ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

Data Transmission Service

User Guide

Product Version: v3.16.2

Document Version: 20220916

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.What is DTS?

Data Transmission Service (DTS) is a data service that is provided by Alibaba Cloud. DTS supports data transmission between various types of data sources, such as relational databases and big data systems.

Features

DTS has the following advantages over traditional data migration and synchronization tools: high compatibility, high performance, security, reliability, and ease of use. DTS allows you to simplify data transmission and focus on business development.

Feature	Description
Data migration	You can use DTS to migrate data between homogeneous and heterogeneous data sources. This feature applies to the following scenarios: data migration to Alibaba Cloud, data migration between instances within Alibaba Cloud, and database splitting and scale-out.
Data synchronization	You can use DTS to synchronize data between data sources. This feature applies to the following scenarios: disaster recovery, data backup, load balancing, cloud BI systems, and real-time data warehousing.
Change tracking	You can use DTS to track data changes from user-created MySQL databases, ApsaraDB RDS for MySQL instances, Cloud Native Distributed Database PolarDB-X instances (formerly known as DRDS), and user-created Oracle databases in real time. This feature applies to the following scenarios: cache updates, business decoupling, asynchronous data processing, synchronization of heterogeneous data, and synchronization of extract, transform, and load (ETL) operations.

2.Log on to the DTS console

This topic describes how to log on to the Data Transmission Service (DTS) console. Google Chrome is used in this example.

Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password that you use to log on to the Apsara Uni-manager Management Console from the operations administrator.

   > **Note** The first time that you log on to the Apsara Uni-manager Management Console, you must change the password of your account. For security purposes, your password must meet the minimum complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, including exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%)

3. Click **Log On**.
4. In the top navigation bar, choose **Products > Data Transmission Service**.
5. Select an **organization** and **region**, and then click **DTS**.

3.Data migration

3.1. Supported databases and migration types

You can use Data Transmission Service (DTS) to migrate data between homogeneous and heterogeneous data sources. Typical scenarios include data migration to Alibaba Cloud, data migration between instances within Alibaba Cloud, and database splitting and scale-out. This topic describes the database types, database versions, or migration types that are supported by the data migration feature.

Source database	Destination database	Migration type
• User-created MySQL database Version 5.1, 5.5, 5.6, 5.7, or 8.0 • ApsaraDB RDS for MySQL All versions	User-created MySQL database Version 5.1, 5.5, 5.6, 5.7, or 8.0	• Schema migration • Full data migration • Incremental data migration
	ApsaraDB RDS for MySQL All versions	• Schema migration • Full data migration • Incremental data migration
	PolarDB-X (formerly known as DRDS) All versions	• Full data migration • Incremental data migration
	User-created Oracle database (RAC or non-RAC architecture) Version 9i, 10g, 11g, 12c, 18c, or 19c	• Full data migration • Incremental data migration

Source database	Destination database	Migration type
	User-created Kafka database Versions 0.1 to 2.0	• Schema migration • Full data migration • Incremental data migration
User-created SQL Server database Version 2005, 2008, 2008 R2, 2012, 2014, 2016, or 2017 **Note** • DTS does not support SQL Server clusters or SQL Server Always On availability groups (AOAGs). • If the version of the source database is 2005, incremental data migration is not supported.	• User-created SQL Server database Version 2005, 2008, 2008 R2, 2012, 2014, 2016, or 2017 **Note** DTS does not support SQL Server clusters or SQL Server Always On availability groups (AOAGs). • ApsaraDB RDS for SQL Server Version 2008, 2008 R2, 2012, 2014, 2016, or 2017	• Schema migration • Full data migration • Incremental data migration
	User-created Oracle database (RAC or non-RAC architecture) Version 9i, 10g, 11g, 12c, 18c, or 19c	• Schema migration • Full data migration • Incremental data migration
	PolarDB Version 9.3, 9.6, 10, or 11	• Schema migration • Full data migration • Incremental data migration

Source database	Destination database	Migration type
User-created Oracle database (RAC or non-RAC architecture) Version 9i, 10g, 11g, 12c, 18c, or 19c	User-created MySQL database Version 5.1, 5.5, 5.6, 5.7, or 8.0	• Schema migration • Full data migration • Incremental data migration
	ApsaraDB RDS for MySQL All versions	• Schema migration • Full data migration • Incremental data migration
	PolarDB-X All versions	• Full data migration • Incremental data migration
	AnalyticDB for MySQL Version 2.0 or 3.0	• Schema migration • Full data migration • Incremental data migration
• User-created PostgreSQL database Version 9.4, 9.5, 9.6, or 10.x • ApsaraDB RDS for PostgreSQL Version 9.4 or 10	• User-created PostgreSQL database Version 9.4, 9.5, 9.6, or 10.x • ApsaraDB RDS for PostgreSQL Version 9.4 or 10	• Schema migration • Full data migration • Incremental data migration
	User-created Kafka database Versions 0.1 to 2.0	Incremental data migration

Source database	Destination database	Migration type
PolarDB Version 9.3, 9.6, 10, or 11	User-created Oracle database (RAC or non-RAC architecture) Version 9i, 10g, 11g, 12c, 18c, or 19c	• Full data migration • Incremental data migration
	PolarDB Version 9.3, 9.6, 10, or 11	• Schema migration • Full data migration • Incremental data migration
User-created Redis database Version 2.8, 3.0, 3.2, 4.0, or 5.0	User-created Redis database Version 2.8, 3.0, 3.2, 4.0, or 5.0	• Full data migration • Incremental data migration
User-created MongoDB database Version 3.0, 3.2, 3.4, 3.6, 4.0 or 4.2	User-created MongoDB database Version 3.0, 3.2, 3.4, 3.6, 4.0 or 4.2	• Full data migration • Incremental data migration

3.2. Create a data migration instance

Before you configure a task to migrate data, you must create a data migration instance. This topic describes how to create a data migration instance in the Data Transmission Service (DTS) console.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. In the upper-right corner, click **Create Migration Task**.
4. In the Create DTS Instances dialog box, select a region, and enter the number of data migration instances that you want to create.

   > **Note** In the Create DTS Instances dialog box, you can view the total number of instances, the number of existing instances, and the number of instances that can be created.

5. Click **Create**.

3.3. Configure data migration tasks

3.3.1. Migrate data from a user-created MySQL database to an ApsaraDB RDS for MySQL instance

This topic describes how to migrate data from a user-created MySQL database to an ApsaraDB RDS for MySQL instance by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. You can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the user-created MySQL database is 5.1, 5.5, 5.6, 5.7, or 8.0.
- If you need to perform incremental data migration, binary logging must be enabled for the user-created MySQL database and the following requirements must be met:
  - The value of the binlog_format parameter is set to row.
  - The value of the binlog_row_image parameter is set to full.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- DTS uses the `ROUND(COLUMN,PRECISION)` function to retrieve values from columns of the FLOAT or DOUBLE data type. If you do not specify a precision, DTS sets the precision for the FLOAT data type to 38 digits and the precision for the DOUBLE data type to 308 digits. You must check whether the precision settings meet your business requirements.
- DTS automatically creates a destination database in the ApsaraDB RDS for MySQL instance. However, if the name of the source database is invalid, you must manually create a database in the ApsaraDB RDS for MySQL instance before you configure the data migration task.
- If you perform a primary/secondary switchover on the source database when the data migration task is running, the task fails.
- Migration latency is the difference between the timestamp of the latest migrated data in the destination database and the current timestamp in the source database. If no data manipulation language (DML) operations are performed on the source database for a long time, the migration latency displayed in the DTS console may be inaccurate. If the latency of the migration task is too high, you can perform a DML operation on the source database to update the latency.

  > **Note** If you select an entire database as the object to be migrated, you can create a heartbeat table. The heartbeat table is updated or receives data every second.

Migration types

- Schema migration

  DTS migrates the schemas of the required objects to the destination instance. DTS supports schema migration for the following types of objects: table, view, trigger, stored procedure, and function.

> **Note**
> - During schema migration, DTS changes the value of the SECURITY attribute in views, stored procedures, and functions from DEFINER to INVOKER.
> - DTS does not migrate user information. Before a user can call views, stored procedures, and functions of the destination database, you must grant the read/write permissions to the user.

- Full data migration

  DTS migrates historical data of the required objects from the user-created MySQL database to the destination database in the ApsaraDB RDS for MySQL instance.

  > **Note** During full data migration, concurrent INSERT operations cause fragmentation in the tables of the destination instance. After full data migration is complete, the tablespace of the destination instance is larger than that of the source database.

- Incremental data migration

  After full data migration is complete, DTS retrieves binary log files from the user-created MySQL database. Then, DTS synchronizes incremental data from the user-created MySQL database to the destination ApsaraDB RDS for MySQL instance. Incremental data migration allows you to ensure service continuity when you migrate data from a user-created MySQL database to Alibaba Cloud.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **MySQL**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created MySQL database.
	Port Number	Enter the service port number of the user-created MySQL database. The default port number is **3306**.

Section	Parameter	Description
	Database Account	Enter the account of the user-created MySQL database. If you need to migrate incremental data, the account must have the SELECT permission on the objects to be migrated, the REPLICATION SLAVE permission, the REPLICATION CLIENT permission, and the SHOW VIEW permission. If you do not need to migrate incremental data, the account must have the SELECT permission on the objects to be migrated.
	Database Password	Enter the password of the source database account.
Destination Database	Instance Type	Select **RDS Instance**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	RDS Instance ID	Select the ID of the destination RDS instance.
	Database Account	Enter the database account of the destination RDS instance. The account must have the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If a whitelist is configured for the user-created MySQL database, you must manually add the CIDR blocks of DTS servers to the whitelist of the database. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the database, click **Next**.

   Note If you do not need to configure a whitelist for the user-created MySQL database, ignore the preceding settings and click **Next**.

6. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.

Setting	Description
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** • You can select columns, tables, or databases as the objects to be migrated. • By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. • If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.2. Migrate data from a user-created MySQL database to a PolarDB-X instance

This topic describes how to migrate data from a user-created MySQL database to a PolarDB-X instance by using Data Transmission Service (DTS). PolarDB-X is formerly known as Distributed Relational Database Service (DRDS).

Precautions

- DTS cannot migrate schemas from a user-created MySQL database to a PolarDB-X instance.

  > **Note** During schema migration, DTS migrates the schemas of the required objects, such as tables, from the source database to the destination database.

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- DTS uses the `ROUND(COLUMN,PRECISION)` function to retrieve values from columns of the FLOAT or DOUBLE data type. If you do not specify a precision, DTS sets the precision for the FLOAT data type to 38 digits and the precision for the DOUBLE data type to 308 digits. You must check whether the precision settings meet your business requirements.
- If you perform a primary/secondary switchover on the source database when the data migration task is running, the task fails.
- Migration latency is the difference between the timestamp of the latest migrated data in the destination database and the current timestamp in the source database. If no data manipulation language (DML) operations are performed on the source database for a long time, the migration latency displayed in the DTS console may be inaccurate. If the latency of the migration task is too high, you can perform a DML operation on the source database to update the latency.

  > **Note** If you select an entire database as the object to be migrated, you can create a heartbeat table. The heartbeat table is updated or receives data every second.

SQL operations that can be synchronized during incremental data migration

INSERT, UPDATE, DELETE, and REPLACE

Permissions required for database accounts

Database	Full data migration	Incremental data migration
User-created MySQL database	The SELECT permission	The REPLICATION SLAVE, REPLICATION CLIENT, SHOW VIEW, and SELECT permissions
PolarDB-X	The read and write permissions	The read and write permissions

Procedure

1. Create databases and tables in the destination PolarDB-X instance based on the schemas of the source tables.
2. Create a data migration instance.
3. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
4. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **MySQL**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created MySQL database.
	Port Number	Enter the service port number of the user-created MySQL database. The default port number is **3306**.
	Database Account	Enter the account of the user-created MySQL database. For more information about the permissions that are required for the account, see Permissions required for database accounts.
	Database Password	Enter the password of the source database account.
Destination Database	Instance Type	Select **DRDS Instance**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the PolarDB-X instance.
	Database Name	Enter the name of the destination database.
	Database Account	Enter the database account of the PolarDB-X instance. For more information about the permissions that are required for the account, see Permissions required for database accounts.
	Database Password	Enter the password of the destination database account.

5. In the lower-right corner of the page, click **Set Whitelist and Next**.
6. If a whitelist is configured for the user-created MySQL database, you must manually add the CIDR

blocks of DTS servers to the whitelist of the database. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the database, click **Next**.

> **Note** If you do not need to configure a whitelist for the user-created MySQL database, ignore the preceding settings and click **Next**.

7. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select only **Full Data Migration**. ◦ To ensure service continuity during data migration, select both **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select one or more objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

8. In the lower-right corner of the page, click **Precheck**.

> **Note** You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

9. After the task passes the precheck, click **Next**.

> **Note** If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.3. Migrate data from an ApsaraDB RDS for MySQL instance to a user-created Oracle database

This topic describes how to migrate data from an ApsaraDB RDS for MySQL instance to a user-created Oracle database by using Data Transmission Service (DTS). DTS supports full data migration and incremental data migration. You can select these migration types to ensure service continuity of the source ApsaraDB RDS for MySQL instance during data migration.

Prerequisites

The destination Oracle database is created. The schema of the Oracle database is the same as the schema of the source database in the ApsaraDB RDS for MySQL instance. This is because DTS does not support schema migration from an ApsaraDB RDS for MySQL instance to a user-created Oracle database.

Permissions required for database accounts

Database\Migration type	Full data migration	Incremental data migration
Source ApsaraDB RDS for MySQL instance	The read and write permissions	The read and write permissions
Destination Oracle database	The read and write permissions	The read and write permissions

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Optional. Enter a name for the task.

   DTS automatically generates a name for each task. Duplicate task names are allowed. You can edit the task name based on your needs. We recommend that you specify an informative name for easy identification.

4. Configure the source and destination databases. The following table describes the parameters.

Section	Parameter	Description
Source Database	Instance Type	Select **RDS Instance** as the type of the source instance.
	Instance Region	The region where the source instance resides.
	RDS Instance ID	Select the ID of the source database.
	Database Account	Enter an account that has the read and write permissions on the source database.
	Database Password	Enter the password of the source database account.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address** as the type of the destination database.
	Instance Region	The region where the destination instance resides.
	Database Type	Select **Oracle**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the Oracle database.
	Port Number	The default port number is 1521.
	Instance Type	○ **Non-RAC Instance**: If you select this option, you must specify the **SID**. ○ **RAC or PDB Instance**: If you select this option, you must specify the **Service Name**.
	Database Account	Enter an account that has the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account.

5. Click **Test Connectivity** and confirm that the test results for both the source and destination databases are **Passed**.
6. In the lower-right corner of the page, click **Set Whitelist and Next**.
7. Select the migration types based on your needs. Select one or more objects from the Available section and click the › icon to move the objects to the **Selected** section.
   - To ensure service continuity during data migration, select **Full Data Migration** and **Incremental Data Migration**.
   - To perform only full data migration, select **Full Data Migration**.
8. Click **Precheck** and wait until the precheck is complete.

> **Note** You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

9. Click **Next** to start the migration task.

> **Note** If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.4. Migrate data from an ApsaraDB RDS for MySQL instance to a user-created Kafka cluster

Kafka is a distributed message queue service that features high throughput and high scalability. Kafka is widely used for big data analytics such as log collection, data aggregation, streaming processing, and online and offline analysis. It is important for the big data ecosystem. You can use Data Transmission Service (DTS) to migrate data from an ApsaraDB RDS for MySQL instance or a user-created MySQL database to a user-created Kafka cluster. The data migration feature allows you to extend message processing capabilities. This topic uses an ApsaraDB RDS for MySQL instance as an example.

Prerequisites

- If the source database is a user-created MySQL database, binary logging must be enabled for the database and the following requirements must be met:
  - The value of the `binlog_format` parameter is set to `row`.
  - The value of the `binlog_row_image` parameter is set to `full`.
- A Kafka cluster is created and the Kafka version is 0.10 to 2.0.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- You can select only tables as the objects to be migrated.

Data format

The data that is migrated to the Kafka cluster is stored in the Avro format. You must parse the migrated data based on the Avro schema. For more information, see DTS Avro schema.

Procedure

1. Create a data migration instance.

2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure a task name.

   DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
4. Configure the source and destination databases.

Section	Parameter	Description
Source Database	Instance Type	Select **RDS Instance**.
	Instance Region	Select the region where the source database resides.
	RDS Instance ID	Select the ID of the source RDS instance.
	Database Account	Enter the database account of the source RDS instance. The account must have the SELECT permission on the objects to be migrated, the REPLICATION CLIENT permission, the REPLICATION SLAVE permission, and the SHOW VIEW permission.
	Database Password	Enter the password of the source database account.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **Kafka**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the Kafka cluster. In this example, enter the public IP address.
	Port Number	Enter the service port number of the Kafka cluster. The default port number is 9092.
	Database Account	Enter the username that is used to log on to the Kafka cluster. If no authentication is enabled for the Kafka cluster, you do not need to enter the username.
	Database Password	Enter the password of the username. If no authentication is enabled for the Kafka cluster, you do not need to enter the password.
	Topic	Click **Get Topic List**, and select a topic name from the drop-down list.

Section	Parameter	Description
	Kafka Version	Select the version of the destination Kafka cluster. **Note** You cannot select Kafka 2.0 in the DTS console. If you are using Kafka 2.0, you must select Kafka 1.0.
	Encryption	Select **Non-encrypted** or **SCRAM-SHA-256** based on your business and security requirements.

5. In the lower-right corner of the page, click **Set Whitelist and Next**.
6. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	○ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ○ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the > icon to move the objects to the **Selected** section. **Note** ○ You can select columns, tables, or databases as the objects to be migrated. ○ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ○ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.

Setting	Description
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.5. Migrate data from a user-created SQL Server database to an ApsaraDB RDS for SQL Server instance

This topic describes how to migrate incremental data from a user-created SQL Server database to an ApsaraDB RDS for SQL Server instance by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. You can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the user-created SQL Server database is 2005, 2008, 2008 R2, 2012, 2014, 2016, or 2017.

> Note
>
> - DTS does not support SQL Server clusters or SQL Server Always On availability groups (AOAGs).
> - If the version of the source database is 2005, incremental data migration is not supported.
> - If you migrate data between different versions of databases, make sure that the database versions are compatible.

- The tables to be migrated from the user-created SQL Server database have primary keys or UNIQUE NOT NULL indexes.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- To ensure that the incremental data migration task runs as expected, do not frequently back up the source database. We recommend that you retain log files for more than three days. Otherwise, you cannot retrieve log files after they are truncated.
- To ensure that the delay time of incremental data migration is accurate, DTS adds a heartbeat table to the user-created SQL Server database. The name of the heartbeat table is `Source table name_dts_mysql_heartbeat` .
- DTS automatically creates a destination database in the ApsaraDB RDS for SQL Server instance. However, if the name of the source database is invalid, you must manually create a database in the ApsaraDB RDS for SQL Server instance before you configure the data migration task.
- If a data migration task fails, DTS automatically resumes the task. Before you switch your workloads to the destination instance, stop or release the data migration task. Otherwise, the data in the source database will overwrite the data in the destination instance after the task is resumed.
- If you perform a primary/secondary switchover on the source database when the data migration task is running, the task fails.
- Migration latency is the difference between the timestamp of the latest migrated data in the destination database and the current timestamp in the source database. If no data manipulation language (DML) operations are performed on the source database for a long time, the migration latency displayed in the DTS console may be inaccurate. If the latency of the migration task is too high, you can perform a DML operation on the source database to update the latency.

> Note If you select an entire database as the object to be migrated, you can create a heartbeat table. The heartbeat table is updated or receives data every second.

Limits

- A single data migration task can migrate incremental data from only one database. To migrate incremental data from multiple databases, you must create a data migration task for each database.
- DTS cannot migrate the schemas of assemblies, service brokers, full-text indexes, full-text catalogs, distributed schemas, distributed functions, CLR stored procedures, CLR scalar-valued functions, CLR table-valued functions, internal tables, systems, or aggregate functions.
- DTS cannot migrate data of the sql_variant type.

- DTS cannot migrate tables that contain computed columns.

Migration types

- Schema migration

  DTS migrates the schemas of the required objects to the destination instance. DTS supports schema migration for the following types of objects: table, view, trigger, synonym, SQL stored procedure, SQL function, plan guide, user-defined type, rule, and default.

- Full data migration

  DTS migrates historical data of the required objects from the user-created SQL Server database to the destination database.

- Incremental data migration

  After full data migration is complete, DTS migrates incremental data from the user-created SQL Server database to the destination database. Incremental data migration allows you to ensure service continuity when you migrate data from a user-created SQL Server database to Alibaba Cloud.

SQL operations that can be synchronized during incremental data migration

- INSERT, UPDATE, and DELETE

  > **Note** DTS does not synchronize the UPDATE operations that update only the large fields.

- CREATE TABLE

  > **Note** If a CREATE TABLE operation creates a partition table or a table that contains functions, DTS does not synchronize the operation.

- ALTER TABLE, including only ADD COLUMN, DROP COLUMN, and RENAME COLUMN
- DROP TABLE
- RENAME TABLE, TRUNCATE TABLE, and CREATE INDEX

Procedure

To prevent data migration failures caused by dependencies among objects, DTS migrates the schemas and data from the source SQL Server database in the following order:

1. Migrate the schemas of tables, views, synonyms, user-defined types, rules, defaults, and plan guides.
2. Perform full data migration.
3. Migrate the schemas of SQL stored procedures, SQL functions, triggers, and foreign keys.
4. Perform incremental data migration.

   > **Note** During schema migration and full data migration, do not perform data definition language (DDL) operations on the required objects. Otherwise, the objects may fail to be migrated.

Before you begin

Before you configure a data migration task, configure log settings on the user-created SQL Server database.

> **Note** Skip this step if you do not need to perform incremental data migration.

1. Run the following command in the user-created SQL Server database to change the recovery mode to full mode:

   ```
   use master;
   GO
   ALTER DATABASE <database_name> SET RECOVERY FULL WITH ROLLBACK IMMEDIATE;
   GO
   ```

   Parameters:

   <database_name>: the name of the source database.

   Example:

   ```
   use master;
   GO
   ALTER DATABASE mytestdata SET RECOVERY FULL WITH ROLLBACK IMMEDIATE;
   GO
   ```

2. Run the following command to create a logical backup for the source database. Skip this step if you have already created a logical backup.

   ```
   BACKUP DATABASE <database_name> TO DISK='<physical_backup_device_name>';
   GO
   ```

   Parameters:

   - <database_name>: the name of the source database.
   - <physical_backup_device_name>: the storage path and file name of the backup file.

   Example:

   ```
   BACKUP DATABASE mytestdata TO DISK='D:\backup\dbdata.bak';
   GO
   ```

3. Run the following command to back up the log entries of the source database:

   ```
   BACKUP LOG <database_name> to DISK='<physical_backup_device_name>' WITH init;
   GO
   ```

   Parameters:

   - <database_name>: the name of the source database.
   - <physical_backup_device_name>: the storage path and file name of the backup file.

   Example:

   ```
   BACKUP LOG mytestdata TO DISK='D:\backup\dblog.bak' WITH init;
   GO
   ```

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **SQL Server**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created SQL Server database. In this example, enter the public IP address.
	Port Number	Enter the service port number of the user-created SQL Server database. The default port number is **1433**.
	Database Account	Enter the account that is used to log on to the user-created SQL Server database. **Note** To perform incremental data migration, the account must have the sysadmin permission. To perform schema migration or full data migration, the account must have the SELECT permission on the objects that you want to migrate.
	Database Password	Enter the password of the source database account.
Destination Database	Instance Type	Select **RDS Instance**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	RDS Instance ID	Select the ID of the destination ApsaraDB RDS for SQL Server instance. **Note** In this scenario, the destination database can be only a SQL Server database.
	Database Account	Enter the database account of the destination ApsaraDB RDS for SQL Server instance. The account must have the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If a whitelist is configured for the user-created SQL Server database, you must manually add the CIDR blocks of DTS servers to the whitelist of the database. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the database, click **Next**.

   > **Note** If you do not need to configure a whitelist for the user-created SQL Server database, ignore the preceding settings and click **Next**.

6. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	○ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ○ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ○ You can select columns, tables, or databases as the objects to be migrated. ○ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ○ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.

Setting	Description
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.6. Migrate data between user-created Oracle databases

This topic describes how to migrate data between user-created Oracle databases by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. You can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the source Oracle database is 9i, 10g, 11g, 12c, 18c, or 19c.

  Note To ensure compatibility, make sure the versions of the source and destination databases are the same.

- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the source Oracle database. For more information, see Supplemental Logging.
- The source Oracle database is running in ARCHIVELOG mode. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

- The available storage space of the destination Oracle database is larger than the total size of the data in the source Oracle database.
- If you perform a primary/secondary switchover on the source database when the data migration task is running, the task fails.

Migration types

Migration type	Description
Schema migration	DTS migrates the schemas of the required objects to the destination Oracle database. DTS supports schema migration for the following types of objects: table, view, synonym, trigger, stored procedure, function, package, and user-defined type. **Note** If an object contains triggers, the data between the source and destination databases will become inconsistent.
Full data migration	DTS migrates historical data of the required objects from the source Oracle database to the destination Oracle database. **Note** During schema migration and full data migration, do not perform DDL operations on the objects to be migrated. Otherwise, the objects may fail to be migrated.
Incremental data migration	After full data migration, DTS retrieves redo log files from the source Oracle database. Then, DTS synchronizes incremental data from the source Oracle database to the destination Oracle database. Incremental data migration allows you to ensure service continuity when you migrate data between Oracle databases.

Permissions required for database accounts

Database	Schema migration	Full data migration	Incremental data migration
Source Oracle database	The owner permission on schemas	The owner permission on schemas	SYSDBA
Destination Oracle database	The owner permission on schemas	The owner permission on schemas	The owner permission on schemas

Note For more information about how to create and authorize an Oracle database account, see CREATE USER and GRANT.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **Oracle**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the source Oracle database.
	Port Number	Enter the port number of the source Oracle database. The default port number is 1521.
	Instance Type	Select **Non-RAC Instance** or **RAC or PDB Instance** based on the architecture of the source Oracle database.
	SID	Enter the system ID (SID) of the source Oracle database. **Note** This parameter is required if you select **Non-RAC Instance** as the instance type.
	Service Name	Enter the server name of the instance. **Note** This parameter is required if you select **RAC or PDB Instance** as the instance type.
	Database Account	Enter the account that is used to connect to the source Oracle database.
	Database Password	Enter the password of the source database account.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **Oracle**.

Section	Parameter	Description
Destination Database	Hostname or IP Address	Enter the endpoint that is used to connect to the destination Oracle database.
	Port Number	Enter the port number of the destination Oracle database. The default port number is 1521.
	Instance Type	Select **Non-RAC Instance** or **RAC or PDB Instance** based on the architecture of the destination Oracle database.
	SID	Enter the SID of the destination Oracle database. **Note** This parameter is required if you select **Non-RAC Instance** as the instance type.
	Service Name	Enter the server name of the instance. **Note** This parameter is required if you select **RAC or PDB Instance** as the instance type.
	Database Account	Enter the account that is used to connect to the destination Oracle database.
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If your Oracle databases have security settings, you must manually add the CIDR blocks of DTS servers to the whitelists of the databases. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the databases, click **Next**.

   Note If you do not need to configure whitelists for your Oracle databases, ignore the preceding settings and click **Next**.

6. Select the migration types and the objects to be migrated.

Setting	Description

Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

> **Note** If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.7. Migrate data from a user-created Oracle database to an ApsaraDB RDS for MySQL instance

This topic describes how to migrate data from a user-created Oracle database to an ApsaraDB RDS for MySQL instance by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. When you migrate data from a user-created Oracle database, you can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the user-created Oracle database is 9i, 10g, 11g, 12c, 18c, or 19c.
- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the user-created Oracle database. For more information, see Supplemental Logging.
- The user-created Oracle database is running in ARCHIVELOG mode. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- Table names in the ApsaraDB RDS for MySQL instance are case-insensitive. If a table name in the user-created Oracle database contains uppercase letters, ApsaraDB RDS for MySQL converts all uppercase letters to lowercase letters before creating the table.

  If the source Oracle database contains identical table names that differ only in capitalization, these table names are identified as duplicate. During schema migration, the following message is returned: "The object already exists". To prevent name conflicts in the destination database, you can change the names of the migrated objects by using the object name mapping feature. For more information, see Object name mapping.
- If you perform a primary/secondary switchover on the source database when the data migration task is running, the task fails.

Migration types

- Schema migration

  DTS supports schema migration for tables and indexes. DTS does not support schema migration for the following types of objects: view, synonym, trigger, stored procedure, function, package, and user-defined type. DTS has the following limits on schema migration for tables and indexes:

  - Schema migration of nested tables is not supported. Clustered tables and index-organized tables (IOTs) are converted into common tables in the destination database.
  - Schema migration of function-based indexes, domain indexes, bitmap indexes, and reverse indexes is not supported.

- Full data migration

  DTS migrates historical data of the required objects from the user-created Oracle database to the destination ApsaraDB RDS for MySQL instance.

- Incremental data migration

  DTS retrieves redo log files from the user-created Oracle database. Then, DTS synchronizes incremental data from the user-created Oracle database to the destination database in the ApsaraDB RDS for MySQL instance. Incremental data migration allows you to ensure service continuity when you migrate data from the user-created Oracle database to the destination database.

SQL operations that can be synchronized during incremental data migration

- INSERT, DELETE, and UPDATE
- CREATE TABLE

  > **Note** If a CREATE TABLE operation creates a table that contains functions, DTS does not synchronize the operation.

- ALTER TABLE, ADD COLUMN, DROP COLUMN, RENAME COLUMN, and ADD INDEX
- DROP TABLE
- RENAME TABLE, TRUNCATE TABLE, and CREATE INDEX

Data type mappings

For more information, see Data type mappings between heterogeneous databases.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
	Instance Type	Select an instance type based on the deployment of the source database. In this example, select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.

Section	Parameter	Description
Source Database	Database Type	Select **Oracle**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created Oracle database.
	Port Number	Enter the service port number of the user-created Oracle database. The default port number is **1521**.
	Instance Type	○ **Non-RAC Instance**: If you select this option, you must specify the **SID**. ○ **RAC or PDB Instance**: If you select this option, you must specify the **Service Name**.
	Database Account	Enter the account of the user-created Oracle database. To perform incremental data migration, the account must have the database administrator (DBA) permission. To perform schema migration or full data migration, the account must have the owner permission on schemas.
	Database Password	Enter the password of the source database account.
Destination Database	Instance Type	Select **RDS Instance**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	RDS Instance ID	Select the ID of the destination ApsaraDB RDS for MySQL instance.
	Database Account	Enter the database account of the destination ApsaraDB RDS for MySQL instance. The account must have the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If your Oracle databases have security settings, you must manually add the CIDR blocks of DTS servers to the whitelists of the databases. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the databases, click **Next**.

   > **Note** If you do not need to configure whitelists for your Oracle databases, ignore the preceding settings and click **Next**.

6. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

> **Note** If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.8. Migrate data from a user-created Oracle database to a PolarDB cluster

This topic describes how to migrate data from a user-created Oracle database to a PolarDB cluster by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. You can select all of the supported migration types to ensure service continuity.

Prerequisites

- The version of the user-created Oracle database is 9i, 10g, 11g, 12c, 18c, or 19c.
- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the user-created Oracle database. For more information, see Supplemental Logging.
- The user-created Oracle database is running in ARCHIVELOG mode. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- DTS supports schema migration for the following types of objects: table, view, synonym, trigger, stored procedure, function, package, and user-defined type.
- If you perform a primary/secondary switchover on the source database when the data migration task is running, the task fails.

Limits

- During schema migration, the reverse indexes and bitmap indexes of the source database are stored as common indexes in the PolarDB cluster.
- During schema migration, partitioned indexes are converted into independent indexes on each partition in the PolarDB cluster.
- Incremental data migration supports only tables that have primary keys or UNIQUE NOT NULL indexes.
- Incremental data migration does not support the LONG data type.
- Data definition language (DDL) operations that are performed during incremental data migration cannot be synchronized to the destination database.
- Materialized views cannot be migrated.

Data type mappings

For more information, see Data type mappings between heterogeneous databases.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select an instance type based on the deployment of the source database. In this example, select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **Oracle**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created Oracle database.
	Port Number	Enter the service port number of the user-created Oracle database. The default port number is **1521**.
	Instance Type	◦ **Non-RAC Instance**: If you select this option, you must specify the **SID**. ◦ **RAC or PDB Instance**: If you select this option, you must specify the **Service Name**.
	Database Account	Enter the account of the user-created Oracle database. To perform incremental data migration, the account must have the database administrator (DBA) permission. To perform schema migration or full data migration, the account must have the owner permission on schemas.
	Database Password	Enter the password of the source database account.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **PolarDB-O**.

Section	Parameter	Description
Destination Database	DNS or IP Address	The endpoint of the destination PolarDB cluster.
	Port Number	Enter the service port number of the destination PolarDB cluster. The default port number is **1433**.
	Database Name	Enter the name of the destination database in the PolarDB cluster.
	Database Account	Enter the database account of the destination PolarDB cluster. The account must have the read and write permissions.
	Database Password	Enter the password of the destination database account. **Note** After you specify the information about the RDS instance, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If your Oracle databases have security settings, you must manually add the CIDR blocks of DTS servers to the whitelists of the databases. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the databases, click **Next**.

   Note If you do not need to configure whitelists for your Oracle databases, ignore the preceding settings and click **Next**.

6. Select the migration types and the objects to be migrated.

Setting	Description

Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

> **Note** If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.9. Migrate data from a user-created Oracle database to a PolarDB-X instance

This topic describes how to migrate data from a user-created Oracle database to a PolarDB-X instance by using Data Transmission Service (DTS). PolarDB-X is formerly known as Distributed Relational Database Service (DRDS). DTS allows you to ensure service continuity when you migrate both historical data and incremental data.

Prerequisites

- The version of the user-created Oracle database is 9i, 10g, 11g, 12c, 18c, or 19c.
- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the user-created Oracle database. For more information, see Supplemental Logging.
- The user-created Oracle database is running in ARCHIVELOG mode. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

Precautions

- DTS cannot migrate schemas from a user-created Oracle database to a PolarDB-X instance.

  > **Note** During schema migration, DTS migrates the schemas of the required objects, such as tables, from the source database to the destination database.

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- If you perform a primary/secondary switchover on the source database when the data migration task is running, the task fails.

Migration types

- Full data migration

  DTS migrates historical data of the required objects from the source Oracle database to the destination database.

- Incremental data migration

  After full data migration is complete, DTS retrieves redo log files from the source Oracle database. Then, DTS synchronizes incremental data from the source Oracle database to the destination database.

> **Note** The following SQL operations can be synchronized during incremental data migration: INSERT, DELETE, and UPDATE. Data definition language (DDL) operations cannot be synchronized during incremental data migration.

Procedure

1. Create databases and tables in the destination PolarDB-X instance based on the schemas of the source tables.

   > **Note** The data types of Oracle and PolarDB-X do not have one-to-one correspondence. You must create the corresponding data types in the destination PolarDB-X instance. For more information, see Data type mappings between heterogeneous databases.

2. Create a data migration instance.
3. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
4. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **Oracle**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created Oracle database. In this example, enter the public IP address.
	Port Number	Enter the service port number of the user-created Oracle database. The default port number is **1521**.
	Instance Type	◦ **Non-RAC Instance**: If you select this option, you must specify the **SID**. ◦ **RAC or PDB Instance**: If you select this option, you must specify the **Service Name**.
	Database Account	Enter the account of the user-created Oracle database. To perform incremental data migration, the account must have the database administrator (DBA) permission. To perform schema migration or full data migration, the account must have the owner permission on schemas.
	Database Password	Enter the password of the source database account.

Section	Parameter	Description
Destination Database	Instance Type	Select **DRDS Instance**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the destination PolarDB-X instance.
	Database Account	Enter the database account of the destination PolarDB-X instance. The account must have the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account.

5. In the lower-right corner of the page, click **Set Whitelist and Next**.
6. If your Oracle databases have security settings, you must manually add the CIDR blocks of DTS servers to the whitelists of the databases. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the databases, click **Next**.

   Note If you do not need to configure whitelists for your Oracle databases, ignore the preceding settings and click **Next**.

7. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select only **Full Data Migration**. ◦ To ensure service continuity during data migration, select both **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.

Setting	Description
Select the objects to be migrated	Select one or more objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

8. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

9. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.10. Migrate data from a self-managed Oracle database to an AnalyticDB for PostgreSQL instance

This topic describes how to migrate data from a self-managed Oracle database to an AnalyticDB for PostgreSQL instance by using Data Transmission Service (DTS). The data migration feature allows you to transfer and analyze data with ease.

Prerequisites

Limits

Category	Description
Limits on the source database	
Other limits	• • • •

Migration types

Migration type	Description
Schema migration	DTS migrates the schemas of required objects from the source database to the destination database. DTS supports schema migration for the following types of objects: table, index, constraint, function, sequence, and view. **Warning** • In this topic, the source and the destination databases are heterogeneous databases. DTS does not ensure that the schemas of the source and destination databases are consistent after schema migration. We recommend that you evaluate the impact of data type conversion on your business. For more information, see Data type mappings between heterogeneous databases. • For partitioned tables, DTS discards the partition definitions. You must define partitions in the destination database.

Migration type	Description
Full data migration	DTS migrates full data of required objects from the source database to the destination database. **Note** During schema migration and full data migration, do not perform DDL operations on the objects to be migrated. Otherwise, the objects may fail to be migrated.
Incremental data migration	After full data migration is complete, DTS retrieves redo log files from the source Oracle database. Then, DTS migrates incremental data from the source Oracle database to the destination database in real time. DTS can synchronize the following SQL operations during incremental data migration: • DML operations: INSERT, UPDATE, and DELETE • DDL operation: ADD COLUMN Incremental data migration ensures service continuity of self-managed applications during data migration.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	The task name that DTS automatically generates. We recommend that you specify a descriptive name that makes it easy to identify the task. You do not need to specify a unique task name.
Source Database	Instance Type	The instance type of the source database. In this example, **User-Created Database with Public IP Address** is selected.
	Instance Region	If you select **User-Created Database with Public IP Address** as the instance type, you do not need to specify the **Instance Region** parameter.
	Database Type	Select **Oracle**.
	Hostname or IP Address	The endpoint that is used to connect to the self-managed Oracle database.
	Port Number	The service port number of the self-managed Oracle database. Default value: **1521**.

Section	Parameter	Description
	Instance Type	◦ If you select **Non-RAC Instance**, you must set the **SID** parameter. ◦ If you select **RAC or PDB Instance**, you must set the **Service Name** parameter.
	Database Account	The account of the self-managed Oracle database. To perform incremental data migration, the account must have the database administrator (DBA) permissions. To perform schema migration or full data migration, the account must have the owner permissions on schemas.
	Database Password	The password of the database account.
Destination Database	Instance Type	Select **AnalyticDB for PostgreSQL**.
	Instance Region	The destination region that you selected on the buy page. You cannot change the value of this parameter.
	The ID of the destination instance.	The ID of the destination AnalyticDB for PostgreSQL instance.
	Database Name	The name of the destination database.
	Database Account	The database account of the destination AnalyticDB for PostgreSQL instance. The account must have the read and write permissions on the instance.
	Database Password	The password of the database account. **Note** After you specify the information about the RDS instance, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If your Oracle databases have security settings, you must manually add the CIDR blocks of DTS servers to the whitelists of the databases. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the databases, click **Next**.

   Note If you do not need to configure whitelists for your Oracle databases, ignore the preceding settings and click **Next**.

6. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7.

8. After the task passes the precheck, click **Next**.

Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.11. Migrate data from a self-managed Oracle database to a self-managed PostgreSQL database

This topic describes how to configure data migration from a self-managed Oracle database to a self-managed PostgreSQL database in the Data Transmission Service (DTS) console. DTS ensures service continuity when you migrate schema data, full data, and incremental data.

Prerequisites

Limits

Category	Description
Limits on the source database	
Other limits	• If you perform a primary/secondary switchover on the source and destination databases when the data migration task is running, the task fails. • • •

Migration types

Migration type	Description
Schema migration	DTS migrates the schemas of required objects from the source database to the destination database. DTS allows you to specify tables, views, and indexes as objects.
Full data migration	DTS migrates full data of required objects from the source database to the destination database. **Note** During schema migration and full data migration, do not perform DDL operations on the objects to be migrated. Otherwise, the objects may fail to be migrated.

Migration type	Description
Incremental data migration	After full data migration is complete, DTS retrieves redo log files from the source Oracle database. Then, DTS migrates incremental data from the source Oracle database to the destination database in real time. DTS can synchronize the following SQL operations during incremental data migration: DML operations: INSERT, UPDATE, and DELETE Incremental data migration ensures service continuity of self-managed applications during data migration.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	The task name that DTS automatically generates. We recommend that you specify a descriptive name that makes it easy to identify the task. You do not need to specify a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If you select **User-Created Database with Public IP Address** as the instance type, you do not need to set the **Instance Region** parameter.
	Database Type	Select **Oracle**.
	Hostname or IP Address	The endpoint that is used to connect to the self-managed Oracle database. In this example, the public IP address of the database is used.
	Port Number	The service port number of the self-managed Oracle database. Default value: **1521**.
	Instance Type	○ If you select **Non-RAC Instance**, you must set the **SID** parameter. ○ If you select **RAC or PDB Instance**, you must set the **Service Name** parameter.
	Database Account	The account of the self-managed Oracle database. To perform incremental data migration, the account must have the database administrator (DBA) permissions. To perform schema migration or full data migration, the account must have the owner permissions on schemas.

Section	Parameter	Description
	Database Password	The password of the database account.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected on the buy page. You cannot change the value of this parameter.
	Database Type	Select **PostgreSQL**.
	Hostname or IP Address	The endpoint that is used to connect to the self-managed PostgreSQL database. In this example, the public IP address is entered.
	Port Number	The service port number of the self-managed PostgreSQL database. Default value: **3433**.
	Database Account	The account of the self-managed PostgreSQL database. The account must have the read and write permissions on the destination database.
	Database Password	The password of the database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If your Oracle databases have security settings, you must manually add the CIDR blocks of DTS servers to the whitelists of the databases. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the databases, click **Next**.

   Note If you do not need to configure whitelists for your Oracle databases, ignore the preceding settings and click **Next**.

6. Select the objects to be migrated and the migration types.

Setting	Description
Select migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ If you want to ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.

Setting	Description
Select the objects to be migrated	Select one or more objects from the **Source Objects** section and click the icon to add the objects to the **Selected Objects** section. **Note** ◦ You can select columns, tables, or databases as objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature to rename an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time range for failed connections to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the following 12 hours. You can specify the retry time range based on your business requirements. If DTS is reconnected to the source and destination databases within the specified time range, DTS resumes the data migration task. Otherwise, the data migration task fails.
Specify the capitalization of object names in the destination instance	Specify the capitalization of database names, table names, and column names in the destination instance. By default, **Consistent with the default policy of the destination database (lowercase)** is selected. You can also select other capitalization policies. For more information, see Specify the capitalization of object names in the destination instance.

7.

8. After the task passes the precheck, click **Next**.

Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.12. Migrate data between ApsaraDB RDS for PostgreSQL instances

This topic describes how to migrate data between ApsaraDB RDS for PostgreSQL instances by using Data Transmission Service (DTS). DTS supports schema migration, full data migration, and incremental data migration. You can select all of the supported migration types to ensure service continuity.

Prerequisites

The network type of both the source and destination ApsaraDB RDS for PostgreSQL instances is VPC.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- To ensure that the delay time of incremental data migration is accurate, DTS adds a heartbeat table named *dts_postgres_heartbeat* to the source database.
- During incremental data migration, DTS creates a replication slot for the source database. The replication slot is prefixed with `dts_sync_`. DTS automatically clears historical replication slots every 90 minutes to reduce storage usage.

  > **Note** If the migration task is released or fails, DTS automatically clears the replication slot. If a primary/secondary switchover is performed on the source ApsaraDB RDS for PostgreSQL instance, you must log on to the secondary database to clear the replication slot.

- If a data migration task fails, DTS automatically resumes the task. Before you switch your workloads to the destination instance, stop or release the data migration task. Otherwise, the data in the source instance will overwrite the data in the destination instance after the task is resumed.

Limits

- A single data migration task can migrate data from only one database. To migrate data from multiple databases, you must create a data migration task for each database.
- During incremental data migration, DTS migrates only data manipulation language (DML) operations. DML operations include INSERT, DELETE, and UPDATE.

Data migration process

The following table describes how DTS migrates the schemas and data of the source PostgreSQL database. The process prevents data migration failures that are caused by dependencies between objects.

Data migration process	Description
1. Schema migration	DTS migrates the schemas of tables, views, sequences, functions, user-defined types, rules, domains, operations, and aggregates to the destination database. **Note** DTS does not migrate functions that are written in the C programming language.
2. Full data migration	DTS migrates historical data of the required objects to the destination database.
3. Schema migration	DTS migrates the schemas of triggers and foreign keys to the destination database.
4. Incremental data migration	DTS migrates incremental data of the required objects to the destination database. **Note** Incremental data migration does not support the BIT data type.

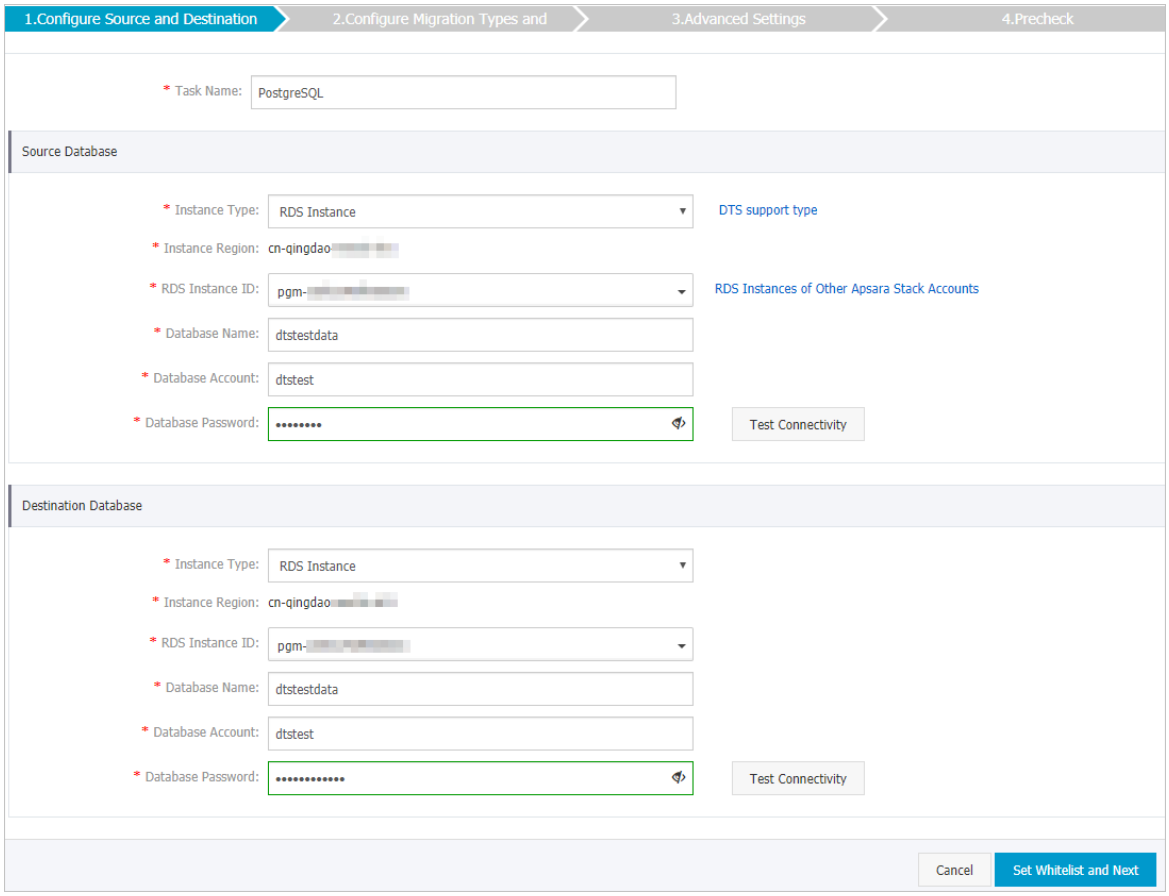
Before you begin

If you need to perform incremental data migration, you must perform the following steps: Log on to the ApsaraDB RDS console. Click the ID of the source ApsaraDB RDS for PostgreSQL instance. In the left-side navigation pane, click **Parameters**. Find the wal_level parameter and change the value to logical.

