ALIBABA CLOUD

阿里云

数据库和应用迁移
用户指南

文档版本：20220424

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.使用流程 07
2.数据库评估 08
2.1. 数据库采集 08
2.2. 数据库画像 14
2.3. 目标库选型建议 16
2.4. 数据库评估分析 17
3.新建数据库档案 20
4.线上改造迁移 23
4.1. 线上改造迁移综述 23
4.2. 新建迁移项目 24
4.3. 配置白名单 25
4.4. 预检查 26
4.5. 源库计划校验 27
4.6. 结构迁移 29
4.7. 定制化结构迁移 30
4.8. 结构订正 31
4.9. 增量源库对比 32
5.数据库割接 34
5.1. 新建割接项目 34
5.2. 业务验证 36
5.3. 建立正向增量同步 39
5.4. 割接验证 39
5.5. 回退验证 40
5.6. 生产割接 41
6.应用评估改造 43
6.1. 应用评估改造摘要 43

6.2. 应用采集 43
6.3. 采集部署 46
6.4. 应用画像 49
6.5. 应用评估 51
6.6. 应用静态改造 55
7.结构订正指南 57
7.1. 不支持FOR UPDATE OF 57
7.2. 不支持自动转换Sample语句 57
7.3. 不支持Bitmap index 58
7.4. 不支持Cluster index 58
7.5. 不支持异常类型 59
7.6. 不支持AGGREGATE关键字 71
7.7. 在PL/SQL中不支持调用其他语言代码 73
7.8. 不支持PARALLEL_ENABLE 74
7.9. 不支持PIPELINED关键字 75
7.10. PolarDB系统保留列名 77
7.11. Trigger不支持非DML事件 78
7.12. Aggregate函数不支持keep关键字 79
7.13. SYS_CONTEXT（）函数支持SESSION_USER、CURRENT_USER... 80
7.14. 不支持USERENV 81
7.15. Varray类型的元素类型不能是刚定义的类型 82
7.16. dbtimezone 83
7.17. NESTED TABLE 84
7.18. REGEXP_LIKE函数 86
7.19. EXTRACTVALUE函数 87
7.20. Unpivot 列转行 88
7.21. DBMS_XMLGEN函数 90
7.22. 不支持DBMS_METADATA.GET_DDL 91

7.23. Date - Date结果不兼容 93
7.24. Forall and Bulk Collect 94
7.25. Interval分区 95
8.迁移实验室 98
8.1. SQL Adapter 98
8.2. SQL周期性采集 103
9.RAM 子账号 109

1.使用流程

ADAM（数据库与应用迁移）使用主要分为数据库评估，数据改造迁移，应用评估改造三个流程。

整体迁移流程简介

1. 数据库评估：帮助用户了解源库现状、提供目标库的选型建议，根据评估结果为用户迁移到目标库提供可行性参考。
2. 数据改造迁移：帮助用户做数据库结构迁移、结构订正、增量结构迁移，并对接数据迁移。
3. 应用评估改造：帮助用户分析应用与数据库的关联关系，并分析从源库迁移到目标库的应用改造点与改造位置。

1. 数据库评估

1. 数据库采集
2. 数据库画像
3. 目标库选型建议
4. 数据库评估分析

基于以上操作，可以评估出Oracle库适合迁移的目标库及兼容度、改造工作量、成本等，用户可根据数据库评估分析里**对象兼容度**详情和**SQL兼容度**详情，根据指导建议自助完成数据库和应用改造。

2. 数据库改造迁移

线上改造迁移：

1. 新建迁移项目
2. 预检查
3. 源库计划校验
4. 结构迁移
5. 定制化结构迁移
6. 增量源库对比

3. 应用评估改造

1. 应用采集
2. 应用画像
3. 应用评估
4. 应用静态改造

> **说明** 应用动态评估改造请使用1，2，3，应用静态改造请使用4。

2.数据库评估

2.1. 数据库采集

ADAM提供在线采集和下载数据采集器采集两种数据库采集方式，您可以根据源库能否通过公网访问自行选择。

背景信息

- 在线采集：需要您将源数据库的网络打通，并在数据库白名单中添加ADAM服务器，确保ADAM服务器可以直接通过在线采集方式进行信息收集。
  - 有公网IP：源数据库具有公网IP，可以通过外网访问。
  - 无公网IP：通过数据库网关DG连接：数据库网关DG（Database Gateway）是一种支持私网数据库远程访问的数据库连接服务。通过数据库网关DG，您可以无需开通公网地址访问并管理本地IDC或其他云厂商的数据库。详情请参见数据库网关DG。

    > **说明**
    >
    > 目前ADAM支持华北2（北京）、华北3（张家口）、华南1（深圳）、华东1（杭州）、华东2（上海）地域的DG实例。

- 下载采集器：对于源库无法通过外网连通的情况，可以下载ADAM数据采集器进行采集。

在线采集

1. 登录ADAM控制台。
2. 单击左侧导航栏中的**数据库评估**。
3. 在**数据库采集**页签下，单击**在线数据库采集**。
4. 单击**创建采集任务**，开始创建采集任务。
5. 登录源数据库配置采集账号，赋予相应的权限。
   - Oracle 10g/11g/12c/18c/19c（非 CDB 模式，创建 LOCAL USER 类型用户）。
     a. 创建采集用户eoa_user, 并设置密码为eoaPASSW0RD。

     ```
     create user eoa_user identified by eoaPASSW0RD default tablespace users;
     ```

     b. 查询权限。

     ```
     grant connect,resource,select_catalog_role,select any dictionary to eoa_user;
     ```

c. DBMS_LOGMNR权限。

> 说明
>
> 版本为 10g 的数据库需要先执行
>
> ```
> create or replace public synonym DBMS_LOGMNR for sys.dbms_logmnr;
> ```

```
grant execute on DBMS_LOGMNR to eoa_user;
```

d. DBMS_METADATA权限，查询数据对象DDL语句。

```
grant execute on DBMS_METADATA to eoa_user;
```

e. 查询事务权限。

```
grant select any transaction to eoa_user;
```

f. 查询表权限。

```
grant select any table to eoa_user;
```

g. 分析表权限。

```
grant analyze any to eoa_user;
```

h. 产生随机编号权限。

```
grant execute on dbms_random to eoa_user;
```

- **Oracle 12c/18c/19c**（CDB 模式，需要连接到 CDB，创建 COMMON USER 类型用户）。

```
create user c##eoa_user identified by "eoaPASSW0RD" default tablespace users;
grant connect,resource,select_catalog_role,select any dictionary to c##eoa_user conta
iner=all;
grant execute on DBMS_LOGMNR to c##eoa_user container=all;
grant execute on dbms_metadata to c##eoa_user container=all;
grant select any table to c##eoa_user container=all;
grant select any transaction to c##eoa_user container=all;
grant analyze any to c##eoa_user container=all;
grant execute on dbms_random to c##eoa_user container=all;
alter user c##eoa_user set container_data=all container=current;
```

- ADAM支持 TD 13 14 15版本的数据库采集。

a. 需要使用具有访问DBC权限的账号。

```
grant select,show on dbc to （username）
```

- ADAM支持Db2 LUW版本的数据库采集。

a. 需要db2look相关权限,采集账号请授予DBA权限执行。

6. 完成源数据库账号配置后，单击**下一步**进入**创建采集任务**面板。

i. **采集任务名**：自定义采集任务名称。

ii. **源库类型**。

- 源库类型选择**ORACLE**。

  服务器/SID：源Oracle数据库的服务名或SID。

- 源库类型选择**TERADATA**或者**Db2_LUW**。

  数据库名：源数据库的数据库名。

iii. 源库网络种类。

a. 选择**具有公网IP的数据库**。

b. 选择**无公网IP：Port的数据库（通过数据库网关DG接入）**。

- **DG区域**：选择源库对应的DG实例所属的地域。
- **DG实例**：选择源库对应的DG实例ID。

iv. 填写源数据库其他信息。

- **主机IP**：源数据库的IP。

> **说明**
>
> 在源库中添加ADAM白名单，以保证ADAM能够成功在线采集数据。ADAM白名单，请参见添加白名单。

- **端口**：源数据库连接端口。
- **用户名**：步骤5中创建的采集用户。
- **口令**：上述用户名对应的密码。
- （可选）**高级配置项**：**编码配置**中输入源数据库的编码方式。

7. 填写完毕后，单击**链接测试**，测试通过后，单击**启动采集**。
8. 待采集任务完成后，勾选采集任务，点击**下一步创建画像**，进入生成数据库画像流程。

下载采集器采集数据

如果源库不能对外使用公网，或者无法从云上访问，需要使用线下流程采集。

1. 下载 ADAM 采集器。

i. 登录 ADAM 控制台。

ii. 单击左侧导航栏中的**数据库评估**。

iii. 在**数据库采集**页签下，单击**下载采集器**。

iv. 根据目标安装 ADAM 客户端设备的操作系统，选择下载对应版本的 ADAM 客户端到本地并解压缩。

> **说明**
>
> 建议不要在待采集的数据库设备上运行。目标安装 ADAM 客户端设备最低应满足以下配置要求：
>
> - 网络：能够连接到待采集的源数据库。
> - CPU：2 core。
> - 内存：8GB。
> - 硬盘：100GB 空余。

v. 配置采集账号，赋予相应的权限。

a. 登录源数据库。

b. 使用具有 SYSDBA 权限的账号创建临时账号，并配置以下权限（如果您已有包含下面权限的账号，请忽略此步骤，直接使用）。

- Oracle 10g/11g/12c/18c/19c（非 CDB 模式，创建 LOCAL USER 类型用户）

  a. 创建采集用户eoa_user, 并设置密码为eoaPASSW0RD。

  ```
  create user eoa_user identified by eoaPASSW0RD default tablespace users;
  ```

  b. 查询权限。

  ```
  grant connect,resource,select_catalog_role,select any dictionary to eoa_u
  ser;
  ```

  c. DBMS_LOGMNR权限。

  > 说明
  >
  > 版本为 10g 的数据库需要先执行：
  >
  > ```
  > create or replace public synonym DBMS_LOGMNR for sys.dbms_logmnr;
  > ```

  ```
  grant execute on DBMS_LOGMNR to eoa_user;
  ```

  d. DBMS_METADATA权限，查询数据对象DDL语句。

  ```
  grant execute on dbms_metadata to eoa_user;
  ```

  e. 查询事务权限。

  ```
  grant select any transaction to eoa_user;
  ```

  f. 查询表权限。

  ```
  grant select any table to eoa_user;
  ```

  g. 分析表权限。

  ```
  grant analyze any to eoa_user;
  ```

  h. 产生随机编号权限。

  ```
  grant execute on dbms_random to eoa_user;
  ```

- Oracle 12c/18c/19c（CDB 模式，需要连接到 CDB，创建 COMMON USER 类型用户）。

```
create user c##eoa_user identified by "eoaPASSW0RD" default tablespace users;
grant connect,resource,select_catalog_role,select any dictionary to c##eoa_us
er container=all;
grant execute on DBMS_LOGMNR to c##eoa_user container=all;
grant execute on dbms_metadata to c##eoa_user container=all;
grant select any table to c##eoa_user container=all;
grant select any transaction to c##eoa_user container=all;
grant analyze any to c##eoa_user container=all;
grant execute on dbms_random to c##eoa_user container=all;
alter user c##eoa_user set container_data=all container=current;
```

- Teradate 13/14/ 15

  a. 需要使用具有访问DBC权限的账号。

  ```
  grant select,show on dbc to （username）
  ```

- Db2 LUW

  a. 需要db2look相关权限,采集账号请授予DBA权限执行。

c. 采集数据库结构性数据（用于生成可行性报告及兼容报告）。

采集器现已支持Oracle 10g/11g/12c/18c/19c版本，Teradate 13 / 14 / 15版本，Db2_LUW版本。（如果采集中遇到任何问题请提交工单，工单中请附上同级目录 logs 下的日志文件。）

a. 执行采集命令（.bat 是在 Windows 环境下命令；.sh 是在 Linux 环境下命令）。

a. Oracle 10g

```
collect_10g[.sh|.bat] -h -u -p -d <service_name>
```

b. Oracle 11g

- Oracle 11g R1

```
collect_11gR1[.sh|.bat] -h -u -p -d <service_name>
```

- Oracle 11g R2

```
collect_11gR2[.sh|.bat] -h -u -p -d <service_name>
```

c. Oracle 12c/18c/19c

> **说明**
>
> 针对Oracle 12c的某个pdb进行采集时，请参考Oracle 11g采集操作说明，使用 `collect_11gR2` 脚本进行采集操作。collect_12c[.sh|.bat] -h <host> -u <username> -p <password> -P <port> -d <service_name> -s <sid>

d. Teradate 13 / 14 / 15

```
collect_td[.sh] -h ip -p password -u username
```

e. Db2_LUW

```
collect_db2_luw[.sh] -h ip -u username -p password -d databasename  -P port
```

-h: 采集数据库的IP地址。 -u: 采集用户 eoa_user。 -p: 采集用户 eoa_user 密码 eoaPASSW0RD。 -P: 采集数据库的端口，如：1521。 -d: 采集数据库的服务名, 12c是指特定PDB的服务名。 -s: 采集数据库实例名。

b. 导出采集结果。

采集完成后，会提示用户生成数据包，并提示数据包路径。日志文件如下：

```
[***] ********************************************************************
[***] *     Collect Successfully!
[***] *
[***] * Complete the file packaging, the package result path is:
[***] *        ~rainmeter/out/data.zip ***********************************
*****************************
```

c. 迁移结束后，清除临时账号。

使用具有 SYSDBA 权限的账号通过终端连接数据库，并执行下面 SQL。

a. Oracle 10g/11g/12c/18c/19c（非 CDB 模式）

```
drop user eoa_user cascade;
```

b. Oracle 12c/18c/19c（CDB 模式）

```
drop user c##eoa_user cascade;
```

> **说明**
>
> ADAM数据库采集器对源数据库的内存占用很少，可以不考虑；其对CPU要求也不高，当涉及到数据库对象的getddl操作时，会在采集开始时CPU负载有一些增加，负载增加量取决于源数据库规格和当前负载。建议在业务低峰期采集，可以提升采集速度，通常业务低峰期采集只需要较短时间就可以完成。

2.2. 数据库画像

数据库画像是数据库评估的基础数据，可以帮助您更好地了解自己的源数据库，在数据库迁移、改造等阶段，可以快速查找源数据库信息，指导迁移与改造。

新建画像

1. 登录ADAM控制台。
2. 在左侧导航栏，单击**数据库评估**。
3. 单击**源库画像分析**，在区域左上角单击 **新建画像**。
4. 在**新建画像**面板中，输入**画像名**，选择数据**类型**，选择**报告语言类型**，单击**上传**，上传数据文件，上传完成后单击**创建**。源库画像创建时间大约为1~30 min，创建时间受源库采集内容的影响。

画像详情

源库画像状态变为**完成**后，在操作列单击**详情**，查看画像分析。

单击**概要**页签，查看源库画像的概要。

- 基本信息统计

  源库画像基本信息主要从复杂度、会话、风险、热点、规模和负载6个维度展示。

  - 会话：衡量数据库的连接状况。分数越高，说明同时连接数据库的会话越多。

- 复杂度：根据数据库的使用场景特性，特性等维度，计算源库画像复杂度。
- 风险：衡量数据库是否存在SQL或对象的执行性能风险。分数越高，说明数据库存在风险的可能性越大。
- 热点：衡量数据库是否存在访问频率非常集中的对象。热点分数越高，说明数据库存在部分数据对象被集中访问的情况 。
- 规模：衡量数据库资源规模的情况。分数越高，说明采集的源数据库规模越大。
- 负载 ： 衡量数据库运行性能的情况。分数越高，说明数据库运行负载使用率高。

- 主要对象分布

  统计源库画像中对象类型的分布。

单击**明细**页签，查看源库画像**性能**、**容量**、**特性**、**外部依赖**、**其他维度**、**对象详情**和**全景搜索**多维度的分析结果。

- 性能

  统计数据库的QPS、TPS、CPU使用率和负载信息。

- 容量

  提供Schema容量排序和对象类型容量占比（表，索引，LOB字段）。

- 特性

  提供一级、二级数据库语法特性的详细展示，帮助用户全面了解自己的源数据库使用了哪些特性。特性以左树右表形式呈现，左树包含该数据库采集到的所有特性列表，点击某个特性，右边列表展示具有该特性的数据库对象。搜索框支持任意字段的查询功能。

- 外部依赖

  提供DBLink列表和详情信息， 对于包含外部依赖的数据库，改造时需要考虑跨库查询问题。

- 其他维度

  ADAM对特殊类型的表和SQL进行识别分析，例如：无主键表，高增长表，带聚合函数SQL等。单击**查看**，查看对象的详细信息。

- 对象详情

  提供多维度对象信息，包括对象特征标签、对象关联关系（依赖的对象，有外键关联的表等），对象包含的特性等。通过对象详情，您可以在不查询源数据库的情况下，点击每个对象的**全景展示**，看到对象详情。

  对象全景展示两部分，上半部分展示对象的基本信息以及详细DDL。下半部分展示对象的特征，依赖，特性等ADAM智能分析出的数据。

- 全景搜索

  全景搜索功能方便用户在数据库应用改造时，快速查找目标对象。搜索功能支持全部匹配，模糊匹配，按类型匹配等多种复合搜索功能，可以任意组合查找画像中的对象信息。按对象schema、DDL、类型和标签等搜索对象，同时可以查看对象的各种关联关系及标签。

追加画像

源库画像支持画像的多版本管理。如果创建画像后，数据库里的数据发生了变化，您可以在原画像的基础上添加新的采集文件。

在**源库画像**配置向导页面，单击操作列中的**追加**，**上传**新采集到的数据文件，单击**创建**，系统会根据全部数据文件自动生成新的画像版本。

授权画像

您可以单击操作列中的**授权**，授权其他用户访问该画像。授权有效期1个月。授权后，其他用户可以查看画像结果，使用该画像创建新的分析项目。

> **注意**
>
> 请妥善使用授权功能，由于授权引起的信息泄露，责任归授权方。

您也可以随时取消授权，收回其他用户访问该画像的权限。

删除画像

对于不再需要的画像，您可以删除该画像，同时该画像生成的所有分析项目也自动删除。

选择目标画像操作列后面的**...**，单击**删除**，删除画像。

后续步骤

选中目标画像，单击**下一步查看目标库选型建议**，进入目标库选型建议。

2.3. 目标库选型建议

阿里云上有很多种数据库，您可能一时无法对迁移的目标库进行选择，目标库选型建议可以帮助您根据现有的数据库画像进行分析，给出迁移到各种目标库的兼容情况分析， 目前目标库选型建议以普遍用户最关心的目标库兼容度为参考依据，为您提供目标库决策的依据。

使用方法

1. 登录ADAM控制台。
2. 在左侧导航栏单击**数据库评估**。
3. 单击配置向导**目标库选型建议**，在源库画像列表中选择需要评估的数据库画像。
4. 单击**下一步查看目标库选型建议**，进入目标库选型指导页面。
5. 单击**兼容性**页签，查看当前数据库迁移到各类目标库的**数据库对象兼容性**和**SQL兼容性**。
   - 数据库对象兼容性包括兼容度、ADAM智能兼容、不兼容。
     - 兼容度：ADAM智能兼容的百分数。
     - ADAM智能兼容：包含直接兼容目标库的对象，以及可以通过ADAM智能转换的对象（使用ADAM数据库改造产品，可以自动帮助您创建ADAM智能兼容的对象到目标数据库）。
     - 不兼容：您必须手工修改数据库对象。（ADAM全面提供改造建议）。
   - 数据库SQL兼容性包括兼容度、兼容、改动后兼容，不兼容。
     - 兼容度：包含兼容的对象和改动后兼容的对象。
     - 兼容：包含直接兼容目标库的对象，以及可以通过ADAM智能转换的对象。
     - 改动后兼容： 数据库SQL改造后可以兼容，需要您对应用进行相应的SQL修改。
     - 不兼容：不能兼容的对象。
6. 对于只想迁移部分Schema的用户，您可以通过兼容性列表顶部的**Schema筛选**，选择自己想要迁移的Schema进行目标库选型建议。避免不想迁移的Schema影响建议的准确性。

7. 您也可以单击**类型推荐**页签，查看ADAM智能推荐的目标库。ADAM根据数据库画像，智能分析源数据库的使用场景（TP型、AP型、HTAP型或测试类型），结合兼容性情况，为您推荐最合适的目标库。

> **注意**
>
> 数据库预估类型受采集数据影响，实际选型时仅供参考。

数据库预估类型	类型说明
SAMPLE	小型或测试数据库。
OLTP	在线交易型数据库。
OLAP	在线分析型数据库。
HTAP	混合场景数据库。

2.4. 数据库评估分析

数据库兼容评估可以帮助您评估目标库兼容性，规格，迁移风险，让用户全面了解数据库上云的可行性以及改造工作量。

新建项目

1. 登录ADAM控制台。
2. 在左侧导航栏单击**数据库评估**。
3. 单击**源库画像分析**，选中一个画像，单击区域底部的**下一步查看目标选型建议**。
4. 在**目标库选型建议**配置向导页面底部，单击**下一步新建目标库评估**，新建目标库评估项目。
5. **新建项目。**
   - 项目名：项目名称，必填。
   - 源库画像：选择已创建的画像名。
   - 项目类型：选择目标数据库类型。
   - 目标库版本：目标数据库版本必选。
   - 内核版本：PolarDB O引擎的内核版本必选。
   - 报告语言：选择目标库兼容评估报告的语言。
6. 单击**创建**，新项目创建成功，并自动启动分析。

> **说明**
>
> 您也可以在**目标库兼容评估**配置向导页面，单击**+新建项目**，创建新项目。

评估综述

在项目列表中可以查看新建项目的分析状态和进度。待分析完成后，点击**详情**查看目标库兼容评估结果。数据库评估结果分三部分展示：项目概要，评估综述，评估详情。

数据库评估综述概括了源数据库迁移到目标库的兼容性、改造点，规格，风险和整体兼容性。

- 兼容性：衡量源数据库到目标数据库的兼容情况，兼容度越高，需要修改的对象与SQL越少。
- 改造：迁移到目标数据库需要改动的具体位置。
  - 对象改造点：使用ADAM数据库改造后无须用户自助改造。
  - 应用改造点：通过数据库采集的SQL分析得到，是对数据库迁移的初步评估。
- 规格：根据采集数据，通过ADAM智能计算出迁移到目标库需要的数据库规格和预估费用。 规格评估受采集环境影响，实际购买需要结合业务综合评估。
- 风险：对用户的迁移改造进行风险预警。包含源库已有的风险点，及迁移到目标库可能发出的风险点。
- 整体兼容性： 显示源数据库到目标数据库整体的兼容情况。

评估详情

评估详情包括六部分：对象兼容度，SQL兼容度，对象改造点、目标库规格、迁移风险和项目外部依赖(Schema)。

在**评估详情**区域，单击各评估项后面的**详情**，查看评估结果。

对象兼容度

Schema兼容性面板列出所有对象的兼容性评估结果，对象类型涉及源数据库的所有对象，包括兼容，不兼容，两种情况。

对修改后兼容的对象，ADAM给出转换后的DDL以及修改点。 对不兼容的对象，ADAM给出不兼容原因，以及修改建议。

SQL兼容度

SQL兼容度是对数据库中采集的SQL进行语法分析的结果。

单击**评估概要**页签，查看总体兼容统计信息。包括：兼容、不兼容，改动后兼容三种情况。

单击**评估详情**页签，查看对象的兼容度细节。包括兼容情况、源SQL和目标SQL。

- 单击**源SQL**下的**查看**，查看源SQL的详细信息。
- 单击**目标SQL**下的**查看**，查看目标SQL的详细信息。单击**错误信息**或**改变信息**页签，查看不兼容的原因或改变的信息。

