云数据库 RDS 版

快速入门PPAS版

快速入门PPAS版

使用限制

为保障实例的稳定及安全，云数据库PPAS版有部分使用上的限制，详情如下。

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RDS管理控制台或OPEN API操作重启实例。
网络设置	若实例的访问模式是高安全模式，禁止在SNAT模式下开启net.ipv4.tcp_timestamps。

使用流程

文档目的

快速入门旨在介绍如何创建RDS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。

目标读者

首次购买RDS实例的用户。

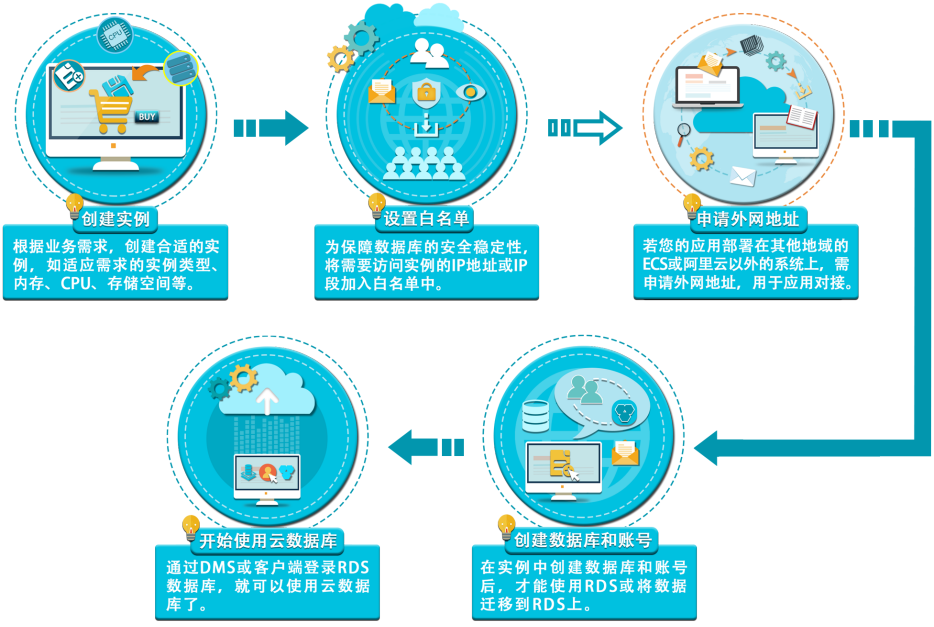
新建实例后需要对其进行基本设置的用户。

想要了解如何连接RDS实例的用户。

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使用限制以及阿里云RDS管理控制台。

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：



创建实例

您可以通过阿里云RDS管理控制台或API创建RDS实例。关于实例计费说明，请参见计费方式。本文将介绍在RDS管理控制台上创建实例的步骤，关于如何通过API创建实例的信息，请参见创建RDS实例。

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。

操作步骤

登录RDS管理控制台。

在**实例列表**页面，单击**新建实例**，进入**创建**页面。

选择**包年包月**或**按量付费**。关于计费方式的选择，请参见计费方式。

选择实例配置，参数说明如下。

基本配置

地域、可用区：选择实例所在的地域和可用区，有的地域支持单可用区和多可用区，有的地域只支持单可用区。关于地域和可用区详情，请参见地域和可用区。

> **注意：**不同地域内的产品内网不互通，且购买后不能更换地域，请谨慎选择。

数据库类型：RDS支持的数据库类型有MySQL、SQL Server、PostgreSQL和PPAS，不同地域支持的数据库类型不同，具体请以实际界面为准。

版本：指数据库版本。目前，RDS支持的数据库版本包括MySQL 5.5/5.6/5.7、SQL Server 2008 R2/2012、PostgreSQL 9.4和PPAS 9.3。不同地域所支持的数据库版本不同，请以实际界面为准。

> 选用MySQL数据库时，建议您选择5.6版本，因其支持TokuDB存储引擎，能极大降低数据文件占用空间，可节约存储费用。
>
> SQL Server 2008 R2和SQL Server 2012版本所支持的功能有差异，详情请参见SQL Server 2008 R2/2012功能差异。

