ALIBABA CLOUD

Alibaba Cloud

云数据库RDS

RDS PostgreSQL 数据库

文档版本：20220407

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.概述 12
2.配额及限制 13
3.功能概览 16
3.1. PostgreSQL功能概览 16
3.2. PostgreSQL 14 16
3.3. PostgreSQL 13 18
3.4. PostgreSQL 12 20
3.5. PostgreSQL 11 21
3.6. PostgreSQL 10 23
3.7. PostgreSQL 9.4 25
4.产品规格 28
4.1. RDS PostgreSQL主实例规格列表 28
4.2. RDS PostgreSQL只读实例规格列表 32
5.计费 35
5.1. 按量付费转包年包月 35
5.2. 包年包月转按量付费 35
5.3. 手动续费 36
5.4. 自动续费 37
6.插件 42
6.1. 支持插件列表 42
6.2. 垂直行业 46
6.2.1. 时序数据库（TimescaleDB） 46
6.2.2. 数组相似度计算（smlar） 49
6.2.3. 高效向量检索（PASE） 50
6.2.4. 位图计算（roaringbitmap） 56
6.2.5. 位图功能扩展（varbitx） 61

6.2.6. 化学分子计算检索（RDKit） 63
6.3. 异构数据库访问 64
6.3.1. 读写外部数据文本文件（oss_fdw） 64
6.3.2. 读写MySQL数据（mysql_fdw） 68
6.3.3. 查询日志（log_fdw） 71
6.3.4. 查询SQL Server数据库（tds_fdw） 73
6.3.5. 同步更新Oracle数据库（oracle_fdw） 75
6.4. 跨库操作（dblink、postgres_fdw） 78
6.5. 基数统计（hll） 80
6.6. 定时任务（pg_cron） 82
6.7. 水平拆分（PL/Proxy） 83
6.8. 模糊查询（pg_bigm） 87
6.9. 集成Elasticsearch（ZomboDB） 90
6.10. SQL防火墙（sql_firewall） 91
6.11. 并发控制（pg_concurrency_control） 93
6.12. 高速全文检索（RUM） 95
6.13. 中文分词（zhparser） 100
6.14. 中文分词（pg_jieba） 101
6.15. 字符串相似性判断（fuzzystrmatch） 102
6.16. 定制执行计划（pg_hint_plan） 105
6.17. 清理表空间（pg_repack） 109
6.18. 逻辑复制（pglogical） 110
6.19. 审计日志（pgAudit） 112
6.20. 调试存储过程（pldebugger） 113
6.21. 索引推荐（index_adviser） 115
6.22. 虚拟索引（hypopg） 119
6.23. UUID生成器（sequential-uuids） 120
6.24. 机器学习（MADlib） 121

7.快速入门 123
7.1. 使用流程 123
7.2. 创建RDS PostgreSQL实例 123
7.3. 设置白名单 127
7.3.1. 设置白名单 127
7.3.2. 设置安全组 133
7.3.3. 白名单常见案例和问题 134
7.4. 创建数据库和账号 136
7.5. 连接PostgreSQL实例 138
8.开发运维建议 143
9.一键上云 147
9.1. 应用场景 147
9.2. 上云前准备 148
9.2.1. （可选）ECS配置安全组 148
9.2.2. 配置postgresql.conf文件 149
9.2.3. 创建迁移账号 151
9.2.4. 更新pg_hba.conf文件 151
9.2.5. 配置服务器防火墙 153
9.3. 一键上云 154
9.4. 解读上云评估报告 158
9.5. 云上应用场景 165
9.5.1. RDS PostgreSQL实例跨地域迁移 165
9.5.2. RDS PostgreSQL实例跨账号迁移 171
9.5.3. RDS PostgreSQL实例缩容 177
10.数据迁移 181
10.1. 数据迁移方案概览 181
10.2. 将自建数据库迁移到云数据库 181
10.2.1. 使用pg_dump和pg_restore将自建PostgreSQL数据库迁移到... 181

10.2.2. 自建PostgreSQL迁移至RDS PostgreSQL 183
10.3. 第三方数据库迁移到云数据库 193
10.3.1. 从Amazon RDS for PostgreSQL增量迁移至阿里云 193
10.3.2. 从Amazon RDS for PostgreSQL全量迁移至阿里云 200
10.4. RDS PostgreSQL间的数据迁移 208
10.5. 迁移RDS PostgreSQL数据到本地PostgreSQL 216
11.数据同步 218
11.1. RDS PostgreSQL间的单向同步 218
11.2. 自建PostgreSQL同步至RDS PostgreSQL 225
11.3. RDS PostgreSQL同步至RDS MySQL 235
11.4. 从RDS PostgreSQL同步至云原生数据仓库AnalyticDB PostgreS... 243
12.升级版本 250
12.1. PostgreSQL基础版升级高可用版 250
12.2. 升级数据库大版本 251
12.3. 解读RDS PostgreSQL大版本升级检查报告 256
12.4. 升级内核小版本 260
13.实例 262
13.1. 创建RDS PostgreSQL实例 262
13.2. 变更配置 265
13.3. 重启实例 267
13.4. 自动或手动主备切换 268
13.5. 设置可维护时间段 270
13.6. 迁移可用区 270
13.7. 释放实例 274
13.8. 开启和关闭实例释放保护 275
13.9. 设置实例参数 276
13.10. 设置实例保护级别 277
13.11. 实例回收站 279

14.只读实例 281
14.1. PostgreSQL只读实例简介 281
14.2. 创建PostgreSQL只读实例 282
15.账号 286
15.1. 创建账号 286
15.2. 重置密码 287
15.3. 授权服务账号 288
15.4. 锁定/删除账号 289
15.5. 账号权限详情 290
15.6. RDS PostgreSQL接入自建域 291
16.数据库连接 303
16.1. 连接PostgreSQL实例 303
16.2. 申请或释放外网地址 307
16.3. 通过DMS登录RDS数据库 307
16.4. 查看或修改内外网地址和端口 309
17.网络/VPC/交换机 311
17.1. 切换虚拟交换机 311
17.2. 切换网络类型 312
17.3. 临时混访方案（同时保留经典网络和专有网络地址） 313
18.数据库 316
18.1. 创建数据库 316
18.2. 删除数据库 316
18.3. 修改时区 317
19.SQL翻译 318
20.监控与报警 319
20.1. 查看资源和引擎监控 319
20.2. 设置监控频率 319
20.3. 查看增强监控 320

20.4. 管理报警 338
21.数据安全/加密 342
21.1. 切换为高安全白名单模式 342
21.2. IP白名单与安全组 343
21.2.1. 设置白名单 343
21.2.2. 设置安全组 348
21.2.3. 白名单常见案例和问题 349
21.3. SSL链路加密 351
21.3.1. 快速配置SSL加密 351
21.3.2. 配置自定义证书 355
21.3.3. 配置客户端CA证书 360
21.3.4. SSL连接RDS PostgreSQL数据库 364
21.4. 云盘加密 369
21.5. 全加密云数据库 371
21.5.1. 什么是全加密功能 371
21.5.2. 配置步骤 374
22.日志/历史事件 378
22.1. SQL审计（数据库审计） 378
22.2. 查看日志 379
22.3. 历史事件 380
23.待处理事件 385
24.备份 387
24.1. 备份费用 387
24.2. 备份PostgreSQL数据 387
24.3. 跨地域备份 390
24.4. 查看备份空间免费额度 398
24.5. 下载数据备份和日志备份 399
24.6. 逻辑备份PostgreSQL数据 399

24.7. 实例级别全量备份pg_basebackup 403
25.恢复 406
25.1. 恢复PostgreSQL数据 406
25.2. 跨地域恢复数据 408
25.3. 使用pg_restore恢复逻辑备份文件数据 409
26.性能优化与诊断 413
26.1. RDS PostgreSQL CPU使用率高问题 413
26.2. RDS PostgreSQL慢SQL问题 415
26.3. RDS PostgreSQL I/O高问题 421
26.4. 数据库自治服务DAS简介 426
26.5. 一键诊断 426
26.5.1. 会话管理 426
26.5.2. 实时性能 427
26.5.3. 空间分析 427
26.5.4. 性能洞察 427
26.6. 性能趋势 428
26.7. 慢SQL 428
26.8. SQL洞察和审计 428
27.标签 430
27.1. 创建标签 430
27.2. 删除标签 431
27.3. 根据标签筛选实例 432
28.最佳实践 434
28.1. RDS PostgeSQL权限管理最佳实践 434
28.2. 快速、轻松地将自建PostgreSQL迁移到阿里云RDS PostgreSQL 437
28.3. 设置PostgreSQL排序规则（Collation） 438
28.4. 批量更新、删除或插入数据 443
28.5. 查找最耗费资源的 SQL（Top SQL） 444

28.6. 逻辑订阅 448
28.7. 使用事件触发器实现DDL回收站、防火墙、增量订阅同步 449
28.8. 自动故障转移和读写分离 452
29.应用解决方案 455
29.1. 实时精准营销（人群圈选） 455
29.2. 图像识别、人脸识别、相似特征检索、相似人群圈选 460
29.3. 秒级闪回实时容灾 469
29.4. 基于pgpool实现读写分离 474
29.5. 用户喜好推荐系统（PostgreSQL近似计算应用） 487
29.6. 将RDS PostgreSQL作为RDS MySQL实时分析库 490
29.7. 中间件ShardingSphere开发指南 499

1.概述

本文简单介绍RDS PostgreSQL及相关概念。

阿里云关系型数据库RDS（Relational Database Service）是一种稳定可靠、可弹性伸缩的在线数据库服务。基于阿里云分布式文件系统和SSD盘高性能存储，RDS支持MySQL、SQL Server、PostgreSQL、PPAS（高度兼容 Oracle）和MariaDB引擎，并且提供了容灾、备份、恢复、监控、迁移等方面的全套解决方案，彻底解决数据库运维的烦恼。关于RDS的优势与价值，请参见RDS与自建数据库对比优势。

如果您需要获取人工帮助，可以在RDS管理控制台的右上角选择**工单 > 提交工单**。如果业务复杂，您也可以购买支持计划，获取由IM企业群、技术服务经理（TAM）、服务经理等提供的专属支持。

有关阿里云关系型数据库RDS更多介绍信息，请查看产品详情 。

声明

本文档中描述的部分产品特性或者服务可能不在您的购买或使用范围之内，请以实际商业合同和条款为准。本文档内容仅作为指导使用，文档中的所有内容不构成任何明示或暗示的担保。

RDS PostgreSQL

RDS PostgreSQL的优点主要集中在对SQL规范的完整实现以及丰富多样的数据类型支持，包括JSON数据、IP数据和几何数据等。除了完美支持事务、子查询、多版本控制（MVCC）、数据完整性检查等特性外，RDS PostgreSQL还集成了高可用和备份恢复等重要功能，减轻您的运维压力。除此之外还提供如下高级功能：

- 专属集群MyBase：是由多台主机（底层服务器，如ECS I2服务器、神龙服务器）组成的集群，相对于全托管数据库，可以满足您更多的需求。
- 全加密云数据库：数据在用户侧加密后传入云数据库，能够有效防御来自云平台外部和内部的安全威胁，时刻保护用户数据，让云上数据成为您的私有资产。
- 只读实例：在对数据库有少量写请求，但有大量读请求的应用场景下，单个实例可能无法承受读取压力，甚至对业务产生影响。为了实现读取能力的弹性扩展，分担数据库压力，您可以创建一个或多个只读实例，利用只读实例满足大量的数据库读取需求，增加应用的吞吐量。

RDS PostgreSQL支持的功能请参见PostgreSQL功能概览。

基本概念

- 实例：一个独立占用物理内存的数据库服务进程，用户可以设置不同的内存大小、磁盘空间和数据库类型。其中内存的规格会决定该实例的性能。实例创建后可以变更配置和删除实例。
- 数据库：在一个实例下创建的逻辑单元，一个实例可以创建多个数据库，数据库在实例内的命名唯一。
- 地域和可用区：地域是指物理的数据中心。可用区是指在同一地域内，电力和网络互相独立的物理区域。更多信息请参见阿里云全球基础设施。

通用描述约定

描述	说明
本地数据库	指代部署在本地机房或者非阿里云RDS上的数据库。
RDS XX（XX 为 MySQL、SQL Server、PostgreSQL、PPAS或MariaDB）	指代某一数据库类型的RDS，如RDS MySQL是指在RDS上开通的数据库引擎为MySQL的实例。

2.配额及限制

云数据库PostgreSQL在配额和使用上有一些限制，用来提高实例的稳定性和安全性。本文介绍PostgreSQL配额与使用限制的具体内容。

规格与性能

资源	规格	说明
磁盘空间大小	• SSD本地盘：最大6,000 GB • SSD云盘：最大6,000 GB • ESSD云盘：最大32,000 GB	基于本地SSD盘的实例存储空间大小与实例规格绑定，云盘版实例的存储空间可购买大小不受实例规格限制。具体请参见主实例规格列表。
连接数	最大76,800	不同实例规格的连接数不同，请参见主实例规格列表。
IOPS	• SSD本地盘：最大50,000 • 云盘：请参见关于云盘IOPS	无
内存大小	• SSD本地盘：最大512GB • 云盘：最大768GB	云 盘实例的底层操作系统以及相关管理服务会占用一部分内存，因此实例实际可用的内存不会达到规格显示的内存大小。占用内存详情如下： • 底层操作系统：约500~700 MB。 • RDS相关管理服务：约500 MB。

配额

配额	限制

只读实例	PG 10或以上版本才支持只读实例，且只读实例必须创建在与主实例相同的地域内。
	• SSD本地盘：
	  ○ 主实例最多创建5个只读实例。
	  ○ 规格必须大于8核32 GB（独享套餐），才支持只读实例。
	• 云盘：
	  ○ 主实例最多创建32个只读实例。
	  ○ 只读实例为单节点架构，没有备节点。
	只读实例的更多信息，请参见PostgreSQL只读实例简介。

标签	标签键必须唯一，最大设置20个。每次最多设置50个实例进行批量标签绑定。创建标签，请参见创建标签。

配额	限制
备份空间免费额度	PostgreSQL云盘实例仅支持快照备份，PostgreSQL本地盘实例仅支持物理备份。超出免费额度的部分 = 数据备份量 + 日志备份量 - 免费额度，单位为GB，只入不舍。 • SSD本地盘：物理备份空间的免费额度=50%×实例购买的存储空间。 • 云盘：快照备份空间的免费额度=200%×实例购买的存储空间。 备份的更多信息，请参见备份PostgreSQL数据。
实例备份保留天数	默认为7天，最大730天。
错误日志保留天数	30天。查看错误日志，请参见查看日志。
慢日志明细保留天数	30天。查看慢日志明细，请参见查看日志。

命名限制

限制项	限制说明
实例名	• 长度为2~256个字符。 • 由大小写字母、中文、数字、下划线（_）或短横线（-）组成。 • 以大小字母或中文开头。
用户名	• 长度为2~63个字符。 • 由小写字母、数字或下划线组成。 • 以字母开头，以字母或数字结尾。 • 不能和已有的账号名重复。 • 不能以pg开头。 • 不能使用SQL关键字。具体请参见SQL关键字。
数据库名	• 长度不超过63个字符。 • 由小写字母、数字、下划线（_）或短横线（-）组成。 • 以字母开头，以字母或数字结尾。 • 不能和已有的数据库名重复。 • 不能使用SQL关键字。具体请参见SQL关键字。

安全限制

限制项	限制说明
密码	密码需要满足以下要求： • 长度为8~32个字符。 • 由大写字母、小写字母、数字、特殊字符中的至少三种组成。特殊字符为：`!@#$%^&*()_+-=`。
端口	RDS PostgreSQL实例的默认端口为5432，允许用户手动修改端口号。

限制项	限制说明
实例参数	出于安全和稳定性考虑，部分参数不支持修改。大部分实例参数可以使用控制台或API进行修改，修改参数方法请参见设置实例参数。
云盘加密	云盘加密只能在创建实例时开启且不能关闭。设置云盘加密，请参见云盘加密。
实例可加入安全组数量	最大10个。 • 如果云数据库RDS实例与云服务器处于不同的安全组，云服务器不能访问RDS。 • RDS实例只能添加与自身网络类型相同的安全组，即实例为专有网络VPC时，只能添加VPC类型的安全组；实例为经典网络时，只能添加经典网络类型的安全组。 设置安全组，请参见设置安全组。
实例可添加白名单分组数量	最大50个。添加白名单，请参见设置白名单。
root权限账号	不可创建，RDS无法向用户提供superuser权限。
高权限账号	高权限账号只能通过控制台或API创建和管理。支持多个高权限账号，可以断开任意账号的连接。 创建账号，请参见创建账号。
普通账号	普通账号可以通过控制台、API或者SQL语句创建和管理。需要手动给普通账号授予特定数据库的权限。不能创建和管理其他账号，也不能断开其他账号的连接。 创建账号，请参见创建账号。

SQL使用限制

RDS PostgreSQL实例中SQL的使用限制与官方保持一致，请参见PostgreSQL SQL命令参考和PostgreSQL限制。

其它限制

限制项	限制说明
外网地址	外网地址需要手动申请。申请外网地址，请参见查看或修改内外网地址和端口。
搭建数据库复制	提供主备复制架构（基础版除外），其中的备（slave）实例不对用户开放，用户应用不能直接访问。
重启RDS实例	必须通过控制台或OpenAPI操作重启实例。
创建表空间	不支持。

3.功能概览

3.1. PostgreSQL功能概览

本文介绍RDS PostgreSQL各版本支持的功能，便于您根据自身需要选购实例或查询已购实例功能。

- PostgreSQL 14
- PostgreSQL 13
- PostgreSQL 12
- PostgreSQL 11
- PostgreSQL 10
- PostgreSQL 9.4

3.2. PostgreSQL 14

本文介绍RDS PostgreSQL 14版本支持的功能，便于您根据自身需要选购实例或查询已购实例功能。

类别	功能	高可用版（云盘）	基础版（云盘）
数据迁移同步	数据迁移同步	✔	✔
实例管理	创建实例	✔	✔
	变更配置	✔	✔
	设置实例参数	✔	✔
	设置实例保护级别	✔	✔
	迁移可用区	✔	▯
	切换主备实例	✔	▯
	升级数据库大版本	▯	▯
	重启实例	✔	✔
	设置可维护时间段	✔	✔
	释放实例	✔	✔
	实例回收站	✔	✔
账号管理	创建账号	✔	✔
	重置密码	✔	✔
	锁定账号	✔	✔
	删除账号	✔	✔
	创建数据库	✔	✔

类别	功能	高可用版（云盘）	基础版（云盘）
数据库管理	删除数据库	✔	✔
	插件	✔	✔
数据库连接	连接PostgreSQL实例	✔	✔
	设置连接地址	✔	✔
	查看连接地址/端口	✔	✔
	申请或释放外网地址	✔	✔
监控报警	查看资源和引擎监控	✔	✔
	设置监控频率	✔	✔
	管理报警	✔	✔
网络管理	切换网络类型	▯	▯
	切换虚拟交换机	✔	✔
只读实例	创建只读实例	✔	▯
安全管理	设置白名单	✔	✔
	设置SSL加密	✔	✔
	设置云盘加密	✔	✔
	切换高安全白名单模式	▯	▯
	全加密云数据库	✔	✔
日志/历史事件	日志管理	✔	✔
	历史事件	✔	✔
数据库备份	备份数据	✔	✔
	跨地域备份	✔	✔
	免费额度	✔	✔
	下载备份	▯	▯
数据库恢复	恢复数据	✔	✔
	跨地域恢复	✔	✔
逻辑订阅	逻辑订阅	✔	✔

类别	功能	高可用版（云盘）	基础版（云盘）
标签管理	创建标签	✔	✔
	删除标签	✔	✔
	根据标签筛选实例	✔	✔

3.3. PostgreSQL 13

本文介绍RDS PostgreSQL 13版本支持的功能，便于您根据自身需要选购实例或查询已购实例功能。

类别	功能	高可用版（云盘）	基础版（云盘）
数据迁移同步	数据迁移同步	✔	✔
实例管理	创建实例	✔	✔
	变更配置	✔	✔
	设置实例参数	✔	✔
	设置实例保护级别	✔	▯
	迁移可用区	✔	▯
	切换主备实例	✔	▯
	升级数据库大版本	✔	✔
	重启实例	✔	✔
	设置可维护时间段	✔	✔
	释放实例	✔	✔
	实例回收站	✔	✔
账号管理	创建账号	✔	✔
	重置密码	✔	✔
	锁定账号	✔	✔
	删除账号	✔	✔
数据库管理	创建数据库	✔	✔
	删除数据库	✔	✔
	插件	✔	✔
	连接PostgreSQL实例	✔	✔

类别	功能	高可用版（云盘）	基础版（云盘）
数据库连接	设置连接地址	✔	✔
	查看连接地址/端口	✔	✔
	申请或释放外网地址	✔	✔
监控报警	查看资源和引擎监控	✔	✔
	设置监控频率	✔	✔
	管理报警	✔	✔
网络管理	切换网络类型	▯	▯
	切换虚拟交换机	✔	✔
只读实例	创建只读实例	✔	▯
安全管理	设置白名单	✔	✔
	设置SSL加密	✔	✔
	设置云盘加密	✔	✔
	切换高安全白名单模式	▯	▯
	全加密云数据库	▯	▯
日志/历史事件	日志管理	✔	✔
	历史事件	✔	✔
数据库备份	备份数据	✔	✔
	跨地域备份	✔	✔
	免费额度	✔	✔
	下载备份	▯	▯
数据库恢复	恢复数据	✔	✔
	跨地域恢复	✔	✔
逻辑订阅	逻辑订阅	✔	✔
标签管理	创建标签	✔	✔
	删除标签	✔	✔
	根据标签筛选实例	✔	✔

3.4. PostgreSQL 12

本文介绍RDS PostgreSQL 12版本支持的功能，便于您根据自身需要选购实例或查询已购实例功能。

类别	功能	高可用版（云盘）	基础版（云盘）
数据迁移同步	数据迁移同步	✔	✔
实例管理	创建实例	✔	✔
	变更配置	✔	✔
	设置实例参数	✔	✔
	设置实例保护级别	✔	▯
	迁移可用区	✔	▯
	切换主备实例	✔	▯
	升级数据库大版本	✔	✔
	重启实例	✔	✔
	设置可维护时间段	✔	✔
	释放实例	✔	✔
	实例回收站	✔	✔
账号管理	创建账号	✔	✔
	重置密码	✔	✔
	锁定账号	✔	✔
	删除账号	✔	✔
数据库管理	创建数据库	✔	✔
	删除数据库	✔	✔
	插件	✔	✔
数据库连接	连接PostgreSQL实例	✔	✔
	设置连接地址	✔	✔
	查看连接地址/端口	✔	✔
	申请或释放外网地址	✔	✔
	查看资源和引擎监控	✔	✔

类别	功能	高可用版（云盘）	基础版（云盘）
监控报警	设置监控频率	✔	✔
	管理报警	✔	✔
网络管理	切换网络类型	▯	▯
	切换虚拟交换机	✔	✔
只读实例	创建只读实例	✔	▯
安全管理	设置白名单	✔	✔
	设置SSL加密	✔	✔
	设置云盘加密	✔	✔
	切换高安全白名单模式	▯	▯
	全加密云数据库	▯	▯
日志/历史事件	日志管理	✔	✔
	历史事件	✔	✔
数据库备份	备份数据	✔	✔
	跨地域备份	✔	✔
	免费额度	✔	✔
	下载备份	▯	▯
数据库恢复	恢复数据	✔	✔
	跨地域恢复	✔	✔
逻辑订阅	逻辑订阅	✔	✔
标签管理	创建标签	✔	✔
	删除标签	✔	✔
	根据标签筛选实例	✔	✔

3.5. PostgreSQL 11

本文介绍RDS PostgreSQL 11版本支持的功能，便于您根据自身需要选购实例或查询已购实例功能。

类别	功能	高可用版（云盘）	基础版（云盘）
数据迁移同步	数据迁移同步	✔	✔

类别	功能	高可用版（云盘）	基础版（云盘）
实例管理	创建实例	✔	✔
	变更配置	✔	✔
	设置实例参数	✔	✔
	设置实例保护级别	✔	▯
	迁移可用区	✔	▯
	切换主备实例	✔	▯
	升级数据库大版本	✔	✔
	重启实例	✔	✔
	设置可维护时间段	✔	✔
	释放实例	✔	✔
	实例回收站	✔	✔
账号管理	创建账号	✔	✔
	重置密码	✔	✔
	锁定账号	✔	✔
	删除账号	✔	✔
数据库管理	创建数据库	✔	✔
	删除数据库	✔	✔
	插件	✔	✔
数据库连接	连接PostgreSQL实例	✔	✔
	设置连接地址	✔	✔
	查看连接地址/端口	✔	✔
	申请或释放外网地址	✔	✔
监控报警	查看资源和引擎监控	✔	✔
	设置监控频率	✔	✔
	管理报警	✔	✔
	切换网络类型	▯	▯

网络管理类别	功能	高可用版（云盘）	基础版（云盘）
	切换虚拟交换机	✔	✔
只读实例	创建只读实例	✔	□
安全管理	设置白名单	✔	✔
	设置SSL加密	✔	✔
	设置云盘加密	✔	✔
	切换高安全白名单模式	□	□
	全加密云数据库	✔	✔
日志/历史事件	日志管理	✔	✔
	历史事件	✔	✔
数据库备份	备份数据	✔	✔
	跨地域备份	✔	✔
	免费额度	✔	✔
	下载备份	□	□
数据库恢复	恢复数据	✔	✔
	跨地域恢复	✔	✔
逻辑订阅	逻辑订阅	✔	✔
标签管理	创建标签	✔	✔
	删除标签	✔	✔
	根据标签筛选实例	✔	✔

3.6. PostgreSQL 10

本文介绍RDS PostgreSQL 10版本支持的功能，便于您根据自身需要选购实例或查询已购实例功能。

类别	功能	高可用版（本地盘）	高可用版（云盘）	基础版（云盘）
	创建实例	✔	✔	✔
	变更配置	✔	✔	✔
	设置实例参数	✔	✔	✔

类别	功能	高可用版（本地盘）	高可用版（云盘）	基础版（云盘）
实例管理	设置实例保护级别	□	✔☺	□
	迁移可用区	✔☺	✔☺	□
	切换主备实例	✔☺	✔☺	□
	重启实例	✔☺	✔☺	✔☺
	设置可维护时间段	✔☺	✔☺	✔☺
	升级数据库大版本	✔☺	✔☺	✔☺
	释放实例	✔☺	✔☺	✔☺
	实例回收站	✔☺	✔☺	✔☺
账号管理	创建账号	✔☺	✔☺	✔☺
	重置密码	✔☺	✔☺	✔☺
	锁定账号	✔☺	✔☺	✔☺
	删除账号	✔☺	✔☺	✔☺
数据库管理	创建数据库	✔☺	✔☺	✔☺
	删除数据库	✔☺	✔☺	✔☺
	插件	✔☺	✔☺	✔☺
数据库连接	连接PostgreSQL实例	✔☺	✔☺	✔☺
	设置连接地址	✔☺	✔☺	✔☺
	查看连接地址/端口	✔☺	✔☺	✔☺
	申请或释放外网地址	✔☺	✔☺	✔☺
监控报警	查看资源和引擎监控	✔☺	✔☺	✔☺
	设置监控频率	✔☺	✔☺	✔☺
	管理报警	✔☺	✔☺	✔☺
网络管理	切换网络类型	✔☺	□	□
	切换虚拟交换机	□	✔☺	✔☺
只读实例	创建只读实例	✔☺	✔☺	□
	设置白名单	✔☺	✔☺	✔☺

类别	功能	高可用版（本地盘）	高可用版（云盘）	基础版（云盘）
安全管理	设置SSL加密	▯	✔☺	✔☺
	设置云盘加密	▯	✔☺	✔☺
	切换高安全白名单模式	✔☺	▯	▯
	全加密云数据库	▯	▯	▯
日志/历史事件	日志管理	✔☺	✔☺	✔☺
	历史事件	✔☺	✔☺	✔☺
数据库备份	备份数据	✔☺	✔☺	✔☺
	跨地域备份	✔☺	✔☺	✔☺
	免费额度	✔☺	✔☺	✔☺
	下载备份	✔☺	▯	▯
数据库恢复	恢复数据	✔☺	✔☺	✔☺
	跨地域恢复	✔☺	✔☺	✔☺
逻辑订阅	逻辑订阅	▯	✔☺	✔☺
标签管理	创建标签	✔☺	✔☺	✔☺
	删除标签	✔☺	✔☺	✔☺
	根据标签筛选实例	✔☺	✔☺	✔☺

3.7. PostgreSQL 9.4

本文介绍RDS PostgreSQL 9.4版本支持的功能，便于您根据自身需要选购实例或查询已购实例功能。

停售公告

RDS PostgreSQL 9.4版本目前已经停止售卖，停售详情请参见停售通知。

支持功能

类别	功能	高可用版（本地盘）
	创建实例	✔☺
	变更配置	▯☺
	设置实例参数	✔☺
	设置实例保护级别	▯☺

类别	功能	高可用版（本地盘）
实例管理	迁移可用区	✔
	切换主备实例	✔
	重启实例	✔
	设置可维护时间段	✔
	升级数据库大版本	✔
	释放实例	✔
	实例回收站	✔
账号管理	创建账号	✔
	重置密码	✔
	锁定账号	✔
	删除账号	✔
数据库管理	创建数据库	✔
	删除数据库	✔
	插件	✔
数据库连接	连接PostgreSQL实例	✔
	设置连接地址	✔
	查看连接地址/端口	✔
	申请或释放外网地址	✔
监控报警	查看资源和引擎监控	✔
	设置监控频率	✔
	管理报警	✔
网络管理	切换网络类型	✔
	切换虚拟交换机	▯
只读实例	创建只读实例	▯
	设置白名单	✔
	设置SSL加密	▯

类型	功能	高可用版（本地盘）
安全管理	设置云盘加密	☹
	切换高安全白名单模式	✔☺
	全加密云数据库	✔☺
日志/历史事件	日志管理	✔☺
	历史事件	✔☺
数据库备份	备份数据	✔☺
	跨地域备份	✔☺
	免费额度	✔☺
	下载备份	✔☺
数据库恢复	恢复数据	☹
	跨地域恢复	☹
逻辑订阅	逻辑订阅	☹
标签管理	创建标签	✔☺
	删除标签	✔☺
	根据标签筛选实例	✔☺

4.产品规格

4.1. RDS PostgreSQL主实例规格列表

本文介绍RDS PostgreSQL的主实例规格，帮助您了解RDS PostgreSQL主实例的最新规格信息和历史规格信息，您可以查看本文了解各个规格的具体配置。

> **说明**
> 云 盘实例的底层操作系统以及相关管理服务会占用一部分内存，因此实例实际可用的内存不会达到规格显示的内存大小。占用内存详情如下：
> - 底层操作系统：约500~700 MB。
> - RDS相关管理服务：约500 MB。

RDS PostgreSQL主实例（本地盘）

系列	版本	规格族	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间
高可用版	10、9.4	通用型	rds.pg.s1.small	1核 2GB	无限制	见关于云盘IOPS	20GB-6000GB
			rds.pg.s2.large	2核 4GB			
			rds.pg.s3.large	4核 8GB			
			rds.pg.c1.large	8核 16GB			
			rds.pg.c1.xlarge	8核 32GB			
			rds.pg.c2.xlarge	16核 64GB			
			rds.pg.c2.2xlarge	16核 128GB			
		独享型（高内存）	pg.x8.medium.2	2核 16GB	2500	4500	
			pg.x8.large.2	4核 32GB	5000	9000	
			pg.x8.xlarge.2	8核 64GB	10000	18000	
			pg.x8.2xlarge.2	16核 128GB	20000	36000	
		独占物理机型	rds.pg.st.h43	60核 470GB	12000	50000	

RDS PostgreSQL主实例（云盘）

系列	版本	规格族	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间

系列	版本	规格族	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间
基础版	14、13、12、11、10	通用型	pg.n2.small.1	1核 2GB	200	见关于云盘IOPS	SSD云盘：20GB-6000GB ESSD PL1云盘：20GB-32000GB ESSD PL2云盘：500GB-32000GB ESSD PL3云盘：1500GB-32000GB
			pg.n2.medium.1	2核 4GB	400		
			pg.n4.medium.1	2核 8GB	800		
			pg.n2.large.1	4核 8GB	800		
			pg.n4.large.1	4核 16GB	1600		
			pg.n2.xlarge.1	8核 16GB	1600		
			pg.n4.xlarge.1	8核 32GB	3200		
			pg.n2.2xlarge.1	16核 32GB	3200		
			pg.n4.2xlarge.1	16核 64GB	6400		
			pg.n8.2xlarge.1	16核 128GB	10000		
			pg.n4.4xlarge.1	32核 128GB	12800		
			pg.n8.4xlarge.1	32核 256GB	20000		
			pg.n4.8xlarge.1	64核 256GB	22000		
			pg.n8.8xlarge.1	64核 512GB	48000		
		通用型（已停售）	pg.n2.small.2c	1核 2GB	200		
			pg.n2.medium.2c	2核 4GB	400		
			pg.x2.medium.2c	2核 4GB	400		
			pg.x4.medium.2c	2核 8GB	800		
			pg.x8.medium.2c	2核 16GB	1600		
			pg.x2.large.2c	4核 8GB	800		
			pg.x4.large.2c	4核 16GB	1600		
			pg.x8.large.2c	4核 32GB	3200		
			pg.x2.xlarge.2c	8核 16GB	1600		
			pg.x4.xlarge.2c	8核 32GB	3200		
			pg.x8.xlarge.2c	8核 64GB	6400		
			pg.x2.3large.2c	12核 24GB	2400		

系列	版本	规格族	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间SSD云盘：
高可用版	14、13、12、11、10	独享型	pg.x4.3large.2c	12核 48GB	4800	见关于云盘IOPS	20GB-6000GB ESSD PL1云盘：20GB-32000GB ESSD PL2云盘：500GB-32000GB ESSD PL3云盘：1500GB-32000GB
			pg.x8.3large.2c	12核 96GB	9600		
			pg.x2.2xlarge.2c	16核 32GB	3200		
			pg.x4.2xlarge.2c	16核 64GB	6400		
			pg.x8.2xlarge.2c	16核 128GB	12800		
			pg.x2.3xlarge2c	24核 48GB	4800		
			pg.x4.3xlarge.2c	24核 96GB	9600		
			pg.x8.3xlarge.2c	24核 192GB	19200		
			pg.x2.4xlarge.2c	32核 64GB	6400		
			pg.x4.4xlarge.2c	32核 128GB	12800		
			pg.x8.4xlarge.2c	32核 256GB	25600		
			pg.x2.13large.2c	52核 104GB	10400		
			pg.x4.13large.2c	52核 192GB	19200		
			pg.x8.13large.2c	52核 384GB	38400		
			pg.x2.8xlarge.2c	64核 128GB	12800		
			pg.x4.8xlarge.2c	64核 256GB	25600		
			pg.x8.8xlarge.2c	64核 512GB	51200		
			pg.x2.13xlarge.2c	104核 192GB	19200		
			pg.x4.13xlarge.2c	104核 384GB	38400		
			pg.x8.13xlarge.2c	104核 768GB	76800		

RDS PostgreSQL主实例（新通用型）

RDS PostgreSQL的新通用型规格为您提供更好的弹性能力以及更高的实例性能，增强了实例的创建、变配速度。

说明 目前新通用性规格暂未支持以下功能的使用。

- 升级大版本
- 一键上云
- 云盘加密
- 配置SSL加密
- AD域控
- PostgreSQL基础版升级高可用版
- 只读实例

如果需要购买新通用型规格，您必须满足以下条件：

- 实例版本为：RDS PostgreSQL 10、11、12、13或14。
- 实例存储类型为：ESSD PL1、ESSD PL2或ESSD PL3云盘。

说明 新通用型规格目前开放了海外新加坡地区，中国的杭州、北京、上海、深圳、香港地区。其他可用区正在逐步开放中。如需使用，请前往 控制台购买。

系列	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间
基础版	pg.n2.2c.1m	2核 4GB	400	见关于云盘IOPS	ESSD PL1云盘：20GB-32000GB ESSD PL2云盘：500GB-32000GB ESSD PL3云盘：1500GB-32000GB
	pg.n4.2c.1m	2核 8GB	800		
	pg.n2.4c.1m	4核 8GB	800		
	pg.n4.4c.1m	4核 16GB	1600		
	pg.n4.6c.1m	6核 24GB	2400		
	pg.n4.8c.1m	8核 32GB	3200		
高可用版	pg.n2.2c.2m	2核 4GB	400	见关于云盘IOPS	ESSD PL1云盘：20GB-32000GB ESSD PL2云盘：500GB-32000GB ESSD PL3云盘：1500GB-32000GB
	pg.n4.2c.2m	2核 8GB	800		
	pg.n4.4c.2m	4核 16GB	1600		
	pg.n4.6c.2m	6核 24GB	2400		
	pg.n4.8c.2m	8核 32GB	3200		
	pg.n4.12c.2m	12核 48GB	4800		

历史规格RDS PostgreSQL

以下为RDS PostgreSQL历史规格列表。新申请实例不再提供历史规格，建议您使用最新规格。

规格代码	CPU核数	内存	最大连接数	最大IOPS

规格代码	CPU核数	内存	最大连接数	最大IOPS
rds.pg.t1.small	1	1GB	100	600
pg.x8.4xlarge.2	32	256GB	20000	50000
pg.n1.micro.1	1	1GB	100	见关于云盘IOPS
pg.gn5i-c2g1.large.1	2	8GB	800	
pg.gn5i-c4g1.xlarge.1	4	16GB	1600	
pg.gn5i-c8g1.2xlarge.1	8	32GB	3200	
pg.gn5i-c16g1.4xlarge.1	16	64GB	6400	
pg.gn5i-c16g1.8xlarge.1	32	128GB	12800	
pg.gn5i-c28g1.14xlarge.1	56	224GB	22000	

4.2. RDS PostgreSQL只读实例规格列表

本文介绍RDS PostgreSQL的只读实例规格，帮助您了解RDS PostgreSQL只读实例的最新规格信息，您可以查看本文了解各个规格的具体配置。

> **说明** RDS PostgreSQL只读实例有包年包月和按量付费两种计费方式，详情请参见 只读实例价格，具体请以实际售卖页价格为准。

RDS PostgreSQL只读实例（本地盘）

类型	版本	规格族	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间
只读实例	10	独享型（高内存）	pg.x8.xlarge.2	8核 64GB	10000	18000	20GB-6000GB
			pg.x8.2xlarge.2	16核 128GB	12000	36000	
		独享型（高CPU）	pg.x4.xlarge.2	8核 32GB	5000	9000	
			pg.x4.2xlarge.2	16核 64GB	10000	18000	
			pg.x4.4xlarge.2	32核 128GB	12000	36000	
		独占物理机型	rds.pg.st.h43	60核 470GB	4000	50000	

RDS PostgreSQL只读实例（云盘）

类型	版本	规格族	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间
只读实例	14、13、12、11、10	通用型（已停售）	pgro.n2.small.1c	1核 2GB	200	见关于云盘IOPS	50GB-32000GB
			pgro.n2.medium.1c	2核 4GB	400		
		独享型	pgro.x2.medium.1c	2核 4GB	400		
			pgro.x2.large.1c	4核 8GB	800		
			pgro.x2.xlarge.1c	8核 16GB	1600		
			pgro.x2.3large.1c	12核 24GB	2400		
			pgro.x2.2xlarge.1c	16核 32GB	3200		
			pgro.x2.3xlarge.1c	24核 48GB	4800		
			pgro.x2.4xlarge.1c	32核 64GB	6400		
			pgro.x2.13large.1c	52核 104GB	10400		
			pgro.x2.8xlarge.1c	64核 128GB	12800		
			pgro.x2.13xlarge.1c	104核 192GB	19200		
			pgro.x4.medium.1c	2核 8GB	800		
			pgro.x4.large.1c	4核 16GB	1600		
			pgro.x4.xlarge.1c	8核 32GB	3200		
			pgro.x4.3large.1c	12核 48GB	4800		
			pgro.x4.2xlarge.1c	16核 64GB	6400		
			pgro.x4.3xlarge.1c	24核 96GB	9600		
			pgro.x4.4xlarge.1c	32核 128GB	12800		
			pgro.x4.13large.1c	52核 192GB	19200		
			pgro.x4.8xlarge.1c	64核 256GB	25600		
			pgro.x4.13xlarge.1c	104核 384GB	38400		
			pgro.x8.medium.1c	2核 16GB	1600		
			pgro.x8.large.1c	4核 32GB	3200		
			pgro.x8.xlarge.1c	8核 64GB	6400		

类型	版本	规格族	规格代码	CPU和内存	最大连接数	最大IOPS	存储空间
			pgro.x8.3large.1c	12核 96GB	9600		
			pgro.x8.2xlarge.1c	16核 128GB	12800		
			pgro.x8.3xlarge.1c	24核 192GB	19200		
			pgro.x8.4xlarge.1c	32核 256GB	25600		
			pgro.x8.13large.1c	52核 384GB	38400		
			pgro.x8.8xlarge.1c	64核 512GB	51200		
			pgro.x8.13xlarge.1c	104核 768GB	76800		

5.计费

5.1. 按量付费转包年包月

您可以根据需求将后付费（按量付费）的RDS实例转变为预付费（包年包月）的计费方式。

前提条件

- 实例规格不能为历史规格（不再售卖的规格）。历史规格列表请参见历史规格。若需将历史规格实例转变为包年包月，请先变更实例规格。具体操作请参见变更配置。
- 实例的计费类型为按量付费。
- 实例状态为运行中。
- 实例没有未完成支付的转包年包月订单。

> **说明** 云盘只读实例暂不支持包年包月。

影响

本操作对实例的运行不会有任何影响。

注意事项

- 如果某实例有未完成支付的转包年包月订单，此时您升级该实例的规格，则该转包年包月订单将失效。您需要先在订单管理页面将其作废，然后重新进行转包年包月的操作。

操作步骤

1. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
2. 找到目标实例，通过以下两种方式均可进入**按量付费转包年包月**页面：
   - 在目标实例右侧**付费类型**列（新版控制台）或**操作**列（旧版控制台）中单击**转包年包月**。
   - 单击实例的ID，然后在**运行状态**区域单击**转包年包月**。
3. 选择**购买时长**并勾选服务协议。
4. 单击**去支付**。

   > **说明** 此时系统会生成一个转包年包月的订单。若该订单未支付或作废，将导致您无法进行新购实例或转包年包月的操作。您可以在 订单管理页面支付或作废该订单。

5. 按照提示支付订单。

相关API

API	描述
变更计费方式	变更RDS实例的计费方式。

5.2. 包年包月转按量付费

您可以根据业务需求，将RDS实例的计费方式由预付费（包年包月）转变为后付费（按量付费）。

前提条件

- 实例的计费方式为包年包月。计费方式详情请参见收费项、计费方式与价格。
- 实例状态为运行中。

> **说明** 如果您的包年包月实例因到期而锁定，您必须先对实例续费。如何续费请参见 手动续费。

- 实例规格不是历史规格。历史规格实例请变更规格之后再转按量付费。

费用

包年包月实例转为按量付费后，会将余款按之前的支付方式原路退回给您。

余款 = 订单实付金额 - 资源已使用金额

- 订单实付金额：仅指以现金方式支付的订单款项，不包含您通过代金券、优惠券抵扣的部分。
- 资源已使用金额：包月天价（原订单月价/30换算成天价）x 实际消费时长 x 已使用时长对应订单的折扣

> **说明** 实际消费时长精确到天，不满一天按一天计算。

影响

本操作对实例的运行不会有任何影响。

> **说明** 转换前请慎重考虑，如果您长期使用实例，建议维持包年包月的计费方式，价格比按量付费更实惠，且购买时长越长，折扣越多。

操作步 骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**基本信息**页面的**运行状态**区域单击**转按量付费**。

3. 确认实例信息后勾选服务协议，单击**去支付**完成支付即可。

相关API

API	描述
变更计费方式	变更RDS实例的计费方式。

5.3. 手动续费

包年包月实例有到期时间，如果到期未续费，会导致业务中断甚至数据丢失，建议您及时手动续费。
包年包月实例到期后的具体影响请参见欠费或到期的影响。

> **说明** 按量付费实例没有到期时间，不涉及续费操作。

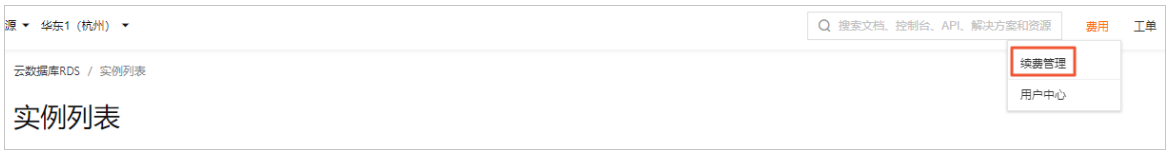
在包年包月实例未到期时或者到期后15天内，您可以手动给实例续费，以延长实例的使用时间。

方法一：RDS控制台续费

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**运行状态**区域右侧单击**续费**。
3. 在**续费**页面中，选择**续费时长**。时间越长，折扣越多。
4. 勾选服务协议，单击**去支付**，完成支付即可。

方法二：续费管理控制台续费

1. 登录RDS管理控制台。
2. 在控制台右上方，选择**费用 > 续费管理**。



3. 通过搜索过滤功能在**手动续费**页签中找到目标实例，您可以单个续费或批量续费：
   - 单个续费
     a. 在目标实例右侧单击**续费**。

     > **说明** 如果目标实例在 **自动续费**或 **到期不续费**页签中，您可以选中目标实例，并在页面下方单击 **恢复为手动续费**，在弹出的对话框中单击 **确认**即可恢复为手动续费。

     b. 选择续费时长，单击**去支付**完成支付即可。
   - 批量续费
     a. 选中目标实例，并在页面下方单击**批量续费**。
     b. 选择每个实例的**续费时长**，单击**去支付**完成支付即可。

自动续费

开通自动续费可以免去您定期手动续费的烦恼，且不会因为忘记续费而导致业务中断。详情请参见自动续费。

5.4. 自动续费

开通自动续费可以免去您定期手动续费的烦恼，且不会因忘记续费而导致业务中断。
包年包月实例有到期时间，如果到期未续费，会导致业务中断甚至数据丢失，具体影响请参见欠费或到期的影响。

> **说明** 按量付费实例没有到期时间，不涉及续费操作。

注意事项

- 自动续费：首次扣款时间为实例到期前第3天的08:00。若未扣费成功，系统将每天扣款一次，持续至实例到期日，直到扣款成功。

  > **说明** 请保证阿里云账号的余额充足，避免续费失败。如果3次自动扣款均失败，请您及时手动续费，避免停机进而导致数据丢失。

- 若您在自动扣款日期前进行了手动续费，则系统将在下一次到期前进行自动续费。
- 自动续费功能于次日生效。若您的实例将于次日到期，为避免业务中断，请手动进行续费，详细步骤请参见手动续费。

在购买实例时开通续费

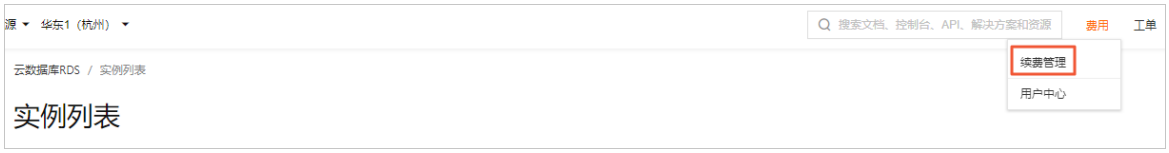
> **说明** 在购买实例时开通自动续费，如果是按月购买，则自动续费周期为1个月；如果是按年购买，则自动续费周期为1年。例如购买6个月实例并勾选自动续费，则快到期时会自动续费1个月。

在购买包年包月实例时，可以勾选**到期自动续费**。

购买实例后开通自动续费

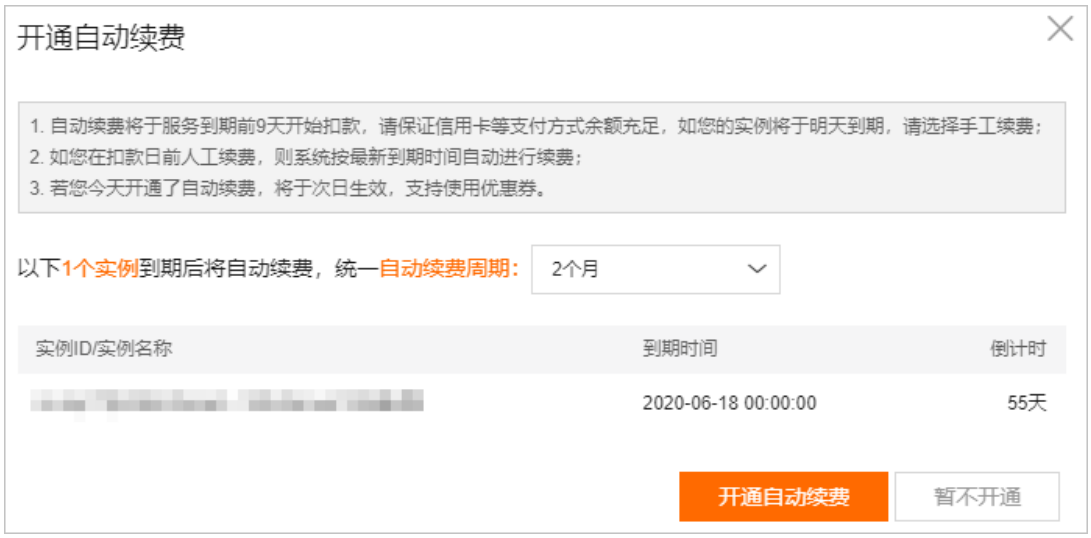
> **说明** 开通自动续费后，系统将根据您选择的续费周期进行自动续费。例如，如果您选择了3个月的续费周期，则每次自动续费时会缴纳3个月的费用。

1. 登录RDS管理控制台。
2. 在控制台右上方，选择**费用 > 续费管理**。

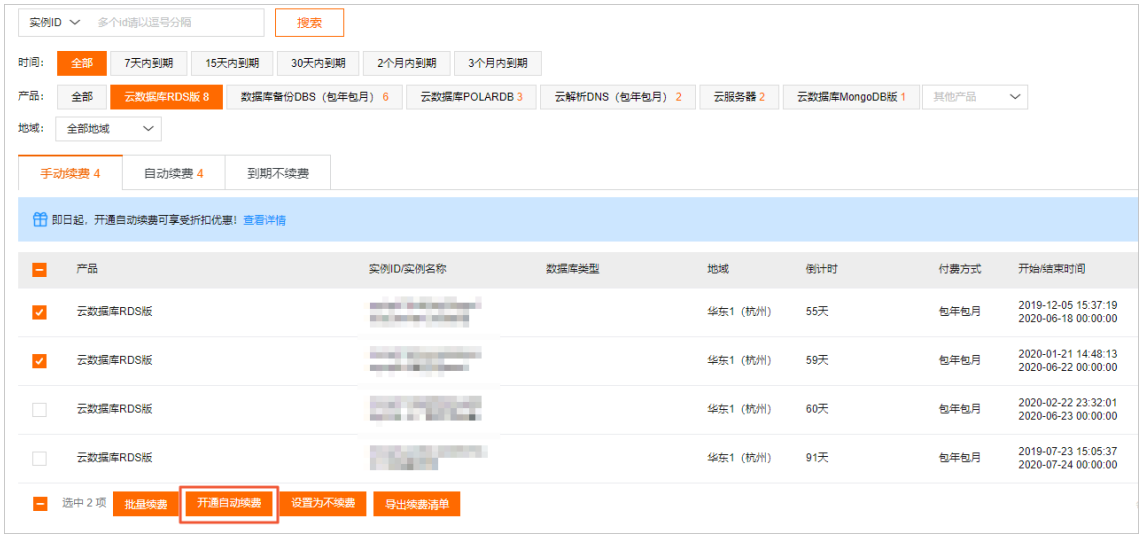


3. 通过搜索过滤功能在**手动续费**或**到期不续费**页签中找到目标实例，您可以单个开通或批量开通：
   - 单个开通
     a. 单击右侧**开通自动续费**。

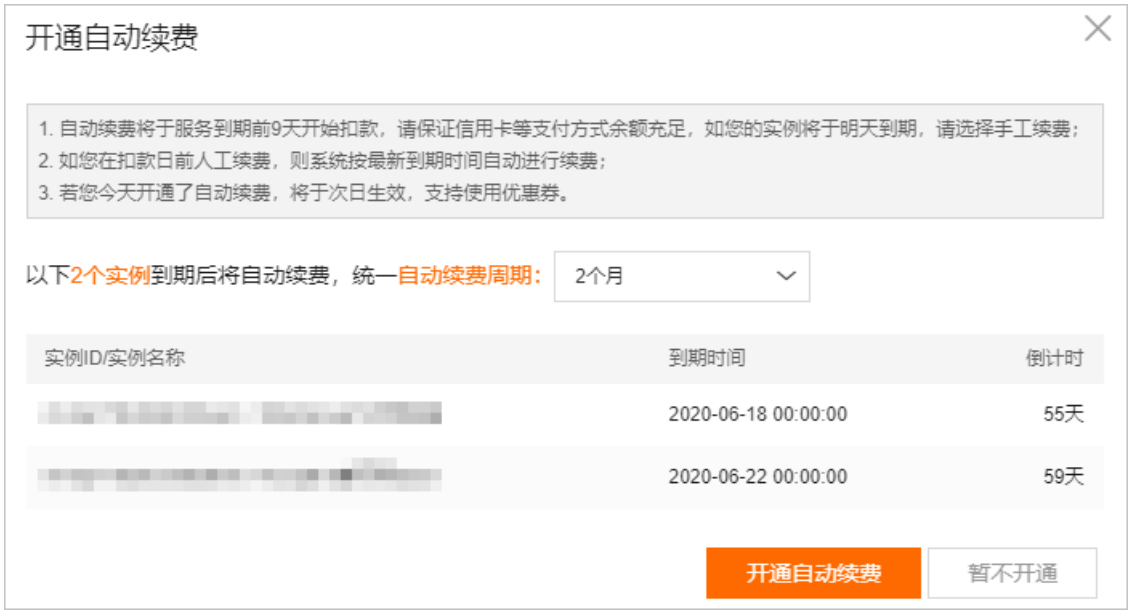
b. 在弹出的对话框中，选择**自动续费周期**，单击**开通自动续费**。



- 批量开通
  勾选目标实例，单击下方**开通自动续费**。

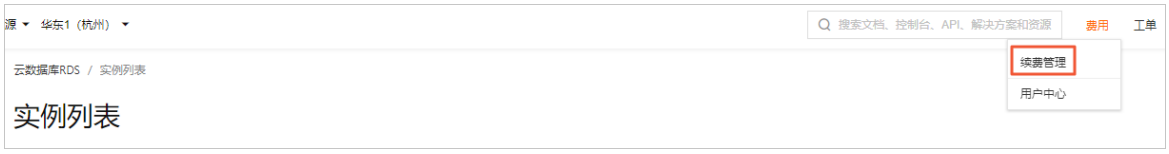


- 在弹出的对话框中，选择 **自动续费周期**，单击 **开通自动续费**。



修改自动续费周期

1. 登录RDS管理控制台。
2. 在控制台右上方，选择**费用 > 续费管理**。

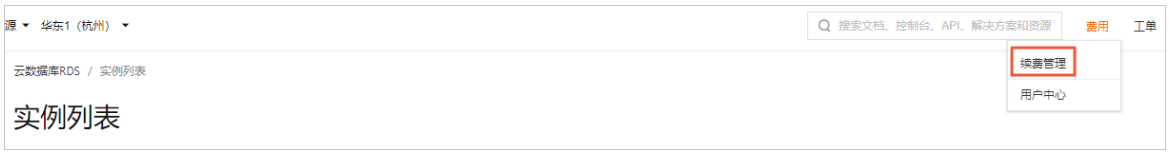


3. 通过搜索过滤功能在**自动续费**页签中找到目标实例，单击右侧**修改自动续费**。

4. 在弹出的对话框中，修改自动续费周期后，单击**确定**。

关闭自动续费

1. 登录RDS管理控制台。
2. 在控制台右上方，选择**费用 > 续费管理**。



3. 通过搜索过滤功能在**自动续费**页签中找到目标实例，单击右侧**恢复手动续费**。

4. 在弹出的对话框中，单击**确定**。

相关API

API	描述
创建RDS实例	创建数据库实例。 **说明** 创建实例时开通自动续费。
手动续费	包年包月实例续费。 **说明** 创建实例后开通自动续费。

6.插件

6.1. 支持插件列表

本文列出RDS PostgreSQL各版本支持的插件及其版本。

> **说明**
> - 如果您的实例实际支持的功能和下表不符，请先升级内核小版本至最新版。
> - 如果您对插件有需求或建议，请提交工单。

插件名	13	12	11	10	9.4	描述
timescaledb	2.5.0	1.7.1	1.7.1	无	无	支持时序数据的自动分片、高效写入、检索、准实时聚合等。
smlar	1.0	1.0	1.0	1.0	1.0	计算两个相同类型数组的相似度。
pase	无	无	0.0.1	无	无	高效向量检索。
roaringbitmap	无	0.5.0	无	无	无	使用位图功能，提高查询性能。
varbitx	无	无	1.0	1.0	1.0	支持多种BIT操作。
rdkit	无	3.8	无	无	3.4	支持化学分子计算、化学分子检索等功能。
oss_fdw	1.1	1.1	1.1	1.1	1.1	读写OSS里的数据。
mysql_fdw	1.1	1.1	1.1	1.1	无	读写RDS MySQL实例或自建MySQL数据库里的数据。
log_fdw	无	无	1.0	无	无	查询数据库日志。
tds_fdw	无	2.0.1	2.0.1	无	无	查询其他类型数据库的数据。
oracle_fdw	无	1.1	无	无	无	支持操作PostgreSQL表同步更新Oracle数据库中的表。
dblink	1.2	1.2	1.2	1.2	1.1	跨库操作表。
postgres_fdw	1.0	1.0	1.0	1.0	1.0	跨库操作表。
hll	2.15	2.14	2.14	无	无	快速预估PV、UV等业务指标。
pg_cron	无	1.1	1.1	1.1	无	设置定时任务。
plproxy	2.10.0	2.9.0	2.9.0	2.8.0	无	包含CLUSTER模式和CONNECT模式，可以帮助您用不同方式访问数据库。

插件名	13	12	11	10	9.4	描述
pg_bigm	1.2	1.2	1.2	1.2	无	创建一个二元语法（2-gram）的GIN索引来加速全文搜索过程。
wal2json	2.3	2.3	2.2	无	无	将逻辑日志文件输出为JSON格式。
zombodb	无	无	4.0	无	无	强大的文本索引和分析功能。
sql_firewall	无	0.8	0.8	0.8	无	数据库层面的防火墙功能，可以防止恶意SQL注入。
pg_concurrency_control	无	无	1.0	1.0	1.0	用于对SQL进行并发控制。
rum	1.3.8	1.3.8	1.3.8	1.3.8	无	高速全文检索。
zhparser	2.1	2.1	1.0	1.0	1.0	中文全文搜索。
pg_jieba	1.1.0	1.1.0	1.1.0	1.1.0	无	中文全文搜索。
fuzzystrmatch	1.1	1.1	1.1	1.1	1.0	判断字符串之间的相似性和距离。
pg_hint_plan	1.3.7	1.3.7	1.3.7	1.3.0	1.1.3	通过特殊的注释语句提示，使PostgreSQL改变其既定的执行计划。
pg_repack	1.4.6	1.4.6	1.4.6	1.4.6	无	在线清理表空间功能。
pglogical	2.4.0	2.4.0	2.4.0	2.4.0	无	提供逻辑流复制发布和订阅的功能。
pgaudit	1.5.0	1.4.1	1.3.2	1.2.2	无	提供详细的会话和对象审计日志记录。
encdb	1.1.7	1.1.7	1.1.7	1.1.7	无	提供全加密云数据库功能。
pldebugger	1.1	1.1	1.1	1.1	无	存储过程调试插件。
address_standardizer	2.5.4	2.5.4	2.5.4	2.5.4	无	基于PAGC标准的地名标准化插件。
address_standardizer_data_us	2.5.4	2.5.4	2.5.4	2.5.4	无	基于PAGC标准的地名标准化（美国）数据插件。
aggs_for_arrays	无	无	无	1.3.1	无	提供计算数字数组的统计信息的扩展函数。
bloom	1.0	1.0	1.0	1.0	无	提供一种基于布鲁姆过滤器的索引访问方法。

插件名	13	12	11	10	9.4	描述
btree_gin	1.3	1.3	1.3	1.2	1.0	提供一个为多种数据类型和所有enum类型实现B树等价行为的GIN操作符类示例。
btree_gist	1.5	1.5	1.5	1.5	1.0	提供一个为多种数据类型和所有enum类型实现B树等价行为的GiST操作符类示例。
chkpass	无	无	无	1.0	1.0	实现了一种数据类型，用来存储加密的口令。
citext	1.6	1.6	1.5	1.4	1.0	提供一种大小写不敏感的字符串类型。
cube	1.4	1.4	1.4	1.2	1.0	提供一种数据类型来表示多维立方体。
decoderbufs	0.1.0	0.1.0	无	无	无	根据Protocol Buffers协议、输出能够适配Debezium平台的数据。
dict_int	1.0	1.0	1.0	1.0	1.0	附加全文搜索词典模板的示例。
earthdistance	1.1	1.1	1.1	1.1	1.0	提供两种不同的方法来计算地球表面的大圆距离。
ganos_address_standardizer	4.4	4.4	4.4	4.4	无	基于PAGC标准的地名标准化插件。
ganos_address_standardizer_data_us	4.4	4.4	4.4	4.4	无	基于PAGC标准的地名标准化美国部分数据插件。
ganos_geometry	4.4	4.4	4.4	4.4	无	提供空间几何类型的计算分析功能。
ganos_geometry_sfcgal	4.4	4.4	4.4	4.4	无	提供空间几何类型的计算分析功能。
ganos_geometry_topology	4.4	4.4	4.4	4.4	无	提供空间几何sfcgal插件扩展功能。
ganos_networking	4.4	4.4	4.4	4.4	无	提供空间几何网络类型的计算分析功能。
ganos_pointcloud	4.4	4.4	4.4	4.4	无	提供点云的存储计算分析功能。
ganos_pointcloud_geometry	4.4	4.4	4.4	4.4	无	提供点云的存储计算分析功能。
ganos_raster	4.4	4.4	4.4	4.4	无	提供空间栅格数据的存储计算分析功能。
ganos_spatialref	4.4	4.4	4.4	4.4	无	提供空间参考的计算分析功能。
ganos_tiger_geocoder	4.4	4.4	4.4	4.4	无	提供USCB Tiger数据类型支持。
ganos_trajectory	4.4	4.4	4.4	4.4	无	ganos移动对象（MOD）数据计算分析功能。

插件名	13	12	11	10	9.4	描述
hstore	1.7	1.6	1.5	1.4	1.3	在单一PostgreSQL值中存储键值对。
imgsmlr	无	无	无	无	1.0	提供相似图片搜索功能。
intagg	1.1	1.1	1.1	1.1	1.0	提供一个整数聚集器和一个枚举器。
intarray	1.3	1.2	1.2	1.2	1.0	提供一些有用的函数和操作符来操纵不含空值的整数数组。
isn	1.2	1.2	1.2	1.1	1.0	按照一个硬编码的前缀列表对输入进行验证，也被用来在输出时连接号码。
ltree	1.2	1.1	1.1	1.1	1.0	用于表示存储在一个层次树状结构中的数据的标签。
jsonbx	无	无	无	无	1.0	PostgreSQL在9.4版本的Jsonb函数扩展。
orafce	无	无	3.8	3.6	3.6	提供一系列Oracle兼容函数。
pg_buffercache	1.3	1.3	1.3	1.3	1.0	提供一种方法实时检查共享缓冲区。
pg_freespacemap	1.2	1.2	1.2	1.2	1.0	检查空闲空间映射（FSM）。
pg_pathman	1.5	1.5.12	1.5.12	1.5	无	高性能分区表插件。
pg_prewarm	1.2	1.2	1.2	1.1	1.0	提供一种方便的方法把数据载入到操作系统缓冲区或者PostgreSQL缓冲区。
pg_sphere	无	无	无	1	1	提供PostgreSQL的球面的数据类型、函数、运算符和索引。
pg_stat_statements	1.8	1.7	1.6	1.6	1.2	提供一种方法追踪服务器执行的所有SQL语句的执行统计信息。
pg_trgm	1.5	1.4	1.4	1.3	1.1	提供字母数字文本相似度的函数和操作符，以及支持快速搜索相似字符串的索引操作符类。
pgcrypto	1.3	1.3	1.3	1.3	1.1	为PostgreSQL提供了密码函数。
pgrouting	无	无	2.6.2	2.6.2	2.0.0	提供空间几何网络的计算分析功能。
pgrowlocks	1.2	1.2	1.2	1.2	1.1	提供一个函数来显示一个指定表的行锁定信息。
pgstattuple	1.5	1.5	1.5	1.5	1.2	提供多种函数来获得元组层的统计信息。
plperl	1.0	1.0	1.0	1.0	1.0	提供perl过程语言。
plpgsql	1.0	1.0	1.0	1.0	1.0	提供SQL过程语言。

插件名	13	12	11	10	9.4	描述
pltcl	无	无	无	无	1.0	提供tcl过程语言。
plv8	2.3.15	2.3.15	2.3.15	2.3.15	1.4.2	可信JavaScript语言扩展。
postgis	3.1.4	3.1.4	3.1.4	3.1.4	2.5.4	PostGIS空间地理信息相关扩展。
postgis_sfcgal	3.1.4	3.1.4	3.1.4	3.1.4	无	PostGIS SFCGAL空间地理信息相关扩展。
postgis_tiger_geocoder	3.1.4	3.1.4	3.1.4	3.1.4	2.5.4	PostGIS TIGER数据空间地理信息相关扩展。
postgis_topology	3.1.4	3.1.4	3.1.4	3.1.4	2.5.4	PostGIS拓扑数据空间地理信息相关扩展。
q3c	无	无	无	1.5.0	1.5.0	用于在球体上进行空间索引。
sslinfo	1.2	1.2	1.2	1.2	1.0	提供当前客户端提供的 SSL 证书的有关信息。
tablefunc	1.0	1.0	1.0	1.0	1.0	包括多个返回表的函数。
tsearch2	无	无	无	无	1.0	为使用旧版本tsearch2的应用提供了向后兼容的文本搜索功能。
tsm_system_rows	1.0	1.0	1.0	无	无	提供表采样方法SYSTEM_ROWS。
tsm_system_time	1.0	1.0	1.0	无	无	提供了表采样方法SYSTEM_TIME。
unaccent	1.1	1.1	1.1	1.1	1.0	文本搜索字典，它能从词位中移除重音（附加符号）。
uuid-ossp	1.1	无	无	1.1	1.0	提供函数使用几种标准算法之一产生通用唯一标识符（UUID）。
xml2	1.1	1.1	1.1	1.1	1.0	提供XPath查询和XSLT功能。

6.2. 垂直行业

6.2.1. 时序数据库（TimescaleDB）

RDS PostgreSQL实例新增TimescaleDB插件版本，支持时序数据的自动分片、高效写入、检索、准实时聚合等。

目前RDS PostgreSQL对外开放的是Open Source版本的TimescaleDB，由于License等问题，可能暂不支持一些高级特性，详情参见TimescaleDB。

前提条件

- 实例版本为PostgreSQL 11、12、13。

> **说明** 如果实例版本为PostgreSQL 13，需要内核小版本大于等于20211130，如需升级内核小版本，请参见 升级内核小版本。

- 使用该插件前，需要将timescaledb加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为**shared_preload_libraries**参数添加**timescaledb**。具体操作，请参见设置实例参数。

> **说明** 部分存量用户可能已经创建过TimescaleDB插件，如果升级内核小版本后，出现如下类似提示：
>
> ```
> ERROR: could not access file "$libdir/timescaledb-1.3.0": No such file or directory
> ```
>
> 请在对应数据库执行如下SQL更新插件：
>
> ```
> alter extension timescaledb update;
> ```

添加TimescaleDB插件

使用pgAdmin客户端连接实例，添加TimescaleDB，命令如下：

```
CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;
```

创建时序表

1. 创建标准表conditions，示例如下：

   ```
   CREATE TABLE conditions ( time TIMESTAMPTZ NOT NULL, location TEXT NOT NULL, temperatur
   e DOUBLE PRECISION NULL, humidity DOUBLE PRECISION NULL );
   ```

2. 创建时序表，示例如下：

   ```
   SELECT create_hypertable('conditions', 'time');
   ```

> **说明** 详细命令说明请参见 Create a Hypertable。

高效写入

您可以使用标准SQL命令将数据插入超表（Hypertables），示例如下：

```
INSERT INTO conditions(time, location, temperature, humidity) VALUES (NOW(), 'office', 70.0
, 50.0);
```

您还可以一次将多行数据插入到超表中，示例如下：

```
INSERT INTO conditions VALUES (NOW(), 'office', 70.0, 50.0), (NOW(), 'basement', 66.5, 60.0
), (NOW(), 'garage', 77.0, 65.2);
```

检索

您可以使用高级SQL查询检索数据，示例如下：

```
--过去3小时内，每15分钟采集一次数据，按时间和温度排序。 SELECT time_bucket('15 minutes', time) AS fifteen_min, location, COUNT(*), MAX(temperature) AS max_temp, MAX(humidity) AS max_hum FROM conditions WHERE time > NOW() - interval '3 hours' GROUP BY fifteen_min, location ORDER BY fifteen_min DESC, max_temp DESC;
```

```
SELECT time_bucket('15 minutes', time) AS fifteen_min,
    location, COUNT(*),
    MAX(temperature) AS max_temp,
    MAX(humidity) AS max_hum
  FROM conditions
  WHERE time > NOW() - interval '3 hours'
  GROUP BY fifteen_min, location
  ORDER BY fifteen_min DESC, max_temp DESC;
```

数据输出 解释 消息 Query History

	fifteen_min timestamp with time zone	location text	count bigint	max_temp double precision	max_hum double precision
1	2019-05-15 14:45:00+08	garage	2	77	65.2
2	2019-05-15 14:45:00+08	office	6	70.1	50.1
3	2019-05-15 14:45:00+08	basement	2	66.5	60

您也可以使用固有的函数进行分析查询，示例如下：

```
--均值查询（Median） SELECT percentile_cont(0.5) WITHIN GROUP (ORDER BY temperature) FROM conditions;
```

```
SELECT percentile_cont(0.5)
  WITHIN GROUP (ORDER BY temperature)
  FROM conditions;
```

数据输出 解释 消息 Query History

	percentile_cont double precision
1	70.05

```
--移动平均数（Moving Average） SELECT time, AVG(temperature) OVER(ORDER BY time ROWS BETWEEN 9 PRECEDING AND CURRENT ROW) AS smooth_temp FROM conditions WHERE location = 'garage' and time > NOW() - interval '1 day' ORDER BY time DESC;
```

```
SELECT time, AVG(temperature) OVER(ORDER BY time
       ROWS BETWEEN 9 PRECEDING AND CURRENT ROW)
     AS smooth_temp
   FROM conditions
   WHERE location = 'garage' and time > NOW() - interval '1 day'
   ORDER BY time DESC;
```

数据输出 解释 消息 Query History

	time timestamp with time zone	smooth_temp double precision
1	2019-05-15 14:52:30.203791+08	77
2	2019-05-15 14:52:27.426082+08	77

6.2.2. 数组相似度计算（smlar）

smlar插件可以用来计算两个相同类型数组的相似度。

前提条件

实例版本如下：

- PostgreSQL 14
- PostgreSQL 13
- PostgreSQL 12（内核小版本20200421及以上）
- PostgreSQL 11（内核小版本20200402及以上）

背景信息

smlar插件提供多种函数计算两个相同类型数组的相似度，同时提供参数来控制相似度计算方法，目前支持所有内置的数据类型。

基本函数介绍

- float4 smlar(anyarray, anyarray)
  计算两个相同数据类型数组的相似度。
- float4 smlar(anyarray, anyarray, bool useIntersect)
  计算两个自定义复合类型（元素，权重）数组的相似度，复合类型如下：

  ```
  CREATE TYPE type_name AS (element_name anytype, weight_name FLOAT4);
  ```

  useIntersect为true时，计算过程只包含重叠元素的部分；为false时计算过程包含所有元素。
- float4 smlar(anyarray a, anyarray b, text formula)
  计算两个相同数据类型数组的相似度，数组通过formula指定。
  formula的预定义变量说明如下：
  - N.i：两个数组的共有元素的个数。
  - N.a：数组a中不重复元素的个数。
  - N.b：数组b中不重复元素的个数。
- float4 set_smlar_limit(float4)
  设置参数smlar.threshold的值。
- float4 show_smlar_limit()
  查看参数smlar.threshold的值。

- anyarray % anyarray
  如果两个数组的相似度大于参数smlar.threshold的值，返回true，否则返回false。
- text[] tsvector2textarray(tsvector)
  转化tsvector类型为text。
- anyarray array_unique(anyarray)
  对数组中的元素排序，排序结果不包含重复元素。
- float4 inarray(anyarray, anyelement)
  如果anyelement存在于anyarray中，返回1，否则返回0。
- float4 inarray(anyarray, anyelement, float4, float4)
  如果anyelement存在于anyarray中，返回第3个参数，否则返回第4个参数。

相关参数说明和支持的数据类型请参见smlar。

使用插件

- 连接实例后创建smlar插件，命令如下：

```
testdb=> create extension smlar;
```

- 验证插件基本功能，示例如下：

```
testdb=> SELECT smlar('{1,4,6}'::int[], '{5,4,6}' ); smlar ---------- 0.666667 (1 row) te
stdb=> SELECT smlar('{1,4,6}'::int[], '{5,4,6}', 'N.i / sqrt(N.a * N.b)' ); smlar -------
--- 0.666667 (1 row)
```

- 卸载插件，命令如下：

```
testdb=> drop extension smlar;
```

6.2.3. 高效向量检索（PASE）

本文介绍RDS PostgreSQL如何通过PASE插件（基于IVFFlat或HNSW算法）实现高效向量检索。

前提条件

实例为RDS PostgreSQL 11。

背景信息

近年来，深度学习领域内的表示学习技术，作为人工智能的代表性技术，取得了长足性进展，在工业界中已经被大量应用，例如广告投放、人脸支付、图像识别、语音识别等场景。数据被嵌入至高维度向量，然后通过向量检索技术来查找相关的项目。

PASE（PostgreSQL ANN search extension）是一款为PostgreSQL数据库研发的高性能向量检索索引插件，使用业界中成熟稳定且高效的ANN（Approximate nearest neighbor）检索算法，包括IVFFlat和HNSW算法，通过这两种算法，可以在PG数据库中实现极高速向量查询。PASE暂时不支持特征向量的抽取与产出，您需要自己检索实体的特征向量，PASE负责的工作是根据已产出的海量级别的向量进行相似向量的检索。

目标读者

限于篇幅，本文不会对机器学习领域的相关名词做详细解释，所以阅读本文需要您有机器学习、搜索推荐相关知识基础。

注意事项

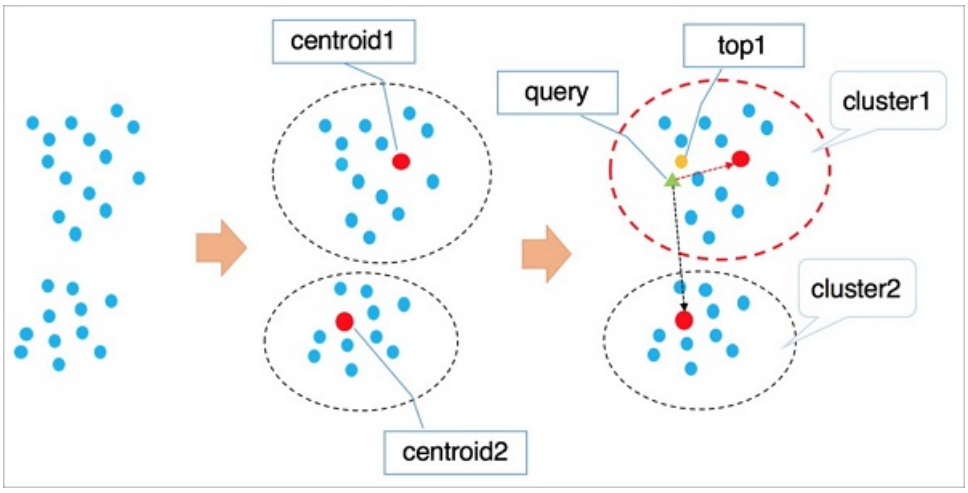
- 索引会有膨胀，大约的膨胀率可以通过 `select pg_relation_size('index名称');` 查询，当膨胀远大于数据的大小，并且查询有明显变慢时，需要重建索引。

- 数据频繁更新后索引会不精确，如果要求绝对准确，需要定期重建索引。
- IVFFlat索引如果使用内部中心点（clustering_type=1），需要先在表中插入一定数据，再创建索引。

PASE算法简述

- IVFFlat算法
  IVFFlat是IVFADC的简化版本，适合于召回精度要求高，但对查询耗时要求不严格（100ms级别）的场景。
  相比其他算法，IVFFlat算法具有以下优点：
  - 如果查询向量是候选数据集中的一员，那么IVFFlat可以达到100%的召回率。
  - 算法简单，因此索引构建更快，存储空间更小。
  - 聚类中心点可以由使用者指定，通过简单的参数调节就可以控制召回精度。
  - 算法参数可解释性强，用户能够完全地控制算法的准确性。

  IVFFlat的算法原理参见下图。

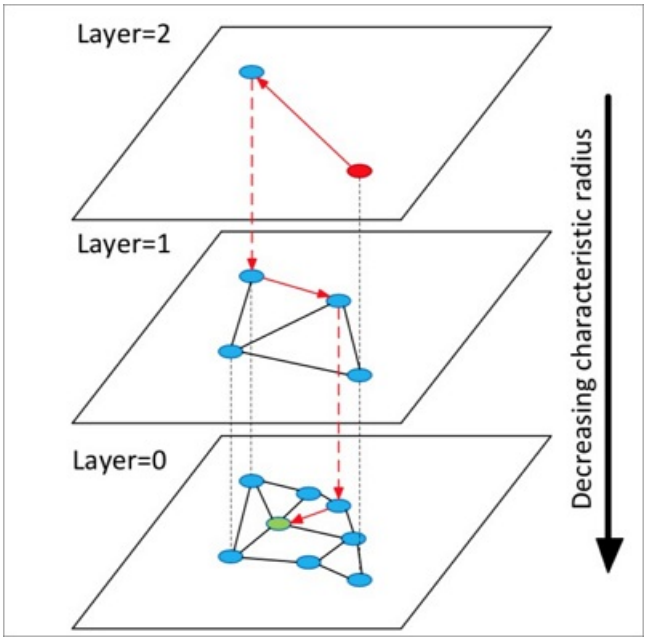


  算法流程说明：

  i. 高维空间中的点基于隐形的聚类属性，按照kmeans等聚类算法对向量进行聚类处理，使得每个类簇有一个中心点。
  ii. 检索向量时首先遍历计算所有类簇的中心点，找到与目标向量最近的n个类簇中心。
  iii. 遍历计算n个类簇中心所在聚类中的所有元素，经过全局排序得到距离最近的k个向量。

  > **说明**
  > - 在查询类簇中心点时，会自动排除远离的类簇，加速查询过程，但是无法保证最优的前k个向量全部在这n个类簇中，因此会有精度损失。您可以通过类簇个数n来控制IVFFlat算法的准确性，n值越大，算法精度越高，但计算量会越大。
  > - IVFFlat和IVFADC的第一阶段完全一样，主要区别是第二阶段计算。IVFADC通过积量化来避免遍历计算，但是会导致精度损失，而IVFFlat是暴力计算，避免精度损失，并且计算量可控。

- HNSW算法
  HNSW（Hierarchical Navigable Small World）算法适合超大规模的向量数据集（千万级别以上），并且对查询延时有严格要求（10ms级别）的场景。
  HNSW基于近邻图的算法，通过在近邻图快速迭代查找得到可能的相近点。在大数据量的情况下，使用HNSW算法的性能提升相比其他算法更加明显，但邻居点的存储会占用一部分存储空间，同时召回精度达到一定水平后难以通过简单的参数控制来提升。
  HNSW的算法原理参见下图。



算法流程说明：

i. 构造多层图，每层图都是下层图的一个缩略，同时构成下层图的跳表，类似高速公路。
ii. 从顶层随机选中一个点开始查询。
iii. 第一次搜索其邻居点，把它们按距离目标的远近顺序存储在定长的动态列表中，以后每一次查找，依次取出动态列表中的点，搜索其邻居点，再把这些新探索的邻居点插入动态列表，每次插入动态列表需要重新排序，保留前k个。如果列表有变化则继续查找，不断迭代直至达到稳态，然后以动态列表中的第一个点作为下一层的入口点，进入下一层。
iv. 循环执行第3步，直到进入最底层。

> **说明** HNSW算法是在NSW算法的单层构图的基础上构造多层图，在图中进行最近邻查找，可以实现比聚类算法更高的查询加速比。

两种算法都有特定的适用业务场景，例如IVFFlat适合高精图像对比场景，HNSW适合搜索推荐的召回场景。后续会陆续集成业界领先的算法实现到PASE中。

使用PASE

1. 创建PASE插件。命令如下：

```
CREATE EXTENSION pase;
```

2. 计算向量相似度。您可以通过以下两种构造方式计算向量相似度：
   - 采用PASE数据类型构造方式计算
     示例

```
SELECT ARRAY[2, 1, 1]::float4[] <?> pase(ARRAY[3, 1, 1]::float4[]) AS distance; SELEC
T ARRAY[2, 1, 1]::float4[] <?> pase(ARRAY[3, 1, 1]::float4[], 0) AS distance; SELECT
ARRAY[2, 1, 1]::float4[] <?> pase(ARRAY[3, 1, 1]::float4[], 0, 1) AS distance;
```

> **说明**
>
> - <?>是PASE类型的操作符，表示计算左右两个向量的相似度。左边向量数据类型必须为float4[]，右边向量数据类型必须为pase。
> - pase类型为插件内定义的数据类型，最多包括三个构造函数。以示例第三条的 `float4[], 0, 1` 部分进行说明：第一个参数为float4[]，代表右向量数据类型；第二个参数在此处没有特殊作用，可以填入0；第三个参数表示相似度计算方式，0表示欧氏距离，1表示点积（内积）。
> - 左右向量的维度必须相等，否则计算会报错。

- 采用字符串构造方式计算
  示例

```
SELECT ARRAY[2, 1, 1]::float4[] <?> '3,1,1'::pase AS distance; SELECT ARRAY[2, 1, 1]::float4[] <?> '3,1,1:0'::pase AS distance; SELECT ARRAY[2, 1, 1]::float4[] <?> '3,1,1:0:1'::pase AS distance;
```

> **说明** 采用字符串构造方式和采用PASE数据类型构造方式都是计算两个向量相似度，区别是字符串构造方式通过英文冒号（:）分隔。以示例第三条的 `3,1,1:0:1` 部分进行说明：第一个参数表示右向量；第二个参数在此处没有特殊作用，可以填入0；第三个参数表示相似度计算方式，0表示欧氏距离，1表示点积（内积）。

3. 创建索引。您可以使用两种算法创建索引：

> **说明** 对于要使用PASE向量索引的用户，如果采用欧氏距离作为向量相似度计算公式，原始向量不需要做任何处理，但如果采用内积或余弦作为向量相似度计算公式，需要对向量进行归一化处理，如原始向量为 $x_1, x_2, x_3, \dots, x_n$，则需要满足：$x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2 = 1$，此时内积和余弦值相同。

- IVFFlat算法创建索引
  示例

```
CREATE INDEX ivfflat_idx ON vectors_table USING pase_ivfflat(vector) WITH (clustering_type = 1, distance_type = 0, dimension = 256, base64_encoded = 0, clustering_params = "10,100");
```

参数说明如下。

参数	说明
clustering_type	IVFFlat算法对向量数据进行的聚类操作类型。必填项。取值： ■ 0：外部聚类，加载外部提供的中心点文件，由参数clustering_params控制。 ■ 1：内部聚类，即构建索引过程首先会在内部进行聚类操作，采用kmeans算法，由参数clustering_params控制。 对于初级用户，建议使用内部聚类方式。

参数	说明
distance_type	相似度计算方式。默认值为0。取值： ■ 0：欧式距离。 ■ 1：点积（内积）。使用此方式需要进行向量归一化，此时点积（内积）值的序和欧氏距离的序是反序关系。 当前仅支持欧式距离，点积（内积）需要向量归一化后，采用附录中提供的方法计算。
dimension	向量维度。必填项，最大支持512。
base64_encoded	数据是否采用base64编码。默认为0。取值： ■ 0：采用float4[]表示向量类型。 ■ 1：采用float[]的base64编码字符串表示向量类型。
clustering_params	对于外部聚类，该参数配置为中心点文件路径；对于内部聚类，该参数配置为聚类参数。格式为：`clustering_sample_ratio,k`。必填项。 ■ clustering_sample_ratio：以1000为分母的聚类采样比例。取值范围为(0,1000]内的整数，例如值为1，表示对表中的数据按照千分之一的比例采样后进行kmeans聚类。值越大查询准确率越高，但创建索引的时间越长，建议采样的数据总量不要超过10万条。 ■ k：聚类中心数，值越大查询准确率越高，但创建索引时间越长，建议取值范围为[100,1000]。

- HNSW算法创建索引
  示例

```
CREATE INDEX hnsw_idx ON vectors_table USING pase_hnsw(vector) WITH (dim = 256, base_
nb_num = 16, ef_build = 40, ef_search = 200, base64_encoded = 0);
```

参数说明如下。

参数	说明
dim	向量维度。必填项，最大支持512。
base_nb_num	图中节点的邻居数。必填项。值越大查询准确率越高，但建索引时间越慢，同时索引量占空间越大，建议取值范围[16-128]。
ef_build	创建索引过程中的堆长度。必填项。越长效果越好，但创建索引越慢，建议取值范围[40,400]。
ef_search	查询过程中的堆长度。必填项。越长效果越好，但查询性能越差，可在查询时指定，该处为默认值：200。
base64_encoded	数据是否采用base64编码。默认值0。取值： ■ 0：采用float4[]表示向量类型。 ■ 1：采用float[]的base64编码字符串表示向量类型。

4. 查询。您可以使用两种索引查询：
   - 使用IVFFlat索引查询
     示例

```
SELECT id, vector <#> '1,1,1'::pase as distance FROM vectors_ivfflat ORDER BY vector
<#> '1,1,1:10:0'::pase ASC LIMIT 10;
```

> **说明**
> - <#>为IVFFlat算法索引的操作符。
> - 向量索引通过ORDER BY语句生效，支持ASC升序排序。
> - pase数据类型为三段式，通过英文冒号（:）分隔。以示例的 `1,1,1:10:0` 进行说明：第一段为查询向量；第二段为IVFFlat的查询效果参数，取值范围为(0,1000]，值越大查询准确率越高，但查询性能越差，建议根据实际数据进行调试确定；第三段为查询时相似度计算方式，0表示欧式距离，1表示点积（内积）。使用点积（内积）方式需要进行向量归一化，此时点积（内积）值的序和欧氏距离的序是反序关系。

   - 使用HNSW索引查询
     示例

```
SELECT id, vector <?> '1,1,1'::pase as distance FROM vectors_ivfflat ORDER BY vector
<?> '1,1,1:100:0'::pase ASC LIMIT 10;
```

> **说明**
> - <?>为HNSW算法索引的操作符。
> - 向量索引通过ORDER BY语句生效，支持ASC升序排序。
> - pase数据类型为三段式，通过英文冒号（:）分隔。以示例的 `1,1,1:10:0` 进行说明：第一段为查询向量；第二段为HNSW的查询效果参数，取值范围为(0,∞)，值越大查询准确率越高，但查询性能越差，建议根据实际数据进行调试确定，初始值建议从40开始；第三段为查询时相似度计算方式，0表示欧式距离，1表示点积（内积）。使用点积（内积）方式需要进行向量归一化，此时点积（内积）值的序和欧氏距离的序是反序关系。

附录

- 点积（内积）方式计算示例
  示例采用HNSW算法索引，创建function示例如下：

```
CREATE OR REPLACE FUNCTION inner_product_search(query_vector text, ef integer, k integer,
table_name text) RETURNS TABLE (id integer, uid text, distance float4) AS $$ BEGIN RETURN
QUERY EXECUTE format(' select a.id, a.vector <?> pase(ARRAY[%s], %s, 1) AS distance from
(SELECT id, vector FROM %s ORDER BY vector <?> pase(ARRAY[%s], %s, 0) ASC LIMIT %s) a ORD
ER BY distance DESC;', query_vector, ef, table_name, query_vector, ef, k); END $$ LANGUAG
E plpgsql;
```

> **说明** 对于归一化的向量，内积=余弦，所以余弦值也可以用上述方式计算。

- IVFFlat索引自定义中心点文件

此功能为高级功能，需要在服务器上指定路径上传中心点文件，并作为索引参数构建索引。详细参数请参见IVFFlat索引参数描述。文件格式如下：

```
向量维度|中心点个数|中心点向量集合
```

示例

```
3|2|1,1,1,2,2,2
```

相关文档

- Product Quantization for Nearest Neighbor Search
  Herv´e J´egou, Matthijs Douze, Cordelia Schmid. Product quantization for nearest neighbor search.
- Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs
  Yu.A.Malkov, D.A.Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs.

6.2.4. 位图计算（roaringbitmap）

RDS PostgreSQL提供roaringbitmap插件，可以使用位图计算功能，提高查询性能。

前提条件

实例为RDS PostgreSQL 12、13、14。

背景信息

Roaring Bitmap算法是将32位的INT类型数据划分为2^{16}个数据块（Chunk），每一个数据块对应整数的高16位，并使用一个容器（Container）来存放一个数值的低16位。Roaring Bitmap将这些容器保存在一个动态数组中，作为一级索引。容器使用两种不同的结构：数组容器（Array Container）和位图容器（Bitmap Container）。数组容器存放稀疏的数据，位图容器存放稠密的数据。如果一个容器里面的整数数量小于4096，就用数组容器来存储值。若大于4096，就用位图容器来存储值。

采用这种存储结构，Roaring Bitmap可以快速检索一个特定的值。在做位图计算（AND、OR、XOR）时，Roaring Bitmap提供了相应的算法来高效地实现在两种容器之间的运算。使得Roaring Bitmap无论在存储和计算性能上都表现优秀。

操作步骤

1. 创建插件。示例如下：

```
CREATE EXTENSION roaringbitmap;
```

2. 创建带有RoaringBitmap数据类型的表。示例如下：

```
CREATE TABLE t1 (id integer, bitmap roaringbitmap);
```

3. 使用rb_build函数插入roaringbitmap的数据。示例如下：

```
--数组位置对应的BIT值为1
INSERT INTO t1 SELECT 1,RB_BUILD(ARRAY[1,2,3,4,5,6,7,8,9,200]);
--将输入的多条记录的值对应位置的BIT值设置为1，最后聚合为一个roaringbitmap
INSERT INTO t1 SELECT 2,RB_BUILD_AGG(e) FROM GENERATE_SERIES(1,100) e;
```

4. 进行Bitmap计算（OR、AND、XOR、ANDNOT）。示例如下：

```
SELECT RB_OR(a.bitmap,b.bitmap) FROM (SELECT bitmap FROM t1 WHERE id = 1) AS a,(SELECT
bitmap FROM t1 WHERE id = 2) AS b;
```

5. 进行Bitmap聚合计算（OR、AND、XOR、BUILD），并生成新的roaringbitmap类型。示例如下：

```
SELECT RB_OR_AGG(bitmap) FROM t1;
SELECT RB_AND_AGG(bitmap) FROM t1;
SELECT RB_XOR_AGG(bitmap) FROM t1;
SELECT RB_BUILD_AGG(e) FROM GENERATE_SERIES(1,100) e;
```

6. 统计基数（Cardinality），即统计roaringbitmap中包含多少个位置为1的BIT位。示例如下：

```
SELECT RB_CARDINALITY(bitmap) FROM t1;
```

7. 从roaringbitmap中返回位置为1的BIT下标（即位置值）。示例如下：

```
SELECT RB_ITERATE(bitmap) FROM t1 WHERE id = 1;
```

Bitmap函数列表

函数名	输入	输出	描述	示例
rb_build	integer[]	roaringbitmap	通过数组创建一个Bitmap。	`rb_build('{1,2,3,4,5}')`
rb_and	roaringbitmap，roaringbitmap	roaringbitmap	And计算。	`rb_and(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_or	roaringbitmap，roaringbitmap	roaringbitmap	Or计算。	`rb_or(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_xor	roaringbitmap，roaringbitmap	roaringbitmap	Xor计算。	`rb_xor(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`

函数名	输入	输出	描述	示例
rb_andnot	roaringbitmap，roaringbitmap	roaringbitmap	AndNot计算。	`rb_andnot(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_cardinality	roaringbitmap	integer	统计基数。	`rb_cardinality(rb_build('{1,2,3,4,5}'))`
rb_and_cardinality	roaringbitmap，roaringbitmap	integer	And计算并返回基数。	`rb_and_cardinality(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_or_cardinality	roaringbitmap，roaringbitmap	integer	Or计算并返回基数。	`rb_or_cardinality(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_xor_cardinality	roaringbitmap，roaringbitmap	integer	Xor计算并返回基数。	`rb_xor_cardinality(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_andnot_cardinality	roaringbitmap，roaringbitmap	integer	AndNot计算并返回基数。	`rb_andnot_cardinality(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`

函数名	输入	输出	描述	示例
rb_is_empty	roaringbitmap	boolean	判断是否为空的Bitmap。	`rb_is_empty(rb_build('{1,2,3,4,5}'))`
rb_equals	roaringbitmap，roaringbitmap	boolean	判断两个Bitmap是否相等。	`rb_equals(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_intersect	roaringbitmap，roaringbitmap	boolean	判断两个Bitmap是否相交。	`rb_intersect(rb_build('{1,2,3}'),rb_build('{3,4,5}'))`
rb_remove	roaringbitmap,integer	roaringbitmap	从Bitmap移除特定的Offset。	`rb_remove(rb_build('{1,2,3}'),3)`
rb_flip	roaringbitmap,integer,integer	roaringbitmap	翻转Bitmap中特定的Offset段。	`rb_flip(rb_build('{1,2,3}'),2,3)`
rb_minimum	roaringbitmap	integer	返回Bitmap中最小的Offset，如果Bitmap为空则返回-1。	`rb_minimum(rb_build('{1,2,3}'))`
rb_maximum	roaringbitmap	integer	返回Bitmap中最大的Offset，如果Bitmap为空则返回0。	`rb_maximum(rb_build('{1,2,3}'))`
rb_rank	roaringbitmap,integer	integer	返回Bitmap中小于等于指定Offset的基数。	`rb_rank(rb_build('{1,2,3}'),3)`

函数名	输入	输出	描述	示例
rb_iterate	roaringbitmap	setof integer	返回Offset List。	`rb_iterate(rb_build('{1,2,3}'))`

Bitmap聚合函数列表

聚合函数名	输入	输出	描述	示例
rb_build_agg	integer	roaringbitmap	将Offset聚合成bitmap。	`rb_build_agg(1)`
rb_or_agg	roaringbitmap	roaringbitmap	Or 聚合计算。	`rb_or_agg(rb_build('{1,2,3}'))`
rb_and_agg	roaringbitmap	roaringbitmap	And 聚合计算。	`rb_and_agg(rb_build('{1,2,3}'))`
rb_xor_agg	roaringbitmap	roaringbitmap	Xor 聚合计算。	`rb_xor_agg(rb_build('{1,2,3}'))`
rb_or_cardinality_agg	roaringbitmap	integer	Or 聚合计算并返回其基数。	`rb_or_cardinality_agg(rb_build('{1,2,3}'))`
rb_and_cardinality_agg	roaringbitmap	integer	And 聚合计算并返回其基数。	`rb_and_cardinality_agg(rb_build('{1,2,3}'))`

聚合函数名	输入	输出	描述	示例
rb_xor_cardinality_agg	roaringbitmap	integer	Xor 聚合计算并返回其基数。	`rb_xor_cardinality_agg(rb_build('{1,2,3}'))`

6.2.5. 位图功能扩展（varbitx）

社区版PostgreSQL内置的varbit插件仅支持的比较简单的BIT类型操作函数，阿里云RDS PostgreSQL对varbit插件进行了扩展，支持更多的BIT操作，可以覆盖更多应用场景，例如实时用户画像推荐系统、门禁广告系统、购票系统等。

前提条件

RDS PostgreSQL版本如下：

- PostgreSQL 11
- PostgreSQL 10

函数说明

函数	说明
`get_bit (varbit a, int b, int c) returns varbit`	从指定位置b开始获取c个BIT位，返回varbit。 例如： `get_bit('111110000011', 3, 5)` 返回11000。
`set_bit_array (varbit a, int b, int c, int[] d) returns varbit`	将下标位置数组d对应的BIT位设置为b（0或1），超出原始长度的部分填充c（0或1）。 例如： `set_bit_array('111100001111', 0, 1, array[1,15])` 返回 1011000011111110。
`bit_count (varbit a, int b, int c, int d) returns int`	从第c位开始，统计d个BIT位中有多少个b（0或1），如果超出长度，则只计算已经存在的。 例如： `bit_count('1111000011110000', 1, 5, 4)` 返回1（0001）。
`bit_count (varbit a, int b) returns int`	统计b（1或0）的个数。 例如： `bit_count('1111000011110000', 1)` 返回8。
`bit_fill (int a, int b) returns varbit`	填充b长度的a（0或1）。 例如： `bit_fill(0,10)` 返回 0000000000。
`bit_rand (int a, int b, float c) returns varbit`	填充a长度的随机BIT，并指定b（0或1）的随机比例c。 例如： `bit_rand(10, 1, 0.3)` 可能返回0101000001。
`bit_posite (varbit a, int b, boolean c) returns int[]`	返回b（0或1）的下标位置数组，下标从0开始计数，c为true时正向返回，c为false时反向返回。 例如： `bit_posite ('11110010011', 1, true)` 返回[0,1,2,3,6,9,10]， `bit_posite ('11110010011', 1, false)` 返回 [10,9,6,3,2,1,0]。

函数	说明
`bit_posite (varbit a, int b, int c, boolean d) returns int[]`	返回b（0或1）的下标位置数组，返回c个为止，下标从0开始计数，d为true时正向返回，d为false时反向返回。 例如：`bit_posite ('11110010011', 1, 3, true)` 返回[0,1,2]，`bit_posite ('11110010011', 1, 3, false)` 返回[10,9,6]。
`get_bit_array (varbit a, int b, int c, int d) returns int[]`	从b位置开始获取c个BIT位，返回d（0或1）的下标位置数组。 例如：`get_bit_array('111110000011', 3, 5, 1)` 返回11000的下标array[3,4]。
`get_bit_array (varbit a, int b, int[] c) returns int[]`	查询指定下标位置数组c对应的BIT位为b（0或1）的，返回下标位置数组，超出部分的下标不统计。 例如：`get_bit_array('111110000011', 1, array[1,5,6,7,10,11])` 返回array[1,10,11]。
`set_bit_array (varbit a, int b, int c, int[] d, int e) returns varbit`	将指定下标位置数组d对应的BIT位设置为b（0或1），设置e位即返回，超出原始长度的部分填充c（0或1）。 例如：`set_bit_array('111100001111', 1, 0, array[4,5,6,7], 2)` 返回111111001111（设置为1，超出补0，设置满2位即返回）。
`set_bit_array_record (varbit a, int b, int c, int[] d) returns (varbit,int[])`	将指定下标位置数组d对应的BIT位设置为b（0或1），超出原始长度的部分填充c（0或1），返回设置后的varbit，同时返回此次操作中被设置为b的下标位置数组。 例如：`set_bit_array_record('111100001111', 0, 1, array[1,15])` 返回1011000011111110（设置为0，超出补1）同时返回array[1,15]。
`set_bit_array_record (varbit a, int b, int c, int[] d, int e) returns (varbit,int[])`	将指定下标位置数组d对应的BIT位设置为b（0或1），超出原始长度的部分填充c（0或1），设置成功e位即返回，返回设置后的varbit，同时返回此次操作被设置为b的下标位置数组。 例如：`set_bit_array_record('111100001111', 1, 0, array[1,4,5,6,7], 2)` 返回111111001111（设置为1，超出补0，设置成功2位即返回）同时返回array[4,5]（array[1]本身为1，未设置成功）。
`bit_count_array (varbit a, int b, int[] c) returns int`	统计指定下标位置数组中0或1的个数。 例如：`bit_count_array('1111000011110000', 1, array[1,2,7,8])` 返回3。

使用方法

- 创建插件

```
CREATE EXTENSION varbitx;
```

- 删除插件

```
DROP EXTENSION varbitx;
```

- 函数使用命令

使用 `select <函数>` 即可，例如：

- bit_count函数

```
select bit_count('1111000011110000', 1, 5, 4);
```

返回结果为：

```
 bit_count ----------- 1 (1 row)
```

- set_bit_array_record函数

```
select set_bit_array_record('111100001111', 1, 0, array[1,4,5,6,7], 2);
```

返回结果为：

```
 set_bit_array_record ------------------------ (111111001111,"{4,5}") (1 row)
```

具体函数及说明请参见函数说明。

6.2.6. 化学分子计算检索（RDKit）

RDS PostgreSQL提供RDKit插件，支持化学分子计算、化学分子检索等功能。

前提条件

实例版本为RDS PostgreSQL 12。

背景信息

RDKit插件支持mol数据类型（描述分子类型）和fp数据类型（描述分子指纹），在此基础上支持比较运算、相似度计算（Tanimoto、Dice）和GiST索引。

更多RDKit插件SQL操作请参见RDKit SQL。

注意事项

- mol数据类型的输入、输出函数遵循简化分子线性输入规范（SMILES）。
- fp数据类型的输入、输出功能遵循bytea（存储二进制数据的字段类型）格式。

创建插件

```
postgres=# create extension rdkit ; CREATE EXTENSION
```

默认参数配置

```
postgres=# show rdkit.tanimoto_threshold ; rdkit.tanimoto_threshold ------------------------
--- 0.5 (1 row) postgres=# show rdkit.dice_threshold; rdkit.dice_threshold ----------------
------ 0.5 (1 row)
```

索引支持

- mol和fp的比较运算操作支持Btree索引、Hash索引。例如：

```
CREATE INDEX molidx ON pgmol (mol); CREATE INDEX molidx ON pgmol (fp);
```

- mol的 `%` 、 `#` 、 `@>` 、 `<@` 操作和fp结构的 `%` 、 `#` 操作支持Gist索引。例如：

```
CREATE INDEX molidx ON pgmol USING gist (mol);
```

函数示例

- tanimoto_sml函数计算tanimoto相似度。

```
postgres=# \df tanimoto_sml List of functions Schema | Name | Result data type | Argument
data types | Type --------+--------------+------------------+---------------------+------
public | tanimoto_sml | double precision | bfp, bfp | func public | tanimoto_sml | double
precision | sfp, sfp | func (2 rows)
```

- dice_sml函数计算dice相似度。

```
postgres=# \df dice_sml List of functions Schema | Name | Result data type | Argument dat
a types | Type --------+----------+------------------+---------------------+------ public
| dice_sml | double precision | bfp, bfp | func public | dice_sml | double precision | sf
p, sfp | func (2 rows)
```

- substruct函数，当函数的第二个参数是第一个参数的子结构时，函数返回结果为TRUE。

```
postgres=# \df substruct List of functions Schema | Name | Result data type | Argument da
ta types | Type --------+-----------+------------------+---------------------+------ publ
ic | substruct | boolean | mol, mol | func public | substruct | boolean | mol, qmol | fun
c public | substruct | boolean | reaction, reaction | func (3 rows)
```

基础操作说明

- `mol % mol` 、 `fp % fp`
  当Tanimoto相似度计算结果小于GUC配置参数rdkit.tanimoto_threshold时，该操作返回结果为TRUE。
- `mol # mol` 、 `fp # fp`
  当Dice相似度计算结果小于GUC配置参数rdkit.dice_threshold时，该操作返回结果为TRUE。
- `mol @> mol`
  如果操作符 `@>` 左边对象包含右边对象，该操作返回结果为TRUE。
- `mol <@ mol`
  如果操作符 `<@` 右边对象包含左边对象，该操作返回结果为TRUE。

6.3. 异构数据库访问

6.3.1. 读写外部数据文本文件（oss_fdw）

阿里云支持通过oss_fdw插件将OSS中的数据加载到PostgreSQL和PPAS数据库中，也支持将PostgreSQL和PPAS数据库中的数据写入OSS中。

前提条件

实例版本如下：

- PostgreSQL 13
- PostgreSQL 12
- PostgreSQL 11
- PostgreSQL 10
- PostgreSQL 9.4

oss_fdw用例

```
# PostgreSQL创建插件 create extension oss_fdw; ---对于PPAS，则执行select rds_manage_extension(
'create','oss_fdw'); # 创建server CREATE SERVER ossserver FOREIGN DATA WRAPPER oss_fdw OPTIO
NS (host 'oss-cn-hangzhou.aliyuncs.com' , id 'xxx', key 'xxx', bucket 'mybucket'); # 创建O
SS外部表 CREATE FOREIGN TABLE ossexample (date text, time text, open float, high float, l
ow float, volume int) SERVER ossserver OPTIONS ( filepath 'osstest/example.csv', delimiter
', ' , format 'csv', encoding 'utf8', PARSE_ERRORS '100'); # 创建表，数据就装载到这张表中。 c
reate table example (date text, time text, open float, high float, low float, volume in
t); # 数据从ossexample装载到example中。 insert into example select * from ossexample; # oss_f
dw能够正确估计OSS上的文件大小，正确的规划查询计划。 explain insert into example select * from oss
example; QUERY PLAN ---------------------------------------------------------------------- I
nsert on example (cost=0.00..1.60 rows=6 width=92) -> Foreign Scan on ossexample (cost=0.00
..1.60 rows=6 width=92) Foreign OssFile: osstest/example.csv.0 Foreign OssFile Size: 728 (4
rows) # 表example中的数据写出到OSS中。 insert into ossexample select * from example; explain i
nsert into ossexample select * from example; QUERY PLAN ----------------------------------
------------------------------ Insert on ossexample (cost=0.00..16.60 rows=660 width=92) ->
Seq Scan on example (cost=0.00..16.60 rows=660 width=92) (2 rows)
```

相关参数说明请参见下文。

oss_fdw参数

oss_fdw和其他fdw接口一样，对外部数据OSS中的数据进行封装。用户可以像使用数据表一样通过oss_fdw读取OSS中存放的数据。oss_fdw提供独有的参数用于连接和解析OSS上的文件数据。

> **说明**
> - 目前oss_fdw支持读取和写入OSS中文件的格式为：text和csv，或者gzip格式的text、csv文件。
> - oss_fdw各参数的值需使用双引号（""）引起来，且不含无用空格。

CREATE SERVER参数

参数	说明
ossendpoint	是内网访问OSS的地址，也称为host。
id oss	账号ID。
key oss	账号Key。
bucket	存储空间，需要先创建OSS账号再设置该参数。

针对导入模式和导出模式，提供下列容错相关参数。网络条件较差时，可以调整以下参数，以保障导入和导出成功。

参数	说明
oss_connect_timeout	设置链接超时，单位秒，默认是10秒。
oss_dns_cache_timeout	设置DNS超时，单位秒，默认是60秒。

参数	说明
oss_speed_limit	设置能容忍的最小速率，默认是1024，即1K。
oss_speed_time	设置能容忍最小速率的最长时间，默认是15秒。

说明 如果使用了oss_speed_limit和oss_speed_time的默认值，表示如果连续15秒的传输速率小于1K，则超时。

CREATE FOREIGN TABLE参数

参数	说明
filepath	OSS中带路径的文件名。 • 文件名包含文件路径，但不包含bucket。 • 该参数匹配OSS对应路径上的多个文件，支持将多个文件加载到数据库。 • 文件命名为filepath和filepath.x 支持被导入到数据库，x要求从1开始，且连续。例如，filepath、filepath.1、filepath.2、filepath.3、filepath.5 ，前4个文件会被匹配和导入，但是 filepath.5将无法导入。
dir	OSS中的虚拟文件目录。 • dir需要以（/）结尾。 • dir指定的虚拟文件目录中的所有文件（不包含子文件夹和子文件夹下的文件）都会被匹配和导入到数据库。
prefix	指定数据文件对应路径名的前缀，不支持正则表达式，且与 filepath、dir 互斥，三者只能设置其中一个。
format	指定文件的格式，目前只支持csv。
encoding	文件中数据的编码格式，支持常见的pg编码，如utf8。
parse_errors	容错模式解析，以行为单位，忽略文件分析过程中发生的错误。
delimiter	指定列的分割符。
quote	指定文件的引用字符。
escape	指定文件的逃逸字符。
null	指定匹配对应字符串的列为null，例如null ‘test’，即列值为’test’的字符串为null。
force_not_null	指定某些列的值不为null。例如，`force_not_null ‘id’` 表示：如果ID列的值为空，则该值为空字符串，而不是null。
compressiontype	设置读取和写入OSS上文件的格式： • none：默认的文件类型，即没有压缩的文本格式。 • gzip：读取文件的格式为gzip压缩格式。

参数	说明
compressionlevel	设置写入OSS的压缩格式的压缩等级，范围1到9，默认为6。

说明

- filepath和dir需要在OPTIONS参数中指定。
- filepath和dir必须指定两个参数中的其中一个，且不能同时指定。
- 导出模式目前只支持虚拟文件夹的匹配模式，即只支持dir，不支持filepath。

CREATE FOREIGN TABLE的导出模式参数

- oss_flush_block_size：单次刷出到OSS的buffer大小，默认32MB，可选范围1到128MB。
- oss_file_max_size：写入OSS的最大文件大小，超出之后会切换到另一个文件续写。默认1024MB，可选范围8到4000 MB。
- num_parallel_worker：写OSS数据的压缩模式中并行压缩线程的个数，范围1到8，默认并发数3。

辅助函数

FUNCTION oss_fdw_list_file (relname text, schema text DEFAULT ‘public’)

- 用于获得某个外部表所匹配的OSS上的文件名和文件的大小。
- 文件大小的单位是字节。

```
select * from oss_fdw_list_file('t_oss'); name | size ---------------------------------+----
------- oss_test/test.gz.1 | 739698350 oss_test/test.gz.2 | 739413041 oss_test/test.gz.3 |
739562048 (3 rows)
```

辅助功能

oss_fdw.rds_read_one_file：在读模式下，指定某个外表匹配的文件。设置后，该外部表在数据导入中只匹配这个被设置的文件。

例如，set oss_fdw.rds_read_one_file = ‘oss_test/example16.csv.1’;

```
set oss_fdw.rds_read_one_file = 'oss_test/test.gz.2'; select * from oss_fdw_list_file('t_os
s'); name | size ---------------------------------+----------- oss_test/test.gz.2 | 73941304
1 (1 rows)
```

oss_fdw注意事项

- oss_fdw是在PostgreSQL FOREIGN TABLE框架下开发的外部表插件。
- 数据导入的性能和PostgreSQL集群的资源（CPU、IO、MEM）相关，也和OSS相关。
- 为保证数据导入的性能，请确保云数据库PostgreSQL与OSS所在Region相同，相关信息请参见OSS endpoint 信息。
- 如果读取外表的SQL时触发 `ERROR: oss endpoint userendpoint not in aliyun white list` ，建议使用阿里云各可用区公共endpoint，详情请参见访问域名和数据中心。如果问题仍无法解决，请通过工单反馈。

错误处理

导入或导出出错时，日志中会出现下列错误提示信息：

- code：出错请求的HTTP状态码。
- error_code：OSS的错误码。
- error_msg：OSS的错误信息。
- req_id：标识该次请求的UUID。当您无法解决问题时，可以凭req_id来请求OSS开发工程师的帮助。

请参见以下链接中的文档了解和处理各类错误，超时相关的错误可以使用oss_ext相关参数处理。

- OSS help 页面
- PostgreSQL CREATE FOREIGN TABLE 手册
- OSS 错误处理
- OSS 错误响应

ID和Key隐藏

CREATE SERVER中的ID和Key信息如果不做任何处理，用户可以使用 `select * from pg_foreign_server` 看到明文信息，会暴露用户的ID和Key。我们通过对ID和Key进行对称加密实现对ID和Key的隐藏（不同的实例使用不同的密钥，最大限度保护用户信息），但无法使用类似GP一样的方法，增加一个数据类型，会导致老实例不兼容。

最终加密后的信息如下：

```
postgres=# select * from pg_foreign_server ; srvname | srvowner | srvfdw | srvtype | srvver
sion | srvacl | srvoptions -----------+----------+--------+---------+------------+--------+
-----------------------------------------------------------------------------------------
---------------------------------------- ---------------------------------- ossserver | 10
| 16390 | | | | {host=oss-cn-hangzhou-zmf.aliyuncs.com, id=MD5xxxxxxxx, key=MD5xxxxxxxx, buc
ket=067862}
```

加密后的信息将会以MD5开头（总长度为len，len%8==3），这样导出之后再导入不会再次加密，但是用户不能创建MD5开头的Key和ID。

6.3.2. 读写MySQL数据（mysql_fdw）

RDS PostgreSQL提供mysql_fdw插件，可以读写RDS MySQL实例或自建MySQL数据库里的数据。

前提条件

- 实例为RDS PostgreSQL 13、12、11、10云盘版。
- 将RDS PostgreSQL的专有网络网段（例如 `172.xx.xx.xx/16` ）添加到MySQL实例的白名单中，允许RDS PostgreSQL访问。

说明 您可以在RDS PostgreSQL实例的**数据库连接**中查看专有网络网段。

背景信息

PostgreSQL从9.6开始就支持并行计算，到11的时候并行计算性能得到巨大提升，10亿数据量的join查询可以实现秒级完成。所以很多用户会使用PostgreSQL作为小的数据仓库使用，同时又能提供高并发访问。

使用mysql_fdw插件能够将PostgreSQL和MySQL连接，同步MySQL数据进行数据分析。

操作步骤

1. 新建mysql_fdw插件。

```
postgres=> create extension mysql_fdw;
CREATE EXTENSION
```

2. 创建MySQL服务器定义。

```
postgres=> CREATE SERVER <server名称>
postgres->      FOREIGN DATA WRAPPER mysql_fdw
postgres->      OPTIONS (host '<连接地址>', port '<连接端口>');
CREATE SERVER
```

> **说明** 服务器定义中的 `host` 必须配置为MySQL的内网地址，`port` 必须配置为MySQL的内网端口。

示例

```
postgres=> CREATE SERVER mysql_server
postgres->      FOREIGN DATA WRAPPER mysql_fdw
postgres->      OPTIONS (host 'rm-xxx.mysql.rds.aliyuncs.com', port '3306');
CREATE SERVER
```

3. 创建用户映射，将MySQL服务器定义映射到PostgreSQL的某个用户上，将来使用这个用户访问MySQL的数据。

```
postgres=> CREATE USER MAPPING FOR <PostgreSQL用户名>
SERVER <server名称>
OPTIONS (username '<MySQL用户名>', password '<MySQL用户对应密码>');
CREATE USER MAPPING
```

示例

```
postgres=> CREATE USER MAPPING FOR pgtest
SERVER mysql_server
OPTIONS (username 'mysqltest', password 'Test1234!');
CREATE USER MAPPING
```

4. 使用上一步骤的PostgreSQL用户创建MySQL的外部表。

> **说明** 外部表的字段名要与MySQL数据库中表的字段名相同，同时可以仅创建您想要查询的字段。例如MySQL数据库中的表有3个字段ID、NAME、AGE，您可以仅创建其中2个字段ID、NAME。

```
postgres=> CREATE FOREIGN TABLE <表名> (<字段名> <数据类型>,<字段名> <数据类型>...) server
<server名称> options (dbname '<MySQL数据库名>', table_name '<MySQL表名>');
CREATE FOREIGN TABLE
```

示例

```
postgres=> CREATE FOREIGN TABLE ft_test (id1 int, name1 text) server mysql_server optio
ns (dbname 'test123', table_name 'test');
CREATE FOREIGN TABLE
```

测试读写

您可以通过外部表读写MySQL数据。

> **说明** MySQL对应的表必须有主键才可以写入数据，否则会报如下错误：
>
> ```
> ERROR:  first column of remote table must be unique for INSERT/UPDATE/DELETE operation.
> ```

```
postgres=> select * from ft_test ;
postgres=> insert into ft_test values (2,'abc');
INSERT 0 1
postgres=> insert into ft_test select generate_series(3,100),'abc';
INSERT 0 98
postgres=> select count(*) from ft_test ;
 count
-------
    99
(1 row)
```

检查执行计划，即PostgreSQL查询MySQL数据的请求在MySQL中是如何执行的。

```
postgres=> explain verbose select count(*) from ft_test ;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=1027.50..1027.51 rows=1 width=8)
   Output: count(*)
   ->  Foreign Scan on public.ft_test  (cost=25.00..1025.00 rows=1000 width=0)
         Output: id, info
         Remote server startup cost: 25
         Remote query: SELECT NULL FROM `test123`.`test`
(6 rows)
postgres=> explain verbose select id from ft_test where id=2;
                               QUERY PLAN
-----------------------------------------------------------------------
 Foreign Scan on public.ft_test  (cost=25.00..1025.00 rows=1000 width=4)
   Output: id
   Remote server startup cost: 25
   Remote query: SELECT `id` FROM `test123`.`test` WHERE ((`id` = 2))
(4 rows)
```

6.3.3. 查询日志（log_fdw）

RDS PostgreSQL提供log_fdw插件，可以查询数据库日志。

前提条件

实例为RDS PostgreSQL 11。

背景信息

log_fdw插件提供如下两个函数，帮助您查询数据库日志。

- `list_postgres_log_files()`：列出所有的csvlog文件。
- `create_foreign_table_for_log_file(IN table_name text, IN log_server text, IN log_file text)`：创建一个与特定csvlog文件相对应的外部表。

操作步骤

1. 创建log_fdw插件。

```
postgres=> create extension log_fdw;
CREATE EXTENSION
```

2. 创建log服务器定义。

```
postgres=> create server <server名称> foreign data wrapper log_fdw;
```

示例

```
postgres=> create server log_server foreign data wrapper log_fdw;
CREATE SERVER
```

3. 调用 `list_postgres_log_files()` 函数，列出所有的csvlog文件。

```
postgres=> select * from list_postgres_log_files() order by 1;
            file_name              | file_size_bytes
-----------------------------------+-----------------
 postgresql-2020-01-10_095546.csv |            3794
 postgresql-2020-01-10_100336.csv |          318318
 postgresql-2020-01-11_000000.csv |          198437
 postgresql-2020-01-11_083546.csv |            4775
 postgresql-2020-01-13_030618.csv |            3347
```

4. 调用 `create_foreign_table_for_log_file(IN table_name text, IN log_server text, IN log_file text)` 函数，创建一个与特定csvlog文件相对应的外部表。

```
postgres=> select create_foreign_table_for_log_file('<外部表名称>', '<log服务器名称>', '<c
svlog文件名称>');
```

示例

```
postgres=> select create_foreign_table_for_log_file('ft1', 'log_server', 'postgresql-20
20-01-13_030618.csv');
 create_foreign_table_for_log_file
-----------------------------------
 t
(1 row)
```

5. 查询外部表即可查询到日志内容。

```
postgres=> select log_time, message from <外部表名称> order by log_time desc limit 2;
```

示例

```
postgres=> select log_time, message from ft1 order by log_time desc limit 2;
          log_time          |                              message
----------------------------+-----------------------------------------------------------
---------
 2020-01-13 03:35:00.003+00 | cron job 1 completed: INSERT 0 1 1
 2020-01-13 03:35:00+00     | cron job 1 starting: INSERT INTO cron_test VALUES ('Hello
World')
(2 rows)
```

外部表结构

```
postgres=> \d+ ft1
                                                        Foreign table "public.ft1"
         Column         |            Type             | Collation | Nullable | Default | FD
W options | Storage  | Stats target | Description
------------------------+-----------------------------+-----------+----------+---------+---
----------+----------+--------------+-------------
 log_time               | timestamp(3) with time zone |           |          |         |
| plain    |              |
 user_name              | text                        |           |          |         |
| extended |              |
 database_name          | text                        |           |          |         |
| extended |              |
 process_id             | integer                     |           |          |         |
| plain    |              |
 connection_from        | text                        |           |          |         |
```

```
 connection_from          | text                        |           |          |         |
| extended |               |
 session_id               | text                        |           |          |         |
| extended |               |
 session_line_num         | bigint                      |           |          |         |
| plain    |               |
 command_tag              | text                        |           |          |         |
| extended |               |
 session_start_time       | timestamp with time zone    |           |          |         |
| plain    |               |
 virtual_transaction_id | text                        |           |          |         |
| extended |               |
 transaction_id           | bigint                      |           |          |         |
| plain    |               |
 error_severity           | text                        |           |          |         |
| extended |               |
 sql_state_code           | text                        |           |          |         |
| extended |               |
 message                  | text                        |           |          |         |
| extended |               |
 detail                   | text                        |           |          |         |
| extended |               |
 hint                     | text                        |           |          |         |
| extended |               |
 internal_query           | text                        |           |          |         |
| extended |               |
 internal_query_pos       | integer                     |           |          |         |
| plain    |               |
 context                  | text                        |           |          |         |
| extended |               |
 query                    | text                        |           |          |         |
| extended |               |
 query_pos                | integer                     |           |          |         |
| plain    |               |
 location                 | text                        |           |          |         |
| extended |               |
 application_name         | text                        |           |          |         |
| extended |               |
Server: log_server
FDW options: (filename 'postgresql-2020-01-13_030618.csv')
```

6.3.4. 查询SQL Server数据库（tds_fdw）

您可以使用tds_fdw插件访问SQL Server数据库的数据。

前提条件

- 实例版本如下：
  - PostgreSQL 12（内核小版本20200421及以上）
  - PostgreSQL 11（内核小版本20200402及以上）

> **说明** 您可以在 **基本信息**页面的 **配置信息**区域查看是否有 **升级内核小版本**按钮。如果有按钮，您可以单击按钮查看当前版本；如果没有按钮，表示已经是最新版。详情请参见 升级内核小版本。

- 将RDS PostgreSQL的专有网络网段（例如 `172.xx.xx.xx/16` ）添加到SQL Server实例的白名单中，允许RDS PostgreSQL访问。

> **说明** 您可以在RDS PostgreSQL实例的 **数据库连接**中查看专有网络网段。

背景信息

tds_fdw是PostgreSQL外部数据包装器，可以连接到使用Tabular Data Stream（TDS）协议的数据库，例如Microsoft SQL server。

详细说明请参见postgres_fdw。

创建插件

连接实例后创建tds_fdw插件，命令如下：

```
create extension tds_fdw;
```

使用插件

1. 创建服务器，示例如下：

```
CREATE SERVER mssql_svr FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername '<连接地址>', p
ort '<连接端口>', database 'tds_fdw_test', tds_version '7.1');
```

> **说明** 服务器定义中的 `servername` 必须配置为SQL Server的内网地址， `port` 必须配置为SQL Server的内网端口。

2. 创建外部表。您可以通过多种方式创建外部表：
   - 使用table_name定义创建外部表，示例如下：

```
CREATE FOREIGN TABLE mssql_table ( id integer, data varchar) SERVER mssql_svr OPTIONS
(table_name 'dbo.mytable', row_estimate_method 'showplan_all');
```

   - 使用schema_name和table_name定义创建外部表，示例如下：

```
CREATE FOREIGN TABLE mssql_table ( id integer, data varchar) SERVER mssql_svr OPTIONS
(schema_name 'dbo', table_name 'mytable', row_estimate_method 'showplan_all');
```

   - 使用query定义创建外部表，示例如下：

```
CREATE FOREIGN TABLE mssql_table ( id integer, data varchar) SERVER mssql_svr OPTIONS
(query 'SELECT * FROM dbo.mytable', row_estimate_method 'showplan_all');
```

- 使用远程列名创建外部表，示例如下：

```
CREATE FOREIGN TABLE mssql_table ( id integer, col2 varchar OPTIONS (column_name 'dat
a')) SERVER mssql_svr OPTIONS (schema_name 'dbo', table_name 'mytable', row_estimate_
method 'showplan_all');
```

3. 创建用户映射，示例如下：

```
CREATE USER MAPPING FOR postgres SERVER mssql_svr OPTIONS (username 'sa', password '123
456');
```

4. 导入外部模式，示例如下：

```
IMPORT FOREIGN SCHEMA dbo EXCEPT (mssql_table) FROM SERVER mssql_svr INTO public OPTION
S (import_default 'true');
```

6.3.5. 同步更新Oracle数据库（oracle_fdw）

RDS PostgreSQL提供oracle_fdw插件，可以连接到Oracle数据库，通过操作PostgreSQL表同步更新Oracle数据库中的表。

前提条件

- 实例为RDS PostgreSQL 12（内核版本20200421及以上）。

> **说明** 您可以执行 `SHOW rds_supported_extensions;` 查看是否支持oracle_fdw，如不支持请升级内核版本。

- Oracle Client版本为11.2及以上。
- Oracle Server版本要求取决于Oracle Client版本。详情请参见Oracle官方文档。
- 将RDS PostgreSQL的专有网络网段（例如 `172.xx.xx.xx/16` ）添加到Oracle数据库实例的白名单中，允许RDS PostgreSQL访问。

> **说明** 您可以在RDS PostgreSQL实例的 **数据库连接** 中查看专有网络网段。

背景信息

oracle_fdw是PostgreSQL外部表插件，可以读取Oracle数据库的数据，也非常方便地实现PostgreSQL与Oracle数据同步。
更多详细信息请参见oracle_fdw。

注意事项

- 如果需要执行UPDATE和DELETE操作，需要在创建外部表时为主键列设置key参数，详情请参见创建外部

表。

- 外部表定义的列数据类型必须是oracle_fdw可以识别并可以转换的，oracle_fdw插件对于数据类型的转换规则请参见Data types。
- WHERE子句和ORDER BY子句支持计算下推，即oracle_fdw会将子句发送给Oracle进行计算。
- JOIN操作支持计算下推，但是有以下注意事项：
  - 两个表必须被定义在相同的映射中。
  - 三个及以上的JOIN操作不支持下推。
  - JOIN操作必须包含在SELECT操作中。
  - 没有JOIN条件的CROSS JOIN操作不支持下推。
  - 如果JOIN子句被下推，ORDER BY子句将不会被下推。
- oracle_fdw支持postgis插件，安装postgis插件后支持空间数据类型，包括：
  - POINT
  - LINE
  - POLYGON
  - MULTIPOINT
  - MULTILINE
  - MULTIPOLYGON

操作步骤

1. 新建oracle_fdw插件。命令如下：

```
CREATE EXTENSION oracle_fdw;
```

2. 创建Oracle数据库映射。有如下两种命令：

```
CREATE SERVER <SERVER名称> FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver '//<Oracle数据库的内网连接地址>:<Oracle数据库的内网连接端口>/<数据库名>');
```

```
CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (host '<Oracle数据库的内网连接地址>', port '<Oracle数据库的内网连接端口>', dbname '<数据库名>');
```

3. 创建用户映射。命令如下：

```
CREATE USER MAPPING FOR <PostgreSQL用户名> SERVER <映射名> OPTIONS (user '<Oracle数据库用户名>', password '<Oracle数据库用户密码>');
```

> **说明** 如果不在PostgreSQL数据库中存储Oracle用户凭证，可以设置user为空字符串，然后提供必须的外部授权。

示例

```
CREATE USER MAPPING FOR pguser SERVER oradb OPTIONS (user 'orauser', password 'orapwd');
```

4. 创建Oracle的外部表。示例如下：

```
CREATE FOREIGN TABLE oratab ( id integer OPTIONS (key 'true') NOT NULL, text character varying(30), floating double precision NOT NULL ) SERVER oradb OPTIONS (table 'ORATAB', schema 'ORAUSER', max_long '32767', readonly 'false', sample_percent, '100', prefetch, '200');
```

