text)::regclass))

 Remote SQL: SELECT c1, c2 FROM ONLY public.t1 WHERE true
 Datanode Name: datanode1
   Partition Iterator
     Output: c1, c2
     Iterations: 3
     ->  Partitioned Seq Scan on public.t1
           Output: c1, c2
           Selected Partitions:  1..3

(15 rows)

--清理示例环境。
gaussdb=# DROP TABLE t1;
```

8.3.1.2.2 参数化路径动态剪枝

参数化路径动态剪枝支持范围如下所示：

1. 支持分区类型：范围分区、哈希分区、列表分区。
2. 支持算子类型：indexscan、indexonlyscan、bitmapscan。
3. 支持表达式类型：比较表达式（<，<=，=，>=，>）、逻辑表达式。

⚠ 注意

参数化路径动态剪枝不支持子查询表达式，不支持stable和volatile函数，不支持跨QueryBlock参数化路径，不支持BitmapOr、BitmapAnd算子。

- 参数化路径动态剪枝支持的典型场景具体示例如下：

  a. 比较表达式

```
--创建分区表和索引
gaussdb=# CREATE TABLE t1 (c1 INT, c2 INT)
PARTITION BY RANGE (c1)
(
    PARTITION p1 VALUES LESS THAN(10),
    PARTITION p2 VALUES LESS THAN(20),
    PARTITION p3 VALUES LESS THAN(MAXVALUE)
);
gaussdb=# CREATE TABLE t2 (c1 INT, c2 INT)
PARTITION BY RANGE (c1)
(
    PARTITION p1 VALUES LESS THAN(10),
    PARTITION p2 VALUES LESS THAN(20),
    PARTITION p3 VALUES LESS THAN(MAXVALUE)
);
gaussdb=# CREATE INDEX t1_c1 ON t1(c1) LOCAL;
gaussdb=# CREATE INDEX t2_c1 ON t2(c1) LOCAL;
gaussdb=# CREATE INDEX t1_c2 ON t1(c2) LOCAL;
gaussdb=# CREATE INDEX t2_c2 ON t2(c2) LOCAL;
```


版权所有 © 华为云计算技术有限公司

```
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT /*+ nestloop(t1 t2) indexscan(t1)
indexscan(t2) */ * FROM t2 JOIN t1 ON t1.c1 = t2.c1;
                                                QUERY
PLAN
------------------------------------------------------------------------------------------------------------------
-----------------------------------
 Data Node Scan
   Output: t2.c1, t2.c2, t1.c1, t1.c2
   Node/s: All datanodes
   Remote query: SELECT/*+ NestLoop(t1 t2) IndexScan(t1) IndexScan(t2)*/ t2.c1, t2.c2, t1.c1,
t1.c2 FROM public.t2 JOIN public.t1 ON t1.c1 = t2.c1

 Remote SQL: SELECT/*+ NestLoop(t1 t2) IndexScan(t1) IndexScan(t2)*/ t2.c1, t2.c2, t1.c1, t1.c2
FROM public.t2 JOIN public.t1 ON t1.c1 = t2.c1
 Datanode Name: datanode1
   Nested Loop
     Output: t2.c1, t2.c2, t1.c1, t1.c2
     ->  Partition Iterator
           Output: t2.c1, t2.c2
           Iterations: 3
           ->  Partitioned Index Scan using t2_c1 on public.t2
                 Output: t2.c1, t2.c2
                 Selected Partitions:  1..3
     ->  Partition Iterator
           Output: t1.c1, t1.c2
           Iterations: PART
           ->  Partitioned Index Scan using t1_c1 on public.t1
                 Output: t1.c1, t1.c2
                 Index Cond: (t1.c1 = t2.c1)
                 Selected Partitions:  1..3 (ppi-pruning)

(23 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT /*+ nestloop(t1 t2) indexscan(t1)
indexscan(t2) */ * FROM t2 JOIN t1 ON t1.c1 < t2.c1;
                          QUERY PLAN
------------------------------------------------------------------------
 Streaming (type: GATHER)
   Output: t2.c1, t2.c2, t1.c1, t1.c2
   Node/s: All datanodes
   ->  Nested Loop
         Output: t2.c1, t2.c2, t1.c1, t1.c2
         ->  Streaming(type: BROADCAST)
               Output: t2.c1, t2.c2
               Spawn on: All datanodes
               Consumer Nodes: All datanodes
               ->  Partition Iterator
                     Output: t2.c1, t2.c2
                     Iterations: 3
                     ->  Partitioned Seq Scan on public.t2
                           Output: t2.c1, t2.c2
                           Distribute Key: t2.c1
                           Selected Partitions:  1..3
         ->  Partition Iterator
               Output: t1.c1, t1.c2
               Iterations: PART
               ->  Partitioned Index Scan using t1_c1 on public.t1
                     Output: t1.c1, t1.c2
                     Distribute Key: t1.c1
                     Index Cond: (t1.c1 < t2.c1)
                     Selected Partitions:  1..3 (ppi-pruning)
(24 rows)

gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT /*+ nestloop(t1 t2) indexscan(t1)
indexscan(t2) */ * FROM t2 JOIN t1 ON t1.c1 < t2.c1;
                          QUERY PLAN
------------------------------------------------------------------------
 Streaming (type: GATHER)
```

 版权所有 © 华为云计算技术有限公司

```
  Output: t2.c1, t2.c2, t1.c1, t1.c2
  Node/s: All datanodes
  ->  Nested Loop
        Output: t2.c1, t2.c2, t1.c1, t1.c2
        ->  Streaming(type: BROADCAST)
              Output: t2.c1, t2.c2
              Spawn on: All datanodes
              Consumer Nodes: All datanodes
              ->  Partition Iterator
                    Output: t2.c1, t2.c2
                    Iterations: 3
                    ->  Partitioned Seq Scan on public.t2
                          Output: t2.c1, t2.c2
                          Distribute Key: t2.c1
                          Selected Partitions:  1..3
        ->  Partition Iterator
              Output: t1.c1, t1.c2
              Iterations: PART
              ->  Partitioned Index Scan using t1_c1 on public.t1
                    Output: t1.c1, t1.c2
                    Distribute Key: t1.c1
                    Index Cond: (t1.c1 > t2.c1)
                    Selected Partitions:  1..3 (ppi-pruning)
(24 rows)
```

b. 逻辑表达式

```
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT /*+ nestloop(t1 t2) indexscan(t1)
indexscan(t2) */ * FROM t2 JOIN t1 ON t1.c1 = t2.c1 AND t1.c2 = 2;
                                                
```

```
Streaming (type: GATHER)
  Output: t2.c1, t2.c2, t1.c1, t1.c2
  Node/s: All datanodes
  ->  Nested Loop
        Output: t2.c1, t2.c2, t1.c1, t1.c2
        ->  Streaming(type: BROADCAST)
              Output: t2.c1, t2.c2
              Spawn on: All datanodes
              Consumer Nodes: All datanodes
              ->  Partition Iterator
                    Output: t2.c1, t2.c2
                    Iterations: 3
                    ->  Partitioned Seq Scan on public.t2
                          Output: t2.c1, t2.c2
                          Distribute Key: t2.c1
                          Selected Partitions:  1..3
        ->  Partition Iterator
              Output: t1.c1, t1.c2
              Iterations: 3
              ->  Partitioned Bitmap Heap Scan on public.t1
                    Output: t1.c1, t1.c2
                    Distribute Key: t1.c1
                    Recheck Cond: ((t1.c1 = t2.c1) OR (t1.c2 = 2))
                    Selected Partitions:  1..3
                    ->  BitmapOr
                          ->  Partitioned Bitmap Index Scan on t1_c1
                                Index Cond: (t1.c1 = t2.c1)
                          ->  Partitioned Bitmap Index Scan on t1_c2
                                Index Cond: (t1.c2 = 2)
(29 rows)
```

b. 隐式转换

```
gaussdb=# CREATE TABLE t3(c1 TEXT, c2 INT);
CREATE TABLE
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 JOIN t3 ON t1.c1 = t3.c1;
WARNING:  Statistics in some tables or columns(public.t1.c1, public.t3.c1) are not collected.
HINT:  Do analyze for them in order to generate optimized plan.
                          QUERY PLAN
------------------------------------------------------------------------
 Streaming (type: GATHER)
   Output: t1.c1, t1.c2, t3.c1, t3.c2
   Node/s: All datanodes
   ->  Nested Loop
         Output: t1.c1, t1.c2, t3.c1, t3.c2
         Join Filter: (t1.c1 = (lengthb(t3.c1)))
         ->  Partition Iterator
               Output: t1.c1, t1.c2
               Iterations: 3
               ->  Partitioned Index Scan using t1_c1 on public.t1
                     Output: t1.c1, t1.c2
                     Distribute Key: t1.c1
                     Selected Partitions:  1..3
         ->  Materialize
               Output: t3.c1, t3.c2, (lengthb(t3.c1))
               ->  Streaming(type: REDISTRIBUTE)
                     Output: t3.c1, t3.c2, (lengthb(t3.c1))
                     Distribute Key: (lengthb(t3.c1))
                     Spawn on: All datanodes
                     Consumer Nodes: All datanodes
                     ->  Seq Scan on public.t3
                           Output: t3.c1, t3.c2, lengthb(t3.c1)
                           Distribute Key: t3.c1
(23 rows)
```

c. 函数

```
gaussdb=# EXPLAIN (VERBOSE ON, COSTS OFF) SELECT * FROM t1 JOIN t3 ON t1.c1 =
LENGTHB(t3.c1);
                      QUERY PLAN
------------------------------------------------------------
 Nested Loop
```

 版权所有 © 华为云计算技术有限公司

```
 Output: t1.c1, t1.c2, t3.c1, t3.c2
 ->  Seq Scan on public.t3
       Output: t3.c1, t3.c2
 ->  Partition Iterator
       Output: t1.c1, t1.c2
       Iterations: 3
       ->  Partitioned Index Scan using t1_c1 on public.t1
             Output: t1.c1, t1.c2
             Index Cond: (t1.c1 = lengthb(t3.c1))
             Selected Partitions:  1..3
(11 rows)

--清理示例环境。
gaussdb=# DROP TABLE t1;
gaussdb=# DROP TABLE t2;
gaussdb=# DROP TABLE t3;
```

8.3.2 分区索引

分区表上的索引共有三种类型：

1. Global Non-Partitioned Index
2. Global Partitioned Index
3. Local Partitioned Index

目前GaussDB支持Global Non-Partitioned Index和Local Partitioned Index类型索引。

图 8-3 Global Non-Partitioned Index

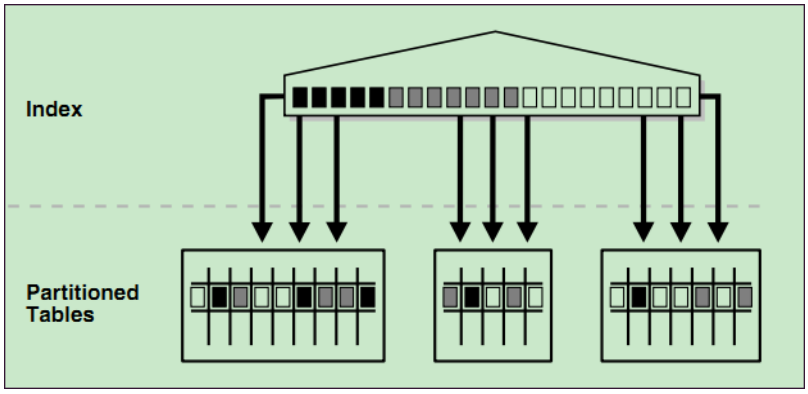
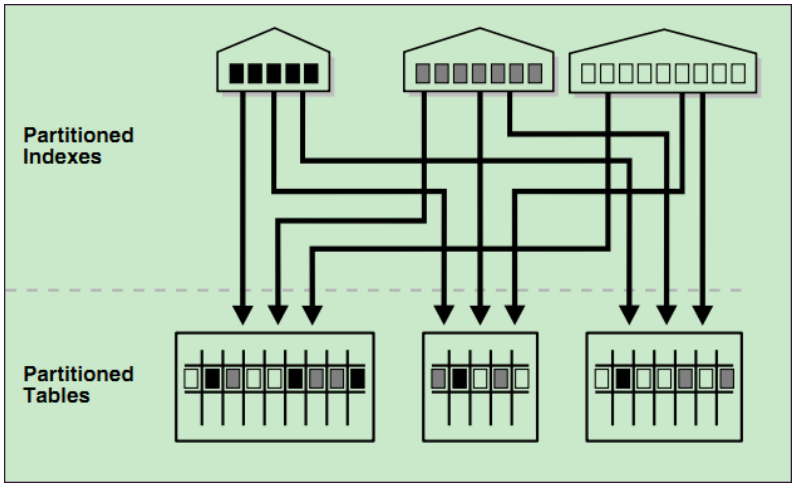
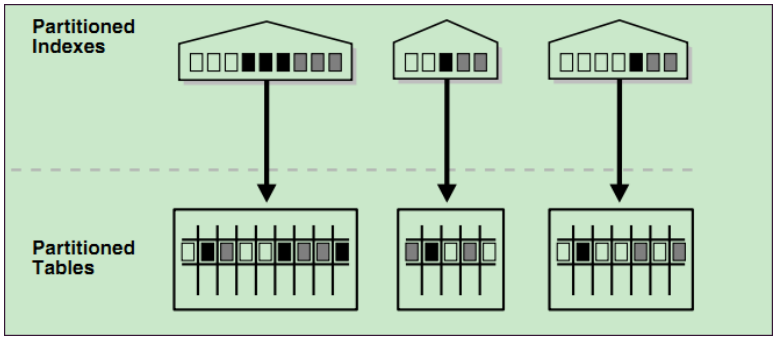


图 8-4 Global Partitioned Index



版权所有 © 华为云计算技术有限公司

图 8-5 Local Partitioned Index



约束

- 分区表索引分为LOCAL索引与GLOBAL索引：LOCAL索引与某个具体分区绑定，而GLOBAL索引则对应整个分区表。
- 唯一约束和主键约束的约束键包含所有分区键则创建LOCAL索引，否则创建GLOBAL索引。
- 在创建LOCAL索引时，可以通过FOR { partition_name | (partition_value [, ...]) }子句，指定在单个分区上创建LOCAL索引，此类索引在其他分区上不生效，后续新增的分区也不会自动创建该索引。需要注意的是，当前仅静态剪枝到单个分区的计划支持生成分类索引的查询路径。

说明

当查询语句在查询数据涉及多个分区时，建议使用GLOBAL索引，反之建议使用LOCAL索引。但需要注意GLOBAL索引在分区维护语法中存在额外的开销。

示例

- 创建表

```
gaussdb=# CREATE TABLE web_returns_p2
(
    ca_address_sk INTEGER NOT NULL ,
    ca_address_id CHARACTER(16) NOT NULL ,
    ca_street_number CHARACTER(10) ,
    ca_street_name CHARACTER VARYING(60) ,
    ca_street_type CHARACTER(15) ,
    ca_suite_number CHARACTER(10) ,
    ca_city CHARACTER VARYING(60) ,
    ca_county CHARACTER VARYING(30) ,
    ca_state CHARACTER(2) ,
    ca_zip CHARACTER(10) ,
    ca_country CHARACTER VARYING(20) ,
    ca_gmt_offset NUMERIC(5,2) ,
    ca_location_type CHARACTER(20)
)
PARTITION BY RANGE (ca_address_sk)
(
    PARTITION P1 VALUES LESS THAN(5000),
    PARTITION P2 VALUES LESS THAN(10000),
    PARTITION P3 VALUES LESS THAN(15000),
    PARTITION P4 VALUES LESS THAN(20000),
    PARTITION P5 VALUES LESS THAN(25000),
    PARTITION P6 VALUES LESS THAN(30000),
    PARTITION P7 VALUES LESS THAN(40000),
    PARTITION P8 VALUES LESS THAN(MAXVALUE)
)
ENABLE ROW MOVEMENT;
```


版权所有 © 华为云计算技术有限公司

- 创建索引
  - 创建分区表LOCAL索引tpcds_web_returns_p2_index1，不指定索引分区的名称。
    ```
    gaussdb=# CREATE INDEX tpcds_web_returns_p2_index1 ON web_returns_p2 (ca_address_id)
    LOCAL;
    ```
    当结果显示为如下信息，则表示创建成功。
    ```
    CREATE INDEX
    ```
  - 创建分区表LOCAL索引tpcds_web_returns_p2_index2，并指定索引分区的名称。
    ```
    gaussdb=# CREATE TABLESPACE example2 LOCATION '/home/omm/example2';
    gaussdb=# CREATE TABLESPACE example3 LOCATION '/home/omm/example3';
    gaussdb=# CREATE TABLESPACE example4 LOCATION '/home/omm/example4';

    gaussdb=# CREATE INDEX tpcds_web_returns_p2_index2 ON web_returns_p2 (ca_address_sk)
    LOCAL
    (
        PARTITION web_returns_p2_P1_index,
        PARTITION web_returns_p2_P2_index TABLESPACE example3,
        PARTITION web_returns_p2_P3_index TABLESPACE example4,
        PARTITION web_returns_p2_P4_index,
        PARTITION web_returns_p2_P5_index,
        PARTITION web_returns_p2_P6_index,
        PARTITION web_returns_p2_P7_index,
        PARTITION web_returns_p2_P8_index
    ) TABLESPACE example2;
    ```
    当结果显示为如下信息，则表示创建成功。
    ```
    CREATE INDEX
    ```
  - 创建分区表GLOBAL索引tpcds_web_returns_p2_global_index。
    ```
    gaussdb=# CREATE INDEX tpcds_web_returns_p2_global_index ON web_returns_p2
    (ca_street_number) GLOBAL;
    ```
    当结果显示为如下信息，则表示创建成功。
    ```
    CREATE INDEX
    ```
  - 创建分类分区索引。

    指定分区名：
    ```
    gaussdb=# CREATE INDEX tpcds_web_returns_for_p1 ON web_returns_p2 (ca_address_id)
    LOCAL(partition ind_part for p1);
    ```
    指定分区键的值：
    ```
    gaussdb=# CREATE INDEX tpcds_web_returns_for_p2 ON web_returns_p2 (ca_address_id)
    LOCAL(partition ind_part for (5000));
    ```
    当结果显示为如下信息，则表示创建成功。
    ```
    CREATE INDEX
    ```
- 修改索引分区的表空间
  - 修改索引分区web_returns_p2_P2_index的表空间为example1。
    ```
    gaussdb=# ALTER INDEX tpcds_web_returns_p2_index2 MOVE PARTITION
    web_returns_p2_P2_index TABLESPACE example1;
    ```
    当结果显示为如下信息，则表示修改成功。
    ```
    ALTER INDEX
    ```
  - 修改索引分区web_returns_p2_P3_index的表空间为example2。
    ```
    gaussdb=# ALTER INDEX tpcds_web_returns_p2_index2 MOVE PARTITION
    web_returns_p2_P3_index TABLESPACE example2;
    ```
    当结果显示为如下信息，则表示修改成功。
    ```
    ALTER INDEX
    ```
- 重命名索引分区
  - 执行如下命令对索引分区web_returns_p2_P8_index重命名web_returns_p2_P8_index_new。

 版权所有 © 华为云计算技术有限公司

```
gaussdb=# ALTER INDEX tpcds_web_returns_p2_index2 RENAME PARTITION
web_returns_p2_P8_index TO web_returns_p2_P8_index_new;
```

当结果显示为如下信息，则表示重命名成功。

```
ALTER INDEX
```

- 查询索引
  - 执行如下命令查询系统和用户定义的所有索引。

    ```
    gaussdb=# SELECT RELNAME FROM PG_CLASS WHERE RELKIND='i' or RELKIND='I';
    ```

  - 执行如下命令查询指定索引的信息。

    ```
    gaussdb=# \di+ tpcds_web_returns_p2_index2
    ```

- 删除索引

  ```
  gaussdb=# DROP INDEX tpcds_web_returns_p2_index1;
  ```

  当结果显示为如下信息，则表示删除成功。

  ```
  DROP INDEX
  ```

- 清理示例

  ```
  gaussdb=# DROP TABLE web_returns_p2;
  ```

8.3.3 分区表统计信息

对于分区表，支持收集分区级统计信息，相关统计信息可以在pg_partition和pg_statistic系统表以及pg_stats和pg_ext_stats视图中查询。分区级统计信息适用于分区表进行静态剪枝后，分区表的扫描范围剪枝到单分区的场景下。分区级统计信息的支持范围为：分区级的page数和tuple数、单列统计信息、多列统计信息、表达式索引统计信息。

分区表统计信息有以下收集方式：

- 级联收集统计信息
- 指定具体单个分区收集统计信息

8.3.3.1 级联收集统计信息

在ANALYZE | ANALYSE分区表时，系统会根据用户指定的或默认的PARTITION_MODE，自动收集分区表中所有符合语义的分区级统计信息，PARTITION_MODE的相关信息请参见《开发指南》中“SQL参考 > SQL语法 > ANALYZE | ANALYSE”中的PARTITION_MODE参数。

⚠ 注意

- 当级联收集复制表、hashbucket表类型的分区表的统计信息，且PARTITION_MODE为ALL时，其行为将转换为ALL COMPLETE模式。
- 分区级统计信息级联收集不支持default_statistics_target为负数的场景。

示例

- 创建分区表并插入数据

  ```
  gaussdb=# CREATE TABLE t1_range_int
  (
     c1 INT,
     c2 INT,
     c3 INT,
     c4 INT
  )
  PARTITION BY RANGE(c1)
  ```


版权所有 © 华为云计算技术有限公司

```
(
    PARTITION range_p00 VALUES LESS THAN(10),
    PARTITION range_p01 VALUES LESS THAN(20),
    PARTITION range_p02 VALUES LESS THAN(30),
    PARTITION range_p03 VALUES LESS THAN(40),
    PARTITION range_p04 VALUES LESS THAN(50)
);
gaussdb=# INSERT INTO t1_range_int SELECT v,v,v,v FROM generate_series(0, 49) AS v;
```

- 级联收集统计信息

```
gaussdb=# ANALYZE t1_range_int WITH ALL;
```

- 查看分区级统计信息

```
gaussdb=# SELECT relname, parttype, relpages, reltuples FROM pg_partition WHERE
parentid=(SELECT oid FROM pg_class WHERE relname='t1_range_int') ORDER BY relname;
   relname    | parttype | relpages | reltuples
--------------+----------+----------+-----------
 range_p00    | p        |        4 |         9
 range_p01    | p        |        7 |        17
 range_p02    | p        |        6 |        13
 range_p03    | p        |        2 |         5
 range_p04    | p        |        4 |         9
 t1_range_int | r        |        0 |         0
(6 rows)

gaussdb=# SELECT
schemaname,tablename,partitionname,subpartitionname,attname,inherited,null_frac,avg_width,n_disti
nct,n_dndistinct,most_common_vals,most_common_freqs,histogram_bounds FROM pg_stats WHERE
tablename='t1_range_int' ORDER BY tablename, partitionname, attname;
 schemaname |   tablename    | partitionname | subpartitionname | attname | inherited | null_frac |
avg_width | n_distinct | n_dndistinct | most_common_vals | most_common_freqs
|                                                histogram_bounds

------------+--------------+---------------+------------------+---------+-----------+-----------+-----------
+------------+--------------+------------------+-------------------
+-----------------------------------------------------------------------------------------------
----------------------------------------------------
 public     | t1_range_int | range_p00     |                  | c1      | f         |         0 |         4 |        -1 |
-1 |              |                  | {0,1,2,3,4,5,6,7,8,9}
 public     | t1_range_int | range_p00     |                  | c2      | f         |         0 |         4 |        -1 |
-1 |              |                  | {0,1,2,3,4,5,6,7,8,9}
 public     | t1_range_int | range_p00     |                  | c3      | f         |         0 |         4 |        -1 |
-1 |              |                  | {0,1,2,3,4,5,6,7,8,9}
 public     | t1_range_int | range_p00     |                  | c4      | f         |         0 |         4 |        -1 |
-1 |              |                  | {0,1,2,3,4,5,6,7,8,9}
 public     | t1_range_int | range_p01     |                  | c1      | f         |         0 |         4 |        -1 |
-1 |              |                  | {10,11,12,13,14,15,16,17,18,19}
 public     | t1_range_int | range_p01     |                  | c2      | f         |         0 |         4 |        -1 |
-1 |              |                  | {10,11,12,13,14,15,16,17,18,19}
 public     | t1_range_int | range_p01     |                  | c3      | f         |         0 |         4 |        -1 |
-1 |              |                  | {10,11,12,13,14,15,16,17,18,19}
 public     | t1_range_int | range_p01     |                  | c4      | f         |         0 |         4 |        -1 |
-1 |              |                  | {10,11,12,13,14,15,16,17,18,19}
 public     | t1_range_int | range_p02     |                  | c1      | f         |         0 |         4 |        -1 |
-1 |              |                  | {20,21,22,23,24,25,26,27,28,29}
 public     | t1_range_int | range_p02     |                  | c2      | f         |         0 |         4 |        -1 |
-1 |              |                  | {20,21,22,23,24,25,26,27,28,29}
 public     | t1_range_int | range_p02     |                  | c3      | f         |         0 |         4 |        -1 |
-1 |              |                  | {20,21,22,23,24,25,26,27,28,29}
 public     | t1_range_int | range_p02     |                  | c4      | f         |         0 |         4 |        -1 |
-1 |              |                  | {20,21,22,23,24,25,26,27,28,29}
 public     | t1_range_int | range_p03     |                  | c1      | f         |         0 |         4 |        -1 |
-1 |              |                  | {30,31,32,33,34,35,36,37,38,39}
 public     | t1_range_int | range_p03     |                  | c2      | f         |         0 |         4 |        -1 |
-1 |              |                  | {30,31,32,33,34,35,36,37,38,39}
 public     | t1_range_int | range_p03     |                  | c3      | f         |         0 |         4 |        -1 |
-1 |              |                  | {30,31,32,33,34,35,36,37,38,39}
 public     | t1_range_int | range_p03     |                  | c4      | f         |         0 |         4 |        -1 |
-1 |              |                  | {30,31,32,33,34,35,36,37,38,39}
 public     | t1_range_int | range_p04     |                  | c1      | f         |         0 |         4 |        -1 |
```

 版权所有 © 华为云计算技术有限公司

```
-1 |               |                | {40,41,42,43,44,45,46,47,48,49}
 public    | t1_range_int | range_p04    |               | c2     | f        |       0 |       4 |       -1 |
-1 |               |                | {40,41,42,43,44,45,46,47,48,49}
 public    | t1_range_int | range_p04    |               | c3     | f        |       0 |       4 |       -1 |
-1 |               |                | {40,41,42,43,44,45,46,47,48,49}
 public    | t1_range_int | range_p04    |               | c4     | f        |       0 |       4 |       -1 |
-1 |               |                | {40,41,42,43,44,45,46,47,48,49}
 public    | t1_range_int |              |               | c1     | f        |       0 |       4 |       -1 |         -1
|               |              |
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49}
 public    | t1_range_int |              |               | c2     | f        |       0 |       4 |       -1 |         -1
|               |              |
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49}
 public    | t1_range_int |              |               | c3     | f        |       0 |       4 |       -1 |         -1
|               |              |
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49}
 public    | t1_range_int |              |               | c4     | f        |       0 |       4 |       -1 |         -1
|               |              |
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49}
(24 rows)
```

- 生成多列数据的分区级统计信息

```
gaussdb=# ALTER TABLE t1_range_int ADD STATISTICS ((c2, c3));
gaussdb=# ANALYZE t1_range_int WITH ALL;
```

- 查看多列数据的分区级统计信息

```
gaussdb=# SELECT
schemaname,tablename,partitionname,subpartitionname,attname,inherited,null_frac,avg_width,n_disti
nct,n_dndistinct,most_common_vals,most_common_freqs,histogram_bounds FROM pg_ext_stats
WHERE tablename='t1_range_int' ORDER BY tablename,partitionname,attname;
 schemaname |   tablename    | partitionname | subpartitionname | attname | inherited | null_frac |
avg_width | n_distinct | n_dndistinct | most_common_vals | most_common_freqs | histogram_bounds
------------+--------------+---------------+------------------+---------+-----------+-----------+-----------
+------------+--------------+------------------+------------------+------------------
 public    | t1_range_int | range_p00    |               | 2 3    | f        |       0 |       8 |       -1 |
-1 |               |                |
 public    | t1_range_int | range_p01    |               | 2 3    | f        |       0 |       8 |       -1 |
-1 |               |                |
 public    | t1_range_int | range_p02    |               | 2 3    | f        |       0 |       8 |       -1 |
-1 |               |                |
 public    | t1_range_int | range_p03    |               | 2 3    | f        |       0 |       8 |       -1 |
-1 |               |                |
 public    | t1_range_int | range_p04    |               | 2 3    | f        |       0 |       8 |       -1 |
-1 |               |                |
 public    | t1_range_int |              |               | 2 3    | f        |       0 |       8 |       -1 |         -1
|               |              |
(6 rows)
```

- 创建表达式索引并生成对应的分区级统计信息

```
gaussdb=# CREATE INDEX t1_range_int_index ON t1_range_int(text(c1)) LOCAL;
gaussdb=# ANALYZE t1_range_int WITH ALL;
```

- 查看表达式索引的分区级统计信息

```
gaussdb=# SELECT
schemaname,tablename,partitionname,subpartitionname,attname,inherited,null_frac,avg_width,n_disti
nct,n_dndistinct,most_common_vals,most_common_freqs,histogram_bounds FROM pg_stats WHERE
tablename='t1_range_int_index' ORDER BY tablename,partitionname,attname;
 schemaname |      tablename      |   partitionname    | subpartitionname | attname | inherited |
null_frac | avg_width | n_distinct | n_dndistinct | most_common_vals | most_common_freqs
|                                                  histogram_bounds

------------+--------------------+--------------------+------------------+---------+-----------+-----------
+-----------+------------+--------------+------------------+------------------
+-------------------------------------------------------------------------------
---------------------------------------------------------------
 public    | t1_range_int_index | range_p00_text_idx |               | text    | f        |       0 |       5
|        -1 |           0 |                |                  | {0,1,2,3,4,5,6,7,8,9}
```

 版权所有 © 华为云计算技术有限公司

```
 public     | t1_range_int_index | range_p01_text_idx |                  | text    | f         |        0 |        6
|         -1 |           0 |                  |                  | {10,11,12,13,14,15,16,17,18,19}
 public     | t1_range_int_index | range_p02_text_idx |                  | text    | f         |        0 |        6
|         -1 |           0 |                  |                  | {20,21,22,23,24,25,26,27,28,29}
 public     | t1_range_int_index | range_p03_text_idx |                  | text    | f         |        0 |        6
|         -1 |           0 |                  |                  | {30,31,32,33,34,35,36,37,38,39}
 public     | t1_range_int_index | range_p04_text_idx |                  | text    | f         |        0 |        6
|         -1 |           0 |                  |                  | {40,41,42,43,44,45,46,47,48,49}
 public     | t1_range_int_index |                  |                  | text    | f         |        0 |        5 |       -1
|           0 |                  |                  |
{0,1,10,11,12,13,14,15,16,17,18,19,2,20,21,22,23,24,25,26,27,28,29,3,30,31,32,33,3
4,35,36,37,38,39,4,40,41,42,43,44,45,46,47,48,49,5,6,7,8,9}
(6 rows)
```

- 删除分区表

```
gaussdb=# DROP TABLE t1_range_int;
```

8.3.3.2 分区级统计信息

指定单分区统计信息收集

当前分区表支持指定单分区统计信息收集，已收集统计信息的分区会在再次收集时自动更新维护。该功能适用于列表分区、哈希分区和范围分区。

```
gaussdb=# CREATE TABLE only_fisrt_part(id int,name varchar)PARTITION BY RANGE (id)
 (PARTITION id11 VALUES LESS THAN (1000000),
PARTITION id22 VALUES LESS THAN (2000000),
PARTITION max_id1 VALUES LESS THAN (MAXVALUE));

gaussdb=# INSERT INTO only_fisrt_part SELECT generate_series(1,5000),'test';


gaussdb=# ANALYZE only_fisrt_part PARTITION (id11);
gaussdb=# ANALYZE only_fisrt_part PARTITION (id22);
gaussdb=# ANALYZE only_fisrt_part PARTITION (max_id1);

gaussdb=# SELECT relname, relpages, reltuples FROM pg_partition WHERE relname IN ('id11', 'id22',
'max_id1');
 relname | relpages | reltuples
---------+----------+-----------
 id11    |     3400 |      5000
 id22    |        0 |         0
 max_id1 |        0 |         0
(3 rows)

gaussdb=# \x
gaussdb=# SELECT * FROM pg_stats WHERE tablename ='only_fisrt_part' AND partitionname ='id11';
-[ RECORD 1 ]----------
+-----------------------
```

```
subpartitionname       |
-[ RECORD 2 ]----------
+-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
schemaname             | public
tablename              | only_fisrt_part
attname                | id
inherited              | f
null_frac              | 0
avg_width              | 4
n_distinct             | -1
n_dndistinct           | 0
most_common_vals       |
most_common_freqs      |
histogram_bounds       |
{1,50,100,150,200,250,300,350,400,450,500,550,600,650,700,750,800,850,900,950,1000,1050,1100,1150,1200,
1250,1300,1350,1400,1450,1500,1550,1600,1650,1700,1750,1800,1850,1900,1950,2000,2050,2100,2150,2200,
2250,2300,2350,2400,2450,2500,2550,2600,2650,2700,2750,2800,2850,2900,2950,3000,3050,3100,3150,3200,
3250,3300,3350,3400,3450,3500,3550,3600,3650,3700,3750,3800,3850,3900,3950,4000,4050,4100,4150,4200,
4250,4300,4350,4400,4450,4500,4550,4600,4650,4700,4750,4800,4850,4900,4950,5000}
correlation            | 1
most_common_elems      |
most_common_elem_freqs |
elem_count_histogram   |
partitionname          | id11
subpartitionname       |
gaussdb=# q \x
-- 删除分区表
gaussdb=# DROP TABLE only_fisrt_part;
```

优化器使用指定分区统计信息

优化器优先使用指定分区的统计信息。如果指定分区未收集统计信息，优化器使用改写分区子句剪枝优化，请参见**通过改写分区子句剪枝优化**。

```
gaussdb=# SET enable_fast_query_shipping = off;
gaussdb=#
CREATE TABLE ONLY_FIRST_PART_TWO
(
    C1 INT,
    C2 BIGINT
)
PARTITION BY RANGE(C1)
(
    PARTITION P_1 VALUES LESS THAN (1000),
    PARTITION P_2 VALUES LESS THAN (3000),
    PARTITION P_3 VALUES LESS THAN (MAXVALUE)
);

gaussdb=# INSERT INTO only_first_part_two SELECT generate_series(1,5000), 0;
gaussdb=# EXPLAIN SELECT * FROM only_first_part_two PARTITION (p_2);
                                QUERY PLAN
---------------------------------------------------------------------------
 Streaming (type: GATHER)  (cost=0.88..2.89 rows=30 width=12)
   Node/s: All datanodes
   ->  Partition Iterator  (cost=0.00..1.14 rows=30 width=12)
         Iterations: 1
         ->  Partitioned Seq Scan on only_first_part_two  (cost=0.00..1.14 rows=30 width=12)
               Selected Partitions:  2
(6 rows)


gaussdb=# EXPLAIN SELECT * FROM only_first_part_two PARTITION (p_1) where c2 = 2;
                                QUERY PLAN
---------------------------------------------------------------------------
Streaming (type: GATHER)  (cost=0.06..1.30 rows=1 width=12)
   Node/s: All datanodes
```

 版权所有 © 华为云计算技术有限公司

```
  ->  Partition Iterator  (cost=0.00..1.18 rows=1 width=12)
        Iterations: 1
        ->  Partitioned Seq Scan on only_first_part_two  (cost=0.00..1.18 rows=1 width=12)
              Filter: (c2 = 0)
              Selected Partitions:  1
(7 rows)

gaussdb=# DROP TABLE only_fisrt_part_two;
```

通过改写分区子句剪枝优化

当没有分区级统计信息时，在优化器行数估算模块，通过在逻辑上对分区子句进行伪谓词的改写，利用改写后的伪谓词影响选择率的计算和整表的统计信息获取一个比较准确的行数估算值。

说明

- 特性只作用于选择率的计算。
- 特性不支持二级分区。
- 特性只支持范围分区（range partition）、列表分区（list partition）。
- 对于范围分区，只支持单列分区键的改写，不支持多列分区键的改写。
- 对于列表分区，出于性能考虑，设置列表指定分区的枚举值个数的阈值为40个。
  - 当指定分区的列表枚举值个数超过40时，本特性不再适用。
  - 对于default分区，其列表枚举值个数是所有非default分区的枚举值个数的总和。

示例1：对于范围分区的改写

```
gaussdb=# CREATE TABLE test_int4_maxvalue(id INT, name VARCHAR)
PARTITION BY RANGE(id)
(
    PARTITION id1 VALUES LESS THAN(1000),
    PARTITION id2 VALUES LESS THAN(2000),
    PARTITION max_id VALUES LESS THAN(MAXVALUE)
);
gaussdb=# INSERT INTO test_int4_maxvalue SELECT GENERATE_SERIES(1,5000),'test';
gaussdb=# ANALYZE test_int4_maxvalue with global;

-- 查询指定分区id1
gaussdb=# EXPLAIN SELECT * FROM test_int4_maxvalue PARTITION(id1);
                                 QUERY PLAN
-----------------------------------------------------------------------------------------
 Data Node Scan  (cost=0.00..0.00 rows=0 width=0)
   Node/s: All datanodes

 Remote SQL: SELECT id, name FROM public.test_int4_maxvalue PARTITION (id1)
 Datanode Name: d1_datanode1
   Partition Iterator  (cost=0.00..7.91 rows=491 width=9)
     Iterations: 1
     ->  Partitioned Seq Scan on test_int4_maxvalue  (cost=0.00..7.91 rows=491 width=9)
           Selected Partitions:  1

 Datanode Name: d1_datanode2
   Partition Iterator  (cost=0.00..8.08 rows=508 width=9)
     Iterations: 1
     ->  Partitioned Seq Scan on test_int4_maxvalue  (cost=0.00..8.08 rows=508 width=9)
           Selected Partitions:  1

(16 rows)

-- 查询指定分区max_id
gaussdb=# EXPLAIN SELECT * FROM test_int4_maxvalue PARTITION(max_id);
                                 QUERY PLAN
-----------------------------------------------------------------------------------------
 Data Node Scan  (cost=0.00..0.00 rows=0 width=0)
```

 版权所有 © 华为云计算技术有限公司

```
  Node/s: All datanodes

 Remote SQL: SELECT id, name FROM public.test_int4_maxvalue PARTITION (max_id)
 Datanode Name: d1_datanode1
   Partition Iterator  (cost=0.00..24.46 rows=1546 width=9)
     Iterations: 1
     ->  Partitioned Seq Scan on test_int4_maxvalue  (cost=0.00..24.46 rows=1546 width=9)
           Selected Partitions:  3

 Datanode Name: d1_datanode2
   Partition Iterator  (cost=0.00..23.55 rows=1455 width=9)
     Iterations: 1
     ->  Partitioned Seq Scan on test_int4_maxvalue  (cost=0.00..23.55 rows=1455 width=9)
           Selected Partitions:  3

(16 rows)

-- 删除分区表
gaussdb=# DROP TABLE test_int4_maxvalue;
```

示例2：对于列表分区的改写

```
gaussdb=# CREATE TABLE test_default
(
    c1 INT,
    c2 BIGINT
)
PARTITION BY LIST(c2)
(
    PARTITION p_1 VALUES (10000, 20000),
    PARTITION p_2 VALUES (300000, 400000, 500000),
    PARTITION p_3 VALUES (DEFAULT)
);
gaussdb=# INSERT INTO test_default SELECT GENERATE_SERIES(1, 1000), 10000;
gaussdb=# INSERT INTO test_default SELECT GENERATE_SERIES(1001, 2000), 600000;
gaussdb=# ANALYZE test_default with global;

-- 查询指定分区p_1
gaussdb=# EXPLAIN SELECT * FROM test_default PARTITION(p_1);
                                QUERY PLAN
-------------------------------------------------------------------------------
 Data Node Scan  (cost=0.00..0.00 rows=0 width=0)
   Node/s: All datanodes

 Remote SQL: SELECT c1, c2 FROM public.test_default PARTITION (p_1)
 Datanode Name: d1_datanode1
   Partition Iterator  (cost=0.00..7.92 rows=492 width=12)
     Iterations: 1
     ->  Partitioned Seq Scan on test_default  (cost=0.00..7.92 rows=492 width=12)
           Selected Partitions:  1

 Datanode Name: d1_datanode2
   Partition Iterator  (cost=0.00..8.08 rows=508 width=12)
     Iterations: 1
     ->  Partitioned Seq Scan on test_default  (cost=0.00..8.08 rows=508 width=12)
           Selected Partitions:  1

(16 rows)

-- 查询指定分区p_3
gaussdb=# EXPLAIN SELECT * FROM test_default PARTITION(p_3);
                                QUERY PLAN
-------------------------------------------------------------------------------
 Data Node Scan  (cost=0.00..0.00 rows=0 width=0)
   Node/s: All datanodes

 Remote SQL: SELECT c1, c2 FROM public.test_default PARTITION (p_3)
 Datanode Name: d1_datanode1
   Partition Iterator  (cost=0.00..8.24 rows=524 width=12)
     Iterations: 1
     ->  Partitioned Seq Scan on test_default  (cost=0.00..8.24 rows=524 width=12)
```

 版权所有 © 华为云计算技术有限公司

```
         Selected Partitions:  3

 Datanode Name: d1_datanode2
   Partition Iterator  (cost=0.00..7.76 rows=476 width=12)
     Iterations: 1
     ->  Partitioned Seq Scan on test_default  (cost=0.00..7.76 rows=476 width=12)
           Selected Partitions:  3

(16 rows)

-- 删除分区表
gaussdb=# DROP TABLE test_default;
```

8.3.4 Partition-wise Join

Partition-wise Join是一种分区级并行的优化技术，是指在符合一定条件的情况下，将两张表之间的Join，分解为两张表中对应的两个分区之间的Join。通过并发执行、减少数据通信量等方式，提升分区表的Join查询的性能。

Partition-wise Join分为SMP场景和非SMP场景。

8.3.4.1 非 SMP 场景下的 Partition-wise Join

在非SMP场景下，Partition-wise Join的路径是基于规则生成的，即只要符合条件，即可生成Partition-wise Join路径，而无需对比路径代价。其开关为GUC参数enable_partitionwise。

使用规格

非SMP场景下的Partition-wise Join的使用规格：

- 只支持一级RANGE分区。
- 支持Hash Join、Nestloop Join、Merge Join。
- 只支持Inner Join。
- 需要设置query_dop的值为1。
- 由于非SMP场景下的Partition-wise Join为规则选择，所以Partition-wise Join计划可能造成性能下降，需要用户自行决定是否启用。
- 仅支持FQS计划。

示例

```
--创建Range分区表。
gaussdb=# CREATE TABLE range_part (
gaussdb(#     a INTEGER,
gaussdb(#     b INTEGER,
gaussdb(#     c INTEGER
gaussdb(# ) PARTITION BY RANGE (a)
gaussdb-# (
gaussdb(# PARTITION range_part_p1 VALUES LESS THAN (10),
gaussdb(# PARTITION range_part_p2 VALUES LESS THAN (20),
gaussdb(# PARTITION range_part_p3 VALUES LESS THAN (30),
gaussdb(# PARTITION range_part_p4 VALUES LESS THAN (40)
gaussdb(# );

--使用FQS计划。
gaussdb=# SET enable_fast_query_shipping= ON;
SET

--设置query_dop为1，关闭SMP。
gaussdb=# SET query_dop = 1;
```

 版权所有 © 华为云计算技术有限公司

```
SET

--关闭非SMP场景下的Partition-wise Join开关。
gaussdb=# SET enable_partitionwise = off;
SET

--查看非Partition-wise Join执行计划。
gaussdb=# SET max_datanode_for_plan = 1;
SET
gaussdb=# EXPLAIN (COSTS OFF) SELECT * FROM range_part t1 INNER JOIN range_part t2 ON (t1.a = t2.a);
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Data Node Scan
   Node/s: All datanodes

 Remote SQL: SELECT t1.a, t1.b, t1.c, t2.a, t2.b, t2.c FROM public.range_part t1 JOIN public.range_part t2 ON
t1.a = t2.a
 Datanode Name: datanode1
   Hash Join
     Hash Cond: (t1.a = t2.a)
     ->  Partition Iterator
           Iterations: 4
           ->  Partitioned Seq Scan on range_part t1
                 Selected Partitions:  1..4
     ->  Hash
           ->  Partition Iterator
                 Iterations: 4
                 ->  Partitioned Seq Scan on range_part t2
                       Selected Partitions:  1..4

(17 rows)

--打开非SMP场景下的Partition-wise Join开关。
gaussdb=# SET enable_partitionwise = on;
SET

--查看非SMP场景下的Partition-wise Join计划。从执行计划中可以看到，Partition Iterator算子被提到了Hash
Join算子的上层。计算方式由原来的依次扫描完所有分区的数据之后再进行Join，改为了每次扫描一对分区，进行
Join，再依次遍历下一个分区。
gaussdb=# EXPLAIN (COSTS OFF) SELECT * FROM range_part t1 INNER JOIN range_part t2 ON (t1.a = t2.a);
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Data Node Scan
   Node/s: All datanodes

 Remote SQL: SELECT t1.a, t1.b, t1.c, t2.a, t2.b, t2.c FROM public.range_part t1 JOIN public.range_part t2 ON
t1.a = t2.a
 Datanode Name: datanode1
   Result
     ->  Partition Iterator
           Iterations: 4
           ->  Hash Join
                 Hash Cond: (t1.a = t2.a)
                 ->  Partitioned Seq Scan on range_part t1
                       Selected Partitions:  1..4
                 ->  Hash
                       ->  Partitioned Seq Scan on range_part t2
                             Selected Partitions:  1..4

(16 rows)

-- 删除分区表
gaussdb=# DROP TABLE range_part;
```

 版权所有 © 华为云计算技术有限公司

8.3.4.2 SMP 场景下的 Full Partition-wise Join

SMP场景下的Partition-wise Join计划是基于代价选择的，在路径生成的过程中，会对比Partition-wise Join和非Partition-wise Join路径的估算代价，选择代价较低的路径。其开关参数为GUC参数enable_smp_partitionwise。

Full Partition-wise Join是指相互Join的两张表为分区策略完全相同的两张分区表，Full Partition-wise Join路径生成条件是两张表的分区键是一对相互匹配的Join key。

使用规格

SMP场景下的Full Partition-wise Join的使用规格：

- 支持一级HASH分区表和一级RANGE分区表。
- Hash分区表的分区策略完全相同是指分区键类型相同、分区数相同。
- Range分区表的分区策略完全相同是指分区键类型相同、分区数相同、分区键数量相同、每个分区的边界值相同。
- 仅支持Stream计划。
- 仅支持分区键和分布键完全一致的场景。
- 仅支持Join算子在单DN内完成计算，即Join算子的数据不跨节点。
- 支持Hash Join和Merge Join。
- 支持Seqscan、Indexscan、Indexonlyscan、Imcvscan。其中，对于Indexscan和Indexonlyscan，只支持分区Local索引，且索引类型为BTREE或UBTREE。
- 相关规格继承SMP规格，不支持SMP场景下的IUD操作。
- 需要开启SMP功能，且设置query_dop的值大于1。

示例

```
--创建Hash分区表。
gaussdb=# CREATE TABLE hash_part
(
    a INTEGER,
    b INTEGER,
    c INTEGER
)
DISTRIBUTE BY HASH(a)
PARTITION BY HASH(a)
(
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4,
    PARTITION p5
);
CREATE TABLE

--使用Stream计划。
gaussdb=# SET enable_fast_query_shipping = off;
SET
gaussdb=# SET enable_stream_operator = on;
SET

--设置query_dop为5，开启SMP。
gaussdb=# SET query_dop = 5;
SET

--关闭SMP场景下的Partition-wise Join开关。
gaussdb=# SET enable_smp_partitionwise = off;
```

 版权所有 © 华为云计算技术有限公司

```
SET

--查看非Partition-wise Join的计划。从计划中可以看出来，在通过Partition Iterator+Partitioned Seq Scan两层算子完成数据扫描之后，通过Streaming(type: LOCAL REDISTRIBUTE)算子对数据进行了一次重分布，用于保证Join算子中数据能够相互匹配。
gaussdb=# EXPLAIN (COSTS OFF) SELECT * FROM hash_part t1, hash_part t2 WHERE t1.a = t2.a;
                            QUERY PLAN
------------------------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Streaming(type: LOCAL GATHER dop: 1/5)
         Spawn on: All datanodes
         ->  Nested Loop
               Join Filter: (t1.a = t2.a)
               ->  Streaming(type: LOCAL REDISTRIBUTE dop: 5/5)
                     Spawn on: All datanodes
                     ->  Partition Iterator
                           Iterations: 5
                           ->  Partitioned Seq Scan on hash_part t1
                                 Selected Partitions:  1..5
               ->  Materialize
                     ->  Streaming(type: LOCAL REDISTRIBUTE dop: 5/5)
                           Spawn on: All datanodes
                           ->  Partition Iterator
                                 Iterations: 5
                                 ->  Partitioned Seq Scan on hash_part t2
                                       Selected Partitions:  1..5
(19 rows)

--打开SMP场景下的Partition-wise Join开关。
gaussdb=# SET enable_smp_partitionwise = on;
SET

--查看Partition-wise Join的执行计划。从计划中可以看出，Partition-wise Join计划消除掉了Streaming算子，即数据不再需要在线程之间重新分布，减少了数据搬运的开销，提升了Join操作的性能。
gaussdb=# EXPLAIN (COSTS OFF) SELECT * FROM hash_part t1, hash_part t2 WHERE t1.a = t2.a;
                          QUERY PLAN
------------------------------------------------------------------------
 Streaming (type: GATHER)
   Node/s: All datanodes
   ->  Streaming(type: LOCAL GATHER dop: 1/5)
         Spawn on: All datanodes
         ->  Hash Join (Partition-wise Join)
               Hash Cond: (t1.a = t2.a)
               ->  Partition Iterator
                     Iterations: 5
                     ->  Partitioned Seq Scan on hash_part t1
                           Selected Partitions:  1..5
               ->  Hash
                     ->  Partition Iterator
                           Iterations: 5
                           ->  Partitioned Seq Scan on hash_part t2
                                 Selected Partitions:  1..5
(15 rows)

-- 删除分区表
gaussdb=# DROP TABLE hash_part;
```

说明

仅在SMP场景下的Partition-wise Join计划中，Join算子右侧会有(Partition-wise Join)提示信息。非SMP场景无该提示信息。

 版权所有 © 华为云计算技术有限公司

8.4 分区表运维管理

分区表运维管理包括分区管理、分区表管理、分区索引管理和分区表业务并发支持等。

- 分区管理：也称分区级DDL，包括新增（Add）、删除（Drop）、交换（Exchange）、清空（Truncate）、分割（Split）、合并（Merge）、移动（Move）、重命名（Rename）共8种。

⚠ 注意

- 对于哈希分区，涉及分区数的变更会导致数据re-shuffling，故当前GaussDB不支持导致Hash分区数变更的操作，包括新增（Add）、删除（Drop）、分割（Split）、合并（Merge）这4种。
- 涉及分区数据变更的操作会使得Global索引失效，可以通过UPDATE GLOBAL INDEX子句来同步更新Global索引，包括删除（Drop）、交换（Exchange）、清空（Truncate）、分割（Split）、合并（Merge）这5种。

说明

- 大部分分区DDL支持partition和partition for指定分区两种写法，前者需要指定分区名，后者需要指定分区定义范围内的任一分区值。比如假设分区part1的范围定义为[100, 200），那么partition part1和partition for(150)这两种写法是等价的。
- 不同分区DDL的执行代价各不相同，由于在执行分区DDL过程中目标分区会被锁住，用户需要评估其代价以及对业务的影响。一般而言，分割（Split）、合并（Merge）的执行代价远大于其他分区DDL，与源分区的大小正相关；交换（Exchange）的代价主要源于Global索引的重建和validation校验；移动（Move）的代价限制于磁盘I/O；其余分区DDL的执行代价都很低。

- 分区表管理：除了继承普通表的功能外，还支持开启/关闭分区表行迁移的功能。
- 分区索引管理：支持用户设置索引/索引分区不可用，或者重建不可用的索引/索引分区，比如由于分区管理操作导致的Global索引失效场景。
- 分区表业务并发支持：分布式分区表的DDL操作会锁全表，不支持跨分区DDL-DQL/DML并发。

8.4.1 新增分区

用户可以在已建立的分区表中新增分区，来维护新业务的进行。当前各种分区表支持的分区上限为1048575，如果达到了上限则不能继续添加分区。同时需要考虑分区占用内存的开销，分区表使用内存大致为（分区数 * 3 / 1024）MB，分区占用内存不允许大于local_syscache_threshold的值，同时还需要预留部分空间以供其他功能使用。

⚠ 注意

- 新增分区不能作用于HASH分区上。
- 新增分区不继承表上的分类索引属性。

版权所有 © 华为云计算技术有限公司

8.4.1.1 向范围分区表新增分区

使用ALTER TABLE ADD PARTITION可以将分区添加到现有分区表的最后面，新增分区的上界值必须大于当前最后一个分区的上界值。

例如，对范围分区表range_sales新增一个分区。

```
ALTER TABLE range_sales ADD PARTITION date_202005 VALUES LESS THAN ('2020-06-01') TABLESPACE tb1;
```

须知

当范围分区表有MAXVALUE分区时，无法新增分区。可以使用ALTER TABLE SPLIT PARTITION命令分割分区。分割分区同样适用于需要在现有分区表的前面/中间添加分区的情形，请参见**对范围分区表分割分区**。

8.4.1.2 向列表分区表新增分区

使用ALTER TABLE ADD PARTITION可以在列表分区表中新增分区，新增分区的枚举值不能与已有的任一个分区的枚举值重复。

例如，对列表分区表list_sales新增一个分区。

```
ALTER TABLE list_sales ADD PARTITION channel5 VALUES ('X') TABLESPACE tb1;
```

须知

当列表分区表有DEFAULT分区时，无法新增分区。可以使用ALTER TABLE SPLIT PARTITION命令分割分区。

8.4.2 删除分区

用户可以使用删除分区的命令来移除不需要的分区。删除分区可以通过指定分区名或者分区值来进行。

⚠ 注意

- 删除分区不能作用于HASH分区上。
- 执行删除分区命令会使得Global索引失效，可以通过UPDATE GLOBAL INDEX子句来同步更新Global索引，或者用户自行重建Global索引。
- 删除分区时，如果该分区上带有仅属于当前分区的分类索引时，则会级联删除分类索引。

使用ALTER TABLE DROP PARTITION可以删除指定分区表的任何一个分区，这个行为可以作用在范围分区表、列表分区表上。

例如，通过指定分区名删除范围分区表range_sales的分区date_202005，并更新Global索引。

```
ALTER TABLE range_sales DROP PARTITION date_202005 UPDATE GLOBAL INDEX;
```

或者，通过指定分区值来删除范围分区表range_sales中'2020-05-08'所对应的分区。由于不带UPDATE GLOBAL INDEX子句，执行该命令后Global索引会失效。

```
ALTER TABLE range_sales DROP PARTITION FOR ('2020-05-08');
```


版权所有 © 华为云计算技术有限公司

须知

- 当分区表只有一个分区时，不支持通过ALTER TABLE DROP PARTITION命令删除分区。
- 当分区表为哈希分区表时，不支持通过ALTER TABLE DROP PARTITION命令删除分区。

8.4.3 交换分区

用户可以使用交换分区的命令来将分区与普通表的数据进行交换。交换分区可以快速将数据导入/导出分区表，实现数据高效加载的目的。在业务迁移的场景，使用交换分区比常规导入会快很多。交换分区可以通过指定分区名或者分区值来进行。

须知

- 执行交换分区命令会使得Global索引失效，可以通过UPDATE GLOBAL INDEX子句来同步更新Global索引，或者用户自行重建Global索引。
- 执行交换分区时，可以申明WITH/WITHOUT VALIDATION，表明是否校验普通表数据满足目标分区的分区键约束规则（默认校验）。数据校验活动开销较大，如果能确保交换的数据属于目标分区，可以申明WITHOUT VALIDATION来提高交换性能。
- 可以申明WITH VALIDATION VERBOSE，此时数据库会校验普通表的每一行，将不满足目标分区的分区键约束规则的数据，插入到分区表的其他分区中，最后再进行普通表与目标分区的交换。

例如，给出如下分区定义和普通表exchange_sales的数据分布，并将分区DATE_202001和普通表exchange_sales做交换，则根据申明子句的不同，存在以下三种行为：

- 申明WITHOUT VALIDATION，数据全部交换到分区DATE_202001中，由于'2020-02-03', '2020-04-08'不满足分区DATE_202001的范围约束，后续业务可能会出现异常。
- 申明WITH VALIDATION，由于'2020-02-03', '2020-04-08'不满足分区DATE_202001的范围约束，数据库给出相应的报错。
- 申明WITH VALIDATION VERBOSE，数据库会将'2020-02-03'插入分区DATE_202002，将'2020-04-08'插入分区DATE_202004，再将剩下的数据交换到分区DATE_202001中。

```
--分区定义
PARTITION DATE_202001 VALUES LESS THAN ('2020-02-01'),
PARTITION DATE_202002 VALUES LESS THAN ('2020-03-01'),
PARTITION DATE_202003 VALUES LESS THAN ('2020-04-01'),
PARTITION DATE_202004 VALUES LESS THAN ('2020-05-01')
-- exchange_sales的数据分布
('2020-01-15', '2020-01-17', '2020-01-23', '2020-02-03', '2020-04-08')
```

⚠ 警告

如果交换的数据不完全属于目标分区，请不要申明WITHOUT VALIDATION交换分区，否则会破坏分区约束规则，导致分区表后续DML业务结果异常。

版权所有 © 华为云计算技术有限公司

进行交换的普通表和分区必须满足如下条件：

- 普通表和分区的列数目相同，对应列的信息严格一致。
- 普通表和分区的表压缩信息严格一致。
- 普通表索引和分区Local索引个数相同，且对应索引的信息严格一致。
- 普通表和分区的表约束个数相同，且对应表约束的信息严格一致。
- 普通表不可以是临时表。
- 普通表和分区表上不可以有动态数据脱敏，行访问控制约束。

使用ALTER TABLE EXCHANGE PARTITION可以对分区表交换分区。

例如，通过指定分区名将范围分区表range_sales的分区date_202001和普通表exchange_sales进行交换，不进行分区键校验，并更新Global索引。

```
ALTER TABLE range_sales EXCHANGE PARTITION (date_202001) WITH TABLE exchange_sales WITHOUT VALIDATION UPDATE GLOBAL INDEX;
```

或者，通过指定分区值将范围分区表range_sales中'2020-01-08'所对应的分区和普通表exchange_sales进行交换，进行分区校验并将不满足目标分区约束的数据插入到分区表的其他分区中。由于不带UPDATE GLOBAL INDEX子句，执行该命令后Global索引会失效。

```
ALTER TABLE range_sales EXCHANGE PARTITION FOR ('2020-01-08') WITH TABLE exchange_sales WITH VALIDATION VERBOSE;
```

8.4.4 清空分区

用户可以使用清空分区的命令来快速清空分区的数据。与删除分区功能类似，区别在于清空分区只会删除分区中的数据，分区的定义和物理文件都会保留。清空分区可以通过指定分区名或者分区值来进行。

⚠ 注意

执行清空分区命令会使得Global索引失效，可以通过UPDATE GLOBAL INDEX子句来同步更新Global索引，或者用户自行重建Global索引。

使用ALTER TABLE TRUNCATE PARTITION可以清空指定分区表的任何一个分区。

例如，通过指定分区名清空范围分区表range_sales的分区date_202005，并更新Global索引。

```
ALTER TABLE range_sales TRUNCATE PARTITION date_202005 UPDATE GLOBAL INDEX;
```

或者，通过指定分区值来清空范围分区表range_sales中'2020-05-08'所对应的分区。由于不带UPDATE GLOBAL INDEX子句，执行该命令后Global索引会失效。

```
ALTER TABLE range_sales TRUNCATE PARTITION FOR ('2020-05-08');
```

8.4.5 分割分区

用户可以使用分割分区的命令来将一个分区分割为两个或多个新分区。当分区数据太大，或者需要对有MAXVALUE的范围分区/DEFAULT的列表分区新增分区时，可以考虑执行该操作。分割分区可以指定分割点将一个分区分割为两个新分区，也可以不指定分割点将一个分区分割为多个新分区。分割分区可以通过指定分区名或者分区值来进行。

版权所有 © 华为云计算技术有限公司

⚠ 注意

- 分割分区不能作用于哈希分区上。
- 执行分割分区命令会使得Global索引失效，可以通过UPDATE GLOBAL INDEX子句来同步更新Global索引，或者用户自行重建Global索引。
- 分割的目标分区如果包含分类索引时，该分区不支持分割。
- 分割后的新分区，可以与源分区名字相同，比如将分区p1分割为p1,p2。但数据库不会将分割前后相同名的分区视为同一个分区。

8.4.5.1 对范围分区表分割分区

使用ALTER TABLE SPLIT PARTITION可以对范围分区表分割分区。

例如，假设范围分区表range_sales的分区date_202001定义范围为['2020-01-01', '2020-02-01')。可以指定分割点'2020-01-16'将分区date_202001分割为两个分区，并更新Global索引。

```
ALTER TABLE range_sales SPLIT PARTITION date_202001 AT ('2020-01-16') INTO
(
    PARTITION date_202001_p1, --第一个分区上界是'2020-01-16'
    PARTITION date_202001_p2  --第二个分区上界是'2020-02-01'
) UPDATE GLOBAL INDEX;
```

或者，不指定分割点，将分区date_202001分割为多个分区，并更新Global索引。

```
ALTER TABLE range_sales SPLIT PARTITION date_202001 INTO
(
    PARTITION date_202001_p1 VALUES LESS THAN ('2020-01-11'),
    PARTITION date_202001_p2 VALUES LESS THAN ('2020-01-21'),
    PARTITION date_202001_p3 --第三个分区上界是'2020-02-01'
)UPDATE GLOBAL INDEX;
```

又或者，通过指定分区值而不是指定分区名来分割分区。

```
ALTER TABLE range_sales SPLIT PARTITION FOR ('2020-01-15') AT ('2020-01-16') INTO
(
    PARTITION date_202001_p1, --第一个分区上界是'2020-01-16'
    PARTITION date_202001_p2  --第二个分区上界是'2020-02-01'
) UPDATE GLOBAL INDEX;
```

须知

若对MAXVALUE分区进行分割，前面几个分区不能申明MAXVALUE范围，最后一个分区会继承MAXVALUE分区范围。

8.4.5.2 对列表分区表分割分区

使用ALTER TABLE SPLIT PARTITION可以对列表分区表分割分区。

例如，假设列表分区表list_sales的分区channel2定义范围为('6', '7', '8', '9')。可以指定分割点('6', '7') 将分区channel2分割为两个分区，并更新Global索引。

```
ALTER TABLE list_sales SPLIT PARTITION channel2 VALUES ('6', '7') INTO
(
    PARTITION channel2_1, --第一个分区范围是('6', '7' )
    PARTITION channel2_2  --第二个分区范围是('8', '9' )
) UPDATE GLOBAL INDEX;
```

或者，不指定分割点，将分区channel2分割为多个分区，并更新Global索引。

版权所有 © 华为云计算技术有限公司

```
ALTER TABLE list_sales SPLIT PARTITION channel2 INTO
(
   PARTITION channel2_1 VALUES ('6'),
   PARTITION channel2_2 VALUES ('8'),
   PARTITION channel2_3 --第三个分区范围是('7', '9' )
)UPDATE GLOBAL INDEX;
```

又或者，通过指定分区值而不是指定分区名来分割分区。

```
ALTER TABLE list_sales SPLIT PARTITION FOR ('6') VALUES ('6', '7') INTO
(
   PARTITION channel2_1, --第一个分区范围是('6', '7' )
   PARTITION channel2_2  --第二个分区范围是('8', '9' )
) UPDATE GLOBAL INDEX;
```

⚠ 注意

若对DEFAULT分区进行分割，前面几个分区不能申明DEFAULT范围，最后一个分区会继承DEFAULT分区范围。

8.4.6 合并分区

用户可以使用合并分区的命令来将多个分区合并为一个分区。合并分区只能通过指定分区名来进行，不支持指定分区值的写法。

⚠ 注意

- 合并分区不能作用于哈希分区上。
- 执行合并分区命令会使得Global索引失效，可以通过UPDATE GLOBAL INDEX子句来同步更新Global索引，或者用户自行重建Global索引。
- 合并前的分区如果包含分类索引则不支持合并。

须知

合并后的新分区，对于范围分区，可以与最后一个源分区名字相同，比如将p1,p2合并为p2；对于列表分区，可以与任一源分区名字相同，比如将p1,p2合并为p1。

如果新分区与源分区名字相同，数据库会将新分区视为对源分区的继承。

使用ALTER TABLE MERGE PARTITIONS可以将多个分区合并为一个分区。

例如，将范围分区表range_sales的分区date_202001和date_202002合并为一个新的分区，并更新Global索引。

```
ALTER TABLE range_sales MERGE PARTITIONS date_202001, date_202002 INTO
   PARTITION date_2020_old UPDATE GLOBAL INDEX;
```

8.4.7 移动分区

用户可以使用移动分区的命令来将一个分区移动到新的表空间中。移动分区可以通过指定分区名或者分区值来进行。

使用ALTER TABLE MOVE PARTITION可以对分区表移动分区。

版权所有 © 华为云计算技术有限公司

例如，通过指定分区名将范围分区表range_sales的分区date_202001移动到表空间tb1中。

```
ALTER TABLE range_sales MOVE PARTITION date_202001 TABLESPACE tb1;
```

或者，通过指定分区值将列表分区表list_sales中'0'所对应的分区移动到表空间tb1中。

```
ALTER TABLE list_sales MOVE PARTITION FOR ('0') TABLESPACE tb1;
```

8.4.8 重命名分区

用户可以使用重命名分区的命令来将一个分区命名为新的名称。重命名分区可以通过指定分区名或者分区值来进行。

8.4.8.1 对分区表重命名分区

使用ALTER TABLE RENAME PARTITION可以对分区表重命名分区。

例如，通过指定分区名将范围分区表range_sales的分区date_202001重命名。

```
ALTER TABLE range_sales RENAME PARTITION date_202001 TO date_202001_new;
```

或者，通过指定分区值将列表分区表list_sales中'0'所对应的分区重命名。

```
ALTER TABLE list_sales RENAME PARTITION FOR ('0') TO channel_new;
```

8.4.8.2 对 Local 索引重命名索引分区

使用ALTER INDEX RENAME PARTITION可以对Local索引重命名索引分区。具体方法与一级分区表重命名分区相同。

8.4.9 分区表行迁移

用户可以使用ALTER TABLE ENABLE/DISABLE ROW MOVEMENT来开启/关闭分区表行迁移。

开启行迁移时，允许通过更新操作将一个分区中的数据迁移到另一个分区中；关闭行迁移时，如果出现这种更新行为，则业务报错。

须知

如果业务明确不允许对分区键所在列进行更新操作，建议关闭分区表行迁移。

例如，创建列表分区表，并开启分区表行迁移，此时可以跨分区更新分区键所在列；关闭分区表行迁移后，对分区键所在列进行跨分区更新会业务报错。

```
CREATE TABLE list_sales
(
    product_id      INT4 NOT NULL,
    customer_id     INT4 PRIMARY KEY,
    time_id         DATE,
    channel_id      CHAR(1),
    type_id         INT4,
    quantity_sold   NUMERIC(3),
    amount_sold     NUMERIC(10,2)
)
PARTITION BY LIST (channel_id)
(
    PARTITION channel1 VALUES ('0', '1', '2'),
    PARTITION channel2 VALUES ('3', '4', '5'),
    PARTITION channel3 VALUES ('6', '7'),
    PARTITION channel4 VALUES ('8', '9')
) ENABLE ROW MOVEMENT;
```


版权所有 © 华为云计算技术有限公司

```
INSERT INTO list_sales VALUES (153241,65143129,'2021-05-07','0',864134,89,34);
--跨分区更新成功，数据从分区channel1迁移到分区channel2
UPDATE list_sales SET channel_id = '3' WHERE channel_id = '0';
--关闭分区表行迁移
ALTER TABLE list_sales DISABLE ROW MOVEMENT;
--跨分区更新失败，报错fail to update partitioned table "list_sales"
UPDATE list_sales SET channel_id = '0' WHERE channel_id = '3';
--分区内更新依然成功
UPDATE list_sales SET channel_id = '4' WHERE channel_id = '3';
```

8.4.10 分区表索引重建/不可用

用户可以通过命令使得一个分区表索引或者一个索引分区不可用，此时该索引/索引分区不再维护。使用重建索引命令可以重建分区表索引，恢复索引的正常功能。

此外，部分分区级DDL操作也会使得Global索引失效，包括删除drop、交换exchange、清空truncate、分割split、合并merge。如果在DDL操作中带UPDATE GLOBAL INDEX子句，则会同步更新Global索引，否则需要用户自行重建索引。

8.4.10.1 索引重建/不可用

使用ALTER INDEX可以设置索引是否可用。

例如，假设分区表range_ sales上存在索引range_sales_idx，可以通过如下命令设置其不可用。

