阿里云
云数据库 MySQL 版

RDS for PPAS 快速入门

文档版本：20190919

为了无法计算的价值 | 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
	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。	禁止: 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
##	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。	`swich` *`{stand | slave}`*

目录

法律声明.....I
通用约定.....I
1 RDS for PPAS使用限制.....1
2 RDS for PPAS使用流程.....2
3 创建RDS for PPAS实例.....3
4 初始化配置.....9
4.1 设置白名单.....9
4.2 申请外网地址.....14
4.3 创建数据库和账号.....17
5 连接实例.....22
6 只读实例.....27
6.1 PPAS只读实例简介.....27
6.2 创建PPAS只读实例.....29
7 使用 oss_fdw 读写外部数据文本文件.....33
8 附录.....38
8.1 附录：用户及 Schema 管理.....38
8.2 附录：PPAS 开发驱动程序.....39
8.3 PPAS 兼容性说明.....39
8.4 常用管理函数.....50

1 RDS for PPAS使用限制

为保障实例的稳定及安全，云数据库PPAS有部分使用上的约束。

RDS for PPAS的使用限制详情如下表所示。

操作	使用约束
修改数据库参数设置	暂不支持。
数据库的root权限	RDS无法向用户提供superuser权限。
数据库备份	只支持通过pg_dump进行数据备份。
数据迁入	只支持通过psql还原由pg_dump备份的数据。
搭建数据库复制	· 系统自动搭建了基于PPAS流复制的HA模式，无需用户手动搭建 · PPAS Standby节点对用户不可见，不能直接用于访问。
重启RDS实例	必须通过控制台或API重启实例。
网络设置	若实例的代理模式是高安全模式，禁止在SNAT模式下开启net.ipv4.tcp_timestamps。

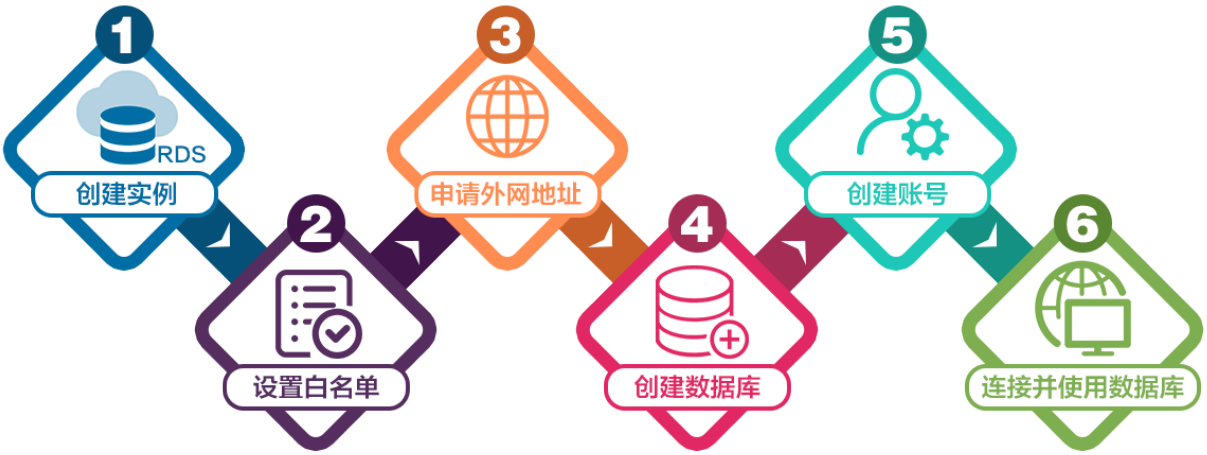
2 RDS for PPAS使用流程

快速入门旨在介绍如何创建RDS for PPAS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#unique_6。

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. #unique_7
2. #unique_8
3. #unique_9
4. #unique_10
5. #unique_11

3 创建RDS for PPAS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 for PPAS实例。

您可以通过阿里云RDS管理控制台或API创建RDS实例。关于如何通过API创建实例，请参见创建RDS实例。

关于实例计费说明，请参见计费方式。

除了新版本的创建实例页面，您也可以切换回旧版创建实例页面。操作详情请参见：

- 创建RDS实例（新版）
- 创建RDS实例（旧版）

前提条件

- 已注册阿里云账号。
- 若您要创建按时付费的实例，请确保您的账户余额大于等于100元。

注意事项

- 包年包月实例无法转为按量付费实例。
- 按量付费实例可以转为包年包月实例，请参见#unique_15。
- 同一个主账号，最多可以创建30个按量付费的RDS实例。如需提高此限额，请提交工单申请。

创建RDS实例（新版）

1. 进入RDS实例创建页面。

说明：

您也可以在当前创建RDS实例页面上方单击返回旧版切换到旧版创建RDS实例页面。

2. 设置以下参数。

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。
地域	实例所在的地域，即实例所在的地理位置。 · 购买后无法更换地域。 · 请根据目标用户所在的地理位置就近选择地域，提升用户访问速度。 · 请确保RDS实例与需要连接的ECS实例创建于同一个地域，否则它们无法通过内网互通，只能通过外网互通，无法发挥最佳性能。
类型	数据库引擎的类型和版本，这里选择PPAS。 当前支持PPAS 9.3、10。 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	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。
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。 · ESSD云盘：增强型（Enhanced）SSD云盘，是阿里云全新推出的超高性能云盘产品。ESSD云盘基于新一代分布式块存储架构，结合25GE网络和RDMA技术，为您提供单盘高达100万的随机读写能力和更低的单路时延。 · SSD云盘：基于分布式存储架构的弹性块存储设备。将数据存储于SSD云盘，即实现了计算与存储分离。 更多信息，请参见存储类型。
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。 您只需要选择将RDS实例的主可用区，备可用区系统会自动选择。

类别	说明
实例规格	· 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。 · 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#unique_18。
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#unique_18。

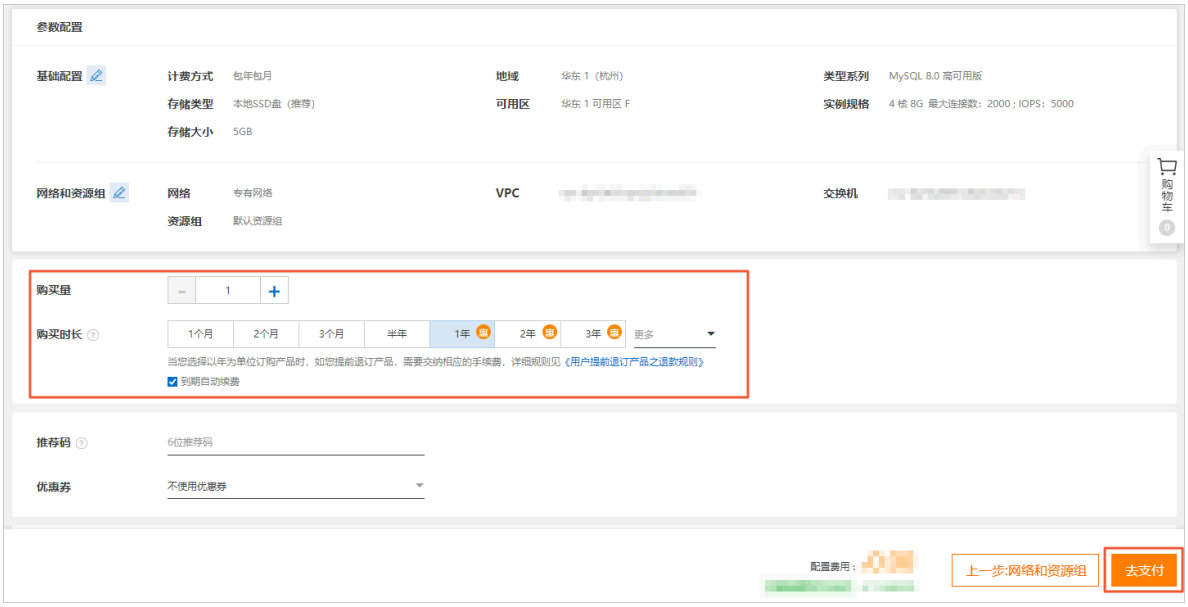
3. 单击下一步：网络和资源组。
4. 设置以下参数。

类别	说明
网络类型	· 经典网络：传统的网络类型。 · 专有网络：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。 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。