说明 外部表的结构需要和Oracle中的映射表结构保持一致。

OPTIONS内的参数说明如下。

参数	说明
key	是否设置对应的列为主键，取值为true或false，默认值为false。如果要执行UPDATE和DELETE操作，必须将所有主键列设置为true。
table	表名，一般是大写，必填参数。可以使用Oracle的SQL来定义table变量的值，例如：`OPTIONS (table '(SELECT col FROM tab WHERE val = ''string'')')`，此时不要使用schema参数。
schema	一般是Oracle用户名，保持大写，用来访问不属于当前连接用户的表。
max_long	限制Oracle表中LONG、LONG RAW、XMLTYPE类型列的最大长度，取值范围是1~1073741823，默认值是32767。
readonly	限制Oracle表为只读，不允许INSERT、UPDATE、DELETE操作。
sample_percent	设置随机选择Oracle表数据的比例，用于PostgreSQL表统计信息，取值范围是0.000001~100，默认值是100。
prefetch	外表扫描时，PostgreSQL和Oracle数据表之间一次性传输的行数，取值范围是0~1024，默认值是200，0代表取消prefetch功能。

完成以上步骤即可通过操作外表来实现对Oracle表的操作。支持DELETE、INSERT、UPDATE、SELECT等基本操作，支持导入外部表定义的操作，命令如下：

```
IMPORT FOREIGN SCHEMA <ora_schema_name> FROM SERVER <server_name> INTO <schema_name> OPTIONS (case 'lower');
```

说明 case取值如下：

- keep：表示保留Oracle上的对象名，通常是大写。
- lower：表示转换所有的对象名为小写。
- smart：表示只将对象名中都是大写字母的替换为小写。

卸载插件

卸载插件命令如下：

```
DROP EXTENSION oracle_fdw;
```

6.4. 跨库操作（dblink、postgres_fdw）

使用PostgreSQL本身提供的扩展插件，例如dblink和postgres_fdw，可以跨库操作表。

背景信息

阿里云RDS for PostgreSQL云盘版实例开放dblink和postgres_fdw插件，支持相同VPC内实例（包括自建PostgreSQL数据库）间的跨库操作。

购买PostgreSQL云盘版实例。

注意事项

PostgreSQL云盘版的dblink和postgres_fdw插件进行跨库操作的注意事项如下：

- 相同VPC内的ECS/RDS PostgreSQL实例可以直接跨库操作。
- 自建PostgreSQL实例可以通过oracle_fdw或mysql_fdw连接VPC外部的Oracle实例或MySQL实例。
- 连接自身跨库操作时，host请填写localhost，port请填写 `show port` 命令返回的本地端口。
- 需要将RDS PostgreSQL的专有网络网段（例如 `172.xx.xx.xx/16` ）添加到目标数据库的白名单中，允许RDS PostgreSQL访问。

> **说明** 您可以在RDS PostgreSQL实例的 **数据库连接** 中查看专有网络网段。

使用dblink

1. 新建dblink插件。

```
create extension dblink;
```

2. 创建dblink连接。

```
postgres=> select dblink_connect('<连接名称>', 'host=<同一VPC下的另一RDS的内网域名> port=<同一VPC下的另一RDS的内网监听端口> user=<远程数据库用户名> password=<密码> dbname=<库名>'); postgres=> SELECT * FROM dblink('<连接名称>', '<SQL命令>') as <表名>(<列名> <列类型>);
```

示例

```
postgres=> select dblink_connect('a', 'host=pgm-bpxxxxx.pg.rds.aliyuncs.com port=3433 user=testuser2 password=passwd1234 dbname=postgres'); postgres=> select * from dblink('a','select * from products') as T(id int,name text,price numeric); //查询远端表
```

```
12
13 select * from dblink('a','select * from products') as T(id int,name text,price numeric);
```

数据输出 解释 消息 Notifications

	id integer	name text	price numeric
1	1	Cheese	9.99

更多详情请参见dblink。

使用postgres_fdw

1. 新建一个数据库。

```
postgres=> create database <数据库名>; //创建数据库 postgres=> \c <数据库名> //切换数据库
```

示例

```
postgres=> create database db1; CREATE DATABASE postgres=> \c db1
```

2. 新建postgres_fdw插件。

```
db1=> create extension postgres_fdw;
```

3. 新建远程数据库服务器。

```
db1=> CREATE SERVER <server名称> FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host '<同一VPC下的另一RDS的内网域名>,port '<同一VPC下的另一RDS的内网监听端口>', dbname '<同一VPC下的另一RDS的库名>'); db1=> CREATE USER MAPPING FOR <本地数据库用户名> SERVER <server名称> OPTIONS ( user '<远程数据库用户名>', password '<远程数据库密码>');
```

示例

```
db1=> CREATE SERVER foreign_server1 FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host 'pgm-bpxxxxx.pg.rds.aliyuncs.com', port '3433', dbname 'postgres'); CREATE SERVER db1=> CREATE USER MAPPING FOR testuser SERVER foreign_server1 OPTIONS (user 'testuser2', password 'passwd1234'); CREATE USER MAPPING
```

4. 导入外部表。

```
db1=> import foreign schema public from server foreign_server1 into <SCHEMA名称>; //导入外部表 db1=> select * from <SCHEMA名称>.<表名> //查询远端表
```

示例

```
db1=> import foreign schema public from server foreign_server1 into ft; IMPORT FOREIGN SCHEMA db1=> select * from ft.products;
```

```
17  select * from ft.products;
18
19
```

数据输出 解释 消息 Notifications

	product_no integer	name text	price numeric
1	1	Cheese	9.99

更多详情请参见postgres_fdw。

6.5. 基数统计（hll）

hll插件支持的数据类型HyperLogLog可以帮助您快速预估PV、UV等业务指标。

前提条件

实例版本如下：

- PostgreSQL 14
- PostgreSQL 13
- PostgreSQL 12（内核小版本20200421及以上）
- PostgreSQL 11（内核小版本20200402及以上）

> **说明** 您可以在 **基本信息**页面的 **配置信息**区域查看是否有 **升级内核小版本**按钮。如果有按钮，您可以单击按钮查看当前版本；如果没有按钮，表示已经是最新版。详情请参见 升级内核小版本。

背景信息

hll插件支持一种可变长、类似集合的数据类型HyperLogLog（hll），常用于在指定精度下返回近似的distinct值，例如1280字节的hll数据类型可以高精度地估计出近百亿的distinct值。hll适合互联网广告分析或其他有类似预估分析计算需求的行业，可以快速预估PV、UV等业务指标。

更多hll插件使用方法请参见postgresql-hll。

详细算法说明请参见论文HyperLogLog: the analysis of a near-optimalcardinality estimation algorithm。

创建hll插件

连接实例后创建hll插件，命令如下：

```
CREATE EXTENSION hll;
```

基础操作

- 创建一个含有hll字段的表，命令如下：

```
create table agg (id int primary key,userids hll);
```

- 将int数据类型转换为hll_hashval，命令如下：

```
select 1::hll_hashval;
```

基本操作符

- hll类型支持如下操作符：
  - =
  - !=
  - <>
  - ||
  - #

  示例如下：

```
select hll_add_agg(1::hll_hashval) = hll_add_agg(2::hll_hashval); select hll_add_agg(1::h
ll_hashval) || hll_add_agg(2::hll_hashval); select #hll_add_agg(1::hll_hashval);
```

- hll_hashval类型支持如下操作符：
  - =
  - !=
  - <>

  示例如下：

```
select 1::hll_hashval = 2::hll_hashval; select 1::hll_hashval <> 2::hll_hashval;
```

基本函数

- 支持hll_hash_boolean、hll_hash_smallint和hll_hash_bigint等hash函数，示例如下：

```
select hll_hash_boolean(true); select hll_hash_integer(1);
```

- 支持hll_add_agg函数，可以将int转换为hll格式，示例如下：

```
select hll_add_agg(1::hll_hashval);
```

- 支持hll_union函数，可以将hll并集，示例如下：

```
select hll_union(hll_add_agg(1::hll_hashval),hll_add_agg(2::hll_hashval));
```

- 支持hll_set_defaults函数，可以设置精度，示例如下：

```
select hll_set_defaults(15,5,-1,1);
```

- 支持hll_print函数，用于打印debug信息，示例如下：

```
select hll_print(hll_add_agg(1::hll_hashval));
```

示例

```
create table access_date (acc_date date unique, userids hll); insert into access_date selec
t current_date, hll_add_agg(hll_hash_integer(user_id)) from generate_series(1,10000) t(user
_id); insert into access_date select current_date-1, hll_add_agg(hll_hash_integer(user_id))
from generate_series(5000,20000) t(user_id); insert into access_date select current_date-2,
hll_add_agg(hll_hash_integer(user_id)) from generate_series(9000,40000) t(user_id); postgre
s=# select #userids from access_date where acc_date=current_date; ?column? ----------------
-- 9725.85273370708 (1 row) postgres=# select #userids from access_date where acc_date=curr
ent_date-1; ?column? ------------------ 14968.6596883279 (1 row) postgres=# select #userids
from access_date where acc_date=current_date-2; ?column? ------------------ 29361.520914991
1 (1 row)
```

6.6. 定时任务（pg_cron）

本文介绍如何通过RDS PostgreSQL提供的pg_cron插件设置定时任务。

背景信息

pg_cron是基于cron的作业调度插件，语法与常规cron相同，但它可以直接从数据库执行PostgreSQL命令。

每一个定时任务分为两部分：

- 定时计划
  规定使用插件的计划，例如每隔1分钟执行一次该任务。
  定时计划使用标准的cron语法，其中*表示**任意时间都运行**，特定数字表示**仅在这个时间时运行**。

  ```
  ┌──────────── 分钟： 0 ~ 59 │ ┌──────────── 小时： 0 ~ 23 │ │ ┌──────────── 日期： 1
  ~ 31 │ │ │ ┌──────────── 月份： 1 ~ 12 │ │ │ │ ┌──────────── 一周中的某一天 ：0 ~ 6
  ，0表示周日。 │ │ │ │ │ │ │ │ │ │ │ │ │ │ * * * * *
  ```

  如
  例如每周六3:30am（GMT）的语法为：

  ```
  30 3 * * 6
  ```

- 定时任务
  用户具体的任务内容，例如 `select * from some_table` 。

注意事项

- 该插件支持RDS PostgreSQL 10、11、12。
- 定时任务执行的时间是GMT时间，请注意换算时间。
- 定时任务都储存于默认数据库postgres中，但是您可以在其他数据库上查询定时任务。
- 由于pg_cron插件已升级，如果您在升级内核小版本20201130之前已经在使用pg_cron，请重新创建插件来使用新的特性，但是重新创建后原有定时任务会丢失。
- 使用该插件前，需要将pg_cron加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为**shared_preload_libraries**参数添加**pg_cron**。具体操作，请参见设置实例参数。

使用方法

- 创建插件

  ```
  CREATE EXTENSION pg_cron;
  ```

> **说明** 仅高权限账号可以执行此命令。

- 删除插件

```
DROP EXTENSION pg_cron;
```

> **说明** 仅高权限账号可以执行此命令。

- 执行某个任务

```
SELECT cron.schedule('<定时计划>', '<定时任务>')
```

示例

```
-- 周六3:30am (GMT) 删除过期数据。 SELECT cron.schedule('30 3 * * 6', $$DELETE FROM events WHERE event_time < now() - interval '1 week'$$); ---------- -- 每天的10:00am (GMT) 执行磁盘清理。 SELECT cron.schedule('0 10 * * *', 'VACUUM'); ---------- -- 每分钟执行指定脚本。 SELECT cron.schedule('* * * * *', 'select 1;'); ---------- -- 每个小时的23分执行指定脚本。 SELECT cron.schedule('23 * * * *', 'select 1;'); ---------- -- 每个月的4号执行指定脚本。 SELECT cron.schedule('* * 4 * *', 'select 1;');
```

- 指定数据库执行任务

```
SELECT cron.schedule('<定时计划>', '<定时任务>', '<指定数据库>')
```

> **说明** 不指定数据库时会使用配置文件中的默认数据库postgres。

- 删除某个任务

```
SELECT cron.unschedule(<定时任务ID>)
```

示例

```
SELECT cron.unschedule(43);
```

- 查看当前任务列表

```
SELECT * FROM cron.job;
```

6.7. 水平拆分（PL/Proxy）

PL/Proxy插件包含CLUSTER模式和CONNECT模式，可以帮助您用不同方式访问数据库。

前提条件

实例版本如下：

- PostgreSQL 14
- PostgreSQL 13
- PostgreSQL 12（内核小版本20200421及以上）
- PostgreSQL 11（内核小版本20200402及以上）

> **说明** 您可以在 **基本信息**页面的 **配置信息**区域查看是否有 **升级内核小版本**按钮。如果有按钮，您可以单击按钮查看当前版本；如果没有按钮，表示已经是最新版。详情请参见 升级内核小版本。

背景信息

PL/Proxy插件包含如下两种模式：

- CLUSTER模式
  支持数据库水平拆分和SQL复制。
- CONNECT模式
  支持将SQL请求路由到指定的数据库。

更多PL/Proxy插件使用方法请参见PL/Proxy。

注意事项

- 相同VPC内的PostgreSQL实例可以直接跨库操作。
- 不同VPC内的PostgreSQL实例可以通过本VPC内的ECS实例进行端口跳转，实现跨库操作。
- 代理节点后端的数据节点数必须是2的N次方。

测试环境

选择一个数据库实例作为代理节点，另外两个数据库实例作为数据节点。详细信息如下。

IP	节点类型	数据库名	用户名
100.xx.xx.136	代理节点	postgres	postgres
100.xx.xx.72	数据节点	pl_db0	postgres
11.xx.xx.9	数据节点	pl_db1	postgres

创建PL/Proxy插件

创建PL/Proxy插件命令如下：

```
create extension plproxy
```

创建PL/Proxy集群

> **说明** CONNECT模式不需要进行本操作。

1. 创建PL/Proxy集群，指定连接的子节点的数据库名、IP地址和端口，示例如下：

   ```
   postgres=# CREATE SERVER cluster_srv1 FOREIGN DATA WRAPPER plproxy postgres-# OPTIONS (
   postgres(# connection_lifetime '1800', postgres(# disable_binary '1', postgres(# p0 'db
   name=pl_db0 host=100.xxx.xxx.72 port=5678', postgres(# p1 'dbname=pl_db1 host=11.xxx.xx
   x.9 port=5678' postgres(# ); CREATE SERVER
   ```

2. 为postgres用户赋予权限，示例如下：

   ```
   postgres=# grant usage on FOREIGN server cluster_srv1 to postgres; GRANT
   ```

3. 创建用户映射，示例如下：

```
postgres=> create user mapping for postgres server cluster_srv1 options (user 'postgres
'); CREATE USER MAPPING
```

创建测试表

在每个数据节点创建测试表（代理节点不需要创建），示例如下：

```
create table users(userid int, name text);
```

CLUSTER模式测试

数据水平拆分测试步骤如下：

1. 在每个数据节点创建插入函数，示例如下：

```
pl_db0=> CREATE OR REPLACE FUNCTION insert_user(i_id int, i_name text) pl_db0-> RETURNS
integer AS $$ pl_db0$> INSERT INTO users (userid, name) VALUES ($1,$2); pl_db0$> SELECT
1; pl_db0$> $$ LANGUAGE SQL; CREATE FUNCTION pl_db1=> CREATE OR REPLACE FUNCTION insert
_user(i_id int, i_name text) pl_db1-> RETURNS integer AS $$ pl_db1$> INSERT INTO users
(userid, name) VALUES ($1,$2); pl_db1$> SELECT 1; pl_db1$> $$ LANGUAGE SQL; CREATE FUNC
TION
```

2. 在代理节点创建同名的插入函数，示例如下：

```
postgres=> CREATE OR REPLACE FUNCTION insert_user(i_id int, i_name text) postgres-> RET
URNS integer AS $$ postgres$> CLUSTER 'cluster_srv1'; postgres$> RUN ON ANY; postgres$>
$$ LANGUAGE plproxy; CREATE FUNCTION
```

3. 在代理节点创建读取函数，示例如下：

```
postgres=> CREATE OR REPLACE FUNCTION get_user_name() postgres-> RETURNS TABLE(userid i
nt, name text) AS $$ postgres$> CLUSTER 'cluster_srv1'; postgres$> RUN ON ALL ; postgre
s$> SELECT userid,name FROM users; postgres$> $$ LANGUAGE plproxy; CREATE FUNCTION
```

4. 在代理节点插入10条测试记录，示例如下：

```
SELECT insert_user(1001, 'Sven'); SELECT insert_user(1002, 'Marko'); SELECT insert_user
(1003, 'Steve'); SELECT insert_user(1004, 'lottu'); SELECT insert_user(1005, 'rax'); SE
LECT insert_user(1006, 'ak'); SELECT insert_user(1007, 'jack'); SELECT insert_user(1008
, 'molica'); SELECT insert_user(1009, 'pg'); SELECT insert_user(1010, 'oracle');
```

5. 由于插入函数执行的是RUN ON ANY，即插入数据时随机选取数据节点，查看每个数据节点的数据如下：

```
pl_db0=> select * from users; userid | name --------+-------- 1001 | Sven 1003 | Steve
1004 | lottu 1005 | rax 1006 | ak 1007 | jack 1008 | molica 1009 | pg (8 rows) pl_db1=>
select * from users; userid | name --------+-------- 1002 | Marko 1010 | oracle (2 rows
)
```

> **说明** 通过查询可以发现10条数据分布在不同数据节点，由于10条数据太少，导致分布不均匀。

6. 在代理节点执行读取函数，由于执行的是RUN ON ALL，即代理节点返回所有数据节点查询结果，示例如下：

```
postgres=> SELECT USERID,NAME FROM GET_USER_NAME(); userid | name --------+-------- 100
1 | Sven 1003 | Steve 1004 | lottu 1005 | rax 1006 | ak 1007 | jack 1008 | molica 1009
| pg 1002 | Marko 1010 | oracle (10 rows)
```

SQL复制测试步骤如下：

1. 在各个节点创建清理函数用于清理表users数据，示例如下：

```
pl_db0=> CREATE OR REPLACE FUNCTION trunc_user() pl_db0-> RETURNS integer AS $$ pl_db0$
> truncate table users; pl_db0$> SELECT 1; pl_db0$> $$ LANGUAGE SQL; CREATE FUNCTION pl
_db1=> CREATE OR REPLACE FUNCTION trunc_user() pl_db1-> RETURNS integer AS $$ pl_db1$>
truncate table users; pl_db1$> SELECT 1; pl_db1$> $$ LANGUAGE SQL; CREATE FUNCTION post
gres=> CREATE OR REPLACE FUNCTION trunc_user() postgres-> RETURNS SETOF integer AS $$ p
ostgres$> CLUSTER 'cluster_srv1'; postgres$> RUN ON ALL; postgres$> $$ LANGUAGE plproxy
; CREATE FUNCTION
```

2. 在代理节点执行清理函数，示例如下：

```
postgres=> SELECT TRUNC_USER(); trunc_user ------------ 1 1 (2 rows)
```

3. 在代理节点创建插入函数，示例如下：

```
postgres=> CREATE OR REPLACE FUNCTION insert_user_2(i_id int, i_name text) postgres-> R
ETURNS SETOF integer AS $$ postgres$> CLUSTER 'cluster_srv1'; postgres$> RUN ON ALL; po
stgres$> TARGET insert_user; postgres$> $$ LANGUAGE plproxy; CREATE FUNCTION
```

4. 在代理节点插入4条测试记录，示例如下：

```
SELECT insert_user_2(1004, 'lottu'); SELECT insert_user_2(1005, 'rax'); SELECT insert_u
ser_2(1006, 'ak'); SELECT insert_user_2(1007, 'jack');
```

5. 查看每个数据节点的数据，示例如下：

```
pl_db0=> select * from users; userid | name --------+------- 1004 | lottu 1005 | rax 10
06 | ak 1007 | jack (4 rows) pl_db1=> select * from users; userid | name --------+-----
-- 1004 | lottu 1005 | rax 1006 | ak 1007 | jack (4 rows)
```

> **说明** 每个数据节点的数据都一样，说明数据复制成功。

6. 在代理节点查询时，只需要执行RUN ON ANY，即在任意一个数据节点读取数据即可，示例如下：

```
postgres=> CREATE OR REPLACE FUNCTION get_user_name_2() postgres-> RETURNS TABLE(userid
int, name text) AS $$ postgres$> CLUSTER 'cluster_srv1'; postgres$> RUN ON ANY ; postgr
es$> SELECT userid,name FROM users; postgres$> $$ LANGUAGE plproxy; CREATE FUNCTION pos
tgres=> SELECT USERID,NAME FROM GET_USER_NAME_2(); userid | name --------+------- 1004
| lottu 1005 | rax 1006 | ak 1007 | jack (4 rows)
```

CONNECT模式测试

使用CONNECT模式时，代理节点可以直接跨实例访问，示例如下：

```
postgres=> CREATE OR REPLACE FUNCTION get_user_name_3() postgres-> RETURNS TABLE(userid int
, name text) AS $$ postgres$> CONNECT 'dbname=pl_db0 host=100.81.137.72 port=56789'; postgr
es$> SELECT userid,name FROM users; postgres$> $$ LANGUAGE plproxy; CREATE FUNCTION postgre
s=> SELECT USERID,NAME FROM GET_USER_NAME_3(); userid | name --------+------- 1004 | lottu
1005 | rax 1006 | ak 1007 | jack (4 rows)
```

6.8. 模糊查询（pg_bigm）

pg_bigm是阿里云产品RDS Postgresql的一款插件，该插件提供了全文本搜索能力，允许创建一个二元语法（2-gram）的GIN索引来加速搜索过程。

前提条件

- RDS PostgreSQL实例需为以下版本之一：
  - PostgreSQL 14
  - PostgreSQL 13
  - PostgreSQL 12
  - PostgreSQL 11
  - PostgreSQL 10
- 使用该插件前，需要将pg_bigm加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为**shared_preload_libraries**参数添加**pg_bigm**。具体操作，请参见设置实例参数。

> **说明** 如果无法创建该插件，请先 升级内核小版本。

与pg_trgm异同

pg_trgm是RDS Postgresql的另一款插件，使用3-gram的模型来实现全文本搜索。pg_bigm插件是在pg_trgm基础上继续开发的，两者的区别如下。

功能和特性	pg_trgm	pg_bigm
全文搜索的短语匹配方法	3-gram	2-gram
支持的索引类型	GIN和GIST	GIN
支持的全文本搜索操作符号	`LIKE`、`ILIKE`、`~`、`~*`	`LIKE`
非字母语言的全文本搜索	不支持	支持
带有1~2个字符的关键字的全文本搜索	慢	快
相似性搜索	支持	支持
最大可以索引的列大小	238,609,291字节（约228 MB）	107,374,180字节（约102 MB）

注意事项

- 建立GIN索引的列的长度不可以超过107,374,180字节（约102 MB）。
- 如果数据库中存储的内容语言是非ASCII，则建议将数据库的编码方式改为UTF8。

> **说明** 查询当前数据库的编码方式命令为 `select pg_encoding_to_char(encoding) from pg_database where datname = current_database();` 。

基本操作

- 创建插件

```
postgres=> create extension pg_bigm; CREATE EXTENSION
```

- 创建索引

```
postgres=> CREATE TABLE pg_tools (tool text, description text); CREATE TABLE postgres=> INSERT INTO pg_tools VALUES ('pg_hint_plan', 'Tool that allows a user to specify an optimizer HINT to PostgreSQL'); INSERT 0 1 postgres=> INSERT INTO pg_tools VALUES ('pg_dbms_stats', 'Tool that allows a user to stabilize planner statistics in PostgreSQL'); INSERT 0 1 postgres=> INSERT INTO pg_tools VALUES ('pg_bigm', 'Tool that provides 2-gram full text search capability in PostgreSQL'); INSERT 0 1 postgres=> INSERT INTO pg_tools VALUES ('pg_trgm', 'Tool that provides 3-gram full text search capability in PostgreSQL'); INSERT 0 1 postgres=> CREATE INDEX pg_tools_idx ON pg_tools USING gin (description gin_bigm_ops); CREATE INDEX postgres=> CREATE INDEX pg_tools_multi_idx ON pg_tools USING gin (tool gin_bigm_ops, description gin_bigm_ops) WITH (FASTUPDATE = off); CREATE INDEX
```

- 执行全文本搜索

```
postgres=> SELECT * FROM pg_tools WHERE description LIKE '%search%'; tool | description ---------+------------------------------------------------------------------------ pg_bigm | Tool that provides 2-gram full text search capability in PostgreSQL pg_trgm | Tool that provides 3-gram full text search capability in PostgreSQL (2 rows)
```

- 使用 `=%` 操作符执行相似性搜索

```
postgres=> SET pg_bigm.similarity_limit TO 0.2; SET postgres=> SELECT tool FROM pg_tools WHERE tool =% 'bigm'; tool --------- pg_bigm pg_trgm (2 rows)
```

- 卸载插件

```
postgres=> drop extension pg_bigm; DROP EXTENSION
```

插件常用函数

- likequery函数
  - 作用：生成可以被LIKE关键字识别的字符串。
  - 参数：1个请求参数，类型为字符串。
  - 返回值：可以被LIKE关键字识别的搜索字符串。
  - 实现原理：
    - 在关键词前后添加 `%` 符号。
    - 使用 `\` 来自动转义符号 `%` 。

- 示例如下：

```
postgres=> SELECT likequery('pg_bigm has improved the full text search performance by 2
00%'); likequery ------------------------------------------------------------------ %p
g\_bigm has improved the full text search performance by 200\%% (1 row) postgres=> SELE
CT * FROM pg_tools WHERE description LIKE likequery('search'); tool | description -----
----+------------------------------------------------------------------- pg_bigm | To
ol that provides 2-gram full text search capability in PostgreSQL pg_trgm | Tool that p
rovides 3-gram full text search capability in PostgreSQL (2 rows)
```

- show_bigm函数
  - 作用：返回给定字符串的所有2-gram元素的集合。
  - 参数：1个请求参数，类型为字符串。
  - 返回值：数组，包含所有的2-gram元素。
  - 实现原理：
    - 在字符串前后添加空格字符。
    - 计算所有的2-gram子串。
  - 示例如下：

```
postgres=> SELECT show_bigm('full text search'); show_bigm ----------------------------
------------------------------------- {" f"," s"," t",ar,ch,ea,ex,fu,"h ","l ",ll,rc,s
e,"t ",te,ul,xt} (1 row)
```

- bigm_similarity函数
  - 作用：计算两个字符串的相似度。
  - 参数：2个请求参数，类型为字符串。
  - 返回值：浮点数，表示相似度。
  - 实现原理：
    - 统计两个字符串共有的2-gram元素。
    - 相似度范围是[0, 1]，0代表两个字符串完全不一样，1代表两个字符串一样。

> **说明**
> - 由于计算2-gram时，会在字符串前后添加空格，于是 `ABC` 和 `B` 的相似度为0， `ABC` 和 `A` 的相似度为0.25。
> - bigm_similarity函数是大小写敏感的，例如 `ABC` 和 `abc` 的相似度为0。

  - 示例如下：

```
postgres=> SELECT bigm_similarity('full text search', 'text similarity search'); bigm_s
imilarity ----------------- 0.5714286 (1 row) postgres=> SELECT bigm_similarity('ABC',
'A'); bigm_similarity ----------------- 0.25 (1 row) postgres=> SELECT bigm_similarity(
'ABC', 'B'); bigm_similarity ----------------- 0 (1 row) postgres=> SELECT bigm_similar
ity('ABC', 'abc'); bigm_similarity ----------------- 0 (1 row)
```

- pg_gin_pending_stats函数
  - 作用：返回GIN索引的pending list中页面和元组的个数。

- 参数：1个，GIN索引的名字或者OID。
- 返回值：2个，pending list中页面的数量和元组的数量。

> **说明** 如果GIN索引创建时，指定参数 FASTUPDATE 为 False ，则该GIN索引不存在pending list，即返回结果为0。

- 示例如下：

```
postgres=> SELECT * FROM pg_gin_pending_stats('pg_tools_idx'); pages | tuples -------+-
------- 0 | 0 (1 row)
```

插件行为控制

- pg_bigm.last_update
  该插件的最后更新日期，只读参数，无法修改。
  示例如下：

```
SHOW pg_bigm.last_update;
```

- pg_bigm.enable_recheck
  决定是否进行recheck。

> **说明** 建议您保持默认值（ON）以保证结果正确性。

示例如下：

```
postgres=> CREATE TABLE tbl (doc text); CREATE TABLE postgres=> INSERT INTO tbl VALUES('H
e is awaiting trial'); INSERT 0 1 postgres=> INSERT INTO tbl VALUES('It was a trivial mis
take'); INSERT 0 1 postgres=> CREATE INDEX tbl_idx ON tbl USING gin (doc gin_bigm_ops); C
REATE INDEX postgres=> SET enable_seqscan TO off; SET postgres=> EXPLAIN ANALYZE SELECT *
FROM tbl WHERE doc LIKE likequery('trial'); QUERY PLAN ----------------------------------
------------------------------------------------------------------------------ Bitmap He
ap Scan on tbl (cost=20.00..24.01 rows=1 width=32) (actual time=0.020..0.021 rows=1 loops
=1) Recheck Cond: (doc ~~ '%trial%'::text) Rows Removed by Index Recheck: 1 Heap Blocks:
exact=1 -> Bitmap Index Scan on tbl_idx (cost=0.00..20.00 rows=1 width=0) (actual time=0.
013..0.013 rows=2 loops=1) Index Cond: (doc ~~ '%trial%'::text) Planning Time: 0.117 ms E
xecution Time: 0.043 ms (8 rows) postgres=> postgres=> SELECT * FROM tbl WHERE doc LIKE l
ikequery('trial'); doc ---------------------- He is awaiting trial (1 row) postgres=> SET
pg_bigm.enable_recheck = off; SET postgres=> SELECT * FROM tbl WHERE doc LIKE likequery('
trial'); doc ------------------------- He is awaiting trial It was a trivial mistake (2
rows)
```

- pg_bigm.gin_key_limit
  限制用于全文本搜索的2-gram元素的最大个数，默认为0，0代表使用所有的2-gram元素。

> **说明** 如果发现使用所有的2-gram元素导致性能下降，可以调整该参数值，限制2-gram元素的个数来提高性能。

- pg_bigm.similarity_limit
  设置相似度阈值，相似度超过这个阈值的元组会做为相似性搜索的结果。

6.9. 集成Elasticsearch（ZomboDB）

ZomboDB是一个PostgreSQL扩展插件，支持原生的访问方式，为PostgreSQL数据库带来了强大的文本索引和分析功能。

前提条件

RDS PostgreSQL实例版本为PostgreSQL 11。

背景信息

ZomboDB提供了一套全方位的查询语言，可以供您自由地查询关系型数据。此外，ZomboDB允许您创建ZomboDB类型的索引，此时ZomboDB完全接管远程的Elasticsearch，并负责文本搜索的事务正确性。

ZomboDB的优势在于允许您直接使用Elasticsearch的强大功能而不用处理同步、通信等问题。

插件的创建与删除

- 创建插件

```
CREATE EXTENSION zombodb;
```

- 删除插件

```
DROP EXTENSION zombodb;
```

示例

1. 创建一个表。

```
CREATE TABLE products ( id SERIAL8 NOT NULL PRIMARY KEY, name text NOT NULL, keywords v
archar(64)[], short_summary text, long_description zdb.fulltext, price bigint, inventor
y_count integer, discontinued boolean default false, availability_date date );
```

2. 为表添加ZomboDB类型的索引。

```
CREATE INDEX idxproducts ON products USING zombodb ((products.*)) WITH (url='localhost:
9200/');
```

> **说明** WITH语句后跟随了Elasticsearch的地址，该地址指向了一个正在服务的Elasticsearch实例。

3. 使用ZomboDB格式的查询语句进行查询。

```
SELECT * FROM products WHERE products ==> '(keywords:(sports OR box) OR long_descriptio
n:"wooden away"~5) AND price:[1000 TO 20000]';
```

> **说明** ZomboDB格式的查询语句详细规则请参见 ZomboDB官方文档。

6.10. SQL防火墙（sql_firewall）

SQL防火墙是数据库层面的防火墙功能，可以防止恶意SQL注入。可以用来学习一些定义好的SQL规则，并将这些规则储存在数据库中作为白名单，学习完成后，可以限制用户执行这些定义规则之外的风险操作。

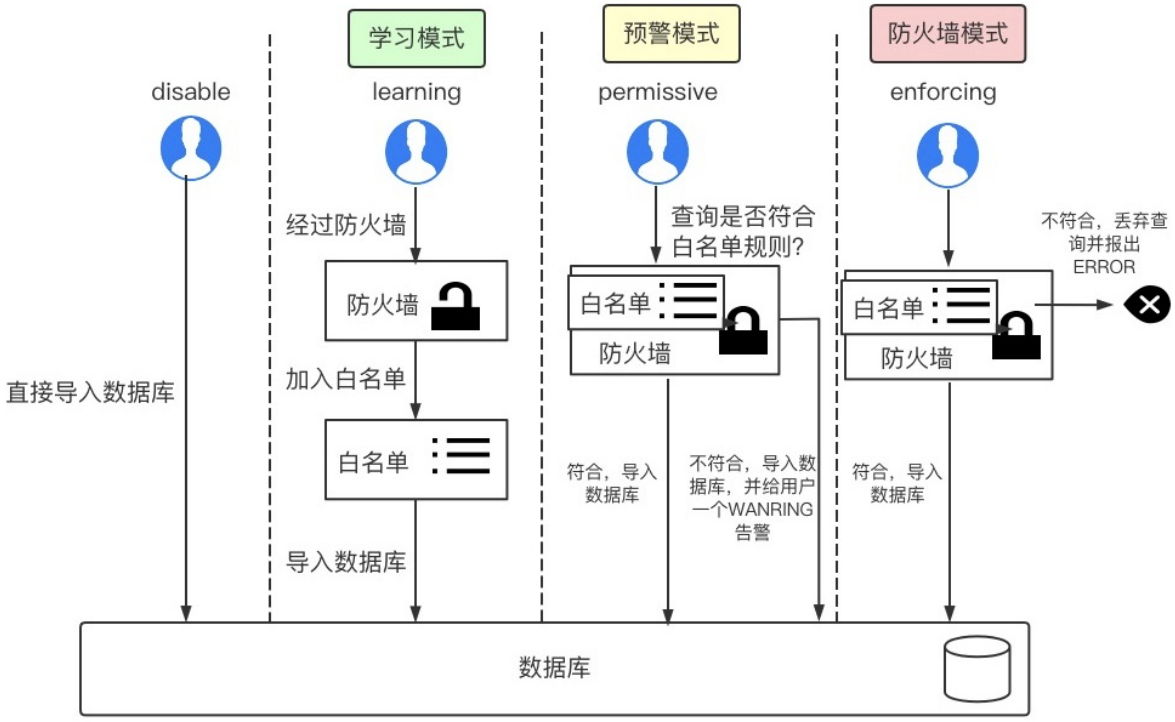
前提条件

RDS PostgreSQL实例需为以下版本之一：

- PostgreSQL 12

- PostgreSQL 11
- PostgreSQL 10

学习模式、预警模式与防火墙模式



SQL防火墙支持如下三种模式：

- 学习模式：防火墙记录用户执行的SQL，并添加到常用SQL白名单。
- 预警模式：防火墙对用户将执行的SQL进行判断，如果SQL不在白名单中，仍然会执行该SQL，但是会告警用户该SQL不在白名单中。
- 防火墙模式：防火墙对用户将执行的SQL进行判断，如果SQL不在白名单中，防火墙会拒绝执行该SQL并返回错误提示。

SQL防火墙的使用步骤

1. 打开防火墙的学习模式，让防火墙记录用户执行的SQL，并加入白名单。这个过程建议持续一段较长的时间，尽量让防火墙学习到所有可能执行的SQL。
2. 切换防火墙为预警模式，这个过程防火墙会对用户的一些不在白名单的SQL进行告警，用户结合自己的业务判断是否为风险SQL，如果这些SQL确实是用户需要的业务语句，则记录这些SQL，之后打开学习模式进行二次学习。
3. 经过前两步，用户常用SQL已经被记录完毕，打开防火墙模式，此时不在白名单的SQL均会被拒绝执行。

操作说明

- 创建插件

```
create extension sql_firewall;
```

- 删除插件

```
drop extension sql_firewall;
```

- 切换模式
  在控制台找到sql_firewall.firewall参数，修改参数值并重启实例。详情请参见设置实例参数。
  sql_firewall.firewall取值如下：
  - disable：关闭SQL防火墙
  - learning：学习模式
  - permissive：预警模式
  - enforcing：防火墙模式
- 功能函数
  - sql_firewall_reset()
    清空白名单。该函数只有在防火墙关闭模式下，rds_superuser权限的用户可执行该函数。
  - sql_firewall_stat_reset()
    清空统计信息。该函数只有在防火墙关闭模式下，rds_superuser权限的用户可执行该函数。
- 视图函数
  - sql_firewall.sql_firewall_statements
    展示目前数据库中所有的白名单及其被调用的次数。

    ```
    postgres=# select * from sql_firewall.sql_firewall_statements; userid | queryid | query
    | calls --------+------------+---------------------------------+------- 10 | 3294787656
    | select * from k1 where uid = ?; | 4 (1 row)
    ```

  - sql_firewall.sql_firewall_stat
    展示预警模式下的警告数量（sql_warning）和防火墙模式下的错误数量（sql_error）。

    ```
    postgres=# select * from sql_firewall.sql_firewall_stat; sql_warning | sql_error ------
    -------+----------- 2 | 1 (1 row)
    ```

示例

```
--预警模式 postgres=# select * from sql_firewall.sql_firewall_statements; WARNING: Prohibite
d SQL statement userid | queryid | query | calls --------+------------+-------------------
-------------+------- 10 | 3294787656 | select * from k1 where uid = 1; | 1 (1 row) postgre
s=# select * from k1 where uid = 1; uid | uname -----+------------- 1 | Park Gyu-ri (1 row)
postgres=# select * from k1 where uid = 3; uid | uname -----+----------- 3 | Goo Ha-ra (1 r
ow) postgres=# select * from k1 where uid = 3 or 1 = 1; WARNING: Prohibited SQL statement u
id | uname -----+---------------- 1 | Park Gyu-ri 2 | Nicole Jung 3 | Goo Ha-ra 4 | Han Seu
ng-yeon 5 | Kang Ji-young (5 rows) --防火墙模式 postgres=# select * from k1 where uid = 3; u
id | uname -----+----------- 3 | Goo Ha-ra (1 row) postgres=# select * from k1 where uid =
3 or 1 = 1; ERROR: Prohibited SQL statement postgres=#
```

6.11. 并发控制（pg_concurrency_control）

RDS PostgreSQL提供pg_concurrency_control插件，用于对SQL进行并发控制。

前提条件

RDS PostgreSQL实例版本为PostgreSQL 10或11。

参数说明

说明 暂不支持在控制台修改以下参数，您可以 提交工单 申请修改。

参数	默认值	说明
pg_concurrency_control.query_concurrency	0	设置Select类型SQL并发控制的排队个数限制。取值范围为0~1024，默认值为0，表示关闭Select类型SQL并发控制。
pg_concurrency_control.bigquery_concurrency	0	设置慢查询类型SQL并发控制的排队个数限制。取值范围为0~1024，默认值为0，表示关闭慢查询类型SQL并发控制。 您可以使用 `hint "/*+bigsql*/"` 的方式来指定一个请求为慢查询。例如： `/*+bigsql*/select * from test;` 此时 `select * from test;` 被认为是一个慢查询。
pg_concurrency_control.transaction_concurrency	0	设置事务块并发控制的排队个数限制。取值范围为0~1024，默认值为0，表示关闭事务块并发控制。
pg_concurrency_control.autocommit_concurrency	0	设置DML类型SQL并发控制的排队个数限制。取值范围为0~1024，默认值为0，表示关闭DML类型SQL并发控制。
pg_concurrency_control.control_timeout	1秒	设置Select类型SQL、DML类型SQL和事务块的并发控制排队等待时间。最小值为30毫秒（ms），最大值为3秒（s）。
pg_concurrency_control.bigsql_control_timeout	1秒	设置慢查询并发控制排队等待时间。最小值为30毫秒（ms），最大值为3秒（s）。
pg_concurrency_control.timeout_action	TCC_break	设置Select类型SQL、DML类型SQL和事务块的并发控制等待超时后的行为。取值： • TCC_break：跳过等待继续执行。 • TCC_rollback：超时后报错，事务回滚。 • TCC_wait：超时后重置时间戳继续等待。
pg_concurrency_control.bigsql_timeout_action	TCC_wait	设置慢查询并发控制等待超时后的行为。取值： • TCC_break：跳过等待继续执行。 • TCC_rollback：超时后报错，事务回滚。 • TCC_wait：超时后重置时间戳继续等待。

使用方法

1. 使用如下命令创建插件：

```
create extension pg_concurrency_control;
```

2. 设置并发控制排队个数限制大于0，即开启插件排队功能。

   例如设置 pg_concurrency_control.query_concurrency =10，即开启Select类型SQL并发控制功能。其余功能开启方式类似。

   说明 暂不支持在控制台修改参数，您可以 提交工单 申请修改。

使用示例

对自定义SQL操作进行并发控制。

1. 使用如下命令查看排队视图：

```
select * from pg_concurrency_control_status();
```

系统输出类似如下结果：

```
 autocommit_count | bigquery_count | query_count | transaction_count ------------------
+----------------+-------------+------------------- 0 | 0 | 0 | 0 (1 row)
```

2. 设置 pg_concurrency_control.query_concurrency 大于0，例如10。
3. 执行慢查询语句：

```
/*+ bigsql */ select pg_sleep(10);
```

4. 再次查看排队视图：

```
select * from pg_concurrency_control_status();
```

系统输出类似如下结果：

```
 autocommit_count | bigquery_count | query_count | transaction_count ------------------
+----------------+-------------+------------------- 0 | 1 | 0 | 0 (1 row)
```

> **说明** 慢查询执行完毕后，排队信息会自动清空。

6.12. 高速全文检索（RUM）

RDS PostgreSQL提供RUM插件，实现高速全文检索。

前提条件

RDS PostgreSQL版本如下：

- PostgreSQL 13
- PostgreSQL 12
- PostgreSQL 11
- PostgreSQL 10

背景信息

GIN索引（通用倒排索引）支持通过tsvector和tsquery两种数据类型进行全文检索，但是有如下几个问题：

- 排序慢
  需要有关词汇的位置信息才能进行排序。GIN索引不存储词汇的位置，因此在索引扫描之后，需要额外的扫描来检索词汇位置。
- 短语查询慢
  GIN索引需要位置信息来执行短语搜索。
- 时间戳排序慢
  GIN索引无法在带有词素的索引中存储一些相关信息，因此需要执行额外的扫描。

基于GIN索引，RDS PostgreSQL提供RUM插件，在RUM索引中存储额外的信息（词汇位置或时间戳的位置信息）来解决以上问题。

RUM索引的缺点是构建和插入时间比GIN索引慢。 这是因为需要存储除密钥之外的其他信息，并且RUM使用通用WAL记录。

通用的操作符

RUM模块提供以下操作符。

操作符	返回值数据类型	描述
tsvector <=> tsquery	float4	返回tsvector与tsquery之间的距离。
timestamp <=> timestamp	float8	返回两个时间戳之间的距离。
timestamp <=| timestamp	float8	只返回左侧时间戳的距离。
timestamp |=> timestamp	float8	只返回右侧时间戳的距离。

> **说明** 后三种操作符也适用于这些数据类型：timestamptz、int2、int4、int8、float4、float8、money、oid。

下文将介绍RUM的各个函数。

rum_tsvector_ops

- 适用数据类型：tsvector
- 说明：该函数存储带有位置信息的tsvector词组，支持按 `<=>` 运算符排序和前缀搜索。
- 示例
  i. 建立一个表格，命令如下：

```
CREATE TABLE test_rum(t text, a tsvector);
CREATE TRIGGER tsvectorupdate
BEFORE UPDATE OR INSERT ON test_rum
FOR EACH ROW EXECUTE PROCEDURE tsvector_update_trigger('a', 'pg_catalog.english', 't'
);
INSERT INTO test_rum(t) VALUES ('The situation is most beautiful');
INSERT INTO test_rum(t) VALUES ('It is a beautiful');
INSERT INTO test_rum(t) VALUES ('It looks like a beautiful place');
```

  ii. 创建RUM插件，命令如下：

```
CREATE EXTENSION rum;
```

  iii. 创建新的索引，命令如下：

```
CREATE INDEX rumidx ON test_rum USING rum (a rum_tsvector_ops);
```

  iv. 测试执行如下两种查询：

```
SELECT t, a <=> to_tsquery('english', 'beautiful | place') AS rank
    FROM test_rum
    WHERE a @@ to_tsquery('english', 'beautiful | place')
    ORDER BY a <=> to_tsquery('english', 'beautiful | place');
```

返回如下输出结果：

```
                t                 |  rank
----------------------------------+---------
 It looks like a beautiful place | 8.22467
 The situation is most beautiful | 16.4493
 It is a beautiful               | 16.4493
(3 rows)
```

```
SELECT t, a <=> to_tsquery('english', 'place | situation') AS rank
    FROM test_rum
    WHERE a @@ to_tsquery('english', 'place | situation')
    ORDER BY a <=> to_tsquery('english', 'place | situation');
```

返回如下输出结果：

```
                t                 |  rank
----------------------------------+---------
 The situation is most beautiful | 16.4493
 It looks like a beautiful place | 16.4493
(2 rows)
```

rum_tsvector_hash_ops

- 适用数据类型：tsvector
- 说明：该函数存储tsvector词组的哈希值和位置信息。支持按 `<=>` 运算符排序， 但不支持前缀搜索。

rum_TYPE_ops

- 适用数据类型：
  - `<` 、 `<=` 、 `=` 、 `>=` 、 `>` 和 `<=>` 操作支持int2、int4、int8、float4、float8、money、oid、time、timetz、date、interval、macaddr、inet、cidr、text、varchar、char、bytea、bit、varbit、numeric、timestamp、timestamptz。
  - `<=|` 和 `|=>` 支持int2、int4、int8、float4、float8、money、oid、timestamp、timestamptz。
- 说明：该函数可以与rum_tsvector_addon_ops、rum_tsvector_hash_addon_ops和rum_anyarray_addon_ops函数一起使用。

rum_tsvector_addon_ops

- 适用数据类型：tsvector
- 说明：该函数存储tsvector词法，以及模块字段支持的任何词法。
- 示例
  i. 建立一个表格和索引，命令如下：

```
CREATE TABLE tsts (id int, t tsvector, d timestamp);
\copy tsts from 'rum/data/tsts.data'
CREATE INDEX tsts_idx ON tsts USING rum (t rum_tsvector_addon_ops, d)
    WITH (attach = 'd', to = 't');
```

ii. 执行如下计划命令：

```
EXPLAIN (costs off)
    SELECT id, d, d <=> '2016-05-16 14:21:25' FROM tsts WHERE t @@ 'wr&qh' ORDER BY d
<=> '2016-05-16 14:21:25' LIMIT 5;
```

返回如下输出结果：

```
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 Limit
   ->  Index Scan using tsts_idx on tsts
         Index Cond: (t @@ '''wr'' & ''qh'''::tsquery)
         Order By: (d <=> 'Mon May 16 14:21:25 2016'::timestamp without time zone)
(4 rows)
```

iii. 执行如下查询命令：

```
SELECT id, d, d <=> '2016-05-16 14:21:25' FROM tsts WHERE t @@ 'wr&qh' ORDER BY d <=>
'2016-05-16 14:21:25' LIMIT 5;
```

返回如下输出结果：

```
 id  |                d                 |   ?column?
-----+----------------------------------+---------------
 355 | Mon May 16 14:21:22.326724 2016 |      2.673276
 354 | Mon May 16 13:21:22.326724 2016 |   3602.673276
 371 | Tue May 17 06:21:22.326724 2016 |  57597.326724
 406 | Wed May 18 17:21:22.326724 2016 | 183597.326724
 415 | Thu May 19 02:21:22.326724 2016 | 215997.326724
(5 rows)
```

说明 当前RUM在使用按引用传递附加信息进行排序来创建索引时可能有缺陷。这是由于后缀树具有固定长度的右边界和固定长度的无子节点后缀项，所以不允许创建此类索引。

rum_tsvector_hash_addon_ops

- 适用数据类型：tsvector
- 说明：该函数存储tsvector词库的哈希值以及任何支持模块的字段，不支持前缀搜索。

rum_tsquery_ops

- 适用数据类型：tsquery
- 说明：在其他信息中存储查询树的分支。
- 示例
  i. 建立一个表格和索引，命令如下：

```
CREATE TABLE test_array (i int2[]);
INSERT INTO test_array VALUES ('{}'), ('{0}'), ('{1,2,3,4}'), ('{1,2,3}'), ('{1,2}'),
('{1}');
CREATE INDEX idx_array ON test_array USING rum (i rum_anyarray_ops);
```

ii. 执行如下计划命令：

```
SET enable_seqscan TO off;
EXPLAIN (COSTS OFF) SELECT * FROM test_array WHERE i && '{1}' ORDER BY i <=> '{1}' AS
C;
```

返回如下输出结果：

```
                QUERY PLAN
-----------------------------------------
 Index Scan using idx_array on test_array
   Index Cond: (i && '{1}'::smallint[])
   Order By: (i <=> '{1}'::smallint[])
(3 rows)
```

iii. 执行如下查询命令：

```
SELECT * FROM test_array WHERE i && '{1}' ORDER BY i <=> '{1}' ASC;
```

返回如下输出结果：

```
     i
-----------
 {1}
 {1,2}
 {1,2,3}
 {1,2,3,4}
(4 rows)
```

rum_anyarray_ops

- 适用数据类型：anyarray
- 说明：该函数存储具有数组长度的anyarrray元素。支持运算符 `&&` 、 `@>` 、 `<@` 、 `=` 、 `%` ，支持按 `<=>` 运算符排序。
- 示例

  i. 建立一个表格和索引，命令如下：

```
CREATE TABLE test_array (i int2[]);
INSERT INTO test_array VALUES ('{}'), ('{0}'), ('{1,2,3,4}'), ('{1,2,3}'), ('{1,2}'),
('{1}');
CREATE INDEX idx_array ON test_array USING rum (i rum_anyarray_ops);
```

  ii. 执行如下计划命令：

```
SET enable_seqscan TO off;
EXPLAIN (COSTS OFF) SELECT * FROM test_array WHERE i && '{1}' ORDER BY i <=> '{1}' AS
C;
```

  返回如下输出结果：

```
                QUERY PLAN
-----------------------------------------
 Index Scan using idx_array on test_array
   Index Cond: (i && '{1}'::smallint[])
   Order By: (i <=> '{1}'::smallint[])
(3 rows)
```

iii. 执行如下查询命令：

```
SELECT * FROM test_array WHERE i && '{1}' ORDER BY i <=> '{1}' ASC;
```

返回如下输出结果：

```
     i
-----------
 {1}
 {1,2}
 {1,2,3}
 {1,2,3,4}
(4 rows)
```

rum_anyarray_addon_ops

- 适用数据类型：anyarray
- 说明：该函数存储anyarrray元素以及模块字段支持的任何元素。

6.13. 中文分词（zhparser）

RDS PostgreSQL支持zhparser插件，用于中文的全文检索。

前提条件

- RDS PostgreSQL实例需为以下版本之一：
  - PostgreSQL 13
  - PostgreSQL 12
  - PostgreSQL 11
  - PostgreSQL 10
  - PostgreSQL 9.4
- 使用该插件前，需要将zhparser加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为**shared_preload_libraries**参数添加**zhparser**。具体操作，请参见设置实例参数。

> **说明** 如果无法创建该插件，请先 升级内核小版本。

背景信息

PostgreSQL自带的parser插件适用于分词比较简单的语言（如英语），按照标点、空格切分语句即可获得有含义的词语，而中文比较复杂，词语之间没有空格分隔，长度也不固定，分词还和语义有关，因此parser不能用来做中文分词，建议您使用zhparser。

zhparser是PostgreSQL的中文分词插件，安装后可以使PostgreSQL支持中文的全文检索（Full Text Search）。

启用中文分词

可以使用下面的命令，启用中文分词：

```
CREATE EXTENSION zhparser;
CREATE TEXT SEARCH CONFIGURATION testzhcfg (PARSER = zhparser);
ALTER TEXT SEARCH CONFIGURATION testzhcfg ADD MAPPING FOR n,v,a,i,e,l WITH simple;
--可选的参数设定
alter role all set zhparser.multi_short=on;
--简单测试
SELECT * FROM ts_parse('zhparser', 'hello world! 2010年保障房建设在全国范围内获全面启动，从中央到地方纷纷加大了保障房的建设和投入力度。2011年，保障房进入了更大规模的建设阶段。住房城乡建设部党组书记、部长姜伟新去年底在全国住房城乡建设工作会议上表示，要继续推进保障性安居工程建设。');
SELECT to_tsvector('testzhcfg','“今年保障房新开工数量虽然有所下调，但实际的年度在建规模以及竣工规模会超以往年份，相对应的对资金的需求也会创历史纪录。”陈国强说。在他看来，与2011年相比，2012年的保障房建设在资金配套上的压力将更为严峻。');
SELECT to_tsquery('testzhcfg', '保障房资金压力');
```

利用分词进行全文索引的方法如下：

```
--为T1表的name字段创建全文索引
create index idx_t1 on t1 using gin (to_tsvector('zhcfg',upper(name) ));
--使用全文索引
 select * from t1 where to_tsvector('zhcfg',upper(t1.name)) @@ to_tsquery('zhcfg','(防火)')
;
```

自定义中文分词词典

自定义中文分词词典，示例如下：

```
-- 确实的分词结果
SELECT to_tsquery('testzhcfg', '保障房资金压力');
-- 往自定义分词词典里面插入新的分词
insert into pg_ts_custom_word values ('保障房资');
-- 使新的分词生效
select zhprs_sync_dict_xdb();
-- 退出此连接
\c
-- 重新查询，可以得到新的分词结果
SELECT to_tsquery('testzhcfg', '保障房资金压力');
```

使用自定义分词的注意事项如下：

- 最多支持一百万条自定义分词，超出部分不做处理，必须保证分词数量在这个范围之内。自定义分词与缺省的分词词典将共同产生作用。
- 每个词的最大长度为128字节，超出部分将会截取。
- 增删改分词之后必须执行 `select zhprs_sync_dict_xdb();` 并且重新建立连接才会生效。

6.14. 中文分词（pg_jieba）

RDS PostgreSQL支持pg_jieba插件，用于中文的全文搜索。

前提条件

- RDS PostgreSQL实例版本需要满足以下条件：
  - 实例大版本：10、11、12或13。
  - 内核小版本：大于等于20211130，如需升级内核小版本，请参见升级内核小版本。
- 使用该插件前，需要将pg_jieba加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为**shared_preload_libraries**参数添加**pg_jieba**。具体操作，请参见设置实例参数。

使用方法

- 创建插件

```
CREATE EXTENSION pg_jieba;
```

> **说明** 仅高权限账号可以执行此命令。

- 删除插件

```
DROP EXTENSION pg_jieba;
```

> **说明** 仅高权限账号可以执行此命令。

- 使用示例1：

```
select * from to_tsvector('jiebacfg', '小明硕士毕业于中国科学院计算所，后在日本京都大学深造'); t
o_tsvector ----------------------------------------------------------------------------------
--------------------------------- '中国科学院':5 '于':4 '后':8 '在':9 '小明':1 '日本京都大学':
10 '毕业':3 '深造':11 '硕士':2 '计算所':6 '，':7 (1 row)
```

- 使用示例2：

```
select * from to_tsvector('jiebacfg', '李小福是创新办主任也是云计算方面的专家'); to_tsvector -
---------------------------------------------------------------------------------------------
- '专家':11 '主任':5 '也':6 '云计算':8 '创新':3 '办':4 '方面':9 '是':2,7 '李小福':1 '的':10 (1
row)
```

6.15. 字符串相似性判断（fuzzystrmatch）

RDS PostgreSQL支持fuzzystrmatch插件，提供Soundex、Levenshtein、Metaphone和Double Metaphone四种功能，能够用来判断字符串之间的相似性和距离。

Soundex

Soundex功能可以将相似发音的名字转换成相同的代码并进行匹配，但是Soundex对于非英语名称效果较差。

Soundex功能提供如下函数：

```
soundex(text) returns text
difference(text, text) returns int
```

- soundex函数将一个字符串转换成它的Soundex代码（例如A550）。
- difference函数将两个字符串转换成它们的Soundex代码并报告两个代码匹配位置的数量。由于Soundex

代码具有四个字符，代码匹配位置数量的取值范围是0~4，0表示没有匹配，4表示完全匹配。

用法示例如下：

```
SELECT soundex('hello world!');
SELECT soundex('Anne'), soundex('Andrew'), difference('Anne', 'Andrew');
SELECT soundex('Anne'), soundex('Margaret'), difference('Anne', 'Margaret');
CREATE TABLE s (nm text);
INSERT INTO s VALUES ('john');
INSERT INTO s VALUES ('joan');
INSERT INTO s VALUES ('wobbly');
INSERT INTO s VALUES ('jack');
SELECT * FROM s WHERE soundex(nm) = soundex('john');
SELECT * FROM s WHERE difference(s.nm, 'john') > 2;
```

Levenshtein

Levenshtein功能可以计算两个字符串之间的编辑距离（Levenshtein距离）。

Levenshtein功能提供如下函数：

```
levenshtein(text source, text target, int ins_cost, int del_cost, int sub_cost) returns int
levenshtein(text source, text target) returns int
levenshtein_less_equal(text source, text target, int ins_cost, int del_cost, int sub_cost,
int max_d) returns int
levenshtein_less_equal(text source, text target, int max_d) returns int
```

参数说明如下。

参数	说明
source	第一个字符串，任意非空字符串， 最长支持255个字符。
target	第二个字符串，任意非空字符串， 最长支持255个字符。
ins_cost	字符插入的开销。
del_cost	字符删除的开销。
sub_cost	字符替换的开销。
max_d	最大Levenshtein距离。

说明 levenshtein_less_equal函数是Levenshtein函数的加速版本，在只计算小距离时使用：

- 如果实际的距离低于或等于max_d，levenshtein_less_equal返回精确值。
- 如果实际的距离大于max_d，levenshtein_less_equal返回大于max_d的值。
- 如果max_d为负值，levenshtein_less_equal函数和levenshtein函数相同。

用法示例如下：

```
SELECT levenshtein('GUMBO', 'GAMBOL');
SELECT levenshtein('GUMBO', 'GAMBOL', 2,1,1);
SELECT levenshtein_less_equal('extensive', 'exhaustive',2);
SELECT levenshtein_less_equal('extensive', 'exhaustive',4);
```

```
test=# SELECT levenshtein('GUMBO', 'GAMBOL');
 levenshtein
-------------
           2
(1 row)

test=# SELECT levenshtein('GUMBO', 'GAMBOL', 2,1,1);
 levenshtein
-------------
           3
(1 row)

test=# SELECT levenshtein_less_equal('extensive', 'exhaustive',2);
 levenshtein_less_equal
------------------------
                      3
(1 row)

test=# SELECT levenshtein_less_equal('extensive', 'exhaustive',4);
 levenshtein_less_equal
------------------------
                      4
(1 row)
```

Metaphone

Metaphone功能的思想和Soundex功能相同，构建一个输入字符串的表示代码，如果两个字符串具有相同的代码则认为它们相似。

Metaphone功能提供如下函数：

```
metaphone(text source, int max_output_length) returns text
```

参数说明如下。

参数	说明
source	任意非空字符串， 最长支持255个字符。
max_output_length	设置输出的metaphone代码的最大长度，如果超长，输出会被截断到这个长度。

用法示例如下：

```
SELECT metaphone('GUMBO', 4);
```

Double Metaphone

Double Metaphone功能能够为一个输入的字符串计算出两个相似的字符串，包括一个主要函数和一个次要函数。在大部分情况下它们是相同的，但是对于非英语名称它们可能有一点不同（取决于发音）。

Double Metaphone功能提供的主要和次要函数如下：

```
dmetaphone(text source) returns text
dmetaphone_alt(text source) returns text
```

用法示例如下：

```
select dmetaphone('gumbo');
select dmetaphone_alt('gumbo');
```

6.16. 定制执行计划（pg_hint_plan）

RDS PostgreSQL提供pg_hint_plan插件，可以通过特殊的注释语句提示，使PostgreSQL改变其既定的执行计划。

前提条件

- RDS PostgreSQL实例需为以下版本之一：
  - PostgreSQL 13
  - PostgreSQL 12
  - PostgreSQL 11
  - PostgreSQL 10
  - PostgreSQL 9.4
- 使用该插件前，需要将pg_hint_plan加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为shared_preload_libraries参数添加pg_hint_plan。具体操作，请参见设置实例参数。

> **说明** 如果无法创建该插件，请先 升级内核小版本。

背景信息

PostgreSQL使用基于代价的优化器，优化路线使用统计数据而非固定的规则。对于一条SQL语句，优化器会去评估每一种可能的代价并最终选择代价最低的去执行，优化器会尽力去选择最好的执行计划，但是由于其并不了解数据中可能存在的一些内在连接关系，这些执行计划可能并不完美。使用pg_hint_plan插件以特殊的注释形式来提示SQL语句应该如何执行，可以优化执行计划。

注释提示

pg_hint_plan的注释以 `/*+` 开头，以 `*/` 结束。提示语句包括提示名和接下来括号包裹的参数，以空格作为分界。为了可读性，每一个提示语句都可以重新换行。

示例

HashJoin作为连接方法，并且使用序列扫描SeqScan来扫描表pgbench_accounts。

```
 /*+ HashJoin(a b) >SeqScan(a) */ EXPLAIN SELECT * FROM pgbench_branches b JOIN pgbench_acc
ounts a ON b.bid = a.bid ORDER BY a.aid;
```

返回如下结果：

```
 QUERY PLAN ----------------------------------------------------------------------------
-------- Sort (cost=31465.84..31715.84 rows=100000 width=197) Sort Key: a.aid -> Hash Join
(cost=1.02..4016.02 rows=100000 width=197) Hash Cond: (a.bid = b.bid) -> Seq Scan on pgbenc
h_accounts a (cost=0.00..2640.00 rows=100000 width=97) -> Hash (cost=1.01..1.01 rows=1 widt
h=100) -> Seq Scan on pgbench_branches b (cost=0.00..1.01 rows=1 width=100) (7 rows)
```

提示表

虽然可以使用注释提示的方式对SQL语句进行提示，但是当SQL语句不可编辑时，这种提示方式就很不方便。对于这种情况，可以将提示放在一张特殊的表hint_plan.hints中。这个表包含了下列字段。

> **说明** 创建pg_hint_plan插件的用户默认拥有提示表的权限，提示表的优先权高于注释提示。

字段	描述
id	提示ID号，唯一且自动填充。
norm_query_string	与要提示的查询匹配的模式。查询中的常量必须替换为 `?` 。空格在模式中很重要。
application_name	应用会话的名称，置空表示任意应用。
hints	提示语句，不需要注释标记。

示例命令如下：

```
INSERT INTO hint_plan.hints(norm_query_string, application_name, hints) VALUES ( 'EXPLAIN (
COSTS false) SELECT * FROM t1 WHERE t1.id = ?;', '', 'SeqScan(t1)' ); INSERT 0 1 postgres=#
UPDATE hint_plan.hints postgres-# SET hints = 'IndexScan(t1)' postgres-# WHERE id = 1; UPDA
TE 1 postgres=# DELETE FROM hint_plan.hints postgres-# WHERE id = 1; DELETE 1
```

提示类型

根据提示短语影响执行计划的方式，可以被分为如下六类：

- 扫描方法提示
  扫描方法提示对目标表强制执行特定的扫描方法。pg_hint_plan通过表格的别名（如果存在的话）来识别目标表。扫描方法是 `SeqScan` 、 `IndexScan` 、 `NoSeqScan` 等。
  扫描提示对普通表、继承表、无日志表、临时表和系统表有效，对外部表、表函数、常量值语句、通用表达式、视图和子查询无效。
  示例命令如下：

  ```
  /*+ SeqScan(t1) IndexScan(t2 t2_pkey) */ SELECT * FROM table1 t1 JOIN table table2 t2 ON
  (t1.key = t2.key);
  ```

- 连接方法提示
  连接方法提示强制指定相关表格聚合在一起的方法。
  连接方法提示对普通表 、继承表、无日志表、临时表、外部表、系统表、表函数、常量值命令和通用表达式有效，对视图和子查询无效。
- 连接顺序提示
  连接顺序提示指定两个或多个表的连接顺序。这里有两种强制指定方法：
  - 强制执行特定的连接顺序，但不限制每个连接级别的方向。

- 强制连接方向。

  示例命令如下：

```
/*+ NestLoop(t1 t2) MergeJoin(t1 t2 t3) Leading(t1 t2 t3) */ SELECT * FROM table1 t1 JOIN table table2 t2 ON (t1.key = t2.key) JOIN table table3 t3 ON (t2.key = t3.key);
```

- 行号纠正提示
  行号纠正提示会纠正由于计划器限制而导致的行号错误。
  示例命令如下：

```
/*+ Rows(a b #10) */ SELECT... ; //设置连接结果的行号为10。 /*+ Rows(a b +10) */ SELECT... ; //行号增加10。 /*+ Rows(a b -10) */ SELECT... ; //行号减去10。 /*+ Rows(a b *10) */ SELECT... ; //将行号扩大至原来的10倍。
```

- 并行执行提示
  并行提示会指定并行的执行计划。
  并行执行提示对普通表、继承表、无日志表和系统表有效，对外部表、常量从句、通用表达式、视图和子查询无效。视图的内部表可以通过其真实名称或别名指定目标对象。
  下面两个示例说明在每个表上执行查询的方式不同：
  - 方式一

```
explain /*+ Parallel(c1 3 hard) Parallel(c2 5 hard) */ SELECT c2.a FROM c1 JOIN c2 ON (c1.a = c2.a);
```

    返回如下结果：

```
 QUERY PLAN ------------------------------------------------------------------------------------ Hash Join (cost=2.86..11406.38 rows=101 width=4) Hash Cond: (c1.a = c2.a) -> Gather (cost=0.00..7652.13 rows=1000101 width=4) Workers Planned: 3 -> Parallel Seq Scan on c1 (cost=0.00..7652.13 rows=322613 width=4) -> Hash (cost=1.59..1.59 rows=101 width=4) -> Gather (cost=0.00..1.59 rows=101 width=4) Workers Planned: 5 -> Parallel Seq Scan on c2 (cost=0.00..1.59 rows=59 width=4)
```

  - 方式二

```
EXPLAIN /*+ Parallel(tl 5 hard) */ SELECT sum(a) FROM tl;
```

    返回如下结果：

```
 QUERY PLAN ------------------------------------------------------------------------------------ Finalize Aggregate (cost=693.02..693.03 rows=1 width=8) -> Gather (cost=693.00..693.01 rows=5 width=8) Workers Planned: 5 -> Partial Aggregate (cost=693.00..693.01 rows=1 width=8) -> Parallel Seq Scan on tl (cost=0.00..643.00 rows=20000 width=4)
```

- 设置临时GUC参数
  在计划的时候临时改变GUC参数。执行计划中的GUC参数会有预期的效果，除非提示语句与其他计划冲突。同样的GUC参数设置以最后一次为准。
  示例命令如下：

```
/*+ Set(random_page_cost 2.0) */ SELECT * FROM table1 t1 WHERE key = 'value';
```

所有提示支持的格式如下表。

类别	格式	说明
扫描方法提示	SeqScan(table)	强制序列扫描。
	TidScan(table)	强制TID扫描。
	IndexScan(table[index...])	强制索引扫描，可以指定某个索引。
	IndexOnlyScan(table[index...])	强制仅使用索引扫描，可以指定某个索引。
	BitmapScan(table[index...])	强制使用Bitmap扫描。
	NoSeqScan(table)	强制不使用序列扫描。
	NoTidScan(table)	强制不使用TID扫描。
	NoIndexScan(table)	强制不使用索引扫描。
	NoIndexOnlyScan(table)	强制不使用索引扫描，仅扫描表。
	NoBitmapScan(table)	强制不使用Bitmap扫描。
连接方法提示	NestLoop(table table[table...])	强制使用嵌套循环连接。
	HashJoin(table table[table...])	强制使用散列连接。
	MergeJoin(table table[table...])	强势使用合并连接。
	NoNestLoop(table table[table...])	强制不使用嵌套循环连接。
	NoHashJoin(table table[table...])	强制不使用散列连接。
	NoMergeJoin(table table[table...])	强制不使用合并连接。
连接顺序提示	Leading(table table[table...])	强制连接的顺序。
	Leading(<join pair>)	强制连接的顺序和方向。
行号纠正提示	Rows(table table[table...] correction)	纠正由指定表组成的联接结果的行号。可用的校正方法为绝对值 `#<n>` 、加法 `+ <n>` 、减法 `-<n>` 和乘法 `* <n>` 。 `<n>` 是strtod函数可以读取的数字。
并行执行提示	Parallel(table <# of workers> [soft|hard])	强制或禁止并行执行指定表。 `<worker#>` 是所需的并行工作程序数量，0表示禁止并行执行。 第三个参数如果是soft（默认），表示仅更改max_parallel_workers_per_gather并将其他所有内容留给计划器选择； 如果是hard，表示所有相关参数都会被强制指定。
设置临时GUC参数	Set(GUC-param value)	规划器运行时，将GUC参数设置为该值。

更多pg_hint_plan的介绍请参见PostgreSQL官方文档。

6.17. 清理表空间（pg_repack）

RDS PostgreSQL支持通过插件pg_repack在线清理表空间，有效解决因对全表大量更新等操作引起的表膨胀问题。pg_repack无需获取排它锁，相比CLUSTER或VACUUM FULL更加轻量化。

前提条件

RDS实例需要满足以下条件：

- 实例大版本为PostgreSQL 10、11、12或13。
- 实例内核小版本为20210331及以上。查看和升级内核小版本，请参见升级内核小版本。

注意事项

- pg_repack需要额外的存储空间。全表repack时，剩余存储空间大小需要至少是待repack表大小的2倍。
- pg_repack无法操作临时表。
- pg_repack无法操作GiST索引。
- pg_repack运行时无法对repack操作中的表执行DDL。pg_repack会持有ACCESS SHARE锁， 禁止DDL执行。
- 重建表和索引时会占用较多的磁盘IO，使用时请提前评估是否影响业务。以ESSD PL1云盘为例，repack 100GB的表时，会达到IO吞吐上限（250MB/s）。

功能介绍

pg_repack插件支持全表repack和索引repack。

- 对整表数据进行repack的实现原理如下：
  i. 创建日志表记录对原表的变更。
  ii. 在原表创建触发器，将原表的INSERT、UPDATE和DELETE操作记录到日志表中。
  iii. 创建新表，与原表的行列相同。
  iv. 在新表中创建索引。
  v. 将日志表里的变更应用到新表。
  vi. 在system catalog交换新旧表。
  vii. 删除旧表。

  > **说明** pg_repack会在第i、ii步和第vi、vii步短暂持有ACCESS EXCLUSIVE锁。其余步骤pg_repack只需要持有原表的ACCESS SHARE锁，不影响原表的INSERT、UPDATE和DELETE。

- 对表中索引进行repack的实现原理如下：
  i. 以CONCURRENTLY方式创建新索引。
  ii. 在system catalog交换新旧索引。
  iii. 删除旧索引。

插件开启与关闭

- 开启插件。

```
CREATE EXTENSION pg_repack;
```

- 关闭插件。

```
DROP EXTENSION pg_repack;
```

安装客户端

pg_repack插件需要配合客户端使用，以ECS Alibaba Cloud Linux 3.2104及以上版本为例，参考以下命令：

1. 安装环境依赖。

```
yum install postgresql* redhat-rpm-config libpq* openssl-devel readline-devel -y
```

2. 添加环境变量。

```
export PATH=$PATH:/usr/lib64/pgsql/postgresql-12/bin
```

3. 下载pg_repack客户端并编译安装。

```
wget https://github.com/reorg/pg_repack/archive/refs/tags/ver_1.4.6.tar.gz tar zxvf ver
_1.4.6.tar.gz cd pg_repack-ver_1.4.6 make && make install
```

使用示例

```
-- 检查但不执行：--dry-run $ pg_repack --dry-run --no-superuser-check --echo --no-order -h 连
接串 -p 端口 -d database1 -U user --table schema1.table1 -- 检查并执行 $ pg_repack --no-super
user-check --echo --no-order -h 连接串 -p 端口 -d database1 -U user --table schema1.table1
```

常见问题

Q：报错“ERROR: pg_repack failed with error: You must be a superuser to use pg_repack”。

A：请使用-k或--no-superuser-check，避免没有权限。

相关参考

pg_repack的详细说明可参考pg_repack官方帮助文档。

6.18. 逻辑复制（pglogical）

pglogical是一个完全作为PostgreSQL扩展实现的逻辑复制系统，为PostgreSQL数据库提供了逻辑流复制发布和订阅的功能。

前提条件

- RDS PostgreSQL实例需为以下版本之一：
  - PostgreSQL 14
  - PostgreSQL 13
  - PostgreSQL 12
  - PostgreSQL 11
  - PostgreSQL 10
- 使用该插件前，需要将pglogical加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为**shared_preload_libraries**参数添加**pglogical**。具体操作，请参见设置实例参数。

背景信息

pglogical是PostgreSQL的拓展模块，使用发布/订阅模型复制数据，使选择性复制更加高效。提供比Slony、Bucardo或Londiste更快的复制速度，支持跨版本升级。支持多场景复制，具体包括：

- 数据库大版本升级。
- 完整的数据库复制。
- 利用复制集，选择性地筛选表、行、列。
- 可从多个上游服务器，做数据的聚集和合并。

使用插件

- 开启pglogical。

```
CREATE EXTENSION pglogical;
```

- 关闭pglogical。

```
DROP EXTENSION pglogical;
```

快速配置逻辑流复制

1. 在发布端创建发布节点。

> **说明** 发布节点中 `host` 固定配置为127.0.0.1， `port` 固定配置为3002。

```
SELECT pglogical.create_node( node_name := 'provider', dsn := 'host=127.0.0.1 port=3002
dbname=test user=provider_user password=provider_pass' );
```

2. 配置复制集。
   将public中的所有表添加到default复制集。

```
SELECT pglogical.replication_set_add_all_tables('default', ARRAY['public']);
```

> **说明**
> - 复制集用来控制将发布节点数据库中的哪些表以及这些表上的哪些操作发布给订阅者。
> - default复制集表示发布所有表及这些表的所有操作。更多复制集定义请参见pglogical官方文档。

3. 在订阅端创建订阅节点。

> **说明** 订阅节点中 `host` 固定配置为127.0.0.1， `port` 固定配置为3002。

```
SELECT pglogical.create_node( node_name := 'subscriber', dsn := 'host=127.0.0.1 port=30
02 dbname=test user=subscriber_user password=subscriber_pass' );
```

4. 在订阅端创建订阅。

> **说明** 订阅中 `host` 必须配置为发布端内网地址， `port` 配置为发布端内网端口。

```
SELECT pglogical.create_subscription( subscription_name := 'subscription', provider_dsn
:= 'host=<provider内网地址> port=<provider内网端口> dbname=test user=provider_user passwo
rd=provider_pass' );
```

创建订阅后，将在后台启动同步和复制流程。

相关参考

pglogical的详细说明，请参见pglogical官方文档。

6.19. 审计日志（pgAudit）

RDS PostgreSQL支持pgAudit插件，为您提供生成审计日志的能力，这些日志通常需要符合政府、金融或ISO 认证。通过审计日志，您还可以对数据库进行故障分析、行为分析等操作，有效帮助您获取数据的执行情况。

前提条件

- RDS PostgreSQL实例需满足以下条件：
  - 实例大版本为PostgreSQL 10、11、12、13或14。
  - 实例内核小版本为20210531及以上。查看和升级内核小版本，请参见升级内核小版本。
- 使用该插件前，需要将pgaudit加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为shared_preload_libraries参数添加pgaudit 。具体操作，请参见设置实例参数。

注意事项

- 根据设置的不同，pgAudit会生成大量日志。使用前请评估使用审计日志的对象，以避免日志过多导致磁盘空间不足。
- 对象重命名后，审计日志会产生在重命名后的对象下。

开启或关闭插件

- 开启pgAudit。

```
CREATE EXTENSION pgaudit;
```

- 关闭pgAudit。

```
DROP EXTENSION pgaudit;
```

配置审计日志

开启pgAudit插件后，您需要通过如下方式配置审计日志：

- 会话审计日志：配置 pgaudit.log 参数来指定会话审计日志记录的语句类别，会话审计日志将会记录被执行的所有此类语句。
- 对象审计日志：配置 pgaudit.role 参数定义用于审计日志记录的角色。当审计角色对执行的命令具有权限或从另一个角色继承权限时，将记录表、视图等。

配置会话审计日志

```
SET pgaudit.log = 'read, ddl';
```

> **说明** 本示例以read和ddl为例，记录SELECT和DDL语句。具体的语句类别划分，请参见 pgAudit官方文档。

配置对象审计日志

```
SET pgaudit.role = 'auditor'; GTANT SELECT, DELETE ON public.account TO auditor;
```

> **说明** 本示例中以角色auditor为例，使角色auditor具有表account的SELECT和DELETE权限。对象审计日志将会记录auditor在表account中执行的所有SELECT或DELETE语句。

相关参考

pgAudit的详细说明，请参考pgAudit官方文档。

6.20. 调试存储过程（pldebugger）

RDS PostgreSQL提供pldebugger插件，用于调试存储过程。

背景信息

RDS PostgreSQL支持多种存储过程语言，例如plpgsql、plpython、plperl、pltcl等等。用户可以使用这些存储过程语言，创建对应的函数或存储过程。

前提条件

- RDS PostgreSQL实例版本需要满足以下条件：
  - 实例大版本：10、11、12或13。
  - 内核小版本：大于等于20211130，如需升级内核小版本，请参见升级内核小版本。
- 使用该插件前，需要将plugin_debugger加入到 shared_preload_libraries 参数中。
  您可以使用RDS PostgreSQL参数设置功能，为**shared_preload_libraries**参数添加**plugin_debugger**。具体操作，请参见设置实例参数。
- 客户端pgAdmin4请确保使用版本大于pgAdmin 4 v4.19（包含pgAdmin 4 v4.19）。pgAdmin4下载链接请参见pgAdmin 4。

使用方法

- 创建插件

  ```
  CREATE EXTENSION pldbgapi;
  ```

  > **说明** 仅高权限账号可以执行此命令。

- 删除插件

  ```
  DROP EXTENSION pldbgapi;
  ```

  > **说明** 仅高权限账号可以执行此命令。

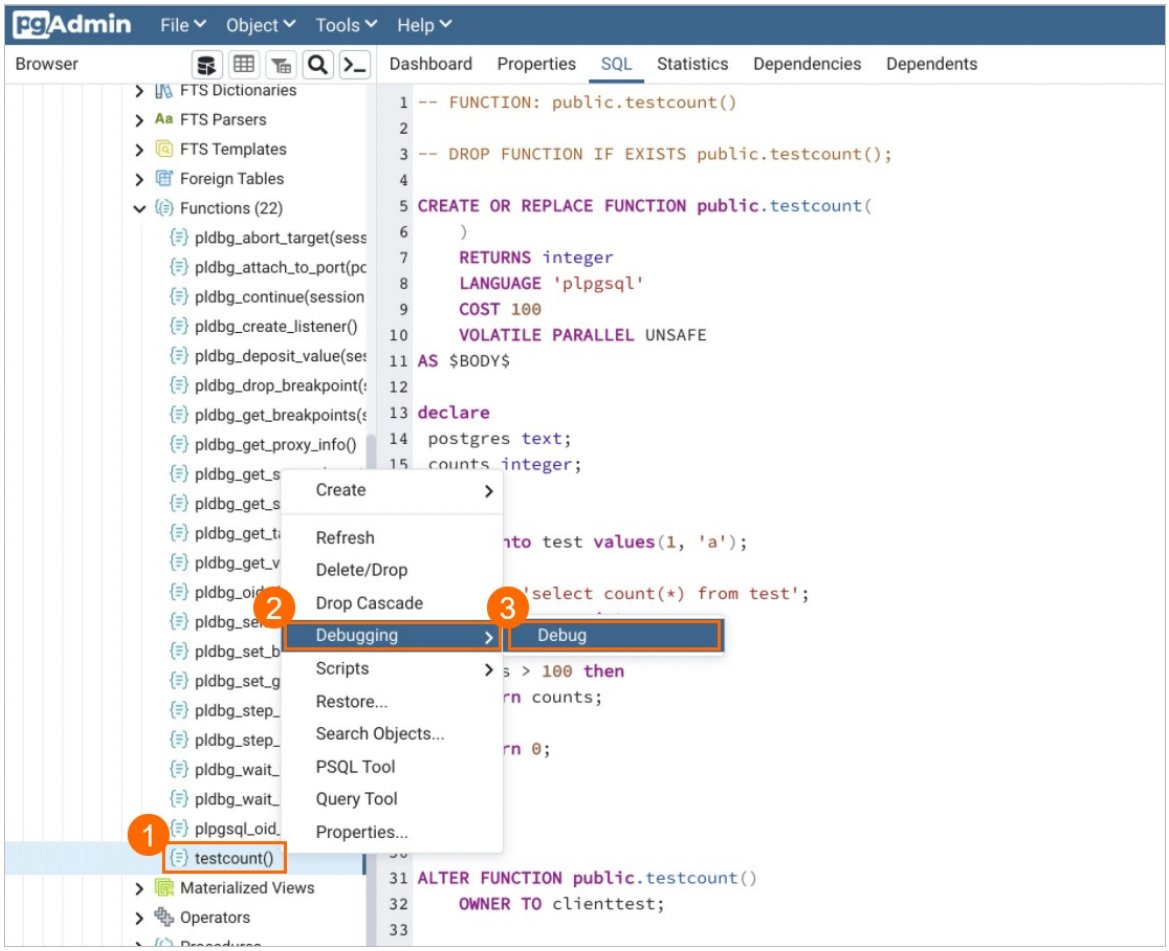
调试示例

1. 使用pgAdmin客户端连接RDS PostgreSQL数据库。连接方法，请参见连接PostgreSQL实例。
2. 创建测试数据库和存储过程。

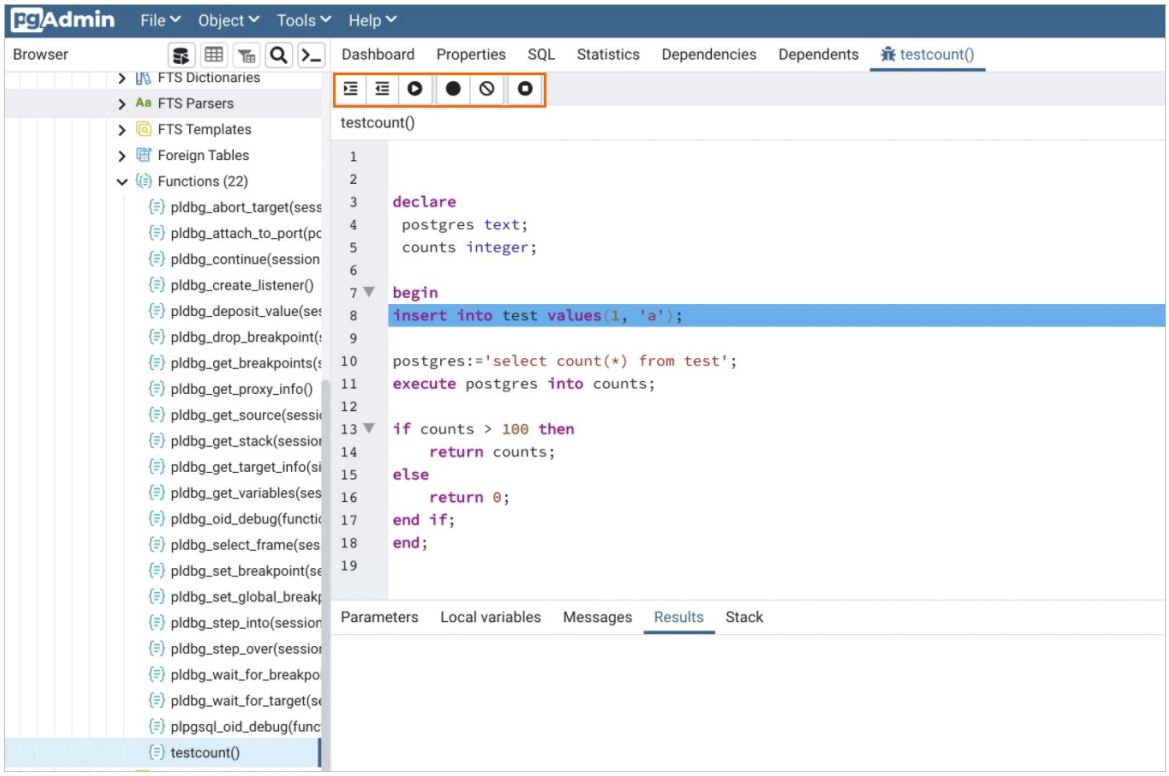
   示例SQL如下：

```
CREATE TABLE test( id int, name VARCHAR(50));

         CREATE OR REPLACE FUNCTION public.testcount()
         RETURNS integer AS $$ DECLARE postgres text; counts integer; BEGIN INSERT INTO
test VALUES(1, 'a'); postgres:='SELECT COUNT(*) FROM test';

         EXECUTE postgres INTO counts;
         IF counts > 100 THEN

          RETURN counts;
         ELSE RETURN 0; END IF; END; $$ language plpgsql;
```

3. 右键选择待调试的 函数。



4. 在pgAdmin界面右边的函数调试框中，您可以对目标函数进行单步调试，包含step into/over，continue，加设断点，停止等操作。底部有调试过程中显示的局部变量信息，调试结果与函数堆栈。



6.21. 索引推荐（index_adviser）

索引推荐（index_adviser）可以帮助您确定应该对哪些列编制索引，以提高指定工作负载中的性能。索引推荐仅识别B树索引类型（单列或复合），不识别可提高性能的其他索引类型，例如GIN、GiST、Hash等。

前提条件

RDS PostgreSQL实例内核小版本升级到20220130。查看和升级内核小版本，请参见 升级内核小版本。

索引推荐（index_adviser）组件

执行创建索引推荐插件语句时，同时会创建index_advisory表、show_index_advisory()函数和select_index_advisory视图。

组件	说明
index_advisory	此表在创建索引推荐（index_adviser）插件时自动创建，将索引编制建议记录在index_advisory表中。
show_index_advisory()	show_index_advisory()是一个PL/pgSQL函数，它解释并显示特定的索引推荐会话（由其后端进程ID标识）期间提出的建议。
select_index_advisory	索引推荐（index_adviser）根据查询分析期间存储在index_advisory表中的信息，创建select_index_advisory视图。该视图所生成输出的格式与show_index_advisory()函数的相同，显示所有索引推荐会话期间提出的建议。

使用方法

1. 创建index_adviser插件

```
postgres=# create extension index_adviser; CREATE EXTENSION
```

2. 加载index_adviser插件

```
postgres=# LOAD 'index_adviser'; LOAD
```

> **说明** 以上加载语句只对当前会话有效。如果您希望所有会话都默认加载index_adviser，您需要配置shared_preload_libraries参数并重启实例（会对性能有一些影响），配置方法如下：
>
> ```
> shared_preload_libraries='index_adviser'
> ```

使用示例

- 创建表

```
CREATE TABLE t( a INT, b INT ); INSERT INTO t SELECT s, 99999 - s FROM generate_series(0, 99999) AS s; ANALYZE t; 所生成的表包含以下各行： a | b -------+------- 0 | 99999 1 | 99998 2 | 99997 3 | 99996 . . . 99997 | 2 99998 | 1 99999 | 0
```

- 查询单条SQL建议
  - 如果您希望索引推荐分析查询并提出索引编制建议但不实际执行查询，请将EXPLAIN关键字作为SQL语句的前缀，示例如下：

```
postgres=# EXPLAIN SELECT * FROM t WHERE a < 10000; QUERY PLAN ------------------------
------------------------------------------------------------ Seq Scan on t (cost=0.00..169
3.00 rows=9983 width=8) Filter: (a < 10000) Result (cost=0.00..0.00 rows=0 width=0) One
-Time Filter: '** plan (using Index Adviser) **'::text -> Index Scan using "<1>t_a_idx"
on t (cost=0.42..256.52 rows=9983 width=8) Index Cond: (a < 10000) (6 rows)
```

```
postgres=# EXPLAIN SELECT * FROM t WHERE a = 100; QUERY PLAN --------------------------
--------------------------------------------------- Seq Scan on t (cost=0.00..1693.00 ro
ws=1 width=8) Filter: (a = 100) Result (cost=0.00..0.00 rows=0 width=0) One-Time Filter
: '** plan (using Index Adviser) **'::text -> Index Scan using "<1>t_a_idx" on t (cost=
0.42..2.64 rows=1 width=8) Index Cond: (a = 100) (6 rows)
```

```
postgres=# EXPLAIN SELECT * FROM t WHERE b = 10000; QUERY PLAN ------------------------
---------------------------------------------------- Seq Scan on t (cost=0.00..1693.00
rows=1 width=8) Filter: (b = 10000) Result (cost=0.00..0.00 rows=0 width=0) One-Time Fi
lter: '** plan (using Index Adviser) **'::text -> Index Scan using "<1>t_b_idx" on t (c
ost=0.42..2.64 rows=1 width=8) Index Cond: (b = 10000) (6 rows)
```

- 您可通过psql命令行查询index_advisory表内存储的索引编制建议，示例如下：

```
postgres=# SELECT * FROM index_advisory; reloid | relname | attrs | benefit | original_
cost | new_cost | index_size | backend_pid | timestamp --------+---------+-------+-----
----+----------+----------+----------+----------+---------------------------
------ 16438 | t | {1} | 1337.43 | 1693 | 355.575 | 2624 | 79370 | 18-JUN-21 08:55:51.4
92388 +00:00 16438 | t | {1} | 1684.56 | 1693 | 8.435 | 2624 | 79370 | 18-JUN-21 08:59:
00.319336 +00:00 16438 | t | {2} | 1684.56 | 1693 | 8.435 | 2624 | 79370 | 18-JUN-21 08
:59:07.814453 +00:00 (3 rows)
```

列名	类型	描述
reloid	oid	索引的基表的OID。
relname	name	索引的基表的名称。
attrs	integer[]	生成建议的索引列（由列编号标识）。
benefit	real	此查询的索引的计算收益。
original_cost	real	使用索引之前的平均代价（即执行SQL的预估时间）。
new_cost	real	使用索引之后的平均代价（即执行SQL的预估时间）。
index_size	integer	磁盘页中的估计索引大小。
backend_pid	integer	生成此建议的进程ID。
timestamp	timestamp	生成此建议的日期及时间。

- 如果语句不带EXPLAIN关键字前缀，索引推荐将在语句执行期间分析语句并记录建议。

> **说明** 不得在只读事务中运行索引推荐（index_adviser）。

- 查询WorkLoad级别建议

- 通过show_index_advisory()函数获取单个会话的WorkLoad建议
  此函数用于获取单个会话的索引推荐（由后端进程ID标识），您可通过指定会话的进程ID来调用该函数：

```
SELECT show_index_advisory( pid );
```

> **说明** pid指当前会话的进程ID，您可以在index_advisory表中查看backend_pid参数来获得。您也可以将null作为传递值为当前会话返回结果集。

```
postgres=# SELECT show_index_advisory(null); show_index_advisory ----------------------
---------------------------------------------------------------------------------------
--------------------------------------- create index idx_t_a on public.t(a);/* size: 26
24 KB, benefit: 3021.99, gain: 1.15167301457103, original_cost: 1693, new_cost: 182.005
006313324 */ create index idx_t_b on public.t(b);/* size: 2624 KB, benefit: 1684.56, ga
in: 0.641983590474943, original_cost: 1693, new_cost: 8.4350004196167 */ (2 rows)
```

> **说明** 结果集中每行的表示意义如下：
> - 创建索引推荐建议的索引所需的SQL语句。
> - 索引页的估计大小。
> - 使用索引的总收益（benefit）。
> - 使用索引的增益（gain=benefit/size）。
> - 使用索引之前的平均代价（即执行SQL的预估时间）。
> - 使用索引之后的平均代价（即执行SQL的预估时间）。

- 通过select_index_advisory视图获取所有会话的WorkLoad建议
  此视图包含计算的指标和**CREATE INDEX**语句，展示当前位于index_advisory表中所有会话的索引编制建议。表t中列a和列b的索引编制建议显示如下：

```
postgres=# SELECT * FROM select_index_advisory; backend_pid | show_index_advisory -----
--------+-------------------------------------------------------------------------------
------------------------------------------------------------------------ 79370 | create i
ndex t_a_idx on public.t(a);/* size: 2624 KB, benefit: 3021.99, gain: 1.15167301457103,
original_cost: 1693, new_cost: 182.005006313324 */ 79370 | create index t_b_idx on publ
ic.t(b);/* size: 2624 KB, benefit: 1684.56, gain: 0.641983590474943, original_cost: 169
3, new_cost: 8.4350004196167 */ (2 rows)
```

在每个会话中，从同一建议的索引中受益的所有查询的结果将被组合起来，以便按每个建议的索引生成一组指标，此指标反映在名为benefit和gain的字段中，字段公式如下所示：

```
size = MAX(index size of all queries) benefit = SUM(benefit of each query) gain = SUM(b
enefit of each query) / MAX(index size of all queries)
```

> **说明** 如果单条SQL建议同时创建多个索引，则index_advisory表中记录的new_cost为创建了多个索引之后的代价，而非创建某一个索引之后的代价。

当对给定会话期间得到的不同建议索引的相对优势进行比较时，gain指标十分有用。gain值越大，从索引中得到的成本效益就越高，这可以抵消索引可能消耗的磁盘空间。

6.22. 虚拟索引（hypopg）

hypopg是一个RDS PostgreSQL的扩展，有助于了解特定索引是否可以提高问题查询的性能。虚拟索引并不是真实存在的索引，因此不耗费CPU、磁盘或其他资源，可以有效验证索引是否有效。

前提条件

- RDS PostgreSQL实例内核小版本升级到20220130。查看和升级内核小版本，请参见升级内核小版本。
- 您的账号类型必须为高权限账号。您可以在RDS控制台目标实例的**账号管理**中查看您的账号权限类型。如果您的账号类型为普通账号，您需要创建高权限账号，创建详情请参见创建账号。

> **说明** 虚拟索引只在当前会话内有效。

开启或关闭hypopg

- 开启hypopg。

```
CREATE EXTENSION hypopg;
```

> **说明** 仅高权限账号可以执行此命令。

- 关闭hypopg。

```
DROP EXTENSION hypopg;
```

> **说明** 仅高权限账号可以执行此命令。

调试示例

1. 创建表并插入测试数据。

```
create extension hypopg; CREATE TABLE hypo (id integer, val text) ; INSERT INTO hypo SE
LECT i, 'line ' || i FROM generate_series(1, 100000) i ; VACUUM ANALYZE hypo ;
```

2. 查看默认执行计划。

```
EXPLAIN SELECT val FROM hypo WHERE id = 1; QUERY PLAN ---------------------------------
----------------------- Seq Scan on hypo (cost=0.00..1791.00 rows=1 width=14) Filter: (
id = 1) (2 rows)
```

3. 创建虚拟索引。

```
SELECT * FROM hypopg_create_index('CREATE INDEX ON hypo (id)') ; indexrelid | indexname
------------+----------------------- 18284 | <18284>btree_hypo_id (1 row)
```

4. 查看虚拟索引的效果。

```
EXPLAIN SELECT val FROM hypo WHERE id = 1; QUERY PLAN ---------------------------------
------------------------------------------------- Index Scan using <18284>btree_hypo_id
on hypo (cost=0.04..8.06 rows=1 width=10) Index Cond: (id = 1) (2 rows)
```

5. 虚拟索引是“虚拟的”，并不会在实际运行SQL语句时使用。可以查看SQL实际的运行计划。

```
EXPLAIN ANALYZE SELECT val FROM hypo WHERE id = 1; QUERY PLAN ------------------------
---------------------------------------------------------------------------- Seq Scan on
hypo (cost=0.00..1791.00 rows=1 width=10) (actual time=0.046..46.390 rows=1 loops=1) Fi
lter: (id = 1) Rows Removed by Filter: 99999 Planning time: 0.160 ms Execution time: 46
.460 ms (5 rows)
```

相关参考

关于hypopg的更多详细说明，请参见HypoPG。

6.23. UUID生成器（sequential-uuids）

RDS PostgreSQL支持Sequential UUID generators插件，该插件提供顺序模式的UUID生成。

前提条件

- RDS PostgreSQL实例版本满足以下要求：
  - 实例大版本为PostgreSQL 14、13、12、11和10 。
  - 内核小版本为20220228以上。查看和升级内核小版本，请参见升级内核小版本。
- 您的账号类型必须为高权限账号。您可以在RDS控制台目标实例的**账号管理**中查看您的账号权限类型。如果您的账号类型为普通账号，您需要创建高权限账号，创建详情请参见创建账号。

插件开启和关闭

- 开启sequential-uuids

```
CREATE EXTENSION sequential_uuids;
```

- 关闭sequential-uuids

```
DROP EXTENSION sequential_uuids;
```

使用示例

该扩展提供了两个函数，使用序列或时间戳生成顺序UUID。

> **说明** 该插件的更多使用方法及参数说明，请参见 Sequential UUID generators官方文档。

- uuid_ sequence_nextval函数
  格式：

```
uuid_sequence_nextval(sequence regclass, block_size int default 65536, block_count int de
fault 65536)
```

  命令示例：

```
CREATE SEQUENCE s; SELECT uuid_sequence_nextval('s'::regclass, 256, 256);
```

  结果示例：

```
 uuid_sequence_nextval -------------------------------------- 00cf26f7-ef7a-4746-8871-08b
9c475713e (1 row)
```

- uuid_time_nextval函数
  格式：

```
uuid_time_nextval(interval_length int default 60, interval_count int default 65536) RETUR
NS uuid
```

命令示例：

```
SELECT uuid_time_nextval(1, 256);
```

结果示例：

```
 uuid_time_nextval ------------------------------------ 179f542c-978c-454c-9dee-5f4ba75
288e1 (1 row)
```

6.24. 机器学习（MADlib）

MADlib是一个在AliPG内核数据库中运行机器学习、图计算模型的开源库。在机器学习方面，MADlib除提供数理统计通用函数、存储过程之外，还提供一系列比较经典的监督、非监督学习算法库。

前提条件

- RDS PostgreSQL实例版本满足以下要求：
  - 实例大版本为PostgreSQL 12、11 。
  - 内核小版本为20220228以上。查看和升级内核小版本，请参见升级内核小版本。
- 您的账号类型必须为高权限账号。您可以在RDS控制台目标实例的**账号管理**中查看您的账号权限类型。如果您的账号类型为普通账号，您需要创建高权限账号，创建详情请参见创建账号。

插件简介

MADlib机器学习模块主要解决以下问题：

- 分类、回归问题：提供一系列算法，如K最近邻、MLP多层感知神经网络、SVM支持向量机、决策树等算法来解决二元分类、回归问题，集成最小二乘法、GLM广义线性回归、逻辑回归、多项式回归等模型来解决回归问题。
- 聚类问题：提供K-Means算法实现聚类分析。
- 关联分析：提供Apriori算法实现关联分析，解决如“啤酒与尿布”的关联问题。
- 时序分析：提供ARIMA自回归移动平均模型预测时间序列数据的未来值。
- 其他：数据降维如通过PCA主成分分析模型来提炼主因子，文档分类与LDA文档主题建模。

MADlib还集成图计算模型，解决诸如最短路径、PageRank排名、社交网络“谁认识谁，谁与谁更熟悉”等问题。图计算模型相关算法如下表：

类别	模型/功能	说明
最短路径	所有顶点之间最短路径	计算所有顶点之间的最短路径并保存到指定结果表，根据结果表查找任意起点到任意终点的最短路径。
	给定顶点到其他之间最短路径	计算给定顶点到其他顶点之间的最短路径并保存到指定结果表，根据结果表查找给定顶点到任意终点的最短路径。
广度优先搜索	广度优先搜索	使用广度优先方法查找从给定源顶点可到达的节点。
HITS	HITS分数	在有向图中找到所有顶点的HITS分数（权威度和中心度）。
网页排名	PageRank	在有向图中找到所有顶点的PageRank。

类别	模型/功能	说明
弱连通分量	弱连通分量	查找图的所有弱连接组件。
度量	平均路径长度	计算图的平均最短路径长度。
	紧密度	计算图中每个节点的紧密度中心值。
	图形直径	计算图的直径。
	入/出度	计算每个顶点的度数（入度/出度）。

插件开启和关闭

- 开启MADlib

> **说明** 开启MADlib插件前，需要使用 `CREATE EXTENSION plpythonu;` 命令先创建plpythonu插件。

```
CREATE EXTENSION madlib;
```

- 关闭MADlib

```
DROP EXTENSION madlib;
```

相关参考

MADlib的详细说明，请参见MADlib官方文档。

7.快速入门

7.1. 使用流程

快速入门旨在介绍如何创建RDS PostgreSQL实例、进行基本设置以及连接实例数据库，使用户能够了解从购买RDS实例到开始使用实例的流程。

快速入门流程图

若您初次使用阿里云RDS，请先了解阿里云RDS使用限制。

通常，从新购实例到可以开始使用实例，您需要完成如下操作。

1. 创建RDS PostgreSQL实例
2. 设置白名单
3. 申请或释放外网地址
4. 创建数据库和账号
5. 连接PostgreSQL实例

7.2. 创建RDS PostgreSQL实例

您可以通过阿里云RDS管理控制台或API创建RDS实例。本文介绍如何通过控制台创建RDS PostgreSQL实例。

> **说明** 首次购买RDS享受折扣价。详情请参见 优惠活动。

前提条件

通过RAM用户创建RDS实例时，该RAM用户必须拥有AliyunRDSFullAccess权限。如何授权，请参见RAM资源授权。

操作步骤

1. 点此打开RDS实例创建页面。
2. 选择**计费方式**。

计费方式	建议	好处
包年包月	长期使用RDS实例，请选择**包年包月**（一次性付费），并在页面左下角选择**购买时长**。	包年包月比按量付费更实惠，且购买时长越长，折扣越多。
按量付费	短期使用RDS实例，请选择**按量付费**（按小时付费）。 您可以先创建**按量付费**的实例，确认实例符合要求后转**包年包月**。	可随时释放实例，停止计费。

> **说明** 可以在页面右下角查看价格。您需要完成后续的配置选择，才能最终确定价格。

3. 设置以下参数。

类别	说明
地域	选择要在哪个地域创建RDS实例。 ○ 如果您要通过云服务器ECS连接该RDS实例，请在此选择ECS实例所在的地域，使得RDS实例与ECS实例位于同一地域（例如，都位于杭州）。否则，ECS实例与RDS实例无法内网互通，只能外网互通，无法发挥最佳性能。 ○ 如果您要通过其它设备（例如本地服务器或电脑）连接该RDS实例，选择一个离该设备较近的地域即可。
类型	本文介绍PostgreSQL，选择**PostgreSQL**。 当前支持PostgreSQL 10、11、12、13和14。 **说明** 不同地域支持的数据库类型不同，请以实际界面为准。
系列	○ **基础版**：一个节点，性价比高，用于学习或测试。 ○ **高可用版**（推荐）：一个主节点和一个备节点，经典高可用架构，用于生产环境。 **说明** ○ 支持的系列取决于您选择的地域和数据库类型。 ○ 关于各系列的详细介绍，请参见产品系列概述。

类别	说明
存储类型	○ **本地SSD盘**：与数据库引擎位于同一节点的SSD盘。将数据存储于本地SSD盘，可以降低I/O延时。当前PostgreSQL10支持本地SSD盘。 ○ **ESSD云盘**：增强型（Enhanced）SSD云盘，分为如下三类： ■ ESSD PL1云盘：PL1性能级别的ESSD云盘。 ■ ESSD PL2云盘：相比PL1大约提升2倍IOPS和吞吐量。 ■ ESSD PL3云盘：相比PL1最高可提升20倍IOPS、11倍吞吐量，适合对极限并发I/O性能要求极高、读写时延极稳定的场景。 ○ **SSD云盘**：基于分布式存储架构的弹性块存储设备。选择SSD云盘，即实现了计算与存储分离。 **说明** ○ 支持的存储类型取决于您选择的实例类型和系列。 ○ 如果选择**ESSD云盘**或**SSD云盘**，可以勾选**云盘加密**，进一步保护数据安全。详情请参见云盘加密。 ○ 更多信息，请参见存储类型。
可用区	可用区是指地域中的一个独立物理区域，例如，杭州地域有可用区H、I、J等。 ○ **多可用区部署（推荐）**：主节点和备节点位于不同可用区，提供跨可用区容灾。 ○ **单可用区部署**：主节点和备节点位于同一可用区。 **说明** 如果实例系列选择为基础版，则只能选择 **单可用区部署**。
实例规格	先选择分类（入门级或企业级），再选择具体规格。 ○ **通用规格（入门级）**：独享被分配的内存和I/O资源，与同一服务器上的其他通用型实例共享CPU和存储资源。 ○ **独享规格（企业级）**：独享型指独享被分配的CPU、内存、存储和I/O资源。独占型是独享型的顶配，独占整台服务器的CPU、内存、存储和I/O资源。 **说明** 详情请参见 主实例规格列表。
存储空间	存储空间包括数据空间、系统文件空间、Binlog文件空间和事务文件空间。调整存储空间时最小单位为5GB。 **说明** 部分本地SSD盘的存储空间大小与实例规格绑定，ESSD云盘或SSD云盘不受此限制。详情请参见 主实例规格列表。

4. 在页面右下角，单击**下一步：实例配置**。
5. 选择 。网络类型

i. **网络类型**：建议选择与ECS实例相同的网络类型，否则，ECS实例与RDS实例无法内网互通。

> 说明
> - 根据准备工作，确认您的ECS实例的网络类型。
> - 如果不需要通过内网访问RDS，那么任选一种网络类型。
> - 部分RDS实例不支持经典网络。

ii. **VPC和交换机**：如果网络类型为专有网络，还需选择VPC和交换机，建议选择与ECS实例相同的VPC。ECS实例与RDS实例位于不同VPC时，无法内网互通。

> 说明
> - 根据准备工作，确认您的ECS实例的VPC名称。
> - VPC相同，交换机不同，ECS实例与RDS实例也可以内网互通。

6. 设置更多自定义参数。如无特殊需求，可保留默认值。

类别	说明
实例释放保护	选中此项可以为按量付费实例开启实例释放保护，防止按量付费实例被意外释放。更多信息，请参见开启和关闭实例释放保护。
资源组	选择默认或自定义的资源组，方便实例管理。
时区	设置实例时区。 说明 ○ 购买主实例时支持选择时区，只读实例不支持选择时区，只读实例将继承主实例时区。 ○ 仅当实例为PostgreSQL云盘时，该参数可配置。 ○ 支持命名时区，不支持UTC时区。命名时区的详细信息，请参见RDS MySQL和RDS PostgreSQL实例的常用命名时区清单。 ○ 如果不配置此参数，系统将根据您购买实例的地域，选择默认时区。地域与默认命名时区的对应关系，请参见RDS PostgreSQL实例地域与默认命名时区对应关系。

7. 在页面右下角，单击**下一步：确认订单**。
8. 确认订单信息、**购买量** 和**购买时长**（仅包年包月实例），勾选**服务协议**，单击**去支付**，并完成支

付。

> **说明** 对于包年包月实例，建议选中 **到期自动续费**，避免因忘记续费而导致业务中断。

控制台提示**支付成功**或**开通成功**。

9. 查看实例。

   进入 实例列表，在上方选择实例所在地域，根据 **创建时间** 找到刚刚创建的实例。

下一步

创建数据库和账号

常见问题

为什么创建实例后，实例列表看不到创建中的实例？

可能原因	说明	建议
地域错误	您所在地域和您创建实例时选择的地域不一致。	在页面左上角切换地域。
可用区内资源不足	可用区内资源不足，导致创建失败。 创建失败 您可以在订单列表里看到退款。	选择其它可用区后重试。
RAM权限策略禁止创建未加密的RDS实例	已配置RAM权限策略，禁止RAM用户创建未加密的RDS实例。 • RAM用户尝试创建本地盘实例，实例创建失败（本地盘实例无法在创建时设置磁盘加密）。 • RAM用户尝试创建云盘实例，但未设置云盘加密，实例创建失败。 更多信息，请参见禁止创建磁盘未加密的RDS实例。	创建实例时，存储类型选择云盘，选中云盘加密并设置密钥后重试。

相关文档

- 通过API创建RDS实例：创建RDS实例
- 创建其它类型实例请参见：
  - 创建RDS MySQL实例
  - 创建RDS SQL Server实例
  - 创建RDS MariaDB实例

7.3. 设置白名单

7.3.1. 设置白名单

创建RDS PostgreSQL实例后，暂时还无法访问该实例，您需要设置RDS PostgreSQL实例的白名单，即IP白名单或安全组，本文介绍如何设置IP白名单。

其他引擎设置白名单请参见：

- MySQL设置白名单
- SQL Server设置白名单
- PPAS设置白名单
- MariaDB设置白名单

操作场景

IP白名单指允许访问RDS实例的IP清单。设置IP白名单可以让RDS实例得到高级别的访问安全保护，建议您定期维护白名单。

通常需要设置IP白名单的场景如下：

- 场景1
  创建RDS实例后，您需要将外部IP地址添加至IP白名单中，外部设备才可以正常访问该RDS实例。
- 场景2
  当数据库连接异常时，您可以检查白名单设置是否正确。
  不同连接场景下，IP白名单的设置请参见下表。

> **说明** 专有网络VPC是阿里云上一种隔离的网络环境，安全性比传统的经典网络更高。

连接场景	网络类型	IP白名单设置
ECS实例和RDS实例连接	两个实例在相同专有网络VPC内（推荐）	添加ECS实例私有IP地址。
	两个实例在不同专有网络VPC内	不同专有网络的实例无法内网互通，您可以参考如下方案： i. 切换RDS专有网络，选择和ECS实例相同的VPC。 **说明** ECS实例和RDS实例需要处于相同地域才能切换到相同VPC。如果地域不同，为业务稳定，建议您通过DTS将RDS实例迁移至ECS实例所属地域。详情请参见RDS PostgreSQL间的数据迁移。 ii. 在IP白名单中添加ECS实例私有IP地址。
	两个实例均为经典网络	添加ECS实例私有IP地址。
	ECS实例为经典网络RDS实例为专有网络	不同网络类型的实例无法内网互通，您可以参考如下方案： i. 将ECS 实例从经典网络迁移到专有网络，选择和RDS实例相同的VPC。 **说明** ECS实例和RDS实例需要处于相同地域才能切换到相同VPC。如果地域不同，为业务稳定，建议您通过DTS将RDS实例迁移至ECS实例所属地域。详情请参见RDS PostgreSQL间的数据迁移。 ii. 在IP白名单中添加ECS实例私有IP地址。

连接场景	网络类型	IP白名单设置
	ECS实例为专有网络 RDS实例为经典网络	不同网络类型的实例无法内网互通，您可以参考如下方案： i. 把RDS实例从经典网络切换为专有网络，选择和ECS实例相同的VPC。 说明 ECS实例和RDS实例需要处于相同地域才能切换到相同VPC。如果地域不同，为业务稳定，建议您通过DTS将RDS实例迁移至ECS实例所属地域。详情请参见RDS PostgreSQL间的数据迁移。 ii. 在IP白名单中添加ECS实例私有IP地址。
云外主机连接RDS实例	无	在IP白名单中添加云外主机的公网IP地址。 说明 ○ 云外主机的应用程序中使用RDS实例的外网连接地址。 ○ 定位本地公网IP地址请参见外网无法连接RDS MySQL或MariaDB：如何正确填写本地设备的公网IP地址。

注意事项

- 单个实例最多支持50个IP白名单分组。
- 设置白名单不会影响RDS实例的正常运行。
- 默认的IP白名单分组（default ）不能删除，只能清空。
- 请勿修改或删除系统自动生成的分组，避免影响相关产品的使用。例如 ali_dms_group （DMS产品IP地址白名单分组）、 hdm_security_ips （DAS产品IP地址白名单分组）。
- 默认的IP白名单只包含127.0.0.1，表示任何IP均无法访问该RDS实例。

设置通用模式IP白名单

通用白名单模式不区分经典网络和专用网络。在通用白名单模式下，设置的IP地址，既可通过经典网络，也可通过专有网络访问RDS实例。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据安全性**。
3. 单击**添加白名单分组**，填写**分组名称**或单击已有分组的**修改** 。
4. 填写需要访问该实例的IP地址或IP段，然后单击。

   说明
   - 用英文逗号隔开多个IP地址或IP段，且逗号前后不能有空格，例如 `192.168.0.1,172.16.213.9` 。
   - 单个实例最多添加1000个IP地址或IP段。当IP地址较多时，建议将零散的IP合并为IP段，例如10.10.10.0/24。

5. （可选）如果当前实例包含只读实例，可以通过参数**白名单同步到只读实例**配置白名单同步，将主实例的白名单同步至指定的只读实例。当有多个只读实例时，支持多选。

6. （可选）单击**加载ECS内网IP**，将显示您当前阿里云账号下所有ECS实例的IP地址，可快速添加ECS私有IP地址到白名单中。

高安全白名单模式IP白名单

高安全白名单模式区分经典网络和专用网络。在高安全白名单模式下，白名单分组需指定网络隔离模式，例如设置在经典网络的白名单IP地址，不可从专有网络访问RDS实例，反之亦然。

仅本地SSD盘实例支持高安全白名单模式，如果实例已经是高安全白名单模式，参见下文进行设置即可。如果需要切换为高安全白名单模式，请参见切换为高安全白名单模式。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据安全性**。
3. 在页签中，添加或修改白名单分组。
   - 添加白名单分组
     a. 单击**添加白名单分组**。
     b. 选择**允许访问实例的网络类型**。
     c. 填写**分组名称**。
   - 修改白名单分组
     单击目标白名单分组右侧的**修改**。
4. 在**组内白名单**框内，填写需要访问该实例的IP地址或IP段，然后单击**确定**。

> **说明**
> - 用英文逗号隔开多个IP地址或IP段，且逗号前后不能有空格，例如 `192.168.0.1,172.16.213.9` 。
> - 单个实例最多添加1000个IP地址或IP段。当IP地址较多时，建议将零散的IP合并为IP段，例如10.10.10.0/24。

5. （可选）如果当前实例包含只读实例，可以通过参数**白名单同步到只读实例**配置白名单同步，将主实例的白名单同步至指定的只读实例。当有多个只读实例时，支持多选。

6. （可选）单击**加载ECS内网IP**，将显示您当前阿里云账号下所有ECS实例的IP地址，可快速添加ECS私有IP地址到白名单中。

下一步

创建数据库和账号

相关API

API	描述
DescribeDBInstanceIPArrayList	查看RDS实例IP白名单
ModifySecurityIps	修改RDS实例IP白名单

7.3.2. 设置安全组

安全组是一种虚拟防火墙，用于控制安全组中的ECS实例的出入流量。在RDS白名单中添加安全组后，该安全组中的ECS实例就可以访问RDS实例。本文介绍如何设置安全组。

操作场景

创建RDS PostgreSQL实例后，暂时无法访问实例。您需要设置RDS PostgreSQL实例的IP白名单或安全组。

关于安全组的更多信息，请参见创建安全组。

> **说明** 您可以同时设置 IP白名单和安全组。IP白名单中的IP地址和安全组中的ECS实例都可以访问该RDS实例。

注意事项

- 您可以同时设置IP白名单和安全组。IP白名单中的IP地址和安全组中的ECS实例都可以访问该RDS实例。
- 实例最多支持添加10个安全组。

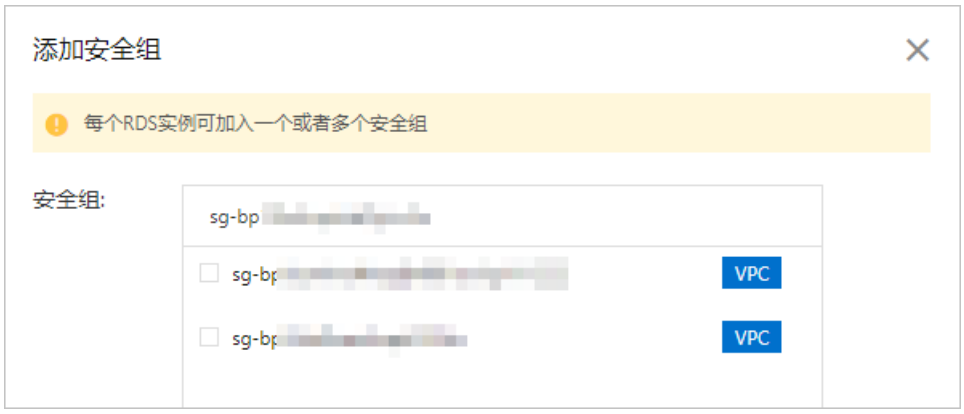
- 白名单中的安全组的更新将实时应用到白名单。
- RDS实例只能添加与自身网络类型相同的安全组，即实例为专有网络VPC时，只能添加VPC类型的安全组；实例为经典网络时，只能添加经典网络类型的安全组。

> **说明**　添加安全组后，如果切换实例网络类型，会导致安全组失效，需要重新添加对应网络类型的安全组。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据安全性**，然后单击**安全组**页签。
3. 单击**添加安全组**。

> **说明**　带有 VPC标识的安全组为专有网络中的安全组。



4. 选择要添加的安全组，单击**确定**。

下一步

创建数据库和账号

相关API

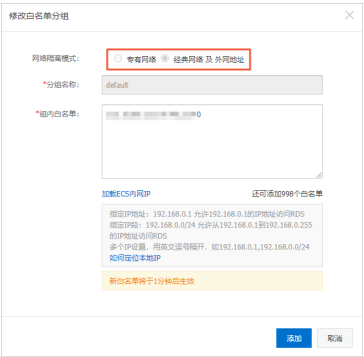
API	描述
DescribeSecurityGroupConfiguration	查询指定RDS实例和ECS安全组的关联信息
ModifySecurityGroupConfiguration	修改指定RDS实例和ECS安全组的关联信息

7.3.3. 白名单常见案例和问题

本文介绍设置白名单时的案例和问题。

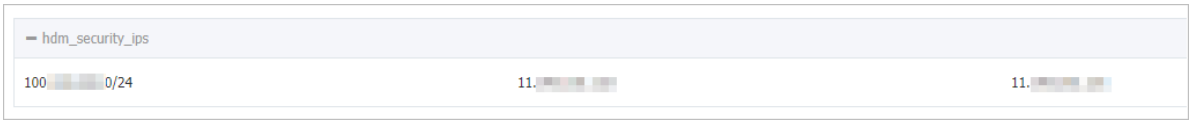
常见错误案例

错误案例	说明	解决办法
没有设置IP白名单，即白名单中只有127.0.0.1。	该地址表示不允许任何IP地址访问RDS实例。	在IP白名单中添加外部IP地址。

错误案例	说明	解决办法
测试连接实例时，添加的IP地址是0.0.0.0。	IP地址格式错误。	修改为0.0.0.0/0。 **注意** 0.0.0.0/0表示允许任何IP访问RDS实例，只建议在测试时使用，请勿在线上业务实例中设置IP白名单为0.0.0.0/0。
IP白名单中添加的设备公网IP地址错误。	• 公网IP地址不固定，可能会变动。 • IP地址查询工具或网站查询的公网IP地址可能不准确。	参见RDS for PostgreSQL/PPAS 如何定位本地 IP。
高安全模式的IP白名单中，网络类型错误。	高安全白名单模式会区分经典网络和专有网络。	在正确的网络类型分组内填写IP地址。例如专有网络的IP白名单分组内添加某个IP地址，则这个IP地址只能在专有网络内访问该RDS实例。 

常见问题

- Q：可以同时设置IP白名单和安全组吗？
  A：可以。IP白名单中的IP地址和安全组中的ECS实例都可以访问该RDS实例。
- Q：设置IP白名单后立刻生效吗？
  A：等待1分钟左右才会生效。
- Q：ali_dms_group和hdm_security_ips白名单分组是什么？
  A：您在数据管理服务DMS和数据库自治服务DAS中接入RDS实例时，经过您的授权后，系统会生成ali_dms_group和hdm_security_ips白名单分组。请勿修改或删除这些分组，避免影响相关服务的使用。这些服务不会操作您任何业务数据。



7.4. 创建数据库和账号

若要使用云数据库RDS，您需要在实例中创建数据库和账号。本文介绍如何为RDS PostgreSQL实例创建数据库和账号。

账号类型

RDS PostgreSQL实例支持两种数据库账号：高权限账号和普通账号。详细说明如下。

账号类型	说明
高权限账号	• 只能通过控制台或API创建和管理。 • 可以创建多个高权限账号，管理所有普通账号和数据库。 • 开放了更多权限，可满足个性化和精细化的权限管理需求，例如可按用户分配不同表的查询权限。 • 可以断开任意账号的连接。 **说明** • 当创建的高权限账号是该实例的第一个高权限账号时，这个账号将是标准系统数据库template1中默认模式Public Schema的Owner。 • CREATE DATABASE命令默认通过复制template1来创建数据库。通过该方式创建的数据库，Public Schema的Owner均为第一个高权限账号。 • 第一个高权限账号的Comment开头会包含说明：template1 public schema owner。
普通账号	• 可以通过控制台、API或者SQL语句创建和管理。 • 一个实例可以创建多个普通账号 。 • 需要手动给普通账号授予特定数据库的权限。 • 普通账号不能创建和管理其他账号，也不能断开其他账号的连接。

注意事项

- 支持在控制台创建多个高权限账号和普通账号，也可以通过SQL命令创建、管理普通账号。
- 如果您要迁移本地数据库到RDS，请在RDS实例中创建与本地数据库一致的迁移账号和数据库。
- 分配数据库账号权限时，请按最小权限原则和业务角色创建账号，并合理分配只读和读写权限。必要时可以把数据库账号和数据库拆分成更小粒度，使每个数据库账号只能访问其业务之内的数据。如果不需要数据库写入操作，请分配只读权限。
- 为保障数据库的安全，请将数据库账号的密码设置为强密码，并定期更换。

创建数据库

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据库管理**。
3. 单击**创建数据库**。
4. 设置如下参数。

参数	说明
数据库（DB）名称	○ 最长63个字符。 ○ 由小写字母、数字、中划线、下划线组成。 ○ 以字母开头，以字母或数字结尾。
支持字符集	数据库的字符集。
Collate	字符串排序规则。
Ctype	字符分类。
授权账号：	设置数据库的所有者，对数据库拥有ALL权限。
备注说明：	填写备注信息。

5. 单击**创建**。

创建账号

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中选择**账号管理**。
3. 单击**创建账号**。
4. 设置如下参数。

参数	说明
数据库账号：	○ 长度为2~63个字符。 ○ 由小写字母、数字或下划线组成。 ○ 以字母开头，以字母或数字结尾。 ○ 不能和已有的账号名重复。 ○ 不能以pg开头。 ○ 不能使用SQL关键字。具体请参见SQL关键字。
账号类型：	RDS PostgreSQL实例支持两种数据库账号：高权限账号和普通账号。 ○ 高权限账号拥有所有数据库的所有操作权限。 ○ 普通账号拥有已授权数据库（owner）的所有操作权限。 **说明** 操作权限包括SELECT、INSERT、UPDATE、DELETE、TRUNCATE、REFERENCES、TRIGGER。
密码：	○ 长度为8~32个字符。 ○ 由大写字母、小写字母、数字、特殊字符中的任意三种组成。 ○ 特殊字符为!@#$%^&*()_+-=

参数	说明
确认密码：	再次输入相同的密码。
备注说明：	填写备注信息。

5. 单击**确定**。

常见问题

创建的账号在只读实例上可以用吗？

答：主实例创建的账号会同步到只读实例，只读实例无法管理账号。账号在只读实例上只能进行读操作，不能进行写操作。

相关API

API	描述
创建数据库账号	创建账号

7.5. 连接PostgreSQL实例

您可以通过数据管理 DMS 平台、PostgreSQL命令行终端工具、pgAdmin客户端或应用程序等方式连接PostgreSQL实例，实现业务目标。本文介绍这些连接方法的具体操作。

前提条件

已完成以下操作：

- 创建RDS PostgreSQL实例
- 创建账号
- 设置白名单
- ECS要通过内网访问RDS，必须满足以下所有条件：
  - ECS和RDS属于同一个阿里云主账号。
  - ECS和RDS位于同一个地域。
  - ECS和RDS必须处于同一个VPC。
  - ECS的私网IP已添加到RDS白名单，具体操作请参见设置白名单。

通过DMS连接

您可以在**基本信息**页面单击**登录数据库**通过DMS链接数据库。

具体操作请参见通过DMS登录RDS数据库。

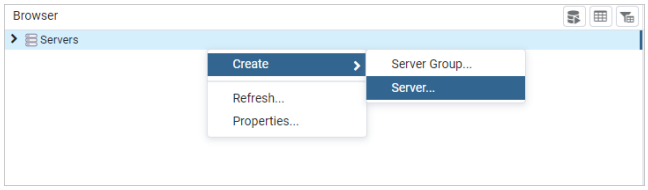
通过pgAdmin客户端连接

通过PostgreSQL官方网站下载 并安装PostgreSQL客户端时，将会自动安装 pgAdmin 4客户端，您也可以单独下载pgAdmin客户端。

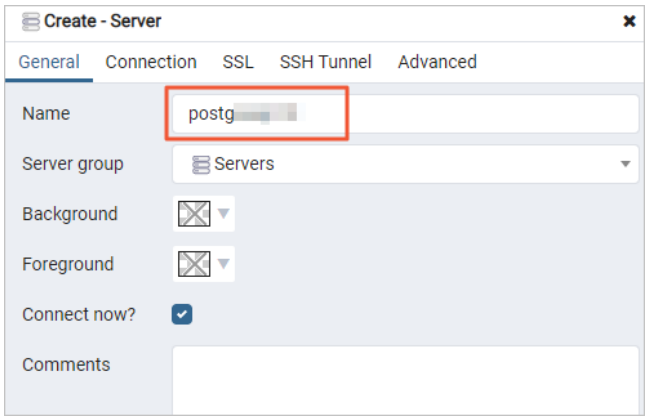
1. 启动pgAdmin 4客户端。

> **说明** 高版本客户端首次登录需要设置Master Password用于保护保存的密码和其他凭据。

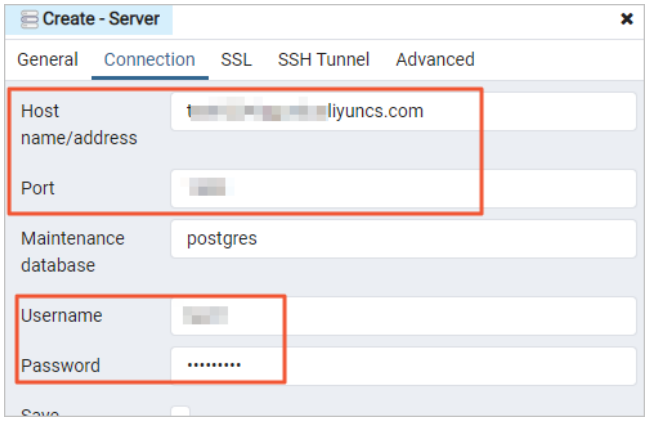
2. 右键单击 Servers，选择Create > Server...。



3. 在General页签设置连接名称。



4. 选择Connection标签页，输入要连接的实例信息。

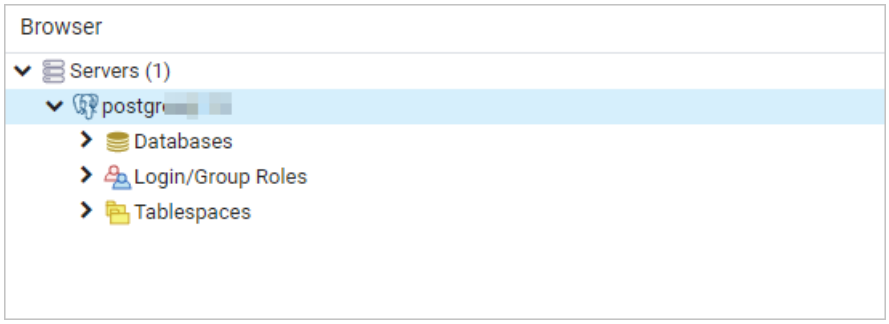


参数	说明
Hostname/address	主机名称/地址。若通过内网连接，需输入RDS实例的内网连接地址；若使用外网连接，需输入RDS实例的外网连接地址。查看RDS实例的内外网地址及端口请参见查看或修改内外网地址和端口。
Port	连接地址对应的端口。
Username	RDS实例的账号名称。创建RDS实例的账号请参见创建数据库和账号。