```
ALTER INDEX range_sales_idx UNUSABLE;
```

可以通过如下命令重建索引range_sales_idx。

```
ALTER INDEX range_sales_idx REBUILD;
```

8.4.10.2 Local 索引分区重建/不可用

- 使用ALTER INDEX PARTITION可以设置Local索引分区是否可用。
- 使用ALTER TABLE MODIFY PARTITION可以设置分区表上指定分区的所有索引分区是否可用。

例如，假设分区表range_sales上存在两张Local索引range_sales_idx1和range_sales_idx2，假设其在分区date_202001上对应的索引分区名分别为range_sales_idx1_part1和range_sales_idx2_part1。

下面给出了维护分区表分区索引的语法：

- 可以通过如下命令设置分区date_202001上的所有索引分区均不可用。
  ```
  ALTER TABLE range_sales MODIFY PARTITION date_202001 UNUSABLE LOCAL INDEXES;
  ```
- 或者通过如下命令单独设置分区date_202001上的索引分区range_sales_idx1_part1不可用。
  ```
  ALTER INDEX range_sales_idx1 MODIFY PARTITION range_sales_idx1_part1 UNUSABLE;
  ```
- 可以通过如下命令重建分区date_202001上的所有索引分区。
  ```
  ALTER TABLE range_sales MODIFY PARTITION date_202001 REBUILD UNUSABLE LOCAL INDEXES;
  ```
- 或者通过如下命令单独重建分区date_202001上的索引分区range_sales_idx1_part1。
  ```
  ALTER INDEX range_sales_idx1 REBUILD PARTITION range_sales_idx1_part1;
  ```

8.5 分区表系统视图&DFX

 版权所有 © 华为云计算技术有限公司

8.5.1 分区表相关系统视图

分区表系统视图根据权限分为3类，具体字段信息请参考《 开发指南 》中“系统表和系统视图 > 系统视图”章节。

1. 所有分区视图：
   - ADM_PART_TABLES：所有分区表信息。
   - ADM_TAB_PARTITIONS：所有分区信息。
   - ADM_PART_INDEXES：所有Local索引信息。
   - ADM_IND_PARTITIONS：所有索引分区信息。
2. 当前用户可访问的视图：
   - DB_PART_TABLES：当前用户可访问的分区表信息。
   - DB_TAB_PARTITIONS：当前用户可访问的分区信息。
   - DB_PART_INDEXES：当前用户可访问的Local索引信息。
   - DB_IND_PARTITIONS：当前用户可访问的索引分区信息。
3. 当前用户拥有的视图：
   - MY_PART_TABLES：当前用户拥有的分区表信息。
   - MY_TAB_PARTITIONS：当前用户拥有的分区信息。
   - MY_PART_INDEXES：当前用户拥有的Local索引信息。
   - MY_IND_PARTITIONS：当前用户拥有的索引分区信息。

8.5.2 分区表相关内置工具函数

前置建表相关信息

- 前置建表：
```
CREATE TABLE test_range_pt (a INT, b INT, c INT)
PARTITION BY RANGE (a)
(
    PARTITION p1 VALUES LESS THAN (2000),
    PARTITION p2 VALUES LESS THAN (3000),
    partition p3 VALUES LESS THAN (4000),
    partition p4 VALUES LESS THAN (5000),
    partition p5 VALUES LESS THAN (MAXVALUE)
)ENABLE ROW MOVEMENT;
```
- 查看分区表OID：
```
SELECT oid FROM pg_class WHERE relname = 'test_range_pt';
oid
-------
49290
(1 row)
```
- 查看分区信息：
```
SELECT oid,relname,parttype,parentid,boundaries FROM pg_partition WHERE parentid = 49290;
oid |    relname    | parttype | parentid | boundaries
-------+---------------+----------+----------+------------
49293 | test_range_pt   | r        |    49290 |
49294 | p1              | p        |    49290 | {2000}
49295 | p2              | p        |    49290 | {3000}
49296 | p3              | p        |    49290 | {4000}
49297 | p4              | p        |    49290 | {5000}
49298 | p5              | p        |    49290 | {NULL}
(6 rows)
```
- 创建索引：

 版权所有 © 华为云计算技术有限公司

```
CREATE INDEX idx_range_a ON test_range_pt(a) LOCAL;
CREATE INDEX
--查看分区索引oid
SELECT oid FROM pg_class WHERE relname = 'idx_range_a';
oid
-------
90250
(1 row)
```

- 查看索引分区信息：

```
SELECT oid,relname,parttype,parentid,boundaries,indextblid FROM pg_partition WHERE parentid =
90250;
oid  | relname  | parttype | parentid | boundaries | indextblid
-------+----------+----------+----------+------------+------------
90255 | p5_a_idx | x        |    90250 |            |      49298
90254 | p4_a_idx | x        |    90250 |            |      49297
90253 | p3_a_idx | x        |    90250 |            |      49296
90252 | p2_a_idx | x        |    90250 |            |      49295
90251 | p1_a_idx | x        |    90250 |            |      49294
(5 rows)
```

工具函数示例

- pg_get_tabledef获取分区表的定义，入参可以为表的OID或者表名。

```
SELECT pg_get_tabledef('test_range_pt');

pg_get_tabledef
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 SET search_path =
public;
                                                +
 CREATE TABLE test_range_pt
(
                                              +
    a
integer,
                                                       +
    b
integer,
                                                       +
    c
integer
                                                       +
 )
                                                          +
 WITH (orientation=row, compression=no, storage_type=USTORE,
segment=off)
           +
 PARTITION BY RANGE
(a)
                                           +

(
                                                            +
    PARTITION p1 VALUES LESS THAN (2000) TABLESPACE
pg_default,
               +
    PARTITION p2 VALUES LESS THAN (3000) TABLESPACE
pg_default,
               +
    PARTITION p3 VALUES LESS THAN (4000) TABLESPACE
pg_default,
               +
    PARTITION p4 VALUES LESS THAN (5000) TABLESPACE
pg_default,
               +
    PARTITION p5 VALUES LESS THAN (MAXVALUE) TABLESPACE
pg_default
```

 版权所有 © 华为云计算技术有限公司

```
            +
)
                                                +
 ENABLE ROW
MOVEMENT;
                                               +
 CREATE INDEX idx_range_a ON test_range_pt USING ubtree (a) LOCAL(PARTITION p1_a_idx,
PARTITION p2_a_idx, PARTITION p3_a_idx, PARTITION p4_a_idx, PARTITION p5_a_idx)  WITH
(storage_type=USTORE) TABLESPACE pg_default;
(1 row)
```

- pg_stat_get_partition_tuples_hot_updated返回给定分区id的分区热更新元组数的统计。

  在分区p1中插入10条数据并更新，统计分区p1的热更新元组数。

```
INSERT INTO test_range_pt VALUES(generate_series(1,10),1,1);
INSERT 0 10
SELECT pg_stat_get_partition_tuples_hot_updated(49294);
pg_stat_get_partition_tuples_hot_updated
------------------------------------------
0
(1 row)
UPDATE test_range_pt SET b = 2;
UPDATE 10
SELECT pg_stat_get_partition_tuples_hot_updated(49294);
pg_stat_get_partition_tuples_hot_updated
------------------------------------------
10
(1 row)
```

- pg_partition_size(oid,oid)指定OID代表的分区使用的磁盘空间。其中，第一个oid为表的OID，第二个oid为分区的OID。

  查看分区p1的磁盘空间。

```
SELECT pg_partition_size(49290, 49294);
pg_partition_size
-------------------
90112
(1 row)
```

- pg_partition_size(text, text)指定名称的分区使用的磁盘空间。其中，第一个text为表名，第二个text为分区名。

  查看分区p1的磁盘空间。

```
SELECT pg_partition_size('test_range_pt', 'p1');
pg_partition_size
-------------------
90112
(1 row)
```

- pg_partition_indexes_size(oid,oid)指定OID代表的分区索引使用的磁盘空间。其中，第一个oid为表的OID，第二个oid为分区的OID。

  查看分区p1的索引分区磁盘空间。

```
SELECT pg_partition_indexes_size(49290, 49294);
pg_partition_indexes_size
---------------------------
204800
(1 row)
```

- pg_partition_indexes_size(text,text)指定名称的分区索引使用的磁盘空间。其中，第一个text为表名，第二个text为分区名。

  查看分区p1的索引分区磁盘空间。

```
SELECT pg_partition_indexes_size('test_range_pt', 'p1');
pg_partition_indexes_size
---------------------------
204800
(1 row)
```

 版权所有 © 华为云计算技术有限公司

- pg_partition_filenode(partition_oid)获取到指定分区表的OID所对应的filenode。
  查看分区p1的filenode。

```
SELECT pg_partition_filenode(49294);
pg_partition_filenode
-----------------------
49294
(1 row)
```

- pg_partition_filepath(partition_oid)指定分区的文件路径名。
  查看分区p1的文件路径。

```
SELECT pg_partition_filepath(49294);
pg_partition_filepath
-----------------------
base/16521/49294
(1 row)
```

 版权所有 © 华为云计算技术有限公司

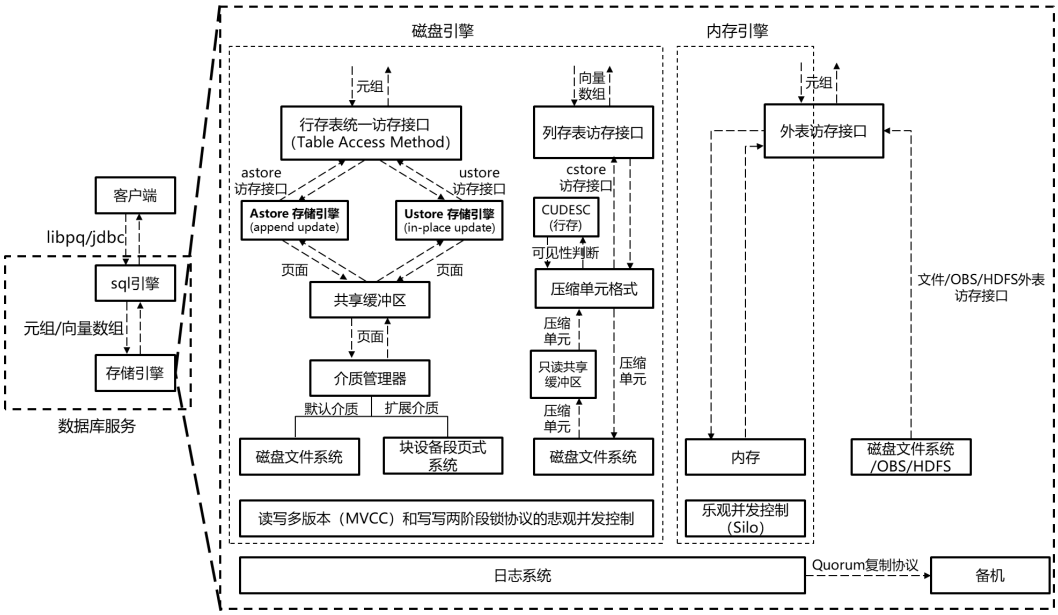
9 存储引擎

9.1 存储引擎体系架构

9.1.1 存储引擎体系架构概述

9.1.1.1 静态编译架构

从整个数据库服务的组成构架来看，存储引擎向上对接SQL引擎，为SQL引擎提供或接收标准化的数据格式（元组或向量数组）；存储引擎向下对接存储介质，按照特定的数据组织方式，以页面、压缩单元（Compress Unit）或其他形式为单位，通过存储介质提供的特定接口，对存储介质中的数据完成读写操作。GaussDB通过静态编译使数据库专业人员可以为特定的应用程序需求选择专用的存储引擎。为了减少对执行引擎的干扰，提供行存访问接口层TableAM，用来屏蔽底层行存引擎带来的差异，使得不同行存引擎可以分别独立演进。如下图所示。



在此基础之上，存储引擎通过日志系统提供数据的持久化和可靠性能力。通过并发控制（事务）系统保证同时执行的、多个读写操作之间的原子性、一致性和隔离性，通

版权所有 © 华为云计算技术有限公司

过索引系统提供对特定数据的加速寻址和查询能力，通过主备复制系统提供整个数据库服务的高可用能力。

行存引擎主要面向OLTP（OnLine Transaction Processing）类业务应用场景，适合高并发、小数据量的单点或小范围数据读写操作。行存引擎向上为SQL引擎提供元组形式的读写接口，向下以页面为单位通过可扩展的介质管理器对存储介质进行读写操作，并通过页面粒度的共享缓冲区来优化读写操作的效率。对于读写并发操作，采用多版本并发控制（MVCC，Multi-Version Concurrency Control）；对于写写并发操作，采用基于两阶段锁协议（2PL，Two-Phase Locking）的悲观并发控制（PCC，Pessimistic Concurrency Control）。当前，行存引擎默认的介质管理器采用磁盘文件系统接口，后续可扩展支持块设备等其他类型的存储介质。GaussDB行存引擎可以选择基于Append update 的Astore或基于In-place update的Ustore。

9.1.1.2 通用数据库服务层

从技术角度来看，存储引擎需要一些基础架构组件，主要包括：

并发：不同存储引擎选择正确的锁可以减少开销，从而提高整体性能。此外提供多版本并发控制或“快照”读取等功能。

事务：均需满足ACID的要求，提供事务状态查询等功能。

内存缓存：不同存储引擎在访问索引和数据时一般会对其进行缓存。缓存池允许直接从内存中处理经常使用的数据，从而加快了处理速度。

检查点：不同存储引擎一般都支持增量checkpoint/double write或全量checkpoint/full page write模式。应用可以根据不同条件进行选择增量或者全量，这个对存储引擎是透明的。

日志：GaussDB采用的是物理日志，其写入/传输/回放对存储引擎透明。

9.1.2 设置存储引擎

存储引擎会对数据库整体效率和性能具有巨大影响，请根据实际需求选择适当的存储引擎。用户可使用WITH ([ORIENTATION | STORAGE_TYPE] [= value] [, ...])为表或索引指定一个可选的存储参数。参数的详细描述如下所示：

ORIENTATION	STORAGE_TYPE
ROW（缺省值）：表的数据将以行式存储。	[USTORE(缺省值)|ASTORE|空]

如果ORIENTATION指定为ROW，且STORAGE_TYPE为空的情况下创建出的表类型取决于GUC参数enable_default_ustore_table（取值为on/off，默认情况为on，参数详情请参见《管理员指南》中“配置运行参数 > GUC参数说明”章节）：如果参数设置为on，创建出的表为Ustore类型；如果为off，创建出的表为Astore类型。

具体示例如下：

```
gaussdb=# CREATE TABLE TEST(a int);
gaussdb=# \d+ test
                    Table "public.test"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 a      | integer |           | plain   |              |
Has OIDs: no
```


版权所有 © 华为云计算技术有限公司

```
Options: orientation=row, compression=no, storage_type=USTORE, segment=off

gaussdb=# CREATE TABLE TEST1(a int) with(orientation=row, storage_type=ustore);
gaussdb=# \d+ test1
Table "public.test1"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 a      | integer |           | plain   |              |
Has OIDs: no
Options: orientation=row, storage_type=ustore, compression=no, segment=off

gaussdb=# CREATE TABLE TEST2(a int) with(orientation=row, storage_type=astore);
gaussdb=# \d+ test2
Table "public.test2"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 a      | integer |           | plain   |              |
Has OIDs: no
Options: orientation=row, storage_type=astore, compression=no
gaussdb=# CREATE TABLE test4(a int) with(orientation=row);
gaussdb=# \d+
                                            List of relations
 Schema | Name  | Type  |   Owner   |  Size   |                          Storage                          | Description
--------+-------+-------+-----------+---------+-----------------------------------------------------------+-------------
 public | test  | table | z7ee88f3a | 0 bytes |
{orientation=row,compression=no,storage_type=USTORE,segment=off} |
 public | test1 | table | z7ee88f3a | 0 bytes |
{orientation=row,storage_type=ustore,compression=no,segment=off} |
 public | test2 | table | z7ee88f3a | 0 bytes | {orientation=row,storage_type=astore,compression=no}            |
 public | test3 | table | z7ee88f3a | 16 kB   |
{orientation=column,storage_type=astore,compression=low}         |
 public | test4 | table | z7ee88f3a | 0 bytes |
{orientation=row,compression=no,storage_type=USTORE,segment=off} |
(5 rows)

gaussdb=# show enable_default_ustore_table;
 enable_default_ustore_table
-----------------------------
 on
(1 row)

gaussdb=#  DROP TABLE test;
gaussdb=#  DROP TABLE test1;
gaussdb=#  DROP TABLE test2;
gaussdb=#  DROP TABLE test4;
```

9.1.3 存储引擎更新说明

9.1.3.1 GaussDB 内核 505 版本

- Ustore支持柔性字段高效存储。
- Ustore支持Toast规模商用。
- Ustore增加页面恢复与逃生技术。
- Ustore支持SMP技术。

9.1.3.2 GaussDB 内核 503 版本

- Ustore适配分布式/并行查询/Global Temp Table/Vacuum full/列约束DEFERRABLE以及INITIALLY DEFERRED。
- Ustore增加在线重建索引。

 版权所有 © 华为云计算技术有限公司

- Ustore增加增强版本B-tree空页面估算，提升优化器代价估算准确度。

- Ustore增加存储引擎可靠性验证框架，Diagnose Page/Page Verify。

- Ustore增强存储引擎相关的解析/检测/修复视图。

- Ustore增强基于WAL日志的定位能力，新增gs_redo_upage系统视图，支持对单页面的不断重放，获取并打印该页面的任何一个历史版本，加速页面损坏类问题的定位。

- Ustore扩展事务槽TD物理格式，为事务内空间复用做好铺垫。

- Ustore增加在线创建索引。

- Ustore适配闪回功能（ for Ustore ）/极致RTO。

9.1.3.3 GaussDB 内核 R2 版本

- Ustore增加新的基于原位更新的行存储引擎Ustore，首次实现新旧版本的记录的分离存储。

- Ustore增加回滚段模块。

- Ustore增加回滚过程，支持同步/异步/页内模式。

- Ustore增加支持事务的增强版本B-tree。

- Astore增加闪回功能，支持闪回表/闪回查询/闪回Drop/闪回Truncate。

- Ustore不支持的特性包括：分布式/并行查询/Table Sampling/Global Temp Table/在线创建/重建索引/极致RTO/Vacuum Full/列约束DEFERRABLE以及INITIALLY DEFERRED。

9.2 Astore 存储引擎

9.2.1 Astore 简介

Astore与Ustore的多版本实现最大的区别在于最新版本和历史版本是否分离存储。Astore不进行分离存储，而Ustore当前也只是分离了数据，索引本身没有分开。

使用 Astore 的优势

1. Astore没有回滚段，而Ustore有回滚段。对于Ustore来说，回滚段是非常重要的，回滚段损坏会导致数据丢失甚至数据库无法启动的严重问题，且Ustore恢复时同步需要Redo和Undo。由于Astore没有回滚段，旧数据都是记录在原先的文件中；所以，当数据库异常crash后恢复时，不会像Ustore数据库那样进行复杂的恢复。
2. 由于旧的数据是直接记录在数据文件中，而不是回滚段中，所以不会经常报Snapshot Too Old错误。
3. 回滚可以很快完成，因为回滚并不删除数据。

> **⚠ 注意**
>
> 回滚时很复杂，在事务回滚时必须清理该事务所进行的修改，插入的记录要删除，更新的记录要更新回来，同时回滚的过程也会再次产生大量的Redo日志。

版权所有 © 华为云计算技术有限公司

4. WAL日志要简单一些，仅需要记录数据文件的变化，不需要记录回滚段的变化。
5. 支持回收站（闪回DROP、闪回Truncate）功能。

9.3 Ustore 存储引擎

9.3.1 Ustore 简介

Ustore（Unified Storage）是GaussDB推出的一款原位更新的存储引擎，其多版本的实现较Astore最大的区别在于最新版本和历史版本的数据是分离存储的，而索引当前还没有分离。

使用 Ustore 的优势

- 最新版本和历史版本分离存储，相比Astore扫描范围小。去除Astore的HOT chain，非索引列/索引列更新，Heap均可原位更新，ROWID可保持不变。历史版本可批量回收，空间膨胀可控。
- B-tree索引增加了事务信息，能够独立进行MVCC，增加了IndexOnlyScan的比例，大大减少回表次数。
- 不依赖Vacuum进行旧版本清理。独立的空间回收能力，索引与堆表解耦，可独立清理，IO平稳度更优。
- 大并发更新同一行的场景，相对于Astore的ROWID会偏移，Ustore的原位更新机制保证了元组ROWID稳定，先到先得，更新时延相对稳定。
- 支持闪回功能。

⚠ 注意

Ustore DML在修改数据页面时，也需要同步生成Undo，因此更新操作开销会稍大一些。此外单条Tuple扫描开销由于需要复制（Astore返回指针）也会大一些。

9.3.1.1 Ustore 特性与规格

9.3.1.1.1 特性约束

类别	特性	是否支持
事务	Serializable	×
	在事务块中对分区表执行DDL操作	×
可扩展性	Hashbucket	×
SQL	Table sampling/物化视图/键值锁	×

9.3.1.1.2 存储规格

1. 数据表最大列数不能超过1600列。

版权所有 © 华为云计算技术有限公司

2. init_td（TD（Transaction Directory，事务目录）是Ustore表独有的用于存储页面事务信息的结构，TD的数量决定该页面支持的最大并发数。在创建表或索引时可以指定初始的TD大小init_td）取值范围[2, 128]，默认值4。单页面支持的最大并发不超过128个。
3. Ustore表（不含toast情况）最大Tuple长度不能超过（8192 - MAXALIGN(56 + init_td * 26 + 4)），其中MAXALIGN表示8字节对齐。当插入数据长度超过阈值时，用户会收到元组长度过长无法插入的报错。其中init_td对于Tuple长度的影响如下：
   - 表init_td数量为最小值2时，Tuple长度不能超过8192 - MAXALIGN(56+2*26+4) = 8080B。
   - 表init_td数量为默认值4时，Tuple长度不能超过8192 - MAXALIGN(56+4*26+4) = 8024B。
   - 表init_td数量为最大值128时，Tuple长度不能超过8192 - MAXALIGN(56+128*26+4) = 4800B。
4. 索引最大列数不能超过32列。全局分区索引最大列数不能超过31列。
5. 索引元组长度不能超过(8192 - MAXALIGN(28 + 3 * 4 + 3 * 10) - MAXALIGN(42))/3, 其中MAXALIGN表示8字节对齐。当插入数据长度超过阈值时，用户会收到索引元组长度过长无法插入的报错，其中索引页头为28B，行指针为4B，元组CTID+INFO标记位为10B，页尾为42B。
6. 回滚段容量最大支持16TB。

9.3.1.2 使用 Ustore 进行测试

创建Ustore表

使用CREATE TABLE语句创建Ustore表。

```
gaussdb=# CREATE TABLE ustore_table(a INT PRIMARY KEY, b CHAR (20)) WITH (STORAGE_TYPE=USTORE);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "ustore_table_pkey" for table
"ustore_table"
CREATE TABLE
gaussdb=# \d+ ustore_table
Table "public.ustore_table"
Column |     Type      | Modifiers | Storage  | Stats target | Description
--------+---------------+-----------+----------+--------------+-------------
a      | integer       | not null  | plain    |              |
b      | character(20) |           | extended |              |
Indexes:
"ustore_table_pkey" PRIMARY KEY, ubtree (a) WITH (storage_type=USTORE) TABLESPACE pg_default
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, storage_type=ustore, compression=no, segment=off
```

删除Ustore表

```
gaussdb=# DROP TABLE ustore_table;
 DROP TABLE
```

为Ustore表创建索引

Ustore当前仅支持BTree类型的多版本索引。在一些场景中，为了区别于Astore的BTree索引，也会将Ustore表的多版本BTree索引称为UBTree（Ustore BTree，UBTree介绍详见**UBTree**章节）。用户可以参照以下方式使用CREATE INDEX语句为Ustore表的“a”属性创建一个UBTree索引。

Ustore表不指定创建索引类型，默认创建的是UBTree索引。

 版权所有 © 华为云计算技术有限公司

⚠ 注意

UBTree索引分为RCR版本和PCR版本，默认创建RCR版本的UBTree。若在创建索引时with选项指定(index_txntype=pcr)或者指定GUC的index_txntype=pcr，则创建的是PCR版本的UBTree。

```
gaussdb=# CREATE TABLE test(a int);
CREATE TABLE
gaussdb=# CREATE INDEX UB_tree_index ON test(a);
CREATE INDEX
gaussdb=# \d+ test
Table "public.test"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 a      | integer |           | plain   |              |
Indexes:
    "ub_tree_index" ubtree (a) WITH (storage_type=USTORE) TABLESPACE pg_default
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off

--删除Ustore表索引。
gaussdb=# DROP TABLE test;
DROP TABLE
```

9.3.1.3 Ustore 的最佳实践

9.3.1.3.1 怎么配置 init_td 大小

TD（Transaction Directory，事务目录）是Ustore表独有的用于存储页面事务信息的结构，TD的数量决定该页面支持的最大并发数。在创建表或索引时可以指定初始的TD大小init_td，默认值为4，即同时支持4个并发事务修改该页面，最大值为128。

用户需要结合业务并发度分析是否需要手动配置init_td。另外也可以结合业务运行过程中“wait available td”等待事件出现的频率来分析是否需要调整，一般“wait available td”等于0。如果“wait available td”一直不为0，就存在等待TD的事件，此时建议增大init_td再进行观察，反复几次，如果大于0的情况属于偶发，不建议调整，多余的TD槽位会占用更多的空间。推荐的增大的方法可以按照倍数进行测试，建议可从小到大尝试8、16、32、48、...、128，并观测对应的等待事件是否有明显减少，尽量取等待事件较少中init_td数量最小的值作为默认值以节省空间。wait available td是wait_status的值之一，wait_status表示当前线程的等待状态，包含等待状态详细信息。通过PG_THREAD_WAIT_STATUS视图可以查询wait_status的值（none表示没在等待任意事件，如果有等待事件即可看到对应wait available td的值），示例如下。init_td的配置和详细描述参见《开发指南》的“SQL参考 > SQL语法 > CREATE TABLE”章节。init_td查看和修改方法的具体示例如下：

```
gaussdb=# CREATE TABLE test1(name varchar) WITH(storage_type = ustore, init_td=2);
gaussdb=# \d+ test1
                        Table "public.test1"
 Column |       Type        | Modifiers | Storage  | Stats target | Description
--------+-------------------+-----------+----------+--------------+-------------
 name   | character varying |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, storage_type=ustore, init_td=2, compression=no, segment=off,
toast.storage_type=ustore, toast.toast_storage_type=enhanced_toast

gaussdb=# ALTER TABLE test1 SET(init_td=8);
```


版权所有 © 华为云计算技术有限公司

```
gaussdb=# \d+ test1
                        Table "public.test1"
 Column |       Type        | Modifiers | Storage  | Stats target | Description
--------+-------------------+-----------+----------+--------------+-------------
 name   | character varying |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, storage_type=ustore, compression=no, segment=off, init_td=8,
toast.storage_type=ustore, toast.toast_storage_type=enhanced_toast

gaussdb=# SELECT * FROM pg_thread_wait_status;
 node_name | db_name  |          thread_name          |    query_id     |       tid       |    sessionid    | lwtid |
psessionid | tlevel | smpid | wait_status | wait_event | locktag | lo
ckmode | block_sessionid | global_sessionid
-----------+----------+-------------------------------+-----------------+-----------------+-----------------+-------
+------------+--------+-------+-------------+------------+---------+---
-------+-----------------+------------------
 sgnode    |          | PageWriter                    |               0 | 139769678919424 | 139769678919424 | 16915
|           |     0 |     0 | none        | none       |         |
      |               | 0:0#0
 sgnode    |          | PageWriter                    |               0 | 139769736066816 | 139769736066816 | 16913
|           |     0 |     0 | none        | none       |         |
      |               | 0:0#0
 sgnode    |          | PageWriter                    |               0 | 139769707755264 | 139769707755264 | 16914
|           |     0 |     0 | none        | none       |         |
      |               | 0:0#0
 sgnode    |          | PageWriter                    |               0 | 139769761756928 | 139769761756928 | 16912
|           |     0 |     0 | none        | none       |         |
      |               | 0:0#0
 sgnode    |          | PageWriter                    |               0 | 139769783772928 | 139769783772928 | 16911
|           |     0 |     0 | none        | none       |         |
      |               | 0:0#0


gaussdb=# DROP TABLE test1;
DROP TABLE
```

9.3.1.3.2 怎么配置 fillfactor 大小

fillfactor是用于描述页面填充率的参数，该参数与页面能存放的元组数量、大小以及表的物理空间直接相关。Ustore表的默认页面填充率为92%，预留的8%空间用于更新的扩展，也可以用于TD列表的扩展空间。fillfactor的配置和详细描述参见《开发指南》的“SQL参考 > SQL语法 > CREATE TABLE”章节。

用户需要结合业务分析是否需要手动配置fillfactor。如果表数据导入后只有查询或定长更新操作，可将页面填充率调整为100%。如果数据导入后存在大量非定长更新操作，建议为不调整页面填充率或者将页面填充率数值调整更小，以减少跨页更新带来的性能损耗。fillfactor查看和修改方法具体示例如下：

```
gaussdb=# CREATE TABLE test(a int) with(fillfactor=100);
gaussdb=# \d+ test
                      Table "public.test"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 a      | integer |           | plain   |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, fillfactor=100, compression=no, storage_type=USTORE, segment=off

gaussdb=# ALTER TABLE test set(fillfactor=92);
gaussdb=# \d+ test
                      Table "public.test"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 a      | integer |           | plain   |              |
```


版权所有 © 华为云计算技术有限公司

```
Has OIDs: no
Options: orientation=row, compression=no, storage_type=USTORE, segment=off， fillfactor=92

gaussdb=# DROP TABLE test;
DROP TABLE
```

9.3.1.3.3 在线校验功能

在线校验是Ustore特有的，在运行过程中可以有效预防页面因编码逻辑错误导致的逻辑损坏，默认开启UPAGE:UBTREE:UNDO三个模块校验。业务现网请保持开启，性能场景除外。

关闭：

```
gs_guc reload -Z coordinator -Z datanode -N all -I all -c "ustore_attr=''"
```

打开：

```
gs_guc reload -Z coordinator -Z datanode -N all -I all -c
"ustore_attr='ustore_verify_level=fast;ustore_verify_module=upage:ubtree:undo'"
```

9.3.1.3.4 怎么配置回滚段大小

一般情况下回滚段大小的参数使用默认值即可。为了达到最佳性能，部分场景下可调整回滚段大小的相关参数。具体场景与设置方法如下：

1. 保留给定时间内的历史版本数据。

   当使用闪回或者支撑问题定位时，通常希望保留更多历史版本数据，此时需要修改undo_retention_time。undo_retention_time默认值是0，取值范围为 0~3天，输入有效单位为s,min,h,d。调整的推荐值为900s，需要注意的是，undo_retention_time的取值越大，对业务的影响除了Undo空间占用增多，也会造成数据空间膨胀，进一步影响数据扫描更新性能。当不使用闪回或者希望减少历史旧版本的磁盘空间占用时，需要将undo_retention_time调小来达到最佳性能。可以通过如下方法选择更适合自己业务模型的取值：

   使用Undo统计信息的系统函数gs_stat_undo，如果入参为false，输出undo_space_limit_size、undo_limit_size_per_transaction 、undo_retention_time 参数的合理化建议。具体详细参数值参见《 开发指南 》的“SQL参考 > 函数和操作符 > undo系统函数”章节。

2. 保留给定空间大小的历史版本数据。

   如果业务中存在长事务或大事务可能导致Undo空间膨胀时，需要将undo_space_limit_size调大，undo_space_limit_size默认值为256GB，取值范围为800MB~16TB。

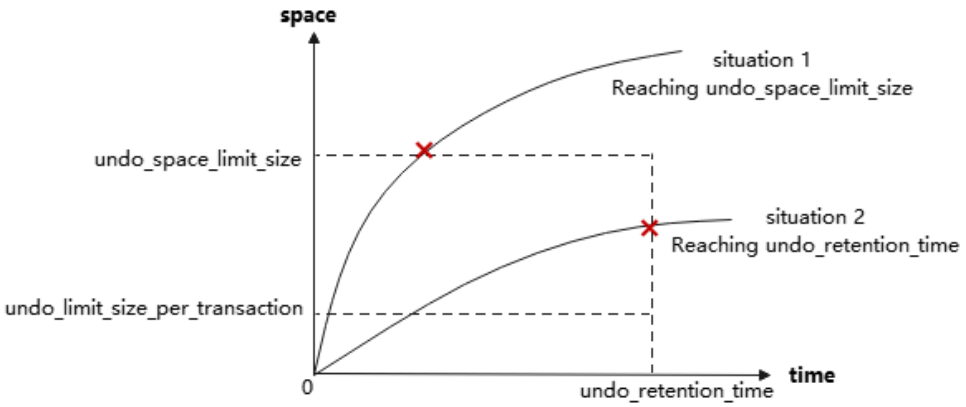
   在磁盘空间允许的条件下，推荐undo_space_limit_size设置翻倍。同时undo_space_limit_size的取值越大则占用磁盘空间越大，可能降低性能。如果查询视图系统函数gs_stat_undo的curr_used_undo_size发现不存在Undo空间膨胀，可以恢复为原值。

   调整undo_space_limit_size后可相应提高单事务平均占用undo空间undo_limit_size_per_transaction的取值，undo_limit_size_per_transaction取值范围为2MB~16TB，默认值为32GB。设置时建议undo_limit_size_per_transaction不超过undo_space_limit_size，即单事务Undo分配空间阈值不大于Undo总空间阈值。

3. 历史版本的保留参数的调整优先级。

   在undo_retention_time、undo_space_limit_size、undo_limit_size_per_transaction中，先触发的空间阈值会先进行约束限制。

 版权所有 © 华为云计算技术有限公司



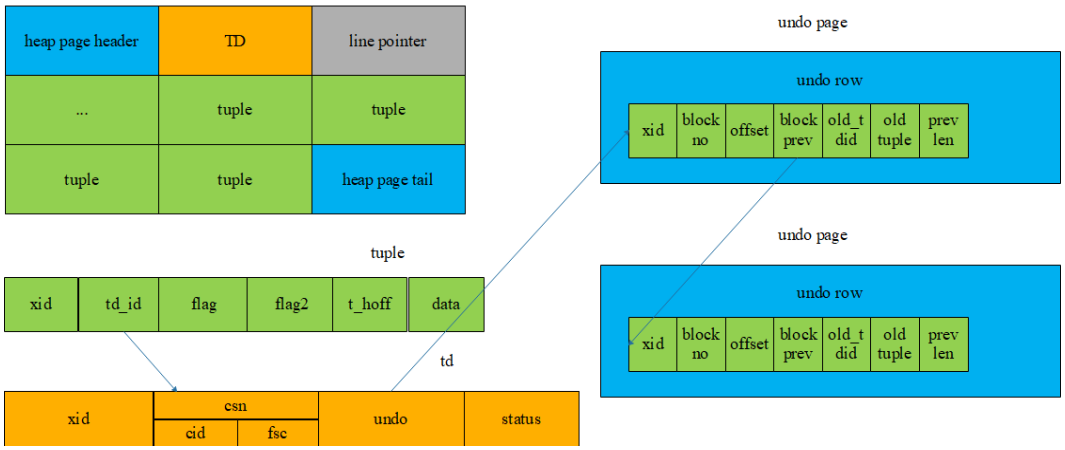
例如：Undo强制回收阈值参数undo_space_limit_size设置为1GB，Undo旧版本保留时间undo_retention_time为900s。如果900s内产生的历史版本数据不足1GB*0.8，则按照900s进行回收限制；否则按照1GB*0.8进行回收限制。遇到该情况时，如果磁盘空闲空间充足，则上调undo_space_limit_size，如果磁盘空闲空间紧缺，则下调undo_retention_time。

9.3.2 存储格式

9.3.2.1 RCR Uheap

9.3.2.1.1 RCR Uheap 多版本管理

Ustore对其使用的heap做了如下重要的增强，简称Uheap。

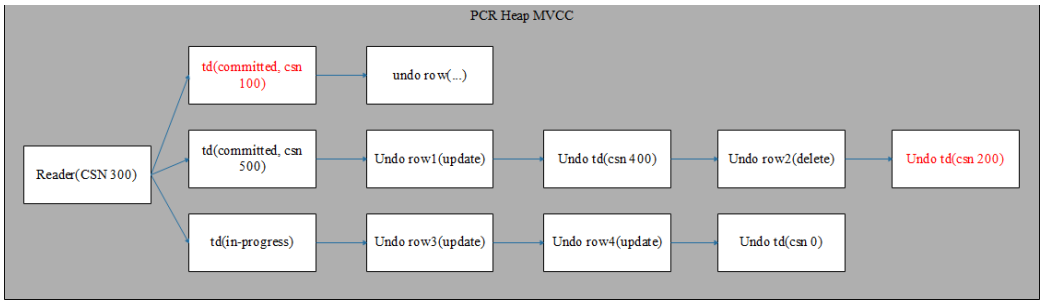


Ustore RCR(Row Consistency Read)的多版本管理是基于数据行的行级多版本管理。不过Ustore将XID记录在了页面的TD(Transaction Directory)区域区别于常见的将XID存储在数据行上，节省了页面空间。事务修改记录时，会将历史数据记录到Undo Row中，在Tuple中的td_id指向的TD槽上记录产生的Undo Row地址(zone_id, block no, page offset)，并将新的数据覆盖写入页面。访问元组时，沿着版本链还原该元组，直到找到自己对应的版本。

9.3.2.1.2 RCR Uheap 可见性机制

Ustore可见性判断是通过构建数据行的一致性版本获得的，老快照可通过Undo记录获取历史版本。

 版权所有 © 华为云计算技术有限公司



9.3.2.1.3 RCR Uheap 空闲空间管理

Ustore使用Free Space Map（FSM）文件记录了每个数据页的潜在空闲空间，并且以树的结构组织起来。每当用户想要对某个表执行插入操作或者是非原位更新操作时，就会从该表对应的FSM中进行快速查找，查看当前FSM上记录的最大空闲空间是否可以满足插入所需的空间要求；如果满足则返回对应的blocknum用于执行插入操作，否则执行拓展页面逻辑。

每一个表或者分区对应的FSM结构存放在一个独立的FSM文件中，该FSM文件与表数据放在相同的目录下。例如，假设表t1对应的数据文件为32181，则其对应的FSM文件为32181_fsm。FSM内部同样是以数据块的格式存储，这里称为FSM block，FSM block之间的逻辑结构组成了一棵有三层节点的树，树的节点在逻辑上是大顶堆关系。每次在FSM上查找时从根节点进行，一直查找到叶子节点，然后在叶子节点内搜索到一个可用的页面并返回给业务用于执行后续操作。

该结构不保证和数据页实际可用空间保持实时一致，会在DML的执行过程中进行维护。Ustore会在Auto Vacuum的过程中概率性对该FSM进行修复重建。当用户执行插入类型的DML语句，类似Insert/Non-Inplace Update(新页面)/Multi Insert时，会查询FSM结构，寻找到一个可以插入当前记录的空间。用户执行完DML操作后会根据当前页面的潜在空闲空间与实际空闲空间的差值来决定是否将该页面的空闲空间刷新到FSM上。该差值越大，即潜在空间大于实际空间越多，则该页面被更新至FSM的几率越大。FSM上会记录数据页的潜在空闲空间 ，在用户执行插入操作找到一个页面时，如果该页面上的空闲空间较大则直接插入，否则如果潜在空间较大则对页面执行清理后插入。最后如果空间不够则重新搜索FSM结构或者拓展总页面数量。更新FSM结构主要有以下几个位置，DML、页面清理、vacuum、拓展页面、分区合并、页面扫描等。

9.3.2.2 UBTree

其使用的btree做了如下重要的增强，简称Ubtree。

- Ubtree索引增加了事务信息，能够独立进行MVCC；增加了IndexOnlyScan的比例，大大减少回表次数。
- 不依赖Vacuum进行旧版本清理。独立的空间回收能力，索引与堆表解耦，可独立清理，IO平稳度更优。

9.3.2.2.1 RCR UBTree

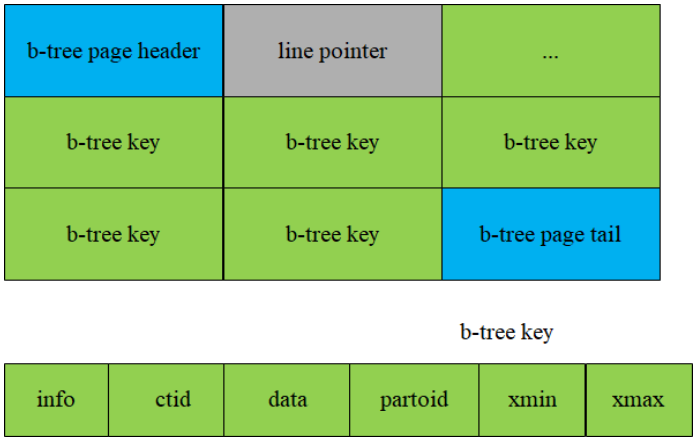
RCR UBTree 多版本管理

RCR(Row Consistency Read) btree 的多版本管理是基于数据行的行级多版本管理。将XID记录在了数据行上，会增加Key的大小，索引会有5-20%左右的膨胀。最新版本和历史版本均在btree上，索引没有记录Undo信息。插入或者删除key时按照key + TID的顺序排列，索引列相同的元组按照对应元组的TID作为第二关键字进行排序，会将

 版权所有 © 华为云计算技术有限公司

xmin、xmax追加到key的后面。索引分裂时，多版本信息随着key的迁移而迁移，如图9-1所示。

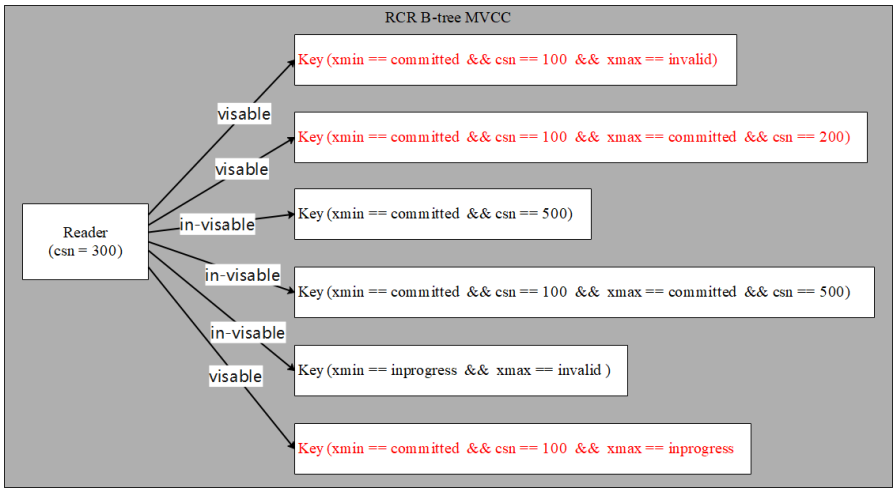
图 9-1 RCR UBTree 多版本管理



RCR UBTree 可见性机制

RCR UBTree可见性判断是通过判断Key上xmin/xmax来确定的，与Astore堆表数据行上xmin/xmax作用类似，如图9-2所示。

图 9-2 RCR UBTree 可见性机制



RCR UBTree 增删改查

- **Insert操作**：Ubtree的插入逻辑基本不变，只需增加索引插入时直接获取事务信息填写xmin字段。
- **Delete操作**：Ubtree额外增加了索引删除流程。索引删除主要步骤与插入相似，获取事务信息填写xmax字段（B-tree索引不维护版本信息，不需要删除操作），同时更新页面上的active_tuple_count。若active_tuple_count被减为0，则尝试页面回收。
- **Update操作**：对于Ustore而言，数据更新对Ubtree索引列的操作也与Astore有所不同。数据更新包含两种情况：索引列和非索引列更新，Ubtree在数据发生更新时的处理如图9-3所示。

 版权所有 © 华为云计算技术有限公司

图 9-3 UBTree 在数据发生更新时的处理

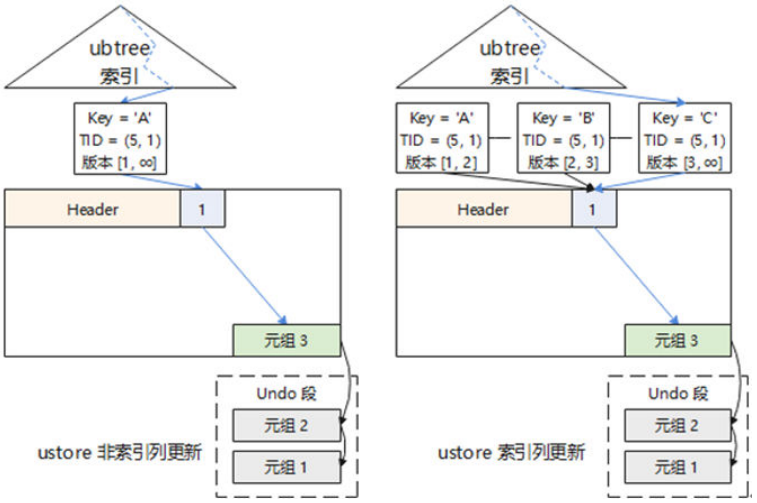


图9-3展示Ubtree在索引列和非索引列更新的差异：

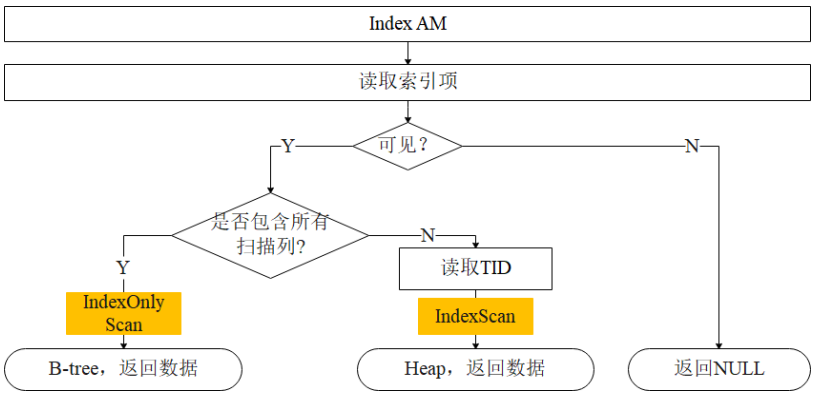
- 在非索引列更新的情况下，索引不发生任何变化。index tuple仍指向第一次插入的data tuple，Uheap不会插入新的data tuple，而是修改当下data tuple并将历史数据存入Undo中。
- 在索引列更新的情况下，Ubtree也会插入新的index tuple，但是会指向同一个data linepointer和同一个data tuple。扫描旧版本的数据则需要从Undo中读取。

- **Scan操作**：用户在读取数据时，可通过使用索引扫描加速，Ubtree支持索引数据的多版本管理及可见性检查，索引层的可见性检查使得索引扫描（Index Scan）及仅索引扫描（IndexOnly Scan）性能有所提升。

  对于索引扫描：

  - 若索引列包含所有扫描列（IndexOnly Scan），则通过扫描条件在索引上进行二分查找，找到符合条件元组即可返回数据。
  - 若索引列不包含所有扫描列（Index Scan），则通过扫描条件在索引上进行二分查找，找到符合条件元组的TID，再通过TID到数据表上查找对应的数据元组。如图4 对应的数据元组所示。

图 9-4 对应的数据元组

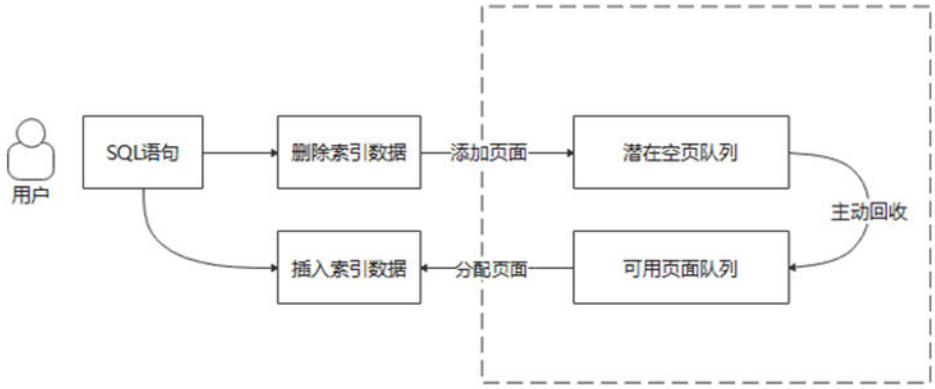


 版权所有 © 华为云计算技术有限公司

RCR UBTree 空间管理

当前Astore的索引依赖AutoVacuum和Free Space Map（FSM）进行空间管理，存在回收不及时的问题，而Ustore的索引使用其特有的URQ（UBTree Recycle Queue，一种基于循环队列的数据结构，即双循环队列）对索引空闲空间进行管理。双循环队列是指有两个循环队列，一个潜在空页队列，另一个可用空页队列。在DML过程中完成索引的空间管理，能有效地缓解DML过程中造成的空间急剧膨胀问题。 索引回收队列单独储存在B-tree索引对应的FSM文件中。

图 9-5 索引页面在双循环队列间流动



如图9-5所示，索引页面在双循环队列间流动如下：

1. **索引空页流动到潜在队列**

   索引页尾字段中记录了页面上活跃元组个数（activeTupleCount）。在DML过程中，删空一个页面的所有元组，即activeTupleCount为零时会将索引页放入潜在队列中。

2. **潜在队列流动到可用队列**

   潜在队列到可用队列的转化主要是达到一个潜在队列收支平衡以及可用队列在拿页时有页可拿的目的。即当从可用队列拿出一个索引空页用完后，建议从潜在队列转化至少一个索引页面到可用队列中，以及每当潜在队列新加入一个索引页面时，能从潜在队列中移除至少一个索引页插入可用队列中，达到潜在队列的收支平衡，以及可用队列有页可用的目的。

3. **可用队列流动到索引空页**

   索引在分裂等获取一个索引空页面时，会先从可用队列中进行查找是否有可以复用的索引页，如果找到则直接进行复用，没有可复用页面则进行物理扩页。

9.3.2.2.2 PCR UBTree

相比于RCR版本的UBTree，PCR版本的UBTree有以下特点。

- 索引元组的事务信息统一由TD槽进行管理。
- 增加了Undo操作，插入和删除前需要先写入Undo，事务abort时需要进行回滚操作。
- 支持闪回。

PCR UBTree通过在创建索引时with选项设置“index_txntype=pcr”或者设置GUC参数“index_txntype=pcr”进行创建。若没有显示指定with选项或者GUC，则默认创建

版权所有 © 华为云计算技术有限公司

RCR版本的UBTree。当前PCR版本的UBTree不支持在线进行创建、极致RTO回放和备机读的功能。

注意，当前版本PCR索引在大数据量的回滚上耗时可能较长（回滚时间随数据量增长可能呈指数型增长，数据量太大可能会导致回滚未完成），回滚时间会在下个版本进行优化。以下是当前版本回滚时间的具体规格：

表 9-1 PCR 索引回滚时间的规格

类型/数据量	100	1000	1万	10万	100万
带PCR索引的回滚时间	0.692 ms	9.610 ms	544.678 ms	52,963.754 ms	89,440,029.048 ms
不带PCR索引的回滚时间	0.226 ms	0.916 ms	8.974 ms	94.903 ms	1206.177 ms
两者比值	3.06	10.49	60.70	558.08	74,151.66

PCR UBTree 多版本管理

与RCR UBTree的区别是，PCR(Page Consistency Read) 的多版本管理是基于页面的多版本管理，所有元组的事务信息统一由TD槽进行管理。

PCR UBTree 可见性机制

PCR UBTree可见性判断是通过把页面回滚到快照可见的时刻得到元组全部可见的页面完成的。

PCR UBTree 增删改查

- **Insert操作**：操作与RCR UBTree基本一致，区别是：插入前需要先申请TD和写入Undo。
- **Delete操作**：操作与RCR UBTree基本一致，区别是：删除前需要先申请TD和写入Undo。
- **Update操作**：操作与RCR UBTree无区别，均转换为一条Delete操作和和一条Insert操作。
- **Scan操作**：操作与RCR UBTree基本一致，区别是：查询操作需要将页面复制一个CR页面出来，将CR页面回滚到扫描快照可见的状态，从而整个页面的元组对于快照都是可见版本。

PCR UBTree 空间管理

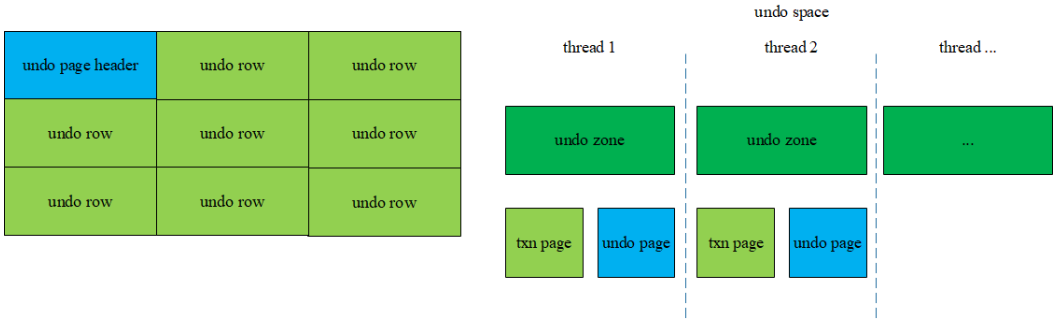
空间管理操作与RCR UBTree基本一致，区别是：PCR UBTree支持闪回，所以页面可回收的时间点由OldestXmin改成了GlobalRecycleXid。

 版权所有 © 华为云计算技术有限公司

9.3.2.3 Undo

历史版本数据集中存放在$node_dir/undo目录中，其中$node_dir为数据库节点路径，回滚段日志是与单个写事务关联的所有撤销日志的集合。支持permanent/unlogged/temp三种表类型。

9.3.2.3.1 回滚段管理



1. 每个undo zone除了管理部分transaction page（用于存储事务回滚的元数据）外，还管理undo page。
2. Undo页面中存储undo row，对数据的修改会将历史版本记录到Undo中。
3. Undo记录也是数据，因此对Undo页面的修改同样会记录Redo。

9.3.2.3.2 文件组织结构

如需查询当前回滚段使用的存储方式是页式或段页式，可以查询系统表。当前仅支持页式。

示例：

```
gaussdb=# SELECT * FROM gs_global_config where name like '%undostoragetype%';
      name       | value
-----------------+---------
 undostoragetype | page
(1 row)
```

- 当回滚段使用的存储方式为页式：
  - txn page所在文件组织结构：
    $node_dir/undo/{permanent|unlogged|temp}/$undo_zone_id.meta.$segno
  - undo row所在文件组织结构：
    $node_dir/undo/{permanent|unlogged|temp}/$undo_zone_id.$segno

9.3.2.3.3 空间管理

Undo子系统依赖后台回收线程进行空闲空间回收。负责主机上Undo模块的空间回收，备机通过回放xLog进行回收。回收线程遍历使用中的undo zone，对该zone中的txn page扫描，依据xid从小到大的顺序进行遍历。回收已提交或者已回滚完成的事务，且该事务的提交时间应早于$(current_time-undo_retention_time)。对于遍历过程中需要回滚的事务，后台回收线程会为该事务添加异步回滚任务。

当数据库中存在运行时间长、修改数据量大的事务，或者开启闪回时间较长的时候，可能出现undo空间持续膨胀的情况。当undo占用空间接近undo_space_limit_size时，就会触发强制回收。只要事务已提交或者已回滚完成，即使事务提交时间晚于$(current_time-undo_retention_time)，在这种情况下也可能被回收掉。

版权所有 © 华为云计算技术有限公司

9.3.2.4 Enhanced Toast

9.3.2.4.1 概述

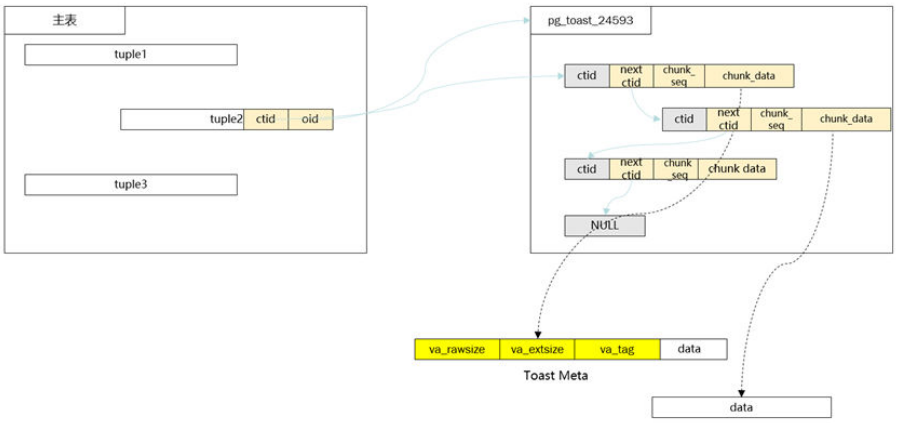
Enhanced Toast是一种用于处理超大字段的技术。首先，减少了Toast Pointer中的冗余信息，存储支持单表超长字段列数超过500列。其次，优化了主表与线外存储表之间的映射关系，无需通过pg_toast_index来存储主表数据与线外存储表数据的关系，降低了用户存储空间。最后，Enhanced Toast技术通过让分割数据自链接，消除了Oid分配的依赖，极大地加快了写入效率。

说明

- Astore存储引擎不支持Enhanced Toast。
- 不支持对Enhanced Toast类型的线外存储表单独进行Vacuum Full操作。

9.3.2.4.2 Enhanced Toast 存储结构

Enhanced Toast技术使用自链接的方式来处理元组间的依赖关系。线外存储表把超长数据按照2K分割成链表块，主表的Toast Pointer指向线外存储表的对应数据链表头。这样极大简化了主表与线外存储表间的映射关系，有效的提升了数据写入与查询的性能。



9.3.2.4.3 Enhanced Toast 使用

新增的GUC参数enable_enhance_toast_table用于控制线外存储结构。
“enable_enhance_toast_table=on”表示使用Enhanced Toast线外存储表，
“enable_enhance_toast_table=off”表示使用Toast线外存储表。

```
gs_guc reload -Z coordinator -Z datanode -N all -I all -c "enable_enhance_toast_table=on"
```

9.3.2.4.4 Enhanced Toast 增删改查

Insert操作：触发Enhanced Toast的写入条件保持与原有Toast一致，除了数据写入时增加了数据间的链接信息之外，插入基本逻辑保持不变。

Delete操作：Enhanced Toast的数据删除流程不再依赖Toast数据索引，仅依靠数据间的链接信息将对应的数据进行遍历删除。

Update操作：Enhanced Toast的更新流程与原有Toast保持一致。

版权所有 © 华为云计算技术有限公司

9.3.2.4.5 Enhanced Toast 相关 DDL 操作

Enhanced Toast表的创建

建表时指定Toast表的存储类型为Enhanced Toast或者Toast：

```
gaussdb=#  CREATE TABLE test_toast (id int, content text) with(toast.toast_storage_type=toast);
CREATE TABLE
gaussdb=#  \d+ test_toast
                    Table "public.test_toast"
 Column  |  Type   | Modifiers | Storage  | Stats target | Description
---------+---------+-----------+----------+--------------+-------------
 id      | integer |           | plain    |              |
 content | text    |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=toast
gaussdb=#  DROP TABLE test_toast;
DROP TABLE
gaussdb=#  CREATE TABLE test_toast (id int, content text) with(toast.toast_storage_type=enhanced_toast);
CREATE TABLE
gaussdb=#  \d+ test_toast
                    Table "public.test_toast"
 Column  |  Type   | Modifiers | Storage  | Stats target | Description
---------+---------+-----------+----------+--------------+-------------
 id      | integer |           | plain    |              |
 content | text    |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=enhanced_toast
gaussdb=#  DROP TABLE test_toast;
DROP TABLE
```

建表时不指定线外存储表的类型，则创建线外存储表类型依赖于GUC参数enable_enhance_toast_table：

```
-- 根据“Enhanced Toast使用”章节打开GUC
gaussdb=#  show enable_enhance_toast_table;
 enable_enhance_toast_table
----------------------------
 on
(1 row)
gaussdb=#  CREATE TABLE test_toast (id int, content text);
CREATE TABLE
gaussdb=#   \d+ test_toast
                    Table "public.test_toast"
 Column  |  Type   | Modifiers | Storage  | Stats target | Description
---------+---------+-----------+----------+--------------+-------------
 id      | integer |           | plain    |              |
 content | text    |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=enhanced_toast

gaussdb=#   DROP TABLE test_toast;
DROP TABLE
gaussdb=#  SET enable_enhance_toast_table = off;
SET
gaussdb=#  show enable_enhance_toast_table;
 enable_enhance_toast_table
----------------------------
 off
(1 row)
```


版权所有 © 华为云计算技术有限公司

```
gaussdb=#   CREATE TABLE test_toast (id int, content text);
CREATE TABLE
gaussdb=#  \d+ test_toast
                    Table "public.test_toast"
 Column  |  Type   | Modifiers | Storage  | Stats target | Description
---------+---------+-----------+----------+--------------+-------------
 id      | integer |           | plain    |              |
 content | text    |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=toast
gaussdb=#  DROP TABLE test_toast;
DROP TABLE
```

线外存储表结构的升级

当GUC参数“enable_enhance_toast_table=on”时，线外存储表支持通过Vacuum Full操作将Toast升级为Enhanced Toast结构。

```
gaussdb=#   CREATE TABLE test_toast (id int, content text);
CREATE TABLE
gaussdb=#  \d+ test_toast
                    Table "public.test_toast"
 Column  |  Type   | Modifiers | Storage  | Stats target | Description
---------+---------+-----------+----------+--------------+-------------
 id      | integer |           | plain    |              |
 content | text    |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=toast
gaussdb=#  VACUUM FULL test_toast;
VACUUM
gaussdb=#  \d+ test_toast
                    Table "public.test_toast"
 Column  |  Type   | Modifiers | Storage  | Stats target | Description
---------+---------+-----------+----------+--------------+-------------
 id      | integer |           | plain    |              |
 content | text    |           | extended |              |
Has OIDs: no
Distribute By: HASH(a)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off,
toast.storage_type=USTORE, toast.toast_storage_type=enhanced_toast
gaussdb=#  DROP TABLE test_toast;
DROP TABLE
```

分区表merge操作

支持将分区表的分区间不同的线外存储表类型进行合并操作。

说明

- 对于相同类型的线外存储分区，合并与原有逻辑保持一致，进行物理合并。
- 对于不同类型的线外存储分区，合并后的分区线外存储表为Enhanced Toast表，需要进行逻辑合并，性能劣于物理合并。

```
gaussdb=#  CREATE TABLE test_partition_table(a int, b text)PARTITION BY range(a)(partition p1 values less
than (2000),partition p2 values less than (3000));
gaussdb=# SELECT relfilenode FROM pg_partition WHERE relname='p1';
 relfilenode
-------------
       17529
(1 row)
```

 版权所有 © 华为云计算技术有限公司

```
gaussdb=#  \d+ pg_toast.pg_toast_part_17529
TOAST table "pg_toast.pg_toast_part_17529"
   Column   |  Type   | Storage
------------+---------+---------
 chunk_id   | oid     | plain
 chunk_seq  | integer | plain
 chunk_data | bytea   | plain
Options: storage_type=ustore, toast_storage_type=toast

gaussdb=# SELECT relfilenode FROM pg_partition WHERE relname='p2';
 relfilenode
-------------
       17528
(1 row)

gaussdb=#  \d+ pg_toast.pg_toast_part_17528
TOAST table "pg_toast.pg_toast_part_17528"
   Column   |  Type   | Storage
------------+---------+---------
 chunk_seq  | integer | plain
 next_chunk | tid     | plain
 chunk_data | bytea   | plain
Options: storage_type=ustore, toast_storage_type=enhanced_toast
gaussdb=#  ALTER TABLE test_partition_table MERGE PARTITIONS p1,p2 INTO partition p1_p2;
ALTER TABLE
gaussdb=#  SELECT reltoastrelid::regclass FROM pg_partition where relname='p1_p2';
         reltoastrelid
-------------------------------
 pg_toast.pg_toast_part_17559
(1 row)
gaussdb=#  \d+ pg_toast.pg_toast_part_17559
TOAST table "pg_toast.pg_toast_part_17559"
   Column   |  Type   | Storage
------------+---------+---------
 chunk_seq  | integer | plain
 next_chunk | tid     | plain
 chunk_data | bytea   | plain
Options: storage_type=ustore, toast_storage_type=enhanced_toast
gaussdb=# DROP TABLE test_partition_table;
DROP TABLE
```

9.3.2.4.6 Enhanced Toast 运维管理

通过gs_parse_page_bypath解析主表中的ToastPointer信息。

```
gaussdb=#  SELECT ctid,next_chunk,chunk_seq FROM pg_toast.pg_toast_part_17559;
 ctid  | next_chunk | chunk_seq
-------+------------+-----------
 (0,1) | (0,0)      |         1
 (0,2) | (0,1)      |         0
 (0,3) | (0,0)      |         1
 (0,4) | (0,3)      |         0
(4 rows)
gaussdb=#  SELECT gs_parse_page_bypath((SELECT * FROM
pg_relation_filepath('test_toast')),0,'uheap',false);
                      gs_parse_page_bypath
-----------------------------------------------------------------
 ${data_dir}/gs_log/dump/1663_13113_17603_0.page
(1 row)
```

解析文件1663_13113_17603_0.page中存储了ToastPointer的相关信息，具体如下：

```
Toast_Pointer:
    column_index: 1
        toast_relation_oid: 17608     --线外存储表OID信息
        ctid: (4, 1)                  --线外存储数据链，头指针
        bucket id: -1                 --bucket id信息
    column_index: 2
```


版权所有 © 华为云计算技术有限公司

```
    toast_relation_oid: 17608
    ctid: (2, 1)
    bucket id: -1
```

Enhanced Toast数据查询，通过直接查询到的Enhanced Toast表数据可以判断其链式结构的完整性。

```
gaussdb=#  SELECT ctid,next_chunk,chunk_seq FROM pg_toast.pg_toast_part_17559;
 ctid  | next_chunk | chunk_seq
-------+------------+-----------
 (0,1) | (0,0)      |         1
 (0,2) | (0,1)      |         0
 (0,3) | (0,0)      |         1
 (0,4) | (0,3)      |         0
(4 rows)
```

9.3.3 Ustore 事务模型

GaussDB事务基础：

1. 事务启动时不会自动分配XID，该事务中的第一条DML/DDL语句运行时才会真正为该事务分配XID。
2. 事务结束时，会产生代表事务提交状态的CLOG（Commit Log），CLOG共有四种状态：事务运行中、事务提交、事务同步回滚、子事务提交。每个事务的 CLOG 状态位为2 bits，CLOG页面上每个字节可以表示四个事务的提交状态。
3. 事务结束时，还会产生代表事务提交顺序的CSN（Commit sequence number）。CSN为实例级变量，每个XID都有自己对应的唯一CSN。CSN可以标记事务的以下状态：事务提交中、事务提交、事务回滚、事务已冻结等。

9.3.3.1 事务提交

针对隐式事务和显式事务，其提交策略如下所示：

1. 隐式事务。单条DML/DDL语句自动触发隐式事务，这种事务没有显式的事务块控制语句（START TRANSACTION/BEGIN/COMMIT/END），DML语句结束后自动提交。
2. 显式事务。显式事务由显式的START TRANSACTION/BEGIN语句控制事务的开始，由COMMIT/END语句控制事务的提交。

   子事务必须存在于显式事务或存储过程中，由SAVEPOINT语句控制子事务开始，由RELEASE SAVEPOINT语句控制子事务结束。如果一个事务在提交时还存在未释放的子事务，该事务提交前会先执行子事务的提交，所有子事务提交完毕后才会进行父事务的提交。

   Ustore支持读已提交隔离级别。语句在执行开始时，获取当前系统的CSN作为当前语句的查询CSN。整个语句的可见结果由语句开始那一刻决定，不受后续其他事务修改影响。Ustore中read committed默认是保持一致性读的。Ustore也支持标准的2PC事务。

9.3.3.2 事务回滚

回滚是在事务运行的过程中发生了故障等异常情形下，事务不能继续执行，系统需要将事务中已完成的修改操作进行撤销。Astore、Ubtree没有回滚段，自然没有这个专门的回滚动作。Ustore为了性能考虑，它的回滚流程结合了同步、异步和页面级回滚等3种形式。

- **同步回滚**

  有三种情况会触发事务的同步回滚：

 版权所有 © 华为云计算技术有限公司

  - 事务块中的ROLLBACK关键字会触发同步回滚。
  - 事务运行过程中如果发生ERROR级别报错，此时的COMMIT关键字与ROLLBACK功能相同，也会触发同步回滚。
  - 事务运行过程中如果发生FATAL/PANIC级别报错，在线程退出前会尝试将该线程绑定的事务进行一次同步回滚。

- **异步回滚**

  同步回滚失败或者在系统宕机后再次重启时，会由Undo回收线程为未回滚完成的事务发起异步回滚任务，立即对外提供服务。由异步回滚任务发起线程undo launch负责拉起异步回滚工作线程undo worker，再由异步回滚工作线程实际执行回滚任务。undo launch线程最多可以同时拉起5个undo worker线程。

- **页面级回滚**

  当事务需要回滚但还未回滚到本页面时，如果其他事务需要复用该事务所占用的TD，就会在复用前对该事务在本页面的所有修改执行页面级回滚。页面级回滚只负责回滚事务在本页面的修改，不涉及其他页面。

  Ustore子事务的回滚由ROLLBACK TO SAVEPOINT语句控制，子事务回滚后父事务可以继续运行，子事务的回滚不影响父事务的事务状态。如果一个事务在回滚时还存在未释放的子事务，该事务回滚前会先执行子事务的回滚，所有子事务回滚完毕后才会进行父事务的回滚。

9.3.4 闪回恢复

闪回恢复功能是数据库恢复技术的一环，可以有选择性地撤销一个已提交事务的影响，将数据从人为不正确的操作中进行恢复。在采用闪回技术之前，只能通过备份恢复、PITR等手段找回已提交的数据库修改，恢复时长需要数分钟甚至数小时。采用闪回技术后，通过闪回Drop和闪回Truncate恢复已提交的数据库Drop/Truncate的数据，只需要秒级，而且恢复时间和数据库大小无关。

说明

- ASTORE引擎只支持闪回DROP/TRUNCATE功能。
- 备机不支持闪回操作。
- 用户可以根据需要开启闪回功能，开启后会带来一定的性能劣化。

9.3.4.1 闪回查询

背景信息

闪回查询可以查询过去某个时间点表的某个snapshot数据，这一特性可用于查看和逻辑重建意外删除或更改的受损数据。闪回查询基于MVCC多版本机制，通过检索查询旧版本，获取指定老版本数据。

前提条件

整体方案分为三部分：旧版本保留、快照的维护和旧版本检索。旧版本保留：新增undo_retention_time配置参数，用来设置旧版本保留的时间，超过该时间的旧版本将被回收清理，若使用闪回查询则需将该参数设置为大于0的值，请联系管理员修改。

语法

