Data Transmission Service

User Manual

User Manual

Data Migration

This section introduces the instance type and data source type supported for data migration.

Instance types

Source Instance	Target Instance
Local self-built IDC databases	RDS in classic networks and RDS instances in VPC networks Self-built ECS databases
Self-built ECS databases	RDS in classic networks and RDS instances in VPC networks Self-built ECS databases
RDS instances	RDS in classic networks and RDS instances in VPC networks Self-built ECS databases
Redis instances in classic networks	Redis instances in classic networks Self-built ECS databases

Data sources

Below are the data sources supported for data migration:

Source Database	Structure Migration	Full Migration	Incremental Migration
Oracle->RDS For MySQL	Supported	Supported	Not Supported
MySQL ->RDS For MySQL	Supported	Supported	Supported
SQLServer->RDS For SQLServer	Supported	Supported	Supported
PostgreSQL-> RDS	Supported	Supported	Supported

For PostgreSQL			
Redis -> Redis	—/—	Supported	Supported

DTS migrates data from a local Oracle to a RDS MySQL instance for structure definition migration and data migration. To ensure data consistency, it is recommended to disable writing on local Oracle during migration.

This section describes how to use DTS to configure data migration tasks from Oracle to RDS for MySQL.

Migration steps

Migration from Oracle to RDS for MySQL only supports structure definition migration and full migration. Below are some limits applied to structure migration and full migration.

- Structure migration
DTS migrates the structure definition of the migrated object to the target instance. It supports only the table object for structure migration rather than other objects such as view, synonym, trigger, procedure, function, package and user define type.

- data migration
  DTS migrates all data of the migrated object in the source database to the target instance. If there is data written into the local Oracle instance during migration, the change data is not necessarily migrated to MySQL. Therefore, to ensure migration data consistency, you should disable writing data into the local Oracle instance during migration.

Migration permission

You should authorize the migration accounts of source and target databases the following permissions for different migration types:

Migration Type	Structure Migration	Full Migration
Local Oracle Instance	Schema Owner	Schema Owner
Target RDS MySQL Instance	Read/Write Permission of the DB to be Migrated	Read/Write Permission of the DB to be Migrated

Migration process

To streamline the dependency between objects and improve success rates of data migration from Oracle to RDS for MySQL, DTS defines the migration process of structure objects and data as below:

1. Structure object migration: table migration

2. data migration
3. Structure object migration: foreign key migration

After data migration is completed, the migration progress in the task list is: **Structure migration 100%, data migration 100%**, but the migration status is **migrating**. At this time, the migration task is migrating the foreign key in Step (3). Do not end the task manually, otherwise it may cause migration data inconsistency.

Data type mapping relationship

Given that Oracle and MySQL data types are not in a one-to-one relationship, DTS maps the data types of Oracle and MySQL during structure migration. Below shows the mapping relationship of the data types of Oracle and MySQL.

Oracle Data Type	MySQL Data Type	Supported by DTS?
varchar2(n [char/byte])	varchar(n)	Supported
nvarchar2[(n)]	national varchar[(n)]	Supported
char[(n [byte/char])]	char[(n)]	Supported
nchar[(n)]]	national char[(n)]	Supported
number[(p[,s])]	decimal[(p[,s])]	Supported
float(p)]	double	Supported
long	longtext	Supported
date	datetime	Supported
binary_float	decimal(65,8)	Supported
binary_double	double	Supported
timestamp[(fractional_seconds_precision)]	datetime[(fractional_seconds_precision)]	Supported
timestamp[(fractional_seconds_precision)]with local time zone	datetime[(fractional_seconds_precision)]	Supported
timestamp[(fractional_seconds_precision)]with local time zone	datetime[(fractional_seconds_precision)]	Supported
clob	longtext	Supported
nclob	longtext	Supported
blob	longblob	Supported
raw	varbinary(2000)	Supported
long raw	longblob	Supported
bfile	-/-	**Not Supported**

interval year(year_precision) to mongth	-/-	**Not Supported**
interval day(day_precision) to second[(fractional_seconds_precision)]	-/-	**Not Supported**

- For char type, when char(n) definition length n exceeds 255, DTS automatically converts the char type to varchar(n).
- Since MySQL does not support bfile, interval year to month and interval day to second data types in Oracle, DTS does not convert these three types during structure migration. If the migrating table contains these three data types, the structure migration will fail. You can modify these three types into the corresponding data types in MySQL based on your business requirements.
- Since MySQL timestamp type does not contain time zone, while Oracle timestamp with [local] time zone contains time zone, DTS convert these two types of data to the UTC time zone and store them into MySQL during migration.

Operation procedure

Below shows how to configure a DTS migration task to migrate data from local Oracle to RDS for MySQL.

Create a database on a RDS instance

During data migration, if the name of the database to be migrated does not comply with the RDS defined specification (that is, the name has a maximum of 64 characters starting with a letter and ending with a letter or digit, and consisting of lowercase letters, digits, underscores and strike-throughs),you need to manually create the database before configuring a migration task.

Create a migration account

When configuring a migration task, you need to provide the migration accounts of the Oracle database and the target RDS instance. For permissions required for the migration accounts, see the **Migration permission** section above.

If you haven’ t created any migration accounts for your Oracle instance, refer to [Oracle Grant Syntax Instructions] (https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_9013.htm “Oracle Grant Syntax Instructions”) to create a migration account meeting the requirements.

For creating and authorizing a RDS for MySQL migration account, see User Manual of RDS Creating an account

Configure a migration task

To configure a migration task, follow the below steps:

1. Log in to DTS Console, click **Create Migration Task** on the top right to start task configuration.
2. Configure the connection information of local Oracle database and the target RDS instance.

Configure the migration task name and the connection information of Oracle and the target RDS instances. Where:
- Task name

DTS generates a name for every task automatically. The task name is not required to be unique. You can modify the name as needed. It is recommended to use a name reflecting specific services.

Source instance

Instance type: select **On premises databases**

- Instance Region: Select the region closest to the source instance.
- Database type: Select **Oracle**
- Host name or IP address: Configure the IP address for accessing Oracle. The address must be in public access mode.
- Port: the listener port of the Oracle instance.
- SID: SID of the Oracle instance
- Account: the connection account of the Oracle instance

Password: the password of the above Oracle connection account

Target instance

Instance type: to select **RDS instance**

- RDS instance ID: Configure the instance ID of the target RDS instance to be migrated. DTS supports RDS instances in classic networks and VPC networks.
- Account: the access account to the RDS instance.
- Password: the password of the above RDS connection account.

After configuring the connection information, click **Authorize Whitelist and enter into next step** to enter into next step. Here, DTS adds the IP address of DTS server to the whitelist of the target RDS instances to avoid migration failure.

3. Configure the migration object and migration type

The migration type includes structure migration and data migration. By default, **Structure migration** and **data migration** are selected.

Select the object to be migrated: database, table or column.
By default, after the object is migrated to the RDS instance, its name remains the same with that in Oracle. Otherwise, you need to use the object name mapping function.

After you configure migration object and migration type, start pre-check.

4. Pre-check

Migration can be started only after pre-check succeeds.

If pre-check fails, click **Failure** next to the check the failure resion , solve the problem and execute pre-check again.

5. Start the migration task

After pre-check succeeds, start the migration task. You can view the migration status and progress in the task list.

So far, the data migration task configuration from a local Oracle database to the RDS For MySQL has been complete.

DTS migrates data from a local MySQL instance to a RDS for MySQL instance without disabling local application services.

This section introduces how to use DTS to configure a data migration task from local MySQL to RDS for MySQL.

Migration types

For data migration from MySQL to RDS for MySQL, DTS supports structure migration, data migration and change data replication. The features and restrictions can be found as below.

- **Structure migration**

In this step, DTS migrates the structure definition of the migrated object to the target instance. DTS currently supports the following objects: table, view, trigger,procedure and function.

- **data migration**

In this step, DTS migrates all data stored in the local MySQL instance to the target RDS instance.

If you only perform data migration, the newly added data to the local MySQL instance during the data migration won’ t be synchronized to the target RDS instance.

If you select change data replication additionally, non-transaction tables without a primary key will be locked in data migration to ensure data consistency. You cannot write any data to the locked tables during the locking period of time, depending on the data volume of these tables. These locked tables will be released only after they are migrated.

- **change data replication**

In this step, DTS synchronizes the change data of the local MySQL instance during the migration to the target RDS instance.

Migration restrictions

- DDL operations are not supported during migration.
- Event migration is not supported in structure migration.
- If object name mapping is enabled, other objects associated with this object may fail to be migrated.
- When change data replication is selected, binlog of source MySQL instances must be enabled.
- When change data replication is selected, binlog_format of the source database must be **row**.
- When change data replication is selected and the source MySQL version is 5.6, binlog_row_image must be **full**.

Migration permission

When DTS is used for data migration from local MySQL to RDS for MySQL, the migration accounts of the source MySQL instance and the target RDS for MySQL instance have different permission requirements for migration types, as follows:

Migration Type	Structure Migration	Migrating existing data	Change data replication
Local MySQL instance	select	select	select replication slave replication client
Target RDS instance	Read/write permission	Read/write permission	Read/write permission

Migration process

To streamline the dependency between objects and improve migration success rates for data migration from MySQL to RDS for MySQL, DTS defines the migration process of structure objects and data as follows:

1. Structure object: migration of tables and views
2. Migrating exists data
3. Structure object: migration of procedures, functions, triggers and foreign keys
4. Change data replication

If you have not set up change data replication for a migration task, after the data migration is completed, the migration progress in the task list is **Structure migration 100%, data migration 100%**, and the migration status is **migrating**. At this time, procedures, functions, triggers and foreign keys are migrated. Do not end the task manually, otherwise it may cause migration data loss.

Configure a migration task

The following shows how to configure a migration task to migrate data from the local MySQL to RDS for MySQL.

Create a database on RDS instance

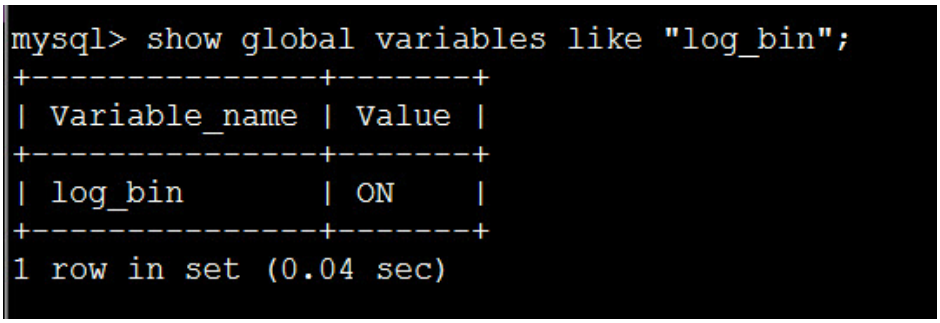
During data migration, if the database to be migrated does not exist in the target RDS instance, DTS will automatically create one. But if the name of the migration database does not comply with the RDS defined specification (that is, the name has a maximum of 64 characters starting with a letter and ending with a letter or digit, and consisting of lowercase letters, digits, underscores and strike-throughs),you need to manually create the database before configuring a migration task.

Create a migration account

When configuring a migration task, you need to provide the migration accounts of the local MySQL instance and the target RDS instance. For permissions required for the migration accounts, see the **Migration permission** section above.

Other Preparations

To perform change data replication , check whether binlog of the source database is enabled and whether the format meets requirements. The items to be checked are as follows:
1) Check whether Binlog is enabled for the source database.



If log_bin is OFF, it indicates that binlog is not enabled for the source database. To enable change data replication migration, modify the binlog configuration of local MySQL.

Enable Binlog, modify log_bin to mysql_bin.
Set the Binlog format to “row”, modify binlog_format to **row**.
Set server_id to be greater than 1, that is, server_id is an integer greater than 1.
If the version is 5.6, modify binlog_row_image to **full**.
After modification is completed, restart the MySQL process.

2. Check whether the binlog format of the source database is row.

```
mysql> show global variables like "binlog_format";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
1 row in set (0.04 sec)
```

Use the above command to check whether the binlog format of local MySQL is **row**. If not, use the following command to change the format to row.

```
mysql> set global binlog_format=ROW;
Query OK, 0 rows affected (0.04 sec)
```

After that, you are suggested to remove the connections and reconnect to MySQL. Otherwise, there may be other connections that still use the statement format to record binlog.

3. When the local MySQL version is or later than 5.6.2, check whether binlog_row_image of the source database is full.

```
mysql> show global variables like "binlog_row_image";
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| binlog_row_image | FULL  |
+------------------+-------+
1 row in set (0.04 sec)
```

If not, use the following command to change it to **full**.

```
mysql> set global binlog_row_image=full;
Query OK, 0 rows affected (0.03 sec)
```

Configure a migration task

When the database and migration accounts are both created, you can start configuring a migration task according to the following steps:

1. Log in to DTS Console, click **Create Migration Task** on the top right to start task configuration.
2. Configure the connection information of local MySQL instance and the target RDS instance.

Configure the migration task name, the migration source instance and connection information of the

target instance. Where:

- **Task name**

DTS generates a name for every task automatically. The task name is not required to be unique. You can modify the name as needed. It is recommended to use a name reflecting specific services.

Source instance

Instance type: **On-premises databases**

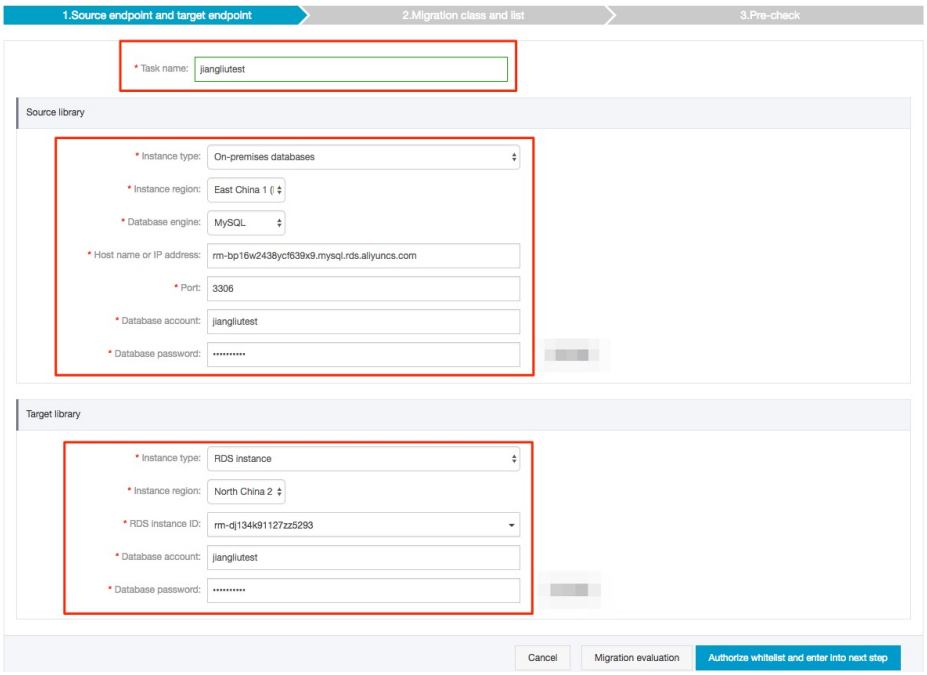
- Instance Region: Select the region closest to the source instance.
- Database type: **MySQL**
- Host name or IP address: configure the address for accessing local MySQL database.
- Port: listener port of the MySQL instance.
- Account: access account of the MySQL instance.