5. 单击下一步：确认订单，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。

说明：

如果有优惠券，可以在本页面选择对应的优惠券。



创建RDS实例（旧版）

1. 进入RDS实例创建页面。
2. 选择包年包月或按量付费。关于计费方式的选择，请参见#unique_19。
3. 选择实例配置，参数说明如下。

参数	说明
地域	实例所在的地理位置。购买后无法更换地域。 · 请根据目标用户所在的地理位置就近选择地域，提升用户访问速度。 · 请确保RDS实例与需要连接的ECS实例创建于同一个地域，否则它们无法通过内网互通，只能通过外网互通，无法发挥最佳性能。
资源组	实例所属的资源组。
数据库类型	即数据库引擎的类型，这里选择PPAS（高度兼容Oracle）。 说明： 不同地域支持的数据库类型不同，请以实际界面为准。
版本	指PPAS的版本。RDS for PPAS支持的版本包括PPAS 9.3、10。 说明： 不同地域所支持的版本不同，请以实际界面为准。

参数	说明
系列	高可用版：一个主节点和一个备节点，经典高可用架构。关于各个系列的详细介绍，请参见产品系列概述。 不同数据库版本支持的系列不同，请以实际界面为准。
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。 更多信息，请参见存储类型。
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实例的主备节点创建在同一可用区或不同可用区。 相比单可用区，多可用区能提供可用区级别的容灾。
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。 说明: 请确保RDS实例与需要连接的ECS实例网络类型一致，否则它们无法通过内网互通。
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#unique_18。 RDS实例有以下规格族： · 通用型：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。 · 独享型：独享被分配的CPU、内存、存储和I/O资源。 · 独占物理机型：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。 例如，8核32GB是通用型实例规格，8核32GB（独享套餐）是独享型实例规格，30核220GB（独占主机）是独占物理机型实例规格。
存储空间	该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。

4. 设置购买时长（仅针对包年包月实例）和实例数量，然后单击右侧的立即购买。

说明:

- 购买包年包月实例时，可以勾选自动续费，系统将根据您的购买时长进行自动续费。例如，您购买3个月的实例并勾选自动续费，则每次自动续费时会缴纳3个月的费用。
- 对于包年包月实例，您也可以单击加入购物车将实例加入到购物车中，最后单击购物车进行结算。

5. 在订单确认页面，勾选相关协议，根据提示完成支付。

下一步

在控制台左上角，选择实例所在的地域即可查看到刚刚创建的实例。

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。

如果连接实例失败，请参见为什么无法连接RDS?

相关API

API	描述
#unique_23	创建RDS实例

操作视频

RDS实例创建

4 初始化配置

4.1 设置白名单

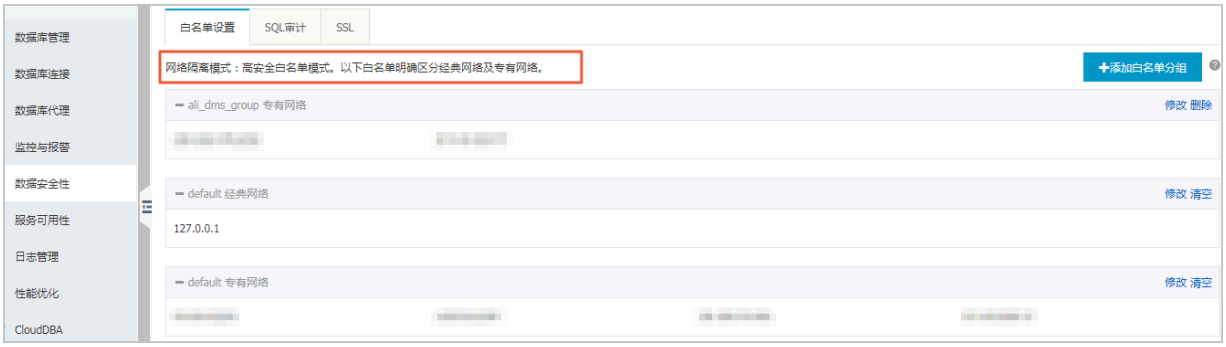
创建RDS实例后，您需要设置RDS实例的白名单，以允许外部设备访问该RDS实例。

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。默认的白名单只包含默认IP地址127.0.0.1，表示任何设备均无法访问该RDS实例。

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。设置白名单不会影响RDS实例的正常运行。

注意事项

- 默认的IP白名单分组只能被修改或清空，不能被删除。
- 每个IP白名单分组最多添加1000个IP或IP段。当IP较多时，建议合并为IP段填入，例如192.168.1.0/24。
- 当未设置白名单登录DMS时，会提示添加IP才可以正常登录，会自动生成相应的白名单分组。
- 设置白名单之前，您需要确认实例处于哪种网络隔离模式，根据模式查看相应的操作步骤。



说明:

RDS实例所处的内网分为经典网络和专有网络两种。

- 经典网络：传统的网络类型。

- 专有网络：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。

操作步骤

高安全白名单模式操作步骤

1. 登录RDS管理控制台。
2. 在页面左上角，选择实例所在地域。

3. 找到目标实例，单击实例ID。
4. 在左侧导航栏中选择数据安全性。
5. 在白名单设置页面中，根据以下连接类型进行后续操作。
   - 专有网络下的ECS实例连接到RDS实例：单击`default 专有网络`分组右侧的修改。
   - 经典网络下的ECS实例连接到RDS实例：单击`default 经典网络`分组右侧的修改。
   - 外网的实例或主机连接到RDS实例：单击`default 经典网络`分组右侧的修改。

说明：

- 若需要ECS实例通过内网地址（专有网络地址和经典网络地址）连接到RDS，请确保两者处于同一地域内，且网络类型相同，否则设置了白名单也无法连接成功。
- 您也可以单击添加白名单分组新建自定义分组，根据连接类型选择专有网络或经典网络 及外网地址。

6. 在弹出的对话框中，填写需要访问该实例的IP地址或IP段，然后单击确定。

- 若填写IP段，如10.10.10.0/24，则表示10.10.10.X的IP地址都可以访问该RDS实例。
- 若您需要添加多个IP地址或IP段，请用英文逗号隔开（逗号前后都不能有空格），例如192.168.0.1,172.16.213.9。
- 单击加载ECS内网IP后，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内网IP地址到白名单中。

说明:

当您在default分组中添加新的IP地址或IP段后，默认地址127.0.0.1会被自动删除。

通用白名单模式操作步骤

1. 登录RDS管理控制台。

2. 在页面左上角，选择实例所在地域。

3. 找到目标实例，单击实例ID。
4. 在左侧导航栏中选择数据安全性。
5. 在白名单设置页面中，单击default白名单分组中的修改，如下图所示。

> **说明：**
> 您也可以单击添加白名单分组新建自定义分组。

6. 在修改白名单分组对话框中，填写需要访问该实例的IP地址或 IP 段，然后单击确定。
   - 若填写IP段，如10.10.10.0/24，则表示10.10.10.X的IP地址都可以访问该RDS实例。
   - 若您需要添加多个IP地址或IP段，请用英文逗号隔开（逗号前后都不能有空格），例如192.168.0.1,172.16.213.9。
   - 单击加载ECS内网IP后，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内网IP地址到白名单中。

> **说明：**

当您在default分组中添加新的IP地址或IP段后，默认地址127.0.0.1会被自动删除。

常见错误案例

- 由于数据安全性 > 白名单设置中只有默认地址127.0.0.1。该地址表示不允许任何设备访问RDS实例。因此需在白名单中添加对端的IP地址。
- 白名单设置成了0.0.0.0，正确格式为0.0.0.0/0。

说明:

0.0.0.0/0表示允许任何设备访问RDS实例，请谨慎使用。

- 如果开启了高安全白名单模式，需进行如下检查：
  - 如果使用的是专有网络的内网连接地址，请确保ECS内网IP地址添加到了专有网络的分组。
  - 如果使用的是经典网络的内网连接地址，请确保ECS内网IP地址添加到了经典网络的分组。
  - 如果使用ClassicLink访问RDS的专有网络地址，请确保ECS内网IP地址添加到了default专有网络分组。
  - 如果通过公网连接，请确保设备公网IP地址添加到了经典网络的分组（专有网络的分组不适用于公网）。
- 您在白名单中添加的设备公网IP地址可能并非设备真正的出口IP地址。原因如下：
  - 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地址不准确。