> **说明**
>
> 对于数据库SQL，ADAM数据库评估是根据数据库记录的执行SQL进行兼容性分析，具体SQL是否为真实业务发出的，需要用户自行判断或者使用ADAM应用评估分析。

单击**规则详情**页签，查看兼容规则详情。

对象改造点

对象改造点主要是数据库对象的改造点汇总，用户可以按照改造点自行改造自己的数据库对象。也可以申请使用ADAM数据库改造功能，自动化改造数据库对象（少量人工订正）。

改造级别：对所有的对象进行分级，区分各个改造点的难易程度，方便项目改造人员直接根据改造点分配规划改造任务。改造级别越高，改造难度越大。

源DDL：单击其下的**查看**，查看源DDL代码。

目标DDL：单击其下的**查看**，查看目标DDL代码。

改造点：单击其下的**数据库**，查看数据库改造点的ID，详细改造方法等。

目标库规格

目标库规格为用户迁移到阿里云数据库提供规格与迁移计划指引。

配置是根据采集到的源库的配置，性能，SQL，外部依赖等以及目标库综合分析计算出的，对于迁移购买具有参考价值。

在**目标数据库方案**页签下，单击对象ID的**详情**操作，查看每个迁移组上的对象的详细信息。

在**跨库对象**页签下，查看跨库对象的**详情**。对于存在多个迁移实例的目标方案，可能存在跨库对象。

迁移风险

迁移风险分为源库风险与目标库分析。

源库风险是在源库采集到的SQL执行时耗费CPU,内存大的SQL列表，分为TOP CPU / TOP Buffer等类型，在测试时需要重点关注这些SQL。

目标库风险是改造数据库结构或者SQL在目标库执行可能存在风险。需要用户关注迁移风险点，避免异构数据库迁移造成的性能差异。

项目外部依赖(Schema)

项目外部依赖评估外部依赖的对象数量，并提供解决方案。

下载报告

ADAM除了提供页面信息展示，也提供了丰富的报告供不同场景人员使用。

在**评估详情**区域，单击**下载所有报告**，一键获取所有ADAM数据库评估相关报告。

后续步骤

在**评估详情**区域，单击**启动数据库改造**，进入数据库改造迁移流程。具体操作，请参见数据库改造迁移。

3.新建数据库档案

本文介绍在ADAM中新建数据库档案的方法。

前提条件

- 已创建目标数据库实例。
- 已创建数据库账号。
- 已设置集群的IP白名单，需要添加的白名单请参见白名单。

支持的数据库类型

- 源数据库：Oracle、Teradata、DB2 for LUW。
- 目标数据库：PolarDB O引擎、PolarDB PostgreSQL引擎、RDS MySQL、AnalyticDB PostgreSQL版、Greenplum、PolarDB-X、RDS PostgreSQL。

操作步骤

本文以PolarDB O引擎数据库为例，创建数据库集群方法请参见创建集群，创建数据库账号请参见创建数据库账号，集群白名单设置方法请参见设置集群白名单。

1. 登录ADAM控制台。
2. 在左侧导航栏单击**数据库管理**。
3. 在**数据库管理**页面，单击**创建数据库**。
4. 在**新建数据库档案**面板中，配置如下信息：

配置项	说明
档案名	请您输入数据库档案名称。
数据库类型	选择PolarDB O引擎数据库。更多数据库类型详情请参见数据库类型。
POLARDB-O实例区域	目标数据库实例所在地域。
POLARDB-O数据库联通的VPC	目标数据库的VPC网络。

配置项	说明
POLARDB-O数据库实例	目标数据库实例ID。
POLARDB-O主机IP	目标数据库IP地址。
数据库名	目标数据库名称。
编码方式（非必填项）	目标数据库的编码方式。
端口	目标数据库外网端口。
用户名	登录目标数据库的用户名。
口令	登录目标数据库的密码。

5. 单击**测试链接**。
6. 测试链接成功后，单击**创建**。

4.线上改造迁移

4.1. 线上改造迁移综述

您可使用ADAM进行线上数据库结构迁移改造。

支持的数据库类型

ADAM线上改造迁移支持将本地Oracle数据库迁移至云数据库，支持的云数据库类型及版本如下：

> **说明**
>
> ADAM支持Oracle源库10g、11g、12C及以上版本。

- 云原生关系型数据库PolarDB O引擎
- 云原生关系型数据库PolarDB PostgreSQL引擎
- 云原生分布式数据库PolarDB-X 1.0 版（只支持Schema维度迁移）
- 云数据库RDS MySQL版
  - 5.6 版本
  - 5.7 版本
  - 8.0 版本
- 云数据库RDS PostgreSQL版
  - 10.0 版本
  - 11.0 版本
  - 12.0 版本
- 云原生数据仓库AnalyticDB PostgreSQL版
  - 4.3 版本

前提条件

进行线上改造迁移前请先确认账号权限：

- 如果您当前使用的是主账号请忽略此步骤。
- 如果是主账号下属的RAM子账号：
  i. 用主账号登录进入数据库改造迁移页面。
  ii. 授权子账号，子账号将自动继承授权信息。

线上改造迁移步骤概述

1. 生成迁云计划：通过数据库评估的结果生成计划在目标库上进行的结构迁移数据。
2. 预检查：检查目标库账号权限、插件、版本号，保证迁移过程流畅进行。

3. 源库计划校验：对比迁云计划和Oracle源库当前现状，判断出哪些对象是新增的、变化的、删除的，帮助用户更新最新的迁云计划，用于迁移到目标库。此节点为可选环节，用户如需使用此功能，需将源Oracle库与ADAM网络打通，确保ADAM服务器能连接源Oracle库进行对比校验。如跳过此环节，可能导致迁云计划非最新数据，解决方案为重新做一次数据库采集评估。
4. 结构迁移/订正：负责将计划迁移的对象尽可能的迁移到目标数据库，ADAM会将不兼容的对象自动化评估校验，对于不能自动改造的提供解决方案，用户根据错误信息提供进行订正重试。
5. 增量源库对比（可跳过）：针对客户数据库结构比原先采集数据结构有些变动或者变动很多，可以通过增量源库对比发现客户改动、新增的DDL，方便客户迁移这些变动、新增的DDL。

4.2. 新建迁移项目

根据评估分析的结果生成从源库到目标数据库的迁移计划，使用迁移工具将源库的Schema快速迁移到目标数据库，其中迁移计划包含的ADAM智能转换结果将保证最大兼容度。本文介绍新建迁移项目的方法。

前提条件

- 已完成数据库评估分析。详情请参见数据库评估分析。
- 已创建数据库评估项目，详情请参见创建方法。
- 已创建目标数据库档案信息，详情请参见新建数据库档案。
- 如果目标库为云原生分布式数据库PolarDB-X 1.0，必须先在PolarDB-X 1.0实例中创建与Schema同名的数据库并绑定RDS。

操作步骤

1. 登录ADAM控制台。
2. 在左侧导航栏单击**数据库改造迁移**。
3. 在**数据库改造迁移**页面，单击**新建迁移项目**。
4. 在**新建迁云项目**面板中，配置以下信息。

配置项	说明
项目名	请输入您的项目名称。
数据库评估	选择目标数据库评估项目。
方案详情	选择目标数据库评估项目后，勾选相应方案详情。
目标库档案	选择目标库档案信息。
改造类型	目前仅支持**线上改造**，支持本地数据库迁移至PolarDB O引擎等数据库中。

5. 单击**测试链接**。
6. 测试显示链接成功后，单击**创建**。

> **说明** 项目创建成功后，如果数据库改造迁移项目列表还未出现项目，您可以单击页面右上角的 进行刷新。

7. 数据库改造迁移项目列表中项目的状态变为 `ACTIVE` 后，单击**详情**。

后续步骤

新建迁移项目完成，即生成迁移计划后，ADAM将自动开始预检查，详情请参见预检查。

4.3. 配置白名单

为了确保ADAM能连接至目标数据库，需要在目标数据库的白名单中添加ADAM的IP地址。

参考下表，添加ADAM的IP地址至目标数据库白名单。关于ADAM的IP地址，请参见ADAM IP地址。

目标数据库	配置白名单
云原生关系型数据库PolarDB O引擎	设置集群白名单
云原生关系型数据库PolarDB PostgreSQL引擎	设置集群白名单
云原生分布式数据库PolarDB-X 1.0版	设置白名单
云数据库RDS MySQL版	设置IP白名单
云数据库RDS PostgreSQL版	设置白名单
云原生数仓AnalyticDB PostgreSQL版	设置白名单

ADAM IP地址

- 如果ADAM与目标数据库在同一地域，需要配置VPC网络IP地址。

地域	IP地址
张家口	100.104.127.0/26
杭州	100.104.20.0/26
上海	100.104.107.64/26
北京	100.104.141.128/26
深圳	100.104.136.192/26
中国（香港）	100.104.87.192/26
新加坡	100.104.246.192/26
马来西亚（吉隆坡）	100.104.3.128/26
印度尼西亚（雅加达）	100.104.88.64/26
日本	100.104.111.128/26

地域	IP地址
美西	100.104.36.0/26

- 如果ADAM与目标数据库不在同一地域，需要配置公网IP地址。

```
39.100.131.119,47.112.134.104,47.93.233.187,149.129.255.124,47.75.108.211,47.89.251.38,47.254.192.180,47.245.13.201,47.241.17.217,47.99.157.96,101.132.180.65
```

4.4. 预检查

生成迁移计划后，ADAM会自动在后台进行预检查，检查目标库账号是否有相关权限、插件等。如果预检查结果有问题，请根据系统提示进行修复。

背景信息

生成迁移计划后，ADAM自动在后台进行预检查。

> **说明** 目前只支持云原生数据库PolarDB O引擎的预检查。迁移至其他目标库将自动跳过此步骤。

操作步骤

1. 数据库改造迁移项目列表中项目的状态变为绿色**ACTIVE**后，单击**详情**。

2. 单击**预检查**查看预检查结果。

   如果状态栏显示黄色的 `ACTIVE` ，您可以单击**详情**，根据预检查结果进行修复。

   - 目标库账号权限检查：如果目标库账号权限不足，请参见目标库账号权限错误。
     - 检查目标库账号是否有创建用户权限。
     - 检查目标库账号是否有授权用户权限。
     - 检查目标库账号是否有创建、删除Schema权限。
     - 检查目标库账号是否有创建、删除DDL权限。
   - 目标库插件检查：

     检查目标库是否已经安装迁移对象依赖的数据库插件。如果目标库缺失插件，请参见目标库插件缺失。
   - 目标库版本号检查：

     检查评估目标库版本与结构迁移目标库版本是否一致。如果目标库版本不一致，请参见目标库版本不一致。
   - 字符集匹配检查

     检查字符集是否匹配
   - 数据库时钟检查

检查数据库时钟是同步

- 目标库资源检查

  检查目标库资源是否充足

> **说明** 如果修复完成后状态栏的 `ACTIVE` 仍是黄色，可以先忽略预校验问题。

后续步骤

单击下一步源库计划校验，请参见源库计划校验。

4.5. 源库计划校验

源库计划校验为可选步骤，如果选择跳过校验进行结构迁移，ADAM将使用之前数据库采集评估分析的结果进行结构迁移，如果数据未发生变化或变化不大，可忽略。如数据变更较大，您可以重新做数据库采集评估迁移。

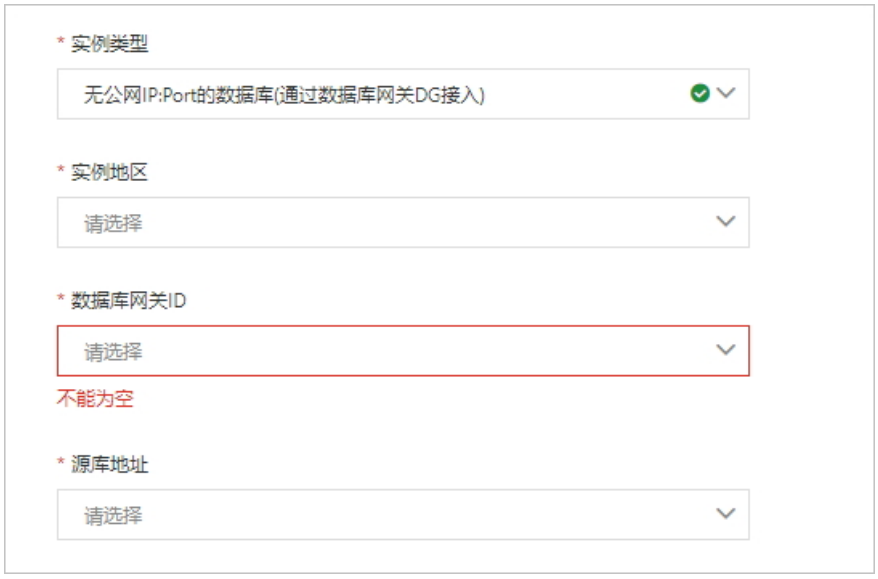
背景信息

- 源库计划校验可帮助您识别出：从上一次数据库采集截止到当前，源库数据及结构是否发生较大变更，更好地协助您完成数据库迁移。
- 执行本步骤需要您将源库的网络打通，并在数据库白名单中添加ADAM服务器 `39.100.131.0/24,47.241.17.0/24,149.129.255.0/24,47.254.192.180,47.89.251.0/24,47.245.13.0/24,47.75.108.0/24` ，确保ADAM服务器能连接进行校验。
  - 源库具有公网IP：通过公网访问。
  - 源库无公网IP，通过数据库网关DG连接：数据库网关DG（Database Gateway）是一种支持私网数据库远程访问的数据库连接服务。通过数据库网关DG，您可以无需开通公网地址访问并管理本地IDC或其他云厂商的数据库。详情请参见数据库网关DG 。

操作步骤

- 跳过校验，直接进行结构迁移
  i. 单击**跳过校验进行结构迁移**。
  ii. 在弹出的确认对话框中单击**确定**。
- 进行源库校验。
  i. 填写源库网络连接信息。
     - 有公网IP数据库
       - 实例地区：源数据库网络所属地域。
       - 主机IP：连接源数据库的IP。

- 无公网IP数据库，通过数据库网关DG接入



  - 实例地区：选择源库对应的数据库网关DG实例所属的地域。
  - 数据库网关ID：选择源库对应的数据库网关DG实例ID。
  - 源库地址：选择源库对应的数据库网关DG实例地址。

ii. 填写源库信息

- 服务器/SID：源数据库的服务名或SID。
- 编码方式：源数据库的编码方式。（非必填项）

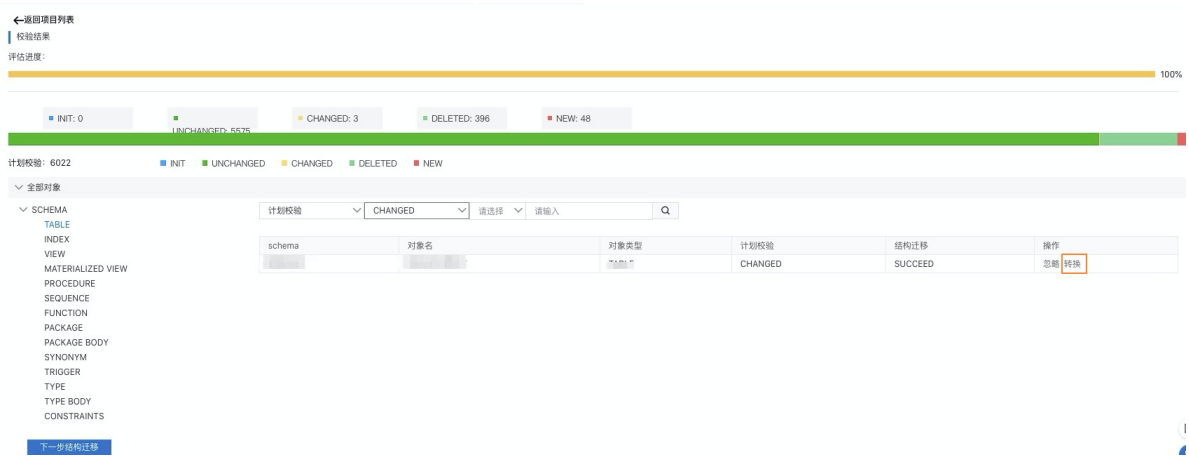
- 端口：源数据库连接端口。
- 用户名：能够进行数据库采集权限的账号。
- 口令：上述用户名对应的密码。

iii. 填写完毕后，单击**链接测试**，测试通过后单击**开始校验**。

校验结果页面

可手动对校验结果进行忽略或转换。

- 忽略：忽略DDL结构迁移。
- 转换：*NEW*、*CHANGED*可以通过智能转换。



>

后续步骤

单击**下一步结构迁移**，请参见结构迁移。

4.6. 结构迁移

ADAM可以通过配置规则控制用户的结构迁移权限，对数据库结构进行迁移。本文介绍配置迁移规则和开启数据库结构迁移的方法。

前提条件

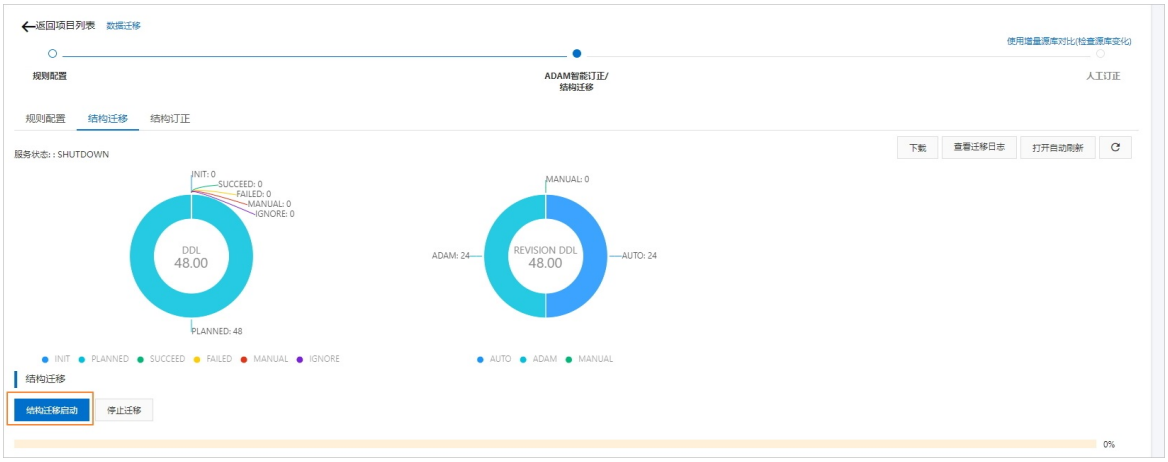
已生成迁移计划，并完成了预检查。

1. 分账户结构迁移配置。在**规则配置**页签下，单击**分账户结构迁移配置**，根据业务情况进行配置。
   - 统一用户：所有的Schema属于同一个用户。
   - 区分用户：每个Schema对应独立用户。
     a. 填写用户对应的密码。
        - 如果您之前已经创建了用户，请填写用户名对应的密码。
        - 如果您未创建用户，ADAM将自动在目标库创建用户，用户密码为您填写的密码，不填写密码将默认与之前登录目标库账号的密码一致。
     b. 根据需要设置或删除用户对应的Schema。
     c. 单击**保存**。
     d. 在弹出的确认对话框中单击**确定**。

> **说明** 迁移至目标库设置的区分用户与源数据库中的用户和Schema相同。

2. 在**结构迁移/订正**配置向导页面，**结构迁移**页签下，单击**结构迁移启动**。在弹出的确认对话框中单击**确定**。

> **注意**
> - 启动结构迁移将会先删除目标库中所有结构对象，请提前确认目标库中没有重要的结构对象。
> - 目标库为云原生分布式数据库PolarDB-X 1.0时，迁移外键约束很慢，请耐心等待。



- 单击**全部结构重新迁移启动**，重新迁移DDL, 包含已经迁移或者未迁移的都会重新被迁移。
- 单击**失败迁移启动**只迁移当前失败的DDL。
- 单击**停止迁移**停止当前迁移的DDL。
- 单击**定制化结构迁移**，详细请参见定制化结构迁移。

后续步骤

单击**结构订正**页签，进行结构订正。详情请参见结构订正。

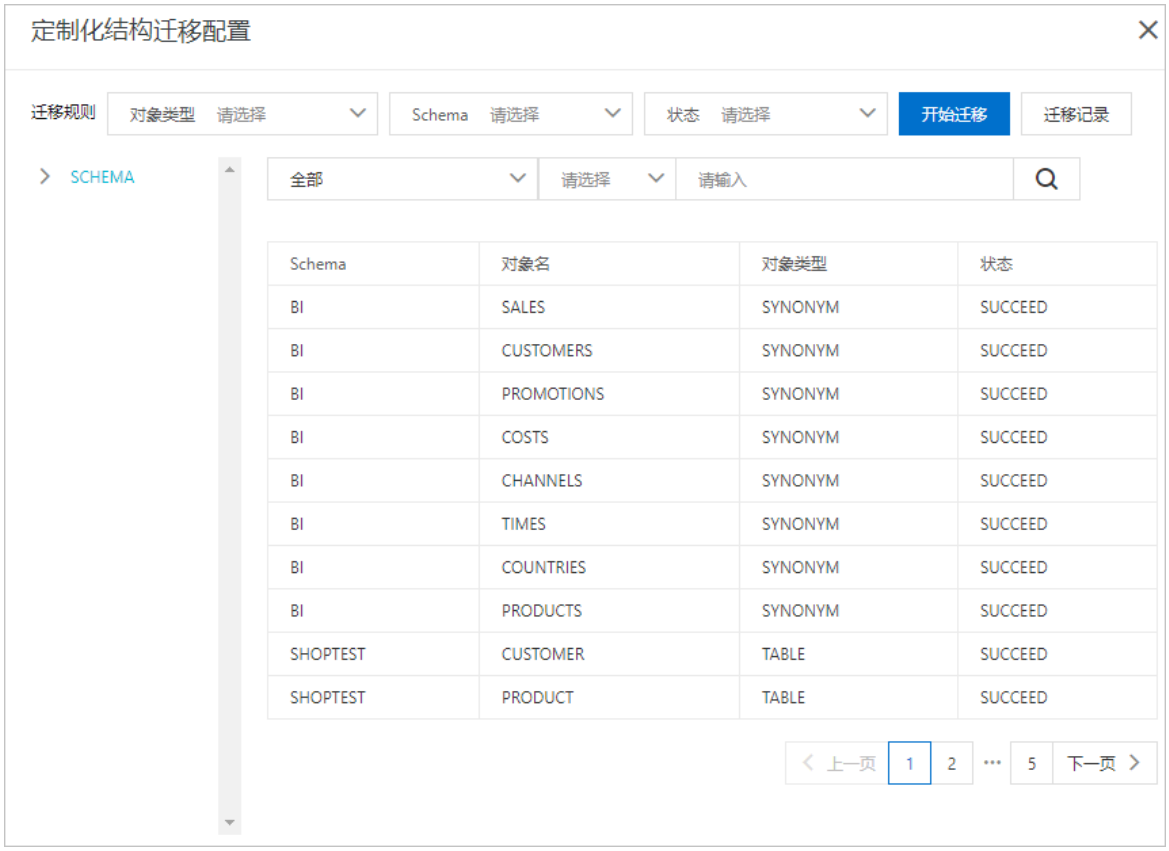
4.7. 定制化结构迁移

背景信息

为了满足用户不同的结构迁移需求，定制化结构迁移可以通过配置**对象类型**、**Schema**、**迁移状态**三个维度灵活定制化结构迁移。

操作步骤

1. 在**结构迁移/订正**配置向导页面，**结构迁移**页签下，单击**定制化结构迁移**。单击**新建定制化结构迁移**，进入**定制化结构迁移配置**面板。