参数	说明
Password	账号对应的密码。

5. 单击Save。
   若连 接信息无误，会出现如下界面，则表示连 接成功。

> **注意**　postgres是RDS实例默认的系统数据库，请勿在该数据库中进行任何操作。



通过PostgreSQL客户端命令行工具连接

通过PostgreSQL官方网站下载并安装PostgreSQL客户端时，将会自动安装**PostgreSQL命令行终端工具（Command Line Tools）**。

在命令行终端中输入如下命令连接RDS PostgreSQL数据库。

```
psql -h <实例连接地址> -U <用户名> -p <端口号> -d <数据库名>
```

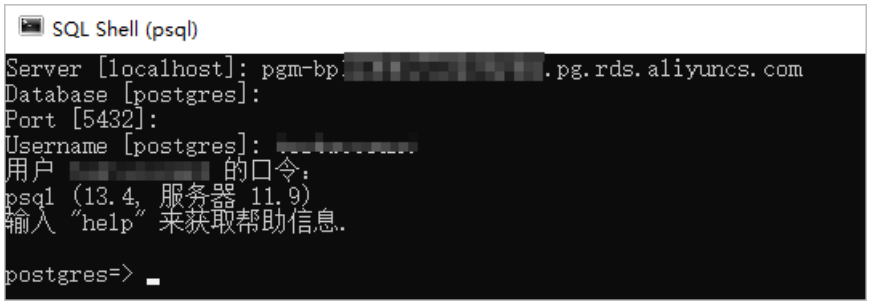
各参数在RDS控制台的获取方法。

参数	获取方法
实例连接地址	RDS PostgreSQL实例的连接地址，具体请参见查看或修改内外网地址和端口。
用户名	通过**账号管理**页面获取用户名。创建账号，请参见创建账号。
端口号	默认为5432，如果您修改过，通过**数据库连接**页面获取。具体请参见查看或修改内外网地址和端口。
数据库名	postgres是RDS PostgreSQL实例默认的系统数据库，请勿在该数据库中进行任何操作。如果您需要连接RDS实例下创建的其他数据库，可从**数据库管理**页面获取。创建数据库，请参见创建数据库。

通过SQL Shell（psql）工具连接

通过PostgreSQL官方网站下载并安装PostgreSQL客户端时，将会自动安装SQL Shell（psql）工具。

在本地Windows系统的**开始**菜单中找到并打开**SQL Shell（psql）**工具，依次输入对应参数连接RDS PostgreSQL实例。



各参数在RDS控制台的获取方法。

参数	获取方法
Server	RDS PostgreSQL实例的连接地址，具体请参见查看或修改内外网地址和端口。
Database	数据库名，不配置默认为系统数据库postgres，请勿在该数据库中进行任何操作。如果您需要连接RDS实例下创建的其他数据库，可从**数据库管理**页面获取。创建数据库，请参见创建数据库。
Port	端口号，不配置默认为5432，如果您修改过，通过**数据库连接**页面获取。具体请参见查看或修改内外网地址和端口。
Username	用户名，通过**账号管理**页面获取用户名。创建账号，请参见创建账号。

通过应用程序连接

说明 本文以Maven项目JDBC连接实例为例，其它编程语言连接方式类似。

1. pom.xml中添加依赖。

```
<dependency> <groupId>postgresql</groupId> <artifactId>postgresql</artifactId> <version
>8.2-504.jdbc3</version> </dependency>
```

2. 连接实例示例代码如下：

```
public class DatabaseConnection { public static void main( String[] args ){ try { Class
.forName("org.postgresql.Driver"); } catch (ClassNotFoundException e) { e.printStackTra
ce(); } //实例连接地址 String hostname = "pgm-bp1i3kkq7321o9****.pg.rds.aliyuncs.com"; //
实例连接端口 int port = 5432; //数据库名称 String dbname = "postgres"; //用户名 String user
name = "username"; //密码 String password = "password"; String dbUrl = "jdbc:postgresql:
//" + hostname + ":" + port + "/" + dbname + "?binaryTransfer=true"; Connection dbConne
ction; try { dbConnection = DriverManager.getConnection(dbUrl, username, password); Sta
tement statement = dbConnection.createStatement(); //输入需要执行的SQL语句。 String select
Sql = "SELECT * FROM information_schema.sql_features LIMIT 10"; ResultSet resultSet = s
tatement.executeQuery(selectSql); while (resultSet.next()) { System.out.println(resultS
et.getString("feature_name")); } } catch (SQLException e) { e.printStackTrace(); } } }
```

通过SSL安全连接

RDS PostgreSQL支持设置SSL链路加密，对网络连接进行加密，保证传输链路的安全。具体配置请参见SSL连接RDS PostgreSQL数据库。

常见问题

Q：我使用函数计算，想获取RDS的数据，要怎么操作呢？

A：您可以为函数安装第三方依赖，使用内置模块获取RDS数据，详情请参见为函数安装第三方依赖。

8.开发运维建议

本文介绍RDS PostgreSQL的一些开发运维建议，帮助您有效提升数据库使用的规范性和稳定性，保持高性能。

客户端连接池

- 使用绑定变量（PreparedStatement），消除数据库SQL硬解析CPU消耗，对性能提升巨大。
- 减少idle connection数量。可以降低RDS PostgreSQL连接内存占用，并提升 `GetSnapshotData()` 效率，提升性能。
- 使用应用连接池，避免短连接损耗和性能下降。如果应用没有连接池，建议在应用层和数据库之间架设连接池，例如使用PgBouncer或者Pgpool-II作为连接池。
- 应用连接池参数的配置建议：
  - `minimumIdle=1` ：控制连接池空闲连接的最小数量，减少idle connection。

    > **说明** 当前大多数连接池已取消了 maxIdle 参数，如果有， maxIdle 参数设置为1。

  - `maxLifetime` ：建议设置为60分钟，用来设置一个connection在连接池中的最大存活时间。可以极大降低连接Relcache内存导致OOM的概率。
  - `maximumPoolSize=15` ： 单个连接池实例允许的最大连接数 ，15已经满足绝大多数应用需求。如果连接池客户端个数较少，负载集中，可以适当增大这些客户端的 maximumPoolSize 。

> **说明**
> 常见应用连接池参数的配置建议如下：
> - HikariCP（Java，推荐的连接池）：
>
>  ```
>  minimumIdle=1, maximumPoolSize=15, idleTimeout=600000 (10 minutes), maxLifetime=3600000 (60 minutes)
>  ```
>
> - GORM（golang，推荐的连接池）：
>
>  ```
>  sqlDB.SetMaxIdleConns(1), sqlDB.SetMaxOpenConns(15), sqlDB.SetConnMaxLifetime(time.Hour)
>  ```
>
> - Druid（Java）：
>
>  ```
>  initialSize=1, minIdle=1, maxIdle=1, maxActive=15, testOnBorrow=false, testOnReturn=false, testWhileIdle=true,minEvictableIdleTimeMillis=600000 (10 minutes), maxEvictableIdleTimeMillis=900000 (15 minutes), timeBetweenEvictionRunsMillis=60000 (1 minutes), maxWait=6000 (6 seconds).
>  ```
>
> 上述配置不涉及 PreparedStatement 配置， PreparedStatement 需要另外配置。

性能与稳定性

- 单个库的表数量建议不要超过 5000 个，1个分区表的子分区个数建议不要过个128个，若按时间分区，使用按月或年分区，不建议按日分区。以此减少每个连接Relcache内存占用。
- 新建索引，对于在线业务，推荐使用**CREATE INDEX CONCURRENTLY**方式创建索引，不堵塞其他会话对被创建索引表的DML（**INSERT**、**UPDATE**或**DELETE**）操作。
- 重建索引，对PostgreSQL 12及以上大版本，使用**REINDEX CONCURRENTLY**。PostgreSQL 11及下大版

本，使用CONCURRENTLY创建新索引成功后，再删除老索引。

- 避免频繁创建和删除临时表，以减少系统表资源的消耗。特别是ON COMMIT DROP，务必慎用。通常临时表功能，可以使用WITH语句替代。
- PostgreSQL13在分区表、HashAggregate(group by)、并行查询等有较大的优化，建议升级PostgreSQL大版本。具体请参见RDS PostgreSQL升级数据库大版本。
- 游标使用后及时关闭。
- 使用TRUNCATE代替DELETE全表，提升性能。
- PostgreSQL支持DDL事务，支持回滚DDL，建议将DDL封装在事务中执行，必要时可以回滚，但是需要注意事务的长度，避免长时间堵塞DDL对象的读操作。
- 如果有大批量的数据入库，建议使用copy语法，或者 `INSERT INTO table VALUES (),(),...();` 的方式，提高写入速度。

实例小版本建议

- 使用Replication Slot（例如逻辑订阅），强烈建议升级小版本到20201230及以上，以开启逻辑复制槽故障转移，同时设置最大复制槽延迟告警，以防止订阅延迟或中断造成Slot位点无法推进，进而造成PostgreSQL数据库wal堆积。
- 开启审计日志或者性能洞察（log_statement=all），强烈建议升级小版本到20211031及以上。

> **说明** `log_statement=all` 在高并发场景（active_conn > 50）下，提升性能约4倍，并且消除了 `log_statement=all` 导致的SysCpu飙升问题。

监控告警

- 强烈建议开启RDS PostgreSQL云监控模版的告警项，参见管理报警。

问题排查

- RDS PostgreSQL慢SQL问题排查：请参见慢SQL统计和分析方法（EXPLAIN）。
- 查找最耗费资源的SQL（Top SQL）：请参见查找最耗费资源的 SQL（Top SQL）。

设计

权限

- 权限管理，以schema/role为单位分配权限，创建readwrite/readonly账号，遵循最小权限原则，请参见RDS PostgeSQL权限管理最佳实践。
- 若使用应用层读写分离，readonly客户端建议使用readonly账号，最小权限原则，避免权限误用。

表

- 表结构中字段定义的数据类型建议与应用程序中的定义保持一致，表之间字段校对规则一致，避免报错或无法使用索引的情况发生。
- 对于存在定期历史数据删除需求的业务，建议数据表按时间分区，按时间月或年区分，删除时使用 `DROP` 或者 `TRUNCATE` 操作对应的子表，不建议使用 `DELETE` 操作。
- 对于频繁更新的表，建议在建表时指定表的 `FILLFACTOR=85` ，每页预留15%的空间用于HOT更新。

```
CREATE TABLE test123(id int, info text) WITH(FILLFACTOR=85);
```

- 临时表建议以 `tmp_` 开头，子表建议根据业务场景以规则结尾，例如按年分区的主表如果为tbl，则子表为tbl_2016、tbl_2017等。

索引

- B-Tree索引字段至多2000字节，如果存在超过2000字节的字段需要新建索引，建议使用函数索引（例如哈希值索引）或分词索引。
- 对于线性顺序存储的数据（如流式数据、时间字段或自增字段），通常查询时使用范围查询，建议使用 `BRIN` 索引，减少索引的大小，加快数据插入速度。

```
CREATE INDEX idx ON tbl using BRIN(id);
```

- 建议避免全表扫描（大数据量扫描的数据分析除外），PostgreSQL支持几乎所有数据类型的索引。
  索引接口包括：B-Tree、Hash、GIN、GiST、SP-GiST、BRIN、RUM（扩展接口）、Bloom（扩展接口）、PASE（扩展接口）。
- 主键索引建议以 `pk_` 开头，唯一索引建议以 `uk_` 开头，普通索引建议以 `idx_` 开头。

数据类型及字符集

- 建议选择合适的数据类型，目标数据为数字时不建议使用字符串，目标数据可以存为树类型时不建议使用字符串。
  使用合理的数据类型，可以提高数据的查询效率。
  PostgreSQL支持的数据类型如下：精确的数字类型、浮点、货币、字符串、字符、字节流、日期、时间、布尔、枚举、几何、网络地址、比特流、文本、UUID、XML、JSON、数组、复合类型、范围类型、对象、行号、大对象、ltree树结构类型、cube多维类型、earth地球类型、hstore类型、pg_trgm相似类型、PostGIS（点、线段、面、路径、经纬度、raster、拓扑等）、HyperLogLog（近似估值统计分析）。
- 字符串排序规则LC_COLLATE推荐使用 'C'，而非 UTF8。LC_COLLATE=UTF8 性能相对差一些，并且索引需要明确指定UTF8 pattern ops，才能支持LIKE查询。

存储过程

- 如果业务逻辑冗长，建议减少数据库和程序之间的交互次数，使用数据库存储过程（如 PL/pgSQL）或内置函数。PostgreSQL内置的PL/pgSQL函数语言提供处理复杂业务逻辑的功能。PostgreSQL还内置了分析函数、聚合函数、窗口函数、普通类型函数、复杂类型函数、数学函数和几何函数等多种函数。

数据查询

- 不建议使用 `COUNT(列名)` 或 `COUNT(常量)` 来替代 `COUNT(*)` ， `COUNT(*)` 是SQL92定义的标准统计行数的语法，会统计NULL值（真实行数），而 `COUNT(列名)` 不会统计。
- 使用 `COUNT(多列列名)` 时，多列列名必须使用括号，例如 `COUNT( (col1,col2,col3) )` 。注意使用 `COUNT(多列列名)` 时，所有NULL行都会被计数，所以效果与 `COUNT(*)` 一致。
- 不建议使用 `SELECT * FROM t` ，用具体的字段列表代替 `*` ，避免返回用不到的字段。
- 除ETL（Extract-Transform-Load）程序外，建议避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。
- 对于需要范围查询的场景，建议使用范围类型以及GiST索引，提高范围检索的查询性能。
- 如果应用经常访问较大结果集的数据（例如100条），建议将数据聚合成1条，例如经常要按ID访问此ID的数据，建议定期按ID聚合数据，查询时返回的记录数越少响应越快。

管理

- 建议开启SQL洞察和审计，该功能支持查询并导出SQL语句执行历史及其对应的各种信息（数据库、状态、执行时间等），对SQL进行健康情况诊断、性能问题排查、业务流量分析等。更多信息，请参见SQL洞察和审计。
- 如果您需要监控并记录阿里云账号的活动，包括通过阿里云控制台、OpenAPI、开发者工具对云上产品和服务的访问和使用行为，建议使用操作审计功能。您还可以将这些行为事件下载或保存到日志服务或OSS存储空间，然后进行行为分析、安全分析、资源变更行为追踪和行为合规性审计等操作。更多信息，请参见什么是操作审计。

- DDL操作之前务必要经过评审，并且选择在低峰期执行。
- 删除和修改记录时，为避免误删除，建议先使用 `SELECT` 确认后，再提交执行。如果业务逻辑上确定只更改1行，则添加 `LIMIT 1` 。
- DDL操作（以及类似的可能获取锁的操作，例如 `VACUUM FULL` 、 `CREATE INDEX` 等）建议设置锁等待，用于防止堵塞所有与该DDL锁对象相关的查询。

```
begin; SET local lock_timeout = '10s'; -- DDL query; end;
```

- **EXPLAIN (ANALYZE)** 语句的工作方式类似于**EXPLAIN**，主要区别在于前者会实际执行SQL。如果SQL涉及数据变更，即DML SQL（**UPDATE**、**INSERT**或**DELETE**），务必在事务中执行**EXPLAIN (ANALYZE)**，查看完成后再进行回滚。

```
begin; EXPLAIN (ANALYZE) <DML(UPDATE/INSERT/DELETE) SQL>; rollback;
```

- 大批量删除和更新数据时，建议分批次操作，不建议在一个事务中完成，以免一次产生较多垃圾。

9.一键上云

9.1. 应用场景

RDS PostgreSQL提供一键上云功能，该功能采用物理流复制方式，具有速度快、操作简单、稳定性好，场景覆盖全等特点，能够有效提高迁移效率。该功能不仅可以实现阿里云ECS或IDC自建PostgreSQL数据库的迁移上云，还可以实现RDS PostgreSQL实例的跨地域、跨账号的迁移备份功能。本文介绍一键上云功能的应用场景。

一键上云功能适用于以下场景：

应用场景	源实例	目标实例	迁移链路	配置参考
• 迁移上云 自建PostgreSQL迁移上云。 • 云上只读 用户自建（IDC自建、ECS自建）作为源实例，RDS PostgreSQL作为云上的只读实例，扩展读取能力，分担负载。 • 云上灾备 用户自建（IDC自建、ECS自建）源实例故障时，RDS PostgreSQL支持手动强制切换为主库，业务延续读写能力。 • 公网迁移 具备公网IP或其他云厂商的PostgreSQL实例迁移至RDS PostgreSQL。	• 阿里云ECS自建PostgreSQL • IDC自建PostgreSQL • 具备公网IP或其他云厂商的PostgreSQL实例	RDS PostgreSQL实例	VPC、公网 • 如果源实例为阿里云ECS自建PostgreSQL，则ECS实例需要与RDS PostgreSQL实例处于相同VPC。如果VPC不同，则需要使用云企业网打通不同VPC，更多信息，请参见云企业网。 • 如果源实例为IDC自建PostgreSQL，则需要使用云企业网、VPN网关、高速通道或智能接入网关实现IDC与RDS PostgreSQL实例内网互通。 • 如果源实例具备公网IP或是其他云厂商的PostgreSQL实例，则可以通过公网实现迁移上云。	一键上云
• 跨地域迁移 RDS PostgreSQL实例间跨地域迁移。 • 异地灾备 RDS PostgreSQL实例间跨地域灾备组网，提升容灾能力。	RDS PostgreSQL实例	同账号异地RDS PostgreSQL实例	VPC 需要使用云企业网实现跨地域RDS PostgreSQL实例间内网互通。	RDS PostgreSQL实例跨地域迁移
跨账号迁移 不同账号下RDS PostgreSQL实例的迁移。	RDS PostgreSQL实例	跨账号RDS PostgreSQL实例	VPC 需要使用云企业网实现跨账号RDS PostgreSQL实例间内网互通。	RDS PostgreSQL实例跨账号迁移
RDS PostgreSQL实例缩容 将源实例通过全量+增量数据实时同步方式迁移至存储空间更小的实例，然后交换连接地址，实现缩容。	RDS PostgreSQL实例	RDS PostgreSQL实例	VPC RDS PostgreSQL实例处于相同VPC。	RDS PostgreSQL实例缩容

9.2. 上云前准备

9.2.1. （可选）ECS配置安全组

RDS PostgreSQL的一键上云功能支持将ECS自建PostgreSQL迁移到RDS PostgreSQL。本文介绍上云前，ECS自建PostgreSQL配置安全组，允许RDS PostgreSQL访问自建PostgreSQL数据库端口。

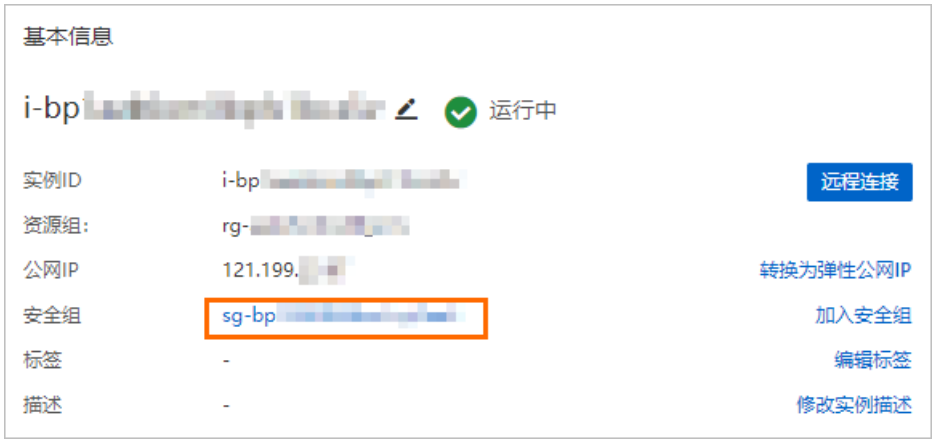
前提条件

如果您需要将ECS自建PostgreSQL迁移到RDS PostgreSQL，则需要执行本文配置。ECS需要满足以下条件：

- 网络：
  - 如果通过内网上云，ECS实例需要与RDS PostgreSQL实例处于相同VPC。
  - 如果通过公网上云，ECS实例需要具备公网IP 。
- ECS实例已部署PostgreSQL数据库。

操作步骤

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 在顶部菜单栏左上角处，选择地域。
4. 找到目标ECS实例，单击实例ID。
5. 在**实例详情**页签的**基本信息**区域，单击安全组链接。



6. 在**安全组**页面的**访问规则**区域，单击**入方向**页签中的**手动添加**按钮。添加一条安全组规则。

安全组中关键参数说明如下：

参数	取值说明
协议类型	TCP，固定配置。

参数	取值说明
端口范围	ECS自建PostgreSQL数据库的端口，可通过 `netstat -a | grep PGSQL` 命令查看。
授权对象	○ 通过内网上云，需要配置RDS PostgreSQL实例的**VPC网段**。 查看方法： a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。 b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。 c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL**后，单击**下一步**。 d. 在**目标库配置**步骤，查看RDS PostgreSQL实例的**当前VPC网段**。 ○ 通过公网上云，需要配置RDS PostgreSQL实例的**公网IP地址**。 查看方法： a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。 b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。 c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**具有公网地址的PostgreSQL迁移（含其他厂商的PostgreSQL迁移场景）**后，单击**下一步**。 d. 在**目标库配置**步骤，单击分配EIP。 e. 刷新后查看RDS PostgreSQL实例的**公网IP地址**。

后续步骤

配置postgresql.conf文件

9.2.2. 配置postgresql.conf文件

RDS PostgreSQL的一键上云功能支持 将部署在ECS或IDC的自建PostgreSQL迁移到RDS PostgreSQL 。本文介绍上云前， 自建PostgreSQL数据库postgresql.conf文件中的参数配置 。

操作步骤

> **说明** 本文以CentOS 7系统 安装 PostgreSQL 13为例进行配置。

1. 连接自建PostgreSQL，分别查看 listen_addresses 和 wal_keep_segments / wal_keep_size 的参数值。

参数	作用	查询命令	期望值
listen_addresses	查看自建PostgreSQL是否允许被远程连接。	`SHOW listen_addresses;`	`*`
wal_keep_segments	查看*pg_wal*目录下所能保留的过去日志文件段的最小数目，建议设置大于等于4096，防止全量备份完成之后由于源实例wal日志被移除导致需要重新备份。 **说明** 适用于PostgreSQL 10、11、12。	`SHOW wal_keep_segments;`	`4096`
wal_keep_size	查看*pg_wal*目录下所能保留的过去日志文件段的最小大小，建议设置大于等于64 GB，防止全量备份完成之后由于源实例wal日志被移除导致需要重新备份。 **说明** 适用于PostgreSQL 13、14。	`SHOW wal_keep_size;`	`65536`

   - 如果查询结果满足期望，则无需配置postgresql.conf文件，执行创建迁移账号。
   - 如果查询结果不满足期望，则执行步骤2。
2. 停止PostgreSQL数据库服务。

   > **说明** 该命令需要使用 `su - postgres` 命令，切换至postgres用户执行。

   ```
   /usr/pgsql-13/bin/pg_ctl stop -m fast
   ```

3. 获取postgresql.conf文件位置。

   > **说明** 该命令需要使用 `su - root` 命令，切换至root用户执行 。

   ```
   find / -name postgresql.conf
   ```

   结果示例：

   ```
   /var/lib/pgsql/13/data/postgresql.conf
   ```

4. 进入postgresql.conf文件目录。

```
cd /var/lib/pgsql/13/data/
```

5. 使用 `vim postgresql.conf` 命令，编辑**postgresql.conf**文件，分别修改 listen_addresses 和 wal_keep_segments / wal_keep_size 的参数值。

```
listen_addresses = '*' # PostgreSQL 10、11、12修改 wal_keep_segments = 4096 # PostgreSQL 13、14修改 wal_keep_size = 65536
```

> **说明** 参数可能默认被注释，修改后注意删除行首的 `#` 。

6. 按**ECS**，然后输入 `:wq` 保存修改并退出。
7. 启动PostgreSQL数据库服务。

> **说明** 该命令需要使用 `su - postgres` 命令，切换至postgres用户执行。

```
/usr/pgsql-13/bin/pg_ctl start
```

后续步骤

创建迁移账号

9.2.3. 创建迁移账号

RDS PostgreSQL的一键上云功能支持将ECS或IDC自建PostgreSQL迁移到RDS PostgreSQL。本文介绍上云前，自建PostgreSQL创建迁移账号，用于迁移时具有相关权限。

操作步骤

迁移账号需要具备CREATE ROLE、REPLICATION和pg_monitor权限。如果您已有相关权限的账号，可进行下一步：更新pg_hba.conf文件。

1. 连接自建PostgreSQL数据库，创建迁移账号， 以migratetest账号为例：

```
CREATE USER migratetest CREATEROLE REPLICATION LOGIN PASSWORD '123456';
```

> **说明** 命令中账号密码仅为示例，您可以自定义。

2. 为migratetest账号授权pg_monitor权限。

```
GRANT pg_monitor TO migratetest;
```

后续步骤

更新pg_hba.conf文件

9.2.4. 更新pg_hba.conf文件

RDS PostgreSQL的一键上云功能支持将ECS或IDC自建PostgreSQL迁移到RDS PostgreSQL。本文介绍上云前，自建PostgreSQL更新pg_hba.conf文件，允许迁移账号从RDS PostgreSQL的VPC网段访问。

操作步骤

> **说明**　本文以CentOS 7系统安装PostgreSQL 13为例进行配置。

1. 登录自建PostgreSQL数据库所在服务器。
2. 获取**pg_hba.conf**文件位置。

   > **说明**　该命令需要切换至root用户执行。

   ```
   find / -name pg_hba.conf
   ```

   结果示例：

   ```
   /var/lib/pgsql/13/data/pg_hba.conf
   ```

3. 进入**pg_hba.conf**文件目录。

   ```
   cd /var/lib/pgsql/13/data/
   ```

4. 使用 `vim pg_hba.conf` 命令，编辑**pg_hba.conf**文件，在末尾添加如下内容。

   ```
   # 通过内网上云 host all       migratetest 172.21.XX.XX/16 md5

           host replication migratetest 172.21.XX.XX/16 md5 # 通过公网上云
          host all       migratetest 121.41.XX.XX/32 md5

           host replication migratetest 121.41.XX.XX/32 md5
   ```

   关键参数解释：

   - `migratetest`：已创建的迁移账号。具体请参见创建迁移账号。
   - `172.21.XX.XX/16` 或 `121.41.XX.XX/32`：RDS PostgreSQL实例的VPC网段或公网IP。
     - 通过内网上云，需要配置RDS PostgreSQL实例的**VPC网段**。
       查看方法：
       a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
       b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。
       c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL**或**阿里云RDS PostgreSQL**或**线下IDC自建PostgreSQL(IDC与VPC打通)**后，单击**下一步**。
       d. 在**目标库配置**步骤，查看RDS PostgreSQL实例的**当前VPC网段**。

- 通过公网上云，需要配置RDS PostgreSQL实例的**公网IP地址**。
  查看方法：
  a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
  b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。
  c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**具有公网地址的PostgreSQL迁移（含其他厂商的PostgreSQL迁移场景）**后，单击**下一步**。
  d. 在**目标库配置**步骤，单击分配EIP。

  e. 刷新后查看RDS PostgreSQL实例的**公网IP地址**。

5. 连接自建PostgreSQL，重新加载配置。

```
SELECT pg_reload_conf();
```

返回结果示例：

```
 pg_reload_conf ---------------- t (1 row)
```

后续步骤

配置服务器防火墙

9.2.5. 配置服务器防火墙

RDS PostgreSQL的一键上云功能支持将ECS或IDC自建PostgreSQL迁移到RDS PostgreSQL。本文介绍上云前，自建PostgreSQL配置服务器防火墙，设置数据库端口允许被访问。

操作步骤

> **说明** 本文以CentOS系统为例进行配置，其他系统请参见对应官方文档。

CentOS 7

1. 连接自建PostgreSQL所在服务器。
2. 查看已开放端口。

```
firewall-cmd --list-ports
```

3. 设置PostgreSQL数据库端口允许被访问。

```
firewall-cmd --zone=public --add-port=5432/tcp --permanent
```

4. 重启防火墙。

```
firewall-cmd --reload
```

CentOS 6及以下

1. 连接自建PostgreSQL所在服务器。
2. 查看已开放端口。

```
/etc/init.d/iptables status
```

3. 开放5432端口。

```
/sbin/iptables -I INPUT -p tcp --dport 85432 -j ACCEPT
```

4. 重启防火墙。

```
service iptables restart
```

说明 你也可以不修改防火墙配置，在上云前关闭防火墙。

- CentOS 7：

```
systemctl stop firewalld.service
```

- CentOS 6或以下版本：

```
service iptables stop
```

后续步骤

一键上云

9.3. 一键上云

本文介绍如何通过RDS PostgreSQL的一键上云功能将ECS或IDC自建PostgreSQL数据库迁移到RDS PostgreSQL。该功能采用物理流复制方式，具有速度快、操作简单、稳定性好、场景覆盖全等特点，能够有效提高迁移效率。

前提条件

- RDS PostgreSQL实例需满足以下条件。
  - 实例大版本与自建PostgreSQL相同，当前支持PostgreSQL 14、13、12、11和10。
  - 实例需要为主实例，只读实例不支持一键上云。
  - 实例的存储类型为云盘。
  - 实例为空，无数据，可用存储空间大于等于自建PostgreSQL中数据大小的总和。
- 自建PostgreSQL需要满足以下条件。

- 网络

迁移来源	网络配置要求
阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL	ECS实例或源RDS PostgreSQL实例需要与目标RDS PostgreSQL实例处于相同VPC。如果VPC不同，则需要使用云企业网打通，更多信息，请参见云企业网。
线下IDC自建PostgreSQL（IDC与VPC打通）	需要打通IDC与RDS PostgreSQL实例之间的内网连接。配置方法请参见VPC连接本地IDC。
具有公网地址的PostgreSQL迁移（含其他厂商的PostgreSQL迁移场景）	需要源实例具备公网IP。

- 如果从ECS自建PostgreSQL迁移至RDS PostgreSQL实例，需先完成（可选）ECS配置安全组。
- 已完成配置：配置postgresql.conf文件。
- 已完成配置：创建迁移账号。
- 已完成配置：更新pg_hba.conf文件。
- 已完成配置：配置服务器防火墙。

注意事项

迁移上云任务期间可以对自建PostgreSQL数据库进行读写，但请勿执行迁移、重启、变配等操作。

步骤一：上云评估

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。

3. 在配置向导的**选择迁移来源**步骤，选择迁移来源后，单击**下一步**。
4. 在**目标库配置**步骤，单击**下一步**。
5. 在**源库配置**步骤，选中所有已完成的配置准备项，单击**下一步**。

> **说明** 源库配置步骤的具体操作，请参见（可选）ECS配置安全组、创建迁移账号和更新pg_hba.conf文件。

6. 在**发起迁移评估**步骤，配置源库信息。

参数	取值说明
迁移任务名	系统自动生成，无需修改。

参数	取值说明
源端 VPC IP/DNS或源端 公网IP/DNS	○ **源端 VPC IP/DNS** ■ ECS自建PostgreSQL数据库一键上云场景，配置ECS实例的**私网IP**。获取方法请参见查看IP地址。 ■ IDC自建PostgreSQL数据库一键上云场景，配置为IDC的内网IP。 ○ **源端 公网IP/DNS** 公网PostgreSQL一键上云场景，配置为源实例的公网IP。
源端Port	自建PostgreSQL数据库的端口，可通过 `netstat -a | grep PGSQL` 命令查看。
用户名	`migratetest` ，**源库配置**步骤创建的数据库账号。
密码	`123456` ，**源库配置**步骤创建的数据库账号的密码。

7. 单击**创建迁移评估任务**。

> **说明**　迁移评估任务期间，实例状态将变更为 **维护实例中**。

迁移评估完成后，您可以在**上云评估**页面查看迁移评估任务的状态。

- 只有 状态 为**成功**，您才能进行迁移上云步骤，具体请参见步骤二：迁移上云。
- 如果 状态 为**失败**，请单击操作列的**查看报告**，根据报错进行处理，常见报错请参见解读上云评估报告。

处理报错后，您可以单击操作列的**重新评估**，重新开始评估任务。

步骤二：迁移上云

> **说明**　只有上云评估状态为成功时，才能执行本步骤。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**一键上云**，切换至**迁移上云**页签，单击创建迁移上云任务。

3. 在**创建迁移上云任务**窗口中，从**关联评估任务**列表中选择步骤一：上云评估中已成功的上云评估任务。

说明 选择**关联评估任务**后，**源库类型**、**源端 IP/DNS**、**源端Port**和**用户名**参数将会自动获取，无需配置。

4. 单击**发起迁移上云**，系统将自动启动上云任务。

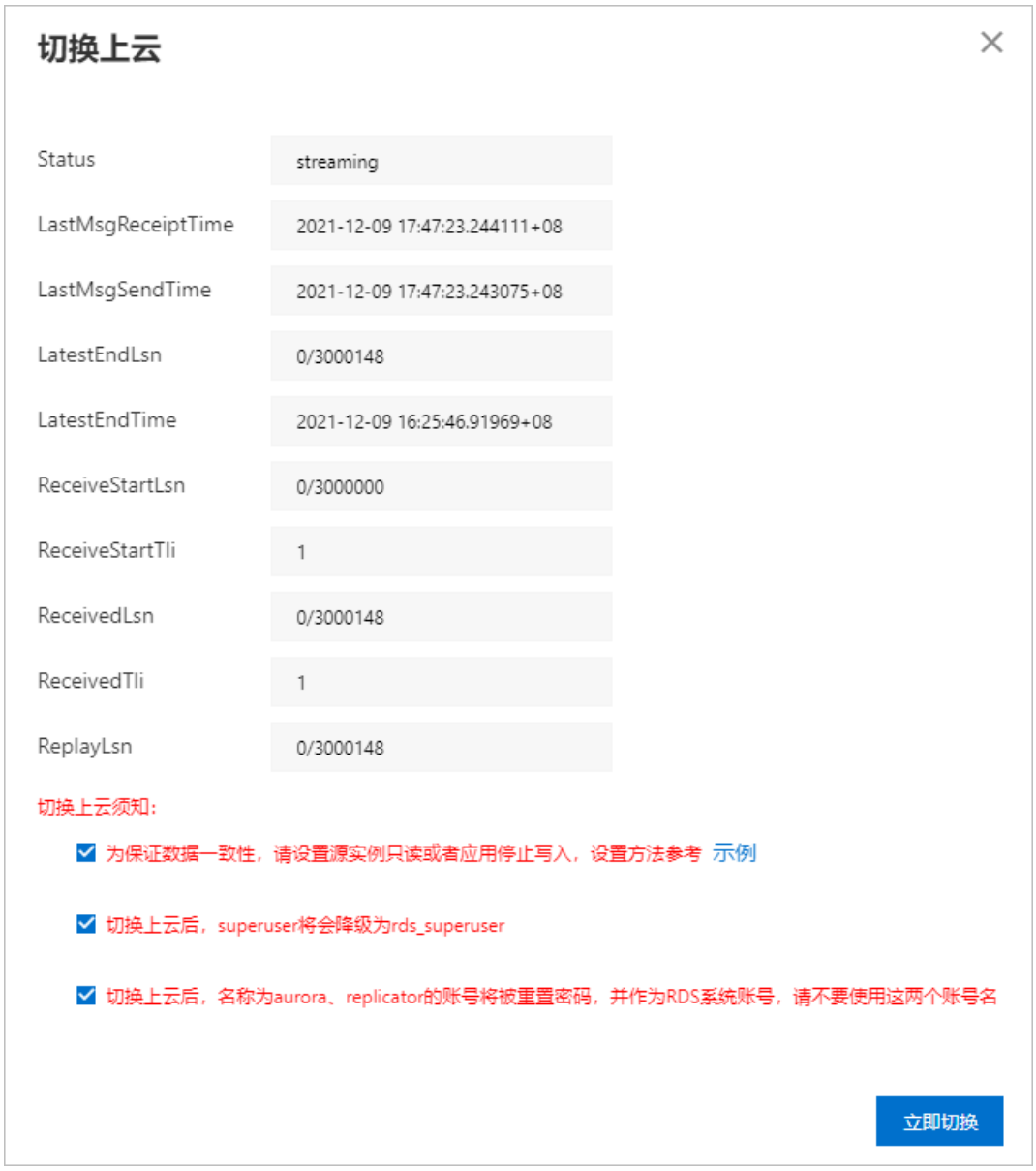
注意 上云任务期间，实例状态将变更为**迁入数据中**。您可以对自建PostgreSQL数据库进行读写，但请勿执行迁移、重启、变配等操作。

5. 上云切换。
   i. 在迁移上云任务中，单击**上云阶段**列的链接，查看当前上云任务进度。

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
ucujgmj2dei	ECS自建	172.21.	5432	migratetest	2021年12月9日 16:15:10	/	2021年12月9日 17:43:33	增量同步	流式传输	运行中	查看日志 上云切换

   ii. 当上云阶段为**增量同步**时，可单击**操作**列的**上云切换**，将RDS PostgreSQL提升为主库，正式提供服务。

iii. 在**切换上云**窗口，根据上云须知提示，设置源实例只读或应用停止写入。



说明　设置自建PostgreSQL数据库只读使用superuser用户配置如下命令：

```
ALTER SYSTEM SET default_transaction_read_only=on; SELECT pg_reload_conf();
```

iv. 选中所有复选框，单击**立即切换**，等待迁移完成。

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
ucujgmj2dei	ECS自建	172.21	5432	migratetest	2021年12月9日 16:15:10	2021年12月9日 17:51:48	2021年12月9日 17:53:33	上云完成	/	成功	查看日志

9.4. 解读上云评估报告

本文介绍RDS PostgreSQL一键上云时上云评估报告的内容详情、常见报错以及处理方式。

如果RDS PostgreSQL一键上云的上云评估报告的**状态**为**失败**，您可以单击上云评估任务对应**操作**列的**查看报告**，查看详细内容，上云评估报告包含以下内容。

- Check rds empty（检查RDS PostgreSQL数据库是否是空库）

- Check source connectivity（检查自建PostgreSQL的连通性）
- Check source version（检查自建PostgreSQL的大版本）
- Check source glibc version（检查自建PostgreSQL的GLIBC版本）
- Check disk size（检查磁盘大小是否充足）
- Check wal keep size（检查wal_keep_size大小）
- Check spec params（检查规格参数）
- Check rds user（检查RDS系统账号是否在源端实例上被占用）
- Check extensions（检查插件兼容性）
- Check triggers（检查触发器是否兼容）
- Check user functions（检查用户自定义函数）

> **说明** RDS PostgreSQL一键上云的更多信息，请参见 一键上云。

Check rds empty（检查RDS PostgreSQL数据库是否是空库）

检查内容

Check rds databases

常见报错

```
error:postgres not empty, check if any table exists
```

说明

RDS PostgreSQL实例创建了数据库并有数据。

解决办法

清空RDS PostgreSQL实例的数据库，仅保留template0、template1和postgres，并且，postgres需要清理除了ha_health_check以外的表。

Check source connectivity（检查自建PostgreSQL的连通性）

- **检查内容1**
  Check ip connectable
  常见报错

  ```
  error:XX.XX.XX.XX is unapproachable
  ```

  说明
  自建PostgreSQL所在服务器IP或DNS无法ping通。
  解决办法
  - ECS自建PostgreSQL一键上云场景，配置ECS实例的**私网IP**。获取方法请参见查看IP地址。
  - IDC自建PostgreSQL一键上云场景，配置为DNS的IP。

- **检查内容2**
  Check port connectable
  常见报错

  ```
  error:5432 is unapproachable
  ```

  说明

- 自建PostgreSQL未开启远程端口监听。
- 自建PostgreSQL设置了防火墙，未设置数据库端口允许被访问。

解决办法

- 修改自建PostgreSQL的**postgresql.conf**文件，配置 `listen_addresses = '*'` 。具体操作请参见配置postgresql.conf文件。
- 设置防火墙，开放5432端口，或上云前关闭防火墙。具体操作请参见配置服务器防火墙。

- **检查内容3**
  Check database connectable
  常见报错

```
error:cannot connect to source database by migratetest:123456
```

说明

- 用户密码不正确。
- 自建PostgreSQL的**pg_hba.conf**配置不正确。

解决办法

- 确认用户名密码是否能连接自建PostgreSQL，无法连接时，可更新用户密码。以migratetest账号为例，命令示例如下：

```
ALTER USER migratetest WITH PASSWORD '123456';
```

- 修改自建PostgreSQL的**pg_hba.conf**文件，以migratetest账号为例，配置如下内容：

```
host all       migratetest <RDS PostgreSQL VPC网段> md5
```

> **说明** 具体操作请参见 更新pg_hba.conf文件。

- **检查内容4**
  Check account replication privilege
  常见报错

```
error:migratetest has no replication privilege
```

说明

- 用户没有REPLICATION权限。
- 自建PostgreSQL的**pg_hba.conf**配置不正确。

解决办法

- 以migratetest账号为例，执行如下命令，为账号添加REPLICATION权限：

```
ALTER ROLE migratetest REPLICATION;
```

- 修改自建PostgreSQL的**pg_hba.conf**文件，以migratetest账号为例，配置如下内容：

```
host replication migratetest <RDS PostgreSQL VPC网段> md5
```

> **说明** 具体操作请参见 更新pg_hba.conf文件。

- **检查内容5**
  Check account createrole privilege
  常见报错

  ```
  error:migratetest has no createrole privilege
  ```

  说明
  用户没有CREATEROLE权限。
  解决办法
  以migratetest账号为例，执行如下命令：

  ```
  ALTER ROLE migratetest CREATEROLE;
  ```

- **检查内容6**
  Check account monitor privilege
  常见报错

  ```
  error:migratetest should be a member of pg_monitor to monitor replication status
  ```

  说明
  用户没有pg_monitor权限。

  > **说明** 该权限能够查询pg_stat_replication和pg_stat_wal_receiver等系统视图，以便于得到复制链路信息。

  解决办法
  以migratetest账号为例，为账号授权：

  ```
  GRANT pg_monitor TO migratetest;
  ```

Check source version（检查自建PostgreSQL的大版本）

检查内容

Check major version consistent

常见报错

```
error:version mismatch, source version:10, current version:13.0
```

说明

自建PostgreSQL版本与RDS PostgreSQL实例大版本不一致。

解决办法

购买与自建PostgreSQL大版本一致的RDS PostgreSQL实例。

Check source glibc version（检查自建PostgreSQL的GLIBC版本）

检查内容

Check source glibc version compatible

常见报错

```
warning:source glibc version is not compatible with rds pg
```

说明

自建PostgreSQL的GLIBC版本与RDS PostgreSQL实例的GLIBC版本不兼容。

> **说明** GLIBC 2.28版本UTF8部分字符排序规则变化，版本不兼容时，数据排序规则不同，存在排序结果不符合预期的风险。

解决办法

按照如下步骤检查。

1. 检查表排序。

```
begin; create temp table testcollation(id varchar(20) collate "en_US.utf8") on commit d
rop; insert into testcollation values('-1'),('1'); select id='1' from testcollation ord
er by id limit 1; rollback;
```

   返回结果：
   - true：结束，上云无风险。
   - false：进行下一步检查。

2. 检查库Collation。

```
SELECT datname,datcollate FROM pg_database where datcollate NOT IN ('C', 'POSIX');
```

   返回结果：
   - 返回为空：结束，上云无风险。
   - 返回非空：进行下一步检查。

3. 在所有库下检查Collation不是 `C` 或者 `POSIX` 的索引。

```
WITH result AS ( WITH defcoll AS ( SELECT datcollate AS coll FROM pg_database WHERE dat
name = current_database() ) SELECT indrelid::regclass::text relname, indexrelid::regcla
ss::text indexname, CASE WHEN c.collname = 'default' THEN defcoll.coll ELSE c.collname
END AS collation FROM (SELECT indexrelid, indrelid, indcollation[i] coll FROM pg_index,
generate_subscripts(indcollation, 1) g(i)) s JOIN pg_collation c ON coll=c.oid CROSS JO
IN defcoll WHERE collprovider IN ('d', 'c') AND collname NOT IN ('C', 'POSIX') ) SELECT
result.relname, result.indexname, result.collation FROM result WHERE result.collation N
OT IN ('C', 'POSIX');
```

   返回结果：
   - 返回为空：结束，上云无风险。
   - 返回非空：上云存在风险。

Check disk size（检查磁盘大小是否充足）

检查内容

Check disk size enough

常见报错

```
error:source_db_size > disk_size * 0.95
```

说明

自建PostgreSQL数据库的大小大于RDS PostgreSQL实例磁盘大小的95%，存储空间不足。

解决办法

1. 使用如下命令查看自建PostgreSQL的存储空间。

```
SELECT SUM(pg_database_size(pg_database.datname))/1024/1024 AS size FROM pg_database;
```

> **说明** 该命令查询结果单位为MB。

2. 计算迁移上云的RDS PostgreSQL实例需要的存储空间大小。
   例如：查询自建PostgreSQL的存储空间为100 GB，那么，RDS PostgreSQL的存储空间至少应调整为110 GB。
3. 对RDS PostgreSQL实例进行变更配置，扩容存储空间。具体请参见变更配置。

Check wal keep size（检查wal_keep_size大小）

检查内容

Check wal keep size large enough

常见报错

```
warning:wal_keep_size X MB is too small. Try to set wal_keep_segments or wal_keep_size larg
e enough ensure pg_basebackup success
```

说明

wal_keep_size 或 wal_keep_segments 参数设置过小，需要调整。

解决办法

- 自建PostgreSQL版本大于等于13：调大RDS PostgreSQL实例的 wal_keep_size 参数取值，增加上云过程中全量备份和增量备份的成功率。
- 自建PostgreSQL版本小于13：调大RDS PostgreSQL实例的 wal_keep_segments 参数值，增加上云过程中全量备份和增量备份的成功率。

> **说明** PostgreSQL版本小于13，wal_keep_size = wal_keep_segments * wal_segment_size。

Check spec params（检查规格参数）

检查内容

Check if spec params too large

常见报错

```
error:max_connections too large, value=XXX error:max_prepared_transactions too large, value
=XXX
```

说明

自建PostgreSQL的max_connections和max_prepared_transactions参数设置过大，大于RDS PostgreSQL的100倍，可能导致搭建复制链路过程中云上实例无法启动。

解决办法

调小自建PostgreSQL的 max_connections 和 max_prepared_transaction 的取值。

> **说明** max_connections 和 max_prepared_transaction 参数变更，需要重启自建PostgreSQL。

Check rds user（检查RDS系统账号是否在源端实例上被占用）

检查内容

Check if rds system user is occupied

常见报错

```
warning:Check if rds system user is occupied ...XXX will be reused in rds
```

说明

自建PostgreSQL的 `aurora` 、 `replicator` 和 `pgxxx` 账号将会被RDS PostgreSQL占用，请不要使用这三个账号作为业务账号。

解决办法

请确认不要使用warning中的账号作为业务账号。

Check extensions（检查插件兼容性）

- **检查内容1**
  Check source supported extensions
  常见报错

  ```
  error:Check source supported extensions XXX not supported
  ```

  说明
  RDS PostgreSQL实例插件与自建PostgreSQL不兼容。
  解决办法
  在自建PostgreSQL中删除不兼容的插件。
- **检查内容2**
  Check source extensions with higher version
  常见报错

  ```
  error:Check source extensions with higher version XXX
  ```

  说明
  自建PostgreSQL插件版本高于RDS PostgreSQL。
  解决办法
  自建PostgreSQL安装低版本插件，使插件版本与RDS PostgreSQL版本一致。
- **检查内容3**
  Check source extensions with lower version
  常见报错

  ```
  warning:Check source extensions with lower version XXX
  ```

  说明
  自建PostgreSQL插件版本低于RDS PostgreSQL。
  解决办法
  无需处理，迁移上云后，将会自动升级插件版本。

Check triggers（检查触发器是否兼容）

检查内容

Check triggers compatible

常见报错

```
Check triggers compatible XXX
```

说明

当前检查项仅做检查，不会报错。

解决办法

不涉及。

Check user functions（检查用户自定义函数）

检查内容

Check user functions compatible

常见报错

```
Check user functions compatible XXX
```

说明

当前检查项仅做检查，不会报错。

解决办法

不涉及。

9.5. 云上应用场景

9.5.1. RDS PostgreSQL实例跨地域迁移

RDS PostgreSQL的一键上云功能不仅支持阿里云ECS或IDC自建PostgreSQL的一键上云，还支持RDS PostgreSQL实例跨地域迁移，本文介绍如何使用一键上云功能，实现RDS PostgreSQL实例间的跨地域迁移。

前提条件

RDS PostgreSQL实例需要满足以下条件：

- 源实例与目标实例大版本相同，当前支持PostgreSQL 14、13、12、11和10。
- 目标实例为主实例，只读实例不支持一键上云。
- 目标实例的存储类型为云盘。
- 目标实例为空，无数据，可用存储空间大于等于源实例中数据大小的总和。
- 源实例和目标实例所使用的VPC网段不能冲突。

注意事项

RDS PostgreSQL实例跨地域迁移时，需要使用云企业网实现跨地域内网互通，会产生相关费用，具体请参见计费概览。

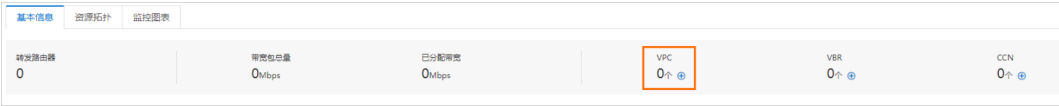
操作步骤

> **说明** 本文以华北2（北京）的RDS PostgreSQL实例迁移到华东1（杭州）为例。

1. 云企业网配置，使不同地域的两个RDS PostgreSQL实例内网互通。
   i. 创建云企业网实例。
      a. 登录云企业网管理控制台。
      b. 在**云企业网实例**页面，单击**创建云企业网实例**。
      c. 在**创建云企业网实例**页面，根据以下信息配置云企业网实例，然后单击**确认**。

配置	说明
名称	云企业网实例的名称。 名称在2~128个字符之间，以大小写字母或中文开头，可包含数字、短划线（-）或下划线（_）。
描述	云企业网实例的描述。 描述可以为空或可以填写2~256个中英文字符，不能以 `http://` 和 `https://` 开头。

   ii. 为云企业网实例添加VPC。
      a. 在**云企业网实例**页面，单击已创建的云企业网实例ID。
      b. 在**基本信息**页面，单击下图中VPC数量后的 ⊕ 图标，为云企业网实例添加VPC。

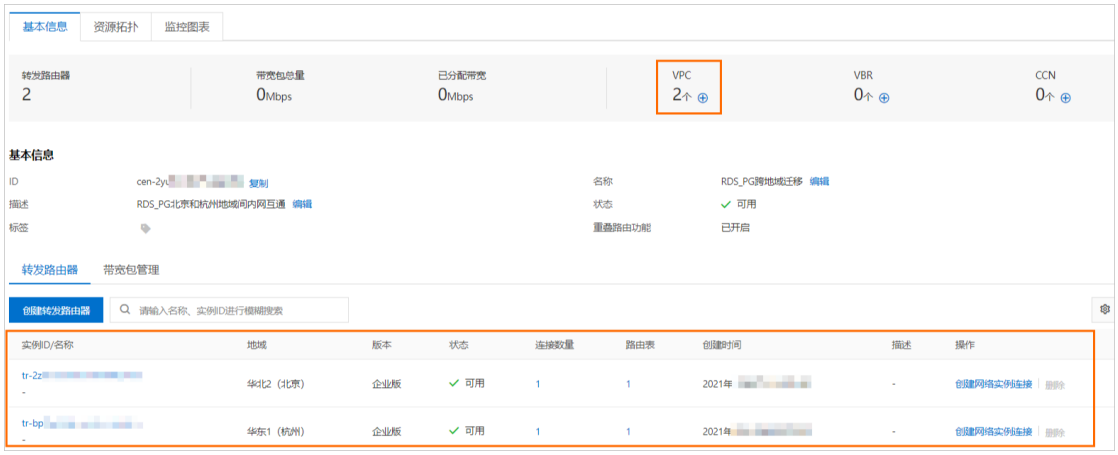


c. 在**连接网络实例**页面，配置网络实例信息后，单击**确认创建**。

> **说明** 本步骤需要 分别添加源实例和目标实例的VPC，因此需要添加两次，下表以添加源实例的VPC为例，添加目标实例VPC的操作方法相同，**地域**和**网络实例**参数，请修改为目标实例的地域和VPC。

参数	取值说明
实例类型	默认为**专有网络（VPC）**，无需修改。
地域	选择源实例所在地域，本示例选择**华北2（北京）**。
转发路由器	默认当前地域下无转发路由器，系统将默认创建，分别选择转发路由器的主可用区和备可用区。
资源归属UID	选择**同账号**。
连接名称	自定义。
网络实例	选择源实例使用的VPC实例。可在RDS管理控制台的数据库连接中查看VPC ID。 选择VPC后，分别选择转发路由器配置的主备可用区对应的交换机。

创建完成后，结果示例如下：



iii. 购买跨地域带宽。

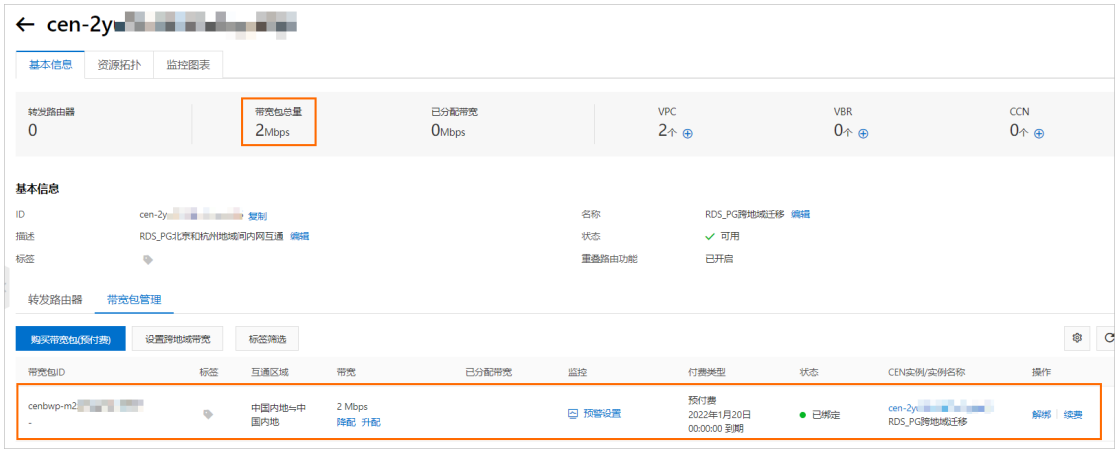
a. 在**云企业网实例**页面，单击已创建的云企业网实例ID。

b. 在**基本信息**页面，切换到**带宽包管理**页签，单击**购买带宽包（预付费）**。

c. 在购买页面，根据以下信息配置带宽包，然后单击**立即购买**并完成支付。

配置项	说明
云企业网	默认选择当前云企业网，无需修改。带宽包购买成功后，带宽包自动绑定至该云企业网实例。
区域-A	选择参与互通的网络实例所在区域。本示例选择**中国内地**。 **说明**　带宽包创建后，不支持修改互通区域。
区域-B	选择参与互通的网络实例所在区域。本示例选择**中国内地**。
计费方式	显示带宽包的计费方式。默认为**按带宽**计费。
带宽值	选择带宽值。单位：Mbps。
带宽包名称	输入带宽包的名称。
购买时长	选择带宽包的购买时长。 选中**到期自动续费**可开启带宽包自动续费功能。

购买完成后，结果示例如下：



iv. 设置跨地域带宽。

a. 在**云企业网实例**页面，单击已创建的云企业网实例ID。

b. 在**基本信息**页面，切换到**带宽包管理**页签，单击**设置跨地域带宽**。

c. 在**连接网络实例**页面，配置跨地域连接信息，然后单击**确定创建**。

配置项	说明
实例类型	选择**跨地域**。
地域	源实例的地域。本示例选择**华北2（北京）**。
转发路由器	自动获取已创建的转发路由器，无需配置。
对端地域	目标实例的地域，本示例选择**华东1（杭州）**。
转发路由器	自动获取已创建的转发路由器，无需配置。
带宽包实例	选择云企业网实例已绑定的带宽包实例。
带宽	输入允许使用的带宽值。单位：Mbps。

创建成功后，结果示例如下：

2. 源实例配置。

i. 添加白名单。

添加白名单的具体操作，请参见 设置白名单， **组内白名单**请填写目标实例的VPC网段。
目标实例的VPC网段查看方法：

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。

c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL**后，单击**下一步**。

d. 在**目标库配置**步骤，查看目标实例的**当前VPC网段**。

ii. 创建高权限账号。

创建账号的具体操作，请参见 创建账号。 **账号类型**请选择高权限账号。

> **说明**　高权限账号用于迁移上云，需要具备CREATE ROLE、REPLICATION和pg_monitor权限。如果您已有高权限的账号，则无需重复创建。

3. 目标实例配置。

i. 上云评估。

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。

c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL**后，单击**下一步**。

d. 在**目标库配置**步骤，单击**下一步**。

e. 在**源库配置**步骤，选中所有配置准备项，单击**下一步**。

f. 在**发起迁移评估**步骤，配置源库信息。

参数	取值说明
迁移任务名	系统自动生成，无需修改。
源端 VPC IP/DNS	填写源实例的内网连接地址。具体请参见查看内网地址。
源端Port	填写源实例的端口，具体请参见查看端口。
用户名	源实例中创建的高权限数据库账号。
密码	源实例中创建的高权限数据库账号的密码。

g. 单击**创建迁移评估任务**。

> **说明**　迁移评估任务期间，实例状态将变更为 **维护实例中**。

迁移评估完成后，您可以在**上云评估**页面查看迁移评估任务的状态。

- 只有 状态 为**成功**，您才能进行迁移上云步骤。
- 如果 状态 为**失败**，请单击操作列的**查看报告**，根据报错进行处理，常见报错请参见解读上云评估报告。

任务名	迁移源类型	IP	Port	账号	创建时间	结束时间	状态	操作
eacetdkwiws	ECS自建	pgm-[illegible].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[illegible]	2021年12月[illegible]	成功	查看报告

ii. 迁移上云。

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，切换至**迁移上云**页签，单击创建迁移上云任务。

c. 在**创建迁移上云任务**窗口中，从**关联评估任务**列表中选择已成功的上云评估任务。

> **说明** 选择**关联评估任务**后，**源库类型**、**源端IP/DNS**、**源端Port**和**用户名**参数将会自动获取，无需配置。

d. 单击**发起迁移上云**，系统将自动启动上云任务。

> **注意** 上云任务期间，实例状态将变更为**迁入数据中**。您可以对源实例进行读写，但请勿执行迁移、重启、变配等操作。

创建成功后，结果示例如下：

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	/	2021年12月[redacted]	预检查	/	运行中	查看日志

iii. 将目标RDS PostgreSQL提升为主库。

a. 在迁移上云任务中，单击**上云阶段**列的链接，查看当前上云任务进度。

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	/	2021年12月[redacted]	增量同步	流式传输	运行中	查看日志 上云切换

b. 当上云阶段为**增量同步**时，可单击**操作**列的**上云切换**，将目标RDS PostgreSQL提升为主库，正式提供服务。

c. 在**切换上云**窗口，根据上云须知提示，设置源实例只读或应用停止写入。

> **说明** 使用高权限用户配置如下命令，设置源实例只读：
>
> ```
> ALTER SYSTEM SET default_transaction_read_only=on; SELECT pg_reload_conf();
> ```

d. 选中所有复选框，单击**立即切换**，等待迁移完成。
切换完成后，结果示例如下：

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	2021年12月[redacted]	2021年12月[redacted]	上云完成	/	成功	查看日志

9.5.2. RDS PostgreSQL实例跨账号迁移

RDS PostgreSQL的一键上云功能不仅支持阿里云ECS或IDC自建PostgreSQL的一键上云，还支持RDS PostgreSQL实例跨账号迁移，本文介绍如何使用一键上云功能，实现RDS PostgreSQL实例间的跨账号迁移。

前提条件

- RDS PostgreSQL实例需要满足以下条件：
  - 源实例与目标实例大版本相同，当前支持PostgreSQL 14、13、12、11和10。
  - 目标实例为主实例，只读实例不支持一键上云。
  - 目标实例的存储类型为云盘。
  - 目标实例为空，无数据，可用存储空间大于等于源实例中数据大小的总和。
  - 源实例和目标实例所使用的VPC网段不能冲突。

- 跨账号迁移需要配置跨账号网络实例授权，具体请参见跨账号网络实例授权。

注意事项

RDS PostgreSQL实例跨账号迁移时，需要使用云企业网实现跨账号内网互通，会产生相关费用，具体请参见计费概览。

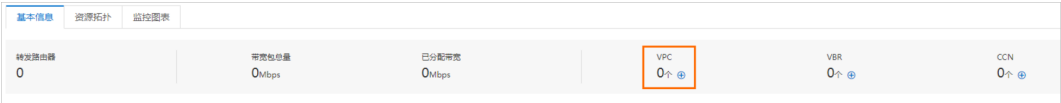
操作步骤

> **说明** 本文以A账号下RDS PostgreSQL实例迁移到B账号下为例。源实例位于A账号下，目标实例和云企业网位于B账号下。

1. 在B账号配置云企业网，使不同账号的两个RDS PostgreSQL实例内网互通。
   i. 创建云企业网实例。
      a. 登录云企业网管理控制台。
      b. 在**云企业网实例**页面，单击**创建云企业网实例**。
      c. 在**创建云企业网实例**页面，根据以下信息配置云企业网实例，然后单击**确认**。

配置	说明
名称	云企业网实例的名称。 名称在2~128个字符之间，以大小写字母或中文开头，可包含数字、短划线（-）或下划线（_）。
描述	云企业网实例的描述。 描述可以为空或可以填写2~256个中英文字符，不能以 `http://` 和 `https://` 开头。

   ii. 为云企业网实例添加VPC。
      a. 在**云企业网实例**页面，单击已创建的云企业网实例ID。
      b. 在**基本信息**页面，单击下图中VPC数量后的 ⊕ 图标，为云企业网实例添加VPC。



c. 在**连接网络实例**页面，配置网络实例信息后，单击**确认创建**。

> **说明** 本步骤需要分别添加源实例和目标实例的VPC，因此需要添加两次，下表以添加源实例的VPC为例，添加目标实例VPC的操作方法相同，**地域**和**网络实例**参数，请修改为目标实例的地域和VPC，**资源归属UID**选择同账号。

参数	取值说明
实例类型	默认为**专有网络（VPC）**，无需修改。
地域	选择源实例所在地域。
转发路由器	默认当前地域下无转发路由器，系统将默认创建，分别选择转发路由器的主可用区和备可用区。
资源归属UID	选择**跨账号**后，填写源实例的**账号ID**。 **说明** 您可以在顶部菜单栏单击用户图标，查看账号ID。
连接名称	自定义。
网络实例	选择源实例使用的VPC实例。可在RDS管理控制台的数据库连接中查看VPC ID。 选择VPC后，分别选择转发路由器配置的主备可用区对应的交换机。

创建完成后，结果示例如下：

iii. （可选）购买跨地域带宽。

> **说明** 如果账号A和账号B的RDS PostgreSQL地域不同，例如账号A的源实例在华东2（北京），账号B的目标实例在华东1（杭州），还需要执行本步骤，如果地域相同，请跳过此步骤，执行步骤2：源实例配置。

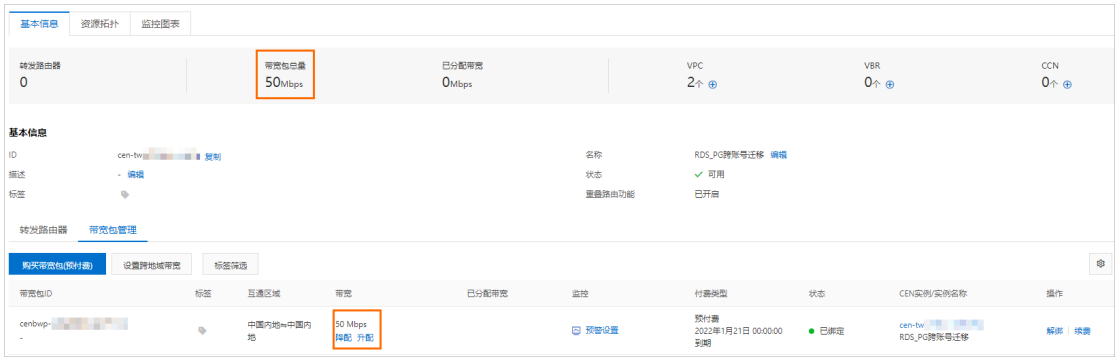
a. 在**云企业网实例**页面，单击已创建的云企业网实例ID。

b. 在**基本信息**页面，切换到**带宽包管理**页签，单击**购买带宽包（预付费）**。

c. 在购买页面，根据以下信息配置带宽包，然后单击**立即购买**并完成支付。

配置项	说明
云企业网	默认选择当前云企业网，无需修改。带宽包购买成功后，带宽包自动绑定至该云企业网实例。
区域-A	选择参与互通的网络实例所在区域。本示例选择**中国内地**。 **说明** 带宽包创建后，不支持修改互通区域。
区域-B	选择参与互通的网络实例所在区域。本示例选择**中国内地**。
计费方式	显示带宽包的计费方式。默认为**按带宽**计费。
带宽值	选择带宽值。单位：Mbps。
带宽包名称	输入带宽包的名称。
购买时长	选择带宽包的购买时长。 选中**到期自动续费**可开启带宽包自动续费功能。

购买完成后，结果示例如下：



iv. （可选）设置跨地域带宽。

> **说明** 如果账号A和账号B的RDS PostgreSQL地域不同，例如账号A的源实例在华东2（北京），账号B的目标实例在华东1（杭州），还需要执行本步骤，如果地域相同，请跳过此步骤，执行步骤2：源实例配置。

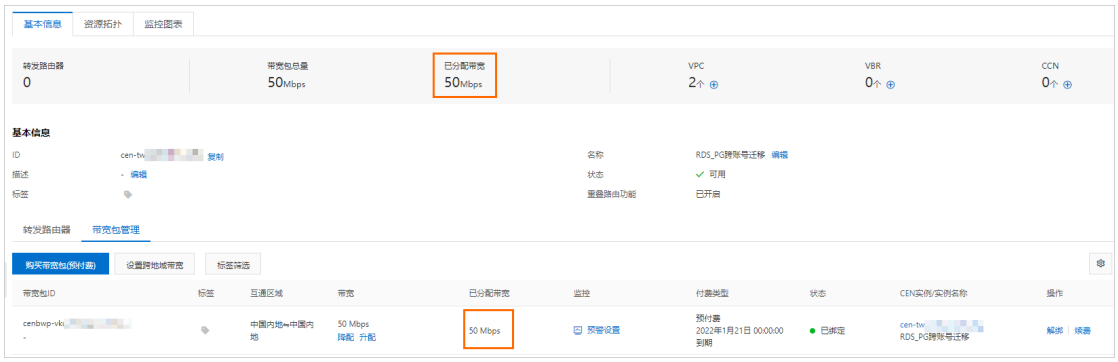
a. 在**云企业网实例**页面，单击已创建的云企业网实例ID。

b. 在**基本信息**页面，切换到**带宽包管理**页签，单击**设置跨地域带宽**。

c. 在**连接网络实例**页面，配置跨地域连接信息，然后单击**确定创建**。

配置项	说明
实例类型	选择**跨地域**。
地域	源实例的地域。本示例选择**华北2（北京）**。
转发路由器	自动获取已创建的转发路由器，无需配置。
对端地域	目标实例的地域，本示例选择**华东1（杭州）**。
转发路由器	自动获取已创建的转发路由器，无需配置。
带宽包实例	选择云企业网实例已绑定的带宽包实例。
带宽	输入允许使用的带宽值。单位：Mbps。

创建成功后，结果示例如下：



2. 源实例配置。

i. 添加白名单。

添加白名单的具体操作，请参见 设置白名单， **组内白名单**请填写目标实例的VPC网段。
目标实例的VPC网段查看方法：

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。

c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL**后，单击**下一步**。

d. 在**目标库配置**步骤，查看目标实例的**当前VPC网段**。

ii. 创建高权限账号。

创建账号的具体操作，请参见 创建账号。 **账号类型**请选择高权限账号。

> **说明**　高权限账号用于迁移上云，需要具备CREATE ROLE、REPLICATION和pg_monitor权限。如果您已有高权限的账号，则无需重复创建。

3. 目标实例配置。

i. 上云评估。

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。

c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL**后，单击**下一步**。

d. 在**目标库配置**步骤，单击**下一步**。

e. 在**源库配置**步骤，选中所有配置准备项，单击**下一步**。

f. 在**发起迁移评估**步骤，配置源库信息。

参数	取值说明
迁移任务名	系统自动生成，无需修改。
源端 VPC IP/DNS	填写源实例的内网连接地址。具体请参见查看内网地址。
源端Port	填写源实例的端口，具体请参见查看端口。
用户名	源实例中创建的高权限数据库账号。
密码	源实例中创建的高权限数据库账号的密码。

g. 单击**创建迁移评估任务**。

> **说明**　迁移评估任务期间，实例状态将变更为 **维护实例中**。

迁移评估完成后，您可以在**上云评估**页面查看迁移评估任务的状态。

- 只有 状态 为**成功**，您才能进行迁移上云步骤。
- 如果 状态 为**失败**，请单击操作列的**查看报告**，根据报错进行处理，常见报错请参见解读上云评估报告。

任务名	迁移源类型	IP	Port	账号	创建时间	结束时间	状态	操作
eacetdkwiws	ECS自建	pgm-[illegible].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[illegible]	2021年12月[illegible]	成功	查看报告

ii. 迁移上云。

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，切换至**迁移上云**页签，单击创建迁移上云任务。

c. 在**创建迁移上云任务**窗口中，从**关联评估任务**列表中选择已成功的上云评估任务。

> **说明** 选择**关联评估任务**后，**源库类型**、**源端IP/DNS**、**源端Port**和**用户名**参数将会自动获取，无需配置。

d. 单击**发起迁移上云**，系统将自动启动上云任务。

> **注意** 上云任务期间，实例状态将变更为**迁入数据中**。您可以对源实例进行读写，但请勿执行迁移、重启、变配等操作。

创建成功后，结果示例如下：

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	/	2021年12月[redacted]	预检查	/	运行中	查看日志

iii. 将目标RDS PostgreSQL提升为主库。

a. 在迁移上云任务中，单击**上云阶段**列的链接，查看当前上云任务进度。

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	/	2021年12月[redacted]	增量同步	流式传输	运行中	查看日志 上云切换

b. 当上云阶段为**增量同步**时，可单击**操作**列的**上云切换**，将目标RDS PostgreSQL提升为主库，正式提供服务。

c. 在**切换上云**窗口，根据上云须知提示，设置源实例只读或应用停止写入。

> **说明** 使用高权限用户配置如下命令，设置源实例只读：
>
> ```
> ALTER SYSTEM SET default_transaction_read_only=on; SELECT pg_reload_conf();
> ```

d. 选中所有复选框，单击**立即切换**，等待迁移完成。
切换完成后，结果示例如下：

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	2021年12月[redacted]	2021年12月[redacted]	上云完成	/	成功	查看日志

9.5.3. RDS PostgreSQL实例缩容

RDS PostgreSQL的一键上云功能不仅支持阿里云ECS或IDC自建PostgreSQL的一键上云，还支持RDS PostgreSQL实例缩容存储空间，本文介绍 RDS PostgreSQL实例缩容的 具体操作。

背景信息

RDS PostgreSQL实例的变更配置，不支持缩小存储空间，而 一 键上云功能可以 实现实例间的迁移，您可以根据实际需要，将实例迁移至存储空间小于当前实例的新实例上，从而达到缩容的目的。

一键上云功能具有以下优势：

- 采用物理流复制方式，速度快、操作简单、稳定性好。
- 使用内网迁移，免费，安全。
- 限制条件少，目标实例为云盘主实例，大版本与源实例一致即可。
- 性能与源实例保持一致，统计信息等与源实例保持一致。

前提条件

已创建目标实例并 满足以下条 件：

- 目标实例与源实例大版本相同，当前支持PostgreSQL 14、13、12、11和10。
- 目标实例为主实例，只读实例不支持一键上云。
- 目标实例的存储类型为云盘。
- 目标实例为空，无数据。
- 目标实例的存储空间需要提前规划，推荐预留20%以上可用存储空间。
  例如，源实例已使用200GB，目标实例需要购买的存储空间至少为 `200÷(1-20%)=250 GB` 。

操作步骤

> **说明** 本文以华东1（杭州）同一VPC下的两个RDS PostgreSQL实例间迁移为例，源实例存储空间为500 GB，实际使用200 GB，迁移至存储空间为250 GB的目标实例。

1. 源实例配置。
   i. 添加白名单。

      添加白名单的具体操作，请参见 设置白名单， **组内白名单**请填写目标实例的VPC网段。
      目标实例的VPC网段查看方法：

      a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
      b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。
      c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL**后，单击**下一步**。
      d. 在**目标库配置**步骤，查看目标实例的**当前VPC网段**。

   ii. 创建高权限账号。

      创建账号的具体操作，请参见 创建账号。 **账号类型**请选择高权限账号。

      > **说明** 高权限账号用于迁移上云，需要具备CREATE ROLE、REPLICATION和pg_monitor权限。如果您已有高权限的账号，则无需重复创建。

2. 目标实例配置。

i. 上云评估。

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，单击**上云评估**页签。

c. 在配置向导的**选择迁移来源**步骤，选择迁移来源为**阿里云ECS自建PostgreSQL或阿里云RDS PostgreSQL**后，单击**下一步**。

d. 在**目标库配置**步骤，单击**下一步**。

e. 在**源库配置**步骤，选中所有配置准备项，单击**下一步**。

f. 在**发起迁移评估**步骤，配置源库信息。

参数	取值说明
迁移任务名	系统自动生成，无需修改。
源端 VPC IP/DNS	填写源实例的内网连接地址。具体请参见查看内网地址。
源端Port	填写源实例的端口，具体请参见查看端口。
用户名	源实例中创建的高权限数据库账号。
密码	源实例中创建的高权限数据库账号的密码。

g. 单击**创建迁移评估任务**。

> **说明** 迁移评估任务期间，实例状态将变更为 **维护实例中**。

迁移评估完成后，您可以在**上云评估**页面查看迁移评估任务的状态。

- 只有 状态 为**成功**，您才能进行迁移上云步骤。
- 如果 状态 为**失败**，请单击操作列的**查看报告**，根据报错进行处理，常见报错请参见解读上云评估报告。

ii. 迁移上云。

a. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

b. 在左侧导航栏单击**一键上云**，切换至**迁移上云**页签，单击创建迁移上云任务。

c. 在**创建迁移上云任务**窗口中，从**关联评估任务**列表中选择已成功的上云评估任务。

> **说明** 选择**关联评估任务**后，**源库类型**、**源端 IP/DNS**、**源端Port**和**用户名**参数将会自动获取，无需配置。

d. 单击**发起迁移上云**，系统将自动启动上云任务。

> **注意** 上云任务期间，实例状态将变更为**迁入数据中**。您可以对源实例进行读写，但请勿执行迁移、重启、变配等操作。

创建成功后，结果示例如下：

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	/	2021年12月[redacted]	预检查	/	运行中	查看日志

iii. 将目标RDS PostgreSQL提升为主库。

a. 在迁移上云任务中，单击**上云阶段**列的链接，查看当前上云任务进度。

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	/	2021年12月[redacted]	增量同步	流式传输	运行中	查看日志 上云切换

b. 当上云阶段为**增量同步**时，可单击**操作**列的**上云切换**，将目标RDS PostgreSQL提升为主库，正式提供服务。

c. 在**切换上云**窗口，根据上云须知提示，设置源实例只读或应用停止写入。

> **说明** 使用高权限用户配置如下命令，设置源实例只读：
>
> ```
> ALTER SYSTEM SET default_transaction_read_only=on; SELECT pg_reload_conf();
> ```

d. 选中所有复选框，单击**立即切换**，等待迁移完成。
切换完成后，结果示例如下：

任务名	迁移源类型	IP	Port	账号	创建时间	切换上云时间	最后修改时间	上云阶段	数据同步进度	任务状态	操作
1kiro28p6mw	ECS自建	pgm-[redacted].pg.rds.aliyuncs.com	5432	migratetest	2021年12月[redacted]	2021年12月[redacted]	2021年12月[redacted]	上云完成	/	成功	查看日志

后续配置

为了确保RDS PostgreSQL实例缩容后，应用侧在不修改业务代码的情况下依然正常 连接 ，还需要在目标实例做如下配置。

1. 修改目标实例白名单设置，与源实例白名单保持一致。更多信息，请参见设置白名单。
2. 修改目标实例的连接地址，与源实例连接地址相同。修改方法请参见修改内外网连接地址。

   以源实例连接地址为 `pgm-aaa.pg.rds.aliyuncs.com` ，目标实例连接地址为 `pgm-bbb.pg.rds.aliyuncs.com` 为例，修改方法如下。

   i. 将源实例连接地址 `pgm-aaa.pg.rds.aliyuncs.com` 修改为 `pgm-ccc.pg.rds.aliyuncs.com` 。

   ii. 将目标实例连接地址 `pgm-bbb.pg.rds.aliyuncs.com` 修改为 `pgm-aaa.pg.rds.aliyuncs.com` 。

10.数据迁移

10.1. 数据迁移方案概览

RDS提供了多种数据迁移方案，可满足不同上云或迁云的业务需求，使您可以在不影响业务的情况下将数据在其他数据库与阿里云云数据库RDS之间平滑迁移。

使用场景	相关操作
将自建数据库迁移到云数据库	• 使用DTS将自建PostgreSQL迁移到RDS PostgreSQL • 使用pg_dump和pg_restore将自建PostgreSQL迁移到RDS PostgreSQL
第三方数据库迁移到云数据库	• 从Amazon RDS for PostgreSQL增量迁移至阿里云 • 从Amazon RDS for PostgreSQL全量迁移至阿里云
RDS实例间的数据库迁移	RDS PostgreSQL间的数据迁移
将RDS数据迁移到本地数据库	迁移RDS PostgreSQL数据到本地PostgreSQL

10.2. 将自建数据库迁移到云数据库

10.2.1. 使用pg_dump和pg_restore将自建PostgreSQL数据库迁移到RDS PostgreSQL

本文介绍如何通过ECS实例将PostgreSQL数据迁移到目标RDS PostgreSQL实例中。

前提条件

准备一台可以连接到RDS和自建库的ECS（建议使用Linux系统）。

操作步骤

1. 在ECS实例上安装PostgreSQL自带工具。

> **说明** pg_dump版本必须与自建库一致，pg_restore版本建议与RDS PostgreSQL一致。本文示例为PostgreSQL 12，您也可根据业务需要自行选择其他版本。

```
--安装RPM #yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm --安装PostgreSQL #yum install -y postgresql12-*
```

2. 配置环境变量，保证导出导入时使用的PostgreSQL版本正确。

```
#su - postgres #vi .bash_profile --追加内容 export PS1="$USER@`/bin/hostname -s`-> " export LANG=en_US.utf8 export PGHOME=/usr/pgsql-12 export LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib:$LD_LIBRARY_PATH export DATE=`date +"%Y%m%d%H%M"` export PATH=$PGHOME/bin:$PATH:. export MANPATH=$PGHOME/share/man:$MANPATH alias rm='rm -i' alias ll='ls -lh' unalias vi
```

3. 导出用户。

> **说明** 一定要先迁移用户，否则可能因为对象的权限或Owner问题导致迁移失败。

```
#pg_dumpall -g -h 127.0.0.1 -p 5432 -U postgres -- -- PostgreSQL database cluster dump
-- SET default_transaction_read_only = off; SET client_encoding = 'UTF8'; SET standard_
conforming_strings = on; -- -- Roles -- CREATE ROLE postgres; ALTER ROLE postgres WITH
SUPERUSER INHERIT CREATEROLE CREATEDB LOGIN REPLICATION BYPASSRLS PASSWORD 'md5d5df0dxx
xxxxxc88a541fec598f'; -- -- PostgreSQL database cluster dump complete --
```

4. 将上一步返回的Roles部分命令略做修改，在RDS PostgreSQL实例中执行命令导入用户，其中SUPERUSER需要修改为rds_superuser。

```
#CREATE ROLE postgres; #ALTER ROLE postgres WITH rds_SUPERUSER INHERIT CREATEROLE CREAT
EDB LOGIN REPLICATION BYPASSRLS PASSWORD 'md5d5df0dxxxxxxxxc88a541fec598f';
```

```
CREATE ROLE postgres;
ALTER ROLE postgres WITH rds_SUPERUSER INHERIT CREATEROLE CREATEDB LOGIN REPLICATION BYPASSRLS PASSWORD 'md5d5df0d43cc619177e8c88a541fec598f';
```

5. 连接到RDS PostgreSQL实例创建目标数据库。

> **说明** encoding需要和自建库相同，本文示例为UTF8。

```
#create database db1 with template template0 encoding 'UTF8' lc_ctype 'en_US.utf8' lc_c
ollate 'C';
```

6. 数据导出及导入。您可以使用如下三种方法：
   - 在线迁移
     当自建库和RDS PstgreSQL实例可以直接连接时，可以使用此方案。
     a. 配置密码文件。格式为：`host:port:dbname:username:password`。

     > **说明** dbname和username必须小写，因为PostgreSQL元数据默认使用小写字母存储， 除非您的数据库和用户在创建时使用双引号包括并且大写。

     ```
     #vi ~/.pgpass pgm-xxx.pg.rds.aliyuncs.com:1921:db1:用户:密码 127.0.0.1:5432:postgr
     es:用户:密码 #chmod 400 ~/.pgpass
     ```

     b. 用管道迁移数据。

     ```
     #nohup pg_dump -F p -h 127.0.0.1 -p 5432 -U postgres -d postgres --no-tablespaces
     | time psql -h pgm-bpxxxxx.pg.rds.aliyuncs.com -p 1921 -U postgres --single-trans
     action db1 > ./pg.dump.log 2>&1 &
     ```

     > **说明** 您可以查看pg.dump.log文件检查错误日志，因为设置了`--single-transaction`选项，所以解决报错后可以直接重新导入。

   - 离线导出并导入
     如果自建库和RDS PstgreSQL实例无法直接连接时，可以先使用pg_dump导出文件，然后将文件拷贝到能连接RDS的主机，最后使用pg_restore导入。

> **说明** 导出时pg_dump版本需要与自建库一致，导入时pg_restore版本需要与RDS一致。

a. 导出文件。

```
#nohup pg_dump -F c -h 127.0.0.1 -p 5432 -U postgres -d postgres --no-tablespaces
-f ./pg.dump > ./pg.dump.log 2>&1 & #ll pg.dump -rw-rw-r-- 1 digoal digoal 4.2M A
ug 31 10:17 pg.dump
```

> **说明** 等待导出完成，检查pg.dump.log文件，确认没有报错。

b. 导入文件。

```
#pg_restore -h pgm-bpxxxxx.pg.rds.aliyuncs.com -p 1921 -U postgres -d db1 --no-ta
blespaces --single-transaction pg1.dump >./pg1.restore.log
```

> **说明** 如果前面步骤设置的MD5密码，可能会报错，建议您在控制台重置postgres用户的密码。重置密码请参见 重置密码。

耐心等待导入结束，您可以查看pg.dump.log文件检查错误日志，因为设置了 `--single-transaction` 选项，所以解决报错后可以直接重新导入。

- 离线导出并导入（并行迁移实现加速）
  与上一方法类似，只是在使用pg_restore导入时，采取并行的选项。

```
nohup pg_restore -U postgres -d db1 --no-tablespaces -j 4 /tmp/pg.dump >./pg.restore.
log 2>&1 &
```

> **说明** 并行参数 `-j` 和 `--single-transaction` 参数不能同时使用。

10.2.2. 自建PostgreSQL迁移至RDS PostgreSQL

本文介绍如何使用数据传输服务DTS（Data Transmission Service），将自建PostgreSQL迁移至RDS PostgreSQL。DTS支持结构迁移、全量数据迁移和增量数据迁移，同时使用这三种迁移类型可以实现在自建应用不停服的情况下，平滑地完成自建PostgreSQL数据库迁移上云。

前提条件

- 已创建源数据库自建PostgreSQL和目标实例RDS PostgreSQL。目标实例RDS PostgreSQL的创建方式和支持的版本，请参见创建RDS PostgreSQL实例和迁移方案概览。

> **说明** 建议源和目标数据库的版本相同，或者从低版本迁移到高版本以保障兼容性。
> 如为高版本迁移至低版本，可能存在数据库兼容性问题。

- 目标实例RDS PostgreSQL的存储空间须大于源数据库自建PostgreSQL占用的存储空间。

注意事项

类型	说明

类型	说明

源库限制	• 带宽要求：源库所属的服务器需具备足够出口带宽，否则将影响数据迁移速率。
	• 待迁移的表需具备主键或唯一约束，且字段具有唯一性，否则可能会导致目标数据库中出现重复数据。
	待迁移的数据库名称中间不能包含短划线（-），例如dts-testdata。
	• 如迁移对象为表级别，且需进行编辑（如表列名映射），则单次迁移任务仅支持迁移至多1000张表。当超出数量限制，任务提交后会显示请求报错，此时建议您拆分待迁移的表，分批配置多个任务，或者配置整库的迁移任务。
	• 如需进行增量迁移，WAL日志：
	  ○ 需开启，即设置 wal_level 参数的值为*logical*。
	  ○ 如为增量迁移任务，DTS要求源数据库的WAL日志保存24小时以上，如为全量迁移和增量迁移任务，DTS要求源数据库的WAL日志至少保留7天以上（您可在全量迁移完成后将WAL日志保存时间设置为24小时以上），否则DTS可能因无法获取WAL日志而导致任务失败，极端情况下甚至可能会导致数据不一致或丢失。由于您所设置的WAL日志保存时间低于DTS要求的时间进而导致的问题，不在DTS的SLA保障范围内。
	• 源库的操作限制：
	  ○ 如自建PostgreSQL进行主备切换，会导致迁移失败。
	  ○ 在库表结构迁移和全量迁移阶段，请勿执行库或表结构变更的DDL操作，否则数据迁移任务会失败。

其他限制	• 由于源库的主备节点可能存在延迟导致数据不一致，执行数据迁移时请使用源库的主节点作为迁移的数据源。
	• 单个数据迁移任务只能迁移一个数据库，如需迁移多个数据库，您需要为每个数据库配置数据迁移任务。
	• 增量数据迁移期间，如果迁移对象的选择粒度为Schema，在待迁移的Schema中创建了新的表或使用RENAME命令重建了待迁移的表，您需要在对该表写入数据前执行 `ALTER TABLE schema.table REPLICA IDENTITY FULL;` 命令。
	  说明 将上述命令中的 `schema` 和 `table` 替换成真实的Schema名和表名。
	• DTS的校验对象为数据内容，暂不支持Sequence等元数据的校验，您需要自行校验。
	• 由于业务切换到目标端后，新写入的Sequence不会按照源库的Sequence最大值作为初始值去递增，您需要在业务切换前，在源库中查询对应Sequence的最大值，然后在目标库中将其作为对应Sequence的初始值。查询源库Sequence值的相关命令如下：
	  `do language plpgsql $$ declare nsp name; rel name; val int8; begin for nsp,rel in select nspname,relname from pg_class t2 , pg_namespace t3 where t2.relnamespace=t3.oid and t2.relkind='S' loop execute format($_$select last_value from %I.%I$_$, nsp, rel) into val; raise notice '%', format($_$select setval('%I.%I'::regclass, %s);$_$, nsp, rel, val+1); end loop; end; $$;`
	• DTS会在源库中创建以下临时表，用于获取增量数据的DDL、增量表的结构、心跳信息等。在迁移期间，请勿删除源库中的临时表，否则会导致DTS任务异常。临时表会在DTS实例释放后自动删除。

类型	说明
	`public.DTS_PG_CLASS` 、 `public.DTS_PG_ATTRIBUTE` 、 `public.DTS_PG_TYPE` 、 `public.DTS_PG_ENUM` 、 `public.DTS_POSTGRES_HEARTBEAT` 、 `public.DTS_DDL_COMMAND` 、 `public.DTS_ARGS_SESSION` 。 • 为保障增量数据迁移延迟时间展示的准确性，DTS会在源库中新增一个名为 `dts_postgres_heartbeat` 的心跳表。 • 源为自建PG的复用源增量数据迁移期间，DTS会在源库中创建前缀为 `dts_sync_` 的replication slot用于复制数据。DTS会每隔90分钟自动清理一次历史replication slot，避免其持续累积占用磁盘空间。 说明 当释放迁移任务或迁移失败时，DTS会主动清理该replication slot；如果RDS PostgreSQL发生了主备切换，则需要您登录备库来手动清理。 • 执行数据迁移前需评估源库和目标库的性能，同时建议业务低峰期执行数据迁移。否则全量数据迁移时DTS占用源和目标库一定读写资源，可能会导致数据库的负载上升。 • 由于全量数据迁移会并发执行INSERT操作，导致目标数据库的表产生碎片，因此全量迁移完成后目标库的表存储空间会比源库的表存储空间大。 • 请确认DTS对数据类型为FLOAT或DOUBLE的列的迁移精度是否符合业务预期。DTS会通过 `ROUND(COLUMN,PRECISION)` 来读取这两类列的值。如果没有明确定义其精度，DTS对FLOAT的迁移精度为38位，对DOUBLE的迁移精度为308位。 • DTS会尝试恢复七天之内迁移失败任务。因此业务切换至目标实例前，请务必结束或释放该任务，或者将DTS访问目标实例账号的写权限用 `revoke` 命令回收掉。避免该任务被自动恢复后，源端数据覆盖目标实例的数据。

迁移类型说明

- 库表结构迁移
  DTS将源库中迁移对象的结构定义迁移到目标库。
- 全量迁移
  DTS在全量迁移的基础上，将源库的增量更新数据迁移到目标库中。通过增量数据迁移可以实现在自建应用不停服的情况下，平滑地完成数据迁移。
- 增量迁移
  DTS在全量迁移的基础上，将源库的增量更新数据迁移到目标库中。通过增量数据迁移可以实现在自建应用不停服的情况下，平滑地完成数据迁移。

支持增量迁移的SQL操作

操作类型	SQL操作语句
DML	INSERT、UPDATE、DELETE

数据库账号的权限要求

数据库	结构迁移	全量迁移	增量迁移
自建PostgreSQL数据库	pg_catalog的usage权限	迁移对象的select权限	superuser

数据库	结构迁移	全量迁移	增量迁移
RDS PostgreSQL实例	迁移对象的create、usage权限	schema的owner权限	schema的owner权限

数据库账号创建及授权方法：

- 自建PostgreSQL数据库请参见CREATE USER和GRANT语法。
- RDS PostgreSQL实例请参见创建账号。

准备工作

如自建PostgreSQL版本为10.1~13，则您需执行如下准备工作。

1. 登录自建PostgreSQL所属的服务器。
2. 修改配置文件*postgresql.conf*，将配置文件中的 `wal_level` 设置为 `logical` 。

```
# - Settings -

wal_level = logical                     # minimal, replica, or logical
                                        # (change requires restart)
```

3. 将DTS的IP地址加入至自建PostgreSQL的配置文件*pg_hba.conf*中。您只需添加目标数据库所在区域对应的DTS IP地址段，详情请参见迁移、同步或订阅本地数据库时需添加的IP白名单。

> **说明** 关于该配置文件的设置请参见 pg_hba.conf文件。如果您已将信任地址配置为 `0.0.0.0/0` （如下图所示），可跳过本步骤。

```
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             0.0.0.0/0               trust
# IPv6 local connections:
host    all             all             ::1/128                 md5
```

4. 根据待迁移对象所属的数据库和schema信息，在目标RDS PostgreSQL中创建相应数据库和schema（schema名称须一致），详情请参见创建数据库和Schema 管理。

如自建PostgreSQL版本为 9.4.8~10.0 ，则您需执行如下准备工作。

1. 从官网下载PostgreSQL源码并完成编译安装。
   i. 根据源库自建PostgreSQL的版本，从PostgreSQL官网下载对应版本的源码。
   ii. 依次执行命令 `./configure` 、 `make` 、 `make install` ，配置、编译并安装源码。

> **注意**
> - 编译安装PostgreSQL时，PostgreSQL的操作系统版本需与GCC（GNU编译器套件）版本保持一致。
> - 如执行 `./configure` 遇到报错，您可以根据报错提示调整命令。例如，报错信息为 `readline library not found. Use --without-readline to disable readline support.` ，则您可以调整命令为 `./configure --without-readline` .
> - 如您选择其他方式安装PostgreSQL，需要在系统版本以及GCC版本相同的测试环境下，编译ali_decoding。

2. 下载DTS提供的插件ali_decoding并完成编译安装。
   i. 下载ali_decoding。
   ii. 将ali_decoding整个目录拷贝至PostgreSQL（已完成编译安装）的contrib目录中。

```
-rw-r--r--  1 1107 1107    384 9月  27 2016 aclocal.m4
drwxrwxrwx  2 1107 1107   4096 9月  27 2016 config
-rw-r--r--  1 root root 374806 9月   7 10:10 config.log
-rwxr-xr-x  1 root root  39032 9月   7 10:10 config.status
-rwxr-xr-x  1 1107 1107 471157 9月  27 2016 configure
-rw-r--r--  1 1107 1107  75195 9月  27 2016 configure.in
drwxrwxrwx 56 1107 1107   4096 9月   7 10:28 contrib
-rw-r--r--  1 1107 1107   1192 9月  27 2016 COPYRIGHT
drwxrwxrwx  3 1107 1107   4096 9月  27 2016 doc
-rw-r--r--  1 root root   3638 9月   7 10:10 GNUmakefile
-rw-r--r--  1 1107 1107   3638 9月  27 2016 GNUmakefile.in
-rw-r--r--  1 1107 1107    283 9月  27 2016 HISTORY
-rw-r--r--  1 1107 1107  75065 9月  27 2016 INSTALL
-rw-r--r--  1 1107 1107   1489 9月  27 2016 Makefile
-rw-r--r--  1 1107 1107   1209 9月  27 2016 README
drwxrwxrwx 16 1107 1107   4096 9月   7 10:10 src
```

   iii. 进入ali_decoding目录，将Makefile文件的内容替换为如下：

```
# contrib/ali_decoding/Makefile MODULE_big = ali_decoding MODULES = ali_decoding OB
JS = ali_decoding.o DATA = ali_decoding--0.0.1.sql ali_decoding--unpackaged--0.0.1.
sql EXTENSION = ali_decoding NAME = ali_decoding #subdir = contrib/ali_decoding #to
p_builddir = ../.. #include $(top_builddir)/src/Makefile.global #include $(top_srcd
ir)/contrib/contrib-global.mk #PG_CONFIG = /usr/pgsql-9.6/bin/pg_config #pgsql_lib_
dir := $(shell $(PG_CONFIG) --libdir) #PGXS := $(shell $(PG_CONFIG) --pgxs) #includ
e $(PGXS) # 源码安装用以下 ifdef USE_PGXS PG_CONFIG = pg_config PGXS := $(shell $(PG_
CONFIG) --pgxs) include $(PGXS) else subdir = contrib/ali_decoding top_builddir = .
./.. include $(top_builddir)/src/Makefile.global include $(top_srcdir)/contrib/cont
rib-global.mk endif
```

   iv. 进入ali_de codin g目录，依次执行命令 `make` 、 `make install` ，编译ali_decoding，并得到安装ali_decoding所需的文件。
   v. 将如下文件复制至指定位置。

```
/usr/bin/install -c -m 755  ali_decoding.so '/usr/local/pgsql/lib/ali_decoding.so'
/usr/bin/install -c -m 644 ./ali_decoding.control '/usr/local/pgsql/share/extension/'
/usr/bin/install -c -m 644 ./ali_decoding--0.0.1.sql ./ali_decoding--unpackaged--0.0.1.sql  '/usr/local/pgsql/share/extension/'
/usr/bin/install -c -m 755  ali_decoding.so '/usr/local/pgsql/lib/'
```

3. 根据待迁移对象所属的数据库和schema信息，在目标RDS PostgreSQL中创建相应数据库和schema（schema名称须一致），详情请参见创建数据库和Schema 管理。

操作步骤

1. 登录新版DTS迁移任务的列表页面。

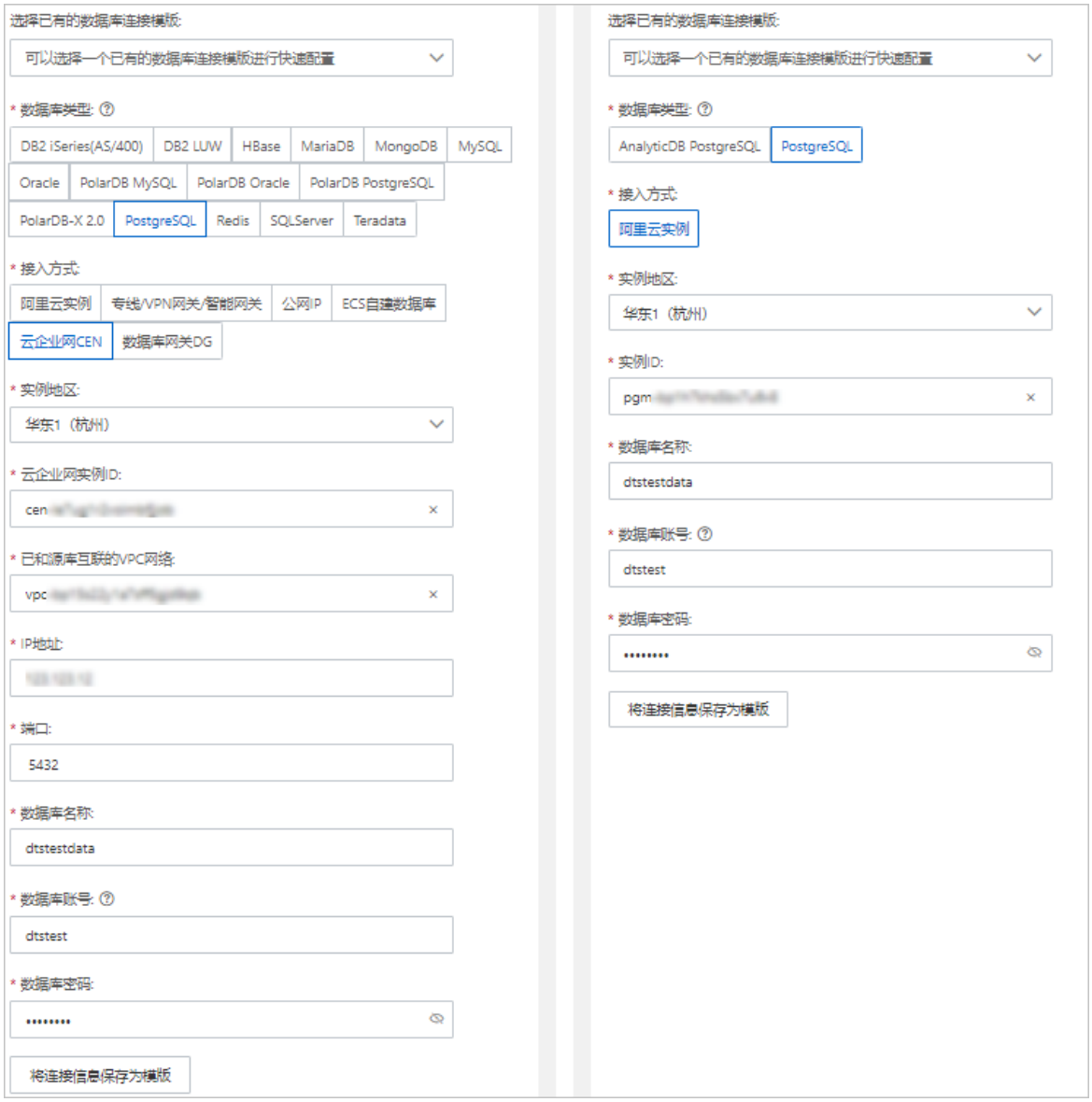
> **说明** 您也可以登录 DMS数据管理服务。在顶部菜单栏中，选择 **集成与开发（DTS）** > **数据迁移**。

2. 在页面左上角，选择迁移实例所属地域。

   换图

3. 单击**创建任务**，配置源库及目标库信息。

> **警告** 选择源和目标实例后，建议您仔细阅读页面上方显示的**使用限制**，以成功创建并执行迁移任务。



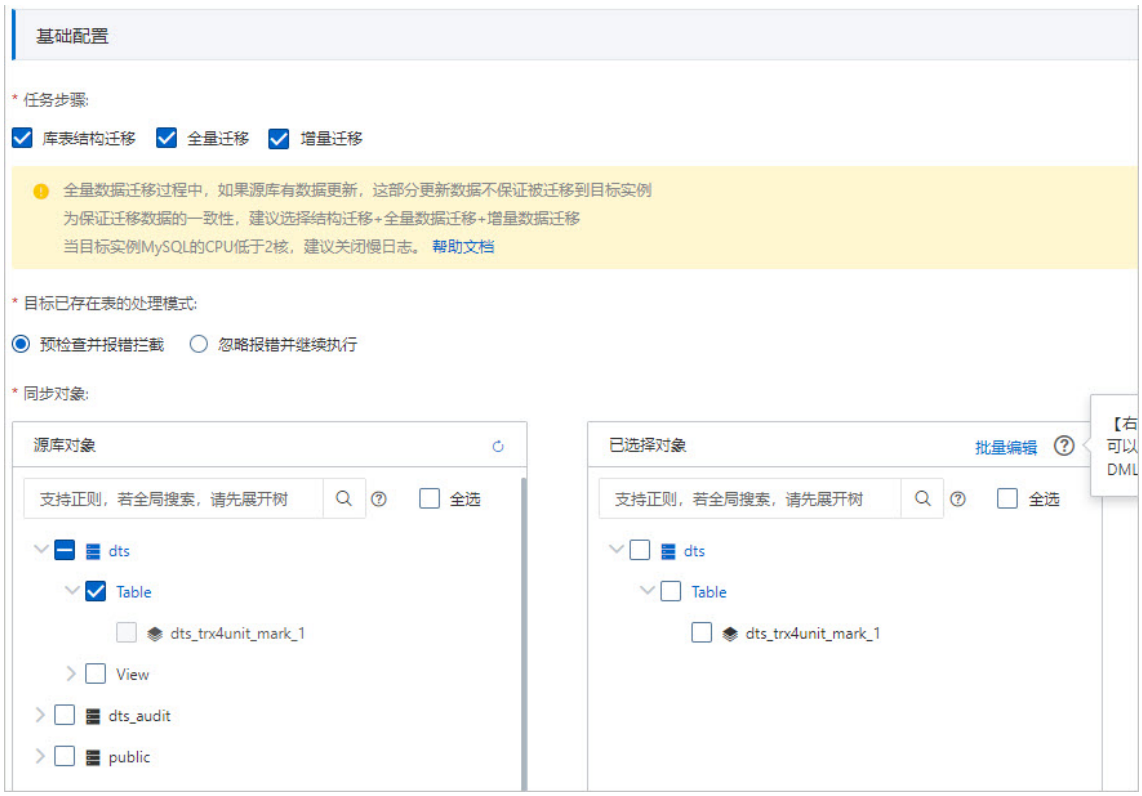
类别	配置	说明
无	**任务名称**	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。

类别	配置	说明
源库信息		
	数据库类型	选择**PostgreSQL**。
	接入方式	根据源库的部署位置进行选择，本文以**云企业网CEN**为例介绍配置流程。 **说明** 当源实例为自建数据库时，您还需要执行相应的准备工作，详情请参见 准备工作概览。
	实例地区	选择自建PostgreSQL数据库所属地域。
	云企业网实例ID	选择自建PostgreSQL数据库所属云企业网的实例ID。
	已和源库互联的VPC网络	选择与自建PostgreSQL数据库互联的VPC网络。
	IP地址	填入自建PostgreSQL数据库的服务器IP地址。
	端口	填入自建SQL Server数据库的服务端口，默认为**5432**。
	数据库名称	填入自建Postg reSQL中 迁移对象所属数据库的名称。
	数据库账号	填入自建PostgreSQL的数据库账号，权限要求，请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号对应的密码。
目标库信息		
	数据库类型	选择**PostgreSQL**。
	接入方式	选择**阿里云实例**。
	实例地区	选择目标RDS PostgreSQL实例所属地域。
	实例ID	选择目标RDS PostgreSQL实例ID。
	数据库名称	填入目标RDS PostgreSQL实例中迁移对象所属数据库的名称。
	数据库账号	填入目标RDS PostgreSQL实例的数据库账号，权限要求，请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号对应的密码。

4. 配置完成后，单击页面右下角的**测试连接以进行下一步**。

> **警告**
>
> - 如果源或目标数据库是阿里云数据库实例（例如RDS MySQL、云数据库MongoDB版等）或ECS上的自建数据库，DTS会自动将对应地区DTS服务的IP地址添加到阿里云数据库实例的白名单或ECS的安全规则中，您无需手动添加，请参见DTS服务器的IP地址段；如果源或目标数据库是IDC自建数据库或其他云数据库，则需要您手动添加对应地区DTS服务的IP地址，以允许来自DTS服务器的访问。
> - 上述场景中，DTS自动添加或您手动添加DTS服务的公网IP地址段可能会存在安全风险，一旦使用本产品代表您已理解和确认其中可能存在的安全风险，并且需要您做好基本的安全防护，包括但不限于加强账号密码强度防范、限制各网段开放的端口号、内部各API使用鉴权方式通信、定期检查并限制不需要的网段，或者使用通过内网（专线/VPN网关/智能网关）的方式接入。
> - DTS任务完成或释放后，建议您手动检测并删除DTS相关的服务器IP地址段。

5. 配置任务对象及高级配置。
   - **基础配置**



配置	说明
任务步骤	全支持 ■ 如果只需要进行全量迁移，请同时选中**库表结构迁移**和**全量迁移**。 ■ 如果需要进行不停机迁移，请同时选中**库表结构迁移**、**全量迁移**和**增量迁移**。 **说明** 如果未选择 **增量迁移**，为保障数据一致性，数据迁移期间请勿在源实例中写入新的数据。

配置	说明
目标已存在表的处理模式	■ **预检查并报错拦截**：检查目标数据库中是否有同名的表。如果目标数据库中没有同名的表，则通过该检查项目；如果目标数据库中有同名的表，则在预检查阶段提示错误，数据迁移任务不会被启动。 **说明** 如果目标库中同名的表不方便删除或重命名，您可以更改该表在目标库中的名称，请参见 库表列名映射。 ■ **忽略报错并继续执行**：跳过目标数据库中是否有同名表的检查项。 **警告** 选择为 **忽略报错并继续执行**，可能导致数据不一致，给业务带来风险，例如： ■ 表结构一致的情况下，在目标库遇到与源库主键的值相同的记录，则会保留目标库中的该条记录，即源库中的该条记录不会迁移至目标库中。 ■ 表结构不一致的情况下，可能导致只能迁移部分列的数据或迁移失败。
同步对象	在**同步对象**框中单击待迁移的对象，然后单击 › 将其移动到**已选择对象**框。 **说明** 迁移对象选择的粒度为schema、表、列。若选择的迁移对象为表或列，其他对象（如视图、触发器、存储过程）不会被迁移至目标库。
映射名称更改	■ 如需更改单个迁移对象在目标实例中的名称，请右击**已选择对象**中的迁移对象，设置方式，请参见库表列名单个映射。 ■ 如需批量更改迁移对象在目标实例中的名称，请单击**已选择对象**方框右上方的 批量编辑 设置方式，请参见库表列名批量映射。 **说明** 如果使用了对象名映射功能，可能会导致依赖这个对象的其他对象迁移失败。
过滤待迁移数据	支持设置条件过滤数据，详情请参见通过SQL条件过滤任务数据。
增量迁移的SQL操作	选择增量迁移SQL操作，请右击**已选择对象**中的迁移对象，在弹跳框中选择所需增量迁移的SQL操作。支持的操作，请参见支持增量迁移的SQL操作。

- **高级配置**

配置	说明
设置告警	是否设置告警，当迁移失败或延迟超过阈值后，将通知告警联系人。 ■ **不设置**：不设置告警。 ■ **设置**：设置告警，您还需要设置告警阈值和告警联系人。
目标库对象名称大小写策略	您可以配置目标实例中迁移对象的库名、表名和列名的英文大小写策略。默认情况下选择**DTS默认策略**，您也可以选择与源库、目标库默认策略保持一致。更多信息，请参见目标库对象名称大小写策略。
源、目标库无法连接重试时间	默认重试720分钟，您也可以在取值范围（10~720分钟）内自定义重试时间，建议设置30分钟以上。如果DTS在设置的时间内重新连接上源、目标库，同步任务将自动恢复。否则，同步任务将失败。 **说明** ■ 针对同源或者同目标的多个DTS实例，如DTS实例A和DTS实例B，设置网络重试时间时A设置30分钟，B设置60分钟，则重后创建任务为准。 ■ 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

6. 上述配置完成后，单击页面右下角的**下一步保存任务并预检查**。

> **说明**
> - 在迁移任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动迁移任务。
> - 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
> - 您可以根据提示修复后重新进行预检查。
> - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

7. **预检查通过率**显示为**100%**时，单击**下一步购买**。
8. 在**购买**页面，选择数据迁移实例的链路规格，详细说明请参见下表。

类别	参数	说明
信息配置	**链路规格**	DTS为您提供了不同性能的迁移规格，迁移链路规格的不同会影响迁移速率，您可以根据业务场景进行选择，详情请参见数据迁移链路规格说明。

9. 配置完成后，阅读并选中**《数据传输（按量付费）服务条款》**。
10. 单击**购买并启动**，迁移任务正式开始，您可在数据迁移界面查看具体进度。

10.3. 第三方数据库迁移到云数据库

10.3.1. 从Amazon RDS for PostgreSQL增量迁移至阿里云

本文介绍如何使用数据传输服务DTS（Data Transmission Service），将Amazon RDS for PostgreSQL迁移至阿里云RDS PostgreSQL。DTS支持结构迁移、全量数据迁移以及增量数据迁移，同时使用这三种迁移类型可以实现在自建应用不停服的情况下，平滑地完成数据库迁移。

前提条件

- Amazon RDS for PostgreSQL的数据库版本为10.4~12的版本。
- 为保障DTS能够通过公网连接Amazon RDS for PostgreSQL，需要将其**公开访问**设置为**是**。
- 为保障DTS能够读取Amazon RDS for PostgreSQL的增量数据变更信息，需要将其参数组中rds.logical_replication 参数的值修改为 *1*。
- 已创建阿里云RDS PostgreSQL，详情请参见创建RDS PostgreSQL。

> **说明**
> - 阿里云RDS PostgreSQL的数据库版本为10或11版本，如需跨版本迁移，请创建一个按量付费的实例以验证兼容性。
> - 阿里云RDS PostgreSQL的存储空间，须大于Amazon RDS for PostgreSQL中待迁移数据库占用的存储空间。

注意事项

- DTS在执行全量数据迁移时将占用源库和目标库一定的读写资源，可能会导致数据库的负载上升，在数据库性能较差、规格较低或业务量较大的情况下（例如源库有大量慢SQL、存在无主键表或目标库存在死锁等），可能会加重数据库压力，甚至导致数据库服务不可用。因此您需要在执行数据迁移前评估源库和目标库的性能，同时建议您在业务低峰期执行数据迁移（例如源库和目标库的CPU负载在30%以下）。
- 如果待迁移对象没有主键或唯一约束，且所有字段没有唯一性，可能会导致目标数据库中出现重复数据或迁移失败。
- 一个数据迁移任务只能迁移一个数据库，如果有多个数据库需要迁移，则需要为每个数据库创建数据迁移任务。
- 仅支持DML操作（INSERT、DELETE、UPDATE）的同步。
- 在数据迁移的过程中，DTS会在Amazon RDS for PostgreSQL中创建前缀为 `dts_sync_` 的replication slot用于复制数据。DTS会每隔90分钟自动清理一次历史replication slot，避免其持续累积占用磁盘空间。

  > **说明** 当释放迁移任务或迁移失败时，DTS会主动清理该replication slot；如果Amazon RDS for PostgreSQL发生了主备切换，则需要您登录备库来手动清理。

- 对于迁移失败的任务，DTS会触发自动恢复。在您将业务切换至目标实例前，请务必先结束或释放该任务，避免该任务被自动恢复后，导致源端数据覆盖目标实例的数据。

费用说明

迁移类型	链路配置费用	公网流量费用
结构迁移和全量数据迁移	不收费。	通过公网将数据迁移出阿里云时将收费，详情请参见产品定价。
增量数据迁移	收费，详情请参见产品定价。	

迁移类型说明

- 结构迁移
  DTS将迁移对象的结构定义迁移至阿里云RDS PostgreSQL，支持结构迁移的对象包含TABLE、TRIGGER、VIEW、SEQUENCE、FUNCTION、USER DEFINED TYPE、RULE、DOMAIN、OPERATION、AGGREGATE。

  > **说明** 不支持迁移使用C语言编写的FUNCTION。

- 全量数据迁移
  将迁移对象的存量数据全部迁移到阿里云RDS PostgreSQL数据库中。
- 增量数据迁移
  DTS在全量数据迁移的基础上，将迁移对象的增量更新迁移到阿里云RDS PostgreSQL数据库中。通过增量数据迁移可以实现在应用不停服的情况下，平滑地完成数据迁移。

数据库账号的权限要求

数据库	结构迁移	全量迁移	增量迁移
Amazon RDS for PostgreSQL	pg_catalog的usage权限	迁移对象的select权限	rds_superuser权限
阿里云RDS PostgreSQL	迁移对象的create、usage权限	schema的owner权限	schema的owner权限

数据迁移流程

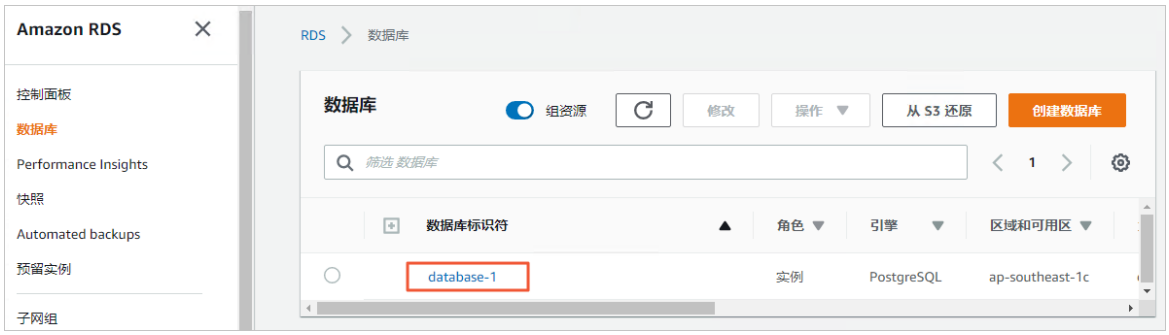
为解决对象间的依赖，提高迁移成功率，DTS对PostgreSQL结构及数据的迁移顺序如下。

1. 执行TABLE、VIEW、SEQUENCE、FUNCTION、USER DEFINED TYPE、RULE、DOMAIN、OPERATION、AGGREGATE的结构迁移。
2. 执行全量数据迁移。
3. 执行TRIGGER、FOREIGN KEY的结构迁移。
4. 执行增量数据迁移。

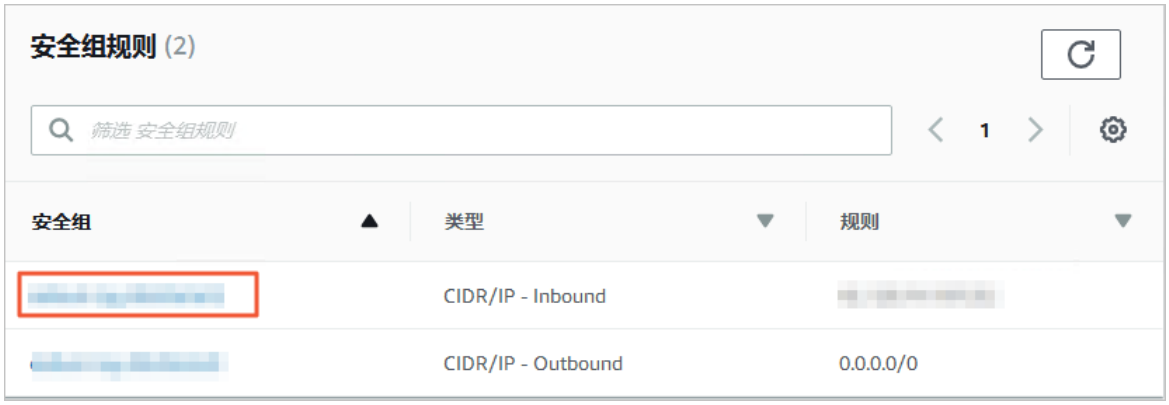
> **说明** 在执行增量数据迁移前，请勿对Amazon RDS for PostgreSQL中的迁移对象执行DDL操作，否则可能导致迁移失败。

准备工作

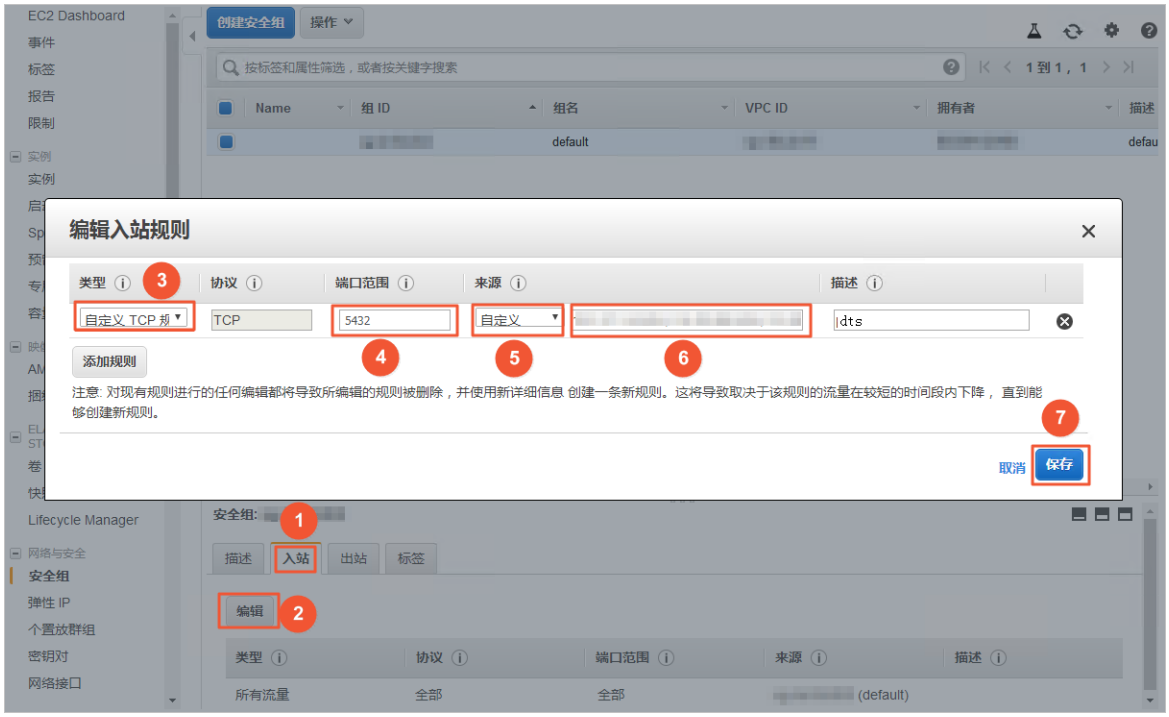
1. 登录Amazon RDS控制台。
2. 在页面右上角，选择目标实例所属的地域。
3. 单击左侧导航栏的**数据库**，然后单击目标数据库标识符，进入**基本信息**页面。



4. 在**安全组规则**区域框，单击入站规则对应的安全组名称。



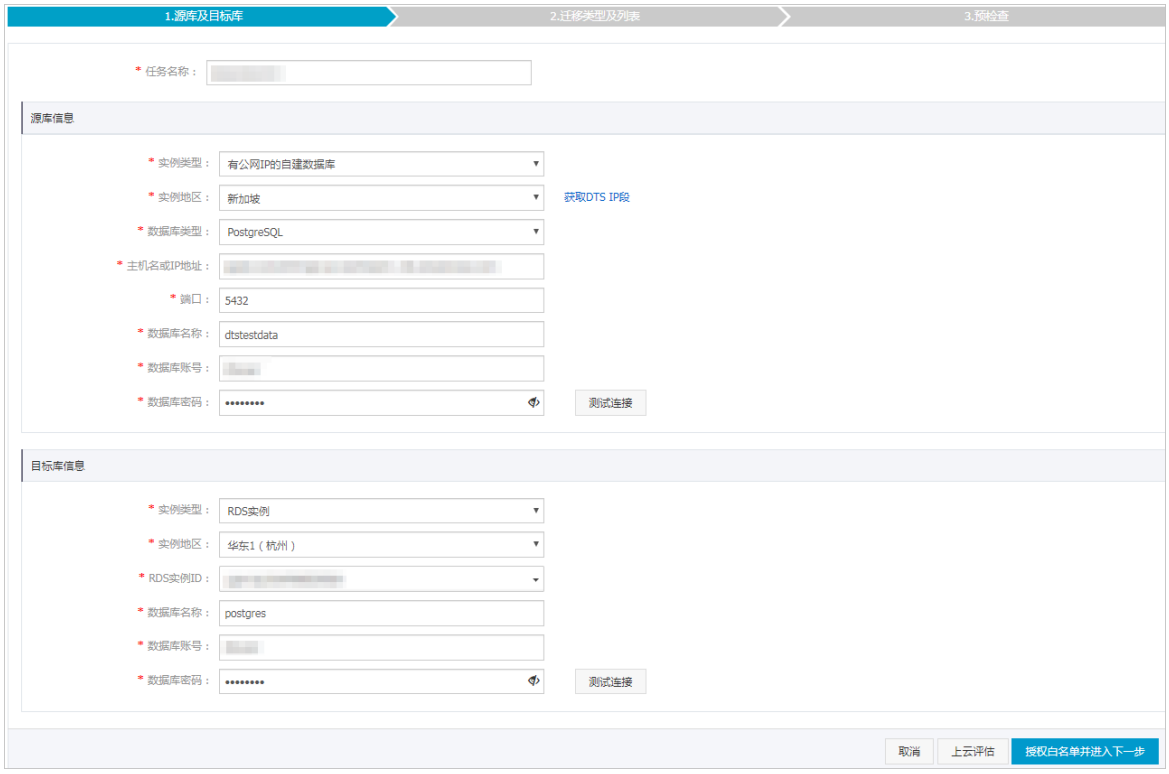
5. 在**安全组设置**页面，将对应区域的DTS服务器地址添加至入站规则中，IP地址段详情请参见迁移、同步或订阅本地数据库时需添加的IP白名单。



> **说明**
>
> - 您只需添加目标数据库所在区域对应的DTS IP地址段。例如，源数据库地区为新加坡，目标数据库地区为杭州，您只需要添加杭州地区的DTS IP地址段。
> - 在加入IP地址段时，您可以一次性添加所需的IP地址，无需逐条添加入站规则。

操作步骤

1. 登录数据传输控制台。
2. 在左侧导航栏，单击**数据迁移**。
3. 在**迁移任务列表**页面顶部，选择迁移的目标集群所属地域。
4. 单击页面右上角的**创建迁移任务**。
5. 配置迁移任务的源库及目标库信息。



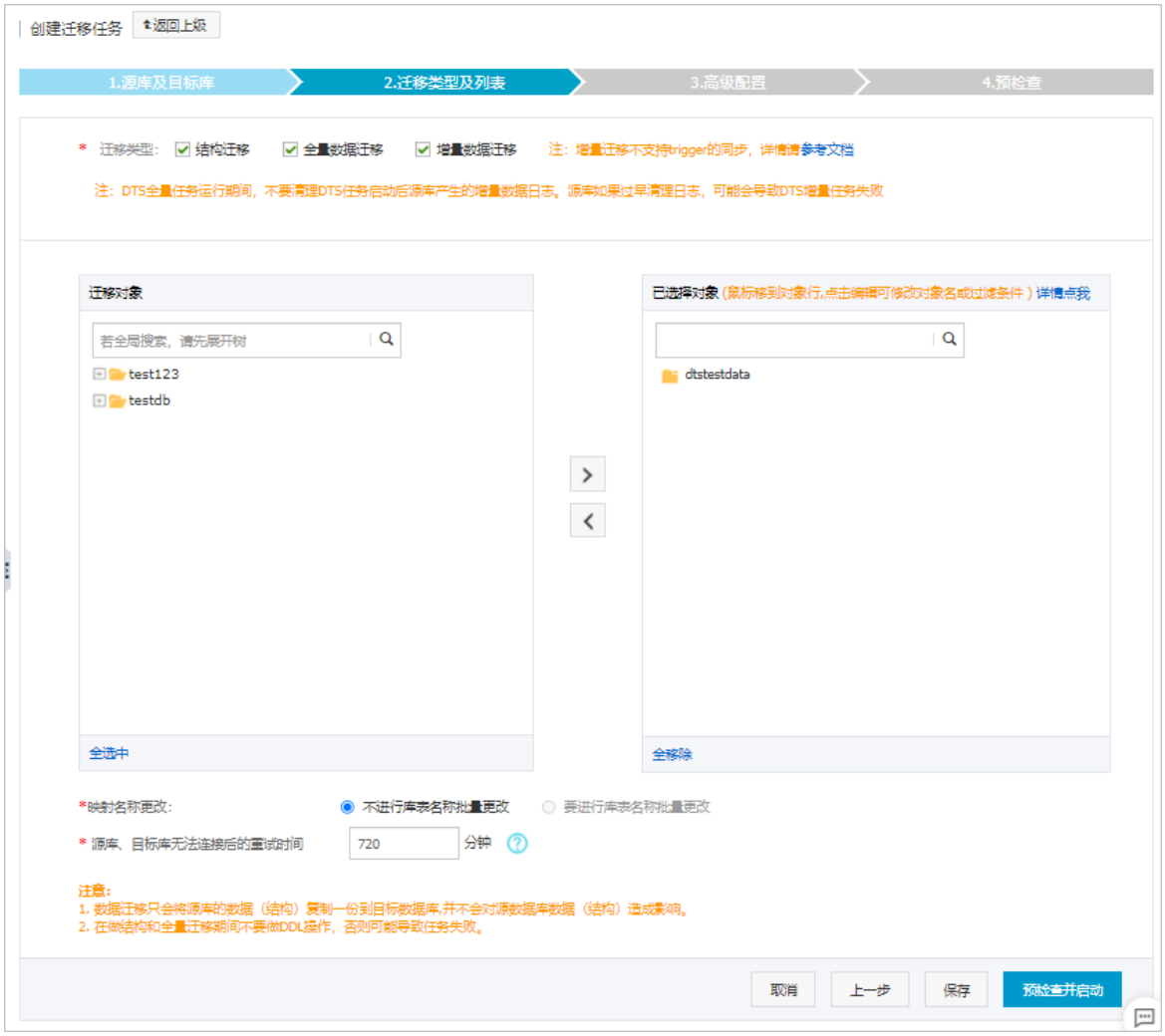
类别	配置	说明
无	任务名称	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
源实例信息	实例类型	选择**有公网IP的自建数据库**。
	实例地区	源实例所在的地区，当实例类型选择为**有公网IP的自建数据库**时，该参数无需设置。
	数据库类型	选择**PostgreSQL**。
	主机名或IP地址	填入Amazon RDS for PostgreSQL的连接地址。 **说明** 您可以在Amazon RDS for PostgreSQL的 **基本信息**页面，获取数据库的连接信息。
	端口	填入Amazon RDS for PostgreSQL的服务端口，默认为**5432**。