系列：RDS实例支持基础版、高可用版和金融版，不同数据库版本所支持的系列不同，请以实际界面为准。

网络类型：RDS支持经典网络和专有网络（Virtual Private Cloud，简称VPC）。专有网络需要事先创建，或者您也可以在创建实例后再更改网络类型，具体请参见设置网络类型。

规格：实例的CPU和内存。不同规格对应不同的连接数和最大IOPS（即读和写分别可以达到的最大值，混合读写最高可以达到指标的2倍）。关于实例规格详情，请参见实例规格表。

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。

购买时长：选择包年或包月的时长。

购买量：购买相同配置的实例个数。

单击**立即购买**，进入**订单确认**页面。

提示：若您需要购买多个不同配置的实例，可以单击**加入清单**将要购买的实例逐个加入购买清单中，然后在**购买清单**中单击**批量购买**，如下图所示。

阅读**关系型数据库RDS服务条款**后，根据后续提示完成支付流程。

包年包月实例：单击**去支付**后，您可以选择付款方式，如使用账户余额、代金券、支付宝等。

按时付费实例：单击**去开通**，实例创建完成。

初始化配置

设置白名单

为了数据库的安全稳定，在开始使用RDS实例前，您需要将访问数据库的IP地址或者IP段加到目标实例的白名单中。正确使用白名单可以让RDS得到高级别的访问安全保护，建议您定期维护白名单。本文将主要介绍设置白名单的操作步骤。

背景信息

访问数据库有如下三种情况，关于不同连接类型（内网和外网）的适用场景，请参见**设置内外网地址**中的**背景信息**介绍。

外网访问RDS数据库

内网访问RDS数据库

内外网同时访问RDS数据库

在您设置实例的连接类型之前，您需要先将应用服务或ECS的IP地址或IP段添加到RDS实例的白名单中。当您设置好白名单后，系统会自动为您生成内网地址。若您需要使用外网地址，请**申请外网地址**。

说明：如果将应用服务IP加入白名单后，还是无法连接RDS，请获取应用服务真实IP。

注意事项

系统会给每个实例创建一个默认的**default**白名单分组，该白名单分组只能被修改或清空，但不能被删

除。

对于新建的RDS实例，系统默认会将回送地址127.0.0.1添加到**default**白名单分组中，IP地址127.0.0.1代表禁止所有IP地址或IP段访问该RDS实例。所以，在您设置白名单时，需要先将127.0.0.1删除，然后再添加您允许访问该RDS实例的IP地址或IP段。

