ALIBABA CLOUD

Alibaba Cloud

Adavanced Database & Application Migration User Guide

Document Version: 20220412

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

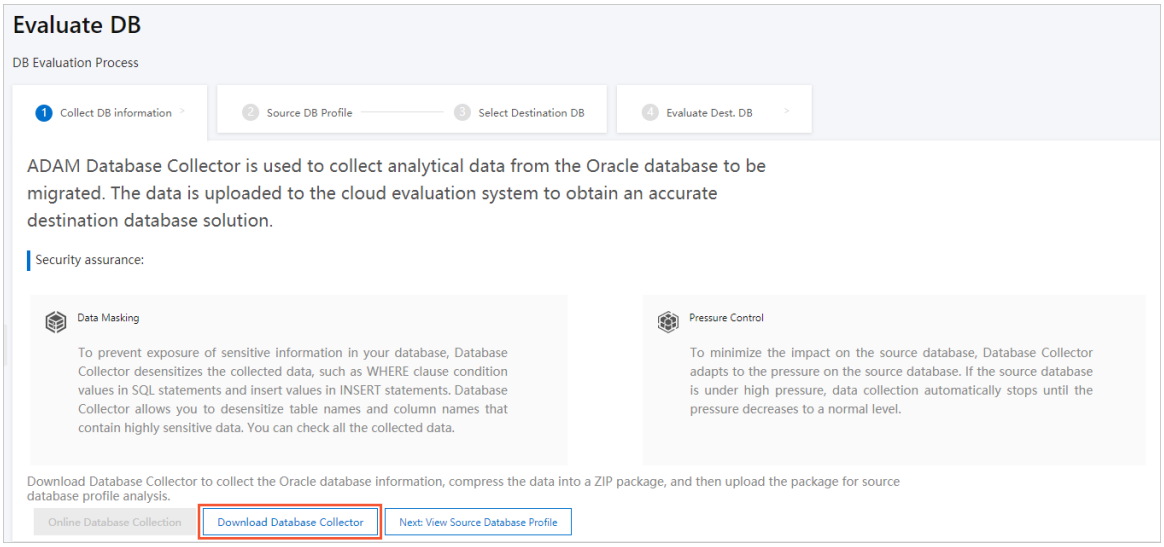
1.Database evaluation

1.1. Collect database information

Download a Database Collector client to collect database information

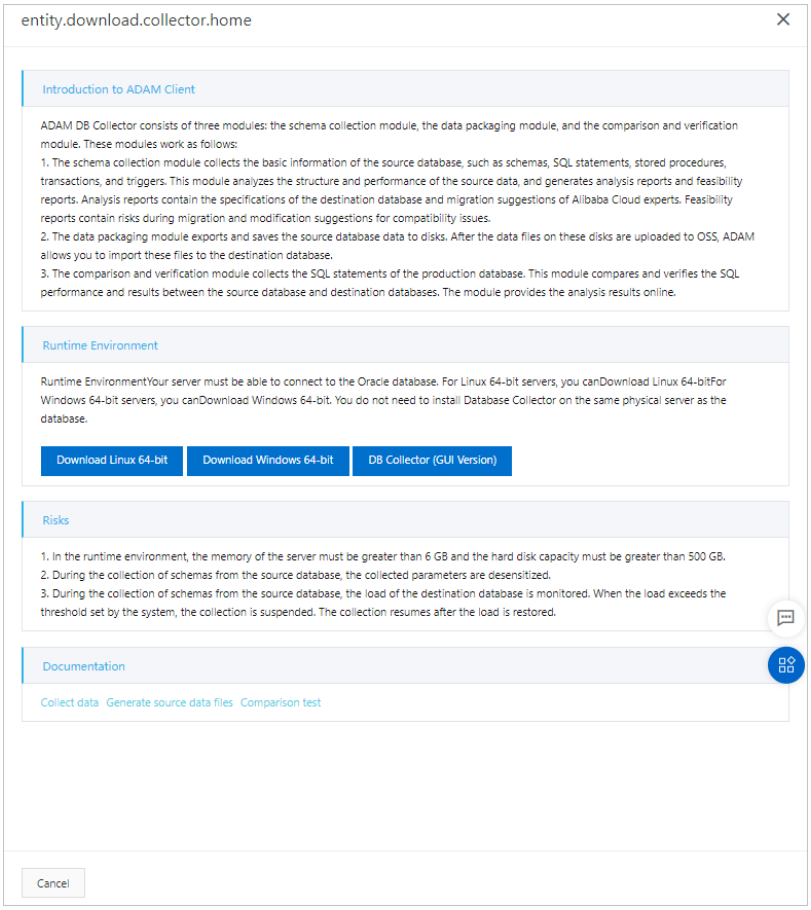
If you cannot access the source database over the Internet or from Alibaba Cloud, you must collect database information offline.

1. Download a Database Collector client
   i. Log on to the ADAM console.
   ii. In the left-side navigation pane, choose **Database Migration** > **Evaluate DB**.
   iii. On the **Collect DB Information** tab, click **Download Database Collector** to go to the Download DB Collector page.



iv. Select a Database Collector version. Download and decompress the corresponding package.

> **Note**
>
> We recommend that you do not run collection tasks on the source database server. The server on which you want to install the Database Collector client must meet the following requirements:
>
> - Network: accessible to the source Oracle database
> - CPU: 2 cores
> - Memory: 8 GB
> - Hard disk: 100 GB free space



v. Create and authorize an account to collect database information.

   a. Log on to the source Oracle database.

b. Create a temporary account by using an account that has the SYSDBA permission, and then grant the following permissions to the temporary account. If you already have an account that has the following permissions, skip this step.

> Note
> The following permissions are used to access the Oracle database. If you have these permissions, you can collect, mask, and export database information.

- **If the source database is an Oracle 10g, 11g, or 12c database, create a local user in a non-CDB architecture.**

  a. Create an account named eoa_user and set the password to eoaPASSW0RD.

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

  b. Grant the SELECT permissions.

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

  c. Grant permissions on the DBMS_LOGMNR package.

  > Note
  > If the source database is an Oracle 10g database, you must first run the following statement:
  >
  > ```
  > create or replace public synonym DBMS_LOGMNR for sys.dbms_logmnr;
  > ```

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

  d. Grant permissions on the DBMS_METADATA package to query DDL statements.

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

  e. Grant permissions to query transactions.

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

  f. Grant permissions to query tables.

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

  g. Grant permissions to analyze tables.

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

  h. Grant permissions to generate random numbers.

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

- **If the source database is an Oracle 12c database, connect to an Oracle 12c container database (CDB) to create a common user.**

```
create user c##eoa_user identified by "eoaPASSW0RD" default tablespace users;
grant connect,resource,select_catalog_role,select any dictionary to c##eoa_user 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;
```

c. Collect structured data to generate feasibility reports and compatibility reports.

The ADAM database collector allows you to collect the information of existing Oracle databases Oracle 10g, 11g, and 12c databases. If you encounter problems during the database collection, submit a ticket and attach the files in the logs directory where the database collector is installed.

a. Run the following command to collect database information. The .bat command is applicable to Database Collector in Windows and the .sh command is applicable to Database Collector in Linux.

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

> **Note**
>
> You can use the `collect_11gR2` script to collect the information of a pluggable database (PDB) in Oracle 12c.

```
collect_12c[.sh|.bat] -h <host> -u <username> -p <password> -P <port> -d <service_name> -s <sid>
```

- -h: specifies the IP address of the Oracle database. - -u: specifies the username as eoa_user. - -p: specifies the password as eoaPASSW0RD. - -P: specifies the port of the Oracle database. Example: 1521. - -d: specifies the service name of the Oracle database. For example, 12c is the service name for a specific PDB. - -s: specifies the name of the Oracle database instance. This option is required only for Oracle 12c.

b. Export the collection results.

After the collection is complete, a data packet is generated and the path of the data packet is displayed. The following script shows the log file:

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

c. After the migration is complete, delete the temporary account.

Use an account that has the SYSDBA permission to access the database in the Database Collector client and execute the following SQL statement:

a. Oracle 10g, 11g, or 12c (non-CDB)

```
drop user eoa_user cascade;
```

b. Oracle 12c (CDB)

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

> **Note**
>
> Database Collector consumes only a small amount of memory in Oracle databases, and does not require high-performance CPUs. In the beginning of a collection task, the CPU usage of your server may increase due to the getDDL operation. The increase rate varies based on the Oracle instance type and the current load. We recommend that you collect database information during off-peak hours to increase the collection speed.

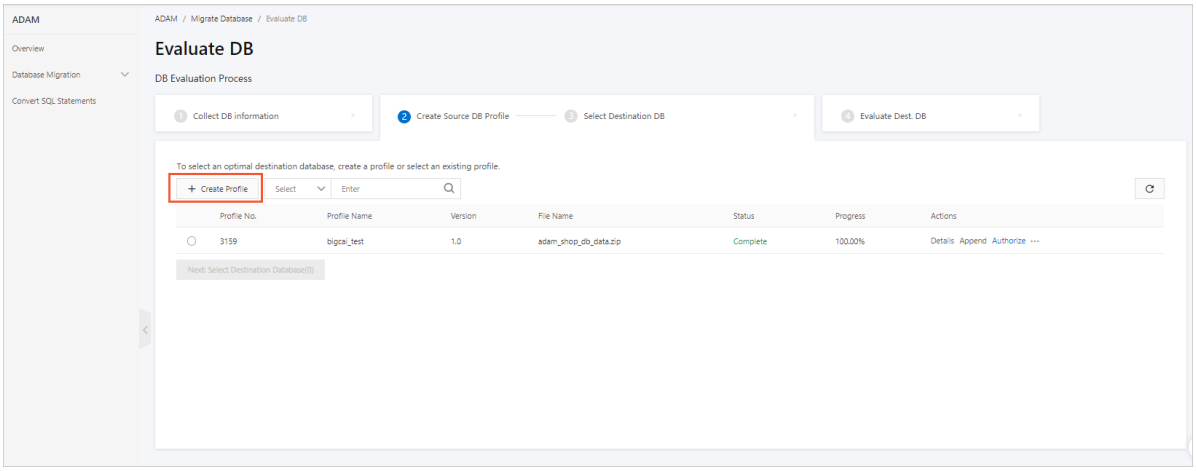
1.2. Manage a database profile

A database profile is used to evaluate a source database,helps you monitor the status of your source Oracle database.When you migrate and transform a source database, you can search for the database based on the database profile.

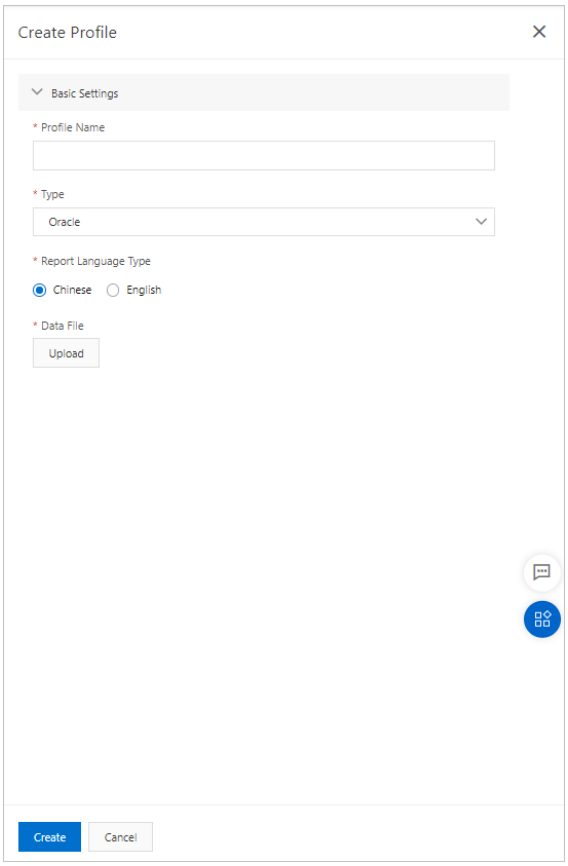
Creae Profile

1. Log on to the ADAM console.
2. In the left-side navigation pane, click **Evaluate DB**.

3. In the upper-left corner of the **Create Source DB Profile** tab, click **Create Profile**.

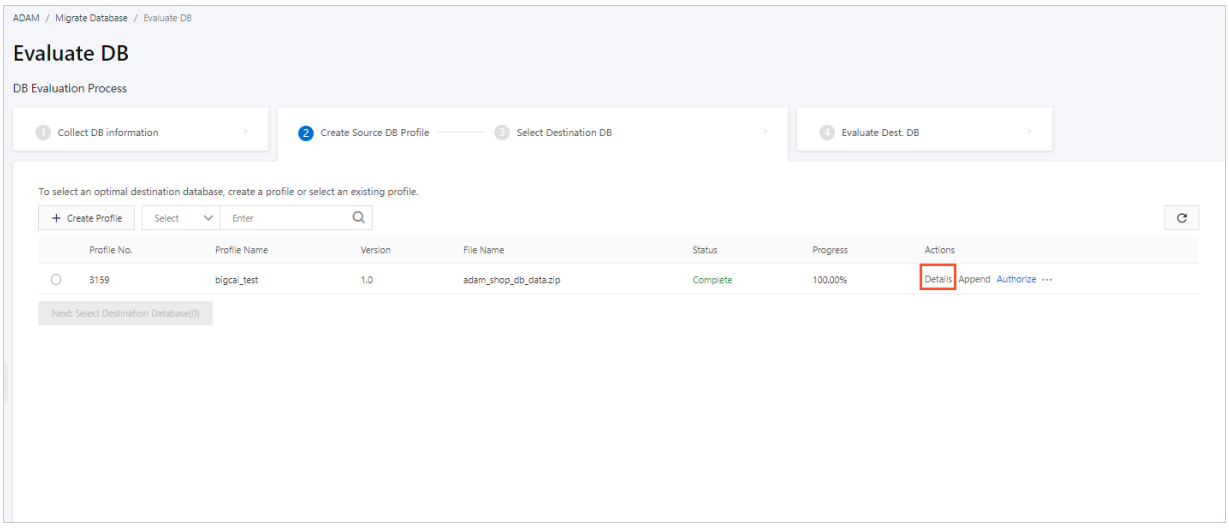


4. In the Create Profile dialog box, specify the profile name and click **Upload** to upload a data file. Then, click **Create**.

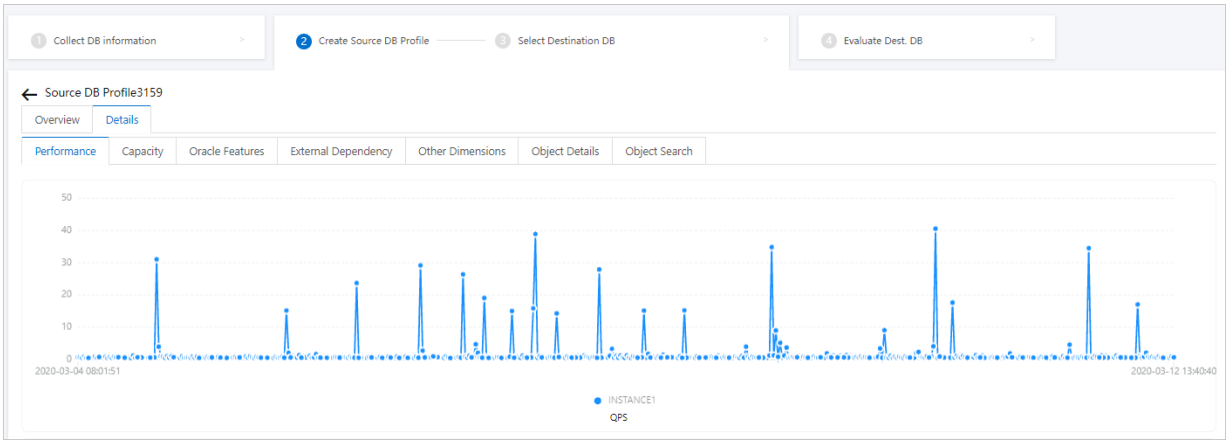


View the profile

After the profile of the source database is analyzed, you can click Details in the Actions column to view the profile details. A database profile is analyzed in 1 to 30 minutes. The required time depends on the collected information of the source database.



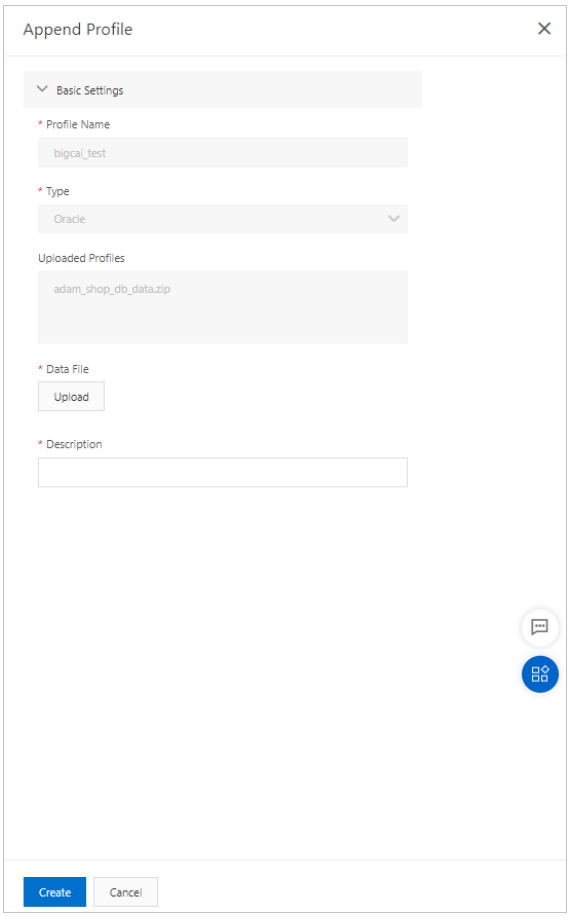
Click **Details** in the Actions column. On the profile details page, you can view the **performance**, **capacity**, **Oracle features**, and **external dependencies** of the source database.



Update a profile

You can create multiple versions for a profile of a source database. If the information of a source database is changed after a profile is created, you can append a new data file to the profile.

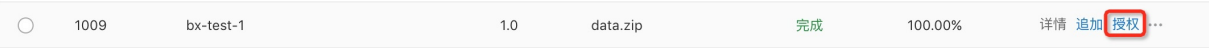
In the profile list, select the profile to be updated. Click **Append** in the Actions list. This way, you can append a new data file to the profile. Advanced Database & Application Migration (ADAM) generates a new version of the profile based on all the data files.



Grant permissions on the profile

ADAM allows you to authorize other users to access your profile. The validity period of the access permissions is one month. Authorized users can view the profile and create new projects based on the profile.

> **Notice**
> Use the authorization feature with caution. You are liable for data disclosures caused by improper authorization.



Revoke permissions on the profile

You can revoke permissions on the profile from other users.

Delete a profile

If you delete a profile, all analysis data of the profile are also deleted. Move the pointer over the More icon in the Actions column and click **Delete**.

Profile details

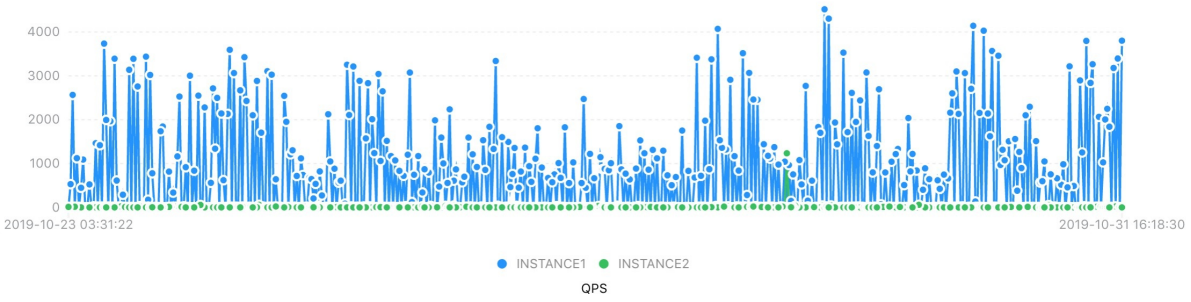
Overview

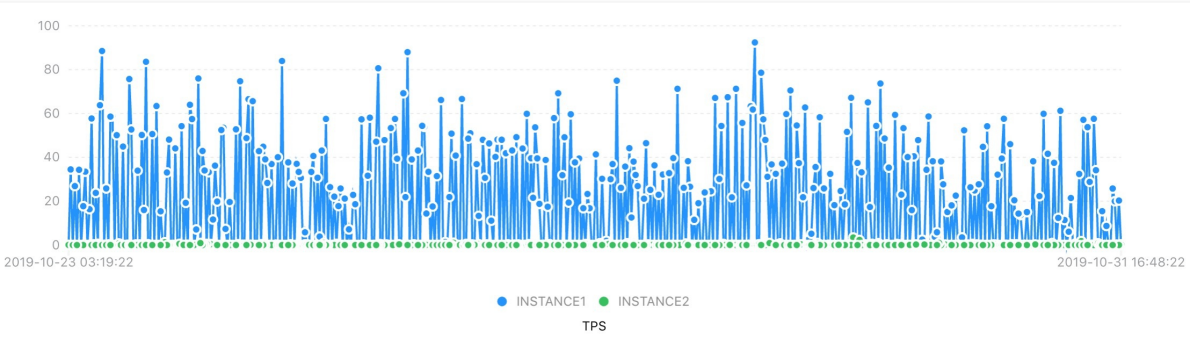
The overview page provides the statistics of a profile by scale, session, risk, hotspot, complexity, and load.