2. 配置**迁移规则**。选择需要进行结构迁移的**对象类型**、**Schema**、**状态**。
3. 单击**开始迁移**。
   - 单击操作列**查看详情**，查看定制化迁移的DDL。
   - 单击操作列**启动**，启动定制化迁移。
   - 单击操作列**停止**，停止定制化迁移，运行状态下才能停止。
   - 单击操作列**删除**，删除定制化迁移。

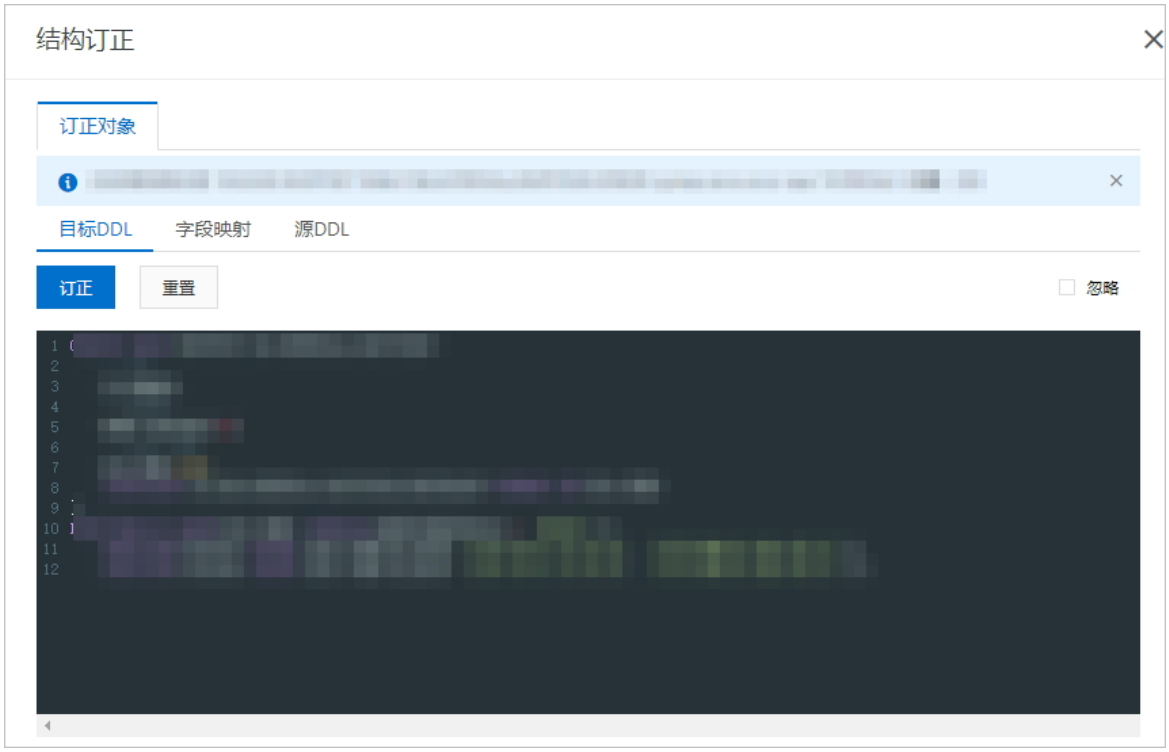
> **说明** 定制化结构迁移列表中查看列表进度需要手动刷新。

4.8. 结构订正

手工对ADAM转换后的DDL进行再修改并执行。

操作步骤

1. 在**结构迁移/订正**配置向导页面，单击**结构订正**页签。
2. 结构迁移状态分别选择**FAILED**和**MANUAL**，单击对象操作列中的**订正**，进行订正。
3. 在**目标DDL**页签中订正、重置DDL。



- **订正**：订正当前的对象。
- **重置**：恢复最开始的对象。
- **忽略**：不迁移对象。

4. 在**详细信息**中查看改造点、依赖对象、被依赖对象和对象特征。
   - **改造点**：查看当前对象智能转换改造点，其中不支持的内容需要您根据建议手动进行修改。
   - **依赖对象**：查看当前对象依赖的对象。
   - **被依赖对象**：查看当前对象被依赖对象。
   - **对象特征**：查看当前对象特征。

后续步骤

如果当前的数据库结构比采集时变动较多，可单击**增量源库对比**页签进行增量源库对比，请参见增量源库对比。

4.9. 增量源库对比

本文介绍如何通过增量源库对比发现改动、新增的DDL。

背景信息

如果您现在的数据库结构，相比采集时数据结构有些变动或者变动很多，可以通过增量源库对比发现改动、新增的DDL，方便您迁移这些变动、新增的DDL。

操作步骤

1. 单击**增量源库对比**页签。
2. 单击**启动**，启动增量对比任务。
3. 待进度条显示100%后，单击**增量校验历史纪录**，查看过去增量校验记录。

说明 如果您在增量源库对比后发现存在新增、删除和变更的对象，您可以重新返回上一个页签，单击**启动**，对新增的数据库结构变更进行同步，具体详情，请参见结构迁移。

5.数据库割接

5.1. 新建割接项目

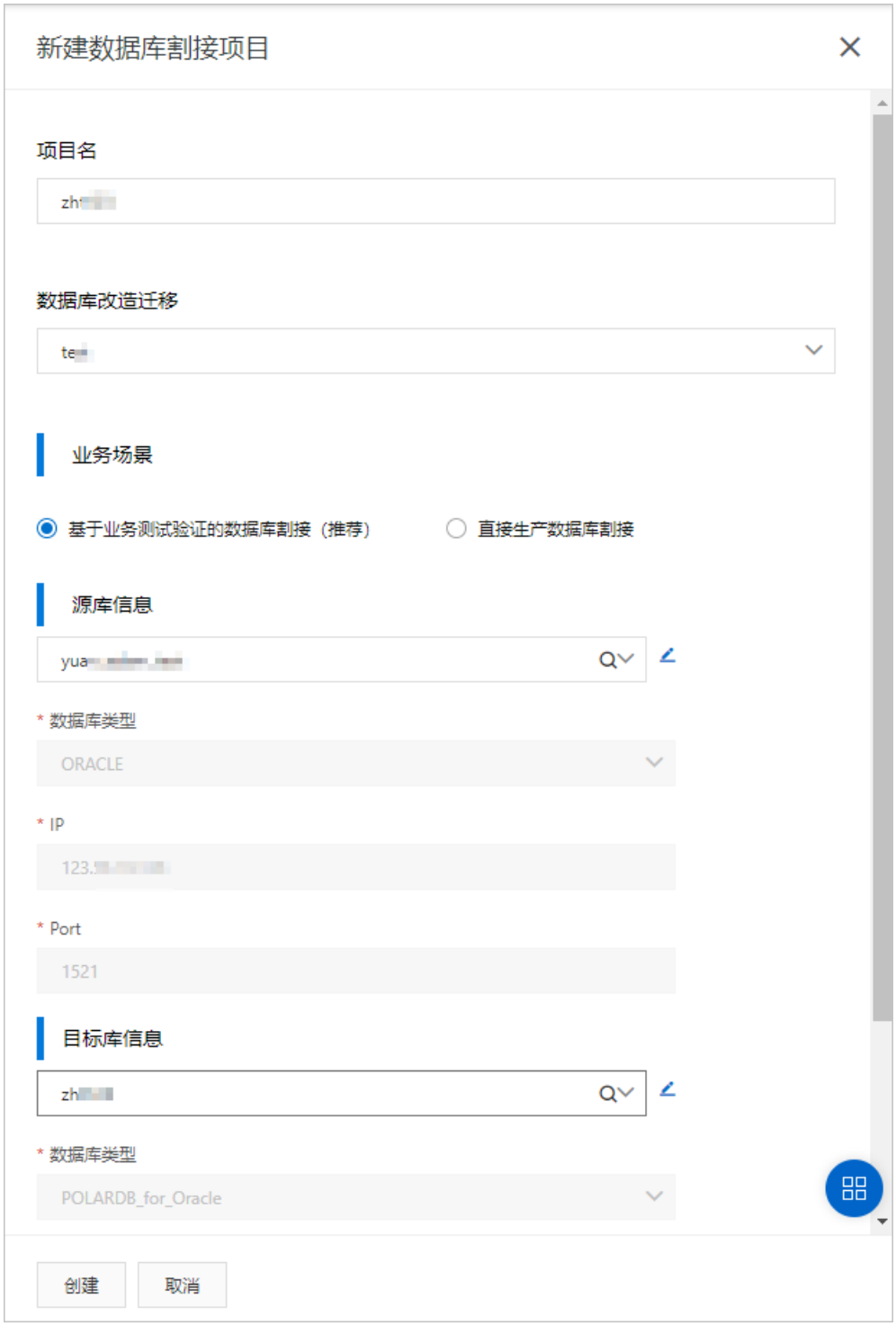
ADAM数据库割接用于帮助用户在完成数据库评估、数据库改造迁移、应用评估改造后，将业务最终平滑地迁移到云上数据库。本文介绍在ADAM中如何新建数据库割接项目。

前提条件

- 已创建源数据库和目标数据库信息，创建方法请参见新建数据库档案。
- 已创建数据库迁移项目，创建方法请参见新建迁移项目。

操作步骤

1. 登录ADAM控制台。
2. 在左侧导航栏单击**数据库割接**。
3. 在数据库割接页面，单击**新建数据库割接项目**。
4. 在**新建数据库割接项目**面板中，配置如下信息：



配置项	说明
项目名	请您输入数据库割接项目的名称。
数据库改造迁移	选择目标数据库迁移项目。

配置项	说明
业务场景	○ **基于业务测试验证的数据库割接**：该场景为了保证生产库最终割接上线是符合预期的，需要在割接前做一些测试验证的工作，例如业务测试、割接验证、回退验证，最后再进行正式生产割接。 ○ **直接生产数据库割接**：该场景适用于有大量源库迁移上云需求的客户，并且已经在开始的几个上云迁移项目里完成了业务以及数据库割接回滚的验证，确认直接生产数据库割接是稳定高效且符合预期的。
源库信息	选择目标源库信息。
目标库信息	选择目标库信息。

5. 单击**创建**。

后续步骤

数据库割接项目创建成功后，单击目标项目**操作**列下的**详情**，进入**业务验证**，详情请参见业务验证。

5.2. 业务验证

ADAM数据库割接用于帮助用户在完成数据库评估、数据库改造迁移、应用评估改造后，将业务最终平滑地迁移到云上数据库。割接项目新建成功后，ADAM的割接进入业务验证阶段，业务验证阶段可以测试目标库的性能是否可以满足预期。

前提条件

- ○ 已新建数据库割接项目，详情请参见新建割接项目。
  - ○ 如果您在业务验证阶段需要进行SQL对比，您需要：
    - ■ 已创建数据库评估项目，详情请参见创建方法。
    - ■ 已创建周期性采集项目，详情请参见SQL周期性采集。
    - ■ 已创建Adapter实例，详情请参见SQL Adapter。

操作步骤

1. 登录ADAM控制台。
2. 在左侧导航栏单击**数据库割接 > 查看数据库割接项目**。
3. 在**数据库割接项目列表**页面，单击目标项目**操作**列下的**详情**。
4. 在**业务验证**页面，您需要进行以下四个步骤的验证：
   i. 目标库参数调整
   - ■ 如果您无需修改任何自定义参数,您可以单击**下一步**，进入**建立正向增量同步（含全量）**配置向导页面。
   - ■ 如果您需要修改某些自定义参数，您可以进行以下操作：
     a. 单击**自定义参数**页签，根据**参数值范围**，按需调整**运行参数据**。
     b. 在页面左上方单击**保存**。
     c. 单击**下一步**，进入**建立正向增量同步（含全量）**配置向导页面。

ii. 建立正向增量同步（含全量）。具体操作，请参见建立正向增量同步。

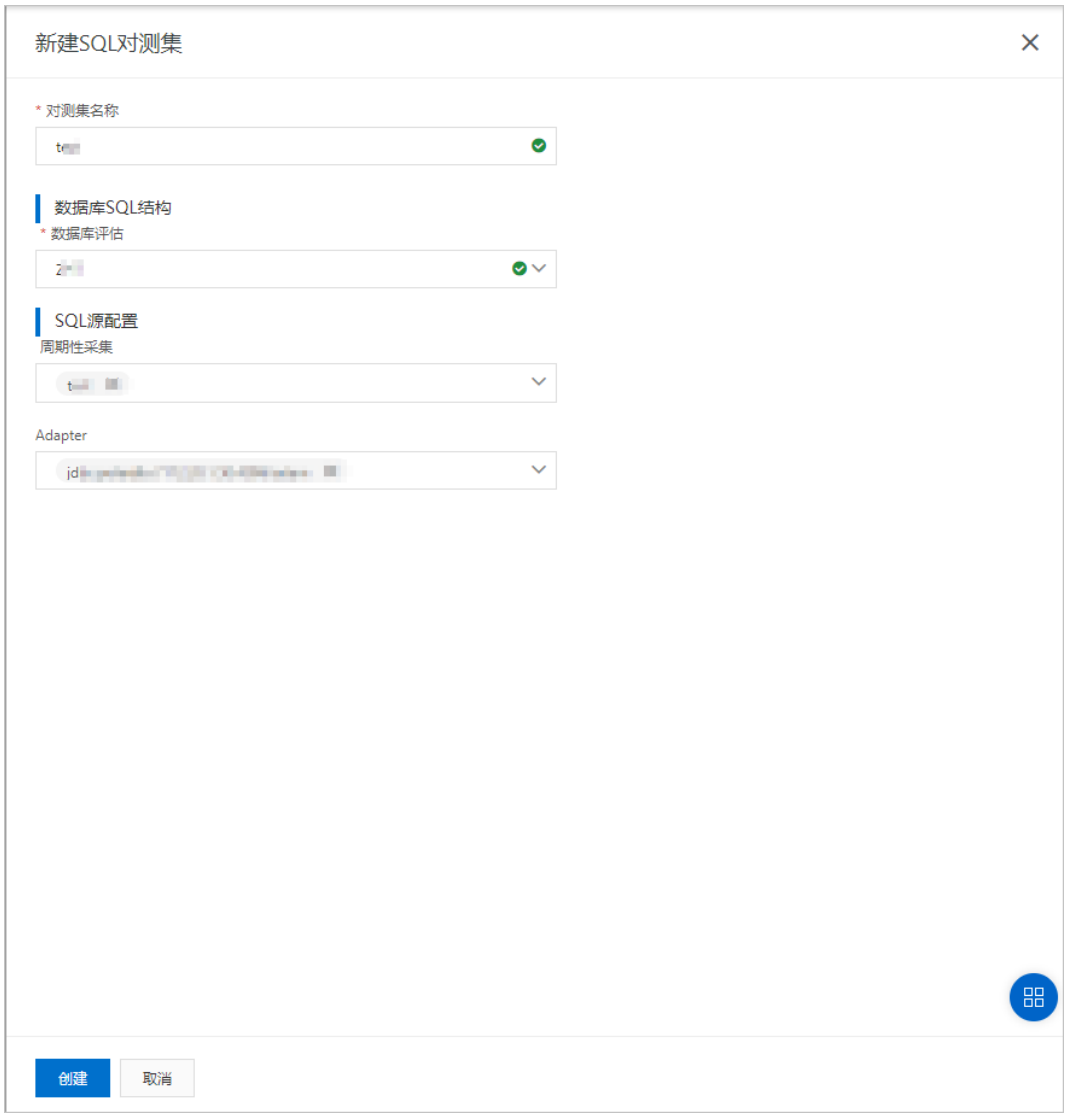
完成建立正向增量同步后，单击**下一步**进入**SQL对比**配置向导页面。

iii. SQL对比

新建SQL对侧集可以验证等价SQL在源和目标库上执行的差异，包括执行时间、返回结果行数、结果集校验等。SQL对比的步骤如下：

> **说明** 此步骤可以直接跳过。您可以单击**SQL对比**页签下的**跳过**。

a. 单击**新建SQL对侧集**，配置参数如下：



配置项	说明
对测集名称	请您输入对侧集的名称。
数据库评估	选择目标数据库评估项目。
周期性采集	选择目标周期性采集项目。
Adapter	选择目标Adapter实例。

b. 在**新建SQL对测集**面板中，单击**创建**。

c. 单击已创建好的目标对侧集**操作**列下的**详情**。

d. 您可以在**SQL对测集详情列表**页面进行SQL订正等操作。

e. 单击**下一步**进入**目标库性能展示**配置向导页面。

iv. 目标库性能展示

在**目标库性能展示**页面，您可以查看到目标库的存储空间、CPU性能的数据信息。

后续步骤

业务验证阶段完成后，在**目标库性能展示**配置向导页面**下一步**，ADAM将进入割接验证阶段，详情请参见割接验证。

5.3. 建立正向增量同步

本文介绍建立正向增量同步的方法。

前提条件

您已完成业务验证的目标库参数调整步骤，并已进入建立正向增量同步配置向导。具体操作，请参见业务验证。

操作步骤

1. 在建立正向增量同步配置向导中，下线数据结构。

   数据迁移前需要先删除触发器、约束、索引等影响数据迁移的结构，在数据迁移完成后，再将这些结构添加回来。下线数据结构的功能是删除触发器、约束、索引等影响数据迁移的结构。

   i. 在**下线结构**页签中，单击**下线启动**。

   ii. 在弹出的**确认**对话框中单击**确定**。

   iii. （可选）您还可以单击**下线结构详情**页签，可以根据条件查询下线结构详情。

2. 数据迁移。

   i. 在**数据迁移**区域，单击**新建迁移项目**。

   ii. 在**新增迁移项目**面板中，配置购买DTS实例的参数，查看源库和目标库信息，并在源库中根据迁移地域添加DTS白名单，选择迁移表范围等。

   iii. 单击**测试链接**。

   iv. 测试链接通过后，单击**创建**。

   迁移项目创建成功后，参数正确的情况下任务会自启动并开始迁移数据，由于单链路迁移大量数据会导致出错的概率大幅度提升，ADAM将为您进行配置迁移对象智能分批处理。

   v. （可选）单击列表右侧的**查看详情**跳转到DTS的管理页面查看任务的具体情况。

   如果数据量超过任务配置的规格，可以点击列表右侧的**升级**按钮重新选择DTS实例的规格大小。

3. 上线数据结构。

   上线数据结构的功能是在数据迁移完成后，将之前删除的触发器、约束、索引等数据结构添加回来。

   i. 单击**上线结构**页签下的**上线启动**。

   ii. 在弹出的**确认**对话框中单击**确定**。

   iii. （可选）单击**上线结构详情**页签，可以根据条件查询上线结构详情。

5.4. 割接验证

ADAM的数据库割接验证阶段帮助您验证修改停机后数据库的应用配置，例如确认同步延迟、停止同步链路等等。同时也支持逆向回退。本文对数据库割接验证阶段的相关步骤进行说明。

前提条件

已完成业务验证阶段，详情请参见业务验证。

割接验证说明

1. 增量同步任务检查

   增量同步任务检查是通过新建迁移项目，完成增量同步任务的迁移，保证了源数据库及目标数据库间任务的同步迁移变更。

2. 停止应用改配置

   停止应用改配置需要您停止应用，检查源数据库是否还有连接，保证数据库连接已经完全停止，并且修改应用内的配置。

   > **说明** 您可以获取当前数据库的链接，当数据的会话值逐渐降低为0时，表示应用已经完全停止。

3. 停止源库任务和触发器

   当您停止应用并修改完配置后，您需要停止源数据库的任务和触发器，以停止源数据库内部数据的流动，防止数据发生变化。当源数据库的数据整体停下后，最终将任务目标指向目标数据库。

4. 停止正向同步任务

   停止源数据库到目标数据库的同步任务，以确保同步任务完全停止。

5. 自增同步

   自增同步表示源数据库与目标数据库间的同步，如果源数据库的数据发生变化，目标数据库可以同步变更。

6. 启动反向同步链路

   启动反向同步链路是割接验证阶段的一个关键步骤，此步骤是为了下一步做回退时，将已写入目标数据库的业务数据同步进源数据库，DTS 配置的任务以原来的目标数据库作为源数据库，原来的源数据库作为目标数据库，建立了反向链接关系。以防止风险的发生，如果发现迁移的数据有误，可以及时实现可逆的操作。

7. 启动目标库任务和触发器

   启动目标库的任务和触发器，帮助下一步的应用启动做充足的准备。

8. 启动应用

   启动应用需要保证目标数据库的连接，此时才能完全启动应用。

后续步骤

您可以单击**启动应用**配置向导页面下的**下一步**，进入回退验证阶段，更多详情，请参见回退验证。

5.5. 回退验证

ADAM的数据库回退验证阶段属于割接验证的逆向操作，帮助您将数据内容从目标数据库回迁到源数据库。本文对数据库回退验证阶段的相关步骤进行说明。

前提条件

已完成割接验证阶段，详情请参见割接验证。

回退验证说明

1. 反向增量同步任务检查

反向增量同步任务检查实现了数据从目标数据库到源数据库的回退检查，检查是否符合预期目标。

2. 检查目标库是否有长事务或任务运行

   检查目标库长事务等是否还有运行中的情况，排除业务的影响。

3. 停应用并修改应用配置

   停止应用改配置需要您停止应用，检查目标数据库是否还有连接，保证数据库连接已经完全停止，并且修改应用内的配置。

   > **说明** 您可以获取当前数据库的链接，当数据的会话值逐渐降低为0时，表示应用已经完全停止。

4. 停止目标库任务和触发器

   当您停止应用和触发器并修改完配置后，您需要停止目标数据库的任务和触发器，以停止目标数据库内部数据的流动，防止数据发生变化。当目标数据库的数据整体停下后，最终将任务目标指向源数据库。

5. 停止反向同步数据链路

   停止目标数据库到源数据库的同步任务，以确保同步任务完全停止。

6. 自增同步

   自增同步表示源数据库与目标数据库间的同步，如果目标数据库的数据发生变化，源数据库可以同步变更。

7. 建立正向增量同步

   建立正向增量同步为了使回退割接流程可多次执行，需要配置正向同步链路，保证数据的同步。

8. 启动源库任务和触发器

   启动源库的任务和触发器，帮助下一步的应用启动做充足的准备。

9. 启动应用

   启动应用需要保证数据库的连接，此时才能完全启动应用。

后续步骤

您可以单击**启动应用**配置向导页面下的**下一步**，进入生产割接阶段，更多详情，请参见生产割接。

5.6. 生产割接

ADAM的数据库生产割接阶段使用户的目标正式指向最终的生产库，此阶段中支持连接信息的变更以及支持增减新的连接方式等等。本文对数据库生产割接阶段的相关步骤进行说明。