类别	配置	说明
	数据库名称	填入Amazon RDS for PostgreSQL中的待迁移的数据库名。
	数据库账号	填入Amazon RDS for PostgreSQL的数据库账号，权限要求请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号的密码。 **说明** 源库信息填写完毕后，您可以单击 **数据库密码**后的 **测试连接**来验证填入的信息是否正确。如果填写正确则提示 **测试通过**；如果提示 **测试失败**，单击 **测试失败**后的 **诊断**，根据提示调整填写的源库信息。
目标实例信息	实例类型	选择**RDS实例**。
	实例地区	选择阿里云RDS PostgreSQL实例所属地域。
	RDS实例ID	选择阿里云RDS PostgreSQL实例ID。
	数据库名称	填入阿里云RDS PostgreSQL中待迁入数据的数据库名，可以和Amazon RDS for PostgreSQL中待迁移的库名不同。 **说明** 该数据库须在阿里云RDS PostgreSQL中存在，如不存在请 创建数据库。
	数据库账号	填入阿里云RDS PostgreSQL的数据库账号，权限要求请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号的密码。 **说明** 目标库信息填写完毕后，您可以单击 **数据库密码**后的 **测试连接**来验证填入的信息是否正确。如果填写正确则提示 **测试通过**；如果提示 **测试失败**，单击 **测试失败**后的 **诊断**，根据提示调整填写的目标库信息。

6. 配置完成后，单击页面右下角的**授权白名单并进入下一步**。

> **说明** 此步骤会将DTS服务器的IP地址自动添加到阿里云RDS PostgreSQL的白名单中，用于保障DTS服务器能够正常连接阿里云RDS PostgreSQL。

7. 选择迁移对象和迁移类型。



配置	说明
迁移类型	同时勾选**结构迁移**、**全量数据迁移**和**增量数据迁移**。
迁移对象	在**迁移对象**框中单击待迁移的对象，然后单击 > 将其移动至**已选择对象**框。 说明 ○ 迁移对象选择的粒度为Schema、表、列。 ○ 默认情况下，迁移完成后，迁移对象名称保持不变。如果您需要改变迁移对象在阿里云RDS PostgreSQL中的名称，需要使用DTS提供的对象名映射功能。使用方法请参见库表列映射。 ○ 如果使用了对象名映射功能，可能会导致依赖这个对象的其他对象迁移失败。
映射名称更改	如需更改迁移对象在目标实例中的名称，请使用对象名映射功能 ，详情请参见库表列映射。

配置	说明
源、目标库无法连接重试时间	默认重试12小时，您也可以自定义重试时间。如果DTS在设置的时间内重新连接上源、目标库，迁移任务将自动恢复。否则，迁移任务将失败。 **说明** 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

8. 上述配置完成后，单击页面右下角的**预检查并启动**。

> **说明**
> - 在迁移任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动迁移任务。
> - 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
> - 您可以根据提示修复后重新进行预检查。
> - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

9. 预检查通过后，单击**下一步**。
10. 在弹出的**购买配置确认**对话框，选择**链路规格**并选中**数据传输（按量付费）服务条款**。
11. 单击**购买并启动**，迁移任务正式开始。

> **说明** 迁移任务不会自动结束，您需要手动结束迁移任务。请选择合适的时间手动结束迁移任务，例如业务低峰期或准备将业务切换至目标实例时。

   i. 观察迁移任务的进度变更为**增量迁移**，并显示为**无延迟**状态时，将源库停写几分钟，此时**增量迁移**的状态可能会显示延迟的时间。
   ii. 等待迁移任务的**增量迁移**再次进入**无延迟**状态后，手动结束迁移任务。

12. 将业务切换至阿里云RDS PostgreSQL。

10.3.2. 从Amazon RDS for PostgreSQL全量迁移至阿里云

本文介绍如何使用数据传输服务DTS（Data Transmission Service），将Amazon RDS PostgreSQL全量迁移至阿里云RDS PostgreSQL。

前提条件

- Amazon RDS PostgreSQL的数据库版本为9.4、9.5、9.6或10.0版本。
- 为保障DTS能够通过公网连接至Amazon RDS PostgreSQL，需要将Amazon RDS PostgreSQL的**公开可用性**设置为**是**。
- 已创建阿里云RDS PostgreSQL，详情请参见创建RDS PostgreSQL。

> **说明** 阿里云RDS PostgreSQL的数据库版本为9.4或10版本。

- 阿里云RDS PostgreSQL的存储空间，须大于Amazon RDS PostgreSQL中待迁移数据库占用的存储空间。

注意事项

- DTS在执行全量数据迁移时将占用源库和目标库一定的读写资源，可能会导致数据库的负载上升，在数据库性能较差、规格较低或业务量较大的情况下（例如源库有大量慢SQL、存在无主键表或目标库存在死锁等），可能会加重数据库压力，甚至导致数据库服务不可用。因此您需要在执行数据迁移前评估源库和目标库的性能，同时建议您在业务低峰期执行数据迁移（例如源库和目标库的CPU负载在30%以下）。
- 不支持增量数据迁移。即需要在执行数据迁移前，停止Amazon RDS PostgreSQL的相关业务，同时为了保障数据一致性，迁移期间请勿在Amazon RDS PostgreSQL中写入新的数据。
- 一个数据迁移任务只能迁移一个数据库，如果有多个数据库需要迁移，则需要为每个数据库创建数据迁移任务。
- 不支持迁移使用C语言编写的function。
- 如果源库中待迁移的表没有主键或唯一约束，且所有字段没有唯一性，可能会导致目标数据库中出现重复数据。
- 对于迁移失败的任务，DTS会触发自动恢复。在您将业务切换至目标实例前，请务必先结束或释放该任务，避免该任务被自动恢复后，导致源端数据覆盖目标实例的数据。
- 为保障迁移任务的正常进行，目前仅支持RDS PostgreSQL 11进行主备切换，且需设置参数 `rds_failover_slot_mode` 为 `sync` ，设置方式，请参见逻辑复制槽故障转移（Logical Replication Slot Failover）。

> **警告** 自建PostgreSQL和其他版本的RDS PostgreSQL进行主备切换，会导致迁移中断。

费用说明

迁移类型	链路配置费用	公网流量费用
结构迁移和全量数据迁移	不收费	通过公网将数据迁移出阿里云时将收费，详情请参见产品定价。

迁移类型说明

- 结构迁移
  DTS将迁移对象的结构定义迁移至阿里云RDS PostgreSQL，支持结构迁移的对象包含table、trigger、view、sequence、function、user defined type、rule、domain、operation及aggregate。
- 全量数据迁移
  将迁移对象的存量数据全部迁移到阿里云RDS PostgreSQL数据库中。

数据库账号的权限要求

数据库	结构迁移	全量迁移
Amazon RDS PostgreSQL	pg_catalog的usage权限	迁移对象的select权限
阿里云RDS PostgreSQL	迁移对象的create、usage权限	schema的owner权限

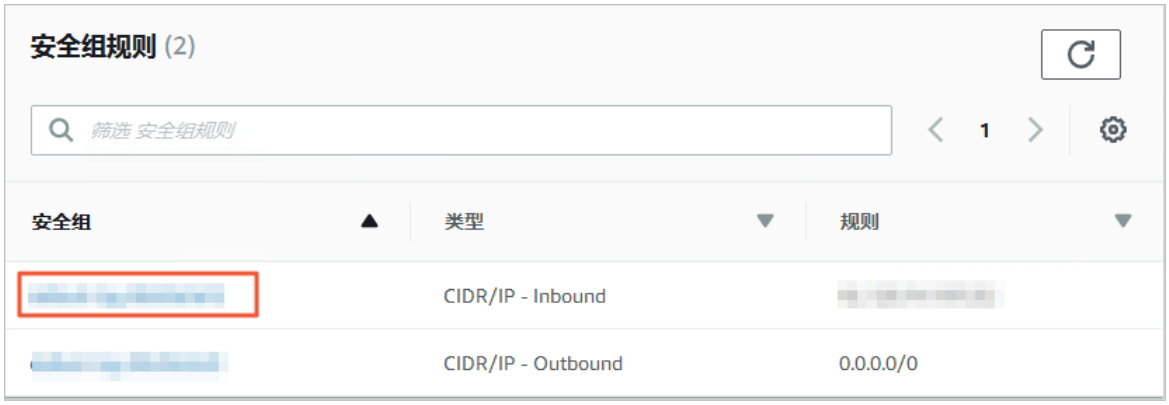
全量数据迁移流程

为解决对象间的依赖，提高迁移成功率，DTS对PostgreSQL结构及数据的迁移顺序如下。

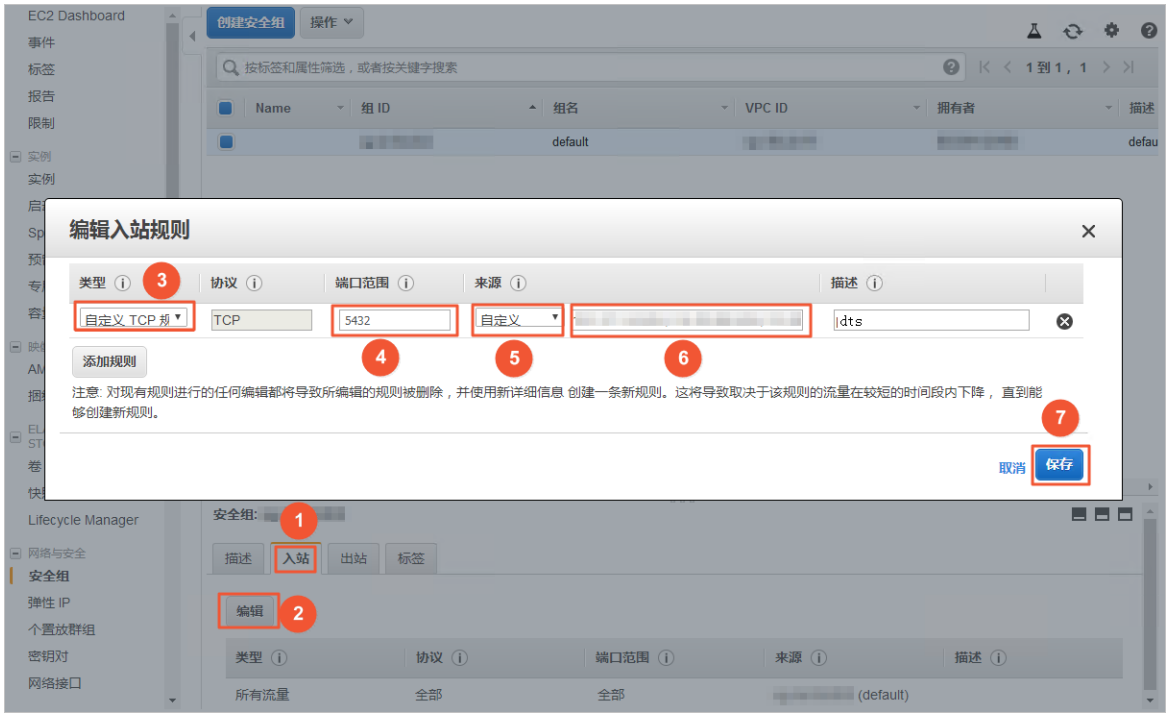
1. 执行Table、view、sequence、function、user defined type、rule、domain、operation、aggregate的结构迁移。
2. 执行全量数据迁移。
3. 执行trigger、foreign key的结构迁移。

准备工作一：调整Amazon的入站规则

1. 登录Amazon RDS控制台。
2. 进入Amazon RDS PostgreSQL的**基本信息**页面。
3. 在**安全组规则**区域框，单击入站规则对应的安全组名称。



4. 在**安全组设置**页面，将对应区域的DTS服务器地址添加至入站规则中，IP地址段详情请参见迁移、同步或订阅本地数据库时需添加的IP白名单。



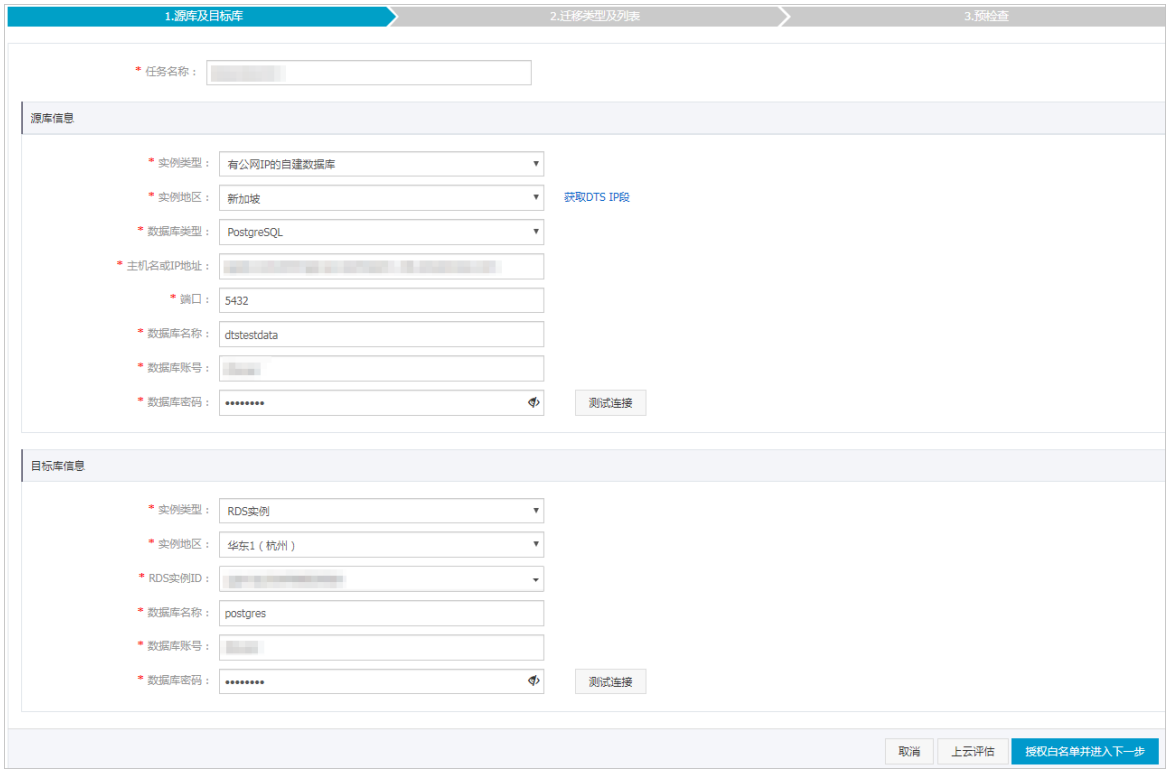
> **说明**
>
> - 您只需添加目标数据库所在区域对应的DTS IP地址段。例如，源数据库地区为新加坡，目标数据库地区为杭州，您只需要添加杭州地区的DTS IP地址段。
> - 在加入IP地址段时，您可以一次性添加所需的IP地址，无需逐条添加入站规则。

准备工作二：在阿里云RDS实例中创建数据库和schema

根据待迁移对象所属的数据库和schema信息，在阿里云RDS PostgreSQL中创建相应数据库和schema（schema名称须一致），详情请参见创建数据库和schema管理。

操作步骤

1. 登录数据传输控制台。
2. 在左侧导航栏，单击**数据迁移**。
3. 在**迁移任务列表**页面顶部，选择迁移的目标集群所属地域。
4. 单击页面右上角的**创建迁移任务**。
5. 配置迁移任务的源库及目标库信息。



类别	配置	说明
无	任务名称	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
源实例信息	实例类型	选择**有公网IP的自建数据库**。
	实例地区	源实例所在的地区，当实例类型选择为**有公网IP的自建数据库**时，该参数无需设置。
	数据库类型	选择**PostgreSQL**。
	主机名或IP地址	填入Amazon RDS PostgreSQL的连接地址。 **说明** 您可以在Amazon RDS PostgreSQL的 **基本信息** 页面，获取数据库的连接信息。
	端口	填入Amazon RDS PostgreSQL的服务端口，默认为**5432**。

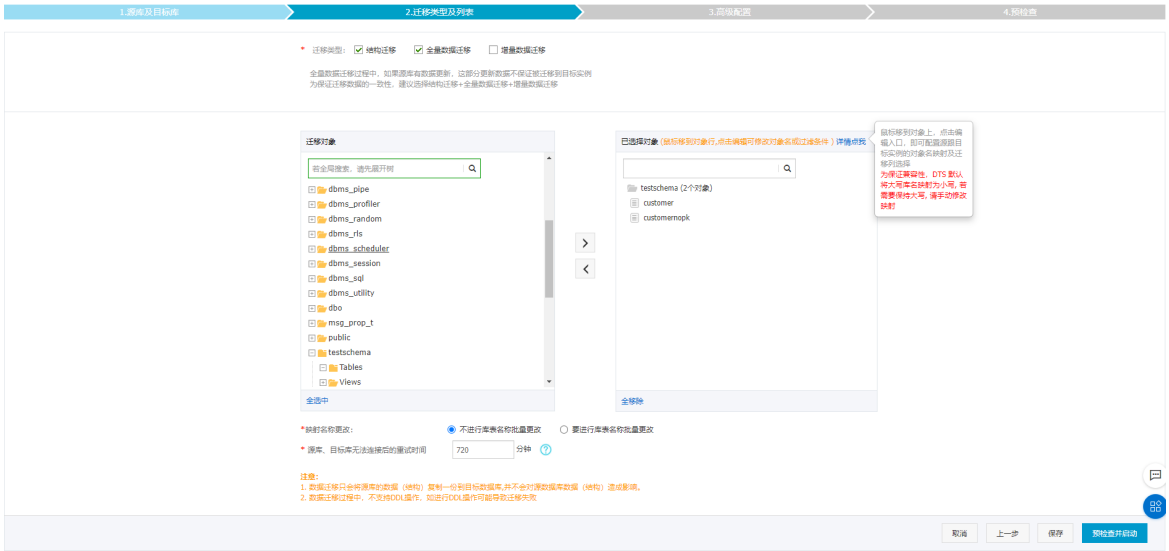
类别	配置	说明
	数据库名称	填入Amazon RDS PostgreSQL中的待迁移的数据库名。
	数据库账号	填入Amazon RDS PostgreSQL的数据库账号，权限要求请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号的密码。 **说明** 源库信息填写完毕后，您可以单击 **数据库密码**后的 **测试连接**来验证填入的信息是否正确。如果填写正确则提示 **测试通过**；如果提示 **测试失败**，单击 **测试失败**后的 **诊断**，根据提示调整填写的源库信息。
目标实例信息	实例类型	选择**RDS实例**。
	实例地区	选择阿里云RDS PostgreSQL实例所属地域。
	RDS实例ID	选择阿里云RDS PostgreSQL实例ID。
	数据库名称	填入阿里云RDS PostgreSQL中待迁入数据的数据库名，可以和Amazon RDS PostgreSQL中待迁移的库名不同。 **说明** 您需要提前在阿里云RDS PostgreSQL中创建相应的数据库和schema，详情请参见 准备工作二：在阿里云RDS实例中创建数据库和schema。
	数据库账号	填入阿里云RDS PostgreSQL的数据库账号，权限要求请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号的密码。 **说明** 目标库信息填写完毕后，您可以单击 **数据库密码**后的 **测试连接**来验证填入的信息是否正确。如果填写正确则提示 **测试通过**；如果提示 **测试失败**，单击 **测试失败**后的 **诊断**，根据提示调整填写的目标库信息。

6. 配置完成后，单击页面右下角的**授权白名单并进入下一步**。

警告

- 如果源或目标数据库是阿里云数据库实例（例如RDS MySQL、云数据库MongoDB版等）或ECS上的自建数据库，DTS会自动将对应地区DTS服务的IP地址添加到阿里云数据库实例的白名单或ECS的安全规则中，您无需手动添加，请参见DTS服务器的IP地址段；如果源或目标数据库是IDC自建数据库或其他云数据库，则需要您手动添加对应地区DTS服务的IP地址，以允许来自DTS服务器的访问。
- 上述场景中，DTS自动添加或您手动添加DTS服务的公网IP地址段可能会存在安全风险，一旦使用本产品代表您已理解和确认其中可能存在的安全风险，并且需要您做好基本的安全防护，包括但不限于加强账号密码强度防范、限制各网段开放的端口号、内部各API使用鉴权方式通信、定期检查并限制不需要的网段，或者使用通过内网（专线/VPN网关/智能网关）的方式接入。
- DTS任务完成或释放后，建议您手动检测并删除DTS相关的服务器IP地址段。

7. 选择迁移对象和迁移类型。



配置	说明
迁移类型	暂不支持**增量数据迁移**，此处同时勾选**结构迁移**和**全量数据迁移**。 **说明** 为保障数据一致性，迁移期间请勿在Amazon RDS PostgreSQL数据库中写入新的数据。

配置	说明
迁移对象	在**迁移对象**框中单击待迁移的对象，然后单击 › 将其移动至**已选择对象**框。 说明 ○ 迁移对象选择的粒度为库、表、列度。 ○ 默认情况下，迁移对象在目标库中的名称与源库保持一致。如果您需要改变迁移对象在目标库中的名称，需要使用对象名映射功能，详情请参见库表列映射。 ○ 如果使用了对象名映射功能，可能会导致依赖这个对象的其他对象迁移失败。
映射名称更改	如需更改迁移对象在目标实例中的名称，请使用对象名映射功能 ，详情请参见库表列映射。
源、目标库无法连接重试时间	默认重试12小时，您也可以自定义重试时间。如果DTS在设置的时间内重新连接上源、目标库，迁移任务将自动恢复。否则，迁移任务将失败。 说明 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

8. 上述配置完成后，单击页面右下角的**预检查并启动**。

说明
- 在迁移任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动迁移任务。
- 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
  - 您可以根据提示修复后重新进行预检查。
  - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

9. 预检查通过后，单击**下一步**。
10. 在弹出的**购买配置确认**对话框，选择**链路规格**并选中**数据传输（按量付费）服务条款**。
11. 单击**购买并启动**，迁移任务正式开始。

说明 请勿手动结束迁移任务，否则可能会导致数据不完整。您只需等待迁移任务完成即可，迁移任务会自动结束。

12. 将业务切换至阿里云RDS PostgreSQL。

10.4. RDS PostgreSQL间的数据迁移

本文介绍如何使用数据传输服务DTS（Data Transmission Service），实现RDS PostgreSQL间的结构迁移、全量数据迁移以及增量数据迁移。同时使用这三种迁移类型可以实现在业务不停服的情况下，平滑地完成数据库的迁移。

前提条件

- 已创建源和目标RDS PostgreSQL实例，详情请参见创建RDS PostgreSQL实例。

> **说明** 建议源和目标数据库的版本相同，或者从低版本迁移到高版本以保障兼容性。如为高版本迁移至低版本，可能存在数据库兼容性问题。

- 目标RDS PostgreSQL实例的存储空间须大于源RDS PostgreSQL实例占用的存储空间。

注意事项

类型	说明

源库限制	• 待迁移的表需具备主键或唯一约束，且字段具有唯一性，否则可能会导致目标数据库中出现重复数据。
	待迁移的数据库名称中间不能包含短划线（-），例如dts-testdata。
	• 如迁移对象为表级别，且需进行编辑（如表列名映射），则单次迁移任务仅支持迁移至多1000张表。当超出数量限制，任务提交后会显示请求报错，此时建议您拆分待迁移的表，分批配置多个任务，或者配置整库的迁移任务。
	• 如需进行增量迁移，WAL日志：
	  ○ 需开启，即设置 wal_level 参数的值为*logical*。
	  ○ 如为增量迁移任务，DTS要求源数据库的WAL日志保存24小时以上，如为全量迁移和增量迁移任务，DTS要求源数据库的WAL日志至少保留7天以上（您可在全量迁移完成后将WAL日志保存时间设置为24小时以上），否则DTS可能因无法获取WAL日志而导致任务失败，极端情况下甚至可能会导致数据不一致或丢失。由于您所设置的WAL日志保存时间低于DTS要求的时间进而导致的问题，不在DTS的SLA保障范围内。
	• 源库的操作限制：
	  ○ 在库表结构迁移和全量迁移阶段，请勿执行库或表结构变更的DDL操作，否则数据迁移任务会失败。
	  ○ 如仅执行全量数据迁移，请勿向源库中写入新的数据，否则会导致源和目标数据不一致。为实时保持数据一致性，建议选择结构迁移、全量数据迁移和增量数据迁移。

	• 为保障迁移任务的正常进行，避免主备切换导致的逻辑订阅中断，需要RDS PostgreSQL支持并开启Logical Replication Slot Failover，设置方式，请参见逻辑复制槽故障转移（Logical Replication Slot Failover）。 • 单个数据迁移任务只能迁移一个数据库，如需迁移多个数据库，您需要为每个数据库配置数据迁移任务。 • 增量数据迁移期间，如果迁移对象的选择粒度为Schema，在待迁移的Schema中创建了新的表或使用RENAME命令重建了待迁移的表，您需要在对该表写入数据前执行 `ALTER TABLE schema.table REPLICA IDENTITY FULL;` 命令。 **说明** 将上述命令中的 `schema` 和 `table` 替换成真实的Schema名和表名。

类型	说明
其他限制	• DTS的校验对象为数据内容，暂不支持Sequence等元数据的校验，您需要自行校验。 • 由于业务切换到目标端后，新写入的Sequence不会按照源库的Sequence最大值作为初始值去递增，您需要在业务切换前，在源库中查询对应Sequence的最大值，然后在目标库中将其作为对应Sequence的初始值。查询源库Sequence值的相关命令如下： `do language plpgsql $$ declare nsp name; rel name; val int8; begin for nsp,rel in select nspname,relname from pg_class t2 , pg_namespace t3 where t2.relnamespace=t3.oid and t2.relkind='S' loop execute format($_$select last_value from %I.%I$_$, nsp, rel) into val; raise notice '%', format($_$select setval('%I.%I'::regclass, %s);$_$, nsp, rel, val+1); end loop; end; $$;` • DTS会在源库中创建以下临时表，用于获取增量数据的DDL、增量表的结构、心跳信息等。在迁移期间，请勿删除源库中的临时表，否则会导致DTS任务异常。临时表会在DTS实例释放后自动删除。 `public.DTS_PG_CLASS`、`public.DTS_PG_ATTRIBUTE`、`public.DTS_PG_TYPE`、`public.DTS_PG_ENUM`、`public.DTS_POSTGRES_HEARTBEAT`、`public.DTS_DDL_COMMAND`、`public.DTS_ARGS_SESSION`。 • 为保障增量数据迁移延迟时间展示的准确性，DTS会在源库中新增一个名为`dts_postgres_heartbeat`的心跳表。 • 增量数据迁移期间，DTS会在源库中创建前缀为`dts_sync_`的replication slot用于复制数据。DTS会每隔90分钟自动清理一次历史replication slot，避免其持续累积占用磁盘空间。 说明 ○ DTS释放实例后会主动删除该replication slot，如果您在同步期间修改了数据库密码，或者删除了DTS的访问IP白名单，则会导致该replication slot无法自动删除，此时需要您在源库手动删除，避免其持续累积占用磁盘空间导致RDS PostgreSQL实例不可用。 ○ 当释放迁移任务或迁移失败时，DTS会主动清理该replication slot；如果RDS PostgreSQL发生了主备切换，则需要您登录备库来手动清理。 • 执行数据迁移前需评估源库和目标库的性能，同时建议业务低峰期执行数据迁移。否则全量数据迁移时DTS占用源和目标库一定读写资源，可能会导致数据库的负载上升。 • 由于全量数据迁移会并发执行INSERT操作，导致目标数据库的表产生碎片，因此全量迁移完成后目标库的表存储空间会比源库的表存储空间大。 • 请确认DTS对数据类型为FLOAT或DOUBLE的列的迁移精度是否符合业务预期。DTS会通过`ROUND(COLUMN,PRECISION)`来读取这两类列的值。如果没有明确定义其精度，DTS对FLOAT的迁移精度为38位，对DOUBLE的迁移精度为308位。 • DTS会尝试恢复七天之内迁移失败任务。因此业务切换至目标实例前，请务必结束或释放该任务，或者将DTS访问目标实例账号的写权限用`revoke`命令回收掉。避免该任务被自动恢复后，源端数据覆盖目标实例的数据。

类型	说明
特殊情况	当源实例为RDS PostgreSQL时，迁移期间，请勿修改RDS PostgreSQL的连接地址和可用区，否则会导致迁移失败。

迁移类型说明

- 库表结构迁移
  DTS将源库中迁移对象的结构定义迁移到目标库。
- 全量迁移
  DTS将源库中迁移对象的存量数据，全部迁移到目标库中。
- 增量迁移
  DTS在全量迁移的基础上，将源库的增量更新数据迁移到目标库中。通过增量数据迁移可以实现在自建应用不停服的情况下，平滑地完成数据迁移。

支持增量迁移的SQL操作

操作类型	SQL操作语句
DML	INSERT、UPDATE、DELETE

数据库账号的权限要求

数据库	库表结构迁移	全量迁移	增量迁移
源RDS PostgreSQL实例	pg_catalog的USAGE权限。	迁移对象的SELECT权限。	高权限账号。 **说明** 当源实例为RDS PostgreSQL 9.4，且仅需增量迁移DML操作时，账号具备REPLICATION权限即可。
目标RDS PostgreSQL实例	目标迁移对象的CREATE、USAGE权限。	Schema的owner权限。	

RDS PostgreSQL数据库账号创建及授权方法，请参见创建账号。

操作步骤

1. 登录新版DTS迁移任务的列表页面。

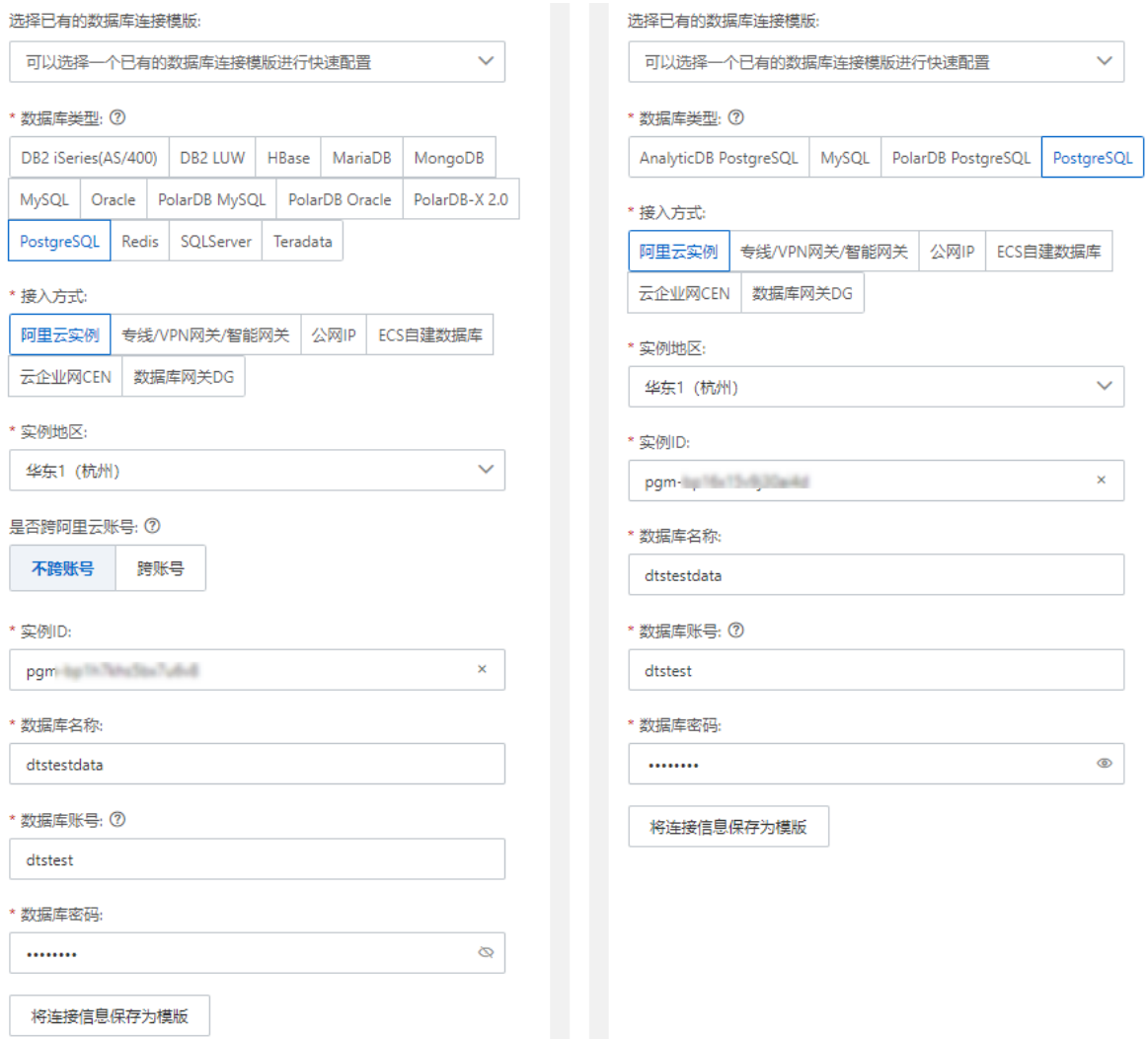
   说明 您也可以登录 DMS数据管理服务。在顶部菜单栏中，选择 **集成与开发（DTS）** > **数据迁移**。

2. 在页面左上角，选择迁移实例所属地域。

   换图

3. 单击**创建任务**，配置源库及目标库信息。

> **警告** 选择源和目标实例后，建议您仔细阅读页面上方显示的**使用限制**，以成功创建并执行迁移任务。



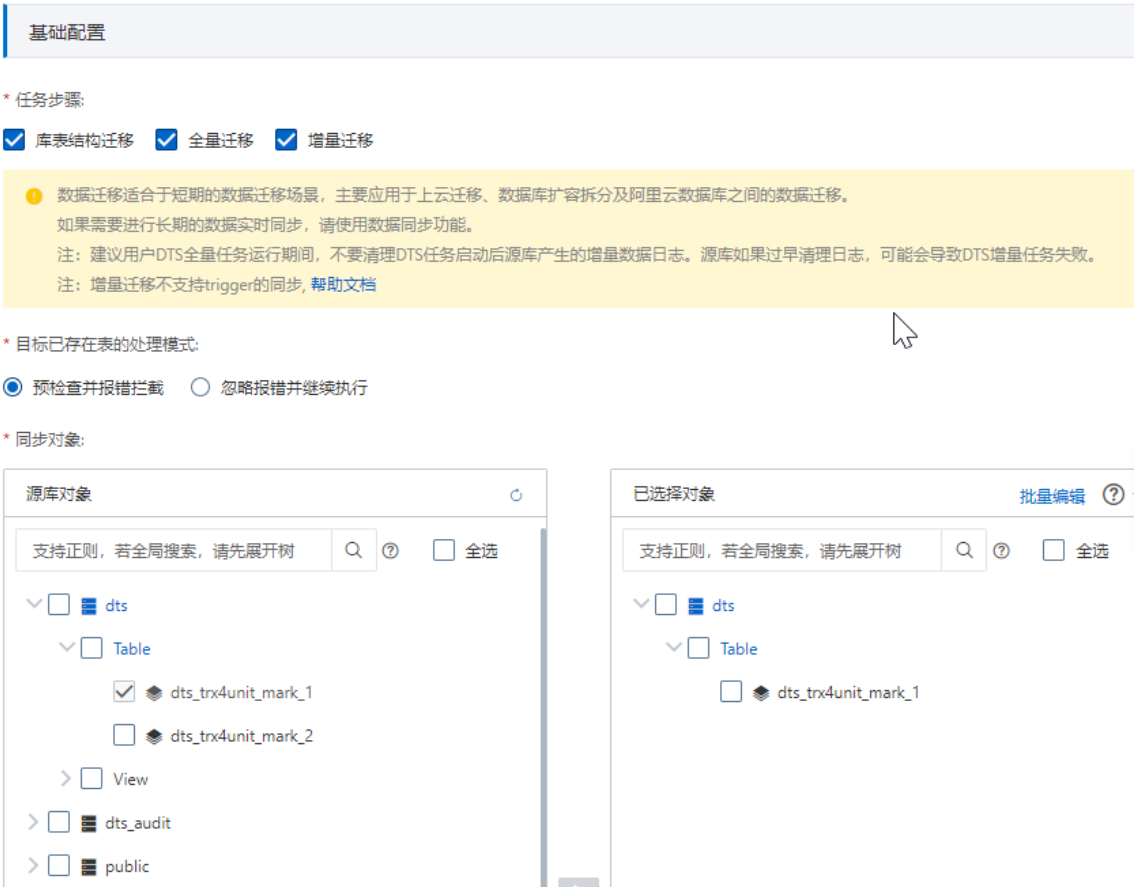
类别	配置	说明
无	**任务名称**	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
	数据库类型	选择PostgreSQL。

类别	配置	说明
源库信息	**接入方式**	选择**阿里云实例**。
	实例地区	选择源RDS PostgreSQL实例所属地域。
	实例ID	选择源RDS PostgreSQL实例ID。
	数据库名称	填入源RDS PostgreSQL实例中迁移对象所属数据库的名称。
	数据库账号	填入源RDS PostgreSQL实例的数据库账号，权限要求，请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号对应的密码。
目标库信息		
	数据库类型	选择**PostgreSQL**。
	接入方式	选择**阿里云实例**。
	实例地区	选择目标RDS PostgreSQL实例所属地域。
	实例ID	选择目标RDS PostgreSQL实例ID。
	数据库名称	填入目标RDS PostgreSQL实例中迁移对象所属数据库的名称。
	数据库账号	填入目标RDS PostgreSQL实例的数据库账号，权限要求，请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号对应的密码。

4. 配置完成后，单击页面右下角的**测试连接以进行下一步**。

> **警告**
>
> - 如果源或目标数据库是阿里云数据库实例（例如RDS MySQL、云数据库MongoDB版等）或ECS上的自建数据库，DTS会自动将对应地区DTS服务的IP地址添加到阿里云数据库实例的白名单或ECS的安全规则中，您无需手动添加，请参见DTS服务器的IP地址段；如果源或目标数据库是IDC自建数据库或其他云数据库，则需要您手动添加对应地区DTS服务的IP地址，以允许来自DTS服务器的访问。
> - 上述场景中，DTS自动添加或您手动添加DTS服务的公网IP地址段可能会存在安全风险，一旦使用本产品代表您已理解和确认其中可能存在的安全风险，并且需要您做好基本的安全防护，包括但不限于加强账号密码强度防范、限制各网段开放的端口号、内部各API使用鉴权方式通信、定期检查并限制不需要的网段，或者使用通过内网（专线/VPN网关/智能网关）的方式接入。
> - DTS任务完成或释放后，建议您手动检测并删除DTS相关的服务器IP地址段。

5. 配置任务对象及高级配置。
   - **基础配置**



配置	说明
任务步骤	全支持 ■ 如果只需要进行全量迁移，请同时选中**库表结构迁移**和**全量迁移**。 ■ 如果需要进行不停机迁移，请同时选中**库表结构迁移**、**全量迁移**和**增量迁移**。 **说明** 如果未选择 **增量迁移**，为保障数据一致性，数据迁移期间请勿在源实例中写入新的数据。

配置	说明
目标已存在表的处理模式	■ **预检查并报错拦截**：检查目标数据库中是否有同名的表。如果目标数据库中没有同名的表，则通过该检查项目；如果目标数据库中有同名的表，则在预检查阶段提示错误，数据迁移任务不会被启动。 **说明** 如果目标库中同名的表不方便删除或重命名，您可以更改该表在目标库中的名称，请参见 库表列名映射。 ■ **忽略报错并继续执行**：跳过目标数据库中是否有同名表的检查项。 **警告** 选择为 **忽略报错并继续执行**，可能导致数据不一致，给业务带来风险，例如： ■ 表结构一致的情况下，在目标库遇到与源库主键的值相同的记录，则会保留目标库中的该条记录，即源库中的该条记录不会迁移至目标库中。 ■ 表结构不一致的情况下，可能导致只能迁移部分列的数据或迁移失败。
同步对象	在**同步对象**框中单击待迁移的对象，然后单击 › 将其移动到**已选择对象**框。 **说明** 迁移对象选择的粒度为schema、表、列。若选择的迁移对象为表或列，其他对象（如视图、触发器、存储过程）不会被迁移至目标库。
映射名称更改	■ 如需更改单个迁移对象在目标实例中的名称，请右击**已选择对象**中的迁移对象，设置方式，请参见库表列名单个映射。 ■ 如需批量更改迁移对象在目标实例中的名称，请单击**已选择对象**方框右上方的 批量编辑 设置方式，请参见库表列名批量映射。 **说明** 如果使用了对象名映射功能，可能会导致依赖这个对象的其他对象迁移失败。
过滤待迁移数据	支持设置条件过滤数据，详情请参见通过SQL条件过滤任务数据。
增量迁移的SQL操作	选择增量迁移SQL操作，请右击**已选择对象**中的迁移对象，在弹跳框中选择所需增量迁移的SQL操作。支持的操作，请参见支持增量迁移的SQL操作。

- **高级配置**

配置	说明
设置告警	是否设置告警，当迁移失败或延迟超过阈值后，将通知告警联系人。 ■ **不设置**：不设置告警。 ■ **设置**：设置告警，您还需要设置告警阈值和告警联系人。
目标库对象名称大小写策略	您可以配置目标实例中迁移对象的库名、表名和列名的英文大小写策略。默认情况下选择**DTS默认策略**，您也可以选择与源库、目标库默认策略保持一致。更多信息，请参见目标库对象名称大小写策略。
源、目标库无法连接重试时间	默认重试720分钟，您也可以在取值范围（10~720分钟）内自定义重试时间，建议设置30分钟以上。如果DTS在设置的时间内重新连接上源、目标库，同步任务将自动恢复。否则，同步任务将失败。 **说明** ■ 针对同源或者同目标的多个DTS实例，如DTS实例A和DTS实例B，设置网络重试时间时A设置30分钟，B设置60分钟，则重后创建任务为准。 ■ 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

6. 上述配置完成后，单击页面右下角的**下一步保存任务并预检查**。

> **说明**
> - 在迁移任务正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动迁移任务。
> - 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
> - 您可以根据提示修复后重新进行预检查。
> - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

7. **预检查通过率**显示为**100%**时，单击**下一步购买**。
8. 在**购买**页面，选择数据迁移实例的链路规格，详细说明请参见下表。

类别	参数	说明
信息配置	**链路规格**	DTS为您提供了不同性能的迁移规格，迁移链路规格的不同会影响迁移速率，您可以根据业务场景进行选择，详情请参见数据迁移链路规格说明。

9. 配置完成后，阅读并选中**《数据传输（按量付费）服务条款》**。
10. 单击**购买并启动**，迁移任务正式开始，您可在数据迁移界面查看具体进度。

10.5. 迁移RDS PostgreSQL数据到本地PostgreSQL

阿里云数据库PostgreSQL版支持通过逻辑备份文件将云上数据迁移到本地数据库。

操作步骤

1. 通过PostgreSQL客户端，连接云数据库。
2. 执行如下命令，备份数据。

```
pg_dump -U username -h hostname -p port databasename -f filename --exclude-table=public
.ha_health_check
```

参数说明如下：

- username：云数据库用户名。
- hostname：云数据库主机名。
- port：云数据库端口号。
- databasename：要备份的数据库名。
- filename：要生成的备份文件名称。
- --exclude-table=public.ha_health_check：用于跳过高可用检查表。

示例

```
pg_dump -U myuser -h rds2z2tp80v3752wb455.pg.rds.aliyuncs.com -p 3433 pg001 -f pg001.sq
l --exclude-table=public.ha_health_check
```

3. 将备份文件*pg001.sql*上传到目标服务器中。

4. 执行如下命令将数据恢复到本地数据库。

```
psql -U username -h hostname -d desintationdb -p port -f dumpfilename.sql
```

参数说明如下：

- username：本地数据库用户名。
- hostname：本地数据库地址。
- port：本地数据库端口号。
- desintationdb：目的数据库名。
- dumpfilename：备份文件名称。

示例

```
psql -U myuser -h localhost -d pg001 -p 5432 -f pg001.sql
```

由于 RDS 数据库的权限设置和本地数据库不一致，在数据导入过程当中可能会出现一些与权限相关的 WARNING 或 ERROR，可以忽略，例如：

```
WARNING: no privileges could be revoked for "xxxxx" ERROR: role "xxxxx" does not exist
```

11.数据同步

11.1. RDS PostgreSQL间的单向同步

本文介绍如何使用数据传输服务DTS（Data Transmission Service），实现RDS PostgreSQL间的单向同步。

前提条件

- 已创建源库和目标数据库 RDS PostgreSQL 实例，创建方式，请参见创建RDS PostgreSQL实例。

> **说明** 建议源库和目标数据库的版本相同，或者从低版本同步到高版本以保障兼容性。如为高版本同步至低版本，可能存在数据库兼容性问题。

- 目标实例RDS PostgreSQL的存储空间须大于源实例RDS PostgreSQL占用的存储空间。

注意事项

类型	说明

源库限制	• 待同步的表需具备主键或唯一约束，且字段具有唯一性，否则可能会导致目标数据库中出现重复数据。
	• 如同步对象为表级别，且需进行编辑（如表列名映射），则单次同步任务仅支持同步至多1000张表。当超出数量限制，任务提交后会显示请求报错，此时建议您拆分待同步的表，分批配置多个任务，或者配置整库的同步任务。
	• WAL日志：
	  ○ 需开启，即设置 wal_level 参数的值为*logical*。
	  ○ 如为增量同步任务，DTS要求源数据库的WAL日志保存24小时以上，如为全量同步和增量同步任务，DTS要求源数据库的WAL日志至少保留7天以上（您可在全量同步完成后将WAL日志保存时间设置为24小时以上），否则DTS可能因无法获取WAL日志而导致任务失败，极端情况下甚至可能会导致数据不一致或丢失。由于您所设置的WAL日志保存时间低于DTS要求的时间进而导致的问题，不在DTS的SLA保障范围内。
	• 源库的操作限制
	为保障同步任务的正常进行，避免主备切换导致的逻辑订阅中断，需要RDS PostgreSQL支持并开启Logical Replication Slot Failover，设置方式，请参见逻辑复制槽故障转移（Logical Replication Slot Failover）。

	• 单个数据同步任务只能同步一个数据库，如需同步多个数据库，您需要为每个数据库配置数据同步任务。
	• 不支持数组（ARRAY）类型的数据同步。
	• 同步期间，如果同步对象的选择粒度为Schema，您需要在对该表写入数据前执行 `ALTER TABLE schema.table REPLICA IDENTITY FULL;` 命令，以保障该表数据同步的一致性。 执行该命令期间，建议您不要有锁表操作，否则会导致表锁死。
	  说明
	  ○ 将上述命令中的 `schema` 和 `table` 替换成真实的Schema名和表名。
	  ○ 建议您在业务低峰期进行操作。
	• DTS的校验对象为数据内容，暂不支持Sequence等元数据的校验，您需要自行校验。

类型	说明
其他限制	• 由于业务切换到目标端后，新写入的Sequence不会按照源库的Sequence最大值作为初始值去递增，您需要在业务切换前，在源库中查询对应Sequence的最大值，然后在目标库中将其作为对应Sequence的初始值。查询源库Sequence值的相关命令如下： `do language plpgsql $$ declare nsp name; rel name; val int8; begin for nsp,rel in select nspname,relname from pg_class t2 , pg_namespace t3 where t2.relnamespace=t3.oid and t2.relkind='S' loop execute format($_$select last_value from %I.%I$_$, nsp, rel) into val; raise notice '%', format($_$select setval('%I.%I'::regclass, %s);$_$, nsp, rel, val+1); end loop; end; $$;` • DTS会在源库中创建以下临时表，用于获取增量数据的DDL、增量表的结构、心跳信息等。在同步期间，请勿删除源库中的临时表，否则会导致DTS任务异常。临时表会在DTS实例释放后自动删除。`public.DTS_PG_CLASS`、`public.DTS_PG_ATTRIBUTE`、`public.DTS_PG_TYPE`、`public.DTS_PG_ENUM`、`public.DTS_POSTGRES_HEARTBEAT`、`public.DTS_DDL_COMMAND`、`public.DTS_ARGS_SESSION`。 • 为保障同步延迟时间展示的准确性，DTS会在源库中新增一个名为*dts_postgres_heartbeat*的心跳表。 • 同步期间，DTS会在源库中创建前缀为`dts_sync_`的replication slot用于复制数据。DTS会每隔90分钟自动清理一次历史replication slot，避免其持续累积占用磁盘空间。 说明 ○ DTS释放实例后会主动删除该replication slot，如果您在同步期间修改了数据库密码，或者删除了DTS的访问IP白名单，则会导致该replication slot无法自动删除，此时需要您在源库手动删除，避免其持续累积占用磁盘空间导致RDS PostgreSQL实例不可用。 ○ 当释放同步任务或同步失败时，DTS会主动清理该replication slot；如果RDS PostgreSQL发生了主备切换，则需要您登录备库来手动清理。 • 执行数据同步前需评估源库和目标库的性能，同时建议业务低峰期执行数据同步。否则全量数据初始化时将占用源库和目标库一定的读写资源，可能会导致数据库的负载上升。 • 全量初始化会并发执行INSERT操作，导致目标数据库的表产生碎片，因此全量初始化完成后目标实例的表空间比源实例的表空间大。 • 对于表级别的数据同步，请勿对源库的同步对象使用gh-ost或pt-online-schema-change等类似工具执行在线DDL变更，否则会导致同步失败。 • 对于表级别的数据同步，如果除DTS以外的数据没有写入目标库，您可以使用数据管理DMS（Data Management Service）来执行在线DDL变更，请参见不锁表结构变更。 • 在DTS同步期间，不允许有除DTS外的数据写入目标库，否则会导致源库与目标库数据不一致。例如，有除DTS外的数据写入目标库时，使用DMS执行在线DDL变更，可能引起目标库数据丢失。

类型	说明
特殊情况	当源实例为RDS PostgreSQL时 同步期间，请勿修改RDS PostgreSQL的连接地址和可用区，否则会导致同步失败。

支持的同步架构

- 一对一单向同步
- 一对多单向同步
- 级联单向同步
- 多对一单向同步

关于各类同步架构的介绍及注意事项，请参见数据同步拓扑介绍。

支持同步的SQL操作

操作类型	SQL操作语句
DML	INSERT、UPDATE、DELETE

操作步骤

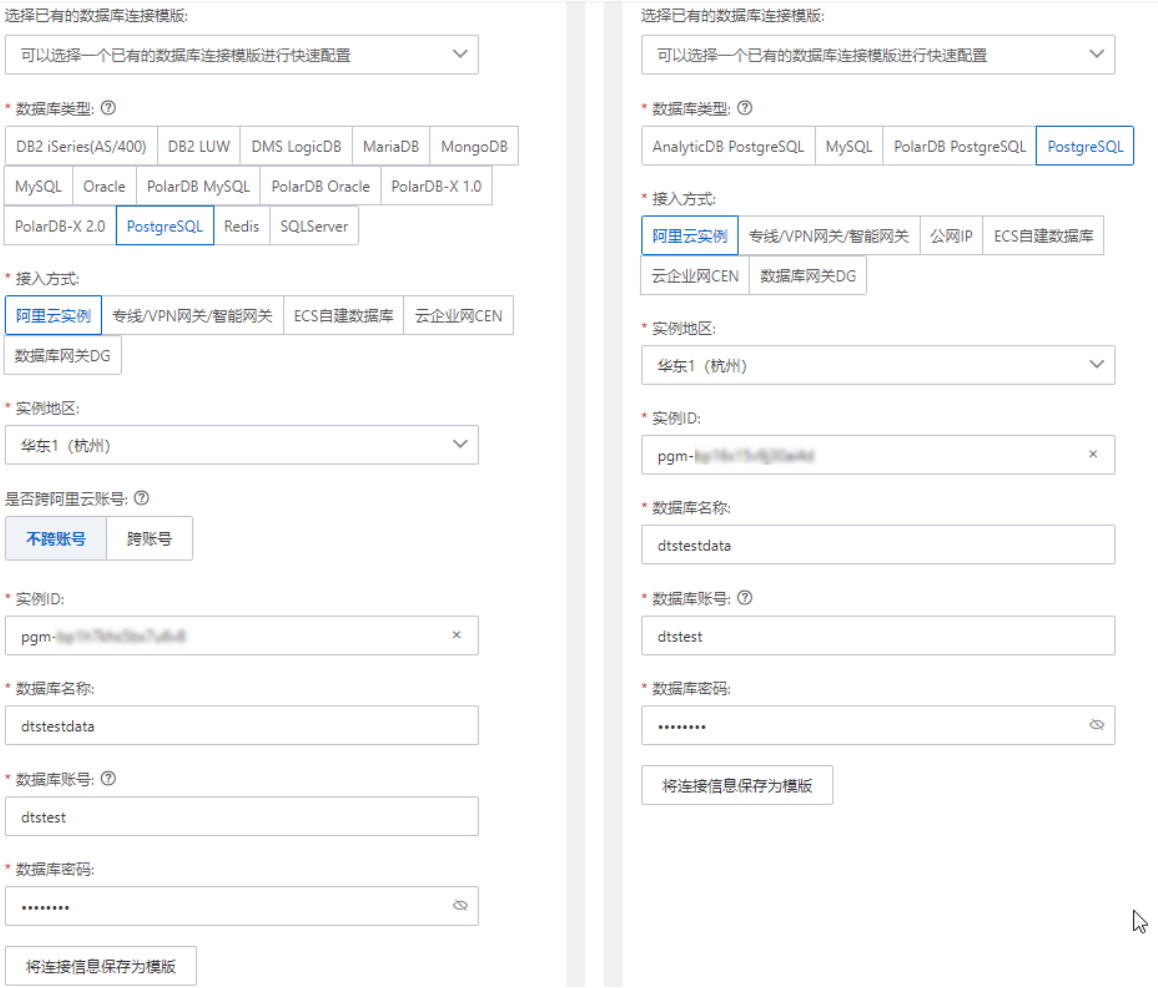
1. 登录新版DTS同步任务的列表页面。

   说明　您也可以登录 DMS数据管理服务。在顶部菜单栏中，选择 **集成与开发（DTS）** > **数据集成** > **数据同步**。

2. 在页面左上角，选择同步实例所属地域。

3. 单击**创建任务**，配置源库及目标库信息。

   警告　选择源和目标实例后，建议您仔细阅读页面上方显示的 **使用限制**，以成功创建并执行同步任务。



类别	配置	说明
无	**任务名称**	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
源库信息		
	数据库类型	选择**PostgreSQL**。
	接入方式	选择**阿里云实例**。
	实例地区	选择源RDS PostgreSQL实例所属地域。
	实例ID	选择源RDS PostgreSQL实例ID。
	数据库名称	填入源RDS PostgreSQL实例中同步对象所属数据库的名称。
	数据库账号	填入源RDS PostgreSQL实例的数据库账号，高权限账号满足要求。 **说明** 当源实例为RDS PostgreSQL 9.4，且仅需同步DML操作时，账号具备REPLICATION权限即可。
	数据库密码	填入该数据库账号对应的密码。

类别	配置	说明
目标库信息		
	数据库类型	选择**PostgreSQL**。
	接入方式	选择**阿里云实例**。
	实例地区	选择目标RDS PostgreSQL实例所属地域。
	实例ID	选择目标RDS PostgreSQL实例ID。
	数据库名称	填入目标RDS PostgreSQL实例中同步对象所属数据库的名称。
	数据库账号	填入目标RDS PostgreSQL实例的数据库账号，高权限账号满足要求。
	数据库密码	填入该数据库账号对应的密码。

4. 配置完成后，单击页面右下角的**测试连接以进行下一步**。

> **说明**
>
> - 如果源或目标数据库是阿里云数据库实例（例如RDS MySQL、云数据库MongoDB版等）或ECS上的自建数据库，DTS会自动将对应地区DTS服务的IP地址添加到阿里云数据库实例的白名单或ECS的安全规则中，您无需手动添加，请参见DTS服务器的IP地址段。
> - DTS任务完成或释放后，建议您手动删除添加的DTS服务器IP地址段。

5. 配置任务对象及高级配置。
   - **基础配置**

配置	说明
任务步骤	固定选中**增量同步**。默认情况下，您还需要同时选中**库表结构同步**和**全量同步**。预检查完成后，DTS会将源实例中待同步对象的全量数据在目标集群中初始化，作为后续增量同步数据的基线数据。

配置	说明
目标已存在表的处理模式	■ **预检查并报错拦截**：检查目标数据库中是否有同名的表。如果目标数据库中没有同名的表，则通过该检查项目；如果目标数据库中有同名的表，则在预检查阶段提示错误，数据同步任务不会被启动。 **说明** 如果目标库中同名的表不方便删除或重命名，您可以更改该表在目标库中的名称，请参见 库表列名映射。 ■ **忽略报错并继续执行**：跳过目标数据库中是否有同名表的检查项。 **警告** 选择为 **忽略报错并继续执行**，可能导致数据不一致，给业务带来风险，例如： ■ 表结构一致的情况下，如在目标库遇到与源库主键的值相同的记录： ■ 全量期间，DTS会保留目标集群中的该条记录，即源库中的该条记录不会同步至目标数据库中。 ■ 增量期间，DTS不会保留目标集群中的该条记录，即源库中的该条记录会覆盖至目标数据库中。 ■ 表结构不一致的情况下，可能会导致无法初始化数据、只能同步部分列的数据或同步失败。
同步拓扑	本场景为单向同步，需选择**单向**。
同步对象	在**源库对象**框中单击待同步对象，然后单击 › 将其移动至**已选择对象**框。 **说明** 同步对象的选择粒度为schema、表或列。
映射名称更改	■ 如需更改单个同步对象在目标实例中的名称，请单击 **已选择对象** 中的同步对象，设置方式，请参见库表列名单个映射。 ■ 如需批量更改同步对象在目标实例中的名称，请单击**已选择对象**方框右上方的**批量编辑**，设置方式，请参见库表列名批量映射。
过滤待同步数据	支持设置WHERE条件过滤数据，请参见通过SQL条件过滤任务数据。
同步的SQL操作	请右击**已选择对象**中的同步对象，在弹跳框中选择所需同步的SQL操作，支持的操作，请参见支持同步的SQL操作。

- **高级配置**

配置	说明
设置告警	是否设置告警，当同步失败或延迟超过阈值后，将通知告警联系人。 ■ **不设置**：不设置告警。 ■ **设置**：设置告警，您还需要设置告警阈值和告警联系人。
源库、目标库无法连接后的重试时间	默认重试120分钟，您也可以在取值范围（10~1440分钟）内自定义重试时间，建议设置30分钟以上。如果DTS在设置的时间内重新连接上源、目标库，同步任务将自动恢复。否则，同步任务将失败。 说明 ■ 针对同源或者同目标的多个DTS实例，如DTS实例A和DTS实例B，设置网络重试时间时A设置30分钟，B设置60分钟，则重试时间以低的30分钟为准。 ■ 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

6. 上述配置完成后，单击页面下方的**下一步保存任务并预检查**。

说明

- 在同步作业正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动同步作业。
- 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
  - 您可以根据提示修复后重新进行预检查。
  - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

7. **预检查通过率**显示为**100%**时，单击**下一步购买**。
8. 在**购买**页面，选择数据同步实例的计费方式、链路规格，详细说明请参见下表。

类别	参数	说明
信息配置	计费方式	○ 预付费（包年包月）：在新建实例时支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。 ○ 后付费（按量付费）：按小时扣费。适合短期需求，用完可立即释放实例，节省费用。
	链路规格	DTS为您提供了不同性能的同步规格，同步链路规格的不同会影响同步速率，您可以根据业务场景进行选择，详情请参见数据同步链路规格说明。
	订购时长	在预付费模式下，选择包年包月实例的时长和数量，包月可选择1~9个月，包年可选择1~3年。 说明 该选项仅在付费类型为 **预付费** 时出现。

9. 配置完成后，阅读并勾选**《数据传输（按量付费）服务条款》**。
10. 单击**购买并启动**，同步任务正式开始，您可在数据同步界面查看具体任务进度。

11.2. 自建PostgreSQL同步至RDS PostgreSQL

本文介绍如何使用数据传输服务DTS（Data Transmission Service），实现自建PostgreSQL同步至RDS PostgreSQL。

前提条件

- 已创建源数据库自建PostgreSQL和目标实例RDS PostgreSQL。目标实例RDS PostgreSQL的创建方式和支持的版本，请参见创建RDS PostgreSQL实例和同步方案概览。

  说明 建议源库和目标数据库的版本相同，或者从低版本同步到高版本以保障兼容性。如为高版本同步至低版本，可能存在数据库兼容性问题。

- 目标实例RDS PostgreSQL的存储空间须大于源数据库自建PostgreSQL占用的存储空间。

注意事项

类型	说明

类型	说明

源库限制	• 待同步的表需具备主键或唯一约束，且字段具有唯一性，否则可能会导致目标数据库中出现重复数据。
	• 如同步对象为表级别，且需进行编辑（如表列名映射），则单次同步任务仅支持同步至多1000张表。当超出数量限制，任务提交后会显示请求报错，此时建议您拆分待同步的表，分批配置多个任务，或者配置整库的同步任务。
	• WAL日志：
	  ○ 需开启，即设置 wal_level 参数的值为*logical*。
	  ○ 如为增量同步任务，DTS要求源数据库的WAL日志保存24小时以上，如为全量同步和增量同步任务，DTS要求源数据库的WAL日志至少保留7天以上（您可在全量同步完成后将WAL日志保存时间设置为24小时以上），否则DTS可能因无法获取WAL日志而导致任务失败，极端情况下甚至可能会导致数据不一致或丢失。由于您所设置的WAL日志保存时间低于DTS要求的时间进而导致的问题，不在DTS的SLA保障范围内。
	• 如自建PostgreSQL进行主备切换，会导致同步失败。

其他限制	• 单个数据同步任务只能同步一个数据库，如需同步多个数据库，您需要为每个数据库配置数据同步任务。
	• 不支持数组（ARRAY）类型的数据同步。
	• 同步期间，如果同步对象的选择粒度为Schema，您需要在对该表写入数据前执行 `ALTER TABLE schema.table REPLICA IDENTITY FULL;` 命令，以保障该表数据同步的一致性。 执行该命令期间，建议您不要有锁表操作，否则会导致表锁死。
	  说明
	  ○ 将上述命令中的 `schema` 和 `table` 替换成真实的Schema名和表名。
	  ○ 建议您在业务低峰期进行操作。
	• DTS的校验对象为数据内容，暂不支持Sequence等元数据的校验，您需要自行校验。
	• 由于业务切换到目标端后，新写入的Sequence不会按照源库的Sequence最大值作为初始值去递增，您需要在业务切换前，在源库中查询对应Sequence的最大值，然后在目标库中将其作为对应Sequence的初始值。查询源库Sequence值的相关命令如下：
	`do language plpgsql $$ declare nsp name; rel name; val int8; begin for nsp,rel in select nspname,relname from pg_class t2 , pg_namespace t3 where t2.relnamespace=t3.oid and t2.relkind='S' loop execute format($_$select last_value from %I.%I$_$, nsp, rel) into val; raise notice '%', format($_$select setval('%I.%I'::regclass, %s);$_$, nsp, rel, val+1); end loop; end; $$;`
	• DTS会在源库中创建以下临时表，用于获取增量数据的DDL、增量表的结构、心跳信息等。在同步期间，请勿删除源库中的临时表，否则会导致DTS任务异常。临时表会在DTS实例释放后自动删除。

类型	说明
	`public.DTS_PG_CLASS` 、 `public.DTS_PG_ATTRIBUTE` 、 `public.DTS_PG_TYPE` 、 `public.DTS_PG_ENUM` 、 `public.DTS_POSTGRES_HEARTBEAT` 、 `public.DTS_DDL_COMMAND` 、 `public.DTS_ARGS_SESSION` 。 • 为保障同步延迟时间展示的准确性，DTS会在源库中新增一个名为*dts_postgres_heartbeat*的心跳表。 • 源为自建PG的源头同步期间，DTS会在源库中创建前缀为 `dts_sync_` 的replication slot用于复制数据。DTS会每隔90分钟自动清理一次历史replication slot，避免其持续累积占用磁盘空间。 说明 当释放同步任务或同步失败时，DTS会主动清理该replication slot；如果RDS PostgreSQL发生了主备切换，则需要您登录备库来手动清理。 • 执行数据同步前需评估源库和目标库的性能，同时建议业务低峰期执行数据同步。否则全量数据初始化时将占用源库和目标库一定的读写资源，可能会导致数据库的负载上升。 • 全量初始化会并发执行INSERT操作，导致目标数据库的表产生碎片，因此全量初始化完成后目标实例的表空间比源实例的表空间大。 • 对于表级别的数据同步，请勿对源库的同步对象使用gh-ost或pt-online-schema-change等类似工具执行在线DDL变更，否则会导致同步失败。 • 对于表级别的数据同步，如果除DTS以外的数据没有写入目标库，您可以使用数据管理DMS（Data Management Service）来执行在线DDL变更，请参见不锁表结构变更。 • 在DTS同步期间，不允许有除DTS外的数据写入目标库，否则会导致源库与目标库数据不一致。例如，有除DTS外的数据写入目标库时，使用DMS执行在线DDL变更，可能引起目标库数据丢失。

支持的同步架构

- 一对一单向同步
- 一对多单向同步
- 级联单向同步
- 多对一单向同步

关于各类同步架构的介绍及注意事项，请参见数据同步拓扑介绍。

支持同步的SQL操作

操作类型	SQL操作语句
DML	INSERT、UPDATE、DELETE

数据库账号的权限要求

数据库	所需权限	创建账号及授权方式
自建PostgreSQL数据库	superuser	CREATE USER和GRANT语法。
RDS PostgreSQL实例	schema的owner权限	RDS PostgreSQL实例请参见创建账号。

准备工作

如自建PostgreSQL版本为9.4.8~10.0，则您需执行如下准备工作。

1. 从官网下载PostgreSQL源码并完成编译安装。
   i. 根据源库自建PostgreSQL的版本，从PostgreSQL官网下载对应版本的源码。
   ii. 依次执行命令 `./configure` 、 `make` 、 `make install` ，配置、编译并安装源码。

> **注意**
>
> - 编译安装PostgreSQL时，PostgreSQL的操作系统版本需与GCC（GNU编译器套件）版本保持一致。
> - 如执行 `./configure` 遇到报错，您可以根据报错提示调整命令。例如，报错信息为 `readline library not found. Use --without-readline to disable readline suppo rt.` ，则您可以调整命令为 `./configure --without-readline` .
> - 如您选择其他方式安装PostgreSQL，需要在系统版本以及GCC版本相同的测试环境下，编译ali_decoding。

2. 下载DTS提供的插件ali_decoding并完成编译安装。
   i. 下载ali_decoding。
   ii. 将ali_decoding整个目录拷贝至PostgreSQL（已完成编译安装）的contrib目录中。

```
-rw-r--r--  1 1107 1107    384 9月  27 2016 aclocal.m4
drwxrwxrwx  2 1107 1107   4096 9月  27 2016 config
-rw-r--r--  1 root root 374806 9月   7 10:10 config.log
-rwxr-xr-x  1 root root  39032 9月   7 10:10 config.status
-rwxr-xr-x  1 1107 1107 471157 9月  27 2016 configure
-rw-r--r--  1 1107 1107  75195 9月  27 2016 configure.in
drwxrwxrwx 56 1107 1107   4096 9月   7 10:28 contrib
-rw-r--r--  1 1107 1107   1192 9月  27 2016 COPYRIGHT
drwxrwxrwx  3 1107 1107   4096 9月  27 2016 doc
-rw-r--r--  1 root root   3638 9月   7 10:10 GNUmakefile
-rw-r--r--  1 1107 1107   3638 9月  27 2016 GNUmakefile.in
-rw-r--r--  1 1107 1107    283 9月  27 2016 HISTORY
-rw-r--r--  1 1107 1107  75065 9月  27 2016 INSTALL
-rw-r--r--  1 1107 1107   1489 9月  27 2016 Makefile
-rw-r--r--  1 1107 1107   1209 9月  27 2016 README
drwxrwxrwx 16 1107 1107   4096 9月   7 10:10 src
```

   iii. 进入ali_decoding目录，将Makefile文件的内容替换为如下：

```
# contrib/ali_decoding/Makefile MODULE_big = ali_decoding MODULES = ali_decoding OB
JS = ali_decoding.o DATA = ali_decoding--0.0.1.sql ali_decoding--unpackaged--0.0.1.
sql EXTENSION = ali_decoding NAME = ali_decoding #subdir = contrib/ali_decoding #to
p_builddir = ../.. #include $(top_builddir)/src/Makefile.global #include $(top_srcd
ir)/contrib/contrib-global.mk #PG_CONFIG = /usr/pgsql-9.6/bin/pg_config #pgsql_lib_
dir := $(shell $(PG_CONFIG) --libdir) #PGXS := $(shell $(PG_CONFIG) --pgxs) #includ
e $(PGXS) # 源码安装用以下 ifdef USE_PGXS PG_CONFIG = pg_config PGXS := $(shell $(PG_
CONFIG) --pgxs) include $(PGXS) else subdir = contrib/ali_decoding top_builddir = .
./.. include $(top_builddir)/src/Makefile.global include $(top_srcdir)/contrib/cont
rib-global.mk endif
```

iv. 进入ali_decoding目录，依次执行命令 `make` 、 `make install` ，编译ali_decoding，并得到安装ali_decoding所需的文件。

v. 将如下文件复制至指定位置。

```
/usr/bin/install -c -m 755  ali_decoding.so '/usr/local/pgsql/lib/ali_decoding.so'
/usr/bin/install -c -m 644 ./ali_decoding.control '/usr/local/pgsql/share/extension/'
/usr/bin/install -c -m 644 ./ali_decoding--0.0.1.sql ./ali_decoding--unpackaged--0.0.1.sql  '/usr/local/pgsql/share/extension/'
/usr/bin/install -c -m 755  ali_decoding.so '/usr/local/pgsql/lib/'
```

3. 根据待同步对象所属的数据库和schema信息，在目标RDS PostgreSQL中创建相应数据库和schema（schema名称须一致），详情请参见创建数据库和Schema 管理。

如自建PostgreSQL版本为10.1~13，则您需执行如下准备工作。

1. 登录自建PostgreSQL所属的服务器。
2. 修改配置文件*postgresql.conf*，将配置文件中的 `wal_level` 设置为 `logical` 。

```
# - Settings -

wal_level = logical                     # minimal, replica, or logical
                                        # (change requires restart)
```

3. 将DTS的IP地址加入至自建PostgreSQL的配置文件*pg_hba.conf*中。您只需添加目标数据库所在区域对应的DTS IP地址段，详情请参见迁移、同步或订阅本地数据库时需添加的IP白名单。

> **说明** 关于该配置文件的设置请参见 pg_hba.conf文件。如果您已将信任地址配置为 `0.0.0.0/0` （如下图所示），可跳过本步骤。

```
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             0.0.0.0/0               trust
# IPv6 local connections:
host    all             all             ::1/128                 md5
```

4. 根据待同步对象所属的数据库和schema信息，在目标RDS PostgreSQL中创建相应数据库和schema（schema名称须一致），详情请参见创建数据库和Schema 管理。

操作步骤

1. 登录新版DTS同步任务的列表页面。

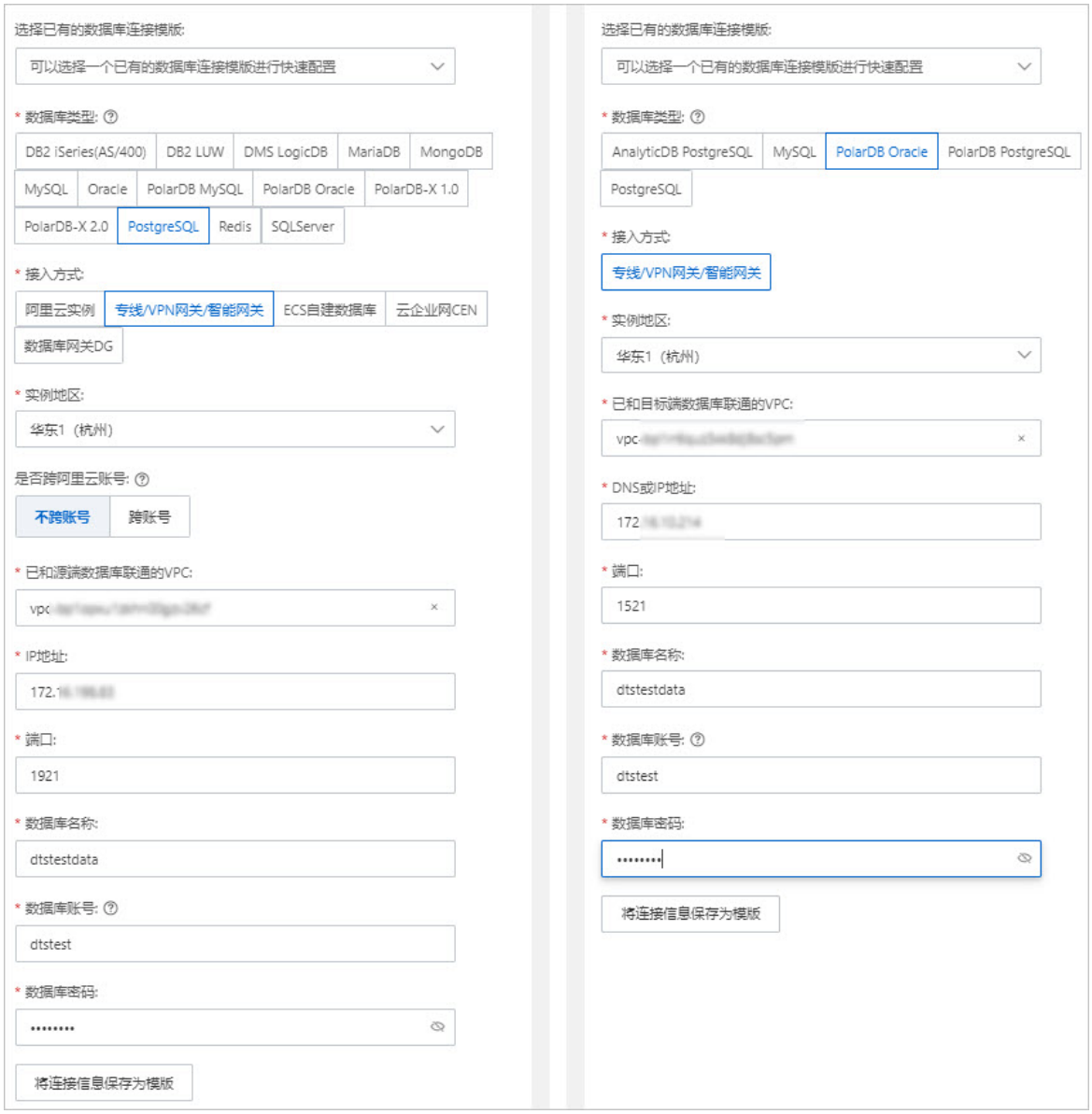
> **说明** 您也可以登录 DMS数据管理服务。在顶部菜单栏中，选择 **集成与开发（DTS）** > **数据集成** > **数据同步**。

2. 在页面左上角，选择同步实例所属地域。

3. 单击**创建任务**，配置源库及目标库信息。

> **警告** 选择源和目标实例后，建议您仔细阅读页面上方显示的**使用限制**，以成功创建并执行同步任务。

待补充



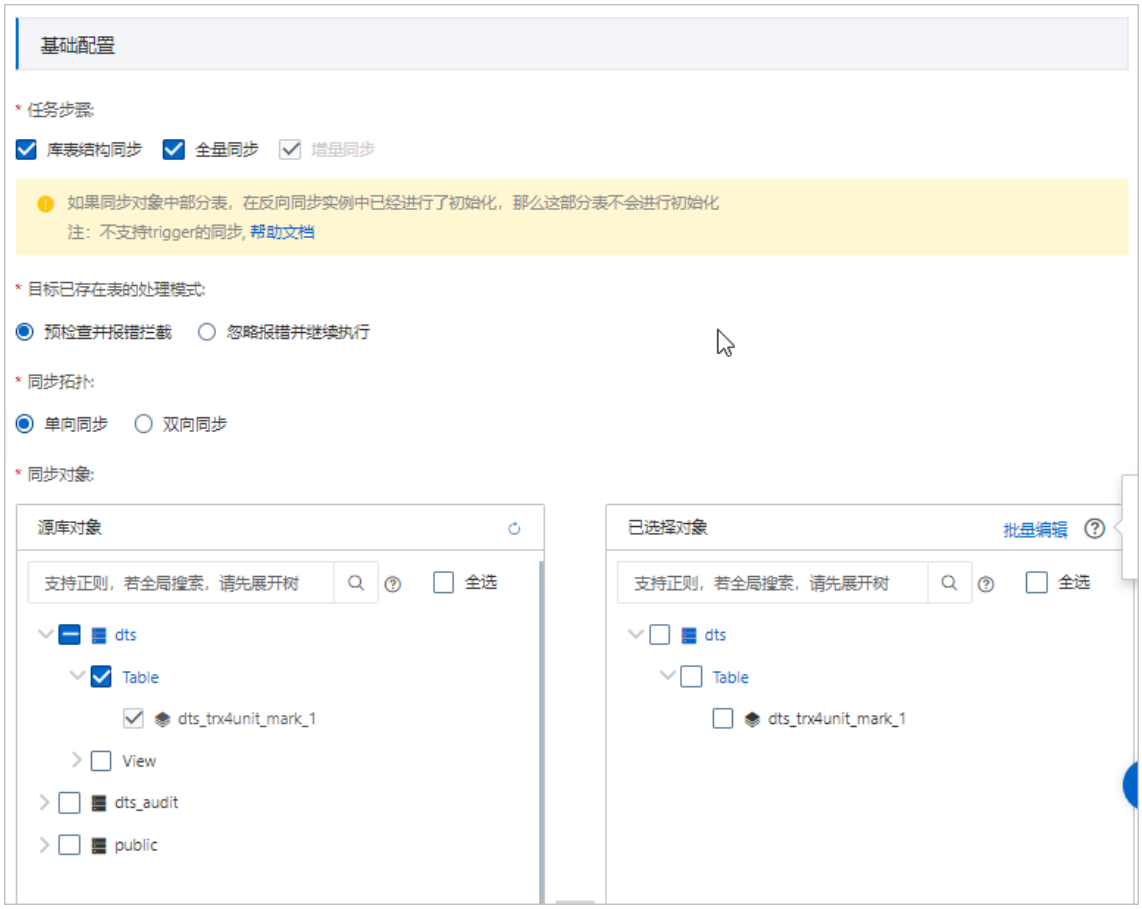
类别	配置	说明
无	**任务名称**	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
	数据库类型	选择PostgreSQL。
	接入方式	选择**云企业网CEN**。
	实例地区	选择自建PostgreSQL数据库所属地域。

类别	配置	说明
源库信息	**云企业网实例ID**	选择自建PostgreSQL数据库所属云企业网的实例ID。
	已和源库互联的VPC网络	选择与自建PostgreSQL数据库互联的VPC网络。
	IP地址	填入自建PostgreSQL数据库的服务器IP地址。
	端口	填入自建PostgreSQL数据库提供服务的端口，默认为5432。
	数据库名称	填入自建Postg reSQL中同步 对象所属数据库的名称。
	数据库账号	填入自建PostgreSQL的数据库账号，权限要求，请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号对应的密码。
目标库信息		
	数据库类型	选择**PostgreSQL**。
	接入方式	选择**阿里云实例**。
	实例地区	选择目标RDS PostgreSQL实例所属地域。
	实例ID	选择目标RDS PostgreSQL实例ID。
	数据库名称	填入目标RDS PostgreSQL实例中同步对象所属数据库的名称。
	数据库账号	填入目标RDS PostgreSQL实例的数据库账号，权限要求，请参见数据库账号的权限要求。
	数据库密码	填入该数据库账号对应的密码。

4. 配置完成后，单击页面右下角的**测试连接以进行下一步**。

> **说明**
>
> - 如果源或目标数据库是阿里云数据库实例（例如RDS MySQL、云数据库MongoDB版等）或ECS上的自建数据库，DTS会自动将对应地区DTS服务的IP地址添加到阿里云数据库实例的白名单或ECS的安全规则中，您无需手动添加，请参见DTS服务器的IP地址段。
> - DTS任务完成或释放后，建议您手动删除添加的DTS服务器IP地址段。

5. 配置任务对象及高级配置。
   - **基础配置**



配置	说明
任务步骤	固定选中**增量同步**。默认情况下，您还需要同时选中**库表结构同步**和**全量同步**。预检查完成后，DTS会将源实例中待同步对象的全量数据在目标集群中初始化，作为后续增量同步数据的基线数据。

配置	说明
目标已存在表的处理模式	■ **预检查并报错拦截**：检查目标数据库中是否有同名的表。如果目标数据库中没有同名的表，则通过该检查项目；如果目标数据库中有同名的表，则在预检查阶段提示错误，数据同步任务不会被启动。 **说明** 如果目标库中同名的表不方便删除或重命名，您可以更改该表在目标库中的名称，请参见 库表列名映射。 ■ **忽略报错并继续执行**：跳过目标数据库中是否有同名表的检查项。 **警告** 选择为 **忽略报错并继续执行**，可能导致数据不一致，给业务带来风险，例如： ■ 表结构一致的情况下，如在目标库遇到与源库主键的值相同的记录： ■ 全量期间，DTS会保留目标集群中的该条记录，即源库中的该条记录不会同步至目标数据库中。 ■ 增量期间，DTS不会保留目标集群中的该条记录，即源库中的该条记录会覆盖至目标数据库中。 ■ 表结构不一致的情况下，可能会导致无法初始化数据、只能同步部分列的数据或同步失败。
同步拓扑	本场景为单向同步，需选择**单向**。
同步对象	在**源库对象**框中单击待同步对象，然后单击 › 将其移动至**已选择对象**框。 **说明** 同步对象的选择粒度为schema、表或列。
映射名称更改	■ 如需更改单个同步对象在目标实例中的名称，请单击 **已选择对象**中的同步对象，设置方式，请参见库表列名单个映射。 ■ 如需批量更改同步对象在目标实例中的名称，请单击**已选择对象**方框右上方的**批量编辑**，设置方式，请参见库表列名批量映射。
过滤待同步数据	支持设置WHERE条件过滤数据，请参见通过SQL条件过滤任务数据。
同步的SQL操作	请右击**已选择对象**中的同步对象，在弹跳框中选择所需同步的SQL操作，支持的操作，请参见支持同步的SQL操作。

- **高级配置**

配置	说明
设置告警	是否设置告警，当同步失败或延迟超过阈值后，将通知告警联系人。 ■ **不设置**：不设置告警。 ■ **设置**：设置告警，您还需要设置告警阈值和告警联系人。
目标库对象名称大小写策略	您可以配置目标实例中迁移对象的库名、表名和列名的英文大小写策略。默认情况下选择**DTS默认策略**，您也可以选择与源库、目标库默认策略保持一致。更多信息，请参见目标库对象名称大小写策略。
源库、目标库无法连接后的重试时间	默认重试120分钟，您也可以在取值范围（10~1440分钟）内自定义重试时间，建议设置30分钟以上。如果DTS在设置的时间内重新连接上源、目标库，同步任务将自动恢复。否则，同步任务将失败。 **说明** ■ 针对同源或者同目标的多个DTS实例，如DTS实例A和DTS实例B，设置网络重试时间时A设置30分钟，B设置60分钟，则重试时间以低的30分钟为准。 ■ 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

6. 上述配置完成后，单击页面下方的**下一步保存任务并预检查**。

> **说明**
> - 在同步作业正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动同步作业。
> - 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
> - 您可以根据提示修复后重新进行预检查。
> - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

7. **预检查通过率**显示为**100%**时，单击**下一步购买**。
8. 在**购买**页面，选择数据同步实例的计费方式、链路规格，详细说明请参见下表。

类别	参数	说明
信息配置	计费方式	○ 预付费（包年包月）：在新建实例时支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。 ○ 后付费（按量付费）：按小时扣费。适合短期需求，用完可立即释放实例，节省费用。
	链路规格	DTS为您提供了不同性能的同步规格，同步链路规格的不同会影响同步速率，您可以根据业务场景进行选择，详情请参见数据同步链路规格说明。
	订购时长	在预付费模式下，选择包年包月实例的时长和数量，包月可选择1~9个月，包年可选择1~3年。 **说明** 该选项仅在付费类型为 **预付费** 时出现。

9. 配置完成后，阅读并勾选**《数据传输（按量付费）服务条款》**。
10. 单击**购买并启动**，同步任务正式开始，您可在数据同步界面查看具体任务进度。

11.3. RDS PostgreSQL同步至RDS MySQL

本文介绍如何使用数据传输服务DTS（Data Transmission Service），将RDS PostgreSQL同步至RDS MySQL。

前提条件

- 已创建源RDS PostgreSQL实例，详情请参见创建RDS PostgreSQL实例。
- 已创建目标RDS MySQL实例，创建方式，请参见创建RDS MySQL实例。
- 目标RDS MySQL实例的存储空间须大于源RDS PostgreSQL实例占用的存储空间。

注意事项

类型	说明

类型	说明

源库限制	• 待同步的表需具备主键或唯一约束，且字段具有唯一性，否则可能会导致目标数据库中出现重复数据。
	• 如同步对象为表级别，且需进行编辑（如表列名映射），则单次同步任务仅支持同步至多1000张表。当超出数量限制，任务提交后会显示请求报错，此时建议您拆分待同步的表，分批配置多个任务，或者配置整库的同步任务。
	• WAL日志：
	  ○ 需开启，即设置 wal_level 参数的值为*logical*。
	  ○ 如为增量同步任务，DTS要求源数据库的WAL日志保存24小时以上，如为全量同步和增量同步任务，DTS要求源数据库的WAL日志至少保留7天以上（您可在全量同步完成后将WAL日志保存时间设置为24小时以上），否则DTS可能因无法获取WAL日志而导致任务失败，极端情况下甚至可能会导致数据不一致或丢失。由于您所设置的WAL日志保存时间低于DTS要求的时间进而导致的问题，不在DTS的SLA保障范围内。
	• 源库的操作限制
	为保障同步任务的正常进行，避免主备切换导致的逻辑订阅中断，需要RDS PostgreSQL支持并开启Logical Replication Slot Failover，设置方式，请参见逻辑复制槽故障转移（Logical Replication Slot Failover）。

类型	说明
其他限制	• 单个数据同步任务只能同步一个数据库，如需同步多个数据库，您需要为每个数据库配置数据同步任务。 • 同步期间，如果同步对象的选择粒度为Schema，您需要在对该表写入数据前执行 `ALTER TABLE schema.table REPLICA IDENTITY FULL;` 命令，以保障该表数据同步的一致性。 执行该命令期间，建议您不要有锁表操作，否则会导致表锁死。 说明 ○ 将上述命令中的 `schema` 和 `table` 替换成真实的Schema名和表名。 ○ 建议您在业务低峰期进行操作。 • DTS会在源库中创建以下临时表，用于获取增量数据的DDL、增量表的结构、心跳信息等。在同步期间，请勿删除源库中的临时表，否则会导致DTS任务异常。临时表会在DTS实例释放后自动删除。 `public.DTS_PG_CLASS` 、 `public.DTS_PG_ATTRIBUTE` 、 `public.DTS_PG_TYPE` 、 `public.DTS_PG_ENUM` 、 `public.DTS_POSTGRES_HEARTBEAT` 、 `public.DTS_DDL_COMMAND` 、 `public.DTS_ARGS_SESSION` 。 • 为保障同步延迟时间展示的准确性，DTS会在源库中新增一个名为*dts_postgres_heartbeat*的心跳表。 • 同步期间，DTS会在源库中创建前缀为 `dts_sync_` 的replication slot用于复制数据。DTS会每隔90分钟自动清理一次历史replication slot，避免其持续累积占用磁盘空间。 说明 ○ DTS释放实例后会主动删除该replication slot，如果您在同步期间修改了数据库密码，或者删除了DTS的访问IP白名单，则会导致该replication slot无法自动删除，此时需要您在源库手动删除，避免其持续累积占用磁盘空间导致RDS PostgreSQL实例不可用。 ○ 当释放同步任务或同步失败时，DTS会主动清理该replication slot；如果RDS PostgreSQL发生了主备切换，则需要您登录备库来手动清理。 • 执行数据同步前需评估源库和目标库的性能，同时建议业务低峰期执行数据同步。否则全量数据初始化时将占用源库和目标库一定的读写资源，可能会导致数据库的负载上升。 • 全量初始化会并发执行INSERT操作，导致目标数据库的表产生碎片，因此全量初始化完成后目标实例的表空间比源实例的表空间大。 • 对于表级别的数据同步，请勿对源库的同步对象使用gh-ost或pt-online-schema-change等类似工具执行在线DDL变更，否则会导致同步失败。 • 对于表级别的数据同步，如果除DTS以外的数据没有写入目标库，您可以使用数据管理DMS（Data Management Service）来执行在线DDL变更，请参见不锁表结构变更。 • 在DTS同步期间，不允许有除DTS外的数据写入目标库，否则会导致源库与目标库数据不一致。例如，有除DTS外的数据写入目标库时，使用DMS执行在线DDL变更，可能引起目标库数据丢失。

类型	说明
特殊情况	• 当源实例为RDS PostgreSQL时 同步期间，请勿修改RDS PostgreSQL的连接地址和可用区，否则会导致同步失败。

支持的同步架构

- 一对一单向同步
- 一对多单向同步
- 级联单向同步
- 多对一单向同步

关于各类同步架构的介绍及注意事项，请参见 数据同步拓扑介绍。

支持同步的SQL操作

操作类型	SQL操作语句
DML	INSERT、UPDATE、DELETE

操作步骤

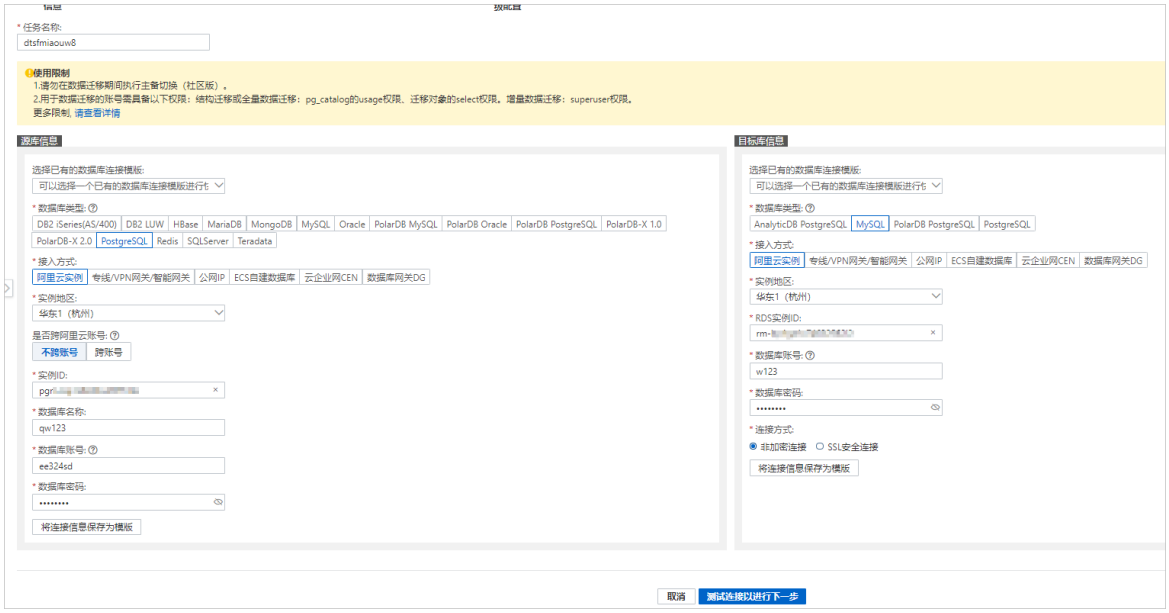
1. 登录新版DTS同步任务的列表页面。

   说明　您也可以登录 DMS数据管理服务。在顶部菜单栏中，选择 **集成与开发（DTS）** > **数据集成** > **数据同步**。

2. 在页面左上角，选择同步实例所属地域。

3. 单击**创建任务**，配置源库及目标库信息。

   警告　选择源和目标实例后，建议您仔细阅读页面上方显示的 **使用限制**，以成功创建并执行同步任务。



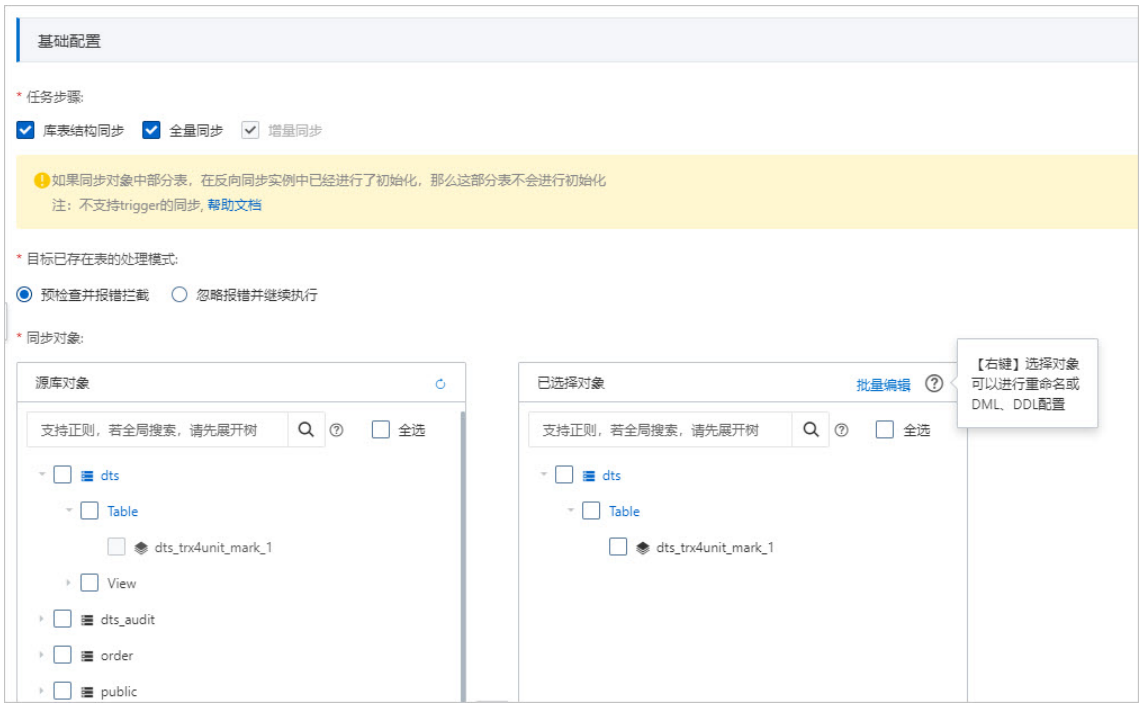
类别	配置	说明
无	**任务名称**	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
源库信息	**数据库类型**	选择**PostgreSQL**。
	接入方式	选择**阿里云实例**。
	实例地区	选择源RDS PostgreSQL实例所属地域。
	实例ID	选择源RDS PostgreSQL实例ID。
	数据库名称	填入源RDS PostgreSQL实例中同步对象所属数据库的名称。
	数据库账号	填入源RDS PostgreSQL实例的数据库账号，高权限账号满足要求。 **说明** 当源实例为RDS PostgreSQL 9.4，且仅需同步DML操作时，账号具备REPLICATION权限即可。
	数据库密码	填入该数据库账号对应的密码。
目标库信息	**数据库类型**	选择**MySQL**。
	接入方式	选择**阿里云实例**。
	实例地区	选择目标RDS MySQL实例所属地域。
	RDS实例ID	选择目标RDS MySQL实例ID。
	数据库账号	填入目标RDS MySQL实例的数据库账号，需具备读写权限。
	数据库密码	填入该数据库账号对应的密码。

类别	配置	说明
	连接方式	rdsmysql连接方式根据需求选择**非加密连接**或**SSL安全连接**。如果设置为**SSL安全连接**，您需要提前开启RDS MySQL实例的SSL加密功能，详情请参见设置SSL加密。

4. 配置完成后，单击页面右下角的**测试连接以进行下一步**。

> **说明**
>
> - 如果源或目标数据库是阿里云数据库实例（例如RDS MySQL、云数据库MongoDB版等）或ECS上的自建数据库，DTS会自动将对应地区DTS服务的IP地址添加到阿里云数据库实例的白名单或ECS的安全规则中，您无需手动添加，请参见DTS服务器的IP地址段。
> - DTS任务完成或释放后，建议您手动删除添加的DTS服务器IP地址段。

5. 配置任务对象及高级配置。
   - **基础配置**



配置	说明
任务步骤	固定选中**增量同步**。默认情况下，您还需要同时选中**库表结构同步**和**全量同步**。预检查完成后，DTS会将源实例中待同步对象的全量数据在目标集群中初始化，作为后续增量同步数据的基线数据。

配置	说明
目标已存在表的处理模式	■ **预检查并报错拦截**：检查目标数据库中是否有同名的表。如果目标数据库中没有同名的表，则通过该检查项目；如果目标数据库中有同名的表，则在预检查阶段提示错误，数据同步任务不会被启动。 **说明** 如果目标库中同名的表不方便删除或重命名，您可以更改该表在目标库中的名称，请参见 库表列名映射。 ■ **忽略报错并继续执行**：跳过目标数据库中是否有同名表的检查项。 **警告** 选择为 **忽略报错并继续执行**，可能导致数据不一致，给业务带来风险，例如： ■ 表结构一致的情况下，如在目标库遇到与源库主键的值相同的记录： ■ 全量期间，DTS会保留目标集群中的该条记录，即源库中的该条记录不会同步至目标数据库中。 ■ 增量期间，DTS不会保留目标集群中的该条记录，即源库中的该条记录会覆盖至目标数据库中。 ■ 表结构不一致的情况下，可能会导致无法初始化数据、只能同步部分列的数据或同步失败。
同步对象	在**源库对象**框中单击待同步对象，然后单击 › 将其移动至**已选择对象**框。 **说明** 同步对象选择的粒度为库、表、列。若选择的同步对象为表或列，其他对象（如视图、触发器、存储过程）不会被同步至目标库。
映射名称更改	■ 如需更改单个同步对象在目标实例中的名称，请单击 **已选择对象** 中的同步对象，设置方式，请参见库表列名单个映射。 ■ 如需批量更改同步对象在目标实例中的名称，请单击**已选择对象**方框右上方的**批量编辑**，设置方式，请参见库表列名批量映射。
过滤待同步数据	支持设置WHERE条件过滤数据，请参见通过SQL条件过滤任务数据。
同步的SQL操作	请右击**已选择对象**中的同步对象，在弹跳框中选择所需同步的DML操作，支持的操作，请参见支持同步的SQL操作。

- **高级配置**

配置	说明
设置告警	是否设置告警，当同步失败或延迟超过阈值后，将通知告警联系人。 ■ **不设置**：不设置告警。 ■ **设置**：设置告警，您还需要设置告警阈值和告警联系人。
源库、目标库无法连接后的重试时间	默认重试120分钟，您也可以在取值范围（10~1440分钟）内自定义重试时间，建议设置30分钟以上。如果DTS在设置的时间内重新连接上源、目标库，同步任务将自动恢复。否则，同步任务将失败。 说明 ■ 针对同源或者同目标的多个DTS实例，如DTS实例A和DTS实例B，设置网络重试时间时A设置30分钟，B设置60分钟，则重试时间以低的30分钟为准。 ■ 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

6. 上述配置完成后，单击页面下方的**下一步保存任务并预检查**。

说明

- 在同步作业正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动同步作业。
- 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
  - 您可以根据提示修复后重新进行预检查。
  - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

7. **预检查通过率**显示为**100%**时，单击**下一步购买**。
8. 在**购买**页面，选择数据同步实例的计费方式、链路规格，详细说明请参见下表。

类别	参数	说明
信息配置	计费方式	○ 预付费（包年包月）：在新建实例时支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。 ○ 后付费（按量付费）：按小时扣费。适合短期需求，用完可立即释放实例，节省费用。
	链路规格	DTS为您提供了不同性能的同步规格，同步链路规格的不同会影响同步速率，您可以根据业务场景进行选择，详情请参见数据同步链路规格说明。
	订购时长	在预付费模式下，选择包年包月实例的时长和数量，包月可选择1~9个月，包年可选择1~3年。 说明 该选项仅在付费类型为 **预付费** 时出现。

9. 配置完成后，阅读并勾选**《数据传输（按量付费）服务条款》**。
10. 单击**购买并启动**，同步任务正式开始，您可在数据同步界面查看具体任务进度。

11.4. 从RDS PostgreSQL同步至云原生数据仓库AnalyticDB PostgreSQL

数据传输服务DTS（Data Transmission Service）支持将RDS PostgreSQL同步至云原生数据仓库AnalyticDB PostgreSQL。通过DTS提供的数据同步功能，可以轻松实现数据的流转，将企业数据集中分析。

前提条件

- RDS PostgreSQL中待同步的数据表必须具备主键。
- 已创建目标云原生数据仓库AnalyticDB PostgreSQL实例，如未创建请参见创建云原生数据仓库AnalyticDB PostgreSQL实例。

注意事项

- 一个数据同步作业只能同步一个数据库，如果有多个数据库需要同步，则需要为每个数据库创建数据同步作业。
- 在数据同步的过程中，如果要将源库中创建的新表作为同步对象，您需要对该表执行如下操作以保障该表数据同步的一致性。

```
ALTER TABLE schema.table REPLICA IDENTITY FULL;
```

- 为保障同步任务的正常进行，目前仅支持RDS PostgreSQL 11进行主备切换，且需设置参数 `rds_failover_slot_mode` 为 `sync` ，设置方式，请参见逻辑复制槽故障转移（Logical Replication Slot Failover）。

  警告 自建PostgreSQL和其他版本的RDS PostgreSQL进行主备切换，会导致同步中断。

同步限制

- 不支持结构初始化，即不支持将源库中待同步对象的结构定义（例如表结构）同步至目标库中。

- 同步对象仅支持数据表。
- 不支持BIT、VARBIT、GEOMETRY、ARRAY、UUID、TSQUERY、TSVECTOR、TXID_SNAPSHOT类型的数据同步。
- 同步过程中，如果对源库中的同步对象执行了DDL操作，需要手动在目标库中执行对应的DDL操作，然后重启数据同步作业。

支持的同步语法

仅支持INSERT、UPDATE、DELETE。

准备工作

1. 调整源RDS实例的 `wal_level` 参数设置。

   警告 修改 `wal_level` 参数后需要重启实例才能生效，请评估对业务的影响，在业务低峰期进行修改。

   i. 登录RDS管理控制台。
   ii. 在页面左上角，选择实例所在地域。
   iii. 找到目标实例，单击实例ID。
   iv. 在左侧导航栏，单击**参数设置**。
   v. 在**参数设置**页面找到 `wal_level` 参数，将参数值改为 `logical` 。

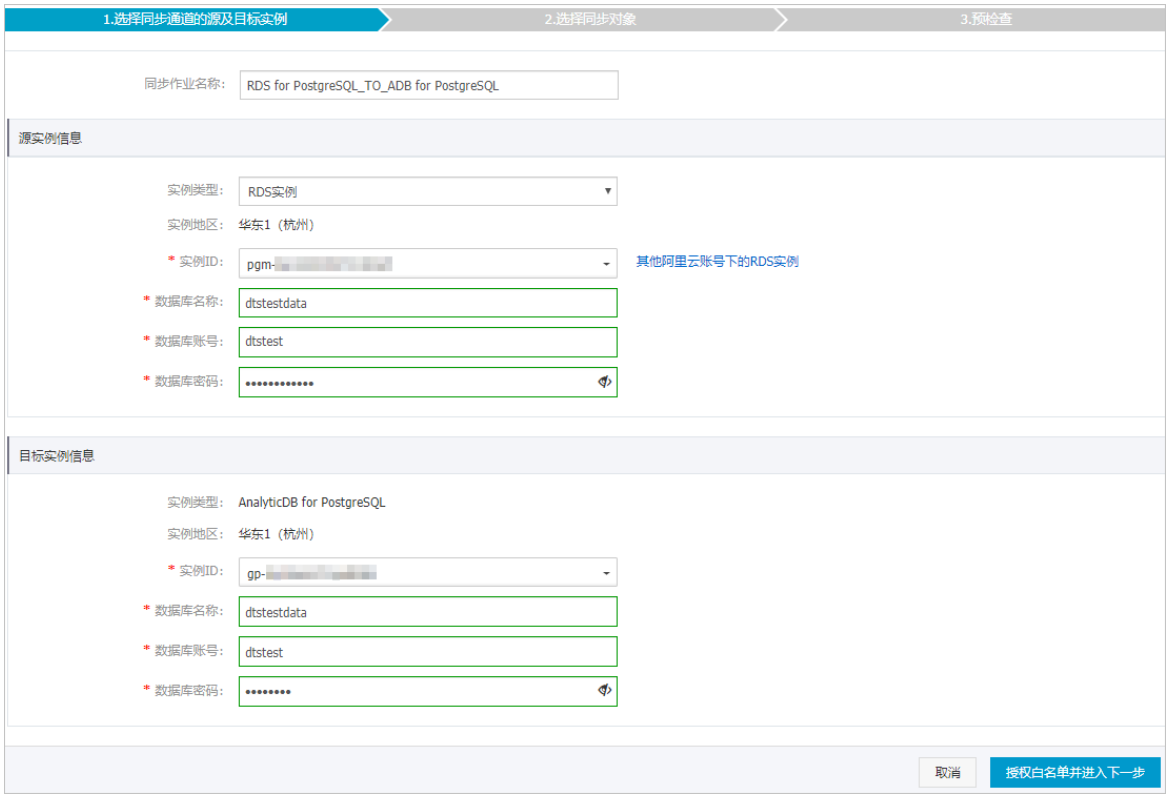
2. 根据源RDS实例中待同步对象的结构，在目标云原生数据仓库AnalyticDB PostgreSQL中创建相应的数据库、Schema、表等结构信息，详情请参见SQL语法。

操作步骤

1. 购买数据同步作业，详情请参见购买流程。

   说明 购买时，选择源实例为 PostgreSQL、目标实例为 AnalyticDB for PostgreSQL，并选择同步拓扑为 **单向同步**。

2. 登录数据传输控制台。
3. 在左侧导航栏，单击**数据同步**。
4. 在**同步作业列表**页面顶部，选择同步的目标实例所属地域。
5. 定位至已购买的数据同步实例，单击**配置同步链路**。
6. 配置同步作业的源实例及目标实例信息。



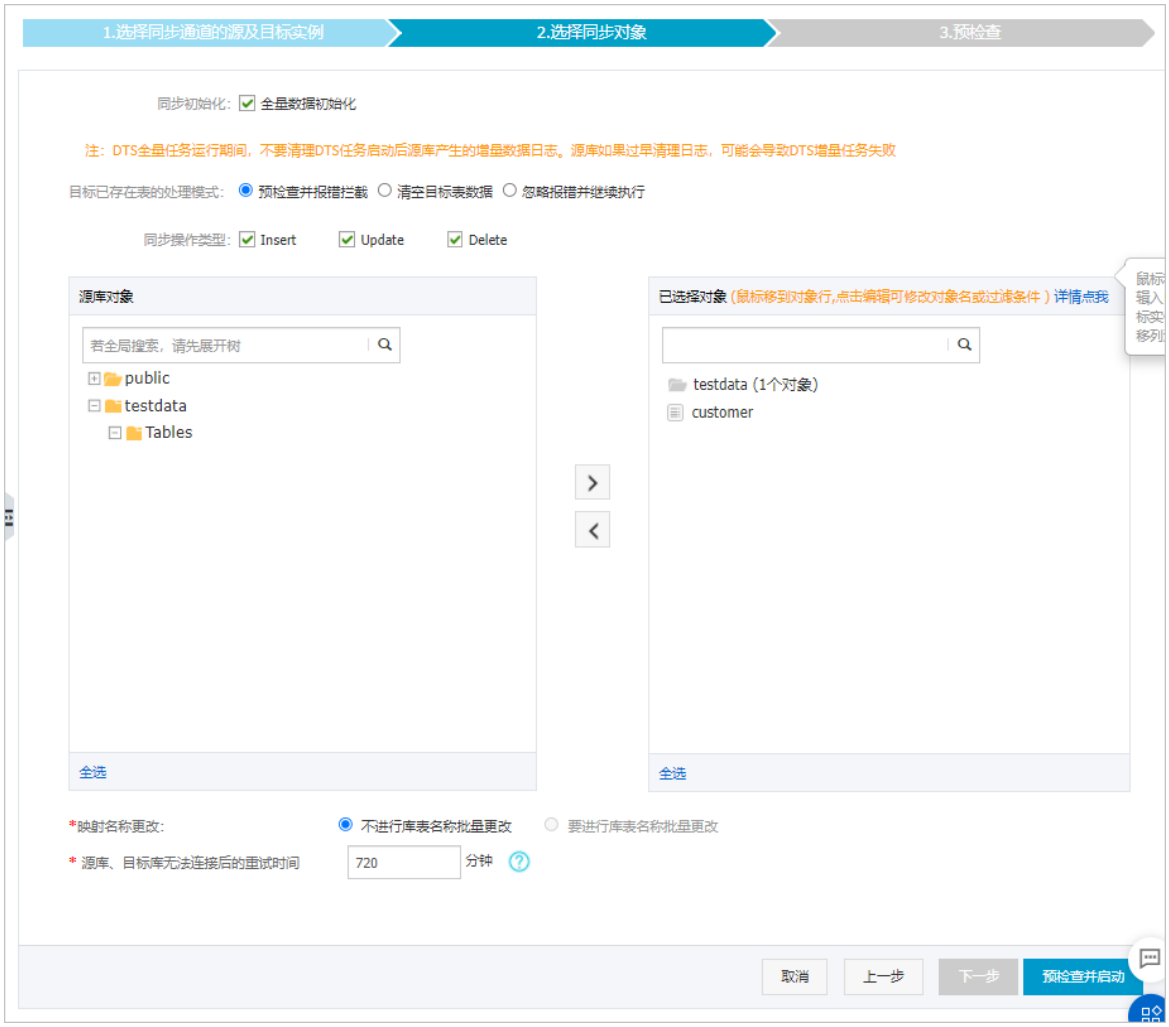
类别	配置	说明
无	同步作业名称	DTS会自动生成一个同步作业名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
源实例信息	实例类型	选择**RDS实例**。
	实例地区	购买数据同步实例时选择的源实例地域信息，不可变更。
	实例ID	选择RDS PostgreSQL实例ID。
	数据库名称	填入待同步的表所属的数据库名称。
	数据库账号	填入RDS PostgreSQL的数据库账号，需具备superuser权限。 **说明** 当源库为RDS PostgreSQL 9.4，且仅需同步DML操作时，账号具备replication权限即可。
	数据库密码	填入该数据库账号对应的密码。
	实例类型	固定为**AnalyticDB for PostgreSQL**，无需设置。
	实例地区	购买数据同步实例时选择的目标实例地域信息，不可变更。
	实例ID	选择云原生数据仓库AnalyticDB PostgreSQL实例ID。

类别	配置	说明
目标实例信息	数据库名称	填入同步目标表所属的数据库名称。 说明 该库须在 云原生数据仓库AnalyticDB PostgreSQL中存在，如不存在请 创建数据库。
	数据库账号	填入云原生数据仓库AnalyticDB PostgreSQL的**初始账号**，详情请参见创建数据库账号。 说明 您也可以填入具备RDS_SUPERUSER权限的账号，创建方法请参见 用户权限管理。
	数据库密码	填入该数据库账号对应的密码。

7. 单击页面右下角的**授权白名单并进入下一步**。

   说明 此步骤会将DTS服务器的IP地址自动添加到RDS PostgreSQL和 云原生数据仓库AnalyticDB PostgreSQL的白名单中，用于保障DTS服务器能够正常连接源和目标实例。

8. 配置同步策略及对象信息。



类别	配置	说明
同步策略配置	同步初始化	默认情况下，您需要勾选**全量数据初始化**。预检查完成后，DTS会将源实例中待同步对象的存量数据同步至目标实例，作为后续增量同步数据的基线数据。
	目标已存在表的处理模式	○ **清空目标表的数据** 在预检查阶段跳过**目标表是否为空**的检查项目。全量初始化之前将目标表的数据清空。适用于完成同步任务测试后的正式同步场景。 ○ **忽略报错并继续执行** 在预检查阶段跳过**目标表是否为空**的检查项目。全量初始化时直接追加迁移数据。适用于多张表同步到一张表的汇总同步场景。

类别	配置	说明
	同步操作类型	根据业务需求选择需要同步的操作类型： 说明 不支持 AlterTable。 ○ Insert ○ Update ○ Delete ○ AlterTable
选择同步对象	无	在**源库对象**框中单击待同步的表，然后单击 › 将其移动至**已选择对象**框。 说明 ○ 同步对象的选择粒度为表。 ○ 如果需要目标表中的列名称与源表不同，则需要使用DTS的字段映射功能，详情请参见设置同步对象在目标实例中的名称。
映射名称更改	无	如需更改同步对象在目标实例中的名称，请使用对象名映射功能，详情请参见库表列映射。
源、目标库无法连接重试时间	无	当源、目标库无法连接时，DTS默认重试720分钟（即12小时），您也可以自定义重试时间。如果DTS在设置的时间内重新连接上源、目标库，同步任务将自动恢复。否则，同步任务将失败。 说明 由于连接重试期间，DTS将收取任务运行费用，建议您根据业务需要自定义重试时间，或者在源和目标库实例释放后尽快释放DTS实例。

9. 上述配置完成后，单击页面右下角的**预检查并启动**。

说明

- 在同步作业正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动同步作业。
- 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
  - 您可以根据提示修复后重新进行预检查。
  - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

10. 在**预检查**对话框中显示**预检查通过**后，关闭**预检查**对话框，同步作业将正式开始。

11. 等待同步作业的链路初始化完成，直至处于**同步中**状态。

    您可以在**数据同步**页面，查看数据同步作业的状态。

12.升级版本

12.1. PostgreSQL基础版升级高可用版

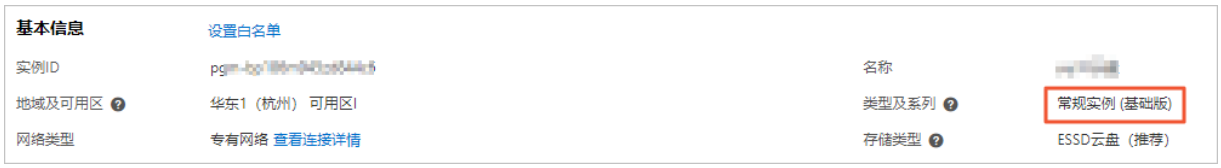
您可以将RDS PostgreSQL的基础版实例升级为高可用版，获得更高可靠性。

前提条件

实例版本如下：

- PostgreSQL 14基础版
- PostgreSQL 13基础版
- PostgreSQL 12基础版
- PostgreSQL 11基础版
- PostgreSQL 10基础版

> **说明** 您可以在实例的**基本信息**页面查看实例的系列。



背景信息

高可用版是适用性较广的云数据库系列。采用一主一备的经典高可用架构，适合80%以上的用户场景，包括互联网、物联网、零售电商、物流、游戏等行业。

详情请参见高可用版。

费用说明

升级版本的费用请参见变配。

影响

- RDS变更配置可能涉及底层数据迁移，请您耐心等待。在迁移完成后会根据您设置的切换时间自动进行切换，切换时会出现30秒左右的闪断，请确保应用具备重连机制。
- 升级后无法回退到基础版。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**基本信息**页面，单击**变更配置**。
3. （仅包年包月实例需要执行此步骤）在弹出的对话框中，选择**立即升配**，单击**下一步**。

4. 设置如下参数。

参数名称	说明
系列	选择**高可用版**。
实例规格	选择实例规格。每种规格都有对应的CPU核数、内存、最大连接数和最大IOPS。详情请参见主实例规格列表。
存储空间	设置存储空间。存储空间只能增加，不能降低。
切换时间	选择升级的切换时间： ○ 数据迁移结束后立即切换 ○ 可维护时间内进行切换

5. 勾选服务协议，单击**去支付**并完成支付。

相关API

API	描述
变更配置	变更RDS实例配置。

12.2. 升级数据库大版本

本文介绍如何升级 RDS PostgreSQL数据 库大版本，例如从PostgreSQL 10升级至11。

PostgreSQL社区每年发布一个大版本，社区对每个大版本支持5年，大版本通常带来功能、性能方面的革命性提升，而超过5年的数据库版本，由于社区不再提供支持，性能和安全的风险也将逐渐增加。

> **注意** 大版本升级会保留原实例，并创建一个新版本的实例，新实例在创建中不收费，创建成功后将会**产生费用**，计费方式为**按量付费**。新实例暂不继承原实例优惠，请根据实际情况选择是否升级。

为了让您享受PostgreSQL新的大版本带来的功能、性能提升，同时降低升级风险，RDS PostgreSQL提供大版本升级功能，步骤如下：

1. 创建升级检查报告

检查当前主实例是否支持大版本升级，并提供检查结果和检查报告，检查报告异常时不允许升级。

2. 执行升级
   - 指定PostgreSQL大版本进行升级，选择**不割接**创建一个新版本的测试实例进行业务测试及确保兼容性。
   - 测试结束后，在原实例上再次执行大版本升级，选择**割接**创建一个新版本的实例用于升级后使用。

注意事项

- 升级大版本前需要满足以下前提条件：
  - 版本：PostgreSQL 13、12、11、10、9.4
  - 系列：高可用版、基础版
  - 网络类型：专有网络。
    如果网络类型为经典网络，您需要将网络类型切换为专有网络，并且切换时不选择**保留原经典网络**。
    如何查看或切换网络类型，请参见切换网络类型。

    > **说明** 切换为专有网络时，如果您选择了**保留原经典网络**，则需要等待经典网络保留时间结束后，才能启动大版本升级任务。

  - 实例不是只读实例或专属集群实例。
  - 实例ID不是以 `pg-cn` 开头。
- 升级大版本将会造成以下影响：
  - 如果选择了割接方式升级，在割接过程中，原实例将会被设置为只读，并会造成分钟级业务闪断，请在业务低峰期进行升级。如果选择不割接方式进行升级，则不受影响。

    > **说明**
    > - 具体的闪断时长取决于数据量以及DNS缓存刷新时间。您可以尝试切换虚拟交换机，通过业务闪断时长来评估 客户端的 DNS缓存刷新时间。
    > - 升级过程的时间长短与实例的数据量相关。割接过程的时间长短，与数据量无关，仅受原数据库的数据定义数量影响，包括表定义、索引定义、视图定义、自定义函数、序列定义等。例如在实例中定义100张表、数据量为1G时的升级时长，与定义100张表、数据量为10TB时的升级时长相同。

  - 如果原实例使用了新版本不支持的参数，在新版本中该参数会被自动删除，如果在低版本的参数取值不在高版本对应参数的合法区间，在新版本中该参数会被置为高版本参数模板的默认值。
  - 新实例不继承原实例的**实例名称**、标签、云监控报警规则以及备份数据。
  - 如果待升级的实例作为数据传输服务（DTS）的源实例或目标实例，则升级后需要重建DTS任务。创建DTS任务详见DTS产品文档。
  - 自建的只读节点和逻辑复制槽在升级后仍然保留在原实例下，不会自动转移到高版本实例，需要在升级后进行重建。
- 当待升级的实例中创建了只读实例时，则无法直接进行升级，需要在升级前后进行以下操作：
  i. 将应用程序中配置的只读实例连接地址修改为主实例连接地址。

     > **说明** 为保证业务稳定，建议在业务低峰期修改应用连接地址。

  ii. 删除只读实例。

iii. 升级大版本。

iv. 升级完成后，在新实例上重新创建PostgreSQL只读实例。

v. 在应用程序中，将升级前修改的实例连接地址修改为新的只读实例连接地址。

功能特点

- 支持**跨版本升级**：例如直接从PostgreSQL 10.0升级到PostgreSQL 13.0。
- 支持**升级演练**：通过**不割接**的方式，可以在不影响原实例的情况下，验证整个升级流程。
- 支持 **平滑升级**：
  - **无需修改应用**：通过**割接**方式升级，新实例地址与原地址相同，无需修改应用。
  - **无需停机**：升级时原实例无需停机，而是暂时变为只读状态（持续时长为分钟级），降低业务中断带来的风险。此外，由于是通过克隆实例进行升级，所以即使升级失败也不会影响原实例。
  - **保留实例设置**：升级后保留原实例的白名单、参数设置、插件（新版本不支持的插件和参数除外）。
    - 升级后保留原实例的白名单、参数设置、插件（新版本不支持的插件和参数除外）。
    - 加密实例升级大版本后仍为加密实例，加密Key保持不变。

计费方式

升级后产生的新实例的计费方式统一为按量付费，您可以在确认业务平稳运行后把新实例转成包年包月以及释放原实例。

升级大版本

1. 如果待升级的实例中创建了只读实例，需在升级前执行以下步骤：

   i. 将应用程序中配置的只读实例连接地址修改为主实例连接地址。

   > **说明** 为保证业务稳定，建议在业务低峰期修改应用连接地址。

   ii. 删除只读实例。

2. 进入**大版本升级**页面。

   i. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

ii. 在左侧导航栏单击**大版本升级**。

> **说明**
> - 如果您在控制台中未看到**大版本升级**，请检查您的RDS PostgreSQL实例版本及系列等配置，具体请参见注意事项。
> - 当前仅新版控制台支持。

3. 在**升级检查**页签中，选择目标升级版本，然后单击**创建升级检查报告**，并等待数分钟。

> **说明**
> - 为确保升级能顺利完成，请确保升级检查报告的检查结果为成功时，再进行后续升级步骤。
> - 升级检查成功后，如果对主实例执行了 `CREATE EXTENSION` 操作，需重新执行检查。

当升级检查报告的检查结果为失败时，可单击**查看信息**查看报告详细内容，常见的报错及原因请参见解读RDS PostgreSQL大版本升级检查报告。

4. 单击**升级实例**页签，阅读警告内容，然后选择升级版本，并单击**继续创建升级实例**。
5. 在弹出的窗口中确认注意事项，单击**确定**，并设置参数（以下仅列举重点参数）：

配置	说明
存储类型	选择实例的存储类型。 版本升级功能基于云盘快照，当前升级后支持选择的存储类型如下。 ○ 当原实例为SSD云盘时，可选SSD云盘。 ○ 当原实例为ESSD云盘时，可选ESSD PL1云盘、ESSD PL2 云盘、ESSD PL3 云盘。 ○ 当原实例为本地SSD盘时，可选ESSD PL1云盘、ESSD PL2 云盘、ESSD PL3 云盘。
目标主可用区	
目标备可用区	

配置	说明
目标主实例交换机	系统支持升级后将新的主备实例配置到其他可用区。
目标备实例交换机	
割接配置	系统将创建一个新实例进行数据迁移。请选择迁移后是否 自动切换至新实例 。 ○ **不割接**：不自动切换。此选项一般用于正式升级之前测试当前业务在新版本中的兼容性。建议第一次执行时选择**不割接**，应用层完全测试验证通过后，释放实例并重复升级大版本操作，选择**割接**开始正式升级。 说明 ■ 迁移过程中，原实例的业务不会受到影响。 ■ 如需通过**不割接**升级实例，需在 迁移完成后 把应用程序中的数据库连接地址改为新实例的连接地址。如何查看连接地址，请参见查看或修改内外网地址和端口。 ○ **割接**：自动切换。您无需更改应用程序里的数据库连接地址，割接结束会自动连接到新实例。此选项一般用于在确认业务可以稳定运行在新版本之后执行正式升级。 说明 ■ 割接后无法回退，请谨慎选择。 ■ 切换过程中，原实例会被变为只读，业务无法写入，请务必在业务低峰期进行。 ■ 如果原实例中创建了只读实例，则无法选择**割接**。只能通过**不割接**升级实例，且原只读实例不会被克隆。升级完成后，您需要在新版本实例中另行创建PostgreSQL只读实例。
割接时间窗口	迁移完成后，业务切换至新实例的时间。 ○ **立刻**：迁移完成后立即切换。 ○ **实例可运维时间**：在可运维时间段内切换。 说明 仅当 **割接配置** 为 **割接** 时可选。
统计信息收集模式	选择在哪个时间点对数据库执行统计信息收集。 ○ **割接前收集**：可以保证业务稳定性。如果实例数据量太大可能会导致升级时间较久。 ○ **割接后收集**：实例升级速度较快。升级后访问未生成统计信息的表可能导致执行计划不准确，业务高峰期还可能导致数据库宕机。 说明 对于 **不割接** 场景，**割接前收集**表示新实例开放读写前收集，**割接后收集**表示新实例开放读写后收集。
存储空间	选择新版本实例的存储空间大小。

配置	说明
实例规格	选择新版本实例的规格。有关实例规格的更多详情，请参见主实例规格列表。

6. 单击**立即创建**。

   此时原实例状态变更为 **版本升级中**，并且实例列表中会多出一个状态为 **创建中**的新实例。当 两个实例的状态变更为 **运行中**即代表创建或升级完成。具体时间取决于数据量，请耐心等待。

   > **说明** 升级任务创建后不支持修改或删除，如需删除升级任务，请提 工单联系售后服务。

7. 如果在步骤一中删除了只读实例，则在升级后，还需执行以下步骤：
   i. 在新实例上重新创建PostgreSQL只读实例。
   ii. 在应用程序中，将步骤一中修改的实例连接地址修改为新的只读实例连接地址。

后续步骤

1. 确认业务已稳定运行在新实例中后，将新实例转成包年包月。
2. 释放原实例。
3. 新实例不包含只读实例，您可以按需创建PostgreSQL只读实例。

相关API

API	描述
UpgradeDBInstanceMajorVersionPrecheck	RDS PostgreSQL实例大版本升级检查
DescribeUpgradeMajorVersionPrecheckTask	查询RDS PostgreSQL实例大版本升级检查报告
UpgradeDBInstanceMajorVersion	RDS PostgreSQL实例升级数据库大版本
DescribeUpgradeMajorVersionTask	查询RDS PostgreSQL实例大版本升级历史任务

后续步骤

12.3. 解读RDS PostgreSQL大版本升级检查报告

本文介绍RDS PostgreSQL大版本升级检查报告的内容详情、常见报错以及处理方式。
如果RDS PostgreSQL大版本升级检查报告的结果为失败，您可以单击**查看信息**查看报告详细内容，大版本升级检查报告包含以下内容。

- user_check_report
- pg_upgrade_internal.log
- pg_upgrade_server.log
- loadable_libraries.txt
- tables_with_oids.txt