- **Session**: indicates the connection status of the database. The higher the value is, the more sessions are established to connect to the database.
- **Risk**: indicates the risk level of the database. The higher the value is, the more likely the database is exposed to performance risks related to SQL statements or objects.
- **Hotspot**: indicates whether the database has objects that are frequently accessed. The higher the value is, the more objects are frequently accessed.
- **Scale**: indicates the scale of database resources. The higher the value, the larger the database scale.
- **Load**: indicates the running performance of the database. The higher the value, the larger the running load of the database.

Performance

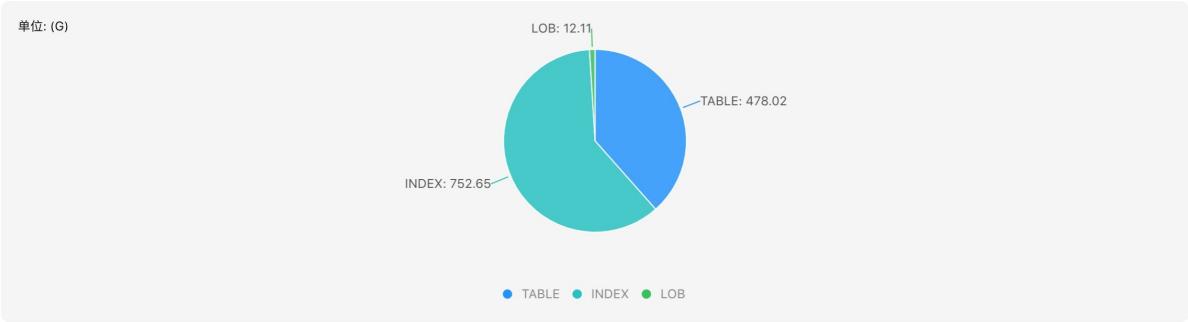
This metric shows the transactions per second (TPS), queries per second (QPS), CPU usage, and load of the database.





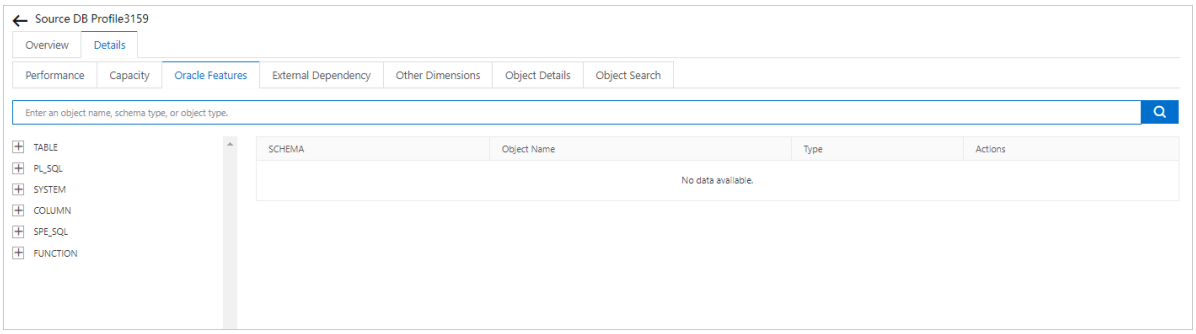
Capacity

This metric shows the capacity ranking of database schemas and the capacity proportion of different object types such as table, index, and large object (LOB).



Oracle features

This metric shows Oracle features in a tree diagram and a table. The tree diagram provides two levels to list all the features collected from the database. If you click a feature, the objects that have the feature are listed on the table. You can search for features by entering a condition in the search box. This shows all Oracle features used by your source database before you migrate the database.

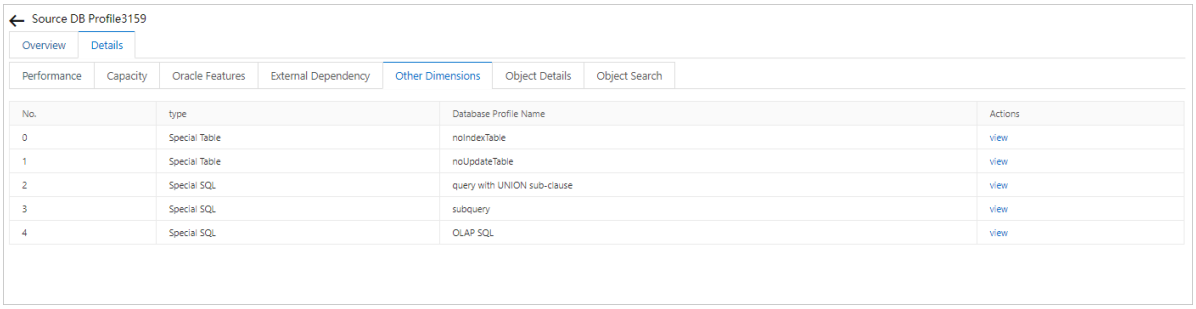


External dependencies

This metric shows database links and link details. You must revise schemas when you transform databases that contain external dependencies.

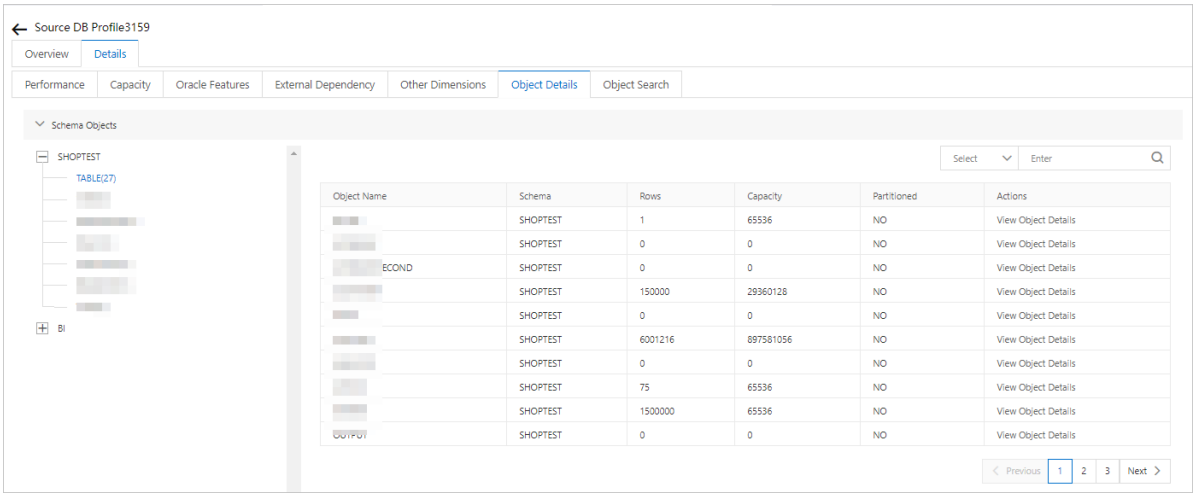
Other dimensions

This metric shows special tables and SQL statements, such as tables without primary keys, high growth tables, and SQL statements with aggregate functions. You can click **View** in the Actions column to view the details.

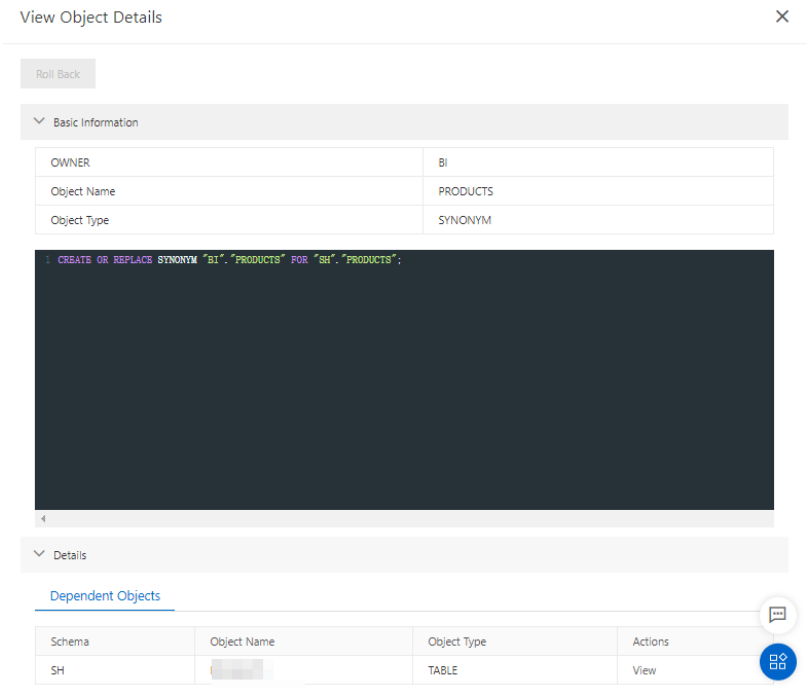


Object details

This metric shows object information from different dimensions, such as object feature tags, relationships such as association and dependency, and Oracle features contained in objects. You can click **View Object Details** in the Actions column to view details.

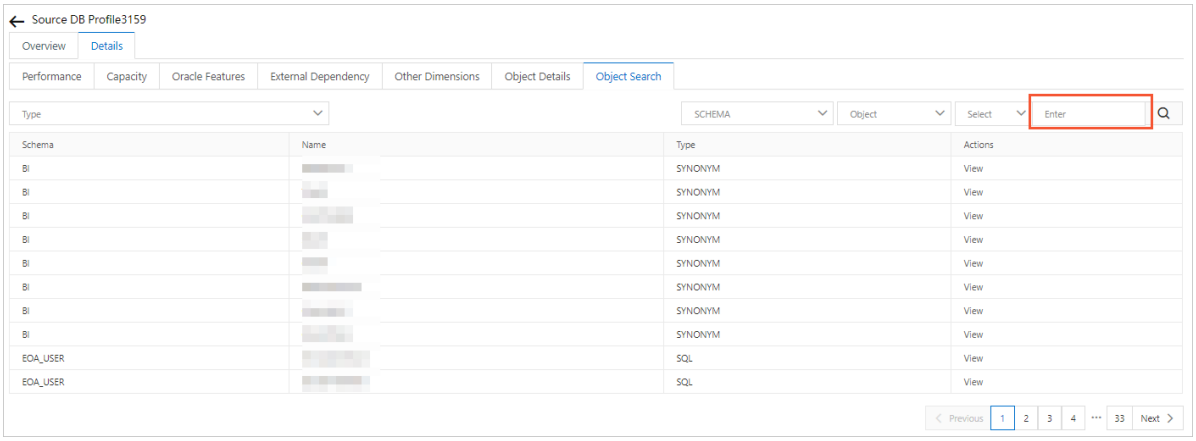


In the Basic Information section of the View Object Details dialog box, you can view the basic information and DDL statements of the object. In the Details section, you can view the analysis data provided by ADAM, such as the features, referenced objects, dependent objects, and Oracle features of the object.



Object search

You can search for objects by schema, DDL, object type, or tag. You can also view the dependencies and tags of objects. Exact matches, fuzzy matches, and type-based matches are supported. You can combine different filter conditions to search for Oracle objects in the profile. The object search feature allows you to search for objects when you transform databases and applications.

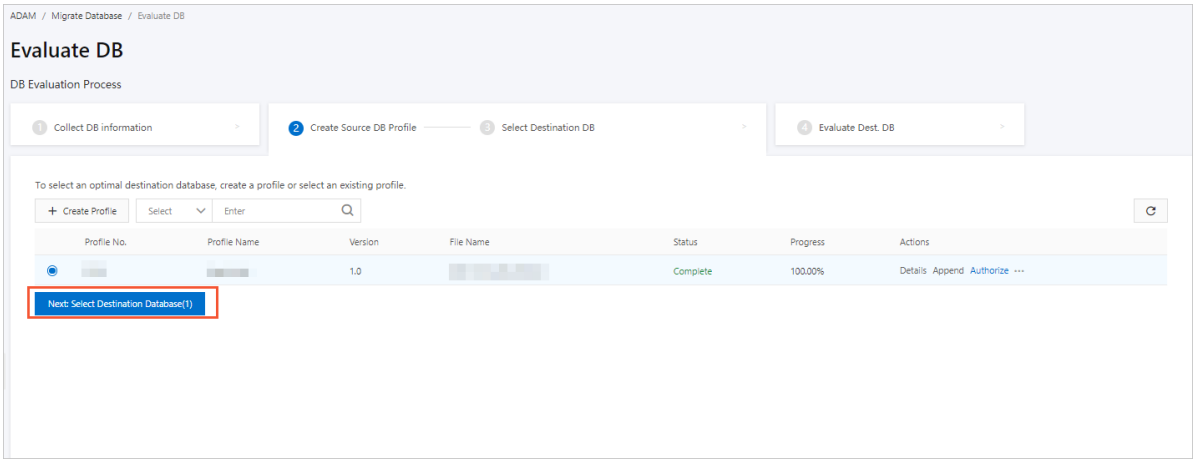


1.3. Select a destination database

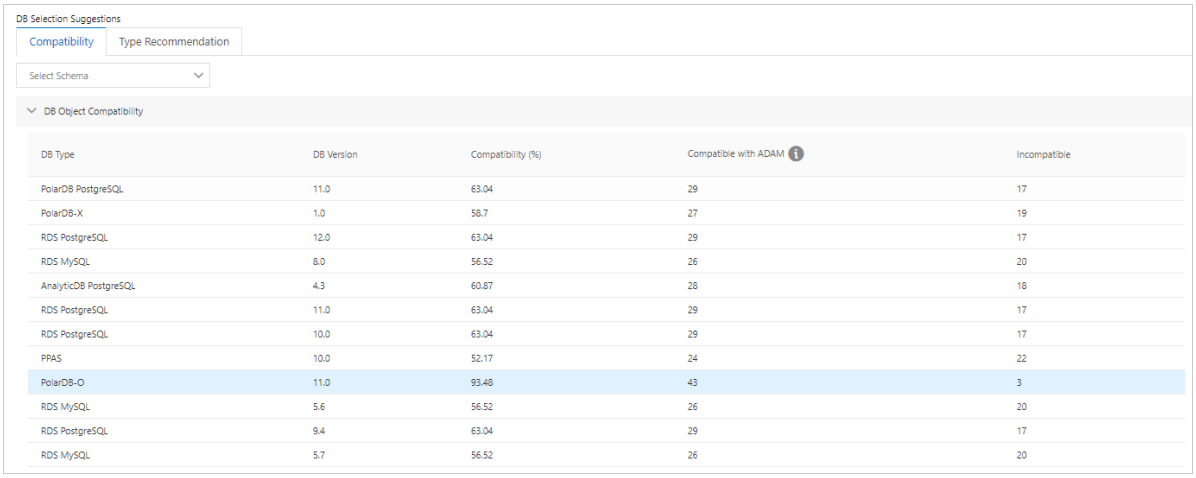
Advanced Database & Application Migration (ADAM) provides multiple databases. You can use the suggestions provided in the ADAM console to analyze the compatibility of a source database with available destination databases. Then, you can select a suitable destination database for your business scenario.

Procedure

1. Log on to the ADAM console.
2. In the left-side navigation pane, click **Evaluate DB**.
3. Click the **Select Destination DB** tab. In the profile list, select a profile to be evaluated.



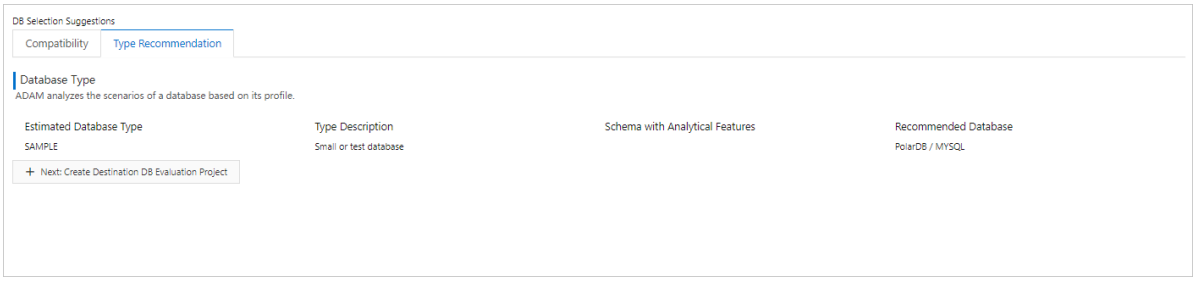
4. Click Next: Select Destination Database.
5. On the **Compatibility** tab of the page that appears, you can view the **DB object compatibility** and **SQL compatibility** of the source database with the destination database.



DB Type	DB Version	Compatibility (%)	Compatible with ADAM	Incompatible
PolarDB PostgreSQL	11.0	63.04	29	17
PolarDB-X	1.0	58.7	27	19
RDS PostgreSQL	12.0	63.04	29	17
RDS MySQL	8.0	56.52	26	20
AnalyticDB PostgreSQL	4.3	60.87	28	18
RDS PostgreSQL	11.0	63.04	29	17
RDS PostgreSQL	10.0	63.04	29	17
PPAS	10.0	52.17	24	22
PolarDB-O	11.0	93.48	43	3
RDS MySQL	5.6	56.52	26	20
RDS PostgreSQL	9.4	63.04	29	17
RDS MySQL	5.7	56.52	26	20

   - The DB Object Compatibility table includes the compatibility, the number of compatible objects, and the number of incompatible objects.
     - Compatibility: The compatibility is equal to the number of compatible objects divided by the total number of objects.
     - Compatible with ADAM: the objects that are compatible with the destination database, and the objects that ADAM can transform. ADAM automatically converts incompatible objects to compatible objects.
     - Incompatible: the objects that are incompatible and must be manually transformed. ADAM provides comprehensive suggestions on how to transform these objects.
   - The SQL Compatibility table includes the compatibility, the number of compatible SQL statements, the number of incompatible SQL statements that can be converted, and the number of incompatible SQL statements.
     - Compatible After Conversion: ADAM provides SQL statements that can be executed in the destination database, but you must modify the corresponding SQL statements in the application.
6. ADAM allows you to obtain suggestions only on the database schemas that you want to migrate. You can select the schemas from the **Select Schema** drop-down list. This prevents the accuracy of suggestions from being affected due to database schemas that you do not want to migrate.
7. On the **Type Recommendation** tab, you can view the recommended destination database. ADAM provides suggestions on appropriate destination databases based on the profile and scenarios of the source database, and the compatibility of the source database with available destination databases. The scenarios include transactional processing (TP), analytical processing (AP), hybrid transaction/analytical processing (HTAP), and tests.

> **Notice**
>
> The recommended destination databases vary with collected data and are for reference only.



Estimated source database type	Description
SAMPLE	Small or test databases.
OLTP	Databases for online transactional processing.
OLAP	Databases for online analytical processing.
HTAP	Databases for hybrid transaction/analytical processing.

1.4. Evaluate a database

Advanced Database & Application Migration (ADAM) allows you to evaluate the compatibility and specifications of destination databases, and migration risks. This helps you further understand the feasibility of migrating databases to the cloud and transformation cost.

Create a project

1. Log on to the ADAM console.
2. In the left-side navigation pane, click **Evaluate DB**.
3. In the **Source DB Profile** step, select a database profile, and click **Next: Select Destination Database**.
4. In the **Select Destination DB** step, click **Next: Create Destination DB Evaluation Project** to create an evaluation project.
5. In the **Create Project** panel, set the following parameters.
   - Project Name: the name of the project. This parameter is required.
   - Source Database Profile: Select a database profile.
   - Project Type: Select the type of the destination database.
   - Destination Database Version: the version of the destination database. This parameter is required.
   - Kernel Version: If you set Project Type to PolarDB for Oracle Edition Engine, this parameter is required.
   - Report Language: the language of the evaluation reports.
6. Click **Create**. The evaluation project is created and automatically performed.