前提条件

已完成回退验证阶段，详情请参见回退验证。

生产割接说明

1. 配置源库目标库

   配置源库目标库将目标更换成您最终的生产库，支持连接信息的变更，可以增减新的连接方式。

2. 清除目标库数据

   在生产割接前将之前用于验证的数据清空，准备正式数据的灌入。您可以单击控制台的**目标库信息检查**批量删除数据或者直接清空所有目标库数据。

3. 增量同步任务检查

   增量同步任务检查是通过新建迁移项目，完成增量同步任务的迁移，保证了源数据库及目标数据库间任务的同步迁移变更。

4. 停止应用改配置

   停止应用改配置需要您停止应用，检查源数据库是否还有连接，保证数据库连接已经完全停止，并且修改应用内的配置。

   > **说明** 您可以获取当前数据库的链接，当数据的会话值逐渐降低为0时，表示应用已经完全停止。

5. 停止源库任务和触发器

   当您停止应用并修改完配置后，您需要停止源数据库的任务和触发器，以停止源数据库内部数据的流动，防止数据发生变化。当源数据库的数据整体停下后，最终将任务目标指向目标数据库。

6. 停止正向同步任务

   停止正向同步任务指您需要停止源数据库到目标数据库的同步任务，以确保同步任务完全停止。

7. 自增同步

   自增同步表示源数据库与目标数据库间的同步，如果源数据库的数据发生变化，目标数据库可以同步变更。

8. 启动反向同步链路

   启动反向同步链路是生产割接阶段的一个关键步骤，将已写入目标数据库的业务数据同步进源数据库，DTS 配置的任务以原来的目标数据库作为源数据库，原来的源数据库作为目标数据库，建立了反向链接关系。以防止风险的发生，如果发现迁移的数据有误，可以及时实现可逆的操作。

9. 启动目标库任务和触发器

   启动目标库的任务和触发器，帮助下一步的应用启动做充足的准备。

10. 启动应用

    启动应用需要保证目标数据库的连接，此时才能完全启动应用。

11. 停止反向同步数据链路

    停止反向同步数据链路指停止目标数据库到源数据库的同步链路。

    > **说明** 确保应用使用正常后，您可以停止反向同步数据链路，断开库间的数据同步。您也可以跳过该操作。

6.应用评估改造

6.1. 应用评估改造摘要

用户完成数据库评估改造后，需要改造自己的应用。应用程序比数据库复杂，涉及的代码可能经历过很多开发者，改造应用已经成为用户迁移数据库上云的痛点：为了解决用户迁移数据库后的应用改造问题，ADAM提供了应用评估改造功能，保证用户快速改造自己的应用。

1. 核心功能

- 提供迁移数据库的应用改造点，包括改造具体位置（调用栈），改造SQL的内容；
- 梳理应用的框架，性能等使用信息；
- 大规模集群迁移的架构梳理，提供架构迁移指南；

2. 使用流程

1. 应用采集
2. 应用画像
3. 应用评估
4. 应用静态改造

> **说明**
>
> 应用动态评估请使用1.2.3，应用静态改造请使用4。

6.2. 应用采集

ADAM可以对Java JDK1.6及以上版本的应用提供采集功能，帮助客户评估分析需要改造的功能点，对于非Java应用暂不支持采集评估。

应用采集概述

- 应用采集客户端包含两个模块：
  - 应用动态采集Agent。收集运行期应用请求数据库的基本信息，比如请求的sql-schema-调用栈、应用系统信息、性能信息、SQL热度等。
  - 数据集中收集Collector。集中收集各应用Agent传输过来的数据，并进行脱敏、加工。
- 应用采集可以完成：
  - 采集应用访问的SQL与调用栈信息。
  - 收集应用运行性能信息。
- 应用采集不能完成：
  - 非Oracle数据库或非Java应用暂不能采集。
  - 无数据库请求的监控不到，比如采集周期内未请求接口则这个接口的请求SQL语句及调用栈采集不到。
  - 触发器等未通过程序直接调用的监控不到。

> **说明**
>
> 如果采集中遇到任何问题请提交工单或直接发邮件到adam_service@alibaba-inc.com，工单中请附上报错信息和联系方式。

说明

- 采集的SQL会做脱敏处理，不采集请求参数及SQL中的具体值。
- 只读保护：不侵入应用。
- 负载控制：业务高峰期自动暂停采集、内存使用量控制在设定范围内。
- 支持 JDK1.6+ Tomcat、Jboss、Weblogic容器Oracle的Java应用动态采集。

应用采集工具下载

部署前必读

- 部署时涉及基本技术知识，请确保由Java研发人员操作。
- 支持Sun JDK 、Oracle JDK、Open JDK1.6及以上版本，不支持IBM JDK。
- 解压后有两个目录：
  - collector：collector是统一收集器，单独部署在没有线上应用的服务器上（数据处理时避免对线上应用造成影响）。
  - javaagent：javaagent目录拷贝到需要监控的应用服务器上，和应用部署在一起，用于采集数据。
- 确保Collector和Agent具有操作权限。

  > 说明
  >
  > UNIX/Linux需要对目录增加级联操作权限 `c.hmod -R 775 collector/` 。

- Collector相当于Server端，可对应1-20个Agent，一个应用服务器部署一个Agent。

  > 说明
  >
  > 如果应用是分布式多机器，根据负载均衡的情况，只抽样几台部署Agent即可。

- 先部署Collector，后部署应用Agent，应用要与Collector网络可达（用于推送数据做集中脱敏等处理），部署Collector的机器需要JDK 1.6+版本，JVM内存4G以上。磁盘与监控的应用数量、监控时长、业务活跃度、SQL数量及SQL大小有关，不会出现爆发式增长，可观察半天来估算，一般一个应用监控7天数据量在1G以下。
- Agent要求：应用部署在JDK1.6及以上版本，待监控的应用有300M的可用JVM heap空间。服务容器支持Tomcat、Jboss、Weblogic、Websphere及k8s集群Docker容器镜像部署。

- Agent监控访问Oracle数据库的SQL和代码调用栈，请确保Agent监控周期内的操作覆盖全。如有周期性任务，需在有这些任务运行时监控，否则数据采集不全。

后续操作

采集部署

6.3. 采集部署

本文介绍了采集部署的流程。

1.预检查

- 确保collector部署在没有线上应用的独立服务器上。
- 确保已配置JAVA_HOME，JDK版本1.6+。

2.启动

```
Unix系统，在collector目录下执行：
`./run.sh`
Windows系统， 在collector目录下执行：
`start /b java -jar javaagent-collector.jar`
```

验证：查看collector/logs/collector.log日志，显示启动成功即部署成功。

如有报错参考应用采集器常见问题。

采集器javaagent

1.启动前的配置

- 确保环境变量已配置JAVA_HOME，否则设置 attach.sh 中 JAVA_HOME 地址为jdk的绝对路径（注意：如客户使用jre而非jdk，需要自己将tools.jar拷贝到${JAVA_HOME}/lib/目录下）。
- 配置javaagent.config：`profiler.collector.ip = 11.23.45.67` # collector的ip `profiler.collector.port = 9996` # collector的端口 `profiler.app.name = adamApp` #应用名，少于20个字符的字母、数字组合 。 `profiler.app.port = 8080` #应用启动端口，应用有很多个不同功能的端口，只配请求的那个端口即可，一个JVM下无论是否一个应用都只配置一个 `profiler.applicationservertype = TOMCAT` # 应用中间件容器类型，TOMCAT\JBOSS\WEBLOGIC等。

以下非必选配置

配置应用需要检测的Java代码的目录前缀。替换下面示例，多个目录用英文逗号分隔，每个至少2级目录，建议不超过5个；
应用采集时根据精准的调用栈信息，在数据库改动时能同步给出明确的应用改动建议。
如不能全部提供则可以不写，后续在阿里云Adam分析页面进行过滤设置。
`profiler.classpath.whitelist = com.alibaba.javaagent,com.alibaba.adam`
如上面白名单已填写，此处可忽略；如果不清楚白名单列表，填写黑名单也可过滤无用的调用栈信息。
`profiler.classpath.blacklist =org.apache,net.sf`
配置当cpu达到多少时暂停数据采集。
`profiler.cpu.threshold = 85`
系统信息收集间隔，默认15分钟。
`profiler.sys.send.interval = 15`
sql动态信息收集间隔，默认15分钟。
`profiler.sql.dynamic.send.interval = 15`

2.启动方式

启动方式一：不重启应用，agent单次临时监控

javaagent目录增加操作权限；确保agent启动的账号和应用启动账号一致；找到应用的进程号PID，在javaagent目录执行（将 `${pid}` 整体替换为应用的进程号）：

```
Unix系统，执行：
`./attach.sh -p ${pid}`
Windows系统，执行：
`java -cp "%JAVA_HOME%\lib\tools.jar;%cd%\javaagent-bootstrap.jar" com.alibaba.adam.javaagent.bootstrap.AgentAttacher -p ${pid}`
```

查看javaagent目录下log文件，提示启动成功，应用有流量进入后，查看collector的目录下有data目录且有数据，说明agent启动成功且正确发送数据到collector。如有报错参考常见问题应用采集器常见问题。

- 确保应用的PID获取正确。
- 启动agent的账号和应用启动账号保持一致，且权限也要一致，否则无法监控。（windows下注册表SYSTEM启动的应用无法通过启动方式一启动，原因是账号权限不一致）。

启动方式二：随应用一起启动监控（推荐）

应用增加javaagent配置重启即可（ `${javaagent_path}` 整体替换为javaagent的目录）。

Unix系统

Tomcat：在catalina.sh启动文件最后一个CATALINA_OPTS配置后面增加：
CATALINA_OPTS="$CATALINA_OPTS -javaagent:${javaagent_path}/javaagent-bootstrap.jar"
Jboss：在run.conf启动文件最后一个JAVA_OPTS配置后面增加：
JAVA_OPTS="$JAVA_OPTS -javaagent:${javaagent_path}/javaagent-bootstrap.jar"
Weblogic：在startWebLogic.sh启动文件最后一个JAVA_OPTIONS配置后面增加：
JAVA_OPTIONS="$JAVA_OPTIONS -javaagent:${javaagent_path}/javaagent-bootstrap.jar"
Websphere：
方式1 配置文件增加： JAVA_OPTS="$JAVA_OPTS -javaagent:/home/admin/javaagent/javaagent-bootstrap.jar"
方式2 页面配置： 通用 JVM 参数(Java Virtual Machine)加上：
name:javaagent value:/home/admin/javaagent/javaagent-bootstrap.jar
k8s集群docker容器：
将javaagent目录放入docker镜像中并对应增加-javaagent配置，一起打包镜像部署。

Windows系统

Tomcat：在catalina.bat启动文件最后一个CATALINA_OPTS配置后面增加：
CATALINA_OPTS="$CATALINA_OPTS -javaagent:${javaagent_path}/javaagent-bootstrap.jar"
Jboss：在run.conf启动文件最后一个JAVA_OPTS配置后面增加：
JAVA_OPTS="$JAVA_OPTS -javaagent:${javaagent_path}/javaagent-bootstrap.jar"
Weblogic：在startWebLogic.cmd启动文件最后一个JAVA_OPTIONS配置后面增加：
JAVA_OPTIONS="$JAVA_OPTIONS -javaagent:${javaagent_path}/javaagent-bootstrap.jar"
Websphere：
方式1 配置文件增加： JAVA_OPTS="$JAVA_OPTS -javaagent:/home/admin/javaagent/javaagent-bootstrap.jar"
方式2 页面配置： 通用 JVM 参数(Java Virtual Machine)加上：
name:javaagent value:/home/admin/javaagent/javaagent-bootstrap.jar

应用启动后查看应用的日志，显示 `Java Agent load successfully!` 表示启动成功，如报错根据错误原因进行修改，如未显示任何agent信息表示配置路径不对导致未加载。

优缺点

启动方式	优点	缺点	适用场景
启动方式一	对全新应用无须重启即可设置agent监控。	每次应用重启agent需手动启动； 启动agent的账号必须和应用保持一致。	应用长期不重启且重启会造成业务影响。
启动方式二	应用重启agent自动启动，应用采集连续；不用考虑账号权限问题。	第一次需重启应用。	应用可重启、持续采集（大多数场景）。

其他说明

- 当应用停止：无论方式一、二启动的监控，应用停止则agent停止。

- agent主动停止监控：
  - 方式一：重启应用或通过命令./attach.sh -p ${pid} -s。
  - 方式二：去掉配置的javaagent后重启应用。
- agent停止监控后想再次做监控，必须重启应用。
  - 方式一：待应用重启后手动启动agent。
  - 方式二：agent随应用重启而启动。
- Jboss区分社区版和企业版，如果有jboss.modules.system.pkgs配置项，则无论用方式一、二，都需要先增加com.alibaba.adam.javaagent目录并重启应用才能生效。

数据收集（一般收集1-7天数据）

- 进入collector目录，将data/下的文件按appname打zip包，分别独立上传ADAM-应用画像，切记一个应用对应一个应用画像，一个应用下不同IP的数据可以打包到一起，但不同appname的数据不能打包到一起，否则上传会报错。
- 下一步上传应用画像。

6.4. 应用画像

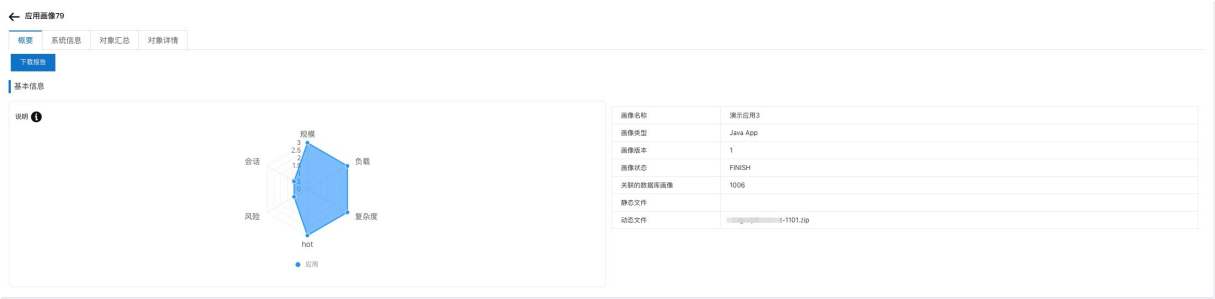
应用画像功能是ADAM根据单个应用采集的数据，通过智能分析算法，向用户直观提供应用系统客观分析内容。

前提条件

已创建应用画像，创建方法请参见应用画像。

应用画像详情

应用画像主要分为四个部分：概要、系统信息、对象汇总、对象详情。



下载报告：通过下载报告查看这个应用画像的报告。

基本信息包括2个部分：应用分析雷达图，应用画像基本信息。

应用分析雷达图是ADAM通过智能分析通过6个纬度对应用进行整体概括：

- 复杂度是根据应用的使用场景，相关数据库的特性等维度，计算出的ADAM画像复杂度。主要用于衡量应用的使用情况，分数越高说明应用越复杂，可能涉及的改造情况越多。
- 会话主要衡量应用的连接状况，分数越高说明应用连接数越多，对应用改造的连接池配置有参考意义。
- 风险主要衡量应用当前可能存在的性能瓶颈与稳定性风险，特别是应用SQL存在的性能风险。
- 热点主要衡量应用所访问的数据库是否存在访问频率非常集中的对象。分数越高，说明应用的数据库访问存在热点情况。

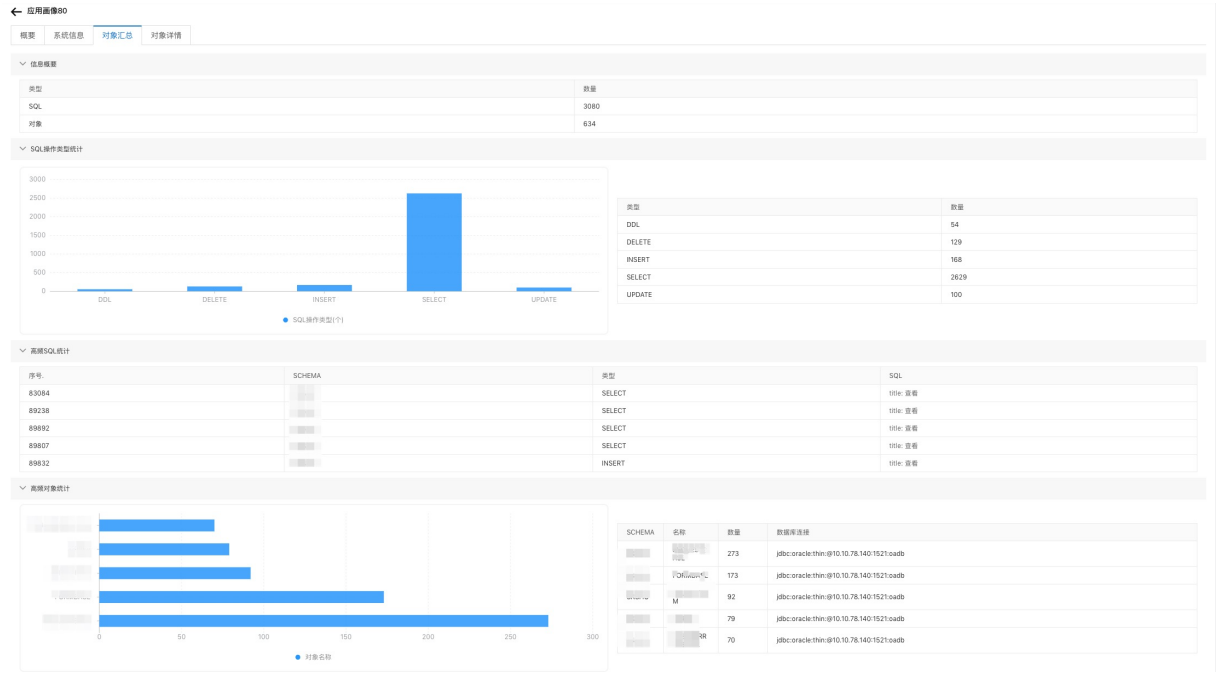
- 规模主要衡量当前应用的部署单元数或者实例数情况。
- 负载主要衡量应用的运行性能情况。

系统信息

系统信息是根据ADAM应用采集器采集的应用运行时系统参数，有助于用户评估应用运行状态。

对象汇总

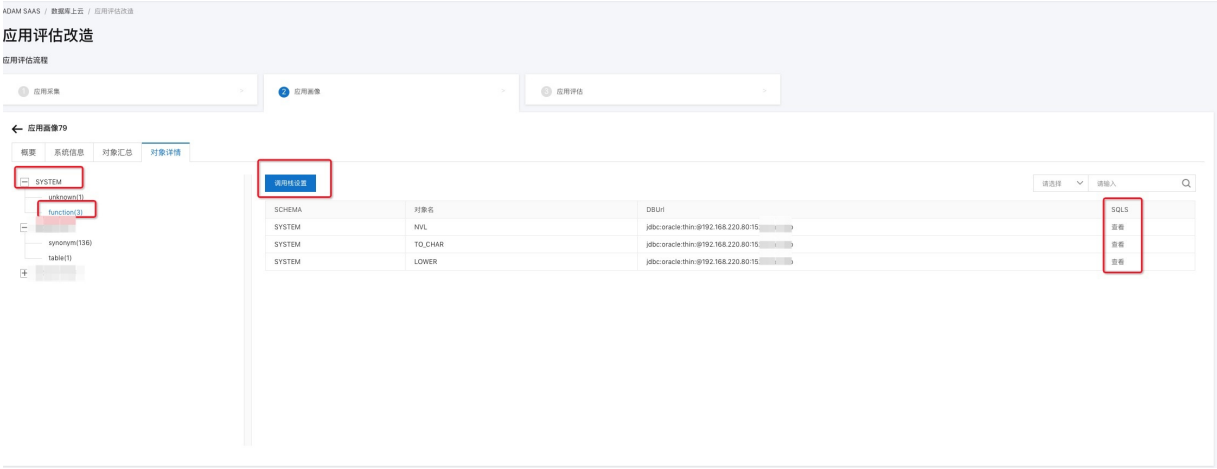
对象汇总主要罗列了应用采集与智能分析出的SQL与数据库对象情况汇总，可以直观通过数据看到应用画像的整体内容。



对象详情

对象详情是详细展示了ADAM智能分析后的数据库对象与应用SQL以及应用代码的关系。其中左边是以SCHEMA和对象类型为纬度，通过树的形式展示应用访问的数据库对象的罗列。右边是具体对象和访问该对象的SQL。

通过调用栈设置，可以配置调用栈黑名单，方便过滤想要的调用栈信息。



6.5. 应用评估

应用评估用于衡量应用与数据库整体迁移改造的情况，可以展示应用需要改造的地方，并给出改造建议。

前提条件

全部应用采集包都已创建应用画像。

背景信息

迁移数据库和应用的过程中存在以下几个难点问题：

- 难以估算应用改造的工作量。
- 难以制定详情地迁移数据库的计划。
- 应用长时间未维护，迁移后难以改造应用。

新建应用评估

1. 填写基本信息，选择需要评估的目标库以及目标库版本。
2. 选择需要评估的应用画像，支持多选。

> **说明**
>
> 只有全部的应用画像关联数据库评估状态是完成，才可以创建应用评估。

3. 选择已经完成的数据库评估项目。

整体评估结果

评估结果是应用评估的核心内容，分为整体、迁移分组、应用节点三个纬度。

首次进入评估结果，展示的是整体信息，即应用与数据库的聚合评估结果。用于衡量应用与数据库整体迁移改造情况。

- 架构列表：ADAM通过数据库与应用的关系，结合智能算法，将整体架构做了分组。迁移分组是一个迁移单位的最小子集，即迁移任何一个迁移分组不会影响到其他迁移分组外的数据依赖。

  > 说明
  >
  > 应用之间调用关系不在考虑范围。

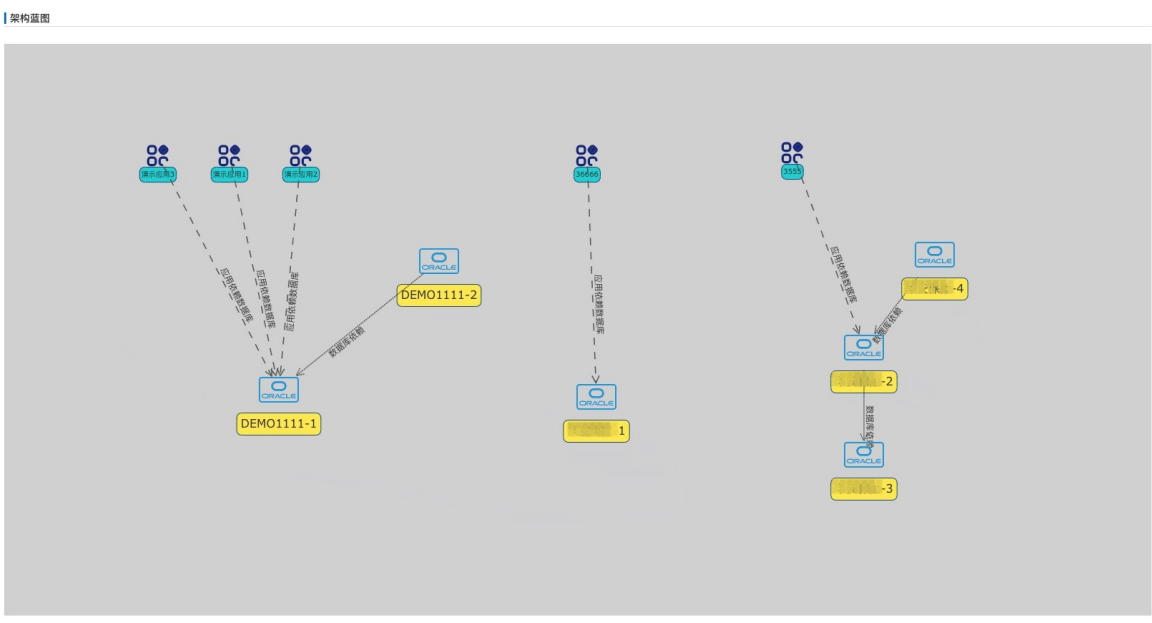
- 迁移评分：ADAM对迁移与改造难度进行的量化打分，分数越高说明迁移改部分的应用改造成本越低。