  解决办法请参见#unique_28。

相关API

API	描述
#unique_29	查看RDS实例IP白名单
#unique_30	修改RDS实例IP白名单

4.2 申请外网地址

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。

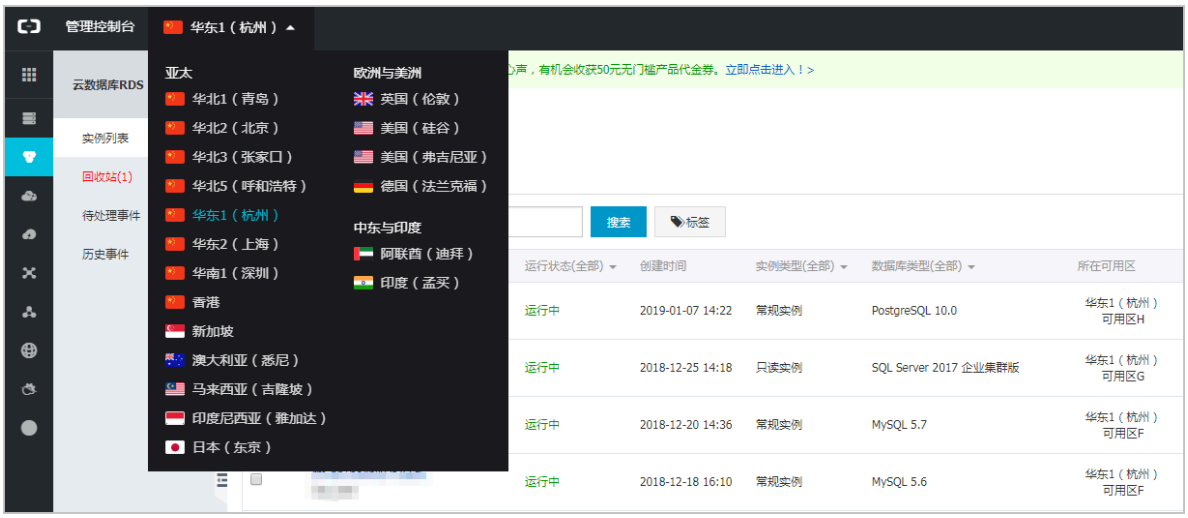
内网地址和外网地址

地址类型	说明
内网地址	· 默认提供内网地址。 · 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。

地址类型	说明
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。

申请外网地址

1. 登录RDS管理控制台。
2. 在页面左上角，选择实例所在地域。



3. 找到目标实例，单击实例ID。
4. 在左侧导航栏中选择数据库连接。

5. 单击申请外网地址。

6. 在弹出的对话框中，单击确定。

   外网地址生成成功。

7. （可选）如果您要修改外网地址或端口号，单击修改连接地址，在弹出的对话框中设置外网地址及端口号，然后单击确定。

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（-）。
- 专有网络下，内外网地址的端口都无法修改。
- 经典网络下，内外网地址的端口都支持修改。

相关API

API	描述
#unique_33	申请实例的外网连接串

4.3 创建数据库和账号

本文介绍如何为RDS for PPAS实例创建账号和数据库。

若要使用云数据库RDS，您需要在实例中创建数据库和账号。对于PPAS类型的实例，您需要通过RDS控制台创建一个初始账号，然后可以通过数据管理（DMS）控制台创建和管理数据库。本文将主要介绍在PPAS类型的实例中创建数据库和账号的操作步骤。

注意事项

- 同一实例下的数据库共享该实例下的所有资源。每个PPAS类型的实例支持创建无数个数据库，支持创建一个初始账号以及无数个普通账号，您可以通过SQL命令创建、管理普通账号和数据库。
- 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。
- 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。

操作步骤

1. 登录RDS管理控制台。
2. 在页面左上角，选择实例所在地域。

3. 找到目标实例，单击实例ID。
4. 在左侧导航栏中，选择账号管理。
5. 单击创建初始账号。

6. 输入要创建的账号信息，如下图所示。

参数说明：

- 数据库账号：长度为2~16个字符，由小写字母、数字或下划线组成。但开头需为字母，结尾需为字母或数字。
- 密码：该账号对应的密码。
  - 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：输入与密码一致的字段，以确保密码正确输入。

7. 单击确定。
8. 单击页面右上角的登录数据库，进入数据管理控制台的快捷登录页面。

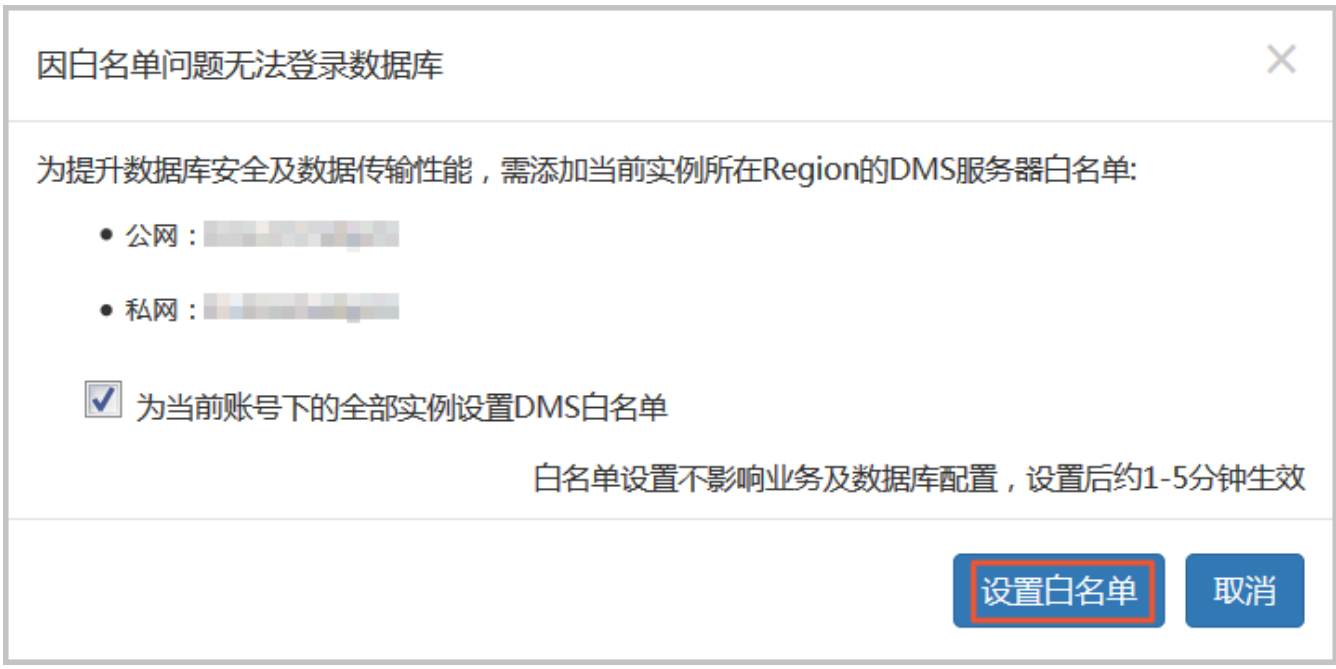
9. 在快捷登录页面，检查阿里云数据库标签页面显示的连接地址和端口信息。若信息正确，填写数据库用户名和密码，如下图所示。

- 参数说明：
  - 1：实例的连接地址和端口信息。
  - 2：要访问数据库的账号名称。
  - 3：上述账户所对应的密码。

10.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，然后再单击登录。

11.若出现将DMS服务器的IP段加入到RDS白名单中的提示，单击设置白名单，如下图所示。若需手动添加，请参见设置白名单。



12.成功添加白名单后，单击登录。

13.成功登录RDS实例后，在页面上方的菜单栏中，选择SQL操作 > SQL窗口 。

14.在SQL窗口中输入如下命令，创建数据库。

```
CREATE DATABASE name
 [ [ WITH ] [ OWNER [=] user_name ]
        [ TEMPLATE [=] template ]
        [ ENCODING [=] encoding ]
        [ LC_COLLATE [=] lc_collate ]
        [ LC_CTYPE [=] lc_ctype ]
        [ TABLESPACE [=] tablespace_name ]
        [ CONNECTION LIMIT [=] connlimit ] ]
```

例如，若您要创建一个名称为test的数据库，可以执行如下命令：

```
create database test;
```

15.单击执行，完成创建数据库。

16.在SQL窗口中输入如下命令，创建普通账号。

```
CREATE USER name [ [ WITH ] option [ ... ] ]
where option can be:
    SUPERUSER | NOSUPERUSER
  | CREATEDB | NOCREATEDB
  | CREATEROLE | NOCREATEROLE
  | CREATEUSER | NOCREATEUSER
  | INHERIT | NOINHERIT
  | LOGIN | NOLOGIN
  | REPLICATION | NOREPLICATION
  | CONNECTION LIMIT connlimit
  | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
```

```
  | VALID UNTIL 'timestamp'
  | IN ROLE role_name [, ...]
  | IN GROUP role_name [, ...]
  | ROLE role_name [, ...]
  | ADMIN role_name [, ...]
  | USER role_name [, ...]
  | SYSID uid
```