说明 大版本升级的更多信息，请参见 升级数据库大版本。

user_check_report

检查是否存在后台添加的多余超级账号或者加密方式不正确的普通账号。

报错	可能原因	解决方法
invalid superuser: ["user_01"]	存在冗余的超级账号，使用此账号连接数据库将会使数据库只读设置无效，从而导致与大版本升级的数据不一致。	冗余的超级账号请提交工单联系售后服务删除。
invalid user: ["user_02"]	存在异常的普通账号，此账号将会导致升级后账号无法连接。	重置报错账号的密码。

pg_upgrade_internal.log

兼容性检查日志。

报错	可能原因	解决方法
A list of problem libraries is in the file: loadable_libraries.txt	loadable_libraries.txt文件中存在高版本不兼容的插件。	排查loadable_libraries.txt报告项中列出的插件，根据实际需要进行评估是否删除，建议在不影响业务稳定的情况下删除后升级。各版本支持的插件列表请参见：支持插件列表。
A list of tables with the problem is in the file: tables_with_oids.txt	建表时声明了 `WITH OIDS` ，该声明在PostgreSQL 12或以上版本不支持。	• 方法一：（推荐）：升级到PostgreSQL 11版本，该版本支持声明 `WITH OIDS` 。 • 方法二：排查tables_with_oids.txt报告项列出的表，请自行评估业务代码是否对OID有依赖，确认不影响业务之后，执行以下语句： `ALTER TABLE {table_name} SET WITHOUT OIDS;`

pg_upgrade_server.log

展示目标实例启动的日志。

loadable_libraries.txt

展示不兼容的library，通常对应了不兼容的插件。

报错	可能原因	解决办法
could not load library "$libdir/pgrouting-2.6.2": ERROR: could not access file "$libdir/pgrouting-2.6.2": No such file or directory	选定的高版本中不支持pgrouting插件。	排查loadable_libraries.txt文件中列出的插件，根据实际需要进行评估是否删除，建议在不影响业务稳定的情况下删除后升级。各版本支持的插件列表请参见：支持插件列表。
could not load library "$libdir/jsonbx": ERROR: could not access file "$libdir/jsonbx": No such file or directory	PostgreSQL 9.4对JSON类型支持不完善，需要jsonbx插件支持，PostgreSQL 10或以上版本原生支持JSON，没有该插件。	请在高版本中验证jsonbx插件的相关函数使用，根据实际需要进行评估是否删除插件，然后在不影响业务稳定的情况下删除后升级。jsonbx插件函数使用差异请参见jsonbx插件函数使用差异。

报错	可能原因	解决办法

• could not load library "$libdir/postgis-2.5": ERROR: could not access file "$libdir/postgis-2.5": No such file or directory
• could not load library "$libdir/postgis_topology-2.2": ERROR: could not access file "$libdir/postgis_topology-2.2": No such file or directory
	• postgis插件版本过低，与选定的高版本lib库不匹配，升级检查失败。
	• postgis_topology插件版本过低，与选定的高版本lib库不匹配，升级检查失败。
		注意 postgis插件各版本间存在差异（例如wkt格式的解析报错不同），升级后可能影响业务处理，建议在升级postgis插件前，先克隆当前实例，在克隆实例中对postgis插件进行升级测试，完全符合业务需求后，再在原实例中升级postgis插件。克隆的具体操作，请参见 备份PostgreSQL数据和 恢复PostgreSQL数据。
		依次执行如下步骤：
		1. 升级内核小版本，更多信息，请参见升级内核小版本。
		2. 更新报错提示的postgis插件：
		○ postgis插件
		`ALTER EXTENSION postgis UPDATE;`
		○ postgis_topology插件
		`ALTER EXTENSION postgis_topology UPDATE;`
		3. 执行 `\dx` 命令查询插件版本，确保postgis插件版本至少为2.5.4。
		4. 重新进行升级检查。
		注意 不同PostgreSQL实例版本安装postgis、postgis_topology或pgrouting插件后，升级大版本存在如下限制：
		○ PostgreSQL 9.4：暂时仅支持升级到PostgreSQL 10或11。
		○ PostgreSQL 10：暂时仅支持升级到PostgreSQL 11。
		○ PostgreSQL 11、12和13：暂不支持升级。

jsonbx插件的部分函数用法在PostgreSQL的不同版本中执行结果可能存在差异，具体如下表所示，您需要在升级前评估是否删除插件。

函数用法	PostgreSQL 9.4执行结果	PostgreSQL 10或以上版本执行结果
select '{"a":1, "b":2, "c":3}'::jsonb - 2;	{"a": 1, "b": 2}	ERROR: cannot delete from object using integer index
select jsonb_delete('{"a":1, "b":2, "c":3}'::jsonb, '{b}'::text[]);	{"a": 1, "c": 3}	ERROR: function jsonb_delete(jsonb, text[]) does not exist

函数用法	PostgreSQL 9.4执行结果	PostgreSQL 10或以上版本执行结果
select '{"a":{"c":1, "d":2}, "b":3}'::jsonb - '{a, c}'::text[];	{"a": {"d": 2}, "b": 3}	{"b": 3}

tables_with_oids.txt

展示声明了 `WITH OIDS` 的表清单。

12.4. 升级内核小版本

云数据库RDS PostgreSQL支持升级内核小版本，内核小版本的升级涉及性能提升、新功能或问题修复等。

RDS PostgreSQL内核小版本功能详情请参见AliPG 小版本 Release Notes。

注意事项

- 升级内核小版本会重启实例，RDS服务可能会出现一次30秒的闪断，请您尽量在业务低峰期执行升级操作，或确保您的应用有自动重连机制。
- 升级内核小版本后无法降级。
- 如果执行了升配等操作，实例会升级到新的内核小版本。

本地盘实例升级内核小版本

本地盘实例暂不支持手动升级内核小版本。您可以通过重启实例自动升级到最新的小版本。具体操作请参见重启实例。

> **说明** 如果当前实例为主实例，拥有只读实例，请先逐个重启所有只读实例，再重启主实例。如果只重启主实例，则只读实例不会升级内核小版本。

云盘实例升级内核小版本

> **说明**
> 如果当前实例为主实例，拥有只读实例，可以通过两种方式进行升级。
> - 对主实例发起内核小版本升级，该主实例下的所有只读实例都会立即进行升级。
> - 如果您不希望所有只读实例立即升级，请先逐个升级所有只读实例，然后再升级主实例。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**配置信息**区域中单击**升级内核小版本**。

3. 在弹出的对话框中，选择**升级时间**，单击**确定**。

查看内核小版本

您可以通过如下两种方法查看实例当前的内核小版本。

- 登录RDS管理控制台，在目标实例的**基本信息**页面查看。

  > **说明** 此方法仅适用于PostgreSQL云盘版实例。

- 连接目标PostgreSQL实例，通过 `show rds_release_date;` 命令查看。

  > **说明** 此方法适用于PostgreSQL云盘版和本地盘版实例。

相关API

API	描述
升级内核小版本	升级数据库的内核小版本。

13.实例

13.1. 创建RDS PostgreSQL实例

您可以通过阿里云RDS管理控制台或API创建RDS实例。本文介绍如何通过控制台创建RDS PostgreSQL实例。

> **说明** 首次购买RDS享受折扣价。详情请参见 优惠活动。

前提条件

通过RAM用户创建RDS实例时，该RAM用户必须拥有AliyunRDSFullAccess权限。如何授权，请参见RAM资源授权。

操作步骤

1. 点此打开RDS实例创建页面。
2. 选择**计费方式**。

计费方式	建议	好处
包年包月	长期使用RDS实例，请选择**包年包月**（一次性付费），并在页面左下角选择**购买时长**。	包年包月比按量付费更实惠，且购买时长越长，折扣越多。
按量付费	短期使用RDS实例，请选择**按量付费**（按小时付费）。 您可以先创建**按量付费**的实例，确认实例符合要求后转**包年包月**。	可随时释放实例，停止计费。

> **说明** 可以在页面右下角查看价格。您需要完成后续的配置选择，才能最终确定价格。

3. 设置以下参数。

类别	说明
地域	选择要在哪个地域创建RDS实例。 ○ 如果您要通过云服务器ECS连接该RDS实例，请在此选择ECS实例所在的地域，使得RDS实例与ECS实例位于同一地域（例如，都位于杭州）。否则，ECS实例与RDS实例无法内网互通，只能外网互通，无法发挥最佳性能。 ○ 如果您要通过其它设备（例如本地服务器或电脑）连接该RDS实例，选择一个离该设备较近的地域即可。
类型	本文介绍PostgreSQL，选择**PostgreSQL**。 当前支持PostgreSQL 10、11、12、13和14。 **说明** 不同地域支持的数据库类型不同，请以实际界面为准。

类别	说明
系列	○ **基础版**：一个节点，性价比高，用于学习或测试。 ○ **高可用版**（推荐）：一个主节点和一个备节点，经典高可用架构，用于生产环境。 说明 ○ 支持的系列取决于您选择的地域和数据库类型。 ○ 关于各系列的详细介绍，请参见产品系列概述。
存储类型	○ **本地SSD盘**：与数据库引擎位于同一节点的SSD盘。将数据存储于本地SSD盘，可以降低I/O延时。当前PostgreSQL10支持本地SSD盘。 ○ **ESSD云盘**：增强型（Enhanced）SSD云盘，分为如下三类： ■ ESSD PL1云盘：PL1性能级别的ESSD云盘。 ■ ESSD PL2云盘：相比PL1大约提升2倍IOPS和吞吐量。 ■ ESSD PL3云盘：相比PL1最高可提升20倍IOPS、11倍吞吐量，适合对极限并发I/O性能要求极高、读写时延极稳定的场景。 ○ **SSD云盘**：基于分布式存储架构的弹性块存储设备。选择SSD云盘，即实现了计算与存储分离。 说明 ○ 支持的存储类型取决于您选择的实例类型和系列。 ○ 如果选择**ESSD云盘**或**SSD云盘**，可以勾选**云盘加密**，进一步保护数据安全。详情请参见云盘加密。 ○ 更多信息，请参见存储类型。
可用区	可用区是指地域中的一个独立物理区域，例如，杭州地域有可用区H、I、J等。 ○ **多可用区部署**（推荐）：主节点和备节点位于不同可用区，提供跨可用区容灾。 ○ **单可用区部署**：主节点和备节点位于同一可用区。 说明 如果实例系列选择为基础版，则只能选择 **单可用区部署**。
实例规格	先选择分类（入门级或企业级），再选择具体规格。 ○ **通用规格（入门级）**：独享被分配的内存和I/O资源，与同一服务器上的其他通用型实例共享CPU和存储资源。 ○ **独享规格（企业级）**：独享型指独享被分配的CPU、内存、存储和I/O资源。独占型是独享型的顶配，独占整台服务器的CPU、内存、存储和I/O资源。 说明 详情请参见 主实例规格列表。

类别	说明
存储空间	存储空间包括数据空间、系统文件空间、Binlog文件空间和事务文件空间。调整存储空间时最小单位为5GB。 **说明** 部分本地SSD盘的存储空间大小与实例规格绑定，ESSD云盘或SSD云盘不受此限制。详情请参见 主实例规格列表。

4. 在页面右下角，单击**下一步：实例配置**。
5. 选择 。网络类型
   i. **网络类型**：建议选择与ECS实例相同的网络类型，否则，ECS实例与RDS实例无法内网互通。

   > **说明**
   > - 根据准备工作，确认您的ECS实例的网络类型。
   > - 如果不需要通过内网访问RDS，那么任选一种网络类型。
   > - 部分RDS实例不支持经典网络。

   ii. **VPC和交换机**：如果网络类型为专有网络，还需选择VPC和交换机，建议选择与ECS实例相同的VPC。ECS实例与RDS实例位于不同VPC时，无法内网互通。

   > **说明**
   > - 根据准备工作，确认您的ECS实例的VPC名称。
   > - VPC相同，交换机不同，ECS实例与RDS实例也可以内网互通。

6. 设置更多自定义参数。如无特殊需求，可保留默认值。

类别	说明
实例释放保护	选中此项可以为按量付费实例开启实例释放保护，防止按量付费实例被意外释放。更多信息，请参见开启和关闭实例释放保护。
资源组	选择默认或自定义的资源组，方便实例管理。
时区	设置实例时区。 **说明** - 购买主实例时支持选择时区，只读实例不支持选择时区，只读实例将继承主实例时区。 - 仅当实例为PostgreSQL云盘时，该参数可配置。 - 支持命名时区，不支持UTC时区。命名时区的详细信息，请参见RDS MySQL和RDS PostgreSQL实例的常用命名时区清单。 - 如果不配置此参数，系统将根据您购买实例的地域，选择默认时区。地域与默认命名时区的对应关系，请参见RDS PostgreSQL实例地域与默认命名时区对应关系。

7. 在页面右下角，单击**下一步：确认订单**。
8. 确认订单信息、**购买量** 和**购买时长**（仅包年包月实例），勾选**服务协议**，单击**去支付**，并完成支付。

   > **说明** 对于包年包月实例，建议选中 **到期自动续费**，避免因忘记续费而导致业务中断。

   控制台提示**支付成功**或**开通成功**。
9. 查看实例。

   进入 实例列表，在上方选择实例所在地域，根据 **创建时间** 找到刚刚创建的实例。

下一步

创建数据库和账号

常见问题

为什么创建实例后，实例列表看不到创建中的实例？

可能原因	说明	建议
地域错误	您所在地域和您创建实例时选择的地域不一致。	在页面左上角切换地域。
可用区内资源不足	可用区内资源不足，导致创建失败。 创建失败 您可以在订单列表里看到退款。	选择其它可用区后重试。
RAM权限策略禁止创建未加密的RDS实例	已配置RAM权限策略，禁止RAM用户创建未加密的RDS实例。 • RAM用户尝试创建本地盘实例，实例创建失败（本地盘实例无法在创建时设置磁盘加密）。 • RAM用户尝试创建云盘实例，但未设置云盘加密，实例创建失败。 更多信息，请参见禁止创建磁盘未加密的RDS实例。	创建实例时，存储类型选择云盘，选中云盘加密并设置密钥后重试。

相关文档

- 通过API创建RDS实例：创建RDS实例
- 创建其它类型实例请参见：
  - 创建RDS MySQL实例
  - 创建RDS SQL Server实例
  - 创建RDS MariaDB实例

13.2. 变更配置

本文介绍如何变更RDS PostgreSQL实例配置，包括系列、规格和存储空间。

前提条件

您的阿里云账号没有未支付的续费订单。

变更项

如您需要横向扩展数据库的读取能力，请参见PostgreSQL只读实例简介和创建PostgreSQL只读实例，通过只读实例来分担主实例的压力。

变更项	说明
存储类型	存储类型的性能级别从低到高分别为：**SSD云盘 < ESSD PL1云盘 < ESSD PL2云盘 < ESSD PL3云盘** 高可用版云盘实例支持升级存储类型，不支持降级。 **说明** 基础版云盘实例如需升级存储类型，请先 升级至高可用版。
实例规格	所有实例类型都支持变更规格。
存储空间	所有实例都支持增加存储空间。 **说明** • 增加存储空间时不能超过该规格的存储空间限制，详情请参见主实例规格列表。 • 实例无法通过变更配置降低存储空间，您可以使用一键上云功能实现缩容，具体请参见RDS PostgreSQL实例缩容。 • 若当前规格对应的存储空间范围无法满足您的需求，请选择其它实例规格。 • 只读实例（云盘）采用在线扩容的方式，不会造成业务的闪断。 • 主实例（云盘）在原地扩容过程中，将会出现短暂的业务闪断。

说明 变更上述配置不会导致实例连接地址的改变。

计费规则

请参见变配的计费规则。

注意事项

- 由于基础版只有一个数据库节点，没有备节点作为热备份，因此当该节点意外宕机或者执行变更配置、版本升级等任务时，会出现较长时间的不可用。如果业务对数据库的可用性要求较高，不建议使用基础版，可选择其他系列（如高可用版）。
- 变更配置可能会进行数据迁移，迁移完成后根据您选择的切换时间进行切换（期间保持增量同步），切换过程中会出现一次约30秒的闪断，而且与数据库、账号、网络等相关的大部分操作都无法执行，请尽量在业务低峰期执行变配操作，或确保您的应用有自动重连机制。
- 变更配置后无需您手动重启实例。
- 变更配置不会导致实例ID和连接地址的改变。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**配置信息**区域单击**变更配置**。
3. （仅包年包月实例需要执行此步骤）在弹出的对话框中，选择变更方式，单击**下一步**。

> **说明** 变更方式说明如下：
> **立即升配**或**立即降配**：
> 变配后，新的配置立即生效。包年包月实例和按量付费实例都支持立即升降配。
> 变更任务下达后，系统将磁盘数据同步到一个新实例，然后根据**立即升降配**确定时间，到时间后系统将原实例的实例ID和连接地址等信息切换到新实例，实例ID、连接地址等不会改变。

4. 修改实例的配置。详情请参见变更项。
5. 选择变更实例配置的执行时间。
   - **数据迁移结束后立即切换**：变更实例配置会涉及到底层的数据迁移，您可以选择在数据迁移后立即切换。
   - **可维护时间内进行切换**：在变更配置生效期间，可能会出现一次约30秒的闪断，而且与数据库、账号、网络等相关的大部分操作都无法执行，因此您可以选择在可维护时间段内执行切换的操作。
6. 勾选服务协议，单击**去支付**，并完成支付。

> **警告**
> - 变配订单提交后无法取消，请在执行变配前详细评估业务需求。
> - 为确保变配的稳定进行，在提交变配订单至变配完成期间，请勿执行DDL操作。

常见问题

1. 云盘版实例如何变更为本地盘实例？
   答：请参见云盘如何变更为本地盘。
2. 仅扩容存储空间，需要迁移数据到新实例吗？
   答：需要检查实例所在主机上是否有足够存储空间用于扩容。如果有则直接扩容，不需要迁移数据；如果没有，则需要迁移数据到拥有足够存储空间的主机上。

相关API

API	描述
ModifyDBInstanceSpec	变更RDS实例配置

13.3. 重启实例

当实例出现连接数满或性能问题时，您可以手动重启实例。

影响

- 重启实例会造成30秒左右的连接中断，重启前请做好业务安排，谨慎操作。

> **说明** RDS基础版实例只有一个数据库节点，没有备节点作为热备份，因此当该节点意外宕机或者执行重启实例、变更配置、版本升级等任务时，会出现较长时间的不可用。如果业务对数据库的可用性要求较高，不建议使用基础版实例，可选择其他系列（如高可用版），部分基础版实例也支持 升级为高可用版。

- 本地盘实例在重启后，内核小版本会自动升级至最新版。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在页面右上角，单击**重启实例**。

3. 在弹出的确认框中，单击**确定**。

相关API

API	描述
重启实例	重启RDS实例。

13.4. 自动或手动主备切换

云数据库RDS PostgreSQL支持高可用切换功能。当主实例不可用时，会自动触发主备切换，将主实例和备实例进行互换，切换后实例地址保持不变，应用自动连接到新的主实例（原备实例），从而保障实例的高可用性。此外，您还可以手动切换主备实例。

前提条件

实例为高可用版实例。

> **说明**
> - 基础版实例没有备实例，所以不支持主备实例的切换。
> - 仅高可用版云盘实例支持临时关闭自动切换功能。

背景信息

- 自动切换：实例默认开启自动切换，当主实例出现故障无法访问时，会自动切换到备实例。主备切换的触发时机，请参见主备切换原因。
- 手动切换：即使自动切换是开启状态，您也可以手动进行主备切换。手动切换可用于容灾演练或多可用区场景下的就近连接等需求。

> **说明** 主备实例的数据会实时同步，您只能访问主实例，备实例仅作为备份形式存在，不提供业务访问。

影响

- 主备实例切换过程中会出现闪断，请确保您的应用程序具有自动重连机制。
- 如果实例下挂载有只读实例，那么主备实例切换后，只读实例的数据会有几分钟的延迟，因为需要重建复制链路、同步增量数据等。
- 主备切换不会导致 变更，但后台对应的IP地址可能会变化。使用实例连接地址时，主备切换不会对您的应用正常运行造成影响。实例连接地址

手动切换主备实例

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

2. 在左侧导航栏中，选择**服务可用性**。
3. 在**实例可用性**区域，单击**主备库切换**。
4. 选择切换时间，然后单击**确定**。

> **说明** 在主备实例切换期间，有很多操作无法执行，例如管理数据库和账号、切换网络类型等，建议您选择**在设置时间内切换**。

临时关闭主备自动切换

自动切换默认为开启，主实例出现故障会自动切换到备实例，在遇到如下情形时您可以选择临时关闭主备自动切换：

- 大促活动等，不希望主备切换影响系统可用性。
- 重要应用系统升级等，不希望主备切换引进其他变数。
- 重大事件或者稳定保障期，不希望主备切换影响系统稳定性。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**服务可用性**。
3. 在**实例可用性**区域，单击**自动切换主备**。

> **说明** 如果看不到**自动切换主备**按钮，请确认您的实例是否为高可用版。

4. 选择**临时关闭**，并设置**临时关闭截止时间**，然后单击**确定**。

> **说明**
> - 到达**临时关闭截止时间**后，实例恢复为自动进行主备切换。
> - 默认为临时关闭1天，最长可设置为7天后的23:59:59。

设置完毕后，您可以在**服务可用性**页面查看临时关闭主备自动切换的截止时间。

查看主备切换日志

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**服务可用性**。
3. 在**主备切换日志**区域，调整查询时间范围，查看主备切换日志。

常见问题

- Q：备实例可以访问吗？

不可以访问。您只能访问主实例，备实例仅作为备份形式存在，不提供业务访问。

- Q：主备切换后，是否需要手动切换回主实例？
  不需要，主备实例的数据完全一致，主备切换后原备实例被选举成为新的主实例，无需进行额外操作。

相关API

API	描述
切换主备实例	切换RDS主备实例。
设置主备自动切换	开启或关闭RDS实例主备自动切换。
查询主备切换设置	查看RDS实例主备自动切换设置。

13.5. 设置可维护时间段

为保障云数据库RDS实例的稳定性，后端系统会不定期对实例进行维护操作。默认可维护时间段为02:00~06:00，您可以根据业务规律，将可维护时间段设置在业务低峰期，以免维护过程中可能对业务造成的影响。

注意事项

- 在进行正式维护前，RDS会给阿里云账号中设置的联系人发送邮件，请注意查收。
- 实例维护当天，为保障整个维护过程的稳定性，实例会在可维护时间段时将实例状态切换为**实例维护中**。当实例处于该状态时，对数据库的访问以及查询类操作（如性能监控）不会受到任何影响，但除了账号管理、数据库管理和IP白名单设置外的变更操作（如升降级、重启等）均暂时无法使用。
- 在可维护时间段内，实例会发生1到2次连接闪断，请确保应用程序具有重连机制。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**配置信息**区域，单击**可维护时间段**后的**设置**。
3. 选择一个合适的可维护时间段并保存。

> **说明** 显示的可维护时间段和登录控制台的电脑的时区一致。

相关API

API	描述
修改可维护时间段	修改RDS实例可维护时间。

13.6. 迁移可用区

您可以将RDS实例迁移至同一地域内的其它可用区。迁移可用区后，实例的所有属性、配置和连接地址都不会改变。迁移所需时间跟实例的数据量有关，通常为几个小时。

前提条件

- 根据实例的存储类型、版本、系列及规格族，该功能的支持情况如下：

存储类型	实例版本	系列	规格族	是否支持
云盘	RDS PostgreSQL 14、13、12、11、10	高可用版	独享型	所有规格均支持
			通用型	支持的规格有： ○ pg.n2.medium.2c ○ pg.n2.small.2c
		基础版	通用型	不支持
本地盘	RDS PostgreSQL 10	高可用版	通用型	所有规格均支持
	RDS PostgreSQL 9.4	高可用版	独享型	
			通用型	

- 实例所在的地域有多个可用区时，才支持迁移可用区功能。关于地域和可用区的详情，请参见地域和可用区。
- 实例为主实例，不是只读实例，且主实例下未创建只读实例。

注意事项

- 云盘实例不支持在保持主可用区不变的情况下迁移备可用区，例如：
  - 不支持将**新加坡 可用区A（主） + 可用区B（备）**迁移至**新加坡 可用区A（主） + 可用区C（备）**。
  - 支持将**新加坡 可用区A（主） + 可用区B（备）**迁移至**新加坡 可用区C（主） + 可用区A（备）**
- 本地盘实例只支持迁移主可用区。

费用

迁移可用区功能免费。即使将实例从单可用区迁移至多个可用区，也不收取费用。

影响

- 切换时实例可用性会受到短暂影响，请确保应用具有自动重连机制。
- 迁移可用区会造成虚拟IP（VIP）的变更，请尽量在您的应用程序中使用连接地址进行连接，不要使用IP地址。
- 请及时清理客户端DNS缓存。客户端采用JVM的应用，建议将JVM配置中的TTL设置为不超过60秒，可确保在连接地址的VIP地址发生变更时，应用程序可以通过重新查询DNS来接收和使用资源的新VIP地址。

> **说明** 下列JVM中设置TTL的方法可供参考：
> - 为所有使用JVM的应用程序设置TTL：将*$JAVA_HOME/jre/lib/security/java.security*文件中的networkaddress.cache.ttl 参数值设置为60。
> - 仅为本地应用程序设置TTL：在首次调用 `InetAddress.getByName()` 前，即建立任何网络连接之前，在应用程序的初始化代码中设置 `networkaddress.cache.ttl java.security.Security.setProperty("networkaddress.cache.ttl" , "60");` 。

- 如果有正在执行的DTS任务，可用区迁移后，需要重启相应的DTS任务。

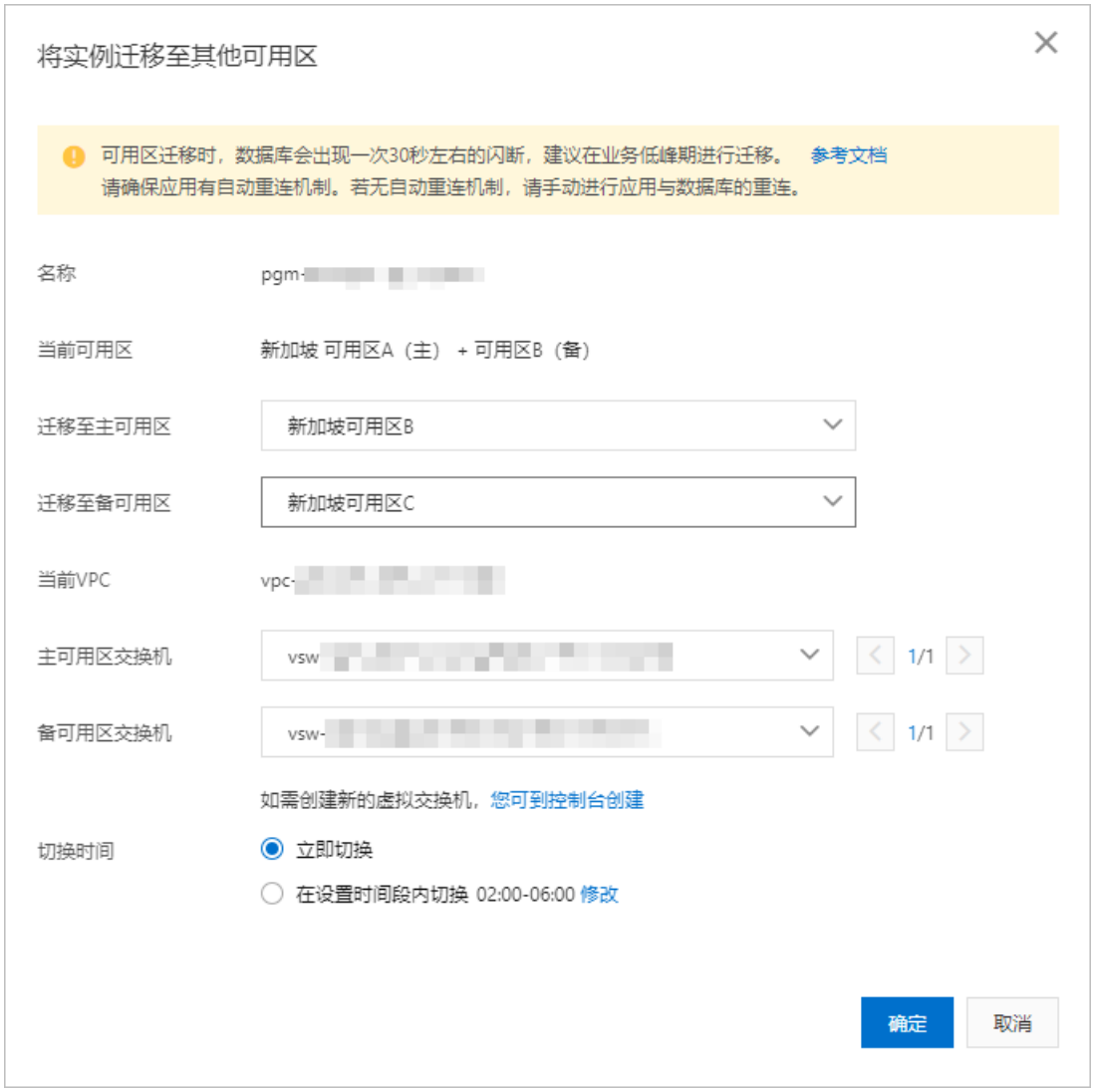
迁移类型

迁移类型	场景
从一个可用区迁移至另一个可用区	实例所在可用区出现满负载或者其它影响实例性能的情况。
从一个可用区迁移至多个可用区	提高实例的容灾能力，实现跨机房容灾。主备实例分别位于不同的可用区。 相对于单可用区实例，多可用区实例可以承受更高级别的灾难。例如，单可用区实例可以承受服务器和机架级别的故障，而多可用区实例可以承受机房级别的故障。
从多个可用区迁移至一个可用区	为了满足特定功能的要求。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**基本信息**区域单击**迁移可用区**。

3. 在弹出的对话框中，选择需要迁移的主备可用区及交换机，然后单击**确定**。



主要参数及配置规则

参数	配置规则
迁移至主可用区	选择新的**主可用区**和**备可用区**。 ○ 如果实例存储类型为本地盘，只支持迁移主可用区，不支持迁移备可用区。 ○ 如果实例存储类型为云盘，支持同时迁移主可用区和备可用区，不支持主可用区不变，只迁移备可用区。
迁移至备可用区	说明 以当前可用区为**新加坡 可用区A（主） + 可用区B（备）**为例，如果需要实现主可用区不变，只迁移备可用区（即迁移后为**新加坡 可用区A（主） + 可用区C（备）**），需要按如下步骤迁移： a. 将**新加坡 可用区A（主） + 可用区B（备）**迁移为**新加坡 可用区C（主） + 可用区A（备）** b. 使用主备切换，将**新 加坡 可用区C（主） + 可用区A（备）** 切换为**新加坡 可用区A（主） + 可用区C（备）**，具体请参见自动或手动主备切换。

参数	配置规则
主可用区交换机	选择主、备可用区对应的虚拟交换机。如果当前可用区无虚拟交换机，请参见创建交换机。
备可用区交换机	
切换时间	○ **立即切换**：配置后立即切换。 ○ **在设置时间段内切换**：单击**修改**，选择切换可用区的时间段。

单击**确定**后，底层开始拷贝数据到目标可用区，不影响实例运行。拷贝完成后，将按您指定的切换时间（**立即切换**或**在可维护时间段内切换**），把流量切换到新链路。

> **说明**
> - 切换时会发生连接闪断。请确保应用具有自动重连机制。若应用没有自动重连机制，需手动重连。
> - 由于客户端DNS缓存可能没有及时刷新，部分流量可能在10分钟后才进行切换，导致第二次闪断。客户端采用JVM的应用，建议将JVM配置中的TTL设置为不超过60秒，可确保在连接地址的VIP地址发生变更时，应用程序可以通过重新查询DNS来接收和使用资源的新VIP地址。设置方法请参见本文的影响章节。

相关API

API	描述
迁移可用区	迁移RDS实例可用区。

13.7. 释放实例

根据业务需求，您可以手动释放按量付费实例或者退订包年包月实例。

注意事项

释放或退订后，实例立即被清除，数据不再保留，请提前备份数据并下载备份。

释放按量付费实例

1. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
2. 通过如下两种方式打开**释放实例**对话框：
   - 找到目标实例，在**操作**列中选择**更多 > 释放实例**。
   - a. 找到目标实例，单击实例ID。
     b. 在**基本信息**页面，单击**释放实例**。
3. 在弹出的对话框中，单击**确定**。

退订包年包月实例

您可以提交工单申请退订包年包月实例。

相关API

API	描述
释放RDS实例	释放按量付费的RDS实例（包年包月实例暂不支持通过API退订）

13.8. 开启和关闭实例释放保护

如果您 的RDS按 量付费实例承载了关键业务，可以为按量付费实例开启实例释放保护，防止手动释放按量付费实例，有效避免因操作疏忽、团队成员沟通不及时等原因造成不可挽回的后果。本文为您介绍如何开启和关闭实例释放保护。

前提条件

实例的付费类型为按量付费。

注意事项

实例释放保护不能阻止因合理原因自动执行的释放行为，包括但不限于：

- 账号欠费超过15天，实例被自动释放。
- 实例存在安全合规风险，被停止或释放。

保护效果

对开启了实例释放保护的实例进行释放的结果如下：

- 在控制台手动释放实例，提示 `已开启实例释放保护，将无法执行释放操作` 。
- 调用DeleteDBInstnace接口释放实例，返回错误码 `OperationDenied.DeletionProtection` 。

创建实例时开启实例释放保护

本步骤重点介绍实例释放保护相关的选项，如需了解其它配置，请参见创建RDS PostgreSQL实例。

1. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
2. 在**实例列表**页面，单击**创建实例**。
3. 在**基础资源**页面，**计费方式**配置为**按量付费**，根据需要配置其余选项，然后单击**下一步：实例配置**。
4. 在**实例配置**页面，选中**防止通过控制台或API误删除释放**，根据需要配置其余选项，然后单击**下一步：确认订单**。
5. 根据需要配置其余选项，直至成功创建实例。

> **说明** 调用 CreateDBInsance、 CloneDBInstance接口创建实例时，您可以通过DeletionProtection 参数控制是否开启实例释放保护。

修改实例释放保护属性

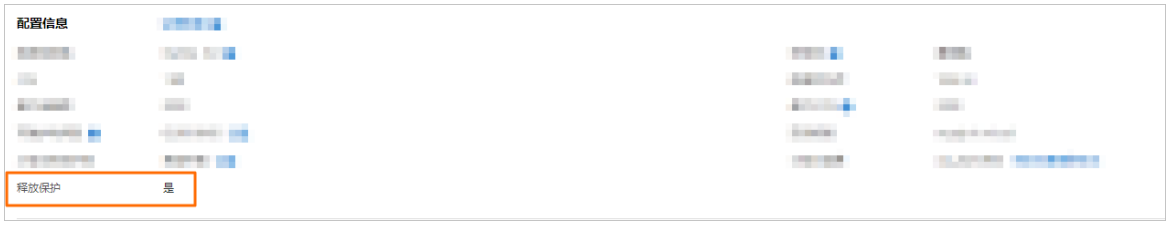
您也可以修改现有实例的信息，开启或关闭实例释放保护。

1. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
2. 在**实例列表**页面找到待修改的实例，在**操作**列，单击**更多 > 修改实例释放保护**。
3. 根据需要打开或关闭**实例释放保护**。
4. 单击**确定**。

> **说明** 您同样可以通过调用 ModifyDBInstanceDeletionProtection接口修改实例释放保护属性。

查看实例释放保护的状态

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**基本信息**页面，查看**配置信息**区域下的**释放保护**。



相关API

API	描 述
创建RDS实例	调用CreateDBInstance接口创建RDS实例。
恢复数据（克隆实例）	调用CloneDBInstance接口将历史数据恢复至一个新实例（称为克隆实例）。
开启或关闭实例释放保护	调用ModifyDBInstanceDeletionProtection接口开启或关闭RDS实例的释放保护功能。

13.9. 设置实例参数

您可以使用控制台或者API查看和修改部分参数的值，以及查询参数修改历史。

操作步骤

注意事项

- 部分参数修改后，在**提交参数**时会立即重启实例，详情请参见控制台上**可修改参数**页面中的**是否重启**列。重启实例会造成连接中断，重启前请做好业务安排，谨慎操作。
- 修改参数值时请参见控制台上**可修改参数**页面中的**可修改参数值**列。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**参数设置**。
3. 在**可修改参数**标签页找到目标参数进行修改。您可以选择修改单个参数或批量修改参数，步骤如下：
   - 修改单个参数
     a. 单击要修改的参数后的

     。
     b. 输入目标值并单击**确定**。
     c. 单击**提交参数**。
     d. 在弹出的对话框中，单击**确定**。

- 批量修改参数
  a. 单击**导出参数**，导出参数文件到本地。
  b. 打开参数文件，修改参数值。
  c. 单击**导入参数**。
  d. 在**导入参数**窗口粘贴要修改的参数及参数值，并单击**确定**。
  e. 确认参数列表中的参数修改结果，单击**提交参数**。

查询参数修改历史

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**参数设置**。
3. 选择**修改历史**标签页。
4. 选择要查询的时间范围，单击**查询**。

相关API

API	描述
修改实例参数	调用ModifyParameter接口修改RDS实例参数。
查询参数模板	调用DescribeParameterTemplates接口查看数据库参数模板。
查询参数配置	调用DescribeParameters接口查询实例当前的参数配置。

参数说明

请参见官方文档：PostgreSQL参数说明。

13.10. 设置实例保护级别

您可以根据自身业务特点，设置RDS PostgreSQL实例的保护级别，提高云数据库可用性或性能。

前提条件

实例为PostgreSQL高可用版（云盘）。

背景信息

您可以通过修改参数synchronous_commit（同步提交参数）、rds_sync_replication_timeout（同步复制超时时间参数）和synchronous_standby_names（同步的备实例名称）设置实例的保护级别。

RDS PostgreSQL的保护级别如下：

保护级别	数据复制方式	说明	参数配置
最高性能（默认）	异步模式	适合数据持久性要求不高的场景，响应速度最快。	synchronous_commit = off 说明 synchronous_commit = off时，rds_sync_replication_timeout无效。
最大保护	强同步模式	适合数据持久性要求极高的场景，响应速度较慢。	• synchronous_commit = remote_write • rds_sync_replication_timeout = 0 • synchronous_standby_names = standby1 警告 如果您有自建的从库，请不要命名为standby1，因为synchronous_standby_names默认值为standby1。如果因为重名导致同步复制或半同步复制选择的节点并不是RDS内的备实例，发生主备切换时，RDS将不予保证数据0丢失。
最高可用性	半同步模式	兼顾数据持久性和响应速度。	• synchronous_commit = remote_write • rds_sync_replication_timeout > 0 说明 rds_sync_replication_timeout的值表示数据同步的超时时长，取值范围为0~300000，单位为ms，建议设置为1000。如果同步超时，会降级为最高性能级别（异步模式），等数据同步后，再次回到高可用级别（半同步模式）。 • synchronous_standby_names = standby1 警告 如果您有自建的从库，请不要命名为standby1，因为synchronous_standby_names默认值为standby1。如果因为重名导致同步复制或半同步复制选择的节点并不是RDS内的备实例，发生主备切换时，RDS将不予保证数据0丢失。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**参数设置**。
3. 在**可修改参数**标签页，找到参数synchronous_commit、rds_sync_replication_timeout和synchronous_standby_names。
4. 单击

修改参数值并单击**确定**。

> **说明** 参数值如何填写请参见上文背景信息部分。

5. 在页面右上角单击**提交参数**。

> **警告** 修改参数rds_sync_replication_timeout，提交时会重启实例造成连接中断，请在业务低峰期进行操作。

13.11. 实例回收站

RDS PostgreSQL实例进入回收站后，您可以在回收站中解锁、重建或销毁实例。

功能说明

RDS PostgreSQL实例在手动释放、到期或退款后均会进入回收站。但如下情况除外：

- 退款或手动释放已创建未超过7天的实例。
- 因阿里云账号欠费被系统自动释放的按量付费实例。
- 实例为只读实例。

充值解锁

按量付费RDS实例因为欠费被锁定时，请检查阿里云账号的支付方式。

续费解锁

包年包月RDS实例因为到期被锁定后，15天内可以在回收站中对实例进行续费解锁，超期则被释放。

1. 登录RDS管理控制台。
2. 在左侧导航栏单击**回收站**，然后在上方选择地域。
3. 找到被锁定的实例，单击**续费解锁**为实例续费。

续费后，实例会立即恢复正常。

重建实例

实例进入回收站后，数据备份将继续保留8天，在这8天里，您可以通过回收站的重建实例功能，将数据恢复到一个新实例。具体规则请参见到期或欠费。

1. 登录回收站。
2. 单击**实例列表**，然后在上方选择地域。
3. 找到目标实例，单击**重建实例**。

默认会在原实例所在的可用区创建相同规格的实例，您也可以选择其它可用区及其它实例规格。

> **说明** RDS PostgreSQL 9.4版本已停售，不支持通过此功能重建实例。更多信息，请参见【通知】2021年07月31日停止售卖RDS PostgreSQL 9.4版本。

销毁实例

RDS实例进入回收站后，您可以在回收站中销毁该实例。

> **警告** 销毁实例会同时销毁除 跨地域备份外的所有备份，包括常规数据备份、归档备份、日志备份，请谨慎操作。

1. 登录回收站。
2. 单击**实例列表**，然后在上方选择 地域 。
3. 找到目标实例，单击**操作**列的**立即销毁**。

相关文档

- 到期或欠费
- 释放实例

14.只读实例

14.1. PostgreSQL只读实例简介

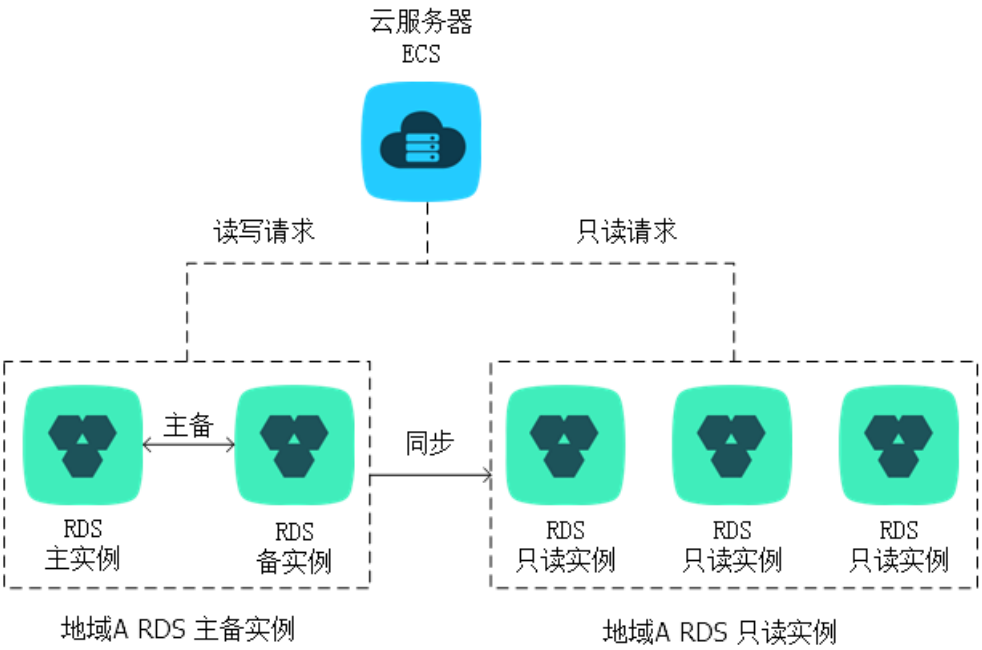
在对数据库有少量写请求，但有大量读请求的应用场景下，单个实例可能无法承受读取压力，甚至对业务产生影响。为了实现读取能力的弹性扩展，分担数据库压力，您可以创建一个或多个只读实例，利用只读实例满足大量的数据库读取需求，增加应用的吞吐量。

简介

创建只读实例时会从备实例复制数据，数据与主实例一致，主实例的数据更新也会在主实例完成操作后立即自动同步到所有只读实例。

> **说明**
>
> - 本地盘主实例最多创建5个只读实例，云盘主实例最多创建32个只读实例。
> - 本地盘实例的只读实例为高可用架构。
> - 云盘实例的只读实例为单节点架构，没有备节点，因此无法保障可用性，建议您购买多个只读实例，使用libpq或JDBC实现自动故障转移，详情请参见自动故障转移和读写分离。

只读实例拓扑图如下图所示。



计费

支持包年包月和按量付费两种方式计费。包年包月费用请以实际购买页为准，按量付费具体费用请参见只读实例规格列表。

功能特点

- 地域和可用区：与主实例在同一地域，可以在不同的可用区。
- 规格和存储空间：本地盘主实例的只读实例的规格和存储空间不能低于主实例。
- 切换网络类型：可以与主实例不一致。
- 账号与数据库管理：不需要维护账号与数据库，全部通过主实例同步。
- 白名单：只读实例创建时会自动复制其主实例的白名单信息，但只读实例和主实例的白名单是相互独立的。若您需要修改只读实例的白名单，请参见设置白名单。
- 监控与报警：提供系统性能指标的监控视图，如磁盘容量、IOPS、连接数、CPU使用率等。

注意事项

- 只读实例的数量：本地盘主实例最多创建5个只读实例，云盘主实例最多创建32个只读实例。
- 实例备份：因主实例已有备份，只读实例暂不支持备份设置以及手动发起备份。
- 数据迁移：不支持将数据迁移至只读实例。
- 数据库管理：不支持创建和删除数据库。
- 账号管理：不支持创建和删除账号，不支持为账号授权以及修改账号密码功能。
- 主实例变配时，如果主实例内存大于只读实例内存，会重启只读实例。
- 只读实例出现内核复制或其他不可预见的错误时，会重搭该只读实例。

常见问题

主实例上创建的账号在只读实例上可以用吗？

答：主实例创建的账号会同步到只读实例，只读实例无法管理账号。账号在只读实例上只能进行读操作，不能进行写操作。

14.2. 创建PostgreSQL只读实例

您可以通过创建只读实例满足大量的数据库读取需求，增加应用的吞吐量。创建只读实例相当于复制了一个主实例，数据与主实例一致，主实例的数据更新也会自动同步到所有只读实例。
关于只读实例的更多介绍，请参见PostgreSQL只读实例简介。

前提条件

- 实例版本为PostgreSQL 10、11、12、13或14。
- 本地盘主实例的规格不低于8核32G（独享套餐），云盘版实例 无规格要求 。
- 实例系列为高可用版。创建PostgreSQL只读实例前，请在**基本信息**页面确认实例系列。如果是基础版，且需要创建只读实例时，请点击**变更配置**，将基础版变更为高可用版后再创建只读实例，具体操作，请参见变更配置。

注意事项

- 只能在主实例内创建只读实例，不能将已有实例切换为只读实例。
- 由于创建只读实例时是从备实例复制数据，因此不会影响主实例。
- 只读实例的参数不继承主实例上的参数设置，会生成默认的参数值，可以在只读实例的控制台上进行修改。
- 本地盘只读实例的规格和存储空间不能低于主实例。
- 云盘版只读实例的规格建议与主实例保持一致，或大于等于主实例规格的1/2，避免只读实例性能与主实例相差过大而导致的复制延迟、OOM等情况。
- 本地盘主实例最多创建5个只读实例，云盘主实例最多创建32个只读实例。
- 本地盘实例的只读实例为高可用架构，云盘实例的只读实例为单节点架构（没有备节点）。

> **说明** 单节点架构没有备节点，因此无法保障可用性，建议您购买多个只读实例，使用libpq或JDBC实现自动故障转移，详情请参见 自动故障转移和读写分离。

- 创建只读实例会产生费用，支持包年包月和按量付费两种方式计费。包年包月费用请以实际购买页为准，按量付费具体费用请参见只读实例规格列表。

创建只读实例

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在**实例分布**区域找到**只读实例**，单击**添加**。

> **说明** 旧版控制台请在右侧单击 **添加只读实例**。

3. 设置只读实例的参数。

参数	说明
计费方式	○ **包年包月**：长期使用RDS只读实例推荐使用包年包月（一次性付费），在页面左下角选择购买时长 ○ **按量付费**：短期使用RDS只读实例，请选择按量付费（按小时付费）。您可以先创建按量付费的只读实例，确认实例符合要求后再转包年包月。
可用区	可用区是地域中的一个独立物理区域，不同可用区之间没有实质性区别。相比单可用区，多可用区能提供可用区级别的容灾。
实例规格	○ **通用规格（入门级）**：通用型的实例规格，独享被分配的内存和I/O资源，与同一服务器上的其他通用型实例共享CPU和存储资源。 ○ **独享规格（企业级）**：独享或独占型的实例规格。独享型指独享被分配的CPU、内存、存储和I/O资源。独占型是独享型的顶配，独占整台服务器的CPU、内存、存储和I/O资源。 **说明** 每种规格都有对应的CPU核数、内存、最大连接数和最大IOPS。详情请参见 RDS PostgreSQL只读实例规格列表。

参数	说明
存储空间	存储空间包括数据空间、系统文件空间、Binlog文件空间和事务文件空间。调整存储空间时最小单位为5GB。 **说明** 本地SSD盘的独享套餐等规格由于资源独享的原因，存储空间大小和实例规格绑定。详情请参见 RDS PostgreSQL只读实例规格列表。

4. 单击**下一步：实例配置**，设置如下参数。

参数	说明
网络类型	○ **经典网络**：传统的网络类型。 ○ **专有网络**（推荐）：也称为VPC（Virtual Private Cloud）。VPC是一种隔离的网络环境，安全性和性能均高于传统的经典网络。选择专有网络时您需要选择对应的**VPC**和**主节点交换机**，如果您在上一步的**基础资源**中配置了**多可用区部署**，则还需要选择**备选节点交换机**。 **说明** 请确保RDS实例与需要连接的ECS实例网络类型一致（如果选择专有网络，还需要保证VPC一致），否则它们无法通过内网互通。

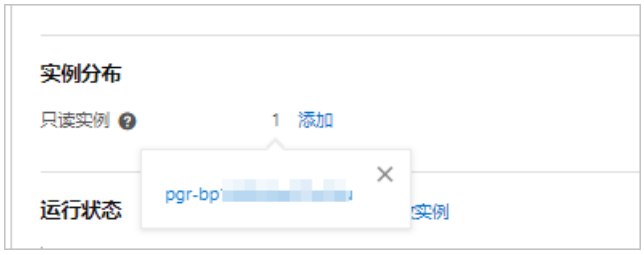
5. 单击**下一步：确认订单**。
6. 勾选服务协议，单击**去支付**，根据提示完成支付。

只读实例创建时间与主实例存储类型及磁盘大小相关，请参考以下预估时间，耐心等待只读实例创建。创建期间不会影响主实例。

- 主实例存储类型为本地SSD盘，创建时间约为一次全量备份时间+10分钟。
- 主实例存储类型为SSD云盘，创建时间约为一次全量备份时间+20分钟。
- 主实例存储类型为ESSD云盘，创建时间约为20分钟。

查看只读实例

- 在实例列表中查看只读实例
  i. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
  ii. 在实例列表中找到只读实例，单击该只读实例的ID。
- 在主实例的基本信息页面查看只读实例
  i. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
  ii. 在主实例的**基本信息**页面，把鼠标悬停于只读实例的数量上，单击只读实例的ID。



查看只读实例的延迟时间

只读实例同步主实例的数据时，可能会有一定的延迟。您可以在只读实例的基本信息页面查看延迟时间。

相关API

API	描述
创建只读实例	创建RDS只读实例

15.账号

15.1. 创建账号

本文介绍如何为RDS PostgreSQL实例创建账号。

账号类型

RDS PostgreSQL实例支持两种数据库账号：高权限账号和普通账号。详细说明如下。

账号类型	说明
高权限账号	• 只能通过控制台或API创建和管理。 • 可以创建多个高权限账号，管理所有普通账号和数据库。 • 开放了更多权限，可满足个性化和精细化的权限管理需求，例如可按用户分配不同表的查询权限。 • 可以断开任意账号的连接。 **说明** • 当创建的高权限账号是该实例的第一个高权限账号时，这个账号将是标准系统数据库template1中默认模式Public Schema的Owner。 • CREATE DATABASE命令默认通过复制template1来创建数据库。通过该方式创建的数据库，Public Schema的Owner均为第一个高权限账号。 • 第一个高权限账号的Comment开头会包含说明：template1 public schema owner。
普通账号	• 可以通过控制台、API或者SQL语句创建和管理。 • 一个实例可以创建多个普通账号 。 • 需要手动给普通账号授予特定数据库的权限。 • 普通账号不能创建和管理其他账号，也不能断开其他账号的连接。

注意事项

- 支持在控制台创建多个高权限账号和普通账号，也可以通过SQL命令创建、管理普通账号。
- 如果您要迁移本地数据库到RDS，请在RDS实例中创建与本地数据库一致的迁移账号和数据库。
- 分配数据库账号权限时，请按最小权限原则和业务角色创建账号，并合理分配只读和读写权限。必要时可以把数据库账号和数据库拆分成更小粒度，使每个数据库账号只能访问其业务之内的数据。如果不需要数据库写入操作，请分配只读权限。
- 为保障数据库的安全，请将数据库账号的密码设置为强密码，并定期更换。

创建账号

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中选择**账号管理**。
3. 单击**创建账号**。
4. 设置如下参数。

参数	说明
数据库账号：	○ 长度为2~63个字符。 ○ 由小写字母、数字或下划线组成。 ○ 以字母开头，以字母或数字结尾。 ○ 不能和已有的账号名重复。 ○ 不能以pg开头。 ○ 不能使用SQL关键字。具体请参见SQL关键字。
账号类型：	RDS PostgreSQL实例支持两种数据库账号：高权限账号和普通账号。 ○ 高权限账号拥有所有数据库的所有操作权限。 ○ 普通账号拥有已授权数据库（owner）的所有操作权限。 **说明** 操作权限包括SELECT、INSERT、UPDATE、DELETE、TRUNCATE、REFERENCES、TRIGGER。
密码：	○ 长度为8~32个字符。 ○ 由大写字母、小写字母、数字、特殊字符中的任意三种组成。 ○ 特殊字符为!@#$%^&*()_+-=
确认密码：	再次输入相同的密码。
备注说明：	填写备注信息。

5. 单击**确定**。

相关API

API	描述
CreateAccount	创建RDS实例的数据库账号。

15.2. 重置密码

在使用RDS过程中，如果忘记数据库账号密码，可以通过控制台重新设置密码。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**账号管理**。
3. 找到要重置密码的账号，单击**重置密码**。

4. 在弹出的对话框中输入新密码并确认，然后单击**确定**。

   密码要求如下：

   - 长度为8~32个字符。
   - 由大写字母、小写字母、数字、特殊字符中的任意三种组成。
   - 特殊字符为!@#$%^&*()_+-=

> **说明** 现在您可以为RDS MySQL 5.7版本实例定制灵活的数据库密码策略，更多信息，请参见自定义密码策略。

相关API

API	描述
ResetAccountPassword	重置RDS实例账号的密码。

15.3. 授权服务账号

当您寻求阿里云的技术支持时，如果技术支持过程中需要对您的数据库实例进行操作，您需要对服务账号授权，技术支持人员才可以通过服务账号提供技术支持服务。在授权有效期结束后，临时服务账号会被自动删除。

前提条件

实例为RDS PostgreSQL云盘版。

授权操作

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**账号管理**。
3. 选择**服务授权账号**页签，针对要授予服务账号的权限，单击**权限开通**列下的滑块。
   - 如果是IP白名单、数据库参数等问题排查，只需要授权**配置权限**。
   - 如果是应用导致的数据库性能问题，则需要授权**数据权限**。

4. 在弹出的对话框中，设置授权过期时间，单击**确定**。

取消授权或修改授权到期时间

给服务账号授权后，您可以在**服务授权账号**页签随时取消服务账号的授权或者修改授权到期时间。

15.4. 锁定/删除账号

RDS PostgreSQL实例支持锁定账号或删除账号。

通过控制台锁定账号

> **说明** 账号锁定后，您无法再通过该账号登录实例。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**账号管理**。
3. 找到要锁定的账号，单击其右侧**操作**列中的**锁定**。

   > **说明** 单击 **锁定**后等待几秒， **状态**列显示 **锁定中**即表示账号已锁定，如果需要继续使用该账号，请单击 **解锁**。

通过控制台删除账号

> **说明** 如果要删除的账号有数据库、表或其他对象的权限，执行删除账号操作会报错 `Some objects depend on account` ，取消相关权限后才能正常删除账号。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**账号管理**。
3. 找到要删除的账号，单击其右侧**操作**列中的**删除**。
4. 在弹出的确认框中，单击**确定**。

通过SQL命令删除普通账号

> **说明** 如果要删除的账号有数据库、表或其他对象的权限，执行删除账号操作会报错 `ERROR: current user cannot be dropped` ，取消相关权限后才能正常删除账号。

1. 通过DMS登录RDS实例，详细步骤请参见通过DMS登录RDS数据库。
2. 登录成功后，在页面上方的菜单栏中，选择**SQL窗口**。
3. 执行如下命令将要删除账号的相关权限分配给其他账号。

```
REASSIGN OWNED BY <要删除的账号名称> TO <其他账号名称>;
```

4. 执行如下命令删除账号：

```
DROP USER <要删除的账号名称>;
```

5. 单击**执行**。

相关API

API	描述
删除数据库账号	删除账号
锁定账号	锁定账号
解锁账号	解锁账号

15.5. 账号权限详情

本文介绍RDS PostgreSQL高权限账号和普通账号的权限。

权限	高权限账号	普通权限账号
CREATEDB	✔	▯
CREATEROLE	✔	▯
REPLICATION	✔	▯
SELECT	✔	✔
INSERT	✔	✔
UPDATE	✔	✔
DELETE	✔	✔
TRUNCATE	✔	✔

权限	高权限账号	普通权限账号
REFERENCES	✔☺	✔☺
TRIGGER	✔☺	✔☺
CREATE	✔☺	✔☺
CONNECT	✔☺	✔☺
TEMPORARY	✔☺	✔☺
EXECUTE	✔☺	✔☺
USAGE	✔☺	✔☺

15.6. RDS PostgreSQL接入自建域

本文介绍如何配置ECS实例的AD域控制器，以及如何将RDS PostgreSQL实例接入域。

背景信息

活动目录AD（Active Directory），是微软提供的目录服务。目录是一种分层结构，用于存储同一局域网络上对象的信息。在目录中可以收录公司的电脑账号、用户账号、组等以提供更好的安全性和更便捷的管理能力。

RDS PostgreSQL提供接入自建AD域功能，用于企业中心化管理，同时提升用户访问安全性，支持数据库级别和用户级别的黑白名单能力。

> **说明** RDS PostgreSQL对AD域服务信息的修改是通过修改RDS PostgreSQL实例上 *pg_hba.conf*文件实现的，修改或导入AD域服务信息的内容即修改实例 的 *pg_hba.conf*文件，所以RDS管理控制台除了可以配置AD域服务之外，还支持其他常用的pg_hba.conf文件配置，具体参考： PostgreSQL pg_hba.conf介绍。

前提条件

- RDS PostgreSQL实例版本需要满足以下条件。
  - 实例大版本：10、11、12、13或14。
  - 内核小版本：大于等于20210228，如需升级内核小版本，请参见升级内核小版本。
  - 存储类型：云盘存储。
- 已创建ECS实例。RDS PostgreSQL接入自建域必须使用内网，ECS需要满足以下条件：
  - AD域控制器所在ECS与目标RDS PostgreSQL实例处于相同VPC。
  - ECS安全组放通RDS的内网IP。详情请参见添加安全组规则。
  - ECS实例系统防火墙默认关闭，如果您开启过，ECS实例系统防火墙需要放通RDS的内网IP。
  - 系统镜像选择Windows Server 2016及以上版本。
- 域账号属于Domain Admins组。
- 登录的阿里云账号为主账号。

操作步骤

1. ECS实例配置AD域控制器。
   i. 远程登录ECS的Windows Server 2016系统。

   > **说明** AD域控制器需要建立在Windows Server操作系统之上，建议使用Windows Server 2016及以上版本，本示例以Windows Server 2016为例，建立可供RDS使用的AD域控制器。

   ii. 搜索**服务器管理器**并打开。
   iii. 在**仪表盘**中单击**添加角色和功能**。

   iv. 在**添加角色和功能向导**窗口中进行如下配置。

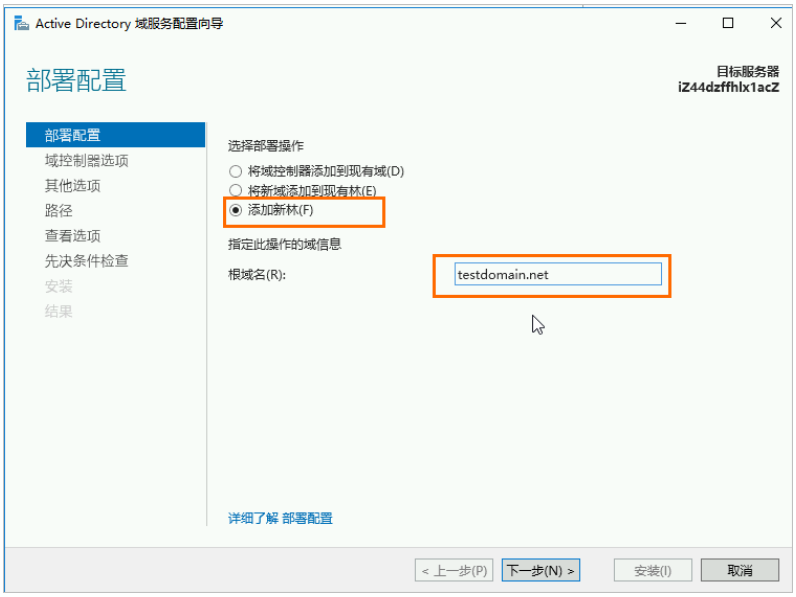
向导页签	设置说明
开始之前	保持默认配置。
安装类型	保持默认配置。
服务器选择	保持默认配置。
服务器角色	■ 选中**Active Directory 域服务**，并在弹出的对话框中单击**添加功能**。 ■ 选中**DNS 服务器**，并在弹出的对话框中单击**添加功能**。 **说明** 如果提示您电脑不是固定IP，建议您修改电脑为固定IP，防止IP自动变更导致DNS服务器无法使用。
功能	保持默认配置。
AD DS	保持默认配置。
DNS 服务器	保持默认配置。
确认	单击**安装**按钮开始安装服务。

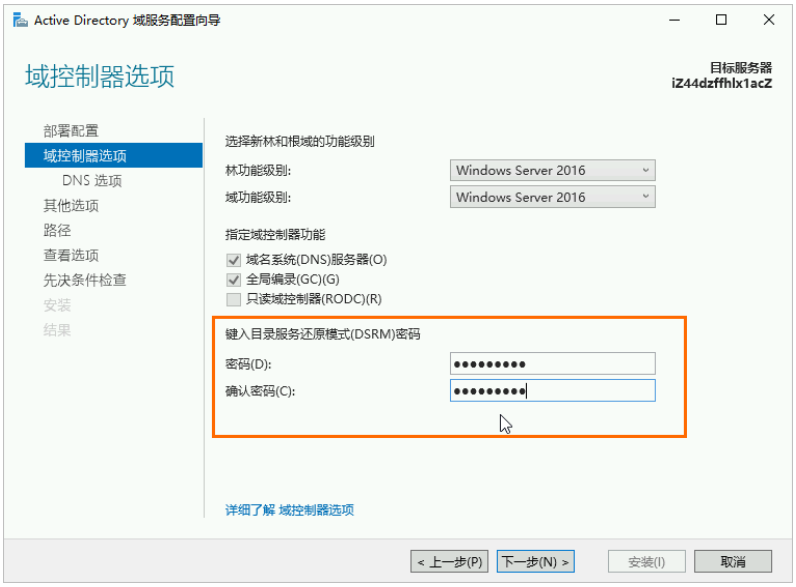
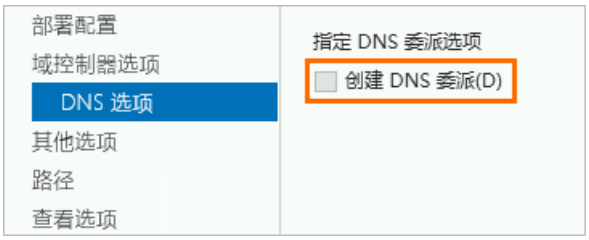
   v. 等待安装完成后，单击**关闭**按钮关闭配置向导。

vi. 在**服务器管理器**页面左侧导航栏单击**AD DS**，然后在右上方单击**更多**。

vii. 在**所有服务器 任务详细信息**窗口中单击**将此服务器提升为域控制器**。

viii. 在**Active Directory 域服务配置向导**窗口中进行如下配置。

向导页签	设置说明
部署配置	选择**添加新林**，设置**根域名**，本示例以 `pgsqldomain.net` 为例。 

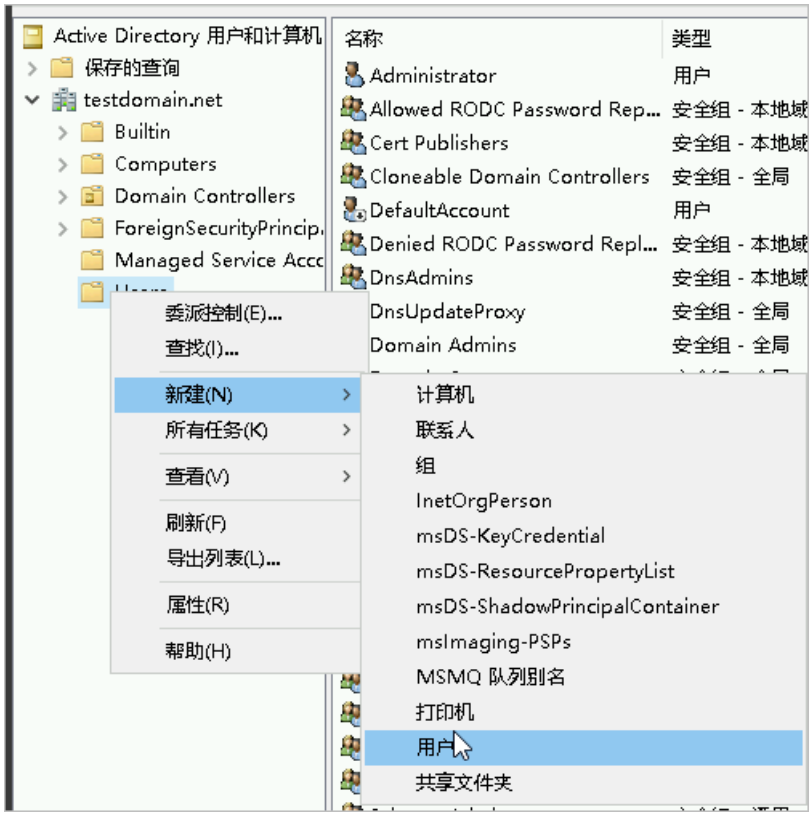
向导页签	设置说明
域控制器选项	设置**目录服务还原模式(DSRM)密码**。 
服务器选择	取消选中**创建 DNS 委派**。 
其他选项	保持默认配置。
路径	保持默认配置。
查看选项	保持默认配置。
先决条件检查	单击**安装**按钮开始安装服务。

> **说明** 设置完成后需要重启ECS服务器，请等待重启完成后再进行后续步骤。

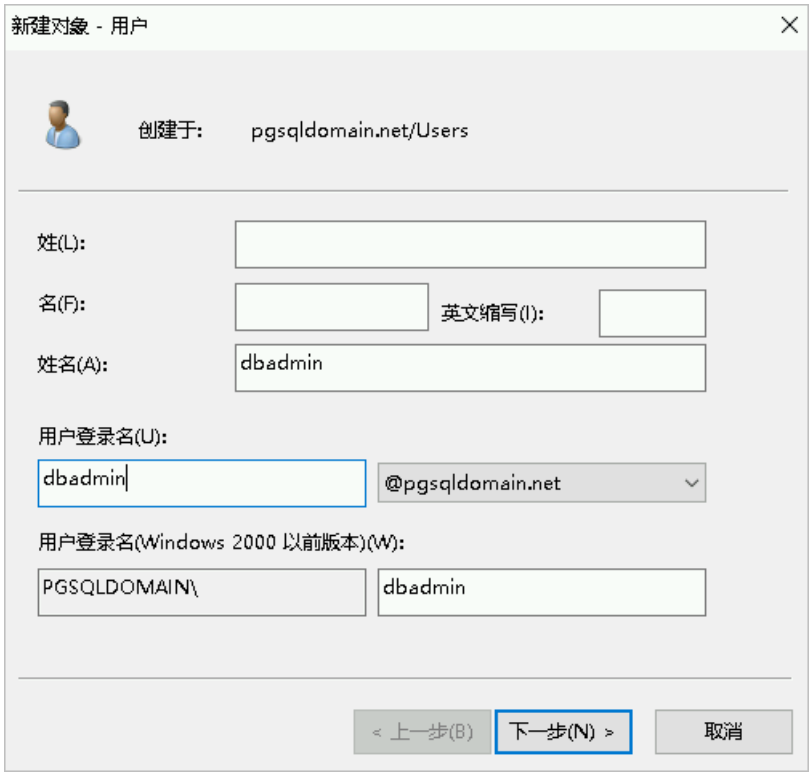
2. AD域控制器中添加管理员用户。
   i. 远程登录ECS服务器，搜索**服务器管理器**并打开。

ii. 在**服务器管理器**左侧导航栏单击**AD DS**，然后在右侧目标域控制器上单击右键，选择**Active Directory 用户和计算机**。

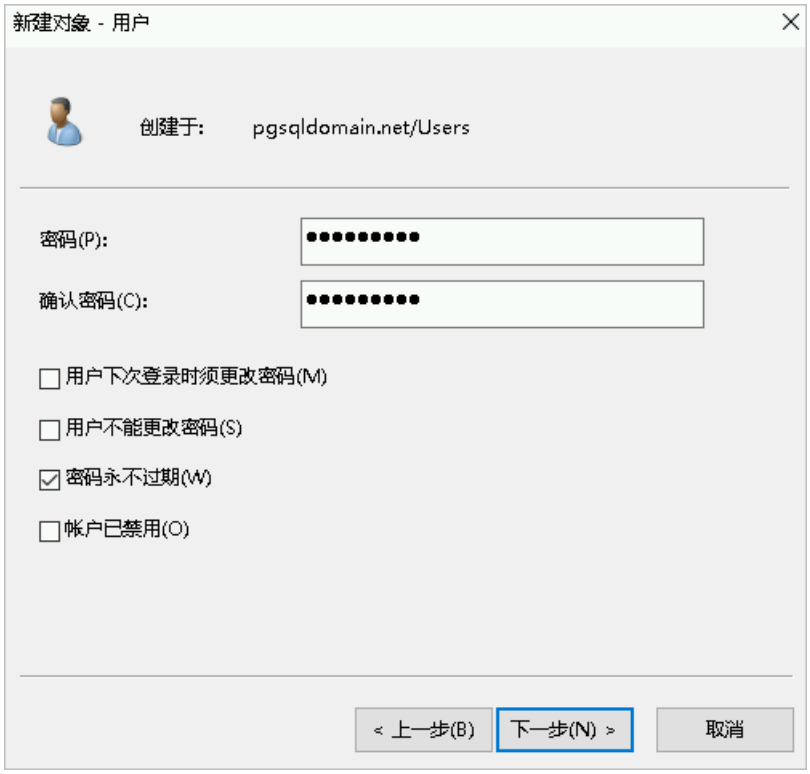
iii. 在**pgsqldomain.net > Users**上单击右键，选择**新建 > 用户**。



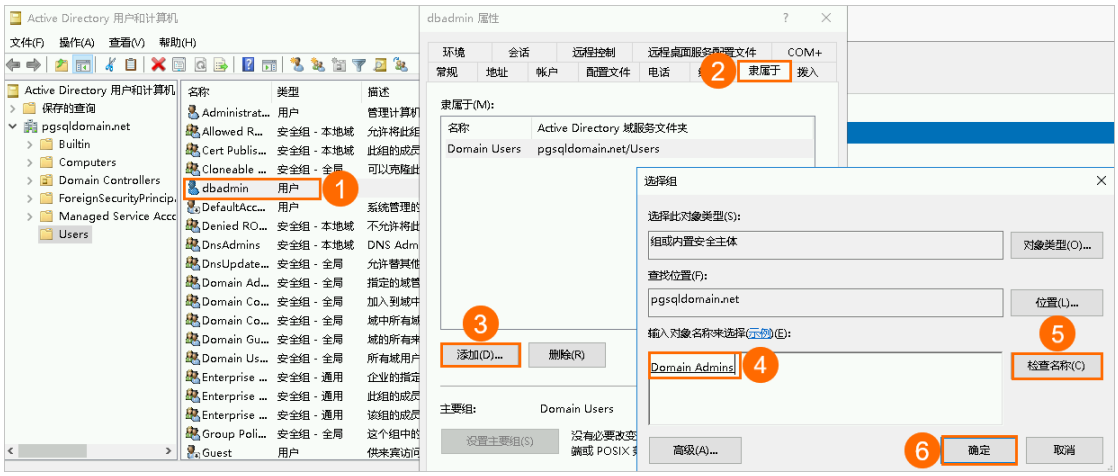
iv. 设置登录的用户名称，单后单击**下一步**。



v. 设置登录密码，勾选**密码永不过期**，最后单击**下一步**及**完成**。



vi. 双击新创建的用户，将用户加入Domain Admins管理员组中。



添加后结果如下：

3. AD域控制器中添加普通用户用于登录。

> **说明** 添加普通用户步骤与 域控制器中添加管理员用户方法相同，普通登录用户不需要加入Domain Admins管理员组。
> 本示例以 `ldapuser` 用户为例，作为AD域控制器中的普通用户，用于登录RDS PostgreSQL实例。

4. 配置ECS实例安全组规则。
   i. 登录ECS管理控制台。
   ii. 在左侧导航栏，选择**实例与镜像 > 实例**。
   iii. 在顶部菜单栏左上角处，选择地域。
   iv. 在**实例列表**页面中，单击目标实例ID。
   v. 在左侧导航栏单击**本实例安全组**，然后在右侧单击**配置规则**。

   > **说明** 域控制器需要开放较多端口，因此不建议和其他ECS实例共享安全组，建议创建单独的安全组使用。

   vi. 在**入方向**页签内单击**手动添加**，允许如下端口访问ECS实例。

协议类型	端口范围	说明
TCP	88	Kerberos认证协议端口。
TCP	135	远程过程调用协议（RPC）端口。
TCP/UDP	389	轻型目录访问协议（LDAP）端口。
TCP	445	通用互联网文档系统协议（CIFS）端口。
TCP	3268	Global Catalog端口。
TCP/UDP	53	DNS端口。
TCP	49152~65535	连接的默认动态端口范围。输入格式为：49152/65535。

5. 配置RDS实例。
   i. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。

ii. 创建名为 `ldapuser` 的 用户账号 。具体操作请参见创建账号。

> **说明** RDS管理控制台创建的账号名需要与AD域中创建的普通账号名相同，密码可以不一样，当启用AD域控制用户访问时，校验AD域控制器中该账号对应的密码，当不启用AD域控制用户访问时，系统校验RDS管理控制台的 **账号管理** 中设置的用户密码。

iii. 单击左侧导航栏的**账号管理**，选择**AD域服务信息**页签。

首次打开**AD域服务信息**页签，系统默认创建了以下两条记录：

```
host all all 0.0.0.0/0 md5 host replication all 0.0.0.0/0 md5
```

该默认配置可以删除或修改。

iv. 单击第一条默认记录后的**编辑**，修改第一条默认记录，填写如下参数取值。

> **说明** 下表中仅简单介绍本示例使用的取值样例及含义，更多参数取值含义及配置方法请参见 PostgreSQL官方文档。

参数	取值示例	说明
优先级	0	表示该条记录的优先级，0优先级最高，系统自动生成。修改第一条默认记录，优先级为0，表示AD域服务的配置优先级最高。
TYPE	host	支持配置以下取值： ■ host：该条记录验证TCP/IP连接，包括SSL连接和非SSL连接。 ■ hostssl：该条记录只验证通过SSL建立的TCP/IP连接。 **说明** 要使用此参数，请打开SSL链路加密，具体请参见 SSL链路加密。 ■ hostnossl：该条记录只验证通过非SSL建立的TCP/IP连接。
DATABASE	all	允许用户访问的数据库，all表示允许用户访问所有数据库。如果配置多个，可通过逗号（,）分隔。

参数	取值示例	说明
USER	ldapuser	允许哪些用户访问数据库，填写AD域中创建的用户名。如果配置多个，可通过逗号（,）分隔。 **说明** 配置在AD域中创建的普通用户的用户名。
ADDRESS	0.0.0.0/0	允许用户从哪个或哪些IP访问数据库，0.0.0.0/0表示允许用户从任意IP地址访问数据库。
MASK	空	掩码。如果 ADDRESS 为IP地址，可以通过此参数指定IP地址的掩码。
METHOD	ldap **说明** LDAP（Lightweight Directory Access Protocol，轻量目录访问协议）是设计用来访问目录数据库的一个协议，本示例以LDAP为例。	认证方法，支持： ■ trust ■ reject ■ scram-sha-256 ■ md5 ■ password ■ gss ■ sspi ■ ldap ■ radius ■ cert ■ pam **说明** 手动输入时请注意此处使用英文小写字母。
OPTION	ldapserver=*<ECS服务器的私网IP>* ldapbasedn="CN=Users,DC=pgsqldomain,DC=net" ldapbinddn="CN=*<AD域中创建的管理员用户名>*,CN=Users,DC=pgsqldomain,DC=net" ldapbindpasswd="*<AD域中创建的管理员用户密码>*" ldapsearchattribute="sAMAccountName"	认证方式对应的可选参数。本示例使用LDAP认证，需要配置。此参数的更多解释，请参见Authentication Methods。

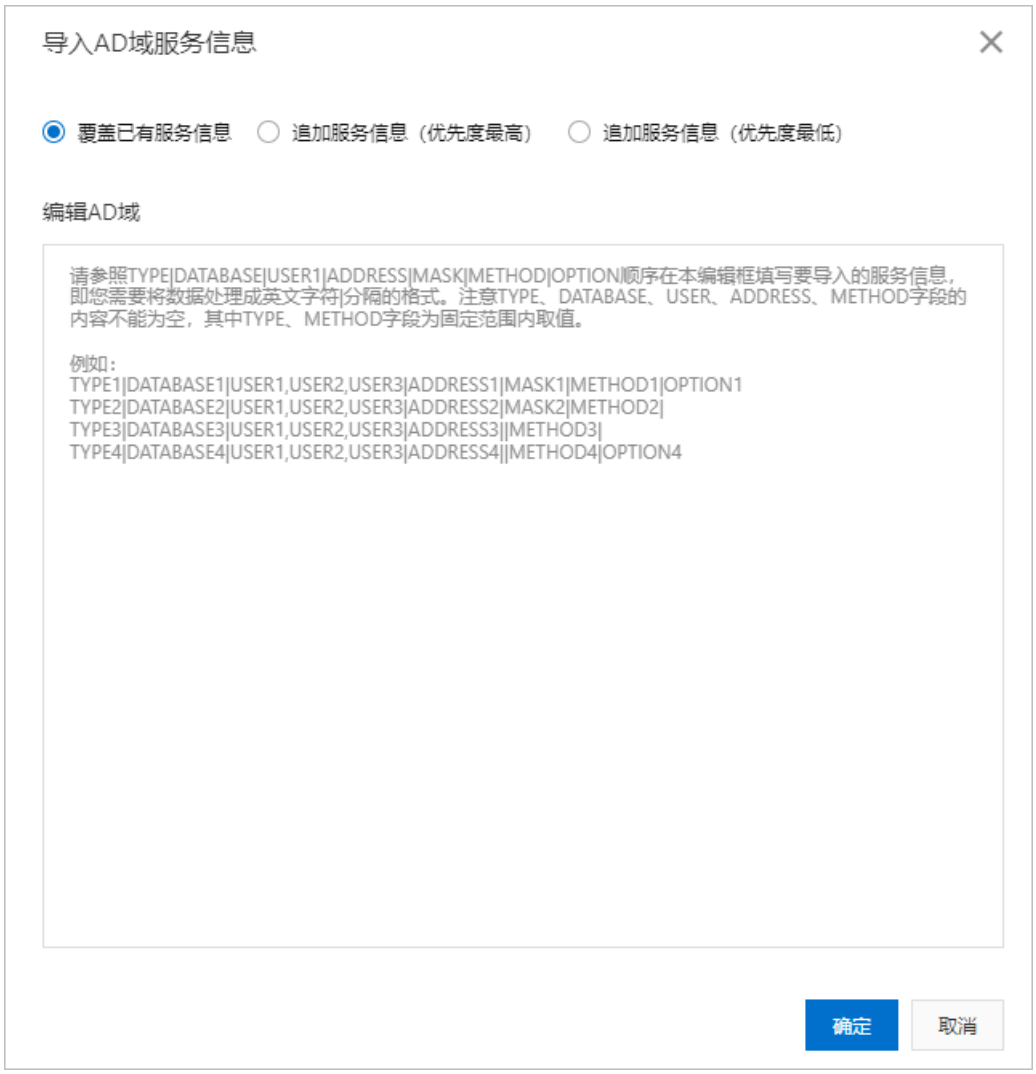
v. 在上一步配置的AD域服务记录后单击**添加**，新增如下记录。

```
host all all 0.0.0.0/0 md5
```

vi. 填写完成后单击**确定**，最后单击**提交按钮**。

> **说明** 提交后系统状态将变为 **维护实例中**，预计时长约1分钟， 修改后的新配置仅对新连接生效，对于旧连接，需要您断开已有连接后重新连接才能生效。

6. （可选）除手动添加AD域服务信息配置外，RDS PostgreSQL还支持通过导入方式批量设置AD域服务信息。



导入方式支持以下三总种模式

- 覆盖已有服务信息
- 追加服务信息（优先度最高）：在已有服务信息前追加AD域记录，优先级高于已有服务信息。
- 追加服务信息（优先度最低）：在已有服务信息最后追加AD域记录，优先级低于已有服务信息。

配置格式：

```
TYPE|DATABASE|USER1|ADDRESS|MASK|METHOD|OPTION
```

请在**编辑AD域**的编辑框中按顺序填写需要导入的服务信息。各参数取值请参见参数说明。
配置示例：

```
host|all|<AD域中创建的普通用户的用户名>|0.0.0.0/0||ldap|ldapserver=<ECS服务器的私网IP> ldapb
asedn="CN=Users,DC=pgsqldomain,DC=net" ldapbinddn="CN=<AD域中创建的管理员用户名>,CN=Users,
DC=pgsqldomain,DC=net" ldapbindpasswd="<AD域中创建的管理员用户密码>" ldapsearchattribute="
sAMAccountName"
```

7. 测试连接。

   使用PostgreSQL命令行终端连接RDS。

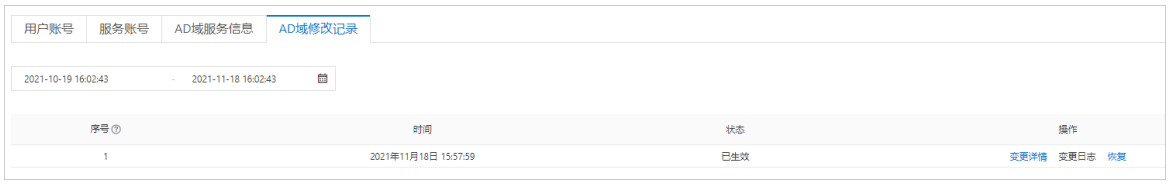
   > **说明** 连接RDS PostgreSQL数据库可以通过多种方式，本示例以PostgreSQL命令行终端为例，连接前需要安装PostgreSQL客户端，具体方法及其他连接方式，请参见 连接PostgreSQL实例。

   使用如下命令连接RDS PostgreSQL数据库，使用在AD域控制器中普通用户的用户名及密码。

```
psql -h <实例连接地址> -U ldapuser -p 5432 -d postgres
```

查看AD域修改记录

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 单击左侧导航栏的**账号管理**，选择**AD域修改记录**页签。
3. 您可以在**操作列**查看**变更详情**，当修改失败时，状态为**未生效**，您可以单击**变更日志**，查看错误信息。



16.数据库连接

16.1. 连接PostgreSQL实例

您可以通过数据管理 DMS 平台、PostgreSQL命令行终端工具、pgAdmin客户端或应用程序等方式连接PostgreSQL实例，实现业务目标。本文介绍这些连接方法的具体操作。

前提条件

已完成以下操作：

- 创建RDS PostgreSQL实例
- 创建账号
- 设置白名单
- ECS要通过内网访问RDS，必须满足以下所有条件：
  - ECS和RDS属于同一个阿里云主账号。
  - ECS和RDS位于同一个地域。
  - ECS和RDS必须处于同一个VPC。
  - ECS的私网IP已添加到RDS白名单，具体操作请参见设置白名单。

通过DMS连接

您可以在**基本信息**页面单击**登录数据库**通过DMS链接数据库。

具体操作请参见通过DMS登录RDS数据库。

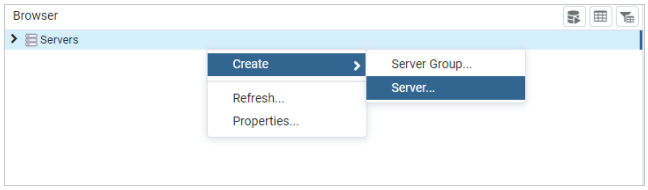
通过pgAdmin客户端连接

通过PostgreSQL官方网站下载 并安装PostgreSQL客户端时，将会自动安装 **pgAdmin 4**客户端，您也可以单独下载pgAdmin客户端。

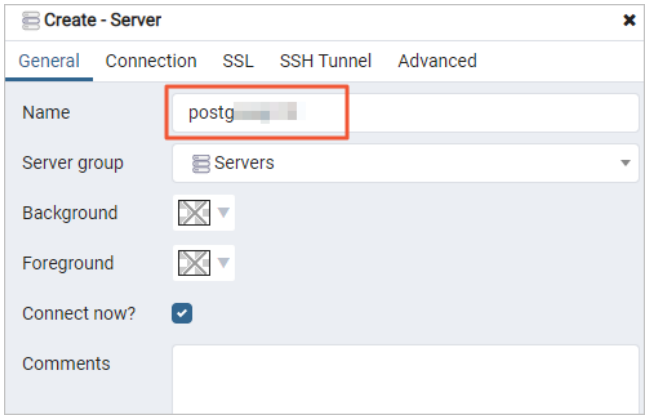
1. 启动**pgAdmin 4**客户端。

   > **说明** 高版本客户端首次登录需要设置Master Password用于保护保存的密码和其他凭据。

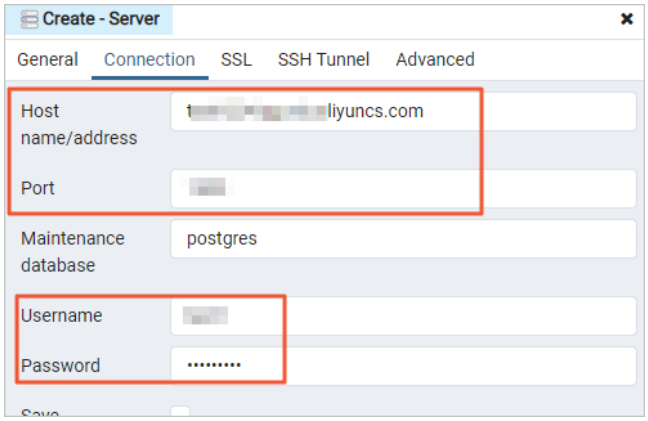
2. 右键单击 **Servers**，选择**Create > Server...**。



3. 在**General**页签设置连接名称。



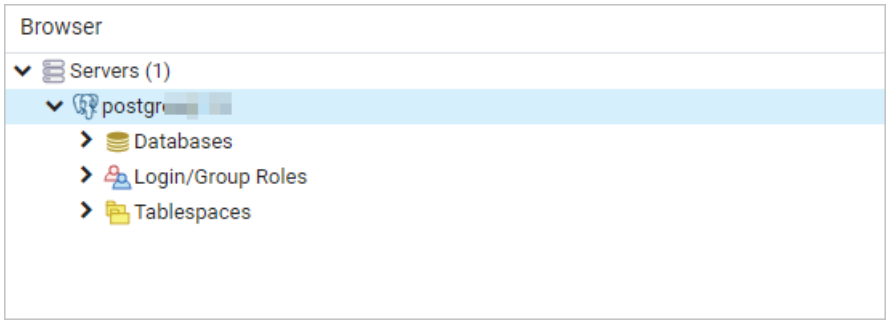
4. 选择Connection标签页，输入要连接的实例信息。



参数	说明
Hostname/address	主机名称/地址。若通过内网连接，需输入RDS实例的内网连接地址；若使用外网连接，需输入RDS实例的外网连接地址。查看RDS实例的内外网地址及端口请参见查看或修改内外网地址和端口。
Port	连接地址对应的端口。
Username	RDS实例的账号名称。创建RDS实例的账号请参见创建数据库和账号。
Password	账号对应的密码。

5. 单击Save。
若连 接信息无误，会出现如下界面，则表示连 接成功。

> **注意** postgres是RDS实例默认的系统数据库，请勿在该数据库中进行任何操作。



通过PostgreSQL客户端命令行工具连接

通过PostgreSQL官方网站下载并安装PostgreSQL客户端时，将会自动安装**PostgreSQL命令行终端工具（Command Line Tools）**。

在命令行终端中输入如下命令连接RDS PostgreSQL数据库。

```
psql -h <实例连接地址> -U <用户名> -p <端口号> -d <数据库名>
```

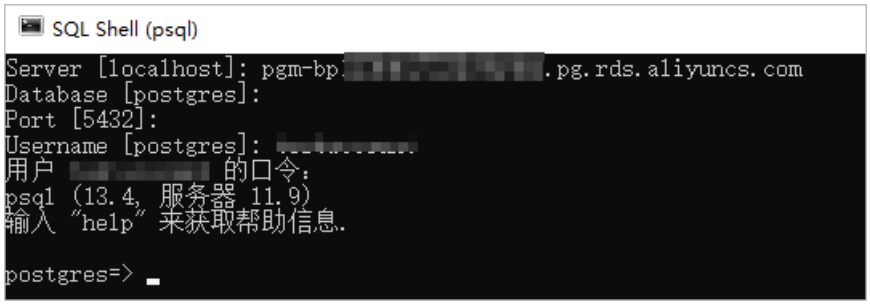
各参数在RDS控制台的获取方法。

参数	获取方法
实例连接地址	RDS PostgreSQL实例的连接地址，具体请参见查看或修改内外网地址和端口。
用户名	通过**账号管理**页面获取用户名。创建账号，请参见创建账号。
端口号	默认为5432，如果您修改过，通过**数据库连接**页面获取。具体请参见查看或修改内外网地址和端口。
数据库名	postgres是RDS PostgreSQL实例默认的系统数据库，请勿在该数据库中进行任何操作。如果您需要连接RDS实例下创建的其他数据库，可从**数据库管理**页面获取。创建数据库，请参见创建数据库。

通过SQL Shell（psql）工具连接

通过PostgreSQL官方网站下载并安装PostgreSQL客户端时，将会自动安装**SQL Shell（psql）**工具。

在本地Windows系统的**开始**菜单中找到并打开**SQL Shell（psql）**工具，依次输入对应参数连接RDS PostgreSQL实例。



各参数在RDS控制台的获取方法。

参数	获取方法
Server	RDS PostgreSQL实例的连接地址，具体请参见查看或修改内外网地址和端口。
Database	数据库名，不配置默认为系统数据库postgres，请勿在该数据库中进行任何操作。如果您需要连接RDS实例下创建的其他数据库，可从**数据库管理**页面获取。创建数据库，请参见创建数据库。
Port	端口号，不配置默认为5432，如果您修改过，通过**数据库连接**页面获取。具体请参见查看或修改内外网地址和端口。
Username	用户名，通过**账号管理**页面获取用户名。创建账号，请参见创建账号。

通过应用程序连接

说明 本文以Maven项目JDBC连接实例为例，其它编程语言连接方式类似。

1. pom.xml中添加依赖。

```
<dependency> <groupId>postgresql</groupId> <artifactId>postgresql</artifactId> <version
>8.2-504.jdbc3</version> </dependency>
```

2. 连接实例示例代码如下：

```
public class DatabaseConnection { public static void main( String[] args ){ try { Class
.forName("org.postgresql.Driver"); } catch (ClassNotFoundException e) { e.printStackTra
ce(); } //实例连接地址 String hostname = "pgm-bp1i3kkq7321o9****.pg.rds.aliyuncs.com"; //
实例连接端口 int port = 5432; //数据库名称 String dbname = "postgres"; //用户名 String user
name = "username"; //密码 String password = "password"; String dbUrl = "jdbc:postgresql:
//" + hostname + ":" + port + "/" + dbname + "?binaryTransfer=true"; Connection dbConne
ction; try { dbConnection = DriverManager.getConnection(dbUrl, username, password); Sta
tement statement = dbConnection.createStatement(); //输入需要执行的SQL语句。 String select
Sql = "SELECT * FROM information_schema.sql_features LIMIT 10"; ResultSet resultSet = s
tatement.executeQuery(selectSql); while (resultSet.next()) { System.out.println(resultS
et.getString("feature_name")); } } catch (SQLException e) { e.printStackTrace(); } } }
```

通过SSL安全连接

RDS PostgreSQL支持设置SSL链路加密，对网络连接进行加密，保证传输链路的安全。具体配置请参见SSL连接RDS PostgreSQL数据库。

常见问题

Q：我使用函数计算，想获取RDS的数据，要怎么操作呢？

A：您可以为函数安装第三方依赖，使用内置模块获取RDS数据，详情请参见为函数安装第三方依赖。

16.2. 申请或释放外网地址

RDS支持内网地址和外网地址两种地址类型，默认提供内网地址供您内部访问RDS实例，如果需要外网访问，您需要申请外网地址。

内网地址和外网地址

地址类型	说明
内网地址	• 默认提供内网地址，无需申请，无法释放，可以切换网络类型。 • 如果您的应用部署在ECS实例，且该ECS实例与RDS实例在同一地域，且网络类型相同，则RDS实例与ECS实例可以通过内网互通，无需申请外网地址。 • 通过内网访问RDS实例时，安全性高，而且可以实现RDS的最佳性能。

外网地址	• 外网地址需要手动申请，不需要时也可以释放。
	• 无法通过内网访问RDS实例时，您需要申请外网地址。具体场景如下：
	  ○ ECS实例访问RDS实例，且ECS实例与RDS实例位于不同地域，或者网络类型不同。
	  ○ 阿里云以外的设备访问RDS实例。
	说明
	• 申请外网地址和后续产生的公网流量暂不收费。
	• 外网地址会降低实例的安全性，请谨慎使用。
	• 为了获得更快的传输速率和更高的安全性，建议您将应用迁移到与您的RDS实例在同一地域且网络类型相同的ECS实例，然后使用内网地址。

申请或释放外网地址

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据库连接**。
3. 您可以执行申请或释放操作：
   - 如果未申请外网地址，可以单击**申请外网地址**。
   - 如果已申请外网地址，可以单击**释放外网地址**。
4. 在弹出的对话框中，单击**确定**。

相关API

API	描述
申请外网地址	申请实例的外网地址
释放外网地址	释放实例的外网地址

16.3. 通过DMS登录RDS数据库

您可以通过阿里云的数据管理服务DMS登录RDS实例的数据库。

背景信息

数据管理DMS是一种集数据管理、结构管理、用户授权、安全审计、数据趋势、数据追踪、BI图表、性能与优化和服务器管理于一体的数据管理服务。

通过新版DMS登录RDS数据库

1. 登录新版DMS管理控制台。
2. 在左侧菜单栏选择目标实例，单击**请先登录**。

> **说明** 管控模式为**安全协同**的实例是授权登录，无需输入账号密码，在**免登录实例**菜单里双击目标数据库即可登录。

3. 输入登录实例数据库的账号密码，单击**确认**。
4. 登录后在左侧**已登录实例**菜单双击目标数据库，即可切换到目标数据库。

通过旧版DMS登录RDS数据库

> **说明** DMS已经升级为新版，建议您使用新版DMS登录RDS数据库。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 单击页面右上角的**登录数据库**，如下图所示，进入DMS的快捷登录页面。

> **说明** 此链接会根据您之前登录的习惯决定跳转到旧版DMS或者新版DMS。

3. 设置如下参数。

参数	说明

参数	说明
网络地址:端口	格式为 `<连接地址>:<连接端口>` ，例如rm-bpxxxxxxx.rds.aliyuncs.com:3433。关于如何查看实例的地址和端口信息，请参见查看或修改内外网地址和端口。
数据库用户名	实例的账号名称。
密码	实例账号对应的密码。

4. 单击**登录**。

> **说明** 若您希望浏览器记住该账号的密码，可以先勾选 **记住密码**，再单击 **登录**。

5. 若出现将DMS服务器的IP段加入到RDS白名单中的提示，单击**设置所有实例**或者**设置本实例**。

6. 成功添加白名单后，单击**登录**。

16.4. 查看或修改内外网地址和端口

在连接RDS实例时，您需要填写RDS的内外网地址和端口。本文将介绍如何在RDS控制台上查看RDS PostgreSQL实例的内外网地址和端口，以及如何修改内外网地址和端口。

修改内外网地址和端口

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**数据库连接**。
3. 单击**修改连接地址**。

4. 在弹出的对话框中，设置内外网连接地址前缀及端口号，单击**确定**。

> **说明**
> - 连接地址前缀以小写字母开头，8~64个字符，支持字母、数字和连字符（-）。
> - 端口范围为1000~5999。
> - 本地SSD盘实例暂不支持修改端口。

常见问题

- 修改连接地址或端口后是否需要修改应用程序内的连接地址或端口？
  需要修改，否则应用程序无法连接上数据库。
- 修改连接地址或端口是立即生效吗？是否需要重启实例？
  是立即生效，不需要重启实例。
- 某个连接地址修改或释放后，可以在其他实例上使用修改或释放前的地址吗？
  可以。
- 主备切换会影响连接地址吗？
  主备切换不会导致连接地址变更，仅后台对应的IP地址会变化，不会对您的应用正常运行造成影响。

17.网络/VPC/交换机

17.1. 切换虚拟交换机

您可以直接切换RDS PostgreSQL云盘实例的虚拟交换机。

前提条件

实例为RDS PostgreSQL云盘版。

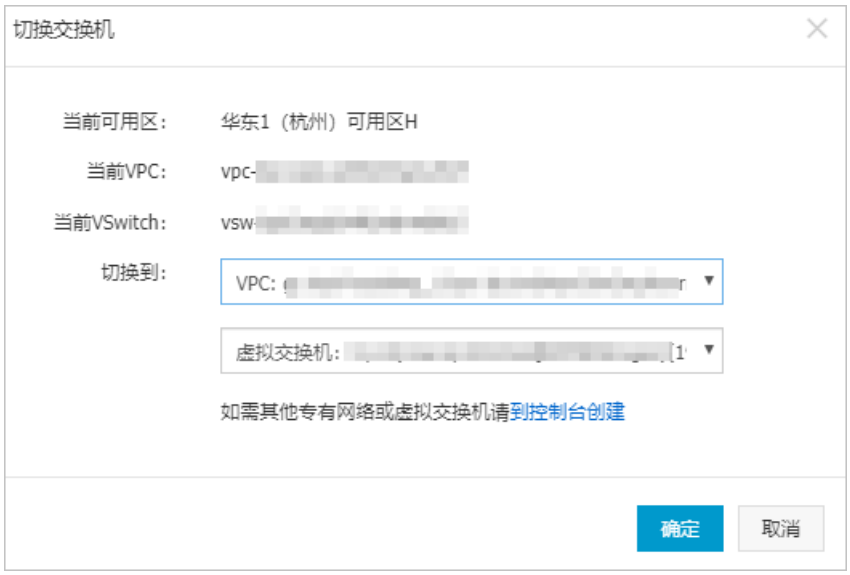
影响

- 切换过程会有30秒闪断，请确保应用程序具有重连机制。
- 切换虚拟交换机会造成虚拟IP（VIP）的变更，请您在应用程序中尽量使用连接地址进行连接，不要使用IP地址。
- VIP的变更会短暂影响到DMS、DTS的使用，变更结束后会自动恢复正常。
- 客户端缓存会导致只能读取数据，无法写入数据，请及时清理缓存。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据库连接**。
3. 在右侧单击**切换交换机**。
4. 选择VPC和虚拟交换机，并单击**确定**。

> **说明** 如需新建VPC或虚拟交换机请单击 **到控制台创建**。



5. 在弹出的风险提示框中单击**确定切换**。

常见问题

不支持直接切换VPC和交换机的实例如何变更VPC?

购买新的实例（购买时选择目的VPC），然后将数据迁移到新的实例。详情请参见RDS PostgreSQL间的数据迁移。

17.2. 切换网络类型

您可以根据业务需求将网络类型为经典网络的RDS PostgreSQL实例切换为专有网络。

前提条件

RDS PostgreSQL实例的网络类型为经典网络。

网络类型

- 经典网络：实例之间不通过网络进行隔离，只能依靠实例自身的白名单策略来阻挡非法访问。
- 专有网络（VPC）：一个VPC就是一个隔离的网络环境。VPC的安全性较高，推荐您使用VPC网络。
  您可以自定义VPC中的路由表、IP 地址范围和网关。此外，您还可以通过专线或者VPN的方式将自建机房与阿里云VPC组合成一个虚拟机房，实现应用平滑上云。

> **说明**
> - 使用经典网络或专有网络，以及切换网络类型均不收取费用。
> - 对于PostgreSQL实例，切换网络类型前，需要先将IP白名单的模式切换为高安全白名单模式。具体操作请参见切换为高安全白名单模式。

从经典网络切换为专有网络（VPC）

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**数据库连接**。
3. 单击**切换为专有网络**。
4. 在弹出的对话框中，选择VPC和交换机，以及是否保留经典网络地址。
   - 选择VPC。建议选择您的ECS实例所在的VPC，否则ECS实例与RDS实例无法通过内网互通（除非在两个VPC之间创建云企业网或VPN网关）。
   - 选择交换机。如果选择的VPC中没有交换机，请创建与实例在同一可用区的交换机。具体操作请参见管理交换机。

   - 选择是否勾选**保留经典网络**，具体说明如下表所述。

操作	说明
不勾选	不保留经典网络地址，原经典网络地址变为VPC地址。 如果不保留经典网络地址，则切换网络类型时，RDS实例会发生一次30秒的闪断，而且经典网络的ECS对该RDS实例的内网访问会立即断开。

操作	说明
勾选	保留经典网络地址，同时生成一个新的VPC地址。表示使用混访模式，即RDS可以同时被经典网络和VPC的ECS通过内网访问。 如果保留经典网络地址，则切换网络类型时，RDS实例不会发生闪断，而且经典网络的ECS对该RDS实例的内网访问也不会断开，直到经典网络地址到期才断开。 在经典网络地址到期前，请将VPC地址配置到VPC的ECS中，以实现业务平滑迁移到VPC。在经典网络地址到期前的7天，系统会每天给您账号绑定的手机发送短信提醒。 更多介绍请参见临时混访方案（同时保留经典网络和专有网络地址）。

5. 将VPC的ECS内网IP地址添加到RDS实例的专有网络白名单分组，使得ECS可以通过内网访问RDS。如果没有专有网络的分组，请新建分组。
6. 在ECS上配置RDS的VPC地址：
   - 如果选择了保留经典网络地址，请在经典网络地址到期前，将RDS的VPC地址配置到VPC的ECS中。
   - 如果选择了不保留经典网络地址，那么切换网络类型后，经典网络的ECS对该RDS实例的内网访问会立即断开。请将RDS的VPC地址配置到VPC的ECS中。

> **说明**
> 如果要使经典网络中的ECS通过内网连接到VPC的RDS，您可以使用ClassicLink，或者将ECS切换到VPC网络。

常见问题

如何变更VPC?

不支持直接变更VPC，您可以使用以下方法变更VPC：

- 对于支持从VPC切换到经典网络，以及支持从经典网络切换到VPC的实例：
  i. 将网络模式从VPC切换为经典网络。
  ii. 将网络模式从经典网络切换至目的VPC。
- 对于不支持网络类型切换的实例：
  购买新的实例（购买时选择目的VPC），然后将数据迁移到新的实例。详情请参见RDS PostgreSQL间的数据迁移。

相关API

API	描述
ModifyDBInstanceNetworkType	修改RDS实例网络类型

17.3. 临时混访方案（同时保留经典网络和专有网络地址）

为满足日益增多的网络迁移需求，RDS新增了网络混访功能，可实现在无访问中断的情况下将经典网络平滑迁移到VPC上。

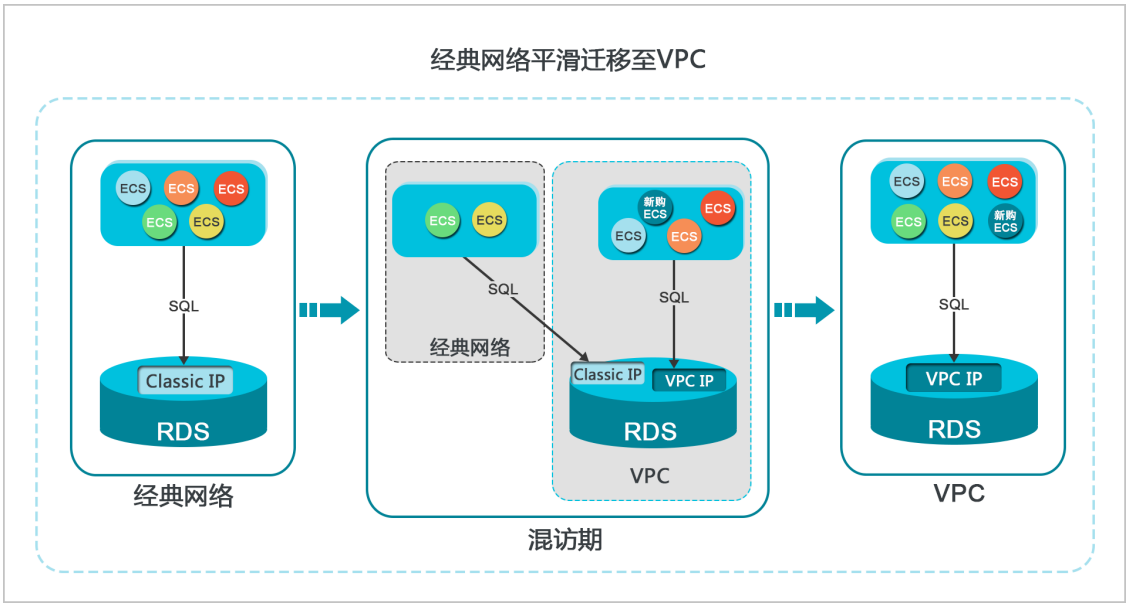
背景信息

以往将RDS实例从经典网络迁移到VPC时，经典网络的内网地址会变为VPC的内网地址（连接字符串没有变化，背后的IP地址有变化），会造成1次30秒内的闪断，而且经典网络中的ECS将不能再通过内网访问该RDS实例，为了能够平滑迁移网络，RDS新增了网络混访功能。

混访是指RDS实例可以同时被经典网络和专有网络中的ECS访问。在混访期间，RDS实例会保留原经典网络的内网地址并新增一个VPC下的内网地址，迁移网络时不会出现闪断。

基于安全性及性能的考虑，我们推荐您仅使用VPC，因此混访期有一定的期限，原经典网络的内网地址在保留时间到期后会被自动释放，应用将无法通过经典网络的内网地址访问数据库。为避免对业务造成影响，您需要在混访期中将VPC下的内网地址配置到您所有的应用中，以实现平滑的网络迁移。

例如，某一公司要将经典网络迁移至VPC时，若选用混访的迁移方式，在混访期内，一部分应用通过VPC访问数据库，一部分应用仍通过原经典网络的内网地址访问数据库，等所有应用都可以通过VPC访问数据库时，就可以将原经典网络的内网地址释放掉。



功能限制

在混访期间，有如下功能限制：

- 不支持切换成经典网络。
- 不支持迁移可用区。

前提条件

- 实例的网络类型是经典网络。
- 实例所在可用区已有可用的VPC和交换机。关于创建VPC和交换机的操作，请参见管理专有网络。

从经典网络迁移至VPC

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据库连接**。
3. 单击**切换为专有网络**。
4. 在弹出的对话框中，选择VPC和交换机，以及是否保留经典网络地址。
   - 选择VPC。建议选择您的ECS实例所在的VPC，否则ECS实例与RDS实例无法通过内网互通（除非在两个VPC之间创建云企业网或VPN网关）。
   - 选择交换机。如果选择的VPC中没有交换机，请创建与实例在同一可用区的交换机。具体操作请参

见管理交换机。

- 选择是否勾选**保留经典网络**，具体说明如下表所述。

操作	说明
不勾选	不保留经典网络地址，原经典网络地址变为VPC地址。 如果不保留经典网络地址，则切换网络类型时，RDS实例会发生一次30秒的闪断，而且经典网络的ECS对该RDS实例的内网访问会立即断开。
勾选	保留经典网络地址，同时生成一个新的VPC地址。表示使用混访模式，即RDS可以同时被经典网络和VPC的ECS通过内网访问。 如果保留经典网络地址，则切换网络类型时，RDS实例不会发生闪断，而且经典网络的ECS对该RDS实例的内网访问也不会断开，直到经典网络地址到期才断开。 在经典网络地址到期前，请将VPC地址配置到VPC的ECS中，以实现业务平滑迁移到VPC。

5. 将VPC的ECS内网IP地址添加到RDS实例的专有网络白名单分组，使得ECS可以通过内网访问RDS。如果没有专有网络的分组，请新建分组。
6.
   - 如果选择了保留经典网络地址，请在经典网络地址到期前，将RDS的VPC地址配置到VPC的ECS中。
   - 如果选择了不保留经典网络地址，那么切换网络类型后，经典网络的ECS对该RDS实例的内网访问会立即断开。请将RDS的VPC地址配置到VPC的ECS中。

> **说明** 如果要使经典网络中的ECS通过内网连接到VPC的RDS，您可以使用 ClassicLink，或者将ECS切换到VPC网络。

修改原经典网络内网地址的过期时间

在混访期间，您可以根据需求随时调整保留原经典网络的时间，过期时间会从变更日期重新开始计时。例如，原经典网络的内网地址会在2017年8月18日过期，但您在2017年8月15日将过期时间变更为“14天后”，则原经典网络的内网地址将会在2017年8月29日被释放。

修改过期时间的操作步骤如下所示：

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据库连接**。
3. 在**实例连接**页签，单击**修改过期时间**。
4. 在**修改过期时间**的确认页面，选择过期时间，单击**确定**。

18.数据库

18.1. 创建数据库

若要使用云数据库RDS，您需要在实例中创建数据库。本文介绍如何为RDS PostgreSQL实例创建数据库。

概念

- 实例：实例是虚拟化的数据库服务器。您可以在一个实例中创建和管理多个数据库。
- 数据库：数据库是以一定方式储存在一起、能与多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合，可以简单理解为存放数据的仓库。
- 字符集：字符集是数据库中字母、符号的集合，以及它们的编码规则。

注意事项

- 支持在控制台创建管理数据库。
- 如果您要迁移本地数据库到RDS，请在RDS实例中创建与本地数据库一致的迁移账号和数据库。

创建数据库

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据库管理**。
3. 单击**创建数据库**。
4. 设置如下参数。

参数	说明
数据库（DB）名称	○ 最长63个字符。 ○ 由小写字母、数字、中划线、下划线组成。 ○ 以字母开头，以字母或数字结尾。
支持字符集	数据库的字符集。
Collate	字符串排序规则。
Ctype	字符分类。
授权账号：	设置数据库的所有者，对数据库拥有ALL权限。
备注说明：	填写备注信息。

5. 单击**创建**。

下一步

连接PostgreSQL实例。

18.2. 删除数据库

本文介绍如何删除RDS PostgreSQL的数据库。

操作步骤

操作步骤

> **说明** 删除数据库后，系统将自动回收数据库使用空间。您可以通过**监控与告警**查看磁盘使用量，查看方法请参见查看资源和引擎监控。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据库管理**。
3. 找到要删除的数据库，单击其右侧**操作**列中的**删除**。
4. 在弹出的确认框中，单击**确定**。

18.3. 修改时区

本文介绍如何修改实例的时区。

前提条件

RDS PostgreSQL云盘版实例。

修改时区

RDS PostgreSQL云盘版实例支持修改配置文件中的时区设置，您可以在控制台的**参数设置**页面修改参数timezone。操作详情请参见设置实例参数。

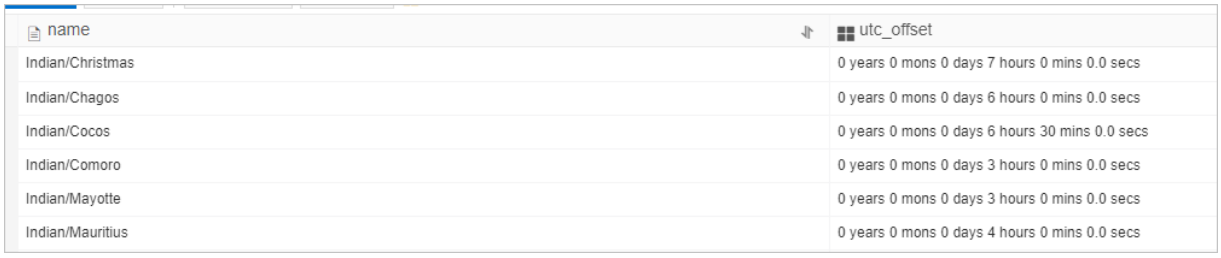
> **说明** RDS PostgreSQL本地盘实例暂不支持timezone参数。

查看支持的时区

您可以使用如下命令查看支持的时区：

```
select name,utc_offset from pg_timezone_names;
```

> **说明** 关于pg_timezone_names表的更多信息，请参见pg_timezone_names。



name	utc_offset
Indian/Christmas	0 years 0 mons 0 days 7 hours 0 mins 0.0 secs
Indian/Chagos	0 years 0 mons 0 days 6 hours 0 mins 0.0 secs
Indian/Cocos	0 years 0 mons 0 days 6 hours 30 mins 0.0 secs
Indian/Comoro	0 years 0 mons 0 days 3 hours 0 mins 0.0 secs
Indian/Mayotte	0 years 0 mons 0 days 3 hours 0 mins 0.0 secs
Indian/Mauritius	0 years 0 mons 0 days 4 hours 0 mins 0.0 secs

19.SQL翻译

RDS PostgreSQL提供SQL翻译功能，支持将其他数据库引擎的SQL翻译成PostgreSQL的SQL，方便您使用RDS PostgreSQL。

背景信息

SQL翻译目前仅支持将Oracle的SQL翻译成PostgreSQL的SQL，后续将增加其他引擎的SQL翻译功能。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏选择**SQL翻译**。
3. 在左边**来源数据库**的输入框中输入的SQL语句。

> **说明** 右边**目标数据库**的输出框会自动显示翻译后的SQL。

20.监控与报警

20.1. 查看资源和引擎监控

RDS提供了丰富的性能监控项，您可以通过RDS管理控制台查看实例的资源监控数据。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**监控与报警**。
3. 在**标准监控**页面选择查询时间，即可查看相应的监控数据，各版本支持的监控项如下。

版本	监控类型	监控项
PostgreSQL 10高可用版（本地SSD盘）、PostgreSQL 10基础版、PostgreSQL 9.4	资源监控	内存使用率
		CPU使用率
		磁盘空间
		IOPS
		当前总连接数
PostgreSQL 13、PostgreSQL 12、PostgreSQL 11、PostgreSQL 10高可用版（云盘）	资源监控	CPU使用率
		内存使用率
		数据盘IOPS
		数据盘空间使用率
		磁盘使用量（MB）
	引擎监控	连接数
		操作行数
		每秒事务数（TPS）
		膨胀点
		延迟（Bytes）
		慢查询
		长事务
		二阶段提交事务（2PC）

20.2. 设置监控频率

RDS PostgreSQL提供3种频率的监控，本文介绍如何设置监控频率。

背景信息

目前RDS PostgreSQL提供3种监控频率：

- 5秒/次

> **说明** 基础版及内存小于8G的数据库实例不支持5秒/次的监控频率。

- 60秒/次
- 300秒/次

> **说明** RDS确保您可以查询到近30天的监控数据，超过30天的监控数据不一定能查询到。

费用

对于付费支持的监控频率，计费方式为按量付费（每小时扣费一次）。例如一个实例使用5秒/次的监控频率，每小时需要USD 0.012。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**监控与报警**。

> **说明** 不同类型的数据库支持的监控项也不同，详情请参见 查看资源和引擎监控。

3. 选择**标准监控**标签页。
4. 单击**监控频率设置**。
5. 在弹出的**监控频率设置**对话框中选择监控频率，然后单击**确定**。

相关API

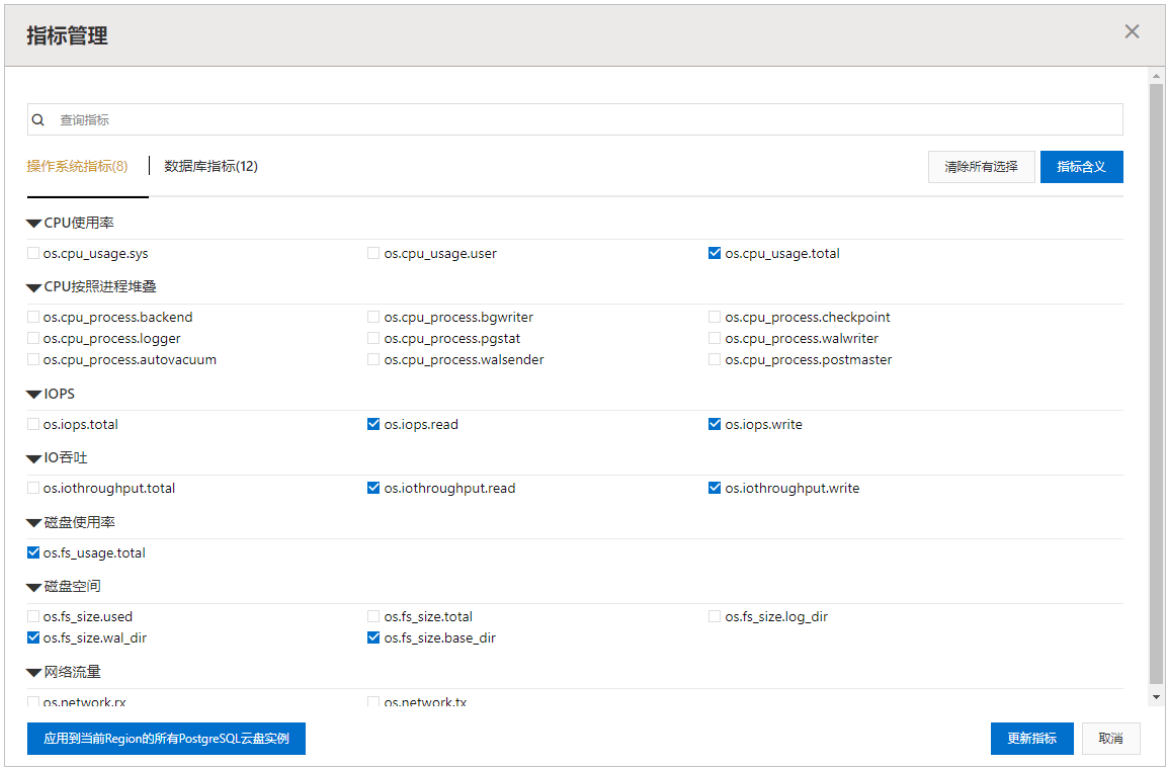
API	描述
查询监控频率	查询监控频率

20.3. 查看增强监控

除标准监控外，RDS PostgreSQL还提供了丰富的增强性能监控项，您可以通过RDS管理控制台查看这些监控数据。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**监控与报警**。
3. 选择**增强监控**页签，单击**指标管理**，分别在**操作系统指标**页签和**数据库指标**页签中选择需要显示的指标项，各指标项详情，请参见参考信息。



说明

- **增强监控**页面中最多能显示30个指标。
- 您可以将已选择显示的指标项方案应用到当前地域的所有RDS PostgreSQL实例：
  - 如果当前为云盘版实例，将会应用到已存在以及以后创建的RDS PostgreSQL云盘实例。
  - 如果当前为本地盘实例，将会应用到已存在以及以后创建的RDS PostgreSQL本地盘实例。

4. 单击**更新指标**后，在**增强监控**页即可看到对应的指标项监控数据。
5. **增强监控**页面还提供了多种查询设置，使您更方便地查看相应的监控数据。

序号	功能	说明
①	查询时间范围	支持按照预设的时间范围和自定义时间范围查询。 ○ 预设时间范围支持：30分钟、1小时、2小时、6小时、1天、7天和30天。 ○ 自定义时间范围：开始时间（YYYY-MM-DD hh:mm:ss）- 结束时间（YYYY-MM-DD hh:mm:ss）

序号	功能	说明
②	聚合方式	支持按照如下聚合方式显示指标结果。 ○ 求平均 ○ 求最大 ○ 求最小
③	布局	支持按如下分布调整显示结果的布局。 ○ 一列 ○ 二列 ○ 三列 ○ 四列
④	时间粒度	表示显示图表中横坐标的时间粒度。 时间粒度与查询时间相关联，对应关系如下： ○ 查询时间范围小于等于1小时，时间粒度为5秒 ○ 查询时间范围大于1小时且小于等于2小时，时间粒度为10秒 ○ 查询时间范围大于2小时且小于等于6小时，时间粒度为30秒 ○ 查询时间范围大于6小时且小于等于12小时，时间粒度为1分钟 ○ 查询时间范围大于12小时且小于等于1天， 时间粒度为2分钟 ○ 查询时间范围大于1天且小于等于5天， 时间粒度为10分钟 ○ 查询时间范围大于5天且小于等于15天，时间粒度为30分钟 ○ 查询时间范围大于15天且小于等于30天，时间粒度为1小时
⑤	光标联动	支持打开光标联动开关，当光标悬浮在图表的某一时间点时，其它图表联动显示对应时间点的指标数据。
⑥	刷新	支持通过手动刷新来更新图表数据。

参考信息

操作系统指标和数据库指标的详细信息如下。

操作系统指标

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
	• 平均值：os.cpu_usage.sys.avg • 最大值：os.cpu_usage.sys.max • 最小值：os.cpu_usage.sys.min	sys cpu使用率，sys cpu使用量 / cpu总量	%	✔	✔

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
CPU使用率	• 平均值：os.cpu_usage.user.avg • 最大值：os.cpu_usage.user.max • 最小值：os.cpu_usage.user.min	user cpu使用率，user cpu使用量 / cpu总量	%	✔☺	✔☺
	• 平均值：os.cpu_usage.total.avg • 最大值：os.cpu_usage.total.max • 最小值：os.cpu_usage.total.min	cpu使用率，(sys + user) cpu使用量 / cpu总量	%	✔☺	✔☺
	• 平均值：os.cpu_process.backend.avg • 最大值：os.cpu_process.backend.max • 最小值：os.cpu_process.backend.min	backend cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺
	• 平均值：os.cpu_process.bgwriter.avg • 最大值：os.cpu_process.bgwriter.max • 最小值：os.cpu_process.bgwriter.min	bgwriter cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺
	• 平均值：os.cpu_process.checkpoint.avg • 最大值：os.cpu_process.checkpoint.max • 最小值：os.cpu_process.checkpoint.min	checkpoint cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
CPU按照进程堆叠	• 平均值：os.cpu_process.logger.avg • 最大值：os.cpu_process.logger.max • 最小值：os.cpu_process.logger.min	logger cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺
	• 平均值：os.cpu_process.pgstat.avg • 最大值：os.cpu_process.pgstat.max • 最小值：os.cpu_process.pgstat.min	pgstat cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺
	• 平均值：os.cpu_process.walwriter.avg • 最大值：os.cpu_process.walwriter.max • 最小值：os.cpu_process.walwriter.min	walwriter cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺
	• 平均值：os.cpu_process.autovacuum.avg • 最大值：os.cpu_process.autovacuum.max • 最小值：os.cpu_process.autovacuum.min	autovacuum cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
	• 平均值：os.cpu_process.walsender.avg • 最大值：os.cpu_process.walsender.max • 最小值：os.cpu_process.walsender.min	walsender cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺
	• 平均值：os.cpu_process.postmaster.avg • 最大值：os.cpu_process.postmaster.max • 最小值：os.cpu_process.postmaster.min	postmaster cpu使用量（1cpu=100%，2cpu=200%，以次类推）	%	✔☺	✔☺
IOPS	• 平均值：os.iops.total.avg • 最大值：os.iops.total.max • 最小值：os.iops.total.min	磁盘读写IOPS	Counts/s	▯	✔☺
	• 平均值：os.iops.read.avg • 最大值：os.iops.read.max • 最小值：os.iops.read.min	磁盘读IOPS	Counts/s	▯	✔☺
	• 平均值：os.iops.write.avg • 最大值：os.iops.write.max • 最小值：os.iops.write.min	磁盘写IOPS	Counts/s	▯	✔☺
	• 平均值：os.iops.data.avg • 最大值：os.iops.data.max • 最小值：os.iops.data.min	本地数据盘IOPS	Counts/s	✔☺	▯
	• 平均值：os.iops.wal.avg • 最大值：os.iops.wal.max • 最小值：os.iops.wal.min	本地日志盘IOPS	Counts/s	✔☺	▯

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
IO吞吐	• 平均值：os.iothroughput.total.avg • 最大值：os.iothroughput.total.max • 最小值：os.iothroughput.total.min	磁盘读写吞吐量	MB/s	▯	✔☺
	• 平均值：os.iothroughput.read.avg • 最大值：os.iothroughput.read.max • 最小值：os.iothroughput.read.min	磁盘读吞吐量	MB/s	▯	✔☺
	• 平均值：os.iothroughput.write.avg • 最大值：os.iothroughput.write.max • 最小值：os.iothroughput.write.min	磁盘写吞吐量	MB/s	▯	✔☺
	• 平均值：os.iothroughput.data.avg • 最大值：os.iothroughput.data.max • 最小值：os.iothroughput.data.min	本地数据盘吞吐	MB/s	✔☺	▯
	• 平均值：os.iothroughput.wal.avg • 最大值：os.iothroughput.wal.max • 最小值：os.iothroughput.wal.min	本地日志盘吞吐	MB/s	✔☺	▯

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
磁盘使用率	• 平均值：os.fs_usage.total.avg • 最大值：os.fs_usage.total.max • 最小值：os.fs_usage.total.min	磁盘使用率	%	▯	✔☺
磁盘空间	• 平均值：os.fs_size.used.avg • 最大值：os.fs_size.used.max • 最小值：os.fs_size.used.min	磁盘已用空间	MB	▯	✔☺
	• 平均值：os.fs_size.total.avg • 最大值：os.fs_size.total.max • 最小值：os.fs_size.total.min	磁盘总空间	MB	▯	✔☺
	• 平均值：os.fs_size.log_dir.avg • 最大值：os.fs_size.log_dir.max • 最小值：os.fs_size.log_dir.min	日志文件大小（包含审计日志、错误日志、慢SQL日志）	MB	✔☺	✔☺
	• 平均值：os.fs_size.wal_dir.avg • 最大值：os.fs_size.wal_dir.max • 最小值：os.fs_size.wal_dir.min	wal文件大小	MB	✔☺	✔☺
	• 平均值：os.fs_size.base_dir.avg • 最大值：os.fs_size.base_dir.max • 最小值：os.fs_size.base_dir.min	数据文件大小（不包含日志和wal）	MB	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
网络流量	• 平均值：os.network.rx.avg • 最大值：os.network.rx.max • 最小值：os.network.rx.min	网络输入流量	MB/s	□	✔☺
	• 平均值：os.network.tx.avg • 最大值：os.network.tx.max • 最小值：os.network.tx.min	网络输出流量	MB/s	□	✔☺