Password: access account password of the MySQL instance.

target RDS instance

Instance type: **RDS instance**

- RDS instance ID: configure the instance ID of the target RDS instance to be migrated. DTS supports RDS instances in classic networks and VPC networks.
- Database account: access account of the RDS instance.
- Database password: access password of the RDS instance.



3. Configure the migration object and migration type

- **Migration type**

DTS supports structure migration, data migration and change data replication.

If you need migration without disabling services, select Migrate object structure, Migrate existing data and Replicate data changes at the same time.

If you only need migrating existing data, select Migrate object structure and Migrate existing data.

- Migration object

The migration object can be a database, a table or a column.

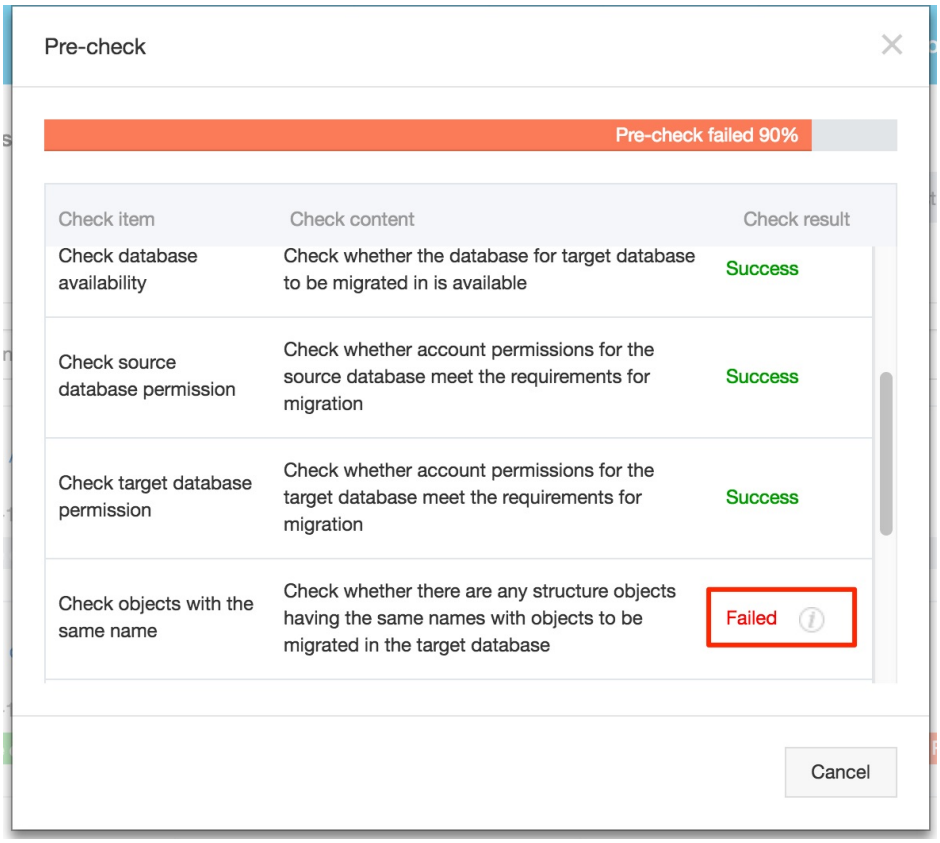
Select the object to be migrated: database, table or column.
By default, after the object is migrated to the RDS instance, its name remains the same with that in the local MySQL instance. Otherwise, you need to use the object name mapping function.

After you configure migration object and migration type, start pre-check.

4. Pre-check

Migration can be started only after pre-check succeeds.

If pre-check fails, click **Failure** next to the check item to view details, solve the problem and execute pre-check again.



5. Start the migration task

After pre-check succeeds, start the migration task. You can view the migration status and progress in the task list.

Replicating change datais a process of dynamic synchronization. So it is recommended that you verify the data on the target database when incremental migration succeeds with no latency. If the data is verified successfully, you can disable the migration task and switch to the target database.

So far, the data migration task configuration from a local MySQL database to the RDS for MySQL has been completed.

DTS can migrates data from a local SQL Server instance to a RDS for SQL Server instance with zero downtime.

This section introduces how to use DTS to configure a data migration task from a local SQL Server to a RDS for SQL Server.

Migration type

For data migration from a local SQL Server to a RDS for SQL Server, DTS supports migrating object structure, migrating existing data and replicating change data. The features and restrictions of migration type can be found below:

- **Migrating object Structure**

In this step, DTS migrates the structure definition of the migrated object to the target instance. Currently DTS supports such objects as tables, views, triggers, synonyms, SQL storage procedures, SQL functions, plan, user defined types, rules and defaults.

- **Migrating existing data**

In this step, DTS migrates all data stored in the SQL Server instance to the target RDS instance. However, the data newly added to the local SQL Server instance during migration will not be synchronized to the target RDS instance.

If you have also selected replicating change data, the data newly added to the local SQL Server instance during migration will be synchronized to the target RDS instance.

Migration restrictions

- DDL operations are not supported during migration.
- Structure migration does not support assemblies, database-level stored procedures, service brokers, full-text indexing, full-text directories, distributed schemas, distributed functions, CLR scalar functions, CLR value functions, inner tables, aggregate functions, and systems.
- If object name mapping is enabled, other objects associated with the migrating object may

fail to be migrated.
- Change data replication only supports the clustered tables with primary key.
- Change data replication does not support synchronization of update statements that only update big fields[A1].
- Change data replication does not support the tables containing computed columns
- An migration task with replicating change data supports migration of only one database at a time. If you want to replicate change data for multiple databases, you need to create multiple migration tasks.

Migration permission

When DTS is used for data migration from local SQL Server to RDS for SQL Server, the migration accounts of the source SQL Server instance and the target RDS for SQL Server instance have different permission requirements for migration types, as follows:

Migration Type	Migrating object structure	Migrating existing data	Replicating change data
Local SQL Server Instance	Select	Select	sysadmin
Target RDS Instance	Read/Write Permission	Read/Write Permission	Read/Write Permission

Migration process

To streamline the dependency between objects and improve migration success rates for data migration from the local SQL Server to the RDS for SQL Server, DTS defines the migration process of structure objects and data as follows:

1. Migrating the structure of the objects of Table, view, synonym, custom type, rule, default and plan.
2. Migrating existing data.
3. Migrating the structure of the objects of SQL stored procedures, SQL functions, triggers and foreign keys.
4. Replicating change data.

If you have not set up replicating change data for a migration task, after the migrating existing data is completed, the migration progress in the task list is **Structure migration 100%, data migration 100%**, and the migration status is **Migrating**. At this time, SQL stored procedures, SQL functions, triggers and foreign keys are migrated. Do not end the task manually, otherwise it may cause migration data loss.

Configure a migration task

The following describes in detail how to configure a DTS migration task to migrate data from the

local SQL Server to the RDS for SQL Server.

Create a database on RDS instance

During data migration, if the database to be migrated does not exist in the target RDS instance, DTS will automatically create one. But if the name of the database name does not comply with the RDS defined specification (that is, the name has a maximum of 64 characters starting with a letter and ending with a letter or digit, and consisting of lowercase letters, digits, underscores and strike-throughs), you need to manually create the database before configuring a migration task.

Create a migration account

When configuring a migration task, you need to provide the migration accounts of the local SQL Server instance and the target RDS for SQL Server instance. For permissions required for the migration accounts, see the **Migration permission** section above.

Other Preparations

To enable migration without disabling local services, you need to set the log format of local SQL Server to **full**. Otherwise, run the below two steps to set the log format of local SQL Server to full:

1. In source database, run
   alter database database_name set recovery_model_desc=' full' ,
   where **database_name** is the name of the database to be migrated.
2. To open full logs effectively, you need to back up the logs in the source database. In the source database, run
   BACKUP LOG database_name to DISK=backup_place WITH init,
   where **database_name** is the name of the database to be migrated, and **backup_place** is the place storing backup files.

Configure a migration task

When the database and migration accounts are both created, you can start configuring a migration task according to the following steps:

1. Log on to DTS Console, click Create Migration Task **on the top right of the console.**
2. Configure the connection information of the local SQL Server and the target RDS instance.

Configure the migration task name as well as the connection information of source instance and target instance. Where:

- **Task name**

By default, DTS generates a name for every task automatically. The task name is not required to be unique. You can modify it as needed. It is recommended to use a name reflecting specific services.

Source instance

Instance type: **On-permises databases**

- Instance region: Select the region closest to the source instance.
- Database type: **SQL Server**
- Host name or IP address: IP address for accessing the local SQL Server database. s.
- Port: listener port of the local SQL Server instance.
- Database account: access account of the local SQL Server instance.

Database password: access password of the above SQL Server access account.

Target RDS instance

Instance type: **RDS instance**

- RDS instance ID: ID of the target RDS instance. DTS supports RDS instances in classic networks and VPC networks.
- Database account: access account of the RDS instance.
- Database password: access password of the above database account.

3. Configure migration object and migration type

- Migration type

DTS supports Migrating object structure,migrating existing data and replicating data changes.

If you need to enable migration without disabling services, select migrating object structure, migrating existing data and replicating data changes at the same time.

If you only need migrating existing data, select migrating object structure and migrating existing data.

- **Migration object**

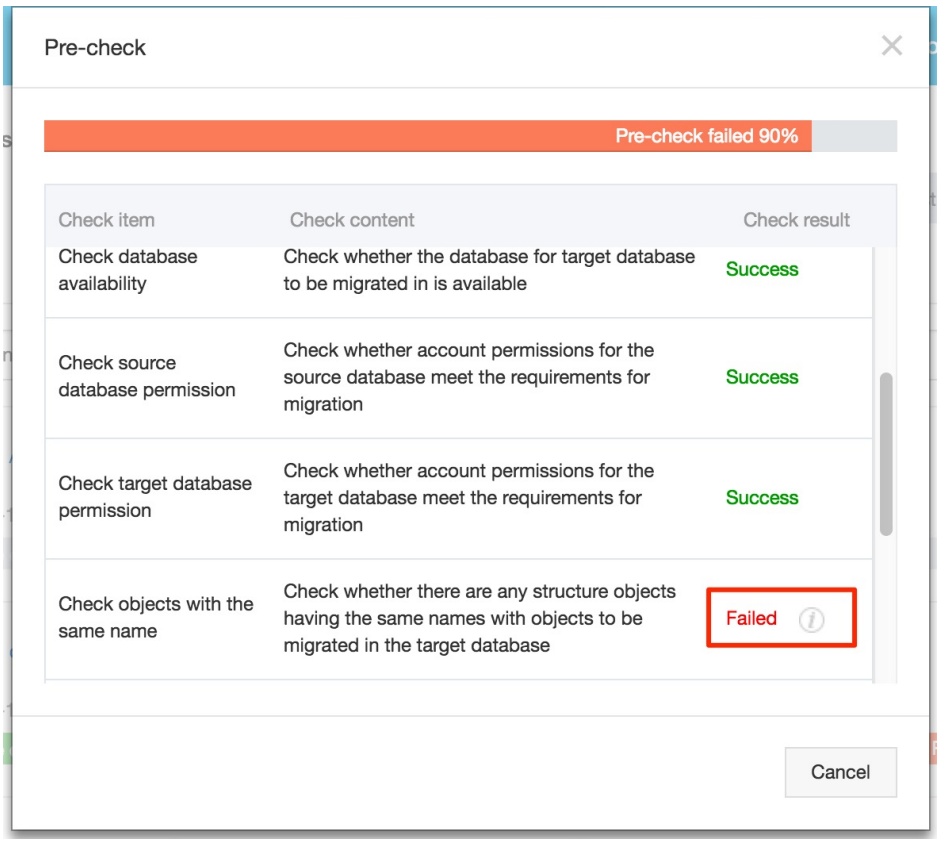
A migration object can be a database, a table or a column. By default, after an object is migrated to a RDS instance, the object’ s name remains the same as that in the local SQL Server instance. Otherwise, use the object name mapping function to maintain the same object name in both source SQL Server instance and the target RDS instance.

After you configure migration object and migration type, start pre-check.

4. Pre-check

Migration can be started only after pre-check succeeds.

If pre-check fails, click **Failure** next to the check item to view details, solve the problem and execute pre-check again.



5. Start the migration task

After pre-check succeeds, start the migration task. You can view the migration status and progress in the task list.

Change data replication is a process of dynamic synchronization. So it is recommended that you verify the data on the target database when incremental migration succeeds with no latency. If the data is verified successfully, you can disable the migration task and switch to the target database.

So far, the data migration task configuration from a local MySQL database to the RDS for MySQL has been completed.

DTS can help to migrate data from a local PostgreSQL instance to a RDS For PostgreSQL instance. PostgreSQL migration supports change data replication so that PostgreSQL database data migration can be fulfilled without stopping local application services.

This section briefs how to use DTS to configure data migration tasks from a local PostgreSQL to RDS for PostgreSQL.

Migration types

- Migrating object sturcture

DTS will migrate the structure definition of the migrated object to the target instance. For

PostgreSQL, DTS supports structure migration of objects including: **Table, trigger, view, sequence, function, user defined type, rule, domain, operation and aggregate**.

- Migrating existing data

DTS will migrate all the data of the migrated source database object to the target instance.

- Replicating change data

During change data replication process, update data of local PostgreSQL instances is synchronized to the target RDS instance, and the local PostgreSQL and target RDS instance enter the dynamic synchronization process.

Migration restrictions

- DTS supports source instances of PostgreSQL 9.2, 9.3 and 9.4 versions
- DDL operations are not supported during migration.
- C-language functions are not supported during migration
- If object name mapping is enabled, other objects dependent on this object may fail to be migrated.