  > 说明
  >
  > 迁移评分受采集数据完备性影响，请结合业务实际情况综合考虑迁移成本。

- 整体兼容度： 衡量应用SQL与数据库对象的兼容性情况。

  > 说明
  >
  > 数据库采集SQL受数据库系统本身影响，兼容度不作为评估参考。

- 架构蓝图： 通过拓扑图的形式，直观展示各迁移分组的情况。



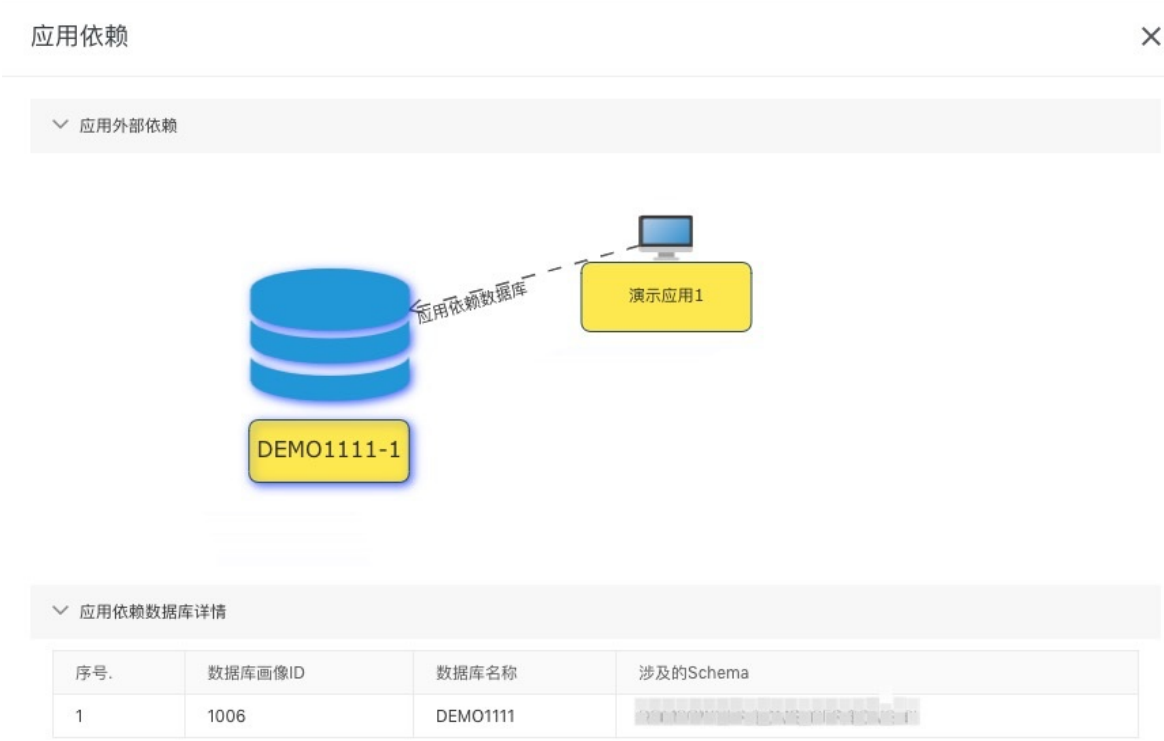
迁移分组评估结果

迁移分组是迁移数据库包含的应用的最小子集，迁移单个应用分组不会对其他分组的数据库节点产生数据依赖。

迁移分组包括**应用节点**以及**数据库节点**。单击应用节点可以查看单个应用节点的评估改造内容。

单应用评估主要分为应用依赖，SQL兼容性，应用改造点三部分：

- 应用依赖：展示目标应用节点依赖的数据库情况，粒度到Schema。



- SQL兼容性：展示应用SQL的兼容性情况，并提供每一条SQL具体转换规则与改造位置。

操作类型	兼容	修改后不兼容	不兼容
INSERT	33	0	0
DELETE	3	0	0
UPDATE	27	0	0
SELECT	412	45	0

- 兼容： 应用无需改造，可以直接使用在目标数据库上；
- 修改后兼容：ADAM提供转换后SQL，直接在应用中替换，即可兼容目标数据库。
- 不兼容： 无法运行在目标数据库上的SQL，需要根据改造建议，修改应用代码或者SQL。

- 应用改造点：展示应用需要改造的地方，并给出改造建议。 目前提供了改造点摘要信息和改造点详情。

6.6. 应用静态改造

通过静态代码扫描方式，对应用的SQL代码进行识别，快速发现因为迁移应用代码需要改造的位置，并使用程序进行自动化改造。对于可以自动替换的SQL进行自动文件改造，对于无法自动替换的SQL提示改造信息。

前提条件

源数据库类型为：Oracle、DB2、Teradata。

新建改造项目

1. 登录ADAM控制台。
2. 在左侧导航栏中，选择**应用评估改造**。
3. 在**应用静态改造**页面的**应用静态改造**步骤中，单击**新建改造项目**。
4. 在**新建改造项目**面板中，填写**源库类型**、**目标库类型**、**目标数据库版本版本**和**框架类型**，上传数据文件，单击**创建**。
5. 创建改造项目需要程序进行自动化改造分析，通常处理为1~10分钟。

> 说明
>
> - 如果目标库为PolarDB O引擎，支持自动改造。
> - 如果目标库为RDS MySQL、AnalyticDB PostgreSQL或PolarDB-X，只展示改造点，不支持自动改造。

应用静态改造详情

改造详情呈现内容主要分为**项目概要**，**改造大盘**，**应用静态改造**三个部分。

项目概要

介绍项目基本信息。

改造大盘

通过图表形式展示特殊代码块和代码块统计。

特殊代码块

- **无需改造**：代码块无需改造，即可运行在目标数据库。
- **自动改造**：代码块由ADAM自动改造，只需要用转换后的文件进行替换。
- **手动改造**：ADAM已指出改造位置和改造内容，需要用户手动操作改造。
- **识别SQL失败**：代码块由于用法特殊或SQL编写异常，导致无法识别出SQL。

代码块统计

- **select**：select代码块数量。
- **insert**：insert代码块数量。
- **update**：update代码块数量。
- **procedure**：存储过程代码块数量。
- **delete**：delete代码块数量。
- **statement**：statement对象执行的代码块数量。

应用静态改造

单击操作列**详情**可以查看代码块改造详情。

7.结构订正指南

7.1. 不支持FOR UPDATE OF

本文介绍不同数据库的表查询方法。

背景说明

Oracle支持Select for update 和 For update of column的语法，这两种语法对于单表来说没有区别，都是锁单表符合条件的相关行。对于多表来说For update会锁多表符合条件的行，For update of 会根据条件锁定相应表的行。例如：

- 单表

  select * from test where id =10 for update

  select * from test where id=10 for update id

  > 说明 锁定id=10的行。

- 多表

  select * from test inner join t1 on test.id=t1.id where test.id=10 for update

  select * from test inner join t1 on test.id=t1.id where test.id=10 for update of test

  > 说明 锁定id=10的test和t1的行。

解决方案

PolarDB目前仅支持For update的语法，不支持For update of的语法。您可以根据业务情况将语法适当调整为For update。

7.2. 不支持自动转换Sample语句

本文介绍不同数据库的采样查询方法。

背景说明

Oracle会使用到Sample的语法进行采样查询，例如：`select * from AAA sample block(name)` 。

解决方案

PolarDB支持使用TABLESAMPLE system（参数），TABLESAMPLE bernoulli（参数） 这两种方式的采样查询。

BERNOULLI以及SYSTEM采样方法传入的参数表示采样表的百分数，取值范围为0-100。BERNOULLI方法扫描整个表并且用指定的几率选择或者忽略行。SYSTEM方法会做块层的采样，每个块都有指定的机会能被选中，被选中块中的所有行都会被返回。在指定较小的采样百分数时，SYSTEM方法要比BERNOULLI方法快很多，但是前者可能由于聚簇效应返回随机性较差的表采样。

示例

```
canno=> create table a(id int);
CREATE TABLE
canno=> insert into a select generate_series(1,1000000);
INSERT 0 1000000
canno=> select count(1) from a tablesample system(1);
 count
-------
  8510
(1 row)
canno=> select count(1) from a tablesample bernoulli(1);
 count
-------
 10004
(1 row)
```

7.3. 不支持Bitmap index

本文介绍Bitmap index以及相关适用场景。

背景说明

Bitmap index是Oracle中的一种特殊索引，适用于low-cardinality字段，可以提高查询速度。由于锁粒度较大，不适合于频繁更新的字段。

对于low-cardinality字段，在PolarDB O引擎中也可以创建Btree索引，创建Btree索引不但能提高查找速度，而且不会出现由于Oracle中大量锁行造成性能下降的问题。

解决方案

根据字段数据评估字段是否为low-cardinality，如果选择性较好而且业务SQL中有这个条件的查询语句，建议创建Btree索引。如果选择性不好，首先评估是否需要创建索引。

示例

- 示例一

  A表的name列选择性较好，在Oracle中有Bitmap index，迁移到PolarDB O引擎后可以创建Btree索引。

- 示例二

  A表的status列选择性不好，90%的值为1，10%的值为0，业务上的查询都条件是status=1，这种情况下可以考虑不创建索引。如果业务上有status=0的查询，建议创建Btree索引。

7.4. 不支持Cluster index

本文介绍Cluster的语法并给出相关示例。

背景说明

Oracle中的 Cluster是一种共享公共列，在相同块中存储相关数据的表。当使用Cluster表时，单个数据块可以包含来自多个表的行，对于经常发生连接的表可以减少磁盘IO。

在PolarDB O引擎中不支持共享列的表，但PolarDB O引擎提供按某一个特定索引重新组织堆表数据的能力。

解决方案

Cluster语法

```
CLUSTER [VERBOSE] table_name [ USING index_name ]
```

Cluster会对数据的物理存储顺序进行调整，重新按照指定索引进行排序。注意Cluster是一次性操作，后续对于表的更改不会自动调整。

示例

```
#Cluster the table employees on the basis of its index employees_ind:
CLUSTER employees USING employees_ind;
#Cluster the employees table using the same index that was used before:
CLUSTER employees;
#Cluster all tables in the database that have previously been clustered:
CLUSTER;
```

7.5. 不支持异常类型

本文对不同数据库解释了异常类型的处理方法 。

背景说明

PolarDB O引擎不支持Oracle的异常类型，如果使用Oracle的异常类型会提示语法错误：

```
ERROR:  "utl_smtp.transient_error" is not a known exception
```

PolarDB O引擎中可以覆盖Oracle的异常类型，请根据PolarDB O引擎的异常处理方式进行改造。

解决方案

异常处理语法：

```
[ <<label>> ]
[ DECLARE
    declarations ]
BEGIN
    statements
EXCEPTION
    WHEN condition [ OR condition ... ] THEN
        handler_statements
    [ WHEN condition [ OR condition ... ] THEN
          handler_statements
      ... ]
END;
```

- 如果没有发生异常，程序会正常执行所有的 `statements` 。
- 如果在 `statements` 内发生了一个异常，会支持跳转到到EXCEPTION部分开始执行。系统会在WHEN列表中匹配第一个 `condition` ，如果没成功匹配，异常会继续向上抛。 `condition` 可以是附录中异常名称的任意一种。

> **说明** PolarDB O引擎支持的异常类型请见附录。

示例

除零错误异常处理示例：

```
INSERT INTO mytab(firstname, lastname) VALUES('Tom', 'Jones');
BEGIN
    UPDATE mytab SET firstname = 'Joe' WHERE lastname = 'Jones';
    x := x + 1;
    y := x / 0;
EXCEPTION
    WHEN division_by_zero THEN
        RAISE NOTICE 'caught division_by_zero';
        RETURN x;
END;
```

附录

Error Code	Condition Name
Class 00 — Successful Completion	
00000	successful_completion
Class 01 — Warning	
01000	warning
0100C	dynamic_result_sets_returned
01008	implicit_zero_bit_padding
01003	null_value_eliminated_in_set_function
01007	privilege_not_granted
01006	privilege_not_revoked
01004	string_data_right_truncation
01P01	deprecated_feature
Class 02 — No Data (this is also a warning class per the SQL standard)	
02000	no_data
02001	no_additional_dynamic_result_sets_returned
Class 03 — SQL Statement Not Yet Complete	
03000	sql_statement_not_yet_complete

Error Code	Condition Name
Class 08 — Connection Exception	
08000	connection_exception
08003	connection_does_not_exist
08006	connection_failure
08001	sqlclient_unable_to_establish_sqlconnection
08004	sqlserver_rejected_establishment_of_sqlconnection
08007	transaction_resolution_unknown
08P01	protocol_violation
Class 09 — Triggered Action Exception	
09000	triggered_action_exception
Class 0A — Feature Not Supported	
0A000	feature_not_supported
Class 0B — Invalid Transaction Initiation	
0B000	invalid_transaction_initiation
Class 0F — Locator Exception	
0F000	locator_exception
0F001	invalid_locator_specification
Class 0L — Invalid Grantor	
0L000	invalid_grantor
0LP01	invalid_grant_operation
Class 0P — Invalid Role Specification	
0P000	invalid_role_specification
Class 0Z — Diagnostics Exception	
0Z000	diagnostics_exception
0Z002	stacked_diagnostics_accessed_without_active_handler
Class 20 — Case Not Found	

Error Code	Condition Name
20000	case_not_found
Class 21 — Cardinality Violation	
21000	cardinality_violation
Class 22 — Data Exception	
22000	data_exception
2202E	array_subscript_error
22021	character_not_in_repertoire
22008	datetime_field_overflow
22012	division_by_zero
22005	error_in_assignment
2200B	escape_character_conflict
22022	indicator_overflow
22015	interval_field_overflow
2201E	invalid_argument_for_logarithm
22014	invalid_argument_for_ntile_function
22016	invalid_argument_for_nth_value_function
2201F	invalid_argument_for_power_function
2201G	invalid_argument_for_width_bucket_function
22018	invalid_character_value_for_cast
22007	invalid_datetime_format
22019	invalid_escape_character
2200D	invalid_escape_octet
22025	invalid_escape_sequence
22P06	nonstandard_use_of_escape_character
22010	invalid_indicator_parameter_value
22023	invalid_parameter_value

Error Code	Condition Name
22013	invalid_preceding_or_following_size
2201B	invalid_regular_expression
2201W	invalid_row_count_in_limit_clause
2201X	invalid_row_count_in_result_offset_clause
2202H	invalid_tablesample_argument
2202G	invalid_tablesample_repeat
22009	invalid_time_zone_displacement_value
2200C	invalid_use_of_escape_character
2200G	most_specific_type_mismatch
22004	null_value_not_allowed
22002	null_value_no_indicator_parameter
22003	numeric_value_out_of_range
2200H	sequence_generator_limit_exceeded
22026	string_data_length_mismatch
22001	string_data_right_truncation
22011	substring_error
22027	trim_error
22024	unterminated_c_string
2200F	zero_length_character_string
22P01	floating_point_exception
22P02	invalid_text_representation
22P03	invalid_binary_representation
22P04	bad_copy_file_format
22P05	untranslatable_character
2200L	not_an_xml_document
2200M	invalid_xml_document

Error Code	Condition Name
2200N	invalid_xml_content
2200S	invalid_xml_comment
2200T	invalid_xml_processing_instruction
Class 23 — Integrity Constraint Violation	
23000	integrity_constraint_violation
23001	restrict_violation
23502	not_null_violation
23503	foreign_key_violation
23505	unique_violation
23514	check_violation
23P01	exclusion_violation
Class 24 — Invalid Cursor State	
24000	invalid_cursor_state
Class 25 — Invalid Transaction State	
25000	invalid_transaction_state
25001	active_sql_transaction
25002	branch_transaction_already_active
25008	held_cursor_requires_same_isolation_level
25003	inappropriate_access_mode_for_branch_transaction
25004	inappropriate_isolation_level_for_branch_transaction
25005	no_active_sql_transaction_for_branch_transaction
25006	read_only_sql_transaction
25007	schema_and_data_statement_mixing_not_supported
25P01	no_active_sql_transaction
25P02	in_failed_sql_transaction

Error Code	Condition Name
25P03	idle_in_transaction_session_timeout
Class 26 — Invalid SQL Statement Name	
26000	invalid_sql_statement_name
Class 27 — Triggered Data Change Violation	
27000	triggered_data_change_violation
Class 28 — Invalid Authorization Specification	
28000	invalid_authorization_specification
28P01	invalid_password
Class 2B — Dependent Privilege Descriptors Still Exist	
2B000	dependent_privilege_descriptors_still_exist
2BP01	dependent_objects_still_exist
Class 2D — Invalid Transaction Termination	
2D000	invalid_transaction_termination
Class 2F — SQL Routine Exception	
2F000	sql_routine_exception
2F005	function_executed_no_return_statement
2F002	modifying_sql_data_not_permitted
2F003	prohibited_sql_statement_attempted
2F004	reading_sql_data_not_permitted
Class 34 — Invalid Cursor Name	
34000	invalid_cursor_name
Class 38 — External Routine Exception	
38000	external_routine_exception
38001	containing_sql_not_permitted
38002	modifying_sql_data_not_permitted
38003	prohibited_sql_statement_attempted

Error Code	Condition Name
38004	reading_sql_data_not_permitted
Class 39 — External Routine Invocation Exception	
39000	external_routine_invocation_exception
39001	invalid_sqlstate_returned
39004	null_value_not_allowed
39P01	trigger_protocol_violated
39P02	srf_protocol_violated
39P03	event_trigger_protocol_violated
Class 3B — Savepoint Exception	
3B000	savepoint_exception
3B001	invalid_savepoint_specification
Class 3D — Invalid Catalog Name	
3D000	invalid_catalog_name
Class 3F — Invalid Schema Name	
3F000	invalid_schema_name
Class 40 — Transaction Rollback	
40000	transaction_rollback
40002	transaction_integrity_constraint_violation
40001	serialization_failure
40003	statement_completion_unknown
40P01	deadlock_detected
Class 42 — Syntax Error or Access Rule Violation	
42000	syntax_error_or_access_rule_violation
42601	syntax_error
42501	insufficient_privilege
42846	cannot_coerce

Error Code	Condition Name
42803	grouping_error
42P20	windowing_error
42P19	invalid_recursion
42830	invalid_foreign_key
42602	invalid_name
42622	name_too_long
42939	reserved_name
42804	datatype_mismatch
42P18	indeterminate_datatype
42P21	collation_mismatch
42P22	indeterminate_collation
42809	wrong_object_type
428C9	generated_always
42703	undefined_column
42883	undefined_function
42P01	undefined_table
42P02	undefined_parameter
42704	undefined_object
42701	duplicate_column
42P03	duplicate_cursor
42P04	duplicate_database
42723	duplicate_function
42P05	duplicate_prepared_statement
42P06	duplicate_schema
42P07	duplicate_table
42712	duplicate_alias

Error Code	Condition Name
42710	duplicate_object
42702	ambiguous_column
42725	ambiguous_function
42P08	ambiguous_parameter
42P09	ambiguous_alias
42P10	invalid_column_reference
42611	invalid_column_definition
42P11	invalid_cursor_definition
42P12	invalid_database_definition
42P13	invalid_function_definition
42P14	invalid_prepared_statement_definition
42P15	invalid_schema_definition
42P16	invalid_table_definition
42P17	invalid_object_definition
Class 44 — WITH CHECK OPTION Violation	
44000	with_check_option_violation
Class 53 — Insufficient Resources	
53000	insufficient_resources
53100	disk_full
53200	out_of_memory
53300	too_many_connections
53400	configuration_limit_exceeded
Class 54 — Program Limit Exceeded	
54000	program_limit_exceeded
54001	statement_too_complex
54011	too_many_columns

Error Code	Condition Name
54023	too_many_arguments
Class 55 — Object Not In Prerequisite State	
55000	object_not_in_prerequisite_state
55006	object_in_use
55P02	cant_change_runtime_param
55P03	lock_not_available
Class 57 — Operator Intervention	
57000	operator_intervention
57014	query_canceled
57P01	admin_shutdown
57P02	crash_shutdown
57P03	cannot_connect_now
57P04	database_dropped
Class 58 — System Error (errors external to PostgreSQL itself)	
58000	system_error
58030	io_error
58P01	undefined_file
58P02	duplicate_file
Class 72 — Snapshot Failure	
72000	snapshot_too_old
Class F0 — Configuration File Error	
F0000	config_file_error
F0001	lock_file_exists
Class HV — Foreign Data Wrapper Error (SQL/MED)	
HV000	fdw_error
HV005	fdw_column_name_not_found

Error Code	Condition Name
HV002	fdw_dynamic_parameter_value_needed
HV010	fdw_function_sequence_error
HV021	fdw_inconsistent_descriptor_information
HV024	fdw_invalid_attribute_value
HV007	fdw_invalid_column_name
HV008	fdw_invalid_column_number
HV004	fdw_invalid_data_type
HV006	fdw_invalid_data_type_descriptors
HV091	fdw_invalid_descriptor_field_identifier
HV00B	fdw_invalid_handle
HV00C	fdw_invalid_option_index
HV00D	fdw_invalid_option_name
HV090	fdw_invalid_string_length_or_buffer_length
HV00A	fdw_invalid_string_format
HV009	fdw_invalid_use_of_null_pointer
HV014	fdw_too_many_handles
HV001	fdw_out_of_memory
HV00P	fdw_no_schemas
HV00J	fdw_option_name_not_found
HV00K	fdw_reply_handle
HV00Q	fdw_schema_not_found
HV00R	fdw_table_not_found
HV00L	fdw_unable_to_create_execution
HV00M	fdw_unable_to_create_reply
HV00N	fdw_unable_to_establish_connection
Class P0 — PL/pgSQL Error	

Error Code	Condition Name
P0000	plpgsql_error
P0001	raise_exception
P0002	no_data_found
P0003	too_many_rows
P0004	assert_failure
Class XX — Internal Error	
XX000	internal_error
XX001	data_corrupted
XX002	index_corrupted

7.6. 不支持AGGREGATE关键字

本文介绍AGGREGATE关键字的使用方法。

背景说明

Oracle中支持使用AGGREGATE创建一个自定义聚合函数，比如创建一个查询第二大值的聚合函数：

```
--创建object type
CREATE or REPLACE type secmax_context AS object(
  firmax NUMBER,
  secmax NUMBER,
  static FUNCTION ODCIAggregateInitialize(sctx IN OUT secmax_context) RETURN NUMBER,
  member FUNCTION ODCIAggregateIterate(self  IN OUT secmax_context,value IN NUMBER) RETURN
NUMBER,
  member FUNCTION ODCIAggregateMerge(self IN OUT secmax_context, ctx2 IN secmax_context)RET
URN NUMBER,
  member FUNCTION ODCIAggregateTerminate(self IN secmax_context,returnValue OUT NUMBER,flag
s IN NUMBER) RETURN NUMBER
);
--实现object type
create or replace type body secmax_context is
static function ODCIAggregateInitialize(sctx IN OUT secmax_context) return number is
begin
  sctx := secmax_context(0, 0);
  return ODCIConst.Success;
end;
member function ODCIAggregateIterate(self IN OUT secmax_context, value IN number) return nu
mber is
begin
  if value > self.firmax then
    self.secmax := self.firmax;
    self.firmax := value;
  elsif value > self.secmax then
```

```
  elsif value > self.secmax then
    self.secmax := value;
  end if;
  return ODCIConst.Success;
end;
member function ODCIAggregateTerminate(self IN secmax_context, returnValue OUT number, flag
s IN number) return number is
begin
  returnValue := self.secmax;
  return ODCIConst.Success;
end;
member function ODCIAggregateMerge(self IN OUT secmax_context, ctx2 IN secmax_context) retu
rn number is
begin
  if ctx2.firmax > self.firmax then
    if ctx2.secmax > self.firmax then
      self.secmax := ctx2.secmax;
          else
      self.secmax := self.firmax;
    end if;
    self.firmax := ctx2.firmax;
  elsif ctx2.firmax > self.secmax then
    self.secmax := ctx2.firmax;
  end if;
  return ODCIConst.Success;
end;
end;
--创建聚合函数
CREATE FUNCTION SecMax (input NUMBER) RETURN NUMBER PARALLEL_ENABLE AGGREGATE USING secmax_
context;
select secmax(id) from test;
```