Warning After you change the value of the `wal_level` parameter, you must restart the instance to apply the change. We recommend that you evaluate the impact on your business and change the parameter setting during off-peak hours.

Procedure

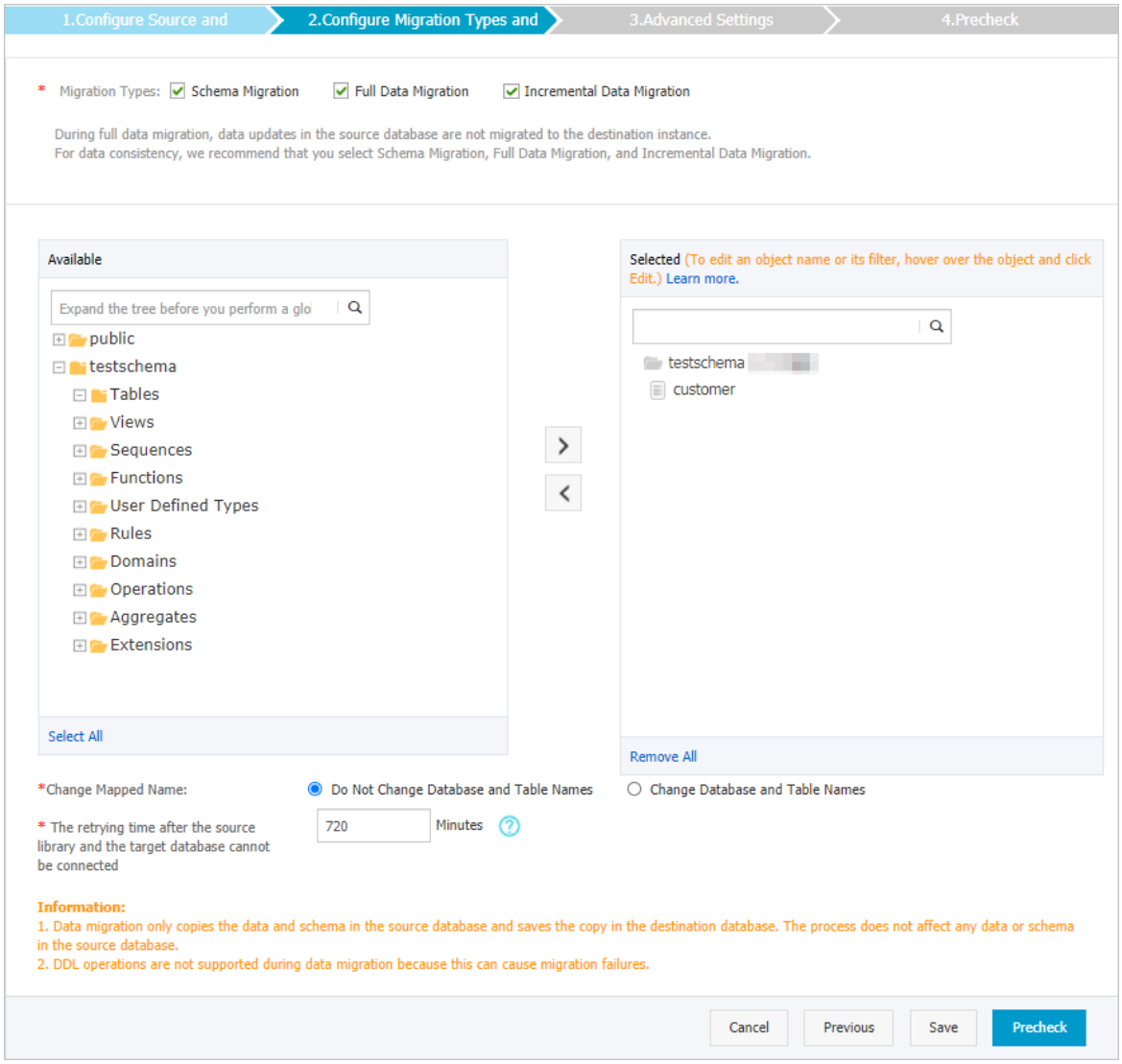
1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.



Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **RDS Instance**.
	Instance Region	Select the region where the source ApsaraDB RDS for PostgreSQL instance resides.
	RDS Instance ID	Select the ID of the source ApsaraDB RDS for PostgreSQL instance.
	Database Name	Enter the name of the source database.
	Database Account	Enter the database account of the source ApsaraDB RDS for PostgreSQL instance. The account must have the read and write permissions on the source database.
	Database Password	Enter the password of the source database account.
	Instance Type	Select **RDS Instance**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.

Section	Parameter	Description
Destination Database	RDS Instance ID	Select the ID of the destination ApsaraDB RDS for PostgreSQL instance.
	Database Name	Enter the name of the destination database. **Note** The name of the destination database can be different from the name of the source database.
	Database Account	Enter the database account of the destination ApsaraDB RDS for PostgreSQL instance. The account must have the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types and the objects to be migrated.



Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.

Setting	Description
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** • You can select columns, tables, or databases as the objects to be migrated. • By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. • If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.13. Migrate data between PolarDB clusters

This topic describes how to migrate data between PolarDB clusters by using Data Transmission Service (DTS).

Prerequisites

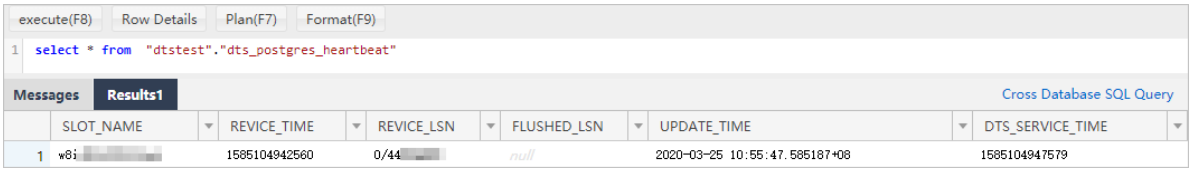
The tables to be migrated from the source PolarDB cluster contain primary keys or UNIQUE NOT NULL indexes.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- A single data migration task can migrate data from only one database. To migrate data from multiple databases, you must create a data migration task for each database.
- If you select a schema as the object to be migrated and create a table in the schema or run the RENAME command to rename the table, you must run the `ALTER TABLE schema.table REPLICA IDENTITY FULL;` command before you write data to the table.

  > **Note** Replace the `schema` and `table` in the preceding sample command with the actual schema name and table name.

- To ensure that the delay time of data migration is accurate, DTS adds a heartbeat table named `dts_postgres_heartbeat` to the source database. The following figure shows the schema of the heartbeat table.



- If a data migration task fails, DTS automatically resumes the task. Before you switch your workloads to the destination database, stop or release the data migration task. Otherwise, the data from the source database will overwrite the data in the destination database after the task is resumed.

Migration types

Migration type	Description
Schema migration	DTS migrates the schemas of the required objects from the source database to the destination PolarDB cluster. DTS supports schema migration for the following types of objects: table, view, synonym, trigger, stored procedure, function, package, and user-defined type. **Notice** However, if an object contains triggers, data will become inconsistent between the source and destination databases.

Migration type	Description
Full data migration	DTS migrates historical data of the required objects from the source database to the destination PolarDB cluster. **Notice** During schema migration and full data migration, do not perform data definition language (DDL) operations on the required objects. Otherwise, the objects may fail to be migrated.
Incremental data migration	DTS retrieves redo log files from the source database. Then, DTS synchronizes incremental data from the source database to the destination PolarDB cluster. DTS can synchronize data manipulation language (DML) operations, including INSERT, UPDATE, and DELETE operations. DTS cannot synchronize DDL operations. Incremental data migration allows you to ensure service continuity when you migrate data between PolarDB clusters.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**. You cannot select PolarDB cluster as the instance type.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **PolarDB-O**.
	DNS or IP Address	The endpoint of the source PolarDB cluster.
	Port Number	Enter the service port number of the PolarDB cluster. The default port number is **3433**.
	Database Name	Enter the name of the source database.
	Database Account	Enter the initial account of the source PolarDB cluster.

Section	Parameter	Description
	Database Password	Enter the password of the source database account. **Note** After you specify the information about the self-managed Oracle database, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **PolarDB-O**.
	DNS or IP Address	The endpoint of the destination PolarDB cluster.
	Port Number	Enter the service port number of the destination PolarDB cluster. The default port number is **1433**.
	Database Name	Enter the name of the destination database.
	Database Account	Enter the database account of the destination PolarDB cluster. The account must have the read and write permissions.
	Database Password	Enter the password of the destination database account. **Note** After you specify the information about the RDS instance, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types and the objects to be migrated.

Setting	Description

Setting	Description
Select the migration types	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

> **Note** If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.14. Migrate data from a PolarDB cluster to a user-created Oracle database

This topic describes how to migrate data from a PolarDB cluster to a user-created Oracle database by using Data Transmission Service (DTS).

Prerequisites

The destination Oracle database is created. The schema of the Oracle database is the same as the schema of the source database in the PolarDB cluster. This is because DTS does not support schema migration from a PolarDB cluster to a user-created Oracle database.

Permissions required for database accounts

Migration types	Full data migration	Incremental data migration
Source PolarDB cluster	The read and write permissions	The read and write permissions
Destination Oracle database	The read and write permissions	The read and write permissions

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination instances.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
	Instance Type	Select **User-Created Database with Public IP Address**. You cannot select PolarDB cluster as the instance type.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **PolarDB-O**.
	DNS or IP Address	Enter the endpoint of the source PolarDB cluster.

Section	Parameter	Description
Source Database	Port Number	Enter the service port number of the source PolarDB cluster. The default port number is **1521**.
	Database Name	Enter the name of the source database.
	Database Account	Enter an account that has the read and write permissions on the source database.
	Database Password	Enter the password of the source database account. **Note** After you specify the information about the self-managed Oracle database, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **Oracle**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created Oracle database.
	Port Number	Enter the service port number of the user-created Oracle database. The default port number is **1521**.
	Instance Type	◦ **Non-RAC Instance**: If you select this option, you must specify the **SID**. ◦ **RAC or PDB Instance**: If you select this option, you must specify the **Service Name**.
	Database Account	Enter an account that has the read and write permissions on the destination database.

Section	Parameter	Description
	Database Password	Enter the password of the destination database account. **Note** After you specify the information about the RDS instance, you can click **Test Connectivity** next to **Database Password** to check whether the information is valid. If the information is valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Then, modify the information based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select only **Full Data Migration**. ◦ To ensure service continuity during data migration, select both **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select one or more objects from the **Available** section and click the > icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.

Setting	Description
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.15. Migrate data from a PolarDB cluster to a user-created Kafka cluster

Kafka is a distributed message queue service that features high throughput and high scalability. Kafka is widely used for big data analytics such as log collection, data aggregation, streaming processing, and online and offline analysis. It is important for the big data ecosystem. This topic describes how to migrate data from a PolarDB cluster to a user-created Kafka cluster by using Data Transmission Service (DTS). The data migration feature allows you to extend message processing capabilities.

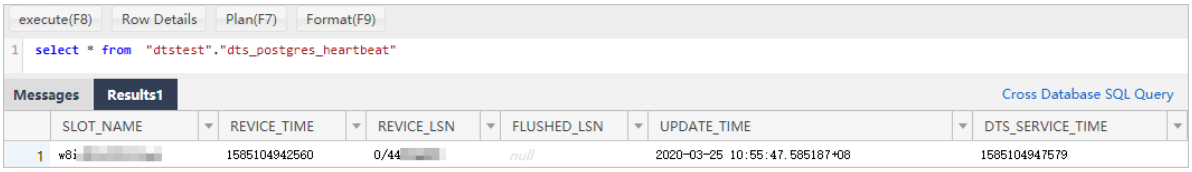
Prerequisites

The tables to be migrated from the source PolarDB cluster contain primary keys or UNIQUE NOT NULL indexes.

Precautions

- DTS can migrate only incremental data from a PolarDB cluster to a user-created Kafka cluster.
- A single data migration task can migrate data from only one database. To migrate data from multiple databases, you must create a data migration task for each database.

- To ensure that the delay time of data migration is accurate, DTS adds a heartbeat table named `dts_postgres_heartbeat` to the source database. The following figure shows the schema of the heartbeat table.



Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination instances.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**. You cannot select PolarDB cluster as the instance type.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **PolarDB-O**.
	DNS or IP Address	Enter the endpoint of the source PolarDB cluster.
	Port Number	Enter the service port number of the source PolarDB cluster. The default port number is **1521**.
	Database Name	Enter the name of the source database.
	Database Account	Enter the privileged account of the source PolarDB cluster.
	Database Password	Enter the password of the source database account.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.

Section	Parameter	Description
Destination Database	Database Type	Select **Kafka**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the Kafka cluster. In this example, enter the public IP address.
	Port Number	Enter the service port number of the Kafka cluster. The default port number is 9092.
	Database Account	Enter the username that is used to log on to the Kafka cluster. If no authentication is enabled for the Kafka cluster, you do not need to enter the username.
	Database Password	Enter the password of the database account. If no authentication is enabled for the Kafka cluster, you do not need to enter the password.
	Topic	Click **Get Topic List**, and select a topic name from the drop-down list.
	Kafka Version	Select the version of the user-created Kafka cluster. **Note** You cannot select Kafka 2.0 in the DTS console. If you are using Kafka 2.0, you must select Kafka 1.0.
	Encryption	Select **Non-encrypted** or **SCRAM-SHA-256** based on your business and security requirements.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the migration type and the objects to be migrated.

Setting	Description
Select the migration type	Select **Incremental Data Migration**.
Select the objects to be migrated	Select one or more tables from the **Available** section and click the icon to move the tables to the **Selected** section. You can select only tables as the objects to be migrated.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. After the task passes the precheck, click **Next**.

> **Note** If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.16. Migrate data between user-created Redis databases

This topic describes how to migrate data between user-created Redis databases by using Data Transmission Service (DTS).

Prerequisites

- The version of the source Redis database is 2.8, 3.0, 3.2, 4.0, or 5.0.
- The source Redis database uses the standalone architecture rather than the cluster architecture.
- The `PSYNC` or `SYNC` command can be executed on the source Redis database.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- If the data eviction policy (`maxmemory-policy`) of the destination database is not set to `noeviction` , data may become inconsistent between the source and destination databases. For more information, see Eviction policies.
- If you run the EVAL or EVALSHA command to call Lua scripts, DTS cannot identify whether these Lua scripts are executed on the destination database. During incremental data migration, the destination database does not explicitly return the execution results of Lua scripts.
- When you run the `PSYNC` or `SYNC` command to transmit data of the LIST type, DTS does not perform the `flush` operation on the existing data. Therefore, the destination database may contain duplicate data records.
- Migration latency is the difference between the timestamp of the latest migrated data in the destination database and the current timestamp in the source database. If no data manipulation language (DML) operations are performed on the source database for a long time, the migration latency displayed in the DTS console may be inaccurate. If the latency of the migration task is too high, you can perform a DML operation on the source database to update the latency.

  > **Note** If you select an entire database as the object to be migrated, you can create a heartbeat table. The heartbeat table is updated or receives data every second.

Migration types

- Full data migration: DTS migrates historical data of the required objects from the source database to the destination database.
- Incremental data migration: DTS migrates incremental data from the source database to the destination database in real time.

Operations that can be synchronized during incremental data migration

- APPEND
- BITOP, BLPOP, BRPOP, and BRPOPLPUSH
- DECR, DECRBY, and DEL
- EVAL, EVALSHA, EXEC, EXPIRE, and EXPIREAT
- FLUSHALL and FLUSHDB
- GEOADD and GETSET
- HDEL, HINCRBY, HINCRBYFLOAT, HMSET, HSET, and HSETNX
- INCR, INCRBY, and INCRBYFLOAT
- LINSERT, LPOP, LPUSH, LPUSHX, LREM, LSET, and LTRIM
- MOVE, MSET, MSETNX, and MULTI
- PERSIST, PEXPIRE, PEXPIREAT, PFADD, PFMERGE, PSETEX, and PUBLISH
- RENAME, RENAMENX, RESTORE, RPOP, RPOPLPUSH, RPUSH, and RPUSHX
- SADD, SDIFFSTORE, SELECT, SET, SETBIT, SETEX, SETNX, SETRANGE, SINTERSTORE, SMOVE, SPOP, SREM, and SUNIONSTORE
- ZADD, ZINCRBY, ZINTERSTORE, ZREM, ZREMRANGEBYLEX, ZUNIONSTORE, ZREMRANGEBYRANK, and ZREMRANGEBYSCORE

Before you begin

To ensure that incremental data migration tasks run as expected, we recommend that you remove the limit on the replication output buffer for the source database. This topic uses a Linux server as an example.

> **Note** If you perform only full data migration, skip the following steps.

1. Use the redis-cli program to connect to the source database.

   > **Note** You can use the redis-cli program after you install the Redis client. For more information, see Redis community official website.

   ```
   redis-cli -h <host> -p <port> -a <password>
   ```

   > **Note**
   > - <host>: the endpoint that is used to connect to the source database. You can use 127.0.0.1 in this example.
   > - <port>: the service port number of the source database. The default port number is 6379.
   > - <password>: the password of the source database.

   Example:

   ```
   redis-cli -h 127.0.0.1 -p 6379 -a Test123456
   ```

2. Run the following command to remove the limit on the replication output buffer:

```
config set client-output-buffer-limit 'slave 0 0 0'
```

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure a task name.

   DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
4. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **Redis**.
	Instance Mode	The value of this parameter is set to **Standalone** and cannot be changed to Cluster.
	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database. The default port number is **6379**.
	Database Password	Enter the password of the source database. If password verification is disabled for the source database, you do not need to enter the password. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.

Section	Parameter	Description
Destination Database	Database Type	Select **Redis**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the destination database.
	Port Number	Enter the service port number of the destination database. The default port number is **6379**.
	Database Password	Enter the password of the destination database. If password verification is disabled for the destination database, you do not need to enter the password. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

5. In the lower-right corner of the page, click **Set Whitelist and Next**.
6. If a whitelist is configured for the user-created Redis database, you must manually add the CIDR blocks of DTS servers to the whitelist of the database. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the database, click **Next**.

   Note If you do not need to configure a whitelist for the user-created Redis database, ignore the preceding settings and click **Next**.

7. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select only **Full Data Migration**. ◦ To ensure service continuity during data migration, select both **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.

Setting	Description
Select the objects to be migrated	Select one or more objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** • You can select columns, tables, or databases as the objects to be migrated. • By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. • If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

8. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

9. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.17. Migrate data between user-created MongoDB databases

This topic describes how to migrate data between user-created MongoDB databases by using Data Transmission Service (DTS).

Prerequisites

The version of the source MongoDB database is 3.0, 3.2, 3.4, 3.6, 4.0, or 4.2.

Precautions

- During full data migration, DTS uses read and write resources of the source and destination databases. This may increase the loads of the database servers. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- If you need to migrate incremental data from a standalone MongoDB database, you must enable oplog. A standalone MongoDB database contains only a primary node. For more information, see Preparation for a standalone MongoDB database.
- If the source database uses the sharded cluster architecture, you must configure a data migration task for each shard.

Migration types

Migration type	Description
Full data migration	DTS migrates historical data of the required objects from the source database to the destination database. **Note** The following types of objects are supported: database, collection, and index.
Incremental data migration	After full data migration is complete, DTS migrates incremental data from the source database to the destination database in real time. **Note** • The create and delete operations that are performed on databases, collections, and indexes can be migrated. • The create, delete, and update operations that are performed on documents can be migrated.

Preparation for a standalone MongoDB database

If you need to migrate incremental data from a standalone MongoDB database, you must enable oplog. A standalone MongoDB database contains only a primary node. If you do not need to perform incremental data migration, skip the following steps.

Note To enable oplog, you must restart the MongoDB service. We recommend that you enable oplog during off-peak hours.

1. Use Mongo Shell to connect to the source database.
2. Run the following commands to shut down the MongoDB service:

```
use admin
db.shutdownServer()
```

3. Run the following command to start the MongoDB service from the backend as a replica set:

```
mongod --port 27017 --dbpath /var/lib/mongodb --logpath /var/log/mongodb/mongod.log --r
eplSet rs0 --bind_ip 0.0.0.0 --auth --fork
```

> **Note**
>
> - The database path used by the preceding command is */var/lib/mongodb*. The log file path is */var/log/mongodb/mongod.log*. You must specify the paths based on your needs.
> - The command uses `0.0.0.0` as the associated IP address of the MongoDB service. This allows you to access the database by using all IP addresses. After the migration is complete, run the `kill` command to end the process, and start the MongoDB service by using the original configuration file.
> - The command enables authentication. You can access the database only after you pass the authentication.

4. Use Mongo Shell to connect to the source database again.
5. Run the following commands to initialize the replica set:

```
use admin
rs.initiate()
```

6. Wait until the role of the current node changes to primary, which indicates that oplog is enabled.

> **Note** You can run the `rs.printReplicationInfo()` command to view the status of oplog.

Procedure

1. Create a data migration instance.
2. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**.
	Database Type	Select **MongoDB**.

Section	Parameter	Description
Source Database	Hostname or IP Address	Enter the endpoint that is used to connect to the source database.
	Port Number	Enter the service port number of the source database.
	Database Name	Enter the name of the authentication database. The database account is created in this database.
	Database Account	Enter the account of the source database. The account must have the read permissions on the source database, admin database, and local database.
	Database Password	Enter the password of the source database account. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
Destination Database	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The destination region that you selected when you created the data migration instance. You cannot change the value of this parameter.
	Database Type	Select **MongoDB**.
	Hostname or IP Address	Enter the endpoint that is used to connect to the destination database.
	Port Number	Enter the service port number of the destination database.
	Database Name	Enter the name of the authentication database. The database account is created in this database. **Note** If the database account is root, enter admin.
	Database Account	Enter the account of the destination database. The account must have the read and write permissions on the destination database.

Section	Parameter	Description
	Database Password	Enter the password of the destination database account. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. If a whitelist is configured for the user-created MongoDB database, you must manually add the CIDR blocks of DTS servers to the whitelist of the database. To obtain the CIDR blocks of DTS servers, click **Copy to Clipboard** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the database, click **Next**.

   Note If you do not need to configure a whitelist for the user-created MongoDB database, ignore the preceding settings and click **Next**.

6. Select the migration types and the objects to be migrated.

Setting	Description
Select the migration types	◦ To perform only full data migration, select only **Full Data Migration**. ◦ To ensure service continuity during data migration, select both **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data to the source database during full data migration. This ensures data consistency between the source and destination databases.
Select the objects to be migrated	Select one or more objects from the **Available** section and click the › icon to move the objects to the **Selected** section. **Note** ◦ You can select columns, tables, or databases as the objects to be migrated. ◦ By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are migrated to the destination database. For more information, see Object name mapping. ◦ If you use the object name mapping feature on an object, other objects that are dependent on the object may fail to be migrated.

Setting	Description
Specify the retry time for failed connection to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 12 hours. You can specify the retry time based on your needs. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** When DTS retries a connection, you are charged for the DTS instance. We recommend that you specify the retry time based on your business needs. You can also release the DTS instance at your earliest opportunity after the source and destination instances are released.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. After the task passes the precheck, click **Next**.

   Note If Incremental Data Migration is not selected, wait until the migration task is completed. If Incremental Data Migration is selected, the migration task does not automatically stop. In this case, you must wait until **The migration task is not delayed** appears and manually stop the task.

3.3.18. Migrate data from a MaxCompute project to a MySQL data source

This topic describes how to configure data migration from a MaxCompute project to a MySQL data source in the Data Transmission Service (DTS) console. In this example, an ApsaraDB RDS for MySQL instance is used as the destination data source.

Usage notes

- DTS uses read and write resources of the source and destination databases during full data migration. This may increase the loads of the database servers. If the database performance is unfavorable, the specification is low, or the data volume is large, database services may become unavailable. For example, DTS occupies a large amount of read and write resources in the following cases: a large number of slow SQL queries are performed on the source database, the tables have no primary keys, or a deadlock occurs in the destination database. Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours. For example, you can migrate data when the CPU utilization of the source and destination databases is less than 30%.
- DTS automatically creates a database in the destination ApsaraDB RDS for MySQL instance. However,

if the name of the source database is invalid, you must manually create a database in the destination instance before you configure the data migration task.

> **Note** For more information about ApsaraDB RDS for MySQL instances and how to create a database in an ApsaraDB RDS for MySQL instance, see the "Create a database" topic in the documentation of ApsaraDB RDS for MySQL.

- To ensure data consistency, we recommend that you do not write data to the source MaxCompute project during full data migration.
- If a data migration task fails, DTS automatically resumes the task. Before you switch your workloads to the destination instance, stop or release the data migration task. Otherwise, the data in the source database overwrites the data in the destination instance after the task is resumed.
- MaxCompute does not support the PRIMARY KEY constraint. If network errors occur, DTS retries the data migration task. In this case, duplicate data records may be migrated to the destination tables that do not have primary keys.
- DTS cannot migrate incremental data from a MaxCompute project to an ApsaraDB RDS for MySQL instance.

> **Note** To ensure data consistency, we recommend that you do not write data to the source MaxCompute project during data migration.

Supported destination database types

You can use DTS to migrate data to the following types of MySQL databases:

- Self-managed database that is hosted on an Elastic Compute Service (ECS) instance
- Self-managed database that is connected over Express Connect, VPN Gateway, or Smart Access Gateway
- Self-managed database that is connected over Database Gateway

In this topic, an **ApsaraDB RDS for MySQL instance** is used to describe how to configure a data migration task. You can also follow the procedure to configure data migration tasks for other types of MySQL databases.

Migration types

Migration type	Description
Schema migration	DTS migrates the schemas of required objects to the destination database. DTS supports schema migration for views, tables, and databases. **Warning** In this topic, the source and the destination databases are heterogeneous databases. DTS does not ensure that the schemas of the source and destination databases are consistent after schema migration. We recommend that you evaluate the impact of data type conversion on your business. For more information, see Data type mappings between heterogeneous databases.

Migration type	Description
Full data migration	DTS migrates full data of required objects to the destination database. **Note** During full data migration, concurrent INSERT operations cause fragmentation in the tables of the destination database. After full data migration is complete, the used tablespace of the destination database is larger than that of the source database.

Procedure

1. Log on to the DTS console. For more information, see Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. In the upper-right corner of the page, click **Create Migration Task**.
4. In the Create DTS Instances dialog box, select a region, and enter the number of data migration instances that you want to create.

   Note In the Create DTS Instances dialog box, you can view the total number of instances, the number of existing instances, and the number of instances that can be created.

5. Click **Create**.
6. Find the data migration instance that you created, and click **Configure Migration Task** in the Actions column.
7. Configure a task name.

   The task name that DTS automatically generates. We recommend that you specify a descriptive name that makes it easy to identify the task. You do not need to specify a unique task name.
8. Configure the source and destination databases.

Section	Parameter	Description
Source Database	Instance Type	Select **MaxCompute**.
	Instance Region	The region in which the source MaxCompute project resides.
	Project	The name of the MaxCompute project. You can view the name of the MaxCompute project on the **Project management** tab in the DataWorks console.
	Instance Type	Select **RDS Instance**.
	Instance Region	The region in which the destination ApsaraDB RDS for MySQL instance resides.
	RDS Instance ID	The ID of the destination ApsaraDB RDS for MySQL instance.

Section	Parameter	Description
Destination Database	Database Account	The database account of the destination ApsaraDB RDS for MySQL instance. The account must have read and write permissions on the destination database.
	Database Password	The password of the database account. After you set the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. **Note** If the specified parameters are valid, the **Passed** message is displayed. If the **Failed** message is displayed, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

9. In the lower-right corner of the page, click **Set Whitelist and Next**.
10. In the lower-right corner of the page, click **Next**. In this step, the permissions on the MaxCompute project are granted to the migration account.
11. Select the migration type and the objects you want to migrate.

Setting	Description
Select migration types	Select **Schema Migration** and **Full Data Migration**. **Note** DTS cannot migrate incremental data from a MaxCompute project to an ApsaraDB RDS for MySQL instance.
Select the objects to be migrated	Select one or more objects from the **Source Objects** section and click the icon to add the objects to the **Selected Objects** section. **Note** - You can select columns, tables, or databases as the objects to be migrated. - By default, after an object is migrated to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to rename the objects that are migrated to the destination database. For more information, see Object name mapping. - If you use the object name mapping feature to rename an object, other objects that are dependent on the object may fail to be migrated.

Setting	Description
Specify the retry time range for failed connections to the source or destination database	By default, if DTS fails to connect to the source or destination database, DTS retries within the following 12 hours. You can specify the retry time range based on your business requirements. If DTS is reconnected to the source and destination databases within the specified time range, DTS resumes the data migration task. Otherwise, the data migration task fails.

12. In the lower-right corner of the page, click **Precheck**.

> Notice
> - Before you can start the data migration task, a precheck is performed. You can start the data migration task only after the task passes the precheck.
> - If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
>   - You can troubleshoot the issues based on the causes and run a precheck again.
>   - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

13. After the task passes the precheck, click **Next**.

3.3.19. Migrate data from a PolarDB-X 2.0 instance

3.3.19.1. Migrate data between PolarDB-X 2.0 instances

This topic describes how to configure data synchronization between PolarDB-X 2.0 instances in the Data Transmission Service (DTS) console.

Prerequisites

- A source PolarDB-X 2.0 instance and a destination PolarDB-X 2.0 instance that are compatible with MySQL 5.7 are created.
- The available storage capacity of the destination PolarDB-X 2.0 instance is larger than the total size of data stored in the source PolarDB-X 2.0 instance.

Limits

Category	Description

Limits on the source database	• The server to which the source database belongs must have sufficient outbound bandwidth. Otherwise, the data migration speed decreases.
	• The tables to be migrated must have PRIMARY KEY or UNIQUE constraints, and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	• If you select tables as the objects to be migrated and you need to modify the tables in the destination database, such as renaming tables or columns, you can migrate up to 1,000 tables in a single data migration task. If you run a task to migrate more than 1,000 tables, a request error occurs. In this case, we recommend that you split the tables and configure multiple tasks to migrate the tables, or configure a task to migrate the entire database.
	• If you want to migrate incremental data, you must make sure that the following requirements are met:
	  ◦ The binary logging feature is enabled. The value of the binlog_row_image parameter is set to full. Otherwise, error messages are returned during precheck and the data migration task cannot be started.
	  ◦ For an incremental data migration task, the binary logs of the source database must be stored for more than 24 hours. For a full data and incremental data migration task, the binary logs of the source database must be stored for at least seven days. After the full data migration is complete, you can set the retention period to more than 24 hours. Otherwise, Data Transmission Service (DTS) may fail to obtain the binary logs and the task may fail. In exceptional circumstances, data inconsistency or loss may occur. Make sure that you set the retention period of binary logs based on the preceding requirements. Otherwise, the Service Level Agreement (SLA) of DTS does not ensure service reliability or performance.
	• Limits on operations:
	  ◦ During schema migration and full data migration, do not perform DDL operations to change the schemas of databases or tables. Otherwise, the data migration task fails.
	  ◦ If you switch the network type of the PolarDB-X V2.0 instance during data migration, you must submit a ticket to update the network connection settings of the data migration task.
	  ◦ If you perform only full data migration, do not write data to the source database during data migration. Otherwise, data inconsistency between the source and destination databases occurs. To ensure data consistency, we recommend that you select schema migration, full data migration, and incremental data migration as the migration types.
	• The PolarDB-X V2.0 instance must be compatible with MySQL 5.7.

Category	Description
Other limits	• The destination PolarDB-X V2.0 instance must be compatible with MySQL 5.7. • Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours. During full data migration, DTS uses the read and write resources of the source and destination databases. This may increase the loads on the database servers. • During full data migration, concurrent INSERT operations cause fragmentation in the tables of the destination database. After the full data migration is complete, the size of used tablespace of the destination database is larger than that of the source database. • DTS attempts to resume data migration tasks that failed within the last seven days. Before you switch workloads to the destination instance, stop or release the data migration task. You can also execute the `REVOKE` statement to revoke the write permissions from the accounts used by DTS to access the destination instance. Otherwise, the data in the source database will overwrite the data in the destination database after the task is resumed.
Usage notes	DTS updates the *`dts_health_check`.`ha_health_check`* table in the source database as scheduled to move forward the binary log file position.

Migration types

- Schema migration

  DTS migrates the schemas of required objects from the source database to the destination database.

- Full data migration

  DTS migrates historical data of required objects from the source database to the destination database.

- Incremental data migration

  After full data migration is complete, DTS migrates incremental data from the source database to the destination database. Incremental data migration allows you to ensure service continuity when you migrate data between self-managed databases.

Permissions required for database accounts

Database	Schema migration	Full data migration	Incremental data migration
Source PolarDB-X 2.0 instance	The SELECT permission	The SELECT permission	The SELECT permission on the objects to be migrated, the REPLICATION SLAVE permission, and the REPLICATION CLIENT permission. DTS automatically grants these permissions to the database account.
Destination PolarDB-X 2.0 instance	The read and write permissions		

SQL operations that can be migrated

Operation type	SQL statements
DML	INSERT, UPDATE, and DELETE

Create a data migration task

1. Log on to the DTS console.
2. In the top navigation bar, move the pointer over **DTS**.
3. Choose **DTS (DTS)** > **Data Migration**.
4. On the Data Migration page, click **Create Task**.
5. In the dialog box that appears, set the **Source Database Type**, **Destination Database Type**, and **Quantity** parameters.

   3

Parameter	Description
Source Database	Select **PolarDB-X 2.0**.
Destination Database Type	Select **PolarDB-X 2.0**.
Quantity	The number of data migration instances that you want to create.

6. click **Purchase**.

Configure a data migration task.

1. In the **Data Migration Tasks** list, find the data migration instance that you create and click **Configure Task** in the rightmost column.
2. Configure the source and destination databases. Set the parameters described in the following table.

Section	Parameter	Description
Source Database	**Database Type**	Select **PolarDB-X 2.0**.
	Access Method	The value is fixed to **Alibaba Cloud Instance**.
	Instance Region	The region in which the source PolarDB-X 2.0 instance resides.
	Instance ID	The ID of the source PolarDB-X 2.0 instance.

Section	Parameter	Description
	Database Account	The database account of the source PolarDB-X 2.0 instance. For information about the permissions that are required for the account, see Permissions required for database accounts.
	Database Password	The password of the database account.
Destination Database	**Database Type**	Select **PolarDB-X 2.0**.
	Access Method	The value is fixed to **Alibaba Cloud Instance**.
	Instance Region	The region in which the destination PolarDB-X 2.0 instance resides.
	Instance ID	The ID of the destination PolarDB-X 2.0 instance.
	Database Account	The database account of the destination PolarDB-X 2.0 instance. For information about the permissions that are required for the account, see Permissions required for database accounts.
	Database Password	The password of the database account.

3. In the lower part of the page, click **Test Connectivity and Proceed**.

> **Warning**
>
> - If the source or destination database is an Alibaba Cloud instance, such as an ApsaraDB RDS for MySQL or ApsaraDB for MongoDB instance, DTS automatically adds the CIDR blocks of DTS servers to the whitelist of the instance. If the source or destination database is a self-managed database hosted on an Elastic Compute Service (ECS) instance, DTS automatically adds the CIDR blocks of DTS servers to the security group rules of the ECS instance. If the source or destination database is a self-managed database that is deployed in a data center or provided by a third-party cloud service provider, you must manually add the CIDR blocks of DTS servers to the whitelist of the database to allow DTS to access the database.
> - If the CIDR blocks of DTS servers are automatically or manually added to the whitelist of the database, Alibaba Cloud instance, or ECS security group rules, security risks may arise. Therefore, before you use DTS to migrate data, you must understand and acknowledge the potential risks and take preventive measures, including but not limited to the following measures: enhance the security of your username and password, limit the ports that are exposed, authenticate API calls, regularly check the whitelist or ECS security group rules and forbid unauthorized CIDR blocks, or connect the database to DTS by using Express Connect, VPN Gateway, or Smart Access Gateway.
> - After the DTS task is complete or released, we recommend that you manually detect and remove the added CIDR blocks from the whitelist of the database, Alibaba Cloud instance, or ECS security group rules.

4. Configure the parameters described in the following table.

Parameter	Description
Task Stages	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If Incremental Data Migration is not selected, we recommend that you do not write data to the source instance during data migration. This ensures data consistency between the source and destination instances.

Parameter	Description
Processing Mode of Conflicting Tables	○ **Precheck and Report Errors**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain identical table names, the precheck is passed. Otherwise, an error is returned during precheck and the data migration task cannot be started. **Note** If the source and destination databases contain identical table names and the tables in the destination database cannot be deleted or renamed, you can use the object name mapping feature to rename the tables that are synchronized to the destination database. For more information, see Object name mapping. ○ **Ignore Errors and Proceed**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore Errors and Proceed**, data consistency is not ensured and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, DTS does not migrate data records that have the same primary keys as data records in the destination database. ■ If the source and destination databases have different schemas, only specific columns are migrated or the data migration task fails.
Select Objects	Select one or more objects from the **Source Objects** section and click the icon to add the objects to the **Selected Objects** section.

5. In the lower part of the page, click **Next: Precheck and Start Task**.

Note

- Before you can start the data migration task, DTS performs a precheck. You can start the data migration task only after the task passes the precheck.
- If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
  - You can troubleshoot the issues based on the causes and run a precheck again.
  - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

6. If the value of **Success Rate** is **100%**, the data migration task starts to run.
   You can view the progress of the task on the Data Migration page.

   5

3.3.19.2. Migrate data from a PolarDB-X 2.0 instance to a MySQL instance

This topic describes how to migrate data from a PolarDB-X 2.0 instance to a MySQL instance by using Data Transmission Service (DTS). In this topic, an ApsaraDB RDS for MySQL instance is used as an example.

Prerequisites

- A PolarDB-X 2.0 instance that is compatible with MySQL 5.7 is created.
- The available storage space of the destination ApsaraDB RDS for MySQL instance is larger than the total size of data in the source PolarDB-X 2.0 instance.

Precautions

Category	Description

Category	Description

Limits on the source database	• The server to which the source database belongs must have sufficient outbound bandwidth. Otherwise, the data migration speed decreases.
	• The tables to be migrated must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	• If you select tables as the objects to be migrated and you need to edit the tables, such as renaming tables or columns, in the destination database, you can migrate up to 1,000 tables in a single data migration task. If you run a task to migrate more than 1,000 tables, a request error occurs. In this case, we recommend that you configure multiple tasks to migrate the tables in batches or configure a task to migrate the entire database.
	• If you want to migrate incremental data, make sure that the following requirements are met:
	  ◦ The binary logging feature must be enabled. The value of the binlog_row_image parameter must be set to full. Otherwise, error messages are returned during the precheck and the data migration task cannot be started.
	  ◦ For an incremental data migration task, binary logs of the source database must be stored for more than 24 hours. For a full data and incremental data migration task, binary logs of the source database must be stored for at least seven days. After full data migration is complete, you can set the retention period to more than 24 hours. Otherwise, DTS may fail to obtain the binary logs and the task may fail. In exceptional circumstances, data inconsistency or loss may occur. Make sure that you set the retention period of binary logs based on the preceding requirements. Otherwise, the service reliability or performance in the Service Level Agreement (SLA) of DTS may not be achieved.
	• Limits on operations to perform on the source database:
	  ◦ During schema migration and full data migration, do not perform DDL operations to change the schemas of databases or tables. Otherwise, the data migration task fails.
	  ◦ If you change the network type of the PolarDB-X V2.0 instance during data migration, you must submit a ticket to update the network connection settings of the data migration task.
	  ◦ If you perform only full data migration, do not write data to the source database during data migration. Otherwise, data inconsistency between the source and destination databases occurs. To ensure data consistency, we recommend that you select schema migration, full data migration, and incremental data migration as the migration types.
	• The PolarDB-X V2.0 instance must be compatible with MySQL 5.7.

Category	Description
Other limits	• Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours. During full data migration, DTS uses the read and write resources of the source and destination databases. This may increase the loads of database servers. • During full data migration, concurrent INSERT operations cause fragmentation in the tables of the destination database. After full data migration is complete, the size of used tablespace of the destination database is larger than that of the source database. • DTS attempts to resume data migration tasks that failed within the last seven days. Before you switch workloads to the destination instance, stop or release the data migration task. You can also execute the `REVOKE` statement to revoke the write permissions from the accounts used by DTS to access the destination instance. Otherwise, the data in the source database overwrites the data in the destination database after the task is resumed.
Special cases	If the destination database runs on an ApsaraDB RDS for MySQL instance, take note of the following limits: DTS automatically creates a destination database in the ApsaraDB RDS for MySQL instance. However, if the name of the source database is invalid, you must manually create a database in the ApsaraDB RDS for MySQL instance before you configure the data migration task. For more information, see **Create databases and accounts for an ApsaraDB RDS for MySQL instance** in the *user guide of ApsaraDB RDS for MySQL*.

Migration types

- Schema migration

  DTS migrates the schemas of required objects from the source database to the destination database.

- Full data migration

  DTS migrates historical data of required objects from the source database to the destination database.

- Incremental data migration

  After full data migration is complete, DTS migrates incremental data from the source database to the destination database. Incremental data migration allows you to ensure service continuity when you migrate data between self-managed databases.

SQL operations that can be migrated

Operation type	SQL statements
DML	INSERT, UPDATE, and DELETE

Permissions required for database accounts

Instance type	Schema migration	Full data migration	Incremental data migration
PolarDB-X 2.0	The SELECT permission	The SELECT permission	The SELECT permission on the objects to be migrated, the REPLICATION SLAVE permission, and the REPLICATION CLIENT permission. DTS automatically grants these permissions to the database account.
ApsaraDB RDS for MySQL	Read and write permissions		

Step 1: Create a data migration task

1. Log on to the DTS console.
2. In the top navigation bar, click **DTS**. In the left-side navigation pane, click **Data Migration**.
3. On the **Data Migration** page, click **Create Task**.
4. In the panel that appears, set the parameters. The following table describes the parameters.

   Data migration task

Parameter	Description
Source Database Type	Select **PolarDB-X 2.0**.
Source Region	Select the region in which the source instance resides.
Destination Database Type	Select **MySQL**.
Destination Region	Select the region in which the destination instance resides.
Quantity	Specify the number of data migration instances that you want to create at a time. Default value: 0.

5. Click **Purchase Instance**.

Step 2: Configure the data migration task

1. In the **Data Migration Tasks** list, find the data migration task that you created and click **Configure Task** in the rightmost column.
2. Configure the source and destination databases.

Section	Parameter	Description
N/A	**Task Name**	DTS automatically generates a task name. We recommend that you specify an informative name to identify the task. You do not need to use a unique task name.
Source Database	**Database Type**	Select **PolarDB-X 2.0**.
	Access Method	The value of this parameter is fixed to **Alibaba Cloud Instance**.
	Instance Region	The system displays the source region that you selected when you created the data migration task. You cannot change the value of this parameter.
	Instance ID	Select the ID of the source instance.
	Database Account	Enter the username of the database account of the source instance. For information about the permissions that are required for the account, see Permissions required for database accounts.
	Database Password	Enter the password of the database account.
Destination Database	**Database Type**	Select **MySQL**.
	Access Method	The value of this parameter is fixed to **Public IP Address**.
	Instance Region	The system displays the destination region that you selected when you created the data migration task. You cannot change the value of this parameter.
	Hostname or IP address	Enter the endpoint of the destination instance.
	Port Number	Enter the port number of the destination instance. Default value: 3306.
	Database Account	Enter the username of the database account of the destination instance. For information about the permissions that are required for the account, see Permissions required for database accounts.
	Database Password	Enter the password of the database account.

3. In the lower part of the page, click **Test Connectivity and Proceed**.

> **Warning**
>
> - If the source or destination database is in an instance hosted on Apsara Stack, such as an ApsaraDB RDS for MySQL or ApsaraDB for MongoDB instance, DTS automatically adds the CIDR blocks of DTS servers to the whitelist of the instance. If the source or destination database is a self-managed database hosted on an Elastic Compute Service (ECS) instance, DTS automatically adds the CIDR blocks of DTS servers to the security group rules of the ECS instance. If the source or destination database is a self-managed database that is deployed in a data center or provided by a third-party cloud service provider, you must manually add the CIDR blocks of DTS servers to the whitelist of the database to allow DTS to access the database.
> - If the CIDR blocks of DTS servers are automatically or manually added to the whitelist of the database or instance, or to the ECS security group rules, security risks may arise. Therefore, before you use DTS to migrate data, you must understand and acknowledge the potential risks and take preventive measures, including but not limited to the following measures: enhance the security of your username and password, limit the ports that are exposed, authenticate API calls, regularly check the whitelist or ECS security group rules and forbid unauthorized CIDR blocks, or connect the database to DTS by using Express Connect, VPN Gateway, or Smart Access Gateway.
> - After the DTS task is complete or released, we recommend that you manually detect and remove the added CIDR blocks from the whitelist of the database or instance, or from the ECS security group rules.

4. Select objects for the task and configure advanced settings.
   - Basic settings

Parameter	Description
Task Stages	

Parameter	Description
Processing Mode of Conflicting Tables	■ **Precheck and Report Errors**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain tables that have the same names, the precheck is passed. Otherwise, an error is returned during the precheck and the data migration task cannot be started. **Note** If the source and destination databases contain tables that have the same names and the tables in the destination database cannot be deleted or renamed, you can use the object name mapping feature to rename the tables that are migrated to the destination database. For more information, see Object name mapping. ■ **Ignore Errors and Proceed**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore Errors and Proceed**, data consistency is not ensured and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, DTS does not migrate data records that have the same primary keys as data records in the destination database. ■ If the source and destination databases have different schemas, only specific columns are migrated or the data migration task fails.
Select Objects	Select one or more objects from the **Source Objects** section and click the › icon to add the objects to the **Selected Objects** section. **Note** You can select columns, tables, or databases as the objects to be migrated. If you select tables or columns as the objects to be migrated, DTS does not migrate other objects such as views, triggers, and stored procedures to the destination database.

- Advanced settings

Setting	Description
Set Alerts	Specify whether to set alerts for the data migration task. If the task fails or the migration latency exceeds the threshold, the alert contacts will receive notifications.
Retry Time for Failed Connections	Specify the retry time range for failed connections. Unit: minutes. Valid values: 10 to 720. Default value: 720. We recommend that you set the retry time range to more than 30 minutes. If DTS reconnects to the source and destination databases within the specified time range, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** If a database serves as the source or destination database of multiple data migration tasks, the retry time range that is set later takes precedence.

5. In the lower part of the page, click **Next: Precheck and Start Task**.

   Note
   - Before you can start the data migration task, DTS performs a precheck. You can start the data migration task only after the task passes the precheck.
   - If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
     - You can troubleshoot the issues based on the causes and run a precheck again.
     - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

6. After the value of **Success Rate** becomes **100%**, the data migration task starts to run.
   You can view the progress of the task on the Data Migration page.