数据库指标

> **说明** 下表中监控指标的更多解释，请参见 PostgreSQL官方文档。

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
Shared Buffers命中率	• 平均值：db.buffers.hit_ratio.avg • 最大值：db.buffers.hit_ratio.max • 最小值：db.buffers.hit_ratio.min	shared_buffers命中率	%	✔☺	✔☺
Shared Buffers命中次数	• 平均值：db.buffers.blks_hit.avg • 最大值：db.buffers.blks_hit.max • 最小值：db.buffers.blks_hit.min	shared_buffers每秒命中数	Blocks/s	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
IO	• 平均值：db.io.blks_read.avg • 最大值：db.io.blks_read.max • 最小值：db.io.blks_read.min	backend进程每秒磁盘read次数（OS buffered read）	Counts/s	✔☺	✔☺
	• 平均值：db.io.buffers_backend.avg • 最大值：db.io.buffers_backend.max • 最小值：db.io.buffers_backend.min	backend进程每秒磁盘write次数（OS buffered write）	Counts/s	✔☺	✔☺
	• 平均值：db.io.buffers_checkpoint.avg • 最大值：db.io.buffers_checkpoint.max • 最小值：db.io.buffers_checkpoint.min	checkpoint进程每秒磁盘write次数（OS buffered write）	Counts/s	✔☺	✔☺
	• 平均值：db.io.buffers_clean.avg • 最大值：db.io.buffers_clean.max • 最小值：db.io.buffers_clean.min	bgwriter进程每秒磁盘write次数（OS buffered write）	Counts/s	✔☺	✔☺
	• 平均值：db.io.buffers_backend_fsync.avg • 最大值：db.io.buffers_backend_fsync.max • 最小值：db.io.buffers_backend_fsync.min	backend进程每秒执行磁盘fsync的次数	Counts/s	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
Checkpoint写时间	• 平均值：db.checkpoints_sync_time.avg • 最大值：db.checkpoints_sync_time.max • 最小值：db.checkpoints_sync_time.min	checkpoint进程平均每秒磁盘fsync时间	ms/s	✔☺	✔☺
	• 平均值：db.checkpoints_write_time.avg • 最大值：db.checkpoints_write_time.max • 最小值：db.checkpoints_write_time.min	checkpoint进程平均每秒磁盘write时间（OS buffered write）	ms/s	✔☺	✔☺
Checkpoint次数	• 平均值：db.checkpoints_timed.avg • 最大值：db.checkpoints_timed.max • 最小值：db.checkpoints_timed.min	每秒内核调度的checkpoints数量	Counts/s	✔☺	✔☺
	• 平均值：db.checkpoints_req.avg • 最大值：db.checkpoints_req.max • 最小值：db.checkpoints_req.min	每秒外部请求的checkpoints数量	Counts/s	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
连接	• 平均值：db.connections.active.avg • 最大值：db.connections.active.max • 最小值：db.connections.active.min	活跃连接数	Counts	✔☺	✔☺
	• 平均值：db.connections.waiting.avg • 最大值：db.connections.waiting.max • 最小值：db.connections.waiting.min	等待连接数	Counts	✔☺	✔☺
	• 平均值：db.connections.idle.avg • 最大值：db.connections.idle.max • 最小值：db.connections.idle.min	空闲连接数	Counts	✔☺	✔☺
	• 平均值：db.connections.total.avg • 最大值：db.connections.total.max • 最小值：db.connections.total.min	连接数	Counts	✔☺	✔☺
	• 平均值：db.connections.spec.avg • 最大值：db.connections.spec.max • 最小值：db.connections.spec.min	规格连接数	Counts	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
事务TPS	• 平均值：db.transactions.xact_commit.avg • 最大值：db.transactions.xact_commit.max • 最小值：db.transactions.xact_commit.min	每秒写事务提交数	Counts/s	✔☺	✔☺
	• 平均值：db.transactions.xact_rollback.avg • 最大值：db.transactions.xact_rollback.max • 最小值：db.transactions.xact_rollback.min	每秒写事务回滚数	Counts/s	✔☺	✔☺
事务状态	• 平均值：db.transactions.active.avg • 最大值：db.transactions.active.max • 最小值：db.transactions.active.min	活跃状态事务数	Counts	✔☺	✔☺
	• 平均值：db.transactions.waiting.avg • 最大值：db.transactions.waiting.max • 最小值：db.transactions.waiting.min	等待状态事务数	Counts	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
	• 平均值：db.transactions.idle.avg • 最大值：db.transactions.idle.max • 最小值：db.transactions.idle.min	idle-in-transaction状态事务数（注意：不合理状态，需要尽快处理）	Counts	✔☺	✔☺
SQL	• 平均值：db.sql.tup_returned.avg • 最大值：db.sql.tup_returned.max • 最小值：db.sql.tup_returned.min	每秒返回行数	Tuples/s	✔☺	✔☺
	• 平均值：db.sql.tup_fetched.avg • 最大值：db.sql.tup_fetched.max • 最小值：db.sql.tup_fetched.min	每秒读取行数	Tuples/s	✔☺	✔☺
	• 平均值：db.sql.tup_inserted.avg • 最大值：db.sql.tup_inserted.max • 最小值：db.sql.tup_inserted.min	每秒插入行数	Tuples/s	✔☺	✔☺
	• 平均值：db.sql.tup_deleted.avg • 最大值：db.sql.tup_deleted.max • 最小值：db.sql.tup_deleted.min	每秒删除行数	Tuples/s	✔☺	✔☺
	• 平均值：db.sql.tup_updated.avg • 最大值：db.sql.tup_updated.max • 最小值：db.sql.tup_updated.min	每秒更新行数	Tuples/s	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
慢SQL	• 平均值：db.slow_sql.one_second.avg • 最大值：db.slow_sql.one_second.max • 最小值：db.slow_sql.one_second.min	已执行1s的SQL数	Counts	✔☺	✔☺
	• 平均值：db.slow_sql.three_seconds.avg • 最大值：db.slow_sql.three_seconds.max • 最小值：db.slow_sql.three_seconds.min	已执行3s的SQL数	Counts	✔☺	✔☺
	• 平均值：db.slow_sql.five_seconds.avg • 最大值：db.slow_sql.five_seconds.max • 最小值：db.slow_sql.five_seconds.min	已执行5s的SQL数	Counts	✔☺	✔☺
	• 平均值：db.long_transactions.active_one_second.avg • 最大值：db.long_transactions.active_one_second.max • 最小值：db.long_transactions.active_one_second.min	已执行1s的事务数	Counts	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
长事务	• 平均值：db.long_transactions.active_three_seconds.avg • 最大值：db.long_transactions.active_three_seconds.max • 最小值：db.long_transactions.active_three_seconds.min	已执行3s的事务数	Counts	✔☺	✔☺
	• 平均值：db.long_transactions.idle_one_second.avg • 最大值：db.long_transactions.idle_one_second.max • 最小值：db.long_transactions.idle_one_second.min	已空闲1s的事务数	Counts	✔☺	✔☺
	• 平均值：db.long_transactions.idle_three_seconds.avg • 最大值：db.long_transactions.idle_three_seconds.max • 最小值：db.long_transactions.idle_three_seconds.min	已空闲3s的事务数	Counts	✔☺	✔☺
	• 平均值：db.long_transactions.idle_five_seconds.avg • 最大值：db.long_transactions.idle_five_seconds.max • 最小值：db.long_transactions.idle_five_seconds.min	已空闲5s的事务数	Counts	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
	• 平均值：db.long_transactions.two_pc_one_second.avg • 最大值：db.long_transactions.two_pc_one_second.max • 最小值：db.long_transactions.two_pc_one_second.min	已执行1s的两阶段事务	Counts	✔☺	✔☺
	• 平均值：db.long_transactions.two_pc_three_seconds.avg • 最大值：db.long_transactions.two_pc_three_seconds.max • 最小值：db.long_transactions.two_pc_three_seconds.min	已执行3s的两阶段事务	Counts	✔☺	✔☺
	• 平均值：db.long_transactions.two_pc_five_seconds.avg • 最大值：db.long_transactions.two_pc_five_seconds.max • 最小值：db.long_transactions.two_pc_five_seconds.min	已执行5s的两阶段事务	Counts	✔☺	✔☺
临时文件数	• 平均值：db.temp.temp_files.avg • 最大值：db.temp.temp_files.max • 最小值：db.temp.temp_files.min	每秒临时文件生成个数	Counts/s	✔☺	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
临时文件大小	• 平均值：db.temp.temp_bytes.avg • 最大值：db.temp.temp_bytes.max • 最小值：db.temp.temp_bytes.min	每秒临时文件数据生成大小	Bytes/s	✔☺	✔☺
膨胀点	• 平均值：db.swell.swell_time.avg • 最大值：db.swell.swell_time.max • 最小值：db.swell.swell_time.min	膨胀点，最长事务已执行时间	s	✔☺	✔☺
数据库最大年龄	• 平均值：db.age.max_age.avg • 最大值：db.age.max_age.max • 最小值：db.age.max_age.min	数据库最大年龄	xids	✔☺	✔☺
只读同步延迟	• 平均值：db.ro_replica.replay_lag.avg • 最大值：db.ro_replica.replay_lag.max • 最小值：db.ro_replica.replay_lag.min	只读实例replay延迟时间	s	□	✔☺
	• 平均值：db.ro_replica.write_lag.avg • 最大值：db.ro_replica.write_lag.max • 最小值：db.ro_replica.write_lag.min	只读实例write延迟时间	s	□	✔☺

指标分类	指标名称	指标含义	指标单位	RDS PostgreSQL 9.4 、10本地盘实例	RDS PostgreSQL 10、11、12、13云盘版实例
	• 平均值：db.ro_replica.flush_lag.avg • 最大值：db.ro_replica.flush_lag.max • 最小值：db.ro_replica.flush_lag.min	只读实例flush延迟时间	s	□	✔☺
逻辑复制	• 平均值：db.slots.max_slot_wal_delay.avg • 最大值：db.slots.max_slot_wal_delay.max • 最小值：db.slots.max_slot_wal_delay.min	最大ReplicationSlot wal复制延迟（注意：复制位点之后的wal需要被保留，该值较大时会导致wal堆积，需要尽快处理）	MB	✔☺	✔☺

相关API

API	描述
DescribeDBInstancePerformance	查询性能数据
DescribeAvailableMetrics	获取增强监控指标列表
ModifyDBInstanceMetrics	变更增强监控指标项
DescribeDBInstanceMetrics	查询实例已开启的增强指标

20.4. 管理报警

您可以打开一键告警或设置自定义告警规则，系统在监控项数据满足条件时，会通知报警联系组中的所有联系人。

背景信息

监控报警是通过阿里云监控产品实现的。通过阿里云监控产品，您可以设置监控项，并在触发监控项的报警规则时，通过邮件通知报警联系组中的所有联系人。您可以维护报警监控项对应的报警联系组，以便发生报警时，能及时通知到相关联系人。

开启一键告警

RDS PostgreSQL实例开启一键告警按钮，能够快速建立RDS的报警体系，让您可以及时知晓关键监控项的异常。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**监控与报警**。
3. 单击**报警**页签。
4. 在页面右侧打开**一键告警**的开关。

打开一键告警后，系统将自动监控如下监控项：

规则名	监控项	统计周期	报警规则
SystemDefault_acs_rds_dashboard_PG_RO_ReadLag	只读同步延迟	60秒	当只读同步延迟出现5次平均值大于等于7200秒时，通知云账号报警联系人。
SystemDefault_acs_rds_dashboard_conn_usgae	连接数使用率	60秒	当连接数使用率出现5次平均值大于等于90%时，通知云账号报警联系人。
SystemDefault_acs_rds_dashboard_local_fs_size_usage	磁盘空间使用率	60秒	当磁盘空间使用率出现5次平均值大于等于90%时，通知云账号报警联系人。
SystemDefault_acs_rds_dashboard_cpu_usage	CPU使用率	60秒	当CPU使用率出现5次平均值大于等于90%时，通知云账号报警联系人。
SystemDefault_acs_rds_dashboard_iops_usage	IOPS使用率	60秒	当IOPS使用率出现5次平均值大于等于80%时，通知云账号报警联系人。
SystemDefault_acs_rds_dashboard_PG_MaxSlotWalDelay	最大复制槽延迟	60秒	当最大复制槽延迟出现5次平均值大于等于1024M时，通知云账号报警联系人。
SystemDefault_acs_rds_dashboard_PG_SwellTime	最长事务执行耗时	60秒	当最长事务执行耗时出现5次平均值大于等于36000秒时，通知云账号报警联系人。

5. （可选）打开一键告警后，如果不指定报警联系人，则报警信息接收人默认为阿里云主账号拥有者，您可以修改报警联系人。
   i. 登录云监控控制台。
   ii. 在左侧导航栏，单击**报警服务 > 报警联系人**。
   iii. 在**报警联系人**页签创建联系人。创建方法，请参见创建报警联系人。
   iv. 在**报警联系组**页签的搜索框中搜索Default Contact Group，单击**操作**列的 图标。

v. 在**新建联系组**页面，修改联系人。

添加自定义告警规则

RDS PostgreSQL除一键告警外，还支持您根据实际需要自定义告警规则，支持的报警项如下：

数据库年龄	非活跃复制槽数量	最慢SQL执行耗时
最大复制槽延迟	只读同步延迟	只读流复制状态
最慢Standby回放延迟	最长事务执行耗时	每CPU平均活跃链接数
连接数使用率	CPU使用率	IOPS使用率
INODE使用率	磁盘空间使用率	内存使用率

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**监控与报警**。
3. 单击**报警**页签。
4. 在右侧单击**报警规则设置**，进入云监控控制台。

5. 创建报警联系组，详情请参见创建报警联系人或报警联系组。
6. 创建报警规则，详情请参见创建阈值报警规则。

> **说明**
>
> - 创建报警规则时，产品选择**云数据库RDS版-PostgreSQL**。
> - 您也可以通过标签自动监控资源。详情请参见通过标签自动监控资源。

21.数据安全/加密

21.1. 切换为高安全白名单模式

RDS实例的IP白名单可以从通用白名单模式切换为高安全白名单模式，从而拥有更高的安全性。

前提条件

实例版本如下：

- PostgreSQL 10高可用版（本地盘）
- PostgreSQL 9.4高可用版（本地盘）

背景信息

RDS实例的IP白名单分为两种模式：

- 通用白名单模式
  白名单中的IP地址不区分经典网络和专有网络（既适用于经典网络也适用于专有网络）。
- 高安全白名单模式
  白名单中区分经典网络的IP白名单分组和专有网络的IP白名单分组。创建IP白名单分组时需要指定网络类型。

切换为高安全模式后的变化

- 对于专有网络的实例，原有的IP白名单将全量复制为一个适用于专有网络的IP白名单分组。
- 对于经典网络的实例，原有的IP白名单将全量复制为一个适用于经典网络的IP白名单分组。
- 对于处于混访模式（专有网络+经典网络）的实例，原有的IP白名单将全量复制为两个完全相同的IP白名单分组，分别适用于专有网络和经典网络。

> **说明**
> - 切换为高安全模式不会影响白名单中的ECS安全组。
> - 切换为高安全模式耗时约三分钟，切换期间不影响应用连接。

注意事项

- 切换为高安全白名单模式后无法切换回通用白名单模式。
- 高安全白名单模式下，经典网络白名单分组也适用于公网访问。如果有公网设备要访问RDS实例，请将公网设备IP地址添加到经典网络白名单分组。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**数据安全性**。
3. 在**白名单设置**页签中，单击**切换高安全白名单模式（推荐）**。
4. 在弹出的对话框中，单击**确认切换**。

常见问题

- 高安全白名单模式时，从外网访问实例，需要将公网IP添加到哪个分组？
  需要添加到经典网络类型的分组。

- 高安全白名单模式的优势在哪里？
  优势在于区分网络类型。例如在专有网络中放通某个IP，该IP仅能在专有网络内访问实例，公网中相同的IP无法访问实例，增加了实例的安全性。

21.2. IP白名单与安全组

21.2.1. 设置白名单

创建RDS实例后，您需要设置RDS实例的白名单，以允许外部设备访问该RDS实例。

其他引擎设置白名单请参见：

- MySQL设置白名单
- SQL Server设置白名单
- PPAS设置白名单
- MariaDB设置白名单

操作场景

IP白名单指允许访问RDS实例的IP清单。设置IP白名单可以让RDS实例得到高级别的访问安全保护，建议您定期维护白名单。

通常需要设置IP白名单的场景如下：

- 场景1
  创建RDS实例后，您需要将外部IP地址添加至IP白名单中，外部设备才可以正常访问该RDS实例。
- 场景2
  当数据库连接异常时，您可以检查白名单设置是否正确。
  不同连接场景下，IP白名单的设置请参见下表。

> **说明** 专有网络VPC是阿里云上一种隔离的网络环境，安全性比传统的经典网络更高。

连接场景	网络类型	IP白名单设置
	两个实例在相同专有网络VPC内（推荐）	添加ECS实例私有IP地址。
	两个实例在不同专有网络VPC内	不同专有网络的实例无法内网互通，您可以参考如下方案： i. 切换RDS专有网络，选择和ECS实例相同的VPC。 **说明** ECS实例和RDS实例需要处于相同地域才能切换到相同VPC。如果地域不同，为业务稳定，建议您通过DTS将RDS实例迁移至ECS实例所属地域。详情请参见RDS PostgreSQL间的数据迁移。 ii. 在IP白名单中添加ECS实例私有IP地址。
	两个实例均为经典网络	添加ECS实例私有IP地址。

连接场景	网络类型	IP白名单设置
ECS实例和RDS实例连接	ECS实例为经典网络 RDS实例为专有网络	不同网络类型的实例无法内网互通，您可以参考如下方案： i. 将ECS 实例从经典网络迁移到专有网络，选择和RDS实例相同的VPC。 说明 ECS实例和RDS实例需要处于相同地域才能切换到相同VPC。如果地域不同，为业务稳定，建议您通过DTS将RDS实例迁移至ECS实例所属地域。详情请参见RDS PostgreSQL间的数据迁移。 ii. 在IP白名单中添加ECS实例私有IP地址。
	ECS实例为专有网络 RDS实例为经典网络	不同网络类型的实例无法内网互通，您可以参考如下方案： i. 把RDS实例从经典网络切换为专有网络，选择和ECS实例相同的VPC。 说明 ECS实例和RDS实例需要处于相同地域才能切换到相同VPC。如果地域不同，为业务稳定，建议您通过DTS将RDS实例迁移至ECS实例所属地域。详情请参见RDS PostgreSQL间的数据迁移。 ii. 在IP白名单中添加ECS实例私有IP地址。
云外主机连接RDS实例	无	在IP白名单中添加云外主机的公网IP地址。 说明 ○ 云外主机的应用程序中使用RDS实例的外网连接地址。 ○ 定位本地公网IP地址请参见外网无法连接RDS MySQL或MariaDB：如何正确填写本地设备的公网IP地址。

注意事项

- 单个实例最多支持50个IP白名单分组。
- 设置白名单不会影响RDS实例的正常运行。
- 默认的IP白名单分组（default ）不能删除，只能清空。
- 请勿修改或删除系统自动生成的分组，避免影响相关产品的使用。例如 ali_dms_group （DMS产品IP地址白名单分组）、 hdm_security_ips （DAS产品IP地址白名单分组）。
- 默认的IP白名单只包含127.0.0.1，表示任何IP均无法访问该RDS实例。

设置通用模式IP白名单

通用白名单模式不区分经典网络和专用网络。在通用白名单模式下，设置的IP地址，既可通过经典网络，也可通过专有网络访问RDS实例。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据安全性**。

3. 单击**添加白名单分组**，填写**分组名称**或单击已有分组的**修改** 。
4. 填写需要访问该实例的IP地址或IP段，然后单击。

> **说明**
> - 用英文逗号隔开多个IP地址或IP段，且逗号前后不能有空格，例如 `192.168.0.1,172.16.213.9` 。
> - 单个实例最多添加1000个IP地址或IP段。当IP地址较多时，建议将零散的IP合并为IP段，例如10.10.10.0/24。

5. （可选）如果当前实例包含只读实例，可以通过参数**白名单同步到只读实例**配置白名单同步，将主实例的白名单同步至指定的只读实例。当有多个只读实例时，支持多选。

6. （可选）单击**加载ECS内网IP**，将显示您当前阿里云账号下所有ECS实例的IP地址，可快速添加ECS私有IP地址到白名单中。

高安全白名单模式IP白名单

高安全白名单模式区分经典网络和专用网络。在高安全白名单模式下，白名单分组需指定网络隔离模式，例如设置在经典网络的白名单IP地址，不可从专有网络访问RDS实例，反之亦然。

仅本地SSD盘实例支持高安全白名单模式，如果实例已经是高安全白名单模式，参见下文进行设置即可。如果需要切换为高安全白名单模式，请参见切换为高安全白名单模式。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据安全性**。
3. 在页签中，添加或修改白名单分组。
   - 添加白名单分组
     a. 单击**添加白名单分组**。
     b. 选择**允许访问实例的网络类型**。
     c. 填写**分组名称**。
   - 修改白名单分组
     单击目标白名单分组右侧的**修改**。
4. 在**组内白名单**框内，填写需要访问该实例的IP地址或IP段，然后单击**确定**。

> **说明**
> - 用英文逗号隔开多个IP地址或IP段，且逗号前后不能有空格，例如 `192.168.0.1,172.16.213.9` 。
> - 单个实例最多添加1000个IP地址或IP段。当IP地址较多时，建议将零散的IP合并为IP段，例如10.10.10.0/24。

5. （可选）如果当前实例包含只读实例，可以通过参数**白名单同步到只读实例**配置白名单同步，将主实例的白名单同步至指定的只读实例。当有多个只读实例时，支持多选。

6. （可选）单击**加载ECS内网IP**，将显示您当前阿里云账号下所有ECS实例的IP地址，可快速添加ECS私有IP地址到白名单中。

下一步

创建数据库和账号

相关API

API	描述
DescribeDBInstanceIPArrayList	查看RDS实例IP白名单
ModifySecurityIps	修改RDS实例IP白名单

21.2.2. 设置安全组

安全组是一种虚拟防火墙，用于控制安全组中的ECS实例的出入流量。在RDS白名单中添加安全组后，该安全组中的ECS实例就可以访问RDS实例。本文介绍如何设置安全组。

操作场景

创建RDS PostgreSQL实例后，暂时无法访问实例。您需要设置RDS PostgreSQL实例的IP白名单或安全组。

关于安全组的更多信息，请参见创建安全组。

> **说明** 您可以同时设置 IP白名单和安全组。IP白名单中的IP地址和安全组中的ECS实例都可以访问该RDS实例。

注意事项

- 您可以同时设置IP白名单和安全组。IP白名单中的IP地址和安全组中的ECS实例都可以访问该RDS实例。
- 实例最多支持添加10个安全组。

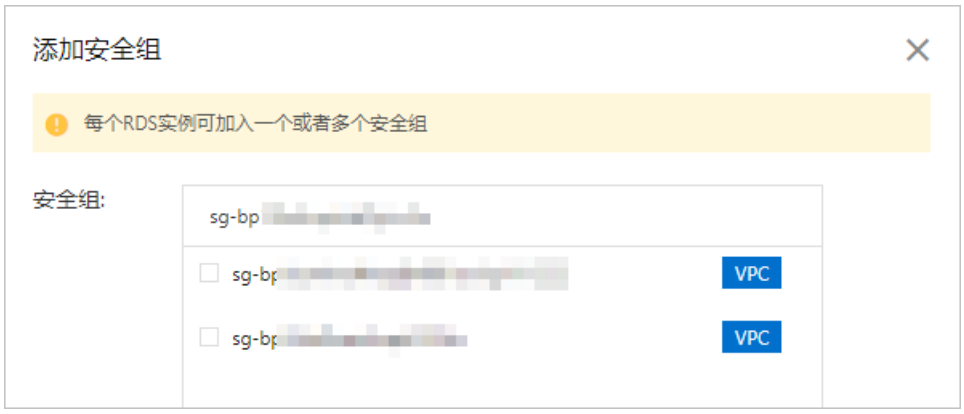
- 白名单中的安全组的更新将实时应用到白名单。
- RDS实例只能添加与自身网络类型相同的安全组，即实例为专有网络VPC时，只能添加VPC类型的安全组；实例为经典网络时，只能添加经典网络类型的安全组。

> **说明** 添加安全组后，如果切换实例网络类型，会导致安全组失效，需要重新添加对应网络类型的安全组。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**数据安全性**，然后单击**安全组**页签。
3. 单击**添加安全组**。

   > **说明** 带有 VPC标识的安全组为专有网络中的安全组。

4. 选择要添加的安全组，单击**确定**。

下一步

创建数据库和账号

相关API

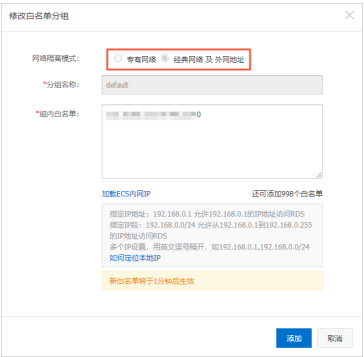
API	描述
DescribeSecurityGroupConfiguration	查询指定RDS实例和ECS安全组的关联信息
ModifySecurityGroupConfiguration	修改指定RDS实例和ECS安全组的关联信息

21.2.3. 白名单常见案例和问题

本文介绍设置白名单时的案例和问题。

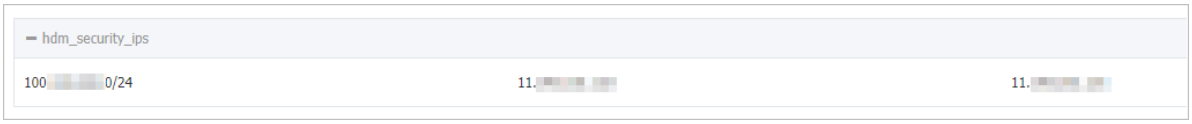
常见错误案例

错误案例	说明	解决办法
没有设置IP白名单，即白名单中只有127.0.0.1。	该地址表示不允许任何IP地址访问RDS实例。	在IP白名单中添加外部IP地址。

错误案例	说明	解决办法
测试连接实例时，添加的IP地址是0.0.0.0。	IP地址格式错误。	修改为0.0.0.0/0。 **注意** 0.0.0.0/0表示允许任何IP访问RDS实例，只建议在测试时使用，请勿在线上业务实例中设置IP白名单为0.0.0.0/0。
IP白名单中添加的设备公网IP地址错误。	• 公网IP地址不固定，可能会变动。 • IP地址查询工具或网站查询的公网IP地址可能不准确。	参见RDS for PostgreSQL/PPAS 如何定位本地 IP。
高安全模式的IP白名单中，网络类型错误。	高安全白名单模式会区分经典网络和专有网络。	在正确的网络类型分组内填写IP地址。例如专有网络的IP白名单分组内添加某个IP地址，则这个IP地址只能在专有网络内访问该RDS实例。 

常见问题

- Q：可以同时设置IP白名单和安全组吗？
  A：可以。IP白名单中的IP地址和安全组中的ECS实例都可以访问该RDS实例。
- Q：设置IP白名单后立刻生效吗？
  A：等待1分钟左右才会生效。
- Q：ali_dms_group和hdm_security_ips白名单分组是什么？
  A：您在数据管理服务DMS和数据库自治服务DAS中接入RDS实例时，经过您的授权后，系统会生成ali_dms_group和hdm_security_ips白名单分组。请勿修改或删除这些分组，避免影响相关服务的使用。这些服务不会操作您任何业务数据。



21.3. SSL链路加密

21.3.1. 快速配置SSL加密

RDS PostgreSQL支持设置SSL链路加密，对网络连接进行加密，保证传输链路的安全。本文介绍开启SSL链路加密功能的入门配置。

背景信息

SSL（Secure Sockets Layer）指安全套接字协议，是为了安全通信、数据安全性专门设计的安全协议，SSL 3.0之后更名为TLS（Transport Layer Security）。本文以配置云端证书为例，指导您快速配置SSL链路加密。

> **说明** RDS PostgreSQL支持的TLS版本为1.0、1.1和1.2。

RDS PostgreSQL实例支持的SSL链路加密配置及功能对比如下：

对比项	配置云端证书	配置自定义证书	配置客户端CA证书
获取方式	由阿里云颁发	由证书认证机构发或自签名证书颁发	由自签名证书颁发
证书有效期	365天	由您自定义	由您自定义
保护的连接地址数	1个	1个或多个	不影响保护的连接地址数，取决于配合使用的云端证书或自定义证书
证书作用	开启SSL链路加密，客户端验证服务端真伪	开启SSL链路加密，客户端验证服务端真伪	服务端验证客户端真伪

> **说明**
> - 配置云端证书或自定义证书为必选配置，用于开启SSL链路加密。
> - 配置客户端CA证书为可选配置，用于服务端验证客户端真伪。

前提条件

- 实例为RDS PostgreSQL 10或以上的云盘版本。
- 已下载pgAdmin 4客户端。

注意事项

- 开启SSL链路加密后，会造成CPU使用率上升、读写时延增长。
- 开启SSL链路加密后，已有连接需要断开重连，加密才会生效。
- 开启云端证书、更改云端证书保护的连接地址或关闭SSL链路加密，将会重启您的数据库实例，用时约3分钟左右，请在业务低峰期操作。

步骤一：使用云端证书开启SSL链路加密

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 进入**数据安全性**>SSL页面。

> **说明** 如果您在控制台中未看到 SSL页面，请检查您的RDS PostgreSQL实例版本及存储类型，具体详见 前提条件。

3. 切换证书来源为**使用云端证书**，单击**配置数据库证书（防止伪装数据库）**后的**配置**按钮，选择需要保护的连接地址。

> **说明**
>
> - 如果您未开启外网地址，**请选择需要保护的连接串**对话框中将会显示RDS PostgreSQL实例的内网连接地址。如果您开启了外网地址，将会同时显示内网和外网两个连接地址。但云端证书只能对一个连接地址进行保护，内网连接相对安全，推荐您对外网连接地址进行保护。查看内网和外网地址请参见查看内外网地址。
> - 如果您需要同时对内网和外网连接地址进行保护，请参见配置自定义证书。
> - 配置云端证书后，实例的**运行状态**将会变更为**修改SSL中**，该状态将持续三分钟左右，请耐心等待运行状态变更为**运行中**后再进行后续操作。

步骤二：下载CA证书

开启云端证书后，RDS PostgreSQL数据库提供数据库CA证书供您下载，当您通过客户端远程连接RDS PostgreSQL数据库时，使用数据库CA证书可以对数据库真伪进行校验。

1. 切换证书来源为**使用云端证书**，单击**下载CA证书**。

2. 解压下载的CA证书。

   下载的文件为压缩包，包含如下三个文件：

   - p7b文件：用于Windows系统中导入CA证书。
   - PEM文件：用于其他系统或应用中导入CA证书。
   - JKS文件：Java中的truststore证书存储文件，密码统一为apsaradb，用于Java程序中导入CA证书链。

步骤三：客户端连接数据库

本连接示例中以pgAdmin客户端通过SSL连接RDS PostgreSQL数据库为例。

RDS PostgreSQL数据库还支持以下方式SSL远程连接：

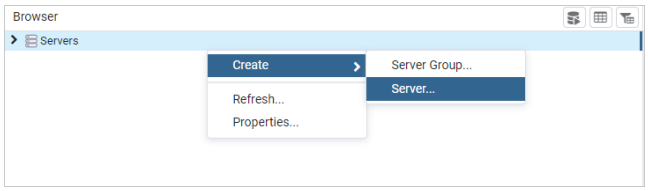
- 使用psql命令行终端SSL连接RDS PostgreSQL数据库
- 使用JDBC SSL连接RDS PostgreSQL数据库

> **说明** 连接数据库前，请确保已在RDS PostgreSQL实例中配置白名单和用户。具体操作，请参见设置白名单和 创建数据库和账号。

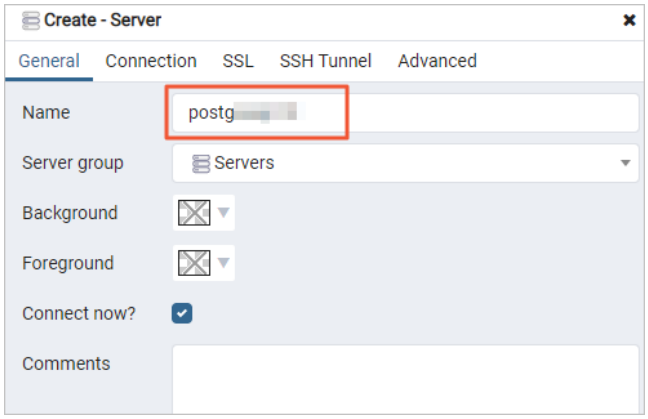
1. 启动pgAdmin 4客户端。

   > **说明** 高版本客户端首次登录需要设置Master Password用于保护保存的密码和其他凭据。

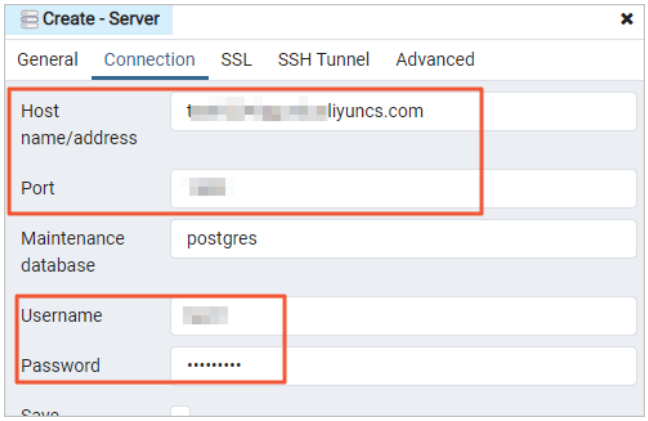
2. 右键单击 Servers，选择Create > Server...。



3. 在General页签设置连接名称。

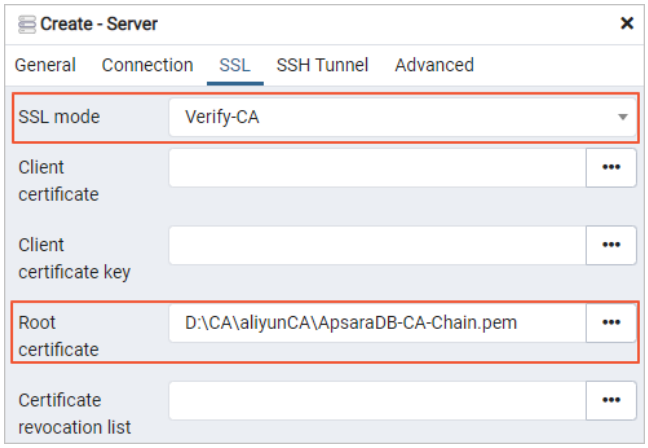


4. 选择Connection标签页，输入要连接的实例信息。



参数	说明
Hostname/address	主机名称/地址。若通过内网连接，需输入RDS实例的内网连接地址；若使用外网连接，需输入RDS实例的外网连接地址。查看RDS实例的内外网地址及端口请参见查看或修改内外网地址和端口。
Port	连接地址对应的端口。
Username	RDS实例的账号名称。创建RDS实例的账号请参见创建数据库和账号。
Password	账号对应的密码。

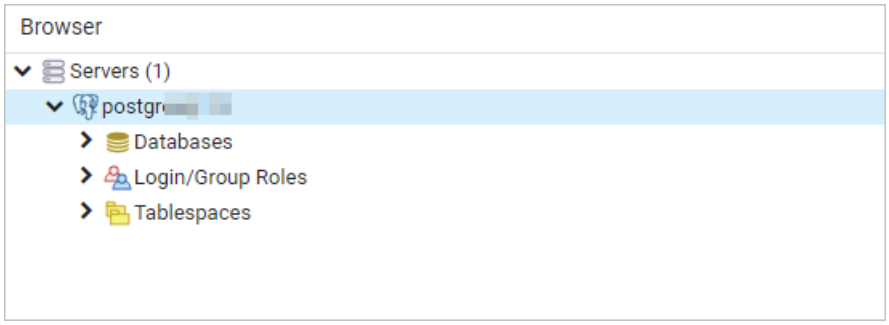
5. 选择SSL标签页，配置SSL认证方式和证书。



参数	说明
SSL mode	为了使用SSL安全连接，请配置Require、Verify-CA或Verify-Full连接方式，参数含义如下： ○ Require：只对数据链路加密，并不验证数据库服务器的真实性。 ○ Verify-CA：加密数据链路，同时验证数据库的真实性。 ○ Verify-Full：加密数据链路，验证数据库的真实性，同时比对证书内的CN或DNS与连接时配置的Host name/address的一致性。
Root certificate	当SSL mode取值为Verify-CA或Verify-Full时，需要配置此参数，表示数据库CA证书路径。 说明 ○ 本示例中，从SSL页面下载的CA证书解压路径为*D:\CA\aliyunCA*，您可以将证书解压到您本地的任意位置。 ○ pgAdmin客户端中使用的数据库CA证书为PEM文件。

6. 单击Save。
   若连 接信息无误，会出现如下界面，则表示连 接成功。

   注意　postgres是RDS实例默认的系统数据库，请勿在该数据库中进行任何操作。



7.

21.3.2. 配置自定义证书

RDS PostgreSQL在配置SSL链路加密时，不仅支持使用阿里云提供的云端证书，还支持使用自定义证书。本文介绍通过自定义证书配置SSL链路加密功能的具体操作。

前提条件

- 实例版本为RDS PostgreSQL 10或以上的云盘版本。
- 已安装OpenSSL工具。

  说明　如果您使用Linux系统，系统会自带OpenSSL工具，无需安装。如果您使用Windows系统，请获取 OpenSSL软件包并安装。

注意事项

- 开启SSL链路加密后，会造成CPU使用率上升、读写时延增长。

- 开启SSL链路加密后，已有连接需要断开重连，加密才会生效。
- 配置自定义证书、更改自定义证书内容或关闭SSL链路加密，将会重启您的数据库实例，用时约3分钟左右，请在业务低峰期操作。

步骤一：获取自定义证书

> **注意** 生成服务器证书私钥或自签名证书私钥时，请勿使用密码加密，否则将无法启用SSL链路加密。

本文以CentOS系统配置为例，如果您使用Windows操作系统，操作步骤中的 `openssl` 命令与CentOS系统中的 `openssl` 命令配置相同， `cp` 命令请替换为拷贝， `vim` 命令请替换为编辑文件。

1. 创建自签名证书（ca.crt）和自签名证书私钥（ca.key）。

```
openssl req -new -x509 -days 3650 -nodes -out ca.crt -keyout ca.key -subj "/CN=root-ca"
```

2. 生成服务器证书请求文件（server.csr）和服务器证书私钥（server.key）

   自定义证书支持对一个连接地址或多个连接地址进行保护。您可以根据需要选择以下两种配置生成服务器证书请求文件。

   - 如果您需要保护一个连接地址，请配置如下命令。

```
openssl req -new -nodes -text -out server.csr -keyout server.key -subj "/CN=pgm-bpxxxxx.pg.rds.aliyuncs.com"
```

> **说明** 本示例中 `pgm-bpxxxxx.pg.rds.aliyuncs.com` 仅用于举例，请修改为需要保护的链接地址，查看连接地址请参见 查看或修改内外网地址和端口。

   - 如果您需要保护多个连接地址，请配置如下命令。

     a. 拷贝openssl配置文件，作为临时使用。

```
cp /etc/pki/tls/openssl.cnf /tmp/openssl.cnf
```

> **说明** 如果您使用Windows系统， *openssl.cnf*文件位于 *openssl安装目录\bin\cnf*目录下，可以将此文件拷贝到任意位置。

     b. 使用如下命令编辑临时拷贝出的openssl.cnf文件。

```
vim /tmp/openssl.cnf
```

     c. 输入 `i` 进入编辑模式，配置以下内容。

```
# 在[ req ]最后补充 req_extensions = v3_req # 新增[ v3_req ] [ v3_req ] basicConstraints = CA:FALSE keyUsage = nonRepudiation, digitalSignature, keyEncipherment subjectAltName = @alt_names # 新增[ alt_names ]，其中DNS后配置需要保护的数据库连接地址 [ alt_names ] DNS.1 = pgm-bpxxxxx.pg.rds.aliyuncs.com DNS.2 = pgm-bpxxxxx.pg.rds.aliyuncs.com
```

     d. 使用 `Esc` 退出编辑模式，输入 `:wq` 保存并退出。

e. 生成服务器证书请求文件（server.csr）和服务器证书私钥（server.key）。

```
openssl req -new -nodes -text -out server.csr -keyout server.key -config /tmp/ope
nssl.cnf
```

生成证书请求文件（server.csr）的过程中，会提示配置以下参数，请根据实际情况配置。

参数	说明	示例
Country Name	ISO国家代码（两位字符）	CN
State or Province Name	所在省份	ZheJiang
Locality Name	所在城市	HangZhou
Organization Name	公司名称	Alibaba
Organizational Unit Name	部门名称	Aliyun
Common Name	申请SSL证书的域名，在openssl.cnf文件中已配置，不需要输入	-
Email Address	不需要输入	-
A challenge password	不需要输入	-
An optional company name	不需要输入	-

3. 生成服务器证书（server.crt）。
   - 如果您需要保护一个连接地址，请配置如下命令。

```
openssl x509 -req -in server.csr -text -days 365 -CA ca.crt -CAkey ca.key -CAcreatese
rial -out server.crt
```

   - 如果您需要保护多个连接地址，请配置如下命令。

```
openssl x509 -req -in server.csr -text -days 365 -CA ca.crt -CAkey ca.key -CAcreatese
rial -out server.crt -extensions v3_req -extfile /tmp/openssl.cnf
```

完成以上配置后，执行ls命令，查看已生成的文件：

```
# ls ca.crt ca.key ca.srl server.crt server.csr server.key
```

关键文件解释如下：

- server.crt：服务器证书文件。
- server.key：服务器私钥文件。
- ca.crt：自签名证书。
- ca.key：自签名证书私钥。

步骤二：使用自定义证书开启SSL链路加密

> **说明** 配置自定义证书后，实例的**运行状态**将会变更为**修改SSL中**，该状态将持续三分钟左右，请耐心等待运行状态变更为**运行中**后再进行后续操作。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 进入**数据安全性**>SSL页面。
3. 切换证书来源为**使用自定义证书**，单击**配置数据库证书（防止伪装数据库）**后的**配置**按钮，填写服务器证书及私钥后，单击**确定**，完成配置。

参数	取值
服务器证书	此参数填写步骤一：获取自定义证书步骤中获取的server.crt文件内容。请填写-----BEGIN CERTIFICATE-----至-----END CERTIFICATE-----间的内容，包括BEGIN和END部分。
服务器证书私钥	此参数填写步骤一：获取自定义证书步骤中获取的server.key文件内容。请填写-----BEGIN PRIVATE KEY-----至-----END PRIVATE KEY-----间的内容，包括BEGIN和END部分。

如果申请证书时配置了多个连接地址，则配置完成后，在**保护主机**参数中，将显示对应多条记录。

步骤三：客户端连接数据库

RDS PostgreSQL数据库支持以下方式通过SSL远程连接：

- 使用pgAdmin客户端通过SSL连接RDS PostgreSQL数据库
- 使用psql命令行终端SSL连接RDS PostgreSQL数据库
- 使用JDBC SSL连接RDS PostgreSQL数据库

步骤四：（可选）更新证书

> **说明** 更新证书会重启实例，请谨慎操作。

如果您需要更新证书，可以在SSL页面单击**配置数据库证书（防止伪装数据库）**后的**修改**按钮，填入新的服务器证书和私钥。

步骤五：（可选）关闭SSL链路加密

> **说明** 关闭SSL加密会重启实例，请谨慎操作。

如果您需要关闭SSL链路加密，可以在**SSL**页面单击**关闭SSL**按钮。

21.3.3. 配置客户端CA证书

如果使用云端证书或自定义证书开启SSL链路加密，则表示在SSL链路中，客户端验证数据库的真实性。如果您需要数据库也验证客户端的真实性，还需要配置客户端CA证书，本文介绍配置客户端CA证书的具体操作。

前提条件

- 已完成快速配置SSL加密或配置自定义证书。
- 已安装OpenSSL工具。

> **说明** 如果您使用Linux系统，系统会自带OpenSSL工具，无需安装。如果您使用Windows系统，请获取 OpenSSL软件包并安装。

注意事项

- 配置客户端CA证书后，已有连接需要断开重连，加密才会生效。
- 配置客户端CA证书、更改客户端CA证书内容、修改客户端证书吊销列表，将会重启您的数据库实例，用时约3分钟左右，请在业务低峰期操作。

步骤一：获取客户端证书

本文以CentOS系统配置为例，如果您使用Windows操作系统，操作步骤中的 `openssl` 命令与CentOS系统中的 `openssl` 命令配置相同。

1. 生成自签名证书（ca1.crt）和自签名证书私钥（ca1.key）。

```
openssl req -new -x509 -days 3650 -nodes -out ca1.crt -keyout ca1.key -subj "/CN=root-ca1"
```

2. 生成客户端证书请求文件（client.csr）和客户端证书私钥（client.key）。

```
openssl req -new -nodes -text -out client.csr -keyout client.key -subj "/CN=<客户端用户名>"
```

> **说明** 该命令中-subj参数后的CN取值请配置为客户端访问数据库的用户名。

3. 生成客户端证书（client.crt）。

```
openssl x509 -req -in client.csr -text -days 365 -CA ca1.crt -CAkey ca1.key -CAcreatese
rial -out client.crt
```

完成以上配置后，执行 `ls` 命令，查看已生成的文件：

```
# ls ca1.crt ca1.key ca1.srl client.crt client.csr client.key
```

关键文件解释如下：

- client.crt：客户端证书文件。
- client.key：客户端私钥文件。
- ca1.crt：自签名证书。
- ca1.key：自签名证书私钥。

步骤二：配置客户端CA证书

> **说明** 配置客户端CA证书后，实例的**运行状态**将会变更为**修改SSL中**，该状态将持续三分钟左右，请耐心等待运行状态变更为**运行中**后再进行后续操作。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 进入**数据安全性**>SSL页面。
3. 单击**启用客户端CA证书**。

4. 将自签名证书文件（ca1.crt）内容填写到对话框中。然后单击**确定**按钮。自签名证书文件（ca1.crt）获取请参见步骤一：获取客户端证书。

步骤三：客户端连接数据库

RDS PostgreSQL数据库支持以下方式通过SSL远程连接：

- 使用pgAdmin客户端通过SSL连接RDS PostgreSQL数据库
- 使用psql命令行终端SSL连接RDS PostgreSQL数据库
- 使用JDBC SSL连接RDS PostgreSQL数据库

步骤四：（可选）配置证书吊销文件

如果您不再使用某一客户端证书时，可以将客户端证书吊销，数据库将拒绝此客户端登录。

> **说明** 配置证书吊销文件后，实例的**运行状态**将会变更为**修改SSL中**，该状态将持续三分钟左右，请耐心等待运行状态变更为**运行中**后再进行后续操作。

1. 配置文件准备。

```
touch /etc/pki/CA/index.txt echo 1000 > /etc/pki/CA/crlnumber
```

> **说明**
> 如果您使用Windows系统，需要按照如下步骤操作：
> i. 在*OpenSSL安装目录\bin*目录下创建CA文件夹。
> ii. 在CA文件夹内创建*index.txt*文件。
> iii. 命令行执行如下命令：
>
> ```
> echo 1000 > <OpenSSL安装目录>\bin\CA\crlnumber
> ```
>
> iv. 修改*C:\Program Files\Common Files\SSL*目录下中的*openssl.cnf*文件。
>
> ```
> # 找到[ CA_default ]配置项 dir = "<OpenSSL安装目录>\\bin\\CA"
> ```

2. 吊销客户端证书（client.crt）。

```
openssl ca -revoke client.crt -cert ca1.crt -keyfile ca1.key
```

> **说明** 该命令需要使用到自签名证书（ca1.crt）及自签名证书私钥（ca1.key），请参见 步骤一：获取客户端证书。

3. 生成证书吊销文件（client.crl）。

```
openssl ca -gencrl -out client.crl -cert ca.crt -keyfile ca.key
```

4. 进入**数据安全性**>**SSL**页面。
5. 单击**启用证书吊销文件**。

6. 将证书吊销文件（client.crl）内容填写到对话框中。

步骤五：（可选）更新客户端证书

> **说明** 更新证书会重启实例，请谨慎操作。

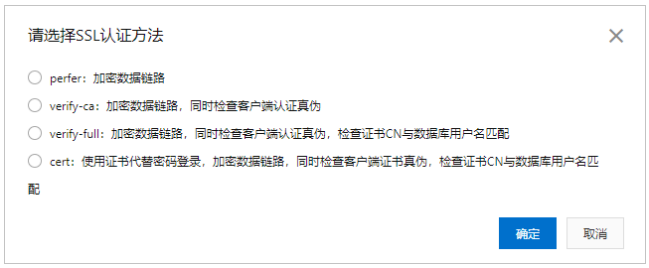
如果您需要更新证书，可以在SSL页面单击**清除客户端CA证书**按钮，清除客户端CA后，重新单击**启用客户端CA证书**。

步骤六：（可选）配置ACL

在配置客户端CA证书后，您可以在RDS PostgreSQL实例中配置ACL控制客户端的访问，此时客户端必须携带客户端证书和私钥并通过客户端对应认证方式的验证才能连接数据库。

> **说明**
> - 配置ACL期间，PostgreSQL数据库实例将不能操作，用时约1分钟左右，请您耐心等待。
> - 如果在RDS PostgreSQL数据库未设置客户端访问控制（默认prefer），客户端可以使用 `PGSSLMODE=disable` 来实现非SSL方式连接数据库。当需要禁止非SSL连接时，请确保在开启SSL链路加密后将ACL配置为除prefer以外的其他认证方式。

单击配置ACL或配置ReplicationACL后的修改，选择不同的客户端连接控制模式。



RDS PostgreSQL支持配置的认证方法及其验证规则如下：

- cert：使用证书代替密码登录，加密数据链路，同时检查客户端证书真伪，检查证书CN与数据库用户名匹配。
- prefer：加密数据链路。客户端使用PGSSLMODE=disable时，可以通过非加密方式连接数据库。
- verify-ca：加密数据链路，同时检查客户端认证真伪。
- verify-full（RDS PostgreSQL 12以上支持）：加密数据链路，同时检查客户端认证真伪，检查证书CN与数据库用户名匹配。

21.3.4. SSL连接RDS PostgreSQL数据库

本文介绍配置了SSL链路加密后，如何通过pgAdmin、psql命令行终端以及JDBC连接数据库。

前提条件

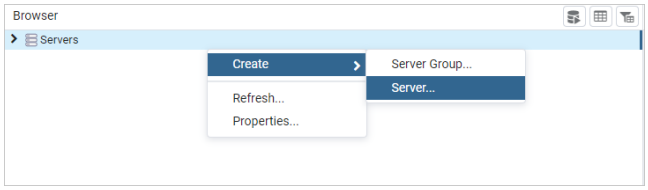
- 已开启SSL链路加密。如果未开启，请参见快速配置SSL加密或配置自定义证书。
- 已获取如下文件。
  - （可选）客户端证书（client.crt）、客户端证书私钥（client.key）：如果在RDS控制台配置了客户端CA证书，则需要准备，获取方法请参见配置客户端CA证书。
  - 数据库CA证书：获取方法请参见下载CA证书或生成自签名证书。

使用pgAdmin客户端通过SSL连接RDS PostgreSQL数据库

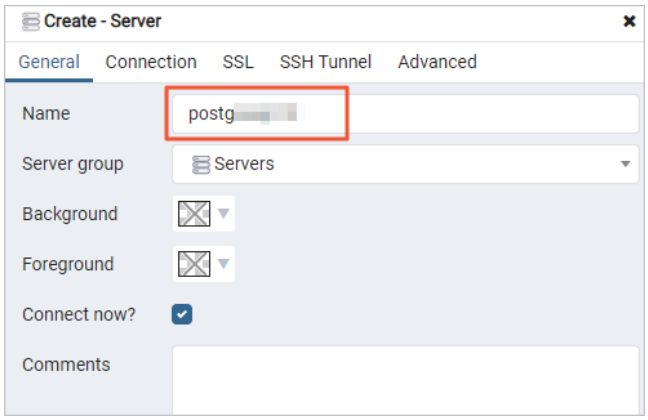
1. 启动pgAdmin 4客户端。

   > **说明** 高版本客户端首次登录需要设置Master Password用于保护保存的密码和其他凭据。

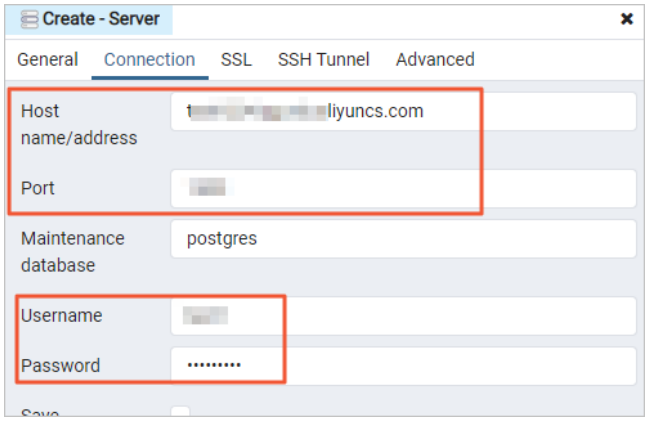
2. 右键单击 Servers，选择Create > Server...。



3. 在General页签设置连接名称。

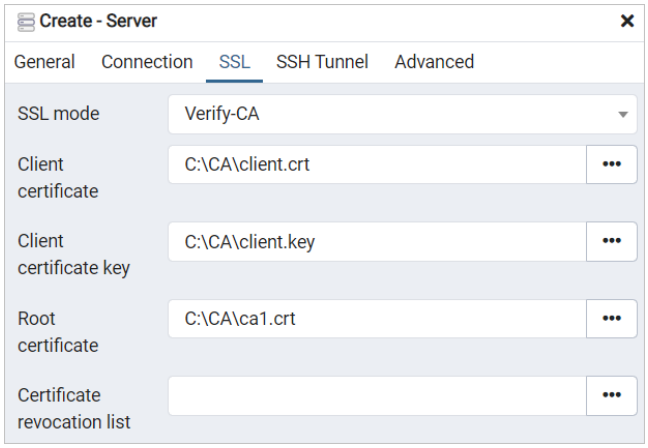


4. 选择Connection标签页，输入要连接的实例信息。



参数	说明
Hostname/address	主机名称/地址。若通过内网连接，需输入RDS实例的内网连接地址；若使用外网连接，需输入RDS实例的外网连接地址。查看RDS实例的内外网地址及端口请参见查看或修改内外网地址和端口。
Port	连接地址对应的端口。
Username	RDS实例的账号名称。创建RDS实例的账号请参见创建数据库和账号。
Password	账号对应的密码。

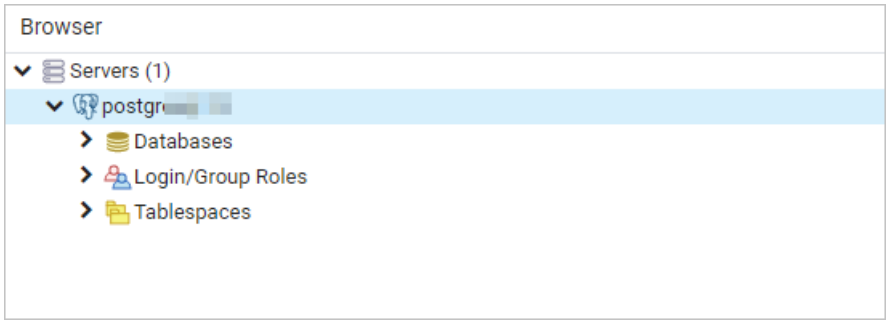
5. 选择SSL标签页，配置SSL认证方式和证书。



参数	说明
SSL mode	PostgreSQL数据库开启SSL链路加密后，表示允许客户端通过SSL连接数据库，在客户端连接数据库时，请根据以下场景选择**SSL mode**参数的取值。 ○ 数据库未配置客户端访问控制 ■ 通过SSL连接时，**SSL mode**选择Require、Verify-CA或Verify-Full。 ■ 不通过SSL连接时，**SSL mode**选择Disable。 ○ 数据库配置了客户端访问控制，客户端必须使用SSL连接，**SSL mode**选择Require、Verify-CA或Verify-Full。 Require、Verify-CA和Verify-Full含义如下： ○ Require：只对数据链路加密，并不验证数据库的真实性。 ○ Verify-CA：加密数据链路，同时验证数据库的真实性。 ○ Verify-Full：加密数据链路，验证数据库的真实性，同时比对证书内的CN或DNS与连接时配置的数据库连接地址是否一致。
Client certificate	如果完成了配置客户端CA证书，则需要配置此参数，表示客户端证书（client.crt）。
Client certificate key	如果完成了配置客户端CA证书，则需要配置此参数，表示客户端证书私钥（client.key）。
Root certificate	当**SSL mode**取值为**Verify-CA**或**Verify-Full**时，需要配置此参数，表示数据库CA证书路径。 **说明** 本示例中，数据库CA证书存放路径为 *C:\CA*，您可以将证书解压到您本地的任意位置。

6. 单击**Save**。
若连 接信息无误，会出现如下界面，则表示连 接成功。

注意 postgres是RDS实例默认的系统数据库，请勿在该数据库中进行任何操作。



7.

使用psql命令行终端SSL连接RDS PostgreSQL数据库

> **说明** 本步骤依赖本地PostgreSQL客户端自带的psql命令行工具，请确保您已安装PostgreSQL客户端，具体请参见PostgreSQL官方文档。

1. 在*var/lib/pgsql*目录下创建*.postgresql*文件夹。

```
mkdir /var/lib/pgsql/.postgresql
```

2. 将如下文件拷贝到*.postgresql*文件夹中。
   - （可选）客户端证书（client.crt）、客户端证书私钥（client.key）：如果在RDS控制台配置了客户端CA证书，则需要配置，获取方法请参见配置客户端CA证书。
   - 数据库CA证书：获取方法请参见下载CA证书或生成自签名证书。

```
cp client.crt client.key ca1.crt /var/lib/pgsql/.postgresql/
```

> **说明** 本示例中，客户端证书（client.crt）、客户端证书私钥（client.key）和数据库CA证书（ca1.crt）存放的路径为当前目录，请根据实际情况配置。

3. 修改.postgresql文件夹权限。

```
chown postgres:postgres /var/lib/pgsql/.postgresql/* chmod 600 /var/lib/pgsql/.postgres
ql/*
```

4. 使用如下命令编辑postgres用户的环境变量。

```
vim /var/lib/pgsql/.bash_profile
```

5. 输入 `i` 进入编辑模式，补充如下内容。

```
export PGSSLCERT="/var/lib/pgsql/.postgresql/client.crt" export PGSSLKEY="/var/lib/pgsq
l/.postgresql/client.key" export PGSSLROOTCERT="/var/lib/pgsql/.postgresql/ca1.crt"
```

6. 使用 `Esc` 退出编辑模式，然后输入 `:wq` 保存并退出。
7. 重新加载环境变量。

```
source .bash_profile
```

8. 设置连接时客户端对数据库的认证方式。

```
export PGSSLMODE="verify-full"
```

PostgreSQL数据库开启SSL链路加密后，表示允许客户端通过SSL连接数据库，在客户端连接数据库时，请根据以下场景选择PGSSLMODE参数的取值。

数据库是否配置客户端访问控制	是否需要通过SSL连接	PGSSLMODE取值
否	是	require、verify-ca或verify-full
	否	disable
是	必须使用SSL	require、verify-ca或verify-full

> **说明**
> require、verify-ca和verify-full含义如下：
> - require：只对数据链路加密，并不验证数据库的真实性。
> - verify-ca：加密数据链路，同时验证数据库的真实性。
> - verify-full：加密数据链路，验证数据库的真实性，同时比对证书内的CN或DNS与连接时配置的数据库连接地址是否一致。

9. 连接数据库。

```
psql -h <数据库连接地址> -U <用户名> -p <端口号> -d <数据库名>
```

各参数在RDS控制台的获取位置：

参数	获取方式
数据库连接地址	加密的连接地址，通过**SSL**页面中**保护主机**参数获取。
用户名	通过**账号管理**页面获取用户名。
端口号	默认5432，如果您修改过，通过**数据库连接**页面获取。
数据库名	postgres是RDS实例默认的系统数据库，请勿在该数据库中进行任何操作。如果您需要连接RDS实例下创建的其他数据库，可从**数据库管理**页面获取。

使用JDBC SSL连接RDS PostgreSQL数据库

1. 下载如下文件到本地。
   - （可选）客户端证书（client.crt）、客户端证书私钥（client.key）：如果在RDS控制台配置了客户端CA证书，则需要获取，获取方法请参见配置客户端CA证书。
   - 数据库CA证书：获取方法请参见下载CA证书或生成自签名证书。
2. 将客户端私钥（client.key）转换为pk8格式。

```
openssl pkcs8 -topk8 -inform PEM -in client.key -outform der -out client.pk8 -v1 PBE-MD5-DES # 需要输入密码，连接时需要使用 Enter Encryption Password: Verifying - Enter Encryption Password:
```

3. 以Maven项目为例，在pom.xml中导入PostgreSQL的Maven依赖。

```
<dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <version>42.2.10</version> </dependency>
```

4. 在业务代码中编写JDBC SSL连接PostgreSQL。

```
// 设置RDS PostgreSQL实例的连接地址 String hostname = "pgm-bpxxxxx.pg.rds.aliyuncs.com"; // 设置RDS PostgreSQL实例的连接端口 String port = "5432"; // 设置待连接的数据库名 String dbname = "postgres"; String jdbcUrl = "jdbc:postgresql://" + hostname + ":" + port + "/" + dbname+"?binaryTransfer=true"; Properties properties = new Properties(); // 设置连接数据库的用户名 properties.setProperty("user", "username"); //设置连接数据库的密码 properties.setProperty("password", "*****"); // 设置证书存放路径 String path= "D:\\ssl\\"; // 配置以ssl访问 properties.setProperty("ssl", "true"); //设置证书授权机构的公钥名 properties.setProperty("sslrootcert", path + "/" + "root.crt"); //设置客户端证书私钥名 properties.setProperty("sslkey", path + "/" + "client.pk8"); //设置客户端证书名 properties.setProperty("sslcert", path + "/" + "client.crt"); //填写将私钥key格式转换为pk8格式时设置的密码 properties.setProperty("sslpassword", "*****"); // 配置ssl模式，可选值为require、verify-ca、verify-full properties.setProperty("sslmode", "verify-ca"); try { Class.forName("org.postgresql.Driver"); Connection connection = DriverManager.getConnection(jdbcUrl, properties); //本示例中，假设在postgres数据库中存在表example，此处以查询表example数据为例。 PreparedStatement preparedStatement = connection.prepareStatement("select * from " + "example"); ResultSet resultSet = preparedStatement.executeQuery(); while (resultSet.next()) { ResultSetMetaData rsmd = resultSet.getMetaData(); int columnCount = rsmd.getColumnCount(); Map map = new HashMap(); for (int i = 0; i < columnCount; i++) { map.put(rsmd.getColumnName(i + 1).toLowerCase(), resultSet.getObject(i + 1)); } System.out.println(map); } } catch (Exception exception) { exception.printStackTrace(); }
```

21.4. 云盘加密

RDS PostgreSQL提供云盘加密功能，能够有效保障您的数据安全。

背景信息

云盘加密能够保障存储的数据安全，您的业务和应用程序无需做额外的改动。同时该云盘生成的快照及这些快照创建的云盘将自动延续加密属性。

云盘加密为免费功能，您在磁盘上的任何读写操作将不会产生额外费用。

前提条件

- 云盘加密只能在创建实例时开启，您必须参照设置云盘加密进行设置，才能在创建实例时设置云盘加密。
- 创建实例时，产品系列、存储类型和规格分类需要按照以下要求配置：

产品系列	存储类型	规格分类
基础版	SSD云盘或ESSD云盘	通用规格 说明 购买页中 **通用型（新）** 页签中的规格不支持云盘加密，具体请参见 RDS PostgreSQL主实例（新通用型）。
高可用版	SSD云盘或ESSD云盘	○ 独享规格 ○ 通用规格（仅 `pg.n2.small.2c` 和 `pg.n2.medium.2c` 支持云盘加密。）

注意事项

- 云盘加密功能开启后无法关闭。
- 开启云盘加密的实例不支持跨地域备份。
- 云盘加密不会影响您的业务，应用程序也无需修改。
- 开启云盘加密后，实例生成的快照以及通过这些快照创建的云盘版实例将自动延续加密属性。
- 密钥管理服务KMS欠费会导致云盘无法解密，整个实例不可用，请确保KMS状态正常。
- 如果您禁用或删除了KMS密钥，将会导致使用了该KMS密钥的RDS实例无法正常工作，受影响的操作包括创建快照、恢复快照和备库重建等。

设置云盘加密

1. 登录密钥管理服务控制台。
2. 左上角选择将要创建实例的地域。
3. 单击**创建密钥**。
4. 设置如下参数。

配置项	说明
KMS实例	KMS实例支持部署到租户的VPC内，满足私有网络接入需求。 更多信息，请参见专属KMS概述。
密钥类型	取值： ○ 对称密钥的类型： ■ Aliyun_AES_256 ■ Aliyun_SM4 ○ 非对称密钥的类型： ■ RSA_2048 ■ RSA_3072 ■ EC_P256 ■ EC_P256K ■ EC_SM2 **说明** Aliyun_SM4或EC_SM2的密钥类型，仅在中国内地使用托管密码机的地域支持。
密钥用途	取值： ○ Encrypt/Decrypt：数据加密和解密。 ○ Sign/Verify：产生和验证数字签名。
别名	用户主密钥的可选标识。 更多信息，请参见别名概述。

配置项	说明
保护级别	取值： ○ Software：通过软件模块对密钥进行保护。 ○ Hsm：将密钥托管在密码机中，使密钥获得高安全等级的专用硬件的保护。
描述	密钥的说明信息。
轮转周期	自动轮转的时间周期。取值： ○ 30天。 ○ 90天。 ○ 180天。 ○ 365天。 ○ 不开启：不开启轮转。 ○ 自定义：7~730天。 **说明** 仅密钥类型为Aliyun_AES_256和Aliyun_SM4的对称密钥支持设置轮转周期。

5. 单击**确定**。
6. （仅第一次使用需要授权）单击云资源访问授权，单击**同意授权**让PostgreSQL可以访问您的云资源。

   说明 仅第一次使用需要授权，您可以在 访问控制RAM控制台查看是否具有AliyunRDSInstanceEncryptionDefaultRole权限。

7. 接下来您可以在创建实例时选择云盘加密，详情请参见创建RDS PostgreSQL实例。

   说明 创建实例后您可以在实例基本信息页面查看到云盘加密的密钥。

21.5. 全加密云数据库

21.5.1. 什么是全加密功能

全加密功能是阿里云数据库的一项数据安全功能。在RDS PostgreSQL实例中开启并使用该功能，您可以对数据库表中的敏感数据列进行加密，这些列中的敏感数据将以密文进行传输、计算和存储。

功能介绍

传统加密技术往往只能保护数据在传输过程中或者在磁盘中存储时的安全，而全加密功能使用基于**Intel SGX 安全增强型**实例的可信执行环境技术对内存中的数据进行加密并进行计算，全程保护数据免受攻击者的窃取。

相比传统的加密方式，全加密功能为用户在RDS PostgreSQL数据库中存储敏感数据提供了更高的数据安全保护。

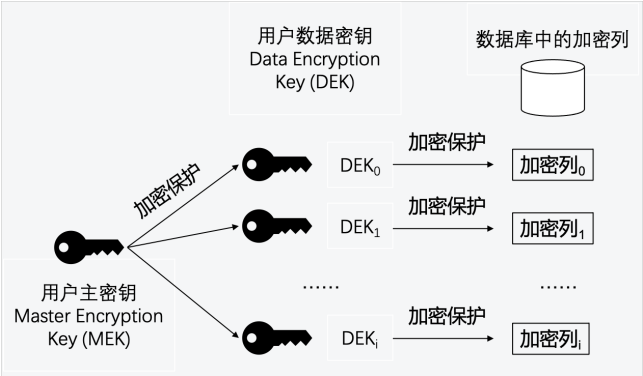
对比项	透明数据加密（TDE）	SSL链路加密	云盘加密	数据库全加密（本功能）
加密方式	在数据被写入存储设备或存储服务时对数据进行加密。	在客户端和数据库实例之间对传输的数据进行加密。	针对RDS云盘版实例，阿里云免费提供云盘加密功能，基于块存储对整个数据盘进行加密。	在全链路（从客户端至数据库实例，含网络、内存和存储）中对数据进行加密。
明文数据可见范围	数据在参与计算前需要被解密，所以其在CPU、缓存和内存中是明文形式。	数据在数据库实例中（含内存和存储）均为明文。	数据在云盘之外时是明文。	数据全程为密文形式，仅在 **Intel SGX 安全增强型** 规格实例内参与计算时被解密为明文。
安全性	保护存储中的数据。	保护数据的通信不被窃听。	保护云盘中数据的安全。	保护全链路中的数据安全。

说明 **Intel SGX 增强型**规格采用Intel CPU的软件防护扩展（Software Guard Extensions，简称SGX）提供可信执行环境（Trusted Execution Environment，简称TEE）来保护被加密的数据及其处理过程。具体规格清单请参见 RDS PostgreSQL主实例规格列表。

加密原理

在RDS PostgreSQL实例中，您可以使用用户主密钥（Master Encryption Key，简称MEK）和用户数据密钥（Data Encryption Key，DEK）对数据库中的列进行加密，保护敏感数据的数据安全。

- 用户数据密钥对选中的数据库列进行加密。如有多个列需要被加密保护，您可以使用一个或多个DEK分别对它们进行加密。数据密钥可以在您授权的情况下被第三方访问使用，但不能被获得。
- 用户主密钥用于对用户数据密钥本身进行加密保护。请您妥善保管，在任何场景中都不应被第三方获得或访问。



在全加密功能中，除数据所有者外的任何人员无法获得用户主密钥和用户数据密钥，从而无法对数据库中的加密列进行解密。

- 对于数据所有者：拥有用户主密钥和数据密钥，可以在客户端一侧使用密钥对数据进行加密，可对数据库执行各类查询和事务，可以看到经过密钥解密的查询结果。
- 对于数据库运维人员、云平台软件、DBA等内部角色： 任何人无法获得用户主密钥，未经授权无法访问用户数据密钥 。可对数据库进行维护或使用数据库的其它功能，但无法查看加密列的明文。

实现方案

全加密功能根据实例类型不同，可通过以下两种方案实现：

- **Intel SGX 安全增强型**实例：全加密功能可以利用该规格为用户提供基于可信执行环境（Trusted Execution Environment，简称TEE）的数据安全保护。此时，全加密功能支持对密文数据进行所有的数据库操作。
  可信执行环境依赖**Intel SGX 安全增强型**规格提供对密文数据进行比较和其它常见计算，具体规格清单请参见RDS PostgreSQL主实例规格列表。
- 其它规格（非**Intel SGX 安全增强型**规格）：全加密功能可以利用密码学方法提供数据安全保护，此时其能够支持的操作类型较少。

全加密功能可以部署在所有实例规格之上，但推荐您选择**Intel SGX 安全增强型**规格，以获得最完整的全加密功能支持。

Intel SGX 安全增强型规格实例与其它规格实例的功能对比如下：

对比项		Intel SGX 安全增强型规格	其它规格
数据定义语言（DDL）		✔☺	✔☺
加密数据类型		✔☺ **说明** 支持7种加密数据类型。	✔☺ **说明** 仅支持rnd_type和det_type两种加密数据类型。
加密算法	确定性加密算法（Deterministic Encryption）	✔☺	✔☺
	随机加密算法（Randomized Encryption）	✔☺	✔☺
SQL查询与事务	对加密数据进行等值判断	✔☺	✔☺ **说明** 仅支持确定性加密的数据的等值比较，安全性较差。
	对加密数据进行比较	✔☺	▯
	对加密数据进行其它计算（表达式求值、`LIKE`等）	✔☺	▯
	功能支持（如`JOIN`、`ORDER BY`、`GROUP BY`等）	✔☺	✔☺ **说明** 仅支持确定性加密的数据的`EQUAL JOIN`操作。
	加密SQL查询性能优化	✔☺	▯

21.5.2. 配置步骤

本文介绍如何实现全加密云数据库功能。

前提条件

- RDS PostgreSQL服务端
  - 已创建云数据库RDS PostgreSQL 10、11、12、13或14版本实例，创建方式请参见创建RDS PostgreSQL实例。
  - 已获取RDS PostgreSQL实例的内外网地址，获取方式请参见查看内外网地址。
  - 已创建数据库高权限账号，创建方式请参见创建账号。
- 本地客户端
  - 已具备Java 1.8+的开发环境。
  - 已根据服务端版本下载对应EncDB依赖包。

RDS PostgreSQL小版本	服务端encdb插件版本号	客户端EncDB依赖包
20210531	1.1.5	libencdb-1.2.1.jar
20210930	1.1.6	libencdb-1.2.4.jar
20211031 ~ 20220130	1.1.7	libencdb-1.2.5.jar
20220228或以上版本	1.1.9	libencdb-1.2.6.jar

  - 已生成用户主密钥。

> **说明**
> - 用户主密钥是您访问加密数据的根凭据，全加密功能不提供用户主密钥的生成和备份服务，您需要自行生成用户主密钥。一旦您丢失密钥，将无法再访问已有的数据。因此我们建议您妥善备份用户主密钥。
> - 用户主密钥由长度为16字节的16进制字符串组成，本示例以 `0x00112233445566778899aabbccddeeff` 为例。常见的生成方法有：密码生成工具（如openssl， `openssl rand -hex 16` ）、编程语言中的random函数。

服务端配置步骤

全加密云数据库在RDS PostgreSQL服务端以扩展（Extension）形式部署。本步骤介绍如何在服务器端安装扩展以及创建加密表。

> **说明** 本步骤需要连接RDS PostgreSQL实例进行操作，连接方法请参见 连接PostgreSQL实例。

1. 安装EncDB扩展。

```
CREATE EXTENSION encdb;
```

2. 创建加密表。

   创建加密表时，您可以根据自身需要，将敏感字段的数据类型替换为对应的加密数据类型。以下表为例，其原始定义如下：

```
CREATE TABLE example ( id INTEGER, name VARCHAR, price INTEGER, miles REAL, secret TEXT
, PRIMARY KEY (id) );
```

如果选择price, miles, secret为敏感数据，创建的加密表定义如下：

```
CREATE TABLE example ( id INTEGER, name VARCHAR, price rnd_type, miles det_type, secret
rnd_type, PRIMARY KEY (id) );
```

> **说明**
> - 全加密数据库功能目前支持det_type和rnd_type两种加密数据类型。其中det_type为确定性加密（同一个敏感数据两次加密得到的密文相同），rnd_type为不确定性加密（同一个敏感数据两次加密得到的密文不同）。
> - 当您在应用程序中使用det_type和rnd_type时，需要将全加密数据库det_type和rnd_type对应的宿主变量转换为string数据类型。
> - 您不能直接修改原有表（即 `ALTER TABLE` ），而是需要新建一个加密表，然后再将原表数据通过SDK加密后导入。

客户端配置步骤

本步骤介绍客户端如何连接数据库并进行数据加密解密操作。

> **说明** 本示例以Maven构建Java项目为例。

1. 安装EncDB依赖包。

```
mvn install:install-file -DgroupId=com.alibaba.encdb -DartifactId=<安装的Jar包名> -Dvers
ion=<安装的Jar版本> -Dpackaging=jar -Dfile=<下载的Jar包所在路径>/<安装的Jar包名.jar>
```

示例：

```
mvn install:install-file -DgroupId=com.alibaba.encdb -DartifactId=libencdb -Dversion=1.
2.4 -Dpackaging=jar -Dfile=D:\encdb\libs\libencdb-1.2.4.jar
```

> **说明** 此示例中EncDB依赖包在 *D:\encdb\libs*路径下。

2. 添加依赖。

您需要在pom.xml中添加如下依赖：

```
 <dependency> <groupId>com.alibaba.encdb</groupId> <artifactId>libencdb</artifactId> <v
ersion>1.2.4</version> </dependency> <!-- https://mvnrepository.com/artifact/org.bouncy
castle/bcprov-jdk15on --> <dependency> <groupId>org.bouncycastle</groupId> <artifactId>
bcprov-jdk15on</artifactId> <version>1.62</version> </dependency> <!-- https://mvnrepos
itory.com/artifact/org.bouncycastle/bcpkix-jdk15on --> <dependency> <groupId>org.bouncy
castle</groupId> <artifactId>bcpkix-jdk15on</artifactId> <version>1.62</version> </depe
ndency> <!-- https://mvnrepository.com/artifact/com.alibaba/fastjson --> <dependency> <
groupId>com.alibaba</groupId> <artifactId>fastjson</artifactId> <version>1.2.62</versio
n> </dependency> <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency> <groupId>org.postgresql</groupId> <artifactId>postgresql</artifactId> <ver
sion>42.2.23</version> </dependency> <dependency> <groupId>com.google.guava</groupId> <
artifactId>guava</artifactId> <version>24.1.1-jre</version> </dependency>
```

3. 全加密功能使用示例。

   i. 连接数据库。

> **说明** 此示例中MEK仅为测试用，实际应用中请使用自己的MEK。参数 Mek 和 EncAlgo 只作为EncDB SDK的加解密模块使用，不会直接传送给服务器端。

```
try { Class.forName("org.postgresql.Driver"); } catch (ClassNotFoundException e) { e.printStackTrace(); } String hostname = "pgm-uf6qw1ap115gr2****.pg.rds.aliyuncs.com"; String port = "5432"; String dbname = "encdbtest"; String username = "username"; String password = "userpassword"; String tableName = "example"; String mek = "0x0011223344556677 8899aabbccddeeff"; String dbUrl = "jdbc:postgresql://" + hostname + ":" + port + "/" + dbname + "?binaryTransfer=true"; try { dbConnection = DriverManager.getConnection(dbUrl, username, password); } catch (SQLException throwables) { throwables.printStackTrace(); } /*创建EncdbSDK实例*/ EncdbSDK sdk = EncdbSDKBuilder.newInstance() .setDbConnection(dbConnection) .setMek(mek) .setEncAlgo(Constants.EncAlgo.SM4_128_CBC) .setDekGenMode(Constants.DekGenMode.LOCAL) .setSdkMode(Constants.SDKMode.Crypto) .build(); Cryptor crypto = sdk.getCryptor();
```

上述示例中关键参数解释及取值示例如下：

参数	取值示例（字符串类型）	说明
Mek	0x0011223344556677 8899aabbccddeeff	用户主密钥。 取值范围：长度为16字节的16进制字符串。
EncAlgo	SM4_128_CBC	加密算法。 取值范围： ■ AES_128_GCM ■ AES_128_CBC ■ SM4_128_CBC（默认值）
DekGenMode	LOCAL	生成数据密钥的方式。固定为LOCAL，不能修改。
SdkMode	Crypto	SDK模式。固定为Crypto，不能修改。

   ii. 插入加密数据。

```
PreparedStatement stmt = dbConnection.prepareStatement("insert into " + tableName + " (id, name, price, miles, secret) values(?,?,?,?,?)"); int price = 1234; float miles = 12.34f; String secret = "aliyun"; stmt.setInt(1, 1); stmt.setString(2, "name"); stmt.setBytes(3, crypto.encrypt(tableName, "price", String.valueOf(price))); stmt.setBytes(4, crypto.encrypt(tableName, "miles", String.valueOf(miles))); stmt.setBytes(5, crypto.encrypt(tableName, "secret", secret)); stmt.execute();
```

iii. 查询解密数据。

```
 String sqlCmd = "SELECT * from " + tableName + " where miles = ?"; PreparedStateme
nt stmt = dbConnection.prepareStatement(sqlCmd); stmt.setBytes(1, crypto.encrypt(ta
bleName, "miles", "12.34")); ResultSet rs = stmt.executeQuery(); while (rs.next())
{ int id = rs.getInt(1); String name = rs.getString(2); int price = Integer.parseIn
t(crypto.decryptString(rs.getBytes(3))); float miles = Float.parseFloat(crypto.decr
yptString(rs.getBytes(4))); String text = crypto.decryptString(rs.getBytes(5)); Sys
tem.out.println(id + ", " + name + ", " + price + ", " + miles + ", " + text); }
```

查询结果：

- 已加密的数据在客户端查询时被解密，您可以查询到明文。

```
1, name, 1234, 12.34, aliyun
```

- 已被加密的数据在服务器端无法查看，能够有效防御来自云平台外部和内部的安全威胁，时刻保护用户数据。

```
select * from example;
```

```
 id | name | price | miles | secret ----+------+-----------------------------------
-----------------------------------+-----------------------------------------------
---------------------------+-------------------------------------------------------
----------------- 1 | name | \xdf4901df087c6a3e0325175bb76942c684191a8dda2a8d0c35f295
dc1e30eaeaa0c0e3 | \x315102ea5ab8a659066ab672e6dfbfd89a3a2b360bf6efe3787931e00f61af05
f7408c | \xed4903dfd1bda1a89ad6aa7e8905c0e6305e15db4bc9ce2d2cfac9e25094d2a3ed367d (1
行记录)
```

22.日志/历史事件

22.1. SQL审计（数据库审计）

您可以通过RDS PostgreSQL的SQL审计功能查看SQL明细、定期审计SQL。开通SQL审计功能后，实例性能不会受到影响。

注意事项

- 开启SQL审计功能之前的记录无法查看到。
- 开通SQL审计功能后，实例性能不会受到影响。
- SQL审计的保存时间为30天。
- SQL审计导出的文件可以保存2天，超过2天的会被系统定时清理。
- SQL语句长度限制为8000字节，超过的部分无法记录。

计费

- SQL审计：该功能默认关闭，开启后实例会产生额外费用。
  收费标准：按小时扣费，不同地域的实例有不同的计费。
  - 0.15美元/（GB*小时）：中国香港、美国（硅谷）、美国（弗吉尼亚）。
  - 0.18美元/（GB*小时）：新加坡、日本（东京）、德国（法兰克福）、阿联酋（迪拜）、澳大利亚（悉尼）、马来西亚（吉隆坡）、印度（孟买）、印度尼西亚（雅加达）、英国（伦敦）。
  - 0.12美元/（GB*小时）：除上述地域外的其他所有地域。
- SQL洞察和审计：开通SQL洞察和审计之后，原SQL审计将停止计费，新版本的SQL洞察和审计将按照DAS专业版进行收费。详细的收费信息，请参见DAS专业版费用说明。

开启 SQL洞察和审计

RDS PostgreSQL目前已支持SQL洞察和审计，增强了**SQL明细搜索**功能，新增了**全量SQL分析**、**SQL洞察-对比查看**和**SQL洞察-来源分析**等功能，您可以在打开**SQL审计**页面时跳出的对话框中查看两者的功能对比以及计费差异。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**数据安全性**。
3. 单击**SQL审计**页签，在弹出的**SQL洞察和审计**对话框中单击**一键升级**。

> **说明** 为了功能保障持续可用及审计数据安全，系统开通功能时默认自动续费。
> - 如果需要关闭自动续期，您可以在**费用 > 续费管理**中进行取消，具体请参见自动续费中的**关闭自动续费**。
> - 如果需要自行配置收费项，您可以参见DAS专业版使用说明手工开通DAS专业版，并参见接入阿里云数据库实例接入目标实例后使用。

开启SQL审计

如果不需要开启**SQL洞察和审计**，使用旧版**SQL审计**功能，请参见如下步骤：

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**数据安全性**。
3. 单击**SQL审计**页签，在弹出的**SQL洞察和审计**对话框中单击**取消**。

4. 单击**立即开启**。

> **说明** 旧版控制台请单击 **开启SQL审计**。

5. 在弹出的确认框中单击**确定**。

开启SQL审计后，您可以通过时间、DB、User、关键字等条件查询SQL信息。

> **说明** 系统是从数据库内采集csvlog文件内容，然后保存到日志服务器后提供给您查看，如果需要调整数据库内csvlog文件的最大保留个数，您可以调整实例参数 rds_max_log_files。

关闭SQL审计

为节约成本，您可以在不需要审计SQL时关闭SQL审计功能，详细步骤如下。

> **说明** SQL审计功能关闭后，包括历史审计内容在内的SQL审计记录会被清空。高可用版实例请将SQL审计内容导出并妥善保存至本地后，再关闭SQL审计功能。基础版实例暂不支持导出审计文件。

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**数据安全性**。
3. 单击**SQL审计**页签，在弹出的**SQL洞察和审计**对话框中单击**取消**。
4. 单击**导出文件**，将SQL审计内容导出并妥善保存至本地。
5. 导出文件后单击**关闭SQL审计**。

6. 在弹出的确认框中，单击**确定**。

22.2. 查看日志

您可以通过控制台或SQL命令查询实例的错误日志和慢日志，帮助故障定位分析。

> **说明** 本文所述的日志是指错误日志和慢日志。关于归档日志，请参见 备份PostgreSQL数据和 下载数据备份和日志备份。

注意事项

无。

查看日志

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中单击**日志管理**。
3. 在**日志管理**页面选择查询错误日志或者慢日志明细，选择时间范围，单击**查询**。

查询项	内容
错误日志	记录1个月内数据库运行出错的日志。
慢日志明细	记录1个月内数据库中执行时间超过1s的SQL语句，并进行相似语句去重。

> **说明**
> - 华北3（张家口）的实例仅保留最近9天的错误日志和慢日志明细。
> - 系统是从数据库内采集csvlog文件内容，然后保存到日志服务器后提供给您查看，如果需要调整数据库内csvlog文件的最大保留个数，您可以调整实例参数rds_max_log_files。保留的csvlog文件个数越多，可以查询的时间越长，但是占用空间也会越大。

相关API

API	描述
DescribeSlowLogs	查看慢日志列表
DescribeSlowLogRecords	查看慢日志明细
DescribeErrorLogs	查看错误日志
DescribeBinlogFiles	查看BINLOG日志
DescribeSQLLogRecords	查询SQL审计日志
DescribeSQLLogFiles	查询SQL审计文件列表

22.3. 历史事件

RDS提供历史事件功能，开启后您可以查看用户和阿里云的运维操作日志，例如在某个时间创建了实例、修改了参数。

计费

公测期间0折优惠，公测结束后如果收费会另行通知。

使用场景

- 实例管理动作追踪。
- 实例操作安全审计。
- 审计云服务提供商管理操作合规性等，例如金融、政务等安全性要求高的行业的审计合规需求。

查看历史事件

1. 登录RDS管理控制台，在左侧单击**事件管理**，然后在上方选择地域。

2. 切换到**历史事件**页签。

页面介绍

RDS的历史事件页面会展示所处地域已发生事件的详细信息，包括资源类型、资源名称、事件类型等等。详细说明如下。

参数	说明
资源类型	RDS资源的类型，当前资源类型仅有**实例**。
资源名称	RDS资源的名称，**资源类型**为**实例**时，**资源名称**列显示实例ID。
事件类型	事件的类型，包括**实例管理**、**数据库管理**、**读写分离**、**网络**等等。详情请参见事件列表。
事件操作	相应事件类型内的具体操作。例如**实例管理**内有**创建**、**删除**、**变配**、**重启**等等。详情请参见事件列表。
执行时间	事件的执行时间。
事件来源	事件的发起者，分为如下三类： • 用户（USER）：通过控制台、API发起操作。 • 系统（SYSTEM）：自动运维操作、周期性的系统任务。 • 内部运维人员（SYSTEM_USER）：通过运维系统发起操作。
事件原因	事件产生的原因，便于您了解非预期事件产生的原因。分为如下两类： • 用户通过控制台、API发起的操作（FROM_USER）。 • 系统或人工运维操作（FROM_SYSTEM_USER和FROM_SYSTEM）。
用户信息	事件执行者的账号ID。
参数信息	用户在控制台发起操作的请求参数。

说明

- 历史事件的展示有延迟，大约为5分钟。
- 历史事件是分地域展示的，如果需要看其他地域的历史事件，请切换地域后进行查看。

事件列表

事件类型（EventType）	事件操作（EventName）
实例管理	重启（RestartDBInstance）
	续费（RenewInstance）
	变配（ModifyDBInstanceSpec）
	可用区迁移（MigrateToOtherZone）
	收缩日志（PurgeDBInstanceLog）
	内核版本升级（UpgradeDBInstanceEngineVersion）
	修改实例备注（ModifyDBInstanceDescription）
	修改可维护时间（ModifyDBInstanceMaintainTime）
	创建只读实例（CreateReadOnlyDBInstance）
	销毁实例（DestroyDBInstance）
	修改内核版本升级模式（ModifyDBInstanceAutoUpgradeMinorVersion）
	修改实例参数 （ModifyParameter）
CloudDBA	创建诊断报告（CreateDiagnosticReport）
数据库管理	创建数据库（CreateDatabase）
	删除数据库（DeleteDatabase）
	修改数据库备注（ModifyDBDescription）
	实例间复制数据库（CopyDatabaseBetweenInstances）
	修改系统库的字符集排序规则和时区（ModifyCollationTimeZone）
读写分离	创建读写分离地址（AllocateReadWriteSplittingConnection）
	查询系统权重分配值（CalculateDBInstanceWeight）
	调整读写分离策略（ModifyReadWriteSplittingConnection）
	释放读写分离地址（ReleaseReadWriteSplittingConnection）
安全	开启白名单高安全模式（MigrateSecurityIPMode）
	开启SSL（ModifyDBInstanceSSL）
	开启TDE（ModifyDBInstanceTDE）
	修改白名单（ModifySecurityIps）

事件类型（EventType）	事件操作（EventName）
账号	创建账号（CreateAccount）
	删除账号（DeleteAccount）
	授权账号访问数据库（GrantAccountPrivilege）
	撤销账号对数据库的访问权限（RevokeAccountPrivilege）
	修改数据库账号的描述（ModifyAccountDescription）
	重置账号密码（ResetAccountPassword）
	重置高权限账号的权限（ResetAccount）
高可用	触发切换主备实例（SwitchDBInstanceHA）
	修改高可用模式（ModifyDBInstanceHAConfig）
网络	申请外网地址（AllocateInstancePublicConnection）
	修改连接地址过期时间（ModifyDBInstanceNetworkExpireTime）
	修改实例的连接地址和端口（ModifyDBInstanceConnectionString）
	切换实例网络类型（ModifyDBInstanceNetworkType）
	释放实例的外网连接地址（ReleaseInstancePublicConnection）
	切换内外网地址（SwitchDBInstanceNetType）
日志管理	开启或关闭审计日志（ModifySQLCollectorPolicy）
备份恢复	创建数据备份（CreateBackup）
	克隆实例（CloneDBInstance）
	创建临时实例（CreateTempDBInstance）
	修改备份策略（ModifyBackupPolicy）
	恢复备份集到原实例（RestoreDBInstance）
	删除数据备份（DeleteBackup）
	恢复数据库（RecoveryDBInstance）
跨地域备份恢复	跨地域恢复数据到新实例（CreateDdrInstance）
	修改RDS跨地域备份设置（ModifyInstanceCrossBackupPolicy）
	将OSS上的备份文件还原到RDS实例（CreateMigrateTask）

SQL Server备份上云事件类型（EventType）	事件操作（EventName）
	在备份数据上云时打开数据库（CreateOnlineDatabaseTask）
监控管理	修改监控频率（ModifyDBInstanceMonitor）
数据迁移	创建SQL Server数据上传信息（CreateUploadPathForSQLServer）
	从其它RDS迁入数据（ImportDatabaseBetweenInstances）
	取消RDS实例迁移任务（CancelImport）
标签管理	为实例绑定标签（AddTagsToResource）
	解绑标签（RemoveTagsFromResource）

相关API

API	描述
查询历史事件	调用DescribeEvents接口查询RDS事件记录列表。
查询历史事件是否开启	调用DescribeActionEventPolicy接口查看RDS历史事件功能开启情况。
开关历史事件	调用ModifyActionEventPolicy接口开启或关闭RDS历史事件功能。

23.待处理事件

当RDS出现待处理事件时，会在控制台提醒您及时处理。
RDS运维事件（如实例迁移、版本升级等）除了在短信、语音、邮件或站内信通知之外，还会在控制台进行通知。您可以查看具体的事件类型、地域、流程和注意事项，以及涉及的实例列表，也可以手动修改计划切换时间。

前提条件

有未处理的运维事件。

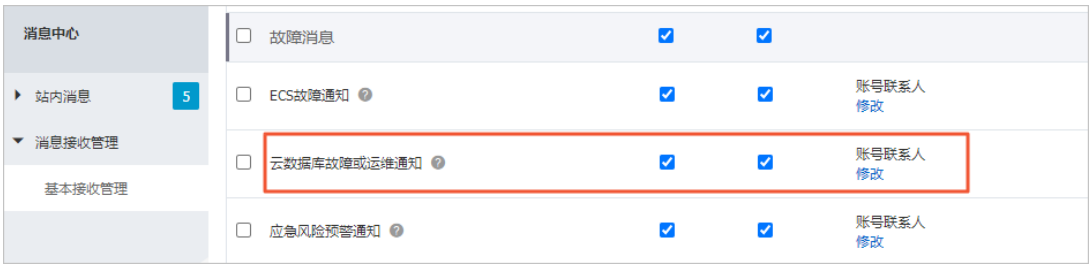
> **说明** 您可以在控制台右上角 **待处理事件**看到提醒。

注意事项

云数据库的待处理事件（如实例迁移、版本升级等）通常至少在执行前的3天通知您，通知方式为语音、邮件、站内信或控制台等。紧急风险修复类事件由于需要尽快修复，通常会在执行前的3天或更短的时间内发送通知并修复。

您需要登录消息中心，确保**云数据库故障或运维通知**的通知开关处于开启状态并设置消息接收人（推荐设置为数据库运维人员），否则您将无法收到相应的通知信息。

消息中心通知设置



操作步骤

1. 登录RDS管理控制台。
2. 在左侧导航栏单击**事件中心**或在右上角单击**待处理事件**。

   > **说明** 强制要求预约时间的运维事件会弹窗提醒，请尽快完成预约。

3. 在**待处理事件**页面选择相应的事件类型、地域。

   > **说明** 不同的 **事件类型**页面会有不同的通知信息，包括该事件的流程和注意事项。

4. 在下方实例列表查看事件的详细信息，如需修改**计划切换时间**，请在左侧勾选对应的实例，然后单击**自定义操作时间**，在弹出的对话框中设置时间并单击**确定**。

   > **说明**
   > - 不同类型的事件显示的信息有所不同，请以实际页面为准。
   > - **计划切换时间**不能晚于**最晚操作时间**。

事件的原因与影响

事件原因	影响类型	影响说明
实例迁移 主备切换 SSL证书更新 备份模式升级	实例闪断	进入 后，将产生下述影响： • 实例或实例中涉及切换的分片将发生连接闪断及30秒以内的只读状态（用于等待数据完全同步），请在业务低峰期执行，并确保应用程序具备重连机制。 • 短暂影响该实例在DMS和DTS中的使用，操作完成后自动恢复正常。 计划切换时间
小版本升级	实例闪断	进入 后，将产生下述影响： • 实例或实例中涉及切换的分片将发生连接闪断及30秒以内的只读状态（用于等待数据完全同步），请在业务低峰期执行，并确保应用程序具备重连机制。 • 短暂影响该实例在DMS和DTS中的使用，操作完成后自动恢复正常。
	小版本号间的差异	不同的小版本号（内核版本号）更新的内容有所区别，您需要关注升级后的小版本和当前小版本的差异，具体请参见相关产品的小版本更新日志（部分产品暂未开放小版本更新日志）： • 云数据库RDS：RDS MySQL（AliSQL）小版本更新日志、RDS PostgreSQL（AliPG）小版本更新日志、SQL Server小版本更新日志。 • 云数据库PolarDB：PolarDB MySQL小版本更新日志、PolarDB O小版本更新日志、PolarDB PostgreSQL小版本更新日志。
代理小版本升级	实例闪断	进入 后，将产生下述影响： • 实例或实例中涉及切换的分片将发生连接闪断及30秒以内的只读状态（用于等待数据完全同步），请在业务低峰期执行，并确保应用程序具备重连机制。 • 短暂影响该实例在DMS和DTS中的使用，操作完成后自动恢复正常。
	小版本号间的差异	不同的小版本号更新的内容有所区别，您需要关注升级后的小版本和当前小版本的差异。
网络升级	实例闪断	进入 后，将产生下述影响： • 实例或实例中涉及切换的分片将发生连接闪断及30秒以内的只读状态（用于等待数据完全同步），请在业务低峰期执行，并确保应用程序具备重连机制。 • 短暂影响该实例在DMS和DTS中的使用，操作完成后自动恢复正常。
	VIP直连影响	部分网络升级过程中可能涉及跨可用区迁移，实例的虚拟IP（VIP）地址会发生改变，如果客户端使用VIP连接云数据库将会引起连接中断。 **说明** 为避免影响，您应当使用实例提供的域名形式的连接地址，同时关闭应用及其所属服务器的DNS缓存。

24.备份

24.1. 备份费用

本文介绍RDS PostgreSQL的备份费用。

费用说明

如果备份大小未超过免费额度，备份不收费。超过后，**备份的每小时费用 = (备份大小 - 免费额度) x 备份单价**。

备份模式	备份大小	免费额度（GB，只入不舍）	备份单价（$/GB）
快照备份	备份大小=数据备份大小+日志备份大小。 在实例的基本信息页的右下角查看。	云盘实例：存储空间的200%	0.00004
物理备份		本地盘实例：存储空间的50%	0.00020

注意

备份费用和备份大小有关，和存储空间使用量无关，因为备份不占用存储空间。

分析备份费用时，请检查备份大小，而不是存储空间使用量。

如何减少备份的费用

提高免费额度

即扩容存储空间，具体请参见变更配置。

免费额度与存储空间有关，例如，把本地盘实例的存储空间从150GB扩容至300GB，免费额度会从75GB升至150GB。

24.2. 备份PostgreSQL数据

RDS会按照默认的备份设置自动备份数据，您也可以修改自动备份设置或手动备份RDS数据。

注意事项

- 实例备份文件占用备份空间，空间使用量超出免费的额度将会产生额外的费用，请合理设计备份周期，以满足业务需求的同时，兼顾备份空间的合理利用。关于免费额度详情，请参见查看备份空间免费额度。
- 关于具体的计费方式与收费项，请参见收费项、计费方式与价格。
- 关于备份空间使用量的计费标准，请参见备份费用。
- 备份期间不要执行DDL操作，避免锁表导致备份失败。
- 尽量选择业务低峰期进行备份。
- 若数据量较大，花费的时间可能较长，请耐心等待。
- 备份文件有保留时间，请及时下载需要保留的备份文件到本地。

备份说明

数据库类型	数据备份	日志备份

数据库类型	数据备份	日志备份
PostgreSQL	数据库的数据文件备份，支持物理备份和快照备份。可用于恢复数据。实例默认会自动进行物理备份或快照备份，支持情况如下： • 本地SSD盘支持全量物理备份。 • 云盘支持快照备份，可恢复至新建实例，不支持下载。 说明 控制台仅支持全量物理备份，您可以参见 逻辑备份PostgreSQL数据导出逻辑备份到本地。	数据库的归档日志文件备份。 说明 基础版不支持日志备份。

修改自动备份设置

阿里云数据库会按照用户设置的备份策略，自动备份数据库。

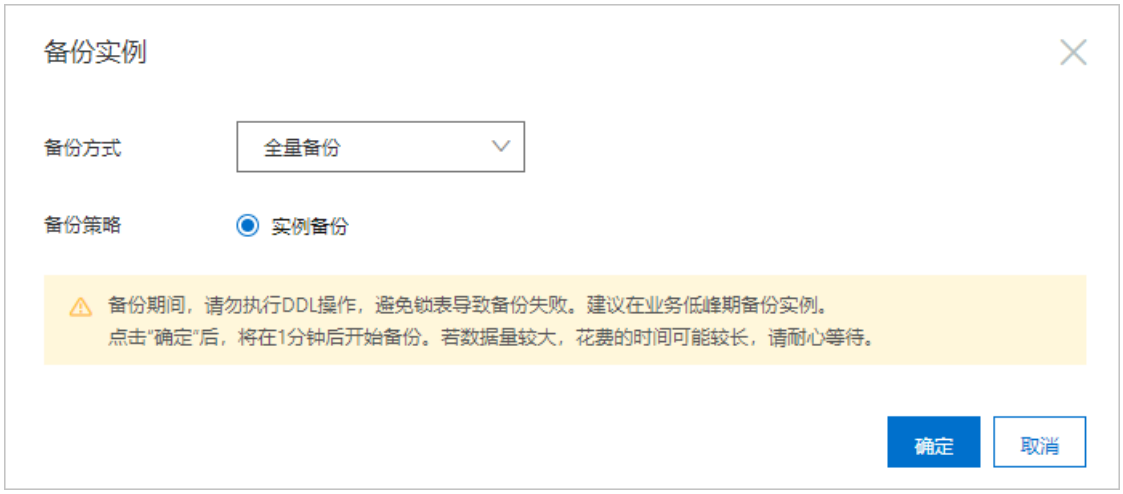
1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**备份恢复**。
3. 在**备份恢复**页面中选择**备份设置**页签，单击**编辑**。
4. 设置如下参数，然后单击 **确定**。

参数	说明
数据备份保留天数	默认为7天，可以设置为7~730天。 说明 ○ PostgreSQL 10基础版的备份数据保存7天，不可修改。 ○ 该参数仅对正常备份产生的数据生效，对**秒级备份**产生的数据无效。
备份周期	可以设置为一星期中的某几天。 说明 为了您的数据安全，一周至少需要备份两次。
秒级备份	开启后，每次备份只需1秒即可完成。 说明 ○ 仅ESSD云盘版实例支持。 ○ 开启后，系统会根据设置的**备份周期**和**备份时间**分别对数据进行一次秒级备份和正常备份。 ○ 开启后，手动备份将以秒级备份的方式进行。 ○ 秒级备份数据固定保留7天。
备份时间	可以设置为任意时段，以小时为单位。

参数	说明
日志备份	日志备份的开关。 关闭日志备份会导致所有日志备份被清除，并且无法使用按时间点恢复数据的功能。
日志备份保留	○ 日志备份文件保留的天数，默认为7天。 ○ 可以设置为7~730天，且必须小于等于数据备份天数。

手动备份

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 单击页面右上角的**备份实例**，打开备份实例对话框，单击**确定**。



3. 在右上角**任务进度**列表查看任务进度，等待任务完成。

> **说明** 备份完成后您可以在 **备份恢复**页面下载备份文件。部分实例不支持下载备份，例如存储类型为云盘的实例不支持下载，详情请参见 下载数据备份和日志备份。

常见问题

1. RDS PostgreSQL的数据备份是否可以关闭？

答：不可以关闭。可以减少备份频率，一周至少2次。

2. RDS PostgreSQL的日志备份是否可以关闭？
   答：可以关闭。备份设置内关闭日志备份开关即可。

相关API

API	描述
创建备份	创建RDS备份。
查看备份列表	查看RDS备份列表。
查询备份设置	查看RDS实例备份设置。
修改备份设置	修改RDS实例备份设置。
删除数据备份	删除RDS实例数据备份文件。
查询备份任务	查询RDS实例的备份任务列表。
查询Binlog日志	查询RDS实例的Binlog文件。

24.3. 跨地域备份

RDS PostgreSQL提供跨地域备份数据（异地备份）功能，可以将数据备份到另一个地域，用于监管和容灾恢复。

背景信息

本文介绍的是RDS PostgreSQL的跨地域备份功能，备份文件存储于另一个地域。支持通过如下方式进行跨地域备份：

- 通过RDS管理控制台跨地域备份
- 通过DBS跨地域备份

如果您已完成跨地域备份，可以跨地域恢复数据，将数据恢复到目的地域的新实例上。详情请参见跨地域恢复数据。

> **说明**
> - 默认的备份功能请参见备份PostgreSQL数据。
> - MySQL跨地域备份请参见MySQL跨地域备份数据。
> - SQL Server跨地域备份请参见SQL Server跨地域备份数据。

跨地域备份和默认备份的区别

对比项	跨地域备份	默认备份
是否默认开启	默认关闭，需要手动开启。	默认开启。
备份存储	备份存储在另一个地域。	备份存储在实例所在地域。

对比项	跨地域备份	默认备份
备份恢复	可以恢复数据到 • 原实例 • 目的地域的新实例 • 已有实例	可以恢复到 • 当前地域的新实例 • 原实例
备份保留时长	独立于实例，实例释放后仍会按照设置的保留时间进行保留。	默认在实例释放后只保留7天。

前提条件

跨地域备份方式	前提条件
通过RDS管理控制台跨地域备份	实例为如下版本： • PostgreSQL 9.4、10高可用版（本地盘） • PostgreSQL 10、11、12、13、14云盘版（不包含开启云盘加密的实例）
通过DBS跨地域备份	• 已购买数据库备份（Database Backup，简称DBS）服务，如何购买请参见创建备份计划。 • DBS地域与原RDS实例的地域不同。 • 备份方式为逻辑备份。 • 源RDS实例已申请外网地址。具体请参见申请或释放外网地址。

计费

跨地域备份方式	计费方式
通过RDS控制台跨地域备份	将产生以下费用。 • 异地存储费用：USD$0.0002/GB/小时。 • 网络流量费用：计费详情请参见网络费用。
通过DBS控制台跨地域备份	将产生备份费用，若将数据存储至DBS内置存储池中还将产生存储费用，计费详情请参见数据库备份DBS产品价格。

注意事项

跨地域备份方式	注意事项
通过RDS管理控制台跨地域备份	• 跨地域备份不支持开启云盘加密的实例。 • 跨地域备份不会影响默认备份，两者同时存在（本地备份复制到另一个地域上）。 • 默认备份完成后就会触发跨地域备份，即默认备份转储到跨地域备份存储设备上。 • 开启跨地域备份时，如果最近24小时没有有效的备份集，会触发一次备库备份。 • 由于网络原因，仅部分地域支持跨地域备份。具体请参见跨地域备份支持地域。
通过DBS跨地域备份	跨地域备份不支持开启云盘加密的实例。

跨地域备份支持地域

源地域	支持备份的目的地域
华东1（杭州）、华东2（上海）、华北1（青岛）、华北2（北京）、华北3（张家口）、华北5（呼和浩特）、华南1（深圳）、中国（香港）、华北6（乌兰察布）、西南1（成都）、华南3（广州）、华南2（河源）	中国（香港）、华东1（杭州）、华东2（上海）、华北1（青岛）、华南1（深圳）、华北3（张家口）、华北5（呼和浩特）、华北2（北京）、华北6（乌兰察布）、西南1（成都）、华南3（广州）、华南2（河源） 说明 支持备份至除源地域自身以外的地域，具体以实际环境为准。
美国（硅谷）	美国（弗吉尼亚）
美国（弗吉尼亚）	美国（硅谷）
华东1 金融云	华东2 金融云、华南1 金融云
华东2 金融云	华东1 金融云、华南1 金融云
华南1 金融云	华东1 金融云、华东2 金融云

通过RDS管理控制台跨地域备份

- **单个实例开启跨地域备份**
  i. 登录RDS管理控制台，在左侧单击**备份管理**，然后在上方选择地域。
  ii. 单击**跨地域备份 > 待开启实例**页签。
  iii. 找到目标实例，单击最右侧的**设置**。
  iv. 设置如下参数。

参数	说明
跨地域备份状态	跨地域备份的总开关，这里选择**开启**。
备份地域	当前可以用于存放备份的地域，本地备份文件将会自动复制到该地域上。
跨地域备份保留	跨地域备份文件可以保留7~1825天，即最多保留5年。 说明 即使RDS实例到期或被释放，该跨地域备份文件的保留时间不受影响，可以在控制台的跨地域备份菜单里看到未到期的备份文件。
跨地域日志备份状态：	跨地域日志备份的开关，开启后本地日志备份文件自动复制到该地域的OSS上。

  v. 单击**确定**。
- **批量开启跨地域备份**

i. 登录RDS控制台，在左侧单击**备份管理**，然后在上方选择地域。

ii. 单击**跨地域备份 > 待开启实例**页签。

iii. 勾选需要开启跨地域备份的实例，单击**备份设置**。

> **说明** 您也可以单击右侧 **设置**开启单个实例的跨地域备份。

iv. 设置如下参数。

参数	说明
跨地域备份状态	跨地域备份的总开关，这里选择**开启**。
备份地域	当前可以用于存放备份的地域，本地备份文件将会自动复制到该地域上。
跨地域备份保留	跨地域备份文件可以保留7~1825天，即最多保留5年。 **说明** 即使RDS实例到期或被释放，该跨地域备份文件的保留时间不受影响，可以在控制台的跨地域备份菜单里看到未到期的备份文件。
跨地域日志备份状态：	跨地域日志备份的开关，开启后本地日志备份文件自动复制到该地域的OSS上。

v. 单击**确定**。

- **修改跨地域备份设置**

i. 登录RDS控制台，在左侧单击**备份管理**，然后在上方选择地域。

ii. 在**跨地域备份 > 备份实例**页签中找到目标实例，单击最右侧的**设置**，就可以修改跨地域备份设置。

> **说明** 如果实例已被释放，则只能修改保留天数，其它选项无法修改。

- **关闭跨地域备份**

如果不需要使用跨地域备份，您可以关闭跨地域备份。

i. 登录RDS控制台，在左侧单击**备份管理**，然后在上方选择地域。

ii. 在**跨地域备份 > 备份实例**页签中找到目标实例，单击最右侧的**设置**。

iii. 修改**跨地域备份状态**为**关闭**，修改**跨地域备份保留**为7天。

> **说明** 关闭跨地域备份之后，不会再产生新的备份，但旧的备份不会马上删除。已有备份有至少保留7天的策略。您需要设置保留时间为7天，等7天后备份文件全部自动清除，就不会再收取跨地域备份的费用。

iv. 单击**确定**。

- **下载备份集**

在RDS控制台发起跨地域备份后，可以下载跨地域存储的备份数据。

i. 登录RDS控制台，在左侧导航栏单击**备份管理**，然后在上方选择地域。

ii. 在**跨地域备份 > 备份实例**页签中找到目标实例，单击实例ID。

iii. 在**数据备份**和**日志备份**页签的备份所在行单击**下载**，分别下载全量备份集和增量备份数据。

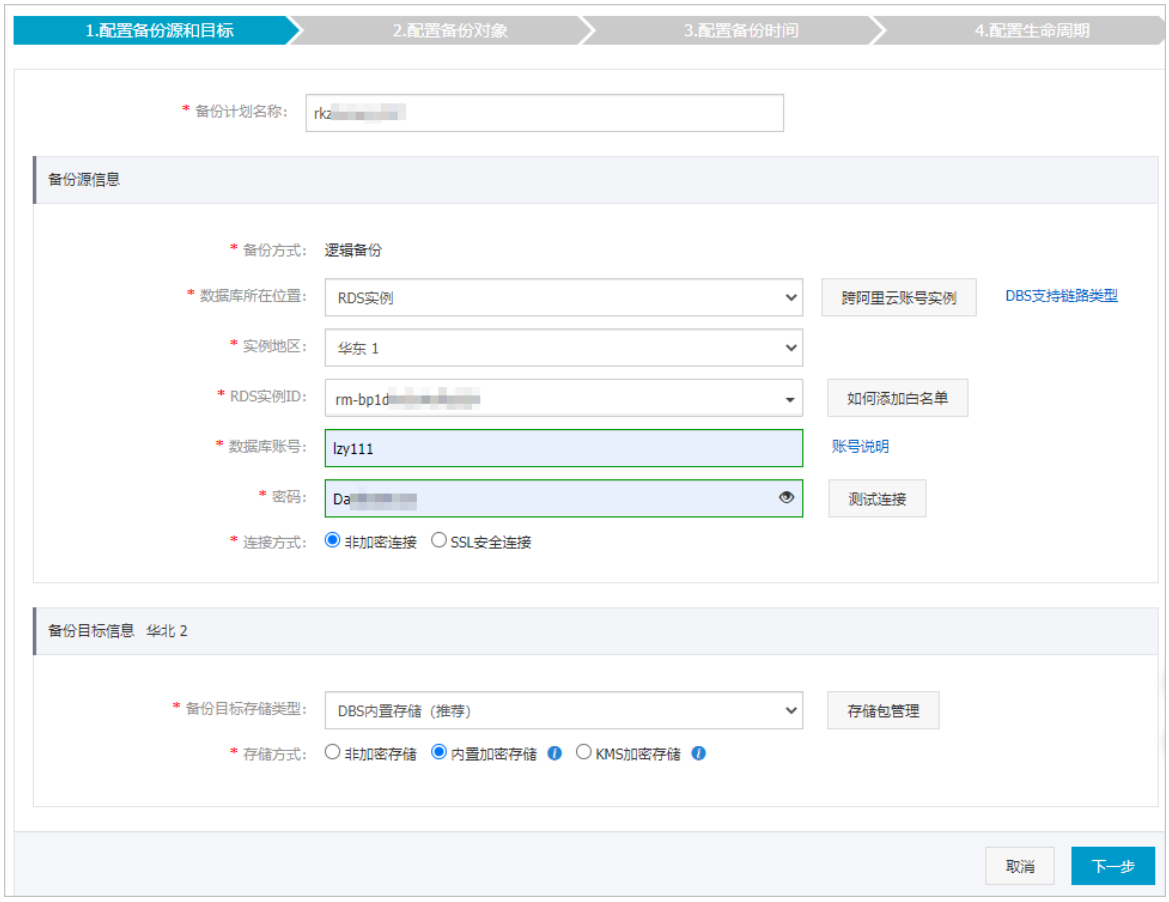
iv. 单击**我了解，要下载**。

> **说明**　下载备份文件时，内网流量免费，外网流量收费，具体请参见 网络费用。

通过DBS跨地域备份

本示例将华东1的RDS PostgreSQL实例备份至华北2。

1. 登录DBS控制台。
2. 单击左侧导航栏中的**备份计划**，然后在上方选择**备份计划**所在的地域，本示例为华北2。
3. 找到目标备份计划ID，单击右侧**操作**列下的**配置备份计划**。
4. 在**配置备份源和目标**页面，配置备份源信息与备份目标信息，并单击页面右下角的下一步。



数据库配置表

类别	配置	说明
无	**备份计划名称**	DBS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
	备份方式	默认为创建计划时购买的备份方式，本示例为逻辑备份。
	数据库所在位置	请选择**RDS实例**。
	实例地区	选择源数据库实例的地域。 因为是异地备份，所以不选择备份计划所在地域，本示例选择华东1。

类别	配置	说明
备份源信息	**RDS实例ID**	选择源RDS PostgreSQL实例ID。
	数据库账号	填入数据库账号，该账号需要具备RDS控制台中的读权限，用于备份数据库，更多信息，请参见账号权限说明。
	密码	填入该数据库账号对应的密码。 账号密码填写完毕后，您可以单击密码右侧的**测试连接**来验证填入的数据库信息是否正确。源库信息填写正确则提示测试通过；如果提示测试失败，单击测试失败后的诊断，根据提示调整填写的源库信息。
	连接方式	选择连接方式，当前支持： ○ **非加密连接** ○ **SSL安全连接**：SSL（Secure Sockets Layer）能在传输层对网络连接进行加密，提升通信数据的安全性和完整性，但会增加网络连接响应时间。 选择**SSL安全连接**前，请先在RDS实例中开启SSL，更多信息，请参见快速配置SSL加密。
备份目标信息	**备份目标存储类型**	备份目标存储类型，支持： ○ **DBS内置存储**（推荐）：无需创建存储空间，数据将自动存入DBS内置存储中，会根据存入DBS的数据量产生费用，计费方式请参见存储费用。 推荐您预购存储包，进一步减低存储费用，详情请参见使用存储包。 ○ **用户OSS**：您需要提前在OSS中创建存储空间（Bucket），更多信息，请参见创建存储空间。 **说明** 本示例为选择 **DBS内置存储**，当您选择 **用户OSS**时，您还需额外配置 **对象存储OSS Bucket名称**参数，且只支持OSS标准存储。
	存储方式	请选择存储方式，当前支持： ○ **内置加密存储**（推荐）：使用行业标准的AES256算法（即256位高级加密标准）进行加密存储。 在对象存储OSS中支持服务器端加密功能。上传文件（Object）时，OSS对收到的文件进行加密，再将得到的加密文件持久化保存；下载文件时，OSS自动将加密文件解密后返回给用户。更多信息，请参见服务器端加密。 ○ **非加密存储**：不开启加密。
	对象存储OSS Bucket名称	请选择您的对象存储OSS Bucket名称。 该参数仅在**备份目标存储类型**参数为**用户OSS**时显示。

5. 在**配置备份对象**页面，将需要备份的库或者表移动到**已选择数据库对象**框中，单击**下一步**。

> **说明** 当您选择备份整个数据库时，会额外备份索引、存储过程等数据库对象，各个数据库支持备份的对象不同，更多信息请参见 支持的数据库引擎与功能。

6. 在**配置备份时间**页面，配置备份时间等信息，并单击页面右下角的**下一步**。

配置	说明
全量备份频率	按需选择**周期备份**或**单次备份**。 **说明** 若您选择 **周期备份**，您还需配置 **全量备份周期**、**全量备份开始时间**。
全量备份周期	勾选备份数据的周期，取值范围为星期一至星期日。每周最少备份1次，最多备份7次。
全量备份开始时间	选择备份开始时间，例如**01:00**，建议设置为业务低峰期。 **说明** 若到了指定备份时间点，仍有上次的全量备份任务在进行中，则会自动跳过一次备份。
增量备份	选择是否开启增量备份。 该参数仅在**全量备份频率**参数为**周期备份**时显示。
全量备份并行线程数上限	填写全量备份并行线程数上限，您可以通过设置该参数调节备份速度，例如降低备份线程数，以减少对数据库的影响。 不同备份计划规格并行线程数上限不同，具体以控制台为准，更多信息，请参见如何调节备份速度。
备份网络限速	网络带宽限制（默认为 `0` ，表示不限速），取值不限。

7. 在**配置生命周期**页面，输入全量备份数据在DBS内置存储中的保存时间。

   若您在上一步开启了 **增量备份**功能，您还需要配置增量备份数据的保存时间，关于备份数据生命周期的更多信息，请参见 如何管理生命周期。

8. 完成上述配置后，单击页面右下角的**预检查并启动**。
9. 在预检查对话框中显示预检查通过后，单击**立即启动**。

> **说明** 待备份计划 **状态**变为 **运行中**，备份计划配置完成。

完成备份后，您可以查看备份计划或恢复备份计划，具体操作，请参见查看备份计划、跨地域恢复数据。

下载备份集

1. 登录DBS控制台。
2. 单击左侧导航栏中的**备份计划**，然后在上方选择备份计划所在的地域，本示例为华北2。
3. 单击备份计划名称，进入备份计划详情页面。
4. 在备份计划详情页面下载备份集有多种方式，具体操作请参见备份集下载功能概览。

常见问题

Q：为什么关闭跨地域备份后还会扣费？

A：关闭跨地域备份之后，不会再收取流量费，也不会再产生新的备份，但旧的备份不会马上删除。已有备份有至少保留7天的策略。您可以设置保留时间为7天，等7天后备份文件全部自动清除，就不会再收取跨地域备份的费用。

相关API

API	描述
预检查跨地域备份	预检查某RDS实例是否有跨地域备份集用于跨地域恢复。
跨地域恢复数据到新实例	跨地域恢复数据到新实例。
修改跨地域备份设置	修改RDS跨地域备份设置。

API	描述
查询跨地域备份设置	查询跨地域备份设置。
查询跨地域数据备份文件列表	查看跨地域数据备份文件列表。
查询跨地域日志备份文件列表	查看跨地域日志备份文件列表。
查询可用跨地域备份地域	查询所选地域当前可以进行跨地域备份的目的地域。
查询跨地域备份可恢复时间段	查询所选备份文件可恢复的时间段。
查询跨地域备份实例	查询所选地域的实例跨地域备份设置。

24.4. 查看备份空间免费额度

不同类型实例的备份空间免费额度不同，本文将介绍如何查看实例备份空间的免费额度以及超出免费额度的部分如何计算。

背景信息

实例的备份文件会占用备份空间，每个RDS实例的备份空间都有一定量的免费额度，超出免费额度的备份空间使用量将会产生额外的费用。

计算公式

PostgreSQL云盘实例仅支持快照备份，快照备份空间的免费额度 = 200%*实例购买的存储空间（单位为GB，只入不舍）

PostgreSQL本地盘实例仅支持物理备份，物理备份空间的免费额度 = 50%*实例购买的存储空间（单位为GB，只入不舍）

超出免费额度的部分 = 数据备份量 + 日志备份量 - 免费额度

例如：本地盘实例数据备份量30GB，日志备份量10GB，存储空间60GB，则 `每小时计费量=30+10-50%*60=10（GB）`，即每小时需要支付这额外10GB的存储费用。

> **说明**
> - 关于超出免费额度的备份空间的每小时费用，请参见备份费用。
> - 部分引擎的基础版实例免费保存最近7天的备份，请以控制台界面显示为准。

通过RDS控制台查看备份空间免费额度

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在页面最下方的使用量统计区域中，查看备份使用量后面的备注信息，即为免费额度。

> **说明** 不同类型实例显示的免费额度信息不同，请以实际界面为准。

常见问题

- 备份文件会占用购买实例时的存储空间吗？
  不会占用。备份文件空间和存储空间是隔离开的。
- 备份空间可以购买包年包月的吗？

暂不支持购买备份空间。

24.5. 下载数据备份和日志备份

为保障用户权益，RDS提供了未加密的数据备份和日志备份下载，方便用户存档，也可用于恢复到本地数据库。

注意事项

- 云盘版实例暂不支持下载快照备份。
- 只读子账号无法下载备份文件，可以在RAM控制台中添加权限到子账号中。

操作步骤

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏单击**备份恢复**。
3. 根据想要下载的备份类型，选择相应的标签页：
   - 下载数据备份：选择**数据备份**标签页。
   - 下载日志备份：选择**归档列表**标签页。
4. 如需查看8天前的备份，请修改时间范围。默认时间范围是近8天。
5. 找到目标的数据备份或归档日志，并单击其对应**操作**列中的**下载**。

   说明
   - 如果您未找到**下载**，请检查实例存储类型，云盘版实例暂不支持下载快照备份。
   - 若数据备份用于数据恢复，请选择离要恢复的时间点最近的数据备份文件。
   - 若归档日志是用于恢复到本地数据库，请注意如下事项：
     - 归档日志的Binlog所在实例编号必须与数据备份的备份所在实例编号一致。
     - 归档日志的起始时间段必须在您选择的数据备份时间点之后，在要恢复数据的时间点之前。

6.