```
{[ ONLY ] table_name [ * ] [ partition_clause ] [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
[ TABLESAMPLE sampling_method ( argument [, ...] ) [ REPEATABLE ( seed ) ] ]
```

 版权所有 © 华为云计算技术有限公司

```
[TIMECAPSULE { TIMESTAMP | CSN } expression ]
|( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
|with_query_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
|function_name ( [ argument [, ...] ] ) [ AS ] alias [ ( column_alias [, ...] | column_definition [, ...] ) ]
|function_name ( [ argument [, ...] ] ) AS ( column_definition [, ...] )
|from_item [ NATURAL ] join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]}
```

语法树中“TIMECAPSULE {TIMESTAMP | CSN} expression”为闪回功能新增表达方式，其中TIMECAPSULE表示使用闪回功能，TIMESTAMP以及CSN表示闪回功能使用具体时间点信息或使用CSN（commit sequence number）信息。

参数说明

- TIMESTAMP
  - 指要查询某个表在TIMESTAMP这个时间点上的数据，TIMESTAMP指一个具体的历史时间。
- CSN
  - 指要查询整个数据库逻辑提交序下某个CSN点的数据，CSN指一个具体逻辑提交时间点，数据库中的CSN为写一致性点，每个CSN代表整个数据库的一个一致性点，查询某个CSN下的数据代表SQL查询数据库在该一致性点的相关数据。

备注：使用时间点进行闪回时，可能会有3s的误差。想要闪回到精确的操作点，需要使用CSN进行闪回。GTM-Free模式下没有全局一致性csn点，暂时不支持以csn的方式进行闪回。

使用示例

- 示例（需将undo_retention_time参数设置为大于0的值）：

```
gaussdb=# DROP TABLE IF EXISTS "public".flashtest;
NOTICE:  table "flashtest" does not exist, skipping
DROP TABLE
--创建表flashtest。
gaussdb=# CREATE TABLE "public".flashtest (col1 INT,col2 TEXT) with(storage_type=ustore);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'col1' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
--查询csn。
gaussdb=# SELECT int8in(xidout(next_csn)) FROM gs_get_next_xid_csn();
  int8in
----------
 79351682
 79351682
 79351682
 79351682
 79351682
 79351682
(6 rows)
--查询当前时间戳。
gaussdb=# SELECT now();
              now
-------------------------------
 2023-09-13 19:35:26.011986+08
(1 row)
--插入数据。
gaussdb=# INSERT INTO flashtest VALUES(1,'INSERT1'),(2,'INSERT2'),(3,'INSERT3'),(4,'INSERT4'),
(5,'INSERT5'),(6,'INSERT6');
INSERT 0 6
gaussdb=# SELECT * FROM flashtest;
 col1 |  col2
------+---------
    3 | INSERT3
```

 版权所有 © 华为云计算技术有限公司

```
   1 | INSERT1
   2 | INSERT2
   4 | INSERT4
   5 | INSERT5
   6 | INSERT6
(6 rows)
--闪回查询某个csn处的表。
gaussdb=# SELECT * FROM flashtest TIMECAPSULE CSN 79351682;
 col1 | col2
------+------
(0 rows)
gaussdb=# SELECT * FROM flashtest;
 col1 |  col2
------+---------
   1 | INSERT1
   2 | INSERT2
   4 | INSERT4
   5 | INSERT5
   3 | INSERT3
   6 | INSERT6
(6 rows)
--闪回查询某个时间戳处的表。
gaussdb=# SELECT * FROM flashtest TIMECAPSULE TIMESTAMP '2023-09-13 19:35:26.011986';
 col1 | col2
------+------
(0 rows)
gaussdb=# SELECT * FROM flashtest;
 col1 |  col2
------+---------
   1 | INSERT1
   2 | INSERT2
   4 | INSERT4
   5 | INSERT5
   3 | INSERT3
   6 | INSERT6
(6 rows)
--闪回查询某个时间戳处的表。
gaussdb=# SELECT * FROM flashtest TIMECAPSULE TIMESTAMP to_timestamp ('2023-09-13
19:35:26.011986', 'YYYY-MM-DD HH24:MI:SS.FF');
 col1 | col2
------+------
(0 rows)
--闪回查询某个csn处的表，并对表进行重命名。
gaussdb=# SELECT * FROM flashtest AS ft TIMECAPSULE CSN 79351682;
 col1 | col2
------+------
(0 rows)
gaussdb=# DROP TABLE IF EXISTS "public".flashtest;
DROP TABLE
```

9.3.4.2 闪回表

背景信息

闪回表可以将表恢复至特定时间点，当逻辑损坏仅限于一个或一组表，而不是整个数据库时，此特性可以快速恢复表的数据。闪回表基于MVCC多版本机制，通过删除指定时间点和该时间点之后的增量数据，并找回指定时间点和当前时间点删除的数据，实现表级数据还原。

前提条件

整体方案分为三部分：旧版本保留、快照的维护和旧版本检索。旧版本保留：新增undo_retention_time配置参数，用来设置旧版本保留的时间，超过该时间的旧版本将被回收清理，请联系管理员修改。

 版权所有 © 华为云计算技术有限公司

语法

```
TIMECAPSULE TABLE table_name TO { TIMESTAMP | CSN } expression
```

使用示例

```
gaussdb=# DROP TABLE IF EXISTS "public".flashtest;
NOTICE:  table "flashtest" does not exist, skipping
DROP TABLE
--创建表
gaussdb=# CREATE TABLE "public".flashtest (col1 INT,col2 TEXT) with(storage_type=ustore);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'col1' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
--查询csn
gaussdb=# SELECT int8in(xidout(next_csn)) FROM gs_get_next_xid_csn();
  int8in
----------
 79352065
 79352065
 79352065
 79352065
 79352065
 79352065
(6 rows)
--查询当前的时间戳
gaussdb=# SELECT now();
              now
-------------------------------
 2023-09-13 19:46:34.102863+08
(1 row)
--查看表flashtest
gaussdb=# SELECT * FROM flashtest;
 col1 | col2
------+------
(0 rows)
--插入数据
gaussdb=# INSERT INTO flashtest VALUES(1,'INSERT1'),(2,'INSERT2'),(3,'INSERT3'),(4,'INSERT4'),
(5,'INSERT5'),(6,'INSERT6');
INSERT 0 6
gaussdb=# SELECT * FROM flashtest;
 col1 |  col2
------+---------
    3 | INSERT3
    1 | INSERT1
    2 | INSERT2
    4 | INSERT4
    5 | INSERT5
    6 | INSERT6
(6 rows)
--闪回表至特定csn
gaussdb=# TIMECAPSULE TABLE flashtest TO CSN 79352065;
TimeCapsule Table
gaussdb=# SELECT * FROM flashtest;
 col1 | col2
------+------
(0 rows)
gaussdb=# SELECT now();
              now
-------------------------------
 2023-09-13 19:52:21.551028+08
(1 row)
--插入数据
gaussdb=# INSERT INTO flashtest VALUES(1,'INSERT1'),(2,'INSERT2'),(3,'INSERT3'),(4,'INSERT4'),
(5,'INSERT5'),(6,'INSERT6');
INSERT 0 6
gaussdb=# SELECT * FROM flashtest;
 col1 |  col2
------+---------
```

 版权所有 © 华为云计算技术有限公司

```
   3 | INSERT3
   6 | INSERT6
   1 | INSERT1
   2 | INSERT2
   4 | INSERT4
   5 | INSERT5
(6 rows)
--闪回表至特定的时间戳
gaussdb=# TIMECAPSULE TABLE flashtest TO TIMESTAMP to_timestamp ('2023-09-13 19:52:21.551028',
'YYYY-MM-DD HH24:MI:SS.FF');
TimeCapsule Table
gaussdb=# SELECT * FROM flashtest;
 col1 | col2
------+------
(0 rows)
gaussdb=# select now();
              now
-------------------------------
 2023-09-13 19:54:00.641506+08
(1 row)
--插入数据
gaussdb=# INSERT INTO flashtest VALUES(1,'INSERT1'),(2,'INSERT2'),(3,'INSERT3'),(4,'INSERT4'),
(5,'INSERT5'),(6,'INSERT6');
INSERT 0 6
gaussdb=# SELECT * FROM flashtest;
 col1 |  col2
------+---------
   3 | INSERT3
   6 | INSERT6
   1 | INSERT1
   2 | INSERT2
   4 | INSERT4
   5 | INSERT5
(6 rows)
--闪回表至特定的时间戳
gaussdb=# TIMECAPSULE TABLE flashtest TO TIMESTAMP '2023-09-13 19:54:00.641506';
TimeCapsule Table
gaussdb=# SELECT * FROM flashtest;
 col1 | col2
------+------
(0 rows)
gaussdb=# DROP TABLE IF EXISTS "public".flashtest;
DROP TABLE
```

9.3.4.3 闪回 DROP/TRUNCATE

背景信息

- 闪回DROP：可以恢复意外删除的表，从回收站（recyclebin）中恢复被删除的表及其附属结构如索引、表约束等。闪回drop是基于回收站机制，通过还原回收站中记录的表的物理文件，实现已drop表的恢复。
- 闪回TRUNCATE：可以恢复误操作或意外被进行truncate的表，从回收站中恢复被truncate的表及索引的物理数据。闪回truncate基于回收站机制，通过还原回收站中记录的表的物理文件，实现已truncate表的恢复。

前提条件

- 开启enable_recyclebin参数（GUC参数在gaussdb.conf文件修改），启用回收站，请联系管理员修改。
- recyclebin_retention_time参数用于设置回收站对象保留时间，超过该时间的回收站对象将被自动清理，请联系管理员修改。

 版权所有 © 华为云计算技术有限公司

相关语法

- 删除表
```
DROP TABLE table_name [PURGE]
```
- 清理回收站对象
```
PURGE { TABLE { table_name }
| INDEX { index_name }
| RECYCLEBIN
}
```
- 闪回被删除的表
```
TIMECAPSULE TABLE { table_name } TO BEFORE DROP [RENAME TO new_tablename]
```
- 截断表
```
TRUNCATE TABLE { table_name } [ PURGE ]
```
- 闪回截断的表
```
TIMECAPSULE TABLE { table_name } TO BEFORE TRUNCATE
```

参数说明

- **DROP/TRUNCATE TABLE table_name PURGE**

  默认将表数据放入回收站中，PURGE直接清理。
- **PURGE RECYCLEBIN**

  表示清理回收站对象。
- **TO BEFORE DROP**

  使用这个子句检索回收站中已删除的表及其子对象。

  可以指定原始用户指定的表的名称，或对象删除时数据库分配的系统生成名称。
  - 回收站中系统生成的对象名称是唯一的。因此，如果指定系统生成名称，那么数据库检索指定的对象。使用“select * from gs_recyclebin;”语句查看回收站中的内容。
  - 在指定了用户指定的名称且回收站中包含多个该名称的对象的情况下，数据库检索回收站中最近移动的对象，如果想要检索更早版本的表，可以执行如下操作：
    - 指定你想要检索的表的系统生成名称。
    - 执行TIMECAPSULE TABLE ... TO BEFORE DROP语句，直到你要检索的表。
  - 恢复DROP表时，只恢复基表名，其他子对象名均保持回收站对象名。用户可根据需要，执行DDL命令手工调整子对象名。
  - 回收站对象不支持DML、DCL、DDL等写操作，不支持DQL查询操作（后续支持）。
  - 闪回点和当前点之间，执行过修改表结构或影响物理结构的语句，闪回失败。执行过DDL的表进行闪回操作报错：“ERROR：The table definition of %s has been changed.”。涉及namespace、表名改变等操作的DDL执行闪回操作报错： ERROR: recycle object %s desired does not exis；
  - 开启enable_recyclebin参数，启用回收站时，如果表上有truncate trigger、truncate表时，无法触发trigger。
- **RENAME TO**

  为从回收站中检索的表指定一个新名称。
- **TO BEFORE TRUNCATE**

 版权所有 © 华为云计算技术有限公司

闪回到TRUNCATE之前。

语法示例

```
-- PURGE TABLE table_name; --
--查看回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+---------------+----------------
+--------------+--------------+--------------+----------+--------------
-+----------------+--------------+-------------+--------------+----------------+-----------
(0 rows)

gaussdb=# DROP TABLE IF EXISTS flashtest;
NOTICE:  table "flashtest" does not exist, skipping
DROP TABLE
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+---------------+----------------
+--------------+--------------+--------------+----------+--------------
-+----------------+--------------+-------------+--------------+----------------+-----------
(0 rows)
--创建表flashtest
gaussdb=# CREATE TABLE IF NOT EXISTS flashtest(id int, name text) with (storage_type = ustore);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'id' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
--插入数据
gaussdb=# INSERT INTO flashtest VALUES(1, 'A');
INSERT 0 1
gaussdb=# SELECT * FROM flashtest;
 id | name
----+------
  1 | A
(1 row)
--DROP表flashtest
gaussdb=# DROP TABLE IF EXISTS flashtest;
DROP TABLE
--查看回收站，删除的表被放入回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |          rcyname          |    rcyoriginname     | rcyoperation | rcytype |
rcyrecyclecsn |        rcyrecycletime         | rcycreatecsn | rcychangecs
n | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid |
rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------------------------+----------------------+--------------+---------
+---------------+-------------------------------+--------------+-------------
--+--------------+----------+---------------+----------------+---------------+-------------+--------------
+----------------+-----------
    18591 |   12737 |    18585 | BIN$31C14EB4899$9737$0==$0   | flashtest            | d            |       0 |
79352606 | 2023-09-13 20:01:28.640664+08 |     79352595 |     7935259
5 |         2200 |       10 |             0 |          18585 | t            | t           | 225492       |         225492 |
    18591 |   12737 |    18588 | BIN$31C14EB489C$12D1BF60==$0 | pg_toast_18585       | d            |       2
|      79352606 | 2023-09-13 20:01:28.641018+08 |            0 |
0 |          99 |       10 |             0 |          18588 | f            | f           | 225492       |         225492 |
(2 rows)
--查看表flashtest，表不存在
gaussdb=# SELECT * FROM flashtest;
ERROR:  relation "flashtest" does not exist
LINE 1: SELECT * FROM flashtest;
                      ^
--PURGE表，将回收站中的表删除
gaussdb=# PURGE TABLE flashtest;
PURGE TABLE
--查看回收站，回收站中的表被删除
gaussdb=# SELECT * FROM gs_recyclebin;
```

 版权所有 © 华为云计算技术有限公司

```
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+--------------+--------------+---------+--------------+----------------
+--------------+-------------+--------------+----------+--------------
-+----------------+---------------+-------------+--------------+----------------+-----------
(0 rows)


-- PURGE INDEX index_name; --
gaussdb=# DROP TABLE IF EXISTS flashtest;
NOTICE:  table "flashtest" does not exist, skipping
DROP TABLE
--创建表flashtest
gaussdb=# CREATE TABLE IF NOT EXISTS flashtest(id int, name text) WITH (storage_type = ustore);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'id' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
--为表flashtest创建索引flashtest_index
gaussdb=# CREATE INDEX flashtest_index ON flashtest(id);
CREATE INDEX
--查看flashtest表的基本信息
gaussdb=# \d+ flashtest
                    Table "public.flashtest"
 Column |  Type   | Modifiers | Storage  | Stats target | Description
--------+---------+-----------+----------+--------------+-------------
 id     | integer |           | plain    |              |
 name   | text    |           | extended |              |
Indexes:
    "flashtest_index" ubtree (id) WITH (storage_type=USTORE) TABLESPACE pg_default
Has OIDs: no
Distribute By: HASH(id)
Location Nodes: ALL DATANODES
Options: orientation=row, storage_type=ustore, compression=no, segment=off,toast.storage_type=ustore,
toast.toast_storage_type=enhanced_toast

--DROP表
gaussdb=# DROP TABLE IF EXISTS flashtest;
DROP TABLE
--查看回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |           rcyname            |     rcyoriginname     | rcyoperation | rcytype |
rcyrecyclecsn |        rcyrecycletime         | rcycreatecsn | rcychangecs
n | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid |
rcyfrozenxid64 | rcybucket
-----------+---------+----------+------------------------------+-----------------------+--------------+---------
+---------------+-------------------------------+--------------+------------
--+--------------+----------+---------------+----------------+---------------+-------------+--------------
+---------------+----------
     18648 |   12737 |    18641 | BIN$31C14EB48D1$9A85$0==$0   | flashtest             | d            |       0 |
79354509 | 2023-09-13 20:40:11.360638+08 |     79354506 |     7935450
8 |         2200 |       10 |            0 |         18641 | t            | t           | 226642       |         226642 |
     18648 |   12737 |    18644 | BIN$31C14EB48D4$12E236A0==$0 | pg_toast_18641        | d            |       2
|      79354509 | 2023-09-13 20:40:11.36112+08  |            0 |
0 |          99 |       10 |            0 |         18644 | f            | f           | 226642       |         226642 |
     18648 |   12737 |    18647 | BIN$31C14EB48D7$9A85$0==$0   | flashtest_index       | d            |       1 |
79354509 | 2023-09-13 20:40:11.361246+08 |     79354508 |     7935450
8 |         2200 |       10 |            0 |         18647 | f            | t           | 0            |             0 |
(3 rows)

--PURGE索引flashtest_index
gaussdb=# PURGE INDEX flashtest_index;
PURGE INDEX
--查看回收站，回收站中的索引flashtest_index被删除
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |           rcyname            |     rcyoriginname     | rcyoperation | rcytype |
rcyrecyclecsn |        rcyrecycletime         | rcycreatecsn | rcychangecs
n | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid |
rcyfrozenxid64 | rcybucket
```

 版权所有 © 华为云计算技术有限公司

```
-----------+---------+----------+-----------------------------+---------------------+--------------+---------
+---------------+-----------------------------+--------------+------------
--+--------------+----------+--------------+---------------+--------------+-------------+--------------
+---------------+-----------
    18648 |   12737 |    18641 | BIN$31C14EB48D1$9A85$0==$0   | flashtest           | d            |       0 |
79354509 | 2023-09-13 20:40:11.360638+08 |     79354506 |     7935450
8 |         2200 |       10 |             0 |          18641 | t            | t           | 226642       |        226642 |
    18648 |   12737 |    18644 | BIN$31C14EB48D4$12E236A0==$0 | pg_toast_18641      | d            |       2
|     79354509 | 2023-09-13 20:40:11.36112+08  |             0 |
0 |          99 |       10 |             0 |          18644 | f            | f           | 226642       |        226642 |
(2 rows)

-- PURGE RECYCLEBIN --
--PURGE回收站
gaussdb=# PURGE RECYCLEBIN;
PURGE RECYCLEBIN
--查看回收站，回收站被清空
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+--------------+----------------
+--------------+--------------+--------------+----------+--------------
-+----------------+---------------+-------------+--------------+----------------+-----------
(0 rows)

-- TIMECAPSULE TABLE { table_name } TO BEFORE DROP [RENAME TO new_tablename] --
gaussdb=# DROP TABLE IF EXISTS flashtest;
NOTICE:  table "flashtest" does not exist, skipping
DROP TABLE
--创建表flashtest
gaussdb=# CREATE TABLE IF NOT EXISTS flashtest(id int, name text) with (storage_type = ustore);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'id' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
--插入数据
gaussdb=# INSERT INTO flashtest VALUES(1, 'A');
INSERT 0 1
gaussdb=# SELECT * FROM flashtest;
 id | name
----+------
  1 | A
(1 row)

--DROP表
gaussdb=# DROP TABLE IF EXISTS flashtest;
DROP TABLE
--查看回收站，表被放入回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |          rcyname           |    rcyoriginname     | rcyoperation | rcytype |
rcyrecyclecsn |        rcyrecycletime         | rcycreatecsn | rcychangecs
n | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid |
rcyfrozenxid64 | rcybucket
-----------+---------+----------+-----------------------------+----------------------+--------------+---------
+---------------+-----------------------------+--------------+------------
--+--------------+----------+---------------+----------------+---------------+-------------+--------------
+----------------+-----------
    18658 |   12737 |    18652 | BIN$31C14EB48DC$9B2B$0==$0   | flashtest           | d            |       0 |
79354760 | 2023-09-13 20:47:57.075907+08 |     79354753 |     7935475
3 |         2200 |       10 |             0 |          18652 | t            | t           | 226824       |        226824 |
    18658 |   12737 |    18655 | BIN$31C14EB48DF$12E46400==$0 | pg_toast_18652      | d            |       2
|     79354760 | 2023-09-13 20:47:57.07621+08  |             0 |
0 |          99 |       10 |             0 |          18655 | f            | f           | 226824       |        226824 |
(2 rows)

--查看表，表不存在
gaussdb=# SELECT * FROM flashtest;
ERROR:  relation "flashtest" does not exist
LINE 1: select * from flashtest;
```

 版权所有 © 华为云计算技术有限公司

```
                ^
--闪回drop表
gaussdb=# TIMECAPSULE TABLE flashtest to before drop;
TimeCapsule Table
--查看表，表被恢复到drop之前
gaussdb=# SELECT * FROM flashtest;
 id | name
----+------
  1 | A
(1 row)

--查看回收站，回收站中的表被删除
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+---------------+----------------
+--------------+--------------+--------------+----------+--------------
-+----------------+---------------+-------------+--------------+----------------+-----------
(0 rows)

--DROP表
gaussdb=# DROP TABLE IF EXISTS flashtest;
DROP TABLE
gaussdb=# SELECT * FROM flashtest;
ERROR:  relation "flashtest" does not exist
LINE 1: SELECT * FROM flashtest;
                ^
--查看回收站，表被放入回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |          rcyname           |        rcyoriginname         | rcyoperation | rcytype |
rcyrecyclecsn |        rcyrecycletime         | rcycreatecsn | rcy
changecsn | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge |
rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+-------------------------------+-------------------------------+--------------+---------
+---------------+--------------------------------+--------------+----
----------+--------------+----------+---------------+----------------+---------------+-------------+--------------
+----------------+-----------
    18664 |   12737 |    18652 | BIN$31C14EB48DC$9B4E$0==$0   | flashtest                     | d            |       0
|      79354845 | 2023-09-13 20:49:17.762977+08 |     79354753 |
 79354753 |         2200 |       10 |            0 |         18652 | t            | t           | 226824       |         226824 |
    18664 |   12737 |    18657 | BIN$31C14EB48E1$12E680A8==$0 | BIN$31C14EB48E1$12E45E00==$0 |
d            |       3 |      79354845 | 2023-09-13 20:49:17.763271+08 |     79354753 |
 79354753 |          99 |       10 |            0 |         18657 | f            | f           | 0            |              0 |
    18664 |   12737 |    18655 | BIN$31C14EB48DF$12E68698==$0 | BIN$31C14EB48DF$12E46400==$0 |
d            |       2 |      79354845 | 2023-09-13 20:49:17.763343+08 |            0 |
       0 |          99 |       10 |            0 |         18655 | f            | f           | 226824       |         226824 |
(3 rows)

--闪回drop表，表名用回收站中的rcyname
gaussdb=# TIMECAPSULE TABLE "BIN$31C14EB48DC$9B4E$0==$0" to before drop;
TimeCapsule Table
----查看回收站，回收站中的表被删除
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+---------------+----------------
+--------------+--------------+--------------+----------+--------------
-+----------------+---------------+-------------+--------------+----------------+-----------
(0 rows)

gaussdb=# SELECT * FROM flashtest;
 id | name
----+------
  1 | A
(1 row)

--DROP表
```

 版权所有 © 华为云计算技术有限公司

```
gaussdb=# DROP TABLE IF EXISTS flashtest;
DROP TABLE
----查看回收站，表被放入回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |           rcyname            |       rcyoriginname        | rcyoperation | rcytype |
rcyrecyclecsn |       rcyrecycletime       | rcycreatecsn | rcy
changecsn | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge |
rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+------------------------------+----------------------------+--------------+---------
+---------------+------------------------------+--------------+----
----------+--------------+----------+---------------+----------------+---------------+-------------+--------------
+----------------+-----------
     18667 |   12737 |    18652 | BIN$31C14EB48DC$9B8D$0==$0   | flashtest                  | d            |       0
|      79354943 | 2023-09-13 20:52:14.525946+08 |      79354753 |
 79354753 |         2200 |       10 |             0 |          18652 | t            | t           | 226824       |         226824 |
     18667 |   12737 |    18657 | BIN$31C14EB48E1$1320B4F0==$0 | BIN$31C14EB48E1$12E680A8==$0 |
d            |       3 |      79354943 | 2023-09-13 20:52:14.526319+08 |      79354753 |
 79354753 |          99 |       10 |             0 |          18657 | f            | f           | 0            |             0 |
     18667 |   12737 |    18655 | BIN$31C14EB48DF$1320BAE0==$0 | BIN$31C14EB48DF$12E68698==$0 |
d            |       2 |      79354943 | 2023-09-13 20:52:14.526423+08 |            0 |
       0 |          99 |       10 |             0 |          18655 | f            | f           | 226824       |         226824 |
(3 rows)

--查看表，表不存在
gaussdb=# SELECT * FROM flashtest;
ERROR:  relation "flashtest" does not exist
LINE 1: SELECT * FROM flashtest;
                      ^
--闪回drop表，并重命名表
gaussdb=# TIMECAPSULE TABLE flashtest to before drop rename to flashtest_rename;
TimeCapsule Table
--查看原表，表不存在
gaussdb=# SELECT * FROM flashtest;
ERROR:  relation "flashtest" does not exist
LINE 1: SELECT * FROM flashtest;
                      ^
--查看重命名后的表，表存在
gaussdb=# SELECT * FROM flashtest_rename;
 id | name
----+------
  1 | A
(1 row)

--查看回收站，回收站中的表被删除
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+---------------+----------------
+--------------+--------------+--------------+----------+--------------
-+----------------+---------------+-------------+--------------+----------------+-----------
(0 rows)
--drop表
gaussdb=# DROP TABLE IF EXISTS flashtest_rename;
DROP TABLE
--清空回收站
gaussdb=# PURGE RECYCLEBIN;
PURGE RECYCLEBIN
--查看回收站，回收站被清空
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+---------------+----------------
+--------------+--------------+--------------+----------+--------------
-+----------------+---------------+-------------+--------------+----------------+-----------
(0 rows)
```

 版权所有 © 华为云计算技术有限公司

```
-- TIMECAPSULE TABLE { table_name } TO BEFORE TRUNCATE --
gaussdb=# DROP TABLE IF EXISTS flashtest;
NOTICE:  table "flashtest" does not exist, skipping
DROP TABLE
--创建表flashtest
gaussdb=# CREATE TABLE IF NOT EXISTS flashtest(id int, name text) WITH (storage_type = ustore);
NOTICE:  The 'DISTRIBUTE BY' clause is not specified. Using 'id' as the distribution column by default.
HINT:  Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
--插入数据
gaussdb=# INSERT INTO flashtest VALUES(1, 'A');
INSERT 0 1
gaussdb=# SELECT * FROM flashtest;
 id | name
----+------
  1 | A
(1 row)

--truncate表
gaussdb=# TRUNCATE TABLE flashtest;
TRUNCATE TABLE
--查看回收站，表的数据被放入回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |           rcyname            |     rcyoriginname     | rcyoperation | rcytype |
rcyrecyclecsn |        rcyrecycletime         | rcycreatecsn | rcychangecs
n | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid |
rcyfrozenxid64 | rcybucket
-----------+---------+----------+------------------------------+----------------------+--------------+---------
+---------------+-------------------------------+--------------+------------
--+--------------+----------+---------------+----------------+---------------+-------------+--------------
+----------------+-----------
    18703 |   12737 |    18697 | BIN$31C14EB4909$9E4C$0==$0   | flashtest            | t            |       0 |
79356608 | 2023-09-13 21:24:42.819863+08 |     79356606 |     7935660
6 |         2200 |       10 |             0 |          18697 | t            | t           | 227927       |         227927 |
    18703 |   12737 |    18700 | BIN$31C14EB490C$132FE3F0==$0 | pg_toast_18697       | t            |       2
|      79356608 | 2023-09-13 21:24:42.820358+08 |            0 |
0 |          99 |       10 |             0 |          18700 | f            | f           | 227927       |         227927 |
(2 rows)

--查看表，表中的数据为空
gaussdb=# SELECT * FROM flashtest;
 id | name
----+------
(0 rows)

--闪回truncate表
gaussdb=# TIMECAPSULE TABLE flashtest to before truncate;
TimeCapsule Table
--查看表，表中的数据被恢复
gaussdb=# SELECT * FROM flashtest;
 id | name
----+------
  1 | A
(1 row)

--查看回收站
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid |           rcyname            |     rcyoriginname     | rcyoperation | rcytype |
rcyrecyclecsn |        rcyrecycletime         | rcycreatecsn | rcychangecs
n | rcynamespace | rcyowner | rcytablespace | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid |
rcyfrozenxid64 | rcybucket
-----------+---------+----------+------------------------------+----------------------+--------------+---------
+---------------+-------------------------------+--------------+------------
--+--------------+----------+---------------+----------------+---------------+-------------+--------------
+----------------+-----------
    18703 |   12737 |    18700 | BIN$31C14EB490C$13300228==$0 | pg_toast_18697       | t            |       2
|      79356610 | 2023-09-13 21:24:42.872732+08 |            0 |
0 |          99 |       10 |             0 |          18706 | f            | f           | 0            |         227928 |
    18703 |   12737 |    18697 | BIN$31C14EB4909$9E4D$0==$0   | flashtest            | t            |       0 |
```

 版权所有 © 华为云计算技术有限公司

```
79356610 | 2023-09-13 21:24:42.872792+08 |     79356606 |     7935660
6 |         2200 |       10 |             0 |          18704 | t             | t           | 0           |         227928 |
(2 rows)

--drop表
gaussdb=# DROP TABLE IF EXISTS flashtest;
DROP TABLE
--清空回收站
gaussdb=# PURGE RECYCLEBIN;
PURGE RECYCLEBIN
--查看回收站，回收站被清空
gaussdb=# SELECT * FROM gs_recyclebin;
 rcybaseid | rcydbid | rcyrelid | rcyname | rcyoriginname | rcyoperation | rcytype | rcyrecyclecsn |
rcyrecycletime | rcycreatecsn | rcychangecsn | rcynamespace | rcyowner | rcytablespace
 | rcyrelfilenode | rcycanrestore | rcycanpurge | rcyfrozenxid | rcyfrozenxid64 | rcybucket
-----------+---------+----------+---------+---------------+--------------+---------+--------------+----------------
+--------------+--------------+--------------+----------+----------+--------------
-+----------------+---------------+-------------+--------------+----------------+-----------
(0 rows)
```

9.3.5 常用视图工具

视图类型	类型	功能描述	使用场景	函数名称
解析	全类型	用于解析指定表页面，并返回存放解析内容的路径。	• 查看页面信息。 • 查看元组（非用户数据）信息。 • 页面或者元组损坏。 • 元组可见性问题。 • 校验报错问题。	gs_parse_page_bypath
	索引回收队列（URQ）	用于解析UB-tree索引回收队列关键信息。	• UB-tree索引空间膨胀。 • UB-tree索引空间回收异常。 • 校验报错问题。	gs_urq_dump_stat

版权所有 © 华为云计算技术有限公司

视图类型	类型	功能描述	使用场景	函数名称
	回滚段（Undo）	用于解析指定Undo Record的内容，不包含旧版本元组的数据。	• undo空间膨胀。 • undo回收异常。 • 回滚异常。 • 日常巡检。 • 校验报错。 • 可见性判断异常。 • 修改参数。	gs_undo_dump_record
		用于解析指定事务生成的所有Undo Record，不包含旧版本元组的数据。		gs_undo_dump_xid
		用于解析指定UndoZone中所有Transaction Slot信息。		gs_undo_translot_dump_slot
		用于解析指定事务对应Transaction Slot信息，包括事务XID和该事务生成的Undo Record范围.		gs_undo_translot_dump_xid
		用于解析指定Undo Zone的元信息，显示Undo Record和Transaction Slot指针使用情况。		gs_undo_meta_dump_zone
		用于解析指定Undo Zone对应Undo Space的元信息，显示Undo Record文件使用情况。		gs_undo_meta_dump_spaces
		用于解析指定Undo Zone对应Slot Space的元信息，显示Transaction Slot文件使用情况。		gs_undo_meta_dump_slot
		用于解析数据页和数据页上数据的所有历史版本，并返回存放解析内容的路径。		gs_undo_dump_parsepage_mv
	预写日志（WAL）	用于解析指定LSN范围之内的xLog日志，并返回存放解析内容的路径。可以通过pg_current_xlog_location()获取当前xLog位置。	• WAL日志出错。 • 日志回放出错。 • 页面损坏。	gs_xlogdump_lsn
		用于解析指定XID的xLog日志，并返回存放解析内容的路径。可以通过txid_current()获取当前事务ID。		gs_xlogdump_xid
		用于解析指定表页面对应的日志，并返回存放解析内容的路径。		gs_xlogdump_tablepath
		用于解析指定表页面和表页面对应的日志，并返回存放解析内容的路径。可以看做一次执行gs_parse_page_bypath和gs_xlogdump_tablepath。该函数执行的前置条件是表文件存在。如果想查看已删除的表的相关日志，请直接调用gs_xlogdump_tablepath。		gs_xlogdump_parsepage_tablepath
统计	回滚段（Undo）	用于显示Undo模块的统计信息，包括Undo Zone使用情况、Undo链使用情况、Undo模块文件创建删除情况和Undo模块参数设置推荐值。	• Undo空间膨胀。 • Undo资源监控。	gs_stat_undo

 版权所有 © 华为云计算技术有限公司

视图类型	类型	功能描述	使用场景	函数名称
	预写日志（WAL）	用于统计预写日志（WAL）写盘时的内存状态表内容。	• WAL写/刷盘监控。 • WAL写/刷盘hang住。	gs_stat_wal_entrytable
		用于统计预写日志（WAL）刷盘状态、位置统计信息。		gs_walwriter_flush_position
		用于统计预写日志（WAL）写刷盘次数频率、数据量以及刷盘文件统计信息。		gs_walwriter_flush_stat
校验	堆表/索引	用于离线校验表或者索引文件磁盘页面数据是否异常。	• 页面损坏或者元组损坏。 • 可见性问题。 • 日志回放出错问题。	ANALYZE VERIFY
		用于校验当前实例当前库物理文件是否存在丢失。	文件丢失。	gs_verify_data_file
	索引回收队列（URQ）	用于校验UB-tree索引回收队列（潜在队列/可用队列/单页面）数据是否异常。	• UB-tree索引空间膨胀。 • UB-tree索引空间回收异常。	gs_verify_urq
	回滚段（Undo）	用于离线校验Undo Record数据是否存在异常。	• Undo Record异常或者损坏。 • 可见性问题。 • 回滚出错或者异常。	gs_verify_undo_record
		用于离线校验Transaction Slot数据是否存在异常。	• Undo Record异常或者损坏。 • 可见性问题。 • 回滚出错或者异常。	gs_verify_undo_slot

版权所有 © 华为云计算技术有限公司

视图类型	类型	功能描述	使用场景	函数名称
		用于离线校验Undo元信息数据是否存在异常。	• 因Undo meta引起的节点无法启动问题。 • Undo空间回收异常。 • Snapshot too old问题。	gs_verify_undo_meta
修复	堆表/索引/Undo文件	用于基于备机修复主机丢失的物理文件。	堆表/索引/Undo文件丢失。	gs_repair_file
	堆表/索引/Undo页面	用于校验并基于备机修复主机受损页面。	堆表/索引/Undo页面损坏。	gs_verify_and_tryrepair_page
		用于基于备机页面直接修复主机页面。		gs_repair_page
		用于基于偏移量对页面的备份进行字节修改。		gs_edit_page_bypath
		用于将修改后的页面覆盖写入到目标页面。		gs_repair_page_bypath
	回滚段（Undo）	用于重建Undo元信息，如果校验发现Undo元信息没有问题则不重建。	Undo元信息异常或者损坏。	gs_repair_undo_byzone
	索引回收队列（URQ）	用于重建UB-tree索引回收队列。	索引回收队列异常或者损坏。	gs_repair_urq

9.3.6 常见问题及定位手段

9.3.6.1 snapshot too old

查询SQL执行时间过长或者其他一些原因，Undo无法保存太久的历史数据就可能因为历史版本被强制回收报错。一般情况下需要扩容回滚段空间，但具体问题需要具体分析。

9.3.6.1.1 长事务阻塞 Undo 空间回收

问题现象

1. gs_log中打印如下错误：

版权所有 © 华为云计算技术有限公司

```
snapshot too old! the undo record has been forcibly discarded
xid xxx, the undo size xxx of the transaction exceeds the threshold xxx. trans_undo_threshold_size
xxx,undo_space_limit_size xxx.
```

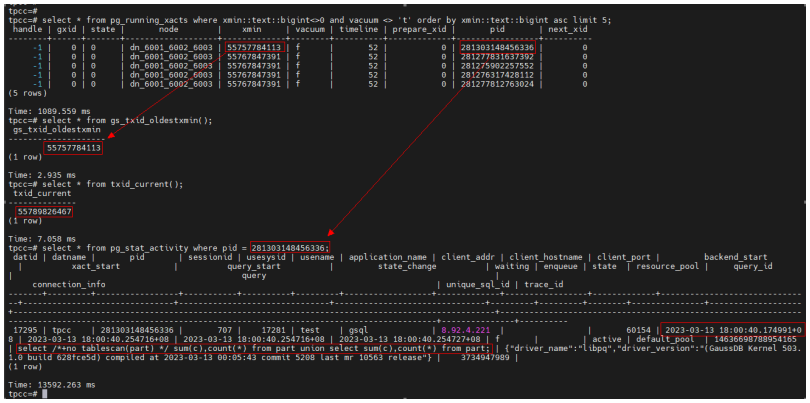
在真实报错信息中，上文中的xxx为实际数据。

2. global_recycle_xid（Undo子系统的全局回收事务XID）长时间不发生变化。

```
gaussdb=# select * from gs_undo_meta_dump_slot(1,-1);
 zone_id | allocate | recycle | frozen_xid | global_frozen_xid | recycle_xid | global_recycle_xid
---------+----------+---------+------------+-------------------+-------------+--------------------
       1 | 280      | 248     | 17028      | 17028             | 17025       | 17028
(1 row)
```

3. pg_running_xacts与pg_stat_activity视图查询存在长事务，阻塞oldestxmin和global_recycle_xid推进。如果pg_running_xacts中查询活跃事务的xmin和gs_txid_oldestxmin相等，且通过pid查询pg_stat_activity查询线程执行语句时间过长，则表明有长事务卡主了回收。

```
SELECT * FROM pg_running_xacts where xmin::text::bigint<>0 and vacuum <> 't' order by
xmin::text::bigint asc limit 5;
SELECT * FROM gs_txid_oldestxmin();
SELECT * FROM pg_stat_activity WHERE pid = 长事务所在线程PID;
```



处理方法

通过pg_terminate_session(pid, sessionid)终止长事务所在的会话（提醒：长事务无固定快速恢复手段，强制结束SQL语句为其中一种常用操作，属于高危操作，执行需谨慎，执行前需与业务及华为技术确认，避免造成业务失败或报错）。

9.3.6.1.2 大量回滚事务拖慢 Undo 空间回收

问题现象

使用gs_async_rollback_xact_status视图查看有大量的待回滚事务，且待回滚的事务数量维持不变或者持续增高。

```
SELECT * FROM gs_async_rollback_xact_status();
```

处理方法

调大异步回滚线程数量，调整方式有以下两种：

方式1：在gaussdb.conf中配置max_undo_workers，然后重启节点。

方式2：gs_guc reload -Z NODE-TYPE [-N NODE-NAME] [-I INSTANCE-NAME | -D DATADIR] -c max_undo_workers=100 重启实例。

版权所有 © 华为云计算技术有限公司

9.3.6.2 storage test error

业务执行过程中，数据页、索引或者Undo页面发生变更后，该页面放锁之前会主动进行逻辑损坏检测，发现页面损坏问题后会输出包含“storage test error”关键字的日志信息到数据库运行日志（gs_log文件），执行事务回滚，页面会恢复到修改前的状态。

问题现象

gs_log中打印“storage test error”关键字。

处理方法

请联系华为技术支持解决。

9.3.6.3 备机读业务报错:"UBTreeSearch::read_page has conflict with recovery, please try again later"

问题现象

业务在使用备机读时，出现报错（错误码43244），错误信息中包含“UBTreeSearch::read_page has conflict with recovery, please try again later”关键字。

问题分析

在开启并行回放或串行回放的情况下（查询GUC参数recovery_parse_workers和recovery_max_workers均是1为串行回放；recovery_parse_workers是1，recovery_max_workers大于1为并行回放），备机的查询线程在做索引扫描时，会先对索引页面加读锁，每当扫到一个元组时会去判可见性。如果该元组对应的事务处于committing状态，需要等待该事务提交后再判断。而备机上的事务提交是依赖日志回放线程推进的，这个过程中会对索引页面进行修改，因此需要加锁。查询线程在等待过程中会释放索引页面的锁，否则会出现查询线程等待回放线程进行事务提交，而回放线程在等待查询线程释放锁。

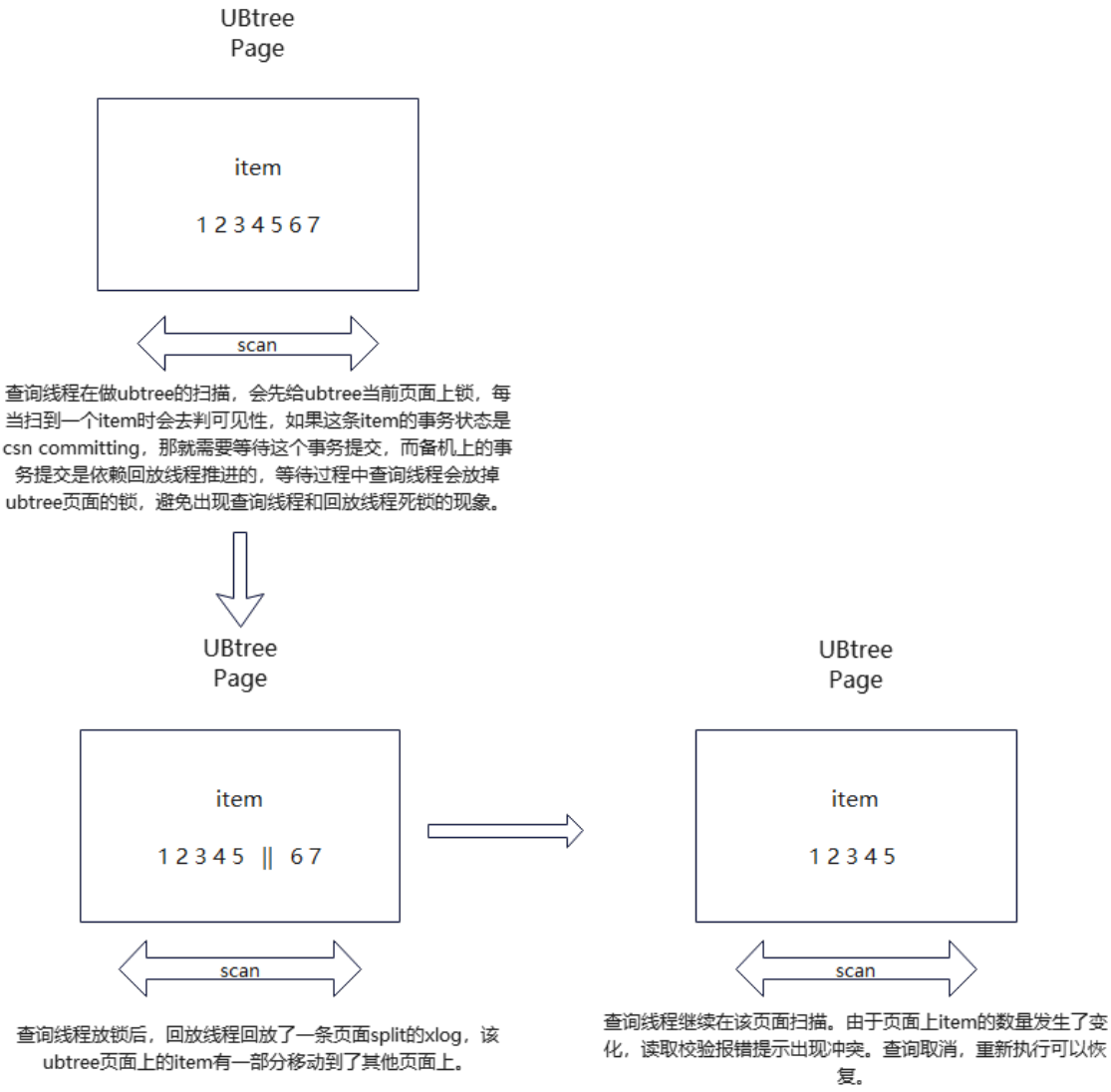
该报错仅出现在查询与回放都需要访问同一个索引页面的场景下，查询线程在释放锁并等待事务结束过程中，访问的页面出现被修改的情况。具体流程图如下图1所示：

说明

- 备机查询在扫到committing状态的元组时，需要等待事务提交是因为事务提交的顺序与产生日志的顺序可能是乱序的。例如主机上tx_1的事务比tx_2先提交，而备机上tx_1的commit日志在tx_2的commit日志之后回放，按照事务提交顺序来看tx_1对tx_2应当是可见的，所以需要等待事务提交。
- 备机查询在扫描索引页面时，发现页面元组数量（包含死元组）发生变化后不可重试，是因为在扫描时可能为正向或反向扫描，而举例来说页面发生分裂后一部分元组移动到右页面，在反向扫描的情况下即使重试只能向左扫描读取，无法再保证结果的正确性，并且由于无法分辨发生分裂或者插入，所以不可重试。

 版权所有 © 华为云计算技术有限公司

图 9-6 问题分析



处理方法

出现报错时，建议重试查询。另外建议选择非频繁更新的索引字段、采用软删除的方式（物理删除操作在业务低谷期执行），可以降低出现该报错的概率。

9.3.6.4 长查询执行期间大量并发更新偶现写入性能下降

问题现象

执行全表扫描类型的长查询，扫描期间页面发生大量并发更新，部分DML写入性能较没有长查询时出现性能下降。

问题分析

对于全表扫描场景下的长查询（例如持续两小时以上），在扫描到某个页面前，该页面发生大量集中并发更新（例如十万次以上更新），后续扫描到该页面时，需要访问

 版权所有 © 华为云计算技术有限公司

大量历史版本获取可见元组（MVCC机制），由于单页扫描期间持有页面读锁，若此时刚好需要写入该页面，写入会被阻塞，直到页面元组读取完成。

定位手段

1. 结合慢SQL告警、statement_history视图等确认是否存在长查询和超时取消的DML语句。
2. 获取1statement_history中查询到的被取消DML的details信息，使用statement_detail_decode系统函数解析details字段，获取等待事件，如等待事件开销占比最高为BufferContentLock则大概率为本问题。

处理方法

事前预防：避免在高并发表上执行全表扫描类型长查询，建议长查询迁移到备机执行。

事中处理：结合慢SQL告警等确认是否触发此场景，可通过中断长查询避免对业务的持续影响。

9.4 数据生命周期管理-OLTP 表压缩

9.4.1 特性简介

OLTP表压缩是GaussDB高级压缩中的一个特性。基于全新的压缩算法、细粒度的自动冷热判定和支持块内压缩等技术创新，可以在提供合理压缩率的同时大幅度降低对业务的影响、增加后台调度、增加查询Job执行状态以及节约空间，能够在支持关键在线业务的容量控制中发挥重要价值。

9.4.2 特性约束

- 不支持系统表、内存表、全局临时表、本地临时表和序列表，不支持压缩toast数据。
- 仅在ORA兼容模式、PG模式以及B兼容模式下有效。
- Ustore不支持编解码，压缩率低于Astore。
- 普通表开启压缩时，扩容空间预留需按照解压后的大小评估。
- HashBucket表不支持DBE_HEAT_MAP.ROW_HEAT_MAP和DBE_COMPRESSION.GET_COMPRESSION_TYPE。
- 扩容期间不支持压缩调度。
- 扩容前请确认当前是否有正在执行的压缩任务，如果有的话，要么等待压缩任务结束，要么执行DBE_ILM.STOP_ILM或DBE_ILM_ADMIN.DISABLE_ILM停掉，扩容完成后再执行

  DBE_ILM_ADMIN.ENABLE_ILM开启。
- 由于当前版本新增了部分解压特性，会导致部分极小的表在判断压缩收益时不通过，从而不进行压缩。但此前没有部分解压的版本可以压缩。例如：单表单字段int1类型的数据从当前版本开始无法进行压缩。

 版权所有 © 华为云计算技术有限公司

9.4.3 特性规格

- TPCC只开启策略、不开调度对原有业务无影响。
- TPCC不开启压缩策略对原有业务无影响。
- TPCC.bmsql_order_line设置ILM策略（只识别完成派送的订单为冷行）不调度，TPmC劣化不高于2%（56核CPU370GB内存+3TB SSD硬盘，350GB SharedBuffer）。
- TPCC.bmsql_order_line设置ILM策略（只识别完成派送的订单为冷行）后台默认参数调度时，TPmC劣化不高于5%（56核CPU370GB内存+3TB SSD硬盘，350GB SharedBuffer）。
- 单线程ILM Job带宽约100MB/秒（56核CPU370GB内存+3TB SSD硬盘，350GB SharedBuffer）。

  度量方式：根据执行压缩的开始时间和结束时间以及压缩的页面个数计算带宽。
- get查询访问压缩数据比非压缩数据性能劣化，驱动侧不高于10%，plsql侧不高于15%（32MB SharedBuffer，6万页面数据）。
- multi-get查询访问压缩数据比非压缩数据性能劣化，驱动侧不高于30%，plsql侧不高于40%（32MB SharedBuffer，6万页面数据）。
- table-scan查询访问压缩数据比非压缩数据性能劣化，驱动侧不高于30%，plsql侧不高于40%（32MB SharedBuffer，6万页面数据）。
- TPCH.lineitem表压缩比（全冷行）不小于2:1。
- 对于TPC-C的Orderline表，以及TPC-H的Lineitem、Orders、Customer、Part表的测试表明，数值型字段较多时，压缩率高于LZ4和ZLIB；而文本型字段较多时，压缩率介于LZ类和LZ+Huffman组合类的压缩算法之间。

9.4.4 使用说明

使用高级压缩的功能，用户必须申请License才能使用。具体情况请联系华为工程师。

步骤1 执行如下命令开启压缩功能：

```
gaussdb=# ALTER DATABASE SET ilm = on;
```

检查当前数据库的public schema中是否存在gsilmpolicy_seq和gsilmtask_seq。

```
gaussdb=# \d
                                List of relations
 Schema |      Name       |   Type   |   Owner    |                  Storage
--------+-----------------+----------+------------+-----------------------------------------------------------
 public | gsilmpolicy_seq | sequence | omm |
 public | gsilmtask_seq   | sequence | omm |
```

或者：

```
gaussdb=# SELECT a.oid, a.relname FROM pg_class a inner join pg_namespace b on a.relnamespace = b.oid
    WHERE (a.relname = 'gsilmpolicy_seq' OR a.relname = 'gsilmtask_seq') AND b.nspname = 'public';

  oid  |     relname
-------+-----------------
 17002 | gsilmpolicy_seq
 17004 | gsilmtask_seq
(2 rows)
```

生成异常会报warning：

```
WARNING:  ILM sequences are already existed while initializing
```

 版权所有 © 华为云计算技术有限公司

步骤2 为表添加压缩策略。

- 新建带策略的表：

```
gaussdb=# CREATE TABLE ilm_table_1 (col1 int, col2 text)
    ilm add policy row store compress advanced row
    after 3 days of no modification on (col1 < 1000);
```

- 为存量表添加策略：

```
gaussdb=# CREATE TABLE ilm_table_2 (col1 int, col2 text);
gaussdb=# ALTER TABLE ilm_table_2 ilm add policy row store
    compress advanced row after 3 days of no modification;
```

- 检查策略视图中是否新增数据：

```
gaussdb=# SELECT * FROM gs_my_ilmpolicies;

 policy_name |  policy_type  | tablespace | enabled | deleted
-------------+---------------+------------+---------+---------
 p1          | DATA MOVEMENT |            | YES     | NO
 p2          | DATA MOVEMENT |            | YES     | NO
(2 rows)
```

- 检查策略详细信息视图中是否新增了符合刚刚设置的策略：

```
gaussdb=# SELECT * FROM gs_my_ilmdatamovementpolicies;

 policy_name | action_type | scope | compression_level | tier_tablespace | tier_status |
condition_type      | condition_days | custom_function | policy_subtype | action_clause | tier_to
-------------+-------------+-------+-------------------+-----------------+-------------+-------------------------
+----------------+-----------------+----------------+---------------+---------
 p1          | COMPRESSION | ROW   | ADVANCED          |                 |             | LAST MODIFICATION
TIME |              3 |                 |               |             |
 p2          | COMPRESSION | ROW   | ADVANCED          |                 |             | LAST MODIFICATION
TIME |              3 |                 |               |             |
(2 rows)
```

- 检查策略与目标表是否对应：

```
gaussdb=# SELECT * FROM gs_my_ilmobjects;
 policy_name | object_owner | object_name | subobject_name | object_type |     inherited_from     |
tbs_inherited_from | enabled | deleted
-------------+--------------+-------------+----------------+-------------+------------------------
+--------------------+---------+---------
 p1          | public       | ilm_table_1 |                | TABLE       | POLICY NOT INHERITED |                    |
YES     | NO
 p2          | public       | ilm_table_2 |                | TABLE       | POLICY NOT INHERITED |                    |
YES     | NO
(2 rows)
```

步骤3 执行压缩评估。

- 手动执行压缩评估。

> **⚠ 注意**
>
> 为方便测试，本功能环境参数中提供POLICY_TIME属性，决定时间条件以天为单位还是以秒为单位。通过下面语句调整：

```
gaussdb=# CALL DBE_ILM_ADMIN.CUSTOMIZE_ILM(11, 1);
```

插入随机数据用于测试：

```
gaussdb=# INSERT INTO ilm_table_1 select *, 'test_data' FROM generate_series(1, 10000);

gaussdb=# DECLARE
    v_taskid number;
gaussdb=# BEGIN
    DBE_ILM.EXECUTE_ILM(OWNER           => 'public',
                        OBJECT_NAME     => 'ilm_table_1',
                        TASK_ID         => v_taskid,
                        SUBOBJECT_NAME  => NULL,
```

 版权所有 © 华为云计算技术有限公司

```
                    POLICY_NAME     => 'ALL POLICIES',
                    EXECUTION_MODE  => 2);
    RAISE INFO 'Task ID is:%', v_taskid;
gaussdb=# END;
/
```

如入参有误，会报对应的错误信息。无误则无输出（上述代码段添加了RAISE INFO语句打印当前task的id）。

```
INFO:  Task ID is:1
```

检查task信息：

```
gaussdb=# SELECT * FROM gs_my_ilmtasks;

task_id | task_owner |   state   |         creation_time         |          start_time           |
completion_time
---------+------------+-----------+-------------------------------+-------------------------------
+-------------------------------
       1 | omm| COMPLETED | 2023-08-29 17:36:38.779555+08 | 2023-08-29 17:36:38.779555+08 |
2023-08-29 17:36:38.879485+08
(1 row)
```

检查评估结果：

```
gaussdb=# SELECT * FROM gs_my_ilmevaluationdetails;

 task_id | policy_name | object_owner | object_name | subobject_name | object_type |
selected_for_execution |      job_name      | comments
---------+-------------+--------------+-------------+----------------+-------------+------------------------
+--------------------+----------
       1 | p1          | public       | ilm_table_1 |                | TABLE       | SELECTED FOR EXECUTION | ilmjob
$_postgres1 |
(1 row)
```

检查压缩job信息：

```
gaussdb=# SELECT * FROM gs_my_ilmresults;

 task_id |     job_name      |       job_state        |          start_time           |       completion_time         |
comments |                statistics
---------+-------------------+------------------------+-------------------------------
+-------------------------------+----------+----------------------------------------
       1 | ilmjob$_postgres1 | COMPLETED SUCCESSFULLY | 2023-08-29 17:36:38.779555+08 |
2023-08-29 17:36:38.879485+08 |          | SpaceSaving=0,BoundTime=0,LastBlkNum=0
(1 row)
```

- 触发后台自动调度评估。

  使用初始用户登录template1数据库，创建维护窗口：

```
gaussdb=#
DECLARE
    V_HOUR INT := 22;
    V_MINUTE INT := 0;
    V_SECOND INT := 0;
    C_ADO_WINDOW_SCHEDULE_NAME TEXT := 'ado_window_schedule';
    C_ADO_WINDOW_PROGRAM_NAME TEXT := 'ado_window_program';
    C_MAINTENANCE_WINDOW_JOB_NAME TEXT := 'maintenance_window_job';
    V_MAINTENCE_WINDOW_REPEAT TEXT;
    V_MAINTENCE_WINDOW_START TIMESTAMPTZ;
    V_BE_SCHEDULE_ENABLE BOOL;
    V_MAINTENANCE_WINDOW_EXIST INT;
BEGIN
    SELECT COUNT(*) INTO V_MAINTENANCE_WINDOW_EXIST FROM PG_CATALOG.PG_JOB WHERE
JOB_NAME = 'maintenance_window_job' AND DBNAME = 'template1';
    IF CURRENT_DATABASE() != 'template1' THEN
        RAISE EXCEPTION 'Create maintenance_window FAILED, current database is not tempalte1';
    END IF;
    IF V_MAINTENANCE_WINDOW_EXIST = 0 AND CURRENT_DATABASE() = 'template1' THEN
        SELECT
            CASE
                WHEN NOW() < CURRENT_DATE + INTERVAL '22 HOUR' THEN CURRENT_DATE +
INTERVAL '22 HOUR'
                ELSE CURRENT_DATE + INTERVAL '1 DAY 22 HOUR'
            END INTO V_MAINTENCE_WINDOW_START;
```

 版权所有 © 华为云计算技术有限公司

```
    --1. prepare for maintence window schedule
    SELECT 'freq=daily;interval=1;byhour='||V_HOUR||';byminute='||V_MINUTE||';bysecond='||
V_SECOND INTO V_MAINTENCE_WINDOW_REPEAT;
    BEGIN
        SELECT
            CASE
                WHEN VALUE = 1 THEN TRUE -- DBE_ILM_ADMIN.ILM_ENABLED
                ELSE FALSE
            END INTO V_BE_SCHEDULE_ENABLE
            FROM PG_CATALOG.GS_ILM_PARAM WHERE IDX = 7; -- DBE_ILM_ADMIN.ENABLED
    EXCEPTION
        WHEN OTHERS THEN
            V_BE_SCHEDULE_ENABLE := FALSE;
    END;
    --2. Create ado window schedule
    DBE_SCHEDULER.CREATE_SCHEDULE(
        SCHEDULE_NAME => C_ADO_WINDOW_SCHEDULE_NAME,
        START_DATE => '9999-01-01 00:00:01',
        REPEAT_INTERVAL => NULL,
        END_DATE => NULL,
        COMMENTS => 'ado window schedule');
    --3. Create ado window program
    DBE_SCHEDULER.CREATE_PROGRAM(
        PROGRAM_NAME => C_ADO_WINDOW_PROGRAM_NAME,
        PROGRAM_TYPE => 'plsql_block',
        PROGRAM_ACTION => 'call prvt_ilm.be_execute_ilm(0);',
        NUMBER_OF_ARGUMENTS => 0,
        ENABLED => TRUE,
        COMMENTS => NULL);
    --4. Create maintenance window master job
    DBE_SCHEDULER.CREATE_JOB(
        JOB_NAME => C_MAINTENANCE_WINDOW_JOB_NAME,
        START_DATE => V_MAINTENCE_WINDOW_START,
        REPEAT_INTERVAL => V_MAINTENCE_WINDOW_REPEAT,
        END_DATE => NULL,
        JOB_TYPE => 'STORED_PROCEDURE'::TEXT,
        JOB_ACTION => 'prvt_ilm.be_active_ado_window'::TEXT,
        NUMBER_OF_ARGUMENTS => 0,
        ENABLED => V_BE_SCHEDULE_ENABLE,
        AUTO_DROP => FALSE,
        COMMENTS => 'maintenance window job',
        destination_name=> 'CCN');
  ELSE
    RAISE EXCEPTION 'CREATE ILM MAINTENANCE WINDOW FAILED';
  END IF;
END;
```

自动调度提供若干参数用于调整：

```
gaussdb=# SELECT * FROM gs_adm_ilmparameters;
              name               | value
---------------------------------+-------
 EXECUTION_INTERVAL              |    15
 RETENTION_TIME                  |    30
 ENABLED                         |     1
 POLICY_TIME                     |     0
 ABS_JOBLIMIT                    |    10
 JOB_SIZELIMIT                   |  1024
 WIND_DURATION                   |   240
 BLOCK_LIMITS                    |    40
 ENABLE_META_COMPRESSION         |     0
 SAMPLE_MIN                      |    10
 SAMPLE_MAX                      |    10
 CONST_PRIO                      |    40
 CONST_THRESHOLD                 |    90
 EQVALUE_PRIO                    |    60
 EQVALUE_THRESHOLD               |    80
 ENABLE_DELTA_ENCODE_SWITCH      |     1
 LZ4_COMPRESSION_LEVEL           |     0
 ENABLE_LZ4_PARTIAL_DECOMPRESSION |     1
(18 rows)
```

 版权所有 © 华为云计算技术有限公司

- EXECUTION_INTERVAL：自动调度任务执行间隔，默认每15分钟执行一次。
- RETENTION_TIME：历史压缩任务记录清理间隔，默认每30天清理一次。
- ENABLED：当前自动调度启用情况，默认为开启。
- POLICY_TIME：策略评估的时间单位，测试使用。默认以天为单位。
- ABS_JOBLIMIT：单次评估生成压缩任务数量上限，默认为10个。
- JOB_SIZELIMIT：单个压缩任务的IO上限，默认为1GB。
- WIND_DURATION：单次维护窗口的持续时间。
- BLOCK_LIMITS：控制实例级的行存压缩速率上限，默认是40，取值范围是0到10000（0表示不限制），单位是block/ms，表示每毫秒最多压缩多少个block。速率上限计算方法：BLOCK_LIMITS*1000*BLOCKSIZE，以默认值40为例，其速率上限为：40*1000*8KB=320000KB/s。
- ENABLE_META_COMPRESSION：是否开启header压缩，默认为0，取值范围为0（关闭）和1（开启）。

  说明

  设置此参数为1时，对于单行数据较短的表，压缩率会有一定提升，但是访问压缩行的性能会有较大幅度的下降。若数据库多是单行数据较长的表，不建议开启此参数。
- SAMPLE_MIN：常量编码和等值编码采样步长最小值，默认为10，取值范围[1, 100]，支持小数输入，小数会自动向下取整。
- SAMPLE_MAX：常量编码和等值编码采样步长最大值，默认为10，取值范围[1, 100]，支持小数输入，小数会自动向下取整。
- CONST_PRIO：常量编码优先级，默认为40，取值范围[0, 100]，100表示关闭常量编码，支持小数输入，小数会自动向下取整。
- CONST_THRESHOLD：常量编码阈值，默认为90，取值范围[1, 100]，表示一列常量值的占比超过该阈值时进行常量编码，支持小数输入，小数会自动向下取整。
- EQVALUE_PRIO：等值编码优先级，默认为60，取值范围[0, 100]，100表示关闭等值编码，支持小数输入，小数会自动向下取整。
- EQVALUE_THRESHOLD：等值编码阈值，默认为80，取值范围[1, 100]，表示两列数据的等值比例超过该阈值时进行等值编码，支持小数输入，小数会自动向下取整。
- ENABLE_DELTA_ENCODE_SWITCH：差值编码开关，默认为1，支持小数输入，0表示关闭，1表示开启，小数会自动向下取整。
- LZ4_COMPRESSION_LEVEL：lz4压缩等级，默认为0，取值范围[0, 16]，支持小数输入，小数会自动向下取整。
- ENABLE_LZ4_PARTIAL_DECOMPRESSION：部分解压开关，默认为1，支持小数输入，0表示关闭，1表示开启，小数会自动向下取整。

以上参数均可通过DBE_ILM_ADMIN.CUSTOMIZE_ILM()接口调整。

维护窗口默认每天晚上22：00（北京时间）开启，可通过DBE_SCHEDULER提供的接口SET_ATTRIBUTE进行设置：

```
\c template1
 CALL DBE_ILM_ADMIN.DISABLE_ILM();
 CALL DBE_ILM_ADMIN.ENABLE_ILM();
DECLARE
    newtime timestamptz := CLOCK_TIMESTAMP() + to_interval('2 seconds');
BEGIN
    DBE_SCHEDULER.set_attribute(
        name          =>          'maintenance_window_job',
        attribute     =>          'start_date',
```

 版权所有 © 华为云计算技术有限公司

```
        value       =>      TO_CHAR(newtime, 'YYYY-MM-DD HH24:MI:SS')
    );
END;
/
```

----结束

9.4.5 维护窗口参数配置

- RETENTION_TIME：评估与压缩记录的保留时长，单位天，默认值30。用户可根据自己存储容量自行调节。
- EXECUTION_INTERVAL：评估任务的执行频率，单位分钟，默认值15。用户可根据自己维护窗口期间业务与资源情况调节。该参数与ABS_JOBLIMIT相互影响。单日单线程最大可产生的I/O为WIND_DURATION/EXECUTION_INTERVAL*JOB_SIZELIMIT。
- JOB_SIZELIMIT：控制单个压缩Job可以处理的最大字节数，单位兆，默认值1024。压缩带宽约为100MB/秒，每个压缩Job限制I/O为1GB时，最多10秒完成。用户可根据自己业务闲时情况以及需要压缩的数据量自行调节。
- ABS_JOBLIMIT：控制一次评估最多生成多少个压缩Job。用户可根据自己设置策略的分区及表数量自己调节。建议最大不超过10，可以使用“select count(*) from gs_adm_ilmobjects where enabled = true”命令查询。
- POLICY_TIME：控制判定冷行的条件单位是天还是秒，秒仅用来做测试用。取值为：ILM_POLICY_IN_SECONDS或ILM_POLICY_IN_DAYS（默认值）。
- WIND_DURATION：维护窗口持续时长，单位分钟，默认240分钟（4小时）。维护窗口默认从北京时间22点开始持续240分钟，用户可根据自己业务闲时情况自行调节。
- BLOCK_LIMITS：控制实例级的行存压缩速率上限，默认是40，取值范围是0到10000(0表示不限制)，单位是block/ms，表示每毫秒最多压缩多少个block。速率上限计算方法：BLOCK_LIMITS*1000*BLOCKSIZE，以默认值40为例，其速率上限为:40*1000*8KB=320000KB/s。
- ENABLE_META_COMPRESSION：是否开启header压缩，默认为0，取值范围为0（关闭）和1（开启）。用户可根据自己的实际情况来进行开启或关闭。
- SAMPLE_MIN：常量编码和等值编码采样步长最小值，默认为10，取值范围[1, 100]，支持小数输入，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。
- SAMPLE_MAX：常量编码和等值编码采样步长最大值，默认为10，取值范围[1, 100]，支持小数输入，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。
- CONST_PRIO：常量编码优先级，默认为40，取值范围[0, 100]，100表示关闭常量编码，支持小数输入，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。
- CONST_THRESHOLD：常量编码阈值，默认为90，取值范围[1, 100]，表示一列常量值的占比超过该阈值时进行常量编码，支持小数输入，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。
- EQVALUE_PRIO：等值编码优先级，默认为60，取值范围[0, 100]，100表示关闭等值编码，支持小数输入，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。
- EQVALUE_THRESHOLD：等值编码阈值，默认为80，取值范围[1, 100]，表示两列数据的等值比例超过该阈值时进行等值编码，支持小数输入，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。

版权所有 © 华为云计算技术有限公司

- ENABLE_DELTA_ENCODE_SWITCH：差值编码开关，默认为1，支持小数输入，0表示关闭，1表示开启，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。
- LZ4_COMPRESSION_LEVEL：lz4压缩等级，默认为0，取值范围[0, 16]，支持小数输入，小数会自动向下取整。用户可根据自己的实际情况来设置具体值。
- ENABLE_LZ4_PARTIAL_DECOMPRESSION：部分解压开关，默认为1，支持小数输入，0表示关闭，1表示开启，小数会自动向下取整。用户可根据自己的实际情况来进行开启或关闭。

示例分析：