> **Note**
> You can also click **+ Create Project** in the **Evaluate Dest. DB** step to create an evaluation project.

Evaluation Overview

In the project list, you can view the status and progress of the evaluation project. After the evaluation is complete, click **Details** in the Actions column to view the compatibility evaluation results of the destination database. The following sections are displayed: Project Summary, Evaluation Overview, and Evaluation Details.

The Evaluation Overview section includes the following subsections: Compatibility, Transformation, Specifications, Risks and Overall Compatibility.

- Compatibility: the compatibility between the source and destination databases. The higher the value, the fewer the SQL statements and objects that need to be transformed.
- Transformation: the number of focus areas of the destination database transformation.
  - Object Transformation: Database objects transformed by ADAM do not need to be modified.
  - Application Transformation: The focus areas of application transformation are obtained by analyzing SQL statements collected from the database during preliminary evaluation.
- Specifications: the required specifications of the destination database and migration cost calculated by ADAM based on collected data. The evaluation of the specifications is subject to the data collection environment. Purchase databases based on your business requirements.
- Risks: the existing risks of the source database and risks that may arise when the database is migrated. This dimension provides you with warnings about database migration and transformation.
- Overall Compatibility: the overall compatibility between the source and destination databases.

Evaluation Details

The Evaluation Details section is shown from the following dimensions: Object Compatibility, SQL Compatibility, Object Transformation, Destination DB Specifications, Migration Risk, and Project Dependency (Schema).

In the **Evaluation Details** section, click **Details** in the Actions column to view the details of each evaluation item.

Object Compatibility

The **Schema Compatibility** panel shows the evaluation results of all database objects, including compatible and incompatible database objects.

ADAM provides the transformed DDL statements and focus areas of transformation for objects that can be compatible after transformation. ADAM also explains why objects are incompatible and provides suggestions to transform them.

SQL Compatibility

The SQL Compatibility panel shows the evaluation results of SQL statements collected from the source database.

Click the **Evaluation Summary** tab to view the overview of the evaluation results. The overall evaluation results of SQL statements that are compatible, incompatible, and compatible after transformation are shown.

Click the **Evaluation Details** tab to view the compatibility details of SQL statements. The detailed evaluation results of SQL statements are shown from the following dimensions: compatibility, source SQL statements, and destination SQL statements.

- Click **View** in the **Source SQL Statements** column to view the details of the source SQL statements.
- Click **View** in the **Destination SQL Statements** column to view the details of the destination SQL statements. On the **Error Message** or **Edit Information** tab of the Destination SQL Statements panel, view the cause of incompatibility or information to be modified.

> **Note**
>
> ADAM evaluates the compatibility of SQL statements based on SQL statements executed and recorded in the database. You need to decide whether a specific SQL statement was issued by your business by using the ADAM application evaluation feature.

Click the **Rule Details** tab to view the detailed compatibility rules.

Object Transformation

The Project Transformation panel lists the focus areas of database transformation. You can transform your database objects based on these focus areas. You can also apply to use the ADAM database transformation feature to transform database objects. Some objects may need to be manually modified.

Transformation Level: All database objects that need to be transformed are classified by transformation difficulty to help you schedule transformation tasks. The higher the transformation level, the more difficult the transformation.

Click **View** in the Source DDL Operation column to view the source DDL operation.

Click **View** in the Destination DDL Operation column to view the destination DDL operation.

Click **Database** in the Transformation column to view the ID and transformation method of the focus area.

Destination DB Specifications

The Destination Database Solution panel provides specifications and migration plan guidelines to help you migrate databases to Alibaba Cloud.

The recommended destination database specifications are generated based on the configurations, performance, SQL, and external dependencies of the source database, and the comprehensive analysis of the source and destination databases.

On the **Destination Database Solution** tab, click **Details** in the Actions column to view the details of the objects in each migration group.

On the **Cross-database Objects** tab, click **Details** to view details of cross-database objects. A solution that has multiple instances may contain cross-database objects.

Migration Risk

Migration risks include risks from source and destination databases.

Risks from the source database may arise when you execute SQL statements that cause high CPU utilization and memory usage in the source database, such as SQL statements of the TOP CPU and TOP Buffer types.

Risks from the destination database may arise when you transform database schemas or execute SQL statements in the destination database. You must focus on migration risks to prevent performance differences caused by heterogeneous database migration.

Project Dependency (Schema)

The Project Dependency (Schema) panel shows dependent objects, referenced objects, and provides suggestions on these objects.

Download the evaluation reports

ADAM allows you to download the evaluation reports of your database.

In the **Evaluation Details** section, click **Download All Reports** to download all the evaluation reports of your database.

What to do next

In the **Evaluation Details** section, click **Start DB Transformation** to start database transformation. For more information, see Transform and migrate a database.

2.Create a database archive

This topic describes how to create a database archive in the Advanced Database & Application Migration (ADAM) console. In this topic, a PolarDB for Oracle database is used as an example.

Prerequisites

- A PolarDB for Oracle cluster is created.
- An account used to log on to the cluster is created.
- An IP whitelist is configured for the cluster. For information about the CIDR blocks of ADAM servers, see Configure the whitelist.

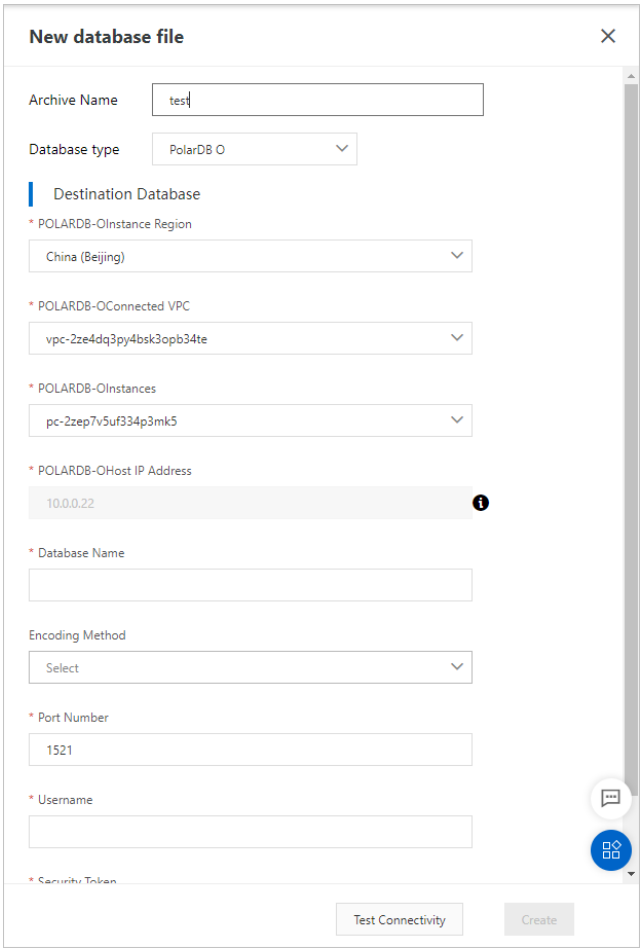
Supported databases

- Source database: Oracle, Teradata, and Db2 for LUW
- Destination database: PolarDB for Oracle, PolarDB for PostgreSQL, ApsaraDB RDS for MySQL, AnalyticDB for PostgreSQL, Greenplum, Distributed Relational Database Service, and ApsaraDB RDS for PostgreSQL

Procedure

This topic uses a PolarDB for Oracle database as an example. For information about how to create a database instance, see Purchase a subscription cluster. For information about how to create a database account, see Create a database account. For information about how to set a cluster whitelist, see Configure an IP whitelist.

1. Log on to the ADAM console.
2. In the left-side navigation pane, click **Database Management**.
3. In the upper-left corner of the **Database management** page, click **Create Database**.
4. In the **New Database File** panel, set the required parameters.



Parameter	Description
Archive Name	The name of the database archive.
Database Type	Set the value to **PolarDB O**. For information about database types supported by ADAM, see Overview.
POLARDB-O Instance Region	The region where the PolarDB for Oracle cluster resides.
POLARDB-O Connected VPC	The virtual private cloud (VPC) of the cluster.

Parameter	Description
POLARDB-O Instances	The ID of the cluster.
POLARDB-O Host IP Address	The IP address of the cluster.
Database Name	The name of the cluster.
Encoding Method (Optional)	The encoding method of the cluster.
Port Number	The public port number of the cluster.
Username	The username used to log on to the cluster.
Security Token	The security token used to log on to the cluster.

5. Click **Test Connectivity**.
6. After the link succeeds, click **Create**.

3.Online Transformation and Migration

3.1. Overview

This topic describes how to migrate and transform databases online.

Supported database types

Advanced Database & Application Migration (ADAM) allows you to migrate data from on-premises Oracle databases to PolarDB O Edition, ApsaraDB RDS for MySQL, ApsaraDB RDS for PostgreSQL, ApsaraDB RDS for PPAS, and AnalyticDB for PostgreSQL databases. The following versions are supported.

> **Note**
> ADAM supports Oracle Database 10g, 11g, 12C, and later.

- PolarDB O Edition
- PolarDB for PostgreSQL
- PolarDB-X version 1.0 (Only schemas can be migrated)
- ApsaraDB RDS for MySQL
  - Version 5.6
  - Version 5.7
  - Version 8.0
- ApsaraDB RDS for PostgreSQL
  - Version 10.0
  - Version 11.0
  - Version 12.0
- AnalyticDB for PostgreSQL
  - Version 4.3

Prerequisites

Before you migrate or transform a database online, make sure that your account has sufficient permissions.

- If you are using an Alibaba Cloud account, skip this step.
- If you are using a RAM user, perform the following steps:
  i. Log on to the ADAM console by using your Alibaba Cloud account.
  ii. On the Transform & Migrate DB page, grant the relevant permissions to the RAM user.

Procedure

1. Create a migration project: Create a migration project based on the evaluation result.
2. Run a recheck: Check the permissions, plug-ins, and version of the destination database to ensure smooth migration.
3. Optional. Verify the source database: Analyze the source Oracle database, track database changes, and update the migration project based on the database changes. To verify a migration project, make sure that the ADAM server can access the source Oracle database. If you skip this step, the migration project may not be created based on the latest data. In this case, you must collect and evaluate the database data again.
4. Migrate and revise schemas: Migrate objects to the destination database as possible as you can. ADAM evaluates and verifies the objects and provides solutions for incompatible objects. You can troubleshoot issues based on the error messages and try again.
5. Optional. Compare the incremental data: Track the major changes to the data and schemas of your source database. This improves the migration efficiency of changed data definition language (DDL) statements.

3.2. Create a migration project

You can create a migration project in the Advanced Database & Application Migration (ADAM) console. Then, you can migrate data from a source database to a destination database based on the database evaluation results. The migration project maximizes the compatibility between the source and destination databases by converting data types. ADAM also provides a schema migration tool that automatically converts data types based on the migration project. This topic describes how to create a migration project.

Prerequisites

- The source database is evaluated. For more information, see Evaluate a database.
- A database evaluation project is created. For more information, see Manage a database profile.
- A destination database archive is created. For more information, see Create a database archive.
- If you migrate data to a PolarDB-X V1.0 instance, you must first create a database and bind it to an RDS instance. The database must have the same name as the schema on the instance.

Procedure

1. Log on to the ADAM console.
2. In the left-side navigation pane, click **Transform & Migrate DB**.
3. On the **Transform and Migrate DB** page, click **Create Migration Project**.

4. In the **Create Migration Project** panel, set the required parameters.

Parameter	Description
Project Name	Enter the name of the migration project.
Evaluate DB	Select the database evaluation project.
Solution Details	Select a migration solution.
Destination Database	Select the archive of the destination database.
Transformation Type	Only **Online Transformation** is available. ADAM allows you to migrate data from on-premises Oracle databases to databases such as PolarDB for Oracle.

5. Click **Test Connectivity**.
6. After the connectivity test succeeds, click **Create**.

> **Note** After a migration project is created, you can click in the upper-right corner to refresh the Transform and Migrate DB page.

7. After the status of the project changes to `ACTIVE`, click **Details**.

What's next

After the migration project is created, ADAM automatically performs a precheck. For more information, see Run a precheck.

3.3. Configure the whitelist

This topic lists the CIDR blocks of ADAM servers in different regions. When you create a destination database, you can add the CIDR blocks of ADAM servers to the whitelist of the database based on the region where the database resides.

Configure the VPC whitelist

Region	Network type	CIDR block
China (Zhangjiakou)	vpc	100.104.127.0/26
China (Hangzhou)	vpc	100.104.20.0/26
China (Shanghai)	vpc	100.104.107.64/26
China (Beijing)	vpc	100.104.141.128/26
China (Shenzhen)	vpc	100.104.136.192/26
China (Hong Kong)	vpc	100.104.87.192/26
Singapore (Singapore)	vpc	100.104.246.192/26
Malaysia (Kuala Lumpur)	vpc	100.104.3.128/26
Indonesia (Jakarta)	vpc	100.104.88.64/26
Japan (Tokyo)	vpc	100.104.111.128/26
US (Silicon Valley)	vpc	100.104.36.0/26

Configure the Internet whitelist

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

3.4. Run a precheck

Context

After you create a migration project, Advanced Database & Application Migration (ADAM) automatically runs a precheck.

> **Note** This operation is supported only when you migrate data to PolarDB O Edition databases or ApsaraDB RDS for PPAS. If you migrate data to other databases, ADAM automatically skips the precheck.

- Check the permissions of the account of the destination database. If your account does not have the required permissions, recreate a migration project. For more information, see Why does the account of the destination database not have sufficient permissions to migrate data?
  - Check whether the account of the destination database has the permissions to create users.
  - Check whether the account of the destination database has the permissions to authorize users.
  - Check whether the account of the destination database has the permissions to create and delete schemas.
  - Check whether the account of the destination database has the permissions to create and delete data definition language (DDL) statements.
- Check the destination database plug-ins

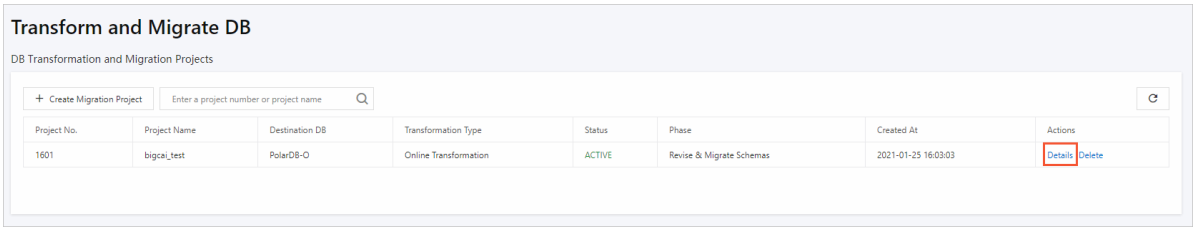
  Check whether the required plug-ins are installed on the destination database. If a plug-in is not installed, execute the corresponding SQL statement to install the plug-in. For more information, see Why is a plug-in of the destination database missing?

- Check the version of the destination database

  Check whether the version of the evaluated destination database is the same as the version of the destination database that is used to migrate schemas. If the two versions are inconsistent, recreate a project to evaluate the destination database. For more information, see Why is the specified database version in a created database evaluation project inconsistent with the actual database version?

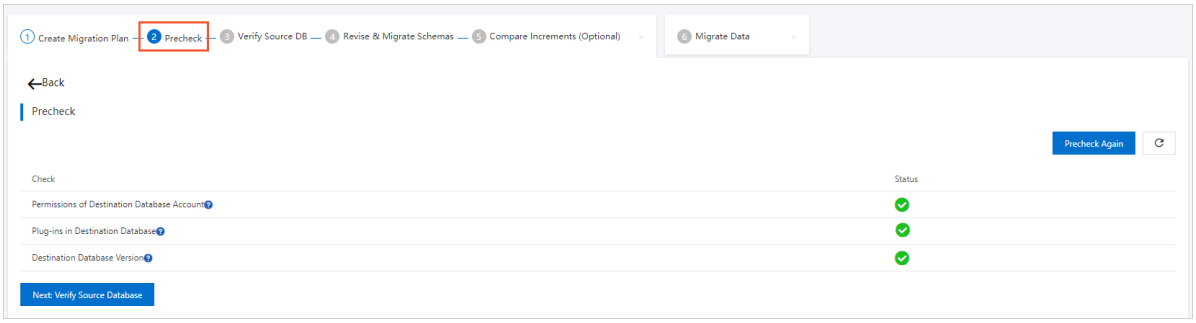
Procedure

1. If the status of the project changes to `ACTIVE` and the font color is green, click **Details**.



   - If the status of the project is `ACTIVE` and the font color is yellow, you can click **Details** and fix the issue based on the precheck result. For more information, see Precheck-related FAQ. If the font color of `ACTIVE` remains yellow after you fix the issue, ignore the issue.

2. Click **Precheck** to view the precheck result.



What's next

Click **Next: Configure Source Database**. For more information, see Verify the source database.

3.5. Verify the source database

If no major changes are performed on to the data or schemas, you can skip the verification and migrate data. Then, the evaluation result of the destination database is used. If major changes are made to the data and schemas, you must recollect and reevaluate data before you migrate data.

Context

- You can verify whether major changes are performed on the data and schemas of the source database since the last data collection.
- You must add the CIDR blocks of Advanced Database & Application Migration (ADAM) servers (`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`) to the source database. This ensures that ADAM can access the source database.
  - If the source database has a public IP address, you can access the database over the Internet.
  - If the source database has no public IP address, you can access the database by using Alibaba Cloud Database Gateway. Database Gateway is a database connection service that supports remote access to private network databases. Database Gateway allows you to access and manage databases that are deployed on data centers or third-party cloud platforms. For more information, see Database Gateway.

Procedure

- Skip the verification and migrate data.
  i. Click **Skip Verification and Migrate Data**.
  ii. In the message that appears, click **OK**.
- Verify the source database.
  i. Enter the network connection information of the source database.
     - Set the Instance Type parameter to Database with Public IP Address.
       - Instance Region: the region where the network of the source database resides.
       - Host IP Address: the IP address of the source database.
     - Set the Instance Type parameter to Database Without Public IP Address or Port Number (Connected over Database Gateway).
       - Instance Region: the region where the database gateway of the source database resides.
       - Database Gateway ID: the ID of the database gateway that is connected to the source database.
       - Source Database Endpoint: the endpoint of the database gateway that is connected to the source database.
  ii. Set the parameters of the source database.
     - Server Name/SID: the server name or system ID (SID) of the source database.

- Encoding Method: the encoding method of the source database.This parameter is optional.
- Port Number: the port that is used to connect with the source database.
- Username: the username of the account. The username is used to collect data. For more information, see Collect database data.
- The password of the account.

iii. Click **Test Connectivity** to perform a test. After the test succeeds, click **Verify Now**.

Result page

You can ignore or convert the verification result.

- Ignore : ignores data definition language (DDL) schema migration.
- Convert : converts the *NEW* and *CHANGED* states.

>

What's next

Click **Next: Schema Migration**. For more information, see Migrate schemas.

3.6. Migrate schemas

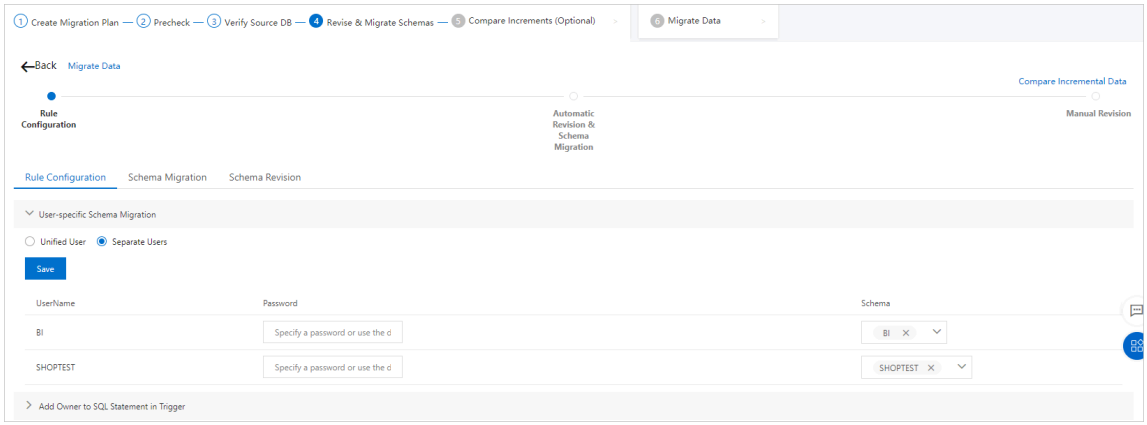
Advanced Database & Application Migration (ADAM) allows multiple users to migrate schemas. You can configure a rule to specify **separate users**. This allows you to manage the permissions of users to migrate and revise specific schemas.