24.6. 逻辑备份PostgreSQL数据

您可以使用PostgreSQL自带的工具pg_dump将实例的数据进行逻辑备份，并导出到本地保存。

背景信息

pg_dump是备份PostgreSQL数据库的工具，可用于备份单个数据库。更多详情请参见 pg_dump。
本文使用CentOS 7操作系统和PostgreSQL 10版本为例进行演示。

前提条件

- RDS PostgreSQL实例已设置白名单放通ECS实例或本地主机的IP。
- ECS实例或本地主机已安装和RDS PostgreSQL实例相同版本的PostgreSQL。

注意事项

建议使用高权限账号进行备份，避免出现权限不足的问题。

备份单个数据库

1. 登录ECS实例或本地主机，使用如下命令备份目标数据库。

```
pg_dump -h '<hostname>' -U <username> -p <port> -Fc <dbname> > <dumpdir>
```

参数	说明
hostname	RDS PostgreSQL实例连接地址。 **说明** ○ ECS实例通过内网连接地址访问RDS实例，需要两者网络类型相同（如果是专有网络，VPC也要相同）。 ○ 本地主机或ECS实例通过外网连接地址访问RDS实例，需要实例已申请或释放外网地址。
username	RDS PostgreSQL实例高权限账号名称。
port	RDS PostgreSQL实例端口。
-Fc	输出格式。`-Fc` 表示输出自定义格式的存档，这个格式适合通过pg_restore进行导入还原。更多选项的详细说明请参见pg_dump。
dbname	目标数据库名称。
dumpdir	导出的备份文件路径及名称。

示例

```
pg_dump -h 'pgm-bpxxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -Fc testdb > /tmp/test
db.dump
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例高权限账号对应的密码，然后按回车键确认。

备份指定表

1. 登录ECS实例或本地主机，使用如下命令备份目标数据库的指定表。

```
pg_dump -h '<hostname>' -U <username> -p <port> -t <table> -Fc <dbname> > <dumpdir>
```

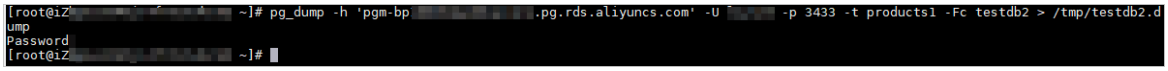
参数	说明

参数	说明
hostname	RDS PostgreSQL实例连接地址。 说明 ○ ECS实例通过内网连接地址访问RDS实例，需要两者网络类型相同（如果是专有网络，VPC也要相同）。 ○ 本地主机或ECS实例通过外网连接地址访问RDS实例，需要实例已申请或释放外网地址。
username	RDS PostgreSQL实例高权限账号名称。
port	RDS PostgreSQL实例端口。
table	需要备份的表。可以用多个 `-t <table>` 选择多个表。
-Fc	输出格式。 `-Fc` 表示输出自定义格式的存档，这个格式适合通过pg_restore进行导入还原。更多选项的详细说明请参见pg_dump。
dbname	目标数据库名称。
dumpdir	导出的备份文件路径及名称。

示例

```
pg_dump -h 'pgm-bpxxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -t products1 -Fc testd
b2 > /tmp/testdb2.dump
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例高权限账号对应的密码，然后按回车键确认。



备份数据库时排除指定表

1. 登录ECS实例或本地主机，使用如下命令备份目标数据库并排除指定表。

```
pg_dump -h '<hostname>' -U <username> -p <port> -T <table> -Fc <dbname> > <dumpdir>
```

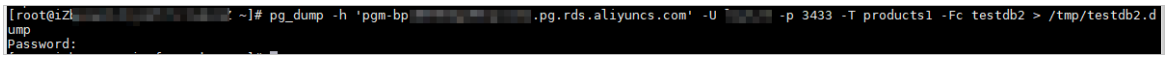
参数	说明

参数	说明
hostname	RDS PostgreSQL实例连接地址。 说明 ○ ECS实例通过内网连接地址访问RDS实例，需要两者网络类型相同（如果是专有网络，VPC也要相同）。 ○ 本地主机或ECS实例通过外网连接地址访问RDS实例，需要实例已申请或释放外网地址。
username	RDS PostgreSQL实例高权限账号名称。
port	RDS PostgreSQL实例端口。
table	需要排除的表。可以用多个 `-T <table>` 排除多个表。
-Fc	输出格式。 `-Fc` 表示输出自定义格式的存档，这个格式适合通过pg_restore进行导入还原。更多选项的详细说明请参见pg_dump。
dbname	目标数据库名称。
dumpdir	导出的备份文件路径及名称。

示例

```
pg_dump -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -T products1 -Fc testdb
2 > /tmp/testdb2.dump
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例高权限账号对应的密码，然后按回车键确认。



备份数据库模式（不备份数据）

1. 登录ECS实例或本地主机，使用如下命令备份目标数据库模式。

```
pg_dump -h '<hostname>' -U <username> -p <port> -s -Fc <dbname> > <dumpdir>
```

参数	说明

参数	说明
hostname	RDS PostgreSQL实例连接地址。 说明 ○ ECS实例通过内网连接地址访问RDS实例，需要两者网络类型相同（如果是专有网络，VPC也要相同）。 ○ 本地主机或ECS实例通过外网连接地址访问RDS实例，需要实例已申请或释放外网地址。
username	RDS PostgreSQL实例高权限账号名称。
port	RDS PostgreSQL实例端口。
-s	仅备份数据库模式，不备份数据。更多选项的详细说明请参见pg_dump。
-Fc	输出格式。`-Fc` 表示输出自定义格式的存档，这个格式适合通过pg_restore进行导入还原。更多选项的详细说明请参见pg_dump。
dbname	目标数据库名称。
dumpdir	导出的备份文件路径及名称。

示例

```
pg_dump -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -s -Fc testdb2 > /tmp/t
estdb2.dump
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例高权限账号对应的密码，然后按回车键确认。

```
[root@iZb[redacted] ~]# pg_dump -h 'pgm-bp[redacted].pg.rds.aliyuncs.com' -U [redacted] -p 3433 -s -Fc testdb2 > /tmp/testdb2.dump
Password:
[root@iZb[redacted] ~]# ll /tmp/
total 16
srwxr-xr-x 1 root root    0 Nov  5 15:28 Aegis-[redacted]
-rw-r--r-- 1 root root    4 Nov  5 15:27 CmsGoAgent.pid
drwx------ 3 root root 4096 Nov  5 15:27 systemd-private-[redacted] service-vhetNf
-rw-r--r-- 1 root root 2013 Nov  7 14:43 testdb2.dump
```

相关文档

如果数据库出现异常需要恢复数据，请参见使用pg_restore恢复逻辑备份文件数据。

24.7. 实例级别全量备份pg_basebackup

您可以使用PostgreSQL自带的工具pg_basebackup将整个实例的数据进行备份，并导出到本地保存。

前提条件

- RDS PostgreSQL实例已放通ECS实例或本地主机的IP。详情请参见设置白名单。
- ECS实例或本地主机已安装和RDS PostgreSQL实例相同版本的PostgreSQL。

背景信息

pg_basebackup可以将整个PostgreSQL实例的数据进行备份，在后续业务中可用于时间点恢复。更多详情请参见 pg_basebackup。
本文以CentOS 7操作系统和PostgreSQL 12版本为例，为您演示如何备份。

注意事项

建议使用高权限账号或具有REPLICATION权限的账号进行备份，避免出现权限不足的问题。

操作步骤

> **说明** pg_basebackup无法备份单个数据库或数据库对象。备份单个数据库或数据库对象请参见逻辑备份PostgreSQL数据。

1. 登录ECS实例或本地主机，使用如下命令备份目标数据库。

```
pg_basebackup -Ft -Pv -Xf -z -D <backupdir> -Z5 -h '<hostname>' -p <port> -U <username>
-W
```

参数说明如下。命令说明请参见pg_basebackup。

参数	说明
backupdir	导出的备份文件路径。系统会自动创建目录，但是如果该目录已经存在且不为空，则会报错。
hostname	RDS PostgreSQL实例连接地址。 如果相同账号内的ECS实例和RDS实例处于相同地域，且网络类型相同（都为VPC时则必须处于同一个VPC），建议使用内网连接地址。 其他情形建议使用外网连接地址。 **说明** 通过外网连接地址访问RDS实例，需要实例已申请外网连接地址。详情请参见 申请或释放外网地址。
port	RDS PostgreSQL实例端口。
username	RDS PostgreSQL实例账号名称。

示例

```
pg_basebackup -Ft -Pv -Xf -z -D /pg12/backup1/ -Z5 -h pgm-bpxxxxx.pg.rds.aliyuncs.com -
p 1433 -U test1 -W
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例账号对应的密码，然后按回车键确认。

```
[root@iZb[redacted] ~]# pg_basebackup -Ft -Pv -Xf -z -D /pg12/backup/ -Z5 -h pgm-bp[redacted].pg.rds.aliyuncs.com -p 1433 -U test1 -W
Password:
pg_basebackup: initiating base backup, waiting for checkpoint to complete
WARNING:  skipping special file "./.s.PGSQL.3002"
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 14/8F000028 on timeline 1
WARNING:  skipping special file "./.s.PGSQL.3002"/base.tar.gz           )
49065/49065 kB (100%), 1/1 tablespace
pg_basebackup: write-ahead log end point: 14/8F0003A0
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
[root@iZbp[redacted]jqZ ~]# ll /pg12/backup/
total 3956
-rw------- 1 root root 4047901 Apr 13 14:04 base.tar.gz
[root@iZbp[redacted] ~]#
```

相关文档

如果需要使用备份文件恢复数据，请参见Recovering Using a Continuous Archive Backup。

25.恢复

25.1. 恢复PostgreSQL数据

如果拥有RDS PostgreSQL实例的数据备份，可以按本文介绍的方法恢复数据。

前提条件

原实例需要满足如下条件：

- 运行中且没有被锁定。
- 当前没有迁移任务。
- 如果要按时间点进行恢复，需要确保日志备份已开启。

> **说明** 基础版实例不支持日志备份，因此不支持按时间点恢复。

- 若要按备份集恢复，则原实例必须至少有一个备份集。

背景信息

RDS PostgreSQL支持按备份集或时间点恢复数据。恢复数据的过程如下：

1. 恢复到一个新实例（此功能原名为克隆实例）。
2. 登录到新实例，验证实例的数据是否正确。
3. 将数据迁移到原实例。

注意事项

- 新实例的白名单设置、备份设置、参数设置和当前实例保持一致。
- 新实例的数据信息和账号信息与备份集或时间点当时的信息一致。

计费方式

与新购实例相同，购买实例的价格与所选地域、规格、存储等配置相关，请前往购买页面了解详情。

恢复数据到新实例

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏选择**备份恢复**。
3. 在页面右上角，单击**数据库恢复（原克隆实例）**。
4. 设置如下参数。

类别	说明
计费方式	○ **包年包月**：属于预付费，即在新建实例时需要支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。 ○ **按量付费**：属于后付费，即按小时扣费。适合短期需求，用完可立即释放实例，节省费用。

类别	说明
还原方式	○ **按时间点**：可以设置为日志备份保留时间内的任意时间点。如要查看或修改日志备份保留时间，请参见备份PostgreSQL数据。 ○ **按备份集** 说明 只有开启了日志备份，才会显示 **按时间点**。
系列	○ **基础版**：单节点，计算与存储分离，性价比高。 ○ **高可用版**：一个主节点和一个备节点，经典高可用架构。 说明 不同地域和数据库版本支持的系列不同，请以实际界面为准。关于各个系列的详细介绍，请参见 产品系列概述
可用区	可用区是地域中的一个独立物理区域，**主节点可用区**指主实例所在可用区，**备节点可用区**指备实例所在可用区。 您可以设置实例为**单可用区部署**或**多可用区部署**： ○ **单可用区部署**：**主节点可用区**和**备节点可用区**都处于相同可用区。 ○ **多可用区部署（推荐）**：**主节点可用区**和**备节点可用区**处于不同可用区，能提供可用区级别的容灾。您需要手动选择**主节点可用区**和**备节点可用区**。 说明 ○ 实例创建后，您可以在实例的**服务可用性**页面查看主备节点信息。 ○ 基础版实例只有一个节点，只能部署在一个可用区内。
实例规格	○ **通用规格（入门级）**：通用型的实例规格，独享被分配的内存和I/O资源，与同一服务器上的其他通用型实例共享CPU和存储资源。 ○ **独享规格（企业级）**：独享或独占型的实例规格。独享型指独享被分配的CPU、内存、存储和I/O资源。独占型是独享型的顶配，独占整台服务器的CPU、内存、存储和I/O资源。 ○ **专属规格**：完全独享虚拟主机或物理主机资源，开放主机权限，可直接在主机上按需分配多个数据库实例。更多信息，请参见添加主机。 说明 每种规格都有对应的CPU核数、内存、最大连接数和最大IOPS。详情请参见 主实例规格列表。
存储空间	存储空间包括数据空间、系统文件空间、Binlog文件空间和事务文件空间。调整存储空间时最小单位为5GB。 说明 本地SSD盘的独享套餐等规格由于资源独享的原因，存储空间大小和实例规格绑定。详情请参见 主实例规格列表。

5. 单击**下一步：网络和资源组**。
6. 设置如下参数。

类别	说明
网络类型	○ **经典网络**：传统的网络类型。 ○ **专有网络**（推荐）：也称为VPC（Virtual Private Cloud）。VPC是一种隔离的网络环境，安全性和性能均高于传统的经典网络。选择专有网络时您需要选择对应的**VPC**和**主节点交换机**，如果您在上一步的**基础资源**中配置了**多可用区部署**，则还需要选择**备选节点交换机**。 **说明** 请确保RDS实例与需要连接的ECS实例网络类型一致（如果选择专有网络，还需要保证VPC一致），否则它们无法通过内网互通。
资源组	实例所属的资源组。

7. 单击**下一步：确认订单**。
8. 确认**参数配置**，选择**购买量**和**购买时长**（仅包年包月实例），勾选服务协议，单击**去支付**完成支付。

登录到新实例并验证数据

关于登录实例的操作，请参见连接实例。

迁移数据到原实例

确认新实例的数据之后，您可以将需要的数据从新实例迁移回原实例。详情请参见RDS实例间的数据迁移。

> **说明** 数据迁移是指将一个实例（称为源实例）的数据复制到另一个实例（称为目标实例），迁移操作不会对源实例造成影响。

25.2. 跨地域恢复数据

如果您已完成跨地域备份数据，可以使用备份文件将数据恢复到原实例和跨地域备份所在地域的新实例上。

前提条件

已开启 跨地域备份。

> **说明**
> - MySQL跨地域恢复请参见MySQL跨地域恢复数据。
> - SQL Server跨地域恢复请参见SQL Server跨地域恢复数据。

操作步骤

1. 登录RDS管理控制台，在左侧单击**备份管理**，然后在上方选择地域。
2. 切换到**跨地域备份 > 备份实例**页签，单击实例ID进入跨地域备份恢复页面，单击右侧**恢复**。
3. 在**数据库恢复**页面选择**包年包月**或**按量付费**页签，然后设置如下参数。

参数	说明

参数	说明
还原方式	○ **按备份集**：恢复备份集内的数据到新实例。 ○ **按时间点**：可以设置为日志备份保留时间内的任意时间点，恢复该时间点的数据到新实例。
备份集	**还原方式**为**按备份集**时，选择用于恢复数据的备份集。
还原时间	**还原方式**为**按时间点**时，选择恢复数据的时间节点。 **说明** 本地日志备份和跨地域日志备份都可用于按时间点恢复。
地域	新实例所在地域。
可用区	可用区是地域中的一个独立物理区域，不同可用区之间没有实质性区别。您可以选择将RDS实例与ECS实例创建在同一可用区或不同的可用区。
规格	每种规格都有对应的CPU核数、内存、最大连接数和最大IOPS。具体请参见主实例规格列表。
存储空间	存储空间包括数据空间、系统文件空间、Binlog文件空间和事务文件空间。
网络类型	○ **经典网络**：传统的网络类型。 ○ **专有网络**（推荐）：也称为VPC（Virtual Private Cloud）。VPC是一种隔离的网络环境，安全性和性能均高于传统的经典网络。您需要选择VPC和相应的虚拟交换机。

说明 数据库类型、版本、系列等参数和源实例保持一致，无法修改。

4. 设置**购买时长**（仅针对包年包月实例）和**数量**，然后单击右侧**立即购买**。
5. 在**订单确认**页面勾选服务条款，根据提示完成支付。

相关文档

创建实例后，您需要设置白名单和创建账号，如果是通过外网连接，还需要申请外网地址。然后就可以连接实例。

25.3. 使用pg_restore恢复逻辑备份文件数据

您可以使用pg_restore命令将逻辑备份文件（.dump文件）中的数据恢复到RDS PostgreSQL实例或本地PostgreSQL数据库中。

背景信息

对于少量的数据，您可以使用逻辑备份进行快速恢复，例如恢复某个表的数据。对于大量的数据，建议您使用全量物理备份恢复数据到新实例上，然后通过数据传输服务DTS 迁移回现有实例。

前提条件

已逻辑备份PostgreSQL数据。

注意事项

- 恢复数据时连接的数据库建议不要用默认数据库postgres。
- 恢复指定表数据时不会尝试还原指定表可能依赖的数据库对象，因此无法保证将指定表还原到干净的数据库时一定会成功。

恢复数据库

1. 登录已备份数据的ECS实例或本地主机，使用如下命令恢复数据。

```
pg_restore -h '<hostname>' -U <username> -p <port> -d <dbname> <dumpdir>
```

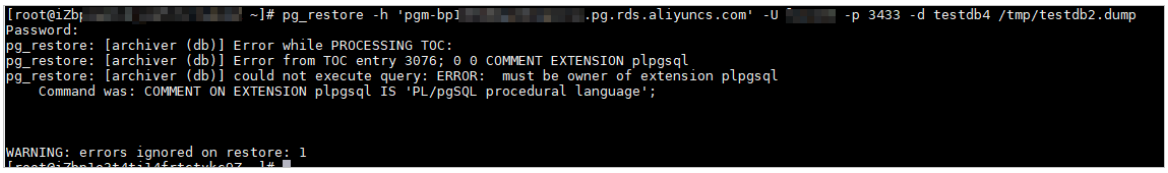
参数	说明
hostname	RDS PostgreSQL实例连接地址。 **说明** ○ ECS实例通过内网连接地址访问RDS实例，需要两者网络类型相同（如果是专有网络，VPC也要相同）。 ○ 本地主机或ECS实例通过外网连接地址访问RDS实例，需要实例已申请或释放外网地址。
username	RDS PostgreSQL实例高权限账号名称。
port	RDS PostgreSQL实例端口。
dbname	待恢复数据的数据库名称。
dumpdir	备份文件路径及名称。

示例

```
pg_restore -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -d testdb2 /tmp/test
db.dump
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例高权限账号对应的密码，然后按回车键确认。

> **说明** 请忽略内嵌插件plpgsql的报警。



恢复指定表数据

1. 登录已备份数据的ECS实例或本地主机，使用如下命令恢复指定表的数据。

```
pg_restore -h '<hostname>' -U <username> -p <port> -d <dbname> -t <table> -c <dumpdir>
```

参数	说明
hostname	RDS PostgreSQL实例连接地址。 说明 ○ ECS实例通过内网连接地址访问RDS实例，需要两者网络类型相同（如果是专有网络，VPC也要相同）。 ○ 本地主机或ECS实例通过外网连接地址访问RDS实例，需要实例已申请或释放外网地址。
username	RDS PostgreSQL实例高权限账号名称。
port	RDS PostgreSQL实例端口。
dbname	待恢复数据的数据库名称。
table	待恢复的表名。
-c	`-c` 表示在恢复时，会先删除数据库对象再恢复。更多选项的详细说明请参见pg_restore。
dumpdir	备份文件路径及名称。

示例

```
pg_restore -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -d testdb2 -t produc
ts -c /tmp/testdb.dump
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例高权限账号对应的密码，然后按回车键确认。

恢复数据库模式（不恢复数据）

1. 登录已备份数据的ECS实例或本地主机，使用如下命令恢复数据库模式。

```
pg_restore -h '<hostname>' -U <username> -p <port> -d <dbname> -s <dumpdir>
```

参数	说明
hostname	RDS PostgreSQL实例连接地址。 说明 ○ ECS实例通过内网连接地址访问RDS实例，需要两者网络类型相同（如果是专有网络，VPC也要相同）。 ○ 本地主机或ECS实例通过外网连接地址访问RDS实例，需要实例已申请或释放外网地址。

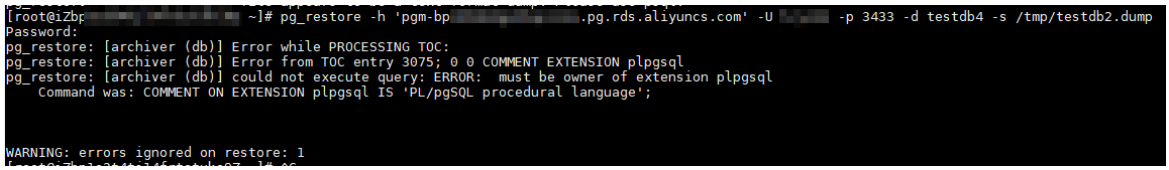
参数	说明
username	RDS PostgreSQL实例高权限账号名称。
port	RDS PostgreSQL实例端口。
dbname	待恢复模式的数据库名称。
-s	`-s` 表示仅恢复模式，不恢复数据。更多选项的详细说明请参见pg_restore。
dumpdir	备份文件路径及名称。

示例

```
pg_restore -h 'pgm-bpxxxxx.pg.rds.aliyuncs.com' -U test123 -p 3433 -d testdb4 -s /tmp/t
estdb2.dump
```

2. 命令行提示 `Password:` 时，输入PostgreSQL实例高权限账号对应的密码，然后按回车键确认。

> **说明** 请忽略内嵌插件plpgsql的报警。



26.性能优化与诊断

26.1. RDS PostgreSQL CPU使用率高问题

当实例CPU使用率持续较高时，很容易导致数据库访问响应慢。本文介绍如何定位CPU使用率高的原因以及如何解决这些问题。

查看CPU使用率

对于RDS PostgreSQL实例来说，CPU使用率持续高于80%，通常表明系统处于高负载的情况，并且很可能存在较严重的性能问题。

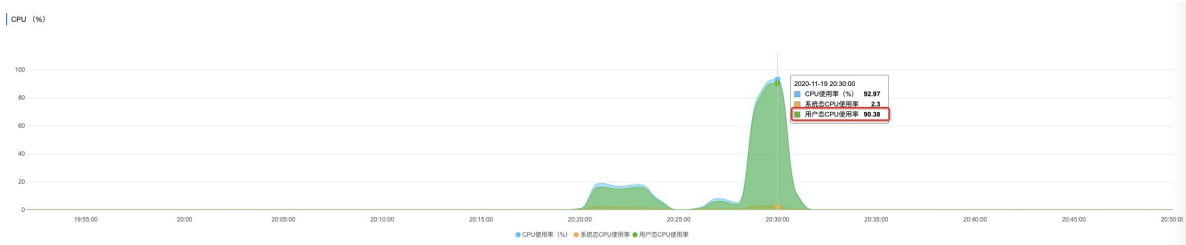
说明 您可以在控制台 **监控与报警** 中的 **资源监控** 里查看CPU使用率信息。

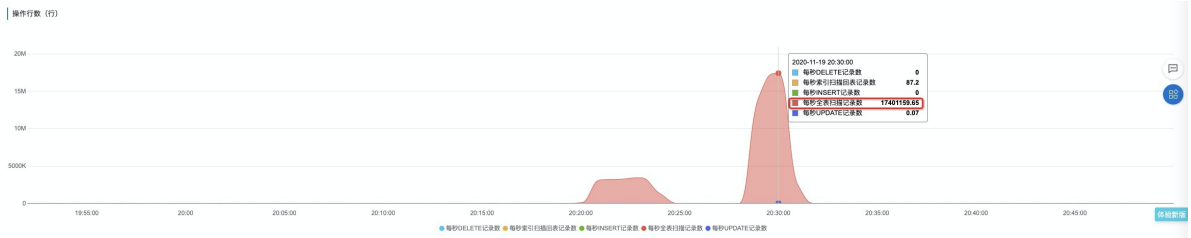
基本概念

- CPU使用率：CPU执行的工作时间的比例，包含了所有符合条件的活动的时钟周期，例如阻塞等待I/O而导致较高的使用率。CPU使用率又被分为内核时间和用户时间。
- 用户时间：执行用户态（User Mode）程序的时间被称为用户时间。
- 内核时间：执行系统态（System Mode）代码的时间为内核时间，包含系统调用，内核线程和中断的时间。
- 上下文切换：内核程序切换CPU让其在不同的地址空间上操作。
- 中断：由物理设备发送给内核的信号，通常是请求I/O服务。

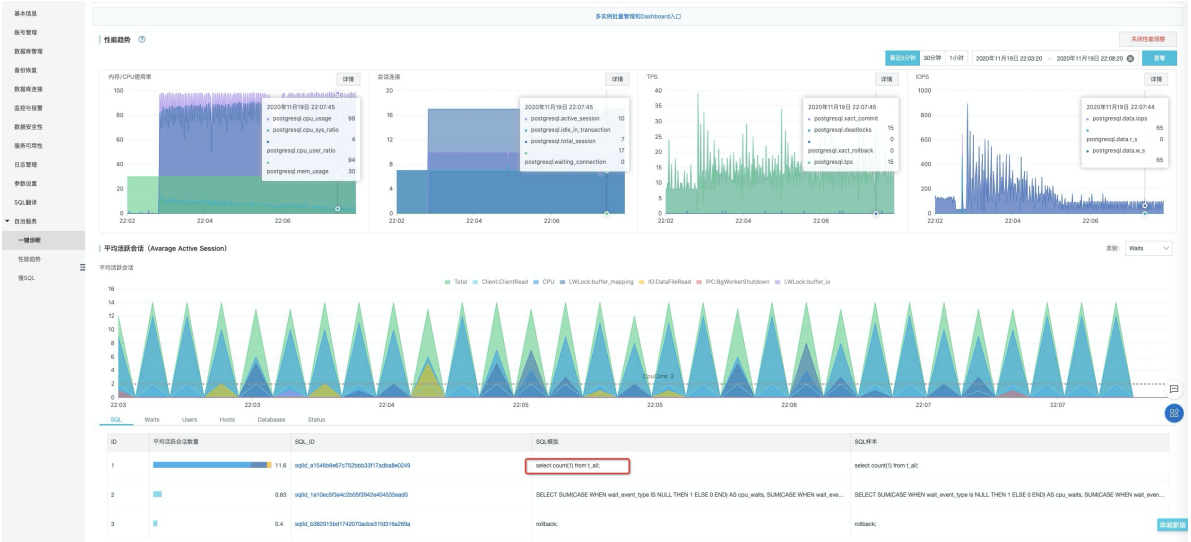
扫描行高导致的CPU使用率高

- 现象：
  在控制台的**资源监控**中可以看到，CPU使用率和操作行数很高。





通过查看**操作行数**的详细信息可以发现，数据库中存在大量的全表扫描行，这是导致CPU使用率高的主要原因。此时应该定位到导致该问题的SQL，对其进行优化。

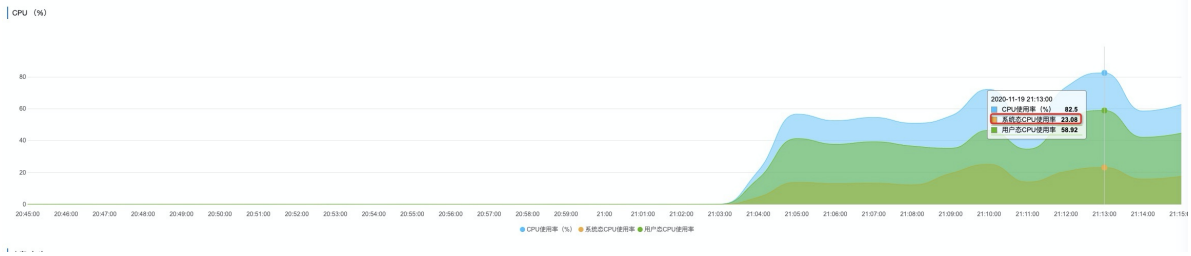


- 解决方案：通过一键诊断功能定位问题SQL，并对其进行优化。更多信息，请参见RDS PostgreSQL慢SQL问题。

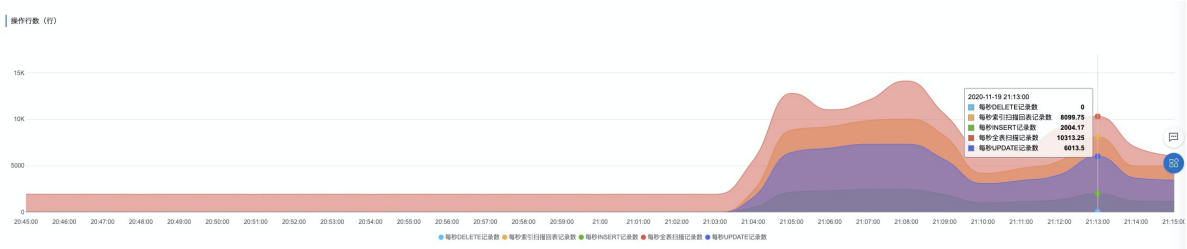
> **说明** 您可以在RDS控制台左侧导航栏中选择 **自治服务 > 一键诊断**，打开一键诊断功能。

活跃会话高导致的CPU使用率高

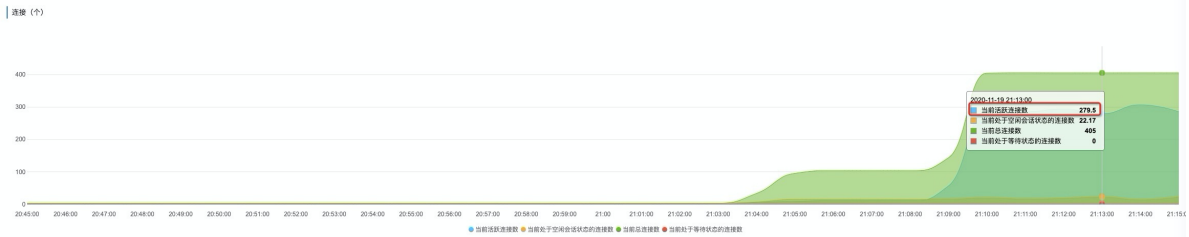
- 现象：
  在控制台的**资源监控**中发现CPU使用率较高，此时系统态占用达到了23%，说明可能存在大量的系统调用或者中断。



通过**操作行数**可以看到，全表扫描数其实并不高。



再查看**连接**，发现此时活跃会话数已达到279个，由于本示例中使用的实例规格为2核4GB，当279个会话同时运行时，CPU会频繁地进行上下文切换，导致系统态占用增加，降低了CPU实际去执行SQL的时间。



> **说明** 正常情况下，合理的活跃会话数量应当是当前CPU核数的2~3倍，例如实例规格为2核4GB时，活跃会话数应当维持在4~6个，此时的CPU使用效率最高。

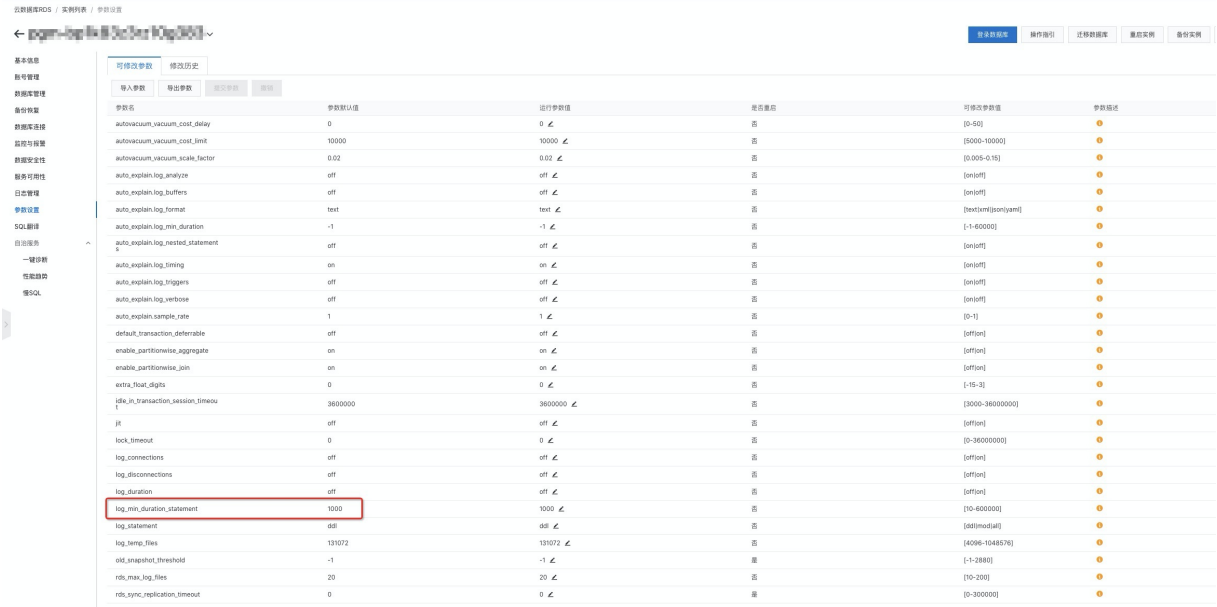
- 解决方案：考虑进行实例规格的升级，或者降低活跃会话的数量，使其保持在一个合理的范围内。

26.2. RDS PostgreSQL慢SQL问题

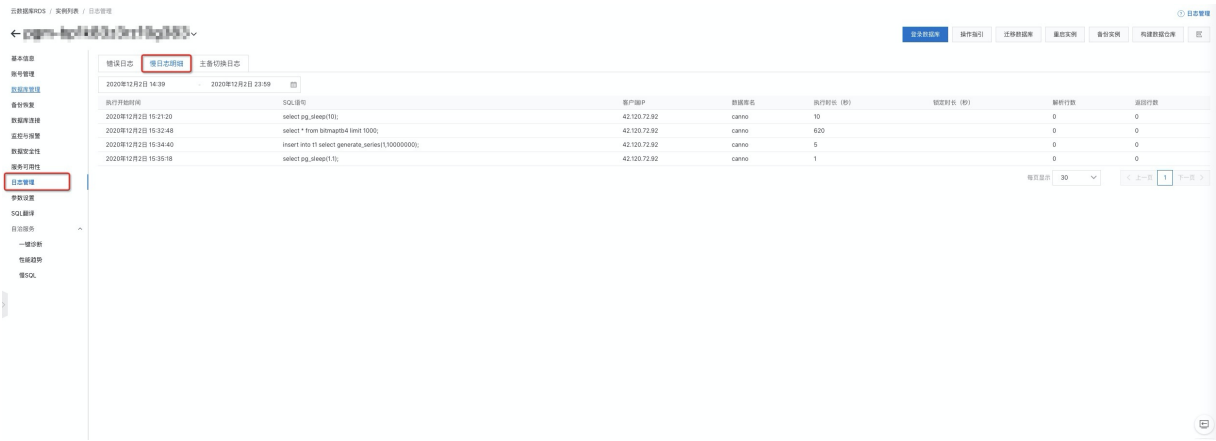
在相同业务场景下，架构设计和库表索引设计会影响查询性能，良好的设计可以提高查询性能，反之会出现很多慢SQL（执行时间很长的SQL语句）。本文介绍导致慢SQL的原因和解决方案。

查看慢SQL

控制台的**参数设置**中，可以找到名为log_min_duration_statement的参数，该参数的作用是设置慢日志判定的阈值，单位为ms。默认值为1000，即响应时间超过1秒的SQL会被记录到慢SQL日志中。



通过查看控制台的**日志管理**中的**慢日志明细**页签，可以查看当前系统中存在的慢SQL。



您只需在**参数设置**中开启auto_explain.log_analyze、auto_explain.log_buffers、auto_explain.log_min_duration三个参数，即可在**慢日志明细**中查看慢SQL的执行计划。

EXPLAIN命令

定位到慢SQL之后，可以通过EXPLAIN命令查询该SQL的执行计划，帮助您深入了解目标SQL的启动代价。

```
EXPLAIN [ ( option [, ...] ) ] statement
```

上面EXPLAIN命令中的option可选参数的说明如下：

- ANALYZE [boolean]：执行SQL语句并返回实际运行时间。默认值：`FALSE`。
- VERBOSE [boolean]：显示关于计划的额外信息。默认值：`FALSE`。
- COST [boolean]：显示执行计划中每一个计划节点的启动时间、总执行代价、预估行数以及每行的宽度。默认值：`TRUE`。
- BUFFERS [boolean]：缓冲区使用的信息。例如：共享块命中、读取、标记为脏和写入的次数、本地块命中、读取、标记为脏和写入的次数、临时块读取和写入的次数。默认值：`FALSE`。
- TIMING [boolean]：显示实际启动时间以及在每个计划节点中花费的时间。仅当ANALYZE为`TRUE`时可用。默认值：`TRUE`。
- SUMMARY [boolean]：在查询计划后面包含摘要信息（例如共计花费多长时间）。当ANALYZE为`TRUE`时自动开启。默认值：`FALSE`。
- FORMAT { TEXT | XML | JSON | YAML }：指定输出格式。默认值：`TEXT`。

例如，执行`EXPLAIN SELECT * from test;`返回的结果如下：

```
                      QUERY PLAN
----------------------------------------------------
 Seq Scan on test  (cost=0.00..1.01 rows=1 width=4)
(1 row)
```

上面的返回结果中，`cost=0.00..1.01 rows=1 width=4` 为估算出的SQL预计执行代价，其中 `0.00` 为启动SQL语句到返回第一行数据所需的启动代价，`1.01` 为返回所有行所需的总代价。

EXPLAIN (ANALYZE) DML-SQL 注意事项

EXPLAIN (ANALYZE) 语句的工作方式类似于**EXPLAIN**，主要区别在于**EXPLAIN (ANALYZE)** 语句实际会执行SQL。如果SQL涉及数据变更，即DML SQL（**UPDATE**、**INSERT**或**DELETE**），务必在事务中执行**EXPLAIN (ANALYZE)**，查看完成后再进行回滚。

命令示例：

```
BEGIN;
EXPLAIN (ANALYZE) <DML(UPDATE/INSERT/DELETE) SQL>;
ROLLBACK;
```

如何阅读执行计划

EXPLAIN命令的输出可以看作是一个树状结构，即查询计划树。树的每一个节点包含节点类型、作用对象及其他属性信息。其中节点类型分如下几大类：

- 控制节点（Control Node）
- 扫描节点（ScanNode）
- 物化节点（Materialization Node）
- 连接节点（Join Node）

其中扫描节点又包含多种扫描方式，本文列举常用的几个进行详细介绍：

- Seq Scan：
  顺序扫描全表，一般用在查询没有索引的表，例如 `explain(ANALYZE,VERBOSE,BUFFERS) select * from class where st_no=2;` 命令的返回结果如下：

```
QUERY PLAN
-----------------------------------------------------------------------------------------------
---------
Seq Scan on public.class (cost=0.00..26.00 rows=1 width=35) (actual time=0.136..0.141 row
s=1 loops=1)
//Seq Scan on public.class表示这个节点的类型和作用对象，即在class表上进行全表扫描。
//(cost=0.00..26.00 rows=1 width=35)表示该节点的代价估计，0.00是节点启动成本，26.00为该节点的代
价，rows是该节点输出行的估计值，width是该节点输出行的平均宽度。
//(actual time=0.136..0.141 rows=1 loops=1) 表示该节点的真实执行信息，0.136表示该节点启动时间，
单位为ms。0.141表示该节点耗时，单位为ms。rows代表实际输出行。loops代表节点循环次数。
    Output: st_no, name //代表SQL输出结果集的各个列，仅在EXPLAIN命令中的VERBOSE选项为on时才会显
示。
    Filter: (class.st_no = 2) //表示Seq Scan节点中Filter操作，即全表扫描时对每行记录进行过滤操
作，过滤条件为class.st_no = 2。
    Rows Removed by Filter: 1199 //表示过滤了多少行记录，属于Seq Scan节点的VERBOSE信息，仅在EX
PLAIN命令中的VERBOSE选项为on时才会显示。
    Buffers: shared hit=11 //表示在共享缓存中命中了11个数据块，属于Seq Scan节点的BUFFERS信息，
仅在EXPLAIN命令中的BUFFERS选项为on时才会显示。
Planning time: 0.066 ms //表示生成查询计划的用时。
Execution time: 0.160 ms //表示实际的SQL执行用时，不包括生成查询计划的用时。
```

- Index Scan：
  索引扫描，主要用在WHERE条件中存在索引列的情况，例如，假设 `st_no` 列中存在索引，则 `explain(ANALYZE,VERBOSE,BUFFERS) select * from class where st_no=2;` 命令的返回结果如下：

```
QUERY PLAN
-----------------------------------------------------------------------------------------------
---------
Index Scan using no_index on public.class (cost=0.28..8.29 rows=1 width=35) (actual time=
0.022..0.023 rows=1 loops=1)
//表示使用public.class表的no_index索引对表进行索引扫描。
     Output: st_no, name
     Index Cond: (class.st_no = 2) //表示索引扫描的条件为class.st_no = 2。
     Buffers: shared hit=3
Planning time: 0.119 ms
Execution time: 0.060 ms
(6 rows)
```

  从上述结果可以看出，使用索引之后，在相同条件下对同一张表的扫描速度变快了。需要扫描的数据块少了之后，需要的代价变小了，速度也更快了。

- Index Only Scan：
  覆盖索引扫描，仅返回指定的索引列，例如 `explain(ANALYZE,VERBOSE,BUFFERS) select st_no from class where st_no=2;` 命令的返回结果如下：

```
QUERY PLAN
-----------------------------------------------------------------------------------------
---------
Index Only Scan using no_index on public.class (cost=0.28..4.29 rows=1 width=4) (actual t
ime=0.015..0.016 rows=1 loops=1)
//表示使用public.class表的no_index索引对表进行覆盖索引扫描。
    Output: st_no Index Cond: (class.st_no = 2)
    Heap Fetches: 0 //表示需要扫描的数据块的个数。
    Buffers: shared hit=3
Planning time: 0.058 ms
Execution time: 0.036 ms
(7 rows)
```

- Bitmap Index Scan：
  利用Bitmap结构扫描。Bitmap Index Scan与Index Scan都是基于索引的扫描，但Bitmap Index Scan节点返回的是一个位图而不是一个元组，位图中每位代表了一个扫描到的数据块。
- Bitmap Heap Scan：
  Bitmap Heap Scan一般作为Bitmap Index Scan的父节点，将Bitmap Index Scan返回的位图转换为对应的元组。相比Index Scan，Bitmap Index Scan将随机读转换成了按照数据块的物理顺序读取，这在数据量比较大的时候会极大地提升扫描性能。例如 `explain(ANALYZE,VERBOSE,BUFFERS) select * from class where st_no=2;` 命令的返回结果如下：

```
QUERY PLAN
-----------------------------------------------------------------------------------------
---------
Bitmap Heap Scan on public.class (cost=4.29..8.30 rows=1 width=35) (actual time=0.025..0.
025 rows=1 loops=1) Output: st_no, name
//表示对public.class表进行Bitmap Heap扫描。
Recheck Cond: (class.st_no = 2)
//表示Bitmap Heap Scan的Recheck操作的条件是class.st_no = 2。
Heap Blocks: exact=1
//表示准确扫描到的数据块个数是1。
Buffers: shared hit=3
-> Bitmap Index Scan on no_index (cost=0.00..4.29 rows=1 width=0) (actual time=0.019..0.0
19 rows=1 loops=1)
//表示使用no_index索引进行位图索引扫描。
Index Cond: (class.st_no = 2)
//表示位图索引的条件为class.st_no = 2。
Buffers: shared hit=2
Planning time: 0.088 ms
Execution time: 0.063 ms
(10 rows)
```

> **说明**
> - 在大多数情况下，Index Scan要比Seq Scan快。
> - 如果获取的结果集在所有数据中占比很大时，Index Scan因为要先扫描索引再读表数据，所以反而会比Seq Scan慢。
> - 如果获取的结果集的占比比较小，但是元组数很多时，Bitmap Index Scan的性能要比Index Scan好。
> - 如果获取的结果集能够被索引覆盖，则Index Only Scan因为仅需扫描索引，正常情况下性能最好。如果存在可见性映射表（Vitual Map）未生成等特殊情况，性能则会下降。

SQL优化示例

本节列举两个示例，分别使用EXPLAIN命令对SQL优化前后的表进行分析，对比一下前后的差距。

无索引表的优化

- SQL优化前：对无索引的表T1执行 `explain analyze select * from t1 where id =1;` 命令的返回结果如下。

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------
--------------------------
 Gather  (cost=1000.00..97331.31 rows=1 width=4) (actual time=0.217..302.316 rows=1 loops
=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Parallel Seq Scan on t1  (cost=0.00..96331.21 rows=1 width=4) (actual time=191.208
..289.240 rows=0 loops=3)
         Filter: (id = 1)
         Rows Removed by Filter: 3333333
 Planning Time: 0.030 ms
 Execution Time: 302.381 ms
(8 rows)
```

  通过计划可以看出， `SELECT` 针对T1表进行了全表扫描，过滤条件为 `id=1` ，总执行时间为 `302.381ms` ，效率不是很理想。

- SQL优化后：通过 `create index on t1(id)` 给T1表中的 `id` 列创建索引，重新执行 `explain analyze select * from t1 where id =1;` 命令，返回结果如下。

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------
--------------------------
 Index Only Scan using t1_id_idx on t1  (cost=0.43..2.45 rows=1 width=4) (actual time=0.0
35..0.036 rows=1 loops=1)
   Index Cond: (id = 1)
   Heap Fetches: 1
 Planning Time: 0.134 ms
 Execution Time: 0.052 ms
(5 rows)
```

  总执行时间缩短至 `0.052ms` ，相比优化前的 `302.381ms` 提升了数千倍。

非最佳索引表的优化

创建一张名为T2的表，插入100万条随机数据，并对 `id` 列创建索引。

```
create table t2(id int,name int);
insert into t2 select random()*(id/100),random()*(id/100) from generate_series(1,1000000) t
(id);
create index on t2(id)
```

- SQL优化前：执行 `explain analyze select * from t2 where id=10 and name=13;` 命令的返回结果如下。

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------
-----------------------
 Index Scan using t2_id_idx on t2  (cost=0.42..12.12 rows=1 width=8) (actual time=0.098..
2.631 rows=88 loops=1)
   Index Cond: (id = 10)
   Filter: (name = 13)
   Rows Removed by Filter: 4461
 Planning Time: 0.081 ms
 Execution Time: 2.655 ms
(6 rows)
```

通过计划可以看出已经做了索引扫描，但是还存在 `name` 列的条件过滤，还有优化的空间。

- SQL优化后：通过 `create index on t2(id,name);` 给T2表中的 `id` 列和 `name` 列各增加一个索引。重新执行 `explain analyze select * from t2 where id=10 and name=13;` 命令，返回结果如下。

```
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------
----------------------------------
 Index Only Scan using t2_id_name_idx on t2  (cost=0.42..15.48 rows=18 width=8) (actual t
ime=0.028..0.134 rows=88 loops=1)
   Index Cond: ((id = 10) AND (name = 13))
   Heap Fetches: 88
 Planning Time: 0.198 ms
 Execution Time: 0.157 ms
(5 rows)
```

总执行时间缩短至 `0.157ms` ，优化成功。

26.3. RDS PostgreSQL I/O高问题

RDS PostgreSQL实例的I/O性能受硬件层存储介质、软件层数据库内核架构和具体SQL语句（扫描或修改数据量）的影响。

基本概念

名称	概念
IO	Input和Output的缩写，即对磁盘进行的读写动作。
IO延时	一个读写操作的执行时间。
逻辑IO	由应用程序发给文件系统的读写操作。

名称	概念
物理IO	由文件系统发给磁盘的读写操作。
顺序IO	按照顺序向磁盘进行读写操作。
随机IO	随机向磁盘进行读写操作。
同步写	等待数据完全写入磁盘后再返回，会阻塞程序直到写入完成。
异步写	不等待数据写入结束直接返回，不会阻塞程序。

SQL扫描行高导致实例I/O高

- 现象：
  数据盘IOPS中发现读占用较高，正常情况下客户端使用SQL访问的数据都是从共享缓冲区返回的，共享缓冲区中未命中的数据会从磁盘中读取，读占用高说明数据大量从磁盘读取到共享缓冲区，因此推断可能存在SQL扫描行较高的情况。

  > **说明** 您可以在控制台的**监控与报警** > **标准监控**页面中查看各项监控指标。

  操作行数中发现全表扫描行明显增加，说明客户端读取了大量的数据。

- 解决方案：
  通过**性能洞察**找到对应的问题SQL，针对该问题SQL进行索引优化或者增加实例规格以承担更高的负载。

  > **说明** 您可以在控制台的**自治服务** > **一键诊断**页面，单击**性能洞察**页签使用**性能洞察**功能。

导入数据导致实例I/O高

- 现象：

数据盘IOPS中发现写占用较高，怀疑存在大量的数据导入动作。

> **说明** 您可以在控制台的**监控与报警** > **标准监控**页面中查看各项监控指标。

- 解决方案：通过**性能洞察**发现是因为导入数据导致的IOPS升高，应尽量避免业务高峰期的数据导入操作。

Vacuum操作导致实例I/O高

Vacuum指的是PostgreSQL实例对垃圾数据进行的清理动作，在执行 `UPDATE` 和 `DELETE` 的过程中，数据表会产生垃圾数据，需要及时通过Vacuum进行清理回收空间。

- 现象：
  数据盘IOPS中发现读写操作均占用了较多的资源，并且**操作行数**中未见明显增加的趋势，此时怀疑是Vacuum或者是CheckPoint引起的IOPS升高。

> **说明** 您可以在控制台的**监控与报警** > **标准监控**页面中查看各项监控指标。

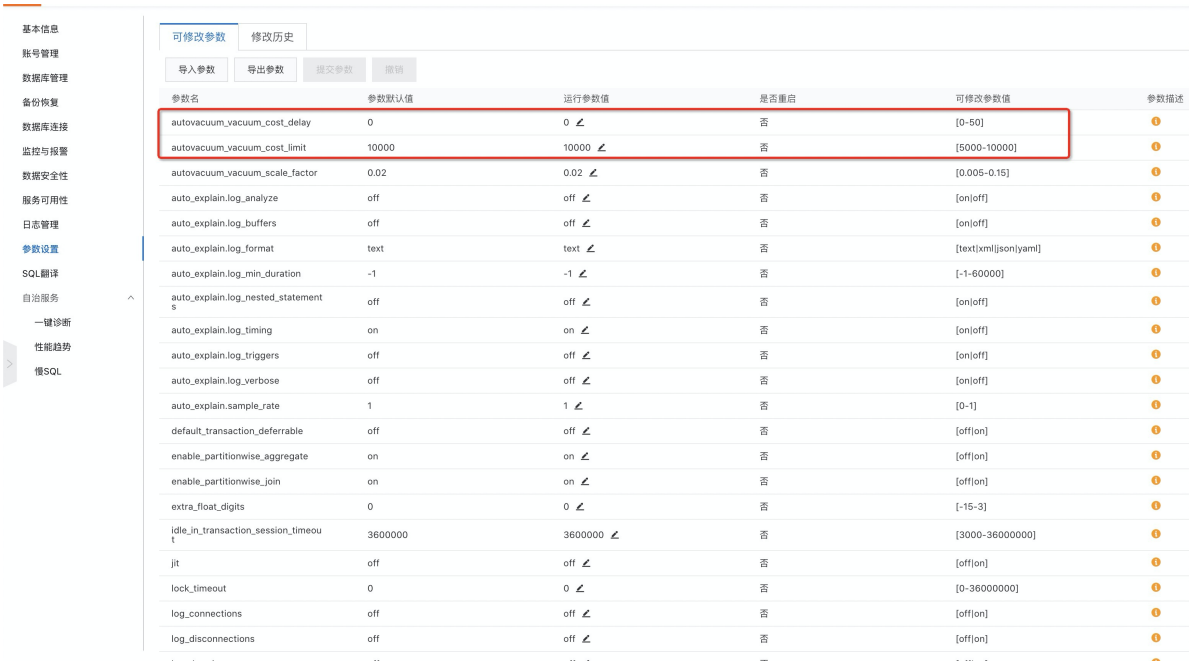
通过 `select * from pg_stat_progress_vacuum ;` 语句查询pg_stat_progress_vacuum可以看到当前正在运行的Vacuum动作以及进度，可以确定是由Vacuum引起的IOPS升高。

```
-[ RECORD 1 ]------+--------------
pid                | 109229
datid              | 13593
datname            | postgres
relid              | 40435
phase              | scanning heap
heap_blks_total    | 943453
heap_blks_scanned  | 937101
heap_blks_vacuumed | 0
index_vacuum_count | 0
max_dead_tuples    | 291
num_dead_tuples    | 0
```

通过 `select * from pg_stat_user_tables where relname ='t_all';` 语句还可以查询到Vacuum的历史执行情况。

```
-[ RECORD 1 ]-------+------------------------------
relid               | 40435
schemaname          | public
relname             | t_all
seq_scan            | 4547
seq_tup_read        | 35959634448
idx_scan            |
idx_tup_fetch       |
n_tup_ins           | 91514896
n_tup_upd           | 0
n_tup_del           | 0
n_tup_hot_upd       | 0
n_live_tup          | 91514896
n_dead_tup          | 0
n_mod_since_analyze | 0
last_vacuum         | 2020-11-20 14:25:19.077927+08
last_autovacuum     |
last_analyze        |
last_autoanalyze    | 2020-11-20 11:59:34.641906+08
vacuum_count        | 1
autovacuum_count    | 0
analyze_count       | 0
autoanalyze_count   | 5
```

- 解决方案：
  在控制台的**参数设置**中根据实际业务场景调整Vacuum相关参数降低Vacuum对I/O产生的影响。



- autovacuum_vacuum_cost_delay：vacuum进程的代价达到autovacuum_vacuum_cost_limit设置的值时，vacuum进程暂停的时间。单位为ms。
- autovacuum_vacuum_cost_limit：评估vacuum进程代价的阈值。

> **注意** Vacuum回收速度过慢也会导致更加严重的表膨胀。

CheckPoint操作导致实例I/O高

- 现象：
  通过 `select * from pg_stat_bgwriter ;` 语句查询pg_stat_bgwriter，分析checkpoints_req指标。该指标指的是手动CheckPoint以及wal日志写入量达到max_wal_size的次数。可以发现此时checkpoints_req的值为160。

```
-[ RECORD 1 ]---------+------------------------------
checkpoints_timed     | 8271
checkpoints_req       | 160
checkpoint_write_time | 9109283
checkpoint_sync_time  | 22878
buffers_checkpoint    | 1228674
buffers_clean         | 829714
maxwritten_clean      | 293
buffers_backend       | 3731358
buffers_backend_fsync | 0
buffers_alloc         | 2290169
stats_reset           | 2020-06-28 15:32:00.021466+08
```

- 解决方案：
  关注是否有大批量导入的情况，建议控制批量导入的大小，建议调整为小批量，分多次导入以减小I/O的压力。

说明　CheckPoint指的是将数据库共享缓冲区的脏页写入到磁盘的操作，如下四种情况会触发CheckPoint操作：

- 定时触发：通过checkpoint_timeout参数定时将数据库共享缓冲区的脏页进行写入到磁盘，此行为为异步写，数据分多次小批量写入磁盘，对I/O影响小。
- wal日志写入量达到max_wal_size时触发：wal日志是循环使用的，当日志量达到max_wal_size，会触发积极的CheckPoint，此行为为同步写，会全速进行数据写入动作，对I/O影响大。
- 手动CheckPoint：此行为为同步写，会全速进行数据写入动作，对I/O影响大。
- 重启实例：此行为为同步写，会全速进行数据写入动作，对I/O影响大。

26.4. 数据库自治服务DAS简介

数据库自治服务DAS（Database Autonomy Service）是一种基于机器学习和专家经验实现数据库自感知、自修复、自优化、自运维及自安全的云服务，帮助您消除人工操作引发的服务故障，有效保障数据库服务的稳定、安全及高效。

前提条件

RDS PostgreSQL数据库实例为高可用版。

功能介绍

RDS PostgreSQL的自治服务主要包含如下功能：

- 会话管理
  查看、导出和结束目标实例的会话。
- 实时性能
  实时查看数据库实例性能。
- 空间分析
  直观地查看数据库实例的空间使用概况、空间剩余可用天数，以及数据库中某个表的空间使用情况、空间碎片、空间异常诊断等，能够帮助您及时发现数据库中的空间异常，避免影响数据库稳定性。
- 性能洞察
  负载监控、分析、性能调优的利器，以简单直观的方式帮助您迅速评估数据库负载，找到性能问题的源头，提升数据库的稳定性。
- 性能趋势
  支持多种查看性能趋势的方式，不仅支持性能趋势区间查看，也支持性能趋势对比查看和自定义性能趋势查看。
- 慢SQL
  查看数据库实例的慢日志趋势、慢日志执行情况和慢日志优化建议等。
- SQL洞察和审计
  在全量请求和安全审计的基础上，融合了搜索和SQL洞察，帮助您更好地获取RDS数据库实例SQL语句的具体信息，排查各种性能问题。

26.5. 一键诊断

26.5.1. 会话管理

数据库自治服务DAS为RDS PostgreSQL提供会话管理功能，方便您查看、导出和结束目标实例的会话。本文介绍如何使用会话管理功能。

前提条件

RDS PostgreSQL数据库实例为高可用版。

进入会话管理页面

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > 一键诊断**。
3. 单击**会话管理**页签。

> **说明** 详细的内容请参见 实例会话。

26.5.2. 实时性能

数据库自治服务DAS为RDS PostgreSQL提供实时性能功能，可以实时查看数据库实例性能。本文介绍如何使用实时性能功能。

前提条件

RDS PostgreSQL数据库实例为高可用版。

进入实时性能页面

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > 一键诊断**。
3. 单击**实时性能**页签。

> **说明** 详细内容请参见 实时性能。

26.5.3. 空间分析

数据库自治服务DAS为RDS PostgreSQL提供空间分析功能，可以直观地查看数据库实例的空间使用概况、空间剩余可用天数，以及数据库中某个表的空间使用情况、空间碎片、空间异常诊断等，能够帮助您及时发现数据库中的空间异常，避免影响数据库稳定性。

前提条件

RDS PostgreSQL数据库实例为高可用版。

进入空间分析页面

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > 一键诊断**。
3. 单击**空间分析**页签。

> **说明** 详细说明请参见 空间分析。

26.5.4. 性能洞察

性能洞察，是数据库自治服务DAS里专注于RDS实例负载监控、关联分析、性能调优的利器，以简单直观的方式帮助用户迅速评估数据库负载，找到性能问题的源头，提升数据库的稳定性。

前提条件

RDS PostgreSQL数据库实例为高可用版。

背景信息

用于性能洞察的数据主要来源如下：

- 如果目标实例已经开启了performance_schema，直接采集和分析performance_schema中的数据。
- 如果目标实例未开启performance_schema，则采集和分析活跃会话数据。

进入性能洞察页面

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > 一键诊断**。
3. 在**性能洞察**页签，单击**开启性能洞察**，在弹出的对话框中单击**确定**开启**性能洞察**。

> **说明** 详细的内容请参见 性能洞察。

26.6. 性能趋势

数据库自治服务DAS为RDS PostgreSQL提供性能趋势功能，支持多种查看性能趋势的方式，不仅支持性能趋势区间查看，也支持性能趋势对比查看和自定义性能趋势查看。

前提条件

RDS PostgreSQL数据库实例为高可用版。

进入性能趋势页面

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > 性能趋势**。

> **说明** 详细内容请参见 性能趋势。

26.7. 慢SQL

数据库自治服务DAS为RDS PostgreSQL提供慢SQL功能，可以查看数据库实例的慢日志趋势、慢日志执行情况和慢日志优化建议等。

前提条件

RDS PostgreSQL数据库实例为高可用版。

进入慢SQL页面

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > 慢SQL**。

> **说明** 详细内容请参见 分析慢日志。

26.8. SQL洞察和审计

数据库自治服务DAS提供SQL洞察和审计功能，在全量请求和安全审计的基础上，融合了搜索和SQL洞察，帮助您更好地获取RDS数据库实例SQL语句的具体信息，排查各种性能问题。

前提条件

- 您的账号已购买DAS专业版。如何购买请参见DAS专业版购买。
- RDS PostgreSQL数据库 实例为高可用版。
- 如果是RAM用户，则必须具备RDS的读写权限，例如AliyunRDSFullAccess权限。如何为RAM用户授权，请参见通过RAM对RDS进行权限管理。

计费

请参见DAS专业版费用说明。

> **说明** 开通SQL洞察和审计之后，原 SQL审计（数据库审计）将停止计费，新版本的SQL洞察和审计将按照DAS专业版进行收费。

功能介绍

- 搜索：查询并导出SQL语句执行历史及其对应的各种信息，如数据库、状态、执行时间等。
- SQL洞察：对SQL进行健康情况诊断、性能问题排查、业务流量分析等。

开启SQL洞察和审计

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > SQL洞察和审计**。
3. 在右侧页面中，如果界面提示您未开启该功能，单击 **一键开启**，并在弹出的窗口中单击 **开启专业版**。

> **说明** 如果页面显示 搜索和 SQL洞察页签，表示您已经开启了 **SQL洞察和审计**功能。

使用搜索功能

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > SQL洞察和审计**。
3. 在右侧页面中单击**搜索**页签，设置查询条件，即可查询相关信息。关于**搜索**的更多信息，请参见搜索。

使用SQL洞察功能

1. 访问RDS实例列表，在上方选择地域，然后单击目标实例ID。
2. 在左侧导航栏中，选择**自治服务 > SQL洞察和审计**。
3. 在右侧页面中单击**SQL洞察**页签即可查看相关信息。关于**SQL洞察**的更多信息，请参见SQL洞察。

27.标签

27.1. 创建标签

如果您有大量实例，可以通过给实例绑定标签，对实例进行分类管理。每个标签由一对键值组成，您可以通过键值，对实例进行二级分类。

限制说明

- 每个实例最多可以绑定20个标签，且标签键必须唯一。相同的标签键会被覆盖。
- 每次最多设置50个实例进行批量标签绑定。
- 不同地域的标签信息是独立的。
- 任一标签在解绑后，如果没有绑定任何实例，则该标签会被删除。

操作步骤

1. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
2. 选择标签添加方式。
   - 单个添加标签：选择目标实例后的**+添加标签** 。

     > **说明** 您也可以选择目标实例后的 **更多** > **编辑标签**。

   - 批量添加标签：勾选要批量添加标签的实例，单击**编辑标签**，如下图所示。

3. 单击**创建标签**，输入标签的**键**和**值**，单击**确定**。

   > **说明** 如果您已经新建了标签，可以单击 **选择标签**，选择历史标签。

4. 如果选择了多个实例批量添加标签，需要选择**绑定方式**，单击**确定**完成绑定。
   - 选择，实例会新增绑定标签，而不解绑原来绑定的标签。例如，原有标签key1:value1，追加新标签key2:value2以后，实例就有了两个标签，分别是key1:value1和key2:value2。

     > **说明** 如果追加标签的 键和已有标签的 键相同，则会覆盖原有的标签。例如，原有标签key1:value1，追加新标签key1:value2以后，实例还是只有一个标签，即key1:value2。

   - 选择，解绑实例的原有标签，并绑定此次选择或创建的标签。例如，原有标签key1:value1，用新标签key2:value2覆盖现有标签以后，实例的标签为key2:value2。

相关API

API	描述
创建标签	绑定标签。

27.2. 删除标签

如果实例调整或者不再需要标签，您可以删除该实例的标签。

限制说明

- 每次解绑的标签数量不能超过20个。
- 任一标签在解绑后，如果没有绑定任何实例，则该标签会被删除。

操作步骤

1. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
2. 鼠标放在目标实例的标签上，单击**编辑** 。

> **说明** 您也可以选择目标实例后的 **更多** > **编辑标签** 。

3. 单击要删除的标签后的X删除标签，如下图所示。

4. 单击**确定**，完成操作。

相关API

API	描述
解绑标签	解绑标签。

27.3. 根据标签筛选实例

实例绑定标签后，您可以根据标签筛选实例。

1. 登录RDS管理控制台，在左侧单击**实例列表**，然后在上方选择地域。
2. 通过标签的**键**和**值**筛选实例。

> **说明** 按标签筛选实例后，如果您需要取消筛选，可以删除 **标签**键右侧的筛选条件。

相关API

API	描述
查询标签	查询标签。

28.最佳实践

28.1. RDS PostgeSQL权限管理最佳实践

本文介绍如何对RDS PostgrSQL实例中的账号权限进行精细化管理。

配置原则

创建角色（Role）作为权限集合，对权限管理的细分操作集中在Role上。Role没有登录权限。在Role的基础上创建user账号来作为登录账号使用。使用 `user = role + login权限` 的账号权限管理模型，当Role的权限变化时，user账号权限会自动跟随Role变化。

权限管理设计模型

该模型简单有效，推荐绝大多 数客户使用。

- 1个RDS PostgreSQL高权限账号，具有所有权限，由少量资深DBA掌握。
- 项目或者团队维度，1个资源owner账号，2个基本Role：{project}_role_readwrite、{project}_role_readonly。

> **说明** 如果有更多细分需求，可以根据需要新建Role 。

- 在Role的基础上创建业务账号： `业务账号user = role + login权限` 。
- 1个项目或团队可以有多个Schema。权限分配尽量以Schema或Role为单位。
- 业务表请勿放到 `schema public` 中。因为PostgreSQL默认所有用户对 `schema public` 都有**CREATE**和 **USAGE**权限。

权限规划示例

本示例以项目维度进行权限管理示例，同样也适用于团队维度。

- DBA拥有RDS PostgreSQL实例的高权限账号，名称是dbsuperuser。
- 业务项目名称是rdspg，新建schema名称是rdspg、rdspg_1。

项目中新增的资源owner账号和Role规划如下：

user/Role	schema中表权限	schema中存储过程权限
rdspg_owner (user)，是唯一的项目资源owner账号	• DDL：CREATE、DROP、ALTER • DQL：SELECT • DML：UPDATE、INSERT、DELETE	• DDL：CREATE、DROP、ALTER • DQL：SELECT，调用存储过程
rdspg_role_readwrite (role)	• DQL：SELECT • DML：UPDATE、INSERT、DELETE	DQL（SELECT，调用存储过程），若存储过程有DDL操作，会抛出权限相关错误。
rdspg_role_readonly (role)	DQL(SELECT)	DQL（SELECT，调用存储过程），若存储过程有DDL或者DML操作，会抛出权限相关错误。

新增业务账号时，根据不同需求，采用如下管理模式创建：

- rdspg_readwrite = rdspg_role_readwrite + login权限
- rdspg_readonly = rdspg_role_readonly + login权限

配置步骤

1. 创建项目资源owner账号rdspg_owner和项目Role。

   DBA使用dbsuperuser高权限账号执行如下操作。

```
--- rdspg_owner 是项目管理账号,此处密码仅为示例，请注意修改。 CREATE USER rdspg_owner WITH LO
GIN PASSWORD 'asdfy181BASDfadasdbfas'; CREATE ROLE rdspg_role_readwrite; CREATE ROLE rd
spg_role_readonly; --- 设置：对于rdspg_owner 创建的表，rdspg_role_readwrite 有 DQL（SELECT
）、DML（UPDATE、INSERT、DELETE）权限。 ALTER DEFAULT PRIVILEGES FOR ROLE rdspg_owner GRA
NT ALL ON TABLES TO rdspg_role_readwrite; --- 设置：对于rdspg_owner 创建的SEQUENCES，rdsp
g_role_readwrite 有 DQL（SELECT）、DML（UPDATE、INSERT、DELETE）权限。 ALTER DEFAULT PRIVI
LEGES FOR ROLE rdspg_owner GRANT ALL ON SEQUENCES TO rdspg_role_readwrite; --- 设置：对
于 rdspg_owner 创建的表， rdspg_role_readonly 只有 DQL（SELECT）权限。 ALTER DEFAULT PRIVI
LEGES FOR ROLE rdspg_owner GRANT SELECT ON TABLES TO rdspg_role_readonly;
```

2. 创建rdspg_readwrite、rdspg_readonly业务账号。

   DBA使用dbsuperuser高权限账号执行如下操作。

```
--- rdspg_readwrite只有 DQL（SELECT）、DML（UPDATE、INSERT、DELETE）权限。 CREATE USER rds
pg_readwrite WITH LOGIN PASSWORD 'dfandfnapSDhf23hbEfabf'; GRANT rdspg_role_readwrite T
O rdspg_readwrite; --- rdspg_readonly只有 DQL（SELECT）权限。 CREATE USER rdspg_readonly
WITH LOGIN PASSWORD 'F89h912badSHfadsd01zlk'; GRANT rdspg_role_readonly TO rdspg_readon
ly;
```

3. 创建schema rdspg，并授权给项目Role。

   DBA使用dbsuperuser高权限账号执行如下操作。

```
--- schema rdspg的owner是 rdspg_owner账号 CREATE SCHEMA rdspg AUTHORIZATION rdspg_owner;
--- 授权ROLE相关SCHEMA访问权限。 GRANT USAGE ON SCHEMA rdspg TO rdspg_role_readwrite; GRA
NT USAGE ON SCHEMA rdspg TO rdspg_role_readonly;
```

> **说明** rdspg_readwrite和rdspg_readonly自动继承了相关Role的权限变更，不需要再额外操作。

应用场景示例

场景1：使用rdspg_owner账号：对schema rdspg中的表进行DDL（CREATE、DROP、ALTER）操作

```
CREATE TABLE rdspg.test(id bigserial primary key, name text); CREATE INDEX idx_test_name on
rdspg.test(name);
```

场景2：使用 rdspg_readwrite/rdspg_readonly 账号进行业务开发

业务开发遵循最小权限原则，尽量使用rdspg_readonly账号，需要DML操作的地方才使用rdspg_readwrite账号。这样也方便在业务层做读写分离。

说明

- 业务层做读写分离，避免了自动读写分离中间件proxy带来的额外成本和性能损耗。
- 即使目前还没有使用只读实例，也建议区分 readonly客户端、readwrite客户端，为使用只读实例做准备。readonly客户端建议使用readonly账号，最小权限原则，规避权限误用。
  - readonly客户端，使用readonly账号，设置JDBC URL：`只读实例1地址,只读实例2地址,读写实例地址`。
  - readwrite客户端，使用readwrite账号，设置JDBC URL：`读写实例地址`。

- 使用rdspg_readwrite账号，对schema rdspg中的表进行DQL（SELECT）、DML（UPDATE、INSERT、DELETE）操作：

```
INSERT INTO rdspg.test (name) VALUES('name0'),('name1'); SELECT id,name FROM rdspg.test L
IMIT 1; --- rdspg_readwrite没有 DDL（CREATE、DROP、ALTER）权限 CREATE TABLE rdspg.test2(id
int); ERROR: permission denied for schema rdspg LINE 1: create table rdspg.test2(id int);
DROP TABLE rdspg.test; ERROR: must be owner of table test ALTER TABLE rdspg.test ADD id2
int; ERROR: must be owner of table test CREATE INDEX idx_test_name on rdspg.test(name); E
RROR: must be owner of table test
```

- 使用rdspg_readonly账号，对schema rdspg中的表进行DQL（SELECT）操作：

```
INSERT INTO rdspg.test (name) VALUES('name0'),('name1'); ERROR: permission denied for tab
le test SELECT id,name FROM rdspg.test LIMIT 1; id | name ----+------- 1 | name0 (1 row)
```

场景3：不同项目交叉授权

如果有另外1个项目employee，需求为账号employee_readwrite增加rdspg项目的表只读权限。DBA使用dbsuperuser高权限账号做如下操作：

```
--- 给账号 employee_readwrite 加上 rdspg_role_readonly 权限集合。 GRANT rdspg_role_readonly TO
employee_readwrite;
```

场景4：项目新增 schema rdspg_2，并授权给项目Role

rdspg_readwrite、rdspg_readonly、employee_readwrite账号自动继承了相关Role的权限变更，不需要再额外操作。DBA使用dbsuperuser 高权限账号做如下操作：

```
CREATE SCHEMA rdspg_1 AUTHORIZATION rdspg_owner; --- 授权ROLE相关SCHEMA访问权限。 --- CREATE
使得 rdspg_role_admin 对schema rdspg_1中的表有 DDL（CREATE、DROP、ALTER）权限。 GRANT USAGE ON
SCHEMA rdspg_1 TO rdspg_role_readwrite; GRANT USAGE ON SCHEMA rdspg_1 TO rdspg_role_readonl
y;
```

账号权限查询

通过本文介绍的账号权限管理模型创建的账号，可以通过如下方式查询具体权限信息。

- 使用PostgreSQL客户端命令行终端连接RDS PostgreSQL数据库，具体请参见连接PostgreSQL实例。然后使用`\du`命令查看：

```
cash=> \du
                                    List of roles
        Role name        |                  Attributes                  |                  Member of
-------------------------+----------------------------------------------+-------------------------------------------
 [illegible]             |                                              | {}
 employee_owner          |                                              | {}
 employee_readonly       |                                              | {employee_role_readonly}
 employee_readwrite      |                                              | {rdspg_role_readonly,employee_role_readwrite}
 employee_role_readonly  | Cannot login                                 | {}
 employee_role_readwrite | Cannot login                                 | {}
 [illegible]             | [illegible]                                  | {}
 rdspg_owner             |                                              | {}
 rdspg_readonly          |                                              | {rdspg_role_readonly}
 rdspg_readwrite         |                                              | {rdspg_role_readwrite}
 rdspg_role_readonly     | Cannot login                                 | {}
 [illegible]             | [illegible]                                  | {}
```

从上述查询结果示例中可以看出：employee_readwrite账号的**Member of**列中，内容为 `rdspg_role_readonly,employee_role_readwrite` ，因此，此账号对employee项目表具有DQL和DML权限，对rdspg项目表具有DQL权限。

- 使用SQL查询：

```
SELECT r.rolname, r.rolsuper, r.rolinherit, r.rolcreaterole, r.rolcreatedb, r.rolcanlogin
, r.rolconnlimit, r.rolvaliduntil, ARRAY(SELECT b.rolname FROM pg_catalog.pg_auth_members
m JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid) WHERE m.member = r.oid) as memberof ,
r.rolreplication , r.rolbypassrls FROM pg_catalog.pg_roles r WHERE r.rolname !~ '^pg_' OR
DER BY 1;
```

28.2. 快速、轻松地将自建PostgreSQL迁移到阿里云RDS PostgreSQL

本文介绍2种方式将自建PostgreSQL迁移到阿里云RDS PostgreSQL。

使用数据传输DTS服务

- 优势
  - 图形化操作，上手简单，出问题方便求助客服。
  - 支持增量迁移，数据割接时停机时间短。
  - 全量迁移免费，增量迁移收费。
- 劣势
  不支持通过上传文件的方式指定表。

具体操作步骤请参见迁移方案概览。

使用原生PostgreSQL工具

如果使用DTS迁移无法满足业务需求，您可以使用PostgreSQL自带的工具进行迁移。

- 优势
  - PostgreSQL原生工具，兼容性好。
  - 执行速度快。
  - 执行选项灵活，支持schema、database，可以排除不需要的表、选择需要的表等。
- 劣势
  - 支持全量迁移，不支持增量迁移。
  - 对运维人员要求较高，需要一定的PostgreSQL和Linux技术基础。

具体操作步骤请参见使用pg_dump和pg_restore将自建PostgreSQL数据库迁移到RDS PostgreSQL。

28.3. 设置PostgreSQL排序规则（Collation）

初始化数据库集群时，可以设置数据库的字符串排序、字符归类方法、数值格式、日期格式、时间格式、货币格式等。另外，为了支持国际化，数据库通常会涉及到LC_COLLATE和LC_CTYPE的概念。

LC_COLLATE	String sort order
LC_CTYPE	字符分类
LC_MESSAGES	消息的语言
LC_MONETARY	货币使用的格式
LC_NUMERIC	数字使用的格式
LC_TIME	时间日期使用的格式

您可以利用这些特性，按本土化需求，输出对应的顺序或者格式。本文将通过示例介绍如何设置数据库的本土化信息以及如何设置输出结果按中文的拼音顺序进行排序。

PostgreSQL 支持的字符集类型

您可以参考PostgreSQL 的官方文档。

Name	Description	Language	Server	Bytes/Char	Aliases
BIG5	Big Five	Traditional Chinese	No	1月2日	WIN950, Windows950
EUC_CN	Extended UNIX Code-CN	Simplified Chinese	Yes	1月3日	-
EUC_JP	Extended UNIX Code-JP	Japanese	Yes	1月3日	-
EUC_JIS_2004	Extended UNIX Code-JP, JIS X 0213	Japanese	Yes	1月3日	-
EUC_KR	Extended UNIX Code-KR	Korean	Yes	1月3日	-
EUC_TW	Extended UNIX Code-TW	Traditional Chinese, Taiwanese	Yes	1月3日	-
GB18030	National Standard	Chinese	No	1月4日	-
GBK	Extended National Standard	Simplified Chinese	No	1月2日	WIN936, Windows936
ISO_8859_5	ISO 8859-5, ECMA 113	Latin/Cyrillic	Yes	1	-

Name	Description	Language	Server	Bytes/Char	Aliases
ISO_8859_6	ISO 8859-6, ECMA 114	Latin/Arabic	Yes	1	-
ISO_8859_7	ISO 8859-7, ECMA 118	Latin/Greek	Yes	1	-
ISO_8859_8	ISO 8859-8, ECMA 121	Latin/Hebrew	Yes	1	-
JOHAB	JOHAB	Korean (Hangul)	No	1月3日	-
KOI8R	KOI8-R	Cyrillic (Russian)	Yes	1	KOI8
KOI8U	KOI8-U	Cyrillic (Ukrainian)	Yes	1	-
LATIN1	ISO 8859-1, ECMA 94	Western European	Yes	1	ISO88591
LATIN2	ISO 8859-2, ECMA 94	Central European	Yes	1	ISO88592
LATIN3	ISO 8859-3, ECMA 94	South European	Yes	1	ISO88593
LATIN4	ISO 8859-4, ECMA 94	North European	Yes	1	ISO88594
LATIN5	ISO 8859-9, ECMA 128	Turkish	Yes	1	ISO88599
LATIN6	ISO 8859-10, ECMA 144	Nordic	Yes	1	ISO885910
LATIN7	ISO 8859-13	Baltic	Yes	1	ISO885913
LATIN8	ISO 8859-14	Celtic	Yes	1	ISO885914
LATIN9	ISO 8859-15	LATIN1 with Euro and accents	Yes	1	ISO885915
LATIN10	ISO 8859-16, ASRO SR 14111	Romanian	Yes	1	ISO885916
MULE_INTERNAL	Mule internal code	Multilingual Emacs	Yes	1月4日	-

Name	Description	Language	Server	Bytes/Char	Aliases
SJIS	Shift JIS	Japanese	No	1月2日	Mskanji, ShiftJIS, WIN932, Windows932
SHIFT_JIS_2004	Shift JIS, JIS X 0213	Japanese	No	1月2日	-
SQL_ASCII	unspecified (see text)	any	Yes	1	-
UHC	Unified Hangul Code	Korean	No	1月2日	WIN949, Windows949
UTF8	Unicode, 8-bit	all	Yes	1月4日	Unicode
WIN866	Windows CP866	Cyrillic	Yes	1	ALT
WIN874	Windows CP874	Thai	Yes	1	-
WIN1250	Windows CP1250	Central European	Yes	1	-
WIN1251	Windows CP1251	Cyrillic	Yes	1	WIN
WIN1252	Windows CP1252	Western European	Yes	1	-
WIN1253	Windows CP1253	Greek	Yes	1	-
WIN1254	Windows CP1254	Turkish	Yes	1	-
WIN1255	Windows CP1255	Hebrew	Yes	1	-
WIN1256	Windows CP1256	Arabic	Yes	1	-
WIN1257	Windows CP1257	Baltic	Yes	1	-
WIN1258	Windows CP1258	Vietnamese	Yes	1	ABC, TCVN, TCVN5712, VSCII

查询字符集支持的LC_COLLATE和LC_CTYPE信息

您可以使用如下SQL查询系统表pg_collation，来获取字符集支持的LC_COLLATE和LC_CTYPE信息。

```
test=> select pg_encoding_to_char(collencoding) as encoding,collname,collcollate,collctype
from pg_collation ;
```

返回结果如下所示，encoding为空时，表示这个collation支持所有的字符集。

```
 encoding | collname | collcollate | collctype ------------+-----------------------+-------
----------------+----------------------- | default | | | C | C | C | POSIX | POSIX | POSIX
UTF8 | aa_DJ | aa_DJ.utf8 | aa_DJ.utf8 LATIN1 | aa_DJ | aa_DJ | aa_DJ LATIN1 | aa_DJ.iso885
91 | aa_DJ.iso88591 | aa_DJ.iso88591 UTF8 | aa_DJ.utf8 | aa_DJ.utf8 | aa_DJ.utf8 UTF8 | aa_
ER | aa_ER | aa_ER UTF8 | aa_ER.utf8 | aa_ER.utf8 | aa_ER.utf8 ....... EUC_CN | zh_CN | zh_
CN | zh_CN UTF8 | zh_CN | zh_CN.utf8 | zh_CN.utf8 EUC_CN | zh_CN.gb2312 | zh_CN.gb2312 | zh
_CN.gb2312 UTF8 | zh_CN.utf8 | zh_CN.utf8 | zh_CN.utf8 UTF8 | zh_HK | zh_HK.utf8 | zh_HK.ut
f8 UTF8 | zh_HK.utf8 | zh_HK.utf8 | zh_HK.utf8 EUC_CN | zh_SG | zh_SG | zh_SG UTF8 | zh_SG
| zh_SG.utf8 | zh_SG.utf8 EUC_CN | zh_SG.gb2312 | zh_SG.gb2312 | zh_SG.gb2312 UTF8 | zh_SG.
utf8 | zh_SG.utf8 | zh_SG.utf8 EUC_TW | zh_TW | zh_TW.euctw | zh_TW.euctw UTF8 | zh_TW | zh
_TW.utf8 | zh_TW.utf8 EUC_TW | zh_TW.euctw | zh_TW.euctw | zh_TW.euctw UTF8 | zh_TW.utf8 |
zh_TW.utf8 | zh_TW.utf8 UTF8 | zu_ZA | zu_ZA.utf8 | zu_ZA.utf8 LATIN1 | zu_ZA | zu_ZA | zu_
ZA LATIN1 | zu_ZA.iso88591 | zu_ZA.iso88591 | zu_ZA.iso88591 UTF8 | zu_ZA.utf8 | zu_ZA.utf8
| zu_ZA.utf8 (869 rows)
```

设置数据库的本土化（collate）信息

- 设置字段的本土化
  前提条件
  执行如下SQL命令，查询当前数据库的字符集（encoding）类型，并了解清楚与您当前数据库字符集兼容的collate。

  ```
  postgres=# select datname,pg_encoding_to_char(encoding) as encoding from pg_database;
  ```

  返回结果如下所示：

  ```
   datname | encoding --------------------+----------- template1 | UTF8 template0 | UTF8 db
  | SQL_ASCII db1 | EUC_CN contrib_regression | UTF8 test01 | UTF8 test02 | UTF8 postgres |
  UTF8 (8 rows)
  ```

  操作步骤

  i. 在创建表时，执行如下命令，指定兼容当前字符集的collate。

  ```
  CREATE TABLE test1 ( a text COLLATE "de_DE", b text COLLATE "es_ES", ... );
  ```

  ii. 执行如下命令，修改列collate。

  > **说明** 修改列collate时，会导致rewrite table，大表请谨慎操作。

  ```
  alter table a alter c1 type text COLLATE "en_US";
  ```

- 在SQL使用本土化

- 使用本土化，改变order by输出排序。命令如下：

```
test=# select * from a order by c1 collate "C";
```

输出结果如下：

```
 c1 -------- Tom Alice (2 rows) test=# select * from a order by c1 collate "en_US"; c1
-------- Alice Tom (2 rows)
```

- 使用本土化，改变操作符的结果。示例如下：
命令：

```
select * from a where c1 > 'Tom' collate "C";
```

输出结果如下：

```
 c1 -------- Alice (1 row)
```

命令：

```
select * from a where c1 > 'Tom' collate "en_US";
```

输出结果如下：

```
 c1 ---- (0 rows)
```

- 使用本土化索引进行排序
排序语句中的collate与索引的collate保持一致，才能使用这个索引进行排序。命令如下：

```
create index idxa on a(c1 collate "en_US"); explain select * from a order by c1 collate "
en_US";
```

输出结果如下：

```
 QUERY PLAN ---------------------------------------------------------------------------- Inde
x Only Scan using idxa on a (cost=0.15..31.55 rows=1360 width=64) (1 row)
```

设置输出结果按拼音排序

您可以通过如下四种方法来设置按拼音排序：

- 使用本土化 SQL。该方法不修改原有数据。命令如下：

```
select * from a order by c1 collate "en_US";
```

输出结果如下：

```
 c1 -------- Alice Tom (2 rows)
```

- 使用本土化字段。若已有数据，使用该方法时需要调整原有数据。命令如下：

```
alter table a alter c1 type text COLLATE "en_US";
```

- 使用本土化索引以及本土化 SQL。该方法不修改原有数据。命令如下：

```
create index idxa on a(c1 collate "en_US"); explain select * from a order by c1 collate "
en_US";
```

输出结果如下：

```
 QUERY PLAN ---------------------------------------------------------------------------- Inde
x Only Scan using idxa on a (cost=0.15..31.55 rows=1360 width=64) (1 row)
```

- 将数据库的collate设置为en_US，数据会将默认使用这个collate按拼音排序。命令如下：

```
create database test03 encoding 'UTF8' lc_collate 'en_US.utf8' lc_ctype 'en_US.utf8' temp
late template0; \c test03 select * from (values ('Alice'),('Tom')) as a(c1) order by c1 ;
```

输出结果如下：

```
 c1 -------- Alice Tom (2 rows)
```

> **说明** 如果是中文，在设置按拼音排序时，要注意多音字。例如重庆（chongqing），在编码时，重可能会按照zhong编码，影响输出。

在Greenplum中设置输出结果按拼音排序

Greenplum不支持单列设置collate，按拼音排序有些许不同。

在Greenplum中，可以使用字符集转换，按对应二进制排序，得到拼音排序的效果，如下面的命令所示：

```
select * from (values ('Alice'), ('Tom')) t(id) order by byteain(textout(convert(id,'UTF8',
'EUC_CN')));
```

输出结果如下：

```
 id -------- Alice Tom (2 rows)
```

参考文档

PostgreSQL 9.6.2 Documentation — Chapter 23. Localization

28.4. 批量更新、删除或插入数据

批量操作可以减少数据库与应用程序的交互次数，提高数据处理的吞吐量。本文将通过示例介绍如何批量插入、更新和删除数据。

您可以通过如下四种方法进行批量插入数据：

- 使用 `INSERT INTO ... SELECT` 的方法。

```
 postgres=# INSERT INTO tbl1 (id, info ,crt_time) SELECT GENERATE_SERIES(1,10000),'test',
NOW(); INSERT 0 10000 postgres=# SELECT COUNT(*) FROM tbl1; count ------- 10001 (1 row)
```

- 使用 `VALUES(),(),...();` 的方法。

```
postgres=# INSERT INTO tbl1 (id,info,crt_time) VALUES (1,'test',NOW()), (2,'test2',NOW())
, (3,'test3',NOW()); INSERT 0 3
```

- 使用 `BEGIN; ...多条INSERT...; END;` 的方法。严格来说，这不属于批量，但可以减少事务提交时的同步等待，同样可以提升性能。

```
postgres=# BEGIN; BEGIN postgres=# INSERT INTO tbl1 (id,info,crt_time) VALUES (1,'test',N
OW()); INSERT 0 1 postgres=# INSERT INTO tbl1 (id,info,crt_time) VALUES (2,'test2',NOW())
; INSERT 0 1 postgres=# INSERT INTO tbl1 (id,info,crt_time) VALUES (3,'test3',NOW()); INS
ERT 0 1 postgres=# END; COMMIT
```

- 使用COPY协议。COPY协议与INSERT协议不一样，更加精简，插入效率高。

```
 test03=# \d test Table "public.test" Column | Type | Modifiers ----------+--------------
---------------+----------- id | integer | not null info | text | crt_time | timestamp wi
thout time zone | Indexes: "test_pkey" PRIMARY KEY, btree (id) test03=# COPY test FROM st
din; Enter data to be copied followed by a newline. End with a backslash and a period on
a line by itself. >> 8 'test' '2017-01-01' >> 9 'test9' '2017-02-02' >> \. COPY 2
```

> **说明**
> 不同的语言驱动，对应的COPY接口不同，请参见如下文档：
> - PostgreSQL JDBC Driver - JDBC 4.2 9.4.1209 API
> - PostgreSQL 9.6.2 Documentation — Functions Associated with the COPY Command

批量更新数据

```
test03=# UPDATE test SET info=tmp.info from (VALUES (1,'new1'),(2,'new2'),(6,'new6')) AS tm
p (id,info) WHERE test.id=tmp.id; UPDATE 3 test03=# SELECT * FROM test; id | info | crt_tim
e ----+--------------+---------------------------- 3 | hello | 2017-04-24 15:31:49.14291 4
| digoal0123 | 2017-04-24 15:42:50.912887 5 | hello digoal | 2017-04-24 15:57:29.622045 1 |
new1 | 2017-04-24 15:58:55.610072 2 | new2 | 2017-04-24 15:28:20.37392 6 | new6 | 2017-04-2
4 15:59:12.265915 (6 rows)
```

批量删除数据

```
test03=# DELETE FROM test USING (VALUES (3),(4),(5)) AS tmp(id) WHERE test.id=tmp.id; DELET
E 3 test03=# SELECT * FROM test; id | info | crt_time ----+---------+----------------------
------ 1 | new1 | 2017-04-24 15:58:55.610072 2 | new2 | 2017-04-24 15:28:20.37392 6 | new6
| 2017-04-24 15:59:12.265915
```

如果要清除全表，建议您使用TRUNCATE。

```
test03=# SET lock_timeout = '1s'; SET test03=# TRUNCATE test; TRUNCATE TABLE test03=# SELEC
T * FROM test; id | info | crt_time ----+------+---------- (0 rows)
```

28.5. 查找最耗费资源的 SQL（Top SQL）

数据库是较大型的应用，对于繁忙的数据库，需要消耗大量的内存、CPU、IO、网络资源。SQL 优化是数据库优化的手段之一，而为了达到 SQL 优化的最佳效果，您首先需要了解最消耗资源的 SQL（Top SQL），例如 IO 消耗最高的 SQL。
数据库资源分为多个维度、CPU、内存、IO 等，为能够从各个维度层面查找最消耗数据库资源的 SQL，您可以使用 pg_stat_statements 插件统计数据库的资源开销和分析 Top SQL。

本文将通过示例介绍如何创建 pg_stat_statements 插件、如何分析 Top SQL 以及如何重置统计信息。

执行如下命令，在需要查询 TOP SQL 的数据库中，创建 pg_stat_statements 插件。

```
CREATE EXTENSION pg_stat_statements;
```

pg_stat_statements 输出内容介绍

通过查询 pg_stat_statements 视图，您可以得到数据库资源开销的统计信息。SQL 语句中的一些过滤条件在 pg_stat_statements 中会被替换成变量，可以减少重复显示的问题。

pg_stat_statements 视图包含了一些重要信息，例如：

- SQL 的调用次数，总耗时，最快执行时间，最慢执行时间，平均执行时间，执行时间的方差（看出抖动），总共扫描、返回或处理了多少行。
- shared buffer 的使用情况：命中、未命中、产生脏块、驱逐脏块。
- local buffer 的使用情况：命中、未命中、产生脏块、驱逐脏块。
- temp buffer 的使用情况：读了多少脏块、驱逐脏块。
- 数据块的读写时间。

下表列出了 pg_stat_statements 输出内容中各参数的含义。

参数名称	类型	参考	说明
userid	oid	pg_authid.oid	OID of user who executed the statement.
dbid	oid	pg_database.oid	OID of database in which the statement was executed.
queryid	bigint	无	Internal hash code, computed from the statement's parse tree.
query	text	无	Text of a representative statement.
calls	bigint	无	Number of times executed.
total_time	double precision	无	Total time spent in the statement, in milliseconds.
min_time	double precision	无	Minimum time spent in the statement, in milliseconds.
max_time	double precision	无	Maximum time spent in the statement, in milliseconds.
mean_time	double precision	无	Mean time spent in the statement, in milliseconds.
stddev_time	double precision	无	Population standard deviation of time spent in the statement, in milliseconds.

参数名称	类型	参考	说明
rows	bigint	无	Total number of rows retrieved or affected by the statement.
shared_blks_hit	bigint	无	Total number of shared block cache hits by the statement.
shared_blks_read	bigint	无	Total number of shared blocks read by the statement.
shared_blks_dirtied	bigint	无	Total number of shared blocks dirtied by the statement.
shared_blks_written	bigint	无	Total number of shared blocks written by the statement.
local_blks_hit	bigint	无	Total number of local block cache hits by the statement.
local_blks_read	bigint	无	Total number of local blocks read by the statement.
local_blks_dirtied	bigint	无	Total number of local blocks dirtied by the statement.
local_blks_written	bigint	无	Total number of local blocks written by the statement.
temp_blks_read	bigint	无	Total number of temp blocks read by the statement.
temp_blks_written	bigint	无	Total number of temp blocks written by the statement.
blk_read_time	double precision	无	Total time the statement spent reading blocks, in milliseconds (if track_io_timing is enabled, otherwise zero).

参数名称	类型	参考	说明
blk_write_time	double precision	无	Total time the statement spent writing blocks, in milliseconds (if track_io_timing is enabled, otherwise zero).

分析 TOP SQL

- **最耗 IO SQL**
  - 执行如下命令，查询单次调用最耗 IO SQL TOP 5。

    ```
    SELECT userid::regrole, dbid, query FROM pg_stat_statements ORDER BY (blk_read_time+blk
    _write_time)/calls DESC LIMIT 5;
    ```

  - 执行如下命令，查询总最耗 IO SQL TOP 5。

    ```
    SELECT userid::regrole, dbid, query FROM pg_stat_statements ORDER BY (blk_read_time+blk
    _write_time) DESC LIMIT 5;
    ```

- **最耗时 SQL**
  - 执行如下命令，查询单次调用最耗时 SQL TOP 5。

    ```
    SELECT userid::regrole, dbid, query FROM pg_stat_statements ORDER BY mean_time DESC LIM
    IT 5;
    ```

  - 执行如下命令，查询总最耗时 SQL TOP 5。

    ```
    SELECT userid::regrole, dbid, query FROM pg_stat_statements ORDER BY total_time DESC LI
    MIT 5;
    ```

- **响应时间抖动最严重 SQL**
  执行如下命令，查询响应时间抖动最严重 SQL。

  ```
  SELECT userid::regrole, dbid, query FROM pg_stat_statements ORDER BY stddev_time DESC LIM
  IT 5;
  ```

- **最耗共享内存 SQL**
  执行如下命令，查询最耗共享内存 SQL。

  ```
  SELECT userid::regrole, dbid, query FROM pg_stat_statements ORDER BY (shared_blks_hit+sha
  red_blks_dirtied) DESC LIMIT 5;
  ```

- **最耗临时空间 SQL**
  执行如下命令，查询最耗临时空间 SQL。

  ```
  SELECT userid::regrole, dbid, query FROM pg_stat_statements ORDER BY temp_blks_written DE
  SC LIMIT 5;
  ```

重置统计信息

pg_stat_statements是累积的统计，如果要查看某个时间段的统计，需要查询快照的信息，详情请参见《PostgreSQL AWR报告（for 阿里云ApsaraDB PgSQL）》。

您也可以通过执行如下命令，来定期清理历史统计信息。

```
SELECT pg_stat_statements_reset();
```

参考文档

PostgreSQL 9.6.2 Documentation — F.29. pg_stat_statements

28.6. 逻辑订阅

RDS PostgreSQL提供逻辑订阅功能，支持在多个RDS PostgreSQL实例之间实现准实时表级单向同步，可以用于数据共享、数据汇总、数据拆分等业务场景。
如果您在全国多个地域部署业务，可以通过逻辑订阅共享数据，例如将中心数据共享给其他地域，或者将其他地域数据汇总到中心进行实时分析查询等。

本文简单介绍逻辑订阅的使用流程，示例的流程如下：

源实例的数据库src创建发布，发布表public.t1，然后在目的实例的数据库dst创建订阅sub1_from_pub1，订阅源库表public.t1。

更多详情请参见逻辑订阅原理与最佳实践。

注意事项

您可以在单个实例的两个表之间实现逻辑订阅，也可以在同一VPC内的两个实例间（必须使用内网地址）实现逻辑订阅。

前提条件

- 实例版本如下：
  - PostgreSQL 14（云盘）
  - PostgreSQL 13（云盘）
  - PostgreSQL 12（云盘）
  - PostgreSQL 11（云盘）
  - PostgreSQL 10（云盘）
- 实例参数wal_level的值为logical，您可以在控制台的**参数设置**页面进行修改，该参数修改后需要重启实例才能生效。重启实例会造成连接中断，请提前做好业务安排，谨慎操作。
- 如果是同一VPC内两个实例间实现逻辑订阅，需要在各自的白名单中放通实例所在VPC的IP段，例如放通172.16.0.0/16。
- 需要具有rds_superuser权限的账号，例如高权限账号或者高权限账号使用 `create role xxx with superuesr` 创建的其他账号。

操作步骤

源库创建新发布

1. 连接PostgreSQL源实例。
2. 在源库里创建新发布，命令如下：

   ```
   CREATE PUBLICATION <发布名称> FOR TABLE <表名>;
   ```

   示例

   ```
   create publication pub1 for table public.t1;
   ```

> 说明
> - 只能发布持久基表。详情请参见CREATE PUBLICATION。
> - 您可以通过 `select * frompg_publication;` 查看当前数据库有哪些发布。

目的库创建订阅

1. 连接PostgreSQL目的实例。
2. 此步骤以源库和目的库在相同的实例为例，如果是两个实例间实现逻辑订阅，请跳转至步骤3。
   执行如下命令：

```
select * from pg_create_logical_replication_slot('<订阅名称>','pgoutput');
```

3. 在目的库创建新订阅，命令如下：

```
CREATE SUBSCRIPTION <订阅名称> CONNECTION '<源库的实例连接信息>' PUBLICATION <源库的发布名称
>;
```

   示例

```
create subscription sub1_from_pub1 connection 'host=pgm-xxxxx.pgsql.singapore.rds.aliyu
ncs.com port=3433 user=test password=xxxxx dbname=src' publication pub1 with (enabled,
create_slot, slot_name='sub1_from_pub1');
```

> 说明
> - 源库的实例连接信息格式为：`host=<源实例内网域名> port=<源实例内网端口> user=<源实例有发布权限账号> password=<源实例账号的密码> dbname=<源实例数据库名>` 。
> - 如果源库和目的库在相同的实例，需要设置 `host=localhost` 、 `create_slot=false` ，port可以使用 `show port` 查询，通常为3002。
> - 您可以通过 `select * from pg_subscription;` 查看整个数据库集群有哪些订阅。
> - 源库的发布名称后可以使用with附加相关订阅参数，详情请参见CREATE SUBSCRIPTION。

28.7. 使用事件触发器实现DDL回收站、防火墙、增量订阅同步

RDS PostgreSQL开放事件触发器，可以实现DDL回收站、DDL防火墙、DDL增量订阅同步等功能，灵活使用事件触发器可以减少维护成本，保护数据安全。

前提条件

实例版本为PostgreSQL 10、11、12云盘版。

背景信息

如果您对数据库安全有非常高的要求，可以基于事件触发器创建DDL回收站规则、DDL防火墙规则，在多个维度保护数据安全：

- 事前防御：防止drop table、drop index、drop database等删库删表危险操作。
- 事后回档：在发生意外删表后，可以从回收站找回。

原理是使用pg_get_ddl_command和pg_get_ddl_drop这两个事件触发器收集并存入DDL语句到表dts_audit.dts_tb_ddl_command中，其具体实现在函数pg_func_ddl_command()中。其中dts_audit.dts_tb_ddl_command表结构如下：

```
 Column | Type | Collation | Nullable | Default | Storage | Stats target | Description ----
-------------+----------------------------+-----------+----------+--------------------+---
-------+--------------+------------- event | text | | | | extended | | tag | text | | | | e
xtended | | Command tag classid | oid | | | | plain | | OID of catalog the object belonged
in objid | oid | | | | plain | | OID the object had within the catalog objsubid | integer |
| | | plain | | Object sub-id (e.g. attribute number for columns) object_type | text | | |
| extended | | Type of the object schema_name | text | | | | extended | | Name of the schem
a the object belonged in, if any; otherwise NULL. No quoting is applied. object_identity |
text | | | | extended | | Text rendering of the object identity, schema-qualified. is_exten
sion | boolean | | | | plain | | True if the command is part of an extension script query |
text | | | | extended | | sql text username | text | | | CURRENT_USER | extended | | datnam
e | text | | | current_database() | extended | | client_addr | inet | | | inet_client_addr(
) | main | | crt_time | timestamp without time zone | | | now() | plain | |
```

创建语句如下：

```
CREATE SCHEMA IF NOT EXISTS dts_audit; CREATE TABLE IF NOT EXISTS dts_audit.dts_tb_ddl_comm
and ( event text, tag text, classid oid, objid oid, objsubid int, object_type text, schema_
name text, object_identity text, is_extension bool, query text, username text default curre
nt_user, datname text default current_database(), client_addr inet default inet_client_addr
(), crt_time timestamp default now() );
```

下文将以示例的方式介绍如何实现DDL回收站、DDL防火墙、DDL增量订阅及同步等功能，您可以根据业务情况修改相关代码。

DDL回收站

1. 执行如下命令创建表、函数和相关触发器。

```
/* external/rds_ddl_pulication/rds_ddl_pulication--1.0.sql */ --create schema CREATE SC
HEMA IF NOT EXISTS dts_audit; --create table for ddl record CREATE TABLE IF NOT EXISTS
dts_audit.dts_tb_ddl_command ( event text, tag text, classid oid, objid oid, objsubid i
nt, object_type text, schema_name text, object_identity text, is_extension bool, query
text, username text default current_user, datname text default current_database(), clie
nt_addr inet default inet_client_addr(), crt_time timestamp default now() ); -- create
function for event triggers create or replace function dts_audit.dts_func_ddl_command()
returns event_trigger as $$ declare v1 text; is_superuser bool = false; r record; begin
-- we don't record ddl command from superusers select u.rolsuper into is_superuser from
pg_catalog.pg_roles u where u.rolname = SESSION_USER; if is_superuser then return; end
if; select query into v1 from pg_stat_activity where pid=pg_backend_pid(); -- RAISE NOT
ICE 'ddl event:%, command:%', tg_event, tg_tag; -- NB:since ddl_command_end cannot coll
ect the details of the drop statement, we use sql_drop if TG_EVENT='ddl_command_end' th
en SELECT * into r FROM pg_event_trigger_ddl_commands(); if r.classid > 0 then insert i
nto dts_audit.dts_tb_ddl_command(event, tag, classid, objid, objsubid, object_type, sch
ema_name, object_identity, is_extension, query) values(TG_EVENT, TG_TAG, r.classid, r.o
bjid, r.objsubid, r.object_type, r.schema_name, r.object_identity, r.in_extension, v1);
end if; end if; if TG_EVENT='sql_drop' then -- To avoid repeated collection, we filtere
d 'ALTER TABLE' and 'ALTER FOREIGN TABLE' if TG_TAG != 'ALTER TABLE' and TG_TAG != 'ALT
ER FOREIGN TABLE' then SELECT * into r FROM pg_event_trigger_dropped_objects(); insert
into dts_audit.dts_tb_ddl_command(event, tag, classid, objid, objsubid, object_type, sc
hema_name, object_identity, is_extension, query) values(TG_EVENT, TG_TAG, r.classid, r.
objid, r.objsubid, r.object_type, r.schema_name, r.object_identity, 'f', v1); end if; e
nd if; end; $$ language plpgsql strict; -- ddl_command_end event trigger CREATE EVENT T
RIGGER pg_get_ddl_command on ddl_command_end EXECUTE PROCEDURE dts_audit.dts_func_ddl_c
ommand(); -- pg_get_ddl_drop event trigger CREATE EVENT TRIGGER pg_get_ddl_drop on sql_
drop EXECUTE PROCEDURE dts_audit.dts_func_ddl_command(); -- grant privileges to all use
r GRANT USAGE ON SCHEMA dts_audit TO PUBLIC; GRANT SELECT, INSERT ON TABLE dts_audit.dt
s_tb_ddl_command TO PUBLIC;
```

说明 执行以上命令后，您的DDL语句就会记录在表dts_audit.dts_tb_ddl_command中。

2. 执行一个DDL语句，测试能否记录变更。

```
create table tb_test(id int); select * from dts_audit.dts_tb_ddl_command;
```

DDL防火墙

您可以根据业务需求创建事件触发器，使用ddl_command_start事件类型，可以阻止相应的DDL语句执行。

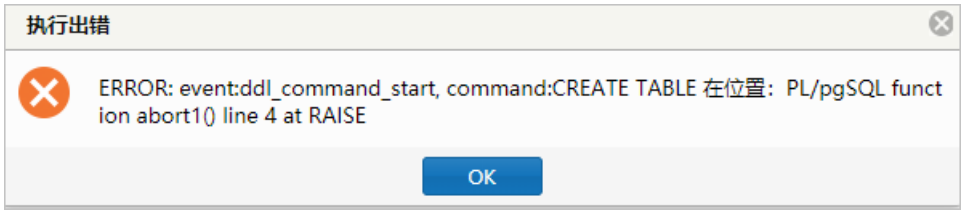
1. 创建触发器函数。

```
CREATE OR REPLACE FUNCTION abort1() RETURNS event_trigger LANGUAGE plpgsql AS $$ BEGIN
if current_user = 'test1' then RAISE EXCEPTION 'event:%, command:%', tg_event, tg_tag;
end if; END; $$;
```

2. 创建触发器阻止创建和删除表的语句。

```
create event trigger b on ddl_command_start when TAG IN ('CREATE TABLE', 'DROP TABLE')
execute procedure abort1();
```

3. 使用对应的用户test1登录实例后创建表，测试能否创建。



说明 DDL语句被成功阻止。

DDL增量订阅同步

在发布端我们将已经执行了的DDL语句存储在dts_audit.dts_tb_ddl_command中。订阅端可以读取记录进行同步。

1. 在发布端执行发布命令。

```
CREATE PUBLICATION my_ddl_publication FOR TABLE ONLY dts_audit.dts_tb_ddl_command;
```

2. 在订阅端创建相同的表。

```
CREATE SCHEMA IF NOT EXISTS dts_audit; CREATE TABLE IF NOT EXISTS dts_audit.dts_tb_ddl_
command ( event text, tag text, classid oid, objid oid, objsubid int, object_type text,
schema_name text, object_identity text, is_extension bool, query text, username text, d
atname text, client_addr inet , crt_time timestamp );
```

3. 在订阅端创建订阅。

```
CREATE SUBSCRIPTION my_ddl_subscriptin CONNECTION 'host=*** port=*** user=*** password=
*** dbname=**' PUBLICATION my_ddl_publication;
```

说明 需要确保实例参数 wal_level的值为 logical，您可以在控制台的参数设置页面进行修改，该参数修改后需要重启实例才能生效。详情请参见 逻辑订阅。

示例

```
CREATE SUBSCRIPTION my_ddl_subscriptin CONNECTION 'host=pgm-bpxxxxx.pg.rds.aliyuncs.com
port=1433 user=test1 password=xxxxx dbname=testdb1' PUBLICATION my_ddl_publication;
```

4. 在订阅端针对dts_audit.dts_tb_ddl_command表创建相应的触发器，实现DDL增量同步。

28.8. 自动故障转移和读写分离

使用PostgreSQL的libpq或JDBC，您可以通过简单的配置实现自动故障转移（failover）和读写分离。

背景信息

从PostgreSQL 10开始，libpq驱动层开始支持简单的故障转移，JDBC驱动层则支持简单的故障转移和负载均衡。

- libpq是PostgreSQL的C应用程序接口，包含一组库函数，允许客户端程序将查询请求发送给PostgreSQL后端服务器并接收这些查询的结果。

- JDBC（Java Database Connectivity）是Java语言中用来规范客户端程序如何访问数据库的应用程序接口，在PostgreSQL中JDBC支持故障转移和负载平衡（Load Balance）。

libpq实现自动故障转移和读写分离

通过libpq函数连接多个数据库，当出现故障时会自动切换到可用的数据库。

命令

```
postgresql://[user[:password]@][netloc][:port][,...][/dbname][?param1=value1&...]
```

示例

如下示例为连接1个RDS PostgreSQL主实例数据库和对应的2个只读实例数据库，只要确保至少有一个数据库可用，读请求就不会失败。

```
postgres://pgm-bpxxx1.pg.rds.aliyuncs.com:3433,pgm-bpxxx2.pg.rds.aliyuncs.com:3433,pgm-bpxx
x3.pg.rds.aliyuncs.com:3433/postgres?target_session_attrs=any
```

target_session_attrs表示允许连接到指定状态的数据库，取值：

- **any**：默认值，表示允许连接到任意数据库，会从所有配置的数据库中随机选择一个尝试连接，如果连接的数据库出现故障导致连接断开，会尝试连接其他数据库，从而实现故障转移。
- **read-write**：只会连接到支持读写的数据库，即从第一个数据库开始尝试连接，如果连接后发现不支持读写，则会断开连接，然后尝试连接第二个数据库，以此类推，直至连接到支持读写的数据库。

更多libpq的使用方法和参数说明请参见Connection Strings。

您可以在应用程序中结合pg_is_in_recovery()函数，判断连接的数据库是主实例数据库的还是只读实例数据库，最终实现读写分离和故障转移，示例如下：

- Python示例

```
$ cat pg_conn.py import psycopg2 conn = psycopg2.connect(database="postgres",host="pgm-bp
xxx1.pg.rds.aliyuncs.com,pgm-bpxxx2.pg.rds.aliyuncs.com,pgm-bpxxx3.pg.rds.aliyuncs.com",
user="testxxx", password="xxxxxx", port="3433", target_session_attrs="read-write") cur =
conn.cursor() cur.execute("select pg_is_in_recovery(), pg_postmaster_start_time()") row =
cur.fetchone() print "recovery =",row[0] print "time =",row[1] $ python pg_conn.py recove
ry = False time = 2020-07-09 15:33:57.79001+08
```

- PHP示例

```
# cat pg_conn.php <?php $conn = pg_connect("host=pgm-bpxxx1.pg.rds.aliyuncs.com,pgm-bpxxx
2.pg.rds.aliyuncs.com,pgm-bpxxx3.pg.rds.aliyuncs.com port=3433 dbname=postgres user=testx
xx password=xxxxxx target_session_attrs=read-write") or die("Could not connect"); $status
= pg_connection_status($conn); if ($status === PGSQL_CONNECTION_OK) { print "Connection s
tatus ok\n"; } else { print "Connection status bad\n"; } $sql = pg_query($conn, "select p
g_is_in_recovery()"); while ($row = pg_fetch_row($sql)) { echo "Recovery-status: $row[0]\
n"; } ?> $ php -f pg_conn.php Connection status ok Recovery-status: f Server: xxx.xxx.xx.
xx
```

JDBC实现读写分离和自动故障转移

您可以在连接URL中定义多个数据库，并用逗号分隔，驱动程序将尝试按顺序连接到它们中的每一个，直到连接成功。如果没有成功，会返回连接异常报错。

命令

```
jdbc:postgresql://node1,node2,node3/accounting?targetServerType=preferSlave&loadBalanceHost
s=true
```

示例

```
jdbc:postgresql://pgm-bpxxx1.pg.rds.aliyuncs.com:3433,pgm-bpxxx2.pg.rds.aliyuncs.com:3433,p
gm-bpxxx3.pg.rds.aliyuncs.com:3433/accounting?targetServerType=preferSlave&loadBalanceHosts
=true
```

参数说明如下：

- **targetServerType**表示允许连接到指定状态的数据库，取值：
  - any：任何数据库。
  - master：主数据库。
  - slave：从数据库。
  - preferSlave：优先从数据库，如果没有从数据库才连接到主数据库。

> **说明** 区别数据库主从的方式是通过查询数据库是否允许写入，允许写入的判断为主数据库，不允许写入的判断为从数据库。

- **loadBalanceHosts**表示尝试连接数据库的顺序，取值：
  - False：默认值，按命令内顺序连接数据库。
  - True：随机连接数据库。

为实现读写分离，您需要在配置JDBC时配置2个数据源，1个设置**targetServerType=master**，一个设置**targetServerType=preferSlave**。需要写操作时，指定master的数据源，需要读操作时，指定preferSlave的数据源。如果需要判断数据源类型，您可以结合pg_is_in_recovery()函数，判断连接的数据库是主实例数据库的还是只读实例数据库，最终实现读写分离和故障转移。

29.应用解决方案

29.1. 实时精准营销（人群圈选）

您可以通过RDS PostgreSQL 12实现实时精准营销，快速圈选目标用户。

前提条件

- 创建RDS PostgreSQL 12实例。
- 设置白名单。
- 创建账号。
- 创建数据库。

背景信息

几乎所有行业（如互联网、游戏、教育等）都有实时精准营销的需求。通过系统生成用户画像，在营销时通过条件组合筛选用户，快速提取目标群体，例如：

- 电商行业中，商家在运营活动前，需要根据活动的目标群体的特征，圈选出一批目标用户进行广告推送或进行活动条件的判断。
- 游戏行业中，商家需要根据玩家的某些特征进行圈选，针对性地发放大礼包，提高玩家活跃度。
- 教育行业中，需要根据学生不同的特征，推送有针对性的习题，帮助学生查缺补漏。
- 搜索、门户、视频网站等业务中，根据用户的关注热点，推送不同的内容。

这些行业都有一些共同的业务痛点：

- 数据量庞大，运算量也极大。
- 用户标签多、字段多，占用存储空间也多。
- 字段多，可能会超过数据库的字段数限制，一般数据库最多支持1000多个字段。
- 使用数组替代多字段存储标签，需要数据库支持倒排索引，但并不是所有数据库都支持倒排索引。
- 使用数组替代多字段存储标签，加上倒排索引，存储空间会暴增。
- 圈选条件组合多样化，没有固定索引可以优化，如果每个字段一个索引，存储空间又会暴增。
- 性能要求高，因为实时营销要求秒级响应。
- 数据更新时效要求高，用户画像要求近实时的更新，否则可能圈选到不精准用户（例如用户昨天在浏览手机，昨晚已下单，但是数据未实时更新，那么手机卖家圈选时这个用户就会被选中，但是该用户实际上已经不是目标群体）。

常见的产品（如MySQL）无法在有限的资源下满足实时圈选目标群体的需求。

本文介绍如何基于RDS PostgreSQL实现实时精准营销，您可以根据业务需要使用三种方案之一。

- 方案一
- 方案二
- 方案三

方案一

> **说明** PostgreSQL和MySQL都支持方案一。

表结构如下：

```
KEY: 用户ID 标签1: 标签2: ... 标签N:
```

索引如下：

```
每个标签字段一个索引
```

搜索方法如下：

```
and , or , not 组合 where 标签a and 标签b and ...
```

方案缺陷

- 用户标签多、字段多，占用存储空间也多。
- 字段多，可能会超过数据库的字段数限制，一般数据库最多支持1000多个字段。也可以设置为多对多的结构，每个标签一条记录，解决字段数限制的问题。
- 圈选条件组合多样化，没有固定索引可以优化，如果每个字段一个索引，存储空间又会暴增。
- 新增一个用户群体标签时，需要更新大量数据。
- 查询性能差。

操作示例

1. 创建人群表，每条记录代表一个人群。示例如下：

```
create table t_tag_dict ( tag int primary key, -- 标签(人群)id info text, -- 人群描述 crt
_time timestamp -- 时间 );
```

2. 生成10万个人群标签。示例如下：

```
insert into t_tag_dict values (1, '男', now()); insert into t_tag_dict values (2, '女',
now()); insert into t_tag_dict values (3, '大于24岁', now()); -- ... insert into t_tag_d
ict select generate_series(4,100000), md5(random()::text), clock_timestamp();
```

3. 创建用户画像表，每个用户N条记录，每条记录代表这个用户贴的某个标签。示例如下：

```
create table t_user_tag ( uid int8, -- 用户id tag int, -- 用户对应标签（人群） mod_time ti
mestamp, -- 时间 primary key (tag,uid) );
```

4. 给2000万个用户打标，每个用户64个随机标签，其中男、女各一半，共12.8亿条记录。示例如下：

```
create or replace function gen_rand_tag(int,int) returns setof int as $$ select case wh
en random() > 0.5 then 1::int else 2::int end as tag union all select ceil(random()*$1)
::int as tag from generate_series(1,$2); $$ language sql strict volatile; insert into t
_user_tag select uid, gen_rand_tag(100000,63) as tag, clock_timestamp() from generate_s
eries(1,20000000) as uid on conflict (uid,tag) do nothing; -- 或使用如下方法加速导入 creat
e sequence seq; vi test.sql insert into t_user_tag select uid, gen_rand_tag(100000,63)
as tag, clock_timestamp() from nextval('seq'::regclass) as uid on conflict(tag,uid) do
nothing; pgbench -M prepared -n -r -P 1 -f ./test.sql -c 50 -j 50 -t 400000
```

5. 查询包含标签1和3的人群。示例如下：

```
1、人群数量 select count(*) from ( select uid from t_user_tag where tag=1 intersect sele
ct uid from t_user_tag where tag=3 ) t; -- Time: 1494.789 ms (00:01.495) 2、提取人群ID s
elect uid from t_user_tag where tag=1 intersect select uid from t_user_tag where tag=3;
-- Time: 3246.184 ms (00:03.246)
```

6. 查询包含标签1、3、10或200的人群。示例如下：

```
1、人群数量 select count(*) from ( select uid from t_user_tag where tag=1 union select u
id from t_user_tag where tag=3 union select uid from t_user_tag where tag=10 union sele
ct uid from t_user_tag where tag=200 ) t; -- Time: 3577.714 ms (00:03.578) 2、提取人群ID
select uid from t_user_tag where tag=1 union select uid from t_user_tag where tag=3 uni
on select uid from t_user_tag where tag=10 union select uid from t_user_tag where tag=2
00; -- Time: 5682.458 ms (00:05.682)
```

方案二

> **说明** 仅PostgreSQL支持方案二。MySQL不支持数组类型、倒排索引。

表结构如下：

```
KEY：用户ID VALUES：标签数组
```

索引如下：

```
标签数组字段：GIN倒排索引
```

搜索方法如下：

```
与、或、非 where VALUES @> array[标签s] -- 与 where VALUES && array[标签s] -- 或 where not VAL
UES @> array[标签s] -- 非
```

方案缺陷

- 使用数组替代多字段存储标签，需要数据库支持倒排索引，但并不是所有数据库都支持倒排索引。
- 使用数组替代多字段存储标签，加上倒排索引，存储空间会暴增。
- 新增一个用户群体标签时，需要更新大量数据。

操作示例

1. 创建人群表，每条记录代表一个人群。示例如下：

```
create table t_tag_dict ( tag int primary key, -- 标签(人群)id info text, -- 人群描述 crt
_time timestamp -- 时间 );
```

2. 生成10万个人群标签。示例如下：

```
insert into t_tag_dict values (1, '男', now()); insert into t_tag_dict values (2, '女',
now()); insert into t_tag_dict values (3, '大于24岁', now()); -- ... insert into t_tag_d
ict select generate_series(4,100000), md5(random()::text), clock_timestamp();
```

3. 创建用户画像表，每个用户一条记录，用数组表示这个用户归属哪些标签。示例如下：

```
create table t_user_tags ( uid int8 primary key, -- 用户id tags int[], -- 用户标签（人群）数组 mod_time timestamp -- 时间 );
```

4. 创建随机生成打标数组的函数。示例如下：

```
create or replace function gen_rand_tags(int,int) returns int[] as $$ select array_agg(ceil(random()*$1)::int) from generate_series(1,$2); $$ language sql strict;
```

5. 在10万个标签内随机提取8个标签。示例如下：

```
select gen_rand_tags(100000, 8); gen_rand_tags ----------------------------------------------------------- {43494,46038,74102,25308,99129,40893,33653,29690} (1 row)
```

6. 给2000万个用户打标，每个用户64个随机标签，其中男、女各一半。示例如下：

```
insert into t_user_tags select generate_series(1,10000000), array_append(gen_rand_tags(100000, 63),1), now(); insert into t_user_tags select generate_series(10000001,20000000), array_append(gen_rand_tags(100000, 63),2), now();
```

7. 创建人群标签字段倒排索引。示例如下：

```
create index idx_t_user_tags_1 on t_user_tags using gin (tags);
```

8. 查询包含标签1和3的人群。示例如下：

```
1、人群数量 select count(uid) from t_user_tags where tags @> array[1,3]; 2、提取人群ID select uid from t_user_tags where tags @> array[1,3];
```

9. 查询包含标签1、3、10或200的人群。示例如下：

```
1、人群数量 select count(uid) from t_user_tags where tags && array[1,3,10,200]; 2、提取人群ID select uid from t_user_tags where tags && array[1,3,10,200];
```

方案三

> **说明** 仅PostgreSQL支持方案三。MySQL不支持位图功能。

方案三通过roaringbitmap插件实现快速查询，相关说明请参见位图计算（roaringbitmap）。

表结构如下：

```
KEY：标签ID VALUES：用户bitmap
```

索引如下：

```
标签ID字段：Btree索引
```

搜索方法如下：

```
聚合bitmap：与、或、非 and_agg(bitmaps) where KEY in (标签s) -- 与 or_agg(bitmaps) where KEY in (标签s) -- 或 except(bitmap1,bitmap2) -- 非
```

- 方案优势
  - 表存储占用空间小。

- 索引存储占用空间小。仅需一个Btree索引，索引记录数少（有多少标签，就有多少条记录，通常标签数在百万以内）。
- 新增一个用户群体标签时，不需要更新大量数据，仅需新增一条人群的bitmap记录。
- 查询性能极好。

- 方案缺陷
  - bitmap最大长度为1GB，用户数超过长度时需要使用offset，方法如下：

    ```
    offset0_bitmap, offset1gb_bitmap, ...
    ```

  - 用户ID需要是数字（建议连续数值），如果没有数值型UID，需要建立映射表。

操作示例

> **说明**
> - 当UID超过40亿（int4）时，使用offset转换。转换方法请参见UID溢出时的处理方法。
> - 更多roaringbitmap的使用方法请参见pg_roaringbitmap。

1. 安装插件。示例如下：

   ```
   create extension roaringbitmap;
   ```

2. 创建用户标签bitmap表。示例如下：

   ```
   create table t_tag_users ( tagid int primary key, -- 用户标签（人群）id uid_offset int, -
   - 由于userid是int8类型，roaringbitmap内部使用int4存储，需要转换一下。 userbits roaringbitmap
   , -- 用户ID聚合的bitmap mod_time timestamp -- 时间 );
   ```

3. 生成UID标签bitmap表。示例如下：

   ```
   insert into t_tag_users select tagid, uid_offset, rb_build_agg(uid::int) as userbits fr
   om ( select unnest(tags) as tagid, (uid / (2^31)::int8) as uid_offset, mod(uid, (2^31):
   :int8) as uid from t_user_tags ) t group by tagid, uid_offset;
   ```

4. 查询包含标签1和3的人群。示例如下：

   ```
   1、人群数量 select sum(ub) from ( select uid_offset,rb_and_cardinality_agg(userbits) as
   ub from t_tag_users where tagid in (1,3) group by uid_offset ) t; 2、提取人群ID select u
   id_offset,rb_and_agg(userbits) as ub from t_tag_users where tagid in (1,3) group by uid
   _offset;
   ```

5. 查询包含标签1、3、10或200的人群。示例如下：

   ```
   1、人群数量 select sum(ub) from ( select uid_offset,rb_or_cardinality_agg(userbits) as u
   b from t_tag_users where tagid in (1,3,10,200) group by uid_offset ) t; 2、提取人群ID se
   lect uid_offset,rb_or_agg(userbits) as ub from t_tag_users where tagid in (1,3,10,200)
   group by uid_offset;
   ```

方案对比

对比项	方案一（MySQL/PostgreSQL）	方案二（PostgreSQL）	方案三（PostgreSQL）	方案三相比方案一的提升
与查询圈选用户速度	1.5秒	0.042秒	0.0015秒	99900%
或查询圈选用户速度	3.6秒	3秒	0.0017秒	211665%
空间占用（表）	63,488 MB	3,126 MB	1390MB	4467%
空间占用（索引）	62,464 MB	3139 MB	2MB	3123100%
创建索引速度	-	20分钟	极快（约为0秒）	-

说明 方案使用的MySQL 8.0和PostgreSQL 12实例规格均为8核CPU、32GB内存、1500GB ESSD云盘。

总结

RDS PostgreSQL 12支持位图功能（roaringbitmap插件），可以高效生成、压缩、解析位图数据，支持最常见的位图聚合操作（与、或、非、异或），满足您在亿级以上用户、千万级标签的大数据量下实时精准营销、快速圈选用户的需求。

相比MySQL的方案，RDS PostgreSQL方案优势非常明显，是一个低成本、高效率的解决方案。

29.2. 图像识别、人脸识别、相似特征检索、相似人群圈选

RDS PostgreSQL使用PASE插件，可以实现低成本、高效率的图像识别、人脸识别、相似特征检索、相似人群圈选。

前提条件

实例为RDS PostgreSQL 11。

说明 未来将在PostgreSQL 11以上的所有版本支持该功能。

背景信息

阿里云RDS PostgreSQL提供PASE插件，支持业界流行的两种向量索引算法ivfflat和hnsw，基于该插件的加速方案优势非常明显，是一个低成本、高效率的图像识别、人脸识别、相似特征检索、相似人群圈选解决方案，广泛应用在互联网、新零售、交通等行业。RDS PostgreSQL未来将持续集成业界优秀的向量索引算法。

说明 PASE插件的使用说明请参见 高效向量检索（PASE）。

方案对比

对比项	方案一（MySQL）	方案二（PostgreSQL）	方案二相比方案一的提升
单次查询响应速度	61.45秒	0.0025秒	2457900%
并发查询QPS	0.055330	1056	1908449%

说明 方案使用的MySQL 8.0和PostgreSQL 11实例规格均为4核CPU、8GB内存、1500GB ESSD云盘，图片数量为100万。

业务挑战

通用关系型数据库（如MySQL）不支持向量检索，需要遍历查询并全部返回到应用端进行计算，性能差，并且需要耗费大量网络带宽。即使数据库支持了向量检索的操作符，但是并不支持向量索引，依然需要遍历计算，无法支持高并发查询。

当图像向量计算上移到应用层实现时，需要从数据库加载所有数据，加载速度慢，而且图像更新后无法实时加载，同时不支持图像识别及其他属性检索的联合过滤，效率低下。

应用场景

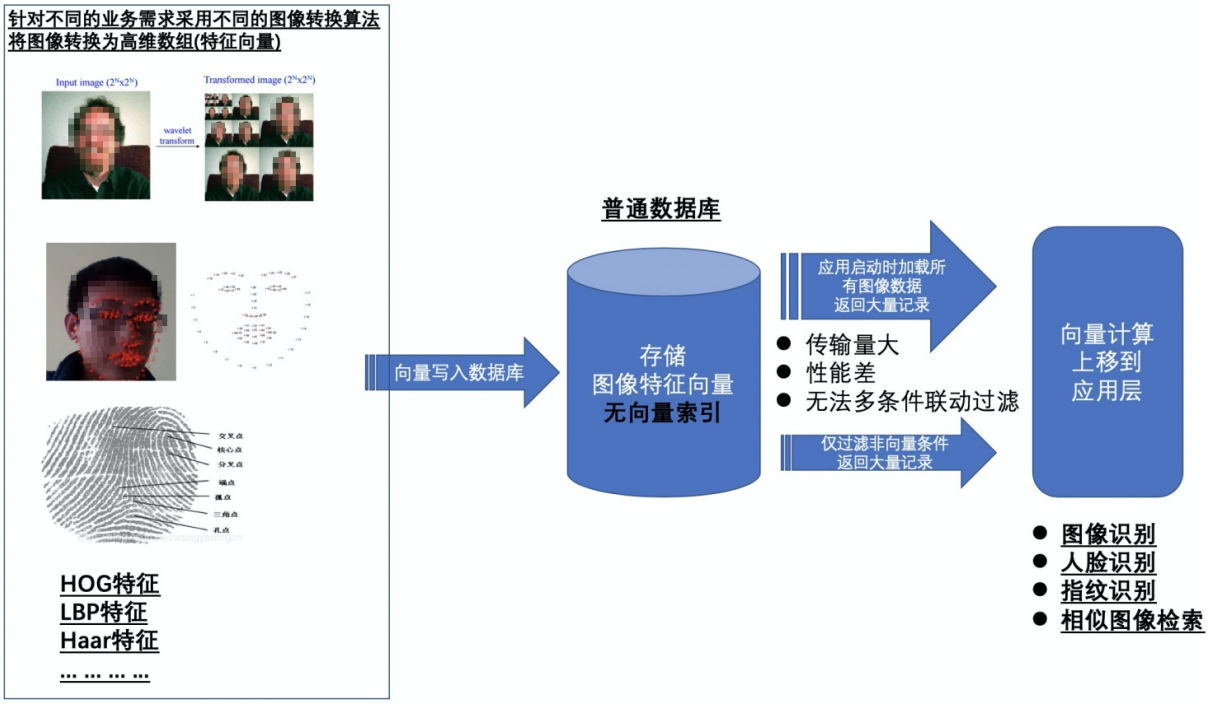
当需要高效率、高精度的以图搜图，同时还有其他条件的过滤需求时，使用RDS PostgreSQL能快速处理查询请求，例如：

- 智能楼宇中，根据人脸识别，识别员工为本大楼的员工时，自动进行打卡，自动按工作楼层设定电梯。
- 智能酒店中，根据人脸识别，办理自动入住，根据会员等级自动对接专属服务等。
- 电商行业中，根据图像识别、搜索类似商品。
- 教育行业中，根据人脸特征，记录学生的听课状态（打盹、走神、小动作、举手等）。
- 交通行业中，自动识别违规驾驶员。
- 新零售行业中，根据人脸识别会员，实现到店提醒、导购引导、定制化运营等。
- 公共交通中，实现刷脸支付。
- 游戏行业中，虚拟现实相关游戏。

准备工作

- 创建RDS PostgreSQL 11实例。
- 设置白名单。
- 创建账号。
- 创建数据库。

技术方案一



- 方案特点
  - 数据库仅存储图像向量，不进行向量计算。
  - 图像向量计算上移到应用层实现。
- 方案缺陷
  - 普通数据库不支持向量索引，无法在数据库中完成向量过滤。
  - 应用需要从数据库加载所有数据，加载速度慢，而且图像更新后无法被应用实时加载。
  - 无法在数据库中实现图像识别及其他属性检索的联合过滤，需要在业务层过滤图像条件，网络传输的记录多、效率低，无法支持高并发场景。

操作示例

1. 创建测试表。示例如下：

```
create table if not exists t_vec_80( id serial PRIMARY KEY, -- 主键 c1 int, -- 其他属性字段 c2 int, c3 text, c4 timestamp, vec float4[] -- 图像特征值向量 );
```

2. 创建生成随机向量的函数（用于模拟图像特征值，实际场景请使用实际图片特征值存入）。示例如下：

```
create or replace function gen_float4_arr(int,int) returns float4[] as $$ select array_agg(trunc(random()*$1)::float4) from generate_series(1,$2); $$ language sql strict volatile;
```

3. 写入100万随机向量。示例如下：

```
insert into t_vec_80 (c1,c2,c3,c4,vec) select random()*100, random()*100000, md5(random()::text), clock_timestamp(), gen_float4_arr(10000, 80) from generate_series(1,1000000);
```

说明 查看结果样例，示例如下：

```
select * from t_vec_80 limit 3; -[ RECORD 1 ]-------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------- id | 1 c1 | 99 c2 | 7428 c3 | 9b74e40ab38ed4f41b5d50b8eedf8b72
c4 | 2020-02-27 15:36:56.895773 vec | {6469,3787,5852,1642,2798,7728,1527,6990,7399
,3460,7802,7682,8102,6499,3428,7687,567,8894,8144,1685,6139,9549,3613,1714,721,9099
,4218,1930,9031,4961,3966,5501,8748,9818,7143,1546,7547,8671,8536,4946,2132,6338,26
29,234,2838,6057,7922,3405,4951,6066,5091,1091,5615,8704,2805,6336,7804,7024,8266,6
836,1985,2233,2337,733,2051,9481,2280,9598,8152,816,4545,285,7155,7174,519,9993,323
2,8441,3399,8183} -[ RECORD 2 ]----------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------ id | 2 c1 | 45 c2 | 84908 c3 | a48d421b772486121ef520eb3e285f95 c4 | 2020-02
-27 15:36:56.896329 vec | {123,7195,2080,6460,5000,9104,4727,1836,1089,6960,4174,18
23,9012,3656,4103,8611,1808,4920,3157,2094,2076,332,2613,2070,3564,1055,5469,1748,5
563,3960,1023,5686,1156,3103,2147,6156,2208,6874,7993,3298,3834,2167,5121,2847,5823
,9225,1458,7632,4145,4615,9726,6222,4947,2340,8292,8511,3395,3762,259,8958,7722,128
2,4644,8878,4386,6792,5035,6594,3666,3028,9892,7501,5196,5014,348,1019,4239,1806,86
52,8384} -[ RECORD 3 ]-------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------------------------- id
| 3 c1 | 64 c2 | 83785 c3 | ea856c452399648fd29b0e0383a169a5 c4 | 2020-02-27 15:36:
56.896395 vec | {1369,718,2899,9880,4113,6661,140,3071,4383,1422,7716,3262,5808,450
9,8298,2403,8175,1326,2295,5676,6523,7309,6024,7542,1549,7831,6194,9934,4253,4573,4
541,5622,5291,7440,5503,9405,4101,5643,2477,8485,7066,194,1748,2875,4703,46,5278,28
78,1373,7574,8555,7896,4884,4580,5439,6433,2411,1633,6367,6664,6207,909,2286,1498,8
349,7789,903,2451,3433,3381,936,499,3575,2685,3374,8278,2731,8653,1157,4105}
```

4. 查询出100万条记录返回给客户端。示例如下：

```
time psql -h xxx.xxx.xxx.xxx -p 3433 -U digoal postgres -c "select * from t_vec_80" >/d
ev/null
```