```
EXECUTION_INTERVAL：15
JOB_SIZELIMIT：10240
WIND_DURATION：240
BLOCK_LIMITS：0
```

此配置下单表分区在一个维护窗口期间可完成240/15*10240MB=160GB数据的评估压缩。压缩带宽为100MB/秒，实际压缩仅耗时160GB/(100MB/秒)=27分钟。其他时间对业务无影响。用户可根据自己业务闲时可支配给压缩的时长来调节参数。

9.4.6 运维 TIPS

须知

高级压缩特性使用前必须先联系工作人员申请License，否则执行相关命令会报错。

1. 手动触发一次压缩（示例中一次压缩102400MB）。
   a. 给表加上冷热分离策略：

   ```
   gaussdb=#  DROP TABLE IF EXISTS ILM_TABLE;
   gaussdb=#  CREATE TABLE ILM_TABLE(a int);
   gaussdb=# ALTER TABLE ILM_TABLE ILM ADD POLICY ROW STORE COMPRESS ADVANCED
      ROW AFTER 3 MONTHS OF NO MODIFICATION;
   ```

   b. 手动触发压缩：

   ```
   DECLARE
    v_taskid number;
   BEGIN
    DBE_ILM_ADMIN.CUSTOMIZE_ILM(11, 1);
   DBE_ILM_ADMIN.CUSTOMIZE_ILM(13, 102400);
    DBE_ILM.EXECUTE_ILM(OWNER         => '$schema_name',
                        OBJECT_NAME     => 'ilm_table',
                        TASK_ID         => v_taskid,
                        SUBOBJECT_NAME => NULL,
                        POLICY_NAME     => 'ALL POLICIES',
                        EXECUTION_MODE => 2);
    RAISE INFO 'Task ID is:%', v_taskid;
   END;
   /
   ```

   c. 查看压缩JOB是否完成，可以看到具体的执行信息：

   ```
   gaussdb=# SELECT * FROM gs_adm_ilmresults ORDER BY task_id desc;

    task_id |                job_name                |     start_time      |   completion_time
   |                    statistics
   ---------+----------------------------------------+---------------------+---------------------
   +-----------------------------------------------------------
      17267 | ilmjob$_2 | 2023-03-29 08:11:25 | 2023-03-29 08:11:25 |
   SpaceSaving=453048,BoundTime=1680145883,LastBlkNum=128
   ```

2. 手动停止压缩。

   ```
   gaussdb=# DBE_ILM.STOP_ILM (task_id => V_TASK, p_drop_running_Jobs => FALSE, p_Jobname =>
   V_JOBNAME);
   ```


版权所有 © 华为云计算技术有限公司

表 9-2 DBE_ILM.STOP_ILM 输入参数

名称	描述
TASK_ID	指定待停止ADO task的描述符ID。
P_DROP_RUNNING_JOBS	是否停止正在执行中的任务，true为强制停止，false为不停止正在执行的任务。
P_JOBNAME	标识待停止的特定JobName，通过GS_MY_ILMEVALUATIONDETAILS视图可以查询。

3. 为表生成策略及后台调度压缩任务。
   a. 给表加上冷热分离策略：
   ```
   gaussdb=# DROP TABLE IF EXISTS ILM_TABLE;
   gaussdb=# CREATE TABLE ILM_TABLE(a int);
   gaussdb=# ALTER TABLE ILM_TABLE ILM ADD POLICY ROW STORE COMPRESS ADVANCED
       ROW AFTER 3 MONTHS OF NO MODIFICATION;
   ```
   b. 设置ILM执行相关参数：
   ```
   BEGIN
     DBE_ILM_ADMIN.CUSTOMIZE_ILM(11, 1);
     DBE_ILM_ADMIN.CUSTOMIZE_ILM(12, 10);
     DBE_ILM_ADMIN.CUSTOMIZE_ILM(1, 1);
     DBE_ILM_ADMIN.CUSTOMIZE_ILM(13, 512);
   END;
   /
   ```
   c. 开启后台的定时调度：
   ```
   gaussdb=# CALL DBE_ILM_ADMIN.DISABLE_ILM();
   gaussdb=# CALL DBE_ILM_ADMIN.ENABLE_ILM();
   ```
   d. 用户可以根据需要，调用DBE_SCHEDULER.set_attribute设置后台维护窗口的开启时间。当前默认22:00开启。
4. 设置ILM执行相关参数。

   控制ADO的条件单位是天还是秒，秒仅用来做测试用。取值为：ILM_POLICY_IN_SECONDS = 1或ILM_POLICY_IN_DAYS = 0（默认值）：

   ```
   gaussdb=# CALL DBE_ILM_ADMIN.CUSTOMIZE_ILM(11, 1);
   ```

   控制一次ADO Task最多生成多少个ADO Job。取值范围大于等于0小于等于2147483647的整数或浮点数，作用时向下取整：

   ```
   gaussdb=# CALL DBE_ILM_ADMIN.CUSTOMIZE_ILM(12, 10);
   ```

   ADO Task的执行频率，单位分钟，默认值15。取值范围大于等于1小于等于2147483647的整数或浮点数，作用时向下取整：

   ```
   gaussdb=# CALL DBE_ILM_ADMIN.CUSTOMIZE_ILM(1, 1);
   ```

   控制单个ADO Job可以处理的最大字节数，单位兆。取值范围大于等于0小于等于2147483647的整数或浮点数，作用时向下取整：

   ```
   gaussdb=# CALL DBE_ILM_ADMIN.CUSTOMIZE_ILM(13, 512);
   ```

5. 评估一张表是否适合压缩及评估压缩后带来多少收益。
   ```
   DBE_COMPRESSION.GET_COMPRESSION_RATIO(
   SCRATCHTBSNAME IN VARCHAR2,
   OWNNAME        IN VARCHAR2,
   OBJNAME        IN VARCHAR2,
   SUBOBJNAME     IN VARCHAR2,
   COMPTYPE       IN NUMBER,
   BLKCNT_CMP     OUT INTEGER,
   BLKCNT_UNCMP   OUT INTEGER,
   ROW_CMP        OUT INTEGER,
   ROW_UNCMP      OUT INTEGER,
   CMP_RATIO      OUT NUMBER,
   ```

 版权所有 © 华为云计算技术有限公司

```
COMPTYPE_STR   OUT VARCHAR2,
SAMPLE_RATIO   IN NUMBER DEFAULT 20,
OBJTYPE        IN INTEGER DEFAULT 1)
```

表 9-3 DBE_COMPRESSION.GET_COMPRESSION_RATIO 输入参数

名称	描述
SCRATCHTBSNAME	数据对象所属表空间。
OWNNAME	数据对象所有者（所属模式）。
OBJNAME	数据对象名称。
SUBOBJNAME	数据子对象名称。
COMPTYPE	● 1：未压缩 ● 2：高级压缩
SAMPLE_RATIO	采样比例，输入为0-100的整数或浮点数，对应为百分之N的采样比例。默认为20，即对20%的行数进行采样。
OBJTYPE	对象类型，支持： 1：表对象

表 9-4 DBE_COMPRESSION.GET_COMPRESSION_RATIO 输出参数

名称	描述
BLKCNT_CMP	样本被压缩后占用的块数。
BLKCNT_UNCMP	样本未压缩占用的块数。
ROW_CMP	样本被压缩后单个块内可容纳的行数。
ROW_UNCMP	样本未被压缩时单个数据块可容纳的行数。
CMP_RATIO	压缩比，blkcnt_uncmp除以blkcnt_cmp。
COMPTYPE_STR	描述压缩类型的字符串。

示例：

```
gaussdb=# ALTER DATABASE set ilm = on;
gaussdb=# CREATE user user1 IDENTIFIED BY '********';
gaussdb=# CREATE user user2 IDENTIFIED BY '********';
gaussdb=# SET ROLE user1 PASSWORD '********';
gaussdb=# CREATE TABLE TEST_DATA (ORDER_ID INT, GOODS_NAME TEXT, CREATE_TIME
TIMESTAMP)
  ILM ADD POLICY ROW STORE COMPRESS ADVANCED ROW AFTER 1 DAYS OF NO MODIFICATION;
INSERT INTO TEST_DATA VALUES (1, '零食大礼包A', NOW());

DECLARE
o_blkcnt_cmp      integer;
o_blkcnt_uncmp    integer;
o_row_cmp         integer;
```

 版权所有 © 华为云计算技术有限公司

```
o_row_uncmp       integer;
o_cmp_ratio       number;
o_comptype_str    varchar2;
begin
dbe_compression.get_compression_ratio(
   SCRATCHTBSNAME  =>  NULL,
   OWNNAME         =>  'user1',
   OBJNAME         =>  'test_data',
   SUBOBJNAME      =>  NULL,
   COMPTYPE        =>  2,
   BLKCNT_CMP      =>  o_blkcnt_cmp,
   BLKCNT_UNCMP    =>  o_blkcnt_uncmp,
   ROW_CMP         =>  o_row_cmp,
   ROW_UNCMP       =>  o_row_uncmp,
   CMP_RATIO       =>  o_cmp_ratio,
   COMPTYPE_STR    =>  o_comptype_str,
   SAMPLE_RATIO    =>  100,
   OBJTYPE         =>  1);
RAISE INFO 'Number of blocks used by the compressed sample of the object        : %', o_blkcnt_cmp;
RAISE INFO 'Number of blocks used by the uncompressed sample of the object      : %',
o_blkcnt_uncmp;
RAISE INFO 'Number of rows in a block in compressed sample of the object        : %', o_row_cmp;
RAISE INFO 'Number of rows in a block in uncompressed sample of the object      : %', o_row_uncmp;
RAISE INFO 'Estimated Compression Ratio of Sample                               : %', o_cmp_ratio;
RAISE INFO 'Compression Type                                                    : %', o_comptype_str;
end;
/
INFO:  Number of blocks used by the compressed sample of the object       : 0
INFO:  Number of blocks used by the uncompressed sample of the object     : 0
INFO:  Number of rows in a block in compressed sample of the object       : 0
INFO:  Number of rows in a block in uncompressed sample of the object     : 0
INFO:  Estimated Compression Ratio of Sample                              : 1
INFO:  Compression Type                                                   : Compress Advanced
```

6. 查询每一行的最后修改时间。

```
DBE_HEAT_MAP.ROW_HEAT_MAP(
OWNER           IN VARCHAR2,
SEGMENT_NAME    IN VARCHAR2,
PARTITION_NAME  IN VARCHAR2     DEFAULT NULL,
CTID            IN TEXT,
V_DEBUG         IN BOOL         DEFAULT FALSE);
```

表 9-5 DBE_HEAT_MAP.ROW_HEAT_MAP 输入参数

名称	描述
OWNER	数据对象所属Schema。
SEGMENT_NAME	数据对象名称。
PARTITION_NAME	数据对象分区名，可选参数，默认为 NULL。
CTID	目标行的ctid，即block_id或row_id。
V_DEBUG	debug调试，增加日志打印。

表 9-6 DBE_HEAT_MAP.ROW_HEAT_MAP 输出参数

名称	描述
OWNER	数据对象的所有者。
SEGMENT_NAME	数据对象名称。

 版权所有 © 华为云计算技术有限公司

名称	描述
PARTITION_NAME	数据对象分区名称，可选参数。
TABLESPACE_NAME	数据所属的表空间名称。
FILE_ID	行所属的绝对文件ID。
RELATIVE_FNO	行所属的相对文件ID（GaussDB中无此逻辑，因此取值同上）。
CTID	行的ctid，即block_id或row_id。
WRITETIME	行的最后修改时间。

示例：

```
gaussdb=# ALTER DATABASE set ilm = on;
gaussdb=# CREATE Schema HEAT_MAP_DATA;
gaussdb=# SET current_schema=HEAT_MAP_DATA;

gaussdb=# CREATE TABLESPACE example1 RELATIVE LOCATION 'tablespace1';
gaussdb=# CREATE TABLE HEAT_MAP_DATA.heat_map_table(id INT, value TEXT) TABLESPACE
example1;
gaussdb=# INSERT INTO HEAT_MAP_DATA.heat_map_table VALUES (1, 'test_data_row_1');

gaussdb=# SELECT * from DBE_HEAT_MAP.ROW_HEAT_MAP(
    owner          =>  'heat_map_data',
    segment_name  =>  'heat_map_table',
    partition_name  => NULL,
    ctid           =>  '(0,1)');
     owner     |  segment_name  | partition_name | tablespace_name | file_id | relative_fno | ctid  |
writetime
---------------+----------------+----------------+-----------------+---------+--------------+-------+-----------
 heat_map_data | heat_map_table |                | example1        |   17291 |        17291 | (0,1) |
(1 row)
```

7. 查询ILM调度与执行的相关环境参数。

```
gaussdb=# SELECT * FROM GS_ADM_ILMPARAMETERS;
              name               | value
---------------------------------+-------
 EXECUTION_INTERVAL              |    15
 RETENTION_TIME                  |    30
 ENABLED                         |     1
 POLICY_TIME                     |     0
 ABS_JOBLIMIT                    |    10
 JOB_SIZELIMIT                   |  1024
 WIND_DURATION                   |   240
 BLOCK_LIMITS                    |    40
 ENABLE_META_COMPRESSION         |     0
 SAMPLE_MIN                      |    10
 SAMPLE_MAX                      |    10
 CONST_PRIO                      |    40
 CONST_THRESHOLD                 |    90
 EQVALUE_PRIO                    |    60
 EQVALUE_THRESHOLD               |    80
 ENABLE_DELTA_ENCODE_SWITCH      |     1
 LZ4_COMPRESSION_LEVEL           |     0
 ENABLE_LZ4_PARTIAL_DECOMPRESSION |     1
(18 rows)
```

8. 查询ILM策略的概要信息，包含策略名称、类型、启用禁用状态、删除状态。

```
gaussdb=# SELECT * FROM GS_ADM_ILMPOLICIES;
policy_name |  policy_type  | tablespace | enabled | deleted
-------------+---------------+------------+---------+---------
```

 版权所有 © 华为云计算技术有限公司

```
 p1          | DATA MOVEMENT |            | YES     | NO


gaussdb=# SELECT * FROM GS_MY_ILMPOLICIES;
 policy_name |  policy_type  | tablespace | enabled | deleted
-------------+---------------+------------+---------+---------
 p1          | DATA MOVEMENT |            | YES     | NO
```

9. 查询ILM策略的数据移动概要信息，包含策略名称、动作类型、条件等。

```
gaussdb=# SELECT * FROM GS_ADM_ILMDATAMOVEMENTPOLICIES;
 policy_name | action_type | scope | compression_level | tier_tablespace | tier_status |
condition_type     | condition_days | custom_function | policy_subtype | action_clause | tier_to
-------------+-------------+-------+-------------------+-----------------+-------------+-----------------------
+----------------+-----------------+----------------+---------------+---------
 p1          | COMPRESSION | ROW   | ADVANCED          |                 |             | LAST MODIFICATION
TIME |             90 |                 |                |               |

gaussdb=# SELECT * FROM GS_MY_ILMDATAMOVEMENTPOLICIES;
 policy_name | action_type | scope | compression_level | tier_tablespace | tier_status |
condition_type     | condition_days | custom_function | policy_subtype | action_clause | tier_to
-------------+-------------+-------+-------------------+-----------------+-------------+-----------------------
+----------------+-----------------+----------------+---------------+---------
 p1          | COMPRESSION | ROW   | ADVANCED          |                 |             | LAST MODIFICATION
TIME |             90 |                 |                |               |
(1 row)
```

10. 查询所有存在ILM策略应用的数据对象与相应策略的概要信息，包含策略名称、数据对象名称、策略的来源、策略的启用删除状态。

```
gaussdb=# SELECT * FROM  GS_ADM_ILMOBJECTS;
 policy_name | object_owner | object_name | subobject_name | object_type |    inherited_from    |
tbs_inherited_from | enabled | deleted
-------------+--------------+-------------+----------------+-------------+----------------------
+--------------------+---------+---------
 p1          | public       | lineitem    |                | TABLE       | POLICY NOT INHERITED |                    |
YES     | NO

gaussdb=# SELECT * FROM  GS_MY_ILMOBJECTS;
 policy_name | object_owner | object_name | subobject_name | object_type |    inherited_from    |
tbs_inherited_from | enabled | deleted
-------------+--------------+-------------+----------------+-------------+----------------------
+--------------------+---------+---------
 p1          | public       | lineitem    |                | TABLE       | POLICY NOT INHERITED |                    |
YES     | NO
```

11. 查询ADO Task的概要信息，包含Task ID，Task Owner，状态以及时间信息。

```
gaussdb=# SELECT * FROM GS_ADM_ILMTASKS;
 task_id | task_owner |   state   |         creation_time         |          start_time           |
completion_time
---------+------------+-----------+-------------------------------+-------------------------------
+-------------------------------
       1 | omm        | COMPLETED | 2023-10-16 12:03:55.113296+08 | 2023-10-16 12:03:55.113296+08 |
2023-10-16 12:03:56.326864+08
(1 row)

gaussdb=# SELECT * FROM GS_MY_ILMTASKS;
 task_id | task_owner |   state   |         creation_time         |          start_time           |
completion_time
---------+------------+-----------+-------------------------------+-------------------------------
+-------------------------------
       1 | omm        | COMPLETED | 2023-10-16 12:03:55.113296+08 | 2023-10-16 12:03:55.113296+08 |
2023-10-16 12:03:56.326864+08
(1 row)
```

12. 查询ADO Task的评估详情信息，包含Task ID，策略信息、对象信息、评估结果以及ADO JOB名称。

```
gaussdb=# SELECT * FROM GS_ADM_ILMEVALUATIONDETAILS;
 task_id | policy_name | object_owner | object_name | subobject_name | object_type |
selected_for_execution |      job_name       | comments

---------+-------------+--------------+-------------+----------------+-------------+------------------------
```

 版权所有 © 华为云计算技术有限公司

```
+-------------------+---------
-
      1 | p2          | public       | ilm_table_1 |               | TABLE       | SELECTED FOR EXECUTION | ilmjob
$_postgres1 |
(1 row)

gaussdb=# SELECT * FROM GS_MY_ILMEVALUATIONDETAILS;
 task_id | policy_name | object_owner | object_name | subobject_name | object_type |
selected_for_execution |     job_name       | comments

---------+-------------+--------------+-------------+----------------+-------------+------------------------
+-------------------+---------
-
      1 | p2          | public       | ilm_table_1 |               | TABLE       | SELECTED FOR EXECUTION | ilmjob
$_postgres1 |
(1 row)
```

13. 查询ADO JOB的执行详情信息，包含Task ID，JOB名称、JOB状态、JOB时间信息等。

```
gaussdb=# SELECT * FROM GS_ADM_ILMRESULTS;
 task_id |     job_name      |       job_state        |          start_time           |       completion_time        |
comments |                  statistics
---------+-------------------+------------------------+-------------------------------
+-------------------------------+----------+-----------------------------------------------------
      1 | ilmjob$_postgres1 | COMPLETED SUCCESSFULLY | 2023-10-16 12:03:56.290176+08 |
2023-10-16 12:03:56.319829+08 |          | SpaceSaving=0,BoundTime=1697429033,LastBlkNum=40
(1 row)

gaussdb=# SELECT * FROM GS_MY_ILMRESULTS;
 task_id | job_name | job_state | start_time | completion_time | comments | statistics
---------+----------+-----------+------------+-----------------+----------+------------
(0 rows)
 task_id |     job_name      |       job_state        |          start_time           |       completion_time        |
comments |                  statistics
---------+-------------------+------------------------+-------------------------------
+-------------------------------+----------+-----------------------------------------------------
      1 | ilmjob$_postgres1 | COMPLETED SUCCESSFULLY | 2023-10-16 12:03:56.290176+08 |
2023-10-16 12:03:56.319829+08 |          | SpaceSaving=0,BoundTime=1697429033,LastBlkNum=40
(1 row)
```

 版权所有 © 华为云计算技术有限公司

10 内存管理

Linux 内存机制

- 物理内存

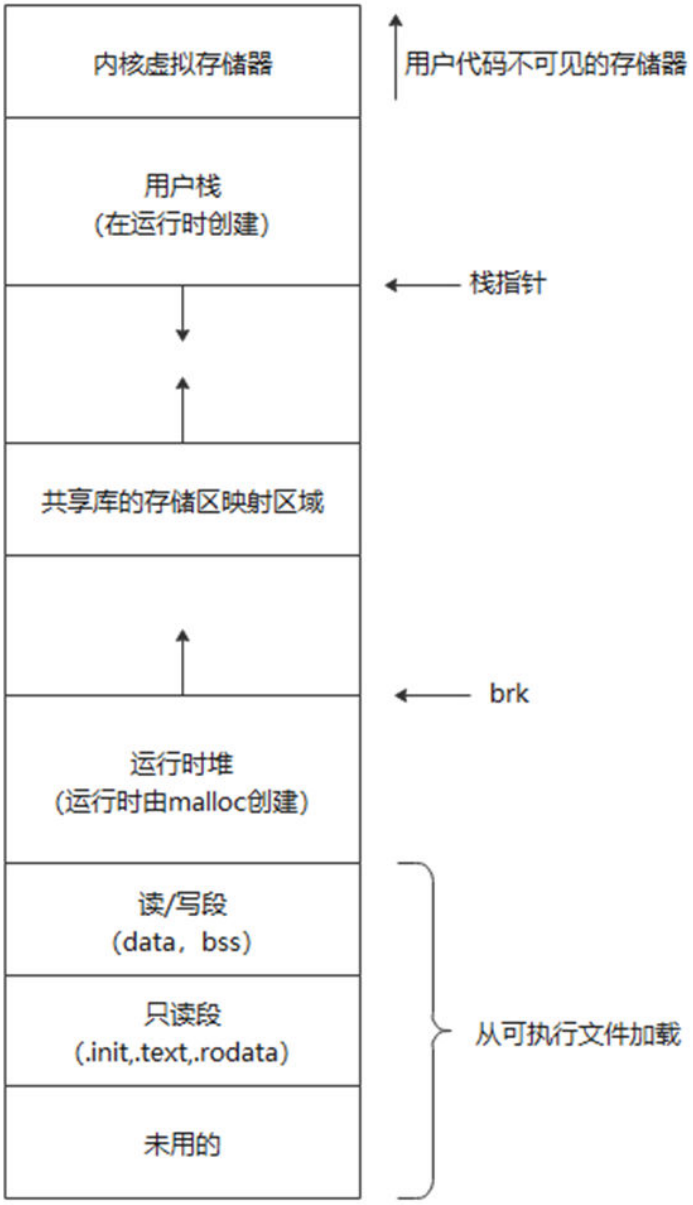
  物理可见的硬件，即内存条。

- 虚拟内存

  进程运行时所有内存空间的总和，以32位系统为例，每个进程有4GB的内存空间，各进程的内存空间具有类似的结构，如图10-1所示：

 版权所有 © 华为云计算技术有限公司

图 10-1 Linux 内存机制



  - 一个新进程建立的时候，将会建立起自己的内存空间，并将此进程的数据、代码等从磁盘复制到自己的内存空间。进程中每一条数据的地址都由进程控制表中的task_struct记录。task_struct包含一张链表，用来记录内存空间的分配情况，哪些地址有数据、哪些地址无数据、哪些可读、哪些可写都可以通过这个链表记录。
  - 每个进程已经分配的内存空间，都与对应的磁盘空间存在映射关系。

    说明

    建议不要对进程虚拟内存大小进行限制。

- 物理内存地址和虚拟内存地址
  - 内存分页机制

 版权所有 © 华为云计算技术有限公司

        - 虚拟内存中的页（page）和物理内存页帧（page frame）相同大小。
        - 页表（page table）维护虚拟内存页到物理内存页的映射。
        - 页面失效（page fault）功能：操作系统找到一个最少使用的页帧，让它失效，并把它写入磁盘，随后把需要访问的页放到页帧中，并修改页表中的映射。
      - 虚拟内存地址：页号（与页表中的页号关联）和偏移量（页的大小）组成。
      - 虚存寻址：页表中找到页帧号，若失效则调入页，页帧号及偏移量传给内存管理单元，组成一个物理存在的地址，即可访问物理内存数据。
  - Buffer
      - 根据磁盘的读写设计的，把分散的写操作集中进行，减少磁盘碎片和硬盘的反复寻道，从而提高系统性能。
      - Linux有一个守护进程定期清空缓冲内容（即写入磁盘），也可以通过sync命令手动清空。
  - Cache
      - 将读取过的数据保存起来，重新读取时若命中（找到需要的数据）则不读取硬盘，若未命中，则读取硬盘。其中的数据会根据读取频率进行组织，把读取最频繁的内容放在最容易找到的位置，把不再读的内容不断往后排，直至从中删除。
  - OOM
      - 内存溢出（Out Of Memory，简称 OOM）是指应用系统中存在无法回收的内存或使用的内存过多，最终使得程序运行要用到的内存大于能提供的最大内存。此时程序无法运行，系统会提示内存溢出，有时会自动关闭软件，通过重启机器或软件释放掉一部分内存后又可以正常运行该软件，而由系统配置、数据流、用户代码等原因导致的内存溢出错误，即使用户重新执行任务依然无法避免。

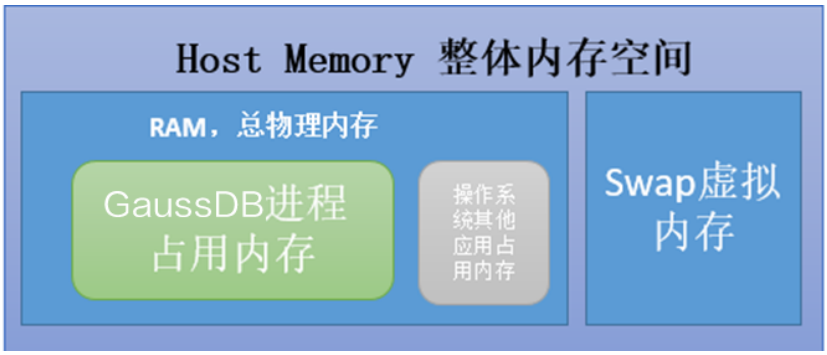
GaussDB 内存资源管理

- 内存管理简介
    - GaussDB的动态内存使用方法均基于内存上下文管理，在内存上下文的机制上，引入了逻辑内存管理机制，同时提供多项视图来追踪内存使用情况。

      当前GaussDB的内存管理分为两级，分别是：
        - 内存节点级别控制：通过max_process_memory参数限制CN、DN上可以使用的内存上限。
        - 内存作业级别控制：限制单条query语句最大可使用的内存上限。
- 节点级别内存控制
    - 在Linux操作系统中，内存空间通常由物理内存RAM和虚拟内存Swap组成。其中，物理内存由数据库程序和非数据库程序共同使用；而数据库程序的内存空间由GaussDB进程共同使用。如**图10-2**所示：

 版权所有 © 华为云计算技术有限公司

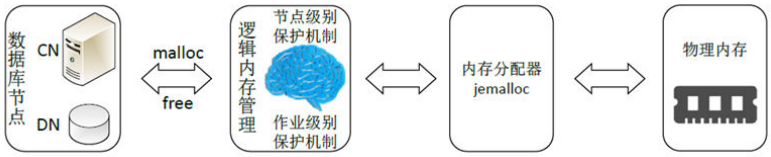
图 10-2 Host Memory 整体内存空间



  - 由于非数据库程序也需要使用内存，一般建议GaussDB占系统可用内存的80%。

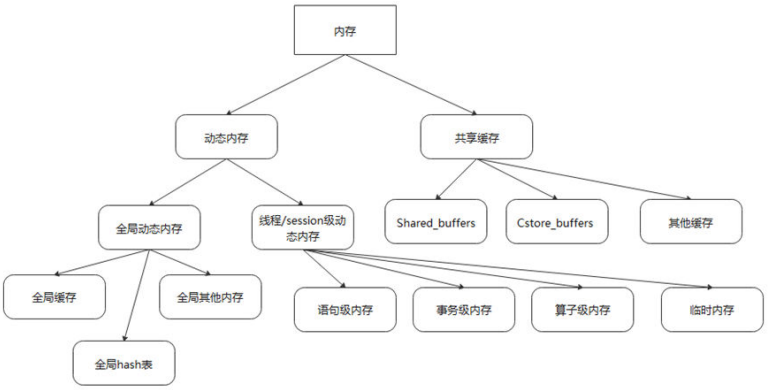
- 作业级别内存控制
  - 提供query_mem参数，根据能使用的内存量和能同时运行的作业数，指定每个作业执行时能够使用的内存量；提供query_max_mem参数，指定每个作业执行时能够使用的内存上限。
  - 提供work_mem参数，设置内部排序操作和Hash表在开始写入临时磁盘文件之前使用的内存大小；提供maintenance_work_mem参数，设置在维护性操作（比如VACUUM、CREATE INDEX、ALTER TABLE ADD FOREIGN KEY等）中可使用的最大内存。
- 逻辑内存管理机制
  - GaussDB内存管理是通过逻辑内存管理来实现的。其实现原理是在原始的内存分配之上增加一层逻辑内存管理，通过检查已分配内存是否超过规定的内存来决定是否为作业分配内存。逻辑内存管理没有更改原有的内存资源分配机制，仅在分配内存之前增加一个逻辑判断层，查看是否达到允许使用的内存上限，来决定是否分配内存。详情如图10-3所示：

图 10-3 逻辑内存管理机制



- 内存分类

图 10-4 内存分类



 版权所有 © 华为云计算技术有限公司

GaussDB的内存使用分类如上所述，其每种类型的说明如下：

- 动态内存
  - 全局动态内存
    - 全局缓存：全局缓存（Cache）用来提高访问效率，Cache中包括一个系统表元组Cache和一个表模式信息Cache（RelCache），SysCache中存放的是最近使用过的系统表的元组，而RelCache中包含所有最近访问过的表的模式信息（包含系统表的信息）。RelCache中存放的不是元组，而是RelationData数据结构，每一个RelationData结构表示一个表的模式信息，这些信息都由系统表元组中的信息构造而来。
    - 全局hash表：保存全局的统计信息或其他数据的内存hash结构。
    - 全局其他内存：保存全局变量等内容。
  - 线程/session级动态内存
    - 语句级内存：生命周期为整个SQL语句执行阶段，语句执行前内存被申请，执行结束后内存被回收释放。
    - 事务级内存：生命周期为整个事务内，事务开始前内存被申请，事务结束后内存被回收释放。
    - 算子级内存：生命周期为单个算子有效期内，算子执行结束后内存被释放回收。
    - 临时内存：一些临时变量申请的内存，使用后及时释放。
- 共享缓存
  - Shared_buffers

    缓存从磁盘读取的数据所使用的内存。
  - 其他缓存

    线程所使用的槽位、锁等资源占用的内存。

- 内存上下文机制
  - 内存上下文介绍
    - 概念：内存上下文用来统一控制内存的申请和释放，内存上下文中记录了其所申请的总内存大小，每一处内存申请的大小，申请内存所在的文件、行号等信息以及与其他内存上下文之间的关联信息。
    - 目的：基于数据库中作业执行逻辑，通过内存上下文管理机制，提升查询内存分配、释放效率。
  - 逻辑内存管理
    - 通过数据库进程内部记账，控制单进程内存使用的上限。
    - 可统计进程总体使用、共享和非共享内存上下文使用及重点模块内存使用（通信库、pooler等）。
    - 逻辑内存的生效机制：enable_memory_limit设置为on，并且max_process_memory减掉缓存等其他预留内存 > 2GB，具体详情请联系管理员。

 版权所有 © 华为云计算技术有限公司

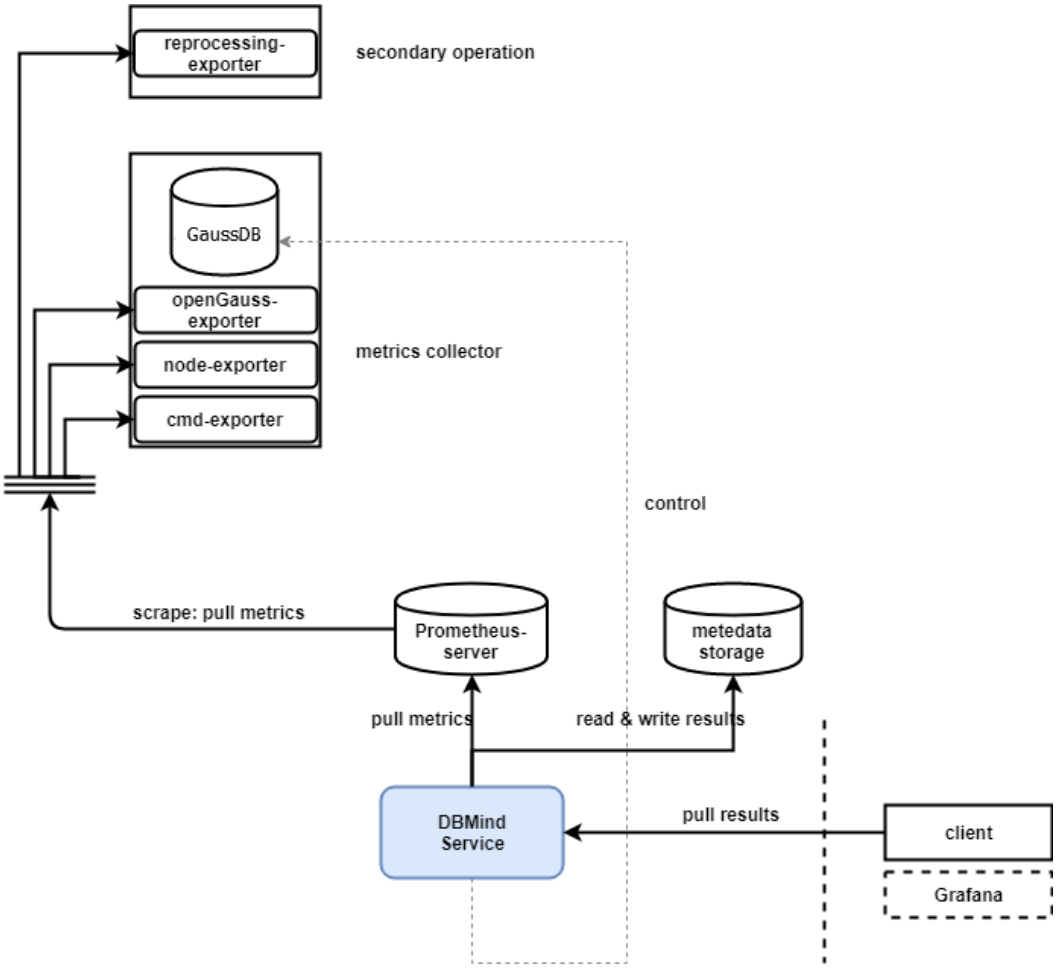
- PV_TOTAL_MEMORY_DETAIL视图用于查询数据库进程的内存使用情况，便于分析内存失败后问题，具体字段可参考《开发指南》中“系统表和系统视图 > 系统视图 > PV_TOTAL_MEMORY_DETAIL”章节。

 版权所有 © 华为云计算技术有限公司

11 DBMind: 数据库自治运维

DBMind承载了GaussDB的AI for Database特性，主要用于对数据库进行自治运维和管理，从而帮助数据库运维人员减少运维工作量。在实现上，DBMind具有监控和服务化的性质，同时也提供即时AI工具包和提供开箱即用的AI自治功能（如：索引推荐）。DBMind的监控平台以开源Prometheus为主，并提供了数据生产者exporter，可与Prometheus平台完成对接。DBMind服务架构如图11-1所示：

图 11-1 DBMind 服务架构



版权所有 © 华为云计算技术有限公司

图中各关键组件说明：

- DBMind Service：DBMind后台服务，可用于定期离线计算，包括慢SQL根因分析、时序预测等；
- Prometheus-server：存储Prometheus监控指标的服务器；
- metadatabase：DBMind在离线计算结束后，将计算结果存储在此处，支持GaussDB、SQLite等数据库；
- client：用户读取DBMind离线计算结果的客户端，目前该客户端仅支持命令行操作；若采用GaussDB等数据库存储DBMind的计算结果，则用户可以自行配置Grafana等可视化工具，用于对该结果进行可视化；
- opengauss-exporter：用户从GaussDB数据库节点上采集监控指标，供DBMind服务进行计算；
- node-exporter：Prometheus官方提供的exporter，可用于监控对应节点的系统指标，如CPU和内存使用情况；
- cmd-exporter：在用户安装数据库的环境上执行命令行，并采集该命令行的执行结果，同时，也尝试将数据库日志内容转化为监控指标；例如通过执行cm_ctl命令，查看数据库集群的状态；
- reprocessing-exporter：用于对Prometheus采集到的指标进行二次加工处理，例如计算CPU使用率等。

DBMind 支持的场景规格

- DBMind目前不支持在元数据库变更的情况下保留原始诊断数据。
- DBMind不支持纳管容灾集群。
- DBMind不支持纳管一主一备一logger集群中的logger节点，否则无法获取相关节点的指标数据。
- DBMind支持纳管多集群，但是随着纳管集群节点越多，消耗的资源也增多。需要在DBMind消耗的资源过多时，停止纳管更多节点，以避免CPU和内存资源不足。
- DBMind实例纳管建议为1个CPU核对应3个节点（例如3个节点（包含主备）的集中式则需要1个CPU核，9个节点（包含CN、DN等）的分布式需要3个CPU核），如果纳管的实例节点数太多，会导致DBMind服务接口超时或服务异常。
- DBMind只支持文档中写明的场景，不支持文档中没有明确的场景、示例中不包括的部署形态、使用方法等超规格使用形式的情况。

版权所有 © 华为云计算技术有限公司

说明

云侧网络情况和DBMind云上部署说明：

- 云上的实例节点拥有多层网络结构，分布式节点拥有三层结构：管理层（CM）、数据层（CN）和数据同步层（DN）。
- 对DBMind主程序部署情况的说明如下：
  - 启动prometheus，监听数据层ip。
  - 启动reprocessing-exporter，监听数据层ip。
  - 启动DBMind，在管理层ip暴露webservice（之所以使用管理层ip，是因为接口调用需要走管理面发起，需要保证管理面网络互通）。
- 对于被纳管的数据库实例，插件部署情况如下：
  - node-exporter、cmd-exporter、opengauss-exporter都监听数据层ip。
  - 对于仲裁、冷备、offline的节点，只会部署node-exporter、cmd-exporter。
  - 分布式如果节点上无CN组件，只会部署node-exporter、cmd-exporter。
  - 其他情况均会部署node-exporter、cmd-exporter、opengauss-exporter。
- DBMind指标采集主要为本身功能服务，在采集过程中可能存在网络故障、采集超时等异常，另外prometheus本身采集机制为通过标签区分不同指标，因此不能保证采集指标的连续性，如果用户需要将指标用于前端展示，在指标不连续的情况下需要自己补齐。

环境配置

DBMind外置AI功能需要运行在Python3.7版本以上，当前DBMind独立出包并且包中已包含工具所有三方依赖项，因此用户不用单独安装。如果用户需要自己配置依赖项，DBMind需要的第三方依赖包记录在AI功能根目录的requirements.txt文件中（包括requirements-x86.txt与requirements-arrch64.txt，用户可根据自己平台类型选择），可以通过pip install命令安装依赖，命令如下：

```
pip3 install -r requirements-x86.txt
```

如果用户没有安装齐全所需的依赖，则当用户执行gs_dbmind命令时，会再次提醒用户安装第三方依赖。需要注意，该文件提供了DBMind所需的第三方依赖，若用户环境存在第三方包冲突等情况，可由用户根据实际情况进行处理。

如果pip版本过低，可能会出现装包失败的情况，此时可使用以下命令对pip进行升级：

```
pip3 install --upgrade pip
```

11.1 DBMind 模式说明

用户在设置环境变量后，可通过gs_dbmind命令调用DBMind下列基本功能：

- 服务功能：service子命令，包括创建并初始化配置目录、启动后台服务、关闭后台服务等。
- 调用组件：component子命令（如索引推荐、参数调优等）可通过该模式进行调用。
- 设置参数：set子命令，通过该命令，可以一键修改配置目录中的配置文件值；用户也可以通过文本编辑器进行手动修改。

例如，可以通过以下方法设置环境变量：

```
echo PATH=`pwd`/openGauss-DBMind:'$PATH' >> ~/.bashrc
echo 'export PATH' >> ~/.bashrc
```


版权所有 © 华为云计算技术有限公司

```
source ~/.bashrc
```

用户可以通过 --help 选项获得上述模式的帮助信息，例如：

```
gs_dbmind --help
usage: [-h] [-v] {service,set,component,upgrade} ...

openGauss DBMind: An autonomous platform for openGauss

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

available subcommands:
  {service,set,component,upgrade}
                        type '<subcommand> -h' for help on a specific subcommand
    service             send a command to DBMind to change the status of the service
    set                 set a parameter
    component           pass command line arguments to each sub-component.
    upgrade             update metadatabase and config files for the new dbmind
```

表 11-1 gs_dbmind 选项基本说明

参数	参数说明	取值范围
-h, --help	帮助命令。	-
-v, --version	版本号。	-
service	服务功能相关的子命令。	-
component	调用组件的子命令。	-
set	修改配置文件的子命令。	-
upgrade	用于升降级DBMind配置文件和元数据库。	-

11.1.1 service 子命令

该子命令可用于对配置目录进行初始化，同时也可以实现启动和停止后台任务。

配置目录初始化

用户可通过“gs_dbmind service setup”子命令进行配置目录的初始化。该配置文件中可包括DBMind的配置文件、日志等内容。该目录中的部分文件说明：

- dbmind.conf：DBMind的参数配置文件，用户可通过“gs_dbmind set”命令进行修改，也可通过文本编辑器进行手动修改。
- dynamic_config.db：DBMind服务保存在本地节点的元信息，主要包括算法的超参数、监控阈值等；该文件为DBMind服务元信息，不可由用户直接配置。
- metric_map.conf：监控指标映射表，可用于适配到不同采集平台中。例如，在DBMind中，监控到的系统cpu使用率名为os_cpu_usage，而用户自行实现的指标采集工具将cpu使用率命名为my_cpu_usage_rate。在该种情况下，如果想要DBMind中代表cpu使用率的指标名为my_cpu_usage_rate，则需要修改该配置选项。即添加“os_cpu_usage = my_cpu_usage_rate”配置项进行映射。对于普通用户，建议直接使用DBMind配套的采集组件和方案，则不涉及修改该配置文件。

版权所有 © 华为云计算技术有限公司

- metric_value_range.conf：指定某些具体监控指标的上下限，以防止对该指标进行预测时，超过合理区间；未在该列表中的指标范围默认为[0, +∞)。
- dbmind.pid：该文件保存正在使用该DBMind配置目录的进程PID。
- dbmind.children.pid：该文件保存DBMind子进程的PID信息，PID之间用逗号隔开。
- VERSION：生成该配置文件目录的DBMind版本，通过验证版本来避免版本不兼容情况下的使用。
- logs：该目录用于存储DBMind服务产生的日志，请不要手动修改该目录下的日志文件，否则可能会导致日志变为不可写状态，无法继续更新DBMind操作记录。
- backtrace.stack：DBMind自身用于定位调试产生的临时文件，包含了当前各个线程的调用栈信息，是通过接收到SIGUSR2信号而产生的。

用户可通过两种方式进行配置目录的初始化，一种为交互式，另一种为非交互式。例如，待初始化的配置目录名为confpath，则分别通过下述方法进行配置：

交互式模式

```
gs_dbmind service setup -c confpath --interactive
```

执行完毕上述命令后，用户可通过命令行终端对配置项进行交互式配置。

非交互式模式

非交互式模式总共分为三个步骤，即创建配置文件、修改配置项、初始化配置。其中第二个步骤需要用户通过文本编辑器手动编辑配置文件。具体步骤如下：

步骤1 创建配置文件，执行下述命令：

```
gs_dbmind service setup -c confpath
```

步骤2 执行完上述命令后，会在confpath目录下生成dbmind.conf配置文件，用户需要利用文本编辑器进行手动修改。相关参数的说明如下：

```
# TSDB部分用于配置时序数据库的连接信息。
# 当前仅支持Prometheus，配置时将name字段编辑为prometheus即可。
[TSDB]
name = prometheus # 时序数据库类型，选项：prometheus。
host = # 时序数据库地址。
port = # 时序数据库端口。
username = (null) # 时序数据库用户名，如果时序数据库并未配置用户，可以为空。
password = (null) # 时序数据库用户名，如果时序数据库并未配置用户，可以为空。
ssl_certfile = (null) # SSL连接的证书文件路径，可选。
ssl_keyfile = (null) # SSL连接的私钥文件路径，可选。
ssl_keyfile_password = (null) # SSL私钥文件的密码，可选。
ssl_ca_file = (null)  # 用于验证请求的CA证书文件路径，可选。

# METADATABASE部分用于指定DBMind生成的分析结果的存储位置。
# 当前支持的数据库类型有SQLite和GaussDB Kernel。如果使用GaussDB数据库，使用时需要注意Python驱动psycopg2的兼容性问题，用户可以选择使用GaussDB官方提供的驱动，也可以通过自行编译或修改GUC参数进行适配。
# 其他信息为连接到该数据库的连接信息，用户需要有数据库创建权限。
[METADATABASE]
dbtype = sqlite # 元数据库类型，选项：sqlite、opengauss、postgresql。
host = # 元数据库地址，如果配置多个元数据库地址，需要用逗号（,）分隔。数据库类型为sqlite时，无需配置。
port = # 元数据库端口，如果配置多个元数据库地址，且port相同，填写一个port即可。如果port不相同，port数量需要和host数量一致，用逗号（,）分隔。数据库为sqlite时，无需配置。
username = # 元数据库用户名，数据库为sqlite时，无需配置。
password = (null) # 元数据库用户密码，数据库为sqlite时，无需配置。
database = # 元数据库名。

# WORKER用于指定DBMind可以使用的worker子进程数量，如果写0则会进行自适应，即尽可能多地使用CPU资源。
```

 版权所有 © 华为云计算技术有限公司

```
[WORKER]
process_num = 0  # 本地节点上的工作进程数，小于或等于零表示自适应。

# AGENT部分用于指定DBMind连接到GaussDB Agent的信息。通过使用该Agent，可以让DBMind获取到被监控实例的即时状态，从而提高分析准确性。同时，也可以向数据库实例下发一些变更动作，如结束某条慢SQL语句（这取决于此处配置的用户是否有足够的权限）。
# 该master_url地址即为Agent的地址，由于opengauss-exporter承担了Agent的角色，该地址即opengauss-exporter的地址。
# opengauss-exporter支持HTTPS协议，可以根据配置指定SSL证书。
[AGENT]
master_url = # 不配置该参数，即可启动自动发现模式，此时所有在TSDB中注册的代理都会被找到，但是需要配置一套统一的用户名/密码/SSL连接信息。否则键入主节点的代理URL，例如，https://127.0.0.1:9187，如果有多个数据库实例，则需要用逗号（,）分隔它们。
username =  # 登录监控数据库的用户名。代理凭证。如果配置监控多个数据库实例，且这些数据库实例的用户名和密码不同，则需要输入多组配置用户名，各配置用户名之间用逗号（,）分隔，且需要与上述master_url选项一一对应。
password = (null) # 监控数据库的登录密码。代理凭证。如果配置监控多个数据库实例，需要用逗号（,）分隔。
ssl_certfile = (null) # SSL连接的证书文件路径，可选。如果配置监控多个数据库实例，需要用逗号（,）分隔。
ssl_keyfile = (null) # SSL连接的私钥文件路径，可选。如果配置监控多个数据库实例，需要用逗号（,）分隔。
ssl_keyfile_password = (null) # SSL私钥文件的密码，可选。如果配置监控多个数据库实例，需要用逗号（,）分隔。
ssl_ca_file = (null)  # 用于验证请求的CA证书文件路径，可选。如果配置监控多个数据库实例，需要用逗号（,）分隔。

[TIMED_TASK_LIST]
explanation = Configure scheduled tasks including start tasks and intervals # 配置计划任务，包括启动任务和间隔。
current_support_task_list = List of currently supported tasks:  # 当前支持的任务列表。
task1 = anomaly_detection          # 1. 指标异常检测
task2 = discard_expired_results    # 2. 清理过期数据
task3 = knob_recommend             # 3. 参数推荐
task4 = slow_query_killer          # 4. 慢SQL查杀
task5 = slow_query_diagnosis       # 5. 慢SQL诊断
task6 = cluster_diagnose           # 6. 集群故障诊断
task7 = agent_update_detect        # 7. 当检测到代理更新时自动更新异常检测器
task8 = calibrate_security_metrics # 8. 校准security_scenarios.yml提到的安全指标
task9 = check_security_metrics     # 9. 检查security_scenarios.yml提到的安全指标
task10 = update_statistics         # 10. 自动更新指定指标的统计信息

TIMED_TASK表示后台任务配置，通过修改参数启动或停止DBMind运行的后台任务以及任务的运行间隔，单位为秒。
运行间隔不能低于30秒，最大值为python3支持的整型数字上限，2的63次方减1。
[TIMED_TASK]
task = discard_expired_results, anomaly_detection, cluster_diagnose, agent_update_detect, update_statistics # 默认reload启动数据清理、异常检测、集群故障诊断、更新异常检测器、校准security_scenarios.yml指标、检查security_scenarios.yml指标和更新指定指标的统计信息任务
anomaly_detection_interval = 180  # 异常检测任务运行间隔，默认180秒。
slow_query_diagnosis_interval = 120  # 慢SQL诊断任务运行间隔，默认120秒。
knob_recommend_interval = 3600  # 参数推荐任务运行间隔，默认一小时。
slow_query_killer_interval = 30  # 慢SQL查杀任务运行间隔，默认30秒。
cluster_diagnose_interval = 30  # 集群故障诊断运行间隔，默认30秒。
discard_expired_results_interval = 3600  # 数据清理任务运行间隔，默认一小时。
agent_update_detect_interval = 30  # 检测agent更新并自动并自动更新异常检测器间隔，默认30秒。
calibrate_security_metrics_interval = 600 # 校准security_scenarios.yml安全指标间隔，默认10分钟。
check_security_metrics_interval = 600 # 检查security_scenarios.yml安全指标间隔，默认10分钟。
update_statistic_interval = 1800  # 自动更新指定指标统计信息间隔，默认30分钟。

# WEB-SERVICE用于DBMind的前台界面展示。
[WEB-SERVICE]
ssl = true # 默认使用安全的协议，用户需要提供证书路径，否则DBMind服务不能启动。
host = 127.0.0.1  # DBMind-service监听地址。
port = 8080  # DBMind-service监听端口。
ssl_certfile = (null) # SSL连接的证书文件路径，可选。ssl为true时，必填。
ssl_keyfile = (null) # SSL连接的私钥文件路径，可选。ssl为true时，必填。
ssl_keyfile_password = (null) # SSl私钥文件的密码，如果没有密码则忽略。
ssl_ca_file = (null)  # 用于验证请求的CA证书文件路径，可选。ssl为true时，必填。

# LOG表示设置DBMind的日志记录信息。
[LOG]
```

 版权所有 © 华为云计算技术有限公司

```
maxbytes = 10485760 # 默认值为10Mb。单个日志文件的最大大小。如果maxbytes为零，则文件无限增长。
backupcount = 1 # 日志文件备份数量。
level = INFO  # 日志级别，选项：DEBUG、INFO、WARNING、ERROR。
log_directory = logs # 日志文件存放的目录。

# 下列内容表示给用户进行交互配置时的提示信息，用户无需配置。
[COMMENT]
worker = The form of executing compute-intensive tasks. Tasks can be executed locally or distributed to multiple nodes for execution.
tsdb = Configure the data source for time series data, which come from monitoring the GaussDB instance.
metadatabase = Configure the database to record meta-data, which the database can store meta-data for the forecasting and diagnosis process. The database should be an GaussDB instance.
self-monitoring = Set up parameters for monitoring and diagnosing GaussDB instance.
self-optimization = Set up parameters for GaussDB optimization.

[IP_MAP]
ip_map = (null) # 对于数据IP和管理IP不一致的环境，需要提供对应关系。以下以一主两备数据库实例为例：“primary_data_ip:primary_management_ip,standby1_data_ip:standby1_management_ip,standby2_data_ip:standby2_management_ip”
```

步骤3 待用户手动修改完上述参数后，需要执行下述命令进行配置项的初始化。在该阶段中，DBMind会初步检查配置项的正确性、初始化用于存储结果数据的元数据库表结构和内容，同时也加密配置项中出现的密码。

```
gs_dbmind service setup --initialize -c confpath
```

步骤4 完成配置目录初始化过程，可基于该配置目录启动DBMind后台服务。

----结束

说明

1. 配置文件注释信息用于在交互模式下对用户进行提示，有特殊含义不要手动修改或删除；
2. 需要确保配置项的值与注释信息之间通过空格符分割，否则系统会将注释信息识别为配置项的值；
3. 配置项中的特殊字符，如果需要转义，则通过转义符“百分号”（%）来转义，例如，用户配置的密码为 "password%"，则应通过“百分号”进行转义，即 "password%%";
4. DBMind Service对外以标准RESTfulAPI的方式提供接口，其默认支持HTTPS通信，建议用户使用。
5. DBMind云侧部署主机目前最小规格8U64G，目前云管控支持容灾主集群纳管，不支持容灾备集群纳管。
7. DBMind在505.2及之后的版本支持IPv6部署。

启动服务

当用户完成配置目录的初始化后，可基于此配置目录启动DBMind后台服务。例如配置目录为confpath，则启动命令如下：

```
gs_dbmind service start -c confpath
```

当执行上述命令后，会提示服务已启动。在未指定任何附加参数时，该命令默认会启动所有的后台任务。如果用户只想启动某一个后台任务，需要添加“--only-run”选项。例如，用户只想启动慢SQL根因分析服务，则为：

```
gs_dbmind service start -c confpath --only-run slow_query_diagnosis
```

DBMind运行过程中如果想启或停止定时任务，首先将配置文件TIMED_TASK下task中相关功能删除，在执行以下命令，此时DBMind会刷新后台定时任务：

```
gs_dbmind service reload -c confpath
```

由于DBMind是一个在后台定期执行的服务，故如果用户只是想跑一次DBMind服务进行测试或调试，则可以通过添加“--dry-run”选项，如：

```
gs_dbmind service start -c confpath --dry-run
```

 版权所有 © 华为云计算技术有限公司

“--dry-run”选项可以与“--only-run”选项合用。

如果当前进程已经在跑，用户希望重启该DBMind服务，则可以使用重启命令，即：

```
gs_dbmind service restart -c confpath
```

默认情况下，restart会等待当前DBMind正在执行的任务执行完毕后才会退出并重启服务。如果用户希望强行终止当前正在运行的任务，则可以添加“-f”或“--force”选项，例如：

```
gs_dbmind service restart --force -c confpath
```

关闭服务

关闭服务与启动服务类似，其命令行结构更加简单，只需指定配置目录的地址即可。例如配置目录为confpath，则为：

```
gs_dbmind service stop -c confpath
```

DBMind服务会在后台执行完正在运行的任务后自行退出，如需要强行退出，则使用“--force”或“-f”选项。

DBMind 的高可用

⚠ 注意

DBMind高可用功能需要在未纳管实例的情况下实现，因此接口不需要token认证。应当使用默认的HTTPS协议以进行SSL双向认证，使用HTTP协议则可能引入攻击者直接发起未授权请求的安全风险。

为保证DBMind云侧使用时的高可靠，DBMind对外提供了服务状态查询和部分异常修复的接口，详细请参见**表11-29**。

- DBMind服务状态查询接口示例：

  ```
  curl -X 'POST'"http://127.0.0.1:8080/v1/api/check-status" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind service start -c dbmindconf "}'
  ```

  如果使用HTTPS协议，则查询示例为：

  ```
  curl -X 'POST'"https://127.0.0.1:8080/v1/api/check-status" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind service start -c dbmindconf "}' --cacert xx.crt --key xx.key --cert xx.crt
  ```

  返回结果示例：

  ```
  {"data":{"error_msg":"","result":{},"state":"NORMAL"},"success":true}
  ```

- DBMind异常修复接口示例：

  ```
  curl -X 'POST'"http://127.0.0.1:8080/v1/api/repair" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind service start -c dbmindconf "}'
  ```

  如果使用HTTPS协议，则修复示例为：

  ```
  curl -X 'POST'"https://127.0.0.1:8080/v1/api/repair" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind service start -c dbmindconf "}' --cacert xx.crt --key xx.key --cert xx.crt
  ```

  返回结果示例：

  ```
  {"data":{"error_msg":"","result":{},"state":"SUCCESS"},"success":true}
  ```

 版权所有 © 华为云计算技术有限公司

说明

当前DBMind服务支持查询的异常场景如下：

- PID文件丢失或文件内容异常：当用户启动服务/组件后，会自动生成“*.pid”文件记录进程对应的进程标识符，也就是PID。PID文件用于云管控判断进程状态，发现进程级别的异常并进行修复。为防止PID文件的误删除与内容的误修改，需要定期判断PID文件是否存在以及其内容是否正确。
- 日志文件丢失：当用户启动服务/组件后，会自动生成*.log文件来记录进程运行的日志。此日志文件用于快速定位问题的根源、追踪程序执行的过程等。为防止日志文件的误删除，需要定期判断日志文件是否存在。
- 资源占用异常：DBMind服务会执行索引推荐、指标分析、智能巡检等功能，需要占用系统资源；exporter组件会采集服务器上的指标，用于后续分析，也会占用系统资源；当服务/组件的CPU或内存占用超过阈值时，说明资源占用过高，服务/组件出现异常。因此，需要监控服务和组件的资源占用情况。
- 元数据库主备切换：DBMind支持数据库的主备切换功能，在主服务器发生故障等条件下，自动将备份服务器切换为主服务器，确保系统的持续运行。当数据库出现主备切换时，需要DBMind能够正确识别并访问对应的服务器，需要定期判断数据库是否发生了主备切换。如果配置文件中只配置一个元数据库的ip，不支持主备切换。
- 元数据库异常：DBMind需要将配置信息、检测到的异常结果、故障信息等内容进行持久化，存储在元数据库中，因此需要确保元数据库的正常运行；如果元数据库发生异常，则可能会导致依赖元数据库存储结果的任务无法向其插入数据，导致数据遗漏。

资源占用阈值及告警设置如下：

- 内存占用阈值：max(总内存的1%, 200MB)。
- CPU多核占用阈值：cmd-exporter 10%、opengauss-exporter 10%、reprocessing-exporter 10%、DBMind 80%。
- CPU单核占用阈值：cmd-exporter 50%、opengauss-exporter 50%、reprocessing-exporter 50%、DBMind不做限制。
- 调用高可用接口会检查资源占用，连续超过3次阈值才会触发资源占用告警。

风险：当占用资源连续超过阈值三次时，会触发告警，管控侧会重启服务，导致正在运行的任务中断。

当前针对DBMind服务异常的修复性说明如下：

- PID文件丢失：可自动修复；
- 日志文件丢失：可自动修复；
- 组件资源占用异常：不可通过接口修复；
- 元数据库主备切换：可自动修复；
- 元数据库异常：不可通过接口修复；
- 针对Exporter组件不可修复的异常，需要云侧进行处理，如重启进程。

 版权所有 © 华为云计算技术有限公司

⚠ 注意

- [METADATABASE]中的元数据库用户需要具有在该数据库下创建表和数据插入的更新权限，否则执行时会出现异常。而且，需要用户提前创建好数据库，否则，会提示无法连接到该数据库的错误。
- DBMind尝试为每个配置文件目录启动一个DBMind实例，并通过PID文件记录DBMind实例PID，不支持同一配置文件下启动多个服务。
- DBMind提供了requirement.txt文件，用户可以通过该文件安装所需的第三方依赖。
- DBMind只支持特定场景的异常修复，对于不能修复的异常则需要云侧进行处理。
- DBMind支持元数据库主备切换场景的自动修复，但在修复期间对元数据库的业务操作会执行失败，导致数据无法正常插入、查询或更新。
- 如果用户在配置目录的dbmind.conf中填写了AGENT下的master_url，当调用**DBMind的AI子功能**中/v1/api/agents接口更新agent时，如果参数force=true时则不支持更新，该参数只有mater_url没有填写时支持（会自动查找）。
- DBMind当前使用进程池执行特性功能，因此接口侧的响应时间与进程池参数设置和环境等有关，比如用户在confpath目录下的dbmind.conf配置文件中process_num设置不合理或硬件性能较差等，使进程池负载较重导致**DBMind的AI子功能**中部分接口超时，比如/v1/api/risk-analysis/{metric}。

命令参考

用户可以通过“--help”选项获得该模式的帮助信息，例如：

```
gs_dbmind service --help
usage:  service [-h] -c DIRECTORY
                [--only-run
{discard_expired_results,anomaly_detection,cluster_diagnose,agent_update_detect,update_statistics,knob_rec
ommend,slow_query_killer,slow_query_diagnosis,calibrate_security_metrics,check_security_metrics}]
                [--dry-run] [-f] [--interactive | --initialize]
                {setup,start,stop,restart,reload}

positional arguments:
  {setup,start,stop,restart,reload}
                        perform an action for service

optional arguments:
  -h, --help            show this help message and exit
  -c DIRECTORY, --conf DIRECTORY
                        set the directory of configuration files
  --only-run
{discard_expired_results,anomaly_detection,cluster_diagnose,agent_update_detect,update_statistics,knob_rec
ommend,slow_query_killer,slow_query_diagnosis,calibrate_security_metrics,check_security_metrics}
                        explicitly set a certain task running in the backend
  --dry-run             run the backend task(s) once. the task to run can be specified by the --only-run
argument
  -f, --force           force to stop the process and cancel all in-progress tasks
  --interactive         configure and initialize with interactive mode
  --initialize          initialize and check configurations after configuring.
```

 版权所有 © 华为云计算技术有限公司

表 11-2 gs_dbmind service 子命令说明

参数	参数说明	取值范围
action	动作参数。	• setup：初始化配置目录。 • start：启动服务。 • restart: 重启服务。 • reload: 重新加载配置文件参数。 • stop：停止服务。
-c, --conf	配置文件目录地址。	-
--initialize	配置参数初始化。	-
--interactive	交互式输入配置参数。	-
--only-run	选择只运行的定时任务模块。	• slow_query_diagnosis：慢SQL根因分析模块。 • anomaly_detection：异常检测模块。 • slow_query_killer：慢SQL查杀。 • cluster_diagnose：集群故障诊断。 • agent_update_detect：代理更新。 • 下列模块也可输入，但是DBMind内部模块使用，无命令行获取接口，对用户透明：knob_recommend、discard_expired_results，update_statistics，calibrate_security_metrics，check_security_metrics。
--dry-run	只跑一次DBMind的任务，任务执行完毕后退出。	-
-f, --force	强行退出当前正在执行的任务，可用于restart和stop命令中。	-
-h, --help	帮助命令。	-

11.1.2 component 子命令

该子命令可以用于启动DBMind的子组件（或插件），包括可用于监控指标的exporter，以及AI功能等。该命令可以将用户通过命令行传入的命令转发给对应的子组

版权所有 © 华为云计算技术有限公司

件，故不同的子组件命令需参考其功能的对应说明，详见后文各个子组件对应章节，此处不再赘述。

命令参考

用户可以通过“--help”选项获得该模式的帮助信息，例如：

```
gs_dbmind component --help
usage:  component [-h] COMPONENT_NAME ...

positional arguments:
  COMPONENT_NAME  choice a component to start. ['anomaly_detection',
                  'cluster_diagnosis', 'cmd_exporter', 'dkr',
                  'extract_log', 'index_advisor', 'opengauss_exporter',
                  'reprocessing_exporter', 'slow_query_diagnosis',
                  'sql_rewriter', 'sqldiag', 'xtuner']
  ARGS            arguments for the component to start

optional arguments:
  -h, --help      show this help message and exit
```

表 11-3 gs_dbmind component 子命令说明

参数	参数说明	取值范围
COMPONENT_NAME	子组件（即插件）名。	'anomaly_detection', 'cluster_diagnosis', 'cmd_exporter', 'dkr','extract_log', 'index_advisor', 'opengauss_exporter', 'reprocessing_exporter', 'slow_query_diagnosis', 'sql_rewriter', 'sqldiag', 'xtuner'
ARGS	子组件的参数。	参考子组件的命令说明。
-h, --help	帮助命令。	-

11.1.3 set 子命令

该命令用于修改配置文件dbmind.conf中的参数值，与用户手动修改配置文件dbmind.conf一般无差异。例如修改配置目录confpath中的配置文件dbmind.conf中TSDB配置部分，host参数的值，并将其设置为127.0.0.1。则可通过下述命令实现：

```
gs_dbmind set TSDB host 127.0.0.1 -c confpath
```

在修改上述普通参数时，与手动修改配置文件dbmind.conf无差异。但由于DBMind的配置文件中不保存明文密码（如果用户使用明文密码，则DBMind会提示并退出），故当用户想要修改密码项时，有两种方法进行修改，一种是先修改dbmind.conf，并通过以下命令实现配置文件的重新初始化：

```
gs_dbmind service setup --initialize -c confpath
```

另一种方法则是直接通过set子命令进行设置，如：

```
gs_dbmind set METADATABASE password ******** -c confpath
```


版权所有 © 华为云计算技术有限公司

设置完成后，需要重启DBMind服务才会生效：

```
gs_dbmind service restart -c confpath
```

说明

1. 该命令对于字符串是大小写敏感的，如果输错则可能出现执行过程错误；

2. 由于set子命令涉及的参数值类型很多，故只会对设置值进行初步检查，用户需要保证输入值的内容正确，如某些值应为正整数而非负数。

命令参考

用户可以通过“--help”选项获得该模式的帮助信息，例如：

```
gs_dbmind set --help
WARN: The SET operation will not detect the integrity of the module, please install the module in advance.
usage:  set [-h] -c DIRECTORY section option target

positional arguments:
  section               which section (case sensitive) to set
  option                which option to set
  target                the parameter target to set

optional arguments:
  -h, --help            show this help message and exit
  -c DIRECTORY, --conf DIRECTORY
                        set the directory of configuration files
```

表 11-4 gs_dbmind set 子命令说明

参数	参数说明	取值范围
-h, --help	帮助命令	-
-c，--conf	配置文件目录confpath	-
section	设置区	-
option	设置项	-
target	设置值	-

11.1.4 upgrade 子命令

该子命令用于升降级DBMind配置文件和元数据库，DBMind整体升降级还需配合DBMind主服务与exporter组件重启等操作，具体步骤如下（以配置文件路径confpath为例）：

步骤1 在当前版本DBMind文件路径下，停止DBMind服务。

```
gs_dbmind service stop -c confpath
```

步骤2 在高版本DBMind文件路径下，升级配置文件和元数据库。

```
gs_dbmind upgrade -c confpath --target dbmindpath
```

步骤3 查询reprocessing-exporter、cmd-exporter、opengauss-exporter的进程号并重启。以reprocessing-exporter为例。

版权所有 © 华为云计算技术有限公司

1. 查询进程号。
   ```
   ps ux | grep reprocessing_exporter
   ```
2. 终止进程。
   ```
   kill -9 pid
   ```
3. 使用目标版本DBMind启动进程，详细见**component子命令**以及后续DBMind的支持组件章节中对exporter进程的启动说明。
   ```
   gs_dbmind components reprocessing_exporter
   ```

步骤4 在目标版本DBMind文件路径下，启动DBMind服务。

```
gs_dbmind service start -c confpath && gs_dbmind service reload -c confpath
```

----结束

须知

- 升降级前需确认DBMind服务的API接口正常，否则，DBMind无法升级或升级后API接口不可用。
- 配置文件与元数据库升降级过程中，如果出现元数据无法连接等异常，会导致升级失败，此时，相关文件会自动回滚，如果回滚失败，返回255。
- 升降级前需保证配置文件夹可读写。
- 升降级过程中的硬件故障如磁盘故障，机器宕机等，无法保证回滚成功。
- 升降级中的回滚失败时，可重新执行升降级命令进行升降级。
- 升级时回滚不成功的话，如需回退至原先版本，需先升级再降级；降级时回滚不成功的话，如需回退至原先版本，则需先降级再升级。
- 升降级回滚失败时，会保留${confpath}_backup和${confpath}_new_backup两个文件夹，用于升降级的重试，需保证这两个文件夹与用户文件不冲突。

命令参考

用户可以通过 --help 选项获得该模式的帮助信息，例如：