1. Configure a migration rule.
   i. **User-specific Schema Migration**

   > **Note** If you configure separate users, you must ensure that these users are consistent with the users of the source Oracle database and specify corresponding schemas for the users.

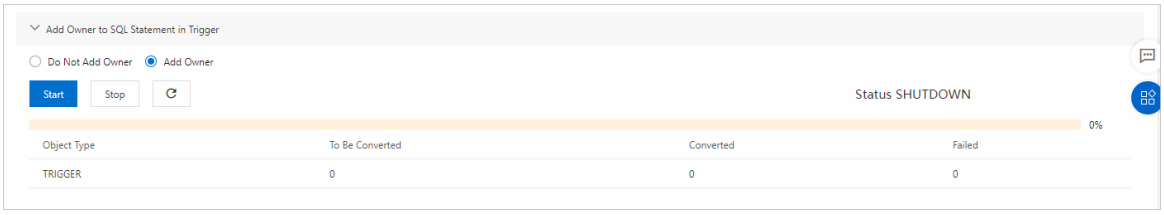
   - Unified User
   - Separate Users
     a. Enter the password of the account.
     - If you have created a user, enter the password of the account.
     - If you have not created a user, ADAM automatically creates a user in the destination database. Then, you must specify a password. Otherwise, the password of the account used to log on to the destination database is specified by default.

     b. Add or remove schemas of users as needed.
     c. Click **Save**.
     d. In the message that appears, click **OK**.



   ii. **Add an owner to an SQL statement in the trigger**

   PolarDB cannot automatically identify the owner of the trigger. You must add the owner to an SQL statement in the trigger before you migrate schemas to PolarDB.

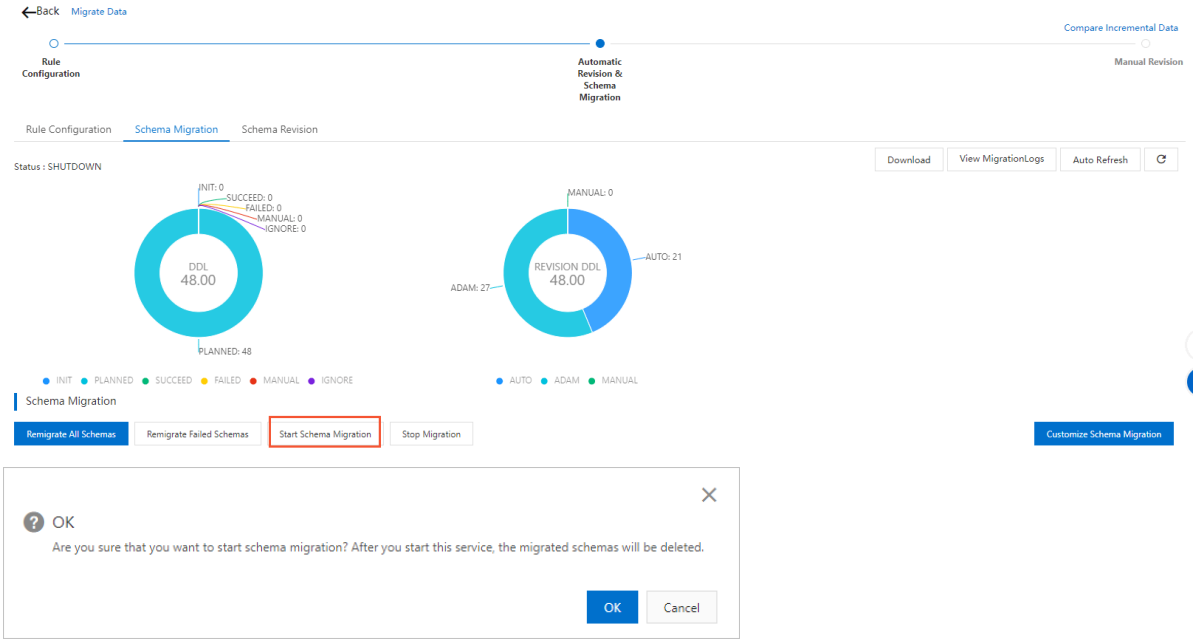


   - Do Not Add Owner
   - Add Owner: In the dialog box that appears, click **Start**. Then, click **OK**. ADAM automatically adds the owner to an SQL statement in the trigger. To stop adding Owner, click **Stop**.

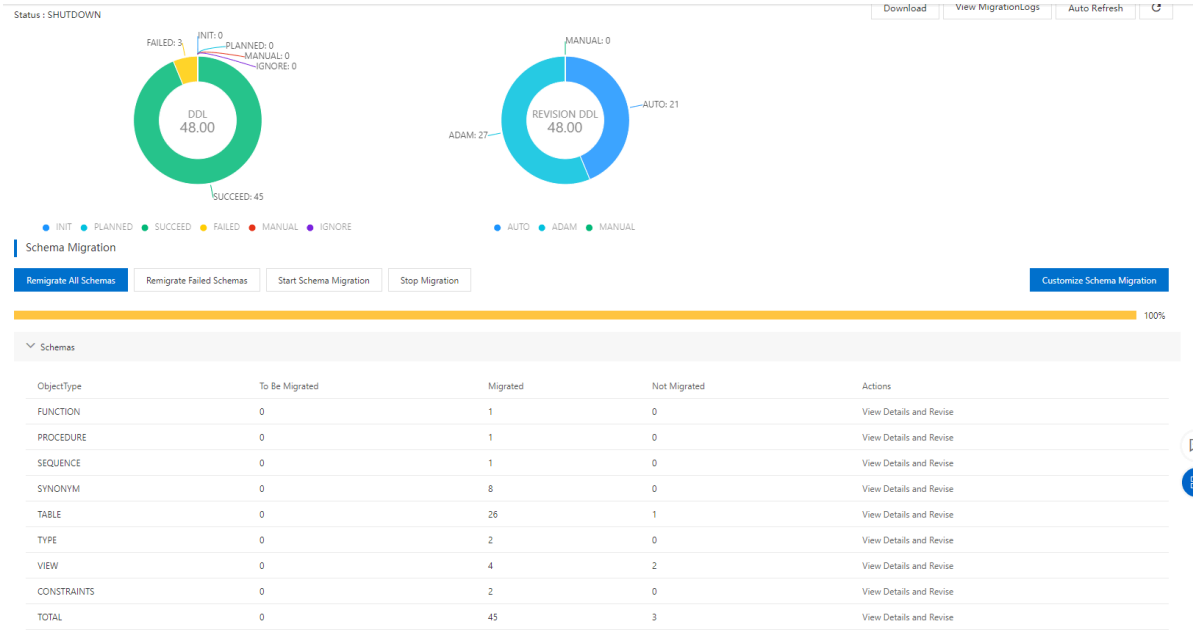
2. In the Schema Migration section, click **Start Schema Migration**.

> **Notice**
> - If you start the schema migration, all schema objects in the destination database are deleted. Make sure that no important schema objects exist in the destination database in advance.
> - If you migrate schemas to a PolarDB-X V1.0 database, the migration may consume much time due to the constraint of foreign keys.

In the message that appears, click **OK**.



3. View the migration result.



What's next

- Click **Remigrate All Schemas** to re-migrate data definition language (DDL) statements. All the DDL statements will be re-migrated.
- Click **Remigrate Failed Schemas** to migrate the DDL statements that fail to be migrated.
- Click **Stop Migration** to stop migrating the current DDL statement.
- Click **Start Schema Migration** to customize migration. For more information, see Customize schema migration.

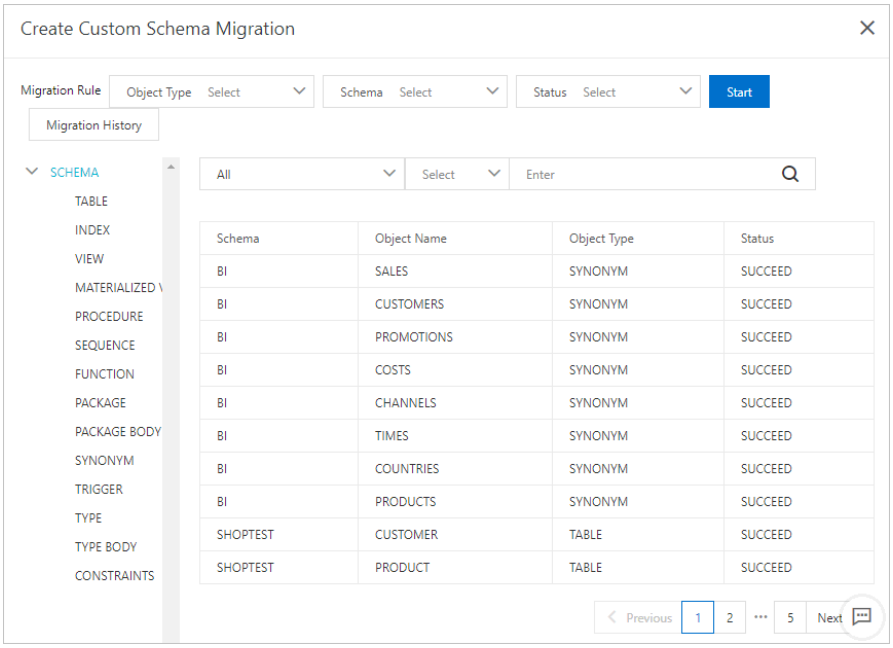
3.7. Customize schema migration

Context

To meet different schema migration requirements, ADAM allows you to customize schema migration. You can migrate schemas by specifying the **Object Type**, **Schema**, or **Status** parameter.

Procedure

1. Click **Customize Schema Migration**. The **Customize Schema Migration** dialog box appears.



2. In the Customize Schema Migration dialog box, configure the **Migration Rule**. You can set the **Object Type**, **Schema**, or **Status** parameter.
3. Click **Start**.

> **Note** To view the progress of custom schema migration, click the Refresh icon.

What's next

- To view the DDL statements of the custom schema migration, click **View Details** in the Actions column.
- To start the custom schema migration, click **Start** in the Actions column.
- To stop the custom schema migration, click **Stop** in the Actions column. The migration can be stopped only when it is in the Running state.
- To delete the custom schema migration, click **Delete** in the Actions column.

3.8. Revise schemas

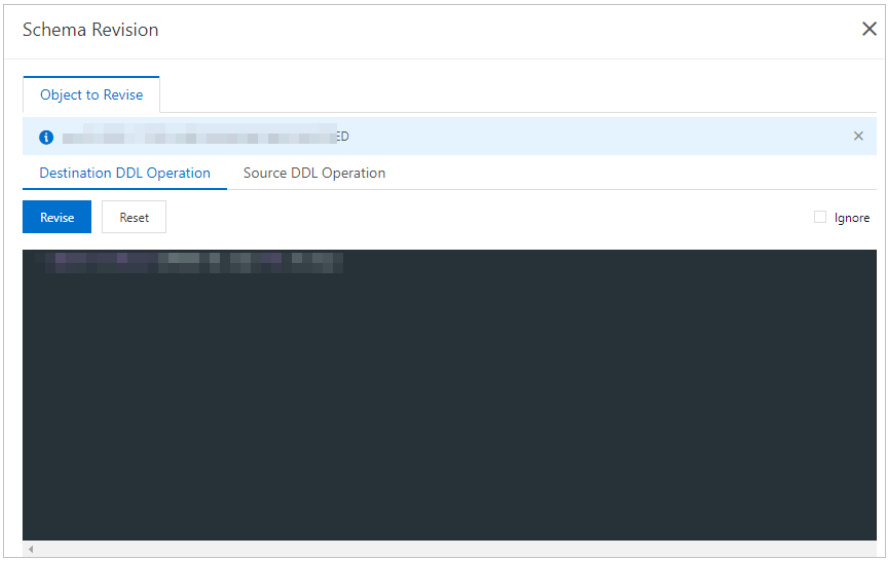
This topic describes how to revise and execute data definition language (DDL) statements.

Context

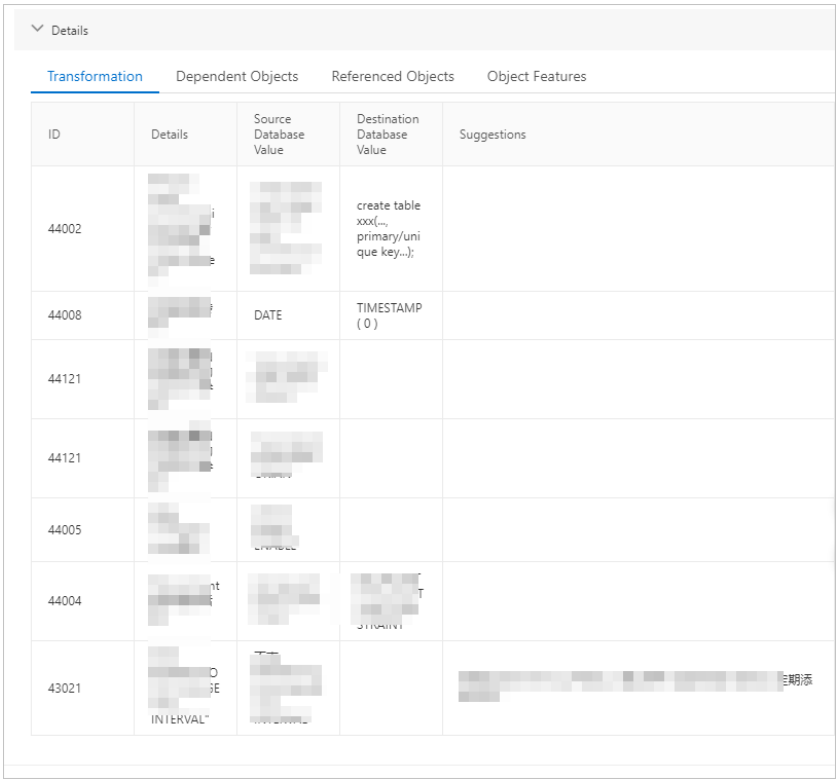
- Advanced Database & Application Migration (ADAM) identifies incompatible objects.
- ADAM depends on the objects of incompatible objects.

Procedure

1. Click the **Schema Revision** tab.
2. Filter the schema objects that are in the `FAILED` or `MANUAL` state. Then, click **Revise** in the Actions column.
3. On the **Object to Revise** tab of the Schema Revision dialog box, set the following parameters.
   - **Revise**: Revise the current object.
   - **Reset**: Reset the initial object.
   - **Ignore**: Skip the migration of the object.



- **Transformation**: View the current DDL statements that ADAM transforms. You must manually transform unsupported DDL statements.
- **Dependent Objects**: View the dependent objects of the current object.
- **Referenced Objects**: View the referenced objects of the current object.
- **Object Features**: View the features of the current object.



What's next

If major data changes occurred in the source database, you can click the **Compare Increments (Optional)** step to compare incremental data.

3.9. Track incremental data by performing data comparison

This topic describes how to track incremental data by performing data comparison.

Context

If the data or schemas of your source database have undergone major changes, you can perform data comparison to track the changes. This feature facilitates the migration of changed data definition language (DDL) statements.

Procedure

1. Click the **Compare Increments (Optional)** tab.

2. Click **Start** to perform data comparison and track the incremental data.
3. After data comparison is completed, click **Incremental Data Comparison History** to view the records.

> **Note** If you discover schema changes after data comparison, you can return to the previous step and click **Start** to synchronize schemas. For more information, see Migrate schemas.

4.Database cutover

4.1. Create a cutover project

Advanced Database & Application Migration (ADAM) allows you to migrate your services to a cloud database after database evaluation, transformation, and migration, and application evaluation and transformation. This topic describes how to create a cutover project in ADAM.

Prerequisites

- A source database archive and destination database archive are created. For more information, see Create a database archive.
- A migration project is created. For more information, see Create a migration project.

Procedure

1. Log on to the ADAM console.
2. In the left-side navigation pane, click **Database Cutover**.
3. On the Cutover page, click **Create Cutover Migration Project**.
4. In the **Create Cutover Migration Project** panel, set the parameters.

Parameter	Description
Project Name	Enter the name of the cutover project.
Transform & Migrate Database	Select the migration project.
Business Scenario	○ **Cutover Based on Business Testing (Recommended)**: To ensure that the cutover result meets your requirements, you need to perform business testing, cutover testing, and rollback testing. ○ **Directly Create Cutover Migration Project**: The scenario is suitable for customers who need to migrate a large number of databases to the cloud, and have completed the testing of business, cutover and rollback for a few migration projects. This ensures the stability and efficiency of the cutover.
Source Database	Select the source database archive.
Destination Database	Select the destination database archive.

5. Click **Create**.

What's next

After the cutover project is created, click **Details** in the **Actions** column. On the **Business Testing** page, you can perform business testing for the cutover.

4.2. Perform business testing

Advanced Database & Application Migration (ADAM) allows you to migrate your services to a cloud database after database evaluation, transformation, and migration, and application evaluation and transformation. After you create a cutover project, you can perform business testing on the project. This way, you can check whether the performance of the destination database can meet your requirements.

Prerequisites

A cutover project is created. For more information, see Create a cutover project.

Procedure

1. Log on to the ADAM console.
2. In the left-side navigation pane, choose **Heterogeneous Database Migration (ADAM) > Database Cutover**. On the Cutover page, click View Cutover Migration Project.
3. On the **Cutover Migration Projects** page, select the project, and click **Details** in the **Actions** column.
4. The **Business Testing** page includes the following steps:
   i. Destination Database Parameters
      - If you do not need to modify custom parameters, click **Next** and skip to the **Forward Incremental Synchronization (Including Full Synchronization)** step.
      - To modify custom parameters, perform the following steps:
        a. Click the **Custom Parameters** tab, specify **Custom Value** based on the **Value Range** parameter.
        b. Click **Save**.
        c. Click **Next** to go to the **Forward Incremental Synchronization (Including Full Synchronization)** step.
   ii. Forward Incremental Synchronization (Including Full Synchronization)

   Before you migrate data, you need to delete schemas, such as triggers, constraints, and indexes. After data is migrated, you can recover the schemas. In the Excluded Schemas section, you can select the schemas that you want to temporarily delete.

   a. For more information, see Migrate data.
   b. Click **Next** to go to the **SQL Comparison** step.

iii. SQL Comparison

The SQL comparison feature compares the execution results of an SQL statement in the source and destination databases. The results show the differences in execution time, number of returned rows, and SQL result set. To implement SQL comparison, perform the following steps:

> **Note** The step is optional. To skip the step, click **Skip** in the **SQL Comparison** step.

a. Click **Create SQL Test Set**. In the Create SQL Test Set panel, set the following parameters.

Parameter	Description
Name	Enter the name of the SQL test set.
Database Evaluation Project	Select the database evaluation project.
SQL Collection Project	Select the SQL collection project.
Adapter	Select the Adapter instance.

b. In the **Create SQL Test Set** panel, click **Create**.

c. Select the SQL test set and click **Details** in the **Actions** column.

d. On the **Test Set Details** tab, revise SQL statements based on your needs.

e. Click **Next** to go to the **Destination Database Performance** step.

iv. Destination Database Performance

In the **c** step, view the storage capacity and CPU utilization of the destination database.

What's next

After business testing is completed, click **Next** in the **Destination Database Performance** step. In the Cutover Testing step, you can perform cutover testing. For more information, see Perform cutover testing.