解决方案

在Polardb中创建自定聚合函数的语法如下所示：

```
CREATE AGGREGATE name ( [ [ argmode ] [ argname ] arg_data_type [ , ... ] ]
                        ORDER BY [ argmode ] [ argname ] arg_data_type [ , ... ] ) (
    SFUNC = sfunc,
    STYPE = state_data_type
    [ , SSPACE = state_data_size ]
    [ , FINALFUNC = ffunc ]
    [ , FINALFUNC_EXTRA ]
    [ , FINALFUNC_MODIFY = { READ_ONLY | SHAREABLE | READ_WRITE } ]
    [ , INITCOND = initial_condition ]
    [ , HYPOTHETICAL ]
    [ , PARALLEL = { SAFE | RESTRICTED | UNSAFE } ]
)
```

示例

该示例为自定义一个聚合函数求最大值

```
canno=> create publication pub1 for all tables ;
CREATE PUBLICATION
canno=> CREATE AGGREGATE max2(int)
canno-> (
canno(>   INITCOND = 0,
canno(>   SFUNC = second_max,
canno(>   STYPE = int
canno(> );
CREATE AGGREGATE
canno=>
canno=> create or replace function re_max2 (int,int) returns int as $$
canno$> declare
canno$>   result int;
canno$> begin
canno$>   if $1 <=$2
canno$>   then
canno$>       result=$2;
canno$>   else
canno$>       result=$1;
canno$>   end if;
canno$>   return result;
canno$> end;
canno$> $$ language plpgsql ;
CREATE FUNCTION
canno=> select max2(id) from a;
 max2
------
   10
(1 row)
```

7.7. 在PL/SQL中不支持调用其他语言代码

本文说明PL/SQL调用其他语言代码的支持情况。

背景说明

目前在PolarDB O引擎中不支持使用其他语言编写的函数、存储过程。

解决方案

使用PolarDB O引擎支持的SPL语言，或Postgresql支持的PL/pgSQL实现相应的业务逻辑。

示例

- 使用Java实现的函数：

```
create or replace function foo return varchar is external language java name 'hello'
```

- 改写为SPL语法的函数：

```
CREATE OR REPLACE FUNCTION foo
    RETURN VARCHAR2
IS
BEGIN
    RETURN 'That''s All Folks!';
END simple_function;
```

详情请参见创建函数。

- 或改写为PL/pgSQL语法的函数：

```
CREATE FUNCTION foo(integer, text) RETURNS integer
AS 'function body text'
LANGUAGE plpgsql;
```

详情请参见 Structure of PL/pgSQL。

7.8. 不支持PARALLEL_ENABLE

本文介绍创建聚合函数的方法。

背景说明

Oracle中支持为自定义聚合函数开启并行，在PolarDB O引擎中同样是支持的，但是语法有些差异。这里介绍下PolarDB O引擎中如何创建自定义聚合函数、开启并行。

解决方案

在PolarDB O引擎中创建自定义聚合函数的语法：

```
CREATE AGGREGATE name ( [ argmode ] [ argname ] arg_data_type [ , ... ] ) (
    SFUNC = sfunc,
    STYPE = state_data_type
    [ , SSPACE = state_data_size ]
    [ , FINALFUNC = ffunc ]
    [ , FINALFUNC_EXTRA ]
    [ , FINALFUNC_MODIFY = { READ_ONLY | SHAREABLE | READ_WRITE } ]
    [ , COMBINEFUNC = combinefunc ]
    [ , SERIALFUNC = serialfunc ]
    [ , DESERIALFUNC = deserialfunc ]
    [ , INITCOND = initial_condition ]
    [ , MSFUNC = msfunc ]
    [ , MINVFUNC = minvfunc ]
    [ , MSTYPE = mstate_data_type ]
    [ , MSSPACE = mstate_data_size ]
    [ , MFINALFUNC = mffunc ]
    [ , MFINALFUNC_EXTRA ]
    [ , MFINALFUNC_MODIFY = { READ_ONLY | SHAREABLE | READ_WRITE } ]
    [ , MINITCOND = minitial_condition ]
    [ , SORTOP = sort_operator ]
    [ , PARALLEL = { SAFE | RESTRICTED | UNSAFE } ]
)
```

示例

PolarDB O引擎中支持常规的聚合函数，如果需要实现特殊功能，您可以自定义相关函数即可（sfunc、stype、FINALFUNC）。

例如，指定符号连接聚合，同时适用于大部分场景：

```
create aggregate launch_concat(text,text) (
  sfunc = pg_catalog.string_agg_transfn,
  stype = internal,
  FINALFUNC = pg_catalog.string_agg_finalfn
);
```

结果为：

```
select launch_concat(id::text, ',') from generate_series(1,10) t(id);
    launch_concat
----------------------
 1,2,3,4,5,6,7,8,9,10
```

7.9. 不支持PIPELINED关键字

本文介绍pipelined的使用场景。

背景说明

Oracle中使用pipelined实现流式返回多条记录，在实际应用中经常使用。在PolarDB O引擎中不支持pipelined语法，但可以使用setof实现同样功能。

解决方案

对于Oracle中Pipelined语法，在PolsrDB-O创建函数时，使用Setof语法替换：

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | TRANSFORM { FOR TYPE type_name } [, ... ]
    | WINDOW
    | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
    | PARALLEL { UNSAFE | RESTRICTED | SAFE }
    | COST execution_cost
    | ROWS result_rows
    | SET configuration_parameter { TO value | = value | FROM CURRENT }
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  } ...
rettype
The return data type (optionally schema-qualified). The return type can be a base, composit
e, or domain type, or can reference the type of a table column. Depending on the implementa
tion language it might also be allowed to specify “pseudo-types” such as cstring. If the fu
nction is not supposed to return a value, specify void as the return type.
When there are OUT or INOUT parameters, the RETURNS clause can be omitted. If present, it m
ust agree with the result type implied by the output parameters: RECORD if there are multip
le output parameters, or the same type as the single output parameter.
The SETOF modifier indicates that the function will return a set of items, rather than a si
ngle item.
The type of a column is referenced by writing table_name.column_name%TYPE.
```

详情请参见https://www.postgresql.org/docs/11/sql-alteraggregate.html。

示例

- 在Oracle中：

```
create or replace function split
(
  p_list varchar2,
  p_del varchar2 := ','
) return split_tbl pipelined
is
  l_idx pls_integer;
  l_list varchar2(32767) := p_list;
  l_value varchar2(32767);
begin
  loop
    l_idx := instr(l_list,p_del);
    if l_idx > 0 then
      pipe row(trim(substr(l_list,1,l_idx-1)));
      l_list := substr(l_list,l_idx+length(p_del));
    else
      pipe row(trim(l_list));
      exit;
    end if;
  end loop;
  return;
end split;
```

- 在PolarDB O引擎中：

```
create or replace function rsf1(id int) returns setof int as $$
declare
begin
  for i in 0..abs(id) loop
    return next i;
  end loop;
end;
$$ language plpgsql strict;
```

7.10. PolarDB系统保留列名

本文对PolarDB系统保留列名做了如下说明。

背景说明

在PolarDB O引擎中不支持使用保留的系统列名：ctid、oid、cmin、cmax、xmin、xmax，使用系统列名会有如下报错：

```
ERROR:  column name "ctid" conflicts with a system column name
```

解决方案

- 您可以通过更改列名的方式处理冲突列名。
- 创建与原表不同名的影子表，修改冲突列名。
- 创建与原表同名的视图，将列名映射为原表列名。

示例

- 更改列名

  原列名为：

  ```
  create table foo(oid varchar(10))
  ```

  修改为：

  ```
  create table foo(p_oid varchar(10))
  ```

- 创建新表

  原表为：

  ```
  create table foo(oid varchar(10), ctid int, xmin int)
  ```

  创建新表：

  ```
  create table __foo(p_oid varchar(10), p_ctid int, p_xmin int);
  ```

  创建视图：

  ```
  create view foo as select p_oid as oid, p_ctid as ctid, p_xmin as xmin from __foo;
  ```

7.11. Trigger不支持非DML事件

背景说明

PolarDB普通触发器不支持DDL语句，如：DROP、CREATE、ALTER等，需要修改为PolarDB的事件触发器。

解决方案

PolarDB事件触发器语法如下表所示：

```
CREATE EVENT TRIGGER name
    ON event
    [ WHEN filter_variable IN (filter_value [, ... ]) [ AND ... ] ]
    EXECUTE { FUNCTION | PROCEDURE } function_name()
```

> **说明**
>
> - 支持的event包含ddl_command_start、ddl_command_end、 table_rewrite和sql_drop。
> - ddl_command_start事件就在CREATE、 ALTER、DROP、SECURITY LABEL、 COMMENT、GRANT或者REVOKE 命令的执行之前发生。在事件触发器引发前不会做受影响对象是否存在的检查。
> - ddl_command_end事件就在同一组命令的执行之后发生。
> - sql_drop事件为任何删除数据库对象的操作在 ddl_command_end事件触发器之前发生。
> - table_rewrite事件在表被命令ALTER TABLE和 ALTER TYPE的某些动作重写之前发生。

示例

在Oracle中：

```
create or replace trigger apps_no_ddl
before create or alter or drop or truncate
on database
begin
raise_application_error(-20001,'不允许用DDL操作APPS用户的对象');
end;
```

在PolarDB中：

```
CREATE OR REPLACE FUNCTION abort_any_command()
  RETURNS event_trigger
 LANGUAGE plpgsql
  AS $$
BEGIN
  RAISE EXCEPTION 'command % is disabled', tg_tag;
END;
$$;
CREATE EVENT TRIGGER apps_no_ddl ON ddl_command_start
   EXECUTE FUNCTION abort_any_command();
```

7.12. Aggregate函数不支持keep关键字

本文介绍了聚合函数的关键字使用。

背景说明

聚合函数的使用不支持keep关键字，例如：

销售表：

```
SQL> select * from criss_sales where dept_id = 'D02' order by sale_date ;
DEPT_ID SALE_DATE   GOODS_TYPE    SALE_CNT
------- ----------- ---------- -----------
D02     2014/3/6    G00                500
D02     2014/3/6    G01                430
D02     2014/4/8    G02                100
D02     2014/4/27   G01                300
D02     2014/5/2    G03                900
```

此时有个新需求,希望查看部门 D02 内，销售记录时间最早，销售量最小的记录。

```
SQL> select
  2      dept_id
  3     ,min(sale_cnt)keep ( dense_rank first order by sale_date) min_early_date
  4  from criss_sales
  5  where dept_id = 'D02'
  6  group by dept_id
  7  ;
DEPT_ID  MIN_EARLY_DATE
------- ----------
D02            430
```

解决方案

您可以通过改写SQL来代替keep语法。

示例

```
canno=> select dept_id,min(sal_cnt) from (select dense_rank() over (partition by dept_id or
der by sale_date),* from  criss_sales where dept_id = 'D02' ) t where  dense_rank=1 group b
y dept_id;
 dept_id | min
---------+-----
 D02     | 430
(1 row)
```

7.13. SYS_CONTEXT（）函数支持SESSION_USER、CURRENT_USER、CURRENT_SCHEMA、HOST、IP_ADDRESS 、SERVER_HOST作为参数

本文介绍SYS_CONTEXT（）函数支持的参数。

背景说明

Oracle官方建议使用SYS_CONTEXT函数获取系统变量，PolarDB O引擎支持SYS_CONTEXT函数的部分功能。

解决方案

SYS_CONTEXT函数使用方法如下表所示：

```
SELECT SYS_CONTEXT('USERENV', attribute) FROM dual;
```

在PolarDB O引擎中attribute只支持：

```
SESSION_USER、CURRENT_USER、CURRENT_SCHEMA、HOST、IP_ADDRESS、SERVER_HOST
```

如果需要使用其他attribute，可以使用自建函数实现相同功能：

```
create or replace function userenv(anynonarray) returns anynonarray as $$
declare
begin
  case lower($1)
  when '按需配置' then
    return 自定义函数();
  when '按需配置' then
    return 自定义函数();
  else
    return null;
  end case;
end;
$$ language plpgsql strict;
```

示例

```
select SYS_CONTEXT('USERENV', 'HOST') from dual;
"42.120.72.81/32"
select SYS_CONTEXT('USERENV', 'CURRENT_USER') from dual;
"admin"
```

7.14. 不支持USERENV

本文介绍获取会话变量的函数。

背景说明

Oracle中使用USERENV函数获取当前会话变量，USERENV是Oracle向下兼容的函数，Oracle官方建议使用SYS_CONTEXT函数进行替换，PolarDB O引擎支持使用SYS_CONTEXT函数获取会话变量。

解决方案

SYS_CONTEXT函数语法：

```
SYS_CONTEXT('USERENV', attribute)
```

在PolarDB O引擎中attribute支持：

```
SESSION_USER、CURRENT_USER、CURRENT_SCHEMA、HOST、IP_ADDRESS、SERVER_HOST
```

如果需要使用其他attribute，可以使用自建函数实现相同功能：

```
create or replace function userenv(anynonarray) returns anynonarray as $$
declare
begin
  case lower($1)
  when '按需配置' then
    return 自定义函数();
  when '按需配置' then
    return 自定义函数();
  else
    return null;
  end case;
end;
$$ language plpgsql strict;
```

示例

```
select SYS_CONTEXT('USERENV', 'HOST') from dual;
"42.120.72.81/32"
select SYS_CONTEXT('USERENV', 'CURRENT_USER') from dual;
"admin"
```

7.15. Varray类型的元素类型不能是刚定义的类型

本文介绍Varray类型的定义和使用方法。

背景说明

Oracle中Varray可以嵌套定义，即Varray定义中可以引用已经定义的Varry，例如：

```
declare
    TYPE VAR_TYP IS VARRAY(20) OF NUMBER;
    TYPE VAR_TYP_2 IS VARRAY(10) OF VAR_TYP;
begin
    return '';
END pkg_subtype;
```

在PolarDB O引擎中，支持Varray类型的定义，但是不支持嵌套定义的语法，需要做简单改造。

解决方案

PolarDB O引擎中Varry的定义方法：

```
TYPE varraytype IS { VARRAY | VARYING ARRAY }(maxsize)
  OF { datatype | objtype };
```

> **说明** OF后的具体类型只支持datatype与objtype，不支持varraytype。

示例

- 对于嵌套定义的Varry做改造：

```
declare
    TYPE VAR_TYP IS VARRAY(20) OF NUMBER;
    TYPE VAR_TYP_2 IS VARRAY(10) OF NUMBER;
begin
    return '';
END pkg_subtype;
```

```
DECLARE
    TYPE dname_varray_typ IS VARRAY(4) OF VARCHAR2(14);
    dname_varray    dname_varray_typ;
    CURSOR dept_cur IS SELECT dname FROM dept ORDER BY dname;
    i               INTEGER := 0;
BEGIN
    dname_varray := dname_varray_typ(NULL, NULL, NULL, NULL);
    FOR r_dept IN dept_cur LOOP
        i := i + 1;
        dname_varray(i) := r_dept.dname;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('DNAME');
    DBMS_OUTPUT.PUT_LINE('----------');
    FOR j IN 1..i LOOP
        DBMS_OUTPUT.PUT_LINE(dname_varray(j));
    END LOOP;
END;
```

7.16. dbtimezone

本文介绍dbtimezone的使用场景。

背景信息

`timestamp at time zone dbtimezone` 中的 `dbtimezone` 在PolarDB O引擎中不支持。

```
SQL> SELECT timestamp '1970-01-01 0:0:0 -0:0' at time zone dbtimezone FROM dual;
TIMESTAMP'1970-01-010:0:0-0:0'ATTIMEZONEDBTIMEZONE
---------------------------------------------------------------------------
01-JAN-70 12.00.00.000000000 AM +00:00
SQL>
SQL>  SELECT CURRENT_TIMESTAMP  FROM dual;
CURRENT_TIMESTAMP
---------------------------------------------------------------------------
04-AUG-20 06.04.42.125059 PM +08:00
```

解决方案

建议依据Oracle数据库中dbtimezone设置修改为具体的时区值（如：+00:00）。PolarDB的时区可以依据timezone参数进行修改，可以在server级调整，也可以在session级调整。

```
van=> set timezone='+08:00';
SET
van=> select current_timestamp;
        current_timestamp
----------------------------------
04-AUG-20 02:03:27.686021 -08:00
(1 row)
van=> set timezone='UTC';
SET
van=> select current_timestamp;
        current_timestamp
         ----------------------------------
 04-AUG-20 10:03:49.071727 +00:00
(1 row)
```

详情请参见https://www.postgresql.org/docs/11/functions-datetime.html。

7.17. NESTED TABLE

本文介绍NESTED TABLE的使用方法和使用场景。

背景介绍

Oracle nested table详细功能请参见http://www.orafaq.com/wiki/NESTED_TABLE。

NESTED TABLE是一种Oracle数据类型，用于支持包含多值属性的列，在本例中，列可以容纳整个子表。

创建具有NESTED TABLE的表：

```
CREATE OR REPLACE TYPE my_tab_t AS TABLE OF VARCHAR2(30);
```

```
CREATE TABLE nested_table (id NUMBER, col1 my_tab_t)
       NESTED TABLE col1 STORE AS col1_tab;
```

将数据插入表：

```
INSERT INTO nested_table VALUES (1, my_tab_t('A'));
INSERT INTO nested_table VALUES (2, my_tab_t('B', 'C'));
INSERT INTO nested_table VALUES (3, my_tab_t('D', 'E', 'F'));
COMMIT;
```

从NESTED TABLE中选择：

```
SQL> SELECT * FROM nested_table;
        ID COL1
---------- ------------------------
         1 MY_TAB_T('A')
         2 MY_TAB_T('B', 'C')
         3 MY_TAB_T('D', 'E', 'F')
```

取消嵌套子表：

```
SQL> SELECT id, COLUMN_VALUE FROM nested_table t1, TABLE(t1.col1) t2;
        ID COLUMN_VALUE
---------- ------------------------
         1 A
         2 B
         2 C
         3 D
         3 E
         3 F
6 rows selected.
```

PostgreSQL Nested Table兼容

PostgreSQL使用数组+复合类型，可以实现同样场景需求。

1. 创建复合类型。

```
postgres=# create type thisisnesttable1 as (c1 int, c2 int, c3 text, c4 timestamp);
CREATE TYPE
or
create table nesttablename (...);  -- 隐含创建composite type
```

> **说明** 如果系统中曾经已经创建了这个类型，或者曾经已经创建过一个即将使用的TABLE，则不需要再次创建。

2. 创建Nested Table。

```
postgres=# create table hello (id int, info text, nst thisisnesttable1[]);
CREATE TABLE
```

> **说明** thisisnesttable1作为hello表的Nested Table

3. 插入数据。

```
postgres=# insert into hello values (1,'test',array['(1,2,"abcde","2018-01-01 12:00:00"
)'::thisisnesttable1,  '(2,3,"abcde123","2018-01-01 12:00:00")'::thisisnesttable1]);
INSERT 0 1
或使用row构造法
insert into hello values (
  1,
  'test',
  (array
    [
      row(1,2,'hello',now()),
      row(1,3,'hello',now())
    ]
  )::thisisnesttable1[]
);
```

> **说明** 多行以数组存入，一个nested table的最大限制1GB（即PostgreSQL varying type的存储上限）。

详情请参见https://www.postgresql.org/docs/11/sql-expressions.html#SQL-SYNTAX-ROW-CONSTRUCTORS。

4. 查询。

```
postgres=# select * from hello ;
 id | info |                                        nst

----+------+-----------------------------------------------------------------------------
-------
  1 | test | {"(1,2,abcde,\"2018-01-01 12:00:00\")","(2,3,abcde123,\"2018-01-01 12:00:0
0\")"}
(1 row)
```

5. 使用unnest可以解开Nested Table的内容。

```
postgres=# select id,info,(unnest(nst)).* from hello ;
 id | info | c1 | c2 |    c3    |         c4
----+------+----+----+----------+--------------------
  1 | test |  1 |  2 | abcde    | 2018-01-01 12:00:00
  1 | test |  2 |  3 | abcde123 | 2018-01-01 12:00:00
(2 rows)
postgres=# select id,info,(unnest(nst)).c1 from hello ;
 id | info | c1
----+------+----
  1 | test |  1
  1 | test |  2
(2 rows)
```

7.18. REGEXP_LIKE函数

本文介绍REGEXP_LIKE函数的使用场景。

背景说明

Oracle支持REGEXP_LIKE函数，但是PolarDB O引擎不支持该函数。

在Oracle中具体使用如下：

```
SQL> SELECT * FROM xmldemo WHERE REGEXP_LIKE (B, '^f([a-z]+)e$');
         A B
---------- --------------------
        20 firstline
```

解决方案

使用POSIX正则表达式进行改写 `，similar to` 和 `'～'` 可以支持正则表达式匹配。

```
van=> SELECT * FROM xmldemo WHERE b SIMILAR to 'f([a-z]+)e';
 a  |     b
    ----+-----------
 20 | firstline
(1 row)
```

或者

```
van=> SELECT * FROM xmldemo WHERE b ~ 'f([a-z]+)e';
a  |     b
    ----+-----------
 20 | firstline
(1 row)
```

详情请参见https://www.postgresql.org/docs/11/functions-matching.html#FUNCTIONS-POSIX-REGEXP

7.19. EXTRACTVALUE函数

本文介绍EXTRACTVALUE函数的使用方法。

背景说明

在Oracle中支持EXTRACTVALUE函数来解析xml。PolarDB O引擎中不支持该函数。

```
SQL> set linesize 300;
SQL> select * from dbmgr.xmldemo;
         A B
---------- ------------------------------------------------------------------------------------------
----------------------------
         6 <A>3</A>
SQL> select EXTRACTVALUE(xmltype(B),'/A') from dbmgr.xmldemo;EXTRACTVALUE(XMLTYPE(B),'/A')
------------------------------------------------------------------------------------------------------
----------------------------
3
```

解决方案

使用xpath函数来实现相同功能，详情请参见https://www.postgresql.org/docs/11/functions-xml.html。

```
van=> create table xmldemo
van=> (
          A NUMBER,
           B VARCHAR2(100)
         );
van=>  insert into xmldemo values(1,'first line');
INSERT 0 1
van=>  insert into xmldemo values(2,'line 2');
INSERT 0 1
van=> SELECT xpath('/a/text()',xmlforest(a AS A)) from xmldemo ;
 xpath
 -------
{1}
{2}
(2 rows)
van=>
```

7.20. Unpivot 列转行

背景说明

Oracle中UNPIVOT语法：