Migration permission requirements

When DTS is used for PostgreSQL migration, the migration accounts of source and target databases must provide the following permissions if different migration types are involved:

Migration Type	Migrationg object structure	Migrating existing data	Replicating data change
Local PostgreSQL Instance	Usage Permissions of pg_catalog	Select of Migration Object	Superuser
Target RDS Instance	Create and Usage Permissions of Migration Object	Schema Owner	Schema Owner

Migration sequence

To streamline the dependency between objects and improve migration success rates for PostgreSQL > RDS For PostgreSQL data migration, DTS defines the migration sequence of structure objects and data as follows:

1. Migrate structure objects: Table, view, sequence, function, user defined type, rule, domain, operation and aggregate.
2. Migrating existing data
3. Migrate structure objects: trigger and foreign key

After the migrating existing data is complete, the migration progress in the task list is: Structure migration 100%, data migration 100%, and the migration status is “Migrating”. At this time, the migration task is migrating the object in Step (3). Please do not end the task manually, otherwise it may cause migration data loss.

Migration steps

The following describes in detail how to configure DTS migration tasks to migrate the local PostgreSQL database to RDS for PostgreSQL.

Create a database on RDS instance

In the data migration process, if the database to be migrated does not exist in the target RDS instance, DTS will automatically create one. But if the name of the database does not comply with RDS-defined Specifications (consisting of lowercase letters, digits, underscores, or strike-throughs, started with a letter and ended with a letter or digit. It has a maximum of 64 characters), you need to manually create the database before configuring the migration task.

Create a migration account

When configuring a migration task, you need to provide the migration account of the Oracle database and the target RDS instance. For permissions required for the migration account, see the Migration permission requirements section above.

If you haven’t created a migration account for your local PostgreSQL or RDS For PostgreSQL instances, you can refer to the following process to create one:

1. Through PostgreSQL client, you can create a migration account in PostgreSQL.

```
create user username password 'password'
```

If you want to use incremental migration, the created account must be superuser, so the account creation statement is adjusted to:

```
create user username with superuser password 'password'
```

Parameter description:

- username: the account to be created
- password: the account login password

2. To authorize the migration account, you can refer to the above table for permission requirements of migration accounts of local PostgreSQL and RDS For PostgreSQL.

```
GRANT privileges ON tablename TO username;
```

Parameter description:

- privileges: the operation permission of the account, such as SELECT, INSERT and UPDATE. If you want to authorize all the permissions to the account, you can use ALL.
- tablename: indicates the table name. If you want to authorize all the table permissions to the account, you can use wildcard character *.
- username: the account name to be authorized

Install logic flow replication plugins

If you want to replicating change data without stopping services, you need to, before task configuration, install the logic flow replication plugin provided by DTS in local PostgreSQL.

1. Download plug-in

Local PostgreSQL 9.4 Version

Local PostgreSQL 9.5 Version

2. Install the plugin

1. Unzip the downloaded zip package.
2. Copy the ali_decoding.so file to the lib directory in the PostgreSQL installation path.

If you want to use the rpm package for the installation, the absolute path is: /usr/pgsql-{$version}/lib/. In specific, $version indicates the version number. For example, if the version is 9.5, the absolute path will be: /usr/pgsql-9.5/lib.

3. Copy the ali_decoding.contorl file to the share/extension directory in the PostgreSQL installation path.

If you want to use the rpm package for the installation, the absolute path is: /usr/pgsql-${version}/share/extension/. In specific, ${version} indicates the PostgreSQL version number. For example, if the version is 9.5, the absolute path will be: /usr/pgsql-9.5/share/extension/.

4. Test whether the installation is successful.

You can log in to the PostgreSQL client with the superuser account and run the following SQL statements to see whether replication slot can be created successfully. If yes, it indicates the plugin has been successfully installed.

```
SELECT * FROM pg_create_logical_replication_slot('replication_slot_test', 'ali_decoding');
```

If the following output result is displayed, it indicates the plug-in has been successfully installed.

```
postgres=# SELECT * FROM pg_create_logical_replication_slot('replication_slot_test', 'ali_decoding');
       slot_name       | xlog_position
-----------------------+---------------
 replication_slot_test | 0/17D3F58
(1 row)
```

After the test is successful, delete the replication slot using the SQL statements below.

```
SELECT pg_drop_replication_slot('replication_slot_test');
```

Migration task configurations

After all of the above prerequisites are met, the data migration process can be started. Next, I will illustrate in detail the migration task configuration process.

1. Log in to DTS Console, click Create Migration Task on the top right of the console to start migration task configuration.
2. Configure the connection information between the local PostgreSQL and target RDS For PostgreSQL instance.

This step mainly serves to configure the migration task name and the connection information of PostgreSQL and target RDS For PostgreSQL instances. Where:

- **Task name**

DTS generates a name for every task automatically. The task name is not required to be unique. You can modify the name as needed. A name indicating the specific services of the task is recommended to facilitate task identification.

- **Source instance**
- Instance type: select **Self-built database with a public IP address**
- Database type: Select **PostgreSQL**
- Host name or IP address: Configure the address for accessing PostgreSQL. The address must be in public access mode.
- Port: the listener port of the local PostgreSQL instance
- Database name: the default database name connecting to the PostgreSQL
- Database account: the connection account of the local PostgreSQL instance

  Database password: the password of the connection account of the local PostgreSQL instance

  Target instance

- Instance type: to select RDS instance
- RDS instance ID: Configure the instance ID of the target RDS instance to be migrated. DTS supports RDS instances in classic networks and VPC networks
- Database name: the default database name connecting to the RDS For PostgreSQL
- Database account: the connection account of RDS For PostgreSQL instance

- Database password: the password of the connection account of the above RDS For PostgreSQL instance

After configuring the connection information, click “Authorize Whitelist and enter into the next step” to authorize entries in white lists. In this step, DTS will add DTS server IP address to the whitelist of the target RDS instances to avoid migration failures because of the whitelist mechanism of RDS instance.

3. Select the migration object and migration type.

- Migration type

For PostgreSQL -> RDS For PostgreSQL, DTS supports migrating object structure,migrating existing data and replicating change data.

If you only need migrating existing data, you can select migrating object structure and migrating existing data as the migration type.

If you need migration without stopping services, you can select migrating object structure + migrating existing data +replicating change data as the migration type.

- Migration object

Select the object to be migrated. The migration object can be a database, a table or a column. By default, after the object is migrated to RDS for PostgreSQL instance, the object name remains the same with that in the local PostgreSQL instance. If the object you migrate has different names on the source and target instances, you need to use the object name mapping function provided by DTS.

4. Pre-check

A pre-check will be performed before a migration task is formally started. Migration can be started only after the pre-check is passed.

If the pre-check fails, check the failure details by clicking the button after the specific check items, rectify the faults accordingly, and perform a pre-check again.

5. Start the migration task

After successful pre-check, the user can start the migration task. After the task is started, the user can view the specific migration status and progress in the task list.

So far, the data migration task configuration from a local PostgreSQL database to the RDS for PostgreSQL instance has been complete.

DTS data migration supports object name mapping, that is, it supports object (including databases, tables or columns) migration from the source instance to the target instance with a different name on the target instance.

This text introduces how to use the object name mapping feature during data migration task

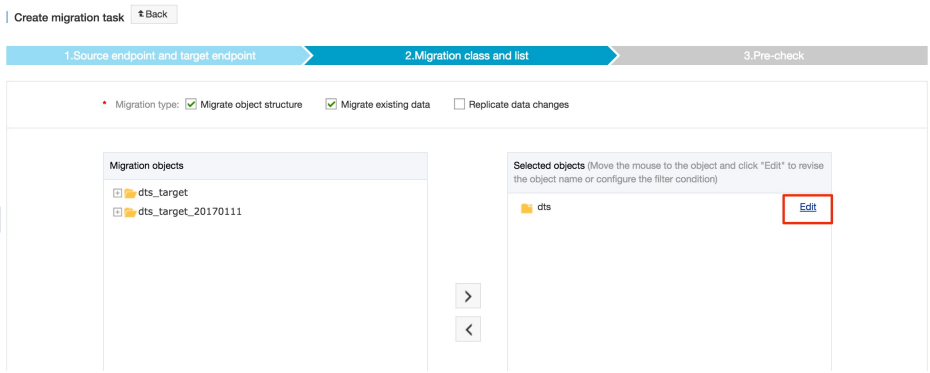
configuration.

Database name mapping

If names of the migrated databases are different in the source and target instances, you can map database names with the object name mapping feature provided by DTS.

Edit entry

You can configure the database name mapping feature in Step 2 **Select Migration Type and Object** of Configure Migration Task.In the **Selected** box, move the cursor to the line of the database object for database name mapping, and the **Edit** entry will be displayed on the right.1. In the **Selected** box, move the cursor to the line of the database object for database name mapping, and the **Edit** entry will be displayed on the right.

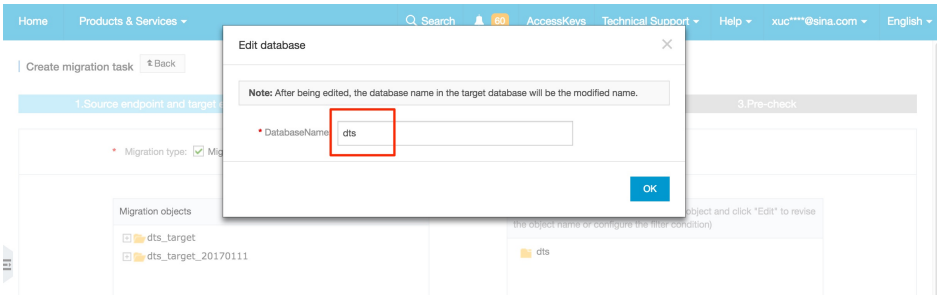


2. Modify the database name

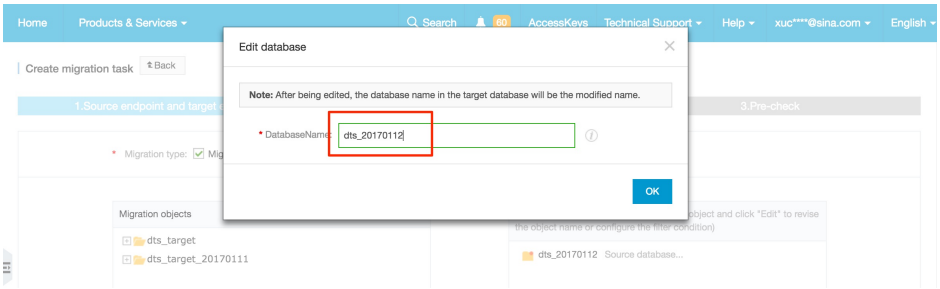
If the database name needs to be modified to dts_20170112 after it is migrated to the target instance, you can click Edit button to enter the database name modification page.

On the database name edit page, you can modify the database name directly. The modified name is the stored name of the database in the target instance.

Suppose before the modification, the database name is dts, as shown in the figure below:



As the database name needs to be modified to dts_20170112 after it is migrated to the target instance, you can directly modify the amptest on the interface to dts_20170112, as shown in the figure below:



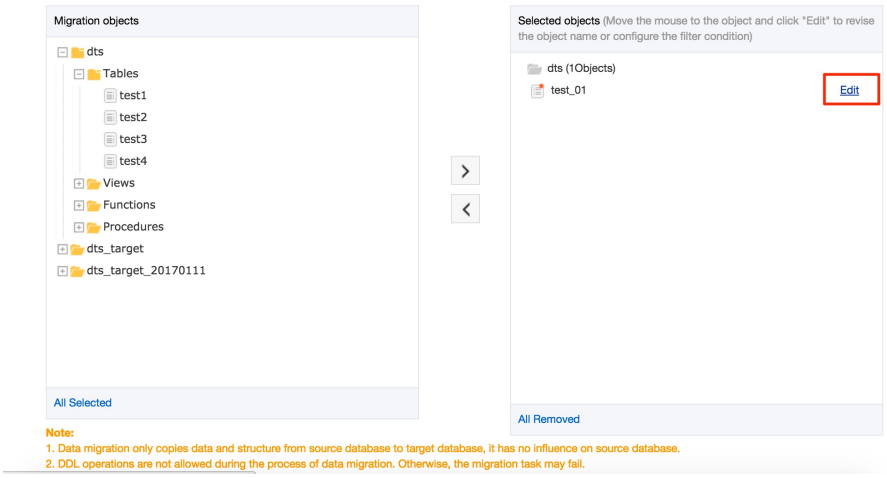
After the database name is modified, the database will use the modified name after it is migrated to the target instance.

Table name mapping

If names of the migrated tables are different in the source and target instances, you can map table names with the object name mapping feature provided by DTS.
If you want to use the table name mapping feature, you cannot select the entire database as the migration object. Instead, you must select specific tables
Besides tables, other structure objects (such as views, stored procedures, storage functions and synonyms) are also available for object name mapping in the same method.

Edit entry

You can configure the table name mapping feature in Step 2 **Select Migration Type and Object** of Configure Migration Task. In the **Selected** box, move the cursor to the line of the object for table name mapping, and the **Edit** entry will be displayed on the right.