3.3.19.3. Migrate data from a PolarDB-X 2.0 instance to a Message Queue for Apache Kafka instance

This topic describes how to migrate data from a PolarDB-X 2.0 instance to a Message Queue for Apache Kafka instance by using Data Transmission Service (DTS). Message Queue for Apache Kafka has extended capabilities to process messages.

Prerequisites

- A PolarDB-X 2.0 instance is created.
- In the destination Message Queue for Apache Kafka instance, a topic is created to receive the migrated data.
- The available storage space of the destination Message Queue for Apache Kafka instance is larger

than the total size of data in the source PolarDB-X 2.0 instance.

Precautions

Category	Description

Limits on the source database	• The server to which the source database belongs must have sufficient outbound bandwidth. Otherwise, the data migration speed decreases.
	• The tables to be migrated must have PRIMARY KEY or UNIQUE constraints, and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	• If you select tables as the objects to be migrated and you need to modify the tables in the destination database, such as renaming tables or columns, you can migrate up to 1,000 tables in a single data migration task. If you run a task to migrate more than 1,000 tables, a request error occurs. In this case, we recommend that you split the tables and configure multiple tasks to migrate the tables, or configure a task to migrate the entire database.
	• If you want to migrate incremental data, you must make sure that the following requirements are met:
	  ◦ The binary logging feature is enabled. The value of the binlog_row_image parameter is set to full. Otherwise, error messages are returned during precheck and the data migration task cannot be started.
	  ◦ For an incremental data migration task, the binary logs of the source database must be stored for more than 24 hours. For a full data and incremental data migration task, the binary logs of the source database must be stored for at least seven days. After the full data migration is complete, you can set the retention period to more than 24 hours. Otherwise, Data Transmission Service (DTS) may fail to obtain the binary logs and the task may fail. In exceptional circumstances, data inconsistency or loss may occur. Make sure that you set the retention period of binary logs based on the preceding requirements. Otherwise, the Service Level Agreement (SLA) of DTS does not ensure service reliability or performance.
	• Limits on operations:
	  ◦ During schema migration and full data migration, do not perform DDL operations to change the schemas of databases or tables. Otherwise, the data migration task fails.
	  ◦ If you switch the network type of the PolarDB-X V2.0 instance during data migration, you must submit a ticket to update the network connection settings of the data migration task.
	  ◦ If you perform only full data migration, do not write data to the source database during data migration. Otherwise, data inconsistency between the source and destination databases occurs. To ensure data consistency, we recommend that you select schema migration, full data migration, and incremental data migration as the migration types.
	• The PolarDB-X V2.0 instance must be compatible with MySQL 5.7.

Category	Description
Other limits	• Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours. During full data migration, DTS uses the read and write resources of the source and destination databases. This may increase the loads on the database servers. • During full data migration, concurrent INSERT operations cause fragmentation in the tables of the destination database. After the full data migration is complete, the size of used tablespace of the destination database is larger than that of the source database. • DTS attempts to resume data migration tasks that failed within the last seven days. Before you switch workloads to the destination instance, stop or release the data migration task. You can also execute the `REVOKE` statement to revoke the write permissions from the accounts used by DTS to access the destination instance. Otherwise, the data in the source database will overwrite the data in the destination database after the task is resumed.
Usage notes	DTS updates the *`dts_health_check`.`ha_health_check`* table in the source database as scheduled to move forward the binary log file position.

Migration types

- Schema migration

  DTS migrates the schemas of required objects from the source database to the destination database.

- Full data migration

  DTS migrates historical data of required objects from the source database to the destination database.

- Incremental data migration

  After full data migration is complete, DTS migrates incremental data from the source database to the destination database. Incremental data migration allows you to ensure service continuity when you migrate data between self-managed databases.

Permissions required for database accounts

Instance type	Schema migration	Full data migration	Incremental data migration
PolarDB-X 2.0	The SELECT permission	The SELECT permission	The SELECT permission on the objects to be migrated, the REPLICATION SLAVE permission, and the REPLICATION CLIENT permission. DTS automatically grants these permissions to the database account.

SQL operations that can be migrated

Operation type	SQL statements
DML	INSERT, UPDATE, and DELETE

Step 1: Create a data migration task

1. Log on to the Data Management (DMS) console.
2. In the top navigation bar, click **DTS**. In the left-side navigation pane, click **Data Migration**.
3. On the **Data Migration** page, click **Create Task**.
4. In the panel that appears, set the parameters. The following table describes the parameters.

Parameter	Description
Source Database Type	Select **PolarDB-X 2.0**.
Source Region	Select the region in which the source instance resides.
Destination Database Type	Select **Kafka**.
Destination Region	Select the region in which the destination instance resides.
Quantity	Specify the number of data migration instances that you want to create at a time. Default value: 0.

5. Click **Purchase Instance**.

Step 2: Configure the data migration task

1. In the **Data Migration Tasks** list, find the data migration task that you created and click **Configure Task** in the rightmost column.
2. Configure the source and destination databases.

Section	Parameter	Description
N/A	**Task Name**	DTS automatically generates a task name. We recommend that you specify an informative name to identify the task. You do not need to use a unique task name.
Source Database	**Database Type**	Select **PolarDB-X 2.0**.
	Access Method	Select **Alibaba Cloud Instance**.
	Instance Region	Select the region in which the source **PolarDB-X 2.0** instance resides.
	Instance ID	Select the ID of the source **PolarDB-X 2.0** instance.

Section	Parameter	Description
	Database Account	Enter the username of the database account of the source **PolarDB-X 2.0** instance. For information about the permissions that are required for the account, see Permissions required for database accounts.
	Database Password	Enter the password of the database account.
Destination Database	**Database Type**	Select **Kafka**.
	Access Method	The value of this parameter is fixed to **Public IP Address**. **Note** If you select Public IP Address, you can use a specific IP address to connect to the destination Message Queue for Apache Kafka instance over the classic network.
	Instance Region	Select the region in which the destination Message Queue for Apache Kafka instance resides.
	Hostname or IP address	Enter an IP address of the destination Message Queue for Apache Kafka instance. **Note** To obtain an IP address, perform the following operations: Log on to the Message Queue for Apache Kafka console and go to the Instance Details page of the Message Queue for Apache Kafka instance. On the **Instance Details** page, obtain an IP address from the **Default Endpoint** parameter.
	Port Number	Enter the service port number of the destination Message Queue for Apache Kafka instance. Default value: 9092.
	Database Account	Enter the username of the database account of the destination Message Queue for Apache Kafka instance. **Note** If the Message Queue for Apache Kafka instance is a **VPC-connected** instance, you do not need to set the **Database Account** or **Database Password** parameter.
	Database Password	Enter the password of the database account.
	Kafka Version	Select the version of the destination Message Queue for Apache Kafka instance.
	Encryption	Specifies whether to encrypt the connection. Select **Non-encrypted** or **SCRAM-SHA-256** based on your business and security requirements.

Section	Parameter	Description
	Topic	Select a topic from the drop-down list.
	Topic That Stores DDL Information	Select a topic from the drop-down list. The topic is used to store the DDL information. If you do not set this parameter, the DDL information is stored in the topic that is specified by the **Topic** parameter.
	Use Kafka Schema Registry	Specifies whether to use Kafka Schema Registry, which provides a serving layer for your metadata. It provides a RESTful API for storing and retrieving your Avro schemas. Valid values: ○ **No**: does not use Kafka Schema Registry. ○ **Yes**: uses Kafka Schema Registry. In this case, you must enter the URL or IP address that is registered in Kafka Schema Registry for your Avro schemas.

3. In the lower part of the page, click **Test Connectivity and Proceed**.

> **Warning**
>
> - If the source or destination database is in an instance hosted on Apsara Stack, such as an ApsaraDB RDS for MySQL or ApsaraDB for MongoDB instance, DTS automatically adds the CIDR blocks of DTS servers to the whitelist of the instance. If the source or destination database is a self-managed database hosted on an Elastic Compute Service (ECS) instance, DTS automatically adds the CIDR blocks of DTS servers to the security group rules of the ECS instance. If the source or destination database is a self-managed database that is deployed in a data center or provided by a third-party cloud service provider, you must manually add the CIDR blocks of DTS servers to the whitelist of the database to allow DTS to access the database.
> - If the CIDR blocks of DTS servers are automatically or manually added to the whitelist of the database or instance, or to the ECS security group rules, security risks may arise. Therefore, before you use DTS to migrate data, you must understand and acknowledge the potential risks and take preventive measures, including but not limited to the following measures: enhance the security of your username and password, limit the ports that are exposed, authenticate API calls, regularly check the whitelist or ECS security group rules and forbid unauthorized CIDR blocks, or connect the database to DTS by using Express Connect, VPN Gateway, or Smart Access Gateway.
> - After the DTS task is complete or released, we recommend that you manually detect and remove the added CIDR blocks from the whitelist of the database or instance, or from the ECS security group rules.

4. Select objects for the task and configure advanced settings.
   - Basic settings

Parameter	Description
Task Stages	

Parameter	Description
Processing Mode of Conflicting Tables	■ **Precheck and Report Errors**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain tables that have the same names, the precheck is passed. Otherwise, an error is returned during the precheck and the data migration task cannot be started. **Note** If the source and destination databases contain tables that have the same names and the tables in the destination database cannot be deleted or renamed, you can use the object name mapping feature to rename the tables that are migrated to the destination database. For more information, see Object name mapping. ■ **Ignore Errors and Proceed**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore Errors and Proceed**, data consistency is not ensured and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, DTS does not migrate data records that have the same primary keys as data records in the destination database. ■ If the source and destination databases have different schemas, only specific columns are migrated or the data migration task fails.
Data Format in Kafka	Select the format in which data is stored in the Message Queue for Apache Kafka instance. ■ If you select **DTS Avro**, data is parsed based on the schema definition of DTS Avro. For more information about the schema definition, visit GitHub. ■ If you select **Canal Json**, data is parsed into the Canal JSON format. For more information about the parameters that are used to parse data into the Canal JSON format, see the section in this topic.
Policy for Shipping Data to Kafka Partitions	Select the policy based on which data is migrated to the Message Queue for Apache Kafka instance.
Select Objects	Select one or more objects from the **Source Objects** section and click the icon to add the objects to the **Selected Objects** section. **Note**

Parameter	Description
Rename Databases and Tables	■ To rename an object that you want to migrate to the destination instance, right-click the object in the **Selected Objects** section. For more information, see Object name mapping. ■ To rename multiple objects at a time, click Batch Edit in the upper-right corner of the **Selected Objects** section. For more information, see Object name mapping. **Note** If you use the object name mapping feature to rename an object, other objects that are dependent on the object may fail to be migrated.
Filter data	Specify WHERE conditions to filter data. For more information, see Specify an SQL condition to filter data.
Select the SQL operations to be migrated	In the **Selected Objects** section, right-click an object. In the dialog box that appears, select the DML operations to be migrated.

Parameters used to parse data into the Canal JSON format

Parameter	Description
`database`	The name of the database.
`es`	The time when the operation is performed on the database. The value is a 13-bit UNIX timestamp. Unit: milliseconds. **Note** You can use a search engine to obtain a UNIX timestamp converter.
`id`	The serial number of the operation.
`isDdl`	Indicates whether the operation is a DDL operation. Valid values: ■ true: The operation is a DDL operation. ■ false: The operation is not a DDL operation.
`mysqlType`	The data type of the field.
`old`	The data before update.
`pkNames`	The name of the primary key.
`sql`	The SQL statement.

Parameter	Description
`sqlType`	The data type of the field after conversion. For example, the type LONG is converted from UNSIGNED INTEGER and BIGDECIMAL from UNSIGNED LONG.
`table`	The table name.
`ts`	The time when the data is written to the destination database. The value is in the 13-bit UNIX timestamp format. Unit: milliseconds. **Note** You can use a search engine to obtain a UNIX timestamp converter.
`type`	The operation type. Valid values: DELETE, UPDATE, and INSERT.

- Advanced settings

Parameter	Description
Set Alerts	Specify whether to set alerts for the data migration task. If the task fails or the migration latency exceeds the threshold, the alert contacts will receive notifications.
Capitalization of Object Names in Destination Instance	Specify the capitalization of database names, table names, and column names in the destination instance. By default, **DTS default policy** is selected. You can select other options to make sure that the capitalization of object names is consistent with that in the source or destination database. For more information, see Specify the capitalization of object names in the destination instance.
Retry Time for Failed Connections	Specify the retry time range for failed connections. Unit: minutes. Valid values: 10 to 720. Default value: 720. We recommend that you set the retry time range to more than 30 minutes. If DTS reconnects to the source and destination databases within the specified time range, DTS resumes the data migration task. Otherwise, the data migration task fails. **Note** If a database serves as the source or destination database of multiple data migration tasks, the retry time range that is set later takes precedence.

5. In the lower part of the page, click **Next: Precheck and Start Task**.

> Note
>
> - Before you can start the data migration task, DTS performs a precheck. You can start the data migration task only after the task passes the precheck.
> - If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
>   - You can troubleshoot the issues based on the causes and run a precheck again.
>   - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

6. After the value of **Success Rate** becomes **100%**, the data migration task starts to run.

3.3.19.4. Migrate data from a PolarDB-X 2.0 instance to an AnalyticDB for PostgreSQL instance

This topic describes how to migrate data from a PolarDB-X 2.0 instance to an AnalyticDB for PostgreSQL instance by using Data Transmission Service (DTS).

Prerequisites

- A source PolarDB-X 2.0 instance that is compatible with MySQL 5.7 is created.
- A destination AnalyticDB for PostgreSQL instance is created. For more information, see Create an AnalyticDB for PostgreSQL instance.

Limits

Category	Description

Category	Description

Limits on the source database	• The server to which the source database belongs must have sufficient outbound bandwidth. Otherwise, the data migration speed decreases.
	• The tables to be migrated must have PRIMARY KEY or UNIQUE constraints, and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	• If you select tables as the objects to be migrated and you need to modify the tables in the destination database, such as renaming tables or columns, you can migrate up to 1,000 tables in a single data migration task. If you run a task to migrate more than 1,000 tables, a request error occurs. In this case, we recommend that you split the tables and configure multiple tasks to migrate the tables, or configure a task to migrate the entire database.
	• If you want to migrate incremental data, you must make sure that the following requirements are met:
	  ◦ The binary logging feature is enabled. The value of the binlog_row_image parameter is set to full. Otherwise, error messages are returned during precheck and the data migration task cannot be started.
	  ◦ For an incremental data migration task, the binary logs of the source database must be stored for more than 24 hours. For a full data and incremental data migration task, the binary logs of the source database must be stored for at least seven days. After the full data migration is complete, you can set the retention period to more than 24 hours. Otherwise, Data Transmission Service (DTS) may fail to obtain the binary logs and the task may fail. In exceptional circumstances, data inconsistency or loss may occur. Make sure that you set the retention period of binary logs based on the preceding requirements. Otherwise, the Service Level Agreement (SLA) of DTS does not ensure service reliability or performance.
	• Limits on operations:
	  ◦ During schema migration and full data migration, do not perform DDL operations to change the schemas of databases or tables. Otherwise, the data migration task fails.
	  ◦ If you switch the network type of the PolarDB-X V2.0 instance during data migration, you must submit a ticket to update the network connection settings of the data migration task.
	  ◦ If you perform only full data migration, do not write data to the source database during data migration. Otherwise, data inconsistency between the source and destination databases occurs. To ensure data consistency, we recommend that you select schema migration, full data migration, and incremental data migration as the migration types.
	• The PolarDB-X V2.0 instance must be compatible with MySQL 5.7.

Category	Description
Other limits	• Before you migrate data, evaluate the impact of data migration on the performance of the source and destination databases. We recommend that you migrate data during off-peak hours. During full data migration, DTS uses the read and write resources of the source and destination databases. This may increase the loads on the database servers. • During full data migration, concurrent INSERT operations cause fragmentation in the tables of the destination database. After the full data migration is complete, the size of used tablespace of the destination database is larger than that of the source database. • DTS attempts to resume data migration tasks that failed within the last seven days. Before you switch workloads to the destination instance, stop or release the data migration task. You can also execute the `REVOKE` statement to revoke the write permissions from the accounts used by DTS to access the destination instance. Otherwise, the data in the source database will overwrite the data in the destination database after the task is resumed.

Migration types

- Schema migration

  DTS migrates the schemas of required objects from the source database to the destination database.

- Full data migration

  DTS migrates historical data of required objects from the source database to the destination database.

- Incremental data migration

  After full data migration is complete, DTS migrates incremental data from the source database to the destination database. Incremental data migration allows you to ensure service continuity when you migrate data between self-managed databases.

SQL operations that can be migrated

Operation type	SQL statements
DML	INSERT, UPDATE, and DELETE

Create a data migration task

1. Log on to the DTS console.
2. In the top navigation bar, move the pointer over **DTS**.
3. Choose **DTS (DTS)** > **Data Migration**.
4. On the Data Migration page, click **Create Task**.
5. In the dialog box that appears, set the **Source Database Type**, **Destination Database Type**, and **Quantity** parameters.

Parameter	Description
Source Database Type	Select **PolarDB-X 2.0**.
Destination Database Type	Select **AnalyticDB for PostgreSQL**.
Quantity	The number of data migration instances that you want to create.

6. click **Purchase**.

Configure a data migration task.

1. In the **Data Migration Tasks** list, find the data migration instance that you create and click **Configure Task** in the rightmost column.
2. Configure the source and destination databases.

Section	Parameter	Description
Source Database	**Database Type**	Select **PolarDB-X 2.0**.
	Access Method	The value is fixed to **Alibaba Cloud Instance**.
	Instance Region	The region in which the source PolarDB-X 2.0 instance resides.
	Instance ID	The ID of the source PolarDB-X 2.0 instance.
	Database Account	The database account of the source PolarDB-X 2.0 instance. For more information about the permissions required for the account, see Permissions required for database accounts.
	Database Password	The password of the database account.
Destination Database	**Database Type**	Select **AnalyticDB for PostgreSQL**.
	Access Method	The value is fixed to **Alibaba Cloud Instance**.
	Instance Region	The region in which the destination AnalyticDB for PostgreSQL instance resides.
	Instance ID	The ID of the destination AnalyticDB for PostgreSQL instance.

Section	Parameter	Description
	Database Account	The database account of the destination AnalyticDB for PostgreSQL instance. For more information about the permissions required for the account, see Permissions required for database accounts.
	Database Password	The password of the database account.

3. In the lower part of the page, click **Test Connectivity and Proceed**.

> **Warning**
>
> - If the source or destination database is an Alibaba Cloud instance, such as an ApsaraDB RDS for MySQL or ApsaraDB for MongoDB instance, DTS automatically adds the CIDR blocks of DTS servers to the whitelist of the instance. If the source or destination database is a self-managed database hosted on an Elastic Compute Service (ECS) instance, DTS automatically adds the CIDR blocks of DTS servers to the security group rules of the ECS instance. If the source or destination database is a self-managed database that is deployed in a data center or provided by a third-party cloud service provider, you must manually add the CIDR blocks of DTS servers to the whitelist of the database to allow DTS to access the database.
> - If the CIDR blocks of DTS servers are automatically or manually added to the whitelist of the self-managed database, Alibaba Cloud instance, or ECS security group rules, security risks may arise. Therefore, before you use DTS to migrate data, you must understand and acknowledge the potential risks and take preventive measures, including but not limited to the following measures: Enhance the security of your username and password, limit the ports that are exposed, authenticate API calls, regularly check the whitelist or ECS security group rules and forbid unauthorized CIDR blocks, or connect to DTS by using Express Connect, VPN Gateway, or Smart Access Gateway.
> - After the DTS task is complete or released, we recommend that you manually detect and remove the added CIDR blocks from the whitelist of the self-managed database, Alibaba Cloud instance, or ECS security group rules.

4. Configure objects to be migrated and advanced settings. The following table describes the parameters.

Parameter	Description

Parameter	Description
Task Stages	◦ To perform only full data migration, select **Schema Migration** and **Full Data Migration**. ◦ To ensure service continuity during data migration, select **Schema Migration**, **Full Data Migration**, and **Incremental Data Migration**. **Note** If Incremental Data Migration is not selected, we recommend that you do not write data to the source instance during data migration. This ensures data consistency between the source and destination instances.
Processing Mode of Conflicting Tables	◦ **Precheck and Report Errors**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain tables that have the same names, the precheck is passed. Otherwise, an error is returned during precheck and the data migration task cannot be started. **Note** If the source and destination databases contain tables that have the same names and the tables in the destination database cannot be deleted or renamed, you can use the object name mapping feature to rename the tables that are migrated to the destination database. For more information, see Object name mapping. ◦ **Ignore Errors and Proceed**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore Errors and Proceed**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, DTS does not migrate data records that have the same primary keys as data records in the destination database. ■ If the source and destination databases have different schemas, only specific columns are migrated or the data migration task fails.

Parameter	Description
Select Objects	Select one or more objects from the **Source Objects** section and click the icon to add the objects to the **Selected Objects** section.

5. In the lower part of the page, click **Next: Precheck and Start Task**.

> **Note**
> - Before you can start the data migration task, DTS performs a precheck. You can start the data migration task only after the task passes the precheck.
> - If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
>   - You can troubleshoot the issues based on the causes and run a precheck again.
>   - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

6. If the value of **Success Rate** is **100%**, the data migration task starts to run.
You can view the progress of the task on the Data Migration page.

5

3.4. Manage data migration tasks

3.4.1. Object name mapping

Data Transmission Service (DTS) provides the object name mapping feature. You can use this feature to change the names of one or more objects that are migrated to the destination instance. This topic describes how to use the object name mapping feature when you configure a data migration task.

Limits

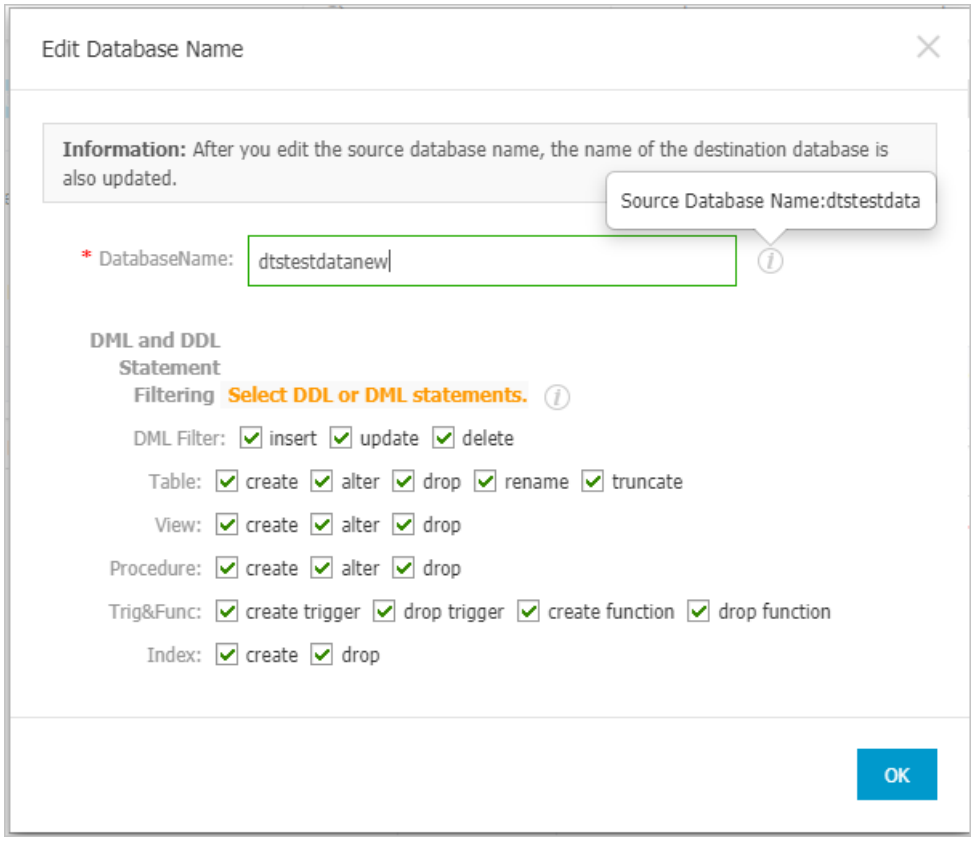
You can use the object name mapping feature only when a data migration task is configured and the current step is **Configure Migration Types and Objects**.

> **Note** Do not use the object name mapping feature after a data migration task is started. Otherwise, data may fail to be migrated.

Procedure

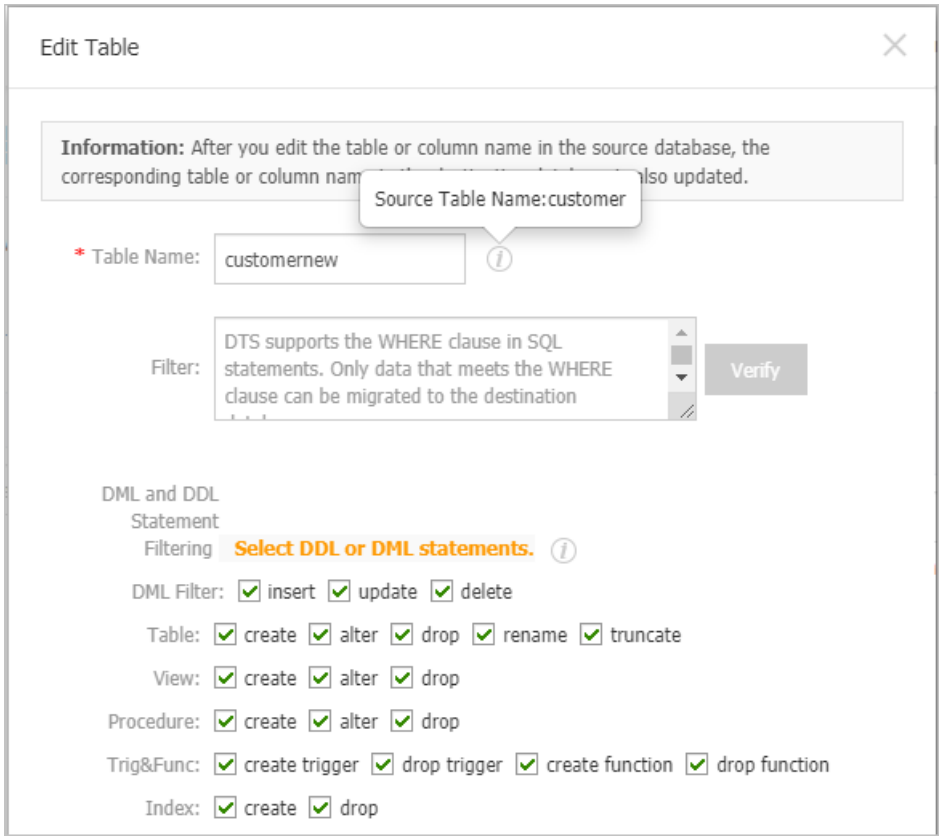
1. In the **Configure Migration Types and Objects** step, move the required objects to the **Selected** section, move the pointer over a database or table, and then click **Edit**.
2. In the dialog box that appears, specify a name for the object in the destination instance.
   - Database name mapping

In the **Edit Database Name** dialog box that appears, enter the database name that you want to use in the destination instance.



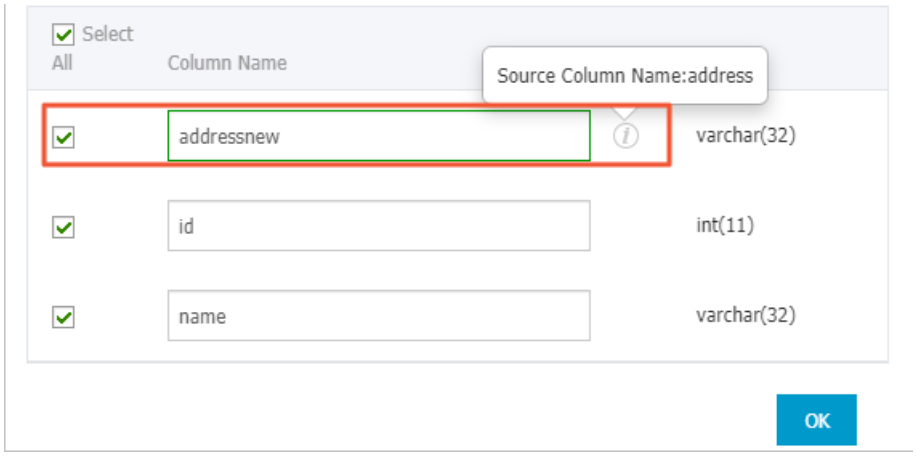
- Table name mapping

  In the **Edit Table** dialog box that appears, enter the table name that you want to use in the destination instance.



- Column name mapping

  In the **Edit Table** dialog box that appears, enter a new name for each column.



> **Note** In this step, you can clear the options of columns that do not need to be synchronized.

3. Click **OK**.
4. Configure other parameters that are required for the data migration task.

3.4.2. Specify an SQL condition to filter data

This topic describes how to specify an SQL condition to filter the data of a specific table when you configure a data migration task.

The SQL condition takes effect only within the table that you select. DTS migrates only the data that meets the SQL condition to the destination database. This feature is applicable to scenarios such as regular data migration and table partitioning.

Limits

An SQL condition applies only to full data migration. If you select **incremental data migration** as the migration type, the SQL condition does not filter incremental data.

Specify an SQL condition

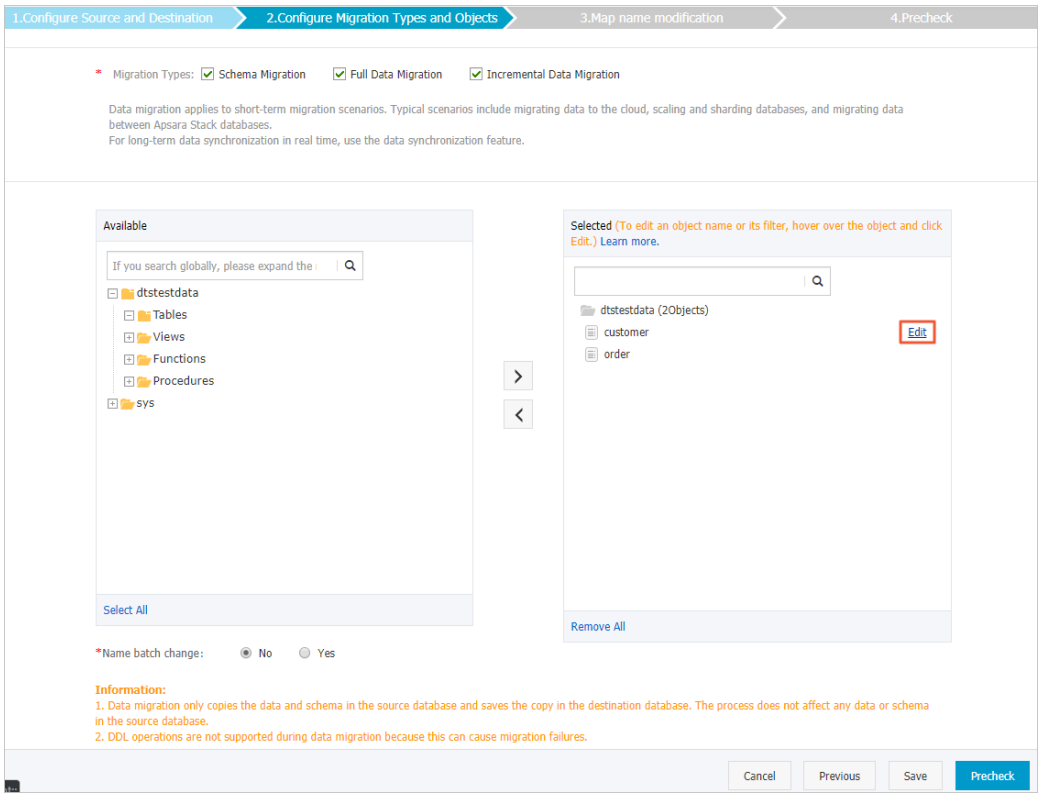
You can specify an SQL condition in the **Configure Migration Types and Objects** step when you configure a data migration task.

To filter the data of a specific table by using an SQL condition, you must select the table as the object that you want to migrate. You cannot select a database as the object. To specify an SQL condition, perform the following steps.

Procedure

1. In the **Configure Migration Types and Objects** step, move the pointer over a table in the **Selected** section. The **Edit** button appears, as shown in Edit button.

   Edit button



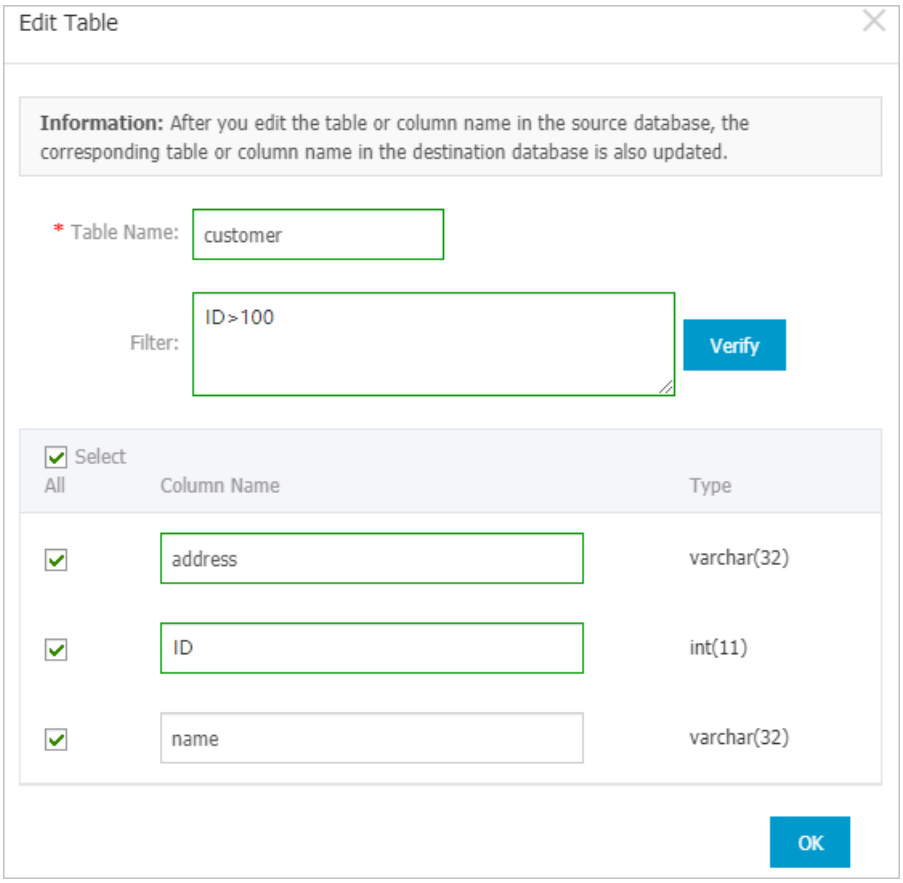
2. Click **Edit**. The Edit Table dialog box appears.

Modify an SQL condition

The SQL conditions in DTS are the same as the standard SQL WHERE conditions for databases. You can use SQL conditions to perform operations and run basic functions.

Enter an SQL condition in the text box. For example, you can enter id>1000 to migrate the records whose IDs are greater than 1,000 to the destination instance, as shown in Modify an SQL condition.

Modify an SQL condition



After the SQL condition is specified, click **OK**.

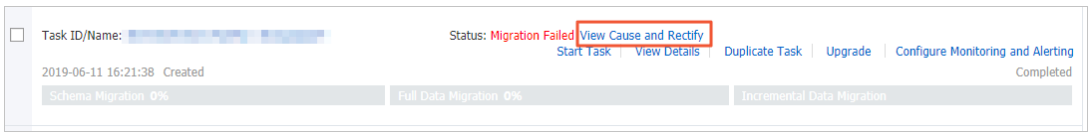
3.4.3. Troubleshoot a failed data migration task

This topic describes how to troubleshoot a failed data migration task. You can use this feature if your data migration task is in the **Migration Failed** state during schema migration or full data migration.

Troubleshoot a failed task during schema migration

DTS supports data migration between heterogeneous data sources. However, if you migrate data of unsupported types to the destination instance during schema migration, the task may fail.

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. Use one of the following methods to troubleshoot the failed task:
   - Method 1
     a. Find the task and click **View Cause and Rectify**.



     b. Troubleshoot the issue based on the cause that is displayed in the View Cause and Rectify message. For example, you can troubleshoot an issue by modifying the schema syntax.

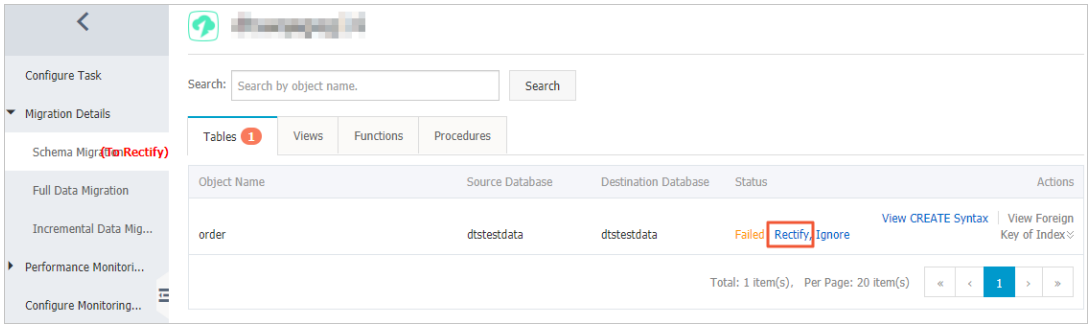
c. Click **Restart Task**.

- Method 2

a. Click the instance ID or task name.

b. In the left-side navigation pane, choose **Migration Details > Schema Migration**.

c. On the **Schema Migration** page, find the object that causes the migration failure and click **Rectify** in the Status column.



d. Troubleshoot the issue based on the cause that is displayed in the **Rectify** dialog box. For example, you can troubleshoot an issue by modifying the schema syntax.

e. Click **Rectify**.

> **Note**
>
> - If the failure persists, the **Rectify** dialog box does not close and shows the failure cause. You must continue troubleshooting based on the failure cause until the troubleshooting is successful.
> - If the troubleshooting is successful, the **Schema Migration** page appears and the status of the object changes to **Finished**.

4. If no objects are in the Failed state, DTS proceeds with the data migration task, for example, entering the full data migration process.

Troubleshoot a failed task during full data migration

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. Find the task and click **View Cause and Rectify**.

   DTS allows you to troubleshoot a task that fails during full data migration due to the following reasons.

   > **Note** If a task fails during full data migration due to other reasons, DTS provides only the **Ignore** option. The object that causes the failure is not migrated to the destination database.

   - The connection to the source or destination database failed or timed out.

     Troubleshoot the issue, make sure that the connection is successful, and then click **Restart Task**.

   - The storage space of the destination instance is insufficient or the instance is locked.

Upgrade the specification of the destination instance or clear the log space, and then click **Restart Task**.

- MyISAM tables in the source database are corrupted.

  Troubleshoot the issue in the source database, and then click **Restart Task**.

4. In the dialog box that appears, troubleshoot the issue based on the failure cause.
5. Click **Restart Task**.

3.4.4. Specify the capitalization of object names in the destination instance

When you configure a data migration or synchronization task, you can specify the capitalization of database names, table names, and column names in the destination instance.

Capitalization policies

Data Transmission Service (DTS) supports the following capitalization policies for object names:

- DTS default policy

  DTS configures the capitalization of database names, table names, and column names in the destination instance based on the database type and related parameters, such as `lower_case_table_names`.

Destination database type	Destination database parameter	Capitalization of object names in the destination instance
○ Self-managed MySQL database or ApsaraDB RDS for MySQL ○ PolarDB-X ○ AnalyticDB for MySQL V3.0	The `lower_case_table_names` parameter is set to 1 or 2.	Database names and table names in the destination instance are in lowercase. The capitalization of column names in the destination instance is the same as that in the source instance.
	The `lower_case_table_names` parameter is set to 0.	The capitalization of database names, table names, and column names in the destination instance is the same as that in the source instance.
○ AnalyticDB for MySQL V2.0 ○ DataHub ○ MaxCompute	N/A	Database names, table names, and column names in the destination instance are in lowercase.

Destination database type	Destination database parameter	Capitalization of object names in the destination instance
Self-managed Oracle database	N/A	Database names, table names, and column names in the destination instance are in uppercase.
◦ Self-managed SQL Server database or ApsaraDB RDS for SQL Server ◦ PolarDB for Oracle ◦ Self-managed PostgreSQL database or ApsaraDB RDS for PostgreSQL ◦ AnalyticDB for PostgreSQL ◦ Self-managed MongoDB database or ApsaraDB for MongoDB ◦ Self-managed Redis database or ApsaraDB for Redis	N/A	Database names and table names in the destination instance are in lowercase. The capitalization of column names in the destination instance is the same as that in the source instance.

- Consistent with the source database

  The capitalization of database names, table names, and column names in the destination instance is the same as that in the source instance.

- Consistent with the default policy of the destination database (uppercase)

  The names of all the databases, tables, and columns that are migrated or synchronized to the destination instance are in uppercase.

- Consistent with the default policy of the destination database (lowercase)

  The names of all the databases, tables, and columns that are migrated or synchronized to the destination instance are in lowercase.

3.5. Precheck items

3.5.1. Source database connectivity

DTS checks whether DTS servers can connect to the source database. DTS creates a connection to the source database by using the JDBC protocol. If the connection fails, the data migration task fails to pass the connectivity check.

The migration task may fail to pass the connectivity check because of the following reasons:

- The database account or password that is specified in the data migration task is invalid.

  Troubleshooting:

Find a server that can connect to the source database. On the server, enter the database account and password that are specified in the data migration task to check whether the account and password are valid. If the database account or password is invalid, the following error message is displayed: Access deny.

Solution:

Log on to the DTS console, modify the database account and password, and then run a precheck again.

- The DTS servers are disallowed to access the source database.

  Troubleshooting:

  - Find a server that can connect to the source database. On the server, enter the database account and password that are specified in the data migration task to check whether the connection is successful. Only authorized DTS servers can connect to the source database. If the CIDR block of a DTS server is not included in the whitelist of the source database, the DTS server cannot connect to the source database.
  - If the source database is a MySQL database, use a MySQL client to connect to the database and run the `SELECT HOST FROM mysql.user WHERE user='Account', password='Password';` command. If the CIDR blocks of DTS servers are not included in the whitelist of the source database, the query result of the preceding command is not %.

  Solution:

  - If the source database is a MySQL database, run the `GRANT ALL ON . TO 'Account'@''%' IDENTIFIED BY 'Password';` command to authorize the database account. Replace Account and Password in the preceding command with your database account and password. After the account is authorized, run a precheck again.

- A firewall is configured on the source database server.

  Troubleshooting: If the server where the source database resides runs Linux, run the `iptables -L` command in the shell to check whether a firewall is configured for the server. If the server where the source database resides runs Windows, find Windows Defender Firewall from the Control Panel and check whether a firewall is configured for the server.

  Solution:

  Disable the firewall and run a precheck again.

- The network between DTS servers and the source database is unavailable.

  If the failure persists, you can check whether the network between DTS servers and the source database is available. In this case, we recommend that you contact Alibaba Cloud engineers by submitting a ticket.

3.5.2. Check the destination database connectivity

This check item checks whether the DTS server can connect to the destination database for migration. DTS creates a connection to the destination database by using the JDBC protocol. If the connection fails, the check item fails.

The destination database connectivity precheck may fail for the following reasons:

- An incorrect account or password is provided when a migration task is created.

  Diagnostics:

  On any network-ready server that can connect to the destination database, use the account and password specified for creating the migration task to connect to the destination database through client software. Check whether the connection succeeds. If an error is reported for the connection and the error message contains Access deny, the account or password is incorrect.

  Troubleshooting:

  Modify the migration task in the DTS console, correct the account and password, and perform the precheck again.

- There is no connectivity between the DTS server and destination database.

  If you check that the password and account are correct, the check item may fail because there is no connectivity between the DTS server and the destination database. In this case, contact the DTS engineers on duty.

3.5.3. Binary logging configurations of the source database

Whether binary logging is enabled in the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether binary logging is enabled in the source database. If binary logging is disabled in the source database, the check result is Failed.

Troubleshooting: Run the `log_bin=mysql_bin` command to modify the configuration file of the source database. Restart the source database and run a precheck again.

Binary log format of the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the binary log format is set to ROW in the source database. If the binary log format is not set to ROW in the source database, the check result is Failed.

Troubleshooting: Run the `set global binlog_format=ROW` command in the source database and run a precheck again. We recommend that you restart the MySQL process. Otherwise, data loss may occur because sessions will continue to be written in a non-ROW mode.

Binary log files in the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether specific binary log files are removed from the source database. If binary log files in the source database are incomplete, the check result is Failed.

Troubleshooting: Run the `PURGE BINARY LOGS TO 'The name of the first binary log file that is not deleted'` command in the source database and run a precheck again.

To find the binary log files that are removed from the source database, click the info icon next to the failed item. In the **View Details** dialog box, the names of deleted binary log files are displayed.

Parameter binlog_row_image of the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the value of the binlog_row_image parameter in the source database is set to FULL. This parameter indicates whether the full image is recorded. If the full image is not recorded in binary log files of the source database, the check result is Failed.

Troubleshooting: Run the `set global binlog_row_image=FULL` command in the source database and run a precheck again.

3.5.4. Integrity of the FOREIGN KEY constraints

DTS checks whether the parent table on which a child table depends is included in the selected objects. The precheck allows DTS to protect the integrity of the FOREIGN KEY constraints.

If the parent table on which a child table depends is not included in the selected objects, the check result is Failed.

Troubleshooting:

- Do not migrate the child tables that cause the check failure. To do this, remove these child tables from the selected objects and run a precheck again.
- Migrate the parent tables rather than the child tables. To do this, add the parent tables to the selected objects and run a precheck again.
- Delete the foreign key dependencies between the parent and child tables in the source database and run a precheck again.

3.5.5. Existence of FEDERATED tables

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the source database contains storage engines that are not supported by incremental data migration. Incremental data migration does not support the FEDERATED and MRG_MyISAM storage engines.

If **the FEDERATED storage engine is used by specific tables in the source database**, the check result is Failed.

If **the MRG_MyISAM storage engine is used by specific tables in the source database**, the check result is Failed.

Solution:

Remove the tables that use the FEDERATED or MRG_MyISAM storage engine from the selected objects. Then, create a separate migration task to perform schema migration and full data migration for these tables.

3.5.6. Permissions

Source database permissions

DTS checks whether the account of the source database has the required permissions to perform data migration. For information about the permissions that are required by each type of database, see the topics about how to configure data migration tasks.

Destination database permissions

DTS checks whether the account of the destination database has the required permissions to perform data migration. For information about the permissions that are required by each type of database, see the topics about how to configure data migration tasks.

3.5.7. Object name conflict

This check item checks for duplicate object names in the destination and source database. If this check item fails, an object in the destination RDS instance has the same name as an object to be migrated. This causes the migration to fail.

When this check item fails, an error message is displayed indicating that an object in the destination database has the same name as an object to be migrated from the source database.

Troubleshooting:

- Use the database and table name mapping feature provided by DTS to migrate the object to be migrated to another object with a different name in the destination database.
- In the destination database, delete or rename the object that has the same name as the object to be migrated.
- Modify the migration task and delete that object to be migrated from the list of objects to be migrated. Do not migrate this object.

3.5.8. Schema existence

This check item checks whether the database to be migrated exists in the destination RDS instance. If no, DTS creates one automatically. However, under the following circumstances, the automatic database creation fails, and this check item prompts a failure:

- The database name contains characters other than lowercase letters, digits, underscores (_), and hyphens (-).

  The cause of the precheck failure is that the name of the **source database** does not comply with the requirements of RDS.