4.3. Migrate data

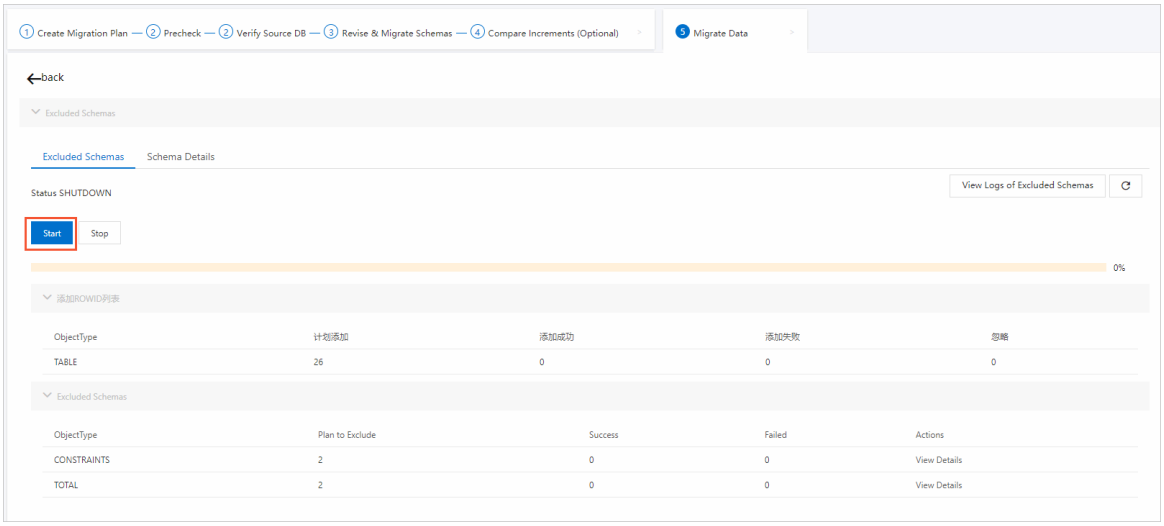
Advanced Database & Application Migration (ADAM) allows you to use Data Transmission Service (DTS) to migrate data. This topic describes how to exclude schemas, migrate data, and include schemas.

Procedure

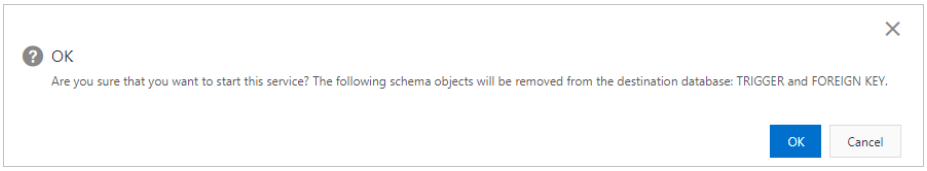
1. Exclude schemas.

i. Click the **Excluded Schemas** tab on the **Excluded Schemas** tab, and then click **Start**.

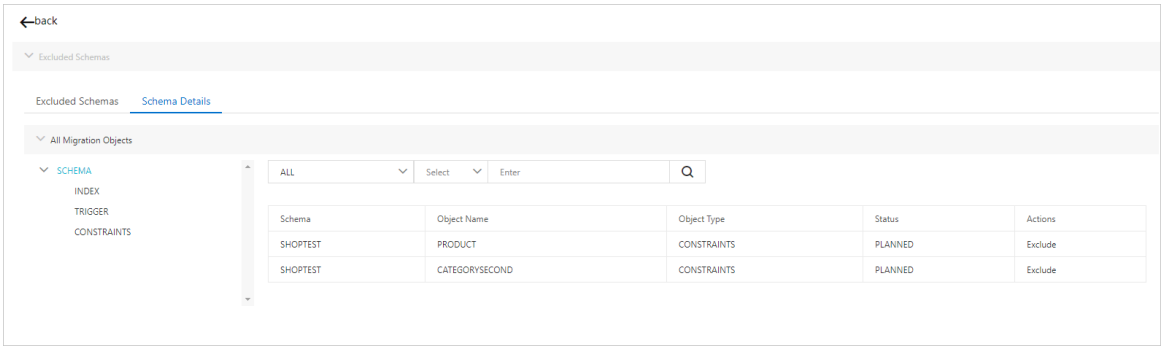
Before you migrate data, you must remove TRIGGER and FOREIGN KEY from the destination database.



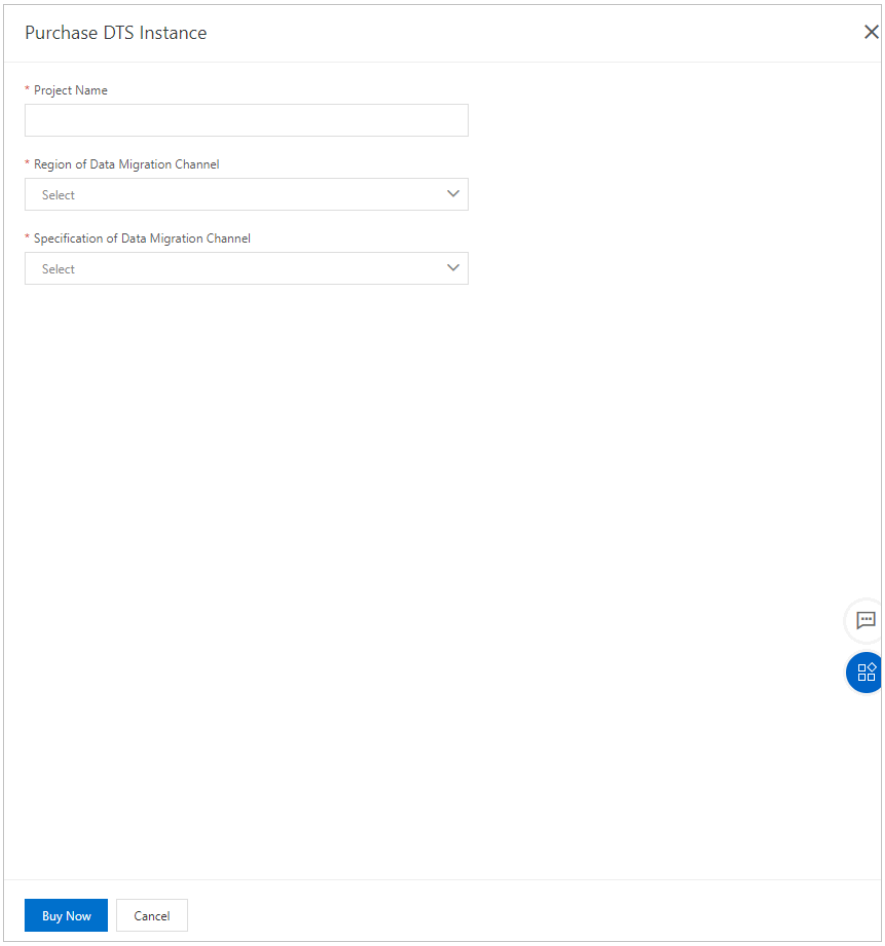
ii. In the message that appears, click **OK**.



iii. Click the **Schema Details** tab to view the details of the schema objects that you want to exclude.



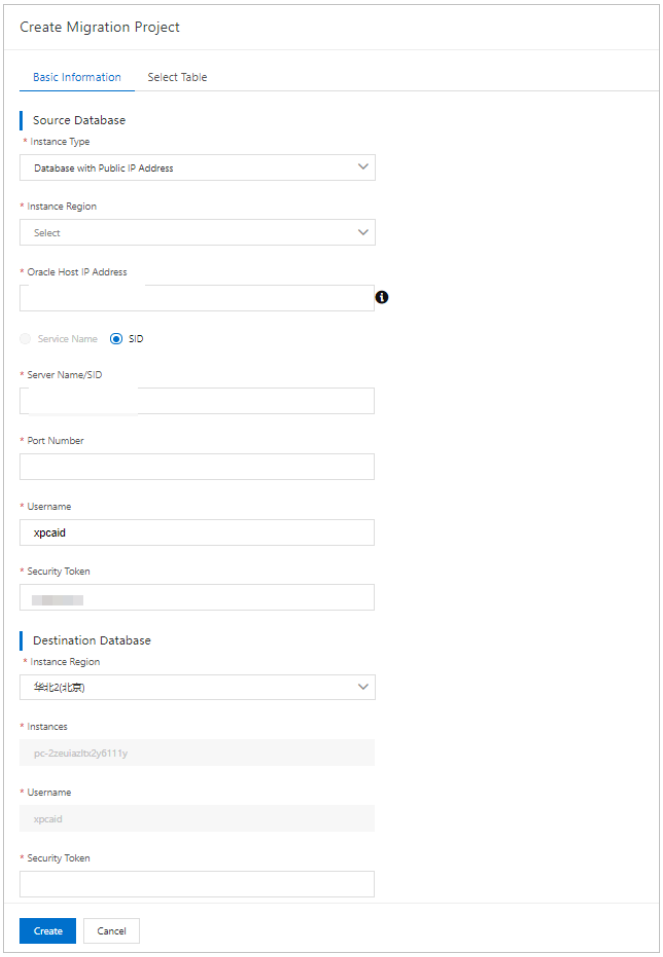
2. Migrate data.
   i. Click the **Data Migration** tab, and then click **Create Migration Project**.
   ii. In the **Purchase DTS Instance** dialog box, enter the required information and click **Buy Now**.



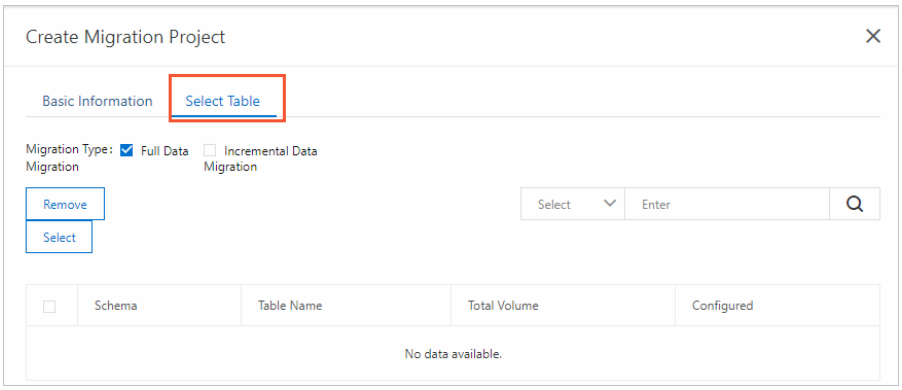
- Project Name: the name of the migration project.
- Region of Data Migration Channel: the region where the destination instance resides.
- Specification of Data Migration Channel: the specification of the data migration instance. For more information, see Specifications of data migration instances.

iii. Click **Configure Migration Task**. In the Create Migration Project dialog box, set the parameters of the source Oracle database and the destination database. Add the CIDR blocks of DTS servers to the whitelist of the source Oracle database.

> **Note** ADAM allows you to migrate Oracle databases that have public IP addresses or Oracle databases that are connected over Database Gateway.



iv. Click **Test Connectivity** to perform a test. After the test succeeds, click the **Select Table** tab. Select **Full Data** or **Incremental Data Migration** based on your business requirements. Click Select and select the tables that you want to migrate.



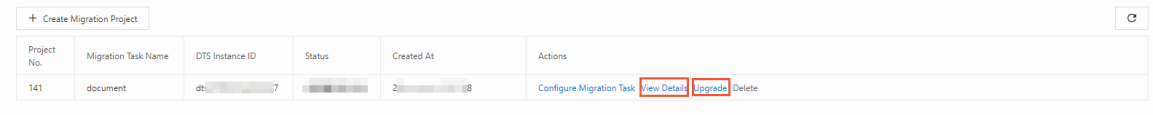
If you select tables, the following message appears: Do you want to select all records?

- **Yes**: Select all the tables in the table list.
- **No**: Select only the tables on the current page.

By default, the tables are sorted by data volume. You can skip the migration of the tables that have a large amount of data, such as logs. We recommend that you remove these tables after you click Yes.

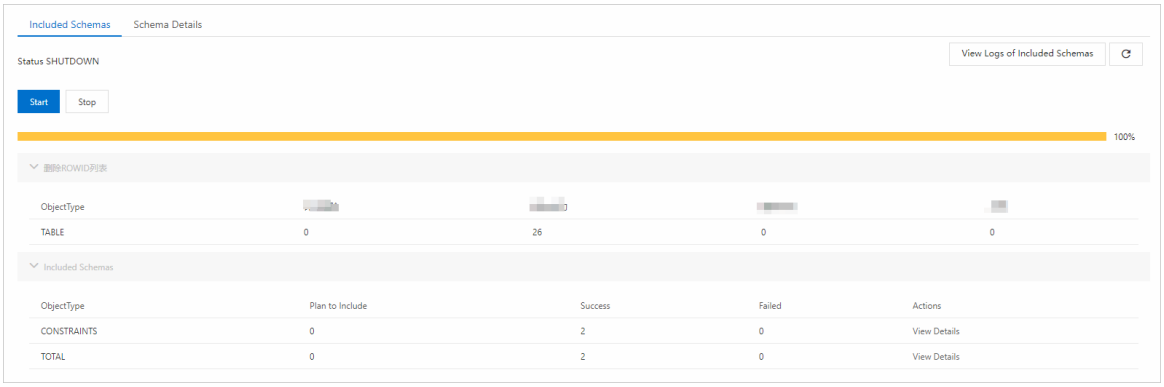
v. Click **Create**. After the configuration is complete, the migration task automatically starts. However, errors may occur if a large amount of data is migrated at one time. Therefore, ADAM migrates data in batches.

vi. Click **View Details** to go to the DTS console. On the Overview page, you can check the details of the migration task. If the data migration instance cannot meet your requirements for data volume, you can click **Upgrade** in the Actions column to upgrade the specification of the data migration instance.

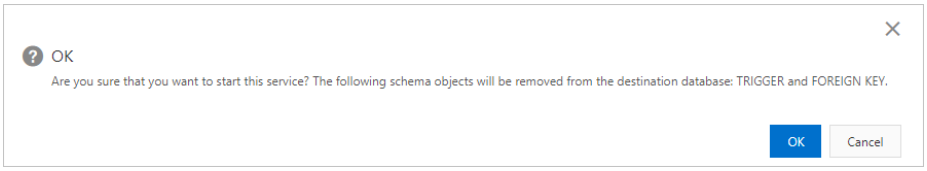


3. Include schemas.
   i. Click **Start** on the **Included Schemas** tab.

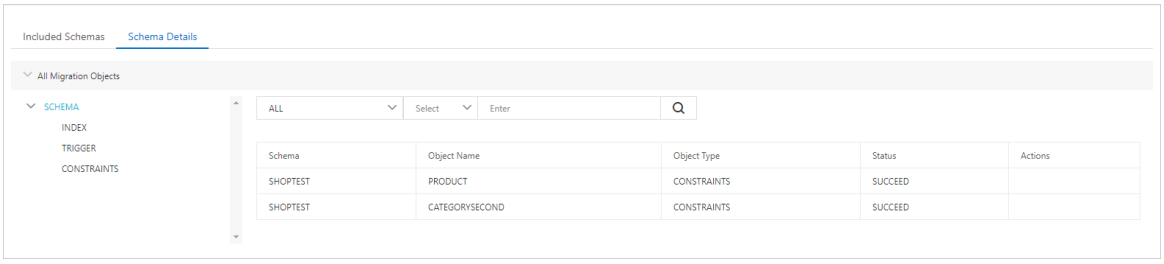
   Create a trigger and foreign key after the data migration is complete.



   ii. In the message that appears, click **OK**.



   iii. Click the **Schema Details** tab to view the details of the schema objects.



4.4. Perform cutover testing

The cutover testing feature allows you to synchronize the task changes of the source database and the destination database, stop forward synchronization tasks, and prepare for data rollback. This topic describes the procedure of cutover testing.

Prerequisites

Business testing is complete. For more information, see Perform business testing.

Cutover testing

1. Check Incremental Synchronization Tasks

   You need to synchronize the task changes of the source database to the destination database.

2. Stop Applications and Modify Settings

   You need to stop the applications and database connection, and modify application configurations.

   > **Note** ADAM allows you to query the connection status of the database. If the value of the Session parameter becomes 0, the applications are stopped.

3. Stop Source Database Tasks and Trigger

   After you stop the applications and modify application configurations, you need to stop the tasks and triggers of the source database. This prevents the data of the source database from changing.

4. Stop Forward Synchronization Tasks

   You need to stop the tasks that synchronize data from the source database to the destination database.

5. Auto Increment Schemas

You can synchronize the data changes of the source database to the destination database.

6. Start Reverse Synchronization Data Links

   The step is designed as the preparations for rollback testing. You need to create a Data Transmission Service (DTS) task to synchronize the data of the destination database to the source database. This ensures that invalid data can be rolled back.

7. Start Destination Database Tasks and Trigger

   You need to start the tasks and triggers of the destination database.

8. Start Applications

   Before you start the applications, you must make sure that the connection of the destination database is running as expected.

What's next

In the **Start Applications** step, click **Next** to perform rollback testing. For more information, see Perform rollback testing.

4.5. Perform rollback testing

The rollback testing feature allows you to synchronize data from the destination database to the source database. This topic describes the procedure of rollback testing.

Prerequisites

Cutover testing is complete. For more information, see Perform cutover testing.

Rollback testing

1. Check Reverse Incremental Synchronization Tasks

   You need to synchronize the task changes of the destination database to the source database. This way, you can check whether the data synchronization meets your requirements.

2. Check Destination Tasks and Long-Running Transactions

   You need to check whether the long-running transactions are running in the destination database to prevent them from affecting your business.

3. Stop Applications and Modify Settings

   You need to stop the applications and database connection, and modify application configurations.

   > **Note** ADAM allows you to query the connection status of the database. If the value of the Session parameter becomes 0, the applications are stopped.

4. Stop Destination Database Tasks and Trigger

   After you stop the applications and modify application configurations, you need to stop the tasks and triggers of the destination database. This prevents the data of the source destination from changing.

5. Stop Reverse Synchronization Data Links

   You need to stop the tasks that synchronize data from the destination database to the source database.

6. Auto Increment Schemas

   You can synchronize the data changes of the destination database to the source database.

7. Perform Forward Incremental Synchronization

   To enable the rollback process to be repeatedly executed, you need to configure a forward synchronization chain to ensure data synchronization.

8. Start Source Database Tasks and Trigger

   You need to start the tasks and triggers of the source database.

9. Start Applications

   Before you start the applications, you must make sure that the connection of the source database is running as expected.

What's next

In the **Start Applications** step, click **Next** to perform cutover migration. For more information, see Perform cutover migration.

4.6. Perform cutover migration

Cutover migration is the final stage of the database cutover process. In this stage, you can modify connection methods based on your needs. This topic describes the procedure of cutover migration.

Prerequisites

Cutover testing is completed. For more information, see Perform rollback testing.

Cutover migration

1. Configure Source and Destination Databases

   You need to configure the information of the source database and the destination database. You can modify connection methods based on your needs.

2. Clear Destination Database Data

   Clear the data of the destination database that is used for testing in the previous steps. You can check destination database information, delete data in batches, or clear all data.

3. Check Incremental Synchronization Tasks

   You need to synchronize the task changes of the source database to the destination database.

4. Stop Applications and Modify Settings

   You need to stop the applications and database connection, and modify application configurations.

   > **Note** ADAM allows you to query the connection status of the database. If the value of the Session parameter becomes 0, the applications are stopped.

5. Stop Source Database Tasks and Trigger

   After you stop the applications and modify application configurations, you need to stop the tasks and triggers of the source database. This prevents the data of the source database from changing.

6. Stop Forward Synchronization Tasks

   You need to stop the tasks that synchronize data from the source database to the destination database.

7. Auto Increment Schemas

   You can synchronize the data changes of the source database to the destination database.

8. Start Reverse Synchronization Data Links

   The step is designed as the preparations for rollback testing. You need to create a Data Transmission Service (DTS) task to synchronize the data of the destination database to the source database. This ensures that invalid data can be rolled back.

9. Start Destination Database Tasks and Trigger

   You need to start the tasks and triggers of the destination database.

10. Start Applications

    Before you start the applications, you must make sure that the connection of the destination database is running as expected.

11. Stop Reverse Synchronization Data Links

    > **Note** If the applications run as expected, you can stop the tasks that synchronize data from the destination database to the source database. The step is optional.