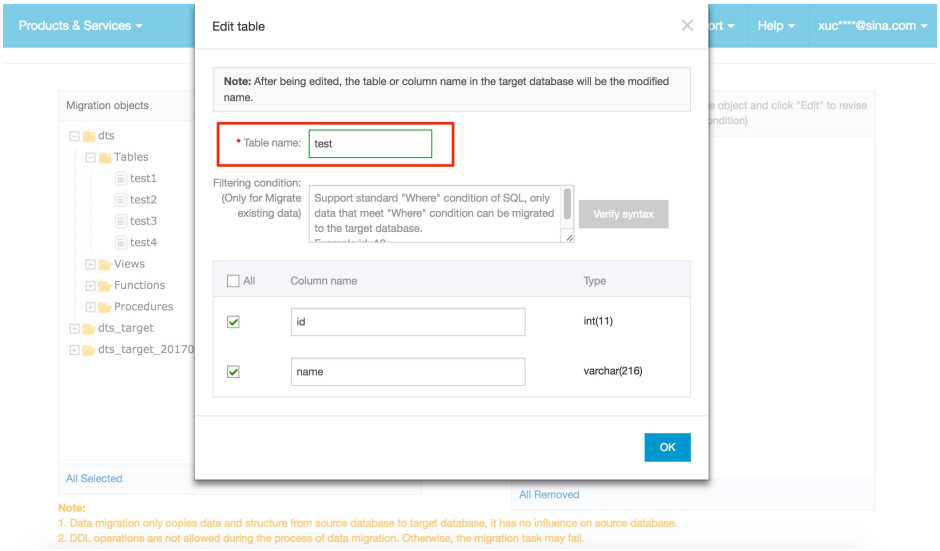


2. Modify table name

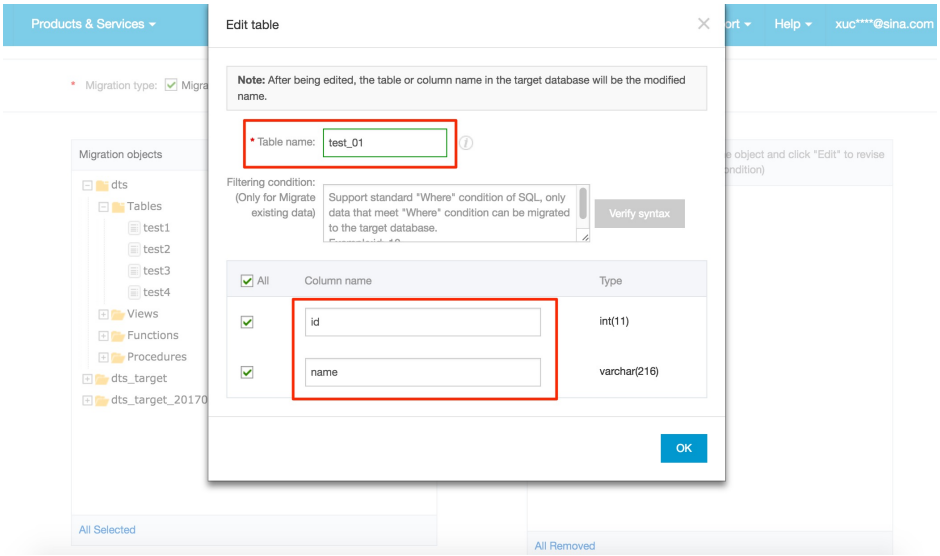
For example, if you want to migrate the Table test to test_01 in the target instance, you can click the **Edit** button after Table test to enter the table name modification page.

On the table edit page, you can modify the table name directly. The modified name is the stored name of the database in the target instance.

Suppose before the modification, the database name is test, as shown in the figure below:



As the table name needs to be modified to test_01 after it is migrated to the target instance, you can directly modify the test on the interface to test_01, as shown in the figure below:



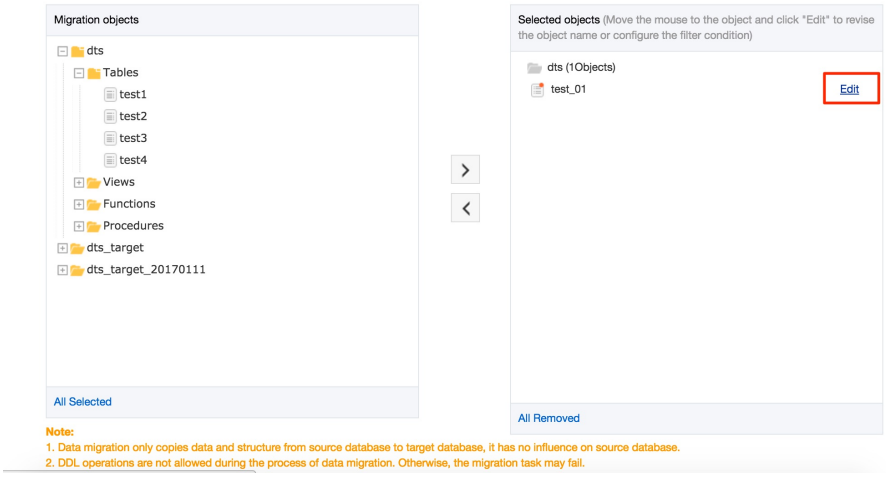
So far, the configuration of table name mapping has been complete.

Column name mapping

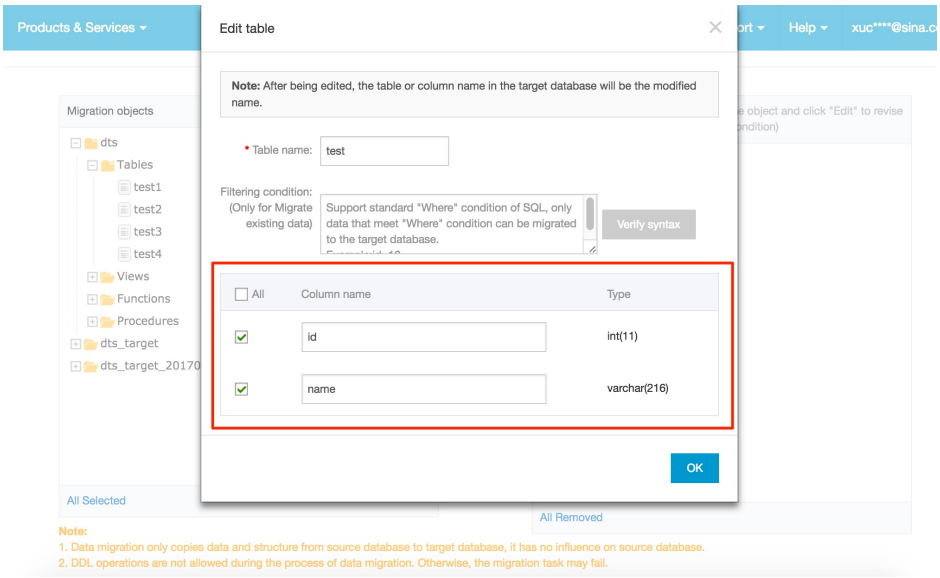
If column names of the migrated tables are different in the source and target instances, you can use the object name mapping feature provided by DTS.

You can configure the column name mapping feature in Step 2 **"Select Migration Type and Object"** of Create Migration Task. If you want to modify the name of the column to be migrated, you must select the table of the column in migration object selection, instead of selecting the database directly. Next, I will briefly introduce how to modify the column name corresponding to the migrated column.

For example, if you want to modify the table corresponding to the column to test, move your cursor to the row of Table test in the selected objects box, an **Edit** button will appear on the right.



Click Edit button to enter the table editing interface.



In this step, you can directly edit the desired column name. After the editing, the column name stored in the target database will be the modified column name.

So far, the configuration of column name mapping has been complete.

DTS supports filtering data for migration in the table through configuring SQL Where conditions. The SQL Where condition only applies to the configured table. DTS can filter the data in the table of the source database according to this SQL Where condition, and only the data meeting this condition can be migrated to the target database. This feature can be used for multiple scenarios including regular change data migration and table splitting. This section briefly introduces how to configure SQL Where conditions for filtering migration data when creating a migration task.

Functional limitations

SQL filtration condition only applies to full migration stage. If you have chosen replication change data as the task type, the SQL filtration condition will not apply.

Configure SQL filtration condition

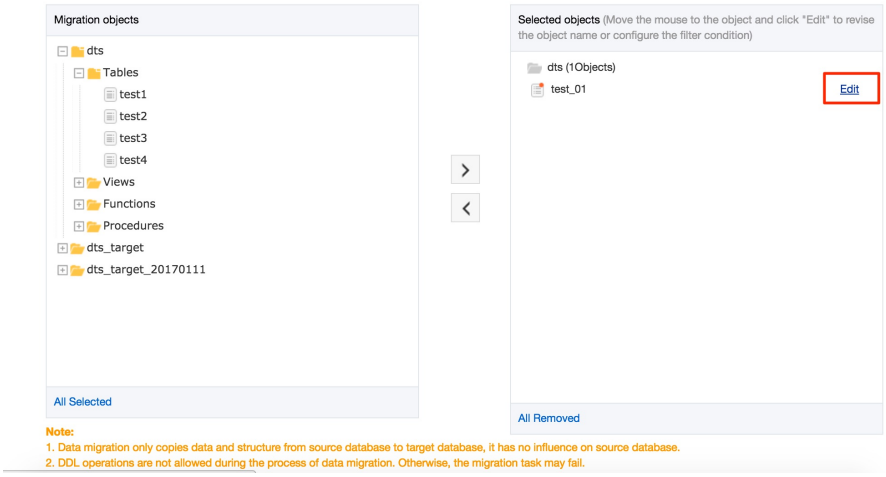
You can configure SQL filtration condition in the **Select Migration Type and List** step in the migration task configuration.

To configure SQL filtration condition for table migration, a table must be selected rather than the whole database when you select the migration object. The following is a brief introduction of the method for SQL filtration condition configuration.

Configuration entry

To configure a table for filtration condition, in the selected objects box of **Select Migration Type and List** step, move the cursor to the target table, and an **Edit** button will appear.

Click the **Edit** button to enter the filtration condition configuration page.

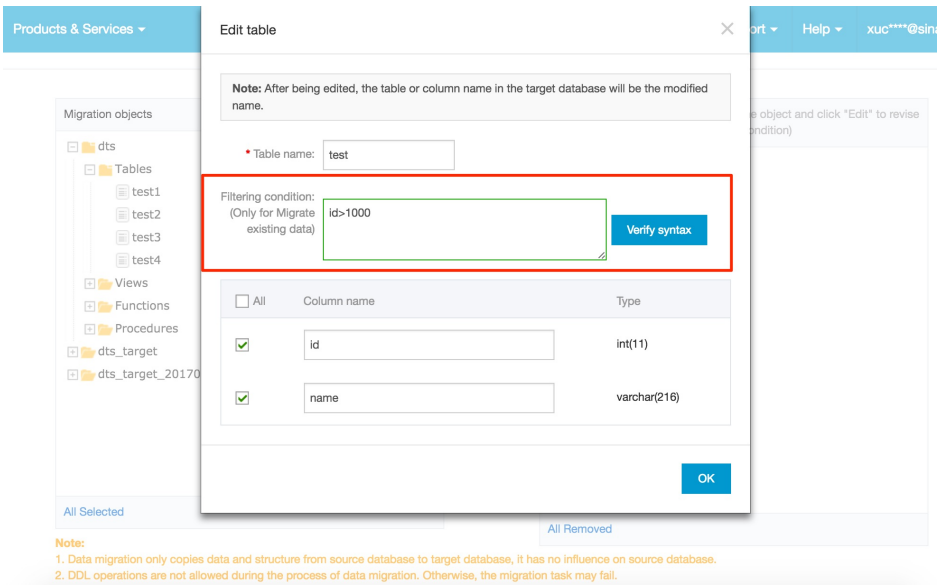


For example, if you want to configure SQL fitering condition for table test1, move your cursor to the row of Table test in the selected objects box, an **Edit** button will appear on the right.

Modify SQL filtration condition

DTS filtration condition, which is the same as the standard SQL WHERE condition of databases, supports calculation and simple functions.

Fill SQL filtration condition in the filtering condition box according to the requirement. For example, if we need to migrate the records of id>1000 in a table to a target instance, the specific configuration is as follows:



So far, the configuration process of data filtration conditions is complete.

Data Synchronization

This section introduces how to use DTS to quickly create real-time synchronization jobs between two RDS (MySQL) instances to achieve real-time synchronization of RDS change data.

Supported functions

- Support data synchronization between two RDS MySQL instances under Alibaba Cloud accounts.
- Support data synchronization between RDS MySQL instances under different Alibaba Cloud accounts.

Synchronization restrictions

Data source

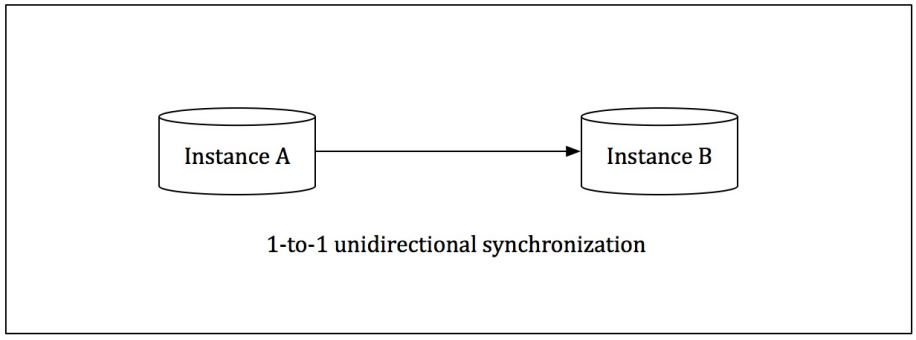
- At present, real time synchronization only supports RDS MySQL instances.
- The target instance does not support RDS instances in standard access mode and with internet connection addresses only.
- Real time synchronization for RDS instances is not supported in Hong Kong Zone A.

Synchronization architecture

At present, real time synchronization only supports three types of synchronization architecture as

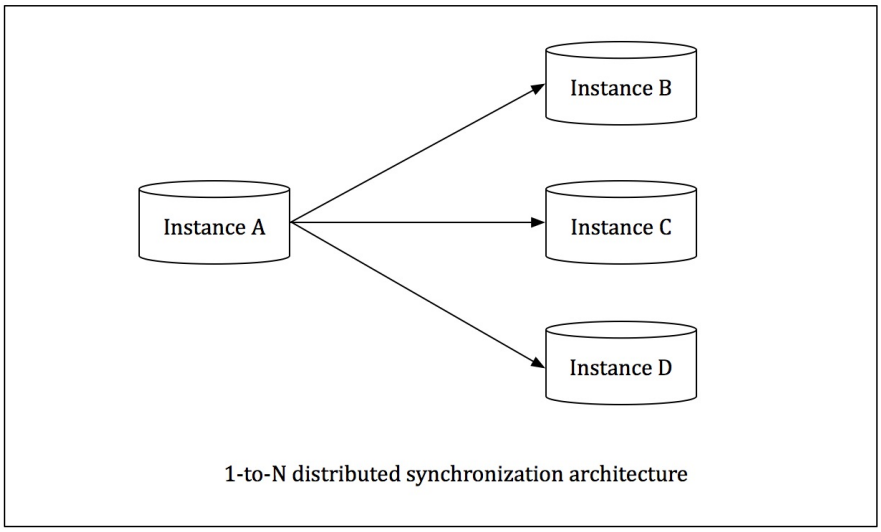
below:

- A->B, that is unidirectional synchronization between two instances. And the synchronized object on B must be read-only, otherwise it may cause synchronization instance exceptions.