若将白名单设置为%或者0.0.0.0/0，代表允许任何IP访问RDS实例。该设置将极大降低数据库的安全性，如非必要请勿使用。

操作步骤

登录RDS管理控制台。

选择目标实例所在地域。

单击目标实例的ID，进入**基本信息**页面。

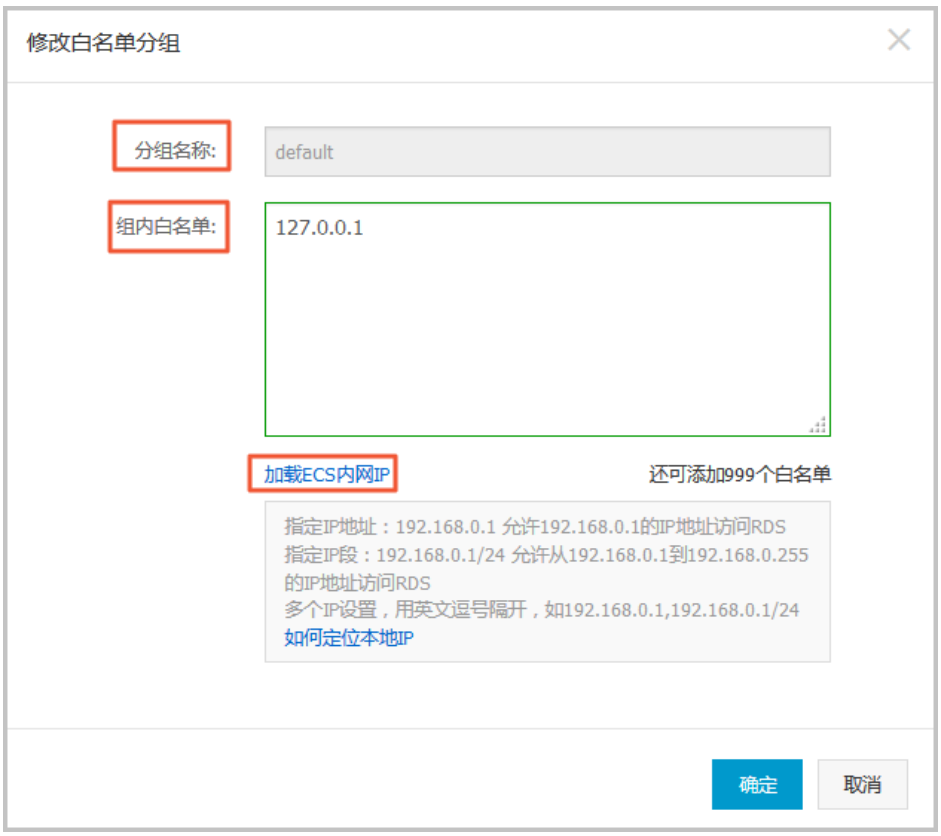
在左侧导航栏中选择**数据安全性**，进入**数据安全性**页面。

在**白名单设置**标签页面中，单击**default**白名单分组中的**修改**，如下图所示。

> **提示**：若您想使用自定义分组，请先单击**default**白名单分组中**清空**以删除默认分组中的IP地址127.0.0.1，然后单击**添加白名单分组**新建自定义分组，其余操作步骤与下述步骤相似。

在**修改白名单分组**页面，在**组内白名单**栏中填写需要访问该实例的IP地址或IP段。若您需要添加ECS的内网IP，请单击**加载ECS内网IP**，然后根据提示选择IP。如下图所示。

> 说明：当您在**default**分组中添加新的IP地址或IP段后，回送地址127.0.0.1会被自动删除。



参数说明：

分组名称：长度为2~32个字符，由小写字母、数字或下划线组成，开头需为小写字母，结尾需为字母或数字。在白名单分组创建成功后，该名称将不能被修改。

组内白名单：填写允许访问RDS实例的IP地址或者IP段。

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，请用英文逗号隔开（逗号前后都不能加空格），例如192.168.0.1,172.16.213.9。

在每个白名单分组中，MySQL、PostgreSQL和PPAS类型的RDS实例可以添加1000个IP，SQL Server类型的RDS实例可以添加800个IP。

加载ECS内网IP：单击该按钮后，将显示同账号下每个ECS实例对应的IP地址，可用于快速添加ECS内网IP到白名单中。

单击**确定**。

修改或删除白名单分组

您可以根据业务需求修改或删除白名单分组，操作步骤如下：

登录RDS管理控制台。

选择目标实例所在地域。

单击目标实例的ID，进入**基本信息**页面。

在左侧导航栏中选择**数据安全性**，进入**数据安全性**页面。

在**白名单设置**标签页面中，单击目标白名单分组中的**修改**或**删除**。

完成修改白名单或确认要删除该白名单分组后，单击**确定**。

申请外网地址

如果您的应用部署在与您的RDS实例在同一地域且网络类型相同的ECS上，则无需申请外网地址。如果您的应用部署在与您的RDS实例在不同地域或网络类型不同的ECS或者阿里云以外的系统上，需申请外网地址，用于应用对接。

说明：只要在同一地域内（可用区可以不同）且网络类型相同的实例，就可以内网互通。

背景信息

RDS提供两种连接地址，即内网地址和外网地址。实例的访问模式和实例版本对连接地址的选择有如下限制。

实例系列	实例版本	访问模式	连接地址
单机基础版	- MySQL 5.7 - SQL Server 2012	标准模式	- 内网地址 - 外网地址 - 内网地址和外网地址
双机高可用版	- MySQL 5.5/5.6 - SQL Server	标准模式	- 内网地址 - 外网地址
		高安全模式	- 内网地址 - 外网地址 - 内网地址和