  Troubleshooting: On the database management page of the RDS console, create a database that complies with the requirements of RDS and grant the migration account the read and write permissions on the new database. Use the database name mapping feature provided by DTS to map the source database to the new database. Then, perform the precheck again.

- The character set of the database is not UTF8, GBK, Latin1, or UTF-8MB4.

  The cause of the precheck failure is that the character set of the **source database** does not comply with the requirements of RDS.

  Troubleshooting: On the database management page of the RDS console, create a database that complies with the requirements of RDS and grant the migration account the read and write permissions on the new database. If the new database and the database to be migrated have different names, you can use the database name mapping feature of DTS to map the database to be migrated to the new database. Then re-run the precheck.

- The migration account of the destination database has no read and write permissions on the database to be migrated.

  The cause of the precheck failure is that you are not authorized to operate on the **source database**.

Troubleshooting: On the database management page of the RDS console, click the Account Management tab. Grant the migration account the read and write permissions on the source database. Then, perform the precheck again.

3.5.9. Value of server_id in the source database

This item is checked only when you migrate incremental data between MySQL databases. DTS checks whether the value of **server-id** in the source database is set to an integer greater than 1.

If the check result is Failed, run the `set global server_id='An integer greater than 1'` command in the source database and perform a precheck again.

3.5.10. Source database version

DTS checks whether the version of the source database is supported. The table lists the source database versions that are supported by DTS.

Source database types and versions

Source database type	Supported version
MySQL	5.0, 5.1, 5.5, 5.6, and 5.7. Only 5.1, 5.5, 5.6, and 5.7 are supported for incremental data migration.

If the check result is Failed, you must upgrade or downgrade the source database to a supported version before you perform a precheck again.

3.6. Data type mappings between heterogeneous databases

Heterogeneous databases support different data types. During schema migration, Data Transmission Service (DTS) converts the data types of the source database into those of the destination database. This topic lists the data type mappings for you to evaluate the impact of data migration on your business.

Data migration from a user-created Oracle database to a user-created MySQL database or an ApsaraDB RDS for MySQL instance

Data type in the Oracle database	Data type in the MySQL database	Supported by DTS
varchar2(n [char/byte])	varchar(n)	Yes
nvarchar2[(n)]	national varchar[(n)]	Yes
char[(n [byte/char])]	char[(n)]	Yes
nchar[(n)]	national char[(n)]	Yes
number[(p[,s])]	decimal[(p[,s])]	Yes

Data type in the Oracle database	Data type in the MySQL database	Supported by DTS
float(p)]	double	Yes
long	longtext	Yes
date	datetime	Yes
binary_float	decimal(65,8)	Yes
binary_double	double	Yes
timestamp[(fractional_seconds_precision)]	datetime[(fractional_seconds_precision)]	Yes
timestamp[(fractional_seconds_precision)]with localtimezone	datetime[(fractional_seconds_precision)]	Yes
timestamp[(fractional_seconds_precision)]with localtimezone	datetime[(fractional_seconds_precision)]	Yes
clob	longtext	Yes
nclob	longtext	Yes
blob	longblob	Yes
raw	varbinary(2000)	Yes
long raw	longblob	Yes
bfile	N/A	No
interval year(year_precision) to month	N/A	No
interval day(day_precision)to second[(fractional_seconds_precision)]	N/A	No

Note

- A char column with a length greater than 255 bytes is converted to the varchar(n) type.
- Data types such as bfile, interval year to month, and interval day to second in Oracle databases are not supported in MySQL databases. They cannot be converted to data types supported by the destination database during schema migration.

  The schema migration fails if the table to be migrated contains these three data types. You must make sure that columns with these three data types are excluded from the objects to be migrated.
- The timestamp data type of MySQL databases does not contain the time zone information. However, the timestamp with time zone and timestamp with local time zone data types in Oracle databases provide time zone information. Therefore, DTS converts the values of these data types based on the time zone to UTC time for storage in the destination instance.

Data migration from a user-created Oracle database to a PolarDB-X instance

Oracle data type	PolarDB-X data type	Supported by DTS
varchar2(n [char/byte])	varchar(n)	Yes
nvarchar2[(n)]	national varchar[(n)]	Yes
char[(n [byte/char])]	char[(n)]	Yes
nchar[(n)]	national char[(n)]	Yes
number[(p[,s])]	decimal[(p[,s])]	Yes
float(p)]	double	Yes
long	longtext	Yes
date	datetime	Yes
binary_float	decimal(65,8)	Yes
binary_double	double	Yes
timestamp[(fractional_seconds_precision)]	datetime[(fractional_seconds_precision)]	Yes
timestamp[(fractional_seconds_precision)]with localtimezone	datetime[(fractional_seconds_precision)]	Yes
timestamp[(fractional_seconds_precision)]with localtimezone	datetime[(fractional_seconds_precision)]	Yes
clob	longtext	Yes

Oracle data type	PolarDB-X data type	Supported by DTS
nclob	longtext	Yes
blob	longblob	Yes
raw	varbinary(2000)	Yes
long raw	longblob	Yes
bfile	None	No
interval year(year_precision) to month	None	No
interval day(day_precision)to second[(fractional_seconds_precision)]	None	No

Note

- If a char field in the Oracle database is greater than 255 bytes in length, DTS converts this field to the varchar(n) type in the PolarDB-X instance.
- The timestamp data type of PolarDB-X does not contain the time zone information. However, the timestamp with time zone and timestamp with local time zone data types in Oracle databases provide the time zone information. Therefore, DTS converts the values of these data types into UTC time in the destination PolarDB-X instance.

Data migration from a user-created Oracle database to a PolarDB cluster

Oracle data type	PolarDB data type	Supported by DTS
varchar2(n [char/byte])	varchar2[(n)]	Yes
nvarchar2[(n)]	nvarchar2[(n)]	Yes
char[(n [byte/char])]	char[(n)]	Yes
nchar[(n)]	nchar[(n)]	Yes
number[(p[,s])]	number[(p[,s])]	Yes
float(p)]	double precision	Yes
long	long	Yes
date	date	Yes
binary_float	real	Yes

Oracle data type	PolarDB data type	Supported by DTS
binary_double	double precision	Yes
timestamp[(fractional_seconds_precision)]	timestamp[(fractional_seconds_precision)]	Yes
timestamp[(fractional_seconds_precision)] with time zone	timestamp[(fractional_seconds_precision)]with time zone	Yes
timestamp[(fractional_seconds_precision)] with local time zone	timestamp[(fractional_seconds_precision)]with time zone	Yes
clob	clob	Yes
nclob	nclob	Yes
blob	blob	Yes
raw	raw(size)	Yes
long raw	long raw	Yes
bfile	None	No
interval year(year_precision) to month	interval year to month	No
interval day(day_precision) to second[(fractional_seconds_precision)]	interval day to second[(fractional_seconds_precision)]	No

Note PolarDB does not support the timestamp[(fractional_seconds_precision)]with local time zone data type. DTS converts the data of this type into UTC time and then stores the data in the destination PolarDB cluster by using the timestamp[(fractional_seconds_precision)]with time zone data type.

4.Data synchronization

4.1. Database types, initial synchronization types, and synchronization topologies

You can use Data Transmission Service (DTS) to synchronize data between various data sources. This topic describes the database types, initial synchronization types, and synchronization topologies that are supported by DTS.

Source database	Destination database	Initial synchronization type	Synchronization topology
• User-created MySQL database 5.1, 5.5, 5.6, and 5.7 • RDS MySQL 5.6 and 5.7	User-created MySQL database 5.1, 5.5, 5.6, and 5.7	Initial schema synchronization Initial full data synchronization	One-way synchronization Two-way synchronization
	RDS MySQL 5.6 and 5.7	Initial schema synchronization Initial full data synchronization	One-way synchronization Two-way synchronization
	AnalyticDB for MySQL 2.0 and 3.0	Initial schema synchronization Initial full data synchronization	One-way synchronization
	AnalyticDB for PostgreSQL 4.3 and 6.0	Initial schema synchronization Initial full data synchronization	One-way synchronization
	Datahub	Initial schema synchronization	One-way synchronization

Source database	Destination database	Initial synchronization type	Synchronization topology
	MaxCompute	Initial schema synchronization Initial full data synchronization	One-way synchronization
Cloud Native Distributed Database PolarDB-X (formerly known as DRDS)	Cloud Native Distributed Database PolarDB-X	Initial full data synchronization	One-way synchronization
	Datahub	Initial schema synchronization	One-way synchronization
	AnalyticDB for MySQL 2.0 and 3.0	Initial schema synchronization Initial full data synchronization	One-way synchronization

4.2. Create a data synchronization instance

Before you configure a task to synchronize data, you must create a data synchronization instance. This topic describes how to create a data synchronization instance in the Data Transmission Service (DTS) console.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. In the upper-right corner of the page, click **Create Synchronization Task**.
4. In the Create DTS Instances dialog box, set the required parameters.

Parameter	Description
Source Instance Region	Select the region where the source instance resides.
Source Instance Type	Select the type of the source instance. ○ **MySQL**: a user-created MySQL database or an ApsaraDB RDS for MySQL instance ○ **Drds**: a Cloud Native Distributed Database PolarDB-X instance (formerly known as DRDS)

Parameter	Description
Destination Instance Region	Select the region where the destination instance resides.
Destination Instance Type	Select the type of the destination instance. ○ **MySQL**: a user-created MySQL database or an ApsaraDB RDS for MySQL instance ○ **AnalyticDB**: AnalyticDB for MySQL ○ **MaxCompute** ○ **DataHub** ○ **Drds**: a Cloud Native Distributed Database PolarDB-X instance (formerly known as DRDS) ○ **AnalyticDB for PostgreSQL**: AnalyticDB for PostgreSQL
Synchronization Mode	Two-way synchronization is available only when you select **MySQL** as the type of both the source and destination instances.
Instances to Create	Set the number of data synchronization instances that you want to create at a time. The default value is 1.

Note In the Create DTS Instances dialog box, you can view the total number of instances, the number of existing instances, and the number of instances that can be created.

5. Click **Create**.

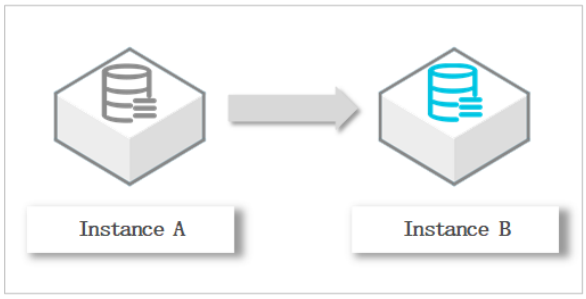
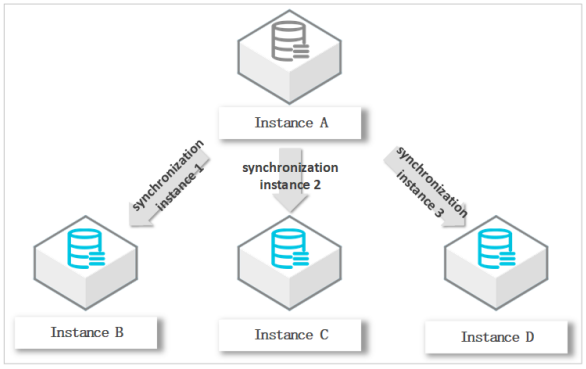
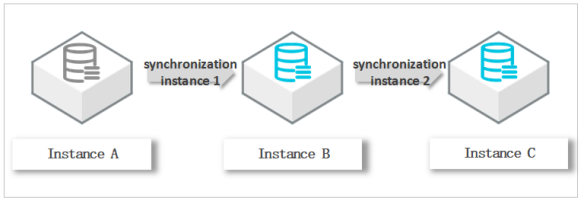
4.3. Synchronization topologies

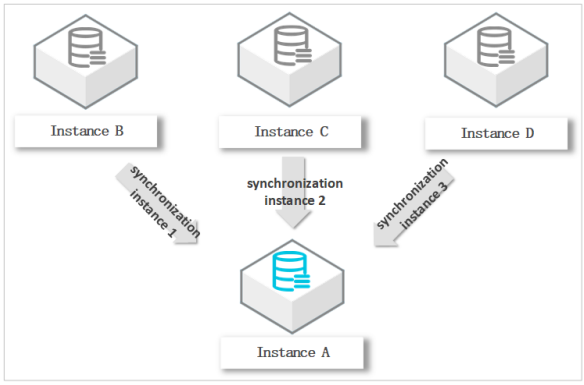
The data synchronization feature supports multiple types of synchronization topologies. You can select a topology for your data synchronization instances based on your business requirements. This topic describes the synchronization topologies that are supported by DTS and how to use these topologies.

One-way synchronization

To ensure data consistency for one-way synchronization, we recommend that you perform only read operations on the objects in the destination instance. Do not modify the objects.

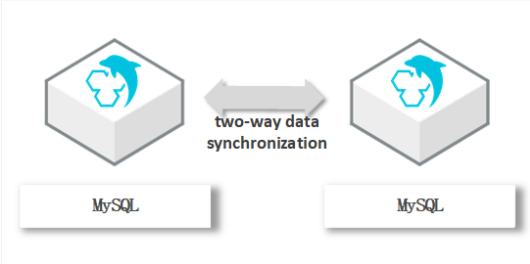
Topology type	Topology	Description

Topology type	Topology	Description
One-way one-to-one synchronization		None
One-way one-to-many synchronization		You must purchase multiple synchronization instances to implement one-way one-to-many synchronization. For example, if you want to synchronize data from Instance A to Instance B, C, and D, you must purchase three synchronization instances.
One-way cascade synchronization		You must purchase multiple synchronization instances to implement one-way cascade synchronization. For example, if you want to synchronize data from Instance A to Instance B and then from Instance B to Instance C, you must purchase two synchronization instances.

Topology type	Topology	Description
One-way many-to-one synchronization		You must purchase multiple synchronization instances to implement one-way many-to-one synchronization. For example, if you want to synchronize data from Instance B, C, and D to Instance A, you must purchase three synchronization instances. **Note** To ensure data consistency, you must select different objects for these synchronization instances.

Two-way synchronization

DTS supports two-way data synchronization only between two MySQL databases. DTS does not support two-way data synchronization between multiple MySQL databases.

Topology type	Topology	Description
Two-way one-to-one synchronization		To ensure data consistency, make sure that records with the same primary key, business primary key, or unique key are updated only on one of the instances.

4.4. Configure data synchronization tasks

4.4.1. Configure data synchronization between ApsaraDB RDS for MySQL instances

This topic describes how to configure one-way data synchronization between ApsaraDB RDS for MySQL instances.

Prerequisites

The source and destination ApsaraDB RDS for MySQL instances are created.

Precautions

- DTS uses the read and write resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours.
- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- The tables to be migrated in the source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- During initial full data synchronization, concurrent INSERT operations cause fragmentation in the tables of the destination instance. After initial full data synchronization, the tablespace of the destination instance is larger than that of the source instance.

Supported synchronization topologies

DTS supports one-way synchronization and two-way synchronization. For more information, see Synchronization topologies.

SQL operations that can be synchronized

Operation type	SQL statement
DML	INSERT, UPDATE, DELETE, and REPLACE
DDL	• ALTER TABLE and ALTER VIEW • CREATE FUNCTION, CREATE INDEX, CREATE PROCEDURE, CREATE TABLE, and CREATE VIEW • DROP INDEX and DROP TABLE • RENAME TABLE • TRUNCATE TABLE

Limits

- Incompatibility with triggers

  If you select a database as the object and the database contains a trigger that updates a table, data inconsistency may occur. To solve this issue, you must delete the trigger in the destination database.

- Limits on RENAME TABLE operations

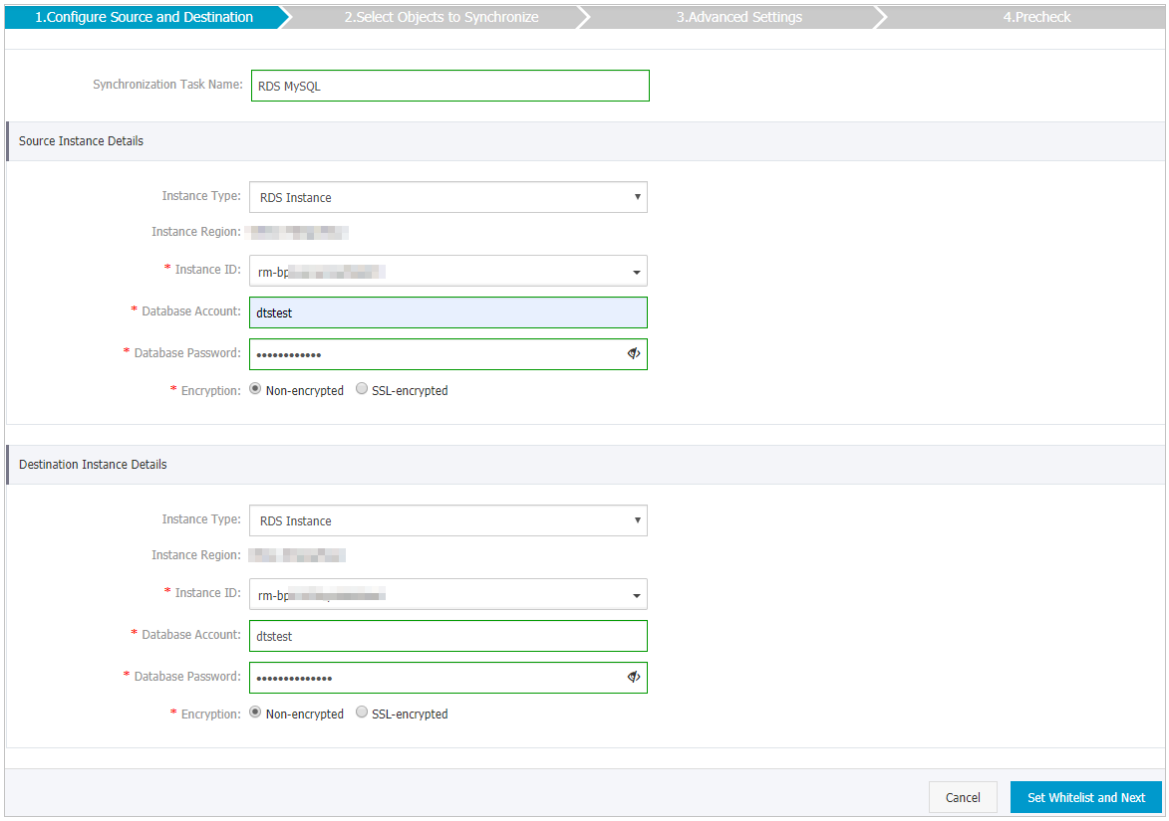
  RENAME TABLE operations may cause data inconsistency between the source and destination databases. For example, if you select a table as the object and rename the table during data synchronization, the data of this table is not synchronized to the destination database. To avoid this situation, you can select the database to which this table belongs as the object when you configure the data synchronization task.

Procedure

1. Create a data synchronization instance.

   > **Note** When you create the data synchronization instance, set both Source Instance Type and Destination Instance Type to **MySQL**, and set Synchronization Mode to **One-Way Synchronization**.

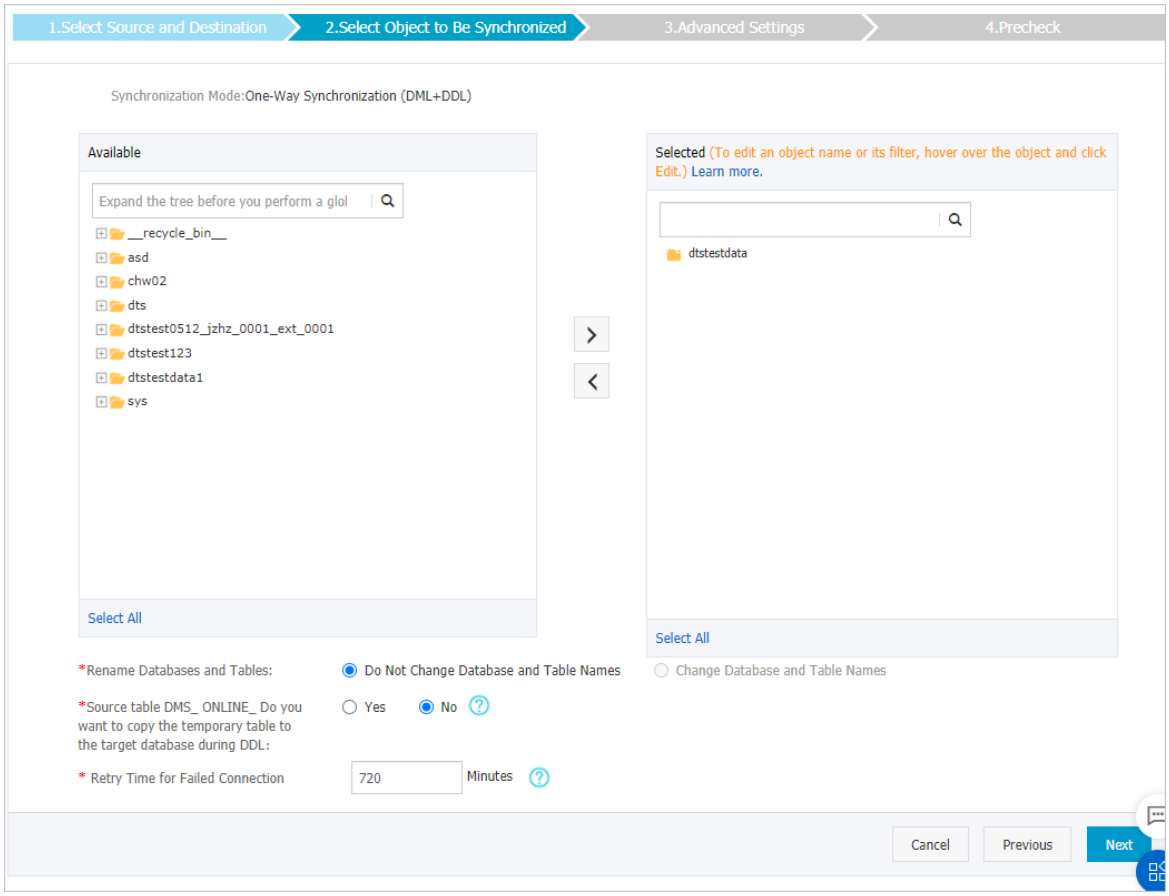
2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.



Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the source RDS instance.

Source Section Instance	Parameter	Description
Details	Database Account	Enter the database account of the source RDS instance. **Note** If the database engine of the source RDS instance is **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the source database account.
	Encryption	Select an encryption method. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.
Destination Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the source instance. The region is the same as the destination region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the destination RDS instance.
	Database Account	Enter the database account of the destination RDS instance. **Note** If the database engine of the destination RDS instance is **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the destination database account.
	Encryption	Select an encryption method. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.

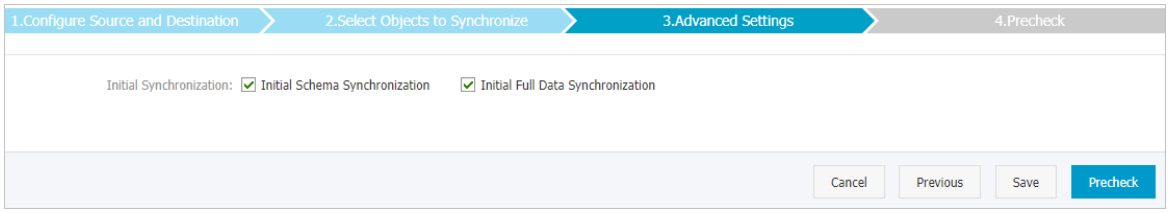
4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the processing mode of conflicting tables, and the objects that you want to synchronize.



Parameter	Description
Processing Mode In Existed Target Table	○ **Pre-check and Intercept**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain identical table names, the precheck is passed. Otherwise, an error is returned during precheck and the data synchronization task cannot be started. ○ **Ignore**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ DTS does not synchronize the data records that have the same primary keys as the data records in the destination database during initial data synchronization. This occurs if the source and destination databases have the same schema. However, DTS synchronizes these data records during incremental data synchronization. ■ If the source and destination databases have different schemas, initial data synchronization may fail. In this case, only specific columns are synchronized or the data synchronization task fails.

Parameter	Description
Select Objects	Select objects (tables or a database) from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** • If you select a database as the object to be synchronized, all schema changes in the database are synchronized to the destination database. • After an object is synchronized to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.

6. In the lower-right corner of the page, click **Next**.
7. Configure initial synchronization.



Note Initial synchronization includes initial schema synchronization and initial full data synchronization. If you select both **Initial Schema Synchronization** and **Initial Full Data Synchronization**, DTS synchronizes the schemas and historical data of the required objects before DTS synchronizes incremental data.

8. In the lower-right corner of the page, click **Precheck**.

Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

Troubleshoot the issues based on the causes and run a precheck again.

9. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.2. Synchronize data from an ApsaraDB RDS for MySQL instance to a MaxCompute project

MaxCompute (formerly known as ODPS) is a fast and fully managed computing platform for large-scale data warehousing. MaxCompute can process exabytes of data. This topic describes how to synchronize data from an ApsaraDB RDS for MySQL instance to a MaxCompute project by using Data Transmission Service (DTS).

Precautions

- DTS uses the read and write resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours.
- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- You can select only tables as the objects to be synchronized.
- MaxCompute does not support the PRIMARY KEY constraint. If network errors occur, DTS may synchronize duplicate data records to MaxCompute.

SQL operations that can be synchronized

- Data definition language (DDL) operation: ADD COLUMN
- Data manipulation language (DML) operations: INSERT, UPDATE, and DELETE

Synchronization process

1. Initial schema synchronization

   DTS synchronizes the schemas of the required objects from the source database to MaxCompute. During initial schema synchronization, DTS adds the _base suffix to the end of the source table name. For example, if the name of the source table is customer, the name of the table in MaxCompute is customer_base.

2. Initial full data synchronization

   DTS synchronizes the historical data of the table from the source database to the destination table in MaxCompute. For example, the customer table in the source database is synchronized to the customer_base table in MaxCompute. The data is the basis for subsequent incremental synchronization.

   > **Note** The destination table that is suffixed with _base is known as a full baseline table.

3. Incremental data synchronization

   DTS creates an incremental data table in MaxCompute. The name of the incremental data table is suffixed with _log, for example, customer_log. Then, DTS synchronizes the incremental data that was generated in the source database to the incremental data table.

   > **Note** For more information, see Schema of an incremental data table.

Procedure

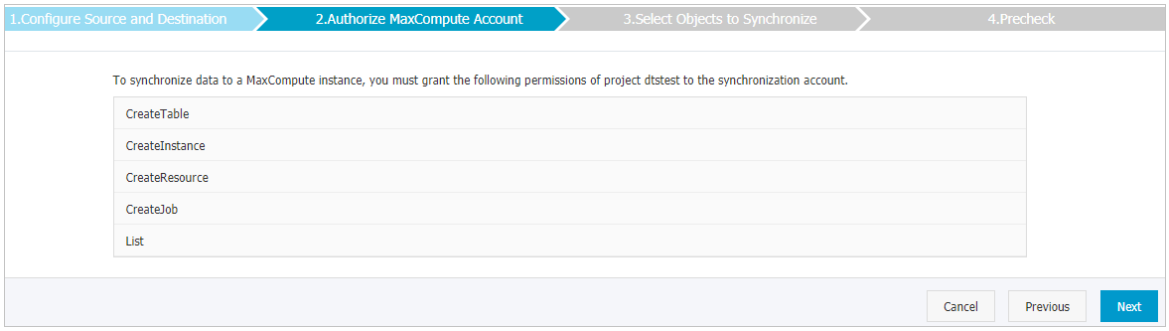
1. Create a data synchronization instance.

> **Note** When you create the data synchronization instance, set Source Instance Type to **MySQL**, set Destination Instance Type to **MaxCompute**, and set Synchronization Mode to **One-Way Synchronization**.

2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.

Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the source RDS instance.
	Database Account	Enter the database account of the source RDS instance. **Note** If the database engine of the source RDS instance is **MySQL 5.5** or **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the source database account.
	Encryption	Select **Non-encrypted** or **SSL-encrypted**. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.
Destination Instance Details	Instance Type	This parameter is set to **MaxCompute** and cannot be changed.
	Instance Region	The region of the destination instance. The region is the same as the destination region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Project	The name of the MaxCompute project.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. In the lower-right corner of the page, click **Next**. In this step, the permissions on the MaxCompute project are granted to the synchronization account.



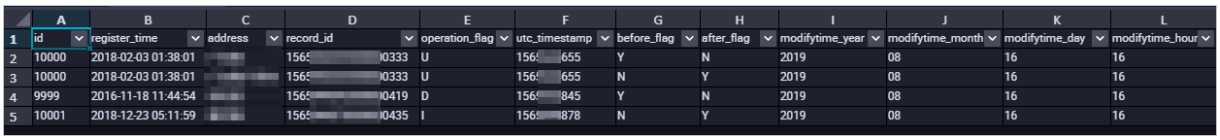
6. Configure the synchronization policy and objects.
7. In the lower-right corner of the page, click **Precheck**.

> **Note** You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

8. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

Schema of an incremental data table

DTS synchronizes incremental data that is generated in the source MySQL database to the incremental data table in MaxCompute. The incremental data table stores incremental data and specific metadata. The following figure shows the schema of an incremental data table.



	A	B	C	D	E	F	G	H	I	J	K	L
1	id	register_time	address	record_id	operation_flag	utc_timestamp	before_flag	after_flag	modifytime_year	modifytime_month	modifytime_day	modifytime_hour
2	10000	2018-02-03 01:38:01		1565 0333	U	156 655	Y	N	2019	08	16	16
3	10000	2018-02-03 01:38:01		1565 0333	U	156 655	N	Y	2019	08	16	16
4	9999	2016-11-18 11:44:54		1565 0419	D	156 845	Y	N	2019	08	16	16
5	10001	2018-12-23 05:11:59		1565 0435	I	156 878	N	Y	2019	08	16	16

> **Note** In the example, the `modifytime_year`, `modifytime_month`, `modifytime_day`, `modifytime_hour`, and `modifytime_minute` fields form the partition key.

The following table describes the schema of an incremental data table.

Field	Description
record_id	The ID of the incremental log entry. **Note** • The ID auto-increments for each new log entry. • If an UPDATE operation is performed, DTS generates two incremental log entries for the operation. The two incremental log entries have the same record ID.

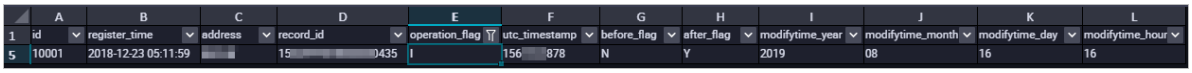
Field	Description
operation_flag	The operation type. Valid values: • I: an INSERT operation. • D: a DELETE operation. • U: an UPDATE operation.
utc_timestamp	The operation timestamp. It is also the timestamp of the binary log file. The timestamp is in the UTC format.
before_flag	Indicates whether the column values are pre-update values. Valid values: Y and N.
after_flag	Indicates whether the column values are post-update values. Valid values: Y and N.

Additional information about the before_flag and after_flag fields

For different operation types, the **before_flag** and **after_flag** fields of an incremental log entry are defined as follows:

- INSERT

  For an INSERT operation, the column values are the newly inserted record values (post-update values). The value of the before_flag field is N and the value of the after_flag field is Y.

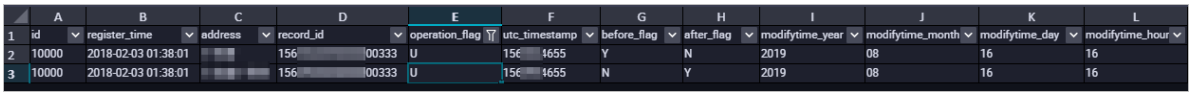


	A	B	C	D	E	F	G	H	I	J	K	L
1	id	register_time	address	record_id	operation_flag	utc_timestamp	before_flag	after_flag	modifytime_year	modifytime_month	modifytime_day	modifytime_hour
5	10001	2018-12-23 05:11:59	[redacted]	15[redacted]0435	I	156[redacted]878	N	Y	2019	08	16	16

- UPDATE

  DTS generates two incremental log entries for an UPDATE operation. The two incremental log entries have the same values for the record_id, operation_flag, and dts_utc_timestamp fields.

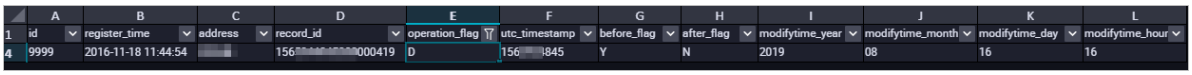
  The second log entry records the pre-update values, so the value of the before_flag field is Y and the value of the after_flag field is N. The second log entry records the post-update values, so the value of the before_flag field is N and the value of the after_flag field is Y.



	A	B	C	D	E	F	G	H	I	J	K	L
1	id	register_time	address	record_id	operation_flag	utc_timestamp	before_flag	after_flag	modifytime_year	modifytime_month	modifytime_day	modifytime_hour
2	10000	2018-02-03 01:38:01	[redacted]	156[redacted]00333	U	156[redacted]1655	Y	N	2019	08	16	16
3	10000	2018-02-03 01:38:01	[redacted]	156[redacted]00333	U	156[redacted]1655	N	Y	2019	08	16	16

- DELETE

  For a DELETE operation, the column values are the deleted record values (pre-update values). The value of the before_flag field is Y and the value of the after_flag field is N.



	A	B	C	D	E	F	G	H	I	J	K	L
1	id	register_time	address	record_id	operation_flag	utc_timestamp	before_flag	after_flag	modifytime_year	modifytime_month	modifytime_day	modifytime_hour
4	9999	2016-11-18 11:44:54	[redacted]	156[redacted]000419	D	156[redacted]1845	Y	N	2019	08	16	16

Merge a full baseline table and incremental data table

After a data synchronization task is started, DTS creates a full baseline table and an incremental data table in MaxCompute. You can use SQL statements to merge the two tables. This allows you to obtain the full data at a specific time point.

This section describes how to merge data for the customer table. The following figure shows the schema of the customer table.

	Field	Type	Null	Key	Default	Extra
1	id	int(11)	NO	PRI	null	
2	register_time	timestamp	YES		null	
3	address	varchar(32)	YES		null	

1. Create a table in MaxCompute based on the schema of the source table. The table is used to store the merged data.

   For example, you can obtain full data of the customer table at the `1565944878` time point. Run the following SQL statements to create the required table:

```
CREATE TABLE `customer_1565944878` (
    `id` bigint NULL,
    `register_time` datetime NULL,
    `address` string);
```

   Note For more information about the data types that are supported by MaxCompute, see MaxCompute User Guide.

2. Run the following SQL statements in MaxCompute to merge the full baseline table and incremental data table and obtain full data at a specific time point:

```
set odps.sql.allow.fullscan=true;
insert overwrite table <result_storage_table>
select <col1>,
       <col2>,
       <colN>
  from(
select row_number() over(partition by t.<primary_key_column>
 order by record_id desc, after_flag desc) as row_number, record_id, operation_flag, af
ter_flag, <col1>, <col2>, <colN>
  from(
select incr.record_id, incr.operation_flag, incr.after_flag, incr.<col1>, incr.<col2>,i
ncr.<colN>
  from <table_log> incr
 where utc_timestamp< <timestamp>
 union all
select 0 as record_id, 'I' as operation_flag, 'Y' as after_flag, base.<col1>, base.<col
2>,base.<colN>
  from <table_base> base) t) gt
where record_num=1
  and after_flag='Y'
```

> **Note**
>
> - <result_storage_table>: the name of the table that stores the merged data.
> - <col1>/<col2>/<colN>: the names of the columns in the table to be merged.
> - <primary_key_column>: the name of the primary key column in the table to be merged.
> - <table_log>: the name of the incremental data table.
> - <table_base>: the name of the full baseline table.
> - <timestamp>: the timestamp that is generated when full data is obtained.

Run the following SQL statements to obtain full data of the customer table at the `1565944878` time point:

```
set odps.sql.allow.fullscan=true;
insert overwrite table customer_1565944878
select id,
       register_time,
       address
  from(
select row_number() over(partition by t.id
 order by record_id desc, after_flag desc) as row_number, record_id, operation_flag, af
ter_flag, id, register_time, address
  from(
select incr.record_id, incr.operation_flag, incr.after_flag, incr.id, incr.register_tim
e, incr.address
  from customer_log incr
 where utc_timestamp< 1565944878
 union all
select 0 as record_id, 'I' as operation_flag, 'Y' as after_flag, base.id, base.register
_time, base.address
  from customer_base base) t) gt
 where gt.row_number= 1
   and gt.after_flag= 'Y';
```

3. Query the merged data from the customer_1565944878 table.

	A	B	C
1	id	register_time	address
2	1	2017-12-09 14:00:12	
3	2	2017-11-16 21:17:39	
4	3	2019-01-29 07:56:20	

4.4.3. Synchronize data from an ApsaraDB RDS for MySQL instance to an AnalyticDB for MySQL cluster

AnalyticDB for MySQL is a real-time online analytical processing (RT-OLAP) service that is developed by Alibaba Cloud for online data analysis with high concurrency. This topic describes how to synchronize data from an ApsaraDB RDS for MySQL instance to an AnalyticDB for MySQL cluster by using Data Transmission Service (DTS). After you synchronize data, you can use AnalyticDB for MySQL to build internal business intelligence (BI) systems, interactive query systems, and real-time report systems.

Prerequisites

- The tables that you want to synchronize from the ApsaraDB RDS for MySQL instance contain primary keys.
- The destination AnalyticDB for MySQL cluster has sufficient storage space.

Precautions

- DTS uses the read and write resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours.
- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- If the disk space usage of nodes in an AnalyticDB for MySQL cluster reaches 80%, the cluster is locked. We recommend that you estimate the required disk space based on the objects to be synchronized. You must ensure that the destination cluster has sufficient storage space.

SQL operations that can be synchronized

- Data definition language (DDL) operations: CREATE TABLE, DROP TABLE, RENAME TABLE, TRUNCATE TABLE, ADD COLUMN, and DROP COLUMN
- Data manipulation language (DML) operations: INSERT, UPDATE, and DELETE

> **Note** We recommend that you do not change the data type of fields in the source table during data synchronization. Otherwise, DTS generates an error message and stops the data synchronization task.

Data type mappings

The data types of ApsaraDB RDS for MySQL and AnalyticDB for MySQL do not have one-to-one correspondence. During initial schema synchronization, DTS converts the data types of the source database into those of the destination database. The following table lists the data types that DTS can convert.

Data type of ApsaraDB RDS for MySQL	Data type of AnalyticDB for MySQL
BIGINT UNSIGNED	DECIMAL(20,0)
BINARY	VARBINARY
BIT	VARCHAR
BLOB	VARBINARY

Data type of ApsaraDB RDS for MySQL	Data type of AnalyticDB for MySQL
CHAR	VARCHAR
DATE	DATE
DATETIME	DATETIME
DECIMAL	DECIMAL
DOUBLE	DOUBLE
ENUM	VARCHAR
FLOAT	FLOAT
GEOMETRY	VARBINARY
GEOMETRYCOLLECTION	VARBINARY
INT UNSIGNED	BIGINT
INTEGER	INT
JSON	JSON
LINESTRING	VARBINARY
LONGBLOB	VARBINARY
LONGTEXT	VARCHAR
MEDIUMBLOB	VARBINARY
MEDIUMINT	INT
MEDIUMINT UNSIGNED	INT
MEDIUMTEXT	VARCHAR
MULTILINESTRING	VARBINARY
MULTIPOINT	VARBINARY
MULTIPOLYGON	VARBINARY
NUMBERIC	DECIMAL
POINT	VARBINARY
POLYGON	VARBINARY
SET	VARCHAR

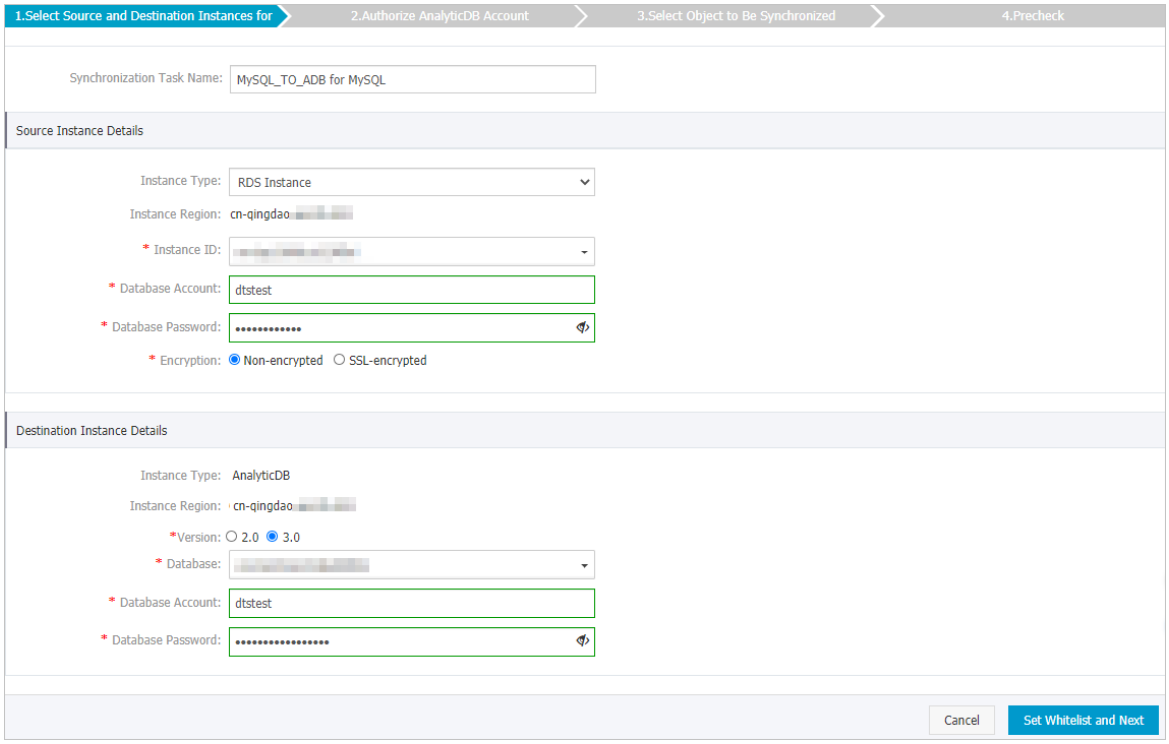
Data type of ApsaraDB RDS for MySQL	Data type of AnalyticDB for MySQL
SMALLINT UNSIGNED	INT
TEXT	VARCHAR
TIME	TIME
TIMESTAMP	TIMESTAMP
TINYBLOB	VARBINARY
TINYINT UNSIGNED	SMALLINT
TINYTEXT	VARCHAR
VARBINARY	VARBINARY
VARCHAR	VARCHAR
YEAR	INT

Procedure

1. Create a data synchronization instance.

   > **Note** When you create the data synchronization instance, set Source Instance Type to **MySQL**, set Destination Instance Type to **AnalyticDB**, and set Synchronization Mode to **One-Way Synchronization**.

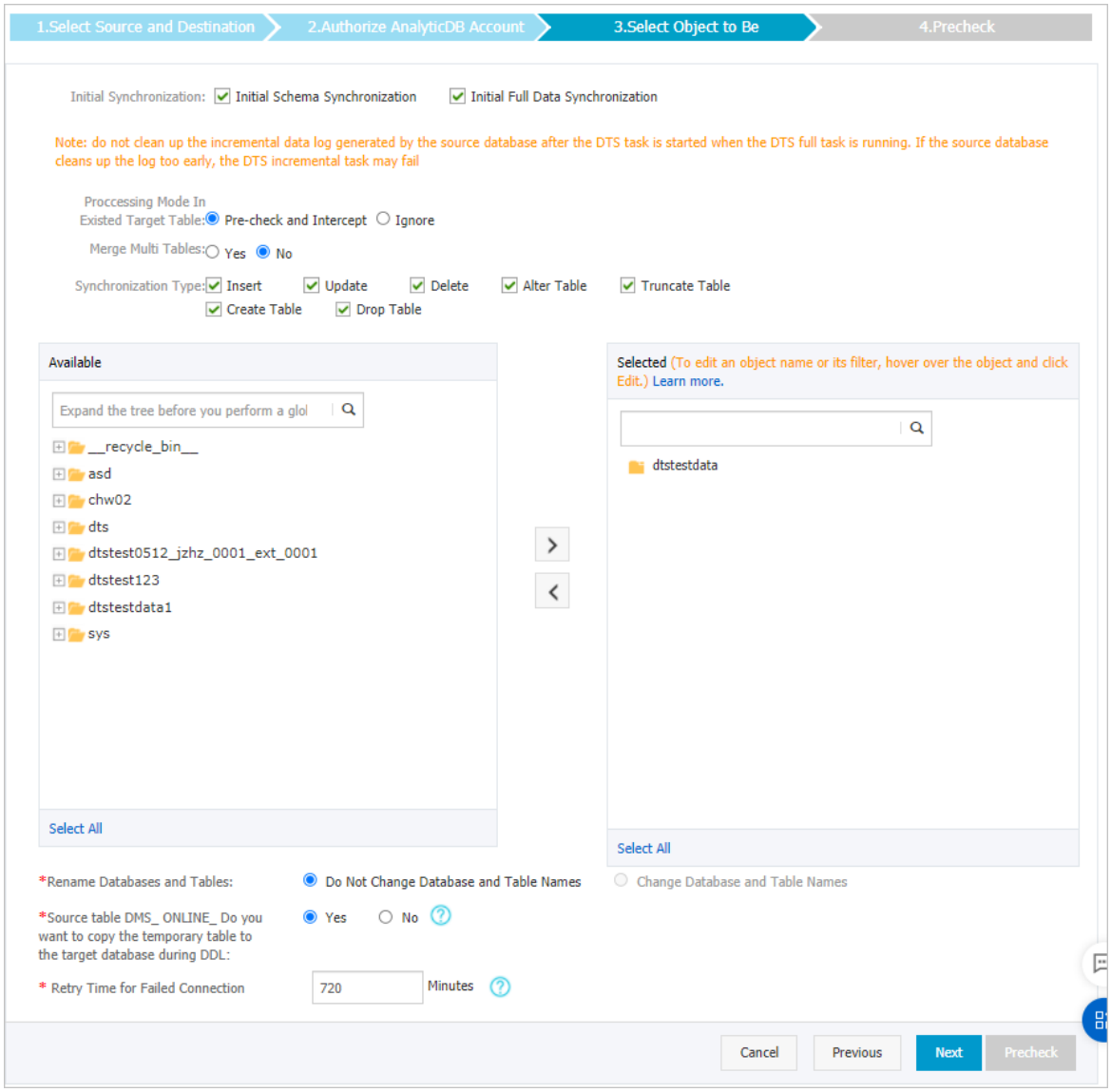
2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.



Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the source RDS instance.
	Database Account	Enter the database account of the source RDS instance. The account must have the REPLICATION CLIENT permission, the REPLICATION SLAVE permission, the SHOW VIEW permission, and the permission to perform SELECT operations on the required objects. **Note** If the database engine of the source RDS instance is **MySQL 5.5** or **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the source database account.
	Encryption	Select an encryption method. If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.

Section	Parameter	Description
Destination Instance Details	Instance Type	This parameter is set to **AnalyticDB** and cannot be changed.
	Instance Region	The region of the destination instance. The region is the same as the destination region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Version	Select **3.0**.
	Database	Select the ID of the destination AnalyticDB for MySQL cluster.
	Database Account	Enter the database account of the destination AnalyticDB for MySQL cluster. The account must have the read and write permissions on the destination database.
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Configure the synchronization policy and objects.