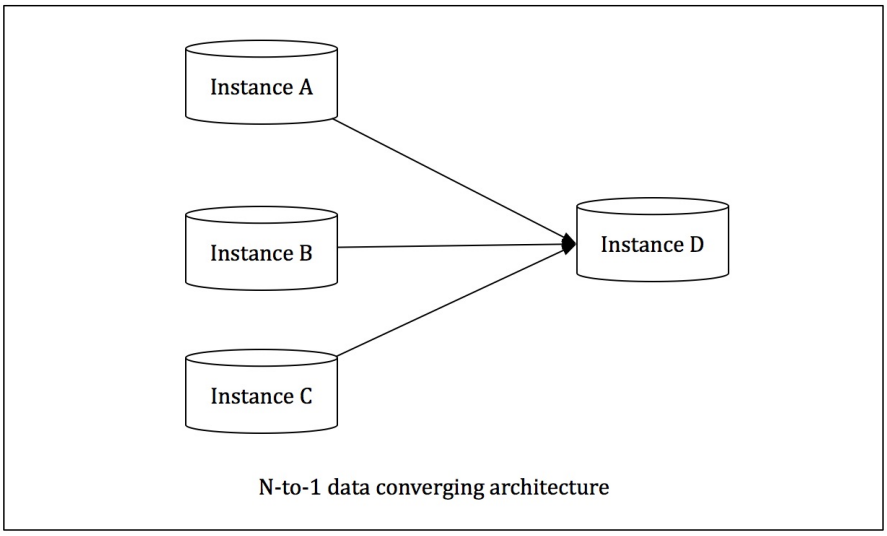
1-to-1 unidirectional synchronization

- A->B/C/D, that is 1-to-N distributed synchronization architecture. The number of target RDS instances is not limited, but the synchronized objects on the target instance must be read-only. Otherwise it may cause synchronization instance exceptions.



1-to-N distributed synchronization architecture

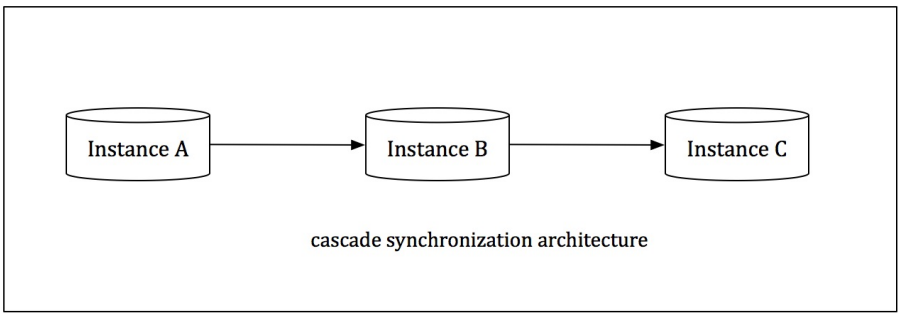
- B/C/D->A, that is N-to-1 data converging architecture. For N-to-1 architecture, the synchronized objects for every synchronization instance must be different to ensure full synchronization.



N-to-1 data converging architecture

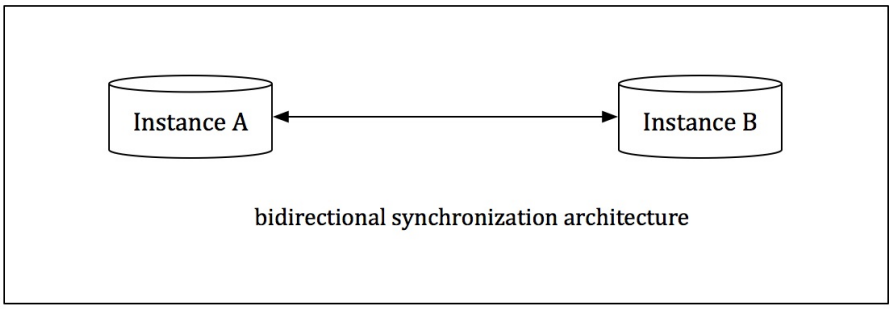
The following architecture is not supported:

- A->B->C, that is, cascade architecture.



cascade synchronization architecture

- A->B->A, that is, bidirectional synchronization between Instance A and Instance B.



bidirectional synchronization architecture

If an unsupported architecture is set when you configure a synchronization instance, checking **Complex Topology** will fail in pre-check.

Incompatible trigger

When the object to be synchronized is an entire database and the database contains a trigger to update the content of synchronizing table, synchronization data may be inconsistent.

If there is a trigger on A. The body of the trigger is After you insert a row of data into Table A, It will

insert a row of data into Table B. In this case, during data synchronization, if you insert a row of data into table A,there will be two row inserted into table B in the target instance .

To solve this problem, you can delete the trigger in the target instance. Data change in Table B is synchronized by DTS.

Configuration steps

Below shows how to create a synchronization job between two RDS instances.

1. Purchase a synchronization instance

(1) Log in to DTS Console and enter the data synchronization page.
(2) Click **Create Synchronization Task** on the top right of the console.
(3) Purchase a synchronization instance before configurations. Synchronization instances now support subscription and pay-as-you-go payment modes. You can select the one you need.

Where:

- **Source region**

The source region is the region where the source RDS instance is located.

- **Target region**

The target region is the region where the target RDS instance is located.

- **Instance specification**

The instance specification affects the performance of synchronization instance. The relationship between instance type and performance can be found in **Data Synchronization Specifications**.

- **Quantity**

The quantity is the number of synchronization instances purchased a time. If you have a pay-as-you-go instance, you can purchase a maximum of 99 instances at a time.

After the synchronization instance is purchased, return to the DTS Console, click **Configure Synchronization instance** on the right of the newly purchased instance to start configurations.

2. Configure synchronization instance connection information

Where:

- **Synchronization job name**
  The synchronization job name is not required to be unique. A name indicating specific services is recommended to facilitate instance identification and management.
  -**RDS instance ID for synchronization instance**

After completing configuration, click **Authorize Whitelist and enter into next step**.

3. Update whitelist for the RDS instance

Add the IP address of the DTS server to the whitelist of the RDS instance for synchronization. This way, you can avoid a failure to create a synchronization job because the DTS server fails to connect to the RDS because of the RDS whitelist.
To ensure the synchronization job stability, do not delete these server IP addresses from the whitelist of RDS instances.

After whitelist authorization, click **Next** to create a synchronization account.

4. Create a synchronization account on the target database

Create a synchronization account on the target RDS instance named **dtssyncwriter**. You cannot delete this account during synchronization. Otherwise, it may cause synchronization job interruption.

5. Select a synchronization object

After a synchronization account is created on the target RDS instance, you can select the synchronization object. The granularity of objects to be synchronized in real time can be subdivided into tables, that is, you can select to synchronize some databases or several tables.

If you select to synchronize the entire database, all the DDL operations in the database (such as create table and drop view) will be synchronized to the target database.

If you select to synchronize a table, only drop/alter/truncate/rename table and create/drop index operations in these tables will be synchronized to the target database.

Note: the rename table operation may cause inconsistent synchronization data. For example, if the synchronization object only includes Table A, instead of Table B and if the source instance executes **rename A to B** operation during synchronization, the operations to the renamed Table B won’ t be synchronized to the target database. To solve this problem, you can choose to synchronize the entire database of Table A and Table B.

After the synchronization object is selected, you can configure the synchronization initialization.

6. Configure synchronize initialization

Initialization is the first step to start a synchronization instance. It will initialize the structure and data of the existing synchronization objects in the source instance on the target instance, use them as the baseline data for subsequent data replication.

Synchronization initialization supports two modes: structural initialization and data initialization. By default, you need to select structural initialization and data initialization.

7. Pre-check

Once all above configuration are completed, pre-check is initiated.

After a synchronization job is configured, DTS checks the limited items. After pre-check succeeds, click **Start** to start the synchronization task.

After a synchronization job started, the synchronization job list is displayed. The just-started task is in **Synchronization Initialization** state. The initialization period depends on the size of the synchronized objects on the source instance. When the initialization is completed, the synchronization instance enters the **Synchronizing** state and a data synchronization is established between the source and target instances.

DTS supports dynamic modification of synchronization objects during the synchronization process. This section briefly introduces how to add synchronization objects during the synchronization process.

Modify limits

Only when the synchronization job is in **Synchronization in process** or **Synchronization failed** status can the synchronization object be modified.

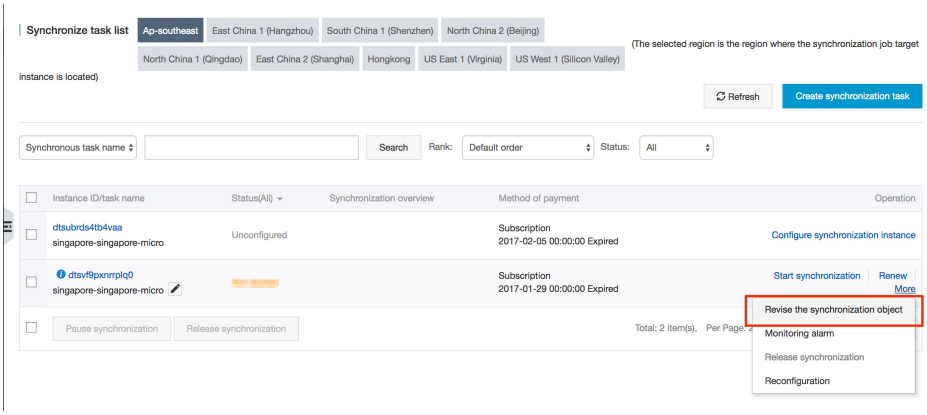
Synchronization start time

When a new synchronization object is added, the synchronization start time varies according to whether the synchronization job is subject to synchronization initialization. The details are as follows:

- If the synchronization job is subject to synchronization initialization, the newly added object will also perform synchronization initialization and then start the data replication.
- If the synchronization job is not subject to synchronization initialization, the synchronization object will start to replication change data generated on the source instance from the current time point.

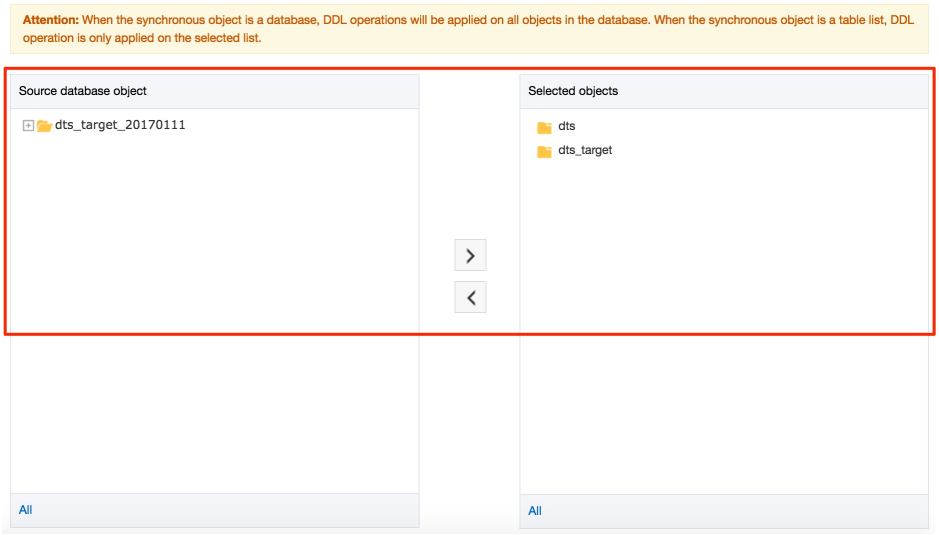
Modification steps

1. Modify entry to the synchronization object



To modify synchronization jobs of the synchronization objects, click **revise the synchronization object** in **More** on the right of the data synchronization instance to modify the synchronization object.

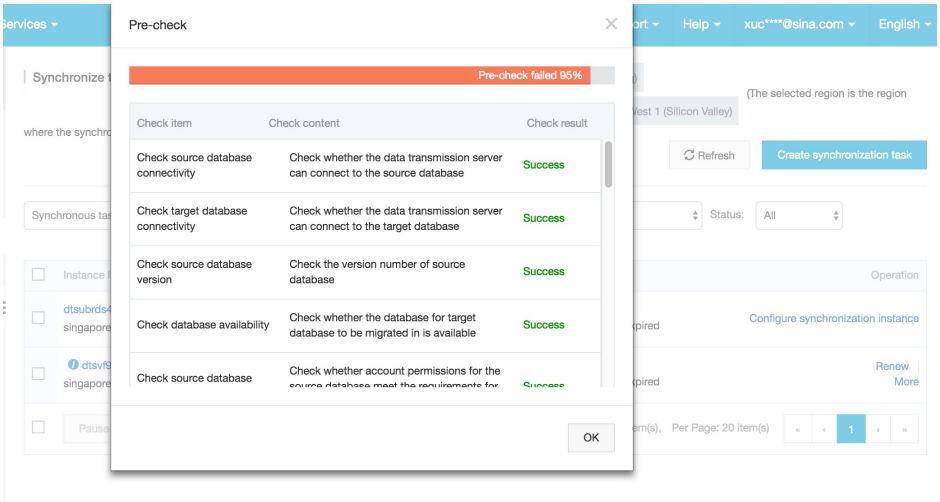
2. Add synchronization object



On the interface for selecting the synchronization object to be modified, you can add the objects to be synchronized as needed.

After the modification to the object is complete, the pre-check is started.

3. Pre-check



After the pre-check is passed, you can click “Start” to complete adding synchronization objects.

After the synchronization object is added, if the synchronization job needs initialization, the task status will change from the original **Synchronizing** to **Synchronizing(Newly added object in initialization Details)**. At this time, the background will restart the synchronization job and the synchronization latency will be changed to -1s. After the restart is successful, the latency and synchronization speed will return to normal.

In the synchronization job list, you can click Details to view the progress details of the initialization of the newly added object. After the newly added synchronization object completes the initialization, the synchronization job status returns to **Synchronizing**.

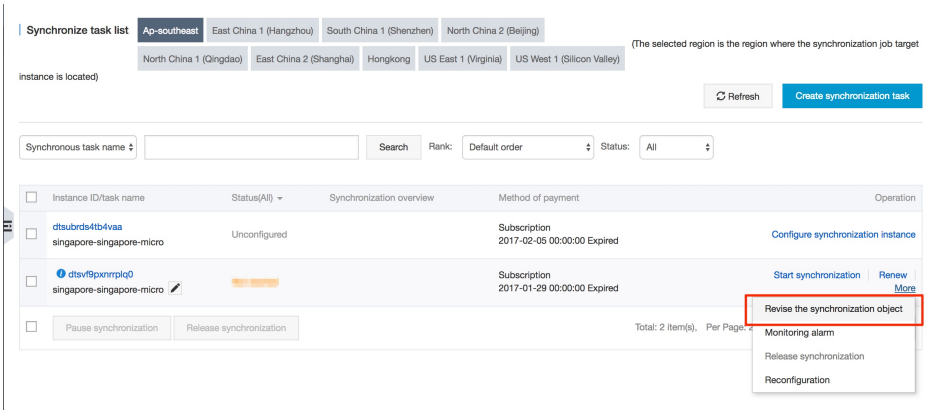
DTS supports dynamic modification of synchronization objects during the synchronization process. This section briefly introduces how to reduce synchronization objects during the synchronization process.