	2008 R2 - PostgreSQL 9.4 - PPAS 9.3		
金融版	MySQL 5.6	标准模式	- 内网地址 - 外网地址
		高安全模式	- 内网地址 - 外网地址 - 内网地址和外网地址

连接地址的使用场景如下所示：

单独使用内网地址：

系统默认提供内网地址，您可以直接修改连接地址。

适用于应用部署在与您的RDS实例在同一地域的ECS上且RDS实例与ECS的网络类型相同时。

单独使用外网地址：

适用于应用部署在与您的RDS在不同地域的ECS上时。

适用于应用部署在阿里云以外的系统上时。

同时使用内外网地址：

适用于应用中的模块同时部署在与您的RDS实例在同一地域且网络类型相同的ECS上和与您的RDS实例在不同的ECS上时。

适用于应用中的模块同时部署在与您的RDS实例在同一地域且网络类型相同的ECS上和阿里云以外的系统上时。

注意事项

在访问数据库前，您需要将访问数据库的IP地址或者IP段加入白名单，操作请参见设置白名单。

RDS会针对外网地址流量收取一定费用，详细收费标准请参见云数据库RDS详细价格信息。

外网地址会降低实例的安全性，请谨慎选择。为了获得更快的传输速率和更高的安全级别，建议您将应用迁移到与您的RDS在同一地域的阿里云服务器ECS上。

操作步骤

登录RDS管理控制台。

选择目标实例所在地域。

单击目标实例的ID，进入**基本信息**页面。

在左侧导航栏中选择**数据库连接**，进入**数据库连接**页面。

单击**申请外网地址**，如下图所示。

在弹出的信息确认框中单击**确定**，生成外网地址。

单击**修改连接地址**，在弹出的窗口中设置内外网连接地址及端口号，如下图所示。

参数说明：

连接类型：根据需要修改连接类型为**内网地址**或者**外网地址**。

连接地址：地址样式为xxx.mysql.rds.aliyuncs.com，其中*xxx*为自定义字段，由字母和数字组成，开头需小写字母，8-64个字符。

端口：RDS对外提供服务的端口号，取值范围是3200~3999之间的任意一个整数。

单击**确定**。

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。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

登录RDS管理控制台。

选择目标实例所在地域。

单击目标实例的ID，进入**基本信息**页面。

在左侧导航栏中，选择**账号管理**，进入**账号管理**页面。

单击**创建初始账号**。

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，由字母、数字、中划线或下划线中的任意三种组成。

确认密码：输入与密码一致的字段，以确保密码正确输入。

单击**确定**。

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。

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。

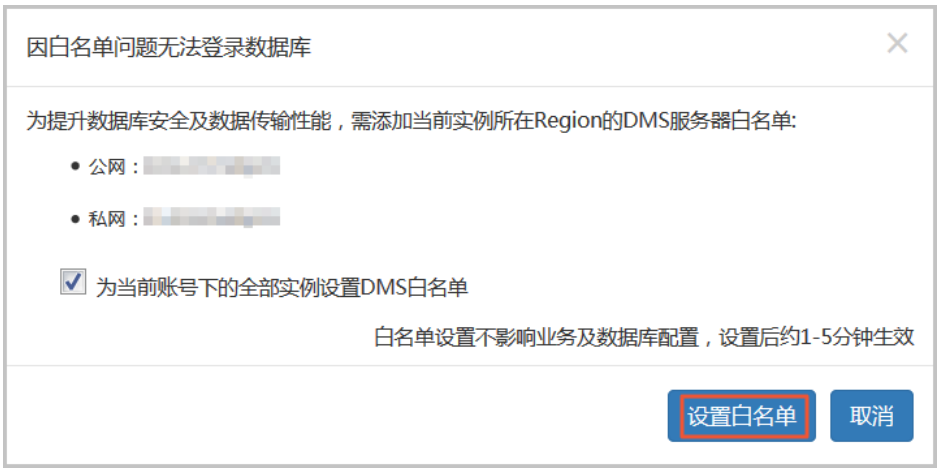
2：要访问数据库的账号名称。

3：上述账户所对应的密码。

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**。

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。



成功添加白名单后，单击**登录**。

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**。

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;
```

单击**执行**，完成创建数据库。

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';
```

单击**执行**，完成创建普通账号。

连接实例

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。

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与要访问的RDS实例在同一地域的ECS上且RDS实例与ECS的网络类型相同时，请使用内网地址。