```
SELECT ...
FROM  ...
UNPIVOT [INCLUDE|EXCLUDE NULLS]
   (unpivot_clause
    unpivot_for_clause
    unpivot_in_clause )
WHERE ...
```

示例如下：

```
SQL> SELECT *
  2  FROM  pivoted_data
  3  UNPIVOT (
  4       deptsal                    --<-- unpivot_clause
  5       FOR saldesc                --<-- unpivot_for_clause
  6       IN (d10_sal, d20_sal, d30_sal, d40_sal)  --<-- unpivot_in_clause
  7          );
JOB       SALDESC       DEPTSAL
---------- ---------- ----------
CLERK     D10_SAL          1430
CLERK     D20_SAL          2090
CLERK     D30_SAL          1045
SALESMAN  D30_SAL          6160
PRESIDENT D10_SAL          5500
MANAGER   D10_SAL          2695
MANAGER   D20_SAL        3272.5
MANAGER   D30_SAL          3135
ANALYST   D20_SAL          6600
```

解决方案

您可以在Polardb-O中使用Crosstab函数接口进行行列转换。

示例

转行示例如下：

```
with a as (  -- A对应原始数据（即需要列转行的数据）
select
  js->>'seller' as seller,
  js->>'se_year' as se_year,
  jan  ,
  feb  ,
  mar  ,
  apr  ,
  may  ,
  jun  ,
  jul  ,
  aug  ,
  sep  ,
  oct  ,
  nov  ,
  dec
from crosstab(
  -- 这个是需要进行行列变换的数据源。
  -- 排序字段为group by字段，最后一个字段为转换后的内容字段，导数第二个字段为行列变换的字段（内容为枚举，比如月份）
  --  （必须在下一个参数中提取出对应的所有枚举值）
  $$select jsonb_build_object('seller', seller, 'se_year', se_year) as js, se_month, sum(se_amount) from tbl_sellers_info group by 1,2 order by 1$$,
  -- 行列转换的行，有哪些值被提取出来作为列。 这个在这里代表的是月份，也就是se_month的值
  -- 或(select * from (values('jan'),...('dec')) t(se_month))
  'select distinct se_month from tbl_sellers_info order by 1'
)
as   -- crosstab 输出格式
(  js jsonb,  -- 第一个参数SQL内对应的order by对应的字段（1个或多个）
   Jan numeric,  -- 第一个参数SQL内对应导数第二个字段的枚举值，（行转列）
   feb numeric,  -- ...同上
   mar numeric,
   apr numeric,
   may numeric,
   jun numeric,
   jul numeric,
   aug numeric,
   sep numeric,
   oct numeric,
   nov numeric,
   dec numeric
)
order by 1,2
)
,
-- b ， 用jsonb把多列合并为一列，并使用jsonb_each展开。
b as (select seller, se_year, jsonb_each(row_to_json(a)::jsonb-'seller'::text-'se_year'::text) as rec  from a)
select seller, se_year, (b.rec).key as month, (b.rec).value as sum from b;
```

7.21. DBMS_XMLGEN函数

本文介绍不同数据库下DBMS_XMLGEN函数的使用场景。

背景说明

Oracle支持DBDBMS_XMLGEN包及其函数对xml类型数据进行操作。

```
SQL> SELECT dbms_xmlgen.newcontext('select * from dbmgr.xmldemo') FROM dual;
DBMS_XMLGEN.NEWCONTEXT('SELECT*FROMDBMGR.XMLDEMO')
--------------------------------------------------
                                                 1
SQL> select dbms_xmlgen.getxml(1) from dual;
DBMS_XMLGEN.GETXML(1)
--------------------------------------------------------------------------------
<?xml version="1.0"?>
<ROWSET>
 <ROW>
  <A>10</A>
  <B>first line</B>
 </ROW>
 <
```

解决方案

虽然PolarDB O引擎不支持DBMS_XMLGEN包及其函数，但是PolarDB有提供XML相关函数，如xmlagg, xmlroot, xmlforest, xmlelement, xmlconcat，xmlcomment，您可以使用这些函数来实现类似功能。

```
 van=> SELECT xmlforest(a AS A, b AS B) from xmldemo;
         xmlforest
----------------------------
<a>10</a><b>first line</b>
<a>20</a><b>line 2</b>
(2 rows)
```

更多方法请参见https://www.postgresql.org/docs/11/functions-xml.html。

7.22. 不支持DBMS_METADATA.GET_DDL

本文介绍针对不同数据库对应的建表语句的获取方法。

背景说明

Oracle可以通过DBMS_METADATA.GET_DDL函数获取表的建表语句，但是PolarDB O引擎目前不支持。

解决方案

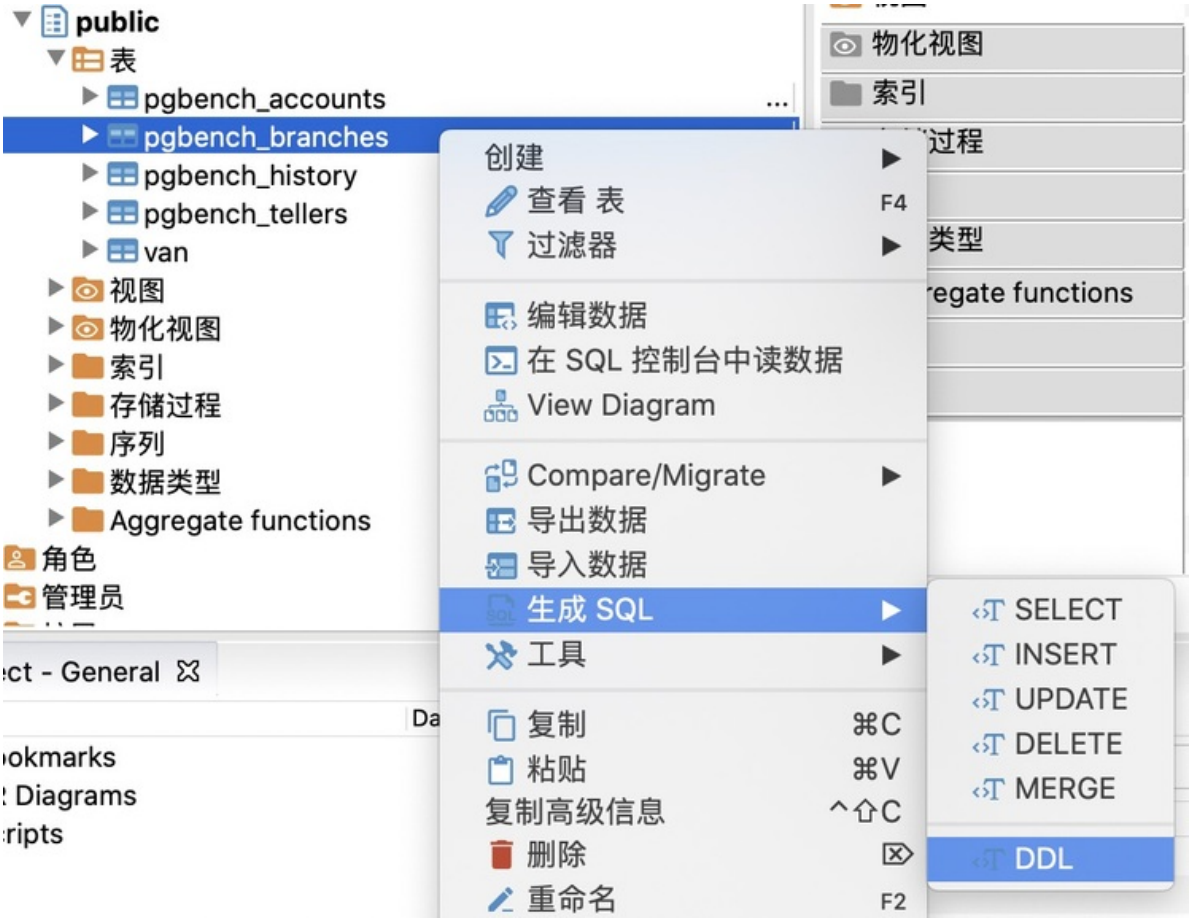
```
an-> \d+ t1
                                          Table "public.t1"
 Column |         Type          | Collation | Nullable | Default | Storage  | Stats targe
t | Description
 --------+-----------------------+-----------+----------+---------+----------+-----------
---+-------------
id     | integer               |           |          |         | plain    |
 |
 name   | character varying(30) |           |          |         | extended |
 |
Indexes:
    "idx1" UNIQUE, btree (id)
    "idx2" btree (name)
Check constraints:
    "con1" CHECK (id < 2000000)
Access method: heap
```

> **说明** `\d+` 可以看到表结构，但不是建表语句。

- 可以通过创建函数来实现：

```
psql=#create extension plperlu;
postgres=# CREATE OR REPLACE FUNCTION GET_DDL(text) RETURNS text
AS 'my $cmd=shift; return `cd /tmp;$cmd`;' LANGUAGE plperlu;
CREATE FUNCTION
postgres=# select GET_DDL('pg_dump -s -t t1 ddl | egrep -v "^--|^$"');
                  get_ddl
-------------------------------------------------------------------
CREATE TABLE public.t1 (                                           +
    id integer,                                                    +
   name character varying(30),                                     +
   CONSTRAINT con1 CHECK ((id < 2000000))                          +
);                                                                 +
ALTER TABLE public.t1 OWNER TO postgres;                      +
CREATE UNIQUE INDEX idx1 ON public.t1 USING btree (id);      +
CREATE INDEX idx2 ON public.t1 USING btree (name);             +
```

- 您可以在客户端管理工具比如pgadmin，dbeaver查看，以dbeaver为例：



7.23. Date - Date结果不兼容

本文介绍Date类型间做减法的结果情况。

背景说明

Oracle中Date与Date类型相减，结果为浮点类型，在PolarDB O引擎中，结果为Interval类型。这种类型差异一般会引起业务SQL中时间计算部分发生语法错误。

解决方案

在PolarDB O引擎中对Date类型减法做简单的语法改造即可适配，改造思路是使用extract函数将Interval类型转换为浮点型，使结果与Oracle保持一致。

extract(*field* from timestamp)	double precision	Get subfield; see Section 9.9.1	extract(hour from timestamp '2001-02-16 20:38:40')	20
extract(*field* from interval)	double precision	Get subfield; see Section 9.9.1	extract(month from interval '2 years 3 months')	3

详情请参见https://www.postgresql.org/docs/11/functions-datetime.html。

示例

以下以使用函数改造为例：

```
CREATE OR REPLACE FUNCTION time_between(TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH TIME ZONE)

RETURNS FLOAT8 AS
$m$
  SELECT EXTRACT(EPOCH FROM $1-$2)/86400;
$m$ LANGUAGE SQL STRICT IMMUTABLE;
-- select sysdate - date '2020-06-28' from dual;
-- 改造为
select time_between(sysdate, date '2020-06-28');
1.29990540226852
```

7.24. Forall and Bulk Collect

本文介绍FORALL语句的使用场景和方法。

背景说明

PolarDB O引擎不支持除"FORALL index IN lower_bound .. upper_bound"类型以外FORALL语句。

原理

在Oracle PL/SQL过程语言handler和SQL之间需要切换，如果是一个较大的LOOP，切换一多，性能就会下降严重。

因此对于在PL/SQL需要多次调用SQL的处理场景，Oracle想到了bulk collect的处理方法。比如用户提交一个数组，要求PL/SQL将这个数组的元素一条条插入到表里面，或者拿来更新表里面的值，又或是删除表里的值。

解决方案

类似Oracle FORALL的批量插入用法，用一个数组表示条件，另一个数组表示VALUE如果有多个条件或者value时，可以用record数组或者hstore（Key-Value类型）数组来表示。

```
CREATE OR REPLACE FUNCTION public.f_bulk_insert1(i_k integer[], i_v text[])
 RETURNS void
 LANGUAGE plpgsql
 STRICT
AS $function$
declare
  i_length int := array_length(i_k,1);
  s timestamp;
  e timestamp;
begin
  s := clock_timestamp();
  raise notice 'start: %', s;
  insert into test select i_k[i], i_v[i] from generate_series(1, i_length) t(i);
  e := clock_timestamp();
  raise notice 'end: %, %', e, e-s;
end;
$function$;
```

7.25. Interval分区

本文介绍Interval的分区说明。

背景说明

Oracle 11g的范围分区表中新增的Interval分区特性，此种范围分区不需要定义MAXVALUE，Oracle会根据分区定义的步长来动态的分配新分区来容纳超过范围的数据。

```
create table BIGTABLE_LOG
(
record_date DATE,
col_1 VARCHAR2(2000),
col_2 VARCHAR2(2000)
)
PARTITION BY RANGE (record_date)
INTERVAL (numtodsinterval(1,'day'))
( PARTITION P1 VALUES LESS THAN (TO_DATE('2014-1-1', 'YYYY-MM-DD')));
```

```
SQL> insert into BIGTABLE_LOG values (to_date('2013-1-1','YYYY-MM-DD'),'','');
1 row created.
SQL> insert into BIGTABLE_LOG values (to_date('2014-1-1','YYYY-MM-DD'),'','');
1 row created.
SQL> insert into BIGTABLE_LOG values (to_date('2014-1-2','YYYY-MM-DD'),'','');
1 row created.
```

```
SQL> select * from BIGTABLE_LOG partition (P1);
RECORD_DATE  COL_1                COL_2
------------ -------------------- --------------------
01-JAN-13
SQL> select * from BIGTABLE_LOG partition (SYS_P24);
RECORD_DATE  COL_1                COL_2
------------ -------------------- --------------------
01-JAN-14
SQL> select * from BIGTABLE_LOG partition (SYS_P25);
RECORD_DATE  COL_1                COL_2
------------ -------------------- --------------------
02-JAN-14
```

解决方案

- PolarDB O引擎目前支持分区类型为列表分区和范围分区（但不支持Interval分区），但可以将Interval分区按步长转化为范围分区。然后通过job定期提前创建相应的分区表。

```
create table BIGTABLE_LOG
(
record_date DATE,
col_1 VARCHAR2(2000),
col_2 VARCHAR2(2000)
)
PARTITION BY RANGE (record_date)
( PARTITION P1 VALUES LESS THAN (TO_DATE('2014-1-1', 'YYYY-MM-DD')),
PARTITION P2 VALUES LESS THAN (TO_DATE('2014-1-2', 'YYYY-MM-DD')),
PARTITION P3 VALUES LESS THAN (TO_DATE('2014-1-3', 'YYYY-MM-DD')),
);
van=> insert into BIGTABLE_LOG values (to_date('2013-1-1','YYYY-MM-DD'),'','');
INSERT 0 1
van=> insert into BIGTABLE_LOG values (to_date('2014-1-1','YYYY-MM-DD'),'','');
INSERT 0 1
van=> insert into BIGTABLE_LOG values (to_date('2014-1-2','YYYY-MM-DD'),'','');
INSERT 0 1
van=> select * from bigtable_log_p1;
record_date     | col_1 | col_2
--------------------+-------+-------
 01-JAN-13 00:00:00 |       |
(1 row)
van=> select * from bigtable_log_p2;
    record_date     | col_1 | col_2
--------------------+-------+-------
01-JAN-14 00:00:00 |       |
(1 row)
van=> select * from bigtable_log_p3;
   record_date     | col_1 | col_2
 --------------------+-------+-------
02-JAN-14 00:00:00 |       |
 (1 row)
```

- 定义定期创建分区的job函数

```
CREATE or replace FUNCTION add_partitions(tablename text,
lessdate text,partitionname text)RETURNS text AS $$
DECLARE results text;
DECLARE sql     text;
BEGIN
   results :='OK';
   sql='ALTER TABLE '|| tablename ||' ADD PARTITION '|| partitionname ||' VALUES LESS THA
N
 (TO_DATE('''||lessdate||''', ''YYYY-MM-DD''))';
   execute sql;
   RETURN results;
END;
$$
  LANGUAGE plpgsql;
```

- 添加分区表成功

```
van=> select add_partitions('bigtable_log','2014-1-4','P4');
 add_partitions
 ----------------
 OK
(1 row)
 van=> \d+ bigtable_log_p4
                                          Table "public.bigtable_log_p4"
   Column    |            Type             | Collation | Nullable | Default | Storage  |
Stats target | Description
 ------------+-----------------------------+-----------+----------+---------+----------+
-------------+-------------
 record_date | timestamp without time zone |           |          |         | plain    |
             |
 col_1       | character varying(2000)     |           |          |         | extended |
             |
 col_2       | character varying(2000)     |           |          |         | extended |
             |
Partition of: bigtable_log FOR VALUES FROM ('03-JAN-14 00:00:00') TO ('04-JAN-14 00:00:00
')
Partition constraint: ((record_date IS NOT NULL) AND (record_date >= '03-JAN-14 00:00:00'
::timestamp without
time zone) AND (record_date < '04-JAN-14 00:00:00'::timestamp without time zone))
van=>
```

> **说明** poarDB-O支持创建job，可以定期自动提前创建相应的分区。

8.迁移实验室

8.1. SQL Adapter

ADAM SQL Adapter是基于PostgreSQL通迅协议的SQL转发代理服务。本文介绍如何使用SQL Adapter实现不兼容SQL的转换。

功能介绍

SQL Adapter目前支持Oracle到PolarDB O或PolarDB PostgreSQL的SQL改造。主要功能有：

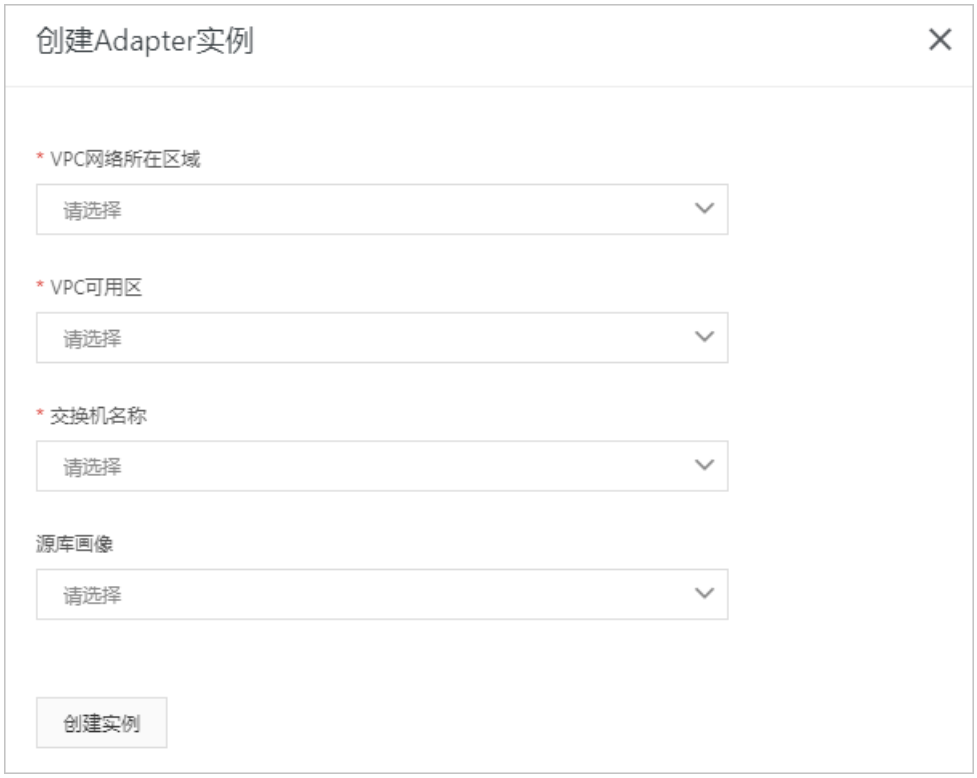
- 实时转换从Oracle迁移到PolarDB O或PolarDB PostgreSQL不完全兼容的SQL。
- 异步记录所有需要改造的SQL。
- 您可以对无法自动转换的SQL进行自定义修改。

前提条件

- 已完成结构迁移。
- 您的应用需要部署在VPC环境内，SQL Adapter功能暂不支持通过公网使用。
- Adapter实例当前仅支持杭州和北京地域，如果您需要使用其它地域，请提交工单。
- 已开通AliyunAdamAccessingDatabaseRole角色。

操作步骤

1. 申请ADAM SQL Adapter权限。
   i. 登录数据库与应用迁移控制台。
   ii. 单击左侧导航栏中的**迁移实验室**。
   iii. 单击**ADAM SQL Adapter**下的**申请**按钮。
   iv. 在申请页面填写公司名称、联系电话、申请原因等信息，单击**提交**，完成申请。
2. 创建Adapter实例。
   i. 登录数据库与应用迁移控制台。
   ii. 单击左侧导航栏中的**迁移实验室**。
   iii. 单击**ADAM SQL Adapter**下的**详情**按钮，进入**ADAM Adapter**页面。
   iv. 单击左上角**申请adapter实例**按钮。
   v. 填写参数，创建Adapter实例。



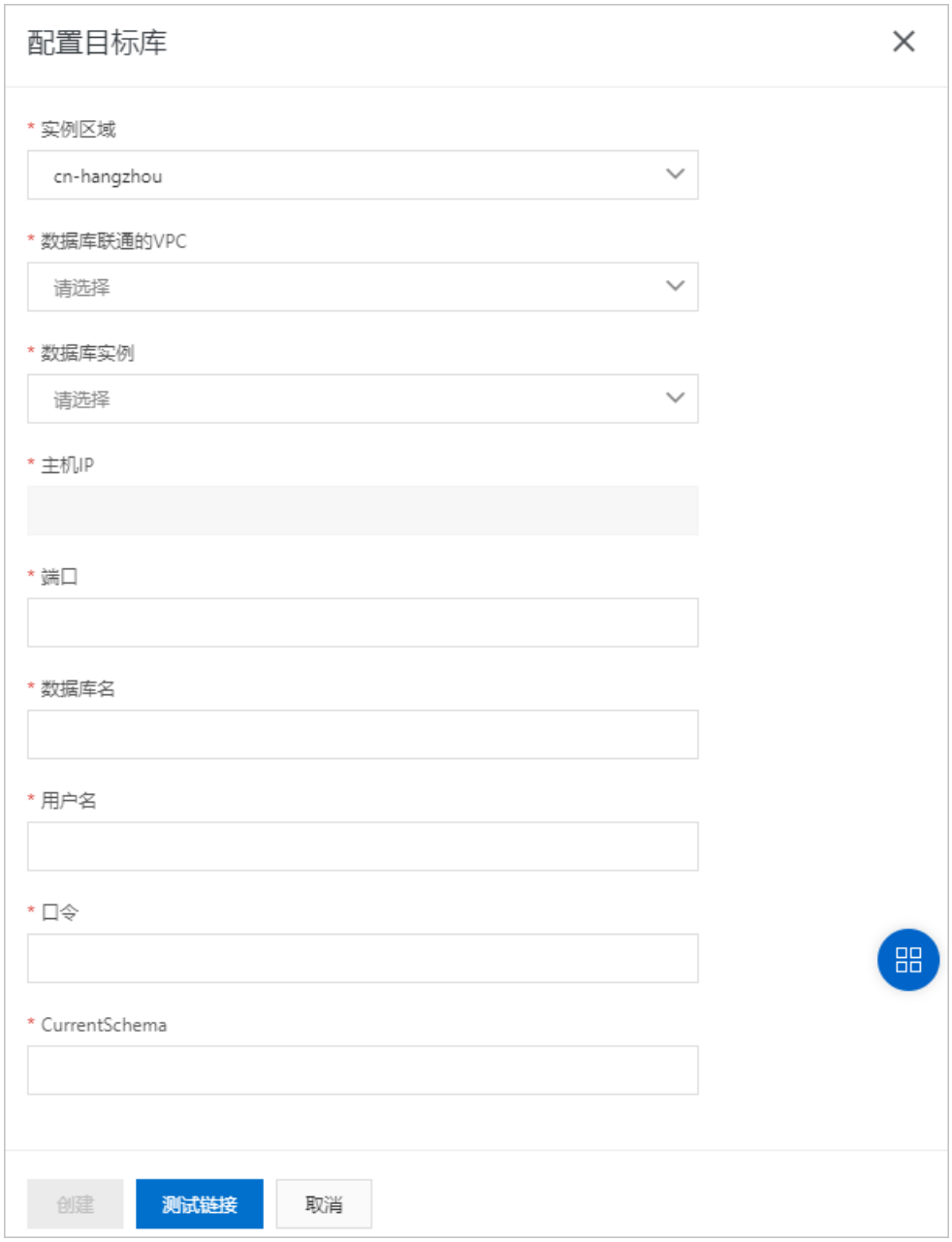
参数	取值及含义
VPC网络所在区域	表示创建Adapter实例的VPC所在地域。 取值范围： ■ cn-hangzhou ■ cn-beijing **说明** Adapter实例当前仅支持杭州和北京地域，如果您需要使用其它地域，请提交工单。
VPC可用区	表示创建Adapter实例的VPC所在可用区。 取值范围： ■ 如果地域选择cn-hangzhou，可用区为cn-hangzhou-b。 ■ 如果地域选择cn-beijing，可用区为cn-beijing-b。 **说明** Adapter当前仅支持B可用区。
交换机名称	表示所选VPC地域及可用区下已创建的交换机。如无可用交换机，请前往专有网络管理控制台创建，具体方法，请参见使用交换机。
源库画像	选择源库画像。如果没有源库画像，请参见数据库画像。

vi. 单击**创建实例**按钮。

3. 配置目标库。

i. 在ADAM Adapter页面单击**配置目标库**。

ii. 单击左上角**配置目标库**按钮，填写目标库相关参数。

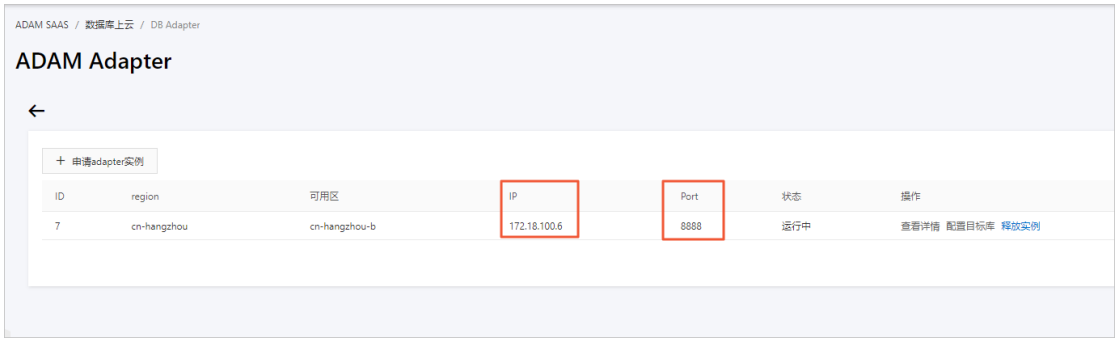


参数	取值及含义
实例区域	不可修改，与创建Adapter实例时取值相同。
数据库连通的VPC	选择目标PolarDB实例的VPC。
数据库实例	选择目标PolarDB实例的实例ID。

参数	取值及含义
主机IP	不可修改，无需配置，自动获取。
端口	无需配置，自动获取。
数据库名	填写PolarDB实例中的目标数据库名称。
用户名	填写目标数据库的用户名。
口令	填写目标数据库的密码。
CurrentSchema	填写当前用户对应的Schema。

iii. 单击**测试链接**按钮。在提示**测试连接成功**后，单击**创建**按钮。

4. 获取ADAM Adapter实例的IP和端口。
   i. 单击左侧导航栏中的**迁移实验室**。
   ii. 单击**ADAM SQL Adapter**下的**详情**按钮，进入**ADAM Adapter**页面。
   iii. 查看ADAM Adapter实例的IP和端口。



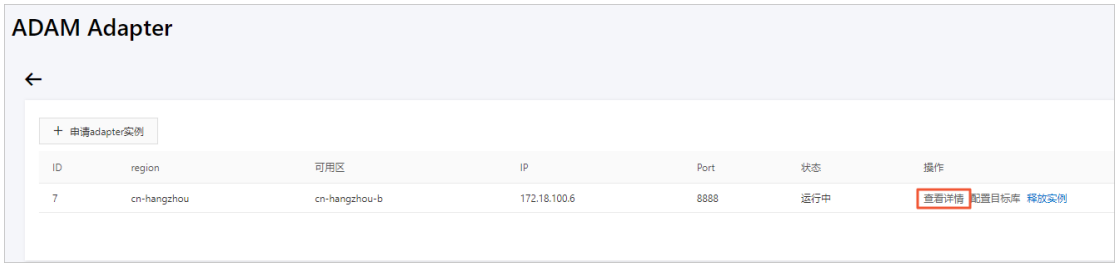
5. 修改应用连接数据库的URL。