Modify limits

Only when the synchronization job is in **Synchronizing** or **Abnormal** status can the synchronization object be modified.

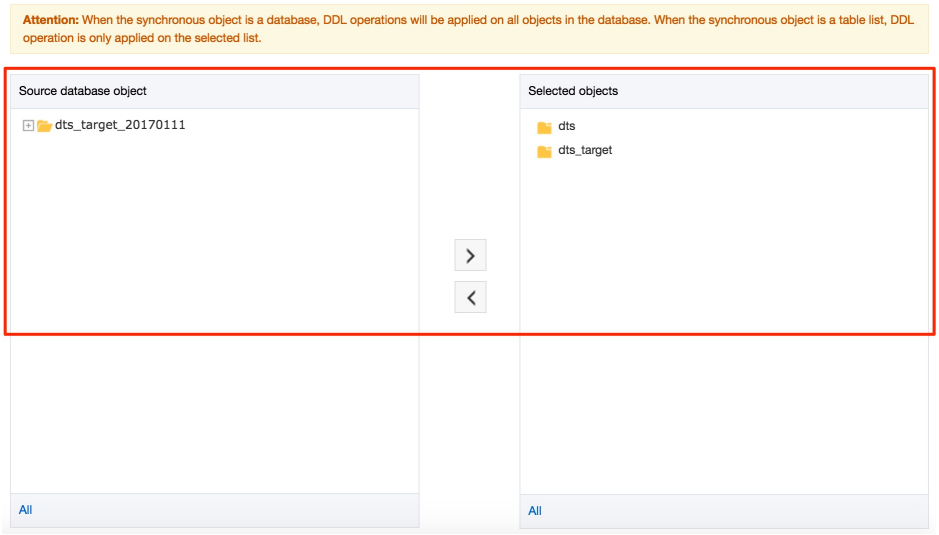
Modification steps

- **Modify entry to the synchronization object**



To modify synchronization jobs of the synchronization objects, click **revise the synchronization object** in **More** on the right of the data synchronization instance to modify the synchronization object.

- **reduce synchronization object**



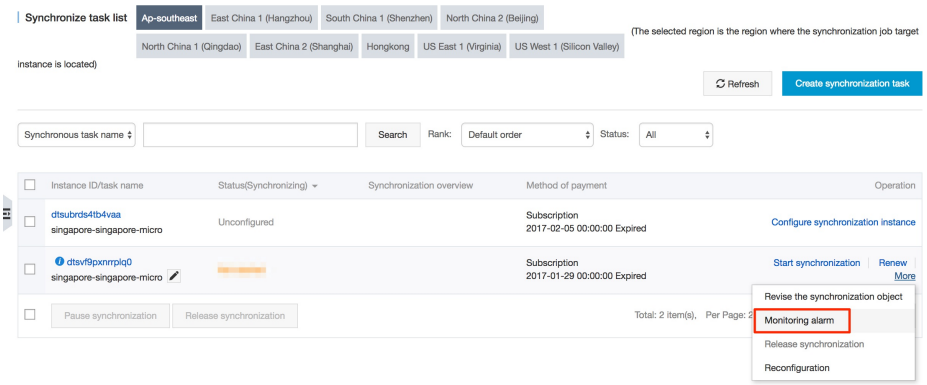
On the interface for selecting the synchronization object to be modified, you can reduce the objects to be synchronized as needed.

So far, the configuration for reducing the synchronization object has been complete.

DTS provides the alarm monitoring feature to monitor the synchronization job status and synchronization latency. You can configure the alarm monitoring feature for synchronization jobs in any status. This section briefly introduces how to configure monitoring alarms for synchronization job status and latency on DTS Console.

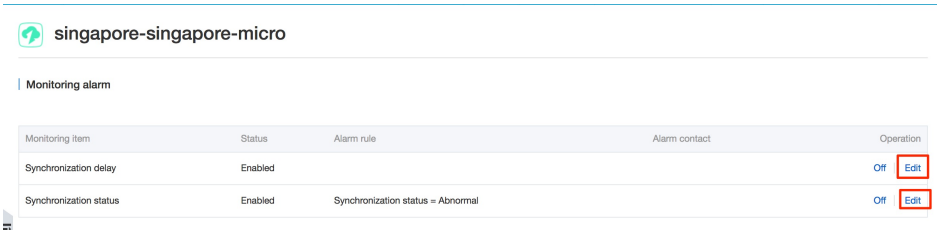
Configuration steps

- **Entry to monitoring alarms**



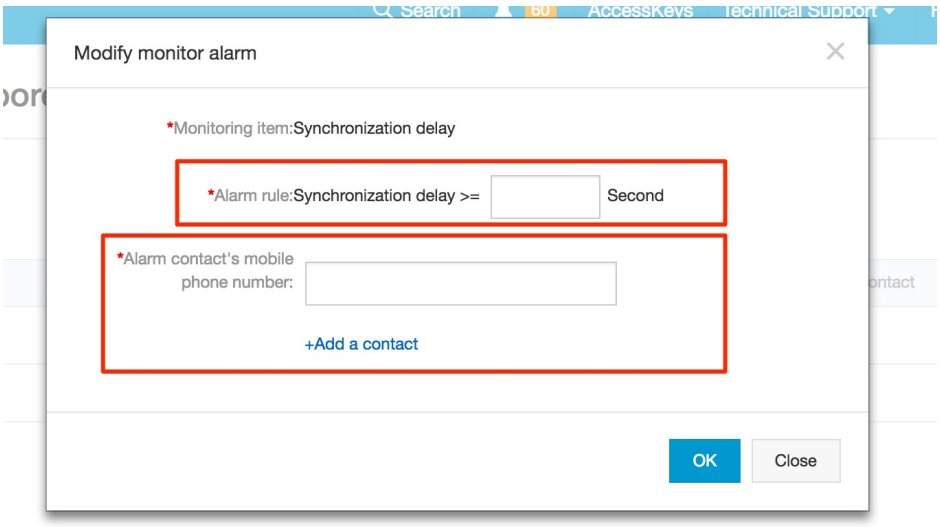
To configure monitoring alarms for synchronization jobs, you can click **Monitoring Alarm** in More on the right of the job to start the configuration.

- **Entry to modifying alarming policies and contacts**



By default, the monitoring items are enabled. If you want to modify the alarm triggering condition and phone contacts, you can click the **Edit** entry after each alarm item to modify them.

- **Modify alarming policies and contacts**

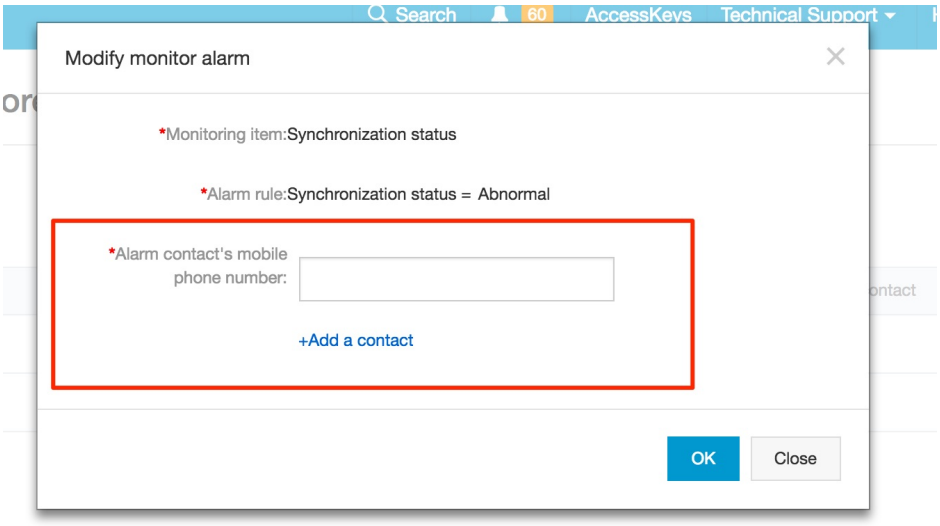


In this diagram, you can modify the latency threshold value of synchronization latency and phone contacts of latency alarms.

Alarm rules: synchronization latency > 10s. It means only when the synchronization job latency is longer than 10s will DTS send alarm messages to the phone contacts of latency alarms. You can configure the latency threshold value based on your services' sensitivity on latency.

DTS supports sending alarm messages to multiple phone contacts and you can add alarm contacts as needed.

You can not modify the monitoring rules when the monitoring items are synchronization status. Only when the synchronization job status is abnormal will the alarm messages be sent. The exceptions of synchronization jobs include: Synchronization initialization failed, synchronization failed and newly added object initialization failed.

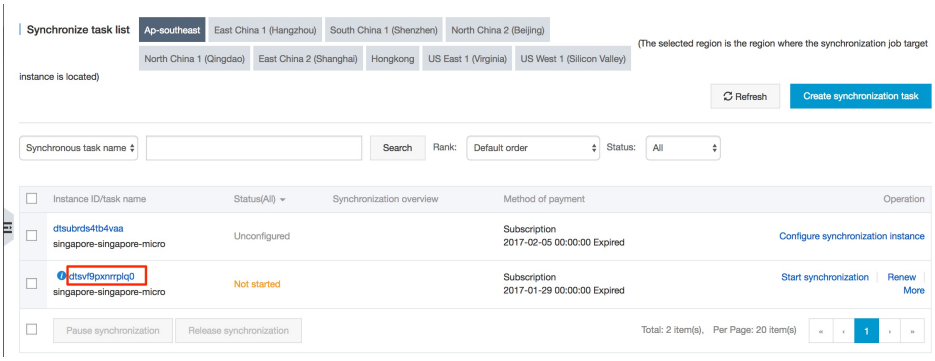


After the configuration, once the monitoring item triggers the alarm rules, DTS will automatically send alarm messages to alarm contacts.

So far, the configuration of monitoring alarms has been complete.

DTS provides three performance trend diagrams, namely for the synchronization latency, synchronization data traffic and synchronization Speed(TPS). You can view the running performance and status of the synchronization jobs in real time. This section briefly introduces how to view the synchronization performance and status.

- **Entry to view synchronization performance**



To view the performance of a synchronization job, you only need to click the synchronization job ID in the synchronization job list to enter the details page of the job. On the details page, click the synchronization performance on the left to view the synchronization performance .

- **Data synchronization performance trend diagram**

DTS provides three performance trend diagrams, namely for the synchronization latency, synchronization speed(TPS) and synchronization data traffic.

- Synchronization data traffic: the data traffic per second that the data writing module pulls from the data pulling module of the DTS, and the unit is MB/s.
- Synchronization Speed(TPS): the number of records per second that the DTS synchronizes to the target RDS instance.
- Synchronization latency: the time difference between the timestamp of the latest synchronized data on the target RDS instance and the current timestamp on the source RDS instance, and the unit is seconds.

Data Subscription

DTS provides an RDS MySQL binlog subscription feature. To subscribe and consump to the binlog of RDS instance, follow the two steps:

1. On the DTS Console, create the data subscription instance for RDS instances.
2. Use the SDK provided by DTS to connect to the subscription instance for subscription to and consumption of the change data.

This section mainly briefs the process for creating data subscription instances on the DTS Console.

Instance creation process

1. Purchase a subscription instance

Log in to DTS Console and enter the data subscription page.

Click the "**Create data subscription**" at the top right corner of the Console to start configuring the data subscription instance.

Purchase a data subscription instance before instance configurations. Data subscription now supports subscription and pay-as-you-go payment modes. You can select the one you need.

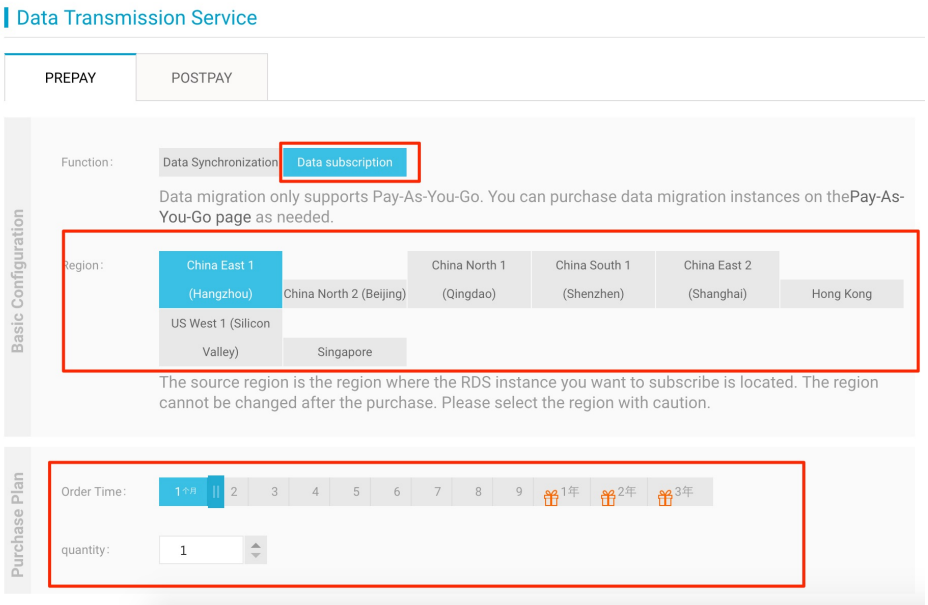
Parameters to be configured on the purchase page include:

- **Source instance region**

The source region is the region of the RDS instance of the data subscription instance.

- **Quantity**

The quantity is the number of subscription channels purchased at a time. If you have a pay-as-you-go instance, you can purchase 99 instances at most at a time.



After the data subscription instance is purchased, return to the DTS Console, click "**Configure Subscription Channel**" on the right of the newly purchased instance to start configurations.

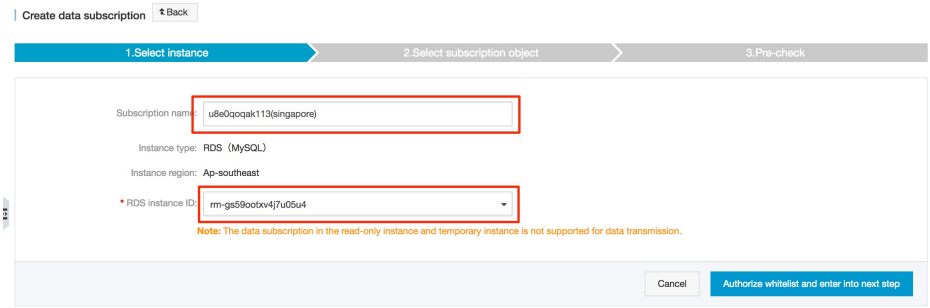
2. Configure RDS instance ID

- **Subscription name**

The subscription name is not required to be unique. For the convenience of identifying specific instances, a name indicating the specific services of the instances is recommended to facilitate instance identification and management.