例如，若您要创建一个名称为test2、密码为123456的数据库，可以执行如下命令：

```
create user test2 password'123456';
```

17.单击执行，完成创建普通账号。

18.将要访问RDS实例的IP地址加入RDS白名单中。关于如何设置白名单，请参见设置白名单。

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
#unique_35	创建账号

5 连接实例

初始化配置后，您可以让ECS连接PPAS实例，也可以本地连接到PPAS实例，实现业务目标。

若您要使用云数据库RDS，可以通过客户端或阿里云数据管理（DMS）连接RDS实例。本章将介绍如何通过DMS和pgAdmin 4客户端连接RDS实例。

如果连接失败，请参见#unique_37。

背景信息

您可以通过RDS管理控制台先登录DMS，然后再连接需要访问的RDS实例。数据管理（Data Management，简称DMS）是一种集数据管理、结构管理、访问安全、BI图表、数据趋势、数据轨迹、性能与优化和服务器管理于一体的数据管理服务。支持MySQL、SQL Server、PostgreSQL、MongoDB、Redis等关系型数据库和NoSQL的数据库管理，同时还支持Linux服务器管理。

您也可以使用客户端连接RDS实例。由于RDS提供的关系型数据库服务与原生的数据库服务完全兼容，所以对用户而言，连接数据库的方式也基本类似。本文以pgAdmin 4客户端为例介绍RDS实例的连接方法，其它客户端可参见此方法。用客户端连接RDS实例时，请注意选择内外网地址：

- 若您的客户端部署在ECS实例上，且ECS实例与要访问的RDS实例的地域、网络类型相同，请使用内网地址。例如ECS实例和RDS实例都是华东1的专有网络实例，使用内网地址连接能提供安全高效的访问。
- 其它情况请使用外网地址。

通过DMS连接实例

关于如何通过DMS连接RDS实例的方法，请参见#unique_39。

通过客户端登录

1. 将要访问RDS实例的IP地址加入RDS白名单中。关于如何设置白名单，请参见设置白名单。
2. 启动pgAdmin 4客户端。

3. 右击Servers，选择创建 > 服务器，如下图所示。

4. 在创建-服务器页面的通常标签页面中，输入服务器名称，如下图所示。

5. 选择Connection标签页，输入要连接的实例信息，如下图所示。

参数说明：

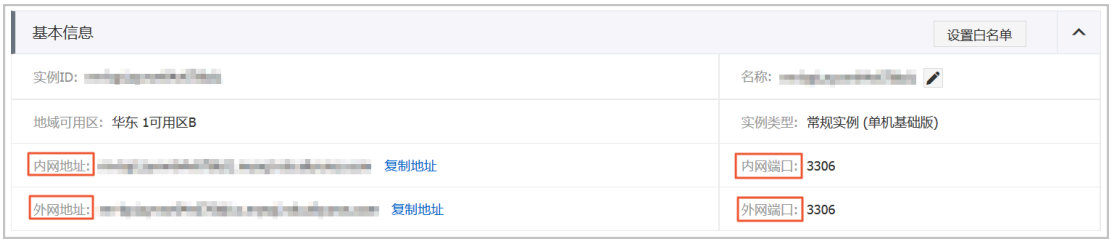
- 主机名称/地址：若使用内网连接，需输入RDS实例的内网地址。若使用外网连接，需输入RDS实例的外网地址。查看RDS实例的内外网地址及端口信息的步骤如下：

  a. 登录RDS管理控制台。

  b. 在页面左上角，选择实例所在地域。

  c. 找到目标实例，单击实例ID。

  d. 在基本信息栏中，即可查看内外网地址及内外网端口信息。



- 端口：若使用内网连接，需输入RDS实例的内网端口。若使用外网连接，需输入RDS实例的外网端口。
- 用户名：RDS实例的初始账号名称。
- 密码：RDS实例的初始账号所对应的密码。

6. 单击保存。
7. 若连接信息无误，选择Servers > 服务器名称 > 数据库 > edb或者postgres，会出现如下界面，则表示连接成功。

> **说明：**
> edb和postgres是RDS实例默认的系统数据库，请勿在这两个数据库中进行任何操作。

连接失败的解决办法

请参见#unique_37。

操作视频

ECS连接RDS

6 只读实例

6.1 PPAS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。

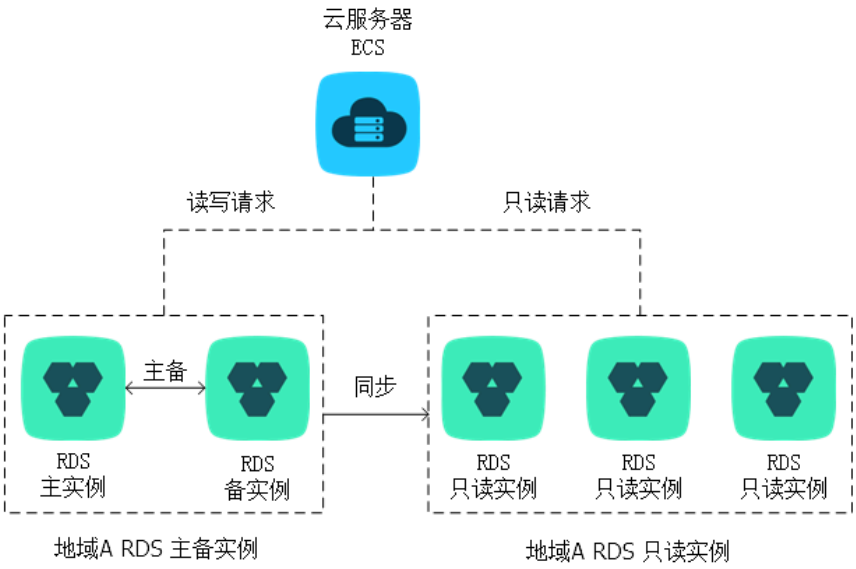
简介

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:

- PPAS 10.0支持只读实例，PPAS 9.3不支持。
- 主实例规格不低于8核32G（独享套餐）。
- 只读实例为单节点的架构（没有备节点）。

只读实例拓扑图如下图所示。



计费

按量付费，即每小时扣费一次。

功能特点

- 计费方式：按量付费，使用更灵活，费用更便宜。
- 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：只读实例的规格和存储空间不能低于主实例。
- #unique_26：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#unique_42。
- 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利用率等。

功能限制

- 只读实例的数量：最多创建5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。

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。

6.2 创建PPAS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#unique_44。

前提条件

- 主实例版本为PPAS 10.0。
- 主实例规格不低于8核32G（独享套餐）。

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。
- 只读实例的规格和存储空间不能低于主实例。
- 最多创建5个只读实例。
- 计费方式：按量付费，即每小时扣费一次。

创建只读实例

1. 登录RDS管理控制台。
2. 在页面左上角，选择实例所在地域。

3. 找到目标实例，单击实例ID。
4. 在页面右侧单击添加只读实例。

5. 在购买页面，设置只读实例的参数，然后单击立即购买。

说明:

- 专有网络VPC：建议选择与主实例相同的VPC。
- 规格和存储空间：只读实例的规格和存储空间不能低于主实例。
- 数量：根据业务量购买，多个只读实例可以提高可用性，最多5个。

6. 在订单确认页面，勾选关系型数据库RDS服务条款，根据提示完成支付。

   几分钟后，该只读实例即创建成功。

几分钟后，该只读实例即创建成功。

查看只读实例

在实例列表中查看只读实例

1. 登录RDS管理控制台。
2. 选择只读实例所在地域。

3. 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

1. 登录RDS管理控制台。
2. 选择主实例所在地域。

3. 找到目标实例，单击实例ID。

4. 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
#unique_45	创建RDS只读实例

7 使用 oss_fdw 读写外部数据文本文件

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。

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/csv、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

- 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： OSSBucket，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。

- 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。
- 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。