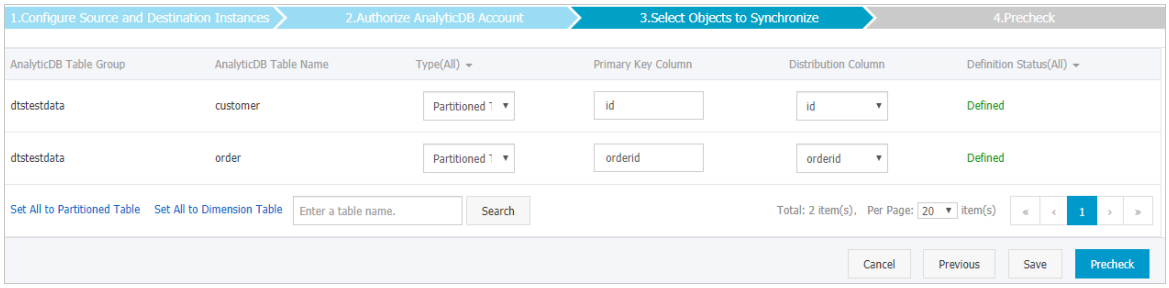


Parameter	Description
Initial Synchronization	You must select both **Initial Schema Synchronization** and **Initial Full Data Synchronization** in most cases. After the precheck, DTS synchronizes the schemas and data of the required objects from the source instance to the destination cluster. The schemas and data are the basis for subsequent incremental synchronization.

Parameter	Description
Processing Mode In Existed Target Table	◦ **Pre-check and Intercept**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain identical table names, the precheck is passed. Otherwise, an error is returned during precheck and the data synchronization task cannot be started. ◦ **Ignore**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ DTS does not synchronize the data records that have the same primary keys as the data records in the destination database during initial data synchronization. This occurs if the source and destination databases have the same schema. However, DTS synchronizes these data records during incremental data synchronization. ■ If the source and destination databases have different schemas, initial data synchronization may fail. In this case, only specific columns are synchronized or the data synchronization task fails.
Merge Multi Tables	◦ If you select **Yes**, DTS adds the `__dts_data_source` column to each table to record data sources. In this case, DDL operations cannot be synchronized. ◦ **No** is selected by default. In this case, DDL operations can be synchronized. **Note** You can merge the data source columns based on tasks rather than tables. To merge only the data source columns of specific tables, you can create two data synchronization tasks.
Synchronization Type	Select the types of operations that you want to synchronize based on your business requirements. All operation types are selected by default. **Note** Only the INSERT, UPDATE, DELETE, and ADD COLUMN operations can be synchronized.

Parameter	Description
Select Objects	Select objects (tables or a database) from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** • If you select a database as the object to be synchronized, all schema changes in the database are synchronized to the destination database. • After an object is synchronized to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.

6. In the lower-right corner of the page, click **Next**.
7. Specify a type for the tables that you want to synchronize to the destination database.



Note After you select **Initial Schema Synchronization**, you must specify the **type**, **primary key column**, and **partition key column** for the tables that you want to synchronize to AnalyticDB for MySQL.

8. In the lower-right corner of the page, click **Precheck**.

Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

Troubleshoot the issues based on the causes and run a precheck again.

9. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.4. Synchronize data from an ApsaraDB RDS for MySQL instance to an AnalyticDB for PostgreSQL instance

This topic describes how to synchronize data from an ApsaraDB RDS for MySQL instance to an AnalyticDB for PostgreSQL instance by using Data Transmission Service (DTS). The data synchronization feature allows you to transfer and analyze data with ease.

Prerequisites

- The tables that you want to synchronize contain primary keys.
- An AnalyticDB for PostgreSQL instance is created.

Precautions

- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- Only one-way synchronization is supported.

Limits

- You can select only tables as the objects to be synchronized.
- DTS does not synchronize the schemas of the required objects from the source database to the destination database.
- DTS does not synchronize the following types of data: JSON, GEOMETRY, CURVE, SURFACE, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION, and BYTEA.

SQL operations that can be synchronized

- Data manipulation language (DML) operations: INSERT, UPDATE, and DELETE
- Data definition language (DDL) operations: ALTER TABLE, ADD COLUMN, DROP COLUMN, and RENAME COLUMN

> **Note** The CREATE TABLE and DROP TABLE operations are not supported. To synchronize data from a new table, you must add the table to the selected objects. For more information, see Add objects to be synchronized.

Term mappings

Term in ApsaraDB RDS for MySQL	Term in AnalyticDB for PostgreSQL
Database	Schema
Table	Table

Create a data structure in the destination instance

Create a database, schema, and table in the destination AnalyticDB for PostgreSQL instance based on the data structure of the source RDS instance.

Configure a data synchronization task

1. Create a data synchronization instance.

   > **Note** When you create the data synchronization instance, set Source Instance Type to **MySQL**, set Destination Instance Type to **AnalyticDB for PostgreSQL**, and set Synchronization Mode to **One-Way Synchronization**.

2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.

Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	Select **RDS Instance**.
	Instance Region	The region where the source RDS instance resides.
	Instance ID	Select the ID of the source RDS instance.
	Encryption	Select **Non-encrypted** or **SSL-encrypted**. **Note** If you select **SSL-encrypted**, you must enable SSL encryption for the RDS instance before you configure the data synchronization task.
Destination Instance Details	Instance Type	This parameter is set to **AnalyticDB for PostgreSQL** and cannot be changed.
	Instance Region	The region where the destination instance resides.
	Instance ID	Select the ID of the destination AnalyticDB for PostgreSQL instance.
	Database Name	Enter the name of the destination database.

Section	Parameter	Description
	Database Account	Enter the database account of the destination AnalyticDB for PostgreSQL instance. **Note** The database account must have the SELECT, INSERT, UPDATE, DELETE, COPY, TRUNCATE, and ALTER TABLE permissions.
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Wait until the synchronization account is created. Then, click **Next**.
6. Configure the synchronization policy and objects.

Section	Parameter	Description
Synchronization policy	Initial Synchronization	Select **Initial Full Data Synchronization**. **Note** DTS synchronizes the historical data of the required objects from the source instance to the destination instance. The historical data is the basis for subsequent incremental synchronization.
	Processing Mode In Existed Target Table	○ **Pre-Check and intercept** (Selected by default) Checks the **Schema Name Conflict** item and generates an error message if the destination table contains data. ○ **Clear Target Table** Skips the **Schema Name Conflict** item during the precheck. Clears the data in the destination table before initial full data synchronization. If you want to synchronize your business data after testing the data synchronization task, you can select this mode. ○ **Ignore** Skips the **Schema Name Conflict** item during the precheck. Adds data to the existing data during initial full data synchronization. If you want to synchronize data from multiple tables to one table, you can select this mode.

Section	Parameter	Description
	Synchronization Type	○ **Insert** ○ **Update** ○ **Delete** ○ **AlterTable** **Note** Select the types of operations that you want to synchronize based on your business requirements.
Select Objects	N/A	You can select only tables as the objects to be synchronized. You can use the object name mapping feature to change the names of the columns that are synchronized to the destination database. For more information, see Specify the name of an object in the destination instance. **Note** The CREATE TABLE operation is not supported. To synchronize data from a new table, you must add the table to the selected objects. For more information, see Add objects to be synchronized.

7. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

8. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.5. Synchronize data from a self-managed PostgreSQL database to an AnalyticDB for PostgreSQL instance

This topic describes how to synchronize data from a self-managed PostgreSQL database to an AnalyticDB for PostgreSQL instance by using Data Transmission Service (DTS). The data synchronization feature provided by DTS allows you to transfer and analyze data with ease.

Prerequisites

- The source self-managed PostgreSQL database and the destination AnalyticDB for PostgreSQL instance are created.

- The available storage space of the AnalyticDB for PostgreSQL instance is larger than the total size of the data in the self-managed PostgreSQL database.

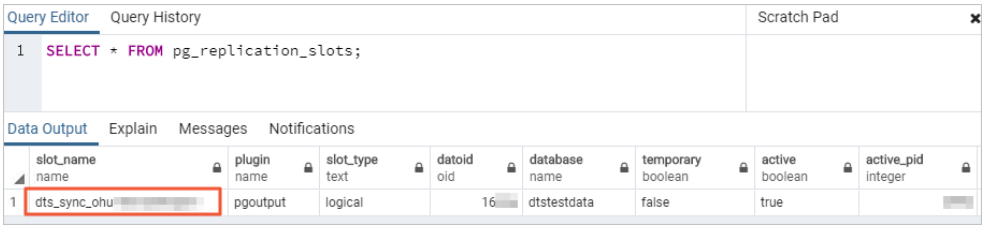
Precautions

Category	Description

Limits on the source database	• The tables to be synchronized must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	• The following requirements for data logs must be met:
	  ◦ The value of the wal_level parameter is set to *logical*.
	  ◦ Data logs are retained for at least seven days during full data synchronization. You can wait until full data synchronization is complete, and then clear the data logs generated in the source database after the DTS task is run.
	Note DTS stores only 50 GB of data logs or the data logs for only the last 24 hours. If the limit is exceeded, DTS automatically clears the cached logs.
	Warning If you clear the data logs of the source database during initial full data synchronization, the data synchronization task may fail. For example, initial full data synchronization takes more than 24 hours due to the large data amount in the source database and abnormal writing in the destination database. In this case, if the data logs of the source database are cleared during initial full data synchronization, DTS cannot obtain the data logs generated 24 hours ago. As a result, the data synchronization task may fail.

	• DTS does not synchronize the schemas of the required objects from the source database to the destination database. You must create a database, schema, or table in the destination AnalyticDB for PostgreSQL instance based on the data structure of the source self-managed PostgreSQL database.
	• Requirements for the objects to synchronize:
	  ◦ Only tables can be selected as objects to be synchronized.
	  ◦ DTS does not synchronize the following data types: BIT, VARBIT, GEOMETRY, ARRAY, UUID, TSQUERY, TSVECTOR, and TXID_SNAPSHOT.
	• A single data synchronization task can synchronize data from only one database. To synchronize data from multiple databases, you must create a data synchronization task for each database.
	• During data synchronization, if you select a schema as the object to synchronize, take note of the following limits: If you create a table in the schema or run the RENAME command to rename the table, you must run the `ALTER TABLE schema.table REPLICA IDENTITY FULL;` command before you write data to the table. This ensures data consistency.
	Note Replace the `schema` and `table` in the preceding sample statement with the actual schema name and table name.
	• To ensure that the latency of data synchronization is accurate, DTS adds a heartbeat table to the source database. The name of the heartbeat table is *dts_postgres_heartbeat*.
	• During data synchronization, DTS creates a replication slot for the source database. The

Category	Description
	replication slot is prefixed with `dts_sync_`. DTS automatically clears historical replication slots every 90 minutes to reduce storage usage.
Other limits	**Note** If the data synchronization task is released or fails, DTS automatically clears the replication slot. If the source is an ApsaraDB RDS for PostgreSQL instance and a primary/secondary switchover is performed on the instance, you must log on to the secondary database to clear the replication slot.



- Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours. During full data synchronization, DTS uses read and write resources of the source and destination databases. This may increase the loads on the database servers.
- During full data synchronization, concurrent INSERT operations cause fragmentation in the tables of the destination database. After full data synchronization is complete, the tablespace of the destination database is larger than that of the source database.
- We recommend that you do not use gh-ost or pt-online-schema-change to perform DDL operations on source tables during data synchronization. Otherwise, data synchronization may fail.
- If you use only DTS to write data to the destination database, you can use Data Management (DMS) to perform online DDL operations on source tables during data synchronization.

  Warning If you use tools other than DTS to write data to the destination database, we recommend that you do not use DMS to perform online DDL operations. Otherwise, data loss may occur in the destination database.

SQL operations that can be synchronized

DML operations: INSERT, UPDATE, and DELETE

Before you begin

DTS does not synchronize the schemas of the required objects from the source database to the destination database. You must create a database, schema, or table in the destination AnalyticDB for PostgreSQL instance based on the data structure of the source self-managed PostgreSQL database.

Procedure

1. Create a data synchronization instance.

   Note When you create the data synchronization instance, select **PostgreSQL** as the source instance type, **AnalyticDB for PostgreSQL** as the destination instance type, and **One-way Synchronization** as the synchronization topology.

2. Find the data synchronization instance, and click **Configure Task** in the **Actions** column.
3. Configure the source and destination databases.

Section	Parameter	Description
N/A	Synchronization Task Name	The task name that DTS automatically generates. We recommend that you specify an informative name to identify the task. You do not need to use a unique task name.
Source Instance Details	Instance Type	The deployment location of the source database. In this example, **User-Created Database Connected Over Express Connect, VPN Gateway, or Smart Access Gateway** is selected.
	Instance Region	The source region that you selected on the buy page. You cannot change the value of this parameter.
	Peer VPC	The ID of the virtual private cloud (VPC) that is connected to the self-managed PostgreSQL database.
	Database Type	This parameter is fixed to **PostgreSQL**.
	IP address	The server IP address of the self-managed PostgreSQL database.
	Port Number	The service port number of the self-managed PostgreSQL database.
	Database Name	The name of the self-managed PostgreSQL database.
	Database Account	The username of the account used to log on to the self-managed PostgreSQL database. The account must have the superuser permissions.
	Database Password	The password of the database account.
Destination Instance Details	Instance Type	The value of this parameter is fixed to **AnalyticDB for PostgreSQL**.
	Instance Region	The destination region that you selected on the buy page. The value of this parameter cannot be changed.
	Instance ID	The ID of the destination AnalyticDB for PostgreSQL instance.
	Database Name	The name of the destination database.
	Database Account	The username of the initial database account of the AnalyticDB for PostgreSQL instance or the database account that has the RDS_SUPERUSER permission.
	Database Password	The password of the database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.

5. If a whitelist is configured for the self-managed PostgreSQL database, you must manually add the CIDR blocks of DTS servers to the whitelist of the database. To obtain the CIDR blocks of DTS servers, click **Copy** in the dialog box that appears. After you add the CIDR blocks of DTS servers to the whitelist of the database, click **Next**.

   > **Note** If you do not need to configure a whitelist for the user-created PostgreSQL database, ignore the preceding settings and click **Next**.

6. Configure the synchronization policy and the objects to be synchronized.

Category	Setting	Description
Synchronization policy	Specify initial synchronization	By default, **Initial Full Data Synchronization** is selected. After the precheck, DTS synchronizes historical data of the required objects from the source database to the destination instance. The data is the basis for subsequent incremental synchronization.
	Select the processing mode of conflicting tables	○ **Pre-check and intercept** (Selected by default) Checks the **Schema Name Conflict** item and generates an error message if the destination table contains data. ○ **Ignore** Skips the check for empty destination tables during the precheck. Adds data to the existing data during initial full data synchronization. If you want to synchronize data from multiple tables to one table, you can select this mode.
	Select types of operations to be synchronized	○ **Insert** ○ **Update** ○ **Delete** > **Note** Select the types of operations that you want to synchronize based on your business requirements.
Objects to be synchronized	Select objects to be synchronized	You can select only tables as objects to be synchronized. You can use the object name mapping feature to rename the columns that are synchronized to the destination database. For more information, see Specify the name of an object in the destination instance.
	Specify whether to rename objects	You can use the object name mapping feature to rename the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.

Category	Setting	Description
	Filter data to be synchronized	You can specify WHERE conditions to filter data. For more information, see Use SQL conditions to filter data. **Notice** Filter conditions do not take effect on incremental data.
Others	Specify the retry time range for a failed connection to the source or destination instance.	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 720 minutes (12 hours). You can specify the retry time range based on your business requirements. If DTS reconnects to the source and destination databases within the specified time range, DTS resumes the data synchronization task. Otherwise, the data synchronization task fails.

7. In the lower-right corner of the page, click **Precheck**.

   Note
   - Before you can start the data synchronization task, DTS performs a precheck. You can start the data synchronization task only after the task passes the precheck.
   - If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
     - You can troubleshoot the issues based on the causes and run a precheck again.
     - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

8. Close the **Precheck** dialog box after the following message is displayed: **Successful**. Then, DTS performs initial synchronization.

4.4.6. Synchronize data between Cloud Native Distributed Database PolarDB-X instances

Cloud Native Distributed Database PolarDB-X is formerly known as Distributed Relational Database Service (DRDS). It is compatible with the MySQL protocol and syntax, and supports automatic sharding, online smooth scaling, auto scaling, and transparent read/write splitting. This topic describes how to synchronize data between Cloud Native Distributed Database PolarDB-X instances by using Data Transmission Service (DTS).

Prerequisites

The tables that you want to synchronize contain primary keys.

Precautions

- DTS uses the read and write resources of the source and destination databases during initial full data

synchronization. This may increase the load of the database server. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours.

- We recommend that you do not change the network type of the Cloud Native Distributed Database PolarDB-X instances during data synchronization.

  > **Note** After you change the network type of a Cloud Native Distributed Database PolarDB-X instance during data synchronization, you must submit a ticket to resume the data synchronization instance.

- We recommend that you do not scale up or down the databases in the Cloud Native Distributed Database PolarDB-X instances. Otherwise, data may fail to be synchronized.

Supported synchronization topologies

DTS supports the following synchronization topologies: one-way one-to-one synchronization, one-way one-to-many synchronization, one-way cascade synchronization, and one-way many-to-one synchronization. For more information, see Synchronization topologies.

SQL operations that can be synchronized

The INSERT, UPDATE, and DELETE operations can be synchronized.

Before you begin

Create a database and tables in the destination instance based on the schemas of the objects in the source instance. This is because DTS does not support **initial schema synchronization** between Cloud Native Distributed Database PolarDB-X instances.

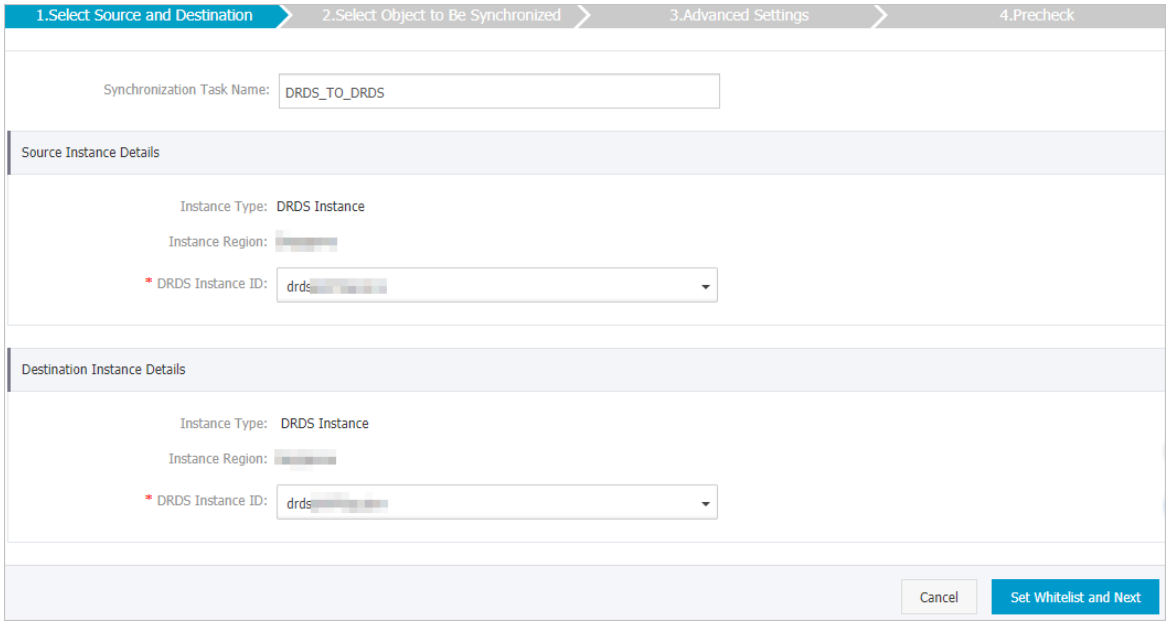
> **Note** During **initial schema synchronization**, DTS synchronizes the schemas of the required objects from the source database to the destination database.

Procedure

1. Create a data synchronization instance.

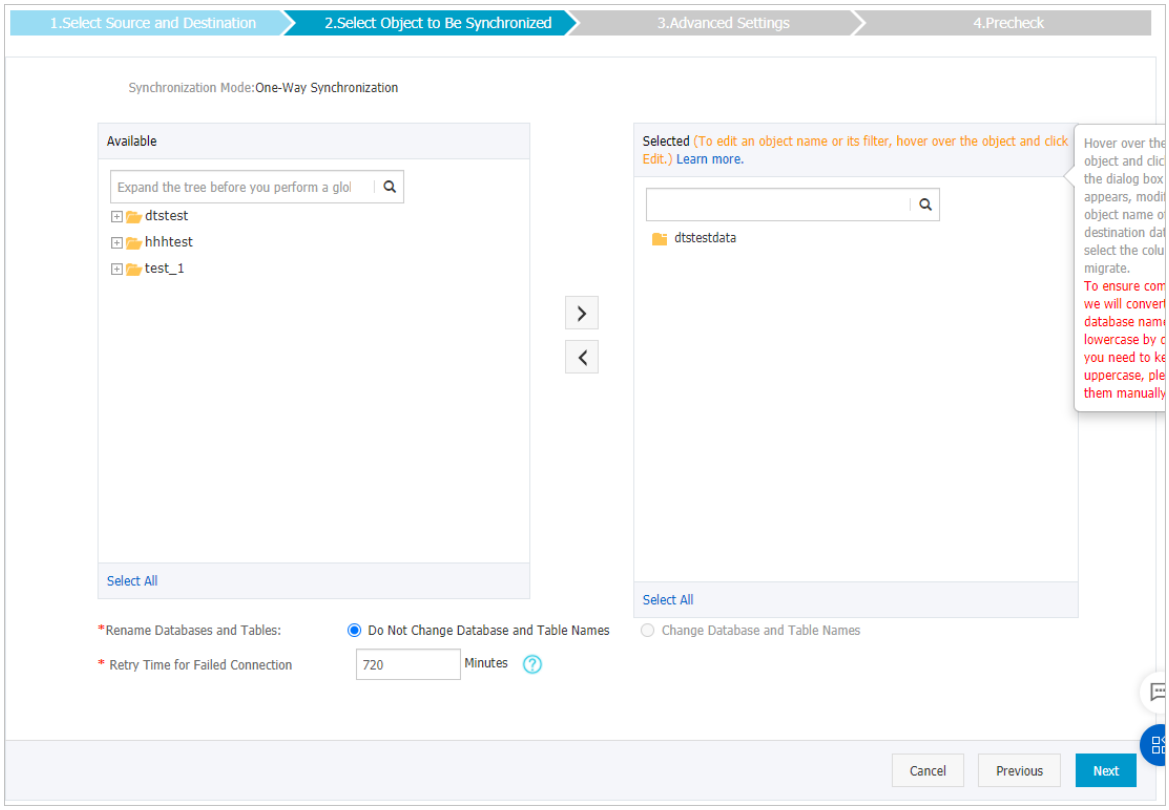
   > **Note** When you create the data synchronization instance, set both Source Instance Type and Destination Instance Type to **Drds**, and set Synchronization Mode to **One-Way Synchronization**.

2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.



Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	This parameter is set to **DRDS Instance** and cannot be changed.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the source Cloud Native Distributed Database PolarDB-X instance.
Destination Instance Details	Instance Type	This parameter is set to **DRDS Instance** and cannot be changed.
	Instance Region	The region of the destination instance. The region is the same as the destination region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the destination Cloud Native Distributed Database PolarDB-X instance.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Configure the synchronization policy and objects.



Parameter	Description
Processing Mode In Existed Target Table	◦ **Pre-check and Intercept**: checks whether the destination tables are empty. If the destination tables are empty, the precheck is passed. If the tables are not empty, an error is returned during the precheck and the data synchronization task cannot be started. ◦ **Ignore**: skips the check for empty destination tables. **Warning** If you select **Ignore**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, DTS does not synchronize the data records that have the same primary keys as the data records in the destination database. ■ If the source and destination databases have different schemas, initial data synchronization may fail. In this case, only specific columns are synchronized or the data synchronization task fails.

Parameter	Description
Select Objects	Select tables from the **Available** section and click the icon to move the tables to the **Selected** section. **Note** ◦ You can select only tables as the objects to be synchronized. ◦ After an object is synchronized to the destination database, the name of the object remains unchanged. You can use the object name mapping feature to change the names of the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.

6. Click **Next**.
7. Specify whether you want to perform initial full data synchronization.

   Note During **initial full data synchronization**, DTS synchronizes the historical data of the required objects from the source database to the destination database. If you do not select Initial Full Data Synchronization, DTS does not synchronize the historical data.

8. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

9. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.7. Synchronize data from a Cloud Native Distributed Database PolarDB-X instance to an AnalyticDB for MySQL cluster

This topic describes how to synchronize data from a Cloud Native Distributed Database PolarDB-X instance to an AnalyticDB for MySQL cluster by using Data Transmission Service (DTS).

Prerequisites

The tables that you want to synchronize contain primary keys.

Precautions

- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- Only one-way synchronization is supported.

Supported synchronization topologies

DTS supports the following synchronization topologies: one-way one-to-one synchronization, one-way one-to-many synchronization, and one-way many-to-one synchronization. For more information, see Synchronization topologies.

SQL operations that can be synchronized

INSERT, UPDATE, and DELETE

Data type mappings

The data types of ApsaraDB RDS for MySQL and AnalyticDB for MySQL do not have one-to-one correspondence. During initial schema synchronization, DTS converts the data types of the source database into those of the destination database. The following table lists the data types that DTS can convert.

Data type of ApsaraDB RDS for MySQL	Data type of AnalyticDB for MySQL
BIGINT UNSIGNED	DECIMAL(20,0)
BINARY	VARBINARY
BIT	VARCHAR
BLOB	VARBINARY
CHAR	VARCHAR
DATE	DATE
DATETIME	DATETIME
DECIMAL	DECIMAL
DOUBLE	DOUBLE
ENUM	VARCHAR

Data type of ApsaraDB RDS for MySQL	Data type of AnalyticDB for MySQL
FLOAT	FLOAT
GEOMETRY	VARBINARY
GEOMETRYCOLLECTION	VARBINARY
INT UNSIGNED	BIGINT
INTEGER	INT
JSON	JSON
LINESTRING	VARBINARY
LONGBLOB	VARBINARY
LONGTEXT	VARCHAR
MEDIUMBLOB	VARBINARY
MEDIUMINT	INT
MEDIUMINT UNSIGNED	INT
MEDIUMTEXT	VARCHAR
MULTILINESTRING	VARBINARY
MULTIPOINT	VARBINARY
MULTIPOLYGON	VARBINARY
NUMBERIC	DECIMAL
POINT	VARBINARY
POLYGON	VARBINARY
SET	VARCHAR
SMALLINT UNSIGNED	INT
TEXT	VARCHAR
TIME	TIME
TIMESTAMP	TIMESTAMP
TINYBLOB	VARBINARY
TINYINT UNSIGNED	SMALLINT

Data type of ApsaraDB RDS for MySQL	Data type of AnalyticDB for MySQL
TINYTEXT	VARCHAR
VARBINARY	VARBINARY
VARCHAR	VARCHAR
YEAR	INT

Procedure

1. Create a data synchronization instance.

   > **Note** When you create the data synchronization instance, set Source Instance Type to **Drds**, set Destination Instance Type to **AnalyticDB**, and set Synchronization Mode to **One-Way Synchronization**.

2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.

Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	This parameter is set to **DRDS Instance** and cannot be changed.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the source Cloud Native Distributed Database PolarDB-X instance.
Destination Instance Details	Instance Type	This parameter is set to **AnalyticDB** and cannot be changed.
	Instance Region	The region of the destination instance. The region is the same as the destination region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Version	Select **3.0**.
	Database	Select the ID of the destination AnalyticDB for MySQL cluster.
	Database Account	Enter the database account of the destination AnalyticDB for MySQL cluster.

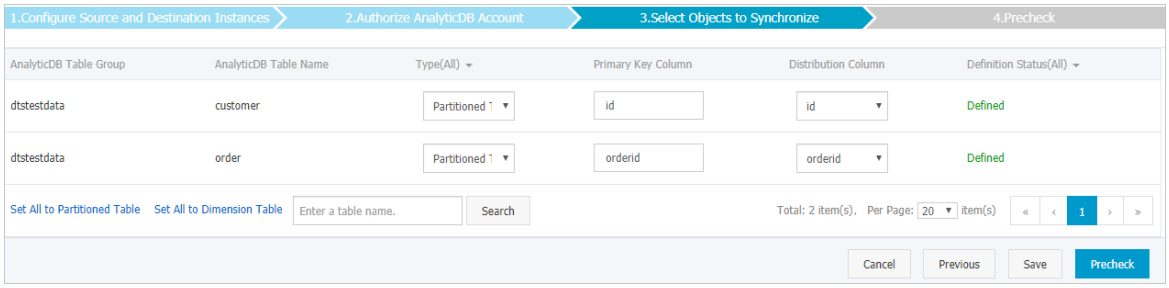
Section	Parameter	Description
	Database Password	Enter the password of the destination database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Configure the synchronization policy and objects

Section	Parameter	Description
Synchronization policy	Initial Synchronization	You must select both **Initial Schema Synchronization** and **Initial Full Data Synchronization** in most cases. After the precheck, DTS synchronizes the schemas and data of the required objects from the source instance to the destination cluster. The schemas and data are the basis for subsequent incremental synchronization.
	Processing Mode In Existed Target Table	○ **Clear Target Table** Skips the **Schema Name Conflict** item during the precheck. Clears the data in the destination table before initial full data synchronization. If you want to synchronize your business data after testing the data synchronization task, you can select this mode. ○ **Ignore** Skips the **Schema Name Conflict** item during the precheck. Adds data to the existing data during initial full data synchronization. If you want to synchronize data from multiple tables to one table, you can select this mode.
	Synchronization Type	Select the types of operations that you want to synchronize based on your business requirements. ○ **Insert** ○ **Update** ○ **Delete**

Section	Parameter	Description
Select Objects	N/A	Select tables from the **Available** section and click the icon to move the tables to the **Selected** section. **Note** ○ You can select only tables as the objects to be synchronized. ○ You can use the object name mapping feature to change the names of the columns that are synchronized to the destination database. For more information, see Specify the name of an object in the destination instance.

6. Click **Next**.
7. Specify a type for the tables that you want to synchronize to the destination database.



Note After you select **Initial Schema Synchronization**, you must specify the **type**, **primary key column**, and **partition key column** for the tables that you want to synchronize to AnalyticDB for MySQL.

8. In the lower-right corner of the page, click **Precheck**.

Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

Troubleshoot the issues based on the causes and run a precheck again.

9. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.8. Synchronize data from a Cloud Native Distributed Database PolarDB-X instance to a DataHub instance

This topic describes how to synchronize data from a Cloud Native Distributed Database PolarDB-X instance to a DataHub instance by using Data Transmission Service (DTS). After you synchronize data, you can use big data services such as Realtime Compute to analyze data in real time.

Prerequisites

- The tables that you want to synchronize have PRIMARY KEY or UNIQUE constraints.
- A DataHub project is created to receive the synchronized data.

Precautions

- If you select one or more tables (not a database) as the required objects, do not use gh-ost or pt-online-schema-change to perform data definition language (DDL) operations on the tables during data synchronization. Otherwise, data may fail to be synchronized.
- The source database must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
- Only one-way synchronization is supported.

Limits

- You can select only tables as the objects to be synchronized.
- Initial full data synchronization is not supported. DTS does not synchronize the historical data of the required objects from the source PolarDB-X instance to the destination DataHub instance.
- DTS does not synchronize data definition language (DDL) operations to the destination database. If you perform a DDL operation on the source PolarDB-X instance during data synchronization, data fails to be synchronized. To solve this issue, you must modify the related topic in the destination DataHub instance and then restart the data synchronization task.

SQL operations that can be synchronized

INSERT, UPDATE, and DELETE

Procedure

1. Create a data synchronization instance.

   > **Note** When you create the data synchronization instance, set Source Instance Type to **Drds**, set Destination Instance Type to **Datahub**, and set Synchronization Mode to **One-Way Synchronization**.

2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.

Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.

Section	Parameter	Description
Source Instance Details	Instance Type	This parameter is set to **DRDS Instance** and cannot be changed.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the source Cloud Native Distributed Database PolarDB-X instance.
Destination Instance Details	Instance Type	This parameter is set to **DataHub** and cannot be changed.
	Instance Region	The region of the destination instance. The region is the same as the destination region that you selected when you created the data synchronization instance. You cannot change the value of this parameter.
	Project	Select the name of the DataHub project.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Configure the synchronization policy and objects.

Parameter	Description
Initial Synchronization	Select **Initial Schema Synchronization**. **Note** After you select **Initial Schema Synchronization**, DTS synchronizes the schemas of the required objects (such as tables) to the destination DataHub instance.

Parameter	Description
Select Objects	Select objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** ◦ You can select only tables as the objects to be synchronized. ◦ You can use the object name mapping feature to change the names of the columns that are synchronized to the destination database. For more information, see Specify the name of an object in the destination instance.

6. In the lower-right corner of the page, click **Precheck**.

   Note You can start the data migration task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.

4.4.9. Synchronize data from a self-managed Oracle database to a DataHub project

DataHub is a real-time data distribution platform that is designed to process streaming data. You can publish and subscribe to streaming data in DataHub and distribute the data to other platforms. DataHub allows you to analyze and use streaming data. This topic describes how to synchronize data from a self-managed Oracle database to a DataHub project by using Data Transmission Service (DTS). After you synchronize data, you can use big data services such as Realtime Compute for Apache Flink to analyze the data in real time.

Prerequisites

- The engine version of the self-managed Oracle database is 9i, 10g, 11g, 12c, 18c, or 19c.
- The self-managed Oracle database is running in ARCHIVELOG mode. Archived log files are accessible, and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the self-managed Oracle database. For more information, see Supplemental Logging.
- A DataHub project is created as the destination of data synchronization. For more information, see *Create a project* in *DataHub User Guide*.

Limits

Category	Description

Limits on the source database	• Requirements for the objects to be synchronized:
	  ◦ The tables to be synchronized must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	  ◦ If the version of your Oracle database is 12c or later, the names of the tables to be synchronized cannot exceed 30 bytes in length.
	  ◦ If you select tables as the objects to be synchronized and you need to edit the tables (such as renaming tables or columns) in the destination database, you can synchronize up to 1,000 tables in a single data synchronization task. If you run a task to synchronize more than 1,000 tables, a request error occurs. In this case, we recommend that you configure multiple tasks to synchronize the tables in batches or configure a task to synchronize the entire database.
	• If the self-managed Oracle database is an Oracle RAC database, you can use only a single virtual IP address (VIP) rather than a Single Client Access Name (SCAN) IP address when you configure the data synchronization task. After you specify the VIP, node failover of the Oracle RAC database is not supported.
	• The following requirements must be met:
	  ◦ The redo logging and archive logging features must be enabled.
	  ◦ If you perform only incremental data synchronization, the redo logs and archive logs of the source database must be stored for more than 24 hours. If you perform both schema synchronization and incremental data synchronization, the redo logs and archive logs of the source database must be stored for at least seven days. Otherwise, DTS may fail to obtain the redo logs and archive logs and the task may fail. In exceptional circumstances, data inconsistency or loss may occur. After the schema synchronization is complete, you can set the retention period to more than 24 hours. Make sure that you set the retention period of redo logs and archive logs based on the preceding requirements. Otherwise, the service reliability and performance stated in the Service Level Agreement (SLA) of DTS may not be achieved.
	• If you perform a primary/secondary switchover on the source database when the data synchronization task is running, the task fails.

Category	Description
Other limits	• You can configure a data synchronization task for this scenario only in the new DTS console. • The objects of initial schema synchronization must be tables. **Warning** In this scenario, DTS is incompatible with triggers. We recommend that you delete the triggers of the source database to prevent data inconsistency caused by triggers. • Full data synchronization is not supported. DTS does not synchronize historical data of the required objects from the source database to the destination DataHub project. • Only tables can be selected as the objects to be synchronized. • We recommend that you do not use gh-ost or pt-online-schema-change to perform DDL operations on objects. Otherwise, data synchronization may fail. • DTS calculates synchronization latency based on the timestamp of the latest synchronized data in the destination database and the current timestamp in the source database. If no DML operation is performed on the source database for a long time, the synchronization latency may be inaccurate. If the latency of the synchronization task is too high, you can perform a DML operation on the source database to update the latency. **Note** If you select an entire database as the object to be synchronized, you can create a heartbeat table. The heartbeat table is updated or receives data every second.

Supported synchronization topologies

- One-way one-to-one synchronization
- One-way one-to-many synchronization
- One-way many-to-one synchronization

SQL operations that can be synchronized

INSERT, UPDATE, and DELETE

Permissions required for database accounts

Database	Required permission	References

Database	Required permission	References
Self-managed Oracle database	Database administrator (DBA)	CREATE USER and GRANT **Notice** If you need to synchronize data from an Oracle database but the DBA permissions cannot be granted to the database account, you can enable archive logging and supplemental logging, and grant fine-grained permissions to the account.

Enable logging and grant fine-grained permissions to an Oracle database account

Notice If you need to synchronize data from an Oracle database but the DBA permissions cannot be granted to the database account, you can enable archive logging and supplemental logging, and grant fine-grained permissions to the account.

1. Enable archive logging and supplemental logging.

Type	Procedure

Archive logging	Execute the following statements to enable archive logging:
	`shutdown immediate;`
	`startup mount;`
	`alter database archivelog;`
	`alter database open;`
	`archive log list;`

<table>
<tr><th>Type</th><th>Procedure</th></tr>
<tr><td>Supplemental logging</td><td>
Enable supplemental logging at the database or table level:
Note You can enable database-level supplemental logging to ensure the stability of DTS tasks. You can enable table-level supplemental logging to reduce the disk usage of the source Oracle database.
○ Enable database-level supplemental logging
a. Execute the following statement to enable minimal supplemental logging:
<code>alter database add supplemental log data;</code>
b. Execute the following statement to enable primary key and unique key supplemental logging at the database level:
<code>alter database add supplemental log data (primary key,unique index) columns;</code>
○ Enable table-level supplemental logging
a. Execute the following statement to enable minimal supplemental logging:
<code>alter database add supplemental log data;</code>
b. Enable table-level supplemental logging by using one of the following methods:
■ Execute the following statement to enable primary key supplemental logging at the table level:
<code>alter table table_name add supplemental log data (primary key) columns;</code>
■ Execute the following statement to enable table-level supplemental logging for all columns:
<code>alter table tb_name add supplemental log data (all) columns ;</code>
</td></tr>
</table>

2. Grant fine-grained permissions to an Oracle database account
   - Oracle 9i to 11g

```
# Create a database account, such as rdsdt_dtsacct, and grant permissions to the acco
unt.
create user rdsdt_dtsacct IDENTIFIED BY rdsdt_dtsacct;
grant create session to rdsdt_dtsacct;
grant connect to rdsdt_dtsacct;
grant resource to rdsdt_dtsacct;
grant execute on sys.dbms_logmnr to rdsdt_dtsacct;
grant select on V_$LOGMNR_LOGS to rdsdt_dtsacct;
grant select on  all_objects to rdsdt_dtsacct;
grant select on  all_tab_cols to rdsdt_dtsacct;
grant select on  dba_registry to rdsdt_dtsacct;
grant select any table to rdsdt_dtsacct;
grant select any transaction to rdsdt_dtsacct;
-- v$log privileges
grant select on v_$log to rdsdt_dtsacct;
-- v$logfile privileges
grant select on v_$logfile to rdsdt_dtsacct;
-- v$archived_log privileges
grant select on v_$archived_log to rdsdt_dtsacct;
-- v$parameter privileges
grant select on v_$parameter to rdsdt_dtsacct;
-- v$database privileges
grant select on v_$database to rdsdt_dtsacct;
-- v$active_instances privileges
grant select on v_$active_instances to rdsdt_dtsacct;
-- v$instance privileges
grant select on v_$instance to rdsdt_dtsacct;
-- v$logmnr_contents privileges
grant select on v_$logmnr_contents to rdsdt_dtsacct;
-- system tables
grant select on sys.USER$ to rdsdt_dtsacct;
grant select on SYS.OBJ$ to rdsdt_dtsacct;
grant select on SYS.COL$ to rdsdt_dtsacct;
grant select on SYS.IND$ to rdsdt_dtsacct;
grant select on SYS.ICOL$ to rdsdt_dtsacct;
grant select on SYS.CDEF$ to rdsdt_dtsacct;
grant select on SYS.CCOL$ to rdsdt_dtsacct;
grant select on SYS.TABPART$ to rdsdt_dtsacct;
grant select on SYS.TABSUBPART$ to rdsdt_dtsacct;
grant select on SYS.TABCOMPART$ to rdsdt_dtsacct;
```

- Oracle 12c to 19c (tenant mode)

```
# Switch to the pluggable database (PDB). Create a database account, such as rdsdt_dt
sacct, and grant permissions to the account.
ALTER SESSION SET container = ORCLPDB1;
create user rdsdt_dtsacct IDENTIFIED BY rdsdt_dtsacct;
grant create  session to rdsdt_dtsacct;
grant connect  to rdsdt_dtsacct;
grant resource to rdsdt_dtsacct;
grant execute on sys.dbms_logmnr to rdsdt_dtsacct;
grant select on  all_objects to rdsdt_dtsacct;
grant select on  all_tab_cols to rdsdt_dtsacct;
grant select on  dba_registry to rdsdt_dtsacct;
```

```
grant select any table to rdsdt_dtsacct;
grant select any transaction to rdsdt_dtsacct;
-- v$log privileges
grant select on v_$log to rdsdt_dtsacct;
-- v$logfile privileges
grant select on v_$logfile to rdsdt_dtsacct;
-- v$archived_log privileges
grant select on v_$archived_log to rdsdt_dtsacct;
-- v$parameter privileges
grant select on v_$parameter to rdsdt_dtsacct;
-- v$database privileges
grant select on v_$database to rdsdt_dtsacct;
-- v$active_instances privileges
grant select on v_$active_instances to rdsdt_dtsacct;
-- v$instance privileges
grant select on v_$instance to rdsdt_dtsacct;
-- v$logmnr_contents privileges
grant select on v_$logmnr_contents to rdsdt_dtsacct;
grant select on sys.USER$ to rdsdt_dtsacct;
grant select on SYS.OBJ$ to rdsdt_dtsacct;
grant select on SYS.COL$ to rdsdt_dtsacct;
grant select on SYS.IND$ to rdsdt_dtsacct;
grant select on SYS.ICOL$ to rdsdt_dtsacct;
grant select on SYS.CDEF$ to rdsdt_dtsacct;
grant select on SYS.CCOL$ to rdsdt_dtsacct;
grant select on SYS.TABPART$ to rdsdt_dtsacct;
grant select on SYS.TABSUBPART$ to rdsdt_dtsacct;
grant select on SYS.TABCOMPART$ to rdsdt_dtsacct;
-- V$PDBS privileges
grant select on V_$PDBS to rdsdt_dtsacct;
# Switch to the container database (CDB). Create a database account and grant permiss
ions to the account by using one of the following methods:
ALTER SESSION SET container = CDB$ROOT;
# Method 1: Create a global account that starts with C##, such as C##rdsdt_dtsacct, a
nd grant permissions to the account. If you use this method, you must submit a ticket
and contact the DTS team to modify some parameters.
create user C##rdsdt_dtsacct IDENTIFIED BY rdsdt_dtsacct;
grant create session to C##rdsdt_dtsacct;
grant connect to C##rdsdt_dtsacct;
grant resource to C##rdsdt_dtsacct;
grant select on v_$logmnr_contents to C##rdsdt_dtsacct;
grant LOGMINING to C##rdsdt_dtsacct;
grant EXECUTE_CATALOG_ROLE to C##rdsdt_dtsacct;
grant execute on sys.dbms_logmnr to C##rdsdt_dtsacct;
# Method 2: Create a database account, such as rdsdt_dtsacct, and grant permissions t
o the account. If you use this method, you must modify the default parameters of the
Oracle database.
alter session set "_ORACLE_SCRIPT"=true;
create user rdsdt_dtsacct IDENTIFIED BY rdsdt_dtsacct;
grant create session to rdsdt_dtsacct;
grant connect to rdsdt_dtsacct;
grant select on v_$logmnr_contents to rdsdt_dtsacct;
grant LOGMINING TO rdsdt_dtsacct;
grant EXECUTE_CATALOG_ROLE to rdsdt_dtsacct;
grant execute on sys.dbms_logmnr to rdsdt_dtsacct;
```

```
grant execute on sys.dbms_logmnr to rdsdt_dtsacct;
```

- Oracle 12c to 19c (non-tenant mode)

```
# Create a database account, such as rdsdt_dtsacct, and grant permissions to the acco
unt.
create user rdsdt_dtsacct IDENTIFIED BY rdsdt_dtsacct;
grant create  session to rdsdt_dtsacct;
grant connect  to rdsdt_dtsacct;
grant resource to rdsdt_dtsacct;
grant select on V_$LOGMNR_LOGS to rdsdt_dtsacct;
grant select on  all_objects to rdsdt_dtsacct;
grant select on  all_tab_cols to rdsdt_dtsacct;
grant select on  dba_registry to rdsdt_dtsacct;
grant select any table to rdsdt_dtsacct;
grant select any transaction to rdsdt_dtsacct;
-- v$log privileges
grant select on v_$log to rdsdt_dtsacct;
-- v$logfile privileges
grant select on v_$logfile to rdsdt_dtsacct;
-- v$archived_log privileges
grant select on v_$archived_log to rdsdt_dtsacct;
-- v$parameter privileges
grant select on v_$parameter to rdsdt_dtsacct;
-- v$database privileges
grant select on v_$database to rdsdt_dtsacct;
-- v$active_instances privileges
grant select on v_$active_instances to rdsdt_dtsacct;
-- v$instance privileges
grant select on v_$instance to rdsdt_dtsacct;
-- v$logmnr_contents privileges
grant select on v_$logmnr_contents to rdsdt_dtsacct;
grant select on sys.USER$ to rdsdt_dtsacct;
grant select on SYS.OBJ$ to rdsdt_dtsacct;
grant select on SYS.COL$ to rdsdt_dtsacct;
grant select on SYS.IND$ to rdsdt_dtsacct;
grant select on SYS.ICOL$ to rdsdt_dtsacct;
grant select on SYS.CDEF$ to rdsdt_dtsacct;
grant select on SYS.CCOL$ to rdsdt_dtsacct;
grant select on SYS.TABPART$ to rdsdt_dtsacct;
grant select on SYS.TABSUBPART$ to rdsdt_dtsacct;
grant select on SYS.TABCOMPART$ to rdsdt_dtsacct;
grant LOGMINING TO rdsdt_dtsacct;
grant EXECUTE_CATALOG_ROLE to rdsdt_dtsacct;
grant execute on sys.dbms_logmnr to rdsdt_dtsacct;
```

Note To create a global account that starts with C## for a multitenant CDB$ROOT in Oracle 12c to 19c (tenant mode), you must contact the O&M engineers to modify some parameters.

Procedure

1. Create a data synchronization instance.

> **Note** When you create the data synchronization instance, select **Oracle** as the source instance type, **DataHub** as the destination instance type, and **One-way Synchronization** as the synchronization topology.

2. Find the data synchronization instance, and click **Configure Synchronization Channel** in the **Actions** column.
3. Configure the source and destination instances.

Section	Parameter	Description
N/A	Synchronization Task Name	The task name that DTS automatically generates. We recommend that you specify a descriptive name that makes it easy to identify the task. You do not need to use a unique task name.
Source Instance Details	Instance Type	Select **User-Created Database with Public IP Address**.
	Instance Region	The source region that you selected on the buy page. You cannot change the value of this parameter.
	Database Type	The value of this parameter is fixed to **Oracle**.
	Hostname or IP Address	The endpoint or IP address that is used to connect to the self-managed Oracle database.
	Port Number	The service port number of the self-managed Oracle database. Default value: 1521.
	Instance Type	◦ **Non-RAC Instance**: If you select this option, you must specify the **SID** parameter. ◦ **RAC or PDB Instance**: You cannot select this option because RAC instances are not supported.
	Database Account	The account of the self-managed Oracle database.
	Database Password	The password of the database account.
Destination Instance Details	Instance Type	The value of this parameter is fixed to **DataHub**.
	Instance Region	The destination region that you selected on the buy page. You cannot change the value of this parameter.
	Project	The name of the DataHub **project**.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Configure the synchronization policy and the objects to be synchronized.