- **RDS instance ID**

Select the RDS instance you want to subscribe to, and only the RDS For MySQL instances under the Alibaba Cloud account are shown in the drop-down menu.



After completing content configurations, click **Authorize Whitelist and enter into next step**.

3. Update whitelist for the RDS instance

This step serves to add the IP address of the DTS servers to the whitelist of the RDS instance for subscription. In this way, you can avoid subscription instance creation failures because the DTS servers fails to connect to the RDS because of the RDS whitelist.

To ensure subscription instance stability, do not delete these servers IP addresses from the whitelists of RDS instances during subscription.

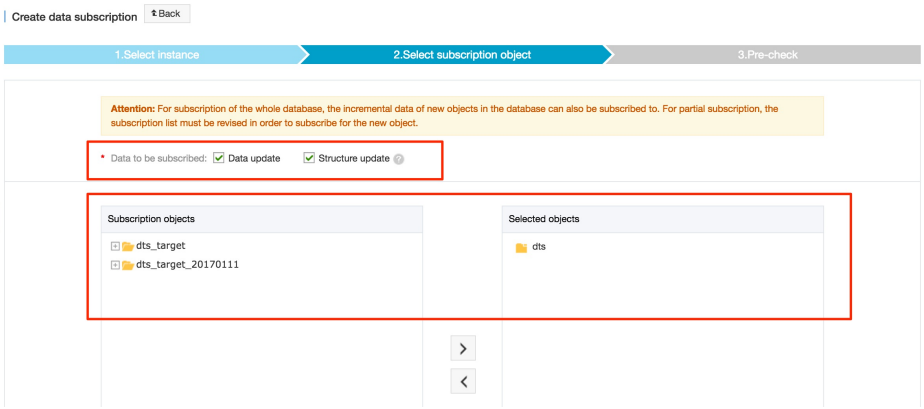
After whitelist authorization, click Next to create the subscription account.

4. Select a subscription object

In this step, you need to choose the objects to be subscribed to.

DTS subscription objects can be further divided into databases and tables. You can choose to subscribe to some databases or several tables.

DTS sub-divides subscription data types into **Data update(DML)** and **Structure update(DDL)**. If subscription object and **Data update** is selected, only three kinds of data updates, namely insert, delete and update, can be subscribed to. If you want to subscribe to structure update (DDL), select the Structure update in the subscription data type. Once you subscribe to Structure update, DTS will pull all the structural changes on the entire RDS instance. You need to filter the data needed using the SDK.



Once subscription object is selected, you will enter the pre-check stage before the start of the subscription instance.

5. Pre-check

When the subscription instance configuration is completed, DTS will perform the limitation pre-check. After the pre-check is passed, you can click **Subscribe** to start the subscription instance.

6. Start subscription

After the subscription instance is created, it will enter the initialization stage. This stage will last for about 1 minute. After initialization, you can View Subscription Data Online, or Subscribe to change Data Using the SDK.

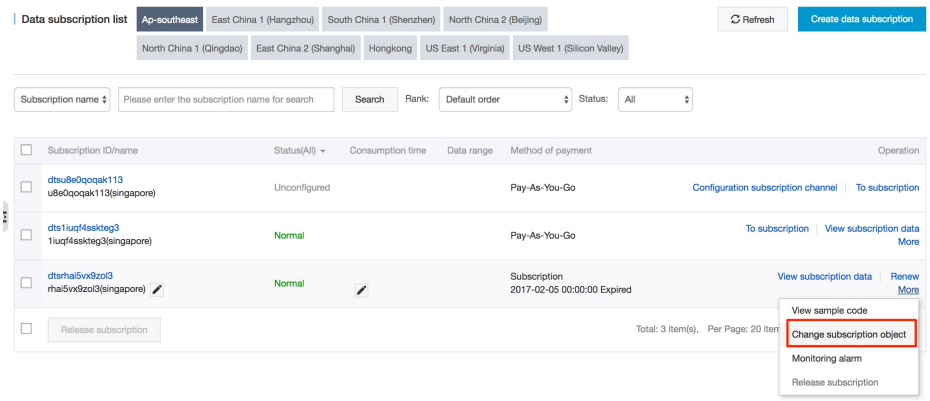
DTS can dynamically add/remove subscription objects in the subscription consumption process. If you have added a subscription object, the subscription instance will pull change data of the new subscription object from the current time after the modification is completed. If you have removed a subscription object, SDK will no longer be able to subscribe to the data of the object after the modification is completed.

This section introduces how to change subscription objects in the console.

Operating steps

1. Entry for modifying subscription objects

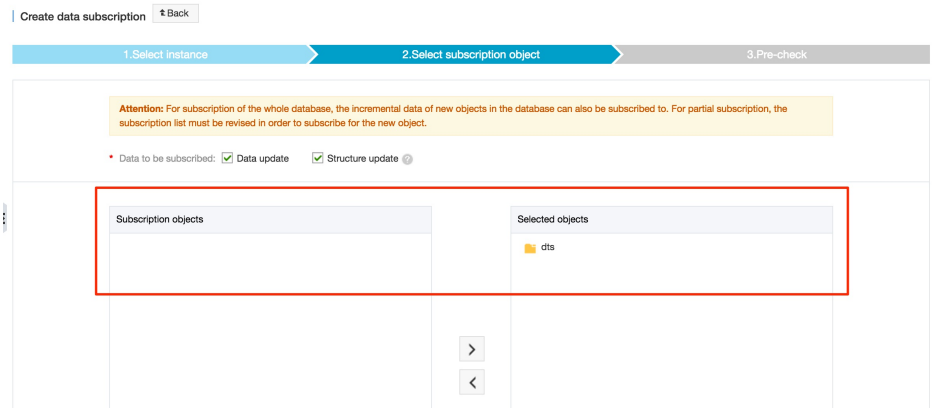
DTS only supports modifying subscription objects in the console and the entry is as follows:



For the subscription instance of the subscription object to be changed, click **More** on the rightmost side and pull down for more operations, one of which is the entry for **Change Subscription Object**.

2. Modify subscription object

Click the entry of "**Change Subscription Object**" and you will be directed to the page of subscription object selection.



On this page, you can add and remove subscription objects, or modify the data type subscribed to. When the change is completed, you will re-enter the pre-check stage.

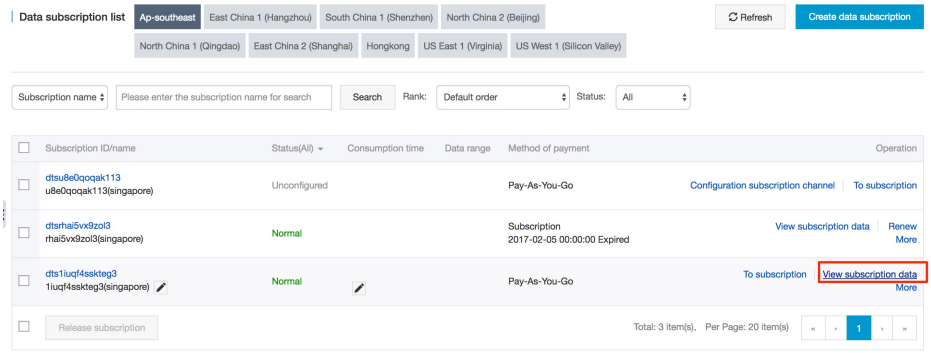
When the pre-check is passed, you can click to subscribe. At this time, the subscription instance will be re-initialized.

When the initialization is completed, the subscription instance will work normally. You can view the subscription data or use the SDK to subscribe to change data.

DTS supports view of subscription data in the DTS console, that is, you can view the change data of the subscription instance on DTS Console. The time range of data available for the view is the data range of the subscription instance.
This section introduces the methods to view subscription data and the meaning of each field in the data record.

Entry for viewing subscription data

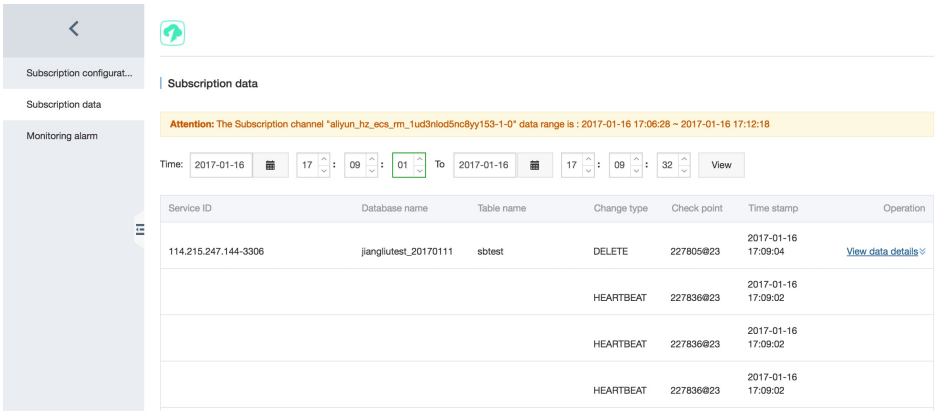
The entry for viewing subscription data on DTS Console is as follows:



Subscription data viewing only applies to subscription channels in the normal and abnormal states. Subscription channels in the initialization and pre-check states do not offer subscription data views.

View subscription data

When the "**View Subscription Data**" option is clicked, the page for viewing subscription data will be prompted, as follows:



You can view change data of a specific time range according to the configured time range. The configured period must be within the time range of the subscription instance.

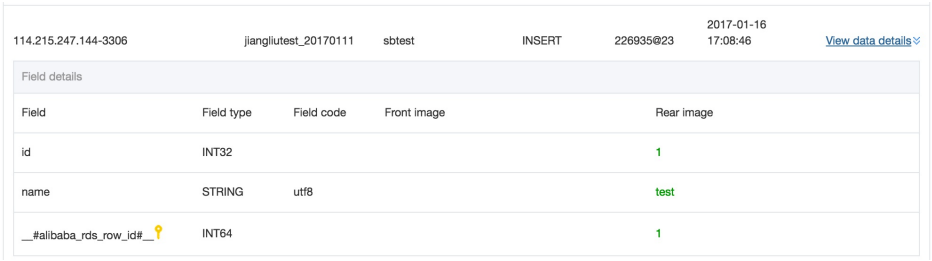
Subscription data field meanings

The subscription data format displayed is in an internal data format of DTS. The meaning of each field in the data record is explained as follows:

- Service ID: The IP:PORT of RDS instance
- Database name: The name of the database where the table corresponding to the change log belongs to.
- Table name: This name of the table corresponding to the change log.
- Change type: the change type of the change log, including : update, delete, insert, replace, begin, commit, ddl, dml and heartbeat.

Heartbeat is the heartbeat record provided by DTS, with each record for each second. This is mainly used to show the subscription instance health condition. If there is one heartbeat every second, the subscription instance is healthy. Otherwise, the subscription channel is abnormal.

- Checkpoint: The checkpoint of this change log in binlog, in the format of binlog_file_offset@binlog_file_id. In specific, binlog_file_id is the postfix of the binlog file name, and binlog_file_offset is the offset of this binlog log in the binlog file.
- Timestamp: The timestamp at which this change log is written into the binlog file.
- Field name: The name of each field corresponding to this change log.
- Field type: The data type of each field corresponding to this change log.
- Field encoding: The character set encoding of the corresponding change log.
- Front image: The values of various fields before changing. There is no pre-image values for the insert operation.

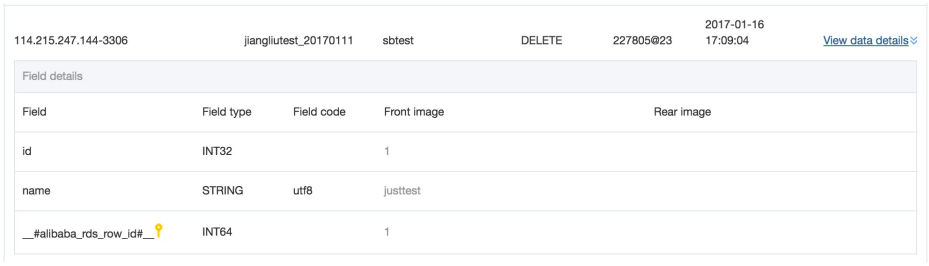


114.215.247.144-3306	jiangliutest_20170111	sbtest	INSERT	226935@23	2017-01-16 17:08:46	View data details

Field details

Field	Field type	Field code	Front image	Rear image
id	INT32			1
name	STRING	utf8		test
__#alibaba_rds_row_id#__	INT64			1

- Rear image: The values of various fields after changing. There is no rear image values for the delete operation.

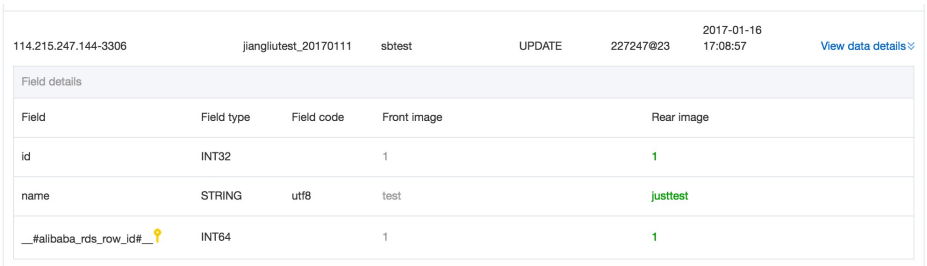


114.215.247.144-3306	jiangliutest_20170111	sbtest	DELETE	227805@23	2017-01-16 17:09:04	View data details

Field details

Field	Field type	Field code	Front image	Rear image
id	INT32		1	
name	STRING	utf8	justtest	
__#alibaba_rds_row_id#__	INT64		1	

There are front image and rear image values for the update operation.



114.215.247.144-3306	jiangliutest_20170111	sbtest	UPDATE	227247@23	2017-01-16 17:08:57	View data details

Field details

Field	Field type	Field code	Front image	Rear image
id	INT32		1	1
name	STRING	utf8	test	justtest
__#alibaba_rds_row_id#__	INT64		1	1

There are only field details for DDL operation, with no field name, field type, field encoding, pre-image or post-image fields, as shown below:

DTS data subscription requires the SDK provided by DTS for subscription and consumption of change data.

Before using the SDK for consumption, you need to Create Subscription instance of RDS Instance for Subscription on the DTS Console.

When the subscription instance is ready, you can use the SDK for real-time subscription of change data in the subscription instance. Currently:

1. DTS only supports JAVA version of SDK.
2. One subscription channel can only be consumed by one SDK. If multiple SDKs are connected with a subscription instance, only one SDK process can pull the change data. If multiple SDKs need to subscribe to the data of the same RDS, we need to create a subscription instance for each downstream SDK.