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

- 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。
- compressionlevel：设置写入OSS的压缩格式的压缩等级，范围1到9，默认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_oss');
              name              |   size
--------------------------------+-----------
  oss_test/test.gz.2  | 739413041
(1 rows)
```

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 OPTIONS
     (host 'oss-cn-hangzhou.aliyuncs.com' ,  id 'xxx',  key 'xxx',
bucket 'mybucket');
# 创建 oss 外部表
CREATE FOREIGN TABLE ossexample
    (date text,  time text,  open float,
     high float,  low 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
```

```
create table example
        (date text,  time text,  open float,
         high float,  low float,  volume int);
# 数据从 ossexample 装载到 example 中。
insert into example select * from ossexample;
# 可以看到
# oss_fdw 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 ossexample;
                            QUERY PLAN
-----------------------------------------------------------------
 Insert 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 insert 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)
```

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 MET）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，相关信息请参考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。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。

请参考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。

最终的加密后的信息如下：

```
postgres=# select * from pg_foreign_server ;
  srvname  | srvowner | srvfdw | srvtype | srvversion | srvacl |

  srvoptions
-----------+----------+--------+---------+------------+--------
+-----------------------------------------------------------------------------------------
----------------------------------
 ossserver |       10 |  16390 |         |            |        | {host
=oss-cn-hangzhou-zmf.aliyuncs.com, id=MD5xxxxxxxx, key=MD5xxxxxxxx,
bucket=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。

8 附录

8.1 附录：用户及 Schema 管理

在使用 RDS 的过程中，由于 superuser 不完全放开，因此我们建议您在使用数据库时遵循单独建立用户并通过 schema 管理您的私有空间。

操作步骤

> **说明:**
> 本例中，myadmin 是建立实例时创建的管理账号，newuser 是当前需要新建的账号。

1. 用初始管理帐号登录数据库，如控制台中已经建立初始帐号的 myadmin。

```
psql -U myadmin -h intranet4example.pg.rds.aliyuncs.com -p 3433
pg001
Password for user myadmin:
psql.bin (9.4.4, server 9.4.1)
Type "help" for help.
```

2. 建立普通帐号 newuser，密码为 password

```
CREATE USER newuser LOGIN PASSWORD'password';
```

参数说明如下：

- USER：要创建的用户名。
- password：用户名对应的密码。

3. 用新用户 newuser 进行数据库登录。

```
psql -U newuser -h intranet4example.pg.rds.aliyuncs.com -p 3433
pg001
Password for user newuser:
psql.bin (9.4.4, server 9.4.1)
Type "help" for help.
```

4. 建立新的业务DATABASE，并建立与用户 newuser 同名的 SCHEMA 作为业务的默认操作空间

```
CREATE DATABASE mydb;
\c mydb;
CREATE SCHEMA newuser;
```

> **说明:**
> - 通过以上操作，用户 myadmin 将保持作为数据库的初始管理帐号。

- 所有用户业务操作都通过 mydb 进行处理，系统原生的系统库 edb 及 postgres 会存有系统表数据，不建议将业务数据放到系统库中。
- 以上操作以后库 mydb 库的主属用户将是 myuser，通过帐号 myuser 或以在此库建立任意对象

8.2 附录：PPAS 开发驱动程序

本文介绍如何配置PPAS开发驱动程序。

RDS for PPAS 开发驱动程序为应用开发提供丰富的驱动接口：

- Linux 版本包括：JAVA / OCI / ODBC
- Windows 版本包括：.Net / JAVA / OCI / ODBC

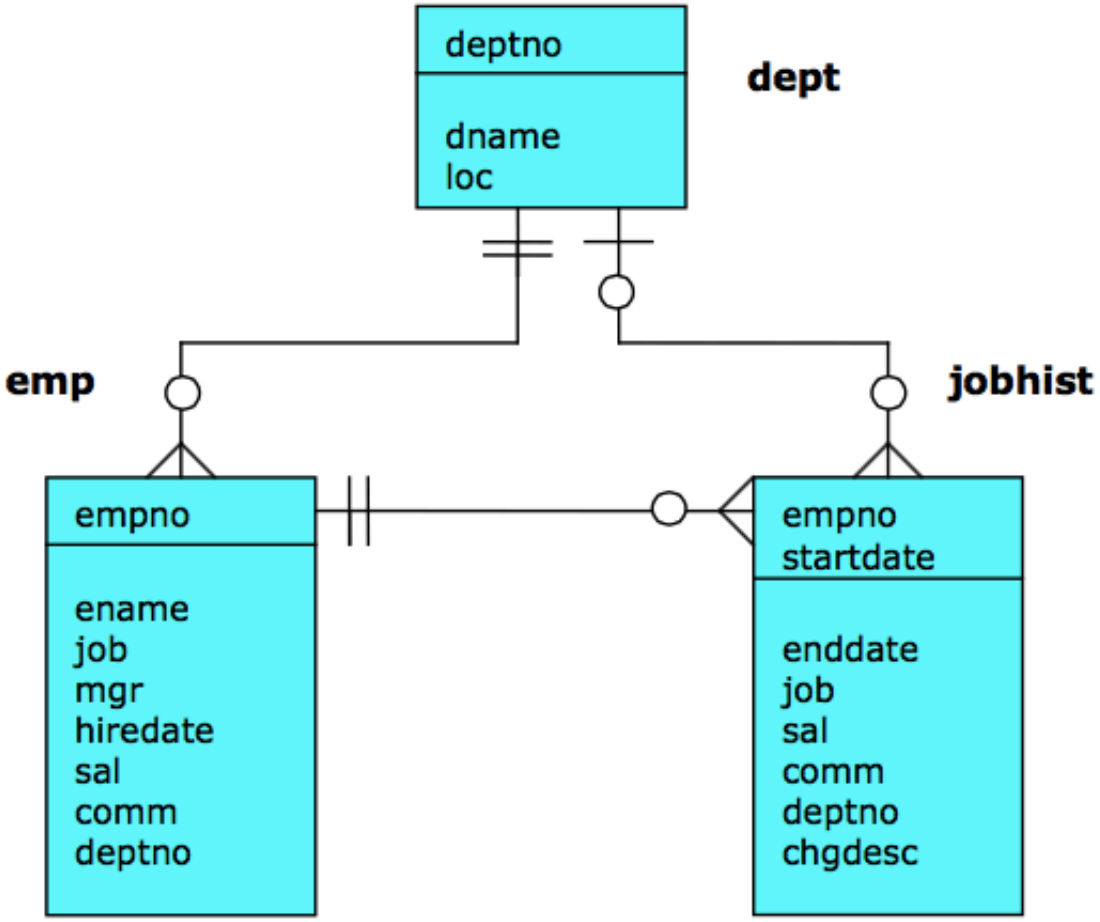
单击 此处 下载 PPAS 开发驱动程序。

- 驱动程序包含以下文件：
  - edb_connectors-9.3.5.14-3-linux-x64.run
  - edb_connectors-9.3.5.14-3-linux.run
  - edb_connectors-9.3.5.14-3-windows-x64.exe
  - edb_connectors-9.3.5.14-3-windows.exe
- 驱动程序默认安装路径为：
  - Linux：`/opt/PostgresPlus/9.3AS/connectors`
  - Windows：`C:/Program Files/PostgresPlus/9.3AS/connectors`

8.3 PPAS 兼容性说明

本文通过示例让Oracle 用户可以快速了解PPAS数据库中的术语及概念，以便在迁移及开发过程中提高效率。

以下所有操作基于一个基础模型，通过此模型用户可以看到 RDS for PPAS 中最基本的创建数据库、创建数据表、管理用户等操作，基础数据模型如下：



同时，为了模拟 Oracle 上类似的环境，我们会建立一名字为 orcl_ppas 的数据库（database），在此数据库中建立名为 scott 的用户，并建立与这个用户同名的 schema 用户空间。

PPAS兼容性手册

关于PPAS兼容性说明的完整内容，请参见阿里云PPAS兼容性手册。

连接数据库 psql

```
 psql -h ppasaddress.ppas.rds.aliyuncs.com -p 3433 -U myuser -d
template1
   用户 myuser 的口令:
   psql.bin (9.4.1.3, 服务器 9.3.5.14)
   输入 "help" 来获取帮助信息.
   template1=>
```

创建并连接数据库 CREATE DATABASE

```
 template1=> CREATE DATABASE orcl_ppas;
   CREATE DATABASE
   template1=> \c orcl_ppas
   psql.bin (9.4.1.3, 服务器 9.3.5.14)
```

创建普通用户 CREATE ROLE