5.Application evaluation and transformation

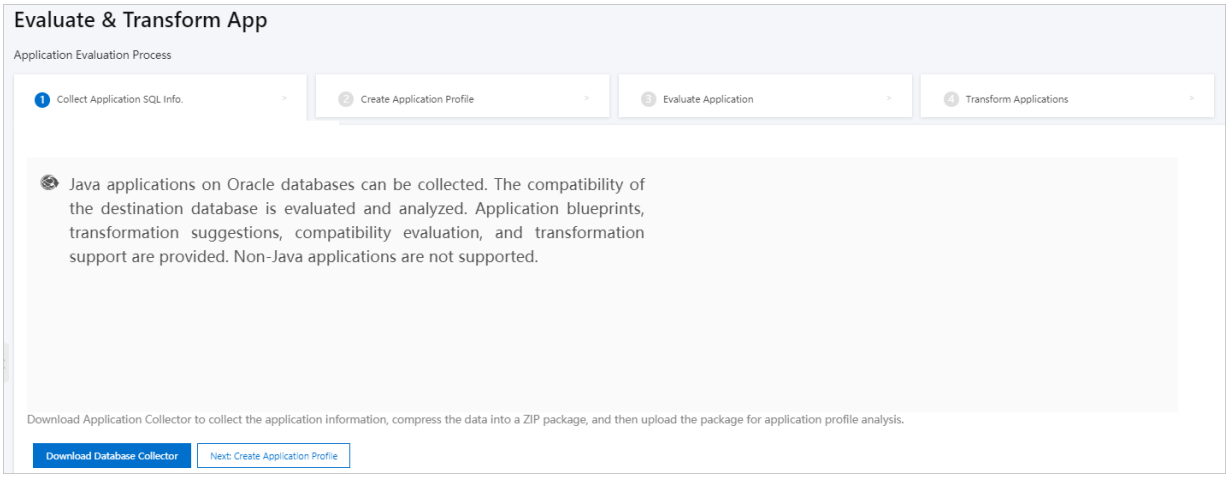
5.1. Overview

After your source Oracle database is evaluated and transformed, Advanced Database & Application Migration (ADAM) allows you to transform your applications. However, applications are much more complex than databases, and the application code may be frequently modified by developers. Therefore, transforming applications has become a pain point when users migrate databases to the cloud.ADAM provides the application evaluation and transformation feature to help you transform your applications.

1. Core features

- Provides application transformation items to migrate databases, call stacks, and SQL statements.
- Analyzes application usage information, such as framework and performance.
- Sorts out architectures to migrate large-scale clusters.

2. Procedure



1. Collect application data
2. Create an application profile
3. Evaluate an application
4. Perform static application transformation

> **Note**
>
> If you perform dynamic application migration, complete Steps 1, 2, and 3. If you perform static application migration, skip to Step 4.

5.2. Collect application information

Advanced Database & Application Migration (ADAM) allows you to collect data from Java applications that use **JDK 1.6 and later** to help you evaluate and analyze which features need to be improved. Non-Java applications do not support data collection.

Supported

- Collects SQL statements and call stack information of the application requests.
- Collects application performance information.

Not supported

- Data cannot be collected from non-Oracle databases or non-Java applications.
- Application requests that do not access any databases cannot be monitored. For example, the SQL statements and call stacks of API operations called outside of the collection period cannot be collected.
- Procedures such as SQL triggers that are not called by programs cannot be monitored.

If you have any questions about collecting application data, submit a ticket or send an email to adam_service@alibaba-inc.com. You must add the error message and contact information to the ticket.

The ADAM client for application data collection consists of a dynamic data collection module and a centralized data collection module.

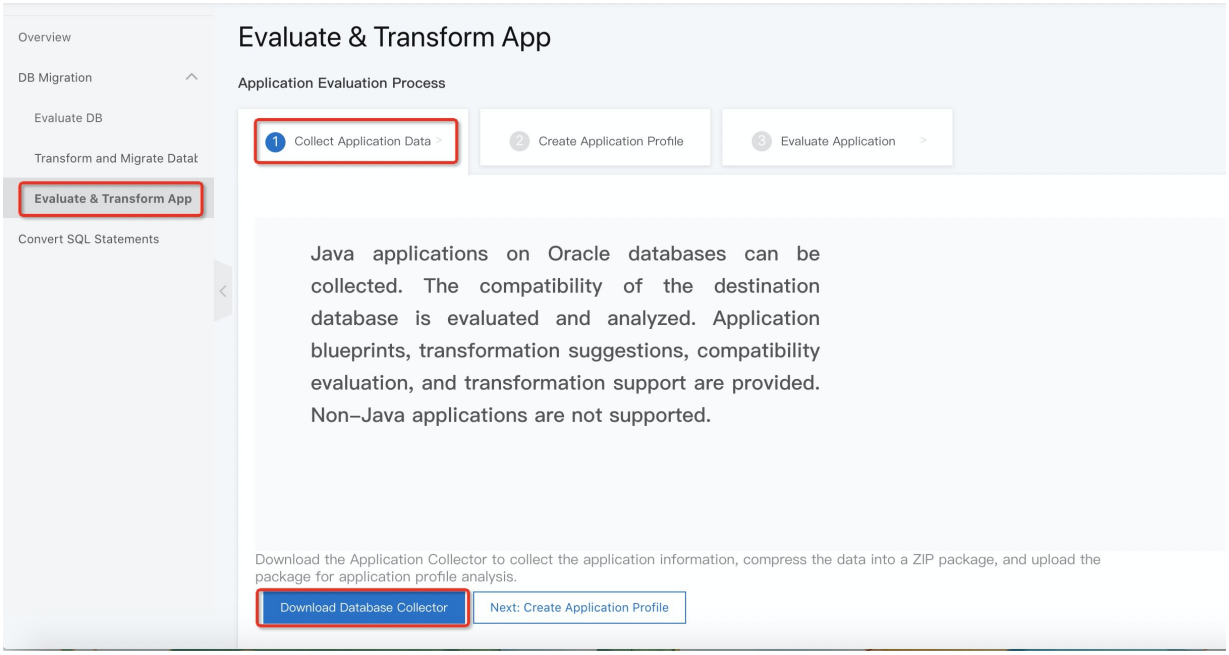
- Agent for dynamic data collection: collects basic information of the database that is requested by the application during the runtime of the application, such as the SQL statements, schemas, and call stacks of requests, as well as system information, performance information, and SQL hotspots.
- Collector for centralized data collection: collects, masks, and processes data from application agents.

Description

- Collected SQL statements are masked and the request parameters, and returned values of SQL statements are not collected.
- Data collection is read-only to prevent intrusions to applications.
- Data collection is automatically suspended during peak hours to restrict the memory usage within the specified range.

- Dynamic data collection of Java applications in in Tomcat, JBoss, and Oracle WebLogic containers that use JDK 1.6 or later is supported.

Download application data collection tools



Things to know before deployment

- Basic technical knowledge is required before you deploy the agent. Make sure that the tools are deployed by Java developers.
- SUN or Oracle JDK, OpenJDK 1.6 and later are supported. IBM JDK is not supported.
- The decompressed package contains the collector and javaagent directories. The collector is a unified collector deployed independently on a server that does not have online applications to avoid affecting online applications during data processing. The javaagent directory must be copied to the application server to be monitored and deployed with applications to collect data.
- Make sure that the collector and agent have system operation permissions. Windows applications do not need these permissions. You must use `chmod -R 775 collector/` to add system operation permissions to the directories in Unix or Linux.
- The collector acts as a server and can be deployed with 1 to 20 agents. An agent can only be deployed on a single application server. You can deploy agents on a few machines of a distributed application based on load balancing.
- Deploy the collector, and then deploy the agent. The application and the collector must be interconnected for centralized masking of data. The machine where the collector is deployed must use JDK 1.6 or later and have a Java Virtual Machine (JVM) memory size of more than 4 GB. The disk volume is determined by the number of monitored applications, monitoring duration, business activity, and the number and sizes of SQL statements. There is no explosive growth of data. You can estimate the data volume based on the data collected within half a day. In most cases, the data volume of a monitored application is less than 1 GB within seven days.
- The agent requires that the application to be monitored is deployed on a server that uses JDK 1.6 or later. The JVM heap size of the application to be monitored must be at least 300 MB. The supported service containers are Tomcat, JBoss, and WebLogic. Docker images can be deployed to a Container Service for Kubernetes cluster.
- The agent monitors the SQL statements and call stacks that are used to access Oracle databases. Make sure that all the operations are monitored during the monitoring period of the agent. The application must be monitored when recurring tasks are running. Otherwise, the collected data is incomplete.

5.3. Application profiling

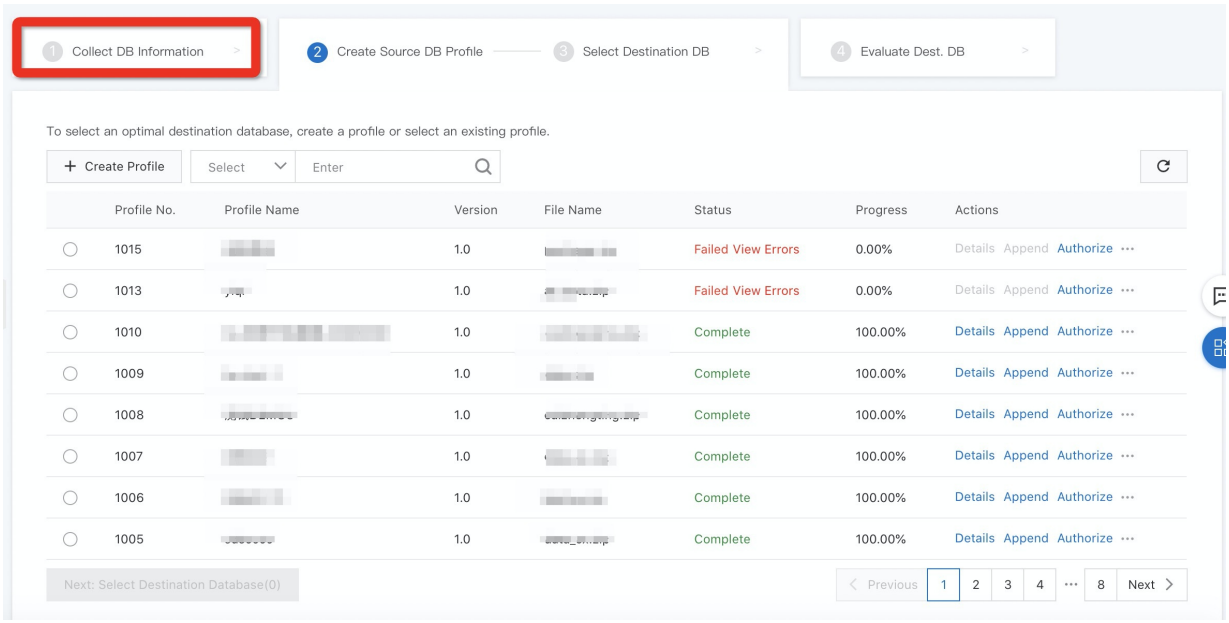
Advanced Database & Application Migration (ADAM) allows you to use application profiling to analyze the data collected from a single application by using intelligent analysis algorithms.

1. Procedure

1.1 After you collect data from an application, click Next: Create Application Profile.

1.2 Create an application profile

Click Create to go to the Create Profile page, as shown in the following figure.

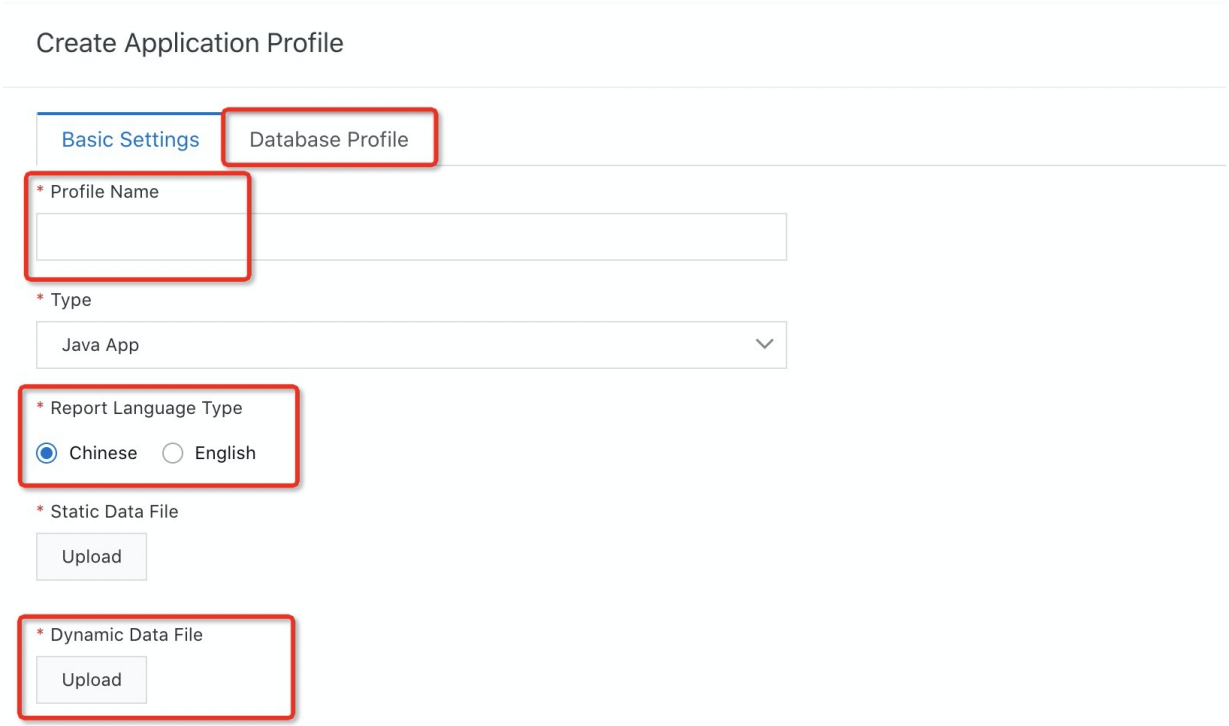


Enter the profile name, select Chinese or English as the report language, and upload a static or dynamic data file.

Static Data File: a WAR package or a collection package of static data.

Dynamic Data File: a ZIP package of collected data. A ZIP package can contain data collected for the same application from different IP addresses. You cannot package or add data of multiple applications to the same application profile.

Database Profile: required. Select profiles of Oracle databases accessed by the application.

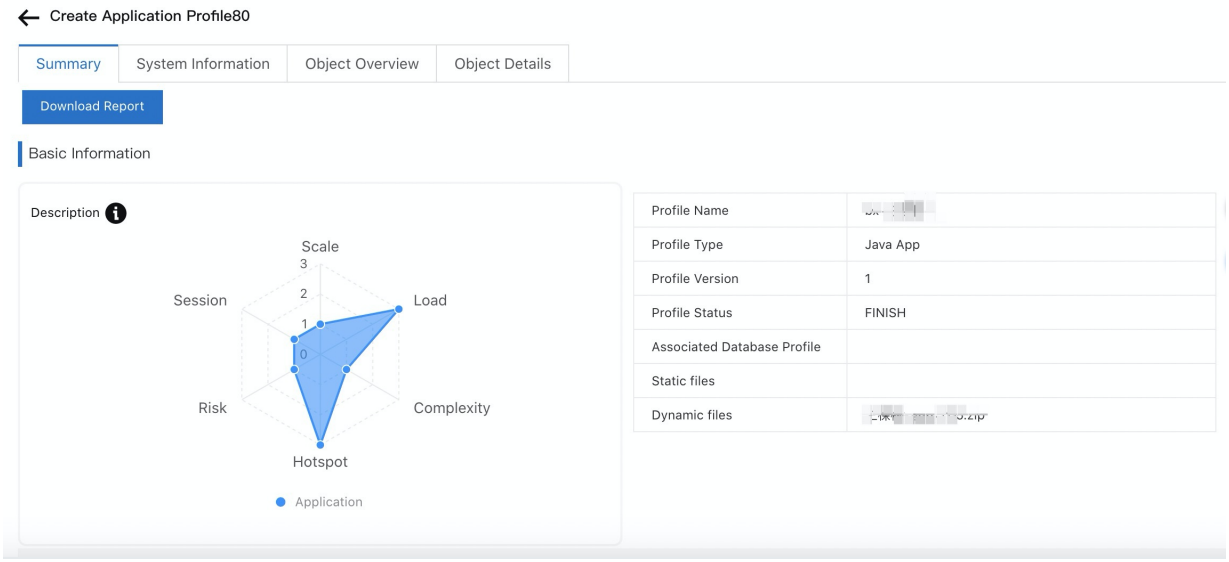


After setting all parameters, click Create. It takes 1 to 10 minutes for application profiling to analyze data.

1.3 Application profile details

An application profile consists of overview, system Information, object overview, and object details.

Overview:



Download Report: You can download the report of the application profile.

The basic information section includes a radar chart for application analysis and the basic information of the application profile.

ADAM uses a radar chart to show the overall performance of an application with respect to six dimensions.

Complexity is calculated based on conditions such as application scenarios and database features. This dimension indicates the application usage. A higher score indicates a more complex application that requires more transformations.

Session indicates the connection status of the application. A higher score indicates a larger number of connections to the application. This dimension is significant for configuring the connection pool when you transform the application.

Risk indicates the potential performance bottlenecks and stability risks of the application, especially the performance risks of SQL operations.

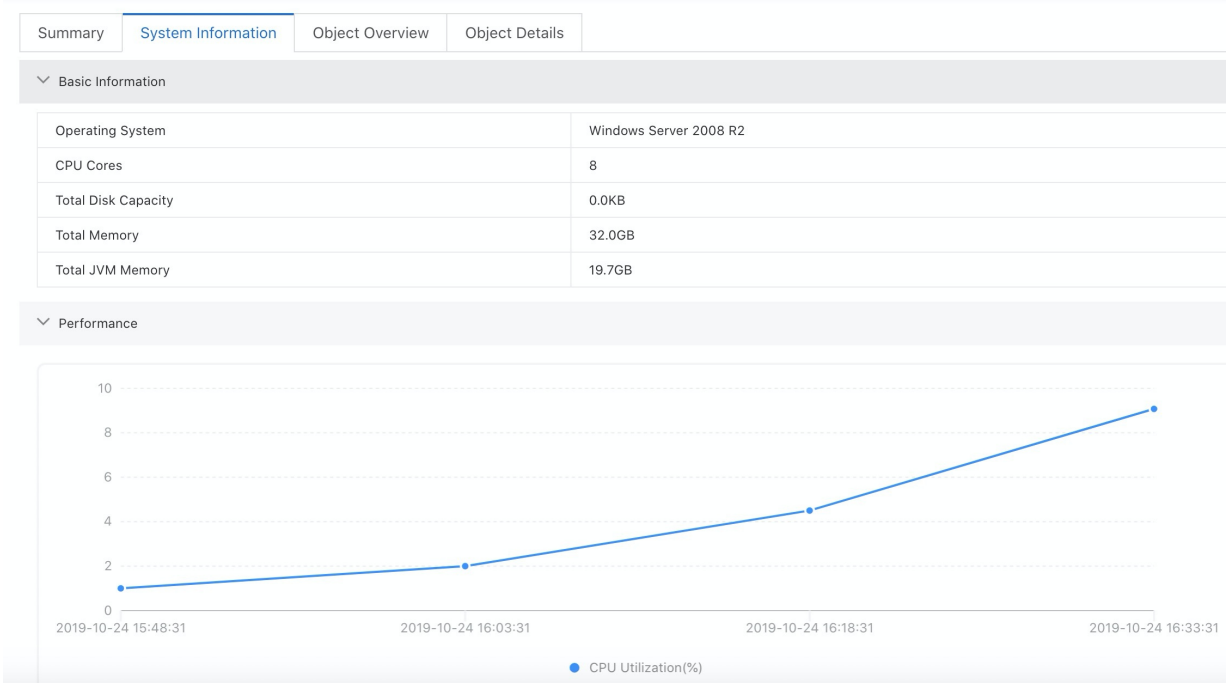
Hotspot indicates whether the database contains frequently accessed objects. A higher score indicates that the database to access contains objects that are frequently accessed.

Scale indicates the number of deployment units or instances of the application.

Load indicates the running performance of the application.