Setting	Description
Specify initial synchronization	Select **Initial Schema Synchronization**. **Note** After you select **Initial Schema Synchronization**, DTS synchronizes the schemas of the required objects such as tables to the destination DataHub instance.
Select objects to be synchronized	Select one or more objects from the **Available** section and click the icon to move the objects to the **Selected** section. **Note** - You can select only tables as the objects to be synchronized. - By default, after an object is synchronized to the destination instance, the name of the object remains unchanged. You can use the object name mapping feature to rename the objects that are synchronized to the destination instance. For more information, see Specify the name of an object in the destination instance.
Specify whether to rename objects	You can use the object name mapping feature to rename the objects that are synchronized to the destination instance. For more information, see Object name mapping.
Specify the retry time range for a failed connection to the source or destination instance.	By default, if DTS fails to connect to the source or destination database, DTS retries within the next 720 minutes (12 hours). You can specify the retry time range based on your business requirements. If DTS reconnects to the source and destination databases within the specified time, DTS resumes the data synchronization task. Otherwise, the data synchronization task fails.
Enable new naming rules for additional columns	After DTS synchronizes data to DataHub, DTS adds additional columns to the destination topic. If the names of additional columns are the same as the names of existing columns in the destination topic, data synchronization fails. Select **Yes** or **No** to specify whether you want to enable new naming rules for additional columns. **Warning** Before you specify this parameter, check whether additional columns and existing columns in the destination topic have name conflicts.

6.
7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.** Then, DTS performs initial synchronization.
8. Wait until initial synchronization is complete and the data synchronization task enters the **Synchronizing** state.

   You can view the state of the data synchronization task on the **Synchronization Tasks** page.

Schema of a DataHub topic

When DTS synchronizes incremental data to a DataHub topic, DTS adds additional columns to store metadata. The following figure shows the schema of a DataHub topic.

> **Note** In this example, `id`, `name`, `address` are data fields. DTS adds the `dts_` prefix to data fields because the previous naming rules for additional columns are used.

dts_id	dts_name	dts_address	dts_record_id	dts_operation_flag	dts_instance_id	dts_db_name	dts_table_name	dts_utc_timestamp	dts_before_flag	dts_after_flag
10006	[redacted]	[redacted]	1574832130000000000	U		dtstestdata	customer	1574832130	Y	N
10006	[redacted]	[redacted]	1574832130000000000	U		dtstestdata	customer	1574832130	N	Y
10009	[redacted]	[redacted]	1574832919000000000	D		dtstestdata	customer	1574832919	Y	N
10112	[redacted]	[redacted]	1574832919000000000	I		dtstestdata	customer	1574832919	N	Y

The following table describes the additional columns in the DataHub topic.

Previous additional column name	New additional column name	Category	Description
`dts_record_id`	`new_dts_sync_dts_record_id`	String	The ID of the incremental log entry. **Note** • By default, the ID auto-increments for each new log entry. In disaster recovery scenarios, rollback may occur, and the ID may not auto-increment. Therefore, some IDs may be duplicated. • If an UPDATE operation is performed, DTS generates two incremental log entries to record the pre-update and post-update values. The values of the `dts_record_id` field in the two incremental log entries are the same.

Previous additional column name	New additional column name	Category	Description
dts_operation_flag	new_dts_sync_dts_operation_flag	String	The operation type. Valid values: • I: an INSERT operation • D: a DELETE operation • U: an UPDATE operation
dts_instance_id	new_dts_sync_dts_instance_id	String	The server ID of the database. The value is set to null. To ensure database security, the actual value is not displayed.
dts_db_name	new_dts_sync_dts_db_name	String	The name of the database.
dts_table_name	new_dts_sync_dts_table_name	String	The name of the table.
dts_utc_timestamp	new_dts_sync_dts_utc_timestamp	String	The operation timestamp, in UTC. It is also the timestamp of the redo log file.
dts_before_flag	new_dts_sync_dts_before_flag	String	Indicates whether the column values are original values. Valid values: Y and N.
dts_after_flag	new_dts_sync_dts_after_flag	String	Indicates whether the column values are updated values. Valid values: Y and N.

Additional information about the dts_before_flag and dts_after_flag fields

The values of the dts_before_flag and dts_after_flag fields in an incremental log entry vary with different operation types:

- INSERT operation

  For an INSERT operation, the column values are the newly inserted record values (updated values). The value of the dts_before_flag field is N, and the value of the dts_after_flag field is Y.

dts_id	dts_name	dts_address	dts_record_id	dts_operation_flag	dts_instance_id	dts_db_name	dts_table_name	dts_utc_timestamp	dts_before_flag	dts_after_flag
10112	[redacted]	[redacted]	1574832919000000000	I		dtstestdata	customer	1574832919	N	Y

- UPDATE operation

  DTS generates two incremental log entries for an UPDATE operation. The two incremental log entries have the same values for the dts_record_id, dts_operation_flag, and dts_utc_timestamp fields.

  The first log entry records the original values. Therefore, the value of the dts_before_flag field is Y, and the value of the dts_after_flag field is N. The second log entry records the updated values. Therefore, the value of the dts_before_flag field is N, and the value of the dts_after_flag field is Y.

dts_id	dts_name	dts_address	dts_record_id	dts_operation_flag	dts_instance_id	dts_db_name	dts_table_name	dts_utc_timestamp	dts_before_flag	dts_after_flag
10006	[redacted]	[redacted]	1574832130000000000	U		dtstestdata	customer	1574832130	Y	N
10006	[redacted]	[redacted]	1574832130000000000	U		dtstestdata	customer	1574832130	N	Y

- DELETE operation

  For a DELETE operation, the column values are the deleted record values (original values). The value of the `dts_before_flag` field is Y, and the value of the `dts_after_flag` field is N.

dts_id	dts_name	dts_address	dts_record_id	dts_operation_flag	dts_instance_id	dts_db_name	dts_table_name	dts_utc_timestamp	dts_before_flag	dts_after_flag
10009	[redacted]	[redacted]	1574832919000000000	D		dtstestdata	customer	1574832919	Y	N

4.4.10. Synchronize data between ApsaraDB for Redis instances

You can use Data Transmission Service (DTS) to perform one-way or two-way data synchronization between ApsaraDB for Redis instances. This feature is applicable to scenarios such as resource migration or resource merging and business architecture adjustment.

Prerequisites

- The engine version of the source ApsaraDB for Redis instance is 4.0 or 5.0 if the instance is a Community Edition instance, and is 5.0 if the instance is an Enhanced Edition instance.

  > **Note** The engine version of the destination ApsaraDB for Redis instance can be 4.0 or 5.0. The engine version of the destination ApsaraDB for Redis instance cannot be earlier than that of the source ApsaraDB for Redis instance. Before you synchronize data between two ApsaraDB for Redis instances that use different engine versions, check the compatibility of the two versions.

- The source ApsaraDB for Redis instance is deployed in a virtual private cloud (VPC).
- SSL encryption is disabled for the source ApsaraDB for Redis instance.
- The available storage space of the destination ApsaraDB for Redis instance is larger than the total size of the data in the source ApsaraDB for Redis instance.
- ApsaraDB for Redis Enhanced Edition (Tair) is integrated with more Redis modules than ApsaraDB for Redis Community Edition. To ensure compatibility between the source and destination instances, the edition of the destination instance must be Enhanced if the edition of the source instance is Enhanced.

Precautions

Category	Description

Category	Description

Limits on the source database	• The collections to synchronize must have PRIMARY KEY or UNIQUE constraints, and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	• To ensure the synchronization quality, DTS adds the following key to the source database: DTS_REDIS_TIMESTAMP_HEARTBEAT. This key is used to record the time when data is synchronized to the destination database.
	• To ensure the stability of data synchronization, we recommend that you increase the value of the *repl-backlog-size* parameter in the `redis.conf` file.
	• The following requirements for append-only file (AOF) logging must be met:
	  ◦ The AOF logging feature must be enabled.
	  ◦ If you perform only incremental data synchronization, the AOF logs of the source database must be stored for more than 24 hours. If you perform both full data synchronization and incremental data synchronization, the AOF logs of the source database must be stored for at least seven days. After the full data synchronization is complete, you can set the retention period to more than 24 hours. Otherwise, DTS may fail to obtain the AOF logs and your data synchronization task may fail. In exceptional circumstances, data inconsistency or loss may occur. Make sure that you set the retention period of AOF logs in accordance with the preceding requirements. Otherwise, the service reliability or performance in the Service Level Agreement (SLA) of DTS may not be achieved.
	  Note This limit is not applied to ApsaraDB for Redis instances.
	• If an expiration policy is enabled for some keys in the source database, these keys may not be deleted in a timely manner after they expired. Therefore, the number of keys in the destination database may be less than that in the source database. You can run the info command to view the number of keys in the destination database.
	• If the data eviction policy (`maxmemory-policy`) of the destination database is not set to `noeviction` , data inconsistency may occur between the source and destination databases.

Category	Description
Other limits	• During data synchronization, if the number of shards in the source ApsaraDB for Redis instance is increased or decreased, or if the database specifications are changed (for example, the memory capacity is scaled up), you must reconfigure the task. To ensure data consistency, we recommend that you clear the data that has been synchronized to the destination Redis instance before you reconfigure the task. • During data synchronization, if the endpoint of the source ApsaraDB for Redis instance is changed, you must submit a ticket to update the change. Otherwise, the AOF logs of the ApsaraDB for Redis instance may be reset. In this case, you must reconfigure the task. • To ensure compatibility, the version of the destination database must be the same as or later than that of the source database. If the version of the destination database is earlier than that of the source database, database compatibility issues may occur. • Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours. During full data synchronization, DTS uses read and write resources of the source and destination databases. This may increase the loads on the database servers. • During data synchronization, we recommend that you use only DTS to write data to the destination database. This prevents data inconsistency between the source and destination databases. If you use tools other than DTS to write data to the destination database, data loss may occur in the destination database when you use Data Management (DMS) to perform online DDL operations.

SQL operations that can be synchronized

Edition	SQL operation

Edition	SQL operation
ApsaraDB for Redis Community Edition	• APPEND • BITOP, BLPOP, BRPOP, and BRPOPLPUSH • DECR, DECRBY, and DEL • EVAL, EVALSHA, EXEC, EXPIRE, and EXPIREAT • GEOADD and GETSET • HDEL, HINCRBY, HINCRBYFLOAT, HMSET, HSET, and HSETNX • INCR, INCRBY, and INCRBYFLOAT • LINSERT, LPOP, LPUSH, LPUSHX, LREM, LSET, and LTRIM • MOVE, MSET, MSETNX, and MULTI • PERSIST, PEXPIRE, PEXPIREAT, PFADD, PFMERGE, and PSETEX • RENAME, RENAMENX, RESTORE, RPOP, RPOPLPUSH, RPUSH, and RPUSHX • SADD, SDIFFSTORE, SELECT, SET, SETBIT, SETEX, SETNX, SETRANGE, SINTERSTORE, SMOVE, SPOP, SREM, and SUNIONSTORE • ZADD, ZINCRBY, ZINTERSTORE, ZREM, ZREMRANGEBYLEX, ZUNIONSTORE, ZREMRANGEBYRANK, and ZREMRANGEBYSCORE • SWAPDB and UNLINK. These two SQL operations can be synchronized only if the engine version of the source instance is Redis 4.0.
ApsaraDB for Redis Enhanced Edition (Tair)	• APPEND • BITOP, BLPOP, BRPOP, and BRPOPLPUSH • DECR, DECRBY, and DEL • EVAL, EVALSHA, EXEC, EXPIRE, and EXPIREAT • GEOADD and GETSET • HDEL, HINCRBY, HINCRBYFLOAT, HMSET, HSET, and HSETNX • INCR, INCRBY, and INCRBYFLOAT • LINSERT, LPOP, LPUSH, LPUSHX, LREM, LSET, and LTRIM • MOVE, MSET, MSETNX, and MULTI • PERSIST, PEXPIRE, PEXPIREAT, PFADD, PFMERGE, and PSETEX • RENAME, RENAMENX, RPOP, RPOPLPUSH, RPUSH, and RPUSHX • SADD, SDIFFSTORE, SELECT, SET, SETBIT, SETEX, SETNX, SETRANGE, SINTERSTORE, SMOVE, SPOP, SREM, and SUNIONSTORE • UNLINK, ZADD, ZINCRBY, ZINTERSTORE, ZREM, ZREMRANGEBYLEX, ZUNIONSTORE, ZREMRANGEBYRANK, and ZREMRANGEBYSCORE

Note

- PUBLISH operations cannot be synchronized.
- If you run the EVAL or EVALSHA command to call Lua scripts, DTS cannot identify whether these Lua scripts are executed on the destination instance. This is because the destination instance does not explicitly return the execution results of Lua scripts during incremental data synchronization.
- When DTS runs the SYNC or PSYNC command to transfer data of the LIST type, DTS does not clear the existing data in the destination instance. As a result, the destination instance may contain duplicate data records.

Step 1: Create a data synchronization task

1. Log on to the new version of the DMS console. For more information, see Log on to the DTS console.
2. In the top navigation bar, click **DTS**. In the left-side navigation pane, click **Data Synchronization**.
3. On the Data Synchronization page, click **Create Task**.
4. In the dialog box that appears, set the required parameters.

Parameter	Description
Source Instance Type	Select **Redis**.
Source Instance Region	Select the region in which the source instance resides.
Destination Instance Type	Select **Redis**.
Destination Instance Region	Select the region in which the destination instance resides.
Synchronization Mode	Select **One-Way Synchronization** or **Two-Way Synchronization** as required. Note You can select only **Two-Way Synchronization** if the source and destination instances reside in the same zone.
Instances to Create	The number of data synchronization instances that you want to purchase at a time. The default value is 0.

5. Click **Purchase**.

Step 2: Configure the source and destination databases

1. In the **Data Synchronization Tasks** list, find the data synchronization task that you create and click **Configure Task** in the rightmost column.
2. Specify the information about the source database and destination database.

Section	Parameter	Description
N/A	**Task Name**	DTS automatically generates a task name. We recommend that you specify an informative name to identify the task. You do not need to use a unique task name.
Source Database	**Database Type**	Select **Redis**.
	Access Method	Select **Alibaba Cloud Instance**.
	Instance Region	Select the region in which the source ApsaraDB for Redis instance resides.
	Replicate Data Across Alibaba Cloud Accounts	Select **No**.
	Instance ID	Select the ID of the source ApsaraDB for Redis instance.
	Database Password	Specify the database account of the source ApsaraDB for Redis instance. The database account must have the read permissions. If you forget the password, you can reset the password. ◦ If you use the default account whose username is the same as the instance ID, you can enter only the password. ◦ If you create a custom account, enter the information in the `<Custom account username>:<Password>` format. Example: testaccount:Test1234. **Note** This parameter is optional and can be left empty if no password is set.
Destination Database	**Database Type**	Select **Redis**.
	Access Method	Select **Alibaba Cloud Instance**.
	Instance Region	Select the region in which the destination ApsaraDB for Redis instance resides.
	Instance ID	Select the ID of the destination ApsaraDB for Redis instance.

Section	Parameter	Description
	Database Password	Specify the database account of the destination ApsaraDB for Redis instance. The database account must have the read permissions. ○ If you use the default account whose username is the same as the instance ID, you can enter only the password. ○ If you create a custom account, enter the information in the `<Custom account username>:<Password>` format. Example: testaccount:Test1234.

3. In the lower part of the page, click **Test Connectivity and Proceed**.
4. Select objects for the task and configure advanced settings.
   - **Basic settings**

Parameter	Description
Task Stages	**Incremental Data Synchronization** is selected by default. You must also select **Full Data Synchronization**. After the precheck, DTS synchronizes the historical data of the selected objects from the source instance to the destination cluster. The historical data is the basis for subsequent incremental synchronization. **Note** Only ApsaraDB for Redis Enhanced Edition (Tair) supports full data synchronization.

Parameter	Description
Processing Mode of Conflicting Tables	■ **Precheck and Report Errors**: checks whether the destination database contains tables that have the same names as tables in the source database. If the source and destination databases do not contain identical table names, the precheck is passed. Otherwise, an error is returned during the precheck and the data synchronization task cannot be started. ■ **Ignore Errors and Proceed**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore Errors and Proceed**, data consistency is not guaranteed and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, and a data record has the same primary key as an existing data record in the destination database: ■ During full data synchronization, DTS does not synchronize the data record to the destination database. The existing data record in the destination database is retained. ■ During incremental data synchronization, DTS synchronizes the data record to the destination database. The existing data record in the destination database is overwritten. ■ If the source and destination databases have different schemas, data may fail to be initialized. In this case, only some columns are synchronized or the data synchronization task fails.
Select Objects	Select one or more objects from the **Source Objects** section and click the › icon to add the objects to the **Selected Objects** section. **Note** You can select only databases as objects to be synchronized. Keys cannot be selected as objects to be synchronized.

- **Advanced settings**

Parameter	Description
Retry Time for Failed Connection	Specify the retry time range for failed connections. Valid values: 10 to 1440. Unit: minutes. Default value: 120. We recommend that you set the retry time range to more than 30 minutes. If DTS reconnects to the source and destination databases within the specified time range, DTS resumes the data synchronization task. Otherwise, the data synchronization task fails. **Note** If multiple DTS instances have the same source or destination database, the lowest value of this parameter takes effect. For example, if the retry time range is set to 30 minutes for Instance A and 60 minutes for Instance B, DTS retries failed connections within 30 minutes.

5. Click **Next: Save Task Settings and Precheck** in the lower part of the page.

> **Note**
> - Before you can start the data synchronization task, DTS performs a precheck. You can start the data synchronization task only after the task passes the precheck.
> - If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
>   - You can troubleshoot the issues based on the causes and run a precheck again.
>   - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

After the value of **Success Rate** becomes **100%**, the data synchronization task starts to run. You can view the progress of the task on the Data Synchronization page.

4.4.11. Configure two-way data synchronization between RDS instances

4.4.11.1. Overview

DTS supports two-way real-time data synchronization between RDS instances on any two clouds. This section describes how to use DTS to create a two-way synchronization task between two ApsaraDB RDS for MySQL instances for active geo-redundancy, geo-disaster recovery, and other scenarios.

4.4.11.2. Supported synchronization statements

Two-way synchronization between ApsaraDB RDS for MySQL instances supports all DML updates (including INSERT, UPDATE, and DELETE) and the following DDL updates:

- ALTER TABLE, ALTER VIEW, ALTER FUNCTION, and ALTER PROCEDURE
- CREATE DATABASE, CREATE SCHEMA, CREATE INDEX, CREATE TABLE, CREATE PROCEDURE, CREATE FUNCTION, CREATE TRIGGER, CREATE VIEW, and CREATE EVENT

- DROP FUNCTION, DROP EVENT, DROP INDEX, DROP PROCEDURE, DROP TABLE, DROP TRIGGER, and DROP VIEW
- RENAME TABLE and TRUNCATE TABLE

> **Note** To ensure the stability of a two-way synchronization channel, you can synchronize DDL updates on the same table in only one direction.
>
> For example, for two-way synchronization, you must enable DDL synchronization in either the **A-to-B** or **B-to-A** direction. If DDL synchronization is configured in one direction, it is not supported in the reverse direction. You can only perform DML synchronization.

4.4.11.3. Detect and resolve conflicts

To ensure data consistency, for two-way synchronized instances, make sure that records with the same primary key, business primary key, or unique key are updated only on one of the instances. If you unexpectedly update a record with the same primary key, business primary key, or unique key on both instances that are two-way synchronized, a synchronization conflict occurs. To maximize the stability of two-way synchronized instances, DTS supports detecting and resolving data conflicts.

Considerations

During two-way synchronization, the system time of the source and destination instances may not be the same. Additionally, synchronization delays may occur. For these reasons, DTS cannot guarantee that its conflict detection mechanism can completely prevent data conflicts. You must refactor certain business logic to ensure that records of the same primary key, business primary key, or unique key are updated only on one of the instances that are two-way synchronized.

Supported conflict types

Currently, DTS supports detecting the following conflict types:

- Uniqueness conflicts caused by INSERT operations

  A uniqueness conflict occurs when the synchronization of an inserted row violates the unique constraint. For example, if two instances in two-way synchronization insert a record with the same primary key value at almost the same time, one of the inserted records fails to be synchronized because a record with the same primary key value already exists in the destination instance.

- Inconsistent records caused by UPDATE operations

  Update conflicts occur in the following scenarios:

  - The records to be updated do not exist in the destination instance. If the records to be updated do not exist, DTS automatically changes the UPDATE operation to the INSERT operation and inserts these records to the destination instance. In this case, duplicate unique key values may occur.
  - The primary keys or unique keys of the records to be updated conflict with each other.

- A DELETE operation is made on non-existent records

  A delete conflict occurs when the records to be deleted do not exist in the destination instance.

  In this case, DTS automatically ignores the DELETE operation regardless of the conflict resolution policy that you have configured.

Supported conflict resolution policies

For the preceding synchronization conflicts, DTS provides the following resolution policies. You can select a conflict resolution policy as required when configuring two-way synchronization.

- TaskFailed: The synchronization task reports an error and automatically exits the process in case of a conflict.

  When the synchronization encounters a conflict of the preceding types, the synchronization task reports an error and automatically exits the process. The task enters a failed state and you must manually resolve the conflict. This method is the default conflict resolution policy.

- Ignore: The records in the destination instance are used in case of a conflict.

  When the synchronization encounters a conflict of the preceding types, the synchronization task skips the current synchronization statement and continues the process. The records in the destination instance are used.

- Overwrite: The conflict records in the destination instance are overwritten in case of a conflict.

  When the synchronization encounters a conflict of the preceding types, the conflict records in the destination instance are overwritten.

4.4.11.4. Synchronization restrictions

This section describes the restrictions in cross-cloud data synchronization using DTS.

Restrictions in data sources

Currently, only ApsaraDB RDS for MySQL instances support two-way synchronization. Other heterogeneous data sources do not support two-way synchronization.

The destination instance cannot be an RDS instance that runs in standard access mode and has only a public network address.

Restrictions in synchronization architecture

Currently, DTS only supports two-way synchronization between two ApsaraDB RDS for MySQL instances. Two-way synchronization between more than two instances is not supported.

Feature restrictions

- Incompatible with triggers

  When you synchronize an entire database and the database contains a trigger that updates the synchronization table, the synchronized data may be inconsistent.

  For example, the object to be synchronized is database A that contains table a and table b. Table a has a trigger that inserts a row to table b after the row is inserted to table a. In this case, if an INSERT operation is performed on table a in the source instance during synchronization, the data in table b is inconsistent between the source and destination instances.

  To resolve this problem, you must delete the trigger in the destination instance, so that the data in table b is only synchronized from the source instance.

- Restrictions in the RENAME TABLE operation

The RENAME TABLE operation may result in inconsistent synchronization data. For example, if the object to be synchronized only includes table a and the **rename a to b** command is executed in the source instance during synchronization, subsequent operations to the renamed table b are not synchronized to the destination database. To solve this problem, you can synchronize the entire database where table a and table b are stored.

- Restrictions in DDL synchronization direction

  To ensure the stability of a two-way synchronization channel, you can synchronize DDL updates on the same table in only one direction. For example, in A-to-B and B-to-A synchronization, you can implement DDL synchronization in either the **A-to-B** or **B-to-A** direction. If DDL synchronization is configured in one direction, it is not supported in the reverse direction.

4.4.11.5. Configure two-way data synchronization between ApsaraDB RDS for MySQL instances across regions

This topic describes how to configure two-way data synchronization between ApsaraDB RDS for MySQL instances across regions.

Prerequisites

The source and destination ApsaraDB RDS for MySQL instances are created.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. On the **Synchronization Tasks** page, click **Create Synchronization Task** in the upper-right corner.

   > **Note**
   >
   > Source Instance Region: Select the region where the source RDS instance resides.
   >
   > Source Instance Type: Select the type of the source instance. In this example, select MySQL.
   >
   > Destination Instance Region: Select the region where the destination RDS instance resides.
   >
   > Destination Instance Type: Select the type of the destination instance. In this example, select MySQL.
   >
   > Synchronization Mode: Select the synchronization mode. In this example, select Two-Way Synchronization.
   >
   > Instances to Create: Set the number of instances that you want to create.

4. After you configure the preceding information, click **Create**.

   After you create a synchronization instance, go back to the **Synchronization Tasks** page. The new synchronization instance is in the Not Configured state and contains two synchronization tasks. You can configure two-way synchronization for the tasks.

5. Find one of the created synchronization tasks and click **Configure Synchronization Channel** in

the Actions column.

6. Configure the parameters for the data synchronization task.
   - Synchronization Task Name

     We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
   - RDS instance ID

     You must specify the ID of the Apsara Stack tenant account to which the destination RDS instance belongs. You can then select an RDS instance ID from the Instance ID drop-down list.

     After you complete the preceding configurations, click **Set Whitelist and Next** to configure the RDS instance whitelists.
7. Configure the RDS instance whitelists.

   In this step, DTS adds the CIDR blocks of DTS servers to the whitelists of the source and destination RDS instances. This ensures that DTS servers can connect to the RDS instances and the data synchronization task can be created.

   We recommend that you do not remove the CIDR blocks of DTS servers from the whitelists of the RDS instances. This ensures the stability of the data synchronization task.

   Click **Next** to create a data synchronization account.
8. Create a data synchronization account in the destination RDS instance.

   Create a data synchronization account named dtssyncwriter in the destination RDS instance. Do not delete the account during data synchronization. Otherwise, an interruption occurs.
9. Configure the synchronization policies and select the objects that you want to synchronize.

   After you create a data synchronization account, you must configure the synchronization policies and select the objects that you want to synchronize.
   - Exclude DDL Statements

     Specify whether to synchronize DDL statements in a specific direction. To include DDL statements, select **No**. To exclude DDL statements, select **Yes**. If you select No, DTS does not synchronize the DDL operations that are performed on a table in the opposite direction.
   - DML Statements for Synchronization

     Select the types of DML operations that you want to synchronize. By default, **Insert**, **Update**, and **Delete** are selected.
   - Conflict Resolution Policy

     Select the resolution policy for synchronization conflicts. By default, **TaskFailed** is selected.

     For example, if Node A is the primary business center and Node B is a secondary business center, you must give the priority to Node A. You must set the conflict resolution policy in the A-to-B direction to **Overwrite** and that in the B-to-A direction to **Ignore**.
   - Select Objects

     You can select databases and tables as the objects to be synchronized.

     If you select an entire database, all schema changes such as the CREATE TABLE and DROP VIEW operations that performed on the objects in the database are synchronized to the destination database.

If you select a table, only the DROP TABLE, ALTER TABLE, TRUNCATE TABLE, RENAME TABLE, CREATE INDEX, and DROP INDEX operations that performed on this table are synchronized to the destination database.

10. Configure initial synchronization.

    Initial synchronization is the first step to start the synchronization task. During initial synchronization, DTS synchronizes the schemas and data of the required objects from the source instance to the destination instance. The schemas and data are the basis for subsequent incremental synchronization.

    Initial synchronization includes **initial schema synchronization** and **initial full data synchronization**. You must select both **Initial Schema Synchronization** and **Initial Full Data Synchronization** in most cases.

    If the tables to be synchronized in one direction are also included in the objects to be synchronized in the opposite direction, DTS does not synchronize these tables during initial synchronization.

11. Run a precheck.

    After the data synchronization task is configured, DTS performs a precheck. Close the **Precheck** dialog box after the task passes the precheck.

    After the task is started, the task list appears. The task is in the **Performing Initial Sync** state. The duration of the initial synchronization depends on the data volume of the objects that you want to synchronize. After initial synchronization, the task status changes to **Synchronizing**. This indicates that the data synchronization task is created.

    After the task is configured in one direction, the source and destination RDS instances of the task in the opposite direction cannot be changed.

12. Repeat steps 5 to 11 to configure the data synchronization task in the opposite direction.

4.4.12. Synchronize data between PolarDB-X 2.0 instances

This topic describes how to configure one-way synchronization between PolarDB-X 2.0 instances in the Data Transmission Service (DTS) console.

Prerequisites

- The source PolarDB-X 2.0 instance and the destination PolarDB-X 2.0 instance that are compatible with MySQL 5.7 are created.
- The available storage capacity of the destination PolarDB-X 2.0 instance is larger than the total size of data stored in the source PolarDB-X 2.0 instance.

Limits

Category	Description

Category	Description

Limits on the source database	• The tables to be synchronized must have PRIMARY KEY or UNIQUE constraints and all fields must be unique. Otherwise, the destination database may contain duplicate data records.
	• If you select tables as the objects to be synchronized and you want to edit the tables (such as renaming tables or columns) in the destination database, up to 1,000 tables can be synchronized in a single data synchronization task. If you run a task to synchronize more than 1,000 tables, a request error occurs. In this case, we recommend that you configure multiple tasks to synchronize the tables in batches or configure a task to synchronize the entire database.
	• The following requirements for binary logs must be met:
	  ◦ Binary logging is enabled on the Calculation layer and Storage layer tabs in the PolarDB-X console. For more information, see the **"Parameter settings"** topic of *PolarDB-X User Guide*. The value of the binlog_row_image parameter must be set to full. Otherwise, error messages are returned during precheck and the data synchronization task cannot be started.
	  ◦ For an incremental data synchronization task, the binary logs of the source database are retained for at least 24 hours. For a full data and incremental data synchronization task, the binary logs of the source database are retained for at least seven days. Otherwise, DTS may fail to obtain the binary logs and the task may fail. In exceptional circumstances, data inconsistency or loss may occur. After the full data synchronization is complete, you can set the retention period to more than 24 hours. Make sure that you set the retention period of binary logs in accordance with the preceding requirements. Otherwise, the service reliability and performance stated in the Service Level Agreement (SLA) of DTS may not be achieved.
	• The source PolarDB-X 2.0 instance must be compatible with MySQL 5.7.

Other limits	• The destination PolarDB-X 2.0 instance must be compatible with MySQL 5.7. • Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours. During initial full data synchronization, DTS uses read and write resources of the source and destination databases. This may increase the loads on the database servers. • During initial full data synchronization, concurrent INSERT operations cause fragmentation in the tables of the destination database. Therefore, after initial full data synchronization is complete, the size of the used tablespace of the destination database is larger than that of the source database. • We recommend that you do not use gh-ost or pt-online-schema-change to perform DDL operations on objects. Otherwise, data synchronization may fail.

SQL operations that can be synchronized

Type	SQL statement
DML	INSERT, UPDATE, and DELETE

Permissions required for database accounts

Database	Schema synchronization	Full data synchronization	Incremental data synchronization
Source **PolarDB-X 2.0** Instance	The SELECT permission	The SELECT permission	The SELECT permission on the objects to be synchronized, the REPLICATION CLIENT permission, and the REPLICATION SLAVE permission
Destination **PolarDB-X 2.0** Instance	Read and write permissions on the objects to be synchronized		

Step 1: Create a data synchronization task

1. Log on to the DTS console.
2. In the top navigation bar, click **DTS**. In the left-side navigation pane, click **Data Synchronization**.
3. On the **Data Synchronization** page, click **Create Task**.
4. In the panel that appears, set the following parameters.

Parameter	Description
Source Database Type	Select **PolarDB-X 2.0**.
Source Region	The region in which the source instance resides.
Destination Database Type	Select **PolarDB-X 2.0**.
Destination Region	The region in which the destination instance resides.
Quantity	The number of data synchronization instances that you want to create at a time. The default value is 0.

5. Click **Purchase Instance**.

Step 2: Configure the data synchronization task

1. In the **Data Synchronization Tasks** list, find the data synchronization task that you create and click **Configure Task** in the rightmost column.
2. Configure the source and the destination database. The following table describes the parameters.

Section	Parameter	Description
N/A	**Task Name**	DTS automatically generates a task name. We recommend that you specify an informative name to identify the task. You do not need to use a unique task name.
	Database Type	Select **PolarDB-X 2.0**.

Section	Parameter	Description
Source Database	Access Method	The value of this parameter is fixed to **Alibaba Cloud Instance**.
	Instance Region	The region in which the source **PolarDB-X 2.0** instance resides.
	Instance ID	The ID of the source **PolarDB-X 2.0** instance.
	Database Account	The database account of the source **PolarDB-X 2.0** instance. The account must have the SELECT permission on the objects to be synchronized, the REPLICATION CLIENT permission, and the REPLICATION SLAVE permission.
	Database Password	The password of the database account.
Destination Database	Database Type	Select **PolarDB-X 2.0**.
	Access Method	The value of this parameter is fixed to **Alibaba Cloud Instance**.
	Instance Region	The region in which the destination **PolarDB-X 2.0** instance resides.
	Instance ID	The ID of the destination **PolarDB-X 2.0** instance.
	Database Account	The database account of the destination **PolarDB-X 2.0** instance. The account must have the read and write permissions on the destination database.
	Database Password	The password of the database account.

3. In the lower part of the page, click **Test Connectivity and Proceed**.
4. Select objects for the task and configure advanced settings.
   - **Basic Settings**

Parameter	Description
Task Stages	**Incremental Data Synchronization** is selected by default. You must also select **Schema Synchronization** and **Full Data Synchronization**. After the precheck, DTS synchronizes the historical data of the selected objects from the source instance to the destination cluster. The historical data is the basis for subsequent incremental synchronization.

Parameter	Description
Processing Mode of Conflicting Tables	■ **Precheck and Report Errors**: checks whether the source and destination databases contain tables that share the same names. If the source and destination databases do not contain identical table names, the precheck is passed. Otherwise, an error is returned during the precheck, and the data synchronization task cannot be started. **Note** You can use the object name mapping feature to rename the tables that are synchronized to the destination database. You can use this feature if the source and destination databases contain identical table names and the tables in the destination database cannot be deleted or renamed. For more information, see Object name mapping. ■ **Ignore Errors and Proceed**: skips the precheck for identical table names in the source and destination databases. **Warning** If you select **Ignore Errors and Proceed**, data consistency is not ensured, and your business may be exposed to potential risks. ■ If the source and destination databases have the same schema, and a data record has the same primary key as an existing data record in the destination database: ■ During full data synchronization, DTS does not synchronize the data record to the destination database. The existing data record in the destination database is retained. ■ During incremental data synchronization, DTS synchronizes the data record to the destination database. The existing data record in the destination database is overwritten. ■ If the source and destination databases have different schemas, data may fail to be initialized. In this case, only some columns are synchronized, or the data synchronization task fails.
Select Objects	Select one or more objects from the **Source Objects** section and click the › icon to add the objects to the **Selected Objects** section. **Note** You can select columns, tables, or databases as the objects to be synchronized. If you select tables or columns as the objects to be synchronized, DTS does not synchronize other objects such as views, triggers, and stored procedures to the destination database.

- **Advanced Settings**

Parameter	Description
Set Alerts	Specify whether to set alerts for the data synchronization task. If the task fails or the synchronization latency exceeds the threshold, the alert contacts will receive notifications.
Retry Time for Failed Connection	Specify the retry time range for failed connections. Valid values: 10 to 1440. Unit: minutes. Default value: 120. We recommend that you set the retry time range to more than 30 minutes. If DTS reconnects to the source and destination databases within the specified time range, DTS resumes the data synchronization task. Otherwise, the data synchronization task fails. **Note** If multiple DTS instances have the same source or destination database, the lowest value of this parameter takes effect. For example, the retry time range is set to 30 minutes for Instance A and 60 minutes for Instance B, DTS retries failed connections within 30 minutes.

5. Click **Next: Save Task Settings and Precheck** in the lower part of the page.

Note

- Before you can start the data synchronization task, DTS performs a precheck. You can start the data synchronization task only after the task passes the precheck.
- If the task fails to pass the precheck, you can click the icon next to each failed item to view details.
  - You can troubleshoot the issues based on the causes and run a precheck again.
  - If you do not need to troubleshoot the issues, you can ignore failed items and run a precheck again.

After the value of **Success Rate** becomes **100%**, the data synchronization task starts to run. You can view the progress of the task on the Data Synchronization page.

4.5. Manage data synchronization instances

4.5.1. Specify the name of an object in the destination instance

After an object, such as a database or table, is synchronized from the source instance to the destination instance, the name of the object remains unchanged. You can use the object name mapping feature provided by DTS to specify a different name for the object in the destination instance.

Notes

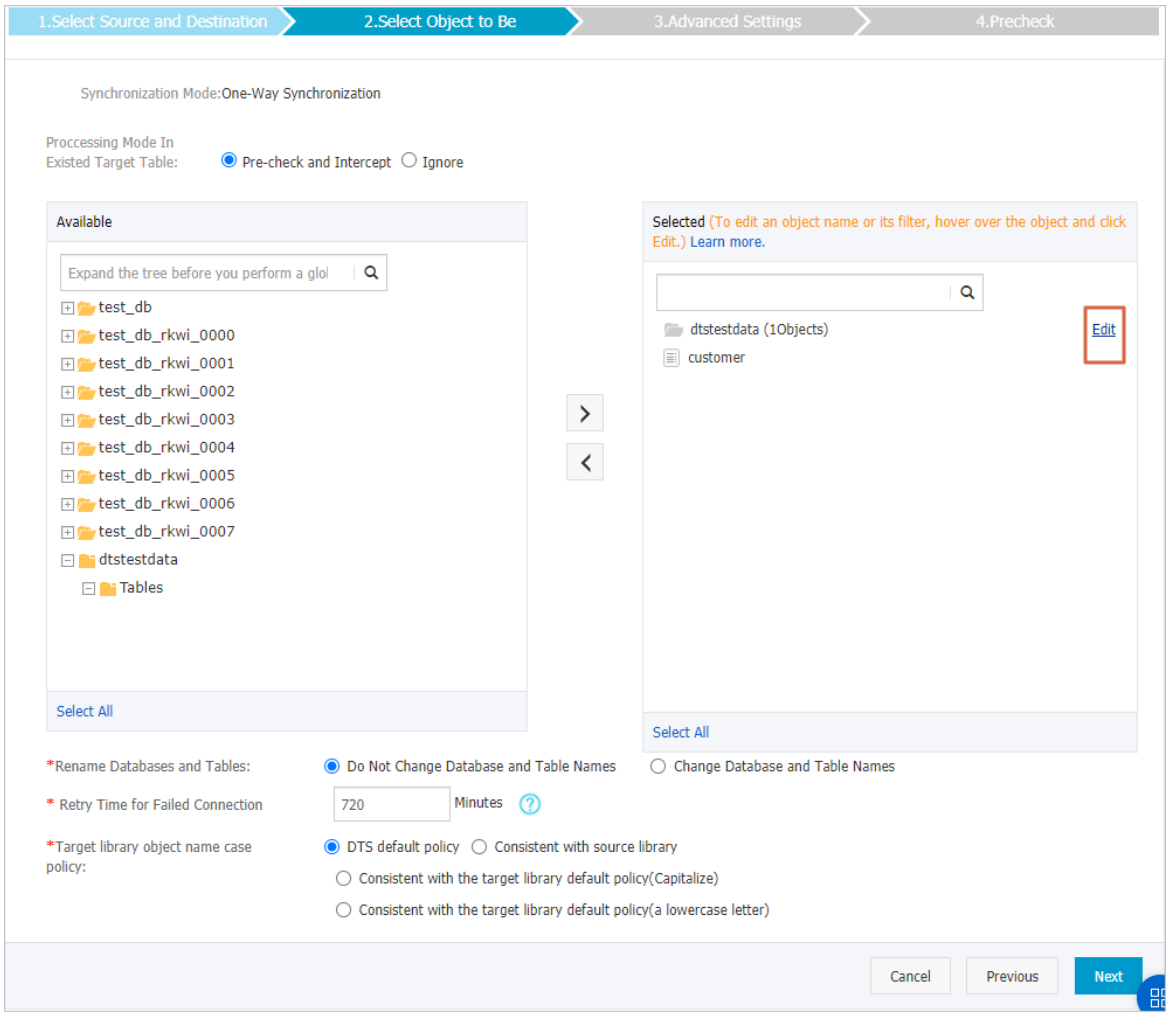
You can perform this operation only when a data synchronization task is configured and the current process is **Select Objects to Synchronize**.

> **Note** Do not perform this operation after the data synchronization task is started. Otherwise, the synchronization may fail.

Procedure

1. On the **Select Objects to Synchronize** page, move the required objects to the **Selected** section, move the pointer over a database or table, and then click **Edit**.

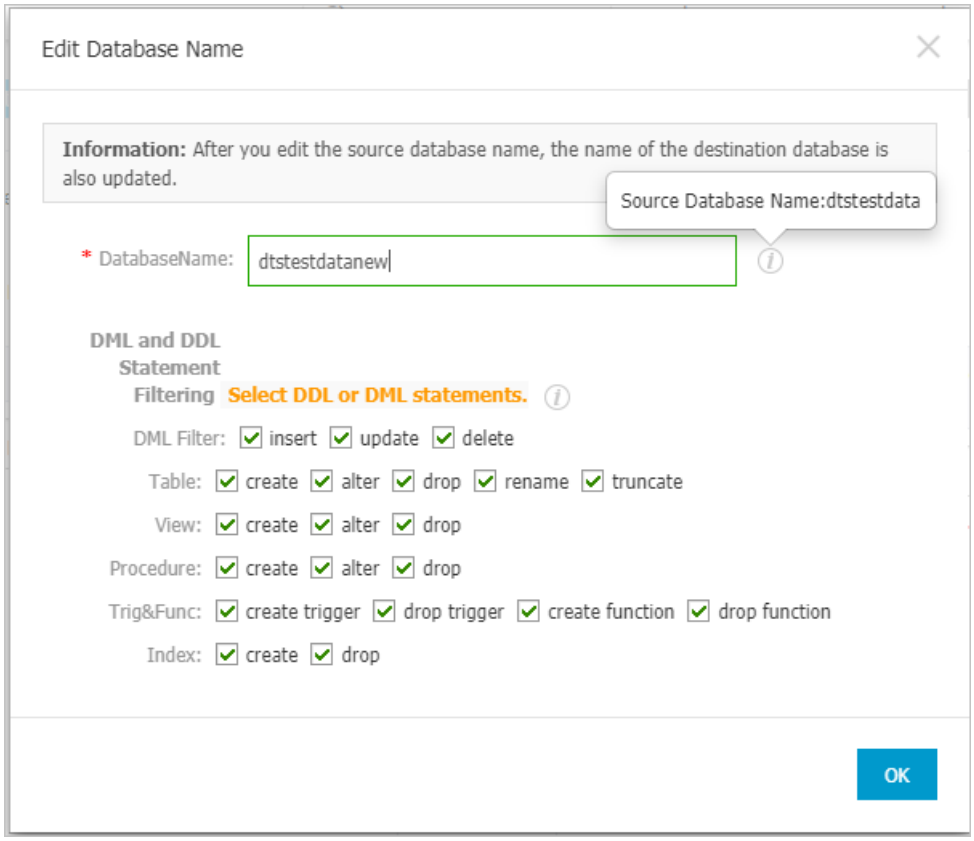
   > **Note** Different database types support different objects. If **Edit** appears when you move the pointer over the target object, the operation is supported.



2. In the dialog box that appears, specify a name for the object in the destination instance.

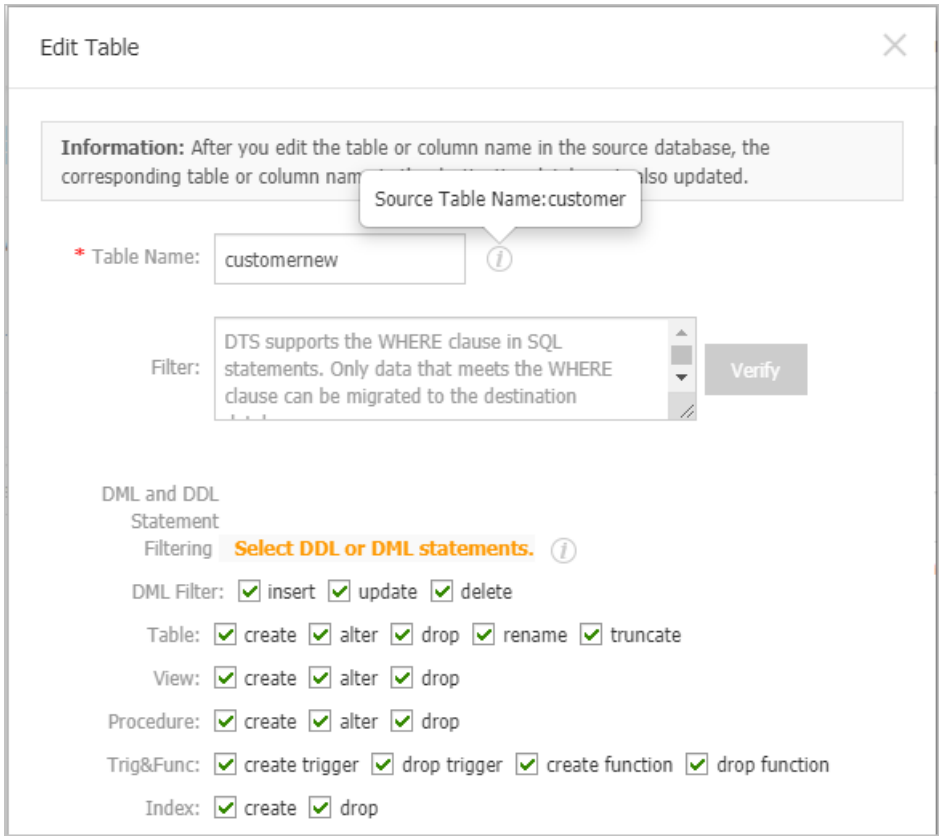
- Database name mapping

  In the **Edit Database Name** dialog box that appears, enter the database name that you want to use in the destination instance.



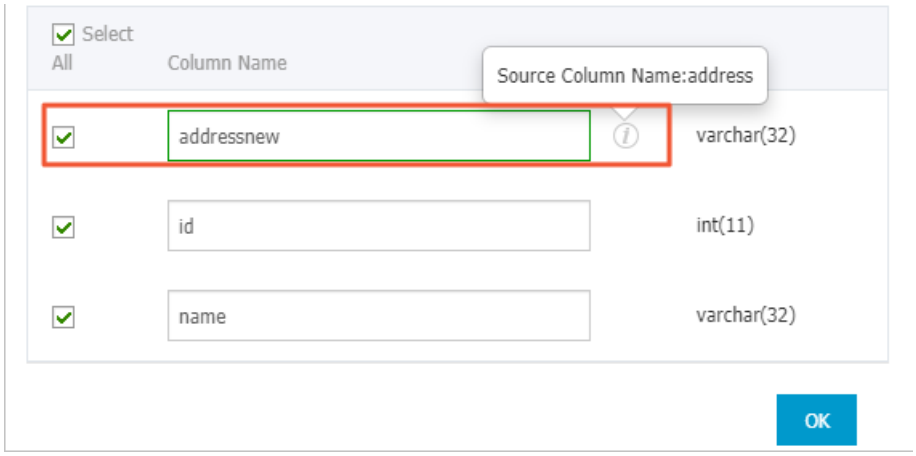
- Table name mapping

  In the **Edit Table** dialog box that appears, enter the table name that you want to use in the destination instance.



- Column name mapping

  In the **Edit Table** dialog box that appears, enter a new name for each column.



> **Note** In this step, you can deselect columns that do not need to be synchronized.