```
  orcl_ppas=> CREATE ROLE scott LOGIN PASSWORD 'scott123';
```

```
CREATE ROLE
```

创建用户的私有空间 CREATE SCHEMA

```
orcl_ppas=> CREATE SCHEMA scott;
CREATE SCHEMA
orcl_ppas=> GRANT scott TO myuser;
GRANT ROLE
orcl_ppas=> ALTER SCHEMA scott OWNER TO scott;
ALTER SCHEMA
orcl_ppas=> REVOKE scott FROM myuser;
REVOKE ROLE
```

说明:

- 如果在进行`ALTER SCHEMA scott OWNER TO scott`之前没有将scott加入到myuser角色，将会出现如下权限问题。`ERROR:must be member of role "scott"`
- 从安全角度出发，在处理完OWNER的授权后，请将scott用户移出myuser角色以提高安全性。

连接到 orcl_ppas 数据库

说明:

此步骤十分重要，以下所有操作都是在scott账号下进行的，否则所建立的数据表及各种数据库对象将不属于scott用户，导致权限问题。

```
[root@localhost bin]# ./psql -h ppasaddress.ppas.rds.aliyuncs.com -p
3433 -U scott -d orcl_ppas
用户 scott 的口令:
psql.bin (9.4.1.3, 服务器 9.3.5.14)
输入 "help" 来获取帮助信息.
orcl_ppas=>
```

创建数据表 CREATE TABLE

```
CREATE TABLE dept (
   deptno          NUMBER(2) NOT NULL CONSTRAINT dept_pk PRIMARY KEY,
   dname           VARCHAR2(14) CONSTRAINT dept_dname_uq UNIQUE,
   lock            VARCHAR2(13)
);
CREATE TABLE emp (
  empno           NUMBER(4) NOT NULL CONSTRAINT emp_pk PRIMARY KEY,
  ename           VARCHAR2(10),
  job             VARCHAR2(9),
  mgr             NUMBER(4),
  hiredate        DATE,
  sal             NUMBER(7,2) CONSTRAINT emp_sal_ck CHECK (sal > 0),
  comm            NUMBER(7,2),
  deptno          NUMBER(2) CONSTRAINT emp_ref_dept_fk
                  REFERENCES dept(deptno)
);
CREATE TABLE jobhist (
  empno           NUMBER(4) NOT NULL,
```

```
  startdate       DATE NOT NULL,
  enddate         DATE,
  job             VARCHAR2(9),
  sal             NUMBER(7,2),
  comm            NUMBER(7,2),
  deptno          NUMBER(2),
  chgdesc         VARCHAR2(80),
  CONSTRAINT jobhist_pk PRIMARY KEY (empno, startdate),
  CONSTRAINT jobhist_ref_emp_fk FOREIGN KEY (empno)
      REFERENCES emp(empno) ON DELETE CASCADE,
  CONSTRAINT jobhist_ref_dept_fk FOREIGN KEY (deptno)
      REFERENCES dept (deptno) ON DELETE SET NULL,
  CONSTRAINT jobhist_date_chk CHECK (startdate <= enddate)
);
```

创建视图 CREATE OR REPLACE VIEW

```
CREATE OR REPLACE VIEW salesemp AS
   SELECT empno, ename, hiredate, sal, comm FROM emp WHERE job = '
SALESMAN';
```

创建序列 CREATE SEQUENCE

```
CREATE SEQUENCE next_empno START WITH 8000 INCREMENT BY 1;
```

插入数据 INSERT INTO

```
INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO dept VALUES (20,'RESEARCH','DALLAS');
INSERT INTO dept VALUES (30,'SALES','CHICAGO');
INSERT INTO dept VALUES (40,'OPERATIONS','BOSTON');
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'17-DEC-80',800,NULL
,20);
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,'20-FEB-81',1600,
300,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'22-FEB-81',1250,
500,30);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'02-APR-81',2975,
NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'28-SEP-81',1250
,1400,30);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'01-MAY-81',2850,
NULL,30);
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'09-JUN-81',2450,
NULL,10);
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'19-APR-87',3000,
NULL,20);
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'17-NOV-81',5000,
NULL,10);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'08-SEP-81',1500
,0,30);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'23-MAY-87',1100,
NULL,20);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'03-DEC-81',950,NULL
,30);
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'03-DEC-81',3000,
NULL,20);
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'23-JAN-82',1300,
NULL,10);
INSERT INTO jobhist VALUES (7369,'17-DEC-80',NULL,'CLERK',800,NULL,20
,'New Hire');
```

```
INSERT INTO jobhist VALUES (7499,'20-FEB-81',NULL,'SALESMAN',1600,300,
30,'New Hire');
INSERT INTO jobhist VALUES (7521,'22-FEB-81',NULL,'SALESMAN',1250,500,
30,'New Hire');
INSERT INTO jobhist VALUES (7566,'02-APR-81',NULL,'MANAGER',2975,NULL,
20,'New Hire');
INSERT INTO jobhist VALUES (7654,'28-SEP-81',NULL,'SALESMAN',1250,1400
,30,'New Hire');
INSERT INTO jobhist VALUES (7698,'01-MAY-81',NULL,'MANAGER',2850,NULL,
30,'New Hire');
INSERT INTO jobhist VALUES (7782,'09-JUN-81',NULL,'MANAGER',2450,NULL,
10,'New Hire');
INSERT INTO jobhist VALUES (7788,'19-APR-87','12-APR-88','CLERK',1000,
NULL,20,'New Hire');
INSERT INTO jobhist VALUES (7788,'13-APR-88','04-MAY-89','CLERK',1040,
NULL,20,'Raise');
INSERT INTO jobhist VALUES (7788,'05-MAY-90',NULL,'ANALYST',3000,NULL,
20,'Promoted to Analyst');
INSERT INTO jobhist VALUES (7839,'17-NOV-81',NULL,'PRESIDENT',5000,
NULL,10,'New Hire');
INSERT INTO jobhist VALUES (7844,'08-SEP-81',NULL,'SALESMAN',1500,0,30
,'New Hire');
INSERT INTO jobhist VALUES (7876,'23-MAY-87',NULL,'CLERK',1100,NULL,20
,'New Hire');
INSERT INTO jobhist VALUES (7900,'03-DEC-81','14-JAN-83','CLERK',950,
NULL,10,'New Hire');
INSERT INTO jobhist VALUES (7900,'15-JAN-83',NULL,'CLERK',950,NULL,30
,'Changed to Dept 30');
INSERT INTO jobhist VALUES (7902,'03-DEC-81',NULL,'ANALYST',3000,NULL,
20,'New Hire');
INSERT INTO jobhist VALUES (7934,'23-JAN-82',NULL,'CLERK',1300,NULL,10
,'New Hire');
```

查询优化器数据分析 ANALYZE