```
gs_dbmind upgrade --help
usage:  upgrade [-h] -c DIRECTORY [--target DIRECTORY]

optional arguments:
  -h, --help            show this help message and exit
  -c DIRECTORY, --conf DIRECTORY
                        set the directory of configuration files
  --target DIRECTORY    set the directory for dbmind of the target version
```

表 11-5 gs_dbmind upgrade 子命令说明

参数	参数说明	取值范围
-h, --help	帮助命令。	-
-c，--conf	配置文件目录confpath。	文件夹路径。
--target	配置目标版本的DBMind目录。	文件夹路径，默认为gs_dbmind文件夹路径。

版权所有 © 华为云计算技术有限公司

11.2 DBMind 的支持组件

支持组件是指DBMind提供的用于支撑整个服务，确保解决方案能够部署和实施的模块。它们本身不是AI功能，却是整个服务体系中非常重要的一环，用于支撑整个自治运维解决方案的快速实施，如用于采集数据库指标的exporter等。

DBMind 的元数据库结构

DBMind元数据库是储存DBMind各类运行数据的数据库，其数据结构如下。

表 11-6 tb_anomaly_detectors 列说明

列名	含义
detector_id	检测器id
cluster_name	集群名称（ip1:port1, ip2:port2）
detector_name	检测器名称
alarm_cause	根因分析手段
alarm_content	异常内容
alarm_level	异常级别
alarm_type	异常类型
extra	额外信息
detector_info	检测器信息
duration	检测时间窗口长度
forecasting_seconds	预测时长
running	运行状态

表 11-7 tb_existing_indexes 列说明

列名	含义
id	索引id
instance	节点地址
db_name	数据库名
tb_name	表名
columns	列
index_stmt	索引表达式

版权所有 © 华为云计算技术有限公司

表 11-8 tb_healing_records 列说明

列名	含义
id	记录id
instance	节点地址
trigger_events	触发事件
trigger_root_causes	触发事件原因
action	动作
called_method	引用方法
success	成功与否
detail	细节
occurrence_at	发生时间

表 11-9 tb_high_availability_status 列说明

列名	含义
id	高可用记录
instance	节点地址
timestamp	时间戳
interface_type	事件类型
interface_info	事件信息

表 11-10 tb_history_alarms 列说明

列名	含义
history_alarm_id	告警id
instance	节点地址
metric_name	指标名称
metric_filter	指标过滤器
alarm_type	异常类型
alarm_level	异常级别
start_at	异常开始时间
end_at	异常结束时间

版权所有 © 华为云计算技术有限公司

列名	含义
alarm_content	告警内容
extra_info	额外信息
anomaly_type	检测器名
alarm_cause	根因分析手段

表 11-11 tb_history_cluster_diagnosis 列说明

列名	含义
diagnosis_id	集群诊断记录id
instance	节点地址
timestamp	时间戳
cluster_role	集群角色
diagnosis_method	诊断方法
cluster_feature	模型输入
diagnosis_result	诊断结果
status_code	诊断结果状态码
alarm_type	异常类型
alarm_level	异常级别

表 11-12 tb_index_recommendation 列说明

列名	含义
id	索引推荐记录id
instance	节点地址
db_name	数据库名
schema_name	schema名
tb_name	表名
columns	列
optimized	是否优化
index_type	索引类型
stmt_count	语句数量

 版权所有 © 华为云计算技术有限公司

列名	含义
select_ratio	select语句比例
insert_ratio	insert语句比例
update_ratio	update语句比例
delete_ratio	delete语句比例
index_stmt	索引表达式

表 11-13 tb_index_recommendation_stats 列说明

列名	含义
id	索引推荐统计id
instance	节点地址
db_name	数据库名
recommend_index_count	推荐索引数量
redundant_index_count	冗余索引数量
invalid_index_count	异常索引数量
stmt_count	语句数量
positive_stmt_count	正向语句数量
table_count	表数量
stmt_source	语句来源
occurrence_time	发生时间

表 11-14 tb_index_recommendation_stmt_details 列说明

列名	含义
id	索引推荐表达式细节id
instance	节点地址
db_name	数据库名
index_id	索引id
template_id	模板id
stmt	语句
optimized	是否优化

 版权所有 © 华为云计算技术有限公司

列名	含义
correlation_type	相关性类型
stmt_count	语句数量

表 11-15 tb_index_recommendation_stmt_templates 列说明

列名	含义
id	索引推荐语句模板id
instance	节点地址
db_name	数据库名
template	模板

表 11-16 tb_knob_recommendation_details 列说明

列名	含义
id	参数推荐细节id
instance	节点地址
name	优化项
current	当前值
recommend	推荐值
min	最小值
max	最大值
restart	是否重启

表 11-17 tb_knob_recommendation_metric_snapshot 列说明

列名	含义
id	参数推荐快照id
instance	节点地址
metric	指标名
value	值

版权所有 © 华为云计算技术有限公司

表 11-18 tb_knob_recommendation_warnings 列说明

列名	含义
id	参数推荐告警id
instance	节点地址
comment	告警内容
level	告警级别

表 11-19 tb_metric_statistics 列说明

列名	含义
stat_id	指标统计id
metric_name	指标名称
method	统计方法
length	保留长度
step	步长
start	开始时间戳
metric_filter	指标过滤项
value_list	数值表

表 11-20 tb_regular_inspections 列说明

列名	含义
id	巡检id
instance	节点地址
inspection_type	巡检类型
report	报告
conclusion	结论
start	开始时间
end	结束时间
state	节点状态
cost_time	耗费时长

版权所有 © 华为云计算技术有限公司

表 11-21 tb_security_anomaly_values 列说明

列名	含义
id	安全异常检测id
host	节点地址
metric_name	指标名称
metric_time	监测时间
metric_value	指标值

表 11-22 tb_security_metric_models 列说明

列名	含义
id	安全异常检测模型id
host	节点地址
metric_name	指标名称
parameter_name	参数名
parameter_value	参数值
time_saved	模型保存时间

表 11-23 tb_slow_queries 列说明

列名	含义
slow_query_id	慢查询id
instance	节点地址
schema_name	schema名
db_name	数据库名
query	查询语句
query_type	查询类型
involving_systable	涉及的系统表
template_id	模板id
hashcode1	哈希码1
hashcode2	哈希码2
insert_at	插入时间

版权所有 © 华为云计算技术有限公司

列名	含义
hit_rate	buffer命中率
fetch_rate	语句fetch率
cpu_time	cpu计算时长
data_io_time	数据io时长
plan_time	计划时长
parse_time	解析时长
db_time	数据库内部时长
root_cause	根因
suggestion	建议

表 11-24 tb_slow_queries_journal 列说明

列名	含义
journal_id	慢查询日报id
slow_query_id	慢查询id
start_at	开始时间
round_start_at	开始日期
duration_time	持续时长
instance	节点地址

表 11-25 tb_killed_slow_queries 列说明

列名	含义
killed_query_id	被查杀的慢查询id
instance	节点地址
db_name	数据库名
query	查询语句
killed	是否被查杀
username	用户名
elapsed_time	花费时间

版权所有 © 华为云计算技术有限公司

列名	含义
killed_time	被查杀时间

11.2.1 Prometheus Exporter 组件

11.2.1.1 概述

Prometheus是业内非常流行的开源监控系统，同时本身也是一款时序数据库。Prometheus的采集端被称为exporter，用来收集被监控模块的指标项。为了与Prometheus平台完成对接，AI工具自带了几款exporter，分别是用来采集数据库指标的opengauss-exporter，用来执行cmd命令并获取返回结果以及采集日志信息的cmd-exporter，以及对采集到的指标进行二次加工的reprocessing-exporter。

须知

- Prometheus和exporter是业内流行的监控和采集平台，部署在内网环境中，不对外部暴露接口，仅供内部监控平台使用。因此，为了增强该平台的安全性，一般需要用户或运维人员配置防火墙等，以便隔离外部访问，从而增强监控平台的安全性。
- Prometheus平台在默认情况下，采用Http协议、并且没有任何安全访问限制。这是因为，该平台一般部署在内网环境中，攻击风险可控。如果用户希望提高安全性，可自行修改Prometheus的TLS配置选项或增加basic_auth_users相关选项，但仍不建议对外部直接暴露访问接口。
- Prometheus默认绑定的IP地址是0.0.0.0，DBMind无法自动获知用户希望绑定的IP地址，故用户应该显式设置Prometheus的绑定IP地址，以减小安全风险。对于Prometheus的其他开源exporter（如node-exporter）也是同理。
- DBMind的exporter与Prometheus交互时，若使用SSL证书，则默认会进行双向认证。
- Prometheus的CA证书默认只支持SAN（Subject Alternative Name）形式，用户在配置Prometheus的CA证书，以及与Prometheus进行认证的证书时，需要使用SAN格式。同时，Prometheus不支持密码保护的私钥文件，用户在为Prometheus生成私钥文件时，请不要使用密码保护。例如，用户可以参考下述示例，通过openssl工具生成SAN形式的自签发证书和私钥文件：

  ```
  openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout prometheus.key -out prometheus.crt -subj "/C=CN/ST=GaussDB/L=GaussDB/O=GaussDB-geek/CN=localhost" -addext "subjectAltName = DNS:localhost"
  ```

  通过执行上述命令，可以生成两个文件prometheus.crt和prometheus.key，其中prometheus.crt文件可以同时作为CA证书使用。
- DBMind的exporter默认支持HTTPS协议，需要用户手工指定证书文件路径，建议用户使用。
- DBMind Service的部分功能会从Prometheus中获取时序数据，该通道也可以使用SSL协议进行加固，prometheus默认支持https通信，建议用户使用。
- Prometheus的配置文件prometheus.yml使用IPv6地址时，ip地址需要使用中括号[]和端口号进行分隔，例：[2407:c080:18c1:33b:5c7a:3859:62ac:f284]:9180
- Prometheus纳管集群时，部分指标需要保证同一个实例（数据库）只有一条数据，否则会导致DBMind中的获取数据信息接口（fetchone/fetchall）获取到的信息不准确，进而导致检测功能出现异常。

版权所有 © 华为云计算技术有限公司

11.2.1.2 环境部署

用户可以从Prometheus的官网上下载Prometheus-server和node-exporter，然后根据官方文档中的说明启动它们；

其中，提供的exporter组件默认采用Https通信协议，因此需要用户默认提供SSL证书和密钥文件，并通过“--ssl-keyfile”、“--ssl-certfile”以及“--ssl-ca-file”选项进行配置。若用户不希望使用HTTPS协议，则可以通过“--disable-https”选项禁用该模式。

说明

由于GaussDB默认模式下的通信加密协议与PostgreSQL不兼容，故导致通过PyPI源安装的基于PostgreSQL编译的Python驱动psycopg2-binary默认无法连接至数据库。

因此，需要用户自行编译psycopg2或修改GUC参数进行适配。也可通过GaussDB官方网站下载基于GaussDB编译的psycopg2（官方网站仅提供部分Python版本的编译包，需要用户鉴别是否与当前Python运行时版本一致）。

11.2.1.3 使用指导

用户可通过gs_dbmind命令启动对应的exporter。下面为用户演示一个完整的Prometheus监控平台的搭建过程。

步骤1 部署Prometheus主进程，运行如下命令：

```
prometheus --config.file=prometheus.yml
```

步骤2 部署opengauss-exporter：opengauss-exporter支持单节点和多节点部署模式，单节点部署下，URL参数中仅包含一个节点地址，下面示例使用默认侦听端口号9187，侦听地址为192.168.1.100，采用HTTPS协议，则命令可以为：

```
gs_dbmind component opengauss_exporter --url postgresql://user:password@ip:port/dbname --web.listen-address 192.168.1.100 --ssl-keyfile server.key --ssl-certfile server.crt --ssl-ca-file server.crt
```

opengauss-exporter多节点部署模式示例，URL参数中包含多个节点地址，下面示例使用默认侦听端口号9187，侦听地址为192.168.1.100，URL参数中包含三个节点地址，采用HTTPS协议，则命令可以为：

```
gs_dbmind component opengauss_exporter --url postgresql://user:password@ip1:port1,ip2:port2,ip3:port3/dbname --web.listen-address 192.168.1.100 --ssl-keyfile server.key --ssl-certfile server.crt --ssl-ca-file server.crt
```

步骤3 部署reprocessing-exporter：启动reprocessing-exporter，采用默认侦听端口号8181，侦听地址为192.168.1.101，Prometheus-server IP与端口号为192.168.1.100:9090，采用HTTPS协议，则命令可以为：

```
gs_dbmind component reprocessing_exporter 192.168.1.100 9090 --web.listen-address 192.168.1.101 --ssl-keyfile server.key --ssl-certfile server.crt --ssl-ca-file server.crt
```

步骤4 部署cmd-exporter: 启动cmd-exporter，使用默认参数，设置采集的GaussDB日志目录路径，并给定证书信息：

```
gs_dbmind component cmd_exporter --ssl-keyfile server.key --ssl-certfile server.crt --ssl-ca-file server.crt --pg-log-dir /path/to/pglog
```

步骤5 部署node-exporter：一般而言，Prometheus监控平台都需要部署node-exporter用于监控Linux操作系统，后文提到的部分AI功能也需要依赖node-exporter采集Linux系统指标，故也需要用户来部署；使用方法详见：https://prometheus.io/docs/guides/node-exporter/#installing-and-running-the-node-exporter。用户可直接运行该node-exporter进程，其默认端口号为9100，启动命令行为：

```
node_exporter
```

----结束

版权所有 © 华为云计算技术有限公司

⚠ 注意

DBMind高可用功能需要在未纳管实例的情况下实现，因此接口不需要token认证。应当使用默认的HTTPS协议以进行SSL双向认证，使用HTTP协议则可能引入攻击者直接发起未授权请求的安全风险。

为保证DBMind云上使用时高可靠，exporter组件提供了组件状态查询和部分异常修复接口，详细请参见**表11-29**。

- 组件状态查询接口示例：

  ```
  curl -X 'POST'"http://127.0.0.1:8080/v1/api/check-status" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind component cmd_exporter --web.listen-address 0.0.0.0 --web.listen-port 9181 --log.filepath /xxx/dbmind_cmd_exporter.log --disable-https"}'
  ```

  如果使用HTTPS协议，则查询示例为：

  ```
  curl -X 'POST'"https://127.0.0.1:8080/v1/api/check-status" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind component cmd_exporter --web.listen-address 0.0.0.0 --web.listen-port 9181 --log.filepath /xxx/dbmind_cmd_exporter.log --ssl-keyfile xx.key --ssl-certfile xx.crt --ssl-ca-file xx.crt"}' --cacert xx.crt --key xx.key --cert xx.crt
  ```

  返回结果示例：

  ```
  {"data":{"error_msg":"","result":{},"state":"NORMAL"},"success":true}
  ```

- 组件异常修复接口示例：

  ```
  curl -X 'POST'"http://127.0.0.1:8080/v1/api/repair" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind component cmd_exporter --web.listen-address 0.0.0.0 --web.listen-port 9181 --log.filepath /xxx/dbmind_cmd_exporter.log --disable-https"}'
  ```

  如果使用HTTPS协议，则修复示例为：

  ```
  curl -X 'POST'"https://127.0.0.1:8080/v1/api/repair" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"cmd": "./gs_dbmind component cmd_exporter --web.listen-address 0.0.0.0 --web.listen-port 9181 --log.filepath /xxx/dbmind_cmd_exporter.log --ssl-keyfile xx.key --ssl-certfile xx.crt --ssl-ca-file xx.crt"}' --cacert xx.crt --key xx.key --cert xx.crt
  ```

  返回结果示例：

  ```
  {"data":{"error_msg":"","result":{},"state":"SUCCESS"},"success":true}
  ```


版权所有 © 华为云计算技术有限公司

说明

exporter启动后，会根据dbmind/components/opengauss_exporter/yamls、dbmind/components/cmd_exporter/yamls目录下的配置文件信息和dbmind/components/reprocessing_exporter/reprocessing_exporter.yml文件中的信息来采集特定的指标。指标的配置信息如下例所示：

```
indicator_name:
  name: indicator_name
  desc: xxx
  query:
    - name: indicator_name
      sql: xxx
      version: xxx
      timeout: 10
      status: enable
  metrics:
    ...
  status: enable
  ttl: 0
  timeout: 10
  null_coverage_window: 30
```

当status值为enable时，会在指定的超时时间内对该指标进行采集。用户可以对默认开启状态status和超时时间timeout进行修改，修改后需重新启动exporter才能生效。

null_coverage_window参数：当单次采集失败时，可以选择上一个成功的采集点来填充到空缺位置，当时钟超过上一个成功的采集时间加上null_coverage_window的秒数的阈值之后，不再进行填充。当null_coverage_window缺失时，会使用统一的默认值300秒来作为缺值补充的时间阈值。

组件状态查询和修复接口只针对DBMind自研exporter，包括opengauss-exporter、cmd-exporter和reprocessing-exporter，对于开源的node-exporter则不支持。

当前Exporter组件支持查询的异常场景如下：

- PID文件丢失或文件内容异常：当用户启动服务/组件后，会自动生成*.pid文件来记录进程对应的进程标识符，也就是PID。PID文件用于云管控来判断进程状态，发现进程级别的异常并进行修复。为防止PID文件的误删除与内容的误修改，需要定期判断pid文件是否存在以及其内容是否正确。
- 日志文件丢失：当用户启动服务/组件后，会自动生成*.log文件来记录进程运行的日志。此日志文件用于快速定位问题的根源、追踪程序执行的过程等。为防止日志文件的误删除，需要定期判断日志文件是否存在。
- 资源占用异常：DBMind服务会执行索引推荐、指标分析、智能巡检等功能，需要占用系统资源；exporter组件会采集服务器上的指标，用于后续分析，也会占用系统资源；当服务/组件的CPU或内存占用超过阈值时，说明资源占用过高，服务/组件出现异常。因此需要监控服务和组件的资源占用情况。
- 数据库异常：opengauss-exporter需要从数据库中获取数据，用于生成采集结果，所以需要判断数据库是否能够正常连接。

资源占用阈值及告警设置如下：

- 内存占用阈值：max(总内存的1%, 200MB)。
- CPU多核占用阈值：cmd-exporter 10%，opengauss-exporter 10%，reprocessing-exporter 10%， DBMind 80%。
- CPU单核占用阈值：cmd-exporter 50%，opengauss-exporter 50%，reprocessing-exporter 50%，DBMind不做限制。
- 调用高可用接口会检查资源占用，连续超过3次阈值才会触发资源占用告警。

风险：当占用资源连续超过阈值三次时，会触发告警，管控侧会重启服务，导致正在运行的任务中断。

当前Exporter针对组件异常的修复性说明如下：

- PID文件丢失：可自动修复；
- 日志文件丢失：可自动修复；

 版权所有 © 华为云计算技术有限公司

- 组件资源占用异常：不可通过接口修复；
- 数据库异常：不可通过接口修复；
- 针对Exporter组件不可修复的异常，需要云侧进行处理，如重启进程。
- 集中式环境下，opengauss-exporter需要使用单节点部署模式以保证获取必须的指标信息。
- 在分布式场景下，CN按照单节点模式启动，同时需要开启agent功能，DN主备节点则需要使用多节点部署模式，分布式示例如下：

  分布式CN、DN节点信息如下，该分布式属于3C3D规格，每个DN有两个备节点，下面对DN部署逻辑给出示例：

  [Coordinator State]

  node node_ip instance state

  ---

  1 node1 ip1 5001 port Normal

  2 node2 ip2 5002 port Normal

  3 node3 ip3 5003 port Normal

  [Datanode State]

  node node_ip instance state | node node_ip instance state | node node_ip instance state

  ---

  1 node1 ip1 6001 port1 P Primary Normal | 2 node2 ip2 6002 port1 S Standby Normal | 3 node3 ip3 6003 port1 S Standby Normal

  2 node2 ip2 6004 port2 P Primary Normal | 3 node3 ip3 6005 port2 S Standby Normal | 1 node1 ip1 6006 port2 S Standby Normal

  3 node3 ip3 6007 port3 P Primary Normal | 1 node1 ip1 6008 port3 S Standby Normal | 2 node2 ip2 6009 port3 S Standby Normal

  每个DN为一主两备的配置，因此需要启动三个opengauss-exporter进程，每个进程对DN的主备进行监控，其详细命令如下（以https方式为例）：

  进程1：gs_dbmind component opengauss_exporter --url postgresql://user:password@ip1:port1,ip2:port1,ip3:port1/dbname --web.listen-address listen_ip1 --web.listen-port listen_port1 --ssl-keyfile server.key --ssl-certfile server.crt --ssl-ca-file server.crt

  进程2：gs_dbmind component opengauss_exporter --url postgresql://user:password@ip2:port2,ip3:port2,ip1:port2/dbname --web.listen-address listen_ip2 --web.listen-port listen_port2 --ssl-keyfile server.key --ssl-certfile server.crt --ssl-ca-file server.crt

  进程3：gs_dbmind component opengauss_exporter --url postgresql://user:password@ip3:port3,ip1:port3,ip2:port3/dbname --web.listen-address listen_ip3 --web.listen-port listen_port3 --ssl-keyfile server.key --ssl-certfile server.crt --ssl-ca-file server.crt

- 以多节点的连接方式启动的opengauss-exporter不支持RPC服务，多节点的url遵循主节点优先，从左至右的原则，周期性获取指标时每次更新
- 以多节点的连接方式启动的opengauss-exporter在URL中最多支持8个IP地址和端口号，以多节点方式连接的情况下，尝试连接每个地址的超时时间为1秒。

 版权所有 © 华为云计算技术有限公司

⚠ 注意

- opengauss-exporter中连接数据库的用户需要monitor admin或以上权限，否则会出现部分指标无法采集的情况。同时opengauss-exporter不支持使用数据库初始用户来进行数据采集。
- opengauss-exporter多节点部署模式下，参数URL中最多支持8个节点地址，否则进程无法正常启动。
- opengauss-exporter多节点部署模式下只会连接主节点，如果提供的地址都为备节点则组件无法正常启动。
- opengauss-exporter多节点部署模式目的在于支持分布式场景下关键亚健康场景的检测与分析，用户在使用时需要保证节点地址来自于同一个实例，否者可能出现问题。
- opengauss-exporter多节点部署模式下，组件不会对URL中节点地址进行约束，在启动时会按照地址先后顺序依次连接，当连接成功时不会继续尝试连接后续地址，启动时如果所有地址均连接失败则进程无法成功，运行过程中如果所有节点均连接失败则进程会固定间隔尝试重新链接，连接成功后组件正常采集。
- opengauss-exporter运行时使用驱动连接数据库执行SQL语句，分布式场景下如果想在服务节点上正常连接实例节点，需要使用支持本地进行驱动连接的认证方式，如kerberos认证;
- opengauss-exporter需要在url参数中连接postgres数据库来获取全局的指标信息，对特定数据库的采集请使用include-databases参数来指定
- opengauss-exporter中连接数据库的用户需要获取dbe_perf模式下的数据，因此需要保证其具有该模式的权限。
- opengauss-exporter会从dbe_perf.statement_history中抽样慢SQL信息，dbe_perf.statement_history视图慢SQL记录与GUC参数log_min_duration_statement和track_stmt_stat_level相关，其中log_min_duration_statement是慢SQL阈值，单位毫秒，具体值由用户设置；track_stmt_stat_level是SQL记录级别，默认为'OFF,L0'，即只记录慢SQL信息，级别为L0，建议用户在详细了解参数意义与作用的情况下谨慎修改。
- opengauss-exporter采集数据库相关信息，主要包括部分系统表和视图中的数据（具体参见代码中opengauss-exporter中的配置文件）。node-exporter采集系统指标信息，主要与系统磁盘、CPU等相关。reprocessing-exporter基于prometheus-server中的某些指标（具体参见代码中reprocessing-exporter中的配置文件）进行二次加工，最终提供加工后的数据供用户使用。
- 当数据库部署在多层网络上时（分布式：管理层-数据层-数据同步层），opengauss-exporter的启动命令中的url参数必须使用数据层的ip地址，否则会影响DBMind的其他功能。
- prometheus-server在拉取exporter数据时有超时机制，超时时间由scrape_timeout（默认10s）控制，因此当exporter采集数据量较大时，用户可根据实际情况增大scrape_timeout以防止超时报错，另外需要注意的是scrape_interval（采集间隔，默认15s）不能比scrape_timeout小，否则会出现异常。
- opengauss-exporter的采集语句存在timeout机制，由于prometheus-server的scrape_timeout默认为10秒，所以采集语句的上限为10秒，如果遇到数据库负载较高查询响应较慢的情况，可能会超过采集语句的timeout时长导致数据缺失。
- 如果数据库时区设置和系统不相同，可能会出现时间相关指标时间与系统时间不一致的情况，因此需要将数据库时区与系统保持同步。
- 当使用https通信时，工具会尝试更改证书与密钥文件权限以及校验证书有效期，如果无法更改文件权限则会出现报警，或者证书有效期小于90天会出现报警。

 版权所有 © 华为云计算技术有限公司

- 当存在指标重复采集时，opengauss-exporter会出现异常，异常信息会打印到日志中。
- 使用opengauss-exporter的--config、--disable-settings-metrics、--disable-statement-history-metrics三个参数时需要注意，其存在以下几种情况：
  - 用户不指定其中任何参数，则工具会同时对dbmind/components/opengauss_exporter/yamls目录下的三个配置文件中的指标进行采集。
  - 用户显式指定--config，则工具不会采集dbmind/components/opengauss_exporter/yamls目录下default.yml中的指标，而会采集用户指定配置文件中的指标，同时pg_settings.yml和statements.yml正常采集，此时用户需要注意指定的配置文件中的指标和pg_settings.yml、statements.yml中的指标不能存在重复采集的现象。
  - 用户显式指定--disable-settings-metrics，则工具不会采集dbmind/components/opengauss_exporter/yamls目录下pg_settings.yml中的指标；用户显式指定--disable-statement-history-metrics，则工具不会采集yamls目录下statements.yml（慢SQL相关）中的指标。
- exporter启动后，一般情况下不会终止进程并退出（如连接的数据库地址不可用、连接的数据库用户被删除或禁用等），而是会在日志中记录报错信息，并在后台重新尝试。
- cmd-exporter默认Linux系统定义了需要监控的gaussDB的日志路径：$GAUSSLOG，如果没有定义$GAUSSLOG则需要进行定义。
- cmd-exporter默认匹配的是GUC参数“log_statement='none'”时的日志格式，当log_statement为其他参数时cmd-exporter会无法识别日志格式。
- 由于cmd-exporter需要访问数据库日志和使用cm工具，所以需要由数据库用户部署cmd-exporter来获取相应的状态信息。
- 实例升级过程中，opengauss-exporter无法正常采集数据，cmd-exporter无法采集数据库的cm相关信息。
- cmd-exporter的日志采集功能对于内核日志的日志文件名和日志格式都有要求，在gaussdb.conf文件中，log_filename的格式要求为'gaussdb-%Y-%m-%d_%H%M%S.log'，log_line_prefix的格式要求为'%m %n %u %d %h %p %S %x %a'。
- cmd-exporter采集的日志指标为散点图形式，各个采集点之间标签不同，所以不能合并到同一个集合中，不适合用折线图的方式进行展示，请采用表格或者散点图的方式进行展示。
- 当cmd-exporter在一个采集周期内没有采集到日志指标时（如gaussdb_log_ffic为空，则表示没有生成ffic文件），则该指标数据为空。
- 在某些特殊情况下由于日志文件激增，考虑到cpu的处理能力和内存限制，cmd-exporter为日志的消息队列设置了长度上限以防止内存泄露。当队列满时将停止新的日志事件入列，直到队列有空间为止。这一设计可能导致在日志负载高的时候丢失部分日志指标。
- reprocessing-exporter中采集指标os_disk_usage时，只针对EXT和XFS文件系统，其他文件系统不会采集。
- 采集组件启动时需要保证侦听端口和侦听地址可用，否则进程无法正常启动。
- 采集系统在启动瞬间的时候，各个exporter会存在初始化过程，可能会影响reprocessing-exporter指标的准确性。

11.2.1.4 获取帮助

用户可以通过“--help”命令获取Exporter组件启动或停止等操作的帮助信息，例如：

版权所有 © 华为云计算技术有限公司

```
gs_dbmind component opengauss_exporter --help
gs_dbmind component reprocessing_exporter --help
gs_dbmind component cmd_exporter --help
```

Exporter组件状态查询和修复的RESETful API列表如下：

API	入参	参数介绍	请求方法	功能描述与预期返回结果
/v1/api/check-status	cmd	组件启动命令，String，必选	POST	获取exporter组件的状态信息并返回状态详情。
/v1/api/repair	cmd	组件启动命令，String，必选	POST	修复exporter组件并返回修复结果。

11.2.1.5 命令参考

reprocessing-exporter的使用帮助详情：

```
gs_dbmind component reprocessing_exporter --help
usage:  [-h] [--prometheus-auth-user PROMETHEUS_AUTH_USER]
        [--prometheus-auth-password PROMETHEUS_AUTH_PASSWORD]
        [--disable-https] [--ssl-keyfile SSL_KEYFILE]
        [--ssl-certfile SSL_CERTFILE] [--ssl-ca-file SSL_CA_FILE]
        [--tsdb-ssl-keyfile TSDB_SSL_KEYFILE] [--tsdb-ssl-certfile TSDB_SSL_CERTFILE]
        [--tsdb-ssl-ca-file TSDB_SSL_CA_FILE]
        [--web.listen-address WEB.LISTEN_ADDRESS]
        [--web.listen-port WEB.LISTEN_PORT]
        [--collector.config COLLECTOR.CONFIG] [--log.filepath LOG.FILEPATH]
        [--log.level {debug,info,warn,error,fatal}] [-v]
        prometheus_host prometheus_port

Reprocessing Exporter: A re-processing module for metrics stored in the
Prometheus server.

positional arguments:
  prometheus_host       from which host to pull data
  prometheus_port       the port to connect to the Prometheus host

optional arguments:
  -h, --help            show this help message and exit
  --prometheus-auth-user PROMETHEUS_AUTH_USER
                        use this user for basic authorization to connect to
                        the Prometheus server
  --prometheus-auth-password PROMETHEUS_AUTH_PASSWORD
                        use this password for basic authorization to connect
                        to the Prometheus server
  --disable-https       disable Https scheme
  --ssl-keyfile SSL_KEYFILE
                        set the path of ssl key file
  --ssl-certfile SSL_CERTFILE
                        set the path of ssl certificate file
  --ssl-ca-file SSL_CA_FILE
                        set the path of ssl ca file
  --tsdb-ssl-keyfile TSDB_SSL_KEYFILE
                        set the path of tsdb ssl key file
  --tsdb-ssl-certfile TSDB_SSL_CERTFILE
                        set the path of tsdb ssl certificate file
  --tsdb-ssl-ca-file TSDB_SSL_CA_FILE
                        set the path of tsdb ssl ca file
  --web.listen-address WEB.LISTEN_ADDRESS
                        address on which to expose metrics and web interface
  --web.listen-port WEB.LISTEN_PORT
                        listen port to expose metrics and web interface
  --collector.config COLLECTOR.CONFIG, --config COLLECTOR.CONFIG
```


版权所有 © 华为云计算技术有限公司

```
                    according to the content of the yaml file for metric
                    collection
--log.filepath LOG.FILEPATH
                    the path to log
--log.level {debug,info,warn,error,fatal}
                    only log messages with the given severity or above.
                    Valid levels: [debug, info, warn, error, fatal]
-v, --version       show program's version number and exit
```

表 11-26 reprocessing-exporter 的命令行参数详情表

参数	参数说明	取值范围
prometheus_host	Prometheus-server的IP地址。	-
prometheus_port	Prometheus-server的服务侦听端口号。	1024-65535。
-h, --help	帮助选项。	-
--prometheus-auth-user	prometheus用户名称。	-
--prometheus-auth-password	prometheus用户密码。	-
--disable-https	禁用HTTPS协议。	-
--ssl-keyfile	HTTPS协议使用的证书私钥文件路径，如果为密文私钥，需要通过管道传入私钥密码，传输内容为json格式，密码填充在ssl-keyfile-password字段。如：`echo {"ssl-keyfile-password":"password"} | gs_dbmind component reprocessing_exporter ...`。	-
--ssl-certfile	HTTPS协议使用的证书文件路径。	-
--ssl-ca-file	HTTPS协议使用的CA证书文件路径。	-
--tsdb-ssl-keyfile	TSDB数据库使用的HTTPS协议证书私钥文件路径。	-
--tsdb-ssl-certfile	TSDB数据库使用的HTTPS协议证书文件路径。	-
--tsdb-ssl-ca-file	TSDB数据库使用的HTTPS协议CA证书文件路径。	-
--web.listen-address	该exporter服务的绑定IP。	-

版权所有 © 华为云计算技术有限公司

参数	参数说明	取值范围
--web.listen-port	该exporter服务的侦听端口。	1024-65535。
--collector.config	显性指定的待采集指标配置文件路径。	-
--log.filepath	日志文件保存路径，默认保存在当前目录下。	-
--log.level	日志文件的打印级别，默认为INFO级别。	DEBUG、INFO、WARN、ERROR、FATAL。
--version	显示版本信息。	-

opengauss-exporter的使用帮助详情：

```
gs_dbmind component opengauss_exporter --help
usage:  [-h] --url URL [--config-file CONFIG_FILE]
        [--include-databases INCLUDE_DATABASES]
        [--exclude-databases EXCLUDE_DATABASES]
        [--constant-labels CONSTANT_LABELS]
        [--scrape-interval-seconds SCRAPE_INTERVAL_SECONDS]
        [--web.listen-address WEB.LISTEN_ADDRESS]
        [--web.listen-port WEB.LISTEN_PORT] [--disable-cache]
        [--disable-settings-metrics] [--disable-statement-history-metrics]
        [--disable-https] [--disable-agent] [--ssl-keyfile SSL_KEYFILE]
        [--ssl-certfile SSL_CERTFILE] [--ssl-ca-file SSL_CA_FILE]
        [--parallel PARALLEL] [--connection-pool-size CONNECTION_POOL_SIZE]
        [--log.filepath LOG.FILEPATH]
        [--log.level {debug,info,warn,error,fatal}] [-v]

openGauss Exporter (DBMind): Monitoring or controlling for openGauss.

optional arguments:
  -h, --help            show this help message and exit
  --url URL, --dsn URL  openGauss database target url. It is recommended to
                        connect to the postgres database through this URL, so
                        that the exporter can actively discover and monitor
                        other databases.
  --config-file CONFIG_FILE, --config CONFIG_FILE
                        path to config file.
  --include-databases INCLUDE_DATABASES
                        only scrape metrics from the given database list. a
                        list of database name (format is label=dbname or
                        dbname) separated by comma(,).
  --exclude-databases EXCLUDE_DATABASES
                        scrape metrics from the all auto-discovered databases
                        excluding the list of database. a list of database
                        name (format is label=dbname or dbname) separated by
                        comma(,).
  --constant-labels CONSTANT_LABELS
                        a list of label=value separated by comma(,).
  --scrape-interval-seconds SCRAPE_INTERVAL_SECONDS
                        specify the scrape interval in seconds to reduce
                        redundant results. If set 0, it means automatically
                        calculate.
  --web.listen-address WEB.LISTEN_ADDRESS
                        address on which to expose metrics and web interface
  --web.listen-port WEB.LISTEN_PORT
                        listen port to expose metrics and web interface
  --disable-cache       force not using cache.
```


版权所有 © 华为云计算技术有限公司

```
--disable-settings-metrics
                    not collect pg_settings.yml metrics.
--disable-statement-history-metrics
                    not collect statement-history metrics (including slow
                    queries).
--disable-https     disable Https scheme
--disable-agent     by default, this exporter also assumes the role of
                    DBMind-Agent, that is, executing database operation
                    and maintenance actions issued by the DBMind service.
                    With this argument, users can disable the agent
                    functionality, thereby prohibiting the DBMind service
                    from making changes to the database.
--ssl-keyfile SSL_KEYFILE
                    set the path of ssl key file
--ssl-certfile SSL_CERTFILE
                    set the path of ssl certificate file
--ssl-ca-file SSL_CA_FILE
                    set the path of ssl ca file
--parallel PARALLEL   number of parallels for metrics scrape.
--connection-pool-size CONNECTION_POOL_SIZE
                    size of connection pool for each database. Set zero to
                    disable connection pool.
--log.filepath LOG.FILEPATH
                    the path to log
--log.level {debug,info,warn,error,fatal}
                    only log messages with the given severity or above.
                    Valid levels: [debug, info, warn, error, fatal]
-v, --version       show program's version number and exit
```

表 11-27 opengauss-exporter 的命令行参数详情表

参数	参数说明	取值范围
--url, --dsn	数据库server的连接地址，支持单节点模式和多节点模式，单节点模式下示例： postgres://user:pwd@host:port/dbname。多节点部署模式下示例： postgres://user:pwd@host1:port1,host2:port2,host3:port3/dbname。其中不同地址之间通过逗号分割，同时节点数量不超过8个，否则进程无法正常启动。 密码字段为空时，需要通过管道密码，传输内容为json格式，密码填充在db-password字段。如：`echo {"db-password":"password"} | gs_dbmind component opengauss_exporter ...`。	如果该url涉及到的各字段URL包含特殊字符（如@，/等），则需要通过URL编码进行转义，例如密码中的"@"应转义为%40，"/"应转义为%2F，否则各字段的含义会被错误识别和切分，具体转义规则可以参考URL编码的转义规则，该URL地址规则遵循RFC-1738标准。
--constant-labels	常量列表，k=v格式，用逗号隔开，表明该exporter自带的常量标签。	格式如：“cluster_name=demo,cluster_id=1”。
-h, --help	帮助选项。	-

版权所有 © 华为云计算技术有限公司

参数	参数说明	取值范围
--disable-https	禁用HTTPS协议。	-
--ssl-keyfile	HTTPS协议使用的证书私钥文件路径，如果为密文私钥，需要通过管道传入私钥密码，传输内容为json格式，密码填充在ssl-keyfile-password字段。如：`echo {"ssl-keyfile-password":"password"} | gs_dbmind component opengauss_exporter ...`。。	-
--ssl-certfile	HTTPS协议使用的证书文件路径。	-
--ssl-ca-file	HTTPS协议使用的CA证书文件路径。	-
--web.listen-address	该exporter服务的绑定IP。	-
--web.listen-port	该exporter服务的侦听端口。	1024-65535。
--config, --config-file	显性指定的待采集指标配置文件路径。	-
--log.filepath	日志文件保存路径，默认保存在当前目录下。	-
--log.level	日志文件的打印级别，默认为INFO级别。	DEBUG、INFO、WARN、ERROR、FATAL。
--version	显示版本信息。	-
--disable-cache	禁止使用缓存。	-
--disable-settings-metrics	禁止采集pg_settings表的值。	-
--disable-statement-history-metrics	禁止采集statement_history表中的慢SQL信息。	-
--disable-agent	禁止agent行为。	-
--include-databases	显性表明待采集的数据库名，指定多个数据库时用逗号（,）隔开。	-
--exclude-databases	显性表明不采集的数据库名，指定多个数据库时用逗号（,）隔开。	-

版权所有 © 华为云计算技术有限公司

参数	参数说明	取值范围
--parallel	指标采集的并行线程数，默认为5。	正整数。
--scrape-interval-seconds	明确抓取间隔避免重复采集指标，默认为0。	>=0的整数，如果为0，则根据上一次采集的时间对采集间隔进行动态调整。
--connection-pool-size	数据库的连接池大小 ，默认为0。	>=0的整数，如果为0，则不采用连接池连接。

cmd-exporter的使用帮助详情：

```
gs_dbmind component cmd_exporter --help
usage:  [-h] [--constant-labels CONSTANT_LABELS] [--web.listen-address WEB.LISTEN_ADDRESS] [--
web.listen-port WEB.LISTEN_PORT] [--disable-https]
       [--config CONFIG] [--ssl-keyfile SSL_KEYFILE] [--ssl-certfile SSL_CERTFILE] [--ssl-ca-file SSL_CA_FILE]
[--parallel PARALLEL]
       [--pg-log-dir PG_LOG_DIR] [--disable-log-exporter] [--log.filepath LOG.FILEPATH] [--log.level
{debug,info,warn,error,fatal}] [-v]

Command Exporter (DBMind): scrape metrics by performing shell commands.