3. Click **OK**.
4. Configure other parameters that are required for the data synchronization task.

4.5.2. Use SQL conditions to filter data

When you select the objects to be synchronized in a data synchronization task, you can specify SQL conditions to filter data. Only the data that meets the specified conditions is synchronized to the destination database. This feature is applicable to scenarios such as regular data synchronization and table partitioning.

Prerequisites

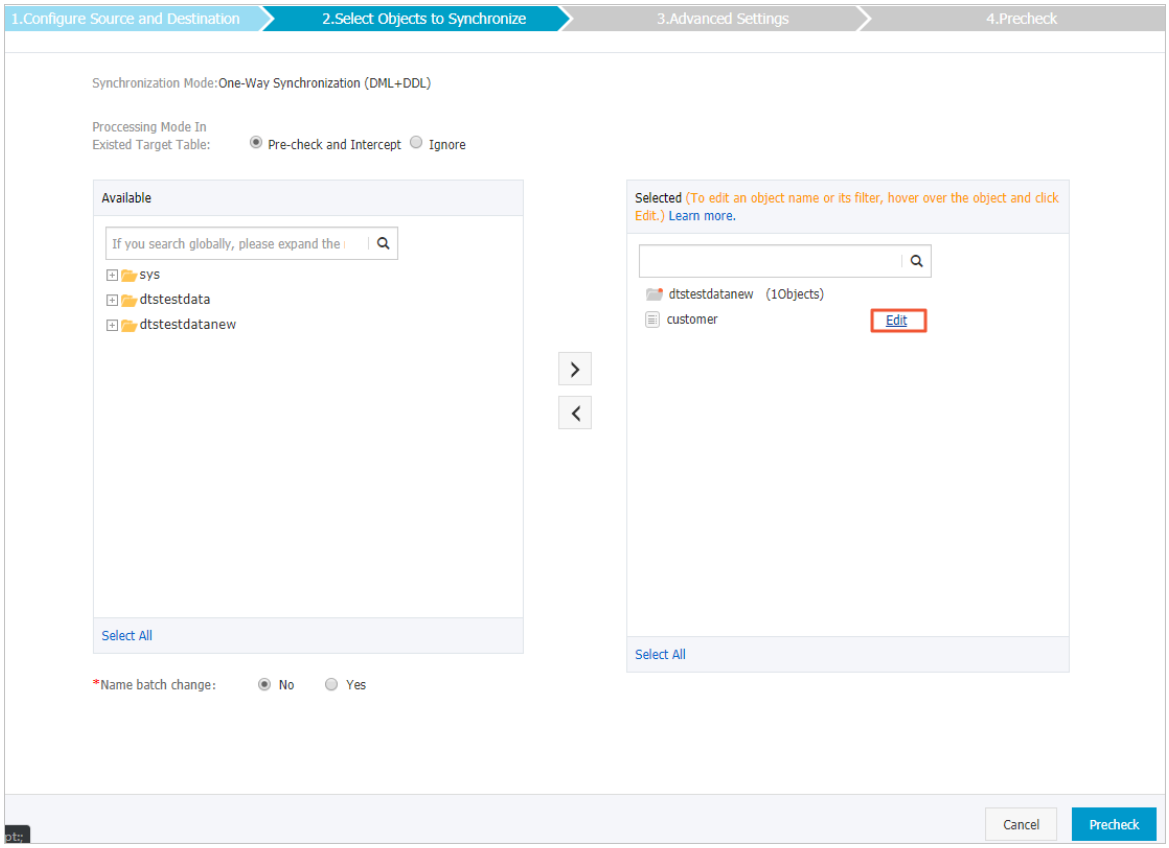
A data synchronization task is being configured. The current step is **Select Objects to Synchronize**. For more information about how to create and configure a data synchronization task, see configuration examples in Data Transmission Service (DTS) documentation.

Limits

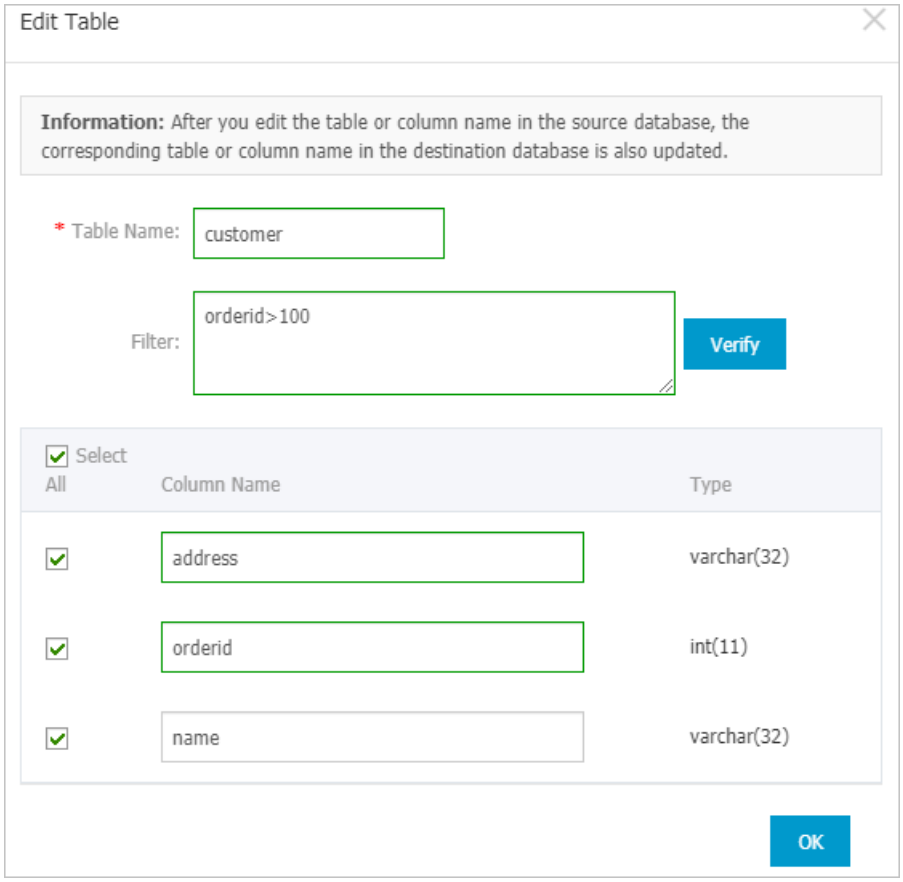
- You can filter the fields only in the current table. Cross-table filtering is not supported.
- If the source database is a self-managed PostgreSQL database or an ApsaraDB RDS for PostgreSQL instance, filter conditions do not take effect on incremental data.

Procedure

1. In the **Select Objects to Synchronize** step, move the required objects to the **Selected** section, move the pointer over a table, and then click **Edit**.



2. In the **Edit Table Name** dialog box, enter one or more SQL conditions in the **Filter Conditions** field.



> **Note**
>
> - An SQL condition is a standard SQL WHERE statement. The following operators are supported: `=`, `!=`, `<`, `>`, and `in`. Only the data that meets the WHERE condition is synchronized to the destination database. In this example, `orderid>100` is entered.
> - You can specify a time condition in an SQL WHERE statement. However, you must make sure that the specified time condition is valid. For example, to filter incremental data created since 2020, you must enter `create_time>'2020-01-01'` or `create_time>'2020-01-01 00:00:00'`. You cannot enter `create_time>'2020'`.
> - You can use single quotation marks (') in an SQL condition if necessary. For example, you can enter `address in('hangzhou','shanghai')`.

3. Click **Verify** to check whether the syntax is valid.

> **Note**
>
> - If the syntax is valid, the **Information** message shows that **the validation is passed**.
> - If the syntax is invalid, the **Error** message appears. In this case, you must modify the condition based on the instructions.

4. Click **OK**.
5. Configure other parameters that are required for the data synchronization task.

4.5.3. Check the synchronization performance

DTS provides the trend charts of data synchronization tasks based on three performance metrics: bandwidth, synchronization speed (TPS), and synchronization delay. You can view the running status of data synchronization tasks in the DTS console.

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. On the Synchronization Tasks page, click the ID of the data synchronization task that you want to check.

   The task details page appears.
4. On the task details page, click **Synchronization Performance** in the left-side navigation pane.
5. View the trend charts of synchronization performance.

   DTS provides the trend charts of data synchronization tasks based on three performance metrics: bandwidth, synchronization speed (TPS), and synchronization delay.

   - Bandwidth: the bandwidth of data that the data writing module pulls from the data pulling module per second. Unit: MB/s.
   - Synchronization speed (TPS): the number of transactions that DTS synchronizes to the destination instance per second.
   - Synchronization delay: the difference between the timestamp of the latest synchronized data in the destination instance and the current timestamp in the source instance. Unit: milliseconds.

4.5.4. Add objects to a data synchronization task

When a data synchronization task is running, you can add objects to the task or remove objects from the task. This topic describes how to add objects to a data synchronization task in the DTS console.

Limits

You can modify the required objects only when the data synchronization task is in the **Synchronizing** or **Synchronization Failed** state.

Start time of data synchronization

The time when DTS synchronizes data of new objects depends on whether initial synchronization is specified for the data synchronization task.

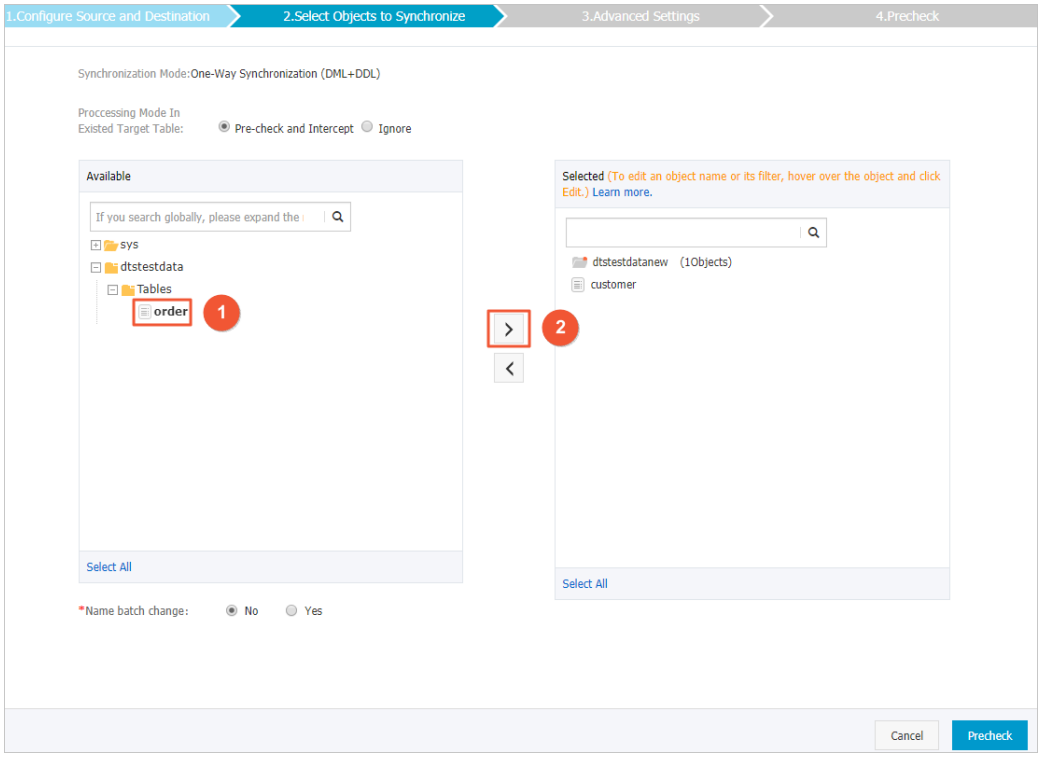
- If initial synchronization is specified, DTS synchronizes schemas and historical data, and then synchronizes incremental data.
- If initial synchronization is not specified, DTS synchronizes data after incremental data is generated on the source instance.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. Find the data synchronization task and choose **More > Modify Objects to Synchronize** in the Actions column.

4. On the **Select Objects to Synchronize** tab, add objects based on your needs, as shown in Add objects to a data synchronization task.

   Add objects to a data synchronization task



5. Click **Precheck**.

   After the task passes the precheck, the objects are added to the data synchronization task.

   After the objects are added, if initial synchronization is specified for the data synchronization task, the task status changes from **Synchronizing** to **Synchronizing (The initial synchronization of the new objects is being performed.)**.

> **Note** You can click **View More** to view the initial synchronization progress of the new objects. After the initial synchronization on the new objects is complete, the task status returns to **Synchronizing**.

4.5.5. Remove objects from a data synchronization task

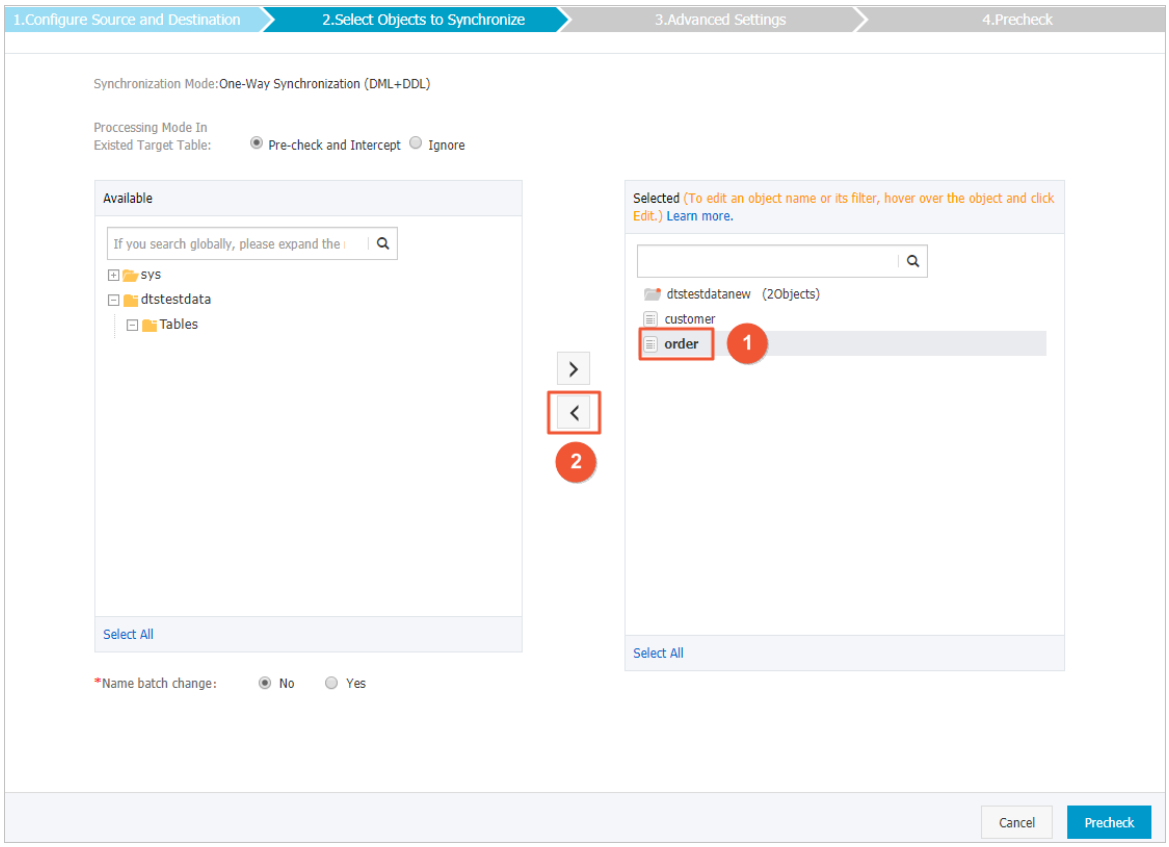
When a data synchronization task is running, you can add objects to the task or remove objects from the task. This topic describes how to remove objects from a data synchronization task in the DTS console.

Limits

You can modify the required objects only when the data synchronization task is in the **Synchronizing** or **Synchronization Failed** state.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Synchronization**.
3. Find the data synchronization task and choose **More > Modify Objects to Synchronize** in the Actions column.
4. On the **Select Objects to Synchronize** tab, remove objects based on your needs.



5. Click **Precheck** to run a precheck.

4.5.6. Troubleshoot precheck failures

Before Data Transmission Service (DTS) runs a data synchronization task, DTS performs a precheck. This topic describes the precheck items and how to troubleshoot precheck failures.

Source database connectivity

- Description

  DTS checks whether DTS servers can connect to the source RDS instance. DTS creates a connection to the source RDS instance by using the JDBC protocol. If the connection fails, the task fails to pass the precheck.

- Cause of failure
  - DTS does not support data synchronization between RDS instances in the region where the source instance resides.
  - The database account or password of the source instance is invalid.

- Solution

Submit a ticket and contact Alibaba Cloud technical support.

Destination database connectivity

- Description

  DTS checks whether DTS servers can connect to the destination RDS instance. DTS creates a connection to the destination RDS instance by using the JDBC protocol. If the connection fails, the task fails to pass the precheck.

- Cause of failure
  - DTS does not support data synchronization between RDS instances in the region where the destination instance resides.
  - The database account or password of the destination instance is invalid.
- Solution

  Submit a ticket and contact Alibaba Cloud technical support.

Source database version

- Description

  DTS checks whether:

  i. The database version of the source RDS instance is supported by the data synchronization feature.

  ii. The database version of the destination RDS instance is the same as the database version of the source RDS instance.

- Cause of failure
  - The database version of the source RDS instance is earlier than the supported database versions. The data synchronization feature supports the following database versions: MySQL 5.1, 5.5, 5.6, and 5.7.
  - The database version of the destination RDS instance is earlier than the database version of the source RDS instance.
- Solution
  - If the database version of the source RDS instance is earlier than the supported database versions, upgrade the source RDS instance to MySQL 5.6 or 5.7 in the RDS console. Then, create a data synchronization task again.
  - If the database version of the destination RDS instance is earlier than the database version of the source RDS instance, upgrade the destination RDS instance to MySQL 5.6 or 5.7 in the RDS console. Then, create a data synchronization task again.

Database existence

DTS checks whether the destination database already exists in the destination instance. If the destination database does not exist in the destination instance, DTS automatically creates a database. However, DTS fails to create the database and reports a failure under the following circumstances:

- The database name contains characters other than lowercase letters, digits, underscores (_), and hyphens (-).
- The character set of the database is not UTF-8, GBK, Latin1, or UTF-8MB4.
- The account of the destination database does not have the read/write permissions on the source

database.

If the data source is an RDS instance, the task passes the precheck.

Source database permissions

DTS checks whether the account of the source database has the required permissions. If the account does not have the required permissions, the task fails to pass the precheck. If the source database is an RDS instance, the task passes the precheck.

Destination database permissions

- Description

  DTS checks whether the account of the destination database has the required permissions. If the account does not have the required permissions, the task fails to pass the precheck.

- Cause of failure
  - DTS fails to create a database account in the destination RDS instance.
  - DTS fails to grant the read/write permissions to the database account of the destination RDS instance.

- Solution

  Submit a ticket and contact Alibaba Cloud technical support.

Object name conflict

- Description

  DTS checks object names only if you select initial synchronization for a data synchronization task. DTS checks whether an object that you want to synchronize has the same name as an object in the destination RDS instance.

- Cause of failure

  If an object in the destination RDS instance has the same name as the object that you want to synchronize, the task fails to pass the precheck.

- Solution
  - Remove the conflicting object from the destination database.
  - Then, create a data synchronization task again. Select both Initial Schema Synchronization and Initial Full Data Synchronization.

Value of server_id in the source database

DTS checks whether the value of the server_id parameter in the source database is set to an integer that is greater than or equal to 2. If the data source is an RDS instance, the task passes the precheck.

Whether binary logging is enabled for the source database

DTS checks whether the binary logging feature is enabled for the source database. If the binary logging feature is disabled for the source database, the task fails to pass the precheck. If the data source is an RDS instance, the task passes the precheck.

Binary log format of the source database

DTS checks whether the binary log format of the source database is set to ROW. If the binary log format of the source database is not set to ROW, the task fails to pass the precheck. If the data source is an RDS instance, the task passes the precheck.

Integrity of the FOREIGN KEY constraints

- Description

  DTS checks whether the parent tables and child tables that have referential relationships with each other are all included in the required objects. The precheck allows DTS to protect the integrity of the FOREIGN KEY constraints.

- Cause of failure

  One or more child tables are included in the required objects. However, the parent tables that are referenced by the child tables are not included in the required objects. This impairs the integrity of the FOREIGN KEY constraints.

- Solution

  The following solutions are available:

  - Create a data synchronization task again and do not synchronize the child tables that fail to pass the precheck.
  - Create a data synchronization task again and add the parent tables to the required objects.
  - Remove the FOREIGN KEY constraints from the child tables that fail to pass the precheck. Then, create a data synchronization task again.

Storage engine

- Description

  DTS checks whether the required objects use the storage engines that are not supported by the data synchronization feature, such as FEDERATED, MRG_MyISAM, and TokuDB.

- Cause of failure

  If the storage engine of a source table is FEDERATED, MRG_MyISAM, or TokuDB, the task fails to pass the precheck.

- Solution

  Change the unsupported storage engine to InnoDB and create a data synchronization task again.

Character set

- Description

  DTS checks whether the required objects use the character sets that are not supported by the data synchronization feature, such as the UCS-2 character set.

- Cause of failure

  If the character sets used by the required objects are not supported by the data synchronization feature, the task fails to pass the precheck.

- Solution

  Change the unsupported character sets to UTF-8, GBK, or Latin1. Then, create a data synchronization task again.

Complicated topologies

- Description

  DTS checks whether the topology that you specify for the source and destination RDS instances is supported.

- Cause of failure
  - The source RDS instance in the current task is being used as the destination instance of another task.
  - The destination RDS instance in the current task is being used as the source or destination instance of another task.
  - The objects that you want to synchronize in the current task are being synchronized by an existing task. The two tasks have the same source and destination RDS instances.

- Solution
  - If the task that you want to create has the same source and destination RDS instances as an existing task, you can add the required objects to the existing task. You do not need to create another task to synchronize these objects.
  - If the task that you want to create conflicts with an existing task, wait until the existing task is completed before you create a data synchronization task again.

Format of the MySQL database password

DTS checks whether the format of the password that is used to access the source database is no longer valid. If the data source is an RDS instance, the task passes the precheck.

4.6. Best practices for data synchronization from PolarDB-X instances

4.6.1. Limits and usage notes for synchronizing data between PolarDB-X instances

PolarDB-X does not provide binary logs. To ensure data quality, we recommend that you take note of the limits when you perform business design, business development, and O&M changes.

Overview

- Limits on the business design
- Limits on the database architecture
- Limits on O&M changes
- Potential risks on data quality
- Suggestions to ensure data quality

Limits on the business design

- All tables must have primary keys. Otherwise, data inconsistency may occur in data synchronization because the destination database may contain duplicate data records.
- We recommend that you do not use the global secondary indexes (GSIs) of PolarDB-X because they are updated asynchronously. If you use GSIs, DTS can ensure only eventual consistency of data.
- The databases that you want to synchronize cannot be deployed in mixed mode. The mixed mode refers to a mode in which both the unit mode and the copy mode are used).

  > **Note** In unit mode, users perform read and write operations in their respective unit nodes. Two-way synchronization is implemented between the databases in each unit node and those in the central node. In copy mode, users write data to the databases in the central node. The data is then synchronized to the databases in each unit node.

- If you configure two-way synchronization between PolarDB-X instances and use underlying MySQL databases, fields of the FLOAT or DOUBLE type are not supported in business tables. If the business tables have fields of the FLOAT or DOUBLE type, convert the field type to DECIMAL. If you configure one-way synchronization between PolarDB-X instances, fields of the FLOAT or DOUBLE type are supported in business tables.
- DTS does not allow you to synchronize objects such as stored procedures, triggers, functions, views, and events in data synchronization between PolarDB-X instances.
- DTS does not support initial schema synchronization between PolarDB-X instances. You must manually create objects such as databases and tables in the destination database.
- Make sure that the available storage capacities in PolarDB-X instances are sufficient for business expansion.
- If the version of the MySQL databases attached to a PolarDB-X instance is 5.7 or 8.0, you cannot configure a change tracking task for the PolarDB-X instance. Instead, you must configure a change tracking task for each MySQL database attached to the PolarDB-X instance.

Limits on the database architecture

- An ApsaraDB RDS for MySQL instance that is attached to a PolarDB-X instance cannot be attached to another PolarDB-X instance.
- The ApsaraDB RDS for MySQL instances attached to the source PolarDB-X instance must have the same specifications as those attached to the destination PolarDB-X instance. For example, if the source PolarDB-X instance has four ApsaraDB RDS for MySQL instances, the destination PolarDB-X instance must have four ApsaraDB RDS for MySQL instances whose specifications are the same as those in the source PolarDB-X instance.
- The sharding rule of the source PolarDB-X instance must be the same as that of the destination PolarDB-X instance. Otherwise, no data synchronization task can be created.
- You can synchronize only business tables between Polar-X instances. You cannot synchronize metadata tables or system tables between PolarDB-X instances.

Limits on O&M changes

Change type	Description	Impact and solution

Change type	Description	Impact and solution
PolarDB-X instances	Change a sharding rule. For example, change the shard key of a database or table, or change the number of shards.	Not supported. You must perform the following steps to recreate a data synchronization task: 1. Stop and delete the original data synchronization task. 2. Wait until the changes are complete in the source database. Then, clear the data that has been synchronized to the destination database. 3. Configure a data synchronization task for each ApsaraDB RDS for MySQL instance that is attached to the source PolarDB-X instance.
	Change the number of instances at the storage layer. For example, scale out instances or migrate frequently-accessed tables.	
Storage	Change the specifications of instances and switch workloads at the storage layer.	Data synchronization is not affected.
	Change parameter settings.	The parameter settings of the source and destination databases must be the same. To change the parameter settings of instances at the storage layer, you must make sure that new parameters do not affect previous parameters. **Note** If you are not sure about the impact of changing parameter settings, you can contact technical support of Database Expert Service.
	Change backup and recovery policies, and enable auditing and diagnostics for instances at the storage layer.	The change takes effect only on the current instance and does not affect other instances with replication relationships.
DTS tasks	Perform DDL operations.	If you configure data synchronization tasks for ApsaraDB RDS for MySQL instances that are attached to the source PolarDB-X instance, the tasks may be delayed after you perform DDL operations. This is caused by the implementation logic of MySQL.

Change type	Description	Impact and solution
DDL operations at the database or table level	Add tables.	Not supported. You must perform the following steps: 1. After you perform DDL operations to create tables in the destination database, perform the same DDL operations to create tables in the source database. 2. Add the new tables to the objects of the data synchronization task. You can write data to the source database only after the preceding operations are complete. If you select tables as the objects of the data synchronization task, you must add the new tables to the synchronization queues on the source and destination instances at the storage layer.
	Add fields, add secondary indexes, delete indexes, and modify indexes (except for replacing secondary indexes with unique indexes).	• If you configure a data synchronization task for a PolarDB-X instance, you must perform DDL operations in the destination database, and then perform the same DDL operations in the source database. • If you configure data synchronization tasks for ApsaraDB RDS for MySQL instances that are attached to the source PolarDB-X instance, DTS automatically synchronizes the operations that you perform. The following operations can be synchronized: Add fields, add secondary indexes, delete indexes, and modify indexes (except for replacing secondary indexes with unique indexes).
	Perform other DDL operations.	Only the preceding DDL operations are supported.

Change type	Description	Impact and solution
Switchover **Note** Switchover: After you use DTS to migrate or synchronize data from the source database to the destination database, you switch workloads from the source database to the destination database.	Perform a switchover.	Before you perform a switchover, make sure that the DTS task is not delayed. Otherwise, data quality issues occur.
	Perform a failover that meets the requirements of recovery point objective (RPO). **Note** RPO represents the maximum amount of data that can be lost after a recovery from a failure. RPO is measured by time. **Warning** Failover: If the source instance or the data center where the source instance resides fails, you can switch workloads to a backup system. A failover is a lossy operation.	If a failure such as network interruption, an equipment failure, or a data center failure occurs and the data synchronization task is delayed, you may need to perform a failover. In this case, if the difference between the time when the last data entry is synchronized to the destination database and the time when the failure occurs is less than the RPO, you can perform a failover to recover your business. For example, if the RPO is 5 minutes, the quality of the data within the 5 minutes cannot be ensured after you perform a failover. You may need to revise the data to ensure consistency. **Note** To query the time when the last data entry is synchronized to the destination database, refer to the documentation about task management.
	Perform a failover that does not meet the requirements of RPO.	The data synchronization task may be delayed due to the following causes: A large number of DDL operations are performed in the source database, a network failure occurs, and the performance of the destination database is unfavorable. In this case, if the data center fails and the difference between the time when the last data entry is synchronized to the destination database and the time when the failure occurs is greater than the RPO, we recommend that you wait until the data center recovers before you perform a failover. For example, if the RPO is 5 minutes, the quality of the data within the 5 minutes cannot be ensured after you perform a failover. You may need to revise the data to ensure consistency.

Potential risks on data quality

Some changes or switchover operations may cause data quality issues such as schema inconsistency between the source and destination databases.

- If data latency occurs between the primary and secondary databases of the source instance, the data written to the primary database is not updated to the secondary database in a timely manner. In this case, if you perform a primary/secondary switchover in the source instance, DTS uses the secondary database of the source instance as the source database for data synchronization. As a result, the data that is not updated to the secondary database is lost.

- If the DTS task is resumed from a network failure after you perform a switchover, DTS attempts to synchronize the data generated before the failure occurs. This mechanism prevents data loss in the destination database. In this case, if the destination tables do not have primary keys, data will be inconsistent between the source and destination databases. If the destination tables have primary keys, data may not be consistent when DTS implements the retry mechanism, but data will remain consistent after the retry ends.
- The data synchronization task may be delayed due to network failures and DDL operations.
- The data synchronization task may be delayed or interrupted due to changes to the source database, unfavorable performance of the destination database, and schema inconsistency.

Alibaba Cloud cannot solve the preceding issues. You must recreate a DTS task or adjust the source and destination databases.

Suggestions to ensure data quality

- You must perform all DDL operations with caution. All DDL operations must be confirmed by the technical engineers to comply with the preceding limits.
- Do not directly perform DDL operations in your program code.

4.6.2. Solutions of data synchronization from a PolarDB-X instance

Data Transmission Service (DTS) supports data synchronization from a source PolarDB-X instance to a destination instance. However, if more than two ApsaraDB RDS for MySQL instances are attached to the source PolarDB-X instance, we recommend that you do not directly configure a data synchronization task for the source PolarDB-X instance. If you do so, you may encounter a performance bottleneck or stability risks. This may affect your business. To ensure good performance and high stability of data synchronization, we recommend that you configure a data synchronization task for each ApsaraDB RDS for MySQL instance attached to the source PolarDB-X instance.

Prerequisites

You have read the Limits and usage notes for synchronizing data between PolarDB-X instances topic and understood the relevant conventions and usage notes.

Synchronization solutions

Solution	Configuration	Description

Solution	Configuration	Description
Solution 1	You can configure a data synchronization task for each ApsaraDB RDS for MySQL instance that is attached to the source PolarDB-X instance. You must map the names of databases and tables in the ApsaraDB RDS for MySQL instances to the names of databases and tables in the destination instance. Compared with Solution 2, this solution ensures good performance and high stability of data synchronization. **Notice** If you use this solution, you can select only tables as the objects to be synchronized.	If you require better performance and higher stability, we recommend that you use this solution. This solution has no limits on the number of ApsaraDB RDS for MySQL instances that are attached to the source PolarDB-X instance.
Solution 2	You can directly configure a data synchronization task for the source PolarDB-X instance. If more than two ApsaraDB RDS for MySQL instances are attached to the source PolarDB-X instance, the performance and stability of data synchronization may deteriorate to the extent that your business may be affected.	If you use this solution, make sure that no more than two ApsaraDB RDS for MySQL instances are attached to the source PolarDB-X instance. Otherwise, the performance and stability of data synchronization may deteriorate.

Comparison between the two solutions

Comparison item	Solution 1	Solution 2
Performance	This solution requires you to configure multiple data synchronization tasks. This way, this solution multiplies the performance of data synchronization and ensures the feasibility of writing a large amount of data from a PolarDB-X instance.	This solution supports only one data synchronization task for the source PolarDB-X instance. If a large amount of data is written to the source PolarDB-X instance by the business side, a performance bottleneck may occur during data synchronization.
Stability	This solution ensures high stability. If a data synchronization task that you configure for one of the ApsaraDB RDS for MySQL instances fails, data synchronization tasks for other ApsaraDB RDS for MySQL instances are not affected. You need to only recover the failed data synchronization task.	This solution provides medium stability. If the data synchronization task for the source PolarDB-X instance fails, data synchronization fails.

Comparison item	Solution 1	Solution 2
Ease of use	You must configure multiple data synchronization tasks. This complicates the configuration process. You must configure database and table name mapping for each task. You must map the names of databases and tables in the ApsaraDB RDS for MySQL instances to the names of the databases and tables in the destination instance.	You need to only configure one data synchronization task. This simplifies the configuration process.
Resource usage	This solution requires multiple data synchronization instances.	This solution requires only one data synchronization instance.

Synchronization of DDL operations

In most cases, data synchronization from a source PolarDB-X instance does not support the synchronization of DDL operations. If you synchronize DDL operations during a data synchronization task, data may fail to be written to the destination instance. To prevent this problem, you can take one of the following measures:

1. Release the data synchronization task.
2. Clear the destination database.
3. Reconfigure the data synchronization task.

However, in a few cases, you can synchronize DDL operations during data synchronization from a source PolarDB-X instance. The following table describes the details of these cases.

Case	Supported operation
You use Solution 1 and select tables as the objects to be synchronized.	• You can add tables to the objects that you select for the data synchronization task. • You can also add columns to or remove columns from the objects. However, you can perform these operations only during data synchronization between PolarDB-X instances. To do this, perform the following steps: i. Add columns to or remove columns from the objects that you select for the data synchronization task. ii. Add or remove columns in the destination database, and then perform the same operations in the source database. When DTS detects that a column already exists in the destination database, DTS ignores the error and does not display a write failure.

Case	Supported operation
You use Solution 2 and select entire databases as the objects to be synchronized.	• You can only add tables to the objects. To do this, you must add tables in the destination database, and then perform the same operations in the source database. • You cannot add or remove columns. **Warning** If you add a column to the database of the source PolarDB-X instance, some physical tables at the underlying layer may contain the column, whereas some physical tables do not contain the column. When DTS executes the assembled SQL statement, DTS may fail to find the column or lose the data in the column.
You use Solution 2 and do not select entire databases as the objects to be synchronized.	• You can only add tables to the objects. To do this, perform the following steps: i. Add tables to the objects that you select for the data synchronization task. ii. Add tables in the destination database, and then perform the same operations in the source database. • You cannot add or remove columns.

5.Change tracking

5.1. Overview

You can use Data Transmission Service (DTS) to track data changes from ApsaraDB RDS for MySQL instances in real time. This feature applies to the following scenarios: lightweight cache updates, business decoupling, asynchronous data processing, and synchronization of extract, transform, and load (ETL) operations.

Supported databases

- User-created MySQL databases or ApsaraDB RDS for MySQL
- PolarDB-X (formerly known as DRDS)
- User-created Oracle database

Objects for change tracking

The objects for change tracking include tables and databases.

In change tracking, data changes include data manipulation language (DML) operations and data definition language (DDL) operations. When you configure change tracking, you must select operation types.

Change tracking tasks

A change tracking task is the basic unit of change tracking and data consumption. To track data changes from an RDS instance, you must create a change tracking task in the DTS console for the RDS instance. The change tracking task pulls data changes from the RDS instance in real time and locally stores the data changes. You can use the DTS SDK to consume the tracked data. You can also create, manage, or delete change tracking tasks in the DTS console.

5.2. Create a change tracking instance

Before you configure a task to track data changes, you must create a change tracking instance. This topic describes how to create a change tracking instance in the Data Transmission Service (DTS) console.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. In the upper-right corner, click **Create Change Tracking Task**.
4. In the Create DTS Instances dialog box, select a region, and enter the number of change tracking instances that you want to create.

> **Note** In the Create DTS Instances dialog box, you can view the total number of instances, the number of existing instances, and the number of instances that can be created.

5. Click **Create**.

5.3. Configure change tracking tasks

5.3.1. Track data changes from a user-created MySQL database or an ApsaraDB RDS for MySQL instance

You can use Data Transmission Service (DTS) to track data changes in real time. This feature applies to the following scenarios: lightweight cache updates, business decoupling, asynchronous data processing, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a user-created MySQL database or an ApsaraDB RDS for MySQL instance.

Prerequisites

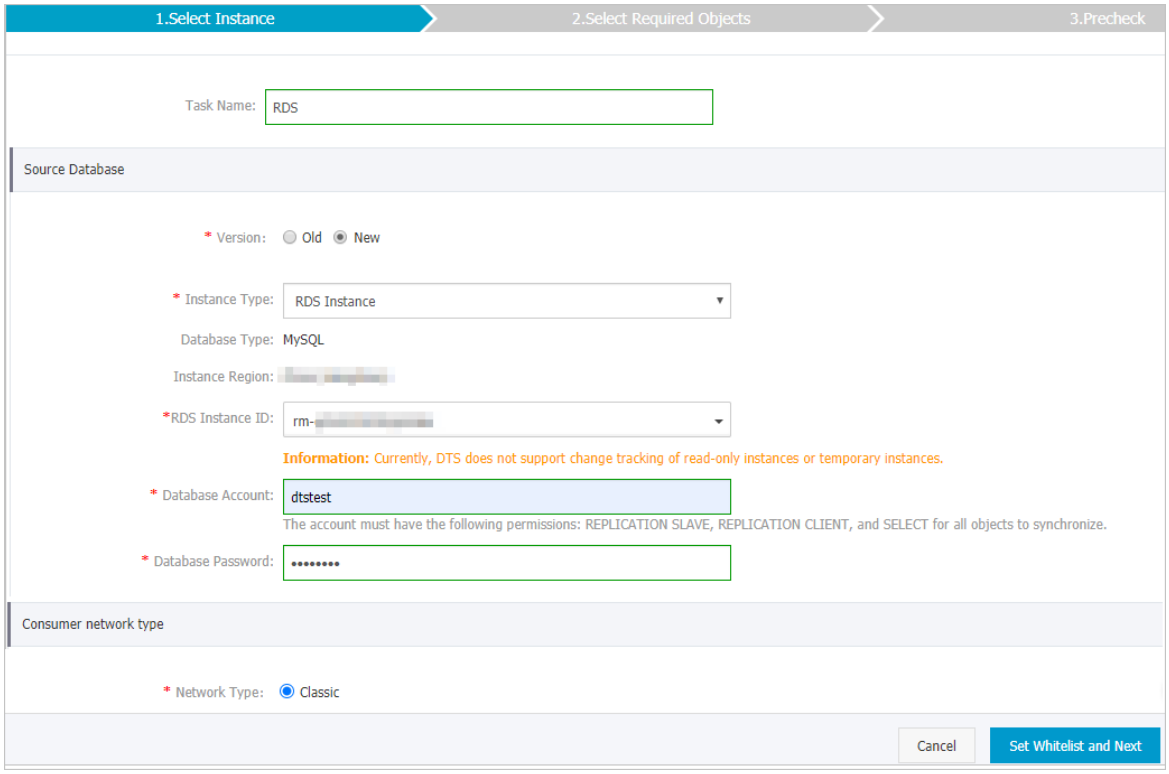
The database version is 5.1, 5.5, 5.6, or 5.7.

Precautions

- DTS does not track data definition language (DDL) operations that are performed by gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflicts.
- If the source database is used in another task, for example, it is used in a running data migration task, DTS may track data changes of other objects. In this case, you must use the change tracking client to filter the tracked data.

Procedure

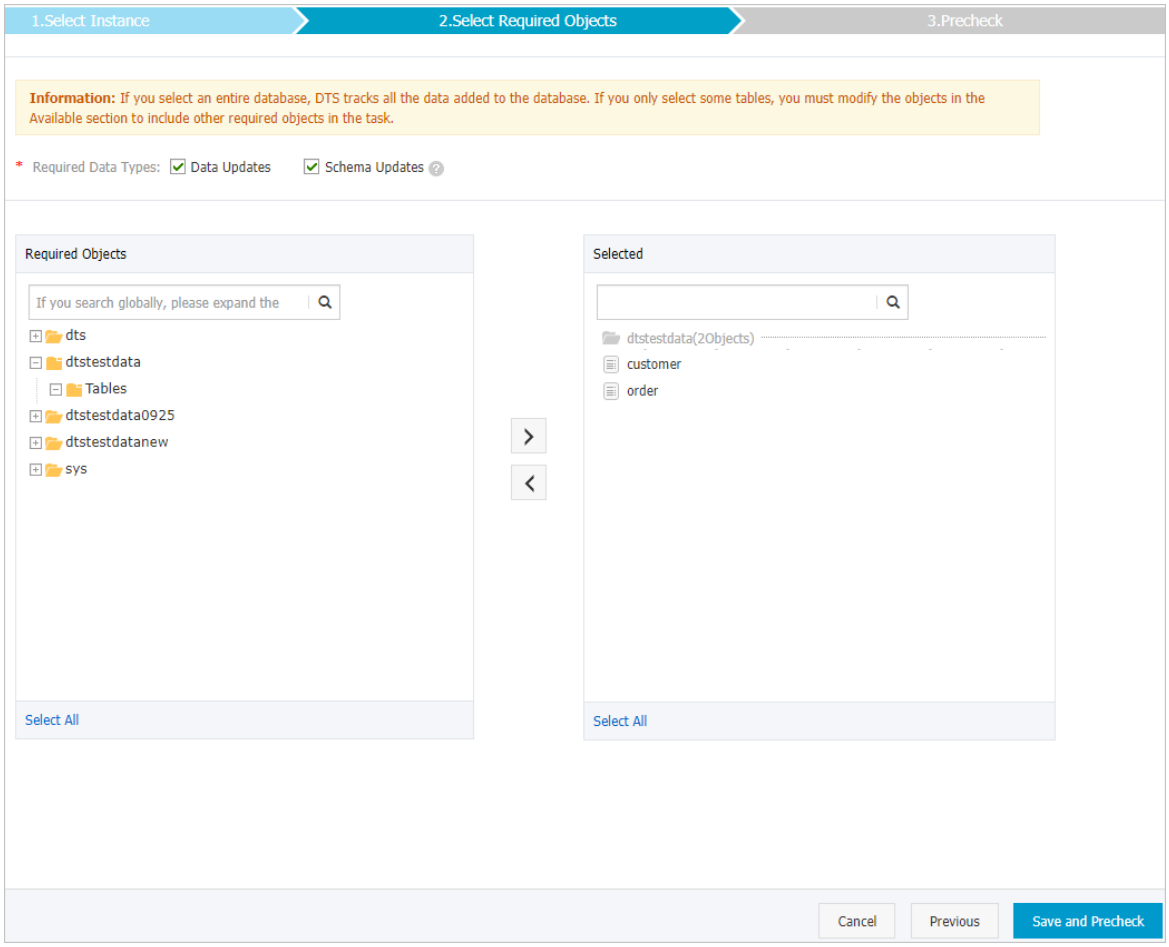
1. Create a change tracking instance.
2. Find the change tracking instance that you created, and click **Configure Channel** in the **Actions** column.
3. Configure the source database.



Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Version	Select a version based on your business requirements: ◦ If the source database is a user-created MySQL database, select **Old**. ◦ If the source database is ApsaraDB RDS for MySQL, select **New**. **Note** You can follow the same procedure to configure a change tracking task when you select Old or New. In this example, select New.
	Instance Type	Select **RDS Instance**.
	Database Type	This parameter is set to **MySQL** and cannot be changed.
	Instance Region	The source region that you selected when you created the change tracking instance. You cannot change the value of this parameter.
	RDS Instance ID	Select the ID of the RDS instance from which you want to track data changes. **Note** A read-only instance or temporary instance cannot be used as the source instance for change tracking.

Section	Parameter	Description
	Database Account	Enter the database account of the source RDS instance. **Note** ◦ The account must have the SELECT permission on the required objects, the REPLICATION SLAVE permission, and the REPLICATION CLIENT permission. ◦ If the database engine of the source RDS instance is **MySQL 5.5** or **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	Enter the password of the source database account.
Consumer Network Type	Network Type	**Classic** is selected by default. **Note** ◦ This parameter is available only if the Version parameter is set to **New**. ◦ If you track data changes over internal networks, the network latency is minimal.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the data change types and objects.



Parameter	Description
Required Data Types	◦ **Data Updates** DTS tracks data updates of the selected objects, including the INSERT, DELETE, and UPDATE operations. ◦ **Schema Updates** DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the required data. **Note** ◦ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ◦ If you select a table as the object, DTS tracks only data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the required objects. For more information, see Modify the objects for change tracking.

Parameter	Description
Required Objects	Select tables or databases from the **Required Objects** section and click the icon to move them to the **Selected** section.

6. In the lower-right corner of the page, click **Save and Precheck**.

> **Note** You can start a change tracking task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**
   After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After the initial change tracking is complete, you can consume the tracked data.

What's next

- Previous change tracking feature: Run the SDK demo code
- New change tracking feature: Use a Kafka client to consume tracked data

5.3.2. Track data changes from a PolarDB-X instance

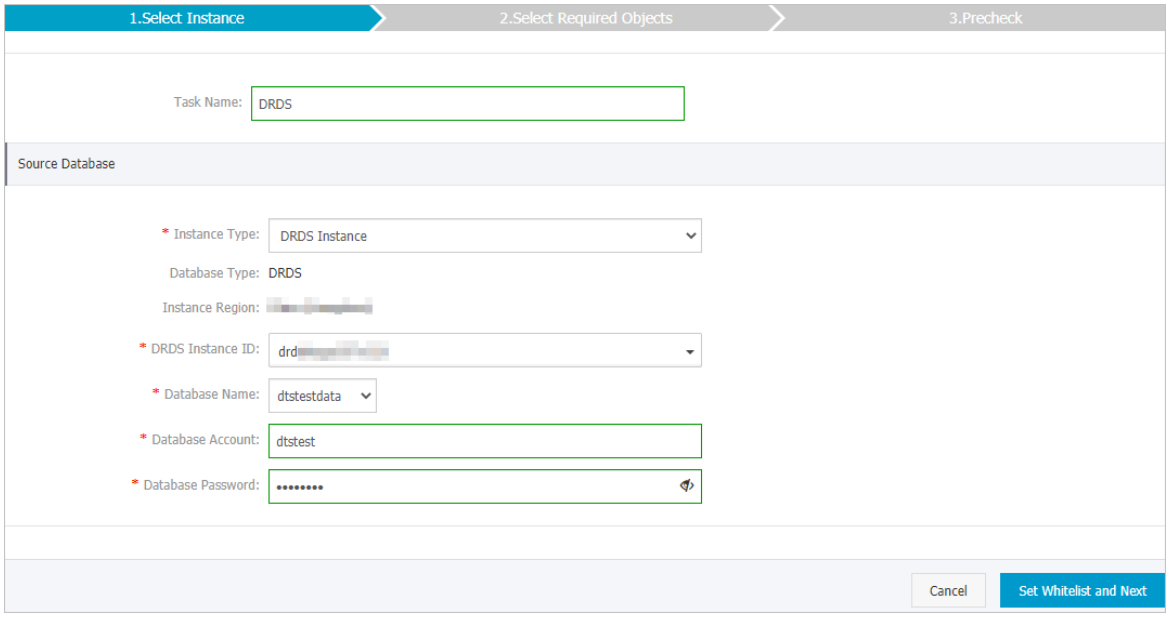
You can use Data Transmission Service (DTS) to track data changes in real time. This feature applies to the following scenarios: lightweight cache updates, business decoupling, asynchronous data processing, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a PolarDB-X instance. PolarDB-X is formerly known as Distributed Relational Database Service (DRDS).