```
ANALYZE dept;
ANALYZE emp;
ANALYZE jobhist;
```

建立存储过程 CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE list_emp
IS
  v_empno         NUMBER(4);
  v_ename         VARCHAR2(10);
  CURSOR emp_cur IS
      SELECT empno, ename FROM emp ORDER BY empno;
BEGIN
  OPEN emp_cur;
  DBMS_OUTPUT.PUT_LINE('EMPNO    ENAME');
  DBMS_OUTPUT.PUT_LINE('-----    -------');
  LOOP
      FETCH emp_cur INTO v_empno, v_ename;
      EXIT WHEN emp_cur%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_empno || '     ' || v_ename);
  END LOOP;
  CLOSE emp_cur;
END;
--
--  Procedure that selects an employee row given the employee
--  number and displays certain columns.
--
```

```
CREATE OR REPLACE PROCEDURE select_emp (
  p_empno         IN  NUMBER
)
IS
  v_ename         emp.ename%TYPE;
  v_hiredate      emp.hiredate%TYPE;
  v_sal           emp.sal%TYPE;
  v_comm          emp.comm%TYPE;
  v_dname         dept.dname%TYPE;
  v_disp_date     VARCHAR2(10);
BEGIN
  SELECT ename, hiredate, sal, NVL(comm, 0), dname
      INTO v_ename, v_hiredate, v_sal, v_comm, v_dname
      FROM emp e, dept d
      WHERE empno = p_empno
        AND e.deptno = d.deptno;
  v_disp_date := TO_CHAR(v_hiredate, 'MM/DD/YYYY');
  DBMS_OUTPUT.PUT_LINE('Number    : ' || p_empno);
  DBMS_OUTPUT.PUT_LINE('Name      : ' || v_ename);
  DBMS_OUTPUT.PUT_LINE('Hire Date : ' || v_disp_date);
  DBMS_OUTPUT.PUT_LINE('Salary    : ' || v_sal);
  DBMS_OUTPUT.PUT_LINE('Commission: ' || v_comm);
  DBMS_OUTPUT.PUT_LINE('Department: ' || v_dname);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Employee ' || p_empno || ' not found');
  WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('The following is SQLERRM:');
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
      DBMS_OUTPUT.PUT_LINE('The following is SQLCODE:');
      DBMS_OUTPUT.PUT_LINE(SQLCODE);
END;
--
--  Procedure that queries the 'emp' table based on
--  department number and employee number or name.  Returns
--  employee number and name as IN OUT parameters and job,
--  hire date, and salary as OUT parameters.
--
CREATE OR REPLACE PROCEDURE emp_query (
  p_deptno        IN     NUMBER,
  p_empno         IN OUT NUMBER,
  p_ename         IN OUT VARCHAR2,
  p_job           OUT    VARCHAR2,
  p_hiredate      OUT    DATE
  p_sal           OUT    NUMBER
)
IS
BEGIN
  SELECT empno, ename, job, hiredate, sal
      INTO p_empno, p_ename, p_job, p_hiredate, p_sal
      FROM emp
      WHERE deptno = p_deptno
        AND (empno = p_empno
         OR  ename = UPPER(p_ename));
END;
--
--  Procedure to call 'emp_query_caller' with IN and IN OUT
--  parameters.  Displays the results received from IN OUT and
--  OUT parameters.
--
CREATE OR REPLACE PROCEDURE emp_query_caller
IS
  v_deptno        NUMBER(2);
  v_empno         NUMBER(4);
```

```
  v_ename         VARCHAR2(10);
  v_job           VARCHAR2(9);
  v_hiredate      DATE;
  v_sal           NUMBER;
BEGIN
  v_deptno := 30;
  v_empno  := 0;
  v_ename  := 'Martin';
  emp_query(v_deptno, v_empno, v_ename, v_job, v_hiredate, v_sal);
  DBMS_OUTPUT.PUT_LINE('Department : ' || v_deptno);
  DBMS_OUTPUT.PUT_LINE('Employee No: ' || v_empno);
  DBMS_OUTPUT.PUT_LINE('Name       : ' || v_ename);
  DBMS_OUTPUT.PUT_LINE('Job        : ' || v_job);
  DBMS_OUTPUT.PUT_LINE('Hire Date  : ' || v_hiredate);
  DBMS_OUTPUT.PUT_LINE('Salary     : ' || v_sal);
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
      DBMS_OUTPUT.PUT_LINE('More than one employee was selected');
  WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No employees were selected');
END;
```

建立函数 CREATE FUNCTION

```
CREATE OR REPLACE FUNCTION emp_comp (
  p_sal           NUMBER,
  p_comm          NUMBER
) RETURN NUMBER
IS
BEGIN
  RETURN (p_sal + NVL(p_comm, 0)) * 24;
END;
--
--  Function that gets the next number from sequence, 'next_empno',
--  and ensures it is not already in use as an employee number.
--
CREATE OR REPLACE FUNCTION new_empno RETURN NUMBER
IS
  v_cnt           INTEGER := 1;
  v_new_empno     NUMBER;
BEGIN
  WHILE v_cnt > 0 LOOP
      SELECT next_empno.nextval INTO v_new_empno FROM dual;
      SELECT COUNT(*) INTO v_cnt FROM emp WHERE empno = v_new_empno;
  END LOOP;
  RETURN v_new_empno;
END;
--
--  EDB-SPL function that adds a new clerk to table 'emp'.  This
function
--  uses package 'emp_admin'.
--
CREATE OR REPLACE FUNCTION hire_clerk (
  p_ename         VARCHAR2,
  p_deptno        NUMBER
) RETURN NUMBER
IS
  v_empno         NUMBER(4);
  v_ename         VARCHAR2(10);
  v_job           VARCHAR2(9);
  v_mgr           NUMBER(4);
  v_hiredate      DATE;
  v_sal           NUMBER(7,2);
```

```
  v_comm          NUMBER(7,2);
  v_deptno        NUMBER(2);
BEGIN
  v_empno := new_empno;
  INSERT INTO emp VALUES (v_empno, p_ename, 'CLERK', 7782,
      TRUNC(SYSDATE), 950.00, NULL, p_deptno);
  SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno INTO
      v_empno, v_ename, v_job, v_mgr, v_hiredate, v_sal, v_comm,
v_deptno
      FROM emp WHERE empno = v_empno;
  DBMS_OUTPUT.PUT_LINE('Department : ' || v_deptno);
  DBMS_OUTPUT.PUT_LINE('Employee No: ' || v_empno);
  DBMS_OUTPUT.PUT_LINE('Name       : ' || v_ename);
  DBMS_OUTPUT.PUT_LINE('Job        : ' || v_job);
  DBMS_OUTPUT.PUT_LINE('Manager    : ' || v_mgr);
  DBMS_OUTPUT.PUT_LINE('Hire Date  : ' || v_hiredate);
  DBMS_OUTPUT.PUT_LINE('Salary     : ' || v_sal);
  DBMS_OUTPUT.PUT_LINE('Commission : ' || v_comm);
  RETURN v_empno;
EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('The following is SQLERRM:');
      DBMS_OUTPUT.PUT_LINE(SQLERRM);
      DBMS_OUTPUT.PUT_LINE('The following is SQLCODE:');
      DBMS_OUTPUT.PUT_LINE(SQLCODE);
      RETURN -1;
END;
--
--  PostgreSQL PL/pgSQL function that adds a new salesman
--  to table 'emp'.
--
CREATE OR REPLACE FUNCTION hire_salesman (
  p_ename         VARCHAR,
  p_sal           NUMERIC,
  p_comm          NUMERIC
) RETURNS NUMERIC
AS $$
DECLARE
  v_empno         NUMERIC(4);
  v_ename         VARCHAR(10);
  v_job           VARCHAR(9);
  v_mgr           NUMERIC(4);
  v_hiredate      DATE;
  v_sal           NUMERIC(7,2);
  v_comm          NUMERIC(7,2);
  v_deptno        NUMERIC(2);
BEGIN
  v_empno := new_empno();
  INSERT INTO emp VALUES (v_empno, p_ename, 'SALESMAN', 7698,
      CURRENT_DATE, p_sal, p_comm, 30);
  SELECT INTO
      v_empno, v_ename, v_job, v_mgr, v_hiredate, v_sal, v_comm,
v_deptno
      empno, ename, job, mgr, hiredate, sal, comm, deptno
      FROM emp WHERE empno = v_empno;
  RAISE INFO 'Department : %', v_deptno;
  RAISE INFO 'Employee No: %', v_empno;
  RAISE INFO 'Name       : %', v_ename;
  RAISE INFO 'Job        : %', v_job;
  RAISE INFO 'Manager    : %', v_mgr;
  RAISE INFO 'Hire Date  : %', v_hiredate;
  RAISE INFO 'Salary     : %', v_sal;
  RAISE INFO 'Commission : %', v_comm;
  RETURN v_empno;
```

```
EXCEPTION
  WHEN OTHERS THEN
      RAISE INFO 'The following is SQLERRM:';
      RAISE INFO '%', SQLERRM;
      RAISE INFO 'The following is SQLSTATE:';
      RAISE INFO '%', SQLSTATE;
      RETURN -1;
END;
```

建立规则 CREATE RULE

```
CREATE OR REPLACE RULE salesemp_i AS ON INSERT TO salesemp
DO INSTEAD
  INSERT INTO emp VALUES (NEW.empno, NEW.ename, 'SALESMAN', 7698,
      NEW.hiredate, NEW.sal, NEW.comm, 30);
CREATE OR REPLACE RULE salesemp_u AS ON UPDATE TO salesemp
DO INSTEAD
  UPDATE emp SET empno    = NEW.empno,
                 ename    = NEW.ename,
                 hiredate = NEW.hiredate,
                 sal      = NEW.sal,
                 comm     = NEW.comm
      WHERE empno = OLD.empno;
CREATE OR REPLACE RULE salesemp_d AS ON DELETE TO salesemp
DO INSTEAD
DELETE FROM emp WHERE empno = OLD.empno;
```