SDK defines many types of class objects. This section briefly introduces the interface definition of these classes of SDK.

RegionContex interface definition

- setAccessKey(accessKey)

Set security credentials. The parameters are AccessKey of the Alibaba Cloud account corresponding to the subscription instance requesting data subscription.

- setSecret(AccessKeySecret)

Set security credentials. The parameters are AccessKeySecret of the Alibaba Cloud account and it can be created and pulled from the AccessKeys page.

- setUsePublicIp(usePublicIp)

Check whether the configured SDK running server uses the internet for data subscription. If yes, the usePublicIp value is true, otherwise it is False.Currently DTS only supports subscribe data through internet.

ClusterClient interface definition

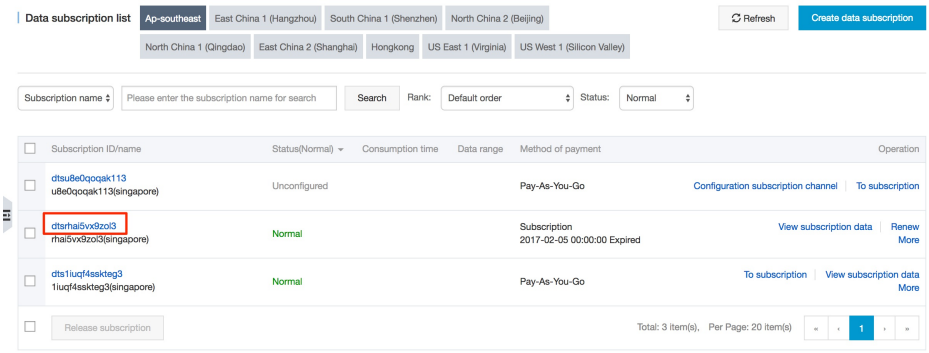
- void addConcurrentListener(ClusterListener arg0)

Add a downstream listener to a ClusterClient in order to subscribe to the change data in the subscription channel.

Parameter of ClusterListener arg0 is an object of ClusterListener class.

- void askForGUID(String arg0)

If you request change data of a subscription instance, the parameter of String arg0 is the ID of the subscription instance and needs to be obtained from DTS Console, as shown below.



- List<ClusterListener> getConcurrentListeners()

Obtain the list of listeners in the ClusterClient. The interface return type is List <ClusterListener >

- void start()

Start SDK client and start the subscription of change data.

- void stop()

Stop SDK client and stop the subscription of change data. Since data pulling and notify callback are performed in the same thread in the SDK, if the notify consumption code has a function that prevents interruption by a signal, the stop function may not work properly to close the client.

ClusterListener interface definition

- void notify(List<ClusterMessage> arg0)

This function is mainly used to define the change data consumption. When the SDK receives data, ClusterListner will be notified to consume data through notify. In the consumption method of the demo, subscription data is printed to the screen.

The input parameter type of this function is: List <ClusterMessage > ClusterMessage is the structure object of the subscription data storage. For detailed definitions, please refer to interface definition of ClusterMessage.

ClusterMessage interface definition

Each ClusterMessage stores the data record of one transaction in RDS, and each record is stored through Record. This section introduces the main interface functions of ClusterMessage.

- Record getRecord()

The interface obtains a binlog record from ClusterMessage. This change log indicates every record, such as begin, commit, update and insert, in the RDS binlog file.

- void ackAsConsumed

To simplify the disaster recovery process of SDKs, the data subscription service end supports consumption timestamp storage of SDKs. When a SDK experiences abnormal downtime and restarts, it will automatically subscribe to and consume data from the last consumption timestamp before the downtime.

When message consumption is completed, you need to pull this interface to report a ACK to DTS server to notify the server to update SDK consumption timestamp, and ensure the integrity of the consumption data after the SDK reboots.

Record interface definition

Record represents every record in the RDS binlog subscribed to, including begin, commit and update.

- String getAttribute(String key)

This function can obtain some of the major property values in Record. Its incoming argument is the property name, and returns the value of the property.

The property names and the corresponding property values of calling this function are as follows:

Key	Description
record_id	ID of this Record. The ID does not guarantee increments during the subscription process
instance	The connection address of the database instance of this Record. The format is: IP:Port
source_type	The engine type of the database instance of this Record. The current value is: mysql
source_category	The type of the Record. The current value is: full_recorded
timestamp	The time when the Records lands at the binlog. The time is also the SQL execution time in RDS
checkpoint	This corresponding binlog file point of this Record. The format is: file_offset@file_name. The file name of filen_name is the numeric suffix of binlog file
record_type	The type of operation corresponding to the Record and main values include: insert/update/delete/replace/ddl/begin/commit/heartbeat
db	The corresponding database name of the updated table of this Record

table_name	Name of the updated table of this Record
record_recording	Encoding of this Record
primary	Primary key column name of the updated table of this Record
fields_enc	Values of each encoding field of this Record. Each field is separated by a comma. The value of non-character type is empty

- Type getOpt()

Get the types of changes of this record, including:

Insert, delete, update, replace, ddl, begin, commit and heartbeat.

The heartbeat is the heartbeat table internally defined in DTS and it is mainly used to check the health of subscription instances. In theory, a heartbeat is generated every second.

- String getCheckpoint()

Obtain the checkpoint of this change log in the binlog, and the returned point format is: binlog_offset@binlog_fid

The binlog_offset refers to the offset of the change log in the binlog file. The binlog_fid refers to the numeric suffix of the binlog file. For example, if the file name of binlog is mysql-bin.0008, the binlog_fid is 8.

- String gettimestamp()

Obtain timestamp of this change log in the binlog.

- String getDbname()

Obtain the name of the database corresponding to the table modified in the change log.

- String getTablename()

Obtain the name of the table corresponding to the change log.

- String getPrimaryKeys()

Obtain primary key column name corresponding to the change log. If it is combined primary key, the column names shall be separated by commas.

- DBType getDbType()

Obtain the database type of the subscription instance. Currently, DTS only supports RDS MySQL, so the value is MySQL.

- String getServerId()

Obtain the RDS MySQL instance running process IP:PORT corresponding to the change log.

- int getFieldCount()

Obtain the number of Fields in the change log.

- List<Field> getFieldList()

The data type returned by the function is List < Field >

List<Field> It contains the definition of all the fields of the change log and the values before and after the change. For definition of Field object, please refer to **Field interface definition**.

- Boolean isFirstInLogevent()

Check whether this Record is the first transaction log in the database batch change. If yes, return True, otherwise return False.

Field interface definition

Field class defines the encoding, type, field name, field value, and whether it is the primary key and other properties of each field. This section introduces the interface definition of each Field class.

- String getEncoding()

Obtain the encoding format for this field value.

- String getFieldname()

Obtain the name of this field.

- Type getType()

Obtain the data type of this field. Please refer to the field type definition below for definition of the type.

- ByteString getValue()

Obtain the value of the field. The returned type is ByteString. When the value is empty, return NULL.

- Boolean isPrimary()

Check whether the field is the primary key column of the table. If yes, return True, otherwise return False.

In this section, you will learn how to complete basic operations with DTS SDK.

Initialize RegionContext

RegionContext is mainly used to save and set security authentication information and network access

mode settings. The following code shows how to initialize RegionContext, set up security authentication credentials and network access mode.

```
import java.util.List;
import com.aliyun.drc.clusterclient.RegionContext;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class MainClass
{
public static void main(String[] args) throws Exception {
// Create RegionContext
RegionContext context = new RegionContext();
// Configure AccessKey and AccessKeySecret of the Alibaba Cloud account
context.setAccessKey("<AccessKey>");
context.setSecret("<AccessKeySecret>");
// Whether the server running the SDK uses public network IP address to connect DTS subscription channel
context.setUsePublicIp(true);
// Below is other calling code...
.....
}
}
```

If you want to use the SDK, you must first initialize RegionContext, configure security certifications and other information of the connected subscription instance.
The above setAccessKey interface sets the AccessKey of Alibaba Cloud account
The setSecret interface sets the AccessKeySecret of Alibaba Cloud account
AccessKey and AccessKeySecret are assigned to the user by Alibaba Cloud system and they are called an ID pair. User ID can be created at the user center of Alibaba Cloud.
The setUsePublicIp tells DTS that whether the SDK running server uses public network IP address to connect the subscription instance. If the setting is true, the subscription data traffic used the public network, otherwise it uses the private network.

Initialize ClusterClient

SDK uses ClusterClient class to connect the subscription instance, and accept change data . The code below creates a ClusterClient:

```
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 RegionContext
RegionContext context = new RegionContext();
context.setAccessKey("<AccessKey>");
context.setSecret("<AccessKeySecret>");
context.setUsePublicIp(true);
```

```
// Create subscription consumers
final ClusterClient client = new DefaultClusterClient(context);

// The following is some other calling code...
.....
}
}
```

Initialize Listener

Change data consumption are achieved through Listener class. After the initialization of ClusterClient, listeners need to be added. Listener will define the notify function to receive subscription data and consume data. The code below realizes the most basic consumption logic and print the subscription data to 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 a RegionContext object
...
//Initialize ClusterClient object
.....
ClusterListener listener = new ClusterListener(){
@Override
public void notify(List<ClusterMessage> messages) throws Exception {
for (ClusterMessage message : messages) {
//Print the subscription change data
System.out.println(message.getRecord() + ":" + message.getRecord().getTablename() + ":"
+ message.getRecord().getOpt());
//After data consumption is completed, report ACK to DTS, and must call
message.ackAsConsumed();
}
}
}
}
```

DTS realizes the mechanism which saves data consumption time of SDK to the DTS server. This simplifies the complexity of SDK disaster recovery for SDK users.
The askAsConsumed () interface in the sample code above reports the timestamp of the latest data consumed by SDK to the DTS server. With the timestamp reported, if SDK restarts after unexpected downtime, it will automatically obtain the consumption time from the DTS server, and then restart from this time point to fix data duplication issues.

Starat ClusterClient

```
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 RegionContext
...
//Initialize ClusterClient
...
//Initialize ClusterListener
...
// Add listeners
client.addConcurrentListener(listener);
// Set requested subscription channel ID
client.askForGUID("dts_rdsrjiei2u2afnb_DSF");
// Start background thread. Note that here will not get blocked and the main thread cannot exit
client.start();
}
```

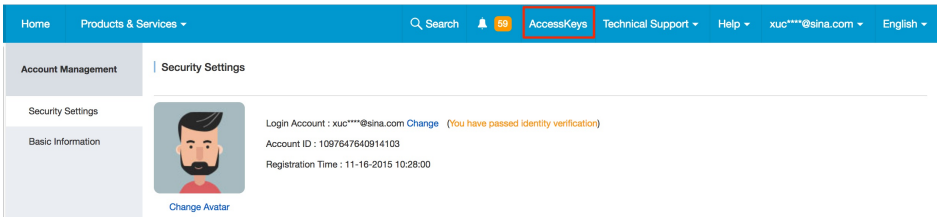
The subscription instance ID requested by the client is set by askForGUID interface in the code above. The ID of this subscription channel is obtained from DTS Console. Once the subscription instance ID is configured, this SDK can obtain the change data of this subscription instance.
You need to add listener to client before you start the client. In this way, when the client pulls change data from the subscription instance, it will start the data consumption by calling the notify method of the listener in a synchronized manner.

This section briefly introduces how to run the demo code provided by DTS Console.

Creating Access Key

When an SDK connects the subscription instance, Access Key certification through Alibaba Cloud account is required. Therefore, before using the SDK, it is necessary to create an Access Key. If the Alibaba Cloud account has created Access Key, you don’ t need to create one again. Otherwise, follow the procedures as below:

1. Visit Alibaba Cloud Official Website and log in to your account.
2. Click your account on the top left and go to the account management page.
3. Click AccessKeys on top right of the Console to enter Access Key management page.



1. Click the Create Acess key button in the upper right corner to create the Access Key.

Install JAVA SDK

The development environment supported by DTS Java SDK is J2SE Development Kit (JDK) 1.5 or later.

Next, we will take an Eclipse project as an example to introduce the steps for installing SDK:

1. Download SDK package consumer.jar from SDK Download;
2. Import jar package using Eclipse as an example: on Eclipse, right-click Project > Properties > Java Build Path > Libraries > Add External JARs, select the path of consumer.jar and add consumer.jar to the project.
3. Select consumer.jar and click the “OK” button. After the above steps, you can use DTS Java SDK in the project.

Run sample code

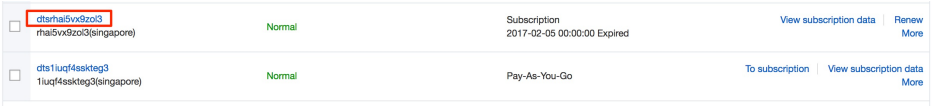
DTS provides demo of DTS SDK. You can duplicate the demo code in the “View Demo Code” function in the DTS Console. Next, we will take an Eclipse project as an example to briefly introduce how to run the SDK demo.

1. Create the MainClass class in the src directory of the Eclipse project.
2. Open the generated Java file MainClass, delete the code template in it.
3. Paste the sample code in the MainClass file.
4. In sample code, modify AccessKeyID, AccessKeySecret and subscription instance ID.

Attention: Click And download SDK Subscribe to data

```java
package com.aliyun.drc.clusterclient;
import java.util.List;
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 {
     //logger.info("start");
      // create a context
      RegionContext context = new RegionContext();
      // connect to dts by  internet
      context.setUsePublicIp(true);
      // aliyun accessKey secret
      context.setAccessKey("xxx");
      context.setSecret("xxx");
      // create a cosummer
      final ClusterClient client = new DefaultClusterClient(context);
      // create listener
      ClusterListener listener = new ClusterListener(){
     @Override
     public void notify(List<ClusterMessage> messages) throws Exception {
      for (ClusterMessage message : messages) {
         // debug
         System.out.println(message.getRecord() + ":" + message.getRecord().getTablename() + ":"
         + message.getRecord().getOpt());
         //you must call ackAsConsumed when you consume the data
         message.ackAsConsumed();
      }
     }
     @Override
     public void noException(Exception e) {
      e.printStackTrace();
     }};
      // add listener
      client.addConcurrentListener(listener);
      // set inanse ID
      client.askForGUID("instanceId");
      // start application
      client.start();
    }
}
```

The ID of the subscription instance should be obtained in the DTS Console.



Change the marked parts in the demo code above to AccessKeyID, AccessKeySecret and the subscription instance ID corresponding to your Alibaba Cloud account.

1. In Eclipse, right click the demo file > Operation Mode > Java Application to run the demo successfully.

Latest version

2016-04-26

java_sdk.jar