Precautions

- DTS does not track data definition language (DDL) operations that are performed by gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflicts.
- If the source database is used in another task, for example, it is used in a running data migration task, DTS may track data changes of other objects. In this case, you must use the change tracking client to filter the tracked data.

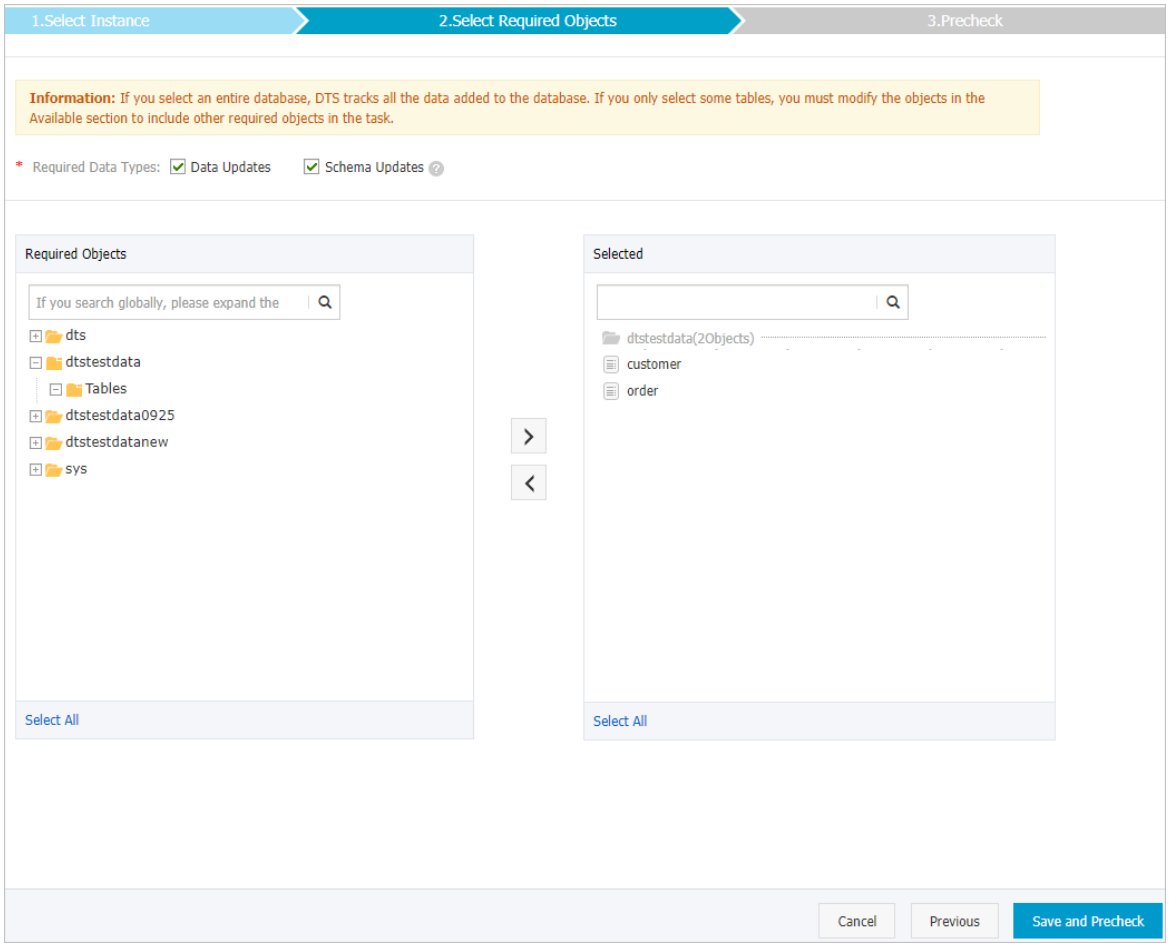
Procedure

1. Create a change tracking instance.
2. Find the change tracking instance that you created, and click **Configure Channel** in the **Actions** column.
3. Configure the source database.



Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	This parameter is set to **DRDS Instance** and cannot be changed.
	Database Type	This parameter is set to **DRDS** and cannot be changed.
	Instance Region	The region of the source instance. The region is the same as the region that you selected when you created the change tracking task. You cannot change the value of this parameter.
	DRDS Instance ID	Select the ID of the PolarDB-X instance.
	Database Name	Select the ID of the source database in the PolarDB-X instance.
	Database Account	Enter the database account of the PolarDB-X instance.
	Database Password	Enter the password of the source database account.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the data change types and objects.



Parameter	Description
Required Data Types	◦ **Data Updates** DTS tracks data updates of the selected objects, including the INSERT, DELETE, and UPDATE operations. ◦ **Schema Updates** DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the required data. **Note** ◦ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ◦ If you select a table as the object, DTS tracks only data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the required objects. For more information, see Modify the objects for change tracking.

Parameter	Description
Required Objects	Select tables or databases from the **Required Objects** section and click the icon to move them to the **Selected** section.

6. In the lower-right corner of the page, click **Save and Precheck**.

> **Note** You can start a change tracking task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.
>
> Troubleshoot the issues based on the causes and run a precheck again.

7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**
   After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After the initial change tracking is complete, you can consume the tracked data.

5.3.3. Track data changes from a user-created Oracle database

You can use Data Transmission Service (DTS) to track data changes in real time. This feature applies to the following scenarios: lightweight cache updates, business decoupling, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a user-created Oracle database.

Prerequisites

- The version of the user-created Oracle database is 9i, 10g, 11g, 12c, 18c, or 19c.
- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the user-created Oracle database. For more information, see Supplemental Logging.
- The user-created Oracle database is running in ARCHIVELOG mode. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

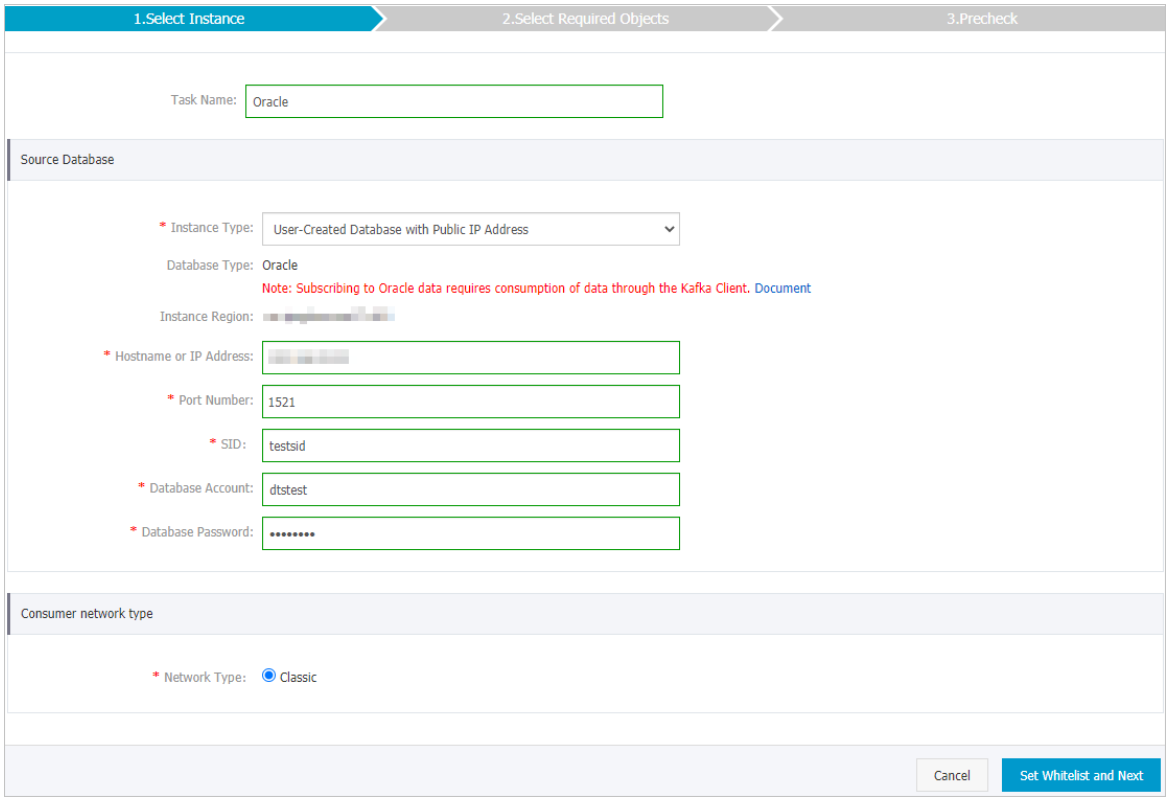
Precautions

- DTS does not track data definition language (DDL) operations that are performed by gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflict.
- If the source database is used in another task, for example, it is used in a running data migration task, DTS may track data changes of other objects. In this case, you must use the change tracking client to filter the tracked data.
- If you perform a primary/secondary switchover on the source database when the change tracking task is running, the task fails.

Procedure

1. Create a change tracking instance.

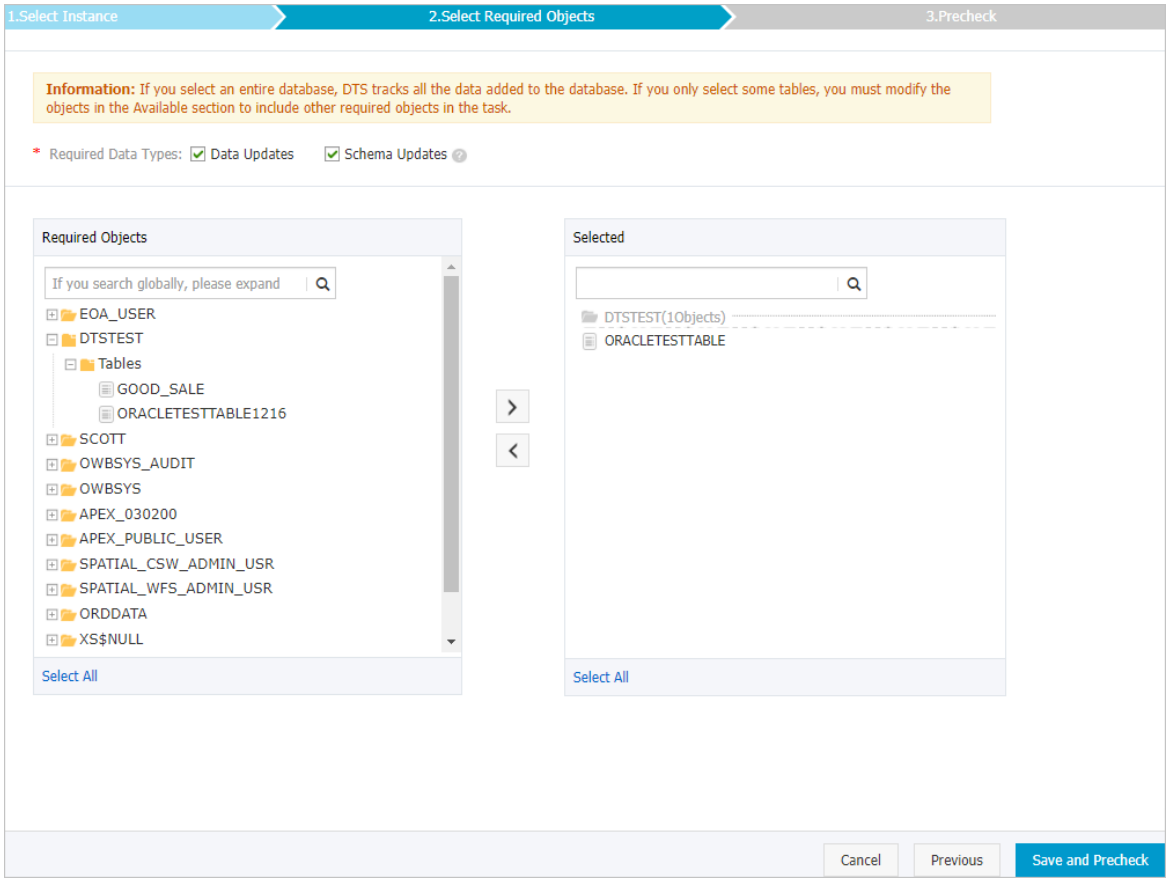
2. Find the change tracking instance that you created, and click **Configure Channel** in the **Actions** column.
3. Configure the source database and network type.



Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select an instance type based on the deployment of the source database. In this example, select **User-Created Database with Public IP Address**.
	Database Type	This parameter is set to Oracle and cannot be changed.
	Instance Region	The source region that you selected when you created the change tracking instance. You cannot change the value of this parameter.
	Hostname or IP Address	Enter the hostname or IP address of the user-created Oracle database.
	Port Number	Enter the service port number of the user-created Oracle database.
	SID	Enter the system ID (SID) of the user-created Oracle database.

Section	Parameter	Description
	Database Account	Enter the account of the user-created Oracle database. **Note** The account must have the database administrator (DBA) permission.
	Database Password	Enter the password of the source database account.
Consumer Network Type	N/A	**Classic** is selected by default. **Note** If you track data changes over internal networks, the network latency is minimal.

4. In the lower-right corner of the page, click **Set Whitelist and Next**.
5. Select the data change types and objects.



Parameter	Description
Required Data Types	◦ **Data Updates** DTS tracks data updates of the selected objects, including the INSERT, DELETE, and UPDATE operations. ◦ **Schema Updates** DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the required data. **Note** ◦ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ◦ If you select a table as the object, DTS only tracks data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the required objects. For more information, see Modify the objects for change tracking.
Required Objects	Select tables or databases from the **Required Objects** section and click the › icon to move them to the **Selected** section.

6. In the lower-right corner of the page, click **Save and Precheck**.

   Note You can start a change tracking task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**
   After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After the initial change tracking is complete, you can consume the tracked data.

What's next

Use a Kafka client to consume tracked data

5.4. Manage change tracking tasks

5.4.1. Modify the consumption checkpoint

During data consumption, you can modify the consumption checkpoint of a change tracking task based on your business requirements. After you modify the consumption checkpoint, the downstream SDK client will consume the data that is generated after the specified time.

Prerequisites

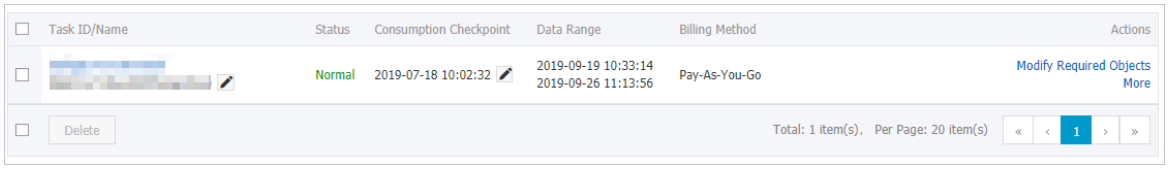
A change tracking task is created. For more information, see Track data changes from a user-created MySQL database or an ApsaraDB RDS for MySQL instance (previous version) or Track data changes from a PolarDB-X instance.

Procedure

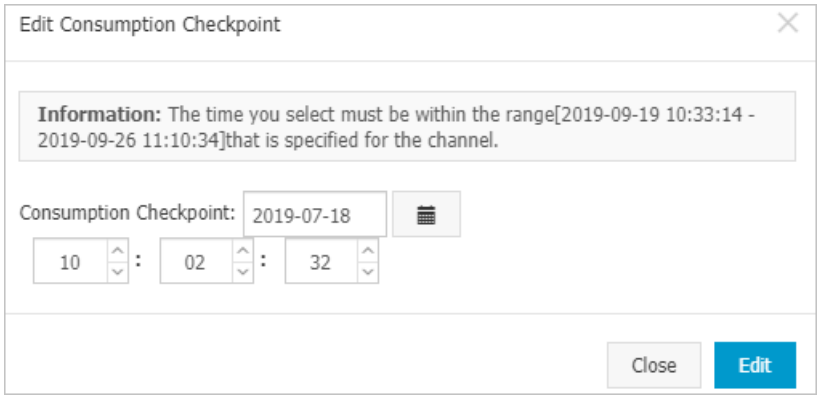
1. Stop all downstream SDK consumption processes.

   > **Note** We recommend that you perform this operation during off-peak hours to avoid service interruption.

2. Log on to the DTS console.
3. In the left-side navigation pane, click **Change Tracking**.
4. Find the change tracking task, move the pointer over the **Consumption Checkpoint** column, and then click the icon.



5. In the Modify Consumption Checkpoint dialog box, specify a new consumption checkpoint.



   > **Note** The selected time range must be within the time range of the tracked data. For more information, see the prompt in the dialog box.

6. Click **Modify**.
7. Restart the downstream SDK consumption processes.
   The downstream SDK client tracks data changes from the new consumption checkpoint.

5.4.2. Modify the objects for change tracking

DTS allows you to add or remove the objects for change tracking in the consumption process. This topic describes how to modify the objects for change tracking.

Procedure

1. Log on to the DTS console.

2. In the left-side navigation pane, click **Change Tracking**.
3. Find the change tracking task, and click **Modify Required Objects** in the **Actions** column.
4. In the **Select Required Objects** step, add or remove the objects for change tracking.
   - Add the objects for change tracking

     In the **Required Objects** section, select one or more objects and click the icon to add the objects to the **Selected** section.
   - Remove the objects for change tracking

     In the **Selected** section, select one or more objects and click the icon to move the objects to the **Required Objects** section.
5. In the lower-right corner of the page, click **Save and Precheck**.

   Note You can start a change tracking task only after the task passes the precheck. If the task fails to pass the precheck, click the icon next to each failed item to view details.

   Troubleshoot the issues based on the causes and run a precheck again.

5.4.3. Create a consumer group

You can manage consumer groups of a change tracking task in the DTS console. This topic describes how to create a consumer group.

Prerequisites

A change tracking task is created. For more information, see Track data changes from a user-created MySQL database or an ApsaraDB RDS for MySQL instance (new version) or Track data changes from a PolarDB-X instance.

Note

- You can create multiple consumer groups (up to 20) in a change tracking instance to repeatedly consume data.
- A consumer group consumes each message only once, and only one consumer can consume data.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. Find the change tracking task and click the task ID.
4. In the left-side navigation pane, click **Consume Data**.
5. On the **Consume Data** page, click **Add Consumer Group** in the upper-right corner.
6. In the dialog box that appears, set the parameters for the consumer group.

Parameter	Description
Consumer Group Name	Enter a new name for the consumer group. We recommend that you use an informative name for easy identification.
Username	Enter the username of the consumer group. ○ A username must contain one or more of the following character types: uppercase letters, lowercase letters, digits, and underscores (_). ○ The username must be 1 to 16 characters in length.
Password	Enter the password that corresponds to the username of the consumer group. ○ A password must contain two or more of the following character types: uppercase letters, lowercase letters, digits, and special characters. ○ The password must be 8 to 32 characters in length.
Confirm Password	Enter the new password again.

7. Click **Create**.

5.4.4. Manage consumer groups

You can manage consumer groups of a change tracking task in the DTS console. This topic describes how to modify the password of a consumer group and how to delete a consumer group.

Prerequisites

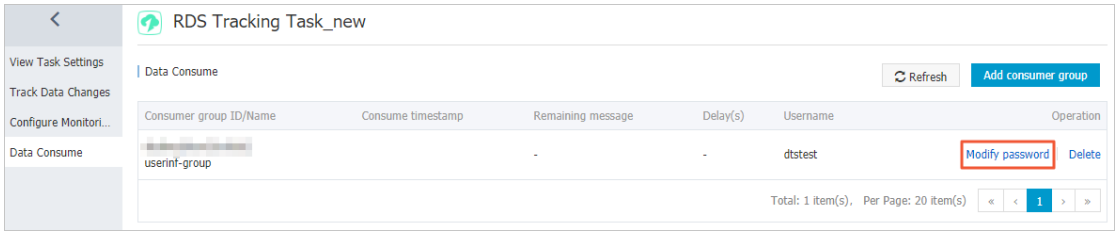
Create a consumer group

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. Find the change tracking task and click the task ID.
4. In the left-side navigation pane, click **Consume Data**.
5. Modify the password of a consumer group or delete a consumer group.

   Modify the password of a consumer group

   i. On the **Consume Data** page, find the target consumer group and click **Modify Password** in the **Actions** column.



ii. In the **Modify Password** dialog box that appears, enter the **old password** and **new password**, and enter the new password again in the **Confirm Password** field.

> **Note**
> - A password must contain two or more of the following character types: uppercase letters, lowercase letters, digits, and special characters.
> - The password must be 8 to 32 characters in length.

iii. Click **Modify**.

Delete a consumer group

> **Note** After a consumer group is deleted, the data in the group will be cleared and cannot be recovered. We recommend that you use caution when performing this operation.

i. On the **Consume Data** page, find the target consumer group and click **Delete** in the **Actions** column.

ii. In the **Delete Consumer Group** message that appears, click **OK**.

5.5. Use the SDK to consume tracked data

5.5.1. Methods provided by SDK

You can use the SDK demo code that is provided by DTS to consume tracked data. This topic describes the methods that are available for the SDK classes.

Methods of the RegionContex class

Method	Description
`setAccessKey(accessKey)`	Specifies the AccessKey ID of the Alibaba Cloud account to which the source instance belongs.
`setSecret(AccessKeySecret)`	Specifies the AccessKey secret of the Alibaba Cloud account to which the source instance belongs.
`setUsePublicIp(usePublicIp)`	Specifies whether to track data changes over the Internet. **Note** DTS can track data changes only over the Internet. Therefore, set the usePublicIp parameter to `true`.
`context.setUseBinary(boolean useBinary)`	Specifies whether to enable the binary packaging feature. Valid values: True and False. We recommend that you enable this feature to improve consumption performance.

Method	Description
`context.setUseDrcNet(boolean useDrcNet)`	Specifies whether to enable the network optimization feature. Valid values: True and False. We recommend that you enable this feature to improve consumption performance.

Methods of the ClusterClient class

Method	Description
`void addConcurrentListener(ClusterListener arg0)`	Adds a downstream listener to retrieve data changes from a change tracking instance. **Note** The `ClusterListener arg0` parameter specifies an object of the `ClusterListener` class.
`void askForGUID(String arg0)`	Retrieves data changes from a change tracking instance. Set the String arg0 parameter to the ID of the change tracking instance.
`List<ClusterListener> getConcurrentListeners()`	Queries the list of listeners in a ClusterClient object. The return type is `List <ClusterListener>`.
`void start()`	Starts the SDK client to start change tracking.
`void stop()`	Stops the SDK client to stop change tracking. **Note** Data pulling and notification callback are performed in the same thread of the SDK client. If the consumption code of the notify() method contains a function that prevents signal interruptions, the stop() function may fail to terminate the SDK client.

Methods of the ClusterListener class

The `void notify(List<ClusterMessage> arg0)` method specifies the consumption mode of tracked data. When the DTS SDK receives the data, it uses the notify() function to notify a ClusterListener object to consume the data. Then, the SDK displays the data on the screen.

Methods of the ClusterMessage class

Note Each ClusterMessage object stores the data record of a transaction. Each data record in the transaction is stored by using a Record object.

Method	Description

Method	Description
`Record getRecord()`	Retrieves a change record from a ClusterMessage object. The change record contains an entry in the binary log file, such as a BEGIN, COMMIT, UPDATE, or INSERT operation.
`void ackAsConsumed`	After the data consumption is complete, you must call this method to send an ACK packet to instruct the DTS server to update the consumer offset. This ensures the integrity of the consumed data after an abnormal SDK client restarts. **Note** If a downstream SDK client restarts after a breakdown, the client resumes change tracking from the last consumer offset.

Methods of the Record class

The `String getAttribute(String key)` method retrieves the attribute values in a Record object. The following table describes the parameters that are available when you call this method.

Parameter	Description
`record_id`	The ID of the record. **Note** The record ID may not increment during the change tracking process.
`instance`	The endpoint that is used to connect to the database instance. The format is <IP address>:<Port number>.
`source_type`	The engine type of the database instance. The value is set to MySQL.
`source_category`	The type of the record. The value is set to full_recorded.
`timestamp`	The binlog timestamp that is generated when the SQL statement is executed in the source database.
`checkpoint`	The checkpoint of the binary log file. The format is `binlog_offset@binlog_file`. **Note** The `binlog_offset` parameter indicates the offset of a record in the binary log file. The `binlog_file` parameter indicates the numerical suffix of the binary log file. For example, if the name of a binary log file is mysql-bin.0008, the value of the binlog_file parameter is 8.

Parameter	Description
record_type	The operation type. Valid values: insert, update, delete, replace, ddl, begin, commit, and heartbeat. **Note** A heartbeat record indicates the heartbeat table that is defined by DTS. The system generates one heartbeat record per second to detect whether the change tracking instance is running as expected.
db	The name of the database.
table_name	The name of the table.
record_recording	The encoding format.
primary	The name of the primary key column. If the primary key is a composite key, separate column names with commas (,).
fields_enc	The encoding of each field value. Separate fields with commas (,). **Note** If a field value is not of the character type, the encoding of this field value is null.

The following table lists the methods that are preset in the SDK demo code. You can call these methods to retrieve the attribute values in a Record object.

Method	Description
Type getOpt()	Queries the operation type.
String getCheckpoint()	Queries the checkpoint of the binary log file.
String gettimestamp()	Queries the timestamp of the binary log file.
String getDbname()	Queries the database name.
String getTablename()	Queries the table name.
String getPrimaryKeys()	Queries the name of the primary key column.
DBType getDbType()	Queries the database type.
String getServerId()	Queries the endpoint that is used to connect to the database instance.
int getFieldCount()	Queries the number of fields.

Method	Description
`List<Field> getFieldList()`	Queries the definitions of all fields, the pre-change image values, and the post-change image values. For more information, see Methods of the Field class.
`Boolean isFirstInLogevent()`	Checks whether the record is the first transaction log in a large volume of data changes. The return value is True or False.

Methods of the Field class

Method	Description
`String getEncoding()`	Obtains the encoding format of the field value.
`String getFieldname()`	Queries the name of the field.
`Type getType()`	Queries the data type of the field.
`ByteString getValue()`	Queries the value of the field. The return type is ByteString. If the field is not specified, the method returns `NULL`.
`Boolean isPrimary()`	Checks whether the field is a primary key column. The return value is True or False.

5.5.2. Quick start

This section describes how to use the DTS Java SDK to perform some basic operations.

Initialize a RegionContext object

A RegionContext object stores the settings of authentication credentials and network access mode. The following code shows how to initialize a RegionContext object.

```
import java.util.List;
import com.aliyun.drc.clusterclient.ClusterClient;
import com.aliyun.drc.clusterclient.DefaultClusterClient;
import com.aliyun.drc.clusterclient.RegionContext;
public class MainClass
{
      public static void main(String[] args) throws Exception {
        // Create a RegionContext object.
          RegionContext context = new RegionContext();
          context.setAccessKey("<AccessKey>");
          context.setSecret("<AccessKeySecret>");
          context.setUsePublicIp(true);
          // Create a ClusterClient object.
          final ClusterClient client = new DefaultClusterClient(context);
          // Other invocation code.
          ...
    }
}
```

Initialize a Listener object

Data consumption is implemented by using an object of the Listener class. After you initialize the ClusterClient object, you must add a Listener object. The Listener object uses the notify() method to receive and consume the tracked data. The following code shows how to display the tracked data on the screen.

```
import com.aliyun.drc.clusterclient.ClusterClient;
import com.aliyun.drc.clusterclient.ClusterListener;
import com.aliyun.drc.clusterclient.DefaultClusterClient;
import com.aliyun.drc.clusterclient.RegionContext;
import com.aliyun.drc.clusterclient.message.ClusterMessage;
public class MainClass
{
    public static void main(String[] args) throws Exception {
        // Initialize the RegionContext object.
        ...
        //Initialize the ClusterClient object.
        ...
        ClusterListener listener = new ClusterListener(){
             @Override
             public void notify(List<ClusterMessage> messages) throws Exception {
                  for (ClusterMessage message : messages) {
                    // Display the tracked data on the screen.
                      System.out.println(message.getRecord() + ":" + message.getRecord().ge
tTablename() + ":"
                      + message.getRecord().getOpt());
                      // Call the following method to send an ACK packet to the DTS server.
                      message.ackAsConsumed();
              }
      }
     }
}
```

DTS saves the consumption checkpoints of the SDK to the DTS server. This simplifies disaster recovery during the use of the SDK. The ackAsConsumed() method sends the checkpoint and timestamp of the latest data record that was consumed by the DTS SDK to the DTS server. If the SDK restarts due to an error, the SDK obtains the consumption checkpoint from the DTS server. The SDK resumes data consumption from the checkpoint. This ensures that the SDK does not consume duplicate data.

Start the ClusterClient object

Use the following code:

```
import java.util.List;
import com.aliyun.drc.clusterclient.ClusterClient;
import com.aliyun.drc.clusterclient.ClusterListener;
import com.aliyun.drc.clusterclient.DefaultClusterClient;
import com.aliyun.drc.clusterclient.RegionContext;
import com.aliyun.drc.clusterclient.message.ClusterMessage;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class MainClass
{
   public static void main(String[] args) throws Exception {
    // Initialize the RegionContext object.
    ...
    // Initialize the ClusterClient object.
    ...
    // Initialize the ClusterListener object.
    ...
    // Add a Listener class.
      client.addConcurrentListener(listener);
      // Specify the ID of the change tracking instance.
      client.askForGUID("dts_rdsrjiei2u2afnb_DSF");
      // Start a background thread. The main thread cannot exit.
      client.start();
}
```

The askForGUID() method sets the ID of the change tracking instance. You can obtain the ID of the change tracking instance from the DTS console. After the ID of the change tracking instance is specified in the askForGUID() method, the SDK retrieves incremental data from this instance.

Before you can start a ClusterClient object, you must add a Listener class to the ClusterClient object. When the ClusterClient object pulls incremental data from the change tracking instance, it also calls the notify() method of the Listener class to consume data.

5.5.3. Parse tracked SQL statements

You can use the DTS SDK to track data changes. DTS records the tracked data changes in a custom format. This topic describes how to parse various types of SQL statements.

Parse a DDL statement

If a data definition language (DDL) operation is performed in the source database, the operation type of the data record is DDL. The DDL statement is stored in the value of the first column. You can use the following sample code to parse the DDL statement:

```
String ddl_string;
Record.Type type=record.getOpt();
if(type.equals(Record.Type.DDL)){
    List<DataMessage.Record.Field> fields = record.getFieldList();
    ddl_string = fields.get(0).getValue().toString();
}
```

Parse an INSERT statement

If an INSERT operation is performed in the source database, the operation type of the data record is INSERT. You can use the following sample code to parse the INSERT statement:

```
StringBuilder insert_string=new StringBuilder();
Record.Type type=record.getOpt();
DataMessage.Record.Field field;
StringBuilder FieldName=new StringBuilder();
StringBuilder FieldValue = new StringBuilder();
if(type.equals(Record.Type.INSERT)){
       int i=0;
       List<DataMessage.Record.Field> fields = record.getFieldList();
       for (; i < fields.size(); i++) {
              field = fields.get(i);                                    FieldName.append('`'
+field.getFieldname().toLowerCase()+'`');
              FieldValue.append("'"+field.getValue()+"'");
              if (i ! = fields.size() - 1) {
                     FieldName.append(',');
                     FieldValue.append(',');
              }
        }
        insert_string.append("insert "+ record.getTablename()+"("+FieldName.toString()+") v
alues("+FieldValue.toString()+");");
}
```

Parse an UPDATE statement

If an UPDATE operation is performed in the source database, the operation type of the data record is UPDATE. The field values prior to the UPDATE operation are stored in `Record.getFieldList()` entries with even indexes. The field values after the UPDATE operation are stored in Record.getFieldList() entries with odd indexes.

If the UPDATE operation is performed on a table that has a primary key, you can use the following sample code to parse the UPDATE statement:

```
StringBuilder update_string=new StringBuilder();
Record.Type type=record.getOpt();
DataMessage.Record.Field field;
StringBuilder SetValue = new StringBuilder();
StringBuilder WhereCondition = new StringBuilder();
String ConditionStr;
boolean hasPk=false;
boolean pkMode=false;
boolean hasSet=false;
if(type.equals(Record.Type.UPDATE)){
    int i=0;
    DataMessage.Record.Field OldField = null;
    DataMessage.Record.Field NewField = null;
    List<DataMessage.Record.Field> fields = record.getFieldList();
    for (; i <fields.size() ; i++) {
        if (i % 2 == 0) {
            OldField = fields.get(i);
            continue;
        }
    NewField = fields.get(i);
```

```
    field = NewField;
    if (field.isPrimary()) {
        if (hasPk) {
            WhereCondition.append(" and ");
        }
        //where old value
        ConditionStr = getFieldValue(OldField);
        if(ConditionStr==null){                                          WhereCondition.appe
nd("`"+field.getFieldname().toLowerCase()+"`" + " " + "is null");
        }else{
               WhereCondition.append("`"+field.getFieldname().toLowerCase()+"`"+" = "+ "'"+
OldField.getValue()+"'");
         }
        hasPk = true;
    }
    if (hasSet) {
        SetValue.append(",");
    }
      SetValue.append("`"+field.getFieldname().toLowerCase()+"`" + " = " + "'"+field.getVal
ue()+"'");
    String setStr = getFieldValue(field);
    hasSet = true;
    }
    update_string.append("Update "+record.getTablename() +" Set " + SetValue + " Where "+Wh
ereCondition +";");
}
protected String getFieldValue(Field field) throws Exception {
    ByteString byteString = field.getValue();
    if (byteString == null) {
        return null;
    }
    else {
        String value;
        if (field.getType() == com.aliyun.drc.client.message.DataMessage.Record.Field.Type.
STRING && field.getEncoding() ! = null && field.getEncoding() ! = "ASCII") {
            value = field.getValue().toString(field.getEncoding());
        }
        else {
          value = byteString.toString();
        }
        return value;
    }
}
```

Parse a DELETE statement

If a DELETE operation is performed in the source database, the operation type of the data record is DELETE. If the DELETE operation is performed on a table that has a primary key, you can use the following sample code to parse the DELETE statement:

```
StringBuilder delete_string=new StringBuilder();
Record.Type type=record.getOpt();
DataMessage.Record.Field field;
StringBuilder FieldName=new StringBuilder();
StringBuilder FieldValue = new StringBuilder();
StringBuilder DeleteCondition = new StringBuilder();
boolean hasPk=false;
boolean pkMode=false;
if(type.equals(Record.Type.DELETE)){
   int i=0;
   List<DataMessage.Record.Field> fields = record.getFieldList();
   delete_string.append("Delete From" + record.getTablename() + "where");

   // Check whether the table has a primary key.
   if (record.getPrimaryKeys() ! = null) {
             pkMode = record.getPrimaryKeys().length() > 0 ? true : false;
   }
   for (; i < fields.size(); i++) {
            if ((pkMode && ! field.isPrimary())) {
                    continue;
            }
            if (hasPk) {
                    delete_string.append(" and ");
            }
            delete_string.append(field.getFieldname() + "=" + field.getValue());
            hasPk = true;
    }
    delete_string.append(";");
}
```

Parse a REPLACE statement

If a REPLACE operation is performed in the source database, the operation type of the data record is UPDATE or INSERT.

- If the value specified in the REPLACE statement does not exist, the operation type of the data record is INSERT.
- If the value specified in the REPLACE statement exists, the operation type of the data record is UPDATE.

Parse a BEGIN statement

If a BEGIN operation is performed in the source database, the operation type of the data record is BEGIN. You do not need to perform operations on fields because the BEGIN statement does not modify fields. You only need to check that the operation is a BEGIN operation. You can use the following sample code to parse the BEGIN statement:

```
StringBuilder sql_string = new StringBuilder();
Record.Type type = record.getOpt();
if(type.equals(Record.Type.BEGIN)){
        sql_string.append("Begin");
}
```

Parse a COMMIT statement

If a COMMIT operation is performed in the source database, the operation type for the data record is COMMIT. You do not need to perform operations on fields because the COMMIT statement does not modify fields. You only need to check that the operation is a COMMIT operation. You can use the following sample code to parse the COMMIT statement:

```
StringBuilder sql_string = new StringBuilder();
Record.Type type = record.getOpt();
if(type.equals(Record.Type.COMMIT)){
        sql_string.append("commit");
}
```

5.5.4. Run the SDK demo code

This section describes how to run the demo code provided by the DTS console.

1. Create an AccessKey.

   Your account must pass the AccessKey authentication before you can use an SDK to connect to a subscription channel. Therefore, before using the SDK, you must obtain an AccessKey. For more information, see the "**Obtain an AccessKey**" section of the *DTS Developer Guide* .

2. Install the Java SDK.

   The development environment supported by the DTS Java SDK is J2SE Development Kit (JDK) V1.5 or later.

   For an Eclipse project, you can follow these steps to install the Java SDK:

   i. Click **View Example Code** and download the SDK package *consumer.jar*.

   ii. Import the JAR package to an Eclipse project as follows:

   In Eclipse, right-click your project and choose **Properties > Java Build Path > Libraries > Add External JARs**. Select the path for storing the *consumer.jar* package consumer.jar.

   iii. Select the *consumer.jar* package and click **OK**.

   Then you can use the DTS Java SDK in the project.

3. Run the demo code.

   DTS provides the SDK demo code. You can copy the demo code by using the View Demo Code option in the DTS console. For an Eclipse project, you can follow these steps to run the demo code:

   i. Create a class named MainClass in the src directory of the Eclipse project.

   ii. Open the generated Java file *MainClass* and delete the code template.

   iii. Paste the demo code into the *MainClass* file.

   iv. Modify the AccessKeyId, AccessKeySecret, and subscription channel ID in the demo code.

   Change the marked parts in the preceding demo code to the AccessKeyId, AccessKeySecret, and subscription channel ID of your account.

   You can obtain the subscription channel ID from the DTS console.

v. In Eclipse, right-click the demo file and choose **Run as > Java Application** to run the demo code.

5.6. Use a Kafka client to consume tracked data

This topic describes how to use the demo code of a Kafka client to consume tracked data. The change tracking feature of the new version allows you to consume tracked data by using a Kafka client from V0.11 to V1.1.

Prerequisites

- A change tracking task is created. For more information, see Track data changes from a user-created MySQL database or an ApsaraDB RDS for MySQL instance (new version) or Track data changes from a PolarDB-X instance.
- One or more consumer groups are created. For more information, see Create a consumer group.

Precautions

- If you enable auto commit when you use the change tracking feature, some data may be committed before it is consumed. This results in data loss. We recommend that you manually commit data.

  > **Note** If data fails to be committed due to a fault, you can restart the client to continue consuming data from the last recorded consumer offset. However, duplicate data may be generated during this period. You must manually filter out the duplicate data.

- Data is serialized and stored in the Avro format. For more information, see Record.avsc.

  > **Note** If the client that you use is not a Kafka client, you must parse the tracked data based on the Avro schema.

- Regarding the `offsetFotTimes` interface, the search unit of DTS is seconds, and the search unit of native Kafka is milliseconds.

Download and run the demo code of the Kafka client

Click here to download the demo code of the Kafka client.

Download and run the demo code of the Kafka client

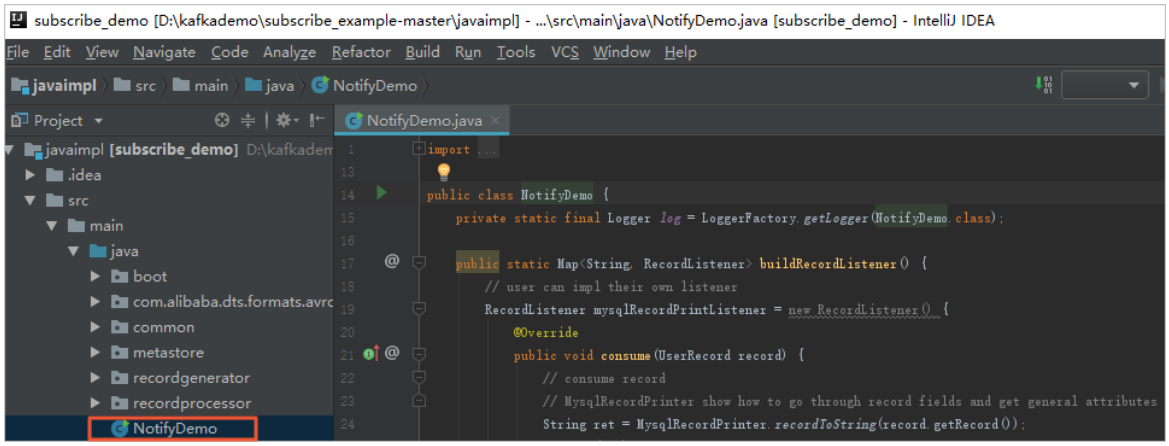
Step	File or directory
1. Use the native Kafka consumer to obtain incremental data from the change tracking instance.	subscribe_example-master/javaimpl/src/main/java/recordgenerator/
2. Deserialize the image of the incremental data, and obtain attributes such as the pre-image and post-image.	subscribe_example-master/javaimpl/src/main/java/boot/MysqlRecordPrinter.java

Step	File or directory
3. Convert the dataTypeNumber values in the deserialized data into MySQL or Oracle data types. **Note** For more information, see Mappings between MySQL data types and dataTypeNumber values and Mappings between Oracle data types and dataTypeNumber values.	subscribe_example-master/javaimpl/src/main/java/recordprocessor/mysql/

Procedure

This procedure uses IntelliJ IDEA (Community Edition 2018.1.4 Windows) as an example.

1. Download the demo code of the Kafka client, and then decompress the package.
2. Open IntelliJ IDEA. In the window that appears, click **Open**.
3. In the dialog box that appears, go to the directory in which the downloaded demo code resides. Find the *pom.xml* file.
4. In the dialog box that appears, select **Open as Project**.
5. On the IntelliJ IDEA page, expand folders to find the demo file of the Kafka client, and double-click the file. The file name is *NotifyDemo.java*.



6. Set the parameters in the *NotifyDemo.java* file.

Parameter	Description	Method to obtain

Parameter	Description	Method to obtain
USER_NAME	The username of the consumer group. **Warning** If you are not using the Kafka client that is described in this topic, you must specify the username in the following format: `<Consumer group account>-<Consumer group ID>`, for example, `dtstest-dtsae******bpv`. Otherwise, the connection fails.	In the DTS console, click the instance ID, and then click **Data Consume**. You can obtain the **Consumer Group ID** and the corresponding **Account** information. **Note** The password of the consumer group account is specified when you create a consumer group.
PASSWORD_NAME	The password of the account.	
SID_NAME	The ID of the consumer group.	
GROUP_NAME	The name of the consumer group. Set this parameter to the consumer group ID.	
KAFKA_TOPIC	The topic of the change tracking task.	In the DTS console, click the instance ID. On the **Track Data Changes** page, you can obtain the **tracked topic**, network address, and port number.
KAFKA_BROKER_URL_NAME	The network address and port number of the change tracking task. **Note** If you track data changes over internal networks, the network latency is minimal. This is applicable if the ECS instance where you deploy the Kafka client belongs to the same VPC or classic network as the change tracking instance.	

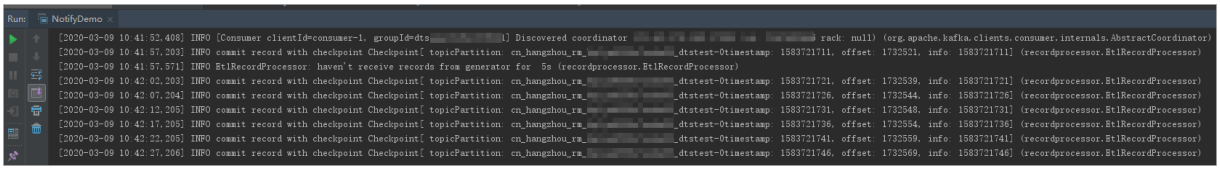
Parameter	Description	Method to obtain
INITIAL_CHECKPOINT_NAME	The consumer offset of consumed data. The value is a UNIX timestamp. **Note** You must save the consumer offset. If the consumption process is interrupted, you can specify the consumer offset on the change tracking client to resume data consumption. This allows you to prevent against data loss. When you start the change tracking client, you can specify the consumer offset to consume data on demand.	When you use the Kafka client to track data changes for the first time, convert the required time point into a UNIX timestamp.
USE_CONFIG_CHECKPOINT_NAME	Default value: *true*. The default value indicates that the client is forced to consume data from the specified consumer offset. This allows you to retain the data that is received but not processed.	None.

7. On the top of the IntelliJ IDEA page, choose **Run** > **Run** to run the client.

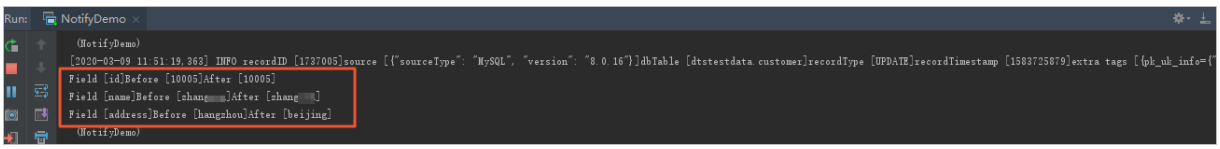
Note When you run IntelliJ IDEA for the first time, it loads and installs the relevant dependency.

Execution result

The following figure shows that the Kafka client can track data changes in the source database.



You can also delete the `//` characters from the `//log.info(ret);` string in line 25 of the *NotifyDemo.java* file. Then, run the client again to view the data change information.



Mappings between MySQL data types and dataTypeNumber values

MySQL data type	Value of dataTypeNumber
MYSQL_TYPE_DECIMAL	0

MySQL data type	Value of dataTypeNumber
MYSQL_TYPE_INT8	1
MYSQL_TYPE_INT16	2
MYSQL_TYPE_INT32	3
MYSQL_TYPE_FLOAT	4
MYSQL_TYPE_DOUBLE	5
MYSQL_TYPE_NULL	6
MYSQL_TYPE_TIMESTAMP	7
MYSQL_TYPE_INT64	8
MYSQL_TYPE_INT24	9
MYSQL_TYPE_DATE	10
MYSQL_TYPE_TIME	11
MYSQL_TYPE_DATETIME	12
MYSQL_TYPE_YEAR	13
MYSQL_TYPE_DATE_NEW	14
MYSQL_TYPE_VARCHAR	15
MYSQL_TYPE_BIT	16
MYSQL_TYPE_TIMESTAMP_NEW	17
MYSQL_TYPE_DATETIME_NEW	18
MYSQL_TYPE_TIME_NEW	19
MYSQL_TYPE_JSON	245
MYSQL_TYPE_DECIMAL_NEW	246
MYSQL_TYPE_ENUM	247
MYSQL_TYPE_SET	248
MYSQL_TYPE_TINY_BLOB	249
MYSQL_TYPE_MEDIUM_BLOB	250
MYSQL_TYPE_LONG_BLOB	251

MySQL data type	Value of dataTypeNumber
MYSQL_TYPE_BLOB	252
MYSQL_TYPE_VAR_STRING	253
MYSQL_TYPE_STRING	254
MYSQL_TYPE_GEOMETRY	255

Mappings between Oracle data types and dataTypeNumber values

Oracle data type	Value of dataTypeNumber
VARCHAR2/NVARCHAR2	1
NUMBER/FLOAT	2
LONG	8
DATE	12
RAW	23
LONG_RAW	24
UNDEFINED	29
XMLTYPE	58
ROWID	69
CHAR and NCHAR	96
BINARY_FLOAT	100
BINARY_DOUBLE	101
CLOB/NCLOB	112
BLOB	113
BFILE	114
TIMESTAMP	180
TIMESTAMP_WITH_TIME_ZONE	181
INTERVAL_YEAR_TO_MONTH	182
INTERVAL_DAY_TO_SECOND	183
UROWID	208

Oracle data type	Value of dataTypeNumber
TIMESTAMP_WITH_LOCAL_TIME_ZONE	231