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，请参见通过DMS登录RDS数据库。

通过客户端登录

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。

启动pgAdmin 4客户端。

右击**Servers**，然后选择**创建** > **服务器**，如下图所示。

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。

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：

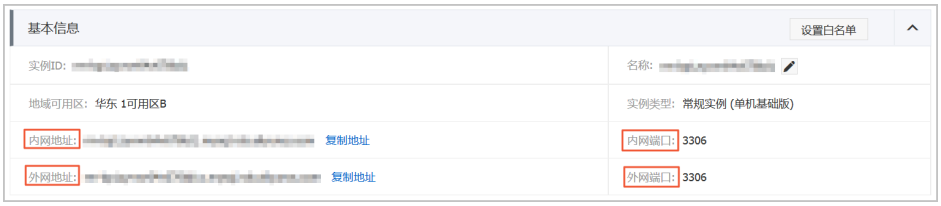
主机名称/地址：若使用内网连接，需输入RDS实例的内网地址。若使用外网连接，需输入RDS实例的外网地址。查看RDS实例的内外网地址及端口信息的步骤如下：

登录RDS管理控制台。

选择目标实例所在地域。

单击目标实例的ID，进入**基本信息**页面。

在基本信息栏中，即可查看内外网地址及内外网端口信息，如下图所示：



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。

单击**保存**。

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。

提示：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。

附录

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，myuser 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. 建立有登录权限的用户。

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

参数说明如下：

- USER：要创建的用户名，如 **newuser**
- password：用户名对应的密码，如 **password**

为新用户建立 schema。

```
CREATE SCHEMA newuser;
GRANT newuser to myuser;
ALTER SCHEMA myuser OWNER TO newuser;
REVOKE newuser FROM myuser;
```

说明：

- 如果在进行 ALTER SCHEMA newuser OWNER TO newuser 之前没有将 newuser 加入到 myuser 角色，将会出现如下权限问题：

```
ERROR:  must be member of role "newuser"
```

- 从安全角度出发，在处理完 OWNER 的授权后，请将 newuser 移出 myuser 角色以提高安全性。

3. 使用 newuser 登录数据库。

```
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.
```

附录：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

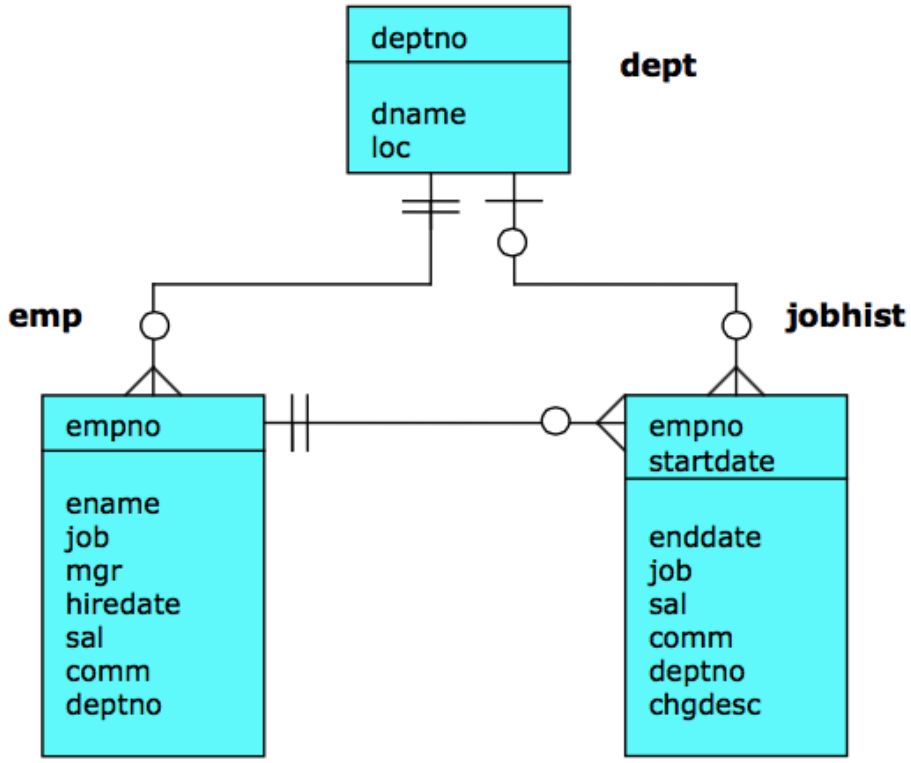
单击 此处 下载 PPAS 开发驱动程序。

- 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

PPAS 兼容性说明

通过本文档中的示例，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

注意：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;
```

常用管理函数

RDS 上 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的 PPAS 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