```
jdbc:polardb://172.18.100.6:8888/polardb_test
```

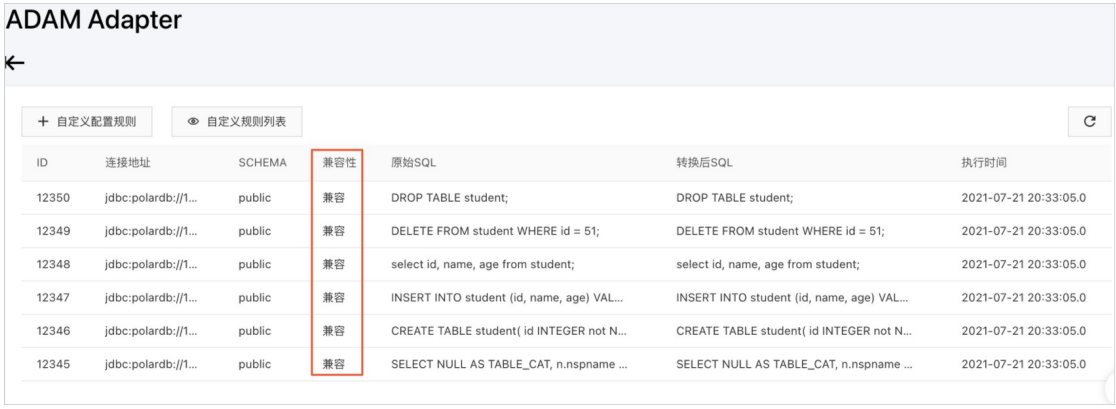
参数	示例	说明
URL前缀	`jdbc:polardb://`	URL统一使用 `jdbc:polardb://` 作为前缀
连接地址（IP）	`172.18.100.6`	ADAM Adapter实例的IP。获取方法请参见获取ADAM Adapter实例的IP和端口。。
端口（Port）	`8888`	ADAM Adapter实例的端口。获取方法请参见获取ADAM Adapter实例的IP和端口。。
数据库名称	`polardb_test`	连接的数据库名称。

> **说明** 用户名及密码无需修改。

6. 查看ADAM Adapter对SQL的兼容性以及SQL转换情况。
   i. 在**ADAM Adapter**页面单击**查看详情**。



   ii. 查看ADAM Adapter对SQL的兼容性以及SQL转换情况。



7. 配置自定义SQL转换规则

   对于ADAM Adapter不兼容无法转换的SQL，请单击左上角**自定义配置规则**。

参数	取值及说明
自定义规则类型	表示自定义规则转换的规则类型。 取值范围： ○ 文本替换。 ○ 正则替换。 **说明** 如果使用正则替换，请谨慎配置正则规则，恶意的正则规则可能导致Adapter实例资源耗尽，目前Adapter暂不提供高可用保证。
匹配文本	填写待匹配替换的文本或正则表达式。
替换文本	填写替换的文本或正则表达式。
规则生效范围	表示配置的替换规则生效的范围。 取值范围： ○ 全局生效。 ○ 指定SQL替换。
不兼容SQL选择	该参数仅当**规则生效范围**选择为**指定SQL替换**时出现。填写替换规则生效的指定SQL。

8.2. SQL周期性采集

本文介绍Oracle数据库周期性采集SQL信息以及数据分析的方法。

限制条件

仅支持Oracle 10g、11g、12c版本。

使用流程

1. 采集SQL信息。

   说明 由于周期性采集目前只在离线采集器中支持，线上采集暂未支持。您需要先自行采集Oracle数据库的SQL信息，详情操作请参见采集信息。

2. 分析SQL信息。

   说明 通过创建**周期性采集项目**，上传数据文件，系统则会为您进行SQL信息自动合并与分析。详情操作请参见分析信息 。

采集信息

1. 下载采集器。
   i. 登录数据库与应用迁移控制台。
   ii. 单击左侧导航栏的**迁移实验室**，单击右侧**SQL周期性采集**下方的**下载**。

iii. 在**数据库采集器下载**页面单击下方的**下载**按钮。

iv. 选择数据库采集器的版本，并同意协议，完成数据库采集器的下载。

> **说明**
>
> - 数据库采集器运行环境要求：
>   - 数据库采集器所在的机器网络可连接Oracle源数据库即可，无需安装在源数据库物理机上。
>   - 内存需达到6 GB以上，同时硬盘空间需满足500 GB以上。
> - 源库结构采集时，对收集结果数据自动进行参数脱敏。
> - 源库结构采集时，自动监测目标库负载。当负载超过系统设定阈值时，自动暂停采集；当负载恢复后继续采集。

2. 登录Oracle源数据库，使用具有SYSDBA权限的账号创建数据采集的临时账号并配置权限。

> **说明**
>
> - 若您已有包含下面权限的账号，请忽略此步骤，直接使用。
> - 以下权限用于连接Oracle数据库，收集相关信息并对结果数据脱敏导出。

- Oracle 10g、11g、12c（非CDB模式，创建LOCAL USER类型用户）

```
--创建采集用户eoa_user，并设置密码为eoaPASSW0RD
create user eoa_user identified by "eoaPASSW0RD" default tablespace users;
--授予查询权限
grant connect,resource,select_catalog_role,select any dictionary to eoa_user;
--授予DBMS_LOGMNR权限
--版本为Oracle 10g的数据库需要先执行如下语句：
CREATE OR REPLACE PUBLIC SYNONYM dbms_logmnr FOR sys.dbms_logmnr;
grant execute on DBMS_LOGMNR to eoa_user;
--授予DBMS_METADATA权限
grant execute on dbms_metadata to eoa_user;
--授予查询事务权限
grant select any transaction to eoa_user;
--授予分析表权限
grant analyze any to eoa_user;
--授予产生随机编号权限
grant execute on dbms_random to eoa_user;
```

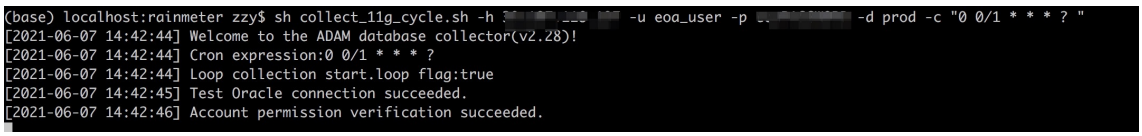
- Oracle 12c（CDB模式，需要连接到CDB，创建COMMON USER类型用户）

```
create user c##eoa_user identified by "eoaPASSW0RD" default tablespace users;
grant connect,resource,select_catalog_role,select any dictionary to c##eoa_user conta
iner=all;
grant execute on DBMS_LOGMNR to c##eoa_user container=all;
grant execute on dbms_metadata to c##eoa_user container=all;
grant select any table to c##eoa_user container=all;
grant select any transaction to c##eoa_user container=all;
grant analyze any to c##eoa_user container=all;
grant execute on dbms_random to c##eoa_user container=all;
alter user c##eoa_user set container_data=all container=current;
```

3. 启动采集（在Windows环境下执行.bat；在Linux 环境下执行.sh，本文以Linux环境介绍如何进行数据采集）。
   - Oracle 10g、11g

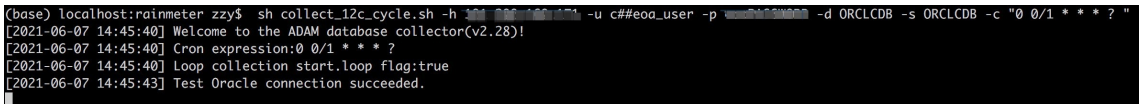
```
sh collect_10g_cycle.sh -h <ip> -u <username> -p <password> -d <service_name> -c <cro
n>
```

```
sh collect_11g_cycle.sh -h <ip> -u <username> -p <password> -d <service_name> -c <cro
n>
```



   - Oracle 12c

```
sh collect_12c_cycle.sh -h <ip> -u <username> -p <password> -d <service_name> -s <sid
> -c <cron>
```



> **说明** 上述命令各参数解释如下：
>
> - -h（必选）：采集数据库的IP地址。
> - -u（必选）：采集用户。
> - -p（必选）：采集用户的密码。
> - -d（必选）： 采集数据库的服务名。
> - -s（仅Oracle 12c需要提供此参数）：采集数据库实例名。
> - -c（必选）：cron表达式，用来指定采集的周期性。格式为：`秒分时 日月周[年]`。

周期性采集会根据cron表达式配置的周期持续进行，为了帮助您理解，下面介绍一些常用的cron表达式示例。同时您也可以随时可以停止采集（kill进程号）。

表达式	说明
0 */1 * * * ?	每隔1分钟触发一次。
0 0 5-15 * * ?	每天5:00~15:00整点触发。
0 0/3 * * * ?	每隔3分钟触发一次。
0 0-5 14 * * ?	每天14:00~14:05期间每隔1分钟触发一次。
0 0/5 14 * * ?	每天14:00~14:55期间每隔5分钟触发一次。
0 0/5 14,18 * * ?	每天14:00~14:55和18:00~18:55两个时间段内每5分钟触发一次。
0 0/30 9-17 * * ?	每天9:00~17:00内每半小时触发一次。
0 0 10,14,16 * * ?	每天10:00、14:00和16:00触发。
0 0 12 ? * WED	每周三12:00触发。
0 0 17 ? * TUES,THUR,SAT	每周二、周四、周六17:00触发。
0 10,44 14 ? 3 WED	每年3月的每周三的14:10和14:44触发。
0 15 10 ? * MON-FRI	周一至周五的上午10:15触发。
0 0 23 L * ?	每月最后一天23:00触发。
0 15 10 L * ?	每月最后一天10:15触发。
0 15 10 ? * 6L	每月最后一个周五10:15触发。
0 15 10 * * ? 2021	2021年的每天10:15触发。
0 15 10 ? * 6L 2021-2025	2021年~2025年的每月的最后一个周五上午10:15触发。
0 15 10 ? * 6#3	每月的第三个周五10:15触发。

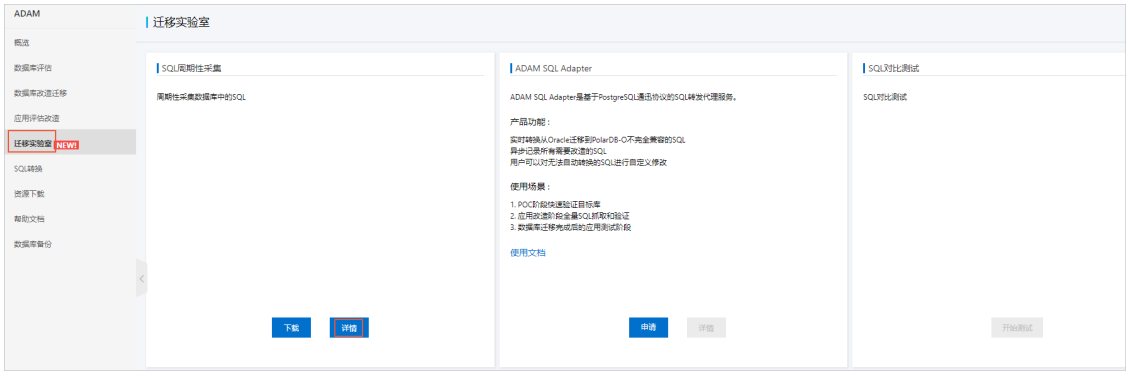
4. 导出采集结果。

   采集结束后，会提示您已生成数据包，日志文件如下：

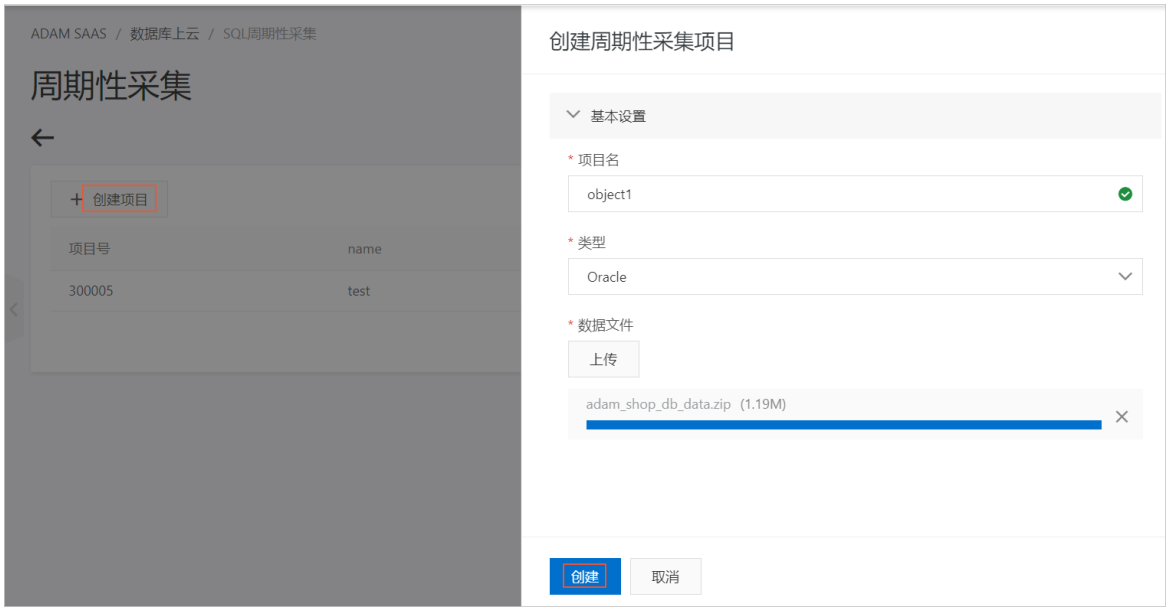
```
[***] ********************************************************************
[***] *     Collect Successfully!
[***] *
[***] * Complete the file packaging, the package result path is:
[***] *        ~rainmeter/out/data.zip ********************************************
******************
```

分析信息

1. 登录数据库与应用迁移控制台。
2. 单击左侧导航栏的**迁移实验室**，单击右侧**SQL周期性采集**下方的**详情**。



3. 单击上方的**创建项目**，填写项目名，数据库类型选择**Oracle**，上传采集到的数据文件，单击**创建**。



4. 单击目标项目操作栏的**详情**，查看采集评估结果。

说明 您可以点击目标SQL的查看按钮查看具体SQL错误信息的分析情况。

9.RAM 子账号

ADAM 支持阿里云主子账号（RAM）访问体系，本文介绍如何授权子账号访问 ADAM 控制台。

前提条件

使用 RAM 子帐号访问 ADAM 控制台，子帐号需要满足以下两个条件：

- 需要为该子账号创建 Access Key ID/Access Secret Key。
- 必须启用该子帐号的控制台登录功能，为该子帐号设置登录用户名、密码。

具体操作请参见创建RAM用户。

背景信息

子账号使用ADAM控制台需要授权。目前ADAM仅支持授予子账号全部控制权限或只读访问权限两种。

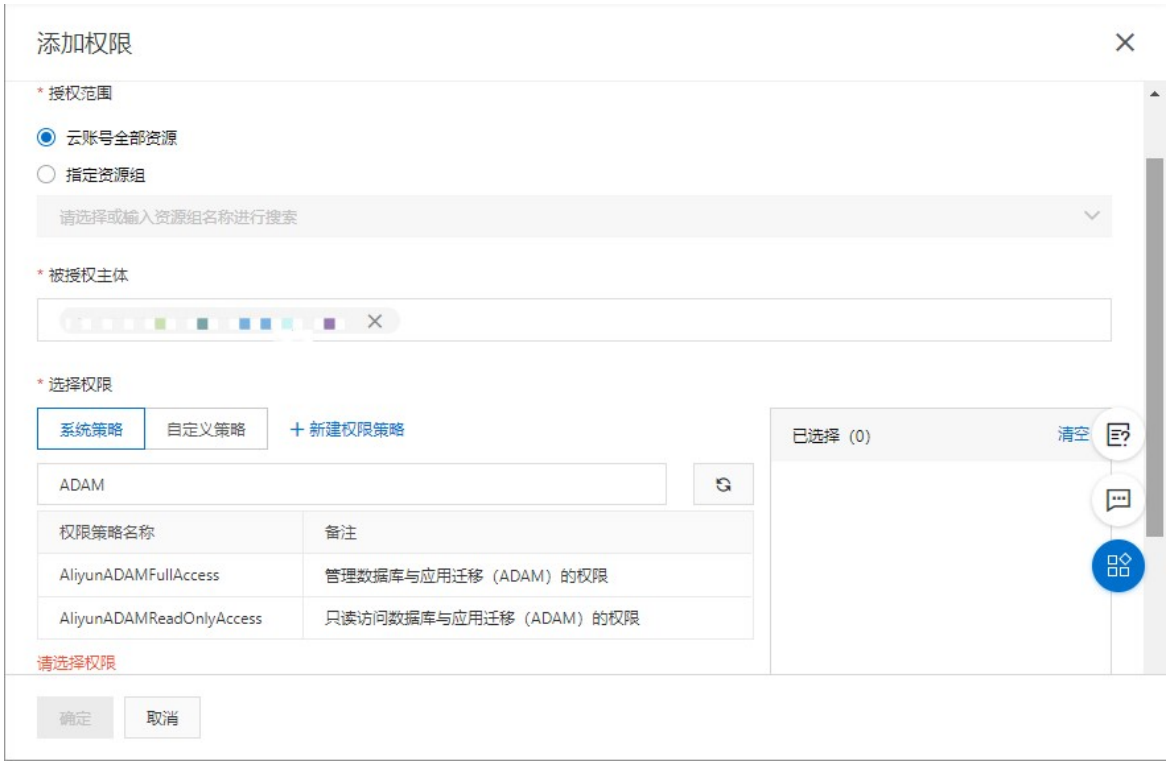
> **说明** ADAM 子账号会继承全部父账号数据，且子账号A产生的数据，同一父账号下的全部子账号也可以看到。

操作步骤

1. 主账号登录RAM 控制台。
2. 单击左侧导航栏的**人员管理 > 用户** 。
3. 在用户列表页面，单击目标子账号右侧操作列中的**添加权限**。
4. 在**添加权限**页面选择**授权范围**、选择对应的**权限策略** 。

   权限策略说明：

   - AliyunADAMFullAccess：管理数据库与应用迁移（ADAM）的权限。
   - AliyunADAMReadOnlyAccess：只读访问数据库与应用迁移（ADAM）的权限。



5. 单击**确定**，授权完成后，即可用子账号登录ADAM 控制台。