optional arguments:
  -h, --help            show this help message and exit
  --constant-labels CONSTANT_LABELS
                        a list of label=value separated by comma(,).
  --web.listen-address WEB.LISTEN_ADDRESS
                        address on which to expose metrics and web interface
  --web.listen-port WEB.LISTEN_PORT
                        listen port to expose metrics and web interface
  --disable-https       disable Https scheme
  --config CONFIG       path to config dir or file.
  --ssl-keyfile SSL_KEYFILE
                        set the path of ssl key file
  --ssl-certfile SSL_CERTFILE
                        set the path of ssl certificate file
  --ssl-ca-file SSL_CA_FILE
                        set the path of ssl ca file
  --parallel PARALLEL   performing shell command in parallel.
  --pg-log-dir PG_LOG_DIR
                        set the directory path of PGLOG, default value is $GAUSSLOG.
  --disable-log-exporter
                        disable log analysis
  --log.filepath LOG.FILEPATH
                        the path to log
  --log.level {debug,info,warn,error,fatal}
                        only log messages with the given severity or above. Valid levels: [debug, info, warn, error,
fatal]
  -v, --version         show program's version number and exit
```

表 11-28 cmd-exporter 的命令行参数详情表

参数	参数说明	取值范围
-h, --help	帮助选项。	-
--disable-https	禁用HTTPS协议。	-

版权所有 © 华为云计算技术有限公司

参数	参数说明	取值范围
--ssl-keyfile	HTTPS协议使用的证书私钥文件路径，如果为密文私钥，需要通过管道传入私钥密码，为json格式，ssl-keyfile-password字段。如：`echo {"ssl-keyfile-password":"password"} | gs_dbmind component cmd_exporter ...`。	-
--ssl-certfile	HTTPS协议使用的证书文件路径。	-
--ssl-ca-file	HTTPS协议使用的CA证书文件路径。	-
--web.listen-address	该exporter服务的绑定IP。	-
--web.listen-port	该exporter服务的侦听端口。	1024-65535。
--config	显性指定的待采集指标配置文件路径。	默认是该功能yamls目录下的default.yml文件，可以参考该配置文件格式，错误配置会报错。
--log.filepath	日志文件保存路径，默认保存在当前目录下。	-
--log.level	日志文件的打印级别，默认为INFO级别。	DEBUG、INFO、WARN、ERROR、FATAL。
--parallel	并行执行shell命令的并发度。	正整数。
--gs-log-dir	日志侦听路径，默认路径为“${GAUSSLOG}/gs_log”。	-
--disable-log-exporter	禁用日志采集功能。	-
--constant-labels	常量列表，k=v格式，用逗号隔开，表明该exporter自带的常量标签。	格式如“cluster_name=demo,cluster_id=1”。
--version	显示版本信息。	-

说明

当启用日志监测功能时，当监控的日志路径发生变动时，请重启cmd-exporter进程。

cmd-exporter进程对容器化环境部分支持。

版权所有 © 华为云计算技术有限公司

11.2.1.6 常见问题处理

1. 提示需要用户提供--ssl-keyfile与--ssl-certfile选项：

   上述exporter默认采用Https模式通信，因此需要用户指定证书及其私钥文件的路径。相反，如果用户只想采用Http模式，则需要显性指定--disable-https选项，从而禁用HTTPS协议。

2. 提示用户需要输入PEM密码（Enter PEM pass phrase）：

   如果用户采用Https模式，并给定了证书及其密钥文件的路径，且该密钥文件是经过加密的，则需要用户输入该加密私钥证书文件的密码。该密码也可以通过标准输入流传递。

   说明

   prometheus不支持私钥加密，若涉及prometheus通信的私钥加密时部分功能将无法实现。

   prometheus的query_range功能支持的最大序列长度是11000个数据，在使用时需要注意。

   prometheus的query_range功能可能会出现单点数据同时出现在连续两次查询结果中，此为正常现象。

11.3 DBMind 的 AI 子功能

用户可以通过gs_dbmind的component子命令启动对应的AI子功能，下述章节展示不同AI功能的具体内容和使用详情。

DBMind进程启动后，其提供的HTTP(S)接口信息如下：

版权所有 © 华为云计算技术有限公司

表 11-29 DBMind Restful API 列表

API	请求方法	功能描述	参数	返回值	示例
/v1/api/token	POST	获取数据库访问令牌。	• param username：必选，用户名。 • param grant_type：可选，授权类型。 • param password：必选，密码。 • param scope：必选，数据库实例。 • param client_id：可选，客户端id。 • param client_secret：可选，客户端密钥。	返回token。	{"access_token":"qmWWgt28VSu0YAH5","token_type":"bearer","expires_in":600}
/v1/api/agents	GET	获取所有代理列表。	无	返回所有代理列表。	{"data":{"ip1":["ip1","ip2","ip3"]}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/agents	PUT	更新代理信息（包含强制更新） **说明** 由于数据库实例状态的更改需要一段时间，因此强制更新默认延迟15秒执行。 更新期间对其他接口没有阻塞，但是由于agent信息未更新，所以等待期间得到的信息可能是更新之前的旧信息	param force：可选，强制更新。	返回更新代理结果。	{"data":true,"success":true}
/v1/api/configs/dynamic-config	PUT	设置DBMind配置。	• param config：必选，配置项分类。 • param name：必选，配置项名称。 • param value：必选，配置项取值。	返回更新配置结果。	{"data":"success","success":true}
/v1/api/configs/dynamic-config	GET	获取DBMind配置。	• param config：必选，配置项分类。 • param name：必选，配置项名称。	返回配置信息。	{"data":"50","success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/configs/dynamic-config/list	GET	获取DBMind配置列表。	param is_default：可选，默认为false。取值为false时从元数据库中获取配置信息；否则从本地配置文件中获取配置信息。	返回配置信息列表。	{"data":{"dynamic":{"detection_params":[["esd_test_alpha","0.05","The Significance level."]...]}...},"success":true}
/v1/api/configs/anomaly-detection/default-settings	GET	获取异常检测默认设置。	无	返回异常检测默认设置信息。	{"data":{"AlarmInfo":{"alarm_cause":null,"alarm_content":null,...,"window":10}},"success":true}
/v1/api/status/schedulers	GET	获取定时任务状态更新。	无	返回定时任务的状态。	{"data":{"header":["name","current_status","running_interval"],"rows":[["anomaly_detection","Running",180]...]},"success":true}
/v1/api/status/collection-system	GET	获取流水采集状态。	• param exporter_type：必选，exporter类型，取值范围：('opengauss_exporter','cmd_exporter','reprocessing_exporter')。 • param ssl_context：必选，SSL上下文信息。	返回采集器状态。	{"data":{"header":["component","listen_address","is_alive"],"rows":[["opengauss_exporter","ip",true]...],"suggestions":[]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/status/data-directory	GET	获取数据库数据目录状态。	• param instance：可选，实例节点的地址。 • param latest_minutes：可选，接口会统计从请求时间往前推latest_minutes分钟的数据，默认为5。	返回数据库目录状态。	{"data":{"free_space":2161.57,"tilt_rate":0.03,"total_space":3298.17...},"success":true}
/v1/api/status/overview	GET	获取数据库概览信息。	param latest_minutes：可选，接口会统计从请求时间往前推latest_minutes分钟的数据，默认为3。	返回数据库概览信息。	{"data":{"deployment_mode":"centralized",...,"strength_version":"GaussDB 2.1.0..."},"success":true}
/v1/api/status/instances	GET	获取数据库实例状态。	无	返回数据库实例状态。	{"data":{"header":["instance","role","state"],"rows":[["ip:port,","primary",fawulse]...]},"success":true}
/v1/api/status/agents	GET	获取数据库代理状态。	无	返回数据库代理状态。	{"data":{"agent_address":"ip","status":true},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/slow-sql-rca	GET	获取慢SQL的根因分析结果。	• param query：必选，SQL查询语句，字符串长度需小于10240。 • param db_name：必选，数据库名称。 • param schema_name：可选，schema名称，默认为public。 • param start_time：可选，13位时间戳，建议传递，开始时间。 • param finish_time：可选，13位时间戳，建议传递，结束时间。 • param template_id：可选，建议传递，归一化SQL ID。 • param debug_query_id：可选，建议传递，唯一SQL ID。 • param n_soft_parse：可选，软解析次数。	返回慢SQL的根因分析结果。	{"data": [true,"public","Seq Scan on t2 (cost=0.00..3.25 rows=1 width=70)\n Filter: (c1 = 'dadsadasds'::text)\n",[[["1. MISSING_INDEXES: (0.53) Missing required index.","2. HEAVY_SCAN_OPERATOR: (0.47) Existing expensive seq scans. Detail: (name: Seq Scan on t2, parent: None, rows:t2(100), cost rate: 100.0%)"]],[["1. Recommended index: (schema: public, index: t2(c1))."," 2. According to business adjustments, try to avoid it"]]]],"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			• param n_hard_parse：可选，硬解析次数。 • param query_plan：可选，执行计划。 • param n_returned_rows：可选，结果中元组个数。 • param n_tuples_fetched：可选，随机扫描行数。 • param n_tuples_returned：可选，顺序扫描行数。 • param n_tuples_inserted：可选，插入行数。 • param n_tuples_updated：可选，更新行数。 • param n_tuples_deleted：可选，删除行数。 • param n_blocks_fetched：可选，buffer的块访问次数。		**说明** 详细参见**注意事项说明**。 • 当用户传入归一化QUERY语句时，内部会使用PBE方式获取执行计划，如果获取失败则不会返回执行计划内容。 • 如果用户执行诊断的用户权限不足，则无法正常返回结果。 • 接口响应时长受数据库负载和资源影响。 • 建议传入query_plan参数，避免内部获取。 • 如果传入的QUERY存在截断且没有传入query_plan参数，则无法进行诊断。 • 接口只支持一条QUERY语句，如果传入多条则只会诊断第一条QUERY。 • 如果用户不传schema信息，则schema默认为public。 • 当前只支持DML语句。

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			• param n_blocks_hit：可选，buffer的块命中次数。 • param db_time：可选，有效的DB时间花费，单位：微秒。 • param cpu_time：可选，CPU时间，单位：微秒。 • param parse_time：可选，解析时间，单位：微秒。 • param plan_time：可选，执行时间，单位：微秒。 • param data_io_time：可选，IO时间，单位：微秒。 • param hash_spill_count：可选，hash过程中，若发生落盘，写文件的次数。 • param sort_spill_count：可选，sort过程中，若发生		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			落盘，写文件的次数。 ● param n_calls：可选，调用次数。 ● param lock_wait_time：可选，加锁等待耗时。 ● param lwlock_wait_time：可选，轻量级加锁时间。 ● param tz: 可选，若存在时区差异则建议传递，慢SQL时区信息，只支持UTC标准。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/correlation	GET	获取性能指标相关性分析结果。	• param metric_name：必选，metric名称。 • param instance：必选，实例地址。 • param start_time：必选，开始时间的时间戳。 • param end_time：必选，结束时间的时间戳。 • 注：start_time和end_time是两个独立参数，并非为选取时间窗口，所以不强制要求start_time早于end_time。 • param metric_filter：可选，metric的标签。 支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式。 – “{'k1':'v1','k2':'v2	返回异常关联指标及其时序数据。	{"data": {"os_mem_usage from ip": [["gs_total_memory_detail_mbytes{\"nodename\": \"dn_6001_6002\", \"type\": \"dynamic_used_memory\"} from ip",1.0,0, [0.32950820234487...]]...]},"success":true}

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			'}”的json字符串格式。		
/v1/api/app/memory-check	GET	获取内存占用情况。	param latest_hours：可选，默认为0。	返回当前数据库实例内存使用情况及其可能的风险。	{"data": {"timestamps": [1684838536049],"values": [4618.0]},"remark":"too little data...","status":"unknown"},...,"topk_session_memory_sql": [{"application_name": "JobScheduler"...}...]}, "success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/risk-analysis/{metric}	GET	获取风险分析结果。	• param metric：必选，metric名称。 • param instance：必选，实例地址。 • param warning_hours：可选，风险分析时长。 • param upper：可选，metric的上限。 • param lower：可选，metric的下限。 • param labels：可选，metric的标签。支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式。 – “{'k1':'v1','k2':'v2'}”的json字符串格式。 – param tz：可选，时区参数，只支持UTC标准。	返回当前指标未来风险分析结果。	{"data":{},"success":true} **说明** 1. 该接口会基于历史数据对指标未来变化趋势进行预测，预测时会取三倍于warning_hours的历史数据作为训练数据，以预测未来metric的变化趋势，因此如果warning_hours取值过小导致历史数据不足，可能会导致训练数据过少而无法进行训练，最终无正常结果输出，另外大于等于48小时的预测结果可能会不准确。 2. 该接口适合预测具有趋势和周期变化的数据。 3. 当该接口中没有传递labels参数进行过滤时，如果满足条件的指标过多可能会导致接口超时。 4. 该预测功能基于AI模型，因此每次预测结果可能存在部分差异。

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例

/v1/api/app/anomaly-detection/detectors/{name}	PUT	添加新的异常检测检测器或修改当前检测器。	param name：必选，检测器的名称。
			需要传入json结构体:
			{
			"duration": 10,
			"forecasting_seconds": 0,
			"alarm_info": {
			"alarm_type": "SYSTEM",
			"alarm_level": "ERROR"
			},
			"detector_info": [
			{
			"detector_name": "EsdTestDetector",
			"metric_name": "gaussdb_blks_hit_ratio",
			"detector_kwargs": {
			"alpha": 0.05
			}
			}
			]
			}
			detector_info字段下检测算法的可选范围及对应检测参数，见表格下方说明。
				返回添加状态:
				• 成功返回Success: add {检测器名称} for {集群IP}
				• 失败返回Failed: add {检测器
					{"data":"Success: add detector for IP","success":true}

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
				名称}for{集群IP}	
/v1/api/app/anomaly-detection/detectors/{name}	DELETE	删除指定的异常检测检测器。	param name：必选，检测器的名称。	返回删除状态。	{"data":"Success: delete detector for IP","success":true}
/v1/api/app/anomaly-detection/detectors/{name}/pause	PUT	暂停指定的异常检测检测器。	param name：必选，检测器的名称。	返回暂停状态。	{"data":"Success: pause detector for IP","success":true}
/v1/api/app/anomaly-detection/detectors/{name}/resume	PUT	继续指定的异常检测检测器。	param name：必选，检测器的名称。	返回继续状态。	{"data":"Success: resume detector for IP","success":true}
/v1/api/app/anomaly-detection/detectors/{name}	GET	查看指定的异常检测检测器信息。	param name：必选，检测器的名称。	返回异常检测器信息。	{"data":{"alarm_info":{"alarm_cause":null,"alarm_content":null,"alarm_level":"ERROR","alarm_type":"SYSTEM","extra":null},"detector_info":[...],"duration":10,"forecasting_seconds":0,"running":1},"success":true}
/v1/api/app/anomaly-detection/detectors/all/rebuild	PUT	根据后端落盘信息重建所有异常检测检测器。	-	返回重建状态。	{"data":"Success: rebuild detectors for IP","success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/anomaly-detection/detectors	DELETE	清空所有异常检测检测器。	-	返回删除状态。	{"data":"Success: clear detectors for IP","success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/workload-collection	GET	获取数据库工作负载数据。	• param data_source：必选，数据源，取值范围：('asp', 'statement_history', 'pg_stat_activity')。 • param databases：可选，数据库列表。 • param schemas：可选，schema列表。 • param start_time：可选，开始时间。 • param end_time：可选，结束时间，仅适用于asp和statement_history。 • param db_users：可选，数据库用户列表。 • param sql_types：可选，SQL语句类型，取值范围：('SELECT', 'UPDATE', 'DELETE', 'INSERT')。	返回工作负载信息。	{"data":{"header":["user_name","db_name","schema_name","application_name","unique_query_id","start_time","finish_time","duration","n_returned_rows","n_tuples_fetched","n_tuples_returned","n_tuples_inserted","n_tuples_updated","n_tuples_deleted","n_blocks_fetched","n_blocks_hit","n_soft_parse","n_hard_parse","db_time","cpu_time","parse_time","plan_time","data_io_time","lock_wait_time","lwlock_wait_time","query"],"rows":[["user1","db1","public"...]...]},"success":true} **说明** • data_source参数数据源解释：asp为采样数据，statement_history为历史慢SQL数据，慢SQL阈值由用户指定，pg_stat_activity为实时SQL数据。 • 为避免接口超时，返回数据量上限1000条。

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			• param template_id：可选，SQL模板id。 • param duration：可选，SQL语句的执行时间，单位ms，仅适用于statement_history和pg_stat_activity。		
/v1/api/app/cluster-diagnosis	GET	获取故障诊断结果。	• param instance：可选，被诊断的实例地址。 • param role：可选，实例的角色，取值范围：('cn', 'dn')。 • param method：可选，诊断模型使用的方法，取值范围：'logical', 'tree'。 • param timestamp：可选，诊断时间的时间戳，单位：时间戳（13位）。	返回故障诊断结果。	{"data": [{"bind_ip_failed":0,"cms_phonydead_restart":0,"cms_restart_pending":0,"dn_disk_damage":0,"dn_manual_stop":0,"dn_nic_down":0,"dn_ping_standby":0,"dn_port_conflict":0,"dn_read_only":0,"dn_status":0,"dn_writable":0,"ffic_updated":0,"ping":1},"DN down/disconnection"],"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/metrics	GET	获取所有指标列表。	无	返回TSDB上的所有指标列表。	{"data":["gaussdb_blks_hit_ratio","gaussdb_blks_read_rate"...],"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/metrics/{name}	GET	获取最新的指定名称序列的指标。	• param name：必选，指定metric的名称。 • param instance：可选，metric所属的实例地址。 • param latest_minutes：可选，指定的时间范围。 • param from_timestamp：可选，开始时间时间戳，适用于latest_minutes为空的情况。 • param to_timestamp：可选，结束时间时间戳，适用于latest_minutes为空的情况。 • param step：可选，获取数据的时间间隔，单位为毫秒。 • param fetch_all：可选，是否获取所有序	返回指定序列的详情。 如果prometheus短期内存入了大量数据，或者查询的区间过大，有可能导致查询超时，或由于内存不足导致查询失败。	{"data":[{"labels":{"device":"device","from_instance":"ip","from_job":"exporter","fstype":"ext4","instance":"ip:port","job":"exporter","mountpoint":"mounpoint"},"name":"os_disk_usage","timestamps":[1684900929939,1684900944939,1684900959939,1684900974939],"values":[0.7115459280083043,0.7115460512887748,0.7115461745692453,0.7115462978497158]}],"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			列或只获取一次。 ● param regex：可选，当instance非空，用正则表达式过滤instance。 ● param labels：可选，用指定标签过滤序列。支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式。 – “{'k1':'v1','k2':'v2'}”的json字符串格式。 ● param regex_labels：可选，使用正则表达式过滤序列。支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式。 – “{'k1':'v1','k2':'v2'}”的json字符串格式。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			• param limit：可选，限制返回结果行数。 • param tz：可选，时区参数，只支持UTC标准。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/metrics/{name}	DELETE	删除指定范围的指标。	• param name：必选，指定metric的名称。 • param instance：可选，metric所属的实例地址。 • param from_timestamp：可选，开始时间时间戳。 • param to_timestamp：可选，结束时间时间戳。 • param regex：可选，当instance非空，用正则表达式过滤instance。 • param labels：可选，用指定标签过滤序列。支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式。 – “{'k1':'v1','k2':'v2'}”的json字符串格式。	返回删除状态（from_timestamp和to_timestamp虽然支持毫秒粒度，但是实际删除时仅支持到秒粒度，所以会出现一些临近边界时间范围的值无法删除的情况）。	{"data":null,"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			• param regex_labels：可选，使用正则表达式过滤序列。支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式。 – “{'k1':'v1','k2':'v2'}”的json字符串格式。 • param flush：可选，删除指标数据后是否刷新磁盘。 • param tz：可选，时区参数，只支持UTC标准。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/alarms	GET	获取历史告警列表。	• param pagesize：可选，每页展示个数，建议传递。 • param current：可选，当前页，建议传递。 • param instance：可选，告警所属实例，如果不传递则返回当前集群内的告警。 • param alarm_type：可选，告警类型，参数可选范围为['SYSTEM', 'SLOW_QUERY', 'ALARM_LOG', 'ALARM', 'SECURITY', 'PERFORMANCE']。 • param alarm_level：可选，告警级别，参数可选范围为['CRITICAL', 'FATAL', 'ERROR', 'WARNING', 'WARN', 'INFO', 'NOTICE',	返回历史告警列表。	{"data":{"header":["history_alarm_id","instance","metric_name","metric_filter","alarm_type","alarm_level","start_at","end_at","alarm_content","extra_info","anomaly_type","alarm_cause"],"rows":[[65,"ip","os_mem_usage",null,"SYSTEM",30,1684762097001,1684762547001,"mem_usage_spike_detector:Find obvious spikes in memory usage.",null,"Spike"]]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			'DEBUG', 'NOTSET']。 ● param metric_name：可选，指标名称。 ● param start_at：可选，查询开始时间戳，建议传递。 ● param end_at：可选，查询结束时间戳，建议传递。 ● param anomaly_type：可选，异常类型。 ● param group：可选，是否根据告警类型和内容划分告警。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/alarms/count	GET	获取历史告警数量。	• param instance：可选，告警所属实例，如果不传递则返回当前集群内的告警。 • param alarm_type：可选，告警类型，参数可选范围为['SYSTEM', 'SLOW_QUERY', 'ALARM_LOG', 'ALARM', 'SECURITY', 'PERFORMANCE']。 • param alarm_level：可选，告警级别，参数可选范围为['CRITICAL', 'FATAL', 'ERROR', 'WARNING', 'WARN', 'INFO', 'NOTICE', 'DEBUG', 'NOTSET']。 • param metric_name：可选，指标名称。 • param start_at：可选，查询开始时间戳，不得晚于	返回历史告警数量。	{"data":30,"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			end_at，建议传递。 ● param end_at：可选，查询结束时间戳，不得早于start_at，建议传递。 ● param anomaly_type：可选，异常类型。 ● param group：可选，是否根据告警类型和内容划分告警。		

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/cluster-diagnosis	GET	获取历史集群诊断列表，集群诊断定时任务结果存储于元数据库中，若元数据库发生故障，结果无法插入表中，会导致部分数据遗漏。	• param pagesize：可选，每页展示个数，建议传递。 • param current：可选，当前页，建议传递。 • param instance：可选，实例IP，如果不传递则返回当前集群内的诊断结果。 • param instance_like：可选，实例IP模糊匹配，特性预埋字段，未实装。 • param start_at：可选，查询开始时间戳，仅返回timestamp大于等于该时间戳的结果，建议传递。 • param end_at：可选，查询结束时间戳，仅返回timestamp小于等于该时间戳的结果，建议传递。	返回历史集群诊断列表。	{"data":{"header":["diagnosis_id","instance","timestamp","cluster_role","diagnosis_method","cluster_feature","diagnosis_result","status_code","alarm_type","alarm_level"],"rows":[[1,ip,1695110107087,"cn","logical","{\"ping\": 0, \"cn_status\": 0, \"bind_ip_failed\": 0, \"panic\": 0, \"ffic_updated\": 0, \"cms_heartbeat_restart\": 0, \"cms_phonydead_restart\": 0, \"cn_dn_disconnected\": 0, \"cn_down_to_delete\": 0, \"cn_restart_time_exceed\": 0, \"cn_read_only\": 0, \"cn_restart\": 0, \"cn_start\": 0, \"cn_manual_stop\": 0, \"cn_disk_damage\": 0, \"cn_nic_down\": 0, \"cn_port_conflict\": 0}","Normal",-1,"ALARM",0]]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例

			• （若start_at和end_at都不传递，则不对timestamp列进行任何值的筛选，返回全量时间戳的结果）。
			• param cluster_role：可选，节点类型。
			• param diagnosis_method：可选，诊断算法。
			• param status_code：可选，状态码，特性预埋字段，未实装。
			• param alarm_type：可选，告警类型，参数可选范围为['SYSTEM', 'SLOW_QUERY', 'ALARM_LOG', 'ALARM', 'SECURITY', 'PERFORMANCE']。
			• param alarm_level：可选，告警级别，参数可选范围为
					说明
					• 返回结果中timestamp列所记录的时间戳以DBMind部署的服务器时间为准，如果元数据库所在服务器时间与DBMind部署服务器时间不一致，会导致元数据库中显示的时间戳结果与元数据库服务器实际时间不一致。
					• 返回结果按照以下两个规则排序：
					  • 按照timestamp从小到大排列；
					  • 当timestamp一致时按照diagnosis_id从小到大排列。

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			['CRITICAL', 'FATAL', 'ERROR', 'WARNING', 'WARN', 'INFO', 'NOTICE', 'DEBUG', 'NOTSET']。 ● param is_normal：可选，是否包含诊断结果为Normal的数据。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/cluster-diagnosis/count	GET	获取历史集群诊断数量，集群诊断定时任务结果存储于元数据库中，若元数据库发生故障，结果无法插入表中，会导致部分数据遗漏。	• param instance：可选，实例IP，如果不传递则返回当前集群内的诊断结果。 • param instance_like：可选，实例IP模糊匹配，特性预埋字段，未实装。 • param start_at：可选，建议传递，查询开始时间戳，仅返回timestamp大于等于该时间戳的结果，建议传递。 • param end_at：可选，建议传递，查询结束时间戳，仅返回timestamp小于等于该时间戳的结果，建议传递。 • （若start_at和end_at都不传递，则不对timestamp列进行任何值的筛选，返回全量时	返回历史集群诊断数量。	{"data":30,"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			间戳的结果）。 ● param cluster_role：可选，节点类型。 ● param diagnosis_method：可选，诊断算法。 ● param status_code：可选，状态码，特性预埋字段，未实装。 ● param alarm_type：可选，告警类型，参数可选范围为['SYSTEM', 'SLOW_QUERY', 'ALARM_LOG', 'ALARM', 'SECURITY', 'PERFORMANCE']。 ● param alarm_level：可选，告警级别，参数可选范围为['CRITICAL', 'FATAL', 'ERROR', 'WARNING', 'WARN', 'INFO', 'NOTICE', 'DEBUG', 'NOTSET']。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			• param is_normal：可选，是否包含诊断结果为Normal的数据。		
/v1/api/summary/sql/killed	GET	获取被终止的查询列表。	• param pagesize：可选，每页展示个数。 • param current：可选，当前页。 • param instance：可选，慢SQL所属实例。 • param query：可选，慢SQL文本。 • param start_time：可选，获取慢SQL的开始时间，单位：时间戳（13位）。 • param end_time：可选，获取慢SQL的结束时间，单位：时间戳（13位）。	返回被终止的查询列表。	{"data":{"header":["instance","schema_name","db_name","query","template_id","hit_rate","fetch_rate","cpu_time","data_io_time","parse_time","plan_time","db_time","root_cause","suggestion","start_at","duration_time"],"rows":[(ip1, 'public', 'db1', 'select 1', t_id, ...), (...), ...]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/sql/killed/count	GET	获取被终止的查询数量。	• param instance：可选，慢SQL所属实例。 • param query：可选，慢SQL文本。 • param start_time：可选，获取慢SQL的开始时间，单位：时间戳（13位）。 • param end_time：可选，获取慢SQL的结束时间，单位：时间戳（13位）。	返回被终止的查询数量。	{"data":0,"success":true}
/v1/api/summary/sql/top	GET	获取执行最频繁的前十个SQL模板。	无	返回执行最频繁的前十个SQL模板。	{"data":{"header":["user_name","unique_sql_id","query","n_calls","min_elapse_time","max_elapse_time","avg_elapse_time","n_returned_rows","db_time","cpu_time","execution_time","parse_time","last_updated","sort_spill_count","hash_spill_count"],"rows":[["user",1961954918,"vacuum;",1,38958701,38958701,38958701,"3891978 1.218781218781",0,38959021...]]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/sql/active	GET	获取当前活跃的SQL列表。	无	返回当前活跃的SQL列表。	{"data":{"header":["datname","username","application_name","client_addr","query_start","waiting","state","query","connection_info"], "rows":[["data1","db1","app1","ip:port","time"...]]}, "success":true}
/v1/api/summary/sql/locking	GET	获取当前锁等待的SQL列表。	无	返回当前锁等待的SQL列表。	{"data":{"header":null,"rows":[]},"success":true}
/v1/api/app/index-recommendation	POST	获取建议的索引列表。	• param instance：必选，数据库实例。 • param database：可选，数据库名称。 • param max_index_num：可选，推荐索引个数的最大值。 • param max_index_storage：可选，推荐索引的内存使用的最大值。 • param sqls：必选，SQL语句列表。	返回建议的索引列表。	{"data":[{"advise_indexes":[],"redundant_indexes":[],"total":0,"useless_indexes":[{"columns":"c1","schemaName":"public","statement":"DROP INDEX t1_c1_idx;","tbName":"t1","type":3},{"columns":"c2","schemaName":"public","statement":"DROP INDEX t2_c2_idx;","tbName":"t2","type":3}]},{}],"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/knob-recommendation/snapshots	GET	获取指标快照。	• param pagesize：可选，每页展示个数。 • param current：可选，当前页。	返回指标快照。	{"data":{"header":["instance","metric","value"],"rows":[]},"success":true}
/v1/api/summary/knob-recommendation/snapshots/count	GET	获取快照数量。	无	返回快照数量。	{"data":0,"success":true}
/v1/api/summary/knob-recommendation/warnings	GET	获取指标报警值。	• param pagesize：可选，每页展示个数。 • param current：可选，当前页。	返回指标报警值。	{"data":{"header":["instance","level","comment"],"rows":[]},"success":true}
/v1/api/summary/knob-recommendation/warnings/count	GET	获取指标报警值数量。	无	返回指标报警值数量。	{"data":0,"success":true}
/v1/api/summary/knob-recommendation/details	GET	获取参数推荐列表。	• param pagesize：可选，每页展示个数。 • param current：可选，当前页。	返回参数推荐列表。	{"data":{"header":["instance","name","current","recommend","min","max"],"rows":[]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/knob-recommendation/details/count	GET	获取参数推荐数量。	无	返回参数推荐数量。	{"data":0,"success":true}
/v1/api/summary/sql/slow	GET	获取慢查询诊断统计信息。	无	返回慢查询诊断统计信息。	{"data":{"distribution":{"delete":0,"insert":0,"select":0,"update":0},"main_slow_queries":0,"mean_buffer_hit_rate":-1,"mean_cpu_time":-1,"mean_fetch_rate":-1,"mean_io_time":-1,"nb_unique_slow_queries":0,"slow_query_count":{"timestamps":[],"values":[]},"slow_query_template":{"header":["template_id","count","query"],"rows":[]},"slow_query_threshold":2.0,"statistics_for_database":{},"statistics_for_schema":{},"systable":{"business_table":0,"system_table":0}},"success":true}
/v1/api/summary/regular-inspection	GET	获取定期巡检结果列表。	param inspection_type：必选，检测类型，取值范围：('daily_check', 'weekly_check', 'monthly_check')。	返回定期巡检结果列表。	{"data":{"header":["instance","report","start","end"],"rows":[["ip:port",{"connection":{"active_connection":{"avg":3.0,"max":4.0,"min":1.0,"the_95th":4.0},"total_connection":{"avg":3.2321,"max":5.0,"min":1.0,"the_95th":4.0}}...},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/regular-inspection/count	GET	获取定期巡检结果数量。	param inspection_type：必选，检测类型，取值范围：('daily_check', 'weekly_check', 'monthly_check')。	返回定期巡检结果数量。	{"data":9,"success":true}
/v1/api/summary/database-list	GET	获取数据库列表。	无	返回数据库列表。	{"data":["db1","db2","db3","db4","db5"...],"success":true}
/v1/api/summary/sql/slow/latest	GET	获取最近的慢查询列表。	• param instance：可选，慢SQL所属的实例地址。 • param query：可选，query文本。 • param start_time：可选，慢SQL开始运行时间，单位：时间戳（13位）。 • param end_time：可选，慢SQL结束运行时间，单位：时间戳（13位）。 • param group：可选，查询时是否使用group逻辑。	返回最近的慢查询列表。	{"data":{"header":["instance","schema_name","db_name","query","template_id","hit_rate","fetch_rate","cpu_time","data_io_time","parse_time","plan_time","db_time","root_cause","suggestion","start_at","duration_time"],"rows":[(ip1, 'public', 'db1', 'select 1', t_id, ...), (...), ...]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/sql/slow/latest/count	GET	获取最近的慢查询数量。	• param pagesize：可选，查询页返回数量。 • param current：可选，当前页数。 • param instance：可选，慢SQL所属的实例地址。 • param distinct：可选，查询时是否使用distinct逻辑。 • param query：可选，query文本。 • param start_time：可选，慢SQL开始运行时间，单位：时间戳（13位）。 • param end_time：可选，慢SQL结束运行时间，单位：时间戳（13位）。 • param group：可选，查询时是否使用group逻辑。	返回最近的慢查询数量。	{"data":0,"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/metric-diagnosis	GET	执行指标异常诊断，当前版本不可用。	• param metric_name：必选，指标名称。取值范围：os_cpu_user_usage，pg_thread_pool_rate，os_mem_usage，os_disk_usage，gaussdb_mount_usage，os_disk_await，xlog_margin，pg_long_transaction_count。 • param metric_filter：必选，筛选指标。支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式； – “{'k1':'v1','k2':'v2'}”的json字符串格式。 • param alarm_cause：必选，选择分析方法。取值范围：high_cpu_us	返回指标异常根因。	{"data":{[{'reason1': 0.0, 'reason2': 1.0}, 'conclusion', 'advice']},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			age, high_thread_pool_rate, high_dynamic_mem_usage, high_shared_mem_usage, high_disk_usage, high_io_delay，mem_leak，high_xlog_count，long_transaction。 ● param start：可选，分析指标开始时间戳，单位毫秒。 ● param end：可选，分析指标结束时间戳，单位毫秒。		

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/real-time-inspection	POST	执行智能巡检功能。	• param inspection_type：必选，巡检类型，取值范围：('real_time_check', 'daily_check', 'weekly_check', 'monthly_check')。 • param start_time：可选，起始时间。 • param end_time：可选，终止时间。 • param instance：必选，实例IP:PORT。 • param inspection_items：必选，巡检项。 • param tz：可选，时区参数，只支持UTC标准。	返回巡检结果。	{ "system_resource": { "os_mem_usage": { "192.168.0.1": { "statistic": { "max": 0.4154, "min": 0.4128, "avg": 0.4149, "the_95th": 0.4153 }, "warnings": { "increase_warning": false, "threshold_warning": [], "forecast_warning": { "occur_time": "", "remaining_hours": 0.0, "risk": "", "timestamps": [], "values": [] } }, "timestamps": [1689210000000, 1689210360000], "data": [0.41450417776587234, 0.41450179472976234] }, "192.168.0.2": { xxx } } },

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
					"instance_status": {}, "database_resource": {}, "database_performance": {}, "diagnosis_optimization": {}, "conclusion": {} }
/v1/api/app/real-time-inspection/list	GET	展示巡检任务的基础信息。	param instance：必选，实例IP:PORT。	返回巡检任务的基础信息。	{"data":{"header":["instance","start","end","id","state","cost_time","inspection_type"],"rows":[["192.168.0.1:8080",1689210000000,1689296400000,5,"success",0.033701,"real_time_check"]]},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/summary/real-time-inspection	GET	获取指定巡检任务的巡检结果。	• param instance：必选，实例IP:PORT。 • param spec_id：必选，巡检任务ID。	返回指定巡检任务的巡检结果。	{ "system_resource": { "os_mem_usage": { "192.168.0.1": { "statistic": { "max": 0.4154, "min": 0.4128, "avg": 0.4149, "the_95th": 0.4153 }, "warnings": { "increase_warning": false, "threshold_warning": [], "forecast_warning": { "occur_time": "", "remaining_hours": 0.0, "risk": "", "timestamps": [], "values": [] } }, "timestamps": [1689210000000, 1689210360000], "data": [0.41450417776587234, 0.41450179472976234] }, "192.168.0.2": { xxx } } },

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
					"instance_status": {}, "database_resource": {}, "database_performance": {}, "diagnosis_optimization": {}, "conclusion": {} }
/v1/api/app/real-time-inspection	DELETE	删除指定的巡检任务。	• param instance：必选，实例IP:PORT。 • param spec_id：必选，巡检任务ID。	返回删除状态。	{"data": {"success":true},"success":true}
/v1/api/check-status	POST	获取DBMind服务的状态信息并返回状态详情。	param cmd：可选，服务启动命令。	返回状态详情。	{"data": {"error_msg":"","result": {},"state":"NORMAL"},"success":true}
/v1/api/repair	POST	修复DBMind服务并返回修复结果。	param cmd：可选，服务启动命令。	返回修复结果。	{"data": {"error_msg":"","result": {},"state":"SUCCESS"},"success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例

/v1/api/summary/sql-trace	GET	SQL全链路查询。	• param instance_id：必选，实例id。
			• param labels：必选，过滤使用的指标标签。
			• param is_online：必选，数据来源是否是在线的。
			• param from_timestamp：可选，查询开始时间。
			• param to_timestamp：可选，查询结束时间。
			• param tz：可选，时区参数，仅支持UTC标准。
				返回SQL全链路信息。	{
					"data": [
					{
					"all_time": 3477357,
					"application_name": "xxx",
					"client_addr": "",
					"client_port": 0,
					"component_id": "xxx",
					"db_name": "xxx",
					"execution_time_details": {
					"kernel_time": {
					"all_time": 3477357,
					"kernel_time_details": {
					"execution_time": 3466386,
					"other_time": 4958,
					"parse_time": 453,
					"plan_time": 5244,
					"rewrite_time": 316
					}
					},
					"resource_time": {
					"all_time": 3477357,
					"resource_time_details": {
					"cpu_time": 98756,
					"data_io_time": 0,
					"other_time": 3378601
					}
					},
					"wait_event_time": {
					"code_wait_event_time": {

 版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例

					"all_time": 3477357,
					"code_wait_event_time_details": {
					"events": [
					{
					"event_name": "xxx",
					"event_time": 16810
					},
					{
					"event_name": "xxx",
					"event_time": 5330
					},
					{
					"event_name": "xxx",
					"event_time": 5195
					},
					{
					"event_name": "xxx",
					"event_time": 197
					}
					],
					"left_time": 0,
					"other_time": 3449825
					}
					},
					"resource_wait_event_time": {
					"all_time": 3477357,
					"other_time": 3477357,
					"resource_wait_event_time_details": {
					"data_io_time": {
					"all_time": 0,
					"data_io_time_details": {
					"events": [

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
					{ "event_name": "xxx", "event_time": 742 }], "left_time": 0, "other_time": -27532 } }, "lock_time": { "all_time": 0, "lock_time_details": { "events": [], "left_time": 0, "other_time": 0 } }, "lwlock_time": { "all_time": 0, "lwlock_time_details": { "events": [], "left_time": 0, "other_time": 0 } } } } } }, "finish_time": "2023-08-01 17:50:04.923701+08", "node_id": "0", "schema_name": "user,public", "session_id": 1658590,

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
					"sql_exec_id": 72339069024193250, "sql_id": 1951225884, "start_time": "2023-08-01 17:50:01.446446+08", "trace_id": "0", "transaction_id": "0", "user_name": "user" }], "success": true }
/v1/api/summary/metric-unit	GET	指标单位查询。	param metric：必选，指标名。	返回**指标单位**。	{"data":"{'en': 'rate', 'cn': '比率'}","success":true}
/v1/api/security/scenarios	GET	自安全，当前版本不可用。	无	返回所有安全异常类型。	{"data": ["scanning_attack", "brute_force_login_attack","user_violation_rate"],"success":true}
/v1/api/security/scenarios/{name}	GET	自安全，当前版本不可用。	param name：必选，安全指标名称，参数可选范围为：['scanning_attack', 'brute_force_login_attack', 'user_violation_rate']。	返回安全异常包含的指标的校准状态。	{"data": [{"metric":"gaussdb_log_errors_rate","status":"CALIBRATED"}, {"metric":"gaussdb_user_violation_rate","status":"CALIBRATED"}], "success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
/v1/api/app/metric-diagnosis-insight	GET	执行指标异常诊断，当前版本不可用。	• param metric_name：必选，指标名称。取值范围：os_cpu_user_usage，pg_thread_pool_rate，os_mem_usage，os_disk_usage，gaussdb_mount_usage，os_disk_await，xlog_margin，pg_long_transaction_count。 • param metric_filter：必选，筛选指标。支持如下两种格式： – “k1=v1,k2=v2”的kv pairs格式； – “{'k1':'v1','k2':'v2'}”的json字符串格式。 • param alarm_cause：必选，选择分析方法。取值范围：high_cpu_us	返回指标异常根因。	{"data":{"insight_1":[{"k1":v1,"k2":v2,...},{"k1":v1,"k2":v2,...},...],"insight_2":[{"k1":v1,"k2":v2,...},{"k1":v1,"k2":v2,...},...]},..."success":true}

版权所有 © 华为云计算技术有限公司

API	请求方法	功能描述	参数	返回值	示例
			age, high_thread_pool_rate, high_dynamic_mem_usage, high_shared_mem_usage, high_disk_usage, high_io_delay，mem_leak，high_xlog_count，long_transaction。 ● param start：可选，分析指标开始时间戳，单位毫秒。 ● param end：可选，分析指标结束时间戳，单位毫秒。		

 版权所有 © 华为云计算技术有限公司

说明

对异常检测器检测算法的说明如下，也可以通过调用/v1/api/configs/anomaly-detection/default-settings获得：

1. 学生化残差检测器-EsdTestDetector：{"alpha":0.05}。
2. 梯度检测器-GradientDetector：{"max_coef":1,"side":["positive",["positive","negative","both"]]}。
3. 单调趋势检测器-IncreaseDetector：{"alpha":null,"side":["positive",["positive","negative","both"]]}。
4. 四分位间距检测器-InterQuartileRangeDetector：{"outliers":[3,3]}。
5. 均值漂移检测器-LevelShiftDetector：{"agg":["median",["median","mean","std"]],"outliers":[null,6], "side":["both",["positive","negative","both"]],"window":5}。
6. 分位数检测器-QuantileDetector：{"high":1,"low":0}。
7. 周期数据检测器-SeasonalDetector：{"high_ac_threshold":0.1,"min_seasonal_freq":2,"outliers":[null,3], "period":null,"side":["positive,["positive","negative","both"]],"window":10}"。
8. 尖峰检测器-SpikeDetector：{"agg":["median",["median","mean","std"]],"outliers":[null,3], "side":["both",["positive","negative","both"]],"window":1}。
9. 阈值检测器-ThresholdDetector：{"high":null,"low":null,"percentage":[null,[0,1]]}。
10. 波动率漂移检测器-VolatilityShiftDetector：{"agg":["std",["median","mean","std"]],"outliers":[null,6], "side":["both",["positive","negative","both"]],"window":10}。

说明

- alarm_level参数的所有级别及其对应的权重：CRITICAL、FATAL：50，ERROR：40，WARNING、WARN：30，INFO、NOTICE：20，DEBUG：10，NOTSET：0。
- alarm_type参数的所有类型：SYSTEM、SLOW_QUERY、ALARM_LOG、ALARM、SECURITY、PERFORMANCE。
- anomaly_type参数的所有类型：high_disk_usage_detector、high_xlog_count_detector、high_mem_usage_detector、mem_leak_detector、session_mem_increase_detector、shared_mem_increase_detector、other_mem_increase_detector、high_cpu_usage_detector、high_thread_pool_rate_detector、high_io_delay_detector、slow_disk_detector、disk_io_jam_detector、lag_detector、packet_loss_detector、leaked_fd_detector、slow_sql_detector、deadlock_detector、core_detector。

指标单位查询接口

/v1/api/summary/metric-unit/{metric} 支持的指标列表如下表所示：

qps	os_disk_io_read_delay	node_network_net_dev_group	node_memory_VmallocTotal_bytes
tps	pg_audit_logout_total	node_network_protocol_type	node_memory_WritebackTmp_bytes
lsn_margin	pg_audit_revoke_total	node_sockstat_FRAG6_memory	node_netstat_Udp6_OutDatagrams
node_load1	pg_cpu_load_total_cpu	node_sockstat_sockets_used	node_netstat_Udp6_RcvbufErrors
node_load5	pg_db_confl_bufferpin	node_softnet_dropped_total	node_netstat_Udp6_SndbufErrors

版权所有 © 华为云计算技术有限公司

node_load15	pg_index_idx_blks_hit	node_textfile_scrape_error	node_netstat_UdpLite6_InErrors
xlog_margin	pg_redo_stat_writetim	node_timex_pps_error_total	node_rapl_package_joules_total
node_os_info	gaussdb_blks_hit_ratio	pg_audit_user_locked_total	node_timex_pps_frequency_hertz
os_disk_iops	gaussdb_blks_read_rate	pg_connections_enqueue_sql	node_timex_pps_stability_hertz
os_mem_usage	gaussdb_conflicts_rate	pg_session_memory_usedsize	pg_checkpoint_full_page_writes
gs_time_value	gaussdb_deadlocks_rate	process_start_time_seconds	pg_lock_time_info_holding_time
load_average1	gaussdb_log_gtm_status	gaussdb_confl_deadlock_rate	pg_recovery_status_current_rto
node_dmi_info	gaussdb_log_node_start	gaussdb_confl_snapshot_rate	pg_replication_slots_delay_lsn
os_cpu_iowait	gs_checkpoint_next_oid	gaussdb_index_increase_rate	pg_stat_bgwriter_total_seconds
os_disk_await	gs_checkpoint_redo_lsn	gaussdb_invalid_logins_rate	pg_stat_replication_flush_diff
os_disk_usage	gs_database_size_bytes	gaussdb_log_dn_ping_standby	pg_stat_replication_replay_lag
pg_time_value	gs_db_confl_tablespace	gaussdb_table_increase_rate	pg_stat_replication_replay_lsn
gs_db_blks_hit	gs_index_idx_blks_read	gaussdb_user_violation_rate	pg_stat_replication_write_diff
node_disk_info	gs_redo_stat_phyblkwrt	gs_connections_idle_session	pg_summary_user_logout_counter
pg_db_blks_hit	gs_sql_count_mergeinto	gs_global_ckpt_status_value	pg_tables_structure_n_dead_tup
gs_class_relage	gs_stat_activity_count	gs_memory_context_totalsize	pg_tables_structure_n_live_tup
gs_db_blks_read	gs_tables_size_relsize	gs_session_memory_totalsize	pg_tables_structure_n_tup_read
gs_db_conflicts	node_boot_time_seconds	gs_sql_statement_full_count	pg_wait_events_total_wait_time
gs_db_deadlocks	node_cpu_seconds_total	gs_stat_get_wal_senders_pid	gaussdb_log_cms_restart_pending

版权所有 © 华为云计算技术有限公司

gs_index_mbytes	node_memory_Slab_bytes	node_context_switches_total	gaussdb_log_recycle_auxilary_db
gs_locker_count	node_network_device_id	node_entropy_available_bits	gs_checkpoint_next_multi_offset
node_intr_total	node_network_mtu_bytes	node_entropy_pool_size_bits	gs_checkpoint_next_multixact_id
node_network_up	node_timex_sync_status	node_filesystem_avail_bytes	gs_checkpoint_oldest_active_xid
node_os_version	node_vmstat_pgmajfault	node_hwmon_temp_max_celsius	gs_stat_replication_replay_diff
node_udp_queues	os_disk_io_write_bytes	node_memory_AnonPages_bytes	gs_tables_structure_last_vacuum
node_uname_info	os_disk_io_write_delay	node_memory_HugePages_Total	node_disk_io_time_seconds_total
os_disk_ioutils	pg_checkpoint_next_oid	node_memory_SwapTotal_bytes	node_disk_reads_completed_total
pg_class_relage	pg_checkpoint_redo_lsn	node_memory_Writeback_bytes	node_memory_AnonHugePages_bytes
pg_db_blks_read	pg_database_size_bytes	node_netstat_Icmp6_InErrors	node_memory_Inactive_anon_bytes
pg_db_conflicts	pg_db_confl_tablespace	node_netstat_IpExt_InOctets	node_memory_Inactive_file_bytes
pg_db_deadlocks	pg_index_idx_blks_read	node_rapl_dram_joules_total	node_netstat_TcpExt_ListenDrops
pg_index_mbytes	pg_redo_stat_phyblkwrt	node_sockstat_TCP_mem_bytes	node_netstat_TcpExt_TCPTimeouts
pg_locker_count	pg_sql_count_mergeinto	node_sockstat_UDP_mem_bytes	node_network_receive_drop_total
process_max_fds	pg_stat_activity_count	node_sockstat_UDPLITE_inuse	node_network_receive_errs_total
gaussdb_cpu_time	pg_tables_size_relsize	node_timex_maxerror_seconds	node_network_receive_fifo_total
gaussdb_log_ffic	replication_lsn_margin	node_timex_pps_jitter_total	node_nf_conntrack_entries_limit
gaussdb_ping_lag	scrape_samples_scraped	pg_connections_idle_session	node_schedstat_timeslices_total
gs_class_relsize	gaussdb_confl_lock_rate	pg_global_ckpt_status_value	pg_checkpoint_next_multi_offset

版权所有 © 华为云计算技术有限公司

gs_db_confl_lock	gaussdb_corrosion_index	pg_memory_context_totalsize	pg_checkpoint_next_multixact_id
gs_db_temp_bytes	gaussdb_idle_connection	pg_session_memory_totalsize	pg_checkpoint_oldest_active_xid
gs_db_temp_files	gaussdb_log_cms_cn_down	pg_sql_statement_full_count	pg_stat_replication_replay_diff
gs_sql_count_dcl	gaussdb_log_errors_rate	pg_stat_get_wal_senders_pid	pg_tables_structure_last_vacuum
gs_sql_count_ddl	gaussdb_log_recycle_cbm	gaussdb_confl_bufferpin_rate	gaussdb_table_increase_amplitude
gs_sql_count_dml	gaussdb_log_recycle_lsn	gaussdb_log_recycle_dcf_else	gs_audit_successful_logins_total
node_arp_entries	gaussdb_qps_by_instance	gaussdb_log_recycle_dcf_zero	gs_logical_replication_slots_lsn
node_disk_io_now	gaussdb_statement_calls	gaussdb_statement_hash_spill	gs_never_used_indexes_index_size
node_forks_total	gaussdb_xlog_oldest_lsn	gaussdb_statement_sort_spill	gs_replication_slots_restart_lsn
non_db_mem_usage	gs_audit_ddl_user_total	gs_audit_user_unlocked_total	gs_tables_structure_last_analyze
pg_class_relsize	gs_connections_max_conn	gs_checkpoint_checkpoint_lsn	gs_tables_structure_vacuum_count
pg_db_confl_lock	gs_memory_context_count	gs_pagewriter_dirty_page_num	node_cpu_package_throttles_total
pg_db_temp_bytes	gs_prepared_xacts_count	gs_sesstype_memory_totalsize	node_cpu_scaling_frequency_hertz
pg_db_temp_files	gs_stat_replication_lsn	gs_shared_memory_detail_rate	node_disk_writes_completed_total
pg_sql_count_dcl	node_hwmon_sensor_label	gs_shared_memory_detail_size	node_hwmon_power_is_battery_watt
pg_sql_count_ddl	node_hwmon_temp_celsius	gs_stat_replication_sent_lsn	node_network_address_assign_type
pg_sql_count_dml	node_memory_Dirty_bytes	gs_tables_structure_tbl_scan	node_network_receive_bytes_total
process_open_fds	node_memory_Shmem_bytes	node_cpu_frequency_max_hertz	node_network_receive_frame_total
dynamic_used_rate	node_netstat_Tcp_InErrs	node_cpu_frequency_min_hertz	node_network_transmit_drop_total

版权所有 © 华为云计算技术有限公司

gaussdb_cpu_usage	node_netstat_Tcp_InSegs	node_cpu_guest_seconds_total	node_network_transmit_errs_total
gaussdb_log_panic	node_network_iface_link	node_disk_reads_merged_total	node_network_transmit_fifo_total
gaussdb_nic_state	node_sockstat_RAW_inuse	node_filesystem_device_error	node_timex_pps_calibration_total
gs_checkpoint_tli	node_sockstat_TCP_alloc	node_hwmon_temp_crit_celsius	pg_audit_successful_logins_total
gs_class_relpages	node_sockstat_TCP_inuse	node_memory_PageTables_bytes	pg_logical_replication_slots_lsn
gs_db_blks_access	node_sockstat_UDP_inuse	node_memory_SUnreclaim_bytes	pg_never_used_indexes_index_size
gs_db_reset_delay	node_timex_tick_seconds	node_memory_SwapCached_bytes	pg_replication_slots_restart_lsn
gs_db_tup_deleted	os_cpu_processor_number	node_netstat_IpExt_OutOctets	pg_tables_structure_last_analyze
gs_db_tup_fetched	os_disk_io_queue_length	node_netstat_Tcp_ActiveOpens	pg_tables_structure_vacuum_count
gs_db_tup_updated	os_network_receive_drop	node_netstat_Tcp_RetransSegs	process_virtual_memory_max_bytes
gs_db_xact_commit	pg_audit_ddl_user_total	node_netstat_Udp_InDatagrams	gaussdb_log_cms_heartbeat_timeout
gs_index_idx_scan	pg_connections_max_conn	node_network_iface_link_mode	gaussdb_log_cms_phonydead_restart
node_network_info	pg_memory_context_count	node_sockstat_UDPLITE6_inuse	gaussdb_log_recycle_dummy_standby
node_time_seconds	pg_prepared_xacts_count	node_softnet_processed_total	gaussdb_log_user_cancel_statement
node_timex_status	pg_stat_replication_lsn	node_timex_pps_shift_seconds	gs_sql_statement_history_exc_time
os_cpu_idle_usage	scrape_duration_seconds	pg_audit_user_unlocked_total	gs_stat_replication_sync_priority
os_cpu_user_usage	gaussdb_log_errors_total	pg_checkpoint_checkpoint_lsn	gs_tables_structure_analyze_count
pg_checkpoint_tli	gaussdb_log_flow_control	pg_pagewriter_dirty_page_num	gs_tables_structure_n_tup_hot_upd
pg_class_relpages	gaussdb_log_login_denied	pg_sesstype_memory_totalsize	node_disk_read_time_seconds_total

版权所有 © 华为云计算技术有限公司

pg_db_blks_access	gaussdb_log_node_restart	pg_shared_memory_detail_rate	node_hwmon_temp_crit_alar_celsius
pg_db_reset_delay	gaussdb_ping_packet_rate	pg_shared_memory_detail_size	node_netstat_TcpExt_TCPSynRetrans
pg_db_tup_deleted	gaussdb_total_connection	pg_stat_replication_sent_lsn	node_network_transmit_bytes_total
pg_db_tup_fetched	gaussdb_tup_deleted_rate	pg_tables_structure_tbl_scan	node_network_transmit_colls_total
pg_db_tup_updated	gaussdb_tup_fetched_rate	process_virtual_memory_bytes	node_softnet_times_squeezed_total
pg_db_xact_commit	gaussdb_tup_updated_rate	gaussdb_confl_temp_bytes_rate	pg_sql_statement_history_exc_time
pg_index_idx_scan	gaussdb_user_locked_rate	gaussdb_confl_temp_files_rate	pg_stat_replication_sync_priority
gaussdb_ping_state	gaussdb_wait_event_spike	gaussdb_log_lock_wait_timeout	pg_tables_structure_analyze_count
gaussdb_state_time	gs_checkpoint_oldest_xid	gaussdb_log_undo_impermission	pg_tables_structure_n_tup_hot_upd
gaussdb_user_login	gs_connections_used_conn	gaussdb_log_undo_invalidation	gaussdb_log_cn_restart_time_exceed
gaussdb_xlog_count	gs_connections_used_rate	gs_audit_invalid_logins_total	gaussdb_log_recycle_standby_backup
gs_checkpoint_time	gs_lock_sql_locked_times	gs_audit_user_violation_total	gs_lock_time_info_p95_holding_time
gs_class_reltuples	gs_sesstype_memory_count	gs_autovacuum_worker_duration	gs_session_memory_detail_totalsize
gs_db_tup_inserted	gs_tables_size_indexsize	gs_checkpoint_oldest_xid_dbid	gs_stat_replication_backend_uptime
gs_db_tup_returned	gs_tables_size_toastsize	gs_session_memory_detail_rate	node_disk_write_time_seconds_total
gs_txid_oldestxmin	gs_tables_size_totalsize	gs_session_memory_detail_size	node_edac_correctable_errors_total
node_network_flags	node_exporter_build_info	gs_stat_replication_flush_lsn	node_hwmon_temp_crit_alarm_celsius
node_procs_blocked	node_filesystem_readonly	gs_stat_replication_sent_diff	node_netstat_TcpExt_SyncookiesRecv

版权所有 © 华为云计算技术有限公司

node_procs_running	node_memory_Active_bytes	gs_stat_replication_write_lsn	node_netstat_TcpExt_SyncookiesSent
node_vmstat_pgpgin	node_memory_Bounce_bytes	gs_summary_user_login_counter	node_network_carrier_changes_total
node_vmstat_pswpin	node_memory_Cached_bytes	gs_tables_structure_dead_rate	node_network_receive_packets_total
os_disk_iocapacity	node_memory_Mapped_bytes	gs_tables_structure_n_tup_del	node_network_transmit_queue_length
os_mem_total_bytes	node_netstat_Icmp_InMsgs	gs_tables_structure_n_tup_ins	node_time_clocksource_current_info
pg_checkpoint_time	node_netstat_Tcp_OutRsts	gs_tables_structure_n_tup_mod	node_timex_estimated_error_seconds
pg_class_reltuples	node_netstat_Tcp_OutSegs	gs_tables_structure_n_tup_upd	pg_lock_time_info_p95_holding_time
pg_db_tup_inserted	node_netstat_Udp_NoPorts	gs_total_memory_detail_mbytes	pg_session_memory_detail_totalsize
pg_db_tup_returned	node_network_speed_bytes	node_cooling_device_cur_state	pg_stat_replication_backend_uptime
pg_txid_oldestxmin	node_sockstat_FRAG_inuse	node_cooling_device_max_state	gaussdb_database_increase_amplitude
gaussdb_mount_usage	node_sockstat_RAW6_inuse	node_cpu_core_throttles_total	gaussdb_log_recycle_extro_read_else
gaussdb_user_logout	node_sockstat_TCP_orphan	node_disk_writes_merged_total	gaussdb_log_recycle_extro_read_zero
gs_db_blk_read_time	node_sockstat_TCP6_inuse	node_disk_written_bytes_total	gs_global_double_write_status_value
gs_db_xact_rollback	node_sockstat_UDP6_inuse	node_hwmon_power_average_watt	gs_sql_statement_statistics_db_time
gs_downstream_count	os_network_receive_bytes	node_memory_Active_anon_bytes	gs_sql_statement_statistics_n_calls
gs_node_info_uptime	os_network_receive_error	node_memory_Active_file_bytes	gs_tables_structure_last_autovacuum

版权所有 © 华为云计算技术有限公司

gs_settings_setting	os_network_transmit_drop	node_memory_CommitLimit_bytes	node_memory_HardwareCorrupted_bytes
gs_sql_count_delete	pg_checkpoint_oldest_xid	node_memory_DirectMap1G_bytes	node_netstat_TcpExt_ListenOverflows
gs_sql_count_insert	pg_connections_used_conn	node_memory_DirectMap2M_bytes	node_network_transmit_carrier_total
gs_sql_count_select	pg_connections_used_rate	node_memory_DirectMap4k_bytes	node_network_transmit_packets_total
gs_sql_count_update	pg_lock_sql_locked_times	node_memory_KernelStack_bytes	pg_global_double_write_status_value
gs_temp_files_count	pg_sesstype_memory_count	node_memory_Unevictable_bytes	pg_sql_statement_statistics_db_time
gs_thread_pool_rate	pg_tables_size_indexsize	node_memory_VmallocUsed_bytes	pg_sql_statement_statistics_n_calls
node_filefd_maximum	pg_tables_size_toastsize	node_netstat_Tcp_PassiveOpens	pg_tables_structure_last_autovacuum
node_vmstat_pgfault	pg_tables_size_totalsize	node_netstat_Udp_OutDatagrams	gaussdb_log_recycle_replication_slot
node_vmstat_pgpgout	gaussdb_active_connection	node_netstat_Udp_RcvbufErrors	gs_tables_structure_last_autoanalyze
node_vmstat_pswpout	gaussdb_log_cms_read_only	node_netstat_Udp_SndbufErrors	node_cpu_scaling_frequency_max_hertz
oldestxmin_increase	gaussdb_log_recycle_build	node_netstat_Udp6_InDatagrams	node_cpu_scaling_frequency_min_hertz
os_cpu_iowait_usage	gaussdb_process_cpu_usage	node_netstat_UdpLite_InErrors	node_edac_uncorrectable_errors_total
os_cpu_system_usage	gaussdb_process_mem_usage	node_scrape_collector_success	node_netstat_TcpExt_SyncookiesFailed

版权所有 © 华为云计算技术有限公司

os_process_fds_rate	gaussdb_statement_db_time	node_time_zone_offset_seconds	node_network_receive_multicast_total
pg_db_blk_read_time	gaussdb_tup_inserted_rate	node_timex_loop_time_constant	node_schedstat_running_seconds_total
pg_db_xact_rollback	gs_long_transaction_count	node_timex_pps_jitter_seconds	node_schedstat_waiting_seconds_total
pg_downstream_count	gs_long_xact_mem_ctx_size	node_timex_tai_offset_seconds	node_time_clocksource_available_info
pg_node_info_uptime	gs_state_memory_totalsize	pg_audit_invalid_logins_total	pg_tables_structure_last_autoanalyze
pg_settings_setting	node_memory_Buffers_bytes	pg_audit_user_violation_total	promhttp_metric_handler_errors_total
pg_sql_count_delete	node_memory_CmaFree_bytes	pg_autovacuum_worker_duration	gs_summary_file_iostat_total_phyblkrd
pg_sql_count_insert	node_memory_MemFree_bytes	pg_checkpoint_oldest_xid_dbid	node_network_receive_compressed_total
pg_sql_count_select	node_memory_Mlocked_bytes	pg_session_memory_detail_rate	node_timex_frequency_adjustment_ratio
pg_sql_count_update	node_netstat_Icmp_OutMsgs	pg_session_memory_detail_size	pg_summary_file_iostat_total_phyblkrd
pg_temp_files_count	node_netstat_Icmp6_InMsgs	pg_stat_replication_flush_lsn	scrape_samples_post_metric_relabeling
pg_thread_pool_rate	node_netstat_Ip6_InOctets	pg_stat_replication_sent_diff	statement_responsetime_percentile_p80
scrape_series_added	node_netstat_Udp_InErrors	pg_stat_replication_write_lsn	statement_responsetime_percentile_p95
gaussdb_state_memory	node_netstat_Udp6_NoPorts	pg_summary_user_login_counter	gs_summary_file_iostat_total_phyblkwrt
gs_audit_grant_total	node_nf_conntrack_entries	pg_tables_structure_dead_rate	net_conntrack_dialer_conn_closed_total
gs_checkpoint_elapse	node_sockstat_FRAG_memory	pg_tables_structure_n_tup_del	net_conntrack_dialer_conn_failed_total

版权所有 © 华为云计算技术有限公司

gs_database_all_size	node_sockstat_FRAG6_inuse	pg_tables_structure_n_tup_ins	node_network_transmit_compressed_total
gs_db_blk_write_time	node_timex_offset_seconds	pg_tables_structure_n_tup_mod	node_scrape_collector_duration_seconds
gs_db_confl_deadlock	os_network_transmit_bytes	pg_tables_structure_n_tup_upd	pg_summary_file_iostat_total_phyblkwrt
gs_db_confl_snapshot	os_network_transmit_error	pg_total_memory_detail_mbytes	promhttp_metric_handler_requests_total
node_network_carrier	pg_long_transaction_count	process_resident_memory_bytes	gaussdb_log_gtm_disconnected_to_primary
node_network_dormant	pg_long_xact_mem_ctx_size	gaussdb_connections_used_ratio	gs_session_connection_total_session_num
node_sockstat_TCP_tw	pg_state_memory_totalsize	gaussdb_database_increase_rate	node_timex_pps_stability_exceeded_total
pg_audit_grant_total	process_cpu_seconds_total	gaussdb_dead_tup_increase_rate	pg_session_connection_total_session_num
pg_checkpoint_elapse	gaussdb_log_bind_ip_failed	gaussdb_log_dn_writable_failed	net_conntrack_listener_conn_closed_total
pg_database_all_size	gaussdb_log_deadlock_count	gaussdb_log_recycle_full_build	node_disk_io_time_weighted_seconds_total
pg_db_blk_write_time	gaussdb_process_leaked_fds	gaussdb_successful_logins_rate	node_edac_csrow_correctable_errors_total
pg_db_confl_deadlock	gs_audit_user_locked_total	gs_checkpoint_full_page_writes	gaussdb_log_cms_heartbeat_timeout_restart
pg_db_confl_snapshot	gs_connections_enqueue_sql	gs_lock_time_info_holding_time	gs_stat_bgwriter_checkpoint_avg_sync_time
xlog_setting_margin	gs_session_memory_usedsize	gs_recovery_status_current_rto	net_conntrack_dialer_conn_attempted_total
gaussdb_cluster_state	node_disk_read_bytes_total	gs_replication_slots_delay_lsn	node_hwmon_power_average_interval_seconds
gaussdb_log_cn_status	node_filesystem_files_free	gs_stat_bgwriter_total_seconds	pg_stat_bgwriter_checkpoint_avg_sync_time

版权所有 © 华为云计算技术有限公司

gaussdb_log_dn_status	node_filesystem_free_bytes	gs_stat_replication_flush_diff	net_conntrack_listener_conn_accepted_total
gaussdb_log_gtm_panic	node_filesystem_size_bytes	gs_stat_replication_replay_lag	node_edac_csrow_uncorrectable_errors_total
gs_audit_logout_total	node_memory_CmaTotal_bytes	gs_stat_replication_replay_lsn	promhttp_metric_handler_requests_in_flight
gs_audit_revoke_total	node_memory_HugePages_Free	gs_stat_replication_write_diff	gs_tables_structure_last_data_changed_delay
gs_cpu_load_total_cpu	node_memory_HugePages_Rsvd	gs_summary_user_logout_counter	net_conntrack_dialer_conn_established_total
gs_db_confl_bufferpin	node_memory_HugePages_Surp	gs_tables_structure_n_dead_tup	pg_tables_structure_last_data_changed_delay
gs_index_idx_blks_hit	node_memory_Inactive_bytes	gs_tables_structure_n_live_tup	gs_sql_statement_statistics_hash_spill_count
gs_redo_stat_writetim	node_memory_MemTotal_bytes	gs_tables_structure_n_tup_read	gs_sql_statement_statistics_sort_spill_count
node_filefd_allocated	node_memory_SwapFree_bytes	gs_wait_events_total_wait_time	pg_sql_statement_statistics_hash_spill_count
node_filesystem_files	node_netstat_Icmp_InErrors	node_memory_Committed_AS_bytes	pg_sql_statement_statistics_sort_spill_count
node_hwmon_chip_names	node_netstat_Icmp6_OutMsgs	node_memory_Hugepagesize_bytes	node_hwmon_power_average_interval_max_seconds
node_network_iface_id	node_netstat_Ip_Forwarding	node_memory_MemAvailable_bytes	node_hwmon_power_average_interval_min_seconds
node_sockstat_TCP_mem	node_netstat_Ip6_OutOctets	node_memory_NFS_Unstable_bytes	gs_stat_bgwriter_checkpoint_proactive_triggering_ratio
node_sockstat_UDP_mem	node_netstat_Tcp_CurrEstab	node_memory_SReclaimable_bytes	pg_stat_bgwriter_checkpoint_proactive_triggering_ratio

版权所有 © 华为云计算技术有限公司

os_disk_io_read_bytes	node_netstat_Udp6_InErrors	node_memory_VmallocChunk_bytes	-

DBMind 定时任务

DBMind内嵌了可定期拉起并自动执行的定时任务，可以在DBMind的配置文件confpath/dbmind.conf中开启或关闭特定的定时任务。DBMind已经支持的定时任务，以及其对应的简要说明，可以在dbmind.conf中的[TIMED_TASK_LIST]中查看。

- 要开启一个定时任务：在dbmind.conf中的[TIMED_TASK_LIST]中找到要开启的定时任务的名称，然后将其编辑加入[TIMED_TASK]的TASK字段中并保存，用","分割多个定时任务。例如，要开启"异常检测"的定时任务，就在[TIMED_TASK]的TASK字段中编辑加入"anomaly_detection"。
- 要关闭一个定时任务：在dbmind.conf中的[TIMED_TASK]的TASK字段中，将需要关闭的定时任务编辑删除并保存。

在dbmind.conf中[TIMED_TASK]的其他字段下可以自行配置各个定时任务自动执行周期，如：配置"anomaly_detection_interval"对应的值，可以对应地修改“异常检测”定时任务的自动执行周期。

修改配置文件后运行如下命令使修改生效：

```
gs_dbmind service reload -c confpath
```

表 11-30 DBMind 内嵌的定时任务说明

定时任务名称	定时任务说明	定时任务自动执行周期配置字段	默认自动执行周期（秒）	是否默认启动	额外说明
anomaly_detection	异常检测	anomaly_detection_interval	180	是	不建议关闭，否则将无法进行实时指标异常监控，历史告警将停止更新。
discard_expired_results	清理元数据库中的超期数据	discard_expired_results_interval	3600	是	常驻，用于清理元数据库中存储的写入于7天（默认，可以通过dynamic config设置）以前的时效性数据，如历史告警、历史集群诊断记录等。
knob_recommend	参数推荐	knob_recommend_interval	3600	否	-
slow_query_killer	慢SQL查杀	slow_query_killer_interval	30	否	只支持CN节点上的查杀操作。

版权所有 © 华为云计算技术有限公司

定时任务名称	定时任务说明	定时任务自动执行周期配置字段	默认自动执行周期（秒）	是否默认启动	额外说明
slow_query_diagnosis	慢SQL诊断	slow_query_diagnosis_interval	120	否	-
cluster_diagnose	集群诊断	cluster_diagnose_interval	30	是	不建议关闭，否则将无法进行实时集群异常诊断，历史集群诊断列表将停止更新。
agent_update_detect	Agent自动检测	agent_update_detect_interval	30	是	不建议关闭，该定时任务可定时检测实例的纳管情况，从而在纳管实例发生变化时（如主备倒换、增加纳管、减少纳管、扩缩容等）及时更新异常检测器的检测范围。
calibrate_security_metrics	自安全指标校准	calibrate_security_metrics_interval	600	否	-
check_security_metrics	自安全异常检测	check_security_metrics_interval	600	否	-
update_statistics	长周期监控指标统计值计算	update_statistic_interval	1800	是	不建议关闭，否则长周期持久化存储的指标将停止更新，影响亚健康相关的诊断结果。

11.3.1 X-Tuner

11.3.1.1 概述

X-Tuner是一款数据库集成的参数调优工具，通过结合深度强化学习和全局搜索算法等AI技术，实现在无需人工干预的情况下，获取最佳数据库参数配置。本功能不强制与数据库环境部署到一起，支持独立部署，脱离数据库安装环境独立运行。

说明

本功能不支持分布式数据库场景。使用X-Tuner功能会尝试建立数据库连接，从而获取数据库规格。如果是分布式数据库，则会提示不支持。

版权所有 © 华为云计算技术有限公司

11.3.1.2 使用准备

前提条件与使用事项

- 数据库状态正常、客户端能够正常连接、且要求数据库内导入数据，以便调优程序可以执行benchmark测试调优效果。
- 使用本工具需要指定登录到数据库的用户身份，要求该登录到数据库上的用户具有足够的权限，以便可以获得充足的数据库状态信息。
- 使用登录到数据库宿主机上的Linux用户，需要将“$GAUSSHOME/bin”添加到PATH环境变量中，即能够直接运行gsql、gs_guc、gs_ctl等数据库运维工具。
- 本工具支持以三种模式运行，其中tune和train模式要求用户配置好benchmark运行环境，并导入数据，本工具将会通过迭代运行benchmark来判断修改后的参数是否有性能提升。
- recommend模式建议在数据库正在执行workload的过程中执行，以便获得更准确的实时workload信息。
- 本工具默认带有TPC-C、TPC-H、TPC-DS以及sysbench的benchmark运行脚本样例，如果用户使用上述benchmark对数据库系统进行压力测试，则可以对上述配置文件进行适度修改或配置。如果需要适配用户自己的业务场景，需要您参照benchmark目录中的template.py文件编写驱动您自定义benchmark的脚本文件。

原理简介

调优程序是一个独立于数据库内核之外的工具，需要提供数据库及其所在实例的用户名和登录密码信息，以便控制数据库执行benchmark进行性能测试；在启动调优程序前，要求用户测试环境交互正常，能够正常跑通benchmark测试脚本、能够正常连接数据库。

说明

如果需要调优的参数中，包含重启数据库后才能使修改生效的参数，那么在调优过程中数据库将会重启多次。如果用户的数据库正在执行作业，请慎用train与tune模式。

调优程序X-Tuner包含三种运行模式，分别是:

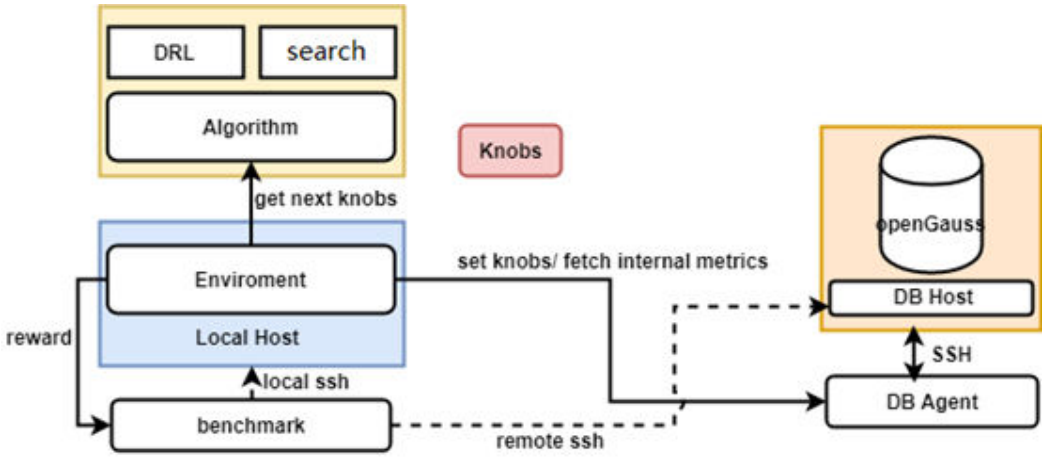
- recommend：通过用户指定的用户名等信息登录到数据库环境中，获取当前正在运行的workload特征信息，根据上述特征信息生成参数推荐报告。报告当前数据库中不合理的参数配置和潜在风险等；输出根据当前正在运行的workload行为和特征；输出推荐的参数配置。该模式是秒级的，不涉及数据库的重启操作，其他模式可能需要反复重启数据库。
- train：通过用户提供的benchmark信息，不断地进行参数修改和benchmark的执行。通过反复的迭代过程，训练强化学习模型，以便用户在后面通过tune模式加载该模型进行调优。
- tune：使用优化算法进行数据库参数的调优，当前支持两大类算法，一种是深度强化学习，另一种是全局搜索算法（全局优化算法）。深度强化学习模式要求先运行train模式，生成训练后的调优模型，而使用全局搜索算法则不需要提前进行训练，可以直接进行搜索调优。

须知

如果在tune模式下，使用深度强化学习算法，要求必须有一个训练好的模型，且要求训练该模型时的参数与进行调优时的参数列表（包括max与min）必须一致。

版权所有 © 华为云计算技术有限公司

图 11-2 X-Tuner 结构图



X-Tuner的整体架构如上方X-Tuner 结构图所示，系统可以分为：

- DB侧：通过DB_Agent模块对数据库实例进行抽象，通过该模块可以获取数据库内部的状态信息、当前数据库参数、以及设置数据库参数等。DB侧包括登录数据库环境使用的SSH连接。
- 算法侧：用于调优的算法包，包括全局搜索算法（如贝叶斯优化、粒子群算法等）和深度强化学习（如DDPG）。
- X-Tuner主体逻辑模块：通过Environment模块进行封装，每一个step就是一次调优过程。整个调优过程通过多个step进行迭代。
- benchmark：由用户指定的benchmark性能测试脚本，用于运行benchmark作业，通过跑分结果反映数据库系统性能优劣。

说明

应确保benchmark脚本跑分结果越大表示性能越好。

例如TPCH这种衡量SQL语句整体执行时长的benchmark，可以通过取总体执行时间的相反数作为benchmark的输出分数。

X-Tuner 的运行和安装方法

执行下述命令即可获取xtuner功能帮助

```
gs_dbmind component xtuner --help
```

用户可据此给定不同的命令行执行相应的功能。

X-Tuner 的配置文件说明

X-Tuner在运行前需要加载配置文件，用户可以通过“--help”命令查看默认加载的配置文件绝对路径：

```
-x TUNER_CONFIG_FILE, --tuner-config-file TUNER_CONFIG_FILE
                      This is the path of the core configuration file of the
                      X-Tuner. You can specify the path of the new
                      configuration file. The default path is DBMINDPATH/dbmind/components/xtuner/tuner/
xtuner.conf.
                      You can modify the configuration file to control the
                      tuning process.
```

 版权所有 © 华为云计算技术有限公司

修改配置文件的配置项可以指引X-Tuner执行不同的动作，用户可以根据自己的不同需求来修改配置文件的内容，配置文件的配置项说明详见**表11-32**。如果需要修改配置文件的加载路径，则可以通过选项“-x”命令行选项来指定。

Benchmark 的选择与配置

Benchmark的驱动脚本存放路径为X-Tuner目录的子目录benchmark中。X-Tuner自带常用的benchmark驱动脚本，例如基于时间周期的探测脚本（默认）、TPC-C、TPC-H等。X-Tuner通过调用benchmark/__init__.py文件中 get_benchmark_instance()命令来加载不同的benchmark驱动脚本，获取benchmark驱动实例。其中，benchmark驱动脚本的格式说明如下：

- 驱动脚本文件名：表示benchmark的名字，该名字用于表示驱动脚本的唯一性，可通过在X-Tuner的配置文件中的配置项benchmark_script来指定选择加载benchmark驱动脚本。
- 驱动脚本内容三要素：path变量、cmd变量以及run函数。

下面分别介绍驱动脚本的内容三要素：

1. path变量：表示benchmark脚本的存放地址，可以直接在驱动脚本中修改，也可以通过配置文件的benchmark_path配置项来指定。
2. cmd变量：表示执行benchmark脚本需要运行的命令，可以直接在驱动脚本中修改，也可以通过配置文件的benchmark_cmd配置项来指定。cmd中的文本允许使用占位符，用于获取某些运行cmd命令时的必要信息，使用示例参见TPC-H驱动脚本示例。这些占位符包括：
   - {host}：数据库宿主机的IP地址。
   - {port}：数据库实例的侦听端口号。
   - {user}：登录数据库系统上的用户名。
   - {password}：与登录数据库系统上的用户相匹配的密码。
   - {db}：正在进行调优的数据库名。
3. run函数：该函数的函数签名为：

   ```
   def run(remote_server, local_host) -> float:
   ```

   其中，返回数据类型为float，表示benchmark执行后的评估分数值，要求该值越大表示性能越好，例如使用TPC-C跑分结果tpmC即可作为返回值，TPC-H的全部SQL语句执行总时间的相反数（取相反数后可保证返回值越大则性能越好）也可作为返回值。

   remote_server变量是X-Tuner程序传递给脚本使用的远端主机（数据库宿主机）的shell命令接口，local_host变量是X-Tuner程序传递给脚本使用的本地主机（运行X-Tuner脚本的主机）的shell命令接口。上述shell命令接口提供的方法包括：

   ```
   exec_command_sync(command, timeout)
   功能：该方法用于在主机上执行shell命令。
   参数列表：
   command 必选，数据类型可以是str, 以及元素为str类型的list或tuple；
   timeout 可选，表示命令执行的超时时长，单位是秒。
   返回值：
   返回二元组 (stdout, stderr)，stdout表示标准输出流结果，stderr表示标准错误流结果，数据类型均为str。
   exit_status
   功能：该属性表示最近一条shell命令执行后的退出状态码(exit status code)。
   说明：一般情况，退出状态码为0表示执行正常，非0表示存在错误。
   ```

Benchmark驱动脚本示例说明

1. TPC-C 驱动脚本

   ```
   from tuner.exceptions import ExecutionError
   ```

 版权所有 © 华为云计算技术有限公司

```
# WARN: You need to download the benchmark-sql test tool to the system,
# replace the PostgreSQL JDBC driver with the GaussDB driver,
# and configure the benchmark-sql configuration file.
# The program starts the test by running the following command:
path = '/path/to/benchmarksql/run'  # TPC-C测试脚本benchmark-sql 的存放路径
cmd = "./runBenchmark.sh props.gs"  # 自定义一个名为 props.gs 的benchmark-sql测试配置文件


def run(remote_server, local_host):
    # 切换到 TPC-C 脚本目录下，清除历史错误日志，然后运行测试命令。
    # 此处建议等待几秒钟，因为benchmark-sql 测试脚本生成最终测试报告是通过一个shell脚本实现的，整个过程会有延迟，
    # 为了保证能够获取到最终的tpmC数值报告，这里选择等待3秒钟。
    stdout, stderr = remote_server.exec_command_sync(['cd %s' % path, 'rm -rf benchmarksql-error.log', cmd, 'sleep 3'])
    # 如果标准错误流中有数据，则报异常退出。
    if len(stderr) > 0:
        raise ExecutionError(stderr)

    # 寻找最终tpmC结果
    tpmC = None
    split_string = stdout.split()  # 对标准输出流结果进行分词。
    for i, st in enumerate(split_string):
        # 在5.0版本的benchmark-sql中，tpmC最终测试结果数值在 ‘(NewOrders)’关键字的后两位，正常情况下，找到该字段后直接返回即可。
        if "(NewOrders)" in st:
            tpmC = split_string[i + 2]
            break
    stdout, stderr = remote_server.exec_command_sync(
        "cat %s/benchmarksql-error.log" % path)
    nb_err = stdout.count("ERROR:")  # 判断整个benchmark运行过程中，是否有报错，记录报错的错误数
    return float(tpmC) - 10 * nb_err  # 这里将报错的错误数作为一个惩罚项，惩罚系数为10，越高的惩罚系数表示越看中报错的数量.
```

2. TPC-H驱动脚本

```
import time

from tuner.exceptions import ExecutionError

# WARN: You need to import data into the database and SQL statements in the following path will be executed.
# The program automatically collects the total execution duration of these SQL statements.
path = '/path/to/tpch/queries'  # 存放TPC-H测试用的SQL脚本目录
cmd = "gsql -U {user} -W {password} -d {db} -p {port} -f {file}"  # 需要运行TPC-H测试脚本的命令，一般使用'gsql -f 脚本文件' 来运行


def run(remote_server, local_host):
    # 遍历当前目录下所有的测试用例文件名
    find_file_cmd = "find . -type f -name '*.sql'"
    stdout, stderr = remote_server.exec_command_sync(['cd %s' % path, find_file_cmd])
    if len(stderr) > 0:
        raise ExecutionError(stderr)
    files = stdout.strip().split('\n')
    time_start = time.time()
    for file in files:
        # 使用 file 变量替换 {file}，然后执行该命令行。
        perform_cmd = cmd.format(file=file)
        stdout, stderr = remote_server.exec_command_sync(['cd %s' % path, perform_cmd])
        if len(stderr) > 0:
            print(stderr)
    # 代价为全部测试用例的执行总时长
    cost = time.time() - time_start
    # 取相反数，适配run 函数的定义：返回结果越大表示性能越好。
    return - cost
```


版权所有 © 华为云计算技术有限公司

11.3.1.3 使用示例

X-Tuner支持三种模式，分别是获取参数诊断报告的recommend模式、训练强化学习模型的train模式以及使用算法进行调优的tune模式。上述三种模式可以通过命令行参数来区别，通过配置文件来指定具体的细节。

配置数据库连接信息

三种模式连接数据库的配置项是相同的，有两种方式：一种是直接通过命令行输入详细的连接信息，另一种是通过JSON格式的配置文件输入，下面分别对两种指定数据库连接信息的方法进行说明。

1. 通过命令行指定：

   分别传递--db-name --db-user --port --host --host-user参数，可选--host-ssh-port参数，例如：

   ```
   gs_dbmind component xtuner recommend --db-name testdb --db-user omm --port 5678 --host 192.168.1.100 --host-user omm
   ```

2. 通过JSON格式的连接信息配置文件指定：

   JSON配置文件的示例如下，并假设文件名为 connection.json：

   ```
   {
     "db_name": "testdb",  # 数据库名
     "db_user": "dba",       # 登录到数据库上的用户名
     "host": "127.0.0.1",    # 数据库宿主机的IP地址
     "host_user": "dba",     # 登录到数据库宿主机的用户名
     "port": 5432,           # 数据库的侦听端口号
     "ssh_port": 22          # 数据库宿主机的SSH侦听端口号
   }
   ```

   则可通过 -f connection.json 传递。

说明

为了防止密码泄露，配置文件和命令行参数中默认都不包含密码信息，用户在输入上述连接信息后，程序会采用交互式的方式要求用户输数据库密码以及操作系统登录用户的密码。

recommend 模式使用示例

对recommend模式生效的配置项为scenario，若为auto，则自动检测workload类型。

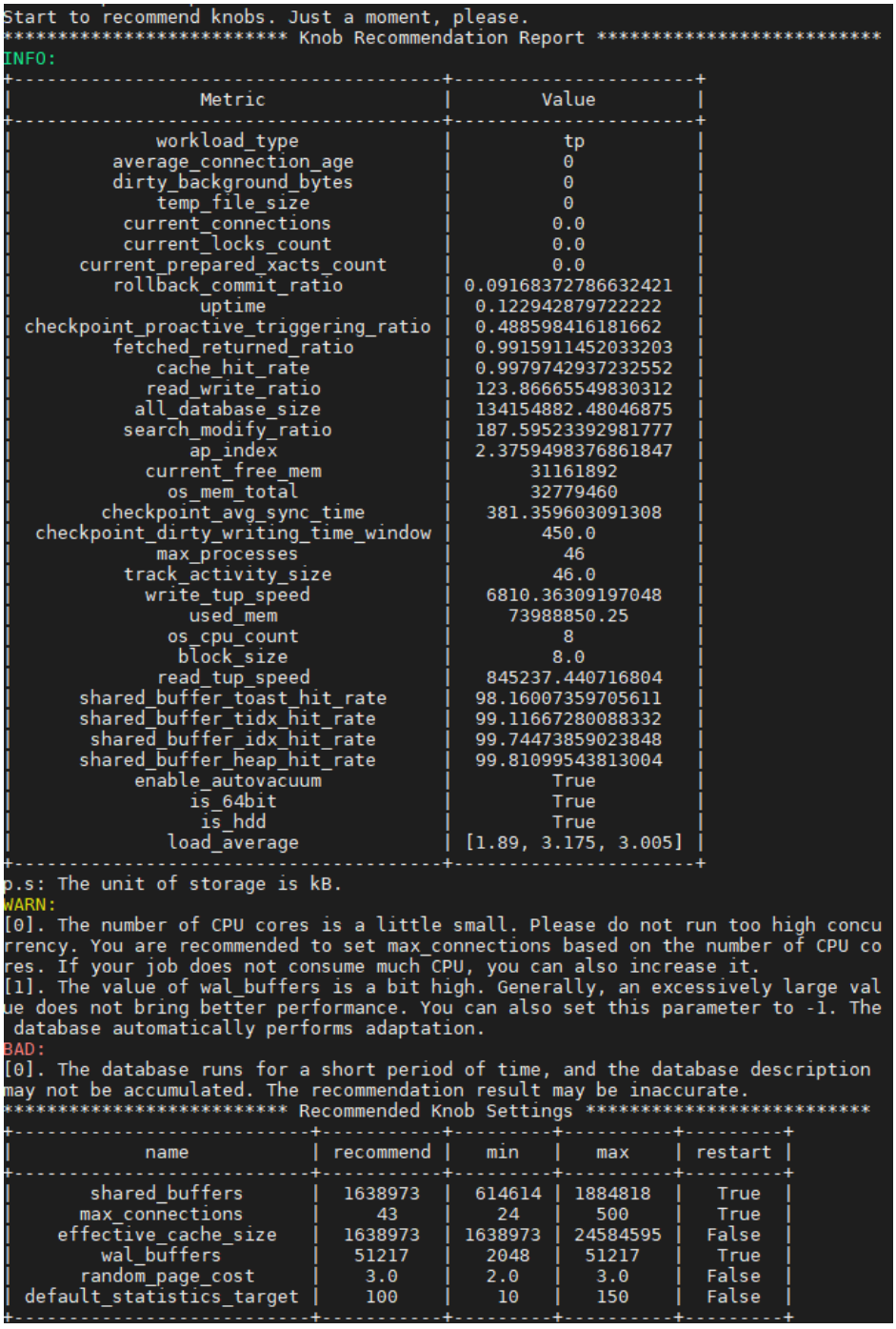
执行下述命令，获取诊断结果：

```
gs_dbmind component xtuner recommend -f connection.json
```

则可以生成诊断报告如图11-3所示。

 版权所有 © 华为云计算技术有限公司

图 11-3 recommend 模式生成的报告示意图



```
Start to recommend knobs. Just a moment, please.
************************** Knob Recommendation Report **************************
INFO:
+--------------------------------------+----------------------+
|                Metric                |        Value         |
+--------------------------------------+----------------------+
|             workload_type            |          tp          |
|         average_connection_age       |          0           |
|         dirty_background_bytes       |          0           |
|             temp_file_size           |          0           |
|          current_connections         |         0.0          |
|          current_locks_count         |         0.0          |
|      current_prepared_xacts_count    |         0.0          |
|          rollback_commit_ratio       | 0.09168372786632421  |
|                 uptime               |  0.122942879722222   |
| checkpoint_proactive_triggering_ratio |  0.488598416181662   |
|         fetched_returned_ratio       |  0.9915911452033203  |
|             cache_hit_rate           |  0.9979742937232552  |
|            read_write_ratio          |  123.86665549830312  |
|          all_database_size           |  134154882.48046875  |
|          search_modify_ratio         |  187.59523392981777  |
|                ap_index              |  2.3759498376861847  |
|            current_free_mem          |       31161892       |
|             os_mem_total             |       32779460       |
|        checkpoint_avg_sync_time      |   381.359603091308   |
|  checkpoint_dirty_writing_time_window |        450.0         |
|             max_processes            |          46          |
|          track_activity_size         |         46.0         |
|            write_tup_speed           |   6810.36309197048   |
|                used_mem              |     73988850.25      |
|              os_cpu_count            |          8           |
|               block_size             |         8.0          |
|             read_tup_speed           |   845237.440716804   |
|      shared_buffer_toast_hit_rate    |  98.16007359705611   |
|      shared_buffer_tidx_hit_rate     |  99.11667280088332   |
|       shared_buffer_idx_hit_rate     |  99.74473859023848   |
|      shared_buffer_heap_hit_rate     |  99.81099543813004   |
|           enable_autovacuum          |         True         |
|                is_64bit              |         True         |
|                 is_hdd               |         True         |
|              load_average            | [1.89, 3.175, 3.005] |
+--------------------------------------+----------------------+
p.s: The unit of storage is kB.
WARN:
[0]. The number of CPU cores is a little small. Please do not run too high concu
rrency. You are recommended to set max_connections based on the number of CPU co
res. If your job does not consume much CPU, you can also increase it.
[1]. The value of wal_buffers is a bit high. Generally, an excessively large val
ue does not bring better performance. You can also set this parameter to -1. The
 database automatically performs adaptation.
BAD:
[0]. The database runs for a short period of time, and the database description
may not be accumulated. The recommendation result may be inaccurate.
************************** Recommended Knob Settings ***************************
+---------------------------+-----------+---------+----------+---------+
|            name           | recommend |   min   |   max    | restart |
+---------------------------+-----------+---------+----------+---------+
|       shared_buffers      |  1638973  |  614614 | 1884818  |   True  |
|      max_connections      |     43    |    24   |   500    |   True  |
|    effective_cache_size   |  1638973  | 1638973 | 24584595 |  False  |
|        wal_buffers        |   51217   |   2048  |   51217  |   True  |
|      random_page_cost     |    3.0    |    2.0  |    3.0   |  False  |
| default_statistics_target |    100    |    10   |    150   |  False  |
+---------------------------+-----------+---------+----------+---------+
```

在上述报告中，推荐了该环境上的数据库参数配置，并进行了风险提示。报告同时生成了当前workload的特征信息，其中有几个特征是比较有参考意义的：

- temp_file_size：产生的临时文件数量，如果该结果大于0，则表明系统使用了临时文件。使用过多的临时文件会导致性能不佳，如果可能的话，需要提高work_mem参数的配置。
- cache_hit_rate：shared_buffer的缓存命中率，表明当前workload使用缓存的效率。

 版权所有 © 华为云计算技术有限公司

- read_write_ratio：数据库作业的读写比例。
- search_modify_ratio：数据库作业的查询与修改数据的比例。
- ap_index：表明当前workload的AP指数，取值范围是0到10，该数值越大，表明越偏向于数据分析与检索。
- workload_type：根据数据库统计信息，推测当前负载类型，分为AP、TP以及HTAP三种类型。
- checkpoint_avg_sync_time：数据库在checkpoint时，平均每次同步刷新数据到磁盘的时长，单位是毫秒。
- load_average：平均每个CPU核心在1分钟、5分钟以及15分钟内的负载。一般地，该数值在1左右表明当前硬件比较匹配workload、在3左右表明运行当前作业压力比较大，大于5则表示当前硬件环境运行该workload压力过大（此时一般建议减少负载或升级硬件）。

说明

- recommend模式会读取数据库中的pg_stat_database以及pg_stat_bgwriter等系统表中的信息，需要登录到数据库上的用户具有足够的权限（建议为**管理员权限**，可通过alter user username sysadmin；授予username相应的权限）。
- 由于某些系统表会一直记录统计信息，这可能会对负载特征识别造成干扰，因此建议先清空某些系统表的统计信息，运行一段时间的workload后再使用recommend模式进行诊断，以便获得更准确的结果。清除统计信息的方法为：

```
select pg_stat_reset_shared('bgwriter');
select pg_stat_reset();
```

train 模式使用示例

该模式是用来训练深度强化学习模型的，与该模式有关的配置项为：

- rl_algorithm：用于训练强化学习模型的算法，当前支持设置为ddpg。
- rl_model_path：训练后生成的强化学习模型保存路径。
- rl_steps：训练过程的最大迭代步数。
- max_episode_steps：每个回合的最大步数。
- scenario：明确指定的workload类型，如果为auto则为自动判断。在不同模式下，推荐的调优参数列表也不一样。
- tuning_list：用户指定需要调哪些参数，如果不指定，则根据workload类型自动推荐应该调的参数列表。如需指定，则tuning_list表示调优列表文件的路径。一个调优列表配置文件的文件内容示例如下：

```
{
  "work_mem": {
    "default": 65536,
    "min": 65536,
    "max": 655360,
    "type": "int",
    "restart": false
  },
  "shared_buffers": {
    "default": 32000,
    "min": 16000,
    "max": 64000,
    "type": "int",
    "restart": true
  },
  "random_page_cost": {
    "default": 4.0,
    "min": 1.0,
    "max": 4.0,
```

 版权所有 © 华为云计算技术有限公司

```
  "type": "float",
  "restart": false
 },
 "enable_nestloop": {
  "default": true,
  "type": "bool",
  "restart": false
 }
}
```

待上述配置项配置完成后，可以通过下述命令启动训练：

```
gs_dbmind component xtuner train -f connection.json
```

训练完成后，会在配置项rl_model_path指定的目录中生成模型文件。

tune 模式使用示例

tune模式支持多种算法，包括基于强化学习（Reinforcement Learning, RL）的DDPG算法、基于全局搜索（Global OPtimization algorithm, GOP）算法的贝叶斯优化算法（Bayesian Optimization）以及粒子群算法（Particle Swarm Optimization, PSO）。

与tune模式相关的配置项为：

- tune_strategy：指定选择哪种算法进行调优，支持rl（使用强化学习模型进行调优）、gop（使用全局搜索算法）以及auto（自动选择）。若该参数设置为rl，则rl相关的配置项生效。除前文提到过的train模式下生效的配置项外，test_episode配置项也生效，该配置项表明调优过程的最大回合数，该参数直接影响了调优过程的执行时间（一般地，数值越大越耗时）。
- gop_algorithm：选择何种全局搜索算法，支持bayes以及pso。
- max_iterations：最大迭代轮次，数值越高搜索时间越长，效果往往越好。
- particle_nums：在PSO算法上生效，表示粒子数。
- scenario与tuning_list见上文train模式中的描述。

待上述配置项配置完成后，可以通过下述命令启动调优：

```
gs_dbmind component xtuner tune -f connection.json
```

⚠ 注意

在使用tune和train模式前，用户需要先导入benchmark所需数据并检查benchmark能否正常跑通。调优过程结束后，调优程序会自动恢复调优前的数据库参数配置。

11.3.1.4 获取帮助

启动调优程序之前，可以通过如下命令获取帮助信息：

```
gs_dbmind component xtuner --help
```

输出帮助信息结果如下：

```
usage:  [-h] [--database DATABASE] [--db-user DB_USER] [--db-port DB_PORT] [--db-host DB_HOST] [--host-user HOST_USER] [--host-ssh-port HOST_SSH_PORT] [-f DB_CONFIG_FILE] [-x TUNER_CONFIG_FILE] [-v]
       {train,tune,recommend}

X-Tuner: a self-tuning tool integrated by openGauss.

positional arguments:
```

 版权所有 © 华为云计算技术有限公司

```
  {train,tune,recommend}
                        Train a reinforcement learning model or tune database by model. And also can recommend
best_knobs according to your workload.

optional arguments:
  -h, --help            show this help message and exit
  -f DB_CONFIG_FILE, --db-config-file DB_CONFIG_FILE
                        You can pass a path of configuration file otherwise you should enter database information
by command arguments manually. Please see the template file share/server.json.template.
  -x TUNER_CONFIG_FILE, --tuner-config-file TUNER_CONFIG_FILE
                        This is the path of the core configuration file of the X-Tuner. You can specify the path of
the new configuration file. The default path is /path/to/config/file. You can modify the configuration file to
control the tuning process.
  -v, --version         show program's version number and exit

Database Connection Information:
  --database DATABASE, --db-name DATABASE
                        The name of database where your workload running on.
  --db-user DB_USER     Use this user to login your database. Note that the user must have sufficient
permissions.
  --db-port DB_PORT, --port DB_PORT
                        Use this port to connect with the database.
  --db-host DB_HOST, --host DB_HOST
                        The IP address of your database installation host.
  --host-user HOST_USER
                        The login user of your database installation host.
  --host-ssh-port HOST_SSH_PORT
                        The SSH port of your database installation host.
```

11.3.1.5 命令参考

表 11-31 命令行参数

参数	参数说明	取值范围
mode	指定调优程序运行的模式。	train，tune，recommend
--tuner-config-file, -x	X-Tuner的核心参数配置文件路径，默认路径为安装目录下的xtuner.conf。	-
--db-config-file, -f	调优程序的用于登录到数据库宿主机上的连接信息配置文件路径，若通过该文件配置数据库连接信息，则下述数据库连接信息可省略。	-
--db-name	指定需要调优的数据库名。	-
--db-user	指定以何用户身份登录到调优的数据库上。	-
--port, --db-port	数据库的侦听端口。	1024-65535
--host, --db-host	数据库实例的宿主机IP。	-
--host-user	指定以何用户身份登录到数据库实例的宿主机上，要求改用户名的环境变量中可以找到gsql、gs_ctl等数据库运维工具。	-
--host-ssh-port	数据库实例所在宿主机的SSH端口号，可选，默认为22。	0-65535

版权所有 © 华为云计算技术有限公司

参数	参数说明	取值范围
--help, -h	返回帮助信息。	-
--version, -v	返回当前工具版本号。	-

表 11-32 配置文件中的参数详解

参数名	参数说明	取值范围
logfile	生成的日志存放路径。	-
output_tuning_result	可选，调优结果的保存路径。	-
verbose	是否打印详情。	on, off
recorder_file	调优中间信息的记录日志存放路径。	-
tune_strategy	调优模式下采取哪种策略。	rl, gop, auto
drop_cache	是否在每一个迭代轮次中进行drop cache，drop cache可以使benchmark跑分结果更加稳定。若启动该参数，则需要将登录的系统用户加入到“/etc/sudoers”列表中，同时为其增加NOPASSWD权限（由于该权限可能过高，建议临时启用该权限，调优结束后关闭）。	on, off
used_mem_penalty_term	数据库使用总内存的惩罚系数，用于防止通过无限量占用内存而换取的性能表现。该数值越大，惩罚力度越大。	建议0 ~ 1
rl_algorithm	选择何种RL算法。	ddpg
rl_model_path	RL模型保存或读取路径，包括保存目录名与文件名前缀。在train 模式下该路径用于保存模型，在tune模式下则用于读取模型文件。	-
rl_steps	深度强化学习算法迭代的步数。	-
max_episode_steps	每个回合的最大迭代步数。	-
test_episode	使用RL算法进行调优模式的回合数。	-
gop_algorithm	采取何种全局搜索算法。	bayes, pso
max_iterations	全局搜索算法的最大迭代轮次（并非确定数值，可能会根据实际情况多跑若干轮）。	-
particle_nums	PSO算法下的粒子数量。	-

版权所有 © 华为云计算技术有限公司

参数名	参数说明	取值范围
benchmark_script	使用何种benchmark驱动脚本，该选项指定加载benchmark路径下同名文件，默认支持TPC-C、TPC-H等典型benchmark。	tpcc, tpch, tpcds, sysbench ...
benchmark_path	benchmark 脚本的存储路径，若没有配置该选项，则使用benchmark驱动脚本中的配置。	-
benchmark_cmd	启动benchmark 脚本的命令，若没有配置该选项，则使用benchmark驱动脚本中的配置。	-
benchmark_period	仅对period benchmark有效，表明整个benchmark的测试周期是多少，单位是秒。	-
scenario	用户指定的当前workload所属的类型。	tp, ap, htap
tuning_list	准备调优的参数列表文件，可参考“share/knobs.json.template”文件。	-

11.3.1.6 常见问题处理

- 数据库实例连接失败：请检查数据库实例的情况，是否数据库实例出现了问题或安全权限配置（gs_hba.conf文件中的配置项）不正确。
- 重启失败：请检查数据库实例健康情况，确保数据库实例工作正常。
- 跑TPC-C作业时发现性能越来越慢：TPC-C等高并发场景下的压力测试，往往伴随着大量的数据修改。由于每一次测试并非是幂等的（TPC-C数据库数据量的增加、没有进行vacuum 清理掉失效元组、数据库没有触发checkpoint、没有进行drop cache等），因此一般建议TPC-C等伴随着较多数据写入的benchmark应该每隔一段时间（视具体并发量、执行时长的不同而异）重新导入一次数据。
- TPC-C 跑作业时，TPC-C驱动脚本报异常 “TypeError: float() argument must be a string or a number, not 'NoneType'”（不能将None转换为float类型）：这是因为没有获取到TPC-C的压测返回结果，造成该问题的原因比较多，请首先手动检测是否能够跑通TPC-C并能够获取返回结果。若无上述问题，则建议将 TPC-C 驱动脚本中的命令列表中的 "sleep" 命令延迟时间设得更大一些。

11.3.2 Index-advisor

11.3.2.1 单 query 索引推荐

单query索引推荐功能支持用户在数据库中直接进行操作，本功能基于查询语句的语义信息和数据库的统计信息，对用户输入的单条查询语句生成推荐的索引。本功能涉及的函数接口如下。

版权所有 © 华为云计算技术有限公司

表 11-33 单 query 索引推荐功能的接口

函数名	参数	功能
gs_index_advise	SQL语句字符串。	针对单条查询语句生成推荐索引。

说明

- 本功能仅支持单条SELECT、INSERT、DELETE、UPDATE类型的语句，不支持其他类型的SQL语句。
- 本功能使用优化器采样结果，用户需要保证最近analyze过，否则优化器结果不准确。
- 本功能暂不支持段页式表、普通视图、物化视图、全局临时表、二级分区表以及密态数据库。
- 如果对ustore表相关语句进行索引推荐，本功能可能无法保证结果的准确性。

使用方法

使用上述函数，获取针对该query生成的推荐索引，推荐结果由索引的表名和列名组成。

例如：