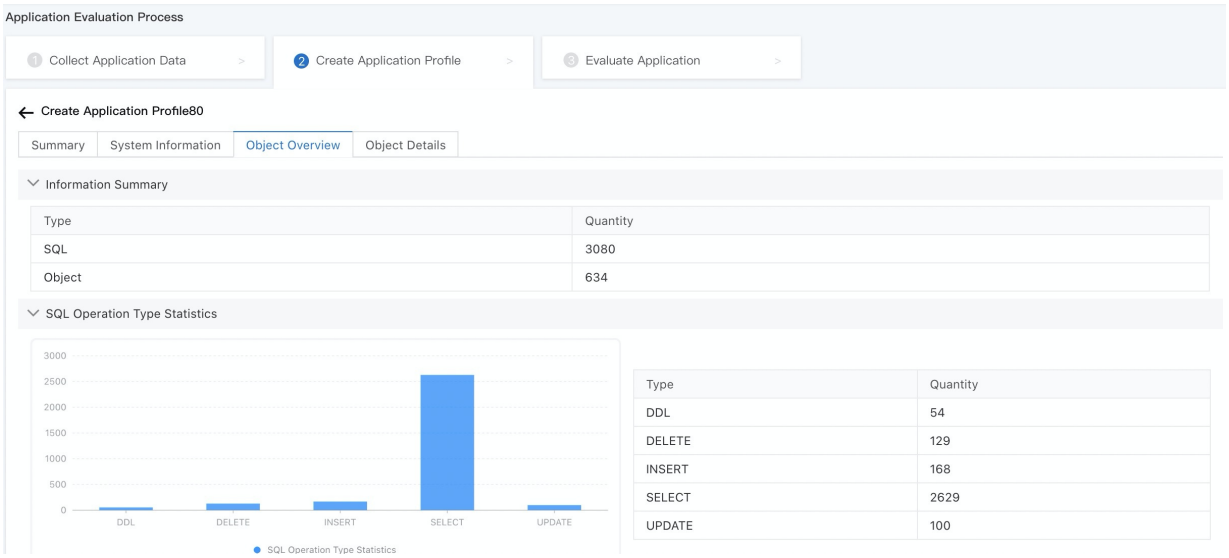
System Information

The System Information tab shows the system parameters that are collected by the ADAM application collector to help you evaluate the application running status.



Object Overview

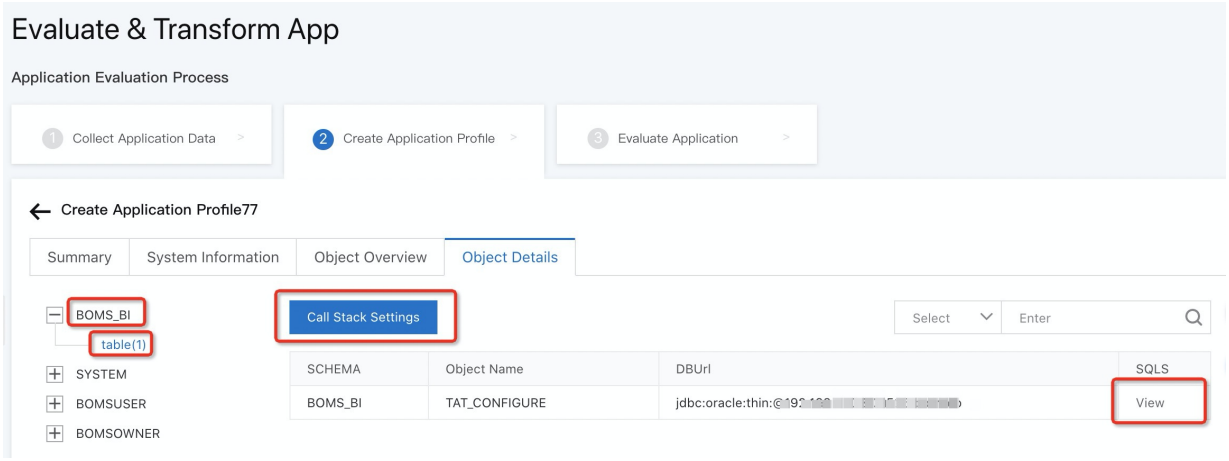
The Object Overview tab shows information about the SQL statements and database objects that the application collects and analyzes.



Object Details

The Object Details tab shows the relationship between database objects, SQL statements, and application code that are analyzed by ADAM. The section on the left side of the tab lists the database objects accessed by the application in the form of a tree diagram that uses schema and object type as dimensions. The section on the right side of the tab lists database objects and the corresponding SQL statements used to access the objects.

You can configure a blacklist when you configure a call stack to obtain required information about the call stack.



5.4. Evaluate applications

Advanced Database & Application Migration (ADAM) shows the combined evaluation results of applications and databases, and provides suggestions on focus areas of application transformation.

Prerequisites

Application profiles are created from all the collection packages.

Background information

Database and application transformation has the following challenges:

- Estimation of application transformation cost
- Database migration planning
- Transformation of applications that have not been maintained for a long time

Create an application evaluation project

1. Enter the basic information of the applications. Select the type and version of the database that you want to evaluate.
2. Select one or more application profiles that you want to evaluate.
3. Select the database evaluation project.

> **Note**
>
> 1. You must select database profiles that are in the Complete state. These profiles are associated with the application profiles that you selected in Step 2.
> 2. You can create an evaluation project only when all associated databases are in the Complete state.
> 3. You must select database profiles that are in the Complete state. These profiles are associated with the application profiles that you selected in Step 2.
> 4. You can create an evaluation project only when all associated databases are in the Complete state.

Application evaluation details

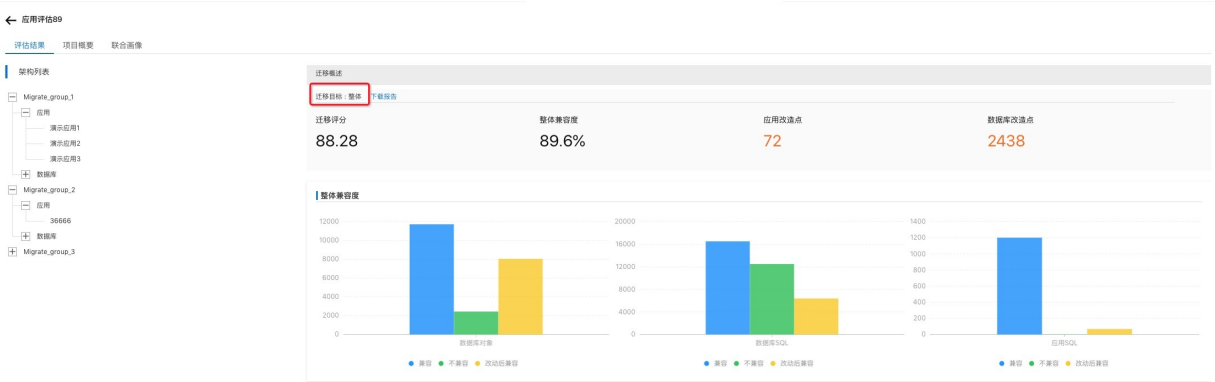
Application evaluation details contain the evaluation results, project overview, and joint profiles.

Evaluation results

Evaluation results are displayed based on the overall results, migration groups, and application nodes.

Evaluation result overview

The evaluation result overview shows the combined evaluation results of applications and databases.

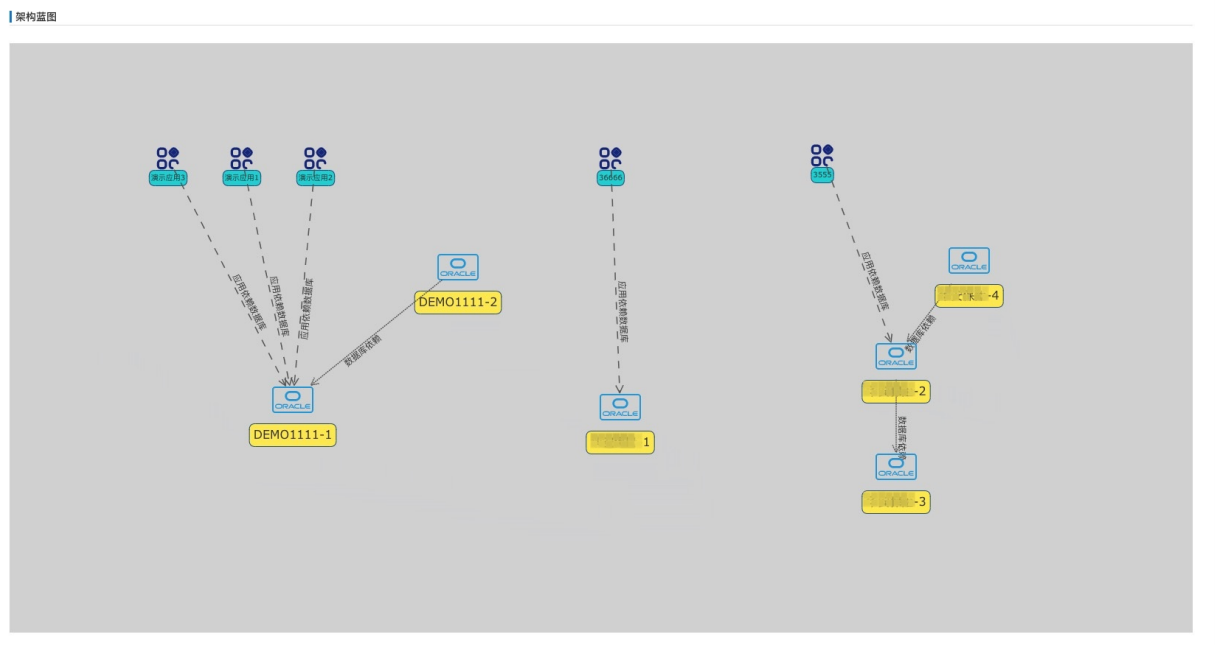


Architecture list: ADAM groups the overall architecture based on the relationship between databases and applications by using intelligent algorithms. A migration group is the smallest subset of a migration unit. Therefore, migration of a migration group will not affect the data dependencies of other migration groups.

Migration score: ADAM scores the difficulty of migration and transformation tasks. A higher score indicates a lower transformation cost for applications. The migration score is subject to the integrity of collected data. We recommend that you consider the migration cost based on your business needs.

Overall compatibility: The metric shows the compatibility of application SQL statements and database objects. SQL statements collected from databases are subject to the database system. Therefore, the compatibility of SQL statements is not used for reference.

Architecture blueprint: the architecture blueprint shows the status of migration groups in a topology.

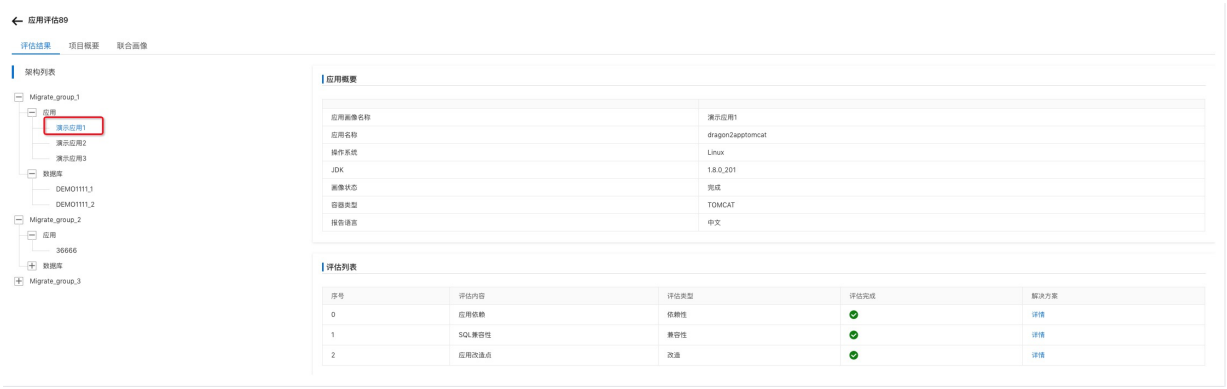


Migration group

A migration group is the smallest subset of a migration unit. Therefore, migration of a migration group will not affect the data dependencies of other migration groups.

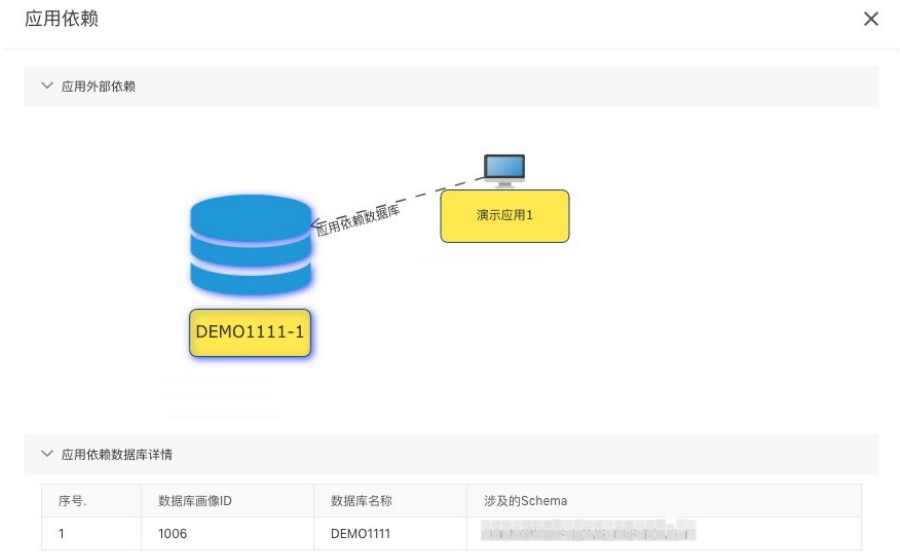
A migration group includes **application nodes** and **database nodes**. You can click an application node to view its evaluation and transformation details.

The evaluation results of a single application includes application dependencies, SQL compatibility, and focus areas of application transformation.



Application dependencies

Application dependencies show the information of databases on which an application node depends. The minimum dependent objects of applications are schemas.



SQL compatibility

SQL compatibility shows the compatibility of application SQL statements, and provides specific transformation rules and focus areas for each SQL statement.

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

Compatible: shows the number of SQL statements that can be executed in the destination database without transformation.

Compatible after Conversion: shows the number of SQL statements transformed by ADAM, which can be executed in the destination database after they are replaced in the application.

Incompatible: shows the number of SQL statements that cannot be executed in the destination database. You must modify the application code or SQL statements based on specific transformation suggestions.

Focus areas of application transformation

ADAM shows focus areas of application transformation and provides suggestions on them. You can check focus areas of application transformation briefly or in detail.

5.5. Perform static application transformation

Advanced Database & Application Migration (ADAM) allows you to scan static code to identify the SQL statements that need to be transformed. ADAM transforms the SQL statements that can be automatically replaced, and provides suggestions if some SQL statements cannot be automatically replaced.

Prerequisites

The type of the source database is Oracle, Db2, or Teradata.

Create a transformation project

1. Log on to the ADAM console.
2. In the left-side navigation pane, click **Evaluate & Transform App**.
3. In the Transform Applications step, click **Create Transformation Project**.
4. In the **Create Transformation Project** panel, specify **Source Database Type**, **Destination Database Type**, **Destination Database Version**, and **Architecture**, upload a data file, and then click **Create**.
5. If you create a migration project, you need to run the built-in analysis program of ADAM. This process takes 1 to 10 minutes.

> Note
> - If the destination database is a PolarDB O Edition database, automatic transformation is supported.
> - If the destination database is an ApsaraDB RDS for MySQL, AnalyticDB for PostgreSQL, or PolarDB-X database, SQL statements can be displayed, but cannot be automatically transformed.

Details of static application transformation

The details of the static application transformation include the following sections in the console: **Project Summary**, **Transformation Overview**, and **Transform Applications**.

Project Summary

The Project Summary section describes the basic information of a transformation project.

Transformation Overview

The Transformation Overview section displays the types and transformation methods of code blocks.

Transformation types of code blocks

- **Not Required**: The code blocks can run in the destination database without transformation.
- **Automatic**: The code blocks are transformed by ADAM. You only need to replace them and run the transformed code blocks in the destination database.
- **Manual**: The transformation requirements of the code blocks are listed. You must manually transform these code blocks.
- **SQL Unrecognized**: The SQL statements in the code blocks cannot be identified because the SQL statements are invalid or code blocks are run in a special way.

Types of code blocks

- **select**: the code blocks of the SELECT type
- **insert**: the code blocks of the INSERT type
- **update**: the code blocks of the UPDATE type
- **procedure**: the code blocks of stored procedures
- **delete**: the code blocks of the DELETE type
- **statement**: the code blocks executed by statements

Transform Applications

Click **Details** in the Actions column to view the transformation details of code blocks.

6.Schema revision

6.1. FOR UPDATE OF is not supported

Background information

Oracle supports Select for update and For update of column syntaxes, which are not different for a single table. They are both locking related rows that meet the conditions For a single table. For multiple tables, update locks the rows that match the conditions in multiple tables. For update of locks the rows in the corresponding tables based on the conditions. Examples:

- Single Table

  select * from test where id =10 for update

  select * from test where id=10 for update id

  > **Note** Lock the row with id=10.

- Multiple tables

  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

  > **Note** Lock the rows test and t1 with id=10.

New solution

Currently, PolarDB only supports the syntax For update and does not support the syntax For For update of. You can adjust the syntax to For update as appropriate based on business situations.

6.2. Automatic conversion of the Sample statement is not supported

Background information

Oracle will use the Sample syntax for sampling query, such as: `select * from AAA sample block(name)` .

New solution

PolarDB supports sampling queries using the TABLESAMPLE system (parameter) and TABLESAMPLE bernoulli (parameter) methods.

BERNOULLI and the parameters passed in by the SYSTEM sampling method indicate the percentage of the sampling table, which ranges from 0 to 100. The BERNOULLI method scans the entire table and selects or ignores rows with a specified probability. The SYSTEM method performs block layer sampling, and each block has a specified opportunity to be selected. All rows in the selected block are returned. When a small sampling percentage is specified, the SYSTEM method is much faster than the BERNOULLI method, but the former method may return Table sampling with poor randomness due to clustering effect.

Examples

```
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)
```

6.3. Bitmap indexes are not supported

Background information

A Bitmap index is a special type of index used in Oracle. It is suitable for low-cardinality fields and increases the query speed. Due to the large lock granularity, it is not suitable for frequently updated fields.

For low-cardinality field in PolarDB O Edition also can create a Btree index, create a Btree index can not only improve the search speed and does not occur due to the Oracle considerable line performance degradation of the problem.

New solution

Evaluate whether a field is low-cardinality based on the field data. If a query statement with good selectivity and this condition in the business SQL statement is used, we recommend that you create a B- tree index. If the selectivity is not good, first evaluate whether an index needs to be created.

Examples

- Example 1

The name column in Table A has good selectivity. A Bitmap index is available in Oracle. The B- tree index can be created after data is migrated to PolarDB O Edition.

- Example 2

  The status column in Table A is poorly selective, for example, the value of 90% is 1 and the value of 10% is 0. In all business queries, the condition is status=1. In this case, you can not create an index. If status=0 query exists in your business, we recommend that you create a B- tree index.

6.4. Cluster index is not supported

Background information

A Cluster in Oracle is a table that shares a common column and stores related data in the same block. When a Cluster table is used, a single data block can contain rows from multiple tables. This reduces disk IO for tables where Joins frequently occur.

Tables that share columns are not supported in PolarDB O Edition, but PolarDB O Edition provide the ability to reorganize heap table data by a particular index.

Solutions

Cluster syntax

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

The Cluster adjusts the order of the physical storage of the data and re-sorts the data by the specified index. Note that the Cluster operation is one-time, and will not automatically adjust for subsequent table changes.

Examples

```
#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;
```

6.5. The exception type is not supported

Background information

The PolarDB O Edition does not support the Oracle exception type. If the Oracle exception type is used, the syntax is incorrect:

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

Oracle exception types can be overwritten in PolarDB O Edition. Modify the exception types based on the exception handling methods of the PolarDB O Edition.

New solution

Exception handling syntax:

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

- If no exception occurs, the program executes all `statements` normally.
- If an EXCEPTION occurs within the `statements`, exceptions are redirected to the EXCEPTION section. The system matches the first `condition` in the WHEN List. If the match is not successful, the exception continues to be thrown upwards. `condition` can be any of the exception names in the appendix.

> **Note** For more information about the exception types supported by PolarDB O Edition, see Appendix.