建立触发器 CREATE TRIGGER

```
CREATE OR REPLACE TRIGGER user_audit_trig
  AFTER INSERT OR UPDATE OR DELETE ON emp
DECLARE
  v_action        VARCHAR2(24);
BEGIN
  IF INSERTING THEN
      v_action := ' added employee(s) on ';
  ELSIF UPDATING THEN
      v_action := ' updated employee(s) on ';
  ELSIF DELETING THEN
      v_action := ' deleted employee(s) on ';
  END IF;
  DBMS_OUTPUT.PUT_LINE('User ' || USER || v_action || TO_CHAR(SYSDATE
,'YYYY-MM-DD'));
END;
CREATE OR REPLACE TRIGGER emp_sal_trig
  BEFORE DELETE OR INSERT OR UPDATE ON emp
  FOR EACH ROW
DECLARE
  sal_diff       NUMBER;
BEGIN
  IF INSERTING THEN
      DBMS_OUTPUT.PUT_LINE('Inserting employee ' || :NEW.empno);
      DBMS_OUTPUT.PUT_LINE('..New salary: ' || :NEW.sal);
  END IF;
  IF UPDATING THEN
      sal_diff := :NEW.sal - :OLD.sal;
      DBMS_OUTPUT.PUT_LINE('Updating employee ' || :OLD.empno);
      DBMS_OUTPUT.PUT_LINE('..Old salary: ' || :OLD.sal);
      DBMS_OUTPUT.PUT_LINE('..New salary: ' || :NEW.sal);
      DBMS_OUTPUT.PUT_LINE('..Raise     : ' || sal_diff);
  END IF;
  IF DELETING THEN
```

```
      DBMS_OUTPUT.PUT_LINE('Deleting employee ' || :OLD.empno);
      DBMS_OUTPUT.PUT_LINE('..Old salary: ' || :OLD.sal);
  END IF;
END;
```

建立包 CREATE PACKAGE

```
CREATE OR REPLACE PACKAGE emp_admin
IS
  FUNCTION get_dept_name (
      p_deptno        NUMBER
  ) RETURN VARCHAR2;
  FUNCTION update_emp_sal (
      p_empno         NUMBER,
      p_raise         NUMBER
  ) RETURN NUMBER;
  PROCEDURE hire_emp (
      p_empno         NUMBER,
      p_ename         VARCHAR2,
      p_job           VARCHAR2,
      p_sal           NUMBER,
      p_hiredate      DATE,
      p_comm          NUMBER,
      p_mgr           NUMBER,
      p_deptno        NUMBER
  );
  PROCEDURE fire_emp (
      p_empno         NUMBER
  );
END emp_admin;
```

建立包体 CREATE PACKAGE BODY

```
--
--  Package body for the 'emp_admin' package.
--
CREATE OR REPLACE PACKAGE BODY emp_admin
IS
  --
  --  Function that queries the 'dept' table based on the department
  --  number and returns the corresponding department name.
  --
  FUNCTION get_dept_name (
      p_deptno        IN NUMBER
  ) RETURN VARCHAR2
  IS
      v_dname         VARCHAR2(14);
  BEGIN
      SELECT dname INTO v_dname FROM dept WHERE deptno = p_deptno;
      RETURN v_dname;
  EXCEPTION
      WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE('Invalid department number ' ||
p_deptno);
          RETURN '';
  END;
  --
  --  Function that updates an employee's salary based on the
  --  employee number and salary increment/decrement passed
  --  as IN parameters.  Upon successful completion the function
  --  returns the new updated salary.
  --
```

```
  FUNCTION update_emp_sal (
      p_empno         IN NUMBER,
      p_raise         IN NUMBER
  ) RETURN NUMBER
  IS
      v_sal           NUMBER := 0;
  BEGIN
      SELECT sal INTO v_sal FROM emp WHERE empno = p_empno;
      v_sal := v_sal + p_raise;
      UPDATE emp SET sal = v_sal WHERE empno = p_empno;
      RETURN v_sal;
  EXCEPTION
      WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE('Employee ' || p_empno || ' not found
');
          RETURN -1;
      WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE('The following is SQLERRM:');
          DBMS_OUTPUT.PUT_LINE(SQLERRM);
          DBMS_OUTPUT.PUT_LINE('The following is SQLCODE:');
          DBMS_OUTPUT.PUT_LINE(SQLCODE);
          RETURN -1;
  END;
  --
  --  Procedure that inserts a new employee record into the 'emp'
table.
  --
  PROCEDURE hire_emp (
      p_empno         NUMBER,
      p_ename         VARCHAR2,
      p_job           VARCHAR2,
      p_sal           NUMBER,
      p_hiredate      DATE,
      p_comm          NUMBER,
      p_mgr           NUMBER,
      p_deptno        NUMBER
  )
  AS
  BEGIN
      INSERT INTO emp(empno, ename, job, sal, hiredate, comm, mgr,
deptno)
          VALUES(p_empno, p_ename, p_job, p_sal,
                 p_hiredate, p_comm, p_mgr, p_deptno);
  END;
  --
  --  Procedure that deletes an employee record from the 'emp' table
based
  --  on the employee number.
  --
  PROCEDURE fire_emp (
      p_empno         NUMBER
  )
  AS
  BEGIN
      DELETE FROM emp WHERE empno = p_empno;
  END;
```

```
END;
```

8.4 常用管理函数

RDS for PPAS由于没有对外开放超级用户，用户无法像线下使用PPAS那样使用superuser账号管理数据库对象。为此，我们推出了一组管理函数，帮助用户顺利使用云上PPAS的各种功能。

管理函数的使用规则

在云上的各类管理函数都要求用户使用 RDS 根账号来执行。RDS 根账号是分配实例时指定的管理账号，具有 createdb createrole login 权限。

- 插件管理函数 rds_manage_extension。

  该函数帮助用户管理云上的插件，用户可以使用该函数创建和删除 PPAS 目前已经支持的插件。

```
rds_manage_extension(operation text, pname text, schema text default
 NULL,logging bool default false)
  operation: create 或 drop
  pname:     支持的插件名
  schema: 插件创建到的目标模式
  logging: 插件创建时的日志信息
  目前支持的插件有:
  pg_stat_statements
  btree_gin
  btree_gist
  chkpass
  citext
  cube
  dblink
  dict_int
  earthdistance
  hstore
  intagg
  intarray
  isn
  ltree
  pgcrypto
  pgrowlocks
  pg_prewarm
  pg_trgm
  postgres_fdw
  sslinfo
  tablefunc
  tsearch2
  unaccent
  postgis
  postgis_topology
  fuzzystrmatch
  postgis_tiger_geocoder
  plperl
  pltcl
  plv8
  "uuid-ossp"
  plpgsql
  oss_fdw
  举例:
  1 创建插件 dblink
```

```
    select rds_manage_extension('create','dblink');
2 删除插件 dblink
    select rds_manage_extension('drop','dblink');
```

- 当前连接会话 rds_pg_stat_activity()。

  该函数类似 pg_stat_activity 视图，返回用户相关的所有连接会话信息。

- 查看慢 SQL 的函数 rds_pg_stat_statements()。

  该函数是视图 pg_stat_statements 的封装，目的是让用户查看自己权限范围内的慢SQL。

- 性能分析函数。

  本组函数，类似 Oracle AWR 报告，提供给用户一组函数帮助用户分析目前 PPAS 实例的试试性能信息。

```
1 rds_truncsnap ()
说明：删除目前保存的所有快照。
2 rds_get_snaps()
说明：获得目前保存的所有快照信息。
3 rds_snap()
说明：产生一个实时快照。
4 rds_report(beginsnap bigint, endsnap bigint)
制定一个初始快照变化和结束快照变化，产生基于快照的性能分析报告。
举例：下面是一个通过产生快照生成性能分析报告的过程
SELECT * FROM rds_truncsnap(); //删除之前保存的快照
SELECT * from rds_snap();    // 产生一个快照
SELECT * from rds_snap(); // 产生一个快照
SELECT * from rds_snap(); // 产生一个快照
SELECT * FROM rds_get_snaps(); //获取目前产生的快照ID: 1 2 3
SELECT * FROM edbreport(1, 3); //根据快照产生一个性能分析报告
```

- 终止会话函数。

```
rds_pg_terminate_backend(upid int)
 rds_pg_cancel_backend(upid int)
 该函数分别对应原生的 pg_terminate_backend 和 pg_cancel_backend，区别仅是他们无法操作 supueruser 建立的连接。
 举例：终止进程号为 123456 的回话
 select rds_pg_cancel_backend(123456);
```

- VPD 函数。

  VPD 即 Virtual Private Database，是兼容 Package DBMS_RLS 的一种封装，参数完全相同。

```
1 rds_drop_policy 对应 DBMS_RLS.DROP_POLICY
  2 rds_enable_policy 对应 DBMS_RLS.ENABLE_POLICY
  3 rds_add_policy 对应 DBMS_RLS.ADD_POLICY
```

VPD 参考链接