```
gaussdb=# select "table", "column" from gs_index_advise('SELECT c_discount from bmsql_customer where c_w_id = 10');
     table      |  column
----------------+----------
 bmsql_customer | c_w_id
(1 row)
```

上述结果表明：应当在 bmsql_customer 的 c_w_id 列上创建索引，例如可以通过下述SQL语句创建索引：

```
CREATE INDEX idx on bmsql_customer(c_w_id);
```

某些SQL语句，也可能被推荐创建联合索引，例如：

```
gaussdb=# select "table", "column" from gs_index_advise('select name, age, gender from t1 where age >= 18 and age < 35 and gender = ''f'';');
 table | column
-------+------------
 t1    | age, gender
(1 row)
```

则上述语句表明应该在表 t1 上创建一个联合索引 (age, gender)， 则可以通过下述命令创建：

```
CREATE INDEX idx1 on t1(age, gender);
```

针对分区表可推荐具体索引类型，例如：

```
gaussdb=# select "table", "column", "indextype" from gs_index_advise('select name, age, gender from range_table where age = 20;');
 table | column | indextype
-------+--------+-----------
 t1    | age    | global
(1 row)
```


版权所有 © 华为云计算技术有限公司

说明

系统函数gs_index_advise()的参数是文本型，如果参数中存在如单引号（'）等特殊字符，可以使用单引号（'）进行转义，可参考上述示例。

11.3.2.2 虚拟索引

虚拟索引功能支持用户在数据库中直接进行操作，本功能将模拟真实索引的建立，避免真实索引创建所需的时间和空间开销，用户基于虚拟索引，可通过优化器评估该索引对指定查询语句的代价影响。

本功能涉及的系统函数接口如下表所示：

表 11-34 虚拟索引功能的接口

函数名	参数	功能
hypopg_create_index	• 参数1：创建索引语句的字符串。 • 参数2：虚拟索引级别，可选参数，可指定global或session，默认为global。	创建虚拟索引。
hypopg_display_index	虚拟索引级别，可选参数，可通过参数指定global或session，默认为global。	显示所有创建的虚拟索引信息。
hypopg_drop_index	索引的oid。	删除指定的虚拟索引。
hypopg_reset_index	虚拟索引级别，可选参数，可通过参数指定global或session，默认为global。	清除所有虚拟索引。
hypopg_estimate_size	索引的oid。	估计指定索引创建所需的空间大小。

本功能涉及的GUC参数如下：

表 11-35 虚拟索引功能的 GUC 参数

参数名	功能	默认值
enable_hypo_index	是否开启虚拟索引功能。	off

使用步骤

案例一：使用虚拟索引，调优等值查询

版权所有 © 华为云计算技术有限公司

在此案例中，存在表bmsql_customer，该表是TPC-C benchmark中的一张表，此处演示在该表的c_w_id列上创建一个索引，是否可以提升某个等值查询的性能，如果该索引被使用了，则预估执行代价(cost)是多少。

步骤1 使用函数hypopg_create_index创建虚拟索引。例如：

```
gaussdb=# select * from hypopg_create_index('create index on bmsql_customer(c_w_id)');
 indexrelid |              indexname
------------+---------------------------------------
     329726 | <329726>btree_bmsql_customer_c_w_id
(1 row)
```

步骤2 开启GUC参数enable_hypo_index，该参数控制数据库的优化器进行EXPLAIN时是否考虑创建的虚拟索引。通过对特定的查询语句执行explain，用户可根据优化器给出的执行计划评估该索引是否能够提升该查询语句的执行效率。例如：

```
gaussdb=# set enable_hypo_index = on;
SET
```

开启GUC参数前，执行EXPLAIN + 查询语句：

```
gaussdb=# explain SELECT c_discount from bmsql_customer where c_w_id = 10;
                            QUERY PLAN
------------------------------------------------------------------------
 Seq Scan on bmsql_customer  (cost=0.00..52963.06 rows=31224 width=4)
   Filter: (c_w_id = 10)
(2 rows)
```

开启GUC参数后，执行EXPLAIN + 查询语句：

```
gaussdb=# explain SELECT c_discount from bmsql_customer where c_w_id = 10;
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 [Bypass]
 Index Scan using <329726>btree_bmsql_customer_c_w_id on bmsql_customer  (cost=0.00..39678.69
rows=31224 width=4)
   Index Cond: (c_w_id = 10)
(3 rows)
```

通过对比两个执行计划可以观察到，该索引预计会降低指定查询语句的执行代价，用户可考虑创建对应的真实索引。

步骤3 （可选）使用函数hypopg_display_index展示所有创建过的虚拟索引。例如：

```
gaussdb=# select * from hypopg_display_index();
                indexname                 | indexrelid |     table      |      column      |                    indexdef
------------------------------------------+------------+----------------+------------------+
+-----------------------------------------------------------
 <329726>btree_bmsql_customer_c_w_id       |     329726 | bmsql_customer | (c_w_id)         |CREATE INDEX
ON bmsql_customer USING btree (c_w_id)
 <329729>btree_bmsql_customer_c_d_id_c_w_id |     329729 | bmsql_customer | (c_d_id, c_w_id) |CREATE
INDEX ON bmsql_customer USING btree (c_d_id, c_w_id)
(2 rows)
```

步骤4 （可选）使用函数hypopg_estimate_size估计虚拟索引创建所需的空间大小（单位：字节）。例如：

```
gaussdb=# select * from hypopg_estimate_size(329729);
 hypopg_estimate_size
----------------------
             15687680
(1 row)
```

步骤5 删除虚拟索引。

使用函数hypopg_drop_index删除指定oid的虚拟索引。例如：

```
gaussdb=# select * from hypopg_drop_index(329726);
 hypopg_drop_index
-------------------
```

 版权所有 © 华为云计算技术有限公司

```
 t
(1 row)
```

使用函数hypopg_reset_index一次性清除所有创建的虚拟索引。例如：

```
gaussdb=# select * from hypopg_reset_index();
 hypopg_reset_index
--------------------


(1 row)
```

----结束

案例二：虚拟索引联合Hint，预测调优效果

Hint可以手动要求数据库优化器使用某种方式生成执行计划，因此，对于某些数据库优化器难以生成最优执行计划的场景，可以手动指定执行计划。例如对某张表中的数据进行扫描操作（Scan），可以采用tablescan,、indexscan,、indexonlyscan， 其分别对应了表扫描、索引扫描、覆盖索引扫描。对于后两种扫描形式，必须要求先在数据库表上存在索引才可以操作。而虚拟索引则可以实现在不创建索引的情况下，测试某个索引扫描的效果。

步骤1 创建一张表t1，并生成一定量数据，供后续测试。

```
create table t1 (id int, name text);
insert into t1 select generate_series(0, 100000), 'test';
analyze t1;
```

步骤2 测试当前优化器默认的范围检索执行计划，并获取其总代价；由于没有创建索引，该SQL语句使用的是全表扫描（SeqScan）。

```
explain select * from t1 where id > 1;
```

步骤3 在t1表的id列上新建虚拟索引。

```
-- 开启参数，以便后续执行explain时能够采用虚拟索引
set enable_hypo_index = on;
-- 创建session级别虚拟索引，该session退出后，这个虚拟索引信息也会被自动清理掉
select hypopg_create_index('create index on t1(id)','session');
```

步骤4 通过explain语句，查看该SQL语句是否能够采用该索引；由于该列的distinct值很大，且涉及回表，优化器默认不会采用该索引，该语句执行计划与步骤2无变化，仍是全表扫描（SeqScan）。

```
explain select * from t1 where id > 1;
```

步骤5 通过hint操作，手动要求走索引扫描，查看能否成功；由于指定了hint, 且存在该索引（尽管是虚拟的），仍然可以通过explain看到优化器使用了索引扫描 IndexScan。

```
-- 其中<57762>btree_t1_id是自动生成的虚拟索引名，实际操作中以创建虚拟索引时的返回值为准
explain select /*+ indexscan(t1 "<57762>btree_t1_id") */ * from t1 where id > 1;
```

----结束

 版权所有 © 华为云计算技术有限公司

说明

- 执行EXPLAIN ANALYZE不会涉及虚拟索引功能。
- 开启虚拟索引功能并执行EXPLAIN语句时，可以生成创建虚拟索引之后的执行计划；同时，indexscan/indexonlyscan hint支持虚拟索引。
- 会话级别虚拟索引在各个会话间的设置互不影响，关闭会话后将被清空。
- 与真实索引不同，虚拟索引的相关操作不可回滚。
- 虚拟索引相关函数，不支持dblink远程调用。
- 本功能暂不支持视图、物化视图。
- 虚拟索引当前不支持FQS执行计划。
- 虚拟索引支持分布式全局二级索引（GSI），但存在如下约束：
  - 虚拟索引GSI不支持段页式、bucket表、全局临时表、物化视图。
  - 虚拟索引GSI不支持创建部分索引、表达式索引。
  - 虚拟索引GSI不支持hint使用。

11.3.2.3 workload 级别索引推荐

对于workload级别的索引推荐，用户可通过运行数据库外的脚本使用此功能，本功能将包含有多条DML语句的workload作为输入，最终生成一批可对整体workload的执行表现进行优化的索引。同时，本功能提供从日志中抽取业务数据SQL流水的功能。

前提条件

- 数据库状态正常、客户端能够正常连接。
- 当前执行用户下安装有gsql工具，该工具路径已被加入到PATH环境变量中。
- 具备Python3.7的环境。
- 若使用本功能提供的业务数据抽取功能，需提前将要收集的节点的GUC参数按如下设置：
  - log_min_duration_statement = 0
  - log_statement= 'all'

    说明

    业务数据抽取完毕建议将上述GUC参数复原，否则容易导致日志文件膨胀。

业务数据抽取脚本使用步骤

步骤1 按前提条件中要求设置相关GUC参数。

步骤2 运行命令如下：

```
gs_dbmind component extract_log [-h] [-d DATABASE] [-U DB_USER] [--start-time START_TIME] [--sql-amount SQL_AMOUNT] [--statement] [--max-reserved-period MAX_RESERVED_PERIOD] [--max-template-num MAX_TEMPLATE_NUM] [--json] log_dir file line_prefix
```

其中的输入参数依次为：

- DATABASE：（可选）数据库名称，不指定默认所有数据库。
- DB_USER：（可选）用户名称，不指定默认所有用户。
- START_TIME：（可选）日志收集的开始时间，不指定默认所有文件。
- SQL_AMOUNT：（可选）收集SQL数量的最大值，不指定默认收集所有sql。

版权所有 © 华为云计算技术有限公司

- statement：（可选）表示收集gs_log日志中statement标识开头的SQL，不指定默认不收集。
- MAX_RESERVED_PERIOD：（可选）指定json模式下，增量收集日志中保留的模板的最大的更新时长，不指定默认都保留，单位：天。
- MAX_TEMPLATE_NUM：（可选）指定json模式下保留的最大模板数量，不指定默认都保留。
- json：（可选）指定收集日志的文件存储格式为SQL归一化后的json，不指定默认格式每条SQL占一行。
- log_dir：gs_log的存放目录。
- file：输出SQL流水文件的保存路径，即抽取出的业务数据存放的文件路径。
- line_prefix：指定每条日志信息的前缀格式，可通过show log_line_prefix查询。

使用示例：

```
gs_dbmind component extract_log $GAUSSLOG/gs_log/dn_6001 sql_log.txt '%m %c %d %p %a %x %n %e' -d testdb -U omm --start_time '2021-07-06 00:00:00' --statement
```

说明

若指定-d/-U参数，日志打印每条日志信息的前缀格式需包含%d、%u，若需要抽取事务，必须指定%p，详见log_line_prefix参数。max_template_num参数设置建议不超5000条，避免workload索引推荐执行时间过长。

步骤3 将步骤1中设置的GUC参数还原为设置前的值。

----结束

索引推荐脚本使用步骤

步骤1 准备好包含有多条DML语句的文件作为输入的workload，文件中每条语句占据一行。用户可从数据库的离线日志中获得历史的业务语句。

步骤2 运行命令如下：

```
gs_dbmind component index_advisor [-h] [--db-host DB_HOST] [-U DB_USER, --db-user DB_USER] [--schema SCHEMA] [--min-improved-rate MIN_IMPROVED_RATE] [--max-index-columns MAX_INDEX_COLUMNS] [--min-reltuples MIN_RELTUPLES] [--max-n-distinct MAX_N_DISTINCT] [--max-index-num MAX_INDEX_NUM] [--max-index-storage MAX_INDEX_STORAGE] [--multi-iter-mode] [--multi-node] [--json] [--driver] [--show-detail] [--show-benefits] [--advise_gsi] [multi_thread_num MULTI_THREAD_NUM] db_port database file
```

密码通过管道输入或交互式输入，对于免密用户，任意输入都可通过检验。

表 11-36 命令行参数

参数	参数说明	取值范围
-h	返回帮助信息。	-
--db-host DB_HOST	（可选）连接数据库的主机号。	-
-U DB_USER	（可选）连接数据库的用户名。	-
--schema SCHEMA	模式名称。	-
--max-index-num MAX_INDEX_NUM	（可选）最大的索引推荐数目。	>=1

 版权所有 © 华为云计算技术有限公司

参数	参数说明	取值范围
--max-index-storage MAX_INDEX_STORAGE	（可选）最大的索引集合空间大小。	>=1
--multi-iter-mode	（可选）算法模式，可通过是否设置该参数来切换算法。	-
--max-n-distinct MAX_N_DISTINCT	（1/distinct数）的最大值，默认为0.01。	0-1
--min-improved-rate MIN_IMPROVED_RATE	索引的最大提升幅度，默认为0.1，即提升10%。	0-1
--max-index-columns MAX_INDEX_COLUMNS	联合索引的最大列数（默认为4）。	>=1
--min-reltuples MIN_RELTUPLES	表的最小行数，默认为10000。	>0
--multi-node	（可选）指定是否为分布式数据库。	-
--json	（可选）指定workload语句的文件路径格式为SQL归一化后的json。	-
--driver	（可选）指定是否使用python驱动器连接数据库，默认gsql连接。	-
--show-detail	（可选）是否显示当前推荐索引集合的详细优化信息。	-
--show-benefits	（可选） 是否显示收益信息。	-
db_port	连接数据库的端口号。	-
database	连接数据库的名字。	-
file	包含workload语句的文件路径。	-
advise_gsi	（可选）推荐分布式全局二级索引（GSI）。	-
multi_thread_num	（可选）以多线程运行脚本，指定线程数。	[1,64]

例如：

```
gs_dbmind component index_advisor 6001 testdb tpcds_log.txt --schema public --max_index_num 10
```

结果在屏幕输出，包含候选索引、推荐索引、已创建索引、无用索引（该给定的workload里面没有用到系统中的索引列表）、冗余索引（当前系统中重复创建的索引）以及历史有效索引，如下：

```
################################ Generate candidate indexes
################################
table:  public.catalog_returns columns:  cr_return_amount
table:  public.catalog_sales columns:  cs_item_sk
table:  public.catalog_sales columns:  cs_sold_date_sk
table:  public.customer_address columns:  ca_city type:  global
table:  public.customer_address columns:  ca_state, ca_county type:  global
table:  public.customer_demographics columns:  cd_demo_sk type:  local
```


版权所有 © 华为云计算技术有限公司

```
table:  public.date_dim columns:  d_month_seq type:  global
table:  public.date_dim columns:  d_year type:  global
table:  public.date_dim columns:  d_date_sk type:  local
table:  public.date_dim columns:  d_month_seq type:  local
table:  public.date_dim columns:  d_year type:  local
table:  public.item columns:  i_class type:  global
table:  public.item columns:  i_manager_id, i_brand_id type:  global
table:  public.item columns:  i_manager_id, i_category_id type:  global
table:  public.item columns:  i_manufact_id type:  global
table:  public.item columns:  i_product_name type:  global
table:  public.store_returns columns:  sr_cdemo_sk
table:  public.store_returns columns:  sr_reason_sk
table:  public.store_returns columns:  sr_return_amt
table:  public.store_sales columns:  ss_item_sk, ss_sold_date_sk
table:  public.store_sales columns:  ss_store_sk
table:  public.time_dim columns:  t_time_sk type:  local
table:  public.web_returns columns:  wr_return_amt
table:  public.web_sales columns:  ws_item_sk
table:  public.web_sales columns:  ws_web_page_sk, ws_ship_hdemo_sk, ws_sold_time_sk
################################## Determine optimal indexes
##################################
CREATE INDEX idx_catalog_sales_cs_item_sk ON public.catalog_sales(cs_item_sk);
CREATE INDEX idx_catalog_sales_cs_sold_date_sk ON public.catalog_sales(cs_sold_date_sk);
CREATE INDEX idx_customer_demographics_local_cd_demo_sk ON
public.customer_demographics(cd_demo_sk) local;
CREATE INDEX idx_item_global_i_manufact_id ON public.item(i_manufact_id) global;
CREATE INDEX idx_store_returns_sr_cdemo_sk ON public.store_returns(sr_cdemo_sk);
CREATE INDEX idx_store_sales_ss_item_sk_ss_sold_date_sk ON public.store_sales(ss_item_sk,
ss_sold_date_sk);
CREATE INDEX idx_store_sales_ss_store_sk ON public.store_sales(ss_store_sk);
CREATE INDEX idx_web_sales_ws_web_page_sk_ws_ship_hdemo_sk_ws_sold_time_sk ON
public.web_sales(ws_web_page_sk, ws_ship_hdemo_sk, ws_sold_time_sk);
################################## Created indexes
##################################
public: CREATE UNIQUE INDEX ship_mode_pkey ON ship_mode USING btree (sm_ship_mode_sk)
LOCAL(PARTITION p_list_15_sm_ship_mode_sk_idx, PARTITION p_list_14_sm_ship_mode_sk_idx, PARTITION
p_list_13_sm_ship_mode_sk_idx, PARTITION p_list_12_sm_ship_mode_sk_idx, PARTITION
p_list_11_sm_ship_mode_sk_idx, PARTITION p_list_10_sm_ship_mode_sk_idx, PARTITION
p_list_9_sm_ship_mode_sk_idx, PARTITION p_list_8_sm_ship_mode_sk_idx, PARTITION
p_list_7_sm_ship_mode_sk_idx, PARTITION p_list_6_sm_ship_mode_sk_idx, PARTITION
p_list_5_sm_ship_mode_sk_idx, PARTITION p_list_4_sm_ship_mode_sk_idx, PARTITION
p_list_3_sm_ship_mode_sk_idx, PARTITION p_list_2_sm_ship_mode_sk_idx, PARTITION
p_list_1_sm_ship_mode_sk_idx)  TABLESPACE pg_default;
public: CREATE INDEX temptable_int2_int3_int4_idx ON temptable USING btree (int2, int3, int4)
TABLESPACE pg_default;
public: CREATE INDEX temptable_int2_int3_idx ON temptable USING btree (int2, int3) TABLESPACE
pg_default;
public: CREATE INDEX temptable_int1_int2_int3_idx ON temptable USING btree (int1, int2, int3)
TABLESPACE pg_default;
public: CREATE INDEX temptable_int1_int2_idx ON temptable USING btree (int1, int2) TABLESPACE
pg_default;
public: CREATE INDEX temptable_int1_idx ON temptable USING btree (int1) TABLESPACE pg_default;
################################## Current workload useless indexes
##################################
DROP INDEX temptable_int2_int3_int4_idx;
DROP INDEX temptable_int2_int3_idx;
DROP INDEX temptable_int1_int2_int3_idx;
DROP INDEX temptable_int1_int2_idx;
DROP INDEX temptable_int1_idx;
################################## Redundant indexes
##################################
DROP INDEX public.test1_age_idx;(CREATE INDEX test1_age_idx ON test1 USING btree (age) TABLESPACE
pg_default)
Related indexes:
        CREATE INDEX test1_age_id_idx ON test1 USING btree (age, id) TABLESPACE pg_default

DROP INDEX public.test1_id_idx;(CREATE INDEX test1_id_idx ON test1 USING btree (id) TABLESPACE
pg_default)
Related indexes:
```

 版权所有 © 华为云计算技术有限公司

```
    CREATE INDEX test1_id_age_idx ON test1 USING btree (id, age) TABLESPACE pg_default
################################ Historical effective indexes
################################
CREATE INDEX idx_temptable_int2 ON ztt_test.temptable(int2);
CREATE INDEX idx_item_i_manufact_id ON public.item(i_manufact_id);
CREATE INDEX idx_item_i_color ON public.item(i_color);
```

----结束

⚠ 注意

如前文所述，"Current workload useless indexes", "Redundant indexes" 分别表示无用索引和冗余索引，判断依据是给定的workload. 由于workload中的SQL语句可能涉及不全（例如日志报错、没有捕获到等原因导致的），故该结论仅供提示，用户需要根据自己的业务逻辑进行排查，防止错误删除。

说明

与单query索引推荐相同，本功能暂不支持段页式表、普通视图、物化视图、全局临时表、二级分区表以及密态数据库。

11.3.3 Slow Query Diagnosis

11.3.3.1 概述

慢SQL一直是数据运维中的痛点问题，如何有效诊断慢SQL根因是当前一大难题，工具结合GaussDB自身特点融合了现网DBA慢SQL诊断经验，支持慢SQL根因分析，能同时按照可能性大小输出多个根因并提供针对性的建议。

11.3.3.2 环境部署

- 数据库运行正常。
- 指标采集系统运行正常。

11.3.3.3 使用指导

目前分布式只支持如下慢SQL根因

序号	根因	根因解释	补充说明
1	LOCK_CONTENTION	锁竞争。	语句执行期间是否被阻塞，导致单SQL执行较慢。
2	HEAVY_SCAN_OPERATOR	扫描算子代价较大。	执行计划中扫描算子代价较大，导致单SQL执行较慢。
3	UNUSED_AND_REDUNDANT_INDEX	无用/冗余索引。	表中存在无用/冗余索引，影响插入更新语句性能。
4	WORKLOAD_CONTENTION	数据库负载集中。	数据库负载集中导致实例整体执行性能较差。

版权所有 © 华为云计算技术有限公司

序号	根因	根因解释	补充说明
5	CPU_RESOURCE_CONTENTION	系统CPU负载集中。	由于外部进程等其他原因导致CPU资源紧张，实例整体SQL执行性能较差。
6	IO_RESOURCE_CONTENTION	系统IO资源集中。	由于外部进程等其他原因导致I/O资源紧张，实例整体SQL执行性能较差。
7	MEMORY_RESOURCE_CONTENTION	系统内存资源集中。	由于外部进程等其他原因导致内存资源紧张，实例整体SQL执行性能较差。
8	ABNORMAL_NETWORK_STATUS	异常网络状态。	网络异常，导致SQL执行性能较差。
9	WAIT_EVENT	等待事件。	SQL执行期间的等待事件信息。
10	MISSING_INDEXES	缺少索引。	缺失索引导致单SQL执行性能较差。
11	POOR_JOIN_PERFORMANCE	join代价较大。	join算子代价较大，影响SQL语句执行性能。
12	REMOTE_QUERY	SQL不下推。	SQL语句不下推，在CN上执行，导致执行效率慢。
13	LARGE_BROADCAST	大表broadcast。	大表广播到其他DN节点，导致执行效率慢。
14	NO_ROOT_CAUSE_FOUND	没有发现根因。	没有发现当前慢SQL的根因。

假设用户已经初始化配置文件目录confpath，则可以通过下述命令实现本特性的功能：

- 仅启动慢SQL诊断功能（慢SQL诊断根因数量由算法运行结果决定，数量不固定），启动命令如下（更多用法参考对service子命令的说明）：
  ```
  gs_dbmind service start -c confpath --only-run slow_query_diagnosis
  ```
- 用户查询慢SQL诊断历史，命令如下：
  ```
  gs_dbmind component slow_query_diagnosis show -c confpath --instance instance --query SQL --start-time timestamps0 --end-time timestamps1
  ```
- 用户交互式诊断慢SQL，命令如下：
  ```
  gs_dbmind component slow_query_diagnosis diagnosis -c confpath --database dbname --schema schema_name --query SQL
  ```
- 启用慢SQL诊断后台任务，首先将opengauss-exporter下的pg_sql_statement_history开启，具体步骤如下：
  ```
  1、停止opengauss-exporter进程；
  2、进入dbmind/components/opengauss_exporter/yamls/statements.yml中，将该指标的status设置为enable；
  3、将slow_query_diagnosis加入到配置文件dbmind.conf下TIMED_TASK的task中，任务之间用逗号隔离；
  4、重启opengauss-exporter进程；
  5、运行 gs_dbmind service reload -c confpath 命令，启动慢SQL诊断后台任务；
  ```
- 用户手动清理历史预测结果，命令如下：
  ```
  gs_dbmind component slow_query_diagnosis clean -c confpath --retention-days DAYS
  ```

 版权所有 © 华为云计算技术有限公司

- 停止已启动的服务，命令如下：
  ```
  gs_dbmind service stop -c confpath
  ```

11.3.3.4 获取帮助

模块命令行说明：

```
gs_dbmind component slow_query_diagnosis --help
usage:  [-h] -c DIRECTORY [--instance INSTANCE] [--database DATABASE] [--schema SCHEMA] [--query
SLOW_QUERY]
       [--start-time TIMESTAMP_IN_MICROSECONDS] [--end-time TIMESTAMP_IN_MICROSECONDS]
       [--retention-days DAYS]
       {show,clean,diagnosis}

Slow Query Diagnosis: Analyse the root cause of slow query

positional arguments:
  {show,clean,diagnosis}
                        choose a functionality to perform

optional arguments:
  -h, --help            show this help message and exit
  -c DIRECTORY, --conf DIRECTORY
                        Set the directory of configuration files
  --instance INSTANCE   Set the instance of slow query. Using in show.
  --database DATABASE   Set the name of database
  --schema SCHEMA       Set the schema of database
  --query SLOW_QUERY    Set a slow query you want to retrieve
  --start-time TIMESTAMP_IN_MICROSECONDS
                        Set the start time of a slow SQL diagnosis result to be retrieved
  --end-time TIMESTAMP_IN_MICROSECONDS
                        Set the end time of a slow SQL diagnosis result to be retrieved
  --retention-days DAYS
                        clear historical diagnosis results and set the maximum number of days to retain data
```

11.3.3.5 命令参考

表 11-37 gs_dbmind component slow_query_diagnosis 命令行说明

参数	参数说明	取值范围
-h, --help	帮助命令。	-
action	动作参数。	• show：结果展示 • clean：清理结果 • diagnosis：交互诊断
-c，--conf	配置目录。	-
--instance	慢SQL实例地址，action为show时使用。	数据库实例地址，样例：127.0.0.1:5432
--query	慢SQL文本。	*
（可选）--start-time	显示开始时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S. action为diagnosis或show时使用。	正整数或日期时间格式

版权所有 © 华为云计算技术有限公司

参数	参数说明	取值范围
（可选）--end-time	显示结束时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S。 action为diagnosis或show时使用。	正整数或日期时间格式
--retention-days	清理天数及结果，action为clean时使用。	实数（当其小于等于0时，会将结果全部删除）
--database	慢SQL关联的数据库。	-
(可选)--schema	慢SQL关联的schema，默认为public。	-

11.3.3.6 注意事项说明

DBMind当前支持三种模式的慢SQL诊断能力，分别是：后台定时任务、命令行交互和API调用，下面对这三种方式的注意事项进行说明。

说明

如果没有特别指出具体使用模式，则表示该项三种模式都适用。

- 当前慢SQL诊断定时任务依赖opengauss-exporter指标采集慢SQL信息，该指标默认关闭，如果该指标采集不启用则慢SQL诊断功能无法正常运行。启动慢SQL采集的方法是手动将opengauss_exporter/yamls/statements.yaml中pg_sql_statement_history指标status设置成enable，再重启opengauss-exporter进程。
- 慢SQL诊断定时任务间隔支持用户配置，用户需要修改配置文件dbmind.conf中TIMED_TASK的slow_query_diagnosis_interval，然后执行reload操作才能生效，命令参考**service子命令**。
- 慢SQL诊断定时任务依赖dbe_perf.statement_history视图，当其数据量较大时可能出现查询超时，此时慢SQL后台诊断任务不会生成新的诊断结果。用户可以根据数据库状况设置该指标的超时时间timeout，以免超时时间太小不能采集到指标，具体修改的方法可以参见**Prometheus Exporter组件**。
- 在慢SQL诊断定时任务中，由于慢SQL流水数据源dbe_perf.statement_history视图中的query字段可能会出现截断导致query不完整，此时如果没有提供执行计划则不能进行诊断。
- 慢SQL诊断定时任务的采集基于opengauss-exporter实现，服务运行时不能保证全量采集，可能会遗漏部分慢SQL数据。
- 使用慢SQL命令行交互诊断功能时，工具基于RPC和数据采集服务获取必要的数据，因此如果RPC和数据采集服务没启动则无法诊断。
- 在使用慢SQL交互诊断功能时，工具会对传入的数据库名和schema进行检测，如果数据库或schema不存在则会进行提示，不能正常进行诊断。
- 在使用慢SQL诊断功能时，工具会对传入的数据库名和schema进行检测，如果其中包含非法字符则会进行提示并拒绝诊断。特殊情况说明：为适配云侧使用场景（将dbe_perf.statement_history作为SQL采集数据源时，其schema可能带'$'字

 版权所有 © 华为云计算技术有限公司

符，例如$user,public ），在调用慢SQL诊断API接口时会对该场景进行特殊处理以支持诊断。

- 慢SQL诊断过程需要获取诊断语句的执行计划，在调用慢SQL诊断API接口时建议传递执行计划（对应query_plan参数），如果前端没有传递执行计划则工具会主动获取执行计划内容，此时需要确保执行诊断的用户具有获取query执行计划的权限。另外获取执行计划过程中如果SQL属于归一化SQL，则工具会基于PBE方式获取执行计划，获取过程中如果由于语法结构不支持、SQL截断等原因导致获取失败则会诊断失败，当SQL不属于归一化SQL时如果SQL截断则也不能正常诊断。
- 在使用慢SQL诊断API接口时，如果在用户传入的db_name不存在则无法诊断，另外当工具在传入的schema_name下获取执行计划失败时会自动其他schema下尝试获取执行计划直到成功获取为止，否则诊断失败。
- 当DBMind纳管多个实例时，使用命令行交互式慢SQL诊断功能时会提示用户选择哪一个实例，然后再进行诊断动作。
- 当前慢SQL诊断只支持DML语句。
- 慢SQL诊断每次只能诊断一条语句，如果输入多条则只会对第一条进行诊断。
- 当前不对系统视图根因进行诊断，根因统一为DATABASE_VIEW。
- 慢SQL诊断过程中如果用户不传schema信息，则schema默认为public。
- 由于reprocessing-exporter在采集磁盘占用率时只支持EXT和XFS文件系统，因此如果非上述文件系统磁盘超过阈值时不会出现磁盘空间不足的根因。
- 如果需要诊断LOCK_CONTENTION或WAIT_EVENT根因，需要使用API调用的方式，并传入debug_query_id参数。在分布式情况下，是使用dbe_perf.get_global_slow_sql_by_timestamp(start_time, end_time)函数查询全局慢SQL的等待事件，为了避免该操作过于耗时，查询时间范围为8小时前至当前时间。

11.3.4 SQLdiag

SQLdiag是GaussDB中SQL语句执行时长预测工具。现有的预测技术主要基于执行计划的预测方法，但这些预测方案仅适用于分析型场景且可以获取执行计划的任务，对于OLTP或者HTAP这样的快速、简单查询是没有太多使用价值的。与上述方案不同，SQLdiag着眼于数据库的历史SQL语句，通过对历史SQL语句的执行表现进行总结归纳，将之再用于推断新的未知业务上。由于短时间内数据库SQL语句执行时长不会有太大的差距，SQLdiag可以从历史数据中检测出与已执行SQL语句相似的语句结果集，并基于SQL向量化技术通过SQL模板化和深度学习这两种方法来预测SQL语句执行时长。本工具有如下优点：

1. 不需要SQL语句的执行计划，对数据库性能不会有任何的影响。
2. 使用场景广泛，目前业内的很多算法局限性比较高，比如只适用于OLTP或者其他场景，而SQLdiag使用场景广泛。
3. 该工具容易理解，只需要简单的操作，就可以训练出自己的预测模型。

本工具的典型应用场景是对一批即将上线的SQL语句进行透视，提前识别风险。

11.3.4.1 概述

SQLdiag是一个SQL语句执行时间预测工具，通过模板化方法或者深度学习方法，实现在不获取SQL语句执行计划的前提下，依据语句逻辑相似度与历史执行记录，预测SQL语句的执行时间并以此发现异常SQL。

 版权所有 © 华为云计算技术有限公司

说明

本功能主要适配集中式部署的数据库，对于分布式数据库可能不能做到全量支持。

11.3.4.2 使用指导

前提条件与使用事项

- 需要保证用户提供训练数据。
- 如果用户通过提供的工具收集训练数据，则需要启用WDR功能，涉及到的参数为track_stmt_stat_level和log_min_duration_statement，具体情况见下面小结。
- 为保证预测准确率，用户提供的历史语句日志应尽可能全面并具有代表性。
- SQLdiag运行时依赖第三方工具，安装方式请参考**环境配置**，若用户环境与第三方工具存在包冲突等情况，可由用户根据实际情况进行处理，第三方工具模块详情如下：

```
tensorflow>=2.1.0
gensim==3.8.3
```

SQL 流水采集方法

说明

SQL流水采集方法暂不支持分布式。

本工具需要用户提前准备数据，训练数据格式如下，每个样本通过换行符分隔：

```
SQL,EXECUTION_TIME
```

预测数据格式如下：

```
SQL
```

其中SQL表示**SQL语句的文本**，EXECUTION_TIME表示**SQL语句的执行时间**，样例数据见sample_data中的train.csv和predict.csv。

用户可以按照要求格式自己收集训练数据，工具也提供了脚本自动采集（load_sql_from_rd），该脚本基于WDR报告获取SQL信息，涉及到的参数有log_min_duration_statement和track_stmt_stat_level：

- 其中log_min_duration_statement表示慢SQL阈值，如果为0则全量收集，时间单位为毫秒；
- track_stmt_stat_level表示信息捕获的级别，建议设置为track_stmt_stat_level='L0,L0'

参数开启后，可能占用一定的系统资源，但一般不大。持续的高并发场景可能产生5%以内的损耗，数据库并发较低的场景，性能损耗可忽略。下述脚本存在于sqldiag根目录（DBMINDPATH/dbmind/components/sqldiag）中。

```
使用脚本获取训练集方式：
usage: load_sql_from_wdr.py [-h] --port PORT --start-time START_TIME --finish-time FINISH_TIME [--save-path SAVE_PATH]

Fetch SQL information based on WDR.

optional arguments:
  -h, --help            show this help message and exit
  --port PORT, --db-port PORT
                        Port of database service.
  --start-time START_TIME
```


版权所有 © 华为云计算技术有限公司

```
                    Start time of query
 --finish-time FINISH_TIME, --end-time FINISH_TIME
                    Finish time of query
 --save-path SAVE_PATH
                    Path to save result

例如：
   python load_sql_from_wdr.py --start-time "2021-04-25 00:00:00" --finish-time "2021-04-26 14:00:00" --
port 5432  --save-path ./data.csv
```

操作步骤

步骤1 提供历史日志以供模型训练

步骤2 进行训练与预测操作。

```
基于模板法的训练与预测：
  gs_dbmind component sqldiag [train, predict] -f FILE --model template --model-path
template_model_path --config-file config_path
基于DNN的训练与预测：
  gs_dbmind component sqldiag [train, predict] -f FILE --model dnn --model-path dnn_model_path --
config-file config_path
```

----结束

使用方法示例

使用提供的测试数据进行模板化训练：

```
gs_dbmind component sqldiag train -f DBMINDPATH/dbmind/components/sqldiag/sample_data/train.csv --
model template --model-path ./template --config-file config_path
```

使用提供的测试数据进行模板化预测：

```
gs_dbmind component sqldiag predict -f DBMINDPATH/dbmind/components/sqldiag/sample_data/
predict.csv --model template --model-path ./template --predicted-file ./result/t_result --config-file
config_path
```

使用提供的测试数据进行模板化模型更新：

```
gs_dbmind component sqldiag finetune -f DBMINDPATH/dbmind/components/sqldiag/sample_data/
train.csv --model template --model-path ./template --config-file config_path
```

使用提供的测试数据进行DNN训练：

```
gs_dbmind component sqldiag train -f DBMINDPATH/dbmind/components/sqldiag/sample_data/train.csv --
model dnn --model-path ./dnn_model --config-file config_path
```

使用提供的测试数据进行DNN预测：

```
gs_dbmind component sqldiag predict -f DBMINDPATH/dbmind/components/sqldiag/sample_data/
predict.csv --model dnn --model-path ./dnn_model --predicted-file --config-file config_path
```

使用提供的测试数据进行DNN模型更新：

```
gs_dbmind component sqldiag finetune -f DBMINDPATH/dbmind/components/sqldiag/sample_data/
train.csv --model dnn --model-path ./dnn_model --config-file config_path
```

11.3.4.3 获取帮助

使用SQLdiag工具前，您可以通过以下指令获取帮助。

```
gs_dbmind component sqldiag --help
```

显示如下帮助信息：

```
usage:   [-h] [-f CSV_FILE] [--predicted-file PREDICTED_FILE]
              [--model {template,dnn}] [--query QUERY] [--threshold THRESHOLD] --model-file
```


版权所有 © 华为云计算技术有限公司

```
MODEL_FILEPATH
            [--config-file CONFIG_FILE]
            {train,predict,finetune}

SQLdiag integrated by DBMind.

positional arguments:
  {train,predict,finetune}
                        The training mode is to perform feature extraction and
                        model training based on historical SQL statements. The
                        prediction mode is to predict the execution time of a
                        new SQL statement through the trained model.

optional arguments:
  -h, --help            show this help message and exit
  -f CSV_FILE, --csv-file CSV_FILE
                        The data set for training or prediction. The file
                        format is CSV. If it is two columns, the format is
                        (SQL statement, duration time). If it is three
                        columns, the format is (timestamp of SQL statement
                        execution time, SQL statement, duration time).
  --predicted-file PREDICTED_FILE
                        The file path to save the predicted result.
  --model {template,dnn}
                        Choose the model model to use.
  --query QUERY         Input the queries to predict.
  --threshold THRESHOLD
                        Slow SQL threshold.
  --model-file MODEL_FILE, --model-path MODEL_FILE
                        The storage path of the model file, used to read or
                        save the model file.
  --config-file CONFIG_FILE, --config CONFIG_FILE
```

11.3.4.4 命令参考

表 11-38 命令行参数说明

参数	参数说明	取值范围
-f	训练或预测文件位置。	-
--predicted-file	预测结果存储位置。	-
--model	模型选择。	template, dnn
--model-path	训练模型存储位置。	-

11.3.4.5 常见问题处理

- 训练场景失败：请检查历史日志文件路径是否正确，且文件格式符合上文规定。
- 预测场景失败：请检查模型路径是否正确。确保待预测负载文件格式正确。

11.3.5 SQL Rewriter

11.3.5.1 概述

SQL Rewriter是一个SQL改写工具，根据预先设定的规则，将查询语句转换为更为高效或更为规范的形式，使得查询效率得以提升。

版权所有 © 华为云计算技术有限公司

说明

- 分布式不支持此功能。
- 本功能不适用包含子查询的语句。
- 本功能只支持SELECT语句和DELETE对整个表格删除的语句。
- 本功能包含12个改写规则，对不符合改写规则的语句，不会进行处理。
- 本功能会对原始查询语句和改写后语句进行屏幕输出，不建议对包含涉敏感信息的SQL语句进行改写。
- union转union all规则避免了去重，从而提升了查询性能，所得结果有可能存在冗余。
- 语句中如包含‘order by’+ 指定列名或‘group by’+ 指定列名，无法适用SelfJoin规则。
- 工具不保证查询语句等价转换，其目的是提升查询语句效率，一些推荐结果，需要结合业务实践进行优化，例如显性要求指定select的字段名。

11.3.5.2 使用指导

前提条件

数据库状态正常、连接正常。

使用方法示例

以tpcc数据库为例：

```
gs_dbmind component sql_rewriter 5030 tpcc queries.sql --db-host 127.0.0.1 --db-user myname --schema
public
```

queries.sql为需要改写的SQL，内容如下：

```
delete from bmsql_config;
delete from bmsql_config where cfg_name='1';
```

结果为多个改写后的查询语句，显示在屏幕（无法改写的语句，显示为空），如下：

```
+-------------------------------------------------------------------------+-----------------------------+
| Raw SQL                                                 | Rewritten SQL             |
+-------------------------------------------------------------------------+-----------------------------+
| delete from bmsql_config;                                  | TRUNCATE TABLE bmsql_config; |
| delete from bmsql_config where cfg_name='1';                  |                           |
+-------------------------------------------------------------------------+-----------------------------+
```

11.3.5.3 获取帮助

使用SQL Rewriter前，您可以通过以下指令获取帮助。

```
gs_dbmind component sql_rewriter --help
```

显示如下帮助信息：

```
usage:  [-h] [--db-host DB_HOST] [--db-user DB_USER] [--schema SCHEMA]
        [-v] db_port database file

SQL Rewriter

positional arguments:
  db_port           Port for database
  database           Name for database
  file              File containing SQL statements which need to rewrite

optional arguments:
  -h, --help         show this help message and exit
```


版权所有 © 华为云计算技术有限公司

```
--db-host DB_HOST  Host for database
--db-user DB_USER  Username for database log-in
--schema SCHEMA    Schema name for the current business data
-v, --version      show program's version number and exit
```

密码通过管道输入或交互式输入，对于免密用户，任意输入都可通过检验。

11.3.5.4 命令参考

表 11-39 命令行参数说明

参数名称	释义
db_port	数据库端口号。
database	数据库名称。
file	包含多个查询语句的文件路径。
db-host	（可选）数据库主机号。
db-user	（可选）数据库用户名。
schema	（可选，模式为public）模式。

11.3.5.5 常见问题处理

- SQL无法改写：请查看SQL是否符合改写规则或SQL语法是否正确。

11.3.6 Anomaly detection

11.3.6.1 概述

Anomaly detection 异常检测模块主要基于统计方法来实现时序数据来发现数据中存在的可能的异常情况。该模块框架解耦，可以实现不同异常检测算法的灵活替换，而且该模块功能可以根据时序数据的不同特征来自动选择算法，支持异常值检测、阈值检测、箱型图检测、梯度检测、增长率检测、波动率检测和状态转换检测。

当前DBMind默认启动的检测器如下：

分类	检测器	告警级别	默认检测逻辑	是否支持诊断	处理措施
磁盘	high_disk_usage_detector（磁盘高占用检测器）	WARNING	磁盘占用超过80%（阈值可配置）	是	**•场景5：磁盘空间占用高异常**
	high_db_disk_usage_detector（数据库磁盘高占用检测器）	WARNING	数据盘占用超过80%（阈值可配置）	是	**•场景5：磁盘空间占用高异常**

版权所有 © 华为云计算技术有限公司

分类	检测器	告警级别	默认检测逻辑	是否支持诊断	处理措施
	high_xlog_count_detector（Xlog堆积检测器）	WARNING	xlog数量超过checkpoint_segments * 2 + wal_keep_segments+1的个数大于0（可配置）	否	-
内存	high_mem_usage_detector（系统内存高占用检测器）	WARNING	系统内存使用率10分钟（可配置）的数据超过80%（可配置）的数据的采样率超过80%（可配置）	是	**•场景3：动态内存使用率异常**； **•场景4：共享内存使用率异常**
	mem_leak_detector（系统内存泄露检测器）	INFO	系统内存在7天-30天的时间范围内持续上涨，检测所需要的最短数据长度是7天，数据长度超过30天之后总是对最近30天的内存进行检测。	是	**•场景2：内存泄漏监测**
	session_mem_increase_detector（会话内存泄露检测器）	INFO	contextname粒度的会话内存2小时（可配置）的数据持续上涨，且该上下文占总会话内存比例超过5%（可配置）的采样率超过80%（可配置），且动态内存使用率超过80%（可配置）数据的采样率超过80%（可配置）	否	在异常信息中通过标签信息获取具体故障的节点和上下文信息，可进一步联系内核运维人员进一步定位 **说明** 对于多线程使用的内存上下文，会被汇总在某一个上下文之下，计算的是该类上下文的总和，不对单个线程进行分析。
	shared_mem_increase_detector（共享缓存泄露检测器）	INFO	contextname粒度的共享内存2小时（可配置）的数据持续上涨，且该上下文占总共享内存比例超过5%（可配置）的采样率超过80%（可配置），且动态内存使用率超过80%（可配置）数据的采样率超过80%（可配置）	否	在异常信息中通过标签信息获取具体故障的节点和上下文信息，可进一步联系内核运维人员进一步定位 **说明** 对于多线程使用的内存上下文，会被汇总在某一个上下文之下，计算的是该类上下文的总和，不对单个线程进行分析。

版权所有 © 华为云计算技术有限公司

分类	检测器	告警级别	默认检测逻辑	是否支持诊断	处理措施
	other_mem_increase_detector（三方组件内存泄露检测器）	INFO	三方组件内存在30分钟（可配置）的数据持续上涨，且系统内存使用率10分钟（可配置）的数据超过80%（可配置）的数据的采样率超过80%（可配置）	否	在异常信息中通过标签信息获取具体故障信息，可进一步联系内核运维人员进一步定位
CPU	high_cpu_usage_detector（CPU高使用率检测器）	WARNING	CPU使用率最近10分钟（可配置）的数据超过80%（可配置）的数据的采样率超过80%（可配置）	是	**•场景1：用户CPU使用率异常**
	high_thread_pool_rate_detector（线程池高使用率检测器）	WARNING	线程池使用率超过90%（可配置）	是	**•场景2：线程池使用率异常**
磁盘IO	high_io_delay_detector（I/O延迟检测器）	INFO	os_disk_await最近10分钟（可配置）的数据中超过30毫秒（可配置）的数据的采样率超过50%（可配置）且超过90毫秒（可配置）的数据的采样率超过10%（可配置）	是	**•场景6：磁盘IO读取时延异常**
	slow_disk_detector（慢盘监测器）	WARNING	os_disk_await数据中25%（可配置）的数据其await time超过15毫秒（可配置）且os_disk_await在7天-30天的数据持续上涨，检测所需要的最短数据长度是7天，数据长度超过30天之后总是对最近30天的内存进行检测。	是	**•场景1：潜在慢盘监测**
	disk_io_jam_detector（I/O阻塞检测器）	INFO	os_disk_io_queue_length大于0（可配置）且os_disk_ioutils大于99%（可配置）且os_disk_await数据中超过30毫秒（可配置）的数据的采样率超过50%（可配置）	否	在异常信息中通过标签信息获取具体故障的节点和硬盘信息，可进一步联系内核运维人员进一步定位

版权所有 © 华为云计算技术有限公司

分类	检测器	告警级别	默认检测逻辑	是否支持诊断	处理措施
网络	lag_detector（来自或去往主DN的网络延迟检测）	INFO	gaussdb_ping_lag最近10分钟（可配置）的数据超过50毫秒（可配置）的数据的采样率超过50%（可配置）	否	检查是否发生主备切换引起的网络条件变化，如果不是请联系现场运维人员进一步排查
	packet_loss_detector（来自或去往主DN的网络丢包率检测）	INFO	gaussdb_ping_packet_rate最近10分钟（可配置）的数据得包率低于90%（可配置）的数据的采样率超过20%（可配置）	否	检查是否发生主备切换引起的网络条件变化，如果不是请联系现场运维人员进一步排查
文件句柄	leaked_fd_detector（文件句柄泄露检测器）	WARNING	数据库进程未释放的文件句柄数gaussdb_process_leaked_fds超过5个（可配置）	否	如需进一步分析文件句柄泄露异常，可以根据标签中的pid，在数据库节点上使用以下命令进行分析 ls -l /proc/{pid}/fd | grep '(deleted)' 可以获取到泄露的文件路径，联系运维人员进一步排查
长事务	slow_sql_detector（长事务检测器）	INFO	存在处于active或者idle in transaction状态且运行时间超过1个小时（3600秒，可配置）的事务 **说明** 由于目标指标pg_long_transaction_count只采集时长超过3600秒的事务，所以当阈值即使设置低于3600秒时，可以检测的长事务仍然是超过3600秒以上的长事务。	否	-

版权所有 © 华为云计算技术有限公司

须知

- 异常检测器的落盘存储依赖于元数据库，请勿在元数据库中对异常检测器进行手动修改。
- 当前版本仅支持，在主备切换、扩容和剔除节点的场景下，同一集群的检测器配置参数会被继承和保留，其他场景均不支持。
- 长事务检测器由长事务整体触发异常，但是计算异常个数的时候实际会计数长事务中超时之后执行的每个SQL。
- 对于网络异常检测器，当延迟超过1000ms时，网络延迟相关指标的采集会开始出现数据丢失现象，无法保证网络数据的完整性，可能会对网络检测器的检测结果产生影响，此时应通过集群异常诊断的断网检测功能上报异常。
- 当前的异常检测器有部分检测项和智能巡检功能的某些检测项比较相似，如：CPU使用率，磁盘使用率，内存使用率，磁盘IO使用率和线程池使用率检测等。由于智能巡检的设计目的和时间跨度与异常检测在设计上有所不同，所以检测阈值和条件也有所区别，所以可能出现相似检测项出现不一致的结果，这些属于正常现象。
- 对于会话内存上下文指标和共享内存上下文指标pg_session_memory_detail_rate和pg_shared_memory_detail_rate，其超时时长为10秒，在查询内存视图耗时很长的情况下，指标所标注的时间会相应滞后。
- 在505.2版本下不支持亚健康功能的xlog堆积和长事务场景。

11.3.6.2 使用指导

假设指标采集系统运行正常，并且用户已经初始化了配置文件目录confpath，则可以通过下述命令实现本特性的功能：

仅启动异常检测功能：

```
gs_dbmind service start --conf confpath --only-run anomaly_detection
```

对于某一指标，在全部节点上，从timestamps1到timestamps2时间段内的数据进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2
```

对于某一指标，在特定节点上，从timestamps1到timestamps2时间段内的数据进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address
```

对于某一指标，在全部节点上，从timestamps1到timestamps2时间段内的数据，以特定异常检测方式进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2 --anomaly anomaly_type
```

对于某一指标，在特定节点，从timestamps1到timestamps2时间段内的数据，以特定异常检测方式进行概览：

```
gs_dbmind component anomaly_detection --conf confpath --action overview --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address --anomaly anomaly_type
```

对于某一指标，在特定节点，从timestamps1到timestamps2时间段内的数据，以特定异常检测方式进行可视化展示：

```
gs_dbmind component anomaly_detection --conf confpath --action plot --metric metric_name --start-time timestamps1 --end-time timestamps2 --host ip_address --anomaly anomaly_type
```

运行异常诊断后台任务：

```
参考10.3.3.3中方法，其对应定时任务为： anomaly_detection
```


版权所有 © 华为云计算技术有限公司

停止已启动的服务：

```
gs_dbmind service stop --conf confpath
```

指标异常分析功能

指标异常根因分析接口调用：

```
curl -X 'GET' http://127.0.0.1:8080/v1/api/app/metric-diagnosis/?
metric_name=os_cpu_user_usage&metric_filter={"from_instance":"127.0.0.1","from_job":"node_exporter","ins
tance":"127.0.0.1:8181","job":"reprocessing_exporter"}&alarm_cause=["high_cpu_usage"]&start=1691482728
000&end=1691482728000 -H 'accept: application/json' -H 'Content-Type: application/json' -H
"Authorization: bearer xxx"
```

如果使用HTTPS协议，则查询示例为：

```
curl -X 'GET' 'https://127.0.0.1:8080/v1/api/app/metric-diagnosis/?
metric_name=os_cpu_user_usage&metric_filter={"from_instance":"127.0.0.1","from_job":"node_exporter","ins
tance":"127.0.0.1:8181","job":"reprocessing_exporter"}&alarm_cause=["high_cpu_usage"]&start=1691482728
000&end=1691482728000' -H 'accept: application/json' -H 'Content-Type: application/json' -H
"Authorization: bearer xxx" --cacert xx.crt --key xx.key --cert xx.crt
```

返回结果格式参考：

```
{"data":{[{'reason1': 0.0, 'reason2': 1.0}, 'conclusion', 'advice']},"success":true}
```

停止已启动的服务：

```
gs_dbmind service stop --conf confpath
```

指标异常分析支持的场景详细情况如下：

- **场景1：用户CPU使用率异常**

  异常判断条件：用户CPU使用率10分钟内持续高于80%。

  可能的异常根因：

  - 业务压力增大导致

    现象：TPS、网络读写、CPU和内存基本存在一定程度的上涨。

    分析：通过与相关指标进行相关性比对。

    建议：根据业务量评估CPU、内存等资源是否满足业务需求，是否需要扩容。
  - iowait延时高导致

    现象：数据库磁盘的读时延和写时延变长。

    建议：增加IO吞吐量，排查可以降低IO的进程。

  分析时提供的信息：

  - 提供pg_stat_activity每个unique_sql_id的总运行时间的快照信息。
- **场景2：线程池使用率异常**

  异常判断条件：默认规则线程池使用率10分钟内持续高于80%

  可能的异常根因：

  - 业务压力增大导致

    现象：TPS、网络读写、内存使用率和线程池使用率基本存在一定程度的关联。

    分析：通过与相关指标进行相关性比对。

    建议：根据业务量评估CPU、内存等资源是否满足业务需求，是否需要扩容。
  - 磁盘读写时延过高导致

    现象：数据库磁盘读写时延增高，导致线程池使用率超过配置的阈值。

版权所有 © 华为云计算技术有限公司

分析：查看产生报警的节点的数据盘每次读写时间的相关性。

建议：若发现数据库磁盘读写时延频繁过高或者有明显劣化趋势，则继续定位是否磁盘硬件故障。

- 工作负载上升

  现象：QPS、CPU使用率和系统内存持续上涨。

  分析：根据算法对QPS、CPU使用率和系统内存持续上涨进行判断。

  建议：数据库负载上升，考虑采用限流措施。

- **场景3：动态内存使用率异常**

  异常判断标准：系统内存超过阈值（默认10分钟连续超过80%），再进行动态内存使用率异常分析

  可能的异常根因：

  - 会话数上涨导致

    现象：在线会话数量指标同时上涨。

    分析：查看同一时间段内会话数量和内存上涨之间的关系，通过皮尔逊计算相关系数，如果绝对值超过阈值的指标会被认为是相关异常。

    建议：停止变更。

  - 动态内存泄露导致

    现象：动态内存持续上涨。

    分析：查看内存占用较大的上下文数量，如未发生很大变化则可能是内存泄露。

    建议：通过pg_terminate_session终止会话或重启DN进程。

  - 非数据库内存泄露导致

    现象：非数据库内存持续上涨。

    分析：查看非数据库内存占用。

    建议：分析系统内存占用，终止节点上其他占用内存较高的进程。

  分析时提供的信息：

  - 提供session_memory_detail的快照信息。

- **场景4：共享内存使用率异常**

  异常判断标准：系统内存超过阈值（默认10分钟连续超过80%），再进行共享内存使用率异常分析。

  可能的异常根因：

  - 未落盘脏页数过高导致

    现象：INSERT或UPDATE操作比例突然增大。

    分析：分析INSERT或UPDATE操作比例突然增大与共享内存的相关性，通过皮尔逊计算相关系数，如果绝对值超过阈值的指标会被认为是相关异常。

    建议：考虑降低pagewriter_sleep参数，加速脏页落盘的速度；考虑降低dirty_page_percent_max参数，降低刷页阈值上限。

  - 共享内存泄露导致

    现象：共享内存持续上涨。

    分析：查看系统内存占用，确认是否有除了gaussdb进程外占用大量内存的进程。

    建议：手动清理，ipcrm -m shmid（此命令操作危险，请谨慎操作）。

  分析时提供的信息：

 版权所有 © 华为云计算技术有限公司

- 提供shared_memory_detail的快照信息

- **场景5：磁盘空间占用高异常**

  异常判断标准：磁盘空间占用超过阈值（默认80%）。

  可能的异常根因：

  - 数据库表空间膨胀导致

    现象：数据库磁盘占用快速上升。

    分析：分析INSERT或UPDATE操作比例和磁盘IO读写来确定是否脏数据增加过快。

    建议：临时情况，无需处理。

  - xlog堆积导致

    现象：xlog路径占用空间过大。

    分析：分析xlog数量是否超过wal_keep_segments + checkpoint_segments * 2。

    建议：查看是否有未推进的逻辑复制槽阻塞xlog回收。

  分析时提供的信息：

  - 提供实时表空间信息；
  - 提供临时文件信息，包括线程和会话信息。

- **场景6：磁盘IO读取时延异常**

  异常判断标准：数据库磁盘IO使用率超过阈值（默认99%）

  可能的异常根因：

  - 数据磁盘读写IO使用率超阈值导致

    现象：数据库磁盘读写IO使用率接近100%

    分析：分析数据库磁盘读写IO使用率和时延之间的关系

    建议：降低IO压力，提高磁盘的IO限制

- **场景7：扫描攻击**

  异常判断标准：SQL执行错误率和用户越权率加权得分超过阈值（默认阈值：提示0.2，告警0.6，严重0.8）

  可能的异常根因：

  - SQL执行错误率和用户越权率增高导致

    现象：SQL执行错误率和用户越权率增高

    分析：用户使用自动化工具扫描目标网络或系统的漏洞，利用这些漏洞获取未经授权的访问权限，窃取敏感数据或破坏系统目标

    建议：及时更新数据库软件和安全补丁，以修复已知漏洞，减少攻击面

- **场景8：暴力登录**

  异常判断标准：用户无效登录率和用户锁定率指标加权得分超过阈值（默认阈值：提示0.1，告警0.3，严重0.4）

  可能的异常根因：

  - 用户无效登录率和用户锁定率增高导致

    现象：用户无效登录率和用户锁定率增高

    分析：攻击者猜测用户名和密码进行暴力登录，导致账户锁定及其他拒绝服务问题

    建议：根据告警信息，及时检查登录日志、采取相应措施

 版权所有 © 华为云计算技术有限公司

- **场景9：违规操作**
- 异常判断标准：用户越权率指标超过阈值（默认阈值：提示0.2，告警0.6，严重0.8）。

  可能的异常根因：

  - 用户越权率增高导致

    现象：用户越权率增高。

    分析：攻击者使用用户凭证进行违规操作。

    建议：对于敏感数据，限制访问权限。

说明

其中，场景7~9的约束如下：

- 用户需有Monitor admin和Audit admin权限，如果没有Audit admin权限，会导致审计指标数据全为0，导致诊断结果不可用。
- 需要开启audit_enabled、audit_log_logout、audit_user_locked和audit_user_violation参数。
- 审计总开关GUC参数audit_enabled支持动态加载。在数据库运行期间修改该配置项的值会立即生效，无需重启数据库。默认值为on，表示开启审计功能。
- 审计项audit_login_logout：默认值为7，表示开启用户登录、退出的审计功能。设置为0表示关闭用户登录、退出的审计功能。
- 审计项audit_user_locked：默认值为1，表示开启审计用户锁定和解锁功能。
- 审计项audit_user_violation：默认值为0，表示关闭用户越权操作审计功能。通过命令gs_guc reload -Z datanode -N all -I all -c "audit_user_violation=1"开启
- 如无开启审计相关参数，只能处理扫描攻击场景。

亚健康诊断功能

亚健康状态是系统介于健康状态和故障状态之间的一种状态，系统仍在运行且功能正常但处于降级模式的一种情况，它的存在会造成系统性能严重低于预期。

亚健康诊断支持的场景如下：

- **场景1：潜在慢盘监测**

  DBMind默认初始化"slow_disk_detector"检测器，在每一次触发异常检测定时任务时对潜在慢盘进行监测。

  - 现象：“慢盘”现象普遍存在于存储架构之中，由于硬盘体质或者频繁读写的原因，部分硬盘会出现性能故障，I/O负载过高等情况进而导致延时变大，读写变慢的现象。
  - 检测逻辑：在最近的过去7天~30天（收集的数据小于7天不进行检测），其磁盘I/O平均读写时间长期在30ms以上并呈现出上升趋势，则认为其发生潜在慢盘。

- **场景2：内存泄漏监测**

  DBMind默认初始化"mem_leak_detector"检测器，在每一次触发异常检测定时任务时对内存泄漏进行监测。

  - 现象：程序中已动态分配的堆内存由于某种原因程序未释放或无法释放，造成系统内存的浪费，导致程序运行速度减慢甚至系统崩溃等严重后果。内存泄漏缺陷具有隐蔽性、积累性的特征，比其他内存非法访问错误更难检测。
  - 检测逻辑：最近的过去7天~30天（收集的数据小于7天不进行检测），其内存占用呈现出上升趋势，则认为其发生内存泄漏。

版权所有 © 华为云计算技术有限公司

可能的异常根因：

- 动态内存泄露导致

  现象：动态内存持续上涨。

  分析：查看内存占用较大的上下文数量，如未发生很大变化则可能是内存泄露。

  建议：通过pg_terminate_session终止会话或重启DN进程。
- 共享内存泄露导致

  现象：共享内存持续上涨。

  分析：查看系统内存占用，确认是否有除了gaussdb进程外占用大量内存的进程。

  建议：手动清理，ipcrm -m shmid（此命令操作危险，请谨慎操作）。
- 第三方软件内存增高导致

  现象：other_used_memory持续上涨。

  分析：第三方软件内存泄露。

  建议：排查第三方软件的内存占用。
- 非数据库内存泄露导致

  现象：非数据库内存持续上涨。

  分析：查看非数据库内存占用。

  建议：分析系统内存占用，终止节点上其他占用内存较高的进程。
- 用户无效登录过高导致

  现象：用户无效登录数超阈值。

  分析：用户无效登录日志过多，存在大量连接失败。

  建议：请联系DBA。

分析时提供的信息：

- 提供session_memory_detail的快照信息。
- 提供shared_memory_detail的快照信息。

- **场景3：锁冲突监测**

  DBMind默认初始化"deadlock_detector"检测器，在每一次触发异常检测定时任务时对锁冲突进行监测。
  - 现象：当发生所冲突时，日志中会记录锁冲突的详细信息。
  - 检测逻辑：当内核日志记录到死锁日志时，则认为其发生锁冲突，并对死锁信息进行收集。

 版权所有 © 华为云计算技术有限公司

说明

- 在输入anomaly detection的参数时，start-time至少要比end-time早30秒以上。
- 异常检测功能依赖于异常检测器，可以通过异常检测器的查询接口/v1/api/app/anomaly-detection/detectors/{name}查看当前已添加的全部异常检测器。
- 根因分析的某些功能依赖opengauss-exporter的指标采集，当数据库处于高负载状况下，由于opengauss-exporter设置了SQL的超时机制来保护业务，可能会导致某些复杂的查询语句超时，进而导致采集的数据为空，当发生采集失败时，可以查询opengauss-exporter的日志来进行进一步的定位。
- 添加检测器或更改检测器参数会将检测器状态变为启用。
- 对于初始化时默认的长期指标检测器（如slow_disk_detector和mem_leak_detector），其检测器的监测时间窗口长度是固定的，不支持修改，对于其duration参数的修改是无效的。
- 对于长期指标检测器，当收集到的数据低于一个星期时，不会进行检测。当数据已经长达一小时没有更新时，不会进行检测。

警告

- 异常检测器的落盘存储依赖于元数据库，请勿在元数据库中对异常检测器进行手动修改。
- 当前版本仅支持，在主备切换、扩容和剔除节点的场景下，同一集群的检测器配置参数会被继承与保留，其他场景均不支持。

11.3.6.3 获取帮助

模块命令行说明：

```
gs_dbmind component anomaly_detection --help
```

显示如下帮助信息：

```
usage: [-h] --action {overview,plot} -c CONF -m METRIC -s START_TIME -e END_TIME [-H HOST] [-a
{level_shift,spike,seasonal,volatility_shift}]

Workload Anomaly detection: Anomaly detection of monitored metric.

optional arguments:
  -h, --help            show this help message and exit
  --action {overview,plot}
                        choose a functionality to perform
  -c CONF, --conf CONF  set the directory of configuration files
  -m METRIC, --metric METRIC
                        set the metric name you want to retrieve
  -s START_TIME, --start-time START_TIME
                        set the start time of for retrieving in ms, supporting UNIX-timestamp with microsecond or
datetime format
  -e END_TIME, --end-time END_TIME
                        set the end time of for retrieving in ms, supporting UNIX-timestamp with microsecond or
datetime format
  -H HOST, --host HOST  set a host of the metric, ip only or ip and port.
  -a {level_shift,spike,seasonal,volatility_shift}, --anomaly {level_shift,spike,seasonal,volatility_shift}
                        set a anomaly detector of the metric from: "level_shift", "spike", "seasonal", "volatility_shift"
```


版权所有 © 华为云计算技术有限公司

11.3.6.4 命令参考

表 11-40 命令行参数说明

参数	参数说明	取值范围
-h, --help	帮助命令。	-
--action	动作参数。	● overview：概览。 ● plot：可视化。
-c，--conf	配置文件目录。	-
-m，--metric	指定显示指标名。	任意能采集到的指标。例如os_mem_usage、os_disk_usage等。
-H, --host	指定数据来源地址信息，通过地址信息进行过滤。	-ip地址或者ip地址加端口号。
-a, --anomaly	指定异常检测方式，用于过滤。	level_shift、spike、seasonal、volatility_shift。
-s, --start-time	显示开始时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S。	正整数或日期时间格式。
-e, --end-time	显示结束时间的时间戳，单位毫秒；或日期时间格式为 %Y-%m-%d %H:%M:%S。	正整数或日期时间格式。

11.3.6.5 常见问题处理

- 概览场景失败：请检查配置文件路径是否正确，且配置文件信息是否完整。检查指标名称是否准确，检查host地址是否正确，检查异常检测类型是否准确，检查起止时间内指标是否存在对应数据。
- 可视化场景失败：请检查配置文件路径是否正确，且配置文件信息是否完整。检查指标名称是否准确，检查host地址是否正确，检查异常检测类型是否准确，检查起止时间内指标是否存在对应数据。

11.3.7 Distributed-key-advisor

对于分布式数据库来说，一个有效的分布键是十分重要的。可以通过使用本功能进行数据库分布键的自动化推荐。

分布键推荐功能提供数据库外部工具和SQL内置接口两种方式，支持数据迁移前与数据迁移后两种场景的分布键推荐。

11.3.7.1 外部工具

版权所有 © 华为云计算技术有限公司

11.3.7.1.1 概述

本分布键推荐功能在数据库内核外部实现，以独立的工具形式呈现，支持数据迁移前与数据迁移后两种场景下的分布键推荐，用户只需提供业务数据与少量相关数据库统计信息。

11.3.7.1.2 使用指导

前提条件

- 数据库运行状态良好，无资源紧张状况。
- 该特性是以外部工具的形式存在，需要用户在使用工具时提供元数据。
- 该特性当前仅针对Hash分布做推荐。
- 本特性暂不考虑计算倾斜、访问倾斜。
- 采用WDR报告中获取业务数据需提前设置guc参数track_stmt_stat_level='L0,L0'，并且要求执行节点是CN，以及指定用户具有monitor admin权限。
- 配置python3.7+环境。

操作步骤

步骤1 准备好包含有多条DML语句的文件作为输入的workload，对于数据迁移前场景，同时准备数据库相关统计信息的json文件。

数据库相关统计信息的JSON文件数据格式为：

```
{ table_name: # 表名
   {"size": size_value, # 表大小(整型)
    "rows": rows_value, # 表行数（整型）
    "primaryKey": "column_value", # 主键第一列（字符串）
    "column":           # 表列名及该列对应的distinct value值(浮点型)
       {"col1": col1_distinct_value,
        "col2": col2_distinct_value,
        "col3": col3_distinct_value
        }
    }
}
```

workload数据格式为：

```
SQL语句文本
```

步骤2 运行命令如下：

```
gs_dbmind component dkr [-f FILE] [-m MODE] [-s STATISTICS] [-p PORT] [-d DATABASE] [--schema SCHEMA] [--host HOST] [-U USER] [--dn DN] [--start_time START_TIME] [--end_time END_TIME] [--min_distinct_threshold MIN_DISTINCT_THRESHOLD] [--cost_type COST_TYPE][--max_replication_table_size MAX_REPLICATION_RTABLE_SIZE] [--prior_distribute_transaction] [--standard][--driver]
```