Examples

Example of exception handling for division-by-zero errors:

```
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;
```

Appendix

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

Error Code	Condition Name
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
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	
20000	case_not_found
Class 21 - Cardinality violation	
21000	cardinality_violation
Class 22 - Data exception	
22000	data_exception
2202E	array_subscript_error

Error Code	Condition Name
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
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

Error Code	Condition Name
22P04	bad_copy_file_format
22P05	untranslatable_character
2200L	not_an_xml_document
2200M	invalid_xml_document
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
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

Error Code	Condition Name
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
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
42803	grouping_error
42P20	windowing_error
42P19	invalid_recursion

Error Code	Condition Name
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
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

Error Code	Condition Name
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
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. The errors that are external to PostgreSQL.	
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
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

Error Code	Condition Name
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	
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

6.6. The AGGREGATE keyword is not supported

Background information

Oracle supports using AGGREGATE to create a custom AGGREGATE function, such as creating an AGGREGATE function that queries the second largest value:

```
-- Create an 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)RETURN NUMBER,
  member FUNCTION ODCIAggregateTerminate(self IN secmax_context,returnValue OUT NUMBER,flags IN NUMBER) RETURN NUMBER
);
-- Implement the 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 number is
begin
  if value > self.firmax then
    self.secmax := self.firmax;
    self.firmax := value;
  elsif value > self.secmax then
    self.secmax := value;
  end if;
  return ODCIConst.Success;
end;
member function ODCIAggregateTerminate(self IN secmax_context, returnValue OUT number, flags 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) return 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;
-- Create an aggregate function.
CREATE FUNCTION SecMax (input NUMBER) RETURN NUMBER PARALLEL_ENABLE AGGREGATE USING secmax_context;
select secmax(id) from test;
```

Solutions

The syntax for creating a custom aggregate function in Polardb is as follows:

```
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 } ]
)
```

Examples

The example is to customize an aggregate function to calculate the maximum value.

```
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)
```

6.7. Calling other language code is not supported in PL/SQL

Background information

Currently, functions and PolarDB O Edition written in other languages are not supported in stored procedure.

Solutions

Use the SPL language supported by PolarDB O Edition, or PL/pgSQL supported by Postgresql to implement the corresponding business logic.

Examples

- Functions implemented in JAVA:

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

- Rewrite the function as SPL syntax:

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

  For more information, see Create a function.

- Or a function that overrides PL/pgSQL syntax:

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

  For more information, see Structure of PL/pgSQL.

6.8. PARALLEL_ENABLE is not supported

Background information

In Oracle, parallel processing can be enabled for custom aggregate functions. In PolarDB O Edition, this is also supported, but with some differences in syntax. This section describes how to create custom aggregate functions and enable parallelism in the next PolarDB O Edition.

Solutions

Syntax for creating a custom aggregate function in PolarDB O Edition:

```
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 } ]
)
```

Examples

Regular aggregate functions are supported in PolarDB O Edition. If you want to implement special features, you can customize the relevant functions as needed (sfunc, stype, FINALFUNC).

For example, specify symbolic join aggregations, which are applicable to most scenarios at the same time:

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

The result is:

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

6.9. The PIPELINED keyword is not supported

Background information

pipelined is commonly used in Oracle to return multiple records in the stream. The pipelined syntax is not supported in PolarDB O Edition. However, you can use setof to implement the same function.

Solutions

For the Pipelined syntax in Oracle, when creating a function in PolsrDB-O, the Setof syntax is used to replace:

```
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 | VOLA
TILE | [ NOT ] LEAKPROOF | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEF
INER | 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, composite, or domain type, or can reference the type of a table column. Depending on the implementation language it might also be allo
wed to specify “pseudo-types” such as cstring. If the function 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 must agree with the result type implied by the output parameters: RECO
RD if there are multiple 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 single item. The type of a column is referenced by writing table_name.column_name%TYPE.
```

For more information, see https://www.postgresql.org/docs/11/sql-alteraggregate.html.

Examples

- In Oracle

```
create or replace function split ( p_list varchar2, p_del varchar2 := ',' ) return split_tbl pipelined is l_idx pls_integer; l_list varchar2(32
767) := 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;
```

- In PolarDB O Edition

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

6.10. Column names reserved by the PolarDB system

Background information

Reserved system column names cannot be used in PolarDB O Edition: ctid, oid, cmin, cmax, xmin, and xmax. The following error is returned if a system column name is used:

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

Solutions

- You can handle conflict column names by changing column names.
- Create a shadow table with a different name from the original table, and modify the conflicting column names.
- Create a view with the same name as the original table, and map the column names to those of the original table.

Examples

- Modify the name of a column

  Original column name:

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

  Modify the key.

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

- Create a table

  The original table is:

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

  Create a new table:

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

  Create a view:

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

6.11. Trigger does not support non-DML events

Background information

PolarDB common triggers do not support DDL statements, such as DROP, CREATE, and ALTER statements. You must change the statement to an event trigger in PolarDB.

New solution

The following table describes the syntax of PolarDB event triggers.

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

> **Note**
>
> - Supported events include ddl_command_start, ddl_command_end, table_rewrite, and sql_drop.
> - ddl_command_start event occurs right before the execution of a CREATE, ALTER, DROP, SECURITY LABEL, COMMENT, GRANT, or REVOKE command. The system does not check whether the affected objects exist before the event trigger is raised.
> - ddl_command_end event occurs right after the execution of the same set of commands.
> - The sql_drop event occurs before the event trigger is ddl_command_end for any drop of database objects.
> - The table_rewrite event occurs before the TABLE is overridden by certain actions of the command ALTER TABLE and ALTER TYPE.

Examples

In Oracle:

```
create or replace trigger apps_no_ddl
before create or alter or drop or truncate
on database
begin
raise_application_error(-20001, 'The object of the app user is not allowed to operate by using DDL ');
end;
```

In PolarDB:

```
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();
```

6.12. Aggregate functions do not support the keep keyword

Background information

The use of the aggregate function does not support the keep keyword. Example:

```
Sales table:
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
At this time, there is a new requirement, and you want to view the record with the earliest sales record time and the smallest sales volume in De
partment D02.
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
```

Solutions

You can replace the keep Syntax by rewriting Sql statements.

Examples

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

6.13. The SYS_CONTEXT() function only supports SESSION_USER,CURRENT_USER,CURRENT_SCHEMA,HOST,IP_ADDRESS and SERVER_HOST as the parameters

Background information

Oracle officially recommends using the SYS_CONTEXT function to obtain system variables. PolarDB O Edition, some features of the SYS_CONTEXT function are supported.

Solutions

The following table describes how to use the SYS_CONTEXT function.

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

In PolarDB O Edition, only the following attributes are supported:

```
SESSION_USER,CURRENT_USER,CURRENT_SCHEMA,HOST,IP_ADDRESS,SERVER_HOST
```

If you need to use other attributes, you can use self-built functions to achieve the same function:

```
create or replace function userenv(anynonarray) returns anynonarray as $$
declare
begin
  case lower($1)
  when 'configure as now'
    return custom function ();
  when 'configure as now'
    return custom function ();
  else
    return null;
  end case;
end;
$$ language plpgsql strict;
```

Examples

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

6.14. USERENV is not supported

Background information

Oracle uses the USERENV function to obtain current session variables. USERENV is an Oracle backward compatible function. Oracle officially recommends using the SYS_CONTEXT function for substitution. PolarDB O Edition, you can use the SYS_CONTEXT function to obtain session variables.

Solutions

Syntax of the SYS_CONTEXT function:

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

In PolarDB O Edition, the attribute support:

```
SESSION_USER,CURRENT_USER,CURRENT_SCHEMA,HOST,IP_ADDRESS,SERVER_HOST
```

If you need to use other attributes, you can use self-built functions to achieve the same function:

```
create or replace function userenv(anynonarray) returns anynonarray as $$
declare
begin
  case lower($1)
  when 'configure as now'
    return custom function ();
  when 'configure as now'
    return custom function ();
  else
    return null;
  end case;
end;
$$ language plpgsql strict;
```

Examples

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

6.15. An element type of the Varray type cannot be a type that is just defined

Background information

In Oracle, a Varray can be nested in the definition, that is, an already defined Varray can be referenced in the Varray definition. For example:

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

In PolarDB O Edition, the definition of the Varray type is supported, but the syntax of the nested definition is not. In this case, you need to make simple modifications.

New solution

Definition of Vory in PolarDB O Edition:

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

> **Note** The specific type after OF only supports datatype and objtype, varraytype is not supported.

Examples

- Transform the Vory with nested definitions:

```
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;
```

6.16. dbtimezone

Background information

`timestamp at time zone dbtimezone` in the `dbtimezone` are not supported in PolarDB O Edition.

```
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 d
ual; CURRENT_TIMESTAMP ---------------------------------------------------------------------------- 04-AUG-20 06.04.42.125059 PM +08:00
```

Solutions

We recommend that you change the time zone to a specific value based on the dbtimezone settings in the ORACLE database, for example, +00:00. The time zone of PolarDB can be modified based on the timezone parameter and can be adjusted at the server level or the session level.

```
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)
```

For more information, see https://www.postgresql.org/docs/11/functions-datetime.html.

6.17. NESTED TABLE

Background information

For Oracle nested table detailed features, see http://www.orafaq.com/wiki/NESTED_TABLE.

NESTED TABLE is an Oracle data type used to support columns that contain multi-value attributes. In this example, the column can accommodate the entire sub-TABLE.

Create a TABLE with a NESTED TABLE:

```
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;
```

Insert data into a table:

```
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;
```

Select from NESTED TABLE:

```
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')
```

Unnest child table:

```
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.
```

Compatible with PostgreSQL Nested Table

PostgreSQL uses array and composite types to meet the same scenario requirements.

1. Create a composite type.

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

   > **Note** If this type has already been created in the system or a TABLE to be used has already been created, you do not need to create it again.

2. Create a Nested Table.

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

   > **Note** thisisnesttable 1 as the Nested Table of the hello Table

3. Inserts data.

   ```
   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 Or use the row construction method insert into hello values ( 1, 'test', (array [ row(1,2,'hello',now()), row(1,3,'hello',now()) ] )::thisisnesttable1[] );
   ```

   > **Note** Multiple rows are stored as an array, and the maximum limit for a nested table is 1GB (that is, the upper limit of storage for PostgreSQL varying type).

   For more information, see https://www.postgresql.org/docs/11/sql-expressions.html#SQL-SYNTAX-ROW-CONSTRUCTORS.

4. Query

   ```
   postgres=# select * from hello ; id | info | nst ----+------+---------- 1 | test | {"(1,2,abcde,\"2018-01-01 12:00:00\")","(2,3,abcde123,\"2018-01-01 12:00:00\")"} (1 row)
   ```

5. You can use unnest to unnest the content of the Nested Table.

   ```
   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)
   ```

6.18. REGEXP_LIKE function

Background information

Oracle supports the REGEXP_LIKE function, but PolarDB O Edition does not.

It is specifically used in Oracle as follows:

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

New solution

Using POSIX regular expressions for rewriting `,similar to` and `'~'` can support regular expression matching.

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

For more information, see https://www.postgresql.org/docs/11/functions-matching.html#FUNCTIONS-POSIX-REGEXP

6.19. EXTRACTVALUE function

Background information

Supports the EXTRACTVALUE function in Oracle to parse xml. The error message returned because the function is not supported in the PolarDB O Edition.

```
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
```

Solutions

Use xpath functions to achieve the same function. For more information, see 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'); IN
SERT 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=>
```

6.20. Unpivot columns to rows

Background information

UNPIVOT syntax in Oracle:

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

Example:

```
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
```

New solution

You can convert the rows and columns in the Polardb-O by using the interface of the Crosstab function.

Examples

An example of changing a row is as follows:

```
with a as ( -- A corresponds to the original data (that is, data whose columns need to be converted to rows)
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(
  -- This is the data source for row and column transformation.
  -- The sorting field is a group by field, the last field is the transformed content field, and the second field of the derivative is the row an
d column transformed field (the content is an enumeration, such as the month).
  -- (All corresponding enumerated values must be extracted in the next parameter.)
  $$select jsonb_build_object('seller', seller, 'se_year', se_year) as js, se_month, sum(se_amount) from tbl_sellers_info group by 1,2 order by 1
$$,
  -- Values are extracted as columns in rows converted from rows. This here represents the month, which is the value of se_month
  -- Or (select * from (values('jan'),...('dec')) t(se_month))
  'select distinct se_month from tbl_sellers_info order by 1'
)
as -- crosstab output format
(js jsonb, -- one or more fields corresponding to the order by clause in the SQL statement of the first parameter
   Jan numeric, -- the enumerated value of the second field of the corresponding derivative in the first parameter SQL, (row to column)
   feb numeric,  -- ... Same as above
   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: use jsonb to merge multiple columns into one column and expand the columns by using jsonb_each.
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;
```

6.21. DBMS_XMLGEN function

Background information

Oracle supports the DBDBMS_XMLGEN package and its functions for manipulation of xml-type data.

```
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.GETX
ML(1) ------------------------------------------------------------------------------- <? xml version="1.0"? > <ROWSET> <ROW>  <A>10</A>  <B>firs
t line</B> </ROW> <
```

Solutions

Although PolarDB O Edition does not support the DBMS_XMLGEN package and its functions, PolarDB provides XML-related functions, such as xmlagg, xmlroot, xmlforest, xmlelement, xmlconcat, and xmlcomment. You can use these functions to implement similar functions.

```
 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)
```

For more information, see https://www.postgresql.org/docs/11/functions-xml.html.

6.22. DBMS_METADATA.GET_DDL is not supported

Background information

You can obtain the table creation statements in Oracle by DBMS_METADATA.GET_DDL functions, but PolarDB O Edition is not supported.

New solution

```
an-> \d+ t1
                                        Table "public.t1"
 Column |         Type          | Collation | Nullable | Default | Storage  | Stats target | 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
```

Note `\d+` can see the table structure, but not the table creation statement.

- This can be achieved by creating a function:

```
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);            +
```

- You can view the client in a management tool such as pgadmin and dbeaver. dbeaver is used as an example:

6.23. Date - Date result incompatibility

Background information

In Oracle, the result is of the floating-point type. In PolarDB O Edition, the result is of the Interval type. Generally, this type of difference causes syntax errors in the time computing part of the business SQL.

New solution

In PolarDB O Edition, the syntax of Date-type subtraction can be adapted. The Transformation idea is to use the extract function to convert the Interval type to the floating point type, to make the result consistent with Oracle.

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

For more information, see https://www.postgresql.org/docs/11/functions-datetime.html.

Examples

The following example shows how to use a function for transformation:

```
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; -- Transform to select time_between(sysdate, date '2
020-06-28'); 1.29990540226852
```

6.24. Forall and Bulk Collect

Background information

PolarDB O Edition does not support FORALL statements except for the "FORALL index IN lower_bound .. upper_bound" type.

How KEDA works

It is necessary to switch between handler and SQL procedural language Oracle PL/SQL. If it is a large LOOP, the performance will be seriously degraded if more is switched.

Therefore, for processing scenarios where PL/SQL needs to call SQL multiple times, Oracle has come up with the bulk collect processing method. For example, when a user submits an array, PL/SQL requires that all elements of the array be inserted into the table, the value in the table be updated, or the value in the table be deleted.

Solutions

Similar to the batch insertion usage of Oracle FORALL, an array is used to represent conditions. Another array indicates that if there are multiple conditions or values, the record array or hstore(Key-VALUE type) array can be used to represent values.

```
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;
$function$;
```

6.25. Interval partition

Background information

The Interval partition feature is added to the range partition table of Oracle 11g. For this type of range partitioning, you do not need to define MAXVALUE. Instead, Oracle dynamically assigns new partitions to contain data exceeding the range based on the step size.

```
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
```

Solutions

- PolarDB O Edition currently partition type drop-down list partitioning and range partitioning (but does not support Interval partition), but can be the Interval partition according to the step size into the scope: Then, create corresponding partition table regularly in advance through the job.

```
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)
```

- Define a job function that periodically creates partitions

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

- Partition table added successfully.

```
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=>
```

> **Note** poarDB-O support creating a job, you can automatically create the corresponding partition in advance on a regular basis.

7.Authorize a RAM user to log on to the ADAM console

Advanced Database & Application Migration (ADAM) is integrated with Alibaba Cloud Resource Access Management (RAM). This topic describes how to authorize a RAM user to log on to the ADAM console.

Prerequisites

Before a RAM user can log on to the ADAM console, the RAM user must meet the following prerequisites:

- An AccessKey pair is created for the RAM user. An AccessKey pair consists of an AccessKey ID and an AccessKey secret.
- Console logon is enabled. A logon username and a password are specified for the RAM user.

For more information, see Create a RAM user.

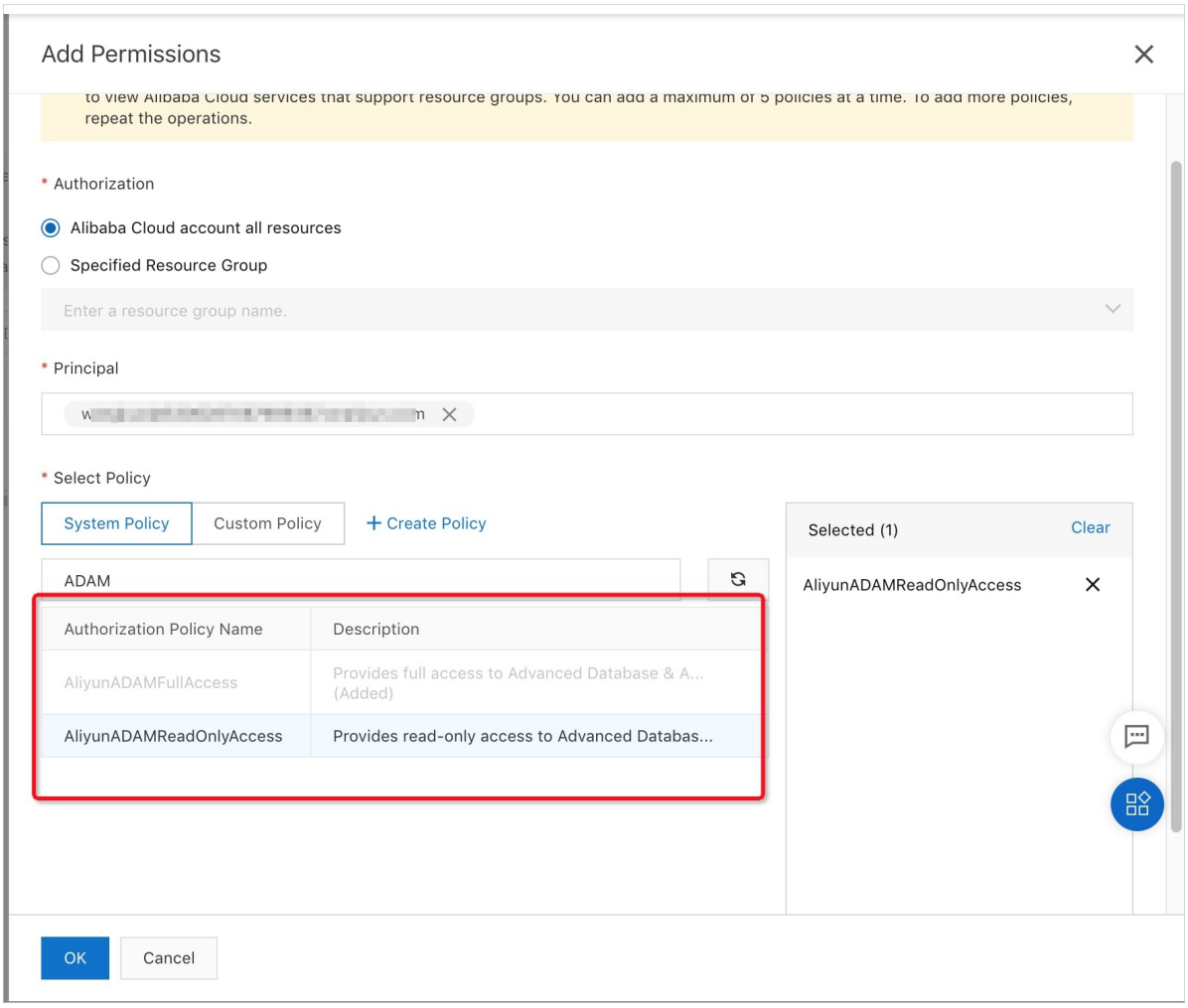
Context

Before a RAM user can use the ADAM console, you must grant required permissions to the RAM user. You can grant full access permissions or read-only permissions to the RAM user in the ADAM console.

> **Note** RAM users of an Alibaba Cloud account inherit the data of the account. The data generated by a RAM user is accessible to all RAM users of the Alibaba Cloud account.

Procedure

1. Log on to the RAM console by using your Alibaba Cloud account.
2. In the left-side navigation pane, choose **Identities > Users**.
3. On the Users page, find the RAM user and click **Add Permissions** in the Actions column.
4. In the **Add Permissions** dialog box, specify the **Authorization** and **Select Policy** parameters.
   You can add the following permission policies for the RAM user:
   - AliyunADAMFullAccess: grants the RAM user full access permissions on ADAM.
   - AliyunADAMReadOnlyAccess: grants the RAM user read-only permissions on ADAM.



5. Click **OK** to complete the authorization. Then, you can log on to the ADAM console as a RAM user.