说明 测试结果如下：

```
real 1m1.450s --单次查询响应速度
```

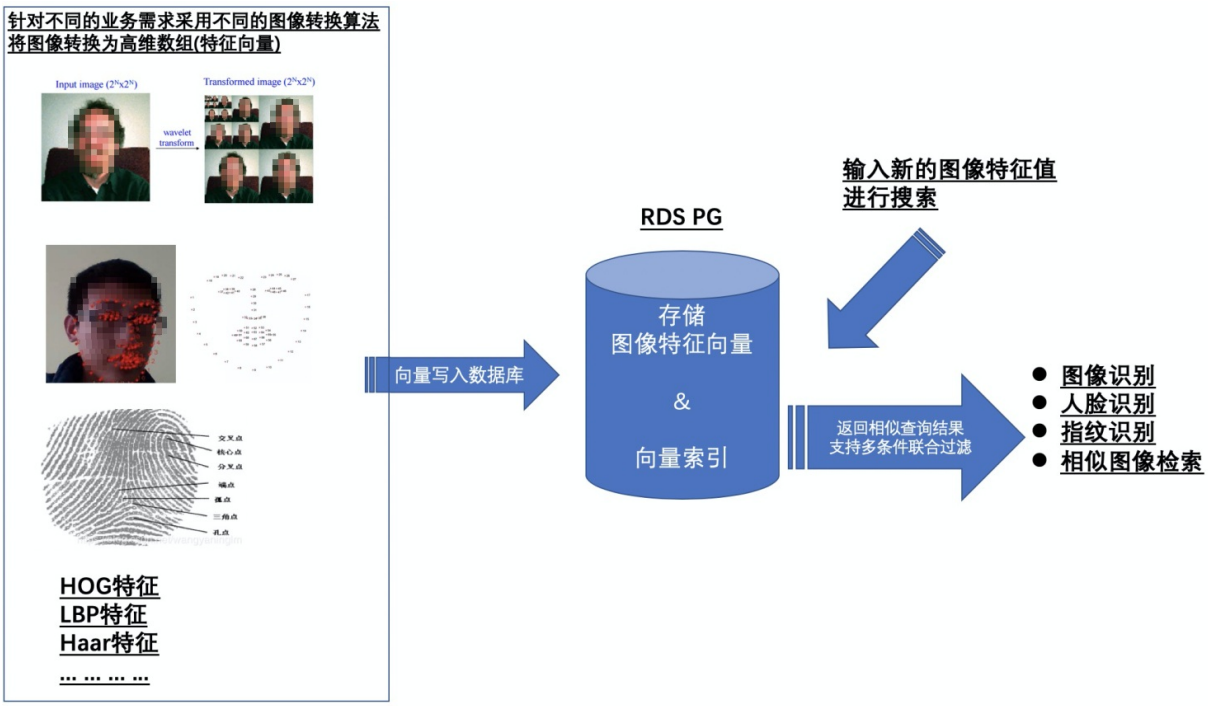
5. 并发能力测试。示例如下：

```
vi test.sql select * from t_vec_80; pgbench -M prepared -n -r -f ./test.sql -c 4 -j 4 -
T 600 -h xxx.xxx.xxx.xxx -p 3433 -U digoal postgres
```

说明　测试结果如下：

```
transaction type: ./test.sql scaling factor: 1 query mode: prepared number of clien
ts: 4 number of threads: 4 duration: 600 s number of transactions actually processe
d: 36 latency average = 72293.794 ms tps = 0.055330 (including connections establis
hing) tps = 0.055330 (excluding connections establishing) statement latencies in mi
lliseconds: 72204.857 select * from t_vec_80;
```

技术方案二



- 方案特点
  - 使用RDS PostgreSQL存储图像向量特征值。
  - 使用RDS PostgreSQL的PASE插件，创建图像特征向量的向量索引。
  - 应用输入特征向量，在数据库中通过向量索引快速搜索到与之相似的图像，支持返回向量距离，以及按向量距离进行排序。
  - 当有多个过滤条件时，数据库可以使用多个索引对多个条件进行合并过。滤
- 方案优势
  - RDS PostgreSQL数据库支持向量索引，图像搜索可以直接在数据库中高效率过滤。
  - RDS PostgreSQL支持索引合并过滤，可以同时过滤图像条件和其他属性条件，通过索引可以最大化收敛条件结果集，大幅度提升速度，并降低传输数据量，单次查询可以毫秒级完成。
  - 通过RDS PostgreSQL只读实例，可以再次提高整体查询吞吐。

操作示例

1. 创建PASE向量索引插件。命令如下：

```
create extension pase;
```

2. 创建测试表。示例如下：

```
create table if not exists t_vec_80( id serial PRIMARY KEY, -- 主键 c1 int, -- 其他属性字段 c2 int, c3 text, c4 timestamp, vec float4[] -- 图像特征值向量 );
```

3. 创建生成随机向量的函数（用于模拟图像特征值，实际场景请使用实际图片特征值存入）。示例如下：

```
-- 创建生成随机向量的函数 create or replace function gen_float4_arr1(int,int) returns floa
t4[] as $$ select array_agg(trunc(random()*$1)::float4) from generate_series(1,$2); $$
language sql strict volatile;
```

```
-- 创建基于数组生成随机附近数组的函数 create or replace function gen_float4_arr(float4[], i
nt) returns float4[] as $$ select array_agg( (u + (u*$2/2.0/100) - u*$2/100*random())::
float4 ) from unnest($1) u; $$ language sql strict volatile;
```

4. 写入100万随机向量。示例如下：

```
do language plpgsql $$ declare v_cent float4[]; begin for i in 1..100 loop -- 100个中心
点 v_cent := gen_float4_arr1(10000,80); -- 取值范围10000, 80个维度 insert into t_vec_80 (
vec) select gen_float4_arr(v_cent, 20) from generate_series(1,10000); -- 1万个点围绕一个
中心点，每个维度的值随机加减20% end loop; end; $$;
```

5. 创建向量索引（使用hnsw算法索引，目前PASE插件支持两种索引ivfflat和hnsw）。示例如下：

> **说明** 实际使用时一定要参考 高效向量检索（PASE），索引参数需要设置正确，特别是维度需要和实际维度一致。

```
CREATE INDEX idx_t_vec_80_1 ON t_vec_80 USING pase_hnsw(vec) WITH (dim = 80, base_nb_nu
m = 16, ef_build = 40, ef_search = 200, base64_encoded = 0);
```

> **说明** 创建索引耗时1282997.955 ms，索引占用空间8138MB。索引创建完成后，未来更新或新增图像特征值时，会自动更新索引，不需要再创建索引。

6. 基于一个随机输入向量查询出5条与之最相似的向量，并按向量距离顺序返回。示例如下：

```
explain select id as v_id, vec <?> '7533.44,3740.27,670.119,994.914,3619.27,2018.17,204
1.34,5483.19,6370.07,4745.54,8762.81,1117.59,8254.75,2009.3,6512.47,3876.7,4775.02,384.
683,2003.78,7926.78,9101.46,6801.24,5397.1,7704.49,7546.87,9129.23,9517.36,5723.4,877.6
49,3117.72,6739.25,8950.36,6397.09,6687.46,9606.15,557.142,9742.48,1714.25,6682.97,5369
.21,6178.99,4983.06,7064.29,5433.98,7120.7,2980.34,8485.47,1651.98,3656.9,1126.65,10260
.1,2139.89,9041.79,4988.89,17.2254,5482.88,3428.6,10370.7,1749.32,4761.6,2806.65,8040.8
9,3176.31,9491.93,4355.37,2898.47,282.75,3233.86,4248.86,7012.86,9238.95,524.011,2285.7
5,5363.21,5558.95,10768.8,8689.83,4907.53,1372.65,1982.05:40:0'::pase as v_dist, vec as
v_vec from t_vec_80 order by vec <?> '7533.44,3740.27,670.119,994.914,3619.27,2018.17,2
041.34,5483.19,6370.07,4745.54,8762.81,1117.59,8254.75,2009.3,6512.47,3876.7,4775.02,38
4.683,2003.78,7926.78,9101.46,6801.24,5397.1,7704.49,7546.87,9129.23,9517.36,5723.4,877
.649,3117.72,6739.25,8950.36,6397.09,6687.46,9606.15,557.142,9742.48,1714.25,6682.97,53
69.21,6178.99,4983.06,7064.29,5433.98,7120.7,2980.34,8485.47,1651.98,3656.9,1126.65,102
60.1,2139.89,9041.79,4988.89,17.2254,5482.88,3428.6,10370.7,1749.32,4761.6,2806.65,8040
.89,3176.31,9491.93,4355.37,2898.47,282.75,3233.86,4248.86,7012.86,9238.95,524.011,2285
.75,5363.21,5558.95,10768.8,8689.83,4907.53,1372.65,1982.05:40:0'::pase limit 5; QUERY
PLAN ----------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
------------------------- ------------------------------------------------------------
----------------------------------------------------------------------------------------
```

```
---------------------------------------------- ----------------------------------------
---------------------------------------------------------------------------------------
------------------------------------------------------------------- -------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
- -------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------- ----------------------------------------------------------------
---------------------------------------------------------------------------------------
------------------------------------------- -------------------------------------------
---------------------------------------------------------------------------------------
------------------ Limit (cost=0.00..7.47 rows=5 width=352) -> Index Scan using idx_t_v
ec_80_1 on t_vec_80 (cost=0.00..1493015.50 rows=1000000 width=352) Order By: (vec <?> '
7533.43994140625,3740.27001953125,670.119018554688,994.914001464844,3619.27001953125,20
18.17004394531,2041.33996582031,5483.18994140625,6370.06982421875,4745.5400390 625,8762
.8095703125,1117.58996582031,8254.75,2009.30004882812,6512.47021484375,3876.69995117188
,4775.02001953125,384.683013916016,2003.78002929688,7926.77978515625,9101.4599609375,68
01.240234375 ,5397.10009765625,7704.490234375,7546.8701171875,9129.23046875,9517.360351
5625,5723.39990234375,877.648986816406,3117.71997070312,6739.25,8950.3603515625,6397.08
984375,6687.4599609375,9606.15039 0625,557.142028808594,9742.48046875,1714.25,6682.9702
1484375,5369.2099609375,6178.990234375,4983.06005859375,7064.2900390625,5433.9799804687
5,7120.7001953125,2980.34008789062,8485.4697265625,165 1.97998046875,3656.89990234375,1
126.65002441406,10260.099609375,2139.88989257812,9041.7900390625,4988.89013671875,17.22
54009246826,5482.8798828125,3428.60009765625,10370.7001953125,1749.31994628 906,4761.60
009765625,2806.64990234375,8040.89013671875,3176.31005859375,9491.9296875,4355.37011718
75,2898.46997070312,282.75,3233.86010742188,4248.85986328125,7012.85986328125,9238.9501
953125,52 4.010986328125,2285.75,5363.2099609375,5558.9501953125,10768.7998046875,8689.
830078125,4907.52978515625,1372.65002441406,1982.05004882812::'::pase) (3 rows) select
id as v_id, vec <?> '7533.44,3740.27,670.119,994.914,3619.27,2018.17,2041.34,5483.19,63
70.07,4745.54,8762.81,1117.59,8254.75,2009.3,6512.47,3876.7,4775.02,384.683,2003.78,792
6.78,9101.46,6801.24,5397.1,7704.49,7546.87,9129.23,9517.36,5723.4,877.649,3117.72,6739
.25,8950.36,6397.09,6687.46,9606.15,557.142,9742.48,1714.25,6682.97,5369.21,6178.99,498
3.06,7064.29,5433.98,7120.7,2980.34,8485.47,1651.98,3656.9,1126.65,10260.1,2139.89,9041
.79,4988.89,17.2254,5482.88,3428.6,10370.7,1749.32,4761.6,2806.65,8040.89,3176.31,9491.
93,4355.37,2898.47,282.75,3233.86,4248.86,7012.86,9238.95,524.011,2285.75,5363.21,5558.
95,10768.8,8689.83,4907.53,1372.65,1982.05:40:0'::pase as v_dist, vec as v_vec from t_v
ec_80 order by vec <?> '7533.44,3740.27,670.119,994.914,3619.27,2018.17,2041.34,5483.19
,6370.07,4745.54,8762.81,1117.59,8254.75,2009.3,6512.47,3876.7,4775.02,384.683,2003.78,
7926.78,9101.46,6801.24,5397.1,7704.49,7546.87,9129.23,9517.36,5723.4,877.649,3117.72,6
739.25,8950.36,6397.09,6687.46,9606.15,557.142,9742.48,1714.25,6682.97,5369.21,6178.99,
4983.06,7064.29,5433.98,7120.7,2980.34,8485.47,1651.98,3656.9,1126.65,10260.1,2139.89,9
041.79,4988.89,17.2254,5482.88,3428.6,10370.7,1749.32,4761.6,2806.65,8040.89,3176.31,94
91.93,4355.37,2898.47,282.75,3233.86,4248.86,7012.86,9238.95,524.011,2285.75,5363.21,55
58.95,10768.8,8689.83,4907.53,1372.65,1982.05:40:0'::pase limit 5; v_id | v_dist | v_ve
c ---------+-------------+-------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------- ----------------------------------------------------------------
---------------------------------------------------------------------------------------
------------------------------------------- -------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------- ----------------------
---------------------------------------------------------- 1000001 | 613508 | {7508.56,
3828.8,670.162,978.82,3654.93,2052.38,2023.41,5518.4,6478.1,4814.47,8689.2,1130.5,8421.
43,2018.39,6534.18,3884.82,4737.2,385.625,2025.83,7917.54,8892.97,6900.71 ,5421.61,7579
```

```
.82,7649.6,9337.72,9530.01,5818.69,873.353,3105.67,6622.92,9102.99,6360.46,6737.99,9374
.82,545.683,9734.36,1699.74,6753.08,5320.49,6062.47,4870.6,6907.26,5304.41,7166.67,2997
.09,850 8.14,1691.62,3595.89,1113.89,10232.1,2107.41,8846.84,4875.69,17.1081,5574.26,35
13.31,10576.6,1763.01,4734.1,2780.48,8165.04,3132.32,9586.17,4345.39,2859.25,286.716,33
06.16,4260.56,7007.33,9126.8 1,528.518,2288.32,5310,5610,10584,8872.31,4843.43,1347.01,
1940.52} 1003628 | 8.55116e+06 | {7532.93,3345.53,694.608,984.268,3507.72,1950.43,2188.
66,6043.55,6832.57,4384.97,8975.91,1290.02,8519.66,2237.75,6985.71,3890.79,4199.22,410.
386,2294.93,7938.11,8989.48, 6374.35,5742.55,7811.5,7492.1,9067.4,9843.13,5885.26,940.3
65,3435.39,6545.54,8069.38,6126.34,6906.32,10175.4,505.915,9504.69,1630.76,6832.68,5477
.68,6446.75,5109.62,6686.55,5688.48,6778.92,3100. 2,9182.86,1733.95,3933.06,1116.63,104
88.3,2346.63,8257.46,5312.34,16.0066,5078.85,3717.24,10262.9,1624.57,4406.59,2983.23,74
05.85,3159.04,9924.56,4947.86,2573.72,276.545,3673.99,4487.34,6820.15, 8524.12,486.187,
2328.58,4769.64,5541.63,10255.7,8280.42,5141.37,1332.7,1989.67} 1004945 | 9.21305e+06 |
{7348.72,3833.3,706.311,985.864,3632.96,2153.75,2172.06,6427.87,6502.42,4678.54,8955.37
,1207.76,8594.73,1958.02,6839.83,3703.57,4091.18,367.272,1970.81,7266.62,9198.17,6 869.
98,5960.79,7658.46,7180.35,9386.35,10320.3,6593.09,900.23,3330.1,6749.94,9182.85,6839.2
5,7254.11,9533.32,580.504,9069.41,1841.88,6840.14,4948.41,6390.41,5102.95,6873.49,5683.
65,7283.23,3124. 15,8727.17,1810.11,3575.12,1111.99,10081.7,2174.01,8797.29,5301.64,17.
779,5196.54,3848.29,9813.85,1514.4,4357.8,2752.47,7138.15,2905.04,10178.2,5025.82,2713.
42,267.272,3557.03,4388.08,6581.85,91 14.22,470.335,2249.53,5274.76,5353.28,10566.6,915
3.67,4746.68,1439.05,1996.43} 1009195 | 9.5047e+06 | {7952.02,3520.88,632.554,1014.25,3
682.3,2152.37,2108.55,5609.13,6663.42,4410.93,7935.51,1272.55,8609.25,2337.6,6845.14,38
49.27,3970,422.706,2090.26,8533.55,9108.23,6752. 42,5636.14,7223.91,7627.38,9467.08,876
3.63,6810.94,819.782,3407.48,6512.03,9083.21,6403.44,6224.57,9703.19,553.033,9508.63,18
23.54,6942.67,5340.35,5954.36,5616.57,6423.06,5320.32,7837.67,2903.61 ,8450.55,1892.85,
3821.65,1140.62,10152.7,2306.96,8871.29,5034.8,17.8199,5573.62,3686.87,10214.3,1688.62,
4667.3,2943.37,7669.45,3079.31,10188.6,4638.13,2907.09,254.251,3438.58,4657.61,6342.84,
948 5.26,465.782,2388.26,5125.77,6048.52,9961.5,8328.46,5174.91,1416.44,1937.93} 100852
3 | 9.65744e+06 | {7255.87,3299.84,671.464,1047.55,3705.29,2031.92,1992.93,5689.99,6486
.58,4153.71,8173.91,1224.91,8320.19,2170.14,6585.28,3911.89,4329.78,401.384,2084.19,834
5.98,9496.74, 7188.78,5137.15,7485.36,6914.55,8471.34,9674.72,6395.1,810.129,3015.94,65
51.72,8213.34,6518.96,6672.72,9064.75,565.507,9560.03,1621.07,7184.9,5224.67,6092.26,48
97.21,6021.32,5271.55,7731.19,3218 .24,8516.33,1660.11,3269.62,1145.53,10584.7,2058.17,
7786.21,4795.73,16.5323,5396.69,3830.83,10393.6,1526.46,4794.47,2644.17,8514.68,3477.77
,9360.25,4510.64,2528.64,238.049,3361.88,4388.69,7549.8 3,9101.76,545.511,2029.84,5622.
08,5770.27,10192.2,8269.93,4979.93,1547.04,2017} (5 rows) Time: 2.502 ms
```

7. 并发能力测试。示例如下：

```
-- 为确保每次输入的图像特征值是随机的，而不是一个确定值，模拟真实场景，使用函数进行测试 -- 测试方法 -- 从表里随机取一条记录，每个维度的浮点值修改5%，生成一个新的随机向量 -- 基于这个新的随机向量，搜索5条最相似的向量，按相似度顺序返回 create or replace function get_vec( in i_id int, in i
_pect int, out v_id int, out v_dist float4, out v_vec float4[] ) returns setof record a
s $$ declare v_vec float4[]; v_pase text; begin select vec into v_vec from t_vec_80 whe
re id=i_id; v_vec := gen_float4_arr(v_vec, i_pect); v_pase := rtrim(ltrim(v_vec::text,
'{'),'}')||':40:0'; -- raise notice '%', v_pase; return query select id as v_id, vec <?
> v_pase::pase as v_dist, vec as v_vec from t_vec_80 order by vec <?> v_pase::pase limi
t 5; end; $$ language plpgsql strict; postgres=> select min(id),max(id) from t_vec_80;
min | max ---------+--------- 1000001 | 2000000 (1 row) vi test.sql \set id random(1000
001,2000000) select * from get_vec(:id, 5); pgbench -M prepared -n -r -f ./test.sql -c
12 -j 12 -T 600 -h xxx.xxx.xxx.xxx -p 3433 -U digoal postgres
```

8. 模拟查询示例如下：

```
postgres=> select * from get_vec(1000001,5); -[ RECORD 1 ]-----------------------------
```

```
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---------------------------------------------------------------------------------- v_id
| 1000001 v_dist | 549580 v_vec | {7508.56,3828.8,670.162,978.82,3654.93,2052.38,2023.4
1,5518.4,6478.1,4814.47,8689.2,1130.5,8421.43,2018.39,6534.18,3884.82,4737.2,385.625,20
25.83,7917.54,8892.97,6900.71,5421.61,7579.82,7649.6,9337.72,9530.01,5818.69,873.353,31
05.67,6622.92,9102.99,6360.46,6737.99,9374.82,545.683,9734.36,1699.74,6753.08,5320.49,6
062.47,4870.6,6907.26,5304.41,7166.67,2997.09,8508.14,1691.62,3595.89,1113.89,10232.1,2
107.41,8846.84,4875.69,17.1081,5574.26,3513.31,10576.6,1763.01,4734.1,2780.48,8165.04,3
132.32,9586.17,4345.39,2859.25,286.716,3306.16,4260.56,7007.33,9126.81,528.518,2288.32,
5310,5610,10584,8872.31,4843.43,1347.01,1940.52} -[ RECORD 2 ]-------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---------------------------------------------------------------------------------- v
_id | 1004945 v_dist | 8.61114e+06 v_vec | {7348.72,3833.3,706.311,985.864,3632.96,2153
.75,2172.06,6427.87,6502.42,4678.54,8955.37,1207.76,8594.73,1958.02,6839.83,3703.57,409
1.18,367.272,1970.81,7266.62,9198.17,6869.98,5960.79,7658.46,7180.35,9386.35,10320.3,65
93.09,900.23,3330.1,6749.94,9182.85,6839.25,7254.11,9533.32,580.504,9069.41,1841.88,684
0.14,4948.41,6390.41,5102.95,6873.49,5683.65,7283.23,3124.15,8727.17,1810.11,3575.12,11
11.99,10081.7,2174.01,8797.29,5301.64,17.779,5196.54,3848.29,9813.85,1514.4,4357.8,2752
.47,7138.15,2905.04,10178.2,5025.82,2713.42,267.272,3557.03,4388.08,6581.85,9114.22,470
.335,2249.53,5274.76,5353.28,10566.6,9153.67,4746.68,1439.05,1996.43} -[ RECORD 3 ]----
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
------------------- v_id | 1003628 v_dist | 9.11551e+06 v_vec | {7532.93,3345.53,694.60
8,984.268,3507.72,1950.43,2188.66,6043.55,6832.57,4384.97,8975.91,1290.02,8519.66,2237.
75,6985.71,3890.79,4199.22,410.386,2294.93,7938.11,8989.48,6374.35,5742.55,7811.5,7492.
1,9067.4,9843.13,5885.26,940.365,3435.39,6545.54,8069.38,6126.34,6906.32,10175.4,505.91
5,9504.69,1630.76,6832.68,5477.68,6446.75,5109.62,6686.55,5688.48,6778.92,3100.2,9182.8
6,1733.95,3933.06,1116.63,10488.3,2346.63,8257.46,5312.34,16.0066,5078.85,3717.24,10262
.9,1624.57,4406.59,2983.23,7405.85,3159.04,9924.56,4947.86,2573.72,276.545,3673.99,4487
.34,6820.15,8524.12,486.187,2328.58,4769.64,5541.63,10255.7,8280.42,5141.37,1332.7,1989
.67} -[ RECORD 4 ]-----------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---------------------------------------- v_id | 1007551 v_dist | 9.44673e+06 v_vec | {
7781.26,3380.16,599.097,902.545,3547.6,1982.01,2408.72,5823.09,5854.29,4392.29,9184.52,
1268.16,8240.44,2106.41,6257.97,3703.93,4635.53,378.289,1987.59,8185.38,8466.11,7341.06
```

```
,5290.8,7422.01,7250.71,8765.47,9341.37,6343.1,865.465,3123.4,5753.41,9331.6,6897.8,641
0.83,8874.91,572.861,9001.73,1567.28,6087.64,5422.22,6226.57,5704.15,6499.31,5340.14,71
57.55,3300.96,8137.33,1648.01,3872.58,1048.15,10322,2171.44,8874.25,4800.68,17.2407,529
7.92,3962.59,10463.2,1482.13,4316.52,2762.2,7293.2,2932.35,10294.3,4539.97,2551.33,266.
689,3879.39,4287.27,7169.59,8934.47,544.819,2246.28,4860.29,5837.37,10389.2,8959.5,4836
.24,1283.66,2118.71} -[ RECORD 5 ]----------------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------
---------------------------------------------------------- v_id | 1008335 v_dist | 9.484
63e+06 v_vec | {7993.58,3279.23,608.321,947.312,3855.4,2190.95,2013.19,6063.82,6356.44,
4670.55,9118.76,1155.98,8339.1,2082.98,6675.26,3565.42,4172.02,432.199,2115.09,7211.91,
8375.44,6845.13,5692.45,7955.92,7269.1,9351.03,9016.28,5845.67,840.522,2964.57,6185.9,9
328.92,6371.88,6985.29,9314.6,575.449,8884.66,1681.17,6381.56,5767.74,5796.38,4839.26,6
309.88,5030.22,7347.04,3403.45,9072.78,1858.26,3753.29,1008.68,10277.5,2072.03,8010.28,
5153.73,17.669,4755.41,3723.93,10381.5,1512.89,4821.96,3179.53,7987.13,3276.66,8983.62,
4408.31,2430.41,284.952,3731.14,4382.78,6574.45,9154.04,520.929,2136.69,4835.47,5222.18
,10158.4,9192.24,4820.05,1417.67,2106.94}
```

说明 查询结果如下：

```
transaction type: ./test.sql scaling factor: 1 query mode: prepared number of clien
ts: 12 number of threads: 12 duration: 600 s number of transactions actually proces
sed: 633784 latency average = 11.361 ms tps = 1056.253960 (including connections es
tablishing) tps = 1056.298691 (excluding connections establishing) statement latenc
ies in milliseconds: 0.001 \set id random(1000001,2000000) 11.358 select * from get
_vec(:id, 5);
```

总结

RDS PostgreSQL通过高维向量索引检索功能（PASE插件），可以实现图像向量的相似匹配高效搜索，单次请求达到毫秒级。高维向量检索不仅能应用在图像搜索，还能应用在任意可以数字化的特征搜索中，例如用户画像特征搜索、营销系统中的相似人群扩选等场景，同时支持索引合并搜索，可以在数据库中一次性完成向量搜索和普通查询条件的联合过滤，大幅度提升效率。

使用RDS PostgreSQL数据库，可以满足在互联网、新零售、交通、智能楼宇、教育、游戏、医疗、社交、公安、机场等行业中的图像识别、人脸识别、相似特征检索、相似人群圈选等高并发查询搜索需求。

视频教程

多维向量相似搜索-图像识别等

优惠活动

新用户9.9元试用、老用户5折升级

29.3. 秒级闪回实时容灾

基于ZFS（Zettabyte File System）的快照功能，您可以设置一个ECS实例作为备节点，通过Cron定期创建快照（秒级快照），就可以实现数据的秒级闪回，从而达到实时容灾的目的。

背景信息

ZFS是一款动态文件管理系统，与其他文件系统的管理方式不同，ZFS采用存储池的概念来管理物理存储空间，把所有设备集中到一个存储池中来进行管理。存储池描述了存储的物理特征（设备的布局、数据的冗余等），使得文件系统不再局限于单独的物理设备，而且允许物理设备把自带的文件系统共享到存储池中。

ZFS使用写时拷贝事务模型技术，写新数据时，包含旧数据的块被保留着，由于ZFS在读写操作时已经存储了所有构建快照的数据，所以快照的创建非常快，而且由于任何文件的修改都是在文件系统和它的快照之间共享的，所以ZFS的快照也是空间优化的。

基于ZFS的这些特性，本文介绍如何设置一个ECS实例作为备节点，通过Cron定期创建快照（秒级快照）。恢复时您就可以找到最近的快照，实现快速恢复数据、实时容灾。

准备工作

- 创建合适规格的ECS实例。建议可用区、专有网络和交换机与RDS PostgreSQL相同，这样内网访问速度最快。如果跨地域容灾，需要使用云企业网连通实例，详情请参见云企业网。
- 设置白名单，允许ECS实例访问RDS实例。

创建快照

1. 连接ECS实例。
2. 安装ZFS。

```
wget http://download.zfsonlinux.org/epel/zfs-release.el7_7.noarch.rpm rpm -ivh zfs-release.el7_7.noarch.rpm yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm yum install -y "kernel-devel-uname-r == $(uname -r)" zfs
```

3. 修改文件rc.local。

```
vi /etc/rc.local /sbin/modprobe zfs chmod +x /etc/rc.local
```

4. 测试ZFS是否正常，然后重启实例。

```
# modprobe zfs # zpool list no pools available # reboot
```

5. 安装PostgreSQL 12。

```
yum install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm yum install postgresql12*
```

6. 在云盘上进行如下配置。

```
parted -a optimal -s /dev/vdb1 mklabel gpt mkpart primary 1MiB 100%FREE // 对齐分区，/dev/vdb1是新增云盘的路径。 zpool create zp1 -f -o ashift=13 vdc1 //配置ZFS存储池。 zfs set canmount=off zp1 //设置canmount属性。
```

7. 创建数据文件目录。

```
zfs create -o mountpoint=/zpdata01 -o recordsize=8K -o atime=off -o primarycache=metadata -o logbias=throughput -o secondarycache=none zp1/zpdata01
```

8. 创建WAL归档目录。

```
zfs create -o mountpoint=/zpdata02 -o recordsize=8K -o atime=off -o primarycache=metadata -o logbias=throughput -o secondarycache=none -o compression=on zp1/zpdata02 zfs set compression=on zp1/zpdata02 # zfs list NAME USED AVAIL REFER MOUNTPOINT zp1 1.29M 1.42T 192K /zp1 zp1/zpdata01 192K 1.42T 192K /zpdata01 zp1/zpdata02 192K 1.42T 192K /zpdata02
```

9. 创建RDS实例的备库文件夹。

```
mkdir /zpdata01/pg12_1921_data mkdir /zpdata02/pg12_1921_wal chown -R postgres:postgres
/zpdata01/pg12_1921_data chown -R postgres:postgres /zpdata02/pg12_1921_wal
```

10. 同步RDS实例数据到ECS实例。

```
su - postgres export PGPASSWORD=<RDS实例高权限账号的密码> nohup pg_basebackup -D /zpdata01
/pg12_1921_data -F p -R -c fast -X stream -h <RDS实例连接地址> -p <RDS实例连接端口> -U <RD
S实例高权限账号名> >./bak.log 2>&1 &
```

> **说明** 请确保白名单已放通ECS实例，放通方法请参见 设置白名单。

```
[root@iZ[redacted]jwZ ~]# mkdir /zpdata01/pg12_1921_data
[root@iZ[redacted]jwZ ~]# mkdir /zpdata02/pg12_1921_wal
[root@iZ[redacted]jwZ ~]# chown -R postgres:postgres /zpdata01/pg12_1921_data
[root@iZ[redacted]jwZ ~]# chown -R postgres:postgres /zpdata02/pg12_1921_wal
[root@iZ[redacted]jwZ ~]# su - postgres
-bash-4.2$ export PGPASSWORD=[redacted]
-bash-4.2$ nohup pg_basebackup -D /zpdata01/pg12_1921_data -F p -R -c fast -X stream -h pgm-bp1[redacted].pg.rds.aliyuncs.com -p 1433 -U l[redacted]3  >./bak.log 2>&1 &
[1] 2295
```

11. 注释*/zpdata01/pg12_1921_data/postgresql.conf*的部分配置。

```
#Fri Mar 13 09:55:03 CST 2020 #ssl_key_file='server.key' #huge_pages=try #auto_explain.
sample_rate=1 #zhparser.multi_zall=off #shared_preload_libraries='pg_stat_statements,au
th_delay,auto_explain,zhparser,timescaledb,pg_pathman' #promote_trigger_file='/data/pos
tgresql.trigger' #ssl=off #rds_max_log_files=20 #pg_pathman.enable_auto_partition=on #s
hared_buffers=32768MB #zhparser.punctuation_ignore=off #pg_pathman.override_copy=on #po
rt=1922 #pg_stat_statements.max=5000 #auth_delay.milliseconds=3s #auto_explain.log_nest
ed_statements=off #track_io_timing=on #zhparser.multi_zmain=off #auto_explain.log_analy
ze=off #archive_mode=on #ssl_cert_file='server.crt' #zhparser.multi_short=off #zhparser
.dict_in_memory=off #auto_explain.log_format=text #auto_explain.log_min_duration=-1 #rd
s.rds_max_non_super_conns=12800 #pg_pathman.enable=on #archive_command='/bin/date' #aut
o_explain.log_verbose=off #log_line_prefix='\1\n\t%p\t%r\t%u\t%d\t%t\t%e\t%T\t%S\t%U\t%
E\t\t' #pg_pathman.enable_runtimemergeappend=on #zhparser.extra_dicts='dict_extra.xdb'
#auto_explain.log_buffers=off #pg_stat_statements.track=top #jit_provider='llvmjit' #pg
_pathman.enable_partitionrouter=off #pg_stat_statements.track_utility=off #pg_stat_stat
ements.save=off #zhparser.dicts_type='EXTRA' #auto_explain.log_timing=on #pg_pathman.en
able_runtimeappend=on #zhparser.seg_with_duality=off #rds.rds_max_super_conns=100 #pg_p
athman.enable_partitionfilter=on #log_destination='stderr,csvlog' #zhparser.multi_duali
ty=off #pg_pathman.insert_into_fdw='postgres' #pg_pathman.enable_bounds_cache=on #rds.r
ds_max_non_super_wal_snd=32 #auto_explain.log_triggers=off #rds_sync_replication_timeou
t=0
```

12. 修改配置文件。

```
vi /zpdata01/pg12_1921_data/postgresql.auto.conf primary_conninfo = 'user=<RDS实例高权限
账号名> password=''<RDS实例高权限账号的密码> '' host=''<RDS实例连接地址>'' port=<RDS实例连接
端口> application_name=hello_rds_pg12' port=1922 shared_buffers=32GB log_destination='cs
vlog' archive_mode=always archive_command='test ! -f /zpdata02/pg12_1921_wal/%f && cp %
p /zpdata02/pg12_1921_wal/%f'
```

13. 修改文件夹权限。

```
chmod 700 /zpdata02/pg12_1921_wal chmod 700 /zpdata01/pg12_1921_data
```

14. 启动数据库。

```
su - postgres /usr/pgsql-12/bin/pg_ctl start -D /zpdata01/pg12_1921_data
```

> **说明** 如果您的ECS实例配置过低，可能会出现如下报错，建议升级配置。
>
> ```
> HINT: This error usually means that PostgreSQL's request for a shared memory segmen
> t exceeded available memory, swap space, or huge pages.
> ```

15. 设置数据库自动启动。

```
vi /etc/rc.local su - postgres -c "/usr/pgsql-12/bin/pg_ctl start -D /zpdata01/pg12_192
1_data"
```

16. 配置数据文件目录自动快照（归档目录不需要快照）。
    i. 创建脚本，配置执行权限。

    ```
    vi /etc/snap.sh STIME=`date +%F%T` /usr/sbin/zfs snapshot zp1/zpdata01@$STIME chmod
    500 /etc/snap.sh
    ```

    ii. 测试快照是否正常。

    ```
    /etc/snap.sh # zfs list -t snapshot NAME USED AVAIL REFER MOUNTPOINT zp1/zpdata01@2
    020-03-2117:06:47 144K - 770M -
    ```

    iii. 自动启动Crond。

    ```
    # systemctl start crond # systemctl enable crond # systemctl status crond ● crond.s
    ervice - Command Scheduler Loaded: loaded (/usr/lib/systemd/system/crond.service; e
    nabled; vendor preset: enabled) Active: active (running) since Sat 2020-03-21 16:16
    :08 CST; 53min ago Main PID: 2526 (crond) CGroup: /system.slice/crond.service └─252
    6 /usr/sbin/crond -n
    ```

    iv. 配置Crontab。

    ```
    # crontab -e 1 1 * * * /etc/snap.sh # crontab -l -u root 1 1 * * * /etc/snap.sh
    ```

> **说明** 您也可以根据磁盘空间和业务需求设置自动清理快照或自动清理归档。手动操作示例如下：
>
> ```
> 手动清理快照示例 # zfs list -t snapshot zNAME USED AVAIL REFER MOUNTPOINT zp1/zpdata0
> 1@2020-03-2117:06:47 144K - 770M - zp1/zpdata01@2020-03-2117:17:01 0B - 786M - # zf
> s destroy zp1/zpdata01@2020-03-2117:06:47 # zfs list -t snapshot NAME USED AVAIL RE
> FER MOUNTPOINT zp1/zpdata01@2020-03-2117:17:01 0B - 786M - 手动清理归档示例 find /zpd
> ata02/pg12_1921_wal/ -type f -mtime +7 -exec rm -f {} \;
> ```

17. 校验备份集可用性。
18. 检测主从延迟。

```
postgres=> select pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_flush_lsn(),sent_lsn))
as sent_delay, pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_flush_lsn(),replay_lsn)) a
s replay_dealy,* from pg_stat_replication ; -[ RECORD 1 ]----+-------------------------
----- sent_delay | 0 bytes replay_dealy | 0 bytes pid | 84098 usesysid | 886185 usename
| rep application_name | hello_rds_pg12 client_addr | 192.168.0.173 client_hostname | c
lient_port | 60402 backend_start | 2020-03-21 16:59:01.890775+08 backend_xmin | state |
streaming sent_lsn | 11D/97002068 write_lsn | 11D/97002068 flush_lsn | 11D/97002068 rep
lay_lsn | 11D/97002068 write_lag | flush_lag | replay_lag | sync_priority | 0 sync_stat
e | async reply_time | 2020-03-21 17:01:17.198139+08
```

至此，您的ECS实例已经实现定期快照备份，可以参见下文进行秒级闪回，实现实时容灾。

秒级闪回

1. 基于快照克隆一个ZFS文件系统。

```
# zfs list -t snapshot NAME USED AVAIL REFER MOUNTPOINT zp1/zpdata01@2020-03-2117:17:01
312K - 786M - # zfs clone -o mountpoint=/test_recovery zp1/zpdata01@2020-03-2117:17:01
zp1/zpdata_test [root@iZbp135pwcjjoxqgfpw9k1Z pg12_1921_wal]# cd /test_recovery [root@i
Zbp135pwcjjoxqgfpw9k1Z test_recovery]# ll total 17 drwx------ 20 postgres postgres 35 M
ar 21 16:59 pg12_1921_data
```

2. 配置数据恢复相关参数。

> **说明** 如果ECS实例的内存不足，可以设置较小的shared_buffer。

```
vi /test_recovery/pg12_1921_data/postgresql.auto.conf port=1923 shared_buffers=32GB log
_destination='csvlog' recovery_end_command = 'cp /zpdata02/pg12_1921_wal/%f %p' recover
y_target_time = '2020-03-21 17:28:37.670338+08' recovery_target_timeline = 'latest' rec
overy_target_action = 'pause'
```

3. 删除克隆文件系统中的socket、pid等文件。

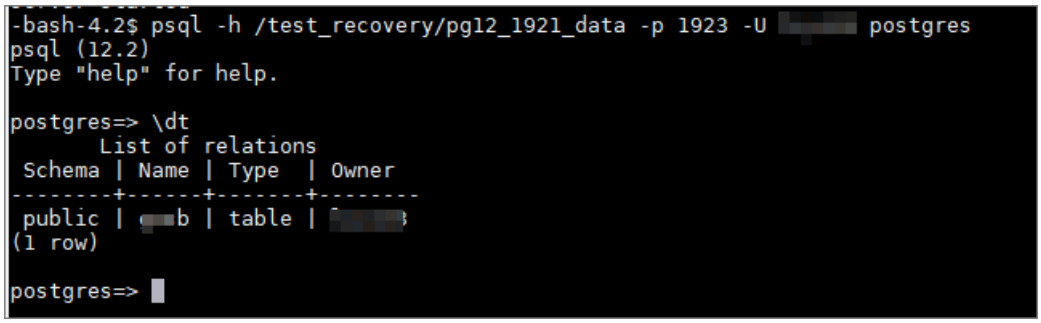
```
rm -f /test_recovery/pg12_1921_data/.s.* rm /test_recovery/pg12_1921_data/postmaster.pi
d
```

4. 进行恢复操作。

```
su - postgres /usr/pgsql-12/bin/pg_ctl start -D /test_recovery/pg12_1921_data
```

5. 查看恢复库中的数据。

```
psql -h /test_recovery/pg12_1921_data -p 1923 -U <RDS实例高权限账号> postgres psql (12.1)
Type "help" for help. postgres=> \dt
```



> **说明** 确认数据正常，您可以使用pgdump工具将数据恢复至RDS实例。

清理恢复库

恢复库使用完成后，按如下命令清理克隆的test_recovery文件系统即可。

```
su - postgres /usr/pgsql-12/bin/pg_ctl stop -m fast -D /test_recovery/pg12_1921_data sudo z
fs destroy zp1/zpdata_test
```

29.4. 基于pgpool实现读写分离

本文介绍ECS实例上的PostgreSQL如何结合pgpool实现读写分离，您也可以通过RDS PostgreSQL实例及只读实例简化操作步骤。

背景信息

不使用pgpool实现数据库的高可用时，pgpool自身是无状态的，性能损耗很小，同时还支持横向扩展，因此搭配自身具有高可用架构的RDS PostgreSQL实例，可以方便快捷地实现读写分离。

部署环境

如果您已经购买PostgreSQL 10本地SSD盘及只读实例（详情请参见创建RDS PostgreSQL实例和创建PostgreSQL只读实例），仅需安装pgpool，然后请跳转到配置pgpool查看后续步骤。

> **说明** PostgreSQL云盘版的只读实例敬请期待。

测试环境：

- ECS实例规格为16核CPU、64GB内存、1.8TB SSD云盘。
- ECS实例系统为CentOS 7.7 x64。

操作步骤如下。

1. 修改配置文件sysctl.conf，命令如下：

```
vi /etc/sysctl.conf # add by digoal.zhou fs.aio-max-nr = 1048576 fs.file-max = 76724600 # 可选：kernel.core_pattern = /data01/corefiles/core_%e_%u_%t_%s.%p # /data01/corefiles 提前建好，权限777，如果是软链接，对应的目录修改为777。 kernel.sem = 4096 2147483647 2147483646 512000 # 信号量，ipcs -l 或 -u 查看，每16个进程一组，每组信号量需要17个信号量。 kernel.shmall = 107374182 # 所有共享内存段相加大小限制 (建议内存的80%)，单位为页。 kernel.shmmax = 274877906944 # 最大单个共享内存段大小 (建议为内存一半)，大于9.2的版本已大幅降低共享内存的使用，单位为字节。 kernel.shmmni = 819200 # 一共能生成多少共享内存段，每个PG数据库集群至少2个共享内存段。 net.core.netdev_max_backlog = 10000 net.core.rmem_default = 262144 # The default setting of the socket receive buffer in bytes. net.core.rmem_max = 4194304 # The maximum receive socket buffer size in bytes net.core.wmem_default = 262144 # The default setting (in bytes) of the socket send buffer. net.core.wmem_max = 4194304 # The maximum send socket buffer size in bytes. net.core.somaxconn = 4096 net.ipv4.tcp_max_syn_backlog = 4096 net.ipv4.tcp_keepalive_intvl = 20 net.ipv4.tcp_keepalive_probes = 3 net.ipv4.tcp_keepalive_time = 60 net.ipv4.tcp_mem = 8388608 12582912 16777216 net.ipv4.tcp_fin_timeout = 5 net.ipv4.tcp_synack_retries = 2 net.ipv4.tcp_syncookies = 1 # 开启SYN Cookies。当出现SYN等待队列溢出时，启用cookie来处理，可防范少量的SYN攻击。 net.ipv4.tcp_timestamps = 1 # 减少time_wait。 net.ipv4.tcp_tw_recycle = 0 # 如果=1则开启TCP连接中TIME-WAIT套接字的快速回收，但是NAT环境可能导致连接失败，建议服务端关闭它。 net.ipv4.tcp_tw_reuse = 1 # 开启重用。允许将TIME-WAIT套接字重新用于新的TCP连接。 net.ipv4.tcp_max_tw_buckets = 262144 net.ipv4.tcp_rmem = 8192 87380 16777216 net.ipv4.tcp_wmem = 8192 65536 16777216 net.nf_conntrack_max = 1200000 net.netfilter.nf_conntrack_max = 1200000 vm.dirty_background_bytes = 409600000 # 系统脏页到达这个值，系统后台刷脏页调度进程 pdflush（或其他） 自动将(dirty_expire_centisecs/100）秒前的脏页刷到磁盘。 # 默认为10%，大内存机器建议调整为直接指定多少字节。 vm.dirty_expire_centisecs = 3000 # 大于这个值的脏页，将被刷到磁盘。3000表示30秒。 vm.dirty_ratio = 95 # 如果系统进程刷脏页太慢，使得系统脏页超过内存 95 % 时，则用户进程如果有写磁盘的操作（如fsync、fdatasync等调用），则需要主动把系统脏页刷出。 # 有效防止用户进程刷脏页，在单机多实例，并且使用CGROUP限制单实例IOPS的情况下非常有效。 vm.dirty_writeback_centisecs = 100 # pdflush（或其他）后台刷脏页进程的唤醒间隔，100表示1秒。 vm.swappiness = 0 # 不使用交换分区。 vm.mmap_min_addr = 65536 vm.overcommit_memory = 0 # 在分配内存时，允许少量over malloc，如果设置为 1，则认为总是有足够的内存，内存较少的测试环境可以使用 1。 vm.overcommit_ratio = 90 # 当overcommit_memory = 2 时，用于参与计算允许指派的内存大小。 vm.swappiness = 0 # 关闭交换分区。 vm.zone_reclaim_mode = 0 # 禁用 numa，或者在vmlinux中禁止。 net.ipv4.ip_local_port_range = 40000 65535 # 本地自动分配的TCP，UDP端口号范围。 fs.nr_open=20480000 # 单个进程允许打开的文件句柄上限。 # 以下参数请注意。 #vm.extra_free_kbytes = 4096000 # 小内存机器不要设置这样大，会无法开机。 #vm.min_free_kbytes = 6291456 # vm.min_free_kbytes 建议每32G内存分配1G vm.min_free_kbytes。 # 如果是小内存机器，以上两个值不建议设置。 # vm.nr_hugepages = 66536 # 建议shared buffer设置超过64GB时 使用大页，页大小 /proc/meminfo Hugepagesize。 #vm.lowmem_reserve_ratio = 1 1 1 # 对于内存大于64G时，建议设置，否则建议默认值 256 256 32。
```

2. 修改配置文件limits.conf，命令如下：

```
vi /etc/security/limits.conf * soft nofile 1024000 * hard nofile 1024000 * soft nproc unlimited * hard nproc unlimited * soft core unlimited * hard core unlimited * soft memlock unlimited * hard memlock unlimited # 注释其他 # 同时注释/etc/security/limits.d/20-nproc.conf。
```

3. 关闭透明大页、配置大页、并自启动PostgreSQL。命令如下：

```
chmod +x /etc/rc.d/rc.local vi /etc/rc.local # 关闭透明大页。 if test -f /sys/kernel/mm/transparent_hugepage/enabled; then echo never > /sys/kernel/mm/transparent_hugepage/enabled fi # 两个实例，每个实例16G shared buffer。 #sysctl -w vm.nr_hugepages=17000 # 自启动两个实例。 su - postgres -c "pg_ctl start -D /data01/pg12_3389/pg_root" su - postgres -c "pg_ctl start -D /data01/pg12_8002/pg_root"
```

4. 创建文件系统。命令如下：

> **警告** 仅针对新磁盘，请确认磁盘（例如新挂载磁盘为vdb而不是vda）后再进行操作，否则可能会导致数据清空。

```
parted -a optimal -s /dev/vdb mklabel gpt mkpart primary 1MiB 100%FREE mkfs.ext4 /dev/v
db1 -m 0 -O extent,uninit_bg -E lazy_itable_init=1 -b 4096 -T largefile -L vdb1 vi /etc
/fstab LABEL=vdb1 /data01 ext4 defaults,noatime,nodiratime,nodelalloc,barrier=0,data=wr
iteback 0 0 mkdir /data01 mount -a
```

5. 启动irq balance。命令如下：

```
systemctl status irqbalance systemctl enable irqbalance systemctl start irqbalance syst
emctl status irqbalance
```

6. 安装PostgreSQL 10、pgpool工具软件。命令如下：

```
yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm y
um install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-r
edhat-repo-latest.noarch.rpm yum search all postgresql yum search all pgpool yum instal
l -y postgresql12* yum install -y pgpool-II-12-extensions
```

7. 初始化数据库数据目录。命令如下：

```
mkdir /data01/pg12_3389 chown postgres:postgres /data01/pg12_3389
```

8. 配置postgres用户环境变量。命令如下：

```
su - postgres vi .bash_profile # 追加 export PS1="$USER@`/bin/hostname -s`-> " export PG
PORT=3389 export PGDATA=/data01/pg12_$PGPORT/pg_root export LANG=en_US.utf8 export PGHO
ME=/usr/pgsql-12 export LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:
/lib:/usr/lib:/usr/local/lib:$LD_LIBRARY_PATH export DATE=`date +"%Y%m%d%H%M"` export P
ATH=$PGHOME/bin:$PATH:. export MANPATH=$PGHOME/share/man:$MANPATH export PGHOST=$PGDATA
export PGUSER=postgres export PGDATABASE=db1 alias rm='rm -i' alias ll='ls -lh' unalias
vi
```

9. 初始化主库。命令如下：

```
initdb -D $PGDATA -U postgres -E UTF8 --lc-collate=C --lc-ctype=en_US.utf8
```

10. 修改配置文件postgresql.conf。命令如下：

```
listen_addresses = '0.0.0.0' port = 3389 max_connections = 1500 superuser_reserved_conn
ections = 13 unix_socket_directories = '., /var/run/postgresql, /tmp' tcp_keepalives_id
le = 60 tcp_keepalives_interval = 10 tcp_keepalives_count = 10 shared_buffers = 16GB hu
ge_pages = on work_mem = 8MB maintenance_work_mem = 1GB dynamic_shared_memory_type = po
six vacuum_cost_delay = 0 bgwriter_delay = 10ms bgwriter_lru_maxpages = 1000 bgwriter_l
ru_multiplier = 10.0 bgwriter_flush_after = 512kB effective_io_concurrency = 0 max_work
er_processes = 128 max_parallel_maintenance_workers = 3 max_parallel_workers_per_gather
= 4 parallel_leader_participation = off max_parallel_workers = 8 backend_flush_after =
256 wal_level = replica synchronous_commit = off full_page_writes = on wal_compression
= on wal_buffers = 16MB wal_writer_delay = 10ms wal_writer_flush_after = 1MB checkpoint
_timeout = 15min max_wal_size = 64GB min_wal_size = 8GB checkpoint_completion_target =
0.2 checkpoint_flush_after = 256kB random_page_cost = 1.1 effective_cache_size = 48GB l
og_destination = 'csvlog' logging_collector = on log_directory = 'log' log_filename = '
postgresql-%a.log' log_truncate_on_rotation = on log_rotation_age = 1d log_rotation_siz
e = 0 log_min_duration_statement = 1s log_checkpoints = on log_connections = on log_dis
connections = on log_line_prefix = '%m [%p] ' log_statement = 'ddl' log_timezone = 'Asi
a/Shanghai' autovacuum = on log_autovacuum_min_duration = 0 autovacuum_vacuum_scale_fac
tor = 0.1 autovacuum_analyze_scale_factor = 0.05 autovacuum_freeze_max_age = 800000000
autovacuum_multixact_freeze_max_age = 900000000 autovacuum_vacuum_cost_delay = 0 vacuum
_freeze_table_age = 750000000 vacuum_multixact_freeze_table_age = 750000000 datestyle =
'iso, mdy' timezone = 'Asia/Shanghai' lc_messages = 'en_US.utf8' lc_monetary = 'en_US.u
tf8' lc_numeric = 'en_US.utf8' lc_time = 'en_US.utf8' default_text_search_config = 'pg_
catalog.english'
```

11. 修改配置文件pg_hba.conf。命令如下：

> **说明** 因为pgpool-II和数据库服务器处于同一ECS实例，所以设置为127.0.0.1时需要密码才能登录。

```
# "local" is for Unix domain socket connections only local all all trust # IPv4 local c
onnections: host all all 127.0.0.1/32 md5 # IPv6 local connections: host all all ::1/12
8 trust # Allow replication connections from localhost, by a user with the # replicatio
n privilege. local replication all trust host replication all 127.0.0.1/32 trust host r
eplication all ::1/128 trust host db123 digoal 0.0.0.0/0 md5
```

12. 创建流复制用户。示例如下：

```
db1=# create role rep123 login replication encrypted password 'xxxxxxx'; CREATE ROLE
```

13. 创建业务用户。示例如下：

```
db1=# create role digoal login encrypted password 'xxxxxxx'; CREATE ROLE db1=# create d
atabase db123 owner digoal; CREATE DATABASE
```

14. 创建pgpool数据库健康心跳用户，检查只读节点回放延迟（wal replay），只要能登录postgres数据库或指定的库即可，配合pgpool参数使用。示例如下：

```
create role nobody login encrypted password 'xxxxxxx';
```

创建从库

为简化测试步骤，在同一ECS实例创建备库。

1. 使用pg_basebackup在线创建从库。命令如下：

```
pg_basebackup -D /data01/pg12_8002/pg_root -F p --checkpoint=fast -P -h 127.0.0.1 -p 33
89 -U rep123
```

2. 修改从库配置文件postgresql.conf。命令如下：

```
cd /data01/pg12_8002/pg_root vi postgresql.conf # 相比主配置 修改如下： port = 8002 primar
y_conninfo = 'hostaddr=127.0.0.1 port=3389 user=rep123' # 不用设置密码，因为主设置了trust
访问。 hot_standby = on wal_receiver_status_interval = 1s wal_receiver_timeout = 10s rec
overy_target_timeline = 'latest'
```

3. 配置从库standby.signal标记。命令如下：

```
cd /data01/pg12_8002/pg_root touch standby.signal
```

4. 查看主从同步是否正常。命令如下：

```
db1=# select * from pg_stat_replication ; -[ RECORD 1 ]----+----------------------------
--- pid | 21065 usesysid | 10 usename | postgres application_name | walreceiver client_
addr | 127.0.0.1 client_hostname | client_port | 47064 backend_start | 2020-02-29 00:26
:28.485427+08 backend_xmin | state | streaming sent_lsn | 0/52000060 write_lsn | 0/5200
0060 flush_lsn | 0/52000060 replay_lsn | 0/52000060 write_lag | flush_lag | replay_lag
| sync_priority | 0 sync_state | async reply_time | 2020-02-29 01:32:40.635183+08
```

配置pgpool

1. 查询pgpool安装位置。命令如下：

```
# rpm -qa|grep pgpool pgpool-II-12-extensions-4.1.1-1.rhel7.x86_64 pgpool-II-12-4.1.1-1
.rhel7.x86_64 # rpm -ql pgpool-II-12-4.1.1
```

2. 修改配置文件pgpool.conf。命令如下：

```
# cd /etc/pgpool-II-12/ cp pgpool.conf.sample-stream pgpool.conf vi pgpool.conf # -----
------------------------ # pgPool-II configuration file # ---------------------------- #
# This file consists of lines of the form: # # name = value # # Whitespace may be used.
Comments are introduced with "#" anywhere on a line. # The complete list of parameter n
ames and allowed values can be found in the # pgPool-II documentation. # # This file is
read on server startup and when the server receives a SIGHUP # signal. If you edit the
file on a running system, you have to SIGHUP the # server for the changes to take effec
t, or use "pgpool reload". Some # parameters, which are marked below, require a server
shutdown and restart to # take effect. # #-------------------------------------------
---------------------------------- # CONNECTIONS #-------------------------------------
---------------------------------------- # - pgpool Connection Settings - listen_addres
ses = '0.0.0.0' # Host name or IP address to listen on: # '*' for all, '' for no TCP/IP
connections # (change requires restart) port = 8001 # Port number # (change requires re
start) socket_dir = '/tmp' # Unix domain socket path # The Debian package defaults to #
/var/run/postgresql # (change requires restart) reserved_connections = 0 # Number of re
served connections. # Pgpool-II does not accept connections if over # num_init_chidlren
- reserved_connections. # - pgpool Communication Manager Connection Settings - pcp_list
en_addresses = '' # Host name or IP address for pcp process to listen on: # '*' for all
, '' for no TCP/IP connections # (change requires restart) pcp_port = 9898 # Port numbe
r for pcp # (change requires restart) pcp_socket_dir = '/tmp' # Unix domain socket path
for pcp # The Debian package defaults to # /var/run/postgresql # (change requires resta
rt) listen_backlog_multiplier = 2 # Set the backlog parameter of listen(2) to # num_ini
t_children * listen_backlog_multiplier. # (change requires restart) serialize_accept =
off # whether to serialize accept() call to avoid thundering herd problem # (change req
```

```
uires restart) # - Backend Connection Settings - backend_hostname0 = '127.0.0.1' # Host
name or IP address to connect to for backend 0 backend_port0 = 3389 # Port number for b
ackend 0 backend_weight0 = 1 # Weight for backend 0 (only in load balancing mode) backe
nd_data_directory0 = '/data01/pg12_3389/pg_root' # Data directory for backend 0 backend
_flag0 = 'ALWAYS_MASTER' # Controls various backend behavior # ALLOW_TO_FAILOVER, DISAL
LOW_TO_FAILOVER # or ALWAYS_MASTER backend_application_name0 = 'server0' # walsender's
application_name, used for "show pool_nodes" command backend_hostname1 = '127.0.0.1' ba
ckend_port1 = 8002 backend_weight1 = 1 backend_data_directory1 = '/data01/pg12_8002/pg_
root' backend_flag1 = 'DISALLOW_TO_FAILOVER' backend_application_name1 = 'server1' # -
Authentication - enable_pool_hba = on # Use pool_hba.conf for client authentication poo
l_passwd = 'pool_passwd' # File name of pool_passwd for md5 authentication. # "" disabl
es pool_passwd. # (change requires restart) authentication_timeout = 60 # Delay in seco
nds to complete client authentication # 0 means no timeout. allow_clear_text_frontend_a
uth = off # Allow Pgpool-II to use clear text password authentication # with clients, w
hen pool_passwd does not # contain the user password # - SSL Connections - ssl = off #
Enable SSL support # (change requires restart) #ssl_key = './server.key' # Path to the
SSL private key file # (change requires restart) #ssl_cert = './server.cert' # Path to
the SSL public certificate file # (change requires restart) #ssl_ca_cert = '' # Path to
a single PEM format file # containing CA root certificate(s) # (change requires restart
) #ssl_ca_cert_dir = '' # Directory containing CA root certificate(s) # (change require
s restart) ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL' # Allowed SSL ciphers # (change req
uires restart) ssl_prefer_server_ciphers = off # Use server's SSL cipher preferences, #
rather than the client's # (change requires restart) ssl_ecdh_curve = 'prime256v1' # Na
me of the curve to use in ECDH key exchange ssl_dh_params_file = '' # Name of the file
containing Diffie-Hellman parameters used # for so-called ephemeral DH family of SSL ci
pher. #------------------------------------------------------------------------------ #
POOLS #------------------------------------------------------------------------------ #
- Concurrent session and pool size - num_init_children = 128 # Number of concurrent ses
sions allowed # (change requires restart) max_pool = 4 # Number of connection pool cach
es per connection # (change requires restart) # - Life time - child_life_time = 300 # P
ool exits after being idle for this many seconds child_max_connections = 0 # Pool exits
after receiving that many connections # 0 means no exit connection_life_time = 0 # Conn
ection to backend closes after being idle for this many seconds # 0 means no close clie
nt_idle_limit = 0 # Client is disconnected after being idle for that many seconds # (ev
en inside an explicit transactions!) # 0 means no disconnection #----------------------
------------------------------------------------------------ # LOGS #----------------------
------------------------------------------------------------ # - Where to log - log_destina
tion = 'syslog' # Where to log # Valid values are combinations of stderr, # and syslog.
Default to stderr. # - What to log - log_line_prefix = '%t: pid %p: ' # printf-style st
ring to output at beginning of each log line. log_connections = on # Log connections lo
g_hostname = off # Hostname will be shown in ps status # and in logs if connections are
logged log_statement = off # Log all statements log_per_node_statement = off # Log all
statements # with node and backend informations log_client_messages = off # Log any cli
ent messages log_standby_delay = 'if_over_threshold' # Log standby delay # Valid values
are combinations of always, # if_over_threshold, none # - Syslog specific - syslog_faci
lity = 'LOCAL0' # Syslog local facility. Default to LOCAL0 syslog_ident = 'pgpool' # Sy
slog program identification string # Default to 'pgpool' # - Debug - #log_error_verbosi
ty = default # terse, default, or verbose messages #client_min_messages = notice # valu
es in order of decreasing detail: # debug5 # debug4 # debug3 # debug2 # debug1 # log #
notice # warning # error #log_min_messages = warning # values in order of decreasing de
tail: # debug5 # debug4 # debug3 # debug2 # debug1 # info # notice # warning # error #
log # fatal # panic #------------------------------------------------------------------
------------ # FILE LOCATIONS #------------------------------------------------------------
---------------------- pid_file_name = '/var/run/pgpool-II-12/pgpool.pid' # PID file na
```

```
me # Can be specified as relative to the" # location of pgpool.conf file or # as an abs
olute path # (change requires restart) logdir = '/tmp' # Directory of pgPool status fil
e # (change requires restart) #------------------------------------------------------
---------------------- # CONNECTION POOLING #-----------------------------------------
------------------------------------- connection_cache = on # Activate connection pools
# (change requires restart) # Semicolon separated list of queries # to be issued at the
end of a session # The default is for 8.3 and later reset_query_list = 'ABORT; DISCARD
ALL' # The following one is for 8.2 and before #reset_query_list = 'ABORT; RESET ALL; S
ET SESSION AUTHORIZATION DEFAULT' #--------------------------------------------------
-------------------------- # REPLICATION MODE #-------------------------------------
------------------------------------- replication_mode = off # Activate replication mo
de # (change requires restart) replicate_select = off # Replicate SELECT statements # w
hen in replication mode # replicate_select is higher priority than # load_balance_mode.
insert_lock = off # Automatically locks a dummy row or a table # with INSERT statements
to keep SERIAL data # consistency # Without SERIAL, no lock will be issued lobj_lock_ta
ble = '' # When rewriting lo_create command in # replication mode, specify table name t
o # lock # - Degenerate handling - replication_stop_on_mismatch = off # On disagreement
with the packet kind # sent from backend, degenerate the node # which is most likely "m
inority" # If off, just force to exit this session failover_if_affected_tuples_mismatch
= off # On disagreement with the number of affected # tuples in UPDATE/DELETE queries,
then # degenerate the node which is most likely # "minority". # If off, just abort the
transaction to # keep the consistency #---------------------------------------------
------------------------------ # LOAD BALANCING MODE #----------------------------------
---------------------------------------------- load_balance_mode = on # Activate load ba
lancing mode # (change requires restart) ignore_leading_white_space = on # Ignore leadi
ng white spaces of each query white_function_list = '' # Comma separated list of functi
on names # that don't write to database # Regexp are accepted black_function_list = 'cu
rrval,lastval,nextval,setval' # Comma separated list of function names # that write to
database # Regexp are accepted black_query_pattern_list = '' # Semicolon separated list
of query patterns # that should be sent to primary node # Regexp are accepted # valid f
or streaming replicaton mode only. database_redirect_preference_list = '' # comma separ
ated list of pairs of database and node id. # example: postgres:primary,mydb[0-4]:1,myd
b[5-9]:2' # valid for streaming replicaton mode only. app_name_redirect_preference_list
= '' # comma separated list of pairs of app name and node id. # example: 'psql:primary,
myapp[0-4]:1,myapp[5-9]:standby' # valid for streaming replicaton mode only. allow_sql_
comments = off # if on, ignore SQL comments when judging if load balance or # query cac
he is possible. # If off, SQL comments effectively prevent the judgment # (pre 3.4 beha
vior). disable_load_balance_on_write = 'transaction' # Load balance behavior when write
query is issued # in an explicit transaction. # Note that any query not in an explicit
transaction # is not affected by the parameter. # 'transaction' (the default): if a wri
te query is issued, # subsequent read queries will not be load balanced # until the tra
nsaction ends. # 'trans_transaction': if a write query is issued, # subsequent read que
ries in an explicit transaction # will not be load balanced until the session ends. # '
always': if a write query is issued, read queries will # not be load balanced until the
session ends. statement_level_load_balance = off # Enables statement level load balanci
ng #------------------------------------------------------------------------------- # MA
STER/SLAVE MODE #---------------------------------------------------------------------
-------- master_slave_mode = on # Activate master/slave mode # (change requires restart
) master_slave_sub_mode = 'stream' # Master/slave sub mode # Valid values are combinati
ons stream, slony # or logical. Default is stream. # (change requires restart) # - Stre
aming - sr_check_period = 3 # Streaming replication check period # Disabled (0) by defa
ult sr_check_user = 'nobody' # Streaming replication check user # This is neccessary ev
en if you disable streaming # replication delay check by sr_check_period = 0 sr_check_p
assword = '' # Password for streaming replication check user # Leaving it empty will ma
```

```
ke Pgpool-II to first look for the # Password in pool_passwd file before using the empt
y password sr_check_database = 'postgres' # Database name for streaming replication che
ck delay_threshold = 512000 # Threshold before not dispatching query to standby node #
Unit is in bytes # Disabled (0) by default # - Special commands - follow_master_command
= '' # Executes this command after master failover # Special values: # %d = failed node
id # %h = failed node host name # %p = failed node port number # %D = failed node datab
ase cluster path # %m = new master node id # %H = new master node hostname # %M = old m
aster node id # %P = old primary node id # %r = new master port number # %R = new maste
r database cluster path # %N = old primary node hostname # %S = old primary node port n
umber # %% = '%' character #------------------------------------------------------------
------------------- # HEALTH CHECK GLOBAL PARAMETERS #--------------------------------
--------------------------------------------- health_check_period = 5 # Health check pe
riod # Disabled (0) by default health_check_timeout = 10 # Health check timeout # 0 mea
ns no timeout health_check_user = 'nobody' # Health check user health_check_password =
'' # Password for health check user # Leaving it empty will make Pgpool-II to first loo
k for the # Password in pool_passwd file before using the empty password health_check_d
atabase = '' # Database name for health check. If '', tries 'postgres' frist, health_ch
eck_max_retries = 60 # Maximum number of times to retry a failed health check before gi
ving up. health_check_retry_delay = 1 # Amount of time to wait (in seconds) between ret
ries. connect_timeout = 10000 # Timeout value in milliseconds before giving up to conne
ct to backend. # Default is 10000 ms (10 second). Flaky network user may want to increa
se # the value. 0 means no timeout. # Note that this value is not only used for health
check, # but also for ordinary conection to backend. #-----------------------------------
--------------------------------------------- # HEALTH CHECK PER NODE PARAMETERS (OPTIO
NAL) #------------------------------------------------------------------------------ #h
ealth_check_period0 = 0 #health_check_timeout0 = 20 #health_check_user0 = 'nobody' #hea
lth_check_password0 = '' #health_check_database0 = '' #health_check_max_retries0 = 0 #h
ealth_check_retry_delay0 = 1 #connect_timeout0 = 10000 #--------------------------------
--------------------------------------------- # FAILOVER AND FAILBACK #--------------
---------------------------------------------------------------- failover_command = ''
# Executes this command at failover # Special values: # %d = failed node id # %h = fail
ed node host name # %p = failed node port number # %D = failed node database cluster pa
th # %m = new master node id # %H = new master node hostname # %M = old master node id
# %P = old primary node id # %r = new master port number # %R = new master database clu
ster path # %N = old primary node hostname # %S = old primary node port number # %% = '
%' character failback_command = '' # Executes this command at failback. # Special value
s: # %d = failed node id # %h = failed node host name # %p = failed node port number #
%D = failed node database cluster path # %m = new master node id # %H = new master node
hostname # %M = old master node id # %P = old primary node id # %r = new master port nu
mber # %R = new master database cluster path # %N = old primary node hostname # %S = ol
d primary node port number # %% = '%' character failover_on_backend_error = off # Initi
ates failover when reading/writing to the # backend communication socket fails # If set
to off, pgpool will report an # error and disconnect the session. detach_false_primary
= off # Detach false primary if on. Only # valid in streaming replicaton # mode and wit
h PostgreSQL 9.6 or # after. search_primary_node_timeout = 300 # Timeout in seconds to
search for the # primary node when a failover occurs. # 0 means no timeout, keep search
ing # for a primary node forever. #-------------------------------------------------
-------------------------- # ONLINE RECOVERY #-----------------------------------------
------------------------------------- recovery_user = 'nobody' # Online recovery user r
ecovery_password = '' # Online recovery password # Leaving it empty will make Pgpool-II
to first look for the # Password in pool_passwd file before using the empty password re
covery_1st_stage_command = '' # Executes a command in first stage recovery_2nd_stage_co
mmand = '' # Executes a command in second stage recovery_timeout = 90 # Timeout in seco
nds to wait for the # recovering node's postmaster to start up # 0 means no wait client
 idle limit in recovery = 0 # Client is disconnected after being idle # for that many s
```

```
_idle_limit_in_recovery = 0 # client is disconnected after being idle # for that many s
econds in the second stage # of online recovery # 0 means no disconnection # -1 means i
mmediate disconnection auto_failback = off # Dettached backend node reattach automatica
lly # if replication_state is 'streaming'. auto_failback_interval = 60 # Min interval o
f executing auto_failback in # seconds. #------------------------------------------------
-------------------------------- # WATCHDOG #-----------------------------------------------
------------------------------------ # - Enabling - use_watchdog = off # Activates watc
hdog # (change requires restart) # -Connection to up stream servers - trusted_servers =
'' # trusted server list which are used # to confirm network connection # (hostA,hostB,
hostC,...) # (change requires restart) ping_path = '/bin' # ping command path # (change
requires restart) # - Watchdog communication Settings - wd_hostname = '' # Host name or
IP address of this watchdog # (change requires restart) wd_port = 9000 # port number fo
r watchdog service # (change requires restart) wd_priority = 1 # priority of this watch
dog in leader election # (change requires restart) wd_authkey = '' # Authentication key
for watchdog communication # (change requires restart) wd_ipc_socket_dir = '/tmp' # Uni
x domain socket path for watchdog IPC socket # The Debian package defaults to # /var/ru
n/postgresql # (change requires restart) # - Virtual IP control Setting - delegate_IP =
'' # delegate IP address # If this is empty, virtual IP never bring up. # (change requi
res restart) if_cmd_path = '/sbin' # path to the directory where if_up/down_cmd exists
# If if_up/down_cmd starts with "/", if_cmd_path will be ignored. # (change requires re
start) if_up_cmd = '/usr/bin/sudo /sbin/ip addr add $_IP_$/24 dev eth0 label eth0:0' #
startup delegate IP command # (change requires restart) if_down_cmd = '/usr/bin/sudo /s
bin/ip addr del $_IP_$/24 dev eth0' # shutdown delegate IP command # (change requires r
estart) arping_path = '/usr/sbin' # arping command path # If arping_cmd starts with "/"
, if_cmd_path will be ignored. # (change requires restart) arping_cmd = '/usr/bin/sudo
/usr/sbin/arping -U $_IP_$ -w 1 -I eth0' # arping command # (change requires restart) #
- Behaivor on escalation Setting - clear_memqcache_on_escalation = on # Clear all the q
uery cache on shared memory # when standby pgpool escalate to active pgpool # (= virtua
l IP holder). # This should be off if client connects to pgpool # not using virtual IP.
# (change requires restart) wd_escalation_command = '' # Executes this command at escal
ation on new active pgpool. # (change requires restart) wd_de_escalation_command = '' #
Executes this command when master pgpool resigns from being master. # (change requires
restart) # - Watchdog consensus settings for failover - failover_when_quorum_exists = o
n # Only perform backend node failover # when the watchdog cluster holds the quorum # (
change requires restart) failover_require_consensus = on # Perform failover when majori
ty of Pgpool-II nodes # aggrees on the backend node status change # (change requires re
start) allow_multiple_failover_requests_from_node = off # A Pgpool-II node can cast mul
tiple votes # for building the consensus on failover # (change requires restart) enable
_consensus_with_half_votes = off # apply majority rule for consensus and quorum computa
tion # at 50% of votes in a cluster with even number of nodes. # when enabled the exist
ence of quorum and consensus # on failover is resolved after receiving half of the # to
tal votes in the cluster, otherwise both these # decisions require at least one more vo
te than # half of the total votes. # (change requires restart) # - Lifecheck Setting -
# -- common -- wd_monitoring_interfaces_list = '' # Comma separated list of interfaces
names to monitor. # if any interface from the list is active the watchdog will # consid
er the network is fine # 'any' to enable monitoring on all interfaces except loopback #
'' to disable monitoring # (change requires restart) wd_lifecheck_method = 'heartbeat'
# Method of watchdog lifecheck ('heartbeat' or 'query' or 'external') # (change require
s restart) wd_interval = 10 # lifecheck interval (sec) > 0 # (change requires restart)
# -- heartbeat mode -- wd_heartbeat_port = 9694 # Port number for receiving heartbeat s
ignal # (change requires restart) wd_heartbeat_keepalive = 2 # Interval time of sending
heartbeat signal (sec) # (change requires restart) wd_heartbeat_deadtime = 30 # Deadtim
e interval for heartbeat signal (sec) # (change requires restart) heartbeat_destination
0 = 'host0_ip1' # Host name or IP address of destination 0 # for sending heartbeat sign
al. # (change requires restart) heartbeat_destination_port0 = 9694 # Port number of des
```

```
tination 0 for sending # heartbeat signal. Usually this is the # same as wd_heartbeat_p
ort. # (change requires restart) heartbeat_device0 = '' # Name of NIC device (such like
'eth0') # used for sending/receiving heartbeat # signal to/from destination 0. # This w
orks only when this is not empty # and pgpool has root privilege. # (change requires re
start) #heartbeat_destination1 = 'host0_ip2' #heartbeat_destination_port1 = 9694 #heart
beat_device1 = '' # -- query mode -- wd_life_point = 3 # lifecheck retry times # (chang
e requires restart) wd_lifecheck_query = 'SELECT 1' # lifecheck query to pgpool from wa
tchdog # (change requires restart) wd_lifecheck_dbname = 'template1' # Database name co
nnected for lifecheck # (change requires restart) wd_lifecheck_user = 'nobody' # watchd
og user monitoring pgpools in lifecheck # (change requires restart) wd_lifecheck_passwo
rd = '' # Password for watchdog user in lifecheck # Leaving it empty will make Pgpool-I
I to first look for the # Password in pool_passwd file before using the empty password
# (change requires restart) # - Other pgpool Connection Settings - #other_pgpool_hostna
me0 = 'host0' # Host name or IP address to connect to for other pgpool 0 # (change requ
ires restart) #other_pgpool_port0 = 5432 # Port number for other pgpool 0 # (change req
uires restart) #other_wd_port0 = 9000 # Port number for other watchdog 0 # (change requ
ires restart) #other_pgpool_hostname1 = 'host1' #other_pgpool_port1 = 5432 #other_wd_po
rt1 = 9000 #------------------------------------------------------------------------
--- # OTHERS #----------------------------------------------------------------------
----- relcache_expire = 0 # Life time of relation cache in seconds. # 0 means no cache
expiration(the default). # The relation cache is used for cache the # query result agai
nst PostgreSQL system # catalog to obtain various information # including table structu
res or if it's a # temporary table or not. The cache is # maintained in a pgpool child
local memory # and being kept as long as it survives. # If someone modify the table by
using # ALTER TABLE or some such, the relcache is # not consistent anymore. # For this
purpose, cache_expiration # controls the life time of the cache. relcache_size = 8192 #
Number of relation cache # entry. If you see frequently: # "pool_search_relcache: cache
replacement happend" # in the pgpool log, you might want to increate this number. check
_temp_table = catalog # Temporary table check method. catalog, trace or none. # Default
is catalog. check_unlogged_table = on # If on, enable unlogged table check in SELECT st
atements. # This initiates queries against system catalog of primary/master # thus incr
eases load of master. # If you are absolutely sure that your system never uses unlogged
tables # and you want to save access to primary/master, you could turn this off. # Defa
ult is on. enable_shared_relcache = on # If on, relation cache stored in memory cache,
# the cache is shared among child process. # Default is on. # (change requires restart)
relcache_query_target = master # Target node to send relcache queries. Default is maste
r (primary) node. # If load_balance_node is specified, queries will be sent to load bal
ance node. #------------------------------------------------------------------------
--- # IN MEMORY QUERY MEMORY CACHE #------------------------------------------------
--------------------------- memory_cache_enabled = off # If on, use the memory cache fu
nctionality, off by default # (change requires restart) memqcache_method = 'shmem' # Ca
che storage method. either 'shmem'(shared memory) or # 'memcached'. 'shmem' by default
# (change requires restart) memqcache_memcached_host = 'localhost' # Memcached host nam
e or IP address. Mandatory if # memqcache_method = 'memcached'. # Defaults to localhost
. # (change requires restart) memqcache_memcached_port = 11211 # Memcached port number.
Mondatory if memqcache_method = 'memcached'. # Defaults to 11211. # (change requires re
start) memqcache_total_size = 67108864 # Total memory size in bytes for storing memory
cache. # Mandatory if memqcache_method = 'shmem'. # Defaults to 64MB. # (change require
s restart) memqcache_max_num_cache = 1000000 # Total number of cache entries. Mandatory
# if memqcache_method = 'shmem'. # Each cache entry consumes 48 bytes on shared memory.
# Defaults to 1,000,000(45.8MB). # (change requires restart) memqcache_expire = 0 # Mem
ory cache entry life time specified in seconds. # 0 means infinite life time. 0 by defa
ult. # (change requires restart) memqcache_auto_cache_invalidation = on # If on, invali
dation of query cache is triggered by corresponding # DDL/DML/DCL(and memqcache_expire)
```

```
. If off, it is only triggered # by memqcache_expire. on by default. # (change requires
restart) memqcache_maxcache = 409600 # Maximum SELECT result size in bytes. # Must be s
maller than memqcache_cache_block_size. Defaults to 400KB. # (change requires restart)
memqcache_cache_block_size = 1048576 # Cache block size in bytes. Mandatory if memqcach
e_method = 'shmem'. # Defaults to 1MB. # (change requires restart) memqcache_oiddir = '
/var/log/pgpool/oiddir' # Temporary work directory to record table oids # (change requi
res restart) white_memqcache_table_list = '' # Comma separated list of table names to m
emcache # that don't write to database # Regexp are accepted black_memqcache_table_list
= '' # Comma separated list of table names not to memcache # that don't write to databa
se # Regexp are accepted
```

涉及修改的重要配置如下：

```
listen_addresses = '0.0.0.0' port = 8001 socket_dir = '/tmp' reserved_connections = 0 p
cp_listen_addresses = '' pcp_port = 9898 pcp_socket_dir = '/tmp' # - Backend Connection
Settings - backend_hostname0 = '127.0.0.1' # Host name or IP address to connect to for
backend 0 backend_port0 = 3389 # Port number for backend 0 backend_weight0 = 1 # Weight
for backend 0 (only in load balancing mode) backend_data_directory0 = '/data01/pg12_338
9/pg_root' # Data directory for backend 0 backend_flag0 = 'ALWAYS_MASTER' # Controls va
rious backend behavior # ALLOW_TO_FAILOVER, DISALLOW_TO_FAILOVER # or ALWAYS_MASTER bac
kend_application_name0 = 'server0' # walsender's application_name, used for "show pool_
nodes" command backend_hostname1 = '127.0.0.1' backend_port1 = 8002 backend_weight1 = 1
backend_data_directory1 = '/data01/pg12_8002/pg_root' backend_flag1 = 'DISALLOW_TO_FAIL
OVER' backend_application_name1 = 'server1' # - Authentication - enable_pool_hba = on #
Use pool_hba.conf for client authentication pool_passwd = 'pool_passwd' # File name of
pool_passwd for md5 authentication. # "" disables pool_passwd. # (change requires resta
rt) allow_clear_text_frontend_auth = off # Allow Pgpool-II to use clear text password a
uthentication # with clients, when pool_passwd does not # contain the user password # -
Concurrent session and pool size - num_init_children = 128 # Number of concurrent sessi
ons allowed # (change requires restart) max_pool = 4 # Number of connection pool caches
per connection # (change requires restart) # - Life time - child_life_time = 300 # Pool
exits after being idle for this many seconds child_max_connections = 0 # Pool exits aft
er receiving that many connections # 0 means no exit connection_life_time = 0 # Connect
ion to backend closes after being idle for this many seconds # 0 means no close client_
idle_limit = 0 # Client is disconnected after being idle for that many seconds # (even
inside an explicit transactions!) # 0 means no disconnection #-------------------------
----------------------------------------------------- # LOGS #-------------------------
----------------------------------------------------- # - Where to log - log_destinatio
n = 'syslog' # Where to log # Valid values are combinations of stderr, # and syslog. De
fault to stderr. log_connections = on # Log connections log_standby_delay = 'if_over_th
reshold' # Log standby delay # Valid values are combinations of always, # if_over_thres
hold, none #-----------------------------------------------------------------------------
--- # FILE LOCATIONS #-------------------------------------------------------------------
------------- pid_file_name = '/var/run/pgpool-II-12/pgpool.pid' # PID file name # Can
be specified as relative to the" # location of pgpool.conf file or # as an absolute pat
h # (change requires restart) logdir = '/tmp' # Directory of pgPool status file # (chan
ge requires restart) #-------------------------------------------------------------------
------------- # CONNECTION POOLING #---------------------------------------------------
--------------------------- connection_cache = on # Activate connection pools # (change
requires restart) # Semicolon separated list of queries # to be issued at the end of a
session # The default is for 8.3 and later reset_query_list = 'ABORT; DISCARD ALL' #---
------------------------------------------------------------------------------ # LOAD BALA
NCING MODE #-----------------------------------------------------------------------------
--- load_balance_mode = on # Activate load balancing mode # (change requires restart) i
gnore_leading_white_space = on # Ignore leading white spaces of each query white_functi
```

```
ghore_leading_white_space = on # Ignore leading white spaces of each query white_functi
on_list = '' # Comma separated list of function names # that don't write to database #
Regexp are accepted black_function_list = 'currval,lastval,nextval,setval' # Comma sepa
rated list of function names # that write to database # Regexp are accepted black_query
_pattern_list = '' # Semicolon separated list of query patterns # that should be sent t
o primary node # Regexp are accepted # valid for streaming replicaton mode only. databa
se_redirect_preference_list = '' # comma separated list of pairs of database and node i
d. # example: postgres:primary,mydb[0-4]:1,mydb[5-9]:2' # valid for streaming replicato
n mode only. app_name_redirect_preference_list = '' # comma separated list of pairs of
app name and node id. # example: 'psql:primary,myapp[0-4]:1,myapp[5-9]:standby' # valid
for streaming replicaton mode only. allow_sql_comments = off # if on, ignore SQL commen
ts when judging if load balance or # query cache is possible. # If off, SQL comments ef
fectively prevent the judgment # (pre 3.4 behavior). disable_load_balance_on_write = 't
ransaction' # Load balance behavior when write query is issued # in an explicit transac
tion. # Note that any query not in an explicit transaction # is not affected by the par
ameter. # 'transaction' (the default): if a write query is issued, # subsequent read qu
eries will not be load balanced # until the transaction ends. # 'trans_transaction': if
a write query is issued, # subsequent read queries in an explicit transaction # will no
t be load balanced until the session ends. # 'always': if a write query is issued, read
queries will # not be load balanced until the session ends. statement_level_load_balanc
e = off # Enables statement level load balancing #------------------------------------
---------------------------------------- # MASTER/SLAVE MODE #------------------------
------------------------------------------------------- master_slave_mode = on # Activat
e master/slave mode # (change requires restart) master_slave_sub_mode = 'stream' # Mast
er/slave sub mode # Valid values are combinations stream, slony # or logical. Default i
s stream. # (change requires restart) # - Streaming - sr_check_period = 3 # Streaming r
eplication check period # Disabled (0) by default sr_check_user = 'nobody' # Streaming
replication check user # This is neccessary even if you disable streaming # replication
delay check by sr_check_period = 0 sr_check_password = '' # Password for streaming repl
ication check user # Leaving it empty will make Pgpool-II to first look for the # Passw
ord in pool_passwd file before using the empty password sr_check_database = 'postgres'
# Database name for streaming replication check delay_threshold = 512000 # Threshold be
fore not dispatching query to standby node # Unit is in bytes # Disabled (0) by default
#------------------------------------------------------------------------------ # HEALT
H CHECK GLOBAL PARAMETERS #------------------------------------------------------------
------------------ health_check_period = 5 # Health check period # Disabled (0) by defa
ult health_check_timeout = 10 # Health check timeout # 0 means no timeout health_check_
user = 'nobody' # Health check user health_check_password = '' # Password for health ch
eck user # Leaving it empty will make Pgpool-II to first look for the # Password in poo
l_passwd file before using the empty password health_check_database = '' # Database nam
e for health check. If '', tries 'postgres' frist, health_check_max_retries = 60 # Maxi
mum number of times to retry a failed health check before giving up. health_check_retry
_delay = 1 # Amount of time to wait (in seconds) between retries. connect_timeout = 100
00 # Timeout value in milliseconds before giving up to connect to backend. # Default is
10000 ms (10 second). Flaky network user may want to increase # the value. 0 means no t
imeout. # Note that this value is not only used for health check, # but also for ordina
ry conection to backend. #-------------------------------------------------------------
----------------- # FAILOVER AND FAILBACK #-------------------------------------------
---------------------------------- failover_on_backend_error = off # Initiates failover
when reading/writing to the # backend communication socket fails # If set to off, pgpoo
l will report an # error and disconnect the session. relcache_expire = 0 # 建议结构变更后
，设置为1,然后reload然后再改回来．当然也可以直接设置为一个时间 # Life time of relation cache
in seconds. # 0 means no cache expiration(the default). # The relation cache is used fo
r cache the # query result against PostgreSQL system # catalog to obtain various inform
ation # including table structures or if it's a # temporary table or not. The cache is
```

```
# maintained in a pgpool child local memory # and being kept as long as it survives. #
If someone modify the table by using # ALTER TABLE or some such, the relcache is # not
consistent anymore. # For this purpose, cache_expiration # controls the life time of th
e cache. relcache_size = 8192 # Number of relation cache # entry. If you see frequently
: # "pool_search_relcache: cache replacement happend" # in the pgpool log, you might wa
nt to increase this number.
```

3. 配置pool_passwd密码文件。命令如下：

> **说明** 通过pgpool连接数据库时需要使用密码文件，可以理解为pgpool支持了PostgreSQL的认证协议。

```
cd /etc/pgpool-II-12 #用法 #pg_md5 --md5auth --username=username password #生成digoal, n
obody密码，自动写入pool_passwd。 pg_md5 --md5auth --username=digoal "xxxxxxx" pg_md5 --md
5auth --username=nobody "xxxxxxx"
```

4. 自动生成pool_passwd文件。命令如下：

```
cd /etc/pgpool-II-12 # cat pool_passwd digoal:md54dd55116da69d3d03bf2e3a1470564f9 nobod
y:md54240e76623e2511d607f431043a5d1c1
```

5. 配置pgpool_hba文件。命令如下：

```
cd /etc/pgpool-II-12 cp pool_hba.conf.sample pool_hba.conf vi pool_hba.conf host all al
l 0.0.0.0/0 md5
```

6. 配置pcp管理密码文件。命令如下：

> **说明** 这里是用来管理pgpool的密码和用户，不是数据库的用户和密码。

```
cd /etc/pgpool-II-12 # pg_md5 abc # 例如密码是abc。 900150983cd24fb0d6963f7d28e17f72 cp p
cp.conf.sample pcp.conf vi pcp.conf # USERID:MD5PASSWD manage:900150983cd24fb0d6963f7d2
8e17f72 #表示使用manage用户来管理pcp。
```

7. 启动pgpool。命令如下：

```
cd /etc/pgpool-II-12 pgpool -f ./pgpool.conf -a ./pool_hba.conf -F ./pcp.conf
```

> **说明** 查看pgpool日志的命令如下：
>
> ```
> less /var/log/messages
> ```

8. 通过pgpool连接数据库。命令如下：

```
psql -h 127.0.0.1 -p 8001 -U digoal postgres
```

```
[root@iZb[illegible]Z pgpool-II-12]# psql -h pgm-bp[illegible].pg.rds.aliyuncs.com -p 3433 -U digoal postgres
Password for user digoal:
psql (12.2, server 10.10)
Type "help" for help.

postgres=>
```

常见问题

- 如何测试读写分离是否成功？
  连接并查询pg_is_in_recovery()，然后断开重连再查询pg_is_in_recovery()，如果交替返回false和true，说明是交替将请求发送给了主库和从库，即读写分离成功。
- 使用pgpool会增加延迟吗？
  会小幅增加延迟，本文测试环境下约增加0.12毫秒延迟。
- pgpool的延迟检测和健康检测机制是什么？
  - pgpool不会将SQL请求发送给回放延迟（wal replay）大于设置值的只读节点，当只读节点延迟小于设置值后，才会再次发送。

    > **说明** 您可以连接主库查询当前数据库wal写入位置的Lsn 1，然后连接只读节点查询当前wal replay位置的Lsn 2，对比Lsn 1和Lsn 2相差的字节。

  - pgpool可以检测后端的健康状态，如果发现不健康，SQL请求不会路由到这个节点。
- 如何停止、重新加载pgpool配置？
  您可以使用 `pgpool --help` 查看帮助命令，例如：

  ```
  cd /etc/pgpool-II-12 pgpool -f ./pgpool.conf -m fast stop
  ```

- 如果有多个只读实例，应该如何配置？
  修改pgpool.conf文件，补充多个只读实例的配置，示例如下：

  ```
  backend_hostname1 = 'xx.xx.xxx.xx' backend_port1 = 8002 backend_weight1 = 1 backend_data_directory1 = '/data01/pg12_8002/pg_root' backend_flag1 = 'DISALLOW_TO_FAILOVER' backend_application_name1 = 'server1' backend_hostname2 = 'xx.xx.xx.xx' backend_port1 = 8002 backend_weight1 = 1 backend_data_directory1 = '/data01/pg12_8002/pg_root' backend_flag1 = 'DISALLOW_TO_FAILOVER' backend_application_name1 = 'server1'
  ```

- 如何通过pcp查询后端状态？
  示例命令如下：

  ```
  # pcp_node_info -U manage -h /tmp -p 9898 -n 1 -v Password: 输入密码 Hostname : 127.0.0.1 Port : 8002 Status : 2 Weight : 0.500000 Status Name : up Role : standby Replication Delay : 0 Replication State : Replication Sync State : Last Status Change : 2020-02-29 00:20:29
  ```

- 监听的端口有哪些？
  监听的端口如下：
  - 主库：3389
  - 备库：8002
  - pgpool：8001
  - pcp：9898

29.5. 用户喜好推荐系统（PostgreSQL近似计算应用）

本文介绍如何使用hll插件实现根据用户喜好推荐相关内容。

背景信息

推荐系统在互联网应用中主要用于提升用户粘性、提高转化率，常见场景如下：

- 电商网站根据用户购买习惯，推荐商品。
- 音乐网站根据用户收听喜好，推荐音乐。
- 新闻网站根据用户浏览习惯，推荐新闻。
- 应用网站根据用户下载和使用应用习惯，推荐应用。

本文以音乐网站为例介绍如何设计推荐系统数据库，以及不同设计方法的差异。

设计背景

1. 用户（uid）完整听完的歌曲（vid），该歌曲有对应的标签（tags），同时一个歌曲可以有多个标签，由此形成映射关系：

```
uid ->> tags ->> musics
```

2. 根据用户每个tag下的歌曲数排行，得到tag热度：

```
tag(count distinct music) ...
```

3. 统计前5个tag及权重：

```
tag1:40% tag2:20% tag3:15% tag4:15% tag5:10%
```

4. 从tag标签的歌曲库中，排除用户听过的，然后根据这些歌曲的推荐权重（例如播放次数倒排），按比例推荐新的歌曲。

普通设计

该设计适合所有数据库，但是缺点是当数据量庞大时，可能会使用聚合查询，效率较低。示例如下：

```
create table t_like( uid int, -- 用户ID tagid int, -- 歌曲标签ID vid int, -- 歌曲ID mod_time
timestamp, -- 最后一次更新时间，仅与上次时间超过1天时更新。 primary key (uid,tagid,vid) ); inser
t into t_like values (:uid, :tagid, :vid, :mod_time) on conflict (uid,tagid,vid) do update
set mod_time=excluded.mod_time where excluded.mod_time - t_like.mod_time > interval '1 day'
; -- 统计最近1天的top 10的tag。 select tagid, count(*) from t_like where uid=:uid and now()-m
od_time < interval '1 day' group by tagid order by count(*) desc limit 10;
```

压力测试示例如下：

```
vi test.sql \set uid random(1,50000) \set tagid random(1,5000) \set vid random(1,10000000)
insert into t_like values (:uid, :tagid, :vid, now()) on conflict (uid,tagid,vid) do update
set mod_time=excluded.mod_time where excluded.mod_time - t_like.mod_time > interval '1 day'
; pgbench -M prepared -n -r -P 1 -f ./test.sql -c 32 -j 32 -T 240 transaction type: ./test.
sql scaling factor: 1 query mode: prepared number of clients: 32 number of threads: 32 dura
tion: 240 s number of transactions actually processed: 80975327 latency average = 0.095 ms
latency stddev = 0.340 ms tps = 337396.279382 (including connections establishing) tps = 33
7406.018908 (excluding connections establishing) statement latencies in milliseconds: 0.000
\set uid random(1,50000) 0.000 \set tagid random(1,5000) 0.000 \set vid random(1,10000000)
0.094 insert into t_like values (:uid, :tagid, :vid, now()) db1=# select tagid, count(*) fr
om t_like where uid=1 and now()-mod_time < interval '1 day' group by tagid order by count(*
) desc limit 10; tagid | count -------+------- 2519 | 4 3049 | 4 3648 | 4 1777 | 3 1352 | 3
1491 | 3 1064 | 3 572 | 3 692 | 3 301 | 3 (10 rows) Time: 3.947 ms
```

基于hll近似计算的设计

使用hll来存储用户（uid）听完的歌曲ID（vid），相比普通设计有如下优势：

- 数据量小，使用近似hll hash聚集代替真实值。
- 查询效率高，支持索引，不需要计算，毫秒响应。
- 支持hash union、add等操作，适合滑窗计算，满足更多业务需求。

> **说明** hll插件使用方法请参见 基数统计（hll）。

1. 每个标签存储一个hll，hll里面存储的是该标签内用户完整听完的歌曲的ID hash值。

```
create table t_like ( uid int, tagid int, -- 标签 w1 hll, w1_mod_time timestamp, -- 周一听完的歌曲对应的vid构成的hash。 w2 hll, w2_mod_time timestamp, -- 周二听完的歌曲对应的vid构成的hash。 w3 hll, w3_mod_time timestamp, -- 周三听完的歌曲对应的vid构成的hash。 w4 hll, w4_mod_time timestamp, -- 周四听完的歌曲对应的vid构成的hash。 w5 hll, w5_mod_time timestamp, -- 周五听完的歌曲对应的vid构成的hash。 w6 hll, w6_mod_time timestamp, -- 周六听完的歌曲对应的vid构成的hash。 w7 hll, w7_mod_time timestamp, -- 周日听完的歌曲对应的vid构成的hash。 whole hll, -- 所有 primary key (uid,tagid) );
```

> **说明** w1~w7是根据业务设置的，如果业务只关心1天的数据，就只需要设置一个字段。

2. 当用户听完一首歌，把这个写入当前日期对应字段，当对应字段已经有value，并且最后修改时间不是今天时，覆盖该value，否则追加hash。使用的是 `insert into on conflict` 语法，示例如下：

```
-- 设置观看历史行为hash。 insert into t_like ( uid, tagid, w5, w5_mod_time, whole ) values ( 1, -- uid 200, -- 标签ID hll_hash_integer(12346)||hll_empty(), -- 观看过的vid, 多个则继续。 now(), hll_hash_integer(12346)||hll_empty() -- 观看过的vid ) on conflict (uid,tagid) do update set w5= case when date(t_like.w5_mod_time) <> current_date then excluded.w5 else hll_union(coalesce(t_like.w5,hll_empty()), excluded.w5) end, w5_mod_time = excluded.w5_mod_time, whole = hll_union(coalesce(t_like.whole,hll_empty()), excluded.whole) where hll_union(coalesce(t_like.w5,hll_empty()), excluded.w5) <> coalesce(t_like.w5,hll_empty()) or hll_union(coalesce(t_like.whole,hll_empty()), excluded.whole) <> coalesce(t_like.whole,hll_empty()) ;
```

> **说明** 实际业务中也可以批量合并更新，针对单个用户的单个标签聚合更新，使用hll union降低更新率。

3. 查询uid 1最近2天的top 10标签，示例如下：

```
select tagid, hll_cardinality( hll_union(coalesce(w4,hll_empty()), coalesce(w5,hll_empty())) ) as vids from t_like where uid = 1 order by 2 desc limit 10; tagid | vids ------+------ 200 | 2 (1 row)
```

4. 创建索引，示例如下：

```
create index idx_t_like_1 on t_like (uid, hll_cardinality( hll_union(coalesce(w4,hll_empty()), coalesce(w5,hll_empty())) ));
```

5. 查看执行计划，示例如下：

```
postgres=# explain select tagid, hll_cardinality( hll_union(coalesce(w4,hll_empty()), coalesce(w5,hll_empty())) ) as vids from t_like where uid = 1 order by 2 desc limit 10; QUERY PLAN ---------------------------------------------------------------------------------------------- Limit (cost=0.11..0.15 rows=1 width=12) -> Index Scan Backward using idx_t_like_1 on t_like (cost=0.11..0.15 rows=1 width=12) Index Cond: (uid = 1) (3 rows)
```

6. 写入几千万数据，进行压力测试，示例如下：

```
vi test.sql \set uid random(1,50000) \set tagid random(1,5000) \set vid random(1,100000
00) insert into t_like ( uid, tagid, w5, w5_mod_time, whole ) values ( :uid, :tagid, hl
l_hash_integer(:vid)||hll_empty(), now(), hll_hash_integer(:vid)||hll_empty() ) on conf
lict (uid,tagid) do update set w5= case when date(t_like.w5_mod_time) <> current_date t
hen excluded.w5 else hll_union(coalesce(t_like.w5,hll_empty()), excluded.w5) end, w5_mo
d_time = excluded.w5_mod_time, whole = hll_union(coalesce(t_like.whole,hll_empty()), ex
cluded.whole) where hll_union(coalesce(t_like.w5,hll_empty()), excluded.w5) <> coalesce
(t_like.w5,hll_empty()) or hll_union(coalesce(t_like.whole,hll_empty()), excluded.whole
) <> coalesce(t_like.whole,hll_empty()) ; pgbench -M prepared -n -r -P 1 -c 32 -j 32 -T
120 -f ./test.sql
```

测试结果如下：

```
transaction type: ./test.sql scaling factor: 1 query mode: prepared number of clients:
32 number of threads: 32 duration: 120 s number of transactions actually processed: 246
36321 latency average = 0.156 ms latency stddev = 0.339 ms tps = 205301.110313 (includi
ng connections establishing) tps = 205354.851711 (excluding connections establishing) s
tatement latencies in milliseconds: 0.001 \set uid random(1,5000000) 0.001 \set tagid r
andom(1,5000) 0.000 \set vid random(1,10000000) 0.154 insert into t_like (
```

7. 查询某个uid的标签热度排序，示例如下：

```
postgres=# select tagid, hll_cardinality( hll_union(coalesce(w4,hll_empty()), coalesce(
w5,hll_empty())) ) as vids from t_like where uid = 1 order by 2 desc limit 10; tagid |
vids -------+------ 200 | 2 1413 | 1 1996 | 1 2642 | 1 3664 | 1 4340 | 1 (6 rows) Time:
0.688 ms
```

> **说明** 响应时间为0.688毫秒。

其他需求，例如过滤用户已经听过的歌曲：

- 判断一个歌曲ID（vid）是否在这个hash里面，不精确操作。示例如下：

```
postgres=# select whole || hll_hash_integer(1) = whole from t_like where uid=1 and tagid=
200; -- 返回false表示不包含vid:1。 ?column? ---------- f (1 row) postgres=# select whole ||
hll_hash_integer(12345) = whole from t_like where uid=1 and tagid=200; -- 返回true表示包含v
id:12345。 ?column? ---------- t (1 row)
```

- 判断一个歌曲ID（vid）是否在这个hash里面，精确操作。建表语句示例如下：

```
create table t_like_lossless ( uid int, vid int, primary key (uid,vid) );
```

> **说明** primary key查询速度非常快。

29.6. 将RDS PostgreSQL作为RDS MySQL实时分析库

本文介绍如何将RDS MySQL数据实时迁移到RDS PostgreSQL，使RDS PostgreSQL成为RDS MySQL的实时分析库。

背景信息

当需要使用RDS PostgreSQL数据库特有的功能来分析MySQL中的数据，使用GIS处理时空数据或进行用户画像分析时，可以使用DTS功能将RDS MySQL中的数据实时迁移到RDS PostgreSQL数据库中，使PostgreSQL数据库作为MySQL的实时分析库。

前提条件

- 已创建源RDS MySQL实例，创建方法，请参见创建RDS MySQL实例。
- 已创建目标RDS PostgreSQL实例，创建方法，请参见创建RDS PostgreSQL实例。
- 目标RDS PostgreSQL实例存储空间须大于源RDS MySQL实例占用的存储空间。

配置步骤

1. 准备源数据库测试数据。
2. 创建迁移任务。
3. MySQL数据实时迁移至PostgreSQL。

准备源数据库测试数据

1. 连接源数据库RDS MySQL。

```
mysql -h <连接地址> -u <用户名> -P <端口> -p
```

2. 创建测试数据库db1。

```
CREATE DATABASE db1;
```

3. 进入db1数据库。

```
USE db1;
```

4. 创建测试数据表test_mm和test_innodb。

```
CREATE TABLE `test_mm` ( `id` INT (11) NOT NULL AUTO_INCREMENT, `user_id` VARCHAR (20) NOT NULL, `group_id` INT (11) NOT NULL, `create_time` datetime NOT NULL, PRIMARY KEY (`id`),KEY `index_user_id` (`user_id`) USING HASH ) ENGINE = innodb AUTO_INCREMENT = 1 DEFAULT CHARSET = utf8;
```

```
CREATE TABLE `test_innodb` ( `id` INT (11) NOT NULL AUTO_INCREMENT, `user_id` VARCHAR (20) NOT NULL, `group_id` INT (11) NOT NULL, `create_time` datetime NOT NULL, PRIMARY KEY (`id`), KEY `index_user_id` (`user_id`) USING HASH ) ENGINE = innodb AUTO_INCREMENT = 1 DEFAULT CHARSET = utf8;
```

5. 创建随机字符串函数。

```
delimiter $$ CREATE FUNCTION rand_string(n int) RETURNS varchar(255) begin declare chars_str varchar(100) default "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"; declare return_str varchar(255) default ""; declare i int default 0; while i < n do set return_str=concat(return_str,substring(chars_str,floor(1+rand()*62),1)); set i=i+1; end while; return return_str; end $$ delimiter ;
```

6. 创建插入测试数据的存储过程。

```
delimiter $$ CREATE PROCEDURE `insert_data`(IN n int) BEGIN DECLARE i INT DEFAULT 1; WHILE (i <= n ) DO INSERT into test_mm (user_id,group_id,create_time ) VALUEs (rand_string(20),FLOOR(RAND() * 100) ,now() ); set i=i+1; END WHILE; END $$ delimiter ;
```

7. 调用存储过程。

```
CALL insert_data(1000000); INSERT INTO test_innodb SELECT * FROM test_mm;
```

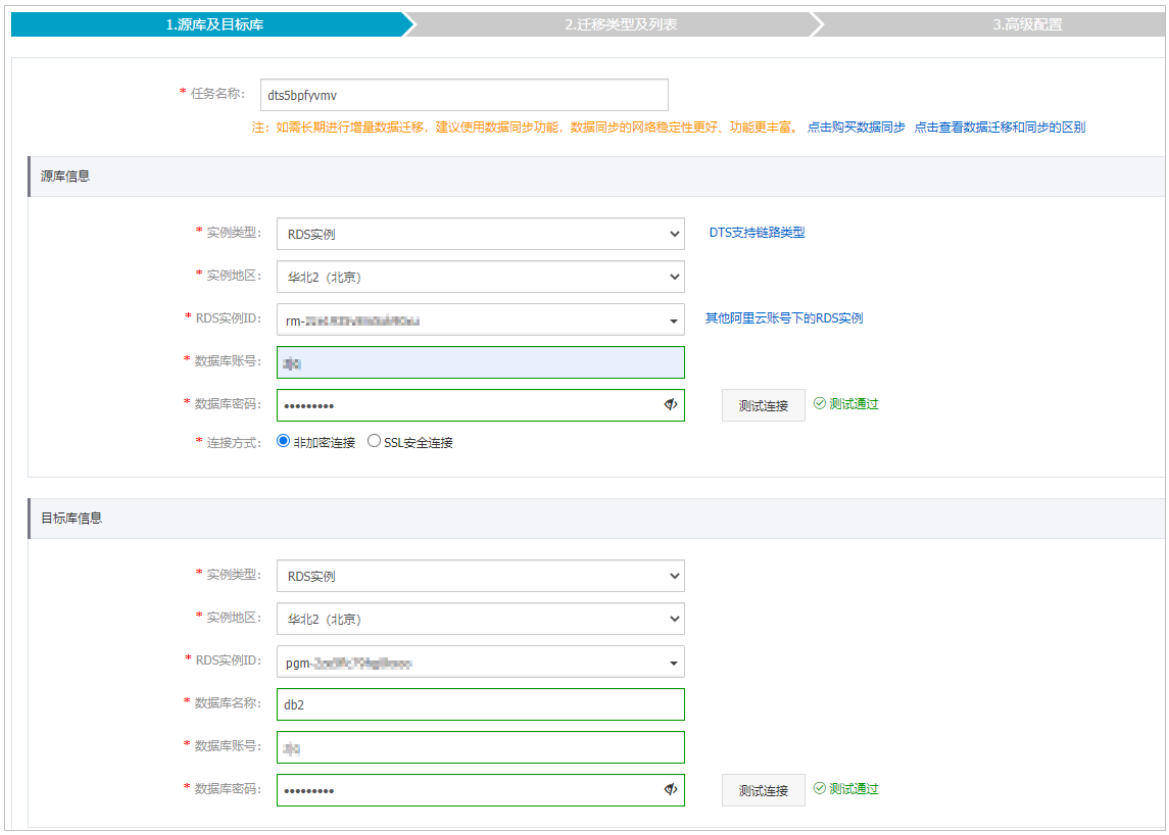
创建迁移任务

1. 创建迁移任务前，需要在RDS PostgreSQL控制台创建数据库，用于从RDS MySQL接收数据。创建方法，请参见创建数据库。

> **说明** 本示例中创建的数据库名为db2。

2. 登录数据传输控制台。
3. 在左侧导航栏，单击**数据迁移**。
4. 在页面顶部的**迁移任务列表**中选择目标实例所属地域。

5. 在页面右上角，单击**创建迁移任务**。
6. 配置源库及目标库信息。



类别	配置	说明
无	**任务名称**	DTS会自动生成一个任务名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。

类别	配置	说明
源库信息	**数据库类型**	选择**RDS实例**。
	实例地区	选择源RDS MySQL实例所属地域。
	实例ID	择源RDS MySQL实例ID。
	数据库账号	填入源RDS MySQL实例的数据库账号。账号需要具备以下权限： ○ 库表结构迁移：SELECT权限。 ○ 全量迁移：SELECT权限。 ○ 增量迁移：REPLICATION CLIENT、REPLICATION SLAVE、SHOW VIEW和SELECT权限。
	数据库密码	填入该数据库账号对应的密码。
	连接方式	根据需求选择**非加密连接**或**SSL安全连接**。如果设置为**SSL安全连接**，您需要提前开启RDS MySQL实例的SSL加密功能，详情请参见设置SSL加密。
目标库信息	**数据库类型**	选择**RDS实例**。
	实例地区	选择目标RDS PostgreSQL实例所属地域。
	RDS实例ID	选择目标RDS PostgreSQL实例ID。
	数据库账号	填入目标RDS PostgreSQL实例的数据库账号，需具备以下权限： ○ LOGIN权限。 ○ 目标库的CONNECT、CREATE权限。 ○ 目标Schema的CREATE权限。
	数据库密码	填入该数据库账号对应的密码。

7. 配置完成后，需要分别单击源库和目标库 数据库密码 参数后的**测试连接**，测试成功后，再进行下一步。
8. 单击页面右下角**授权白名单并进入下一步**。
9. 配置迁移类型及列表。

配置	说明
迁移类型	选择结构迁移、全量数据迁移和增量数据迁移。
迁移对象	选择待迁移的表。本示例为test_innodb和test_mm。
已选择对象	此处展示已选择待迁移的表。
映射名称更改	默认为**不进行库表名称批量修改**，如果需要对库表名称进行批量修改，可在选择**要进行库表名称批量更改**后，单击页面右下角**高级设置**，批量修改数据库表名称。
源库、目标库无法连接后的重试时间	默认为720分钟，无需修改。

10. 上述配置完成后，单击页面下方的**预检查并启动**。

> **说明**
> - 在迁移作业正式启动之前，会先进行预检查。只有预检查通过后，才能成功启动迁移作业。
> - 如果预检查失败，单击具体检查项后的 ⓘ，查看失败详情。
> - 您可以根据提示修复后重新进行预检查。
> - 如无需修复告警检测项，您也可以选择**确认屏蔽**、**忽略告警项并重新进行预检查**，跳过告警检测项重新进行预检查。

11. 当预检查页面中显示为**预检查通过100%**时，单击**下一步**。

12. 在**购买配置确认**页面，选择数据迁移实例的链路规格。

> **说明** DTS为您提供了不同性能的迁移规格，迁移链路规格的不同会影响迁移速率，您可以根据业务场景进行选择，详情请参见 数据迁移链路规格说明。

13. 勾选**《数据传输（按量付费）服务条款》**。
14. 单击**购买并启动**，迁移任务正式开始，您可在任务列表查看具体任务进度。

MySQL数据实时迁移至PostgreSQL

1. 查看当前全量迁移状态。

i. 连接RDS PostgreSQL数据库。

```
psql -h <数据库连接地址> -U <用户名> -p <端口号> -d db2
```

> **说明** 由于在 创建迁移任务步骤中创建了数据库db2来接收MySQL迁移的数据，因此，连接PostgreSQL数据库时， 数据库名 直接配置成db2。

ii. 执行 `\dn` 命令，查看是否将MySQL数据库db1映射为PostgreSQL中的Schema。

```
 List of schemas Name | Owner --------+------------ db1 | test1 public | pg*******
(2 rows)
```

> **说明** MySQL中的数据库名称，在迁移到PostgreSQL数据库后，将会映射到PostgreSQL中的Schema。

iii. 执行 `\dt+ db1.*` 命令，查看db1中的表状态。

```
 List of relations Schema | Name | Type | Owner | Persistence | Size | Description
--------+-------------+-------+-------+-------------+-------+------------- db1 | te
st_innodb | table | test1 | permanent | 65 MB | db1 | test_mm | table | test1 | per
manent | 65 MB | (2 rows)
```

iv. 执行如下命令，分别查询test_innodb和test_mm数据表记录数。

- 查询test_innodb记录数。

```
SELECT COUNT(*) FROM db1.test_innodb;
```

```
 count --------- 1000000 (1 row)
```

- 查询test_mm记录数。

```
SELECT COUNT(*) FROM db1.test_mm;
```

```
 count --------- 1000000 (1 row)
```

说明

由于MySQL数据库在PostgreSQL中被映射为Schema，在PostgreSQL中查询db1中数据时，需要指定Schema。

如果您不想每次查询db1数据时都指定Schema，可以设置 search_path 参数。

a. 查看 `search_path` 。

```
show search_path;
```

```
 search_path ----------------- "$user", public (1 row)
```

b. 设置 `search_path` 。

```
set search_path = db1, "$user", public;
```

```
SET
```

c. 查看修改结果。

```
show search_path;
```

```
 search_path ---------------------- db1, "$user", public (1 row)
```

2. 新增数据实时迁移测试。

i. RDS MySQL中新增数据。

```
INSERT INTO test_innodb (user_id, group_id, `create_time`) VALUES ('testuser', 1, '
2021-07-29 12:00:00');
```

执行结果

```
SELECT * FROM test_Innodb WHERE user_id = 'testuser';
```

```
+---------+----------+----------+---------------------+ | id | user_id | group_id |
create_time | +---------+----------+----------+---------------------+ | 1000001 | t
estuser | 1 | 2021-07-29 12:00:00 | +---------+----------+----------+--------------
-------+ 1 row in set (0.03 sec)
```

ii. RDS PostgreSQL中查看新增的记录是否迁移。

```
SELECT * FROM test_Innodb WHERE user_id = 'testuser';
```

```
 id | user_id | group_id | create_time ---------+----------+----------+------------
--------- 1000001 | testuser | 1 | 2021-07-29 12:00:00 (1 row)
```

3. 更新数据实时迁移测试。

i. RDS MySQL中更新数据。

```
UPDATE test_innodb set group_id = 2 WHERE user_id = 'testuser';
```

执行结果：

```
SELECT * FROM test_innodb WHERE user_id = 'testuser';
```

```
+---------+----------+----------+---------------------+ | id | user_id | group_id |
create_time | +---------+----------+----------+---------------------+ | 1000001 | t
estuser | 2 | 2021-07-29 12:00:00 | +---------+----------+----------+--------------
-------+ 1 row in set (0.03 sec)
```

ii. RDS PostgreSQL中查看更新的记录是否迁移。

```
SELECT * FROM test_innodb WHERE user_id = 'testuser';
```

```
 id | user_id | group_id | create_time ---------+----------+----------+------------
--------- 1000001 | testuser | 2 | 2021-07-29 12:00:00 (1 row)
```

4. 删除数据实时迁移测试。

i. RDS MySQL中删除数据。

```
DELETE FROM test_innodb WHERE user_id = 'testuser';
```

执行结果：

- 查询user_id为testuser的记录。

```
SELECT * FROM test_innodb WHERE user_id = 'testuser';
```

```
Empty set (0.03 sec)
```

- 查询MAX(id)。

```
SELECT MAX(id) FROM test_innodb;
```

```
+---------+ | MAX(id) | +---------+ | 1000000 | +---------+ 1 row in set (0.03 se
c)
```

> **说明** 未新增记录时，id最大为1000000，新增数据后，id自增到1000001，删除后，id又减少至1000000。

ii. RDS PostgreSQL中查看删除记录后数据是否迁移。

```
SELECT * FROM test_innodb WHERE user_id = 'testuser';
```

```
 id | user_id | group_id | create_time ----+---------+----------+------------- (0 rows)
```

```
SELECT MAX(id) FROM test_innodb;
```

```
 max --------- 1000000 (1 row)
```

29.7. 中间件ShardingSphere开发指南

RDS PostgreSQL支持中间件ShardingSphere，本文介绍如何基于ShardingSphere进行开发。

前提条件

RDS PostgreSQL所有版本实例均支持ShardingSphere。

背景信息

ShardingSphere适合分片彻底、数据库逻辑分片极其清晰的业务，支持如下功能：

- 数据分片
  - 分库和分表
  - 读写分离
  - 分片策略定制化
  - 无中心化分布式主键
- 分布式事务
  - 标准化事务接口
  - XA强一致事务
  - 柔性事务
- 数据库治理
  - 配置动态化
  - 编排和治理
  - 数据脱敏
  - 可视化链路追踪
  - 弹性伸缩（规划中）

更多详情请参见ShardingSphere官方手册。

ShardingSphere模式介绍

ShardingSphere支持多种模式，建议根据实际业务情况选择。各模式介绍如下表。

类别	Sharding-JDBC	Sharding-Proxy	Sharding-Sidecar
支持数据库	任意实现JDBC规范的数据库（MySQL、PostgreSQL、Oracle、SQL Server等）	MySQL、PostgreSQL	MySQL、PostgreSQL

类别	Sharding-JDBC	Sharding-Proxy	Sharding-Sidecar
连接消耗数	高	低	高
异构语言支持	仅Java	任意	任意
性能	损耗低	损耗略高	损耗低
无中心化	是	否	是
静态入口	无	有	无

设置配置模板

1. ECS实例上使用如下命令进入模板目录。

```
cd apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin cd conf
```

2. 使用ll命令查看目录内所有文件。

   结果示例如下：

```
total 24 -rw-r--r-- 1 501 games 3019 Jul 30 2019 config-encrypt.yaml -rw-r--r-- 1 501 g
ames 3582 Apr 22 2019 config-master_slave.yaml -rw-r--r-- 1 501 games 4278 Apr 22 2019
config-sharding.yaml -rw-r--r-- 1 501 games 1918 Jul 30 2019 server.yaml
```

   说明

   - config-encrypt.yaml：数据脱敏配置文件
   - config-master_slave.yaml：读写分离配置文件
   - config-sharding.yaml：数据分片配置文件
   - server.yaml：公共配置文件

3. 修改配置文件。

说明　配置文件的详细说明请参见ShardingSphere官方手册，本文仅以数据分片和公共配置进行简单说明。

- 数据分片示例

```
schemaName: #逻辑数据源名称 dataSources: #数据源配置，可配置多个data_source_name。
<data_source_name>: #与Sharding-JDBC配置不同，无需配置数据库连接池。 url: #数据库URL连接 username: #数据库用户名 password: #数据库密码 connectionTimeoutMilliseconds: 30000 #连接超时毫秒数 idleTimeoutMilliseconds: 60000 #空闲连接回收超时毫秒数 maxLifetimeMilliseconds: 1800000 #连接最大存活时间毫秒数 maxPoolSize: 65 #最大连接数 shardingRule: #省略数据分片配置，与Sharding-JDBC配置一致。
```

- 公共配置示例
  - Proxy属性

```
#省略与Sharding-JDBC一致的配置属性 props: acceptor.size: #用于设置接收客户端请求的工作线程个数，默认为CPU核数*2。 proxy.transaction.type: #默认为LOCAL事务，允许LOCAL，XA，BASE三个值，XA采用Atomikos作为事务管理器，BASE类型需要拷贝实现ShardingTransactionManager的接口的jar包至lib目录中。 proxy.opentracing.enabled: #是否开启链路追踪功能，默认为不开启。 check.table.metadata.enabled: #是否在启动时检查分表元数据一致性，默认值：false。 proxy.frontend.flush.threshold: # 对于单个大查询,每多少个网络包返回一次。
```

  - 权限验证

```
# 用于执行登录Sharding Proxy的权限验证。配置用户名、密码、可访问的数据库后，必须使用正确的用户名、密码才可登录Proxy。 authentication: users: root: # 自定义用户名 password: root # root用户对应的密码 sharding: # 自定义用户名 password: sharding # sharding用户对应的密码 authorizedSchemas: sharding_db, masterslave_db # 该用户授权可访问的数据库，多个用逗号分隔。缺省将拥有root权限，可访问全部数据库。
```

准备测试环境

- ECS实例安装Java测试环境

```
yum install -y java
```

- 配置PostgreSQL 12实例
  - 创建账号：r1
  - 账号对应密码：PW123321!
  - 创建数据库：db0、db1、db2、db3，并且owner为r1。
  - 实例白名单放通ECS的IP。

说明
- 创建方法请参见创建RDS PostgreSQL实例和创建数据库和账号。
- 放通ECS的IP请参见设置白名单。

数据库管理

数据库名	数据库状态	字符集	绑定账号
db0	运行中	UTF8	r1
db1	运行中	UTF8	r1
db2	运行中	UTF8	r1
db3	运行中	UTF8	r1

- 使用vi命令编辑公共配置文件。

```
vi /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/conf/server.yaml
```

- 参考如下示例，配置公共配置文件。

```
authentication: users: r1: password: PW123321! authorizedSchemas: db0,db1,db2,db3 props:
executor.size: 16 sql.show: false
```

水平分片测试

1. 修改数据分片配置文件。
   i. 使用vi命令打开公共配置文件。

```
vi /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/conf/config-shar
ding.yaml
```

   ii. 参考如下示例，配置公共配置文件。

```
schemaName: sdb dataSources: db0: url: jdbc:postgresql://pgm-bpxxxxx.pg.rds.aliyunc
s.com:1433/db0 username: r1 password: PW123321! connectionTimeoutMilliseconds: 3000
0 idleTimeoutMilliseconds: 60000 maxLifetimeMilliseconds: 1800000 maxPoolSize: 65 d
b1: url: jdbc:postgresql://pgm-bpxxxxx.pg.rds.aliyuncs.com:1433/db1 username: r1 pa
ssword: PW123321! connectionTimeoutMilliseconds: 30000 idleTimeoutMilliseconds: 600
00 maxLifetimeMilliseconds: 1800000 maxPoolSize: 65 db2: url: jdbc:postgresql://pgm
-bpxxxxx.pg.rds.aliyuncs.com:1433/db2 username: r1 password: PW123321! connectionTi
meoutMilliseconds: 30000 idleTimeoutMilliseconds: 60000 maxLifetimeMilliseconds: 18
00000 maxPoolSize: 65 db3: url: jdbc:postgresql://pgm-bpxxxxx.pg.rds.aliyuncs.com:1
433/db3 username: r1 password: PW123321! connectionTimeoutMilliseconds: 30000 idleT
imeoutMilliseconds: 60000 maxLifetimeMilliseconds: 1800000 maxPoolSize: 65 sharding
Rule: tables: t_order: actualDataNodes: db${0..3}.t_order${0..7} databaseStrategy:
inline: shardingColumn: user_id algorithmExpression: db${user_id % 4} tableStrategy
: inline: shardingColumn: order_id algorithmExpression: t_order${order_id % 8} keyG
enerator: type: SNOWFLAKE column: order_id t_order_item: actualDataNodes: db${0..3}
.t_order_item${0..7} databaseStrategy: inline: shardingColumn: user_id algorithmExp
ression: db${user_id % 4} tableStrategy: inline: shardingColumn: order_id algorithm
Expression: t_order_item${order_id % 8} keyGenerator: type: SNOWFLAKE column: order
_item_id bindingTables: - t_order,t_order_item defaultTableStrategy: none:
```

2. 启动ShardingSphere，监听8001端口。

```
cd ~/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/bin/ ./start.sh 8001
```

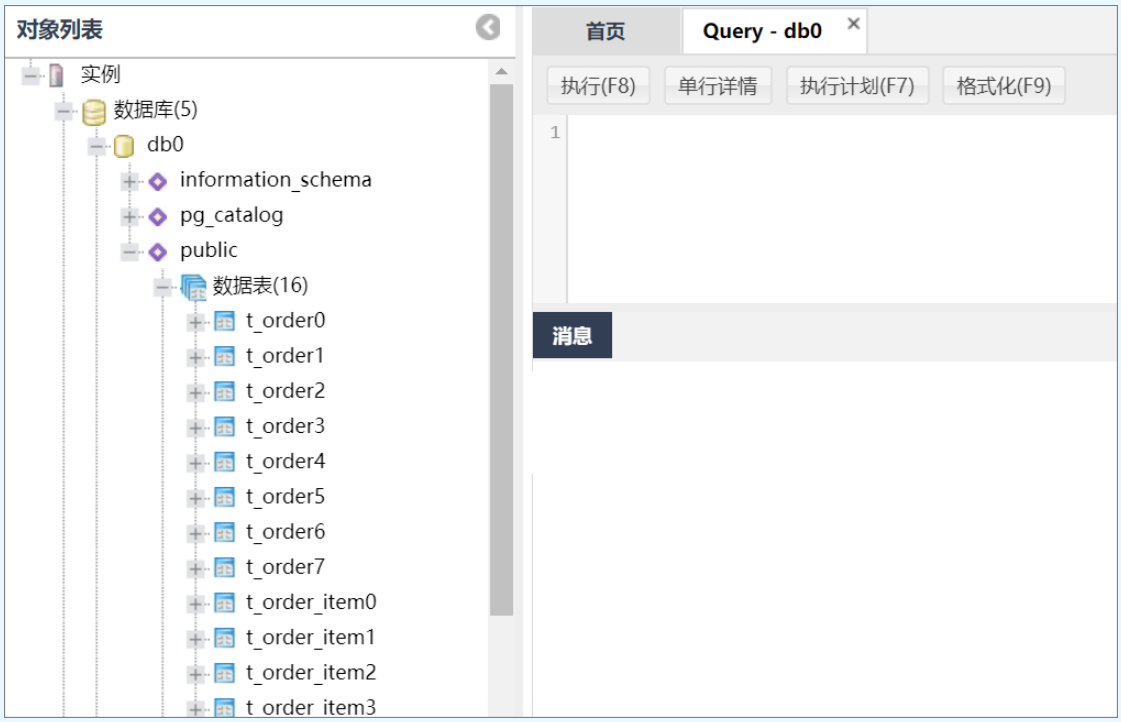
3. 测试连接数据库。

```
psql -h 127.0.0.1 -p 8001 -U r1 sdb
```

4. 创建表。

```
create table t_order(order_id int8 primary key, user_id int8, info text, c1 int, crt_time timestamp); create table t_order_item(order_item_id int8 primary key, order_id int8, user_id int8, info text, c1 int, c2 int, c3 int, c4 int, c5 int, crt_time timestamp);
```

说明 创建表会根据配置的策略，在目标库中自动创建水平分片。



常见问题

- 如果想知道ShardingSphere的解析SQL、路由SQL的语句，请参见如下配置：
  - 使用vi命令打开公共配置文件。

```
vi /root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/conf/server.yaml
```

  - 参考如下示例，配置公共配置文件。

```
authentication: users: r1: password: PW123321! authorizedSchemas: db0,db1,db2,db3 props: executor.size: 16 sql.show: true # 表示把解析的SQL打印到日志
```

- 如果需要做写入、查询等测试，请参见如下命令：

- 写入示例

```
insert into t_order (user_id, info, c1, crt_time) values (0,'a',1,now()); insert into t
_order (user_id, info, c1, crt_time) values (1,'b',2,now()); insert into t_order (user_
id, info, c1, crt_time) values (2,'c',3,now()); insert into t_order (user_id, info, c1,
crt_time) values (3,'c',4,now());
```

- 查询示例1

```
select * from t_order;
```

结果示例

```
 order_id | user_id | info | c1 | crt_time --------------------+---------+------+----+-
--------------------------- 433352561047633921 | 0 | a | 1 | 2020-02-09 19:48:21.856555
433352585668198400 | 1 | b | 2 | 2020-02-09 19:48:27.726815 433352610813050881 | 2 | c
| 3 | 2020-02-09 19:48:33.721754 433352628370407424 | 3 | c | 4 | 2020-02-09 19:48:37.9
07683 (4 rows)
```

- 查询示例2

```
sdb=> select * from t_order where user_id=1;
```

结果示例

```
 order_id | user_id | info | c1 | crt_time --------------------+---------+------+----+-
--------------------------- 433352585668198400 | 1 | b | 2 | 2020-02-09 19:48:27.726815
(1 row)
```

- 如果需要查看ShardingSphere日志，路径如下：

```
/root/apache-shardingsphere-incubating-4.0.0-sharding-proxy-bin/logs/stdout.log
```

- 如果需要使用pgbench压测，请参见如下命令：
  i. 创建测试sql文件并打开。

```
vi test.sql
```

  ii. 参考如下示例，配置pgbench压测命令。

```
\set user_id random(1,100000000) \set order_id random(1,2000000000) \set order_item_i
d random(1,2000000000) insert into t_order (user_id, order_id, info, c1 , crt_time) v
alues (:user_id, :order_id,random()::text, random()*1000, now()) on conflict (order_i
d) do update set info=excluded.info,c1=excluded.c1,crt_time=excluded.crt_time; insert
into t_order_item (order_item_id, user_id, order_id, info, c1,c2,c3,c4,c5,crt_time) v
alues (:order_item_id, :user_id,:order_id,random()::text, random()*1000,random()*1000
,random()*1000,random()*1000,random()*1000, now()) on conflict(order_item_id) do noth
ing; pgbench -M simple -n -r -P 1 -f ./test.sql -c 24 -j 24 -h 127.0.0.1 -p 8001 -U r
1 sdb -T 120 progress: 1.0 s, 1100.9 tps, lat 21.266 ms stddev 6.349 progress: 2.0 s,
1253.0 tps, lat 18.779 ms stddev 7.913 progress: 3.0 s, 1219.0 tps, lat 20.083 ms std
dev 13.212
```