----结束

使用方法示例

当需要将业务数据从其他类型的数据库迁移到GaussDB分布式数据库时（即“数据迁移前”场景），则可以使用下述命令启动推荐：

```
gs_dbmind component dkr -f test.sql -m offline -s table_info.json --dn 6 --min_distinct_threshold 0.5 --cost_type naive
```

当业务数据已经迁移到GaussDB分布式数据库中后，对于需要对分布键进行调优的场景（即“数据迁移后”场景），则可使用下述命令启动推荐：

版权所有 © 华为云计算技术有限公司

```
echo huawei.123 | gs_dbmind component dkr -f test.sql -m online -p 53400 -d tpch500high --host linux173 -
U dba --dn 6  --min_distinct_threshold 0.5 --cost_type optimizer
```

运行结果示例如下，其中“CREATE TABLE”表示应该创建分布键的列名，“HINT”提示可以创建为复制表的表名：

```
################################## Distribution key recommended result
##################################
CREATE TABLE customer(...) DISTRIBUTE BY HASH(c_custkey);
CREATE TABLE lineitem(...) DISTRIBUTE BY HASH(l_orderkey);
CREATE TABLE orders(...) DISTRIBUTE BY HASH(o_orderkey);
CREATE TABLE part(...) DISTRIBUTE BY HASH(p_partkey);
CREATE TABLE partsupp(...) DISTRIBUTE BY HASH(ps_partkey);
CREATE TABLE region(...) DISTRIBUTE BY HASH(r_regionkey);
CREATE TABLE supplier(...) DISTRIBUTE BY HASH(s_suppkey);
HINT: table nation can be set as a replication table. (table rows = 25)
```

11.3.7.1.3 获取帮助

启动调优程序之前，可以通过如下命令获取帮助信息：

```
gs_dbmind component dkr --help
```

输出帮助信息结果如下：

```
usage:              [-h] [-f FILE] [--standard] [-s STATISTICS]
                    -m {offline,online} [--dn DN] [-p PORT] [-d DATABASE]
                    [--host HOST] [--schema SCHEMA] [-U USER]
                    [--start_time START_TIME] [--end_time END_TIME]
                    [--min_distinct_threshold MIN_DISTINCT_THRESHOLD]
                    [--cost_type {naive,optimizer}]
                    [--max_replication_table_size MAX_REPLICATION_TABLE_SIZE]
                    [--prior_distribute_transaction]
                    [--driver]

Generate a set of distributed keys for workload. two modes are supported:offline: (--file -m
-s) online: (--file -p -d --host -U --start_time --end_time --schema)

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  File contains workload queries (One query per line) or
                        oracle stored procedure
  --standard            queries in the file as standard format
  -s STATISTICS, --statistics STATISTICS
                        File contains statistical information
  -m {offline,online}, --mode {offline,online}
                        the current mode
  --dn DN               The number of data nodes
  -p PORT, --port PORT  Port of database
  -d DATABASE, --database DATABASE
                        Name of database
  --host HOST           Host for database
  --schema SCHEMA       Schema name for the current business data
  -U USER, --user USER  Username for database log-in
  --start_time START_TIME
                        Collect WDR report start time
  --end_time END_TIME   Collect WDR report end time
  --min_distinct_threshold MIN_DISTINCT_THRESHOLD
                        Minimum value of distinct value
  --cost_type {naive,optimizer}
                        Which cost estimation algorithm to use
  --max_replication_table_size MAX_REPLICATION_TABLE_SIZE
                        Maximum number of rows in the replication table
  --prior_distribute_transaction
                        Prioritize the processing of distributed transaction
  --driver              Whether to employ python-driver
```


版权所有 © 华为云计算技术有限公司

11.3.7.1.4 常见问题处理

- 提示No recommended results for distribution key and replication table。检查数据库参数或schema参数是否输入正确。
- 获取WDR报告为空，检查时间参数格式是否正确，或者确认指定用户是否有monitor admin权限。

11.3.7.1.5 命令参考

表 11-41 命令行参数

参数	参数说明	取值范围
MODE	指定当前业务场景。	offline,online
STATISTICS	数据迁移前场景下，数据库统计信息文件路径；和UGO对接需要。	-
FILE	包含workload语句的文件路径。	-
DATABASE	连接数据库的名字。	-
SCHEMA	（可选）模式名称。	-
PORT	连接数据库的端口号。	-
DN	DN节点个数。	-
HOST	（可选）连接数据库主机号。	-
USER	（可选）连接数据库用户名。	-
START_TIME	（可选）收集WDR报告业务数据开始时间。	timestamp with time zone
END_TIME	（可选）收集WDR报告业务数据结束时间。	timestamp with time zone
MIN_DISTINCT_THRESHOLD	（可选）列的distinct value值最低，默认0.5。	-
COST_TYPE	（可选）代价计算类型，默认naive。	naive,optimizer
standard	（可选）指定当前业务文件格式与索引推荐章节介绍的extract_log脚本输出文件格式一致。	-
max_replication_table_size	（可选）指定复制表最大行数。	-
prior_distribute_transaction	（可选）指定对整个业务优先考虑分布式事物场景。	-
driver	（可选）指定使用python驱动连接数据库。	-

版权所有 © 华为云计算技术有限公司

11.3.7.2 SQL 内置接口

11.3.7.2.1 概述

分布键推荐SQL内置接口针对的是在分布式数据库下分布列以及分布方式的推荐，目的是在进行业务迁移或业务上线时，减少选择表分布列的人力成本。

11.3.7.2.2 使用指导

功能介绍

分布键推荐主要分为三种模式：

- 启发式模式：基于专家规则、预置权重的方式进行推荐。
- WhatIf代价模式：依赖于优化器的代价进行推荐。
- 半在线模式：在线收集普通SQL、存储过程中的DML、DQL语句，离线调用启发式和WhatIf代价模式进行分析。

应用场景

适用于有数据、无数据两种情况。无数据仅支持Hash分布推荐，有数据支持Hash分布、复制表推荐。

业务场景有以下几类：

1. 业务无数据，但是建立好了表模型，以及设计好了SQL负载，在此种情况下用户明确表模型中的维度表、事实表，所以在此种情况下复制表推荐的原则为用户指定维度表，然后对事实表采用启发式规则推荐。
2. 业务有数据，但无意向导入数据，则按照1进行。
3. 业务有数据，且有意向使用实际数据进行推荐。采用接近实际业务的数据量和SQL负载进行测试，有益于推荐更加准确的分布列，否则可能会出现分布列和分布方式推荐不准确的问题。
4. 在业务已有数据的测试环境上，不想导入SQL负载，可以通过使用跑业务的方式收集负载，并离线分析。

使用限制

- 使用推荐函数前需要创建对应的表，分析不支持临时表。
- 如果使用WhatIf代价模式，必须analyze。
- 需要关闭负载均衡，不关闭会导致收集部分负载。
- 推荐过程中不能做表结构的变更。
- 半在线模式只能有系统管理员调用。
- 半在线模式只会收集业务负载DML、DQL语句。
- 同一时间只能有一个数据库的负载收集功能打开。

11.3.7.2.3 命令参考

分布键推荐SQL内置接口主要使用以下系统函数，具体描述请参见《开发指南》中“SQL参考 > 函数和操作符 > 分布列推荐函数”章节。

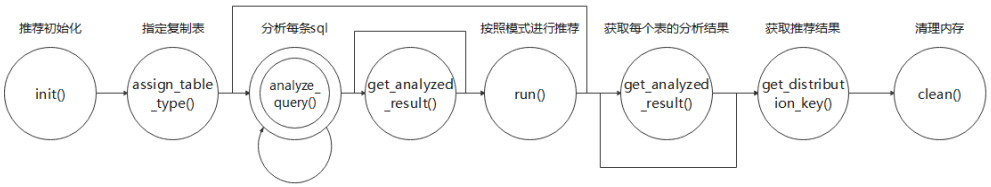
版权所有 © 华为云计算技术有限公司

表 11-42 内置系统函数

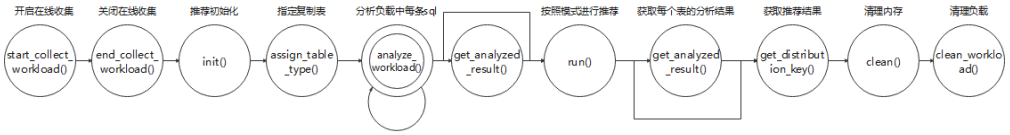
函数名	函数说明
sqladvisor.init	初始化分布键推荐参数。
sqladvisor.set_weight_params	设置启发式规则不同成分的权重。
sqladvisor.set_cost_params	使用Whtif代价模式可以设置的参数。
sqladvisor.analyze_query	导入需要推荐的SQL语句，并对语句的成分进行分析。
sqladvisor.analyze_workload	分析在线收集的负载信息。
sqladvisor.get_analyzed_result	获取当前表提取出来的有益成分。
sqladvisor.run	根据指定的模式和输入的SQL进行计算分析。
sqladvisor.clean	清理session中推荐过程中的全部内存。
sqladvisor.start_collect_workload	开启在线收集负载.
sqladvisor.end_collect_workload	关闭在线收集负载。
sqladvisor.clean_workload	清理负载中的内存。

使用建议

- 启发式、WhatIf代价推荐模式调用状态机。



- 半在线推荐模式调用状态机。



11.3.8 Cluster Diagnosis

11.3.8.1 概述

在现网业务中需要对发生的故障原因进行快速定位定界，本功能可以通过收集数据库集群中各个组件（如CMS、DN）等的信息和即时状态（如网络连通性），来判断集群环境是否存在故障，以及故障根因。可用于实现集群级别的故障根因诊断。

版权所有 © 华为云计算技术有限公司

DBMind对cmd-exporter进行加强，本版本支持CN、DN、CMS、CMA、ffic、OM_Monitor等日志采集，同时也支持基于节点间网络连通（如ping）状态采集。同时DBMind对现网故障场景进行了梳理，并对数据集进行枚举扩充，最终实现DN、CN故障快速定位。

说明

由于该功能是根据日志来进行诊断的，所以诊断结果中的时间可能因为日志的延迟或者日志的延迟处理，导致诊断结果中的时间晚于故障发生的时间。

表 11-43 现支持诊断的 CN、DN 故障根因列表

CN故障根因	DN故障根因
未知原因/Unknown。	未知原因/Unknown。
CN心跳超时/CN heartbeat timeout。	实例被停止/DN manual stop。
CN进程僵死/CN phony dead。	磁盘故障/DN disk Damage。
CORE/Core。	网卡故障/DN NIC down。
CN IP丢失/CN ip lost。	端口冲突/DN port conflict。
磁盘损坏或磁盘满/CN disk Damage。	CM Server仲裁重启DN/DN restarted by cms。
端口冲突/CN port conflict。	进程僵死重启/DN phony dead。
网卡故障/CN NIC down。	CORE/Core。
手动停止/CN manual stop。	只读/DN read only。
主机断网或宕机/CN down/ disconnection。	主机断网或宕机/DN down/disconnection。
CN与所有主DN断连/CN disconnected from dn。	主备DN间网络异常/DN Primary disconnected with Standby。
只读/CN read only。	DN IP丢失/DN ip lost。

说明

当cm_ctl query的集群状态输出结果异常时，一般是发生了调用栈输出，这种情况下难以获取集群状态，无法获取集群的诊断结果，相关状态标记为"abnormal_output_from_cm_ctl_query"，诊断结果为Unknown。

当dn节点处于"Offline"状态时，不对其进行集群故障诊断，返回状态为Normal，状态码-1。

11.3.8.2 使用指导

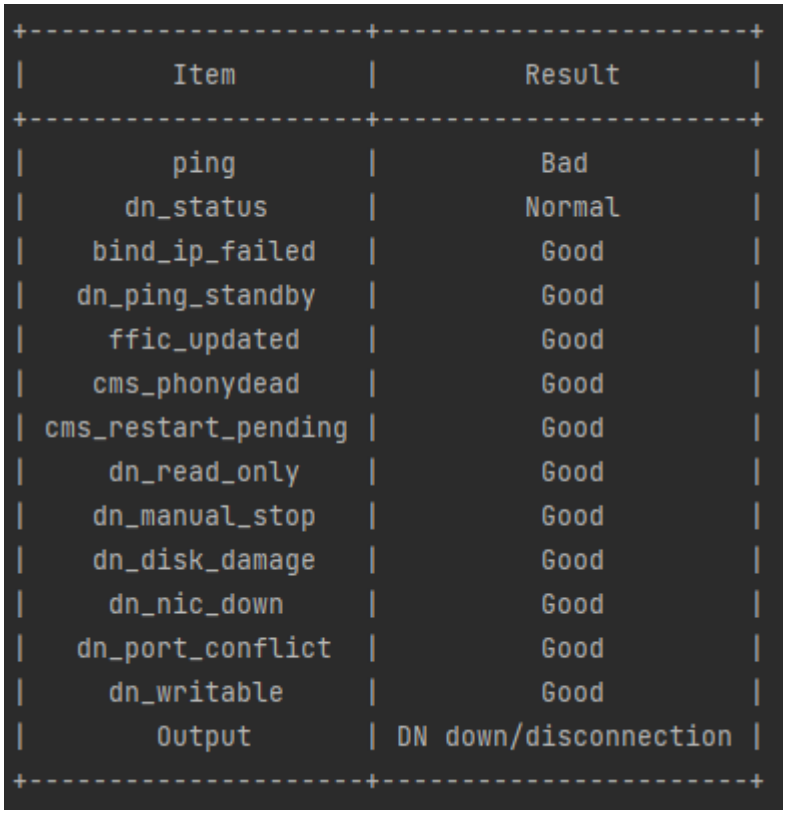
在DN集群产生异常告警时，一个完整的，用于启动集群故障分析功能的命令是：

```
gs_dbmind component cluster_diagnosis --conf {confpath} --host {ip_address} --role dn --time "2023-04-20 16:00:00" --method tree
```


版权所有 © 华为云计算技术有限公司

输入此命令后，系统读取所设定时间前3分钟的日志记录，并对选定的DN集群使用选定方法进行分析，分析的结果示例如图11-4所示：

图 11-4 DN 集群使用选定方法分析



Item	Result
ping	Bad
dn_status	Normal
bind_ip_failed	Good
dn_ping_standby	Good
ffic_updated	Good
cms_phonydead	Good
cms_restart_pending	Good
dn_read_only	Good
dn_manual_stop	Good
dn_disk_damage	Good
dn_nic_down	Good
dn_port_conflict	Good
dn_writable	Good
Output	DN down/disconnection

返回结果前半部分的字典给出对日志的解析结果，其中Good表示该项正常，Bad表示该项有异常；最后的Output表示输出结果，详情见表11-43。

说明

- 单次诊断读取的是诊断时间点之前三分钟的日志和节点状态，由于网络延迟，模型计算用时等因素，实际时间会略短于3分钟，综合各种因素，以150秒内的诊断结果更为准确。
- 集群故障诊断功能的网络连通性诊断是通过各个数据库节点之间的连通性来判断的，对于单节点集群，不存在数据库节点之间的连通性，所以集群诊断不支持单节点集群的故障诊断。
- 集群诊断功能中，判断网络连通性的超时时长为1秒，当网络延迟达到1秒及以上时，节点会被判断为断开连接。
- 尝试对于刚刚纳管的集群进行集群诊断时，由于采集数据存在延迟，可能会出现短暂的网络异常。

11.3.8.3 获取帮助

模块命令行说明：

```
gs_dbmind component cluster_diagnosis --help
```

显示如下帮助信息：

```
usage: [-h] --conf CONF --host HOST --role {cn,dn} [--time TIME] [--method {logical,tree}]
Cluster diagnosis.
optional arguments:
  -h, --help            show this help message and exit
```


版权所有 © 华为云计算技术有限公司

```
--conf CONF           set the directory of configuration files
--host HOST           set the host of the cluster node, ip only.
--role {cn,dn}        set the role of instance for diagnosis. roles: [cn]
                    are not supported for centralized DB.
--time TIME           set time for diagnosis in timestamp(ms) or datetime format
--method {logical,tree}
                    set method for the model: logical: if-else, tree: xgboost.
```

11.3.8.4 命令参考

表 11-44 命令行参数说明

参数	参数说明	取值范围
-h, --help	帮助命令。	-
--conf	连接时序数据库需要的配置文件地址。	-
--host	分析的目标节点的IP地址。	-
--role	分析的目标节点的角色。	目前仅支持{cn, dn}。
--time	分析的异常发生的时间点，默认值是当前时间，日期时间格式为 %Y-%m-%d %H:%M:%S。	日期时间格式或者以毫秒为单位的时间戳。
--method	分析的方法，目前提供故障定位逻辑模型以及决策树模型两种方法。	{logical, tree}。

11.3.8.5 常见问题处理

如果尝试启动集群诊断时发现系统不能返回预期的诊断结果，需按照以下顺序逐项排查：

- 检查连接时序数据库需要的配置文件地址是否输入正确，是否包含集群诊断需要读取的日志文件，所用账户是否有权限读写日志，以及日志文件是否损坏，能否正常读写。
- 检查目标节点的IP地址是否有误。
- 检查是否支持对所选定的集群进行诊断，集群诊断目前仅支持对CN与DN的诊断，后期将会加入对GTM与ETCD的支持，并检查参数是否输入正确。
- 检查所键入的异常发生时间点是否符合日期时间格式，此外，集群诊断仅支持对当前及过去时刻的诊断。
- 检查所选择的诊断方法是否有输入正确，并且是否超出支持范围。
- 对于网卡故障的监测，建议在时序数据库采集数据时将cmd-exporter同时挂载在多个网卡上，以便在其中某一网卡故障时，其他网卡仍然能正常将消息送出。

11.3.9 智能巡检

11.3.9.1 概述

在现网业务中需要定期对数据库实例进行巡检并输出健康报告。DBMind构建了20+个巡检项，包括硬件状态、实例状态、数据库资源、数据库性能与诊断优化等类别。对

版权所有 © 华为云计算技术有限公司

其中关键资源指标趋势进行风险判断，避免不易发现的潜在问题影响实例健康。同时，支持生成日报、周报、月报供用户查看，并基于当前巡检项情况给出健康评分。支持用户对巡检项自定义设置告警阈值，方便用户根据业务特性进行调整，避免无意义告警。

说明

- 月报与周报生成基于日报：如果未构建日报，则无法生成周报与月报；当日报数量不满足连续7天时，无法生成这7天对应的周报；当日报数量不满足连续14天以上时，无法生成对应的月报。此外，周报和月报的生成基于实例，比如出现实例IP和port的变化，会导致周报/月报无法生成；如果节点实例被删除，会导致周报/月报无法生成。采用定时方式生成周报和月报时，建议控制并发量，避免影响实时任务。配置文件中需要将实例的IP映射关系写到ip_map中，否则无法保证巡检结果的实例为管理IP。
- 周报和月报仅支持所有巡检项的巡检，不支持部分巡检项的拼接；因此周报和月报的自定义阈值需要由前端传入，规则与实时巡检等一致，没有传入的巡检项默认不进行告警（除了部分由前端控制的巡检项，见**10.3.9.2-使用指导**）
- 调用巡检接口时需要传递实例，该实例必须为集群主节点实例。
- 智能巡检中db_size、buffer_hit_rate、db_tmp_file、db_deadlock、db_transaction等巡检项仅展示部署了opengauss-exporter的CN和DN的数据库结果。如果出现DN主备切换的情况，需要部署opengauss-exporter才可以获取新的节点结果。
- 智能巡检中db_size、buffer_hit_rate、db_transaction、db_tmp_file、db_deadlock等巡检项展示部署了opengauss-exporter的CN和DN的数据库结果；user_login_out、db_latency、thread_pool等巡检项展示CN的结果；xlog_accumulate巡检项展示DN的结果。
- log_error_check和core_dump两个巡检项为了不漏掉关键信息，在实时巡检时仅支持45小时内数据；其他巡检项不能超过TSDB数据存储范围。
- 日报无论是否选择GUC参数巡检，均不会影响周检和月检的全量校验，依然能够生成周报/月报；此外，周报和月报不选择GUC参数巡检，不会获取GUC参数巡检结果；选择GUC参数巡检，均会生成GUC参数巡检结果。
- 智能巡检暂不支持容灾集群。
- 分布式多节点集群，需要将opengauss-exporter部署在所有CN实例上，才可以获取到巡检结果。
- 当实例集群只有单节点时，gaussdb_ping_packet_rate指标没有数据，无法进行网络状态巡检获取对应数据。
- 动态内存、程序内存、其他内存三个巡检项依赖于"pg_total_memory_detail_mbytes"指标，当"pv_total_memory_detail"视图无法查询时，无法获取相关数据进行巡检。
- 负载高、网络异常和集群不可用的场景下，巡检功能不可用。

  oldestXmin表示长时间未推进巡检项。因为需要采集足够12个小时的数据来判断是否推进，所以在前12小时内，该巡检项无数据。

表 11-45 巡检项与权重设置

巡检类	巡检项	巡检字段	分数权重
系统资源 system_resource	CPU使用率。	os_cpu_usage	0.06
	系统磁盘占用率。	os_disk_usage	0.06
	内存使用率。	os_mem_usage	0.06
	磁盘I/O使用率。	os_disk_ioutils	0.06
	网络状况。	network_packet_loss	0.06

版权所有 © 华为云计算技术有限公司

巡检类	巡检项	巡检字段	分数权重
实例状态 instance_status	组件异常。	component_error	0.05
数据库资源 database_resource	数据库目录占用率。	data_directory	0.04
	数据库日志目录占用率。	log_directory	0.04
	数据库大小。	db_size	0
数据库性能 database_performance	Buffer命中率。	buffer_hit_rate	0.03
	用户登录或者退出登录次数。	user_login_out	0
	活跃Session率。	active_session_rate	0.03
	日志异常检查。	log_error_check	0.03
	线程池占用率。	thread_pool	0.03
	数据库时延。	db_latency	0.08
	数据库事务。	db_transaction	0
	数据库临时文件大小。	db_tmp_file	0
	数据库执行语句。	db_exec_statement	0
	数据库死锁。	db_deadlock	0.03
	数据库TPS性能。	db_tps	0
	数据库Top Query。	db_top_query	0
	长事务。	long_transaction	0.05
	xLog堆积。	xlog_accumulate	0.05
	oldestXmin长时间未推进。	xmin_stuck	0.05
诊断优化 diagnosis_optimization	Core dump。	core_dump	0.04
	动态内存。	dynamic_memory	0.04
	程序内存。	process_memory	0.04
	其他内存。	other_memory	0.03
	GUC参数。	guc_params	0.04

版权所有 © 华为云计算技术有限公司

各巡检项实现细节与告警设置

- 系统资源
  - CPU使用率：获取TSDB中的“os_cpu_user_usage”和“os_cpu_iowait_usage”两个指标。
    - 对于os_cpu_user_usage，观测数值是否大于0.7，若超出阈值，则进行告警；最后，进行趋势预测，判断未来24小时数值是否大于0.7，若超出阈值，则进行告警。
    - 对于os_cpu_iowait_usage，当数值大于0.3时，进行告警。
  - 系统磁盘占用率：获取TSDB中的“os_disk_usage”指标，观测其中系统盘（mount_point='/'）的磁盘使用率，当数值大于0.8时，进行告警。
  - 内存使用率：获取TSDB中的“os_mem_usage”指标，当发现持续上升时会进行告警；当数值大于0.7时，进行告警；最后，进行趋势预测，如果判断未来24小时数值大于0.8，进行告警。
  - 磁盘I/O使用率：获取TSDB中的“os_disk_ioutils”指标，当数值大于0.8时，进行告警。
  - 网络状况：获取TSDB中的“gaussdb_ping_packet_rate”指标，得出数据库节点间的网络连接状况，当丢包率大于0.05时，进行告警。
- 实例状态
  - 组件异常：根据instance从元数据库中获取节点的状态信息，得出时间范围内当前集群中CN和DN的状态，当出现异常状态（即值不为-1）时，进行告警。
- 数据库资源
  - 数据库目录占用率：获取TSDB中的“pg_node_info_uptime”指标，解析出各个数据库实例的数据目录，同时获取“gaussdb_process_cpu_usage”指标，匹配判断是否存在change_root，存在则组合成正确的目录路径，然后获取“os_disk_usage”指标，匹配判断对应数据目录的占用率，当发现持续上升时，进行告警；当数值大于0.8时，进行告警；最后，进行趋势预测，如果判断未来24小时数值大于0.8，进行告警；判断文件系统是否有误，不匹配则进行告警。
  - 数据库日志目录占用率：获取TSDB中的“pg_node_info_uptime”指标，然后解析出各个数据库实例的日志目录，同时获取“gaussdb_process_cpu_usage”指标，匹配判断是否存在change_root，存在则组合成正确的目录路径，然后获取“os_disk_usage”指标，匹配判断对应日志目录的占用率，当发现持续上升时会进行告警，当数值大于0.8时，同样进行告警，再进行趋势预测，如果判断未来24小时数值大于0.8，也会进行告警，最后还会判断文件系统是否有误，不匹配则进行告警。
  - 数据库大小：获取TSDB中的“pg_database_size_bytes”指标，获取所有数据库的大小并进行记录，用于前端进行趋势展示。暂无告警。
- 数据库性能
  - Buffer命中率：获取TSDB中的“pg_db_blks_access”指标，当数值小于0.9时，进行告警。
  - 用户登录或者退出登录次数：获取TSDB中的“gaussdb_user_login”和“gaussdb_user_logout”两个指标，获取1分钟内平均每秒用户登录或者退出登录次数，用于前端进行趋势展示。暂无告警。
  - 活跃Session率：获取TSDB中的活跃会话数gaussdb_active_connection和总会话数gaussdb_total_connection指标，活跃session率即为

 版权所有 © 华为云计算技术有限公司

gaussdb_active_connection / gaussdb_total_connection，当其比值小于0.8时，进行告警。

- 日志异常检查：获取TSDB中“gaussdb_log_*”的指标，统计每种日志错误出现的次数并进行相加，如果出现日志异常，则将日志异常的出现次数返回前端，进行告警。具体指标为'deadlock_count', 'login_denied', 'errors_rate', 'panic', 'dn_ping_standby', 'node_restart', 'node_start', 'cn_status', 'dn_status', 'gtm_status', 'dn_writable_failed', 'etcd_io_overload', 'cms_heartbeat_timeout_restart', 'cms_phonydead_restart', 'cms_cn_down', 'cn_restart_time_exceed', 'cms_read_only', 'cms_restart_pending', 'cms_heartbeat_timeout', 'etcd_restart', 'etcd_not_connect_dial_tcp', 'etcd_auth_failed', 'etcd_overload', 'etcd_sync_timeout', 'etcd_disk_full', 'etcd_be_killed', 'bind_ip_failed', 'gtm_disconnected_to_primary', 'gtm_panic', 'ffic'。
- 线程池占用率：获取TSDB中的“pg_thread_pool_rate”指标，当发现持续上升时会进行告警，当数值大于0.9时，进行告警。
- 数据库时延：获取TSDB中的“statement_responsetime_percentile_p95”和“statement_responsetime_percentile_p80”两个指标暂无告警。
- 数据库事务：获取TSDB中的“pg_db_xact_commit”和“pg_db_xact_rollback”两个指标，获取数据库中提交和回滚的次数，用于前端进行趋势展示。暂无告警。
- 数据库临时文件大小：获取TSDB中的“pg_db_temp_files”指标，获取数据库临时文件的大小，用于前端进行趋势展示。暂无告警。
- 数据库执行语句：获取TSDB中的“pg_sql_count_select”、“pg_sql_count_update”、“pg_sql_count_insert”和“pg_sql_count_delete”四个指标，获取数据库中select、update、insert、delete的执行次数，用于前端进行趋势展示。暂无告警。
- 数据库死锁：获取TSDB中的“pg_db_deadlocks”指标，获取数据库中出现的死锁情况，当死锁数不为0时，表示数据库出现死锁。数值表示死锁数量。暂无告警。
- 数据库TPS性能：获取TSDB中的“gaussdb_qps_by_instance”和“qps”两个指标，获取数据库的QPS和TPS信息，用于前端进行趋势展示。暂无告警。
- 数据库Top Query：通过执行SQL语句从dbe_perf.statement查询出现调用次数最多的TopK个查询，用于前端展示。当前K为10。暂无告警。
- 长事务：通过执行SQL语句从pg_catalog.pg_stat_activity查询超过12小时未结束的活跃事务，当返回结果列表的长度大于0时，表示存在长事务，进行告警。
- xLog堆积：获取TSDB中的“gaussdb_xlog_count”指标，获取xlog目录下的文件数量，当数量大于3000时，进行告警。
- oldestXmin长时间未推进：获取TSDB中的“oldestxmin_increase”指标，当oldestxmin超过12小时未出现变化，则表示长时间未推进，此时巡检返回数据中数值为0，进行告警。

- 诊断优化
  - Core dump：获取TSDB中的“gaussdb_log_ffic”指标，当返回值中count大于0时，表示出现core dump，进行告警。
  - 动态内存：获取TSDB中的“pg_total_memory_detail_mbytes”指标，计算其中“dynamic_used_memory”相对于“max_dynamic_memory”的比例，当数值大于0.8时，进行告警；计算其中“dynamic_used_shrctx”相对于“max_dynamic_memory”的比例，当数值大于0.8时，同样进行告警。

 版权所有 © 华为云计算技术有限公司

- 程序内存：获取TSDB中的“pg_total_memory_detail_mbytes”指标，计算其中“process_used_memory”相对于“max_process_memory”的比例，当数值大于0.8时，进行告警。
- 其他内存：获取TSDB中的“pg_total_memory_detail_mbytes”指标，获取其中“other_used_memory”的值，当发现持续上升时会进行告警，当超过20G时，进行告警。
- GUC参数：获取TSDB中的“pg_total_memory_detail_mbytes”和“pg_settings_setting”指标，获取其中“max_process_memory”、“shared_buffers”和“work_mem”的值，将当前的值与DBMind根据workload推算出的最优参数进行对比，如果当前参数与最优参数相差大，结果中的warning字段为true，进行告警。对于"work_mem"，集中式、分布式独立部署CN/DN时，最优值a= (内存/16G ）* 16MB，如果a>128MB，设置为128MB，若a < 16MB, 设置 16MB。基于最优值取[0.9a, 1.1a]作为可选区间，当实际work_mem参数在[0.9a, 1.1a]范围外时，进行告警。。对于"max_process_memory"，1. 内存> 256GB, 设置：内存 *0.875，2. 内存 <=256GB && 内存>=128GB, 设置： 内存*0.75，3. 内存<128GB, 设置：内存 * 0.625。对于"shared_buffer"，集中式：内存/4不可以为0，最小值64MB，数值满足8M整数倍。

说明

健康评分根据各巡检项的重要性权重以及异常程度来给出评分，最后汇总成当前实例的健康评分，具体计算公式如下：

$$score = \sum_{i=1}^{n} w_i * S_i$$

上述：

$$\sum_{i=1}^{n} w_i = 100$$

S_i = 0 or 1（其中0代表巡检项异常，1代表正常）。

此外，结论中会提供TOP3列表，TOP3指的是异常情况最严重的三个巡检项，通过统计各个巡检项的异常数量，再结合巡检项的权重进行排序，取前3个作为TOP3列表，如果没有异常，则为空。

各巡检项的异常数量：网络状态按照双向通信异常进行计算；buffer_hit_rate巡检项按照数据库实例异常数量计算；长事务按照长事务数量进行计算；其他巡检项按照节点异常数量进行计算。

支持的巡检自定义阈值

当前巡检项的告警类型与默认设置如下表所示。

- √和数字表示该巡检项已设置默认告警阈值。
- -表示该巡检项支持设置对应告警类型。
- ×表示该巡检项不支持设置对应告警类型：

版权所有 © 华为云计算技术有限公司

巡检项	巡检项	数值范围	异常类型				判断标准
			超过阈值	预测超过阈值	持续上升	文件类型有误	
CPU使用率-os_cpu_user_usage	os_cpu_usage	[0, 1]	>0.7	>0.7	×	×	任意异常出现则告警。
CPU使用率-os_cpu_iowait_usage	os_cpu_usage	[0, 1]	>0.3	-	-	×	任意异常出现则告警。
磁盘I/O使用率	os_disk_ioutils	[0, 1]	>0.8	-	-	×	任意异常出现则告警。
系统磁盘占用率	os_disk_usage	[0, 1]	>0.8	-	-	×	任意异常出现则告警。
内存使用率	os_mem_usage	[0, 1]	>0.7	>0.8	√	×	任意异常出现则告警。
网络状况	network_packet_loss	[0, 1]	>0.05	-	×	×	任意异常出现则告警。
组件异常	component_error	×	×	×	×	×	任意异常出现则告警。
数据库目录占用率	data_directory	[0, 1]	>0.8	>0.8	-	√	任意异常出现则告警。
数据库日志目录占用率	log_directory	[0, 1]	>0.8	>0.8	-	√	任意异常出现则告警。
数据库大小	db_size	[0, ∞)	-	-	-	×	暂无告警。
Buffer命中率	buffer_hit_rate	[0, 1]	<0.9	-	×	×	任意异常出现则告警。
用户登录或者退出登录次数	user_login_out	[0, ∞)	-	-	-	×	暂无告警。
活跃Session率	active_session_rate	[0, 1]	<0.8	-	×	×	任意异常出现则告警。
日志异常检查	log_error_check	×	×	×	×	×	存在则告警。
线程池占用率	thread_pool	[0, 1]	>0.9	-	√	×	任意异常出现则告警。
数据库时延	db_latency	[0, ∞)	-	-	×	×	任意异常出现则告警。
数据库事务	db_transaction	[0, ∞)	-	-	-	×	暂无告警。

版权所有 © 华为云计算技术有限公司

数据库临时文件大小	db_tmp_file	[0, ∞)	-	-	-	×	暂无告警。
数据库执行语句	db_exec_statement	[0, ∞)	-	-	×	×	暂无告警。
数据库死锁	db_deadlock	[0, ∞)	-	-	-	×	暂无告警。
数据库TPS性能	db_tps	[0, ∞)	-	-	-	×	暂无告警。
数据库Top Query	db_top_query	×	×	×	×	×	暂无告警。
长事务	long_transaction	×	×	×	×	×	存在则告警。
oldestXmin长时间未推进	xmin_stuck	[0, ∞)	×	×	×	×	任意异常出现则告警。
xLog堆积	xlog_accumulate	[0, ∞)	>3000	-	-	×	任意异常出现则告警。
Core dump	core_dump	×	×	×	×	×	存在则告警。
动态内存	dynamic_memory	[0, 1]	>0.8	-	-	×	任意异常出现则告警。
程序内存	process_memory	[0, 1]	>0.8	-	-	×	任意异常出现则告警。
其他内存	other_memory	[0, ∞)	>20*1024	-	√	×	任意异常出现则告警。
GUC参数	guc_params	×	×	×	×	×	warning中存在值则告警。

说明

- 持续上升和预测超过阈值两类告警涉及检测算法，会占用DBMind性能，配置时减少开启的数量，避免影响服务性能以及返回时间等。
- 持续上升检测器除了判断数据本身持续上升以外，还需要时间范围内80%以上的数据超过阈值范围，才判断为持续上升告警。
- 预测超过阈值（告警）需要设置预测的时间范围，以分钟为单位，设置范围(0, 48 * 60]。建议设置的预测时间长度小于巡检数据的时间长度，或者通过自定义阈值调整预测的时间长度。否则，预测告警的准确率无法得到保证。
- 预测超过阈值（告警）会预测未来时间范围内的数值，判断数值是否存超过阈值。预测超过阈值对于巡检数据不进行判断，如果巡检数据已经超过阈值，仍会继续进行预测，判断未来是否会一直超过阈值，还是会在某个时间点回归正常范围。

版权所有 © 华为云计算技术有限公司

11.3.9.2 使用指导

通过巡检相关 API 实现功能调用

- 智能巡检接口

  示例：

  ```
  curl -X 'POST' "http://127.0.0.1:8080/v1/api/app/real-time-inspection?inspection_type=real_time_check&start_time=1689210000000&end_time=1689296400000&instance=127.0.0.1:5432" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"system_resource": ["os_mem_usage"], "instance_status": [], "database_resource": [], "database_performance": [], "diagnosis_optimization": []}' -H "Authorization: bearer xxx"
  ```

  如果使用HTTPS协议，则查询示例：

  ```
  curl -X 'POST' "https://127.0.0.1:8080/v1/api/app/real-time-inspection?inspection_type=real_time_check&start_time=1689210000000&end_time=1689296400000&instance=127.0.0.1:5432" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"system_resource": ["os_mem_usage"], "instance_status": [], "database_resource": [], "database_performance": [], "diagnosis_optimization": []}' -H "Authorization: bearer xxx" --cacert xx.crt --key xx.key --cert xx.crt
  ```

  如果使用自定义阈值，查询示例：

  ```
  curl -X 'POST' "https://127.0.0.1:8080/v1/api/app/real-time-inspection?inspection_type=real_time_check&start_time=1689210000000&end_time=1689296400000&instance=127.0.0.1:5432" -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"system_resource": [{"os_mem_usage": {"increase": false, "threshold": [], "forecast": [1440, 0.0, 0.8]}}], "instance_status": [], "database_resource": [], "database_performance": [], "diagnosis_optimization": []}' -H "Authorization: bearer xxx" --cacert xx.crt --key xx.key --cert xx.crt
  ```

  返回结构示例如下：

  ```
  {
    "data": {
      "conclusion": {
        "full_score": 0.06,
        "health_score": 0.06,
        "health_status": "bad",
        "top3": []
      },
      "database_performance": {},
      "database_resource": {},
      "diagnosis_optimization": {},
      "instance_status": {},
      "system_resource": {
        "os_mem_usage": {
          "127.0.0.1": {
            "data": [0.31643905373281667],
            "statistic": {
              "avg": 0.311,
              "max": 0.3166,
              "min": 0.3057,
              "the_95th": 0.3153
            },
            "timestamps": [1694674713000],
            "warnings": {
              "increase_warning": true
            }
          }
        }
      }
    },
    "success": true
  }
  ```

- 展示巡检任务接口示例：

  ```
  curl -X 'GET' "http://127.0.0.1:8080/v1/api/app/real-time-inspection/list?instance=127.0.0.1:5432" -H 'accept: application/json' -H "Authorization: bearer xxx"
  ```

  如果使用HTTPS协议，则查询示例：

版权所有 © 华为云计算技术有限公司

```
curl -X 'GET' "https://127.0.0.1:8080/v1/api/app/real-time-inspection/list?instance=127.0.0.1:5432" -H 'accept: application/json' -H "Authorization: bearer xxx" --cacert xx.crt --key xx.key --cert xx.crt
```

返回结构如下：

```
{"data":{"header":["instance","start","end","id","state","cost_time","inspection_type"],"rows":[["127.0.0.1:5432",1689210000000,1689296400000,5,"success",0.033701,"real_time_check"]]},"success":true}
```

- 获取指定巡检任务的巡检结果接口示例：

```
curl -X 'GET' "http://127.0.0.1:8080/v1/api/summary/real-time-inspection?spec_id=5&instance=127.0.0.1:5432" -H 'accept: application/json' -H "Authorization: bearer xxx"
```

如果使用HTTPS协议，则查询示例为：

```
curl -X 'GET' "https://127.0.0.1:8080/v1/api/summary/real-time-inspection?spec_id=5&instance=127.0.0.1:5432" -H 'accept: application/json' -H "Authorization: bearer xxx" --cacert xx.crt --key xx.key --cert xx.crt
```

返回结构如下，与**智能巡检接口返回结构**一致。

- 删除指定的巡检任务接口示例：

```
curl -X 'DELETE' "http://127.0.0.1:8080/v1/api/app/real-time-inspection?spec_id=5&instance=127.0.0.1:5432" -H 'accept: application/json' -H "Authorization: bearer xxx"
```

如果使用HTTPS协议，则删除示例：

```
curl -X 'DELETE' "https://127.0.0.1:8080/v1/api/app/real-time-inspection?spec_id=5&instance=127.0.0.1:5432" -H 'accept: application/json' -H "Authorization: bearer xxx" --cacert xx.crt --key xx.key --cert xx.crt
```

返回结构如下：

```
{"data":{"success":true},"success":true}
```

【自定义阈值参数设置】

- 自定义阈值传参方式，各告警类型对应的key如**表1**所示：

表 11-46 告警类型以及对应的 key

告警类型	键
持续上升	increase
超过阈值	threshold
预测超过阈值	forecast
文件类型有误	ftype

- 启用默认告警配置（505.1.0之前的版本）

```
{
"system_resource": [
"os_mem_usage"
],
"instance_status": [],
"database_resource": [],
"database_performance": [],
"diagnosis_optimization": []
}
```

- 启用默认告警配置（505.1.0及之后版本）

```
{
"system_resource": [
{
"os_mem_usage": true
}
```


版权所有 © 华为云计算技术有限公司

```
],
"instance_status": [],
"database_resource": [],
"database_performance": [],
"diagnosis_optimization": []
}
```

- 不启用告警

```
{
"system_resource": [
{
"os_mem_usage": false
}
],
"instance_status": [],
"database_resource": [],
"database_performance": [],
"diagnosis_optimization": []
}
```

- 启用自定义告警。如下所示，os_mem_usage启用持续上升告警、自定义设置阈值告警和预测阈值告警；os_disk_usage不启用持续上升告警、阈值告警，启用预测阈值告警：

```
{
"system_resource": [
{
"os_mem_usage": {
"increase": true,
"threshold": [0.0, 0.8],
"forecast": [1440, 0.0, 0.8]
},
"os_disk_usage": {
"forecast": [1440, 0.0, 0.8]
}
}
],
"instance_status": [],
"database_resource": [],
"database_performance": [],
"diagnosis_optimization": []
}
```

- 部分巡检项不支持自定义配置阈值：组件异常、日志异常检查、数据库Top Query、长事务、oldestXmin长时间未推进、Core dump、GUC参数，是否告警由云侧前端控制，忽略由前端传入的参数。
- 各个巡检项支持的阈值类型见**10.3.9-概述**中自定义阈值表格，前端传入不支持的告警类型会报错。
- 对于部分巡检项，存在子巡检项，如**表11-47**所示：

表 11-47 巡检项及子巡检项

巡检项	子巡检项
os_cpu_usage	cpu_user
	cpu_iowait
user_login_out	login
	logout
db_latency	p95
	p80

 版权所有 © 华为云计算技术有限公司

巡检项	子巡检项
db_transaction	commit
	rollback
db_exec_statement	select
	update
	insert
	delete
db_tps	tps
	qps
dynamic_memory	dynamic_used_memory
	dynamic_used_shrctx

当子巡检项的告警配置一致时，可以省略重复值，具体方式如下：

- 重复设置os_cpu_usage阈值自定义

```
{
"system_resource": [
{
"os_cpu_usage": {
"cpu_user": {
"increase": false,
"threshold": [],
"forecast": [1440, 0.0, 0.8]
},
"cpu_iowait": {
"increase": false,
"threshold": [],
"forecast": [1440, 0.0, 0.8]
}
}
}
],
"instance_status": [],
"database_resource": [],
"database_performance": [],
"diagnosis_optimization": []
}
```

- 简化设置os_cpu_usage阈值自定义

```
{
"system_resource": [
{
"os_cpu_usage": {
"increase": false,
"threshold": [],
"forecast": [1440, 0.0, 0.8]
}
}
],
"instance_status": [],
"database_resource": [],
"database_performance": [],
"diagnosis_optimization": []
}
```

 版权所有 © 华为云计算技术有限公司

巡检项结果返回值示例

- 集群：

  以3CN、3主DN、6备DN为例，共3个节点，且opengauss-exporter部署在CN和主DN上（分布式仅部署在CN上，结果仅展示CN，无法获取DN节点上信息）：

  3CN: 127.0.0.1:9080, 127.0.0.2:9080, 127.0.0.3:9080

  3主DN: 127.0.0.1:40001, 127.0.0.2:50001, 127.0.0.3:60001

  6备DN: 127.0.0.1:5000, 127.0.0.1:60001, 127.0.0.2:40001, 127.0.0.2:60001, 127.0.0.3:40001, 127.0.0.3:50001

- 返回值结构类型：
  - a)：以节点为key：{"127.0.0.1": xxx, "127.0.0.2": xxx, "127.0.0.3": xxx}
  - b)：以CN为key：{"127.0.0.1:9080": xxx, "127.0.0.2:9080": xxx, "127.0.0.3:9080": xxx}
  - c)：以cn和主dn为key(以opengauss-exporter部署的cn和dn为key)：{"127.0.0.1:9080": xxx, "127.0.0.2:9080": xxx, "127.0.0.3:9080": xxx, "127.0.0.1:40001": xxx, "127.0.0.2:50001": xxx, "127.0.0.3:60001": xxx}
  - d)：以dn为key：{"127.0.0.1:40001": xxx, "127.0.0.2:50001": xxx, "127.0.0.3:60001": xxx, "127.0.0.2:40001": xxx, "127.0.0.3:50001": xxx, "127.0.0.1:60001": xxx , "127.0.0.3:40001": xxx, "127.0.0.1:50001": xxx, "127.0.0.2:60001": xxx }

巡检项对应的返回结构如**表11-48**所示：

表 11-48 巡检项返回结构

巡检项	分布式	备注
os_cpu_usage	a)	两层结构，子巡检项：cpu_user、cpu_iowait。
os_disk_ioutils	a)	-
os_disk_usage	a)	-
os_mem_usage	a)	-
network_packet_loss	a)	两层结构，展示节点到节点的网络状况。
component_error	a)	两层结构，子巡检项：cn、dn。
data_directory	c)	-
log_directory	c)	-
db_size	c)	两层结构，{"ip:port": {"db": xxx}}
buffer_hit_rate	c)	两层结构，{"ip:port": {"db": xxx}}
user_login_out	c)	两层结构，子巡检项：login、logout

版权所有 © 华为云计算技术有限公司

巡检项	分布式	备注
active_session_rate	c)	-
log_error_check	a)	-
thread_pool	c)	-
db_latency	c)	两层结构，子巡检项：p80、p95。
db_transaction	c)	三层结构：{"commit": {"ip:port": {"db": xxx}}}。
db_tmp_file	c)	两层结构，{"ip:port": {"db": xxx}}。
db_exec_statement	c)	两层结构，子巡检项：select、update、insert、delete。
db_deadlock	c)	两层结构，{"ip:port": {"db": xxx}}。
db_tps	c)	两层结构，子巡检项：qps、tps。
db_top_query	b)	两层结构，{"ip:port": [{xxx}]}。
long_transaction	b)	两层结构，{"ip:port": [{xxx}]}。
xmin_stuck	c)	-
xlog_accumulate	d)	-
core_dump	a)	-
dynamic_memory	c)	两层结构，子巡检项：dynamic_used_memory、dynamic_used_shrctx。
process_memory	c)	-
other_memory	c)	-
guc_params	c)	-

说明

巡检结果会存储到DBMind元数据库中，DBMind会定期清除老数据以避免磁盘膨胀。

版权所有 © 华为云计算技术有限公司

11.3.9.3 命令参考

表 11-49 智能巡检接口

API	入参	参数介绍	请求方法	功能描述与预期返回结果
/v1/api/app/real-time-inspection	inspection_type start_time end_time tz instance inspection_items	• 巡检类型，String，必选。 • 起始时间，String，可选。 • 终止时间，String，可选。 • 时区信息，String，可选。 • 实例IP:PORT，String，必选。 • 巡检项，dict，必选。	POST	执行智能巡检功能并返回巡检结果。
/v1/api/app/real-time-inspection/list	instance	实例IP:PORT，String，必选	GET	展示巡检任务的基础信息。
/v1/api/summary/real-time-inspection	instance spec_id	• 实例IP:PORT，String，必选。 • 巡检任务ID，String，必选。	GET	获取指定巡检任务的巡检结果。
/v1/api/app/real-time-inspection	instance spec_id	• 实例IP:PORT，String，必选。 • 巡检任务ID，String，必选。	DELETE	删除指定的巡检任务。

11.3.9.4 常见问题处理

- 月报与周报生成基于日报：如果未构建日报，则无法生成周报与月报；当日报数量不满足连续7天时，无法生成这7天对应的周报；当日报数量不满足连续14天以上时，无法生成对应的月报。此外，周报和月报的生成基于实例，比如出现实例

版权所有 © 华为云计算技术有限公司

IP和port的变化，会导致周报/月报无法生成；比如节点实例被删除，会导致周报/月报无法生成。采用定时方式生成周报和月报时，建议控制并发量，避免影响实时任务。配置文件中需要将实例的IP映射关系写到ip_map中，否则无法保证巡检结果的实例为管理IP。

- 周报和月报仅支持所有巡检项的巡检，不支持部分巡检项的拼接；因此周报和月报的自定义阈值需要由前端传入，规则与实时巡检等一致，没有传入的巡检项默认不进行告警（除了部分由前端控制的巡检项，见**10.3.9.2-使用指导**）。
- 调用巡检接口时需要传递实例，该实例需要为集群主节点实例，否则在节点出现异常时无法返回所有节点的巡检结果。
- 智能巡检中db_size、buffer_hit_rate、db_tmp_file、db_deadlock、db_transaction等巡检项仅展示部署了opengauss-exporter的CN和DN的数据库结果。如果出现DN主备切换的情况，需要部署opengauss-exporter才可以获取新的节点结果。
- 智能巡检中db_size、buffer_hit_rate、db_transaction、db_tmp_file、db_deadlock等巡检项展示部署了opengauss-exporter的CN和DN的数据库结果；user_login_out、db_latency、thread_pool等巡检项展示CN的结果；xlog_accumulate巡检项展示DN的结果。
- log_error_check和core_dump两个巡检项为了不漏掉关键信息，在实时巡检时仅支持45小时内数据；其他巡检项不能超过TSDB数据存储范围。

版权所有 © 华为云计算技术有限公司

12 ABO 优化器

12.1 智能基数估计

12.1.1 概述

智能基数估计使用库内贝叶斯网络模型对多列数据样本联和分布进行建模，从而能够对多列等值查询提供更加准确的基数估算。更加准确的基数估算能够显著提高优化器对于计划和算子选择的准确性，从而提高数据库整体吞吐量。

12.1.2 前置条件

数据库运行正常，GUC参数enable_ai_stats设置为on，multi_stats_type设置为'BAYESNET'或者'ALL'。

12.1.3 使用指导

步骤1 采样方式设置为按照采样率采样，即设置GUC参数default_statistics_target为[-100, -1]之间的整数，表示采样百分比。

步骤2 使用ANALYZE(([column_name,])) 进行数据统计和模型创建。

步骤3 输入查询，如果查询涉及到的等值查询列上有统计模型创建，那么会自动是使用统计模型进行选择率估计。

步骤4 不再需要智能统计模型的时候，使用ALTER TABLE [table_name] DELETE STATISTICS (([column_name,]))进行统计信息以及模型删除。

----结束

其他使用方法和接口详见《开发指南》中“SQL参考 > SQL语法 > ALTER TABLE和ANALYZE | ANALYSE”章节。

12.1.4 最佳实践

生成如下数据表：

```
benchmark=# \d part;
id              | integer                 |
```

 版权所有 © 华为云计算技术有限公司

```
p_brand        | character varying(256)   |
p_type         | character varying(256)   |
p_container    | character varying(256)   |
p_mfgr         | character varying(256)   |
```

插入10,000,000行数据：

```
benchmark=# select count(1) from part1;
10000000
```

在数据表上创建四种不同的多列索引：

```
benchmark=# select * from pg_indexes where tablename='part1';
public     | part1     | brand_type_container   |              | CREATE INDEX brand_type_container ON part1
USING btree (p_brand, p_type, p_container) TABLESPACE pg_default
public     | part1     | brand_type_mfgr        |              | CREATE INDEX brand_type_mfgr ON part1 USING
btree (p_brand, p_type, p_mfgr) TABLESPACE pg_default
public     | part1     | brand_container_mfgr   |              | CREATE INDEX brand_container_mfgr ON part1
USING btree (p_brand, p_container, p_mfgr) TABLESPACE pg_default
public     | part1     | type_container_mfgr    |              | CREATE INDEX type_container_mfgr ON part1
USING btree (p_type, p_container, p_mfgr) TABLESPACE pg_default
```

针对数据表生成一批包含多列等值条件的查询，如下：

```
explain analyze select * from part1 where p_container='LG CASE' AND p_brand='Brand#34' AND
p_mfgr='Manufacturer#2' AND p_type='SMALL BRUSHED COPPER';
```

分别测试不创建多列统计信息和创建ABO统计信息场景下的执行计划：

```
benchmark=# explain analyze select * from part1 where p_container='LG CASE' AND p_brand='Brand#34'
AND p_mfgr='Manufacturer#2' AND p_type='SMALL BRUSHED COPPER';
  Bitmap Heap Scan on part1 (cost=5.30..336.06 rows=17 width=56) (actual time=0.953..7.061 rows=103
loops=1)
  Recheck Cond: (((p_brand)::text = 'Brand#34'::text) AND ((p_type)::text = 'SMALL BRUSHED COPPER'::text)
AND ((p_container)::text = 'LG CASE'::text))
  Filter: ((p_mfgr)::text = 'Manufacturer#2'::text)
  Rows Removed by Filter: 773
  Heap Blocks: exact=871
  ->  Bitmap Index Scan on brand_type_container  (cost=0.00..5.30 rows=84 width=0) (actual
time=0.704..0.704 rows=876 loops=1)
      Index Cond: (((p_brand)::text = 'Brand#34'::text) AND ((p_type)::text = 'SMALL BRUSHED COPPER'::text)
AND ((p_container)::text = 'LG CASE'::text))
Total runtime: 7.213 ms
benchmark=# explain analyze select * from part1 where p_container='LG CASE' AND p_brand='Brand#34'
AND p_mfgr='Manufacturer#2' AND p_type='SMALL BRUSHED COPPER';
  Bitmap Heap Scan on part1  (cost=10.59..723.97 rows=110 width=56) (actual time=0.112..0.434 rows=103
loops=1)
  Recheck Cond: (((p_type)::text = 'SMALL BRUSHED COPPER'::text) AND ((p_container)::text = 'LG
CASE'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))
  Filter: ((p_brand)::text = 'Brand#34'::text)
  Rows Removed by Filter: 64
  Heap Blocks: exact=167
  ->  Bitmap Index Scan on type_container_mfgr  (cost=0.00..10.54 rows=183 width=0) (actual
time=0.081..0.081 rows=167 loops=1)
      Index Cond: (((p_type)::text = 'SMALL BRUSHED COPPER'::text) AND ((p_container)::text = 'LG
CASE'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))
Total runtime: 0.533 ms
```

通过以上操作可以看出，在这个场景下，ABO基数估算查询加速了10+倍。

12.1.5 常见问题处理

如果遇到异常场景导致模型无法创建，ABO优化器会只创建传统统计信息，请根据对应的告警信息进行相应处理。

版权所有 © 华为云计算技术有限公司

13 Foreign Data Wrapper

GaussDB的FDW（Foreign Data Wrapper）可以实现各个GaussDB数据库及远程服务器（包括数据库、文件系统）之间的跨库操作。目前支持的外部数据封装器类型包括file_fdw。

13.1 file_fdw

file_fdw模块提供了外部数据封装器file_fdw， 可以用来在服务器的文件系统中访问数据文件。数据文件必须是COPY FROM可读的格式，具体请参见《开发指南》中“SQL参考 > SQL语法 > COPY”章节。使用file_fdw访问的数据文件是当前可读的，不支持对该数据文件的写入操作。

当前GaussDB会默认编译file_fdw，initdb的时候会在pg_catalog schema中创建该插件。

file_fdw对应的server和外表只允许数据库的初始用户、系统管理员或开启运维模式时的运维管理员创建。

使用file_fdw创建的外部表可以有下列选项：

- filename
  指定要读取的文件，必需的参数，且必须是一个绝对路径名。
- format
  远端server的文件格式，支持text、csv、binary三种格式，和COPY语句的FORMAT选项相同。
- header
  指定的文件是否有标题行，与COPY语句的HEADER选项相同。
- delimiter
  指定文件的分隔符，与COPY的DELIMITER选项相同。
- quote
  指定文件的引用字符，与COPY的QUOTE选项相同。
- escape
  指定文件的转义字符，与COPY的ESCAPE选项相同。
- null
  指定文件的null字符串，与COPY的NULL选项相同。

 版权所有 © 华为云计算技术有限公司

- encoding

  指定文件的编码，与COPY的ENCODING选项相同。

- force_not_null

  这是一个布尔选项。如果为真，则声明字段的值不应该匹配空字符串（也就是，文件级别null选项）。与COPY的FORCE_NOT_NULL选项里的字段相同。

说明

- file_fdw不支持COPY的OIDS和 FORCE_QUOTE选项。
- 这些选项只能为外部表或外部表的字段声明，不是file_fdw的选项，也不是使用file_fdw的服务器或用户映射的选项。
- 修改表级别的选项需要系统管理员权限。因为安全原因，只有系统管理员能够决定读取的文件。
- 对于一个使用file_fdw的外部表，EXPLAIN可显示要读取的文件名和文件大小（单位：字节）。指定了COSTS OFF关键字之后，不显示文件大小。

使用 file_fdw

- 创建服务器对象：CREATE SERVER。
- 创建用户映射：CREATE USER MAPPING。
- 创建外表：CREATE FOREIGN TABLE。

  说明

  - 外表的表结构需要与指定的文件的数据保持一致。
  - 对外表做查询操作，写操作不被允许。
- 删除外表：DROP FOREIGN TABLE。
- 删除用户映射：DROP USER MAPPING。
- 删除服务器对象：DROP SERVER。

注意事项

- 使用file_fdw需要指定要读取的文件，请先准备好该文件，并授予数据库对该文件的读取权限。
- 不支持DROP EXTENSION file_fdw操作。

 版权所有 © 华为云计算技术有限公司

14 GTM 模式

为适应不同的并发和一致性要求，GaussDB提供了三种不同的GTM模式，GTM、GTM-Lite、GTM-Free。三种模式之间的主要区别是中心事务管理节点GTM的压力和事务处理流程不同，其中GTM模式下，中心事务处理节点的压力最大，比较容易成为性能和并发瓶颈；在GTM-Lite模式下，中心事务处理节点的压力得到减轻，事务处理流程进一步优化，GTM的性能和并发瓶颈得到减轻，在保证一致性的情况下，事务处理能力得到更大限度的提升；在GTM-Free模式下，中心事务管理节点不再参与事务管理，消除了GTM单点瓶颈，可达到更高的事务处理性能，但是在一致性方面，支持所有事务运行完，保证读的外部一致性，不支持分布式事务强一致性读，不支持insert into select * from等依赖于查询结果的事务一致性。

当前版本暂不支持三种模式之间的相互切换，建议使用安装时默认的GTM模式，支持升级前后GTM模式不变化。

相关的GUC参数包括enable_gtm_free和gtm_option，可通过gsql执行show语句查询当前GTM模式：

```
SHOW enable_gtm_free;
SHOW gtm_option;
```

具体模式判断方法如下。

- GTM模式：enable_gtm_free=off 且 gtm_option=0；
- GTM-Lite模式：enable_gtm_free=off 且 gtm_option=1;
- GTM-Free模式：enable_gtm_free=on 或 gtm_option=2。

GTM-lite模式和GTM模式下，GaussDB支持分布式事务强一致和完整的语法功能。GTM-FREE模式下，由于采用了分布式事务最终一致的执行和并发控制机制，因此约束了部分语法的使用场景和使用方式。如果业务确有使用被限制语法的需求，那么在明确了解最终一致性行为的基础之上，需要对业务进行一定程度的改造。主要涉及的约束语法和改造建议如下：

1. 总体原则：所有用户表必须指定分布键（DISTRIBUTE BY），且选择合理：
   - 考虑数据分布均匀。
   - 尽量选择查询中的关联条件作为分布键，保证关联查询不会引起DN节点之间的数据流动。
   - 考虑将表的主键作为分布键。
2. SELECT：
   - 表查询时，WHERE条件应包含所有分布键字段等值查询条件。

 版权所有 © 华为云计算技术有限公司

- 避免在SELECT目标列中使用子查询，可能导致计划无法下推到DN执行，影响执行性能。

3. DML:

   默认不支持跨节点事务，如果所执行的DML语句包含跨节点事务，会报错处理，具体分为两种场景：

   a. 如果用户语句在数据库内部被拆分成多条独立语句执行，会报错：INSERT/UPDATE/DELETE/MERGE contains multiple remote queries under GTM-free mode Unsupport DML two phase commit under gtm free mode. modify your SQL to generate light-proxy or fast-query-shipping plan。这时，需要修改语句，来单节点执行。

      举例：

      insert into t select * from b where b.c = xx;

      假设t表和b表的分布键不同，且上述where条件只会过滤出一条数据。在不打开enable_stream_operator的情况下， 上面的查询在数据内部会被拆分成两条独立语句串行执行：首先执行select * from b where b.c = xx从某个DN节点抽取到目标记录；然后再执行insert into t语句，将抽取的目标记录下发到另一个节点DN节点完成插入。在gtm-free模式下，这样的语句执行方式会返回上述报错。类似的，create table as select * from、带子查询的delete/join/insert等语句，也可能会出现类似报错。

      业务改造方案：在业务执行之前，需要加上set enable_stream_operator=on命令，打开流算子，使得业务语句可以被整体下推执行。

   b. 如果同一个用户语句在数据库内部涉及多节点执行，会报错：Your SQL needs more than one datanode to be involved in。这时，建议对语句进行修改，使得能够单节点执行。

      举例：

      insert into t values(3,3),(1,1);

      假设(3,3)和(1,1)被分布在不同的DN节点上，那么上述语句在数据库内部的执行过程会涉及两个DN节点。在gtm-free模式下，这样的语句执行方式会返回上述报错。

      业务改造方式：对于上述语句，如果业务确有需要在多个节点上执行，需要在语句中添加一个hint来避免报错，如下：

      insert /*+ multinode */ into t values(3,3),(1,1);

      类似的，对delete和update语句也有类似约束，一般建议用户在delete和update语句的where条件中加上分布键等值过滤条件。

4. 建议开发阶段在jdbc连接串内设置application_type=perfect_sharding_type，这样所有跨节点读写操作的SQL都会报错，用来提示开发人员尽早优化语句。

 版权所有 © 华为云计算技术有限公司

15 敏感数据发现

敏感数据发现功能提供函数gs_sensitive_data_discovery()和gs_sensitive_data_discovery_detail()，通过调用的不同函数，指定扫描对象和敏感数据分类器，得到对应扫描对象不同明细级别的敏感数据信息。

使用敏感数据发现函数扫描数据

步骤1 创建扫描目标。

```
gaussdb=# CREATE SCHEMA test_schema1;
gaussdb=# CREATE TABLE test_schema1.normal_table1 (email VARCHAR(50), phonenum text, creditcard
bytea, name BPCHAR(50), encryted_data NVARCHAR2(500));
gaussdb=#  INSERT INTO test_schema1.normal_table1 VALUES ('Gauss_1@huawei.com', '13345678901',
'4951-7843-2960-0886', '高斯',  'r1Pj4DKQopSmZfSr/mXwieGcrE+O2iX3G04w1Tda0zqX/pD/
Az90tLuvy0ZnCmUg0/Kl/g89ZqxxAZzta8jQ6bvicRpdXrDCQTHAfsXYSEScB9Idqtwh7kdm1LHq89qs');
```

步骤2 调用敏感数据发现函数扫描目标。

- 调用gs_sensitive_data_discovery()函数，扫描目标数据，返回统计的扫描结果。

```
gaussdb=# SELECT * FROM gs_sensitive_data_discovery('test_schema1.normal_table1',  'all');
           target           | percentage_of_sensitive_data |
classifier_type_distribution
----------------------------+------------------------------
+-----------------------------------------------------------------------------------------
 test_schema1.normal_table1 | 5/5                          |  Email: 1/5  CreditCard: 1/5  PhoneNumber: 1/5
ChineseName: 1/5  EncryptedContent: 1/5
(1 row)
```

- 调用gs_sensitive_data_discovery_detail()函数，扫描目标数据，返回详细的扫描结果。

```
   schema    |     table     |    column     |    classifier    | laws_and_regulations
-------------+---------------+---------------+------------------+----------------------
 test_schema1 | normal_table1 | email         | Email            | GDPR
 test_schema1 | normal_table1 | phonenum      | PhoneNumber      | GDPR
 test_schema1 | normal_table1 | creditcard    | CreditCard       | GDPR
 test_schema1 | normal_table1 | name          | ChineseName      | GDPR
 test_schema1 | normal_table1 | encryted_data | EncryptedContent | GDPR
(5 rows)
```

步骤3 清理数据。

```
gaussdb=# DROP TABLE test_schema1.normal_table1;
gaussdb=# DROP SCHEMA test_schema1;
(3 row)
```

----结束

 版权所有 © 华为云计算技术有限公司

16 在线 DDL

在线DDL支持对正常业务影响较小的情况下进行DDL，减少对业务的影响，提高业务运行的连续性和数据库的可用性。部分在线DDL只需要修改元数据，不涉及重建数据，操作时长较快，对业务影响较小。传统DDL操作若涉及重建数据，例如修改列类型、移动表空间、VACUUM FULL等操作，会对操作的表加长时间的排他锁，阻塞对该表的并发读写操作，影响用户业务运行。阻塞时长与数据量成正比，当数据量较大时，阻塞时长可达数小时、甚至数天。相比之下，在线DDL不会长时间加排他锁，执行中允许对该表的并发读写操作，减少对用户业务运行影响。

说明

- 在线DDL基本原理请参见《特性描述》的“高可用 > 在线DDL”章节。
- 使用DDL功能时，数据库需正常运行。

使用在线 DDL 操作过程中进行并发读写操作

步骤1 创建表。

```
gaussdb=# CREATE TABLE t1(c1 INT,c2 INT);
gaussdb=# INSERT INTO t1 VALUES(generate_series(1,10), generate_series(1,10));

-- 查看表格
gaussdb=# \d+ t1
                    Table "public.t1"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 c1     | integer |           | plain   |              |
 c2     | integer |           | plain   |              |
Has OIDs: no
Distribute By: HASH(c1)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off
```

步骤2 历史版本已支持的在线DDL操作（仅部分示例，全部历史版本支持的在线DDL操作请参见《开发指南》的“SQL参考 > SQL语法 > Online DDL及其操作”）。

- 在线添加列。

  ```
  gaussdb=# ALTER TABLE t1 ADD COLUMN c3 integer;
  ALTER TABLE
  ```

- 在线删除列。

  ```
  gaussdb=# ALTER TABLE t1 drop COLUMN c3;
  ALTER TABLE
  ```

- 在线重命名表。

  ```
  gaussdb=# ALTER TABLE t1 RENAME TO t2;
  ALTER TABLE
  ```

 版权所有 © 华为云计算技术有限公司

```
gaussdb=# ALTER TABLE t2 RENAME TO t1;
ALTER TABLE
```

步骤3 设置ONLINE关键字并进行相关操作。

- 在线修改列类型。
  ```
  gaussdb=# ALTER TABLE ONLINE t1 ALTER c2 TYPE BIGINT;
  ALTER TABLE
  ```
- 在线添加非空约束。
  ```
  gaussdb=# ALTER TABLE ONLINE t1 ALTER c1 SET NOT NULL;
  ALTER TABLE
  ```
- 在线添加检查约束。
  ```
  gaussdb=# ALTER TABLE ONLINE t1 ADD CONSTRAINT c2_k CHECK(c2 > 0);
  ALTER TABLE
  ```
- 在线修改添加唯一约束。
  ```
  gaussdb=# ALTER TABLE ONLINE t1 ADD UNIQUE(c1);
  NOTICE:  ALTER TABLE / ADD UNIQUE will create implicit index "t1_c1_key" for table "t1"
  ALTER TABLE
  ```
- 在线表级移动表空间。
  ```
  gaussdb=# CREATE TABLESPACE tbs RELATIVE LOCATION 'tablespace1/tbs';
  CREATE TABLESPACE
  gaussdb=# ALTER TABLE ONLINE t1 SET TABLESPACE tbs;
  ALTER TABLE
  ```
- 以上在线DDL操作中，可进行并发读写操作。
  ```
  gaussdb=# INSERT INTO t1 VALUES(11, 11);
  INSERT 0 1
  ```

步骤4 查看表结构。

```
gaussdb=# \d+ t1
                        Table "public.t1"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 c1     | integer | not null  | plain   |              |
 c2     | bigint  |           | plain   |              |
Indexes:
    "t1_c1_key" UNIQUE CONSTRAINT, ubtree (c1) WITH (storage_type=USTORE) TABLESPACE pg_default
Check constraints:
    "c2_k" CHECK (c2 > 0)
Has OIDs: no
Tablespace: "tbs"
Distribute By: HASH(c1)
Location Nodes: ALL DATANODES
Options: orientation=row, compression=no, storage_type=USTORE, segment=off
```

步骤5 使用VACUUM FULL在线回收表空间。

```
gaussdb=# VACUUM FULL t1 ONLINE;
VACUUM
```

----结束

 版权所有 © 华为云计算技术有限公司

17 bucket 分布表

bucket分布表底层采用段页式存储，按照用户指定的不同分布方式对用户数据进行物理切分。方便用户数据上涨，扩容DN分片后，能够通过物理文件搬迁和日志多流追增的方式进行在线扩容。

目前bucket分布表支持：Astore Hash分布表和Astore range分布表；不支持复制表、临时表（本地临时表和全局临时表）、unlogged表、Ustore表。

17.1 hashbucket

步骤1 通过给DATABASE绑定node group生成CN上Hash分布和DN上bucket的映射关系，语法如下：

```
ALTER DATABASE databasename TO GROUP groupname;
```

语法详情请参见《开发指南》中“SQL参考 > SQL语法 > A > ALTER DATABASE”章节中相关描述。

步骤2 创建hashbucket表时，只需要指定with(hashbucket=on)即可：

```
CREATE TABLE table_name(a int, b int) with(hashbucket=on, storage_type=astore);
```

----结束

 版权所有 © 华为云计算技术有限公司