ALIBABA CLOUD

Alibaba Cloud

MaxCompute

Best Practices

Document Version: 20220630

Alibaba Cloud

Legal disclaimer

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

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

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

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

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

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

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

Document conventions

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

Table of Contents

1.SQL

1.1. Write MaxCompute SQL statements

This topic describes the common scenarios of using MaxCompute SQL statements and how to write them.

Prepare a dataset

The emp and dept tables are used as the dataset in this example. You can create a table on a MaxCompute project and upload data to the table. For more information about how to import data, see Overview.

Download data files of the emp table and data files of the dept table.

Execute the following statement to create the emp table:

```
CREATE TABLE IF NOT EXISTS emp (
  EMPNO STRING,
  ENAME STRING,
  JOB STRING,
  MGR BIGINT,
  HIREDATE DATETIME,
  SAL DOUBLE,
  COMM DOUBLE,
  DEPTNO BIGINT);
```

Execute the following statement to create the dept table:

```
CREATE TABLE IF NOT EXISTS dept (
  DEPTNO BIGINT,
  DNAME STRING,
  LOC STRING);
```

Examples

- Example 1: Query all departments that have at least one employee.

  We recommend that you use the JOIN clause to avoid large amounts of data in the query. Execute the following SQL statement:

  ```
  SELECT d. *
  FROM dept d
  JOIN (
      SELECT DISTINCT deptno AS no
      FROM emp
  ) e
  ON d.deptno = e.no;
  ```

- Example 2: Query all employees who have higher salaries than Smith.

The following code shows how to use MAPJOIN in the SQL statement for this scenario:

```
SELECT /*+ MapJoin(a) */ e.empno
    , e.ename
    , e.sal
FROM emp e
JOIN (
    SELECT MAX(sal) AS sal
    FROM `emp`
    WHERE `ENAME` = 'SMITH'
) a
ON e.sal > a.sal;
```

- Example 3: Query the names of all employees and the names of their immediate superiors.

  The following code shows how to use EQUI JOIN in the SQL statement for this scenario:

```
SELECT a.ename
    , b.ename
FROM emp a
LEFT OUTER JOIN emp b
ON b.empno = a.mgr;
```

- Example 4: Query all jobs that have basic salaries higher than USD 1,500.

  The following code shows how to use the HAVING clause in the SQL statement for this scenario:

```
SELECT emp. `JOB`
    , MIN(emp.sal) AS sal
FROM `emp`
GROUP BY emp. `JOB`
HAVING MIN(emp.sal) > 1500;
```

- Example 5: Query the number of employees in each department, the average salary, and the average length of service.

  The following code shows how to use built-in functions in the SQL statement for this scenario:

```
SELECT COUNT(empno) AS cnt_emp
    , ROUND(AVG(sal), 2) AS avg_sal
    , ROUND(AVG(datediff(getdate(), hiredate, 'dd')), 2) AS avg_hire
FROM `emp`
GROUP BY `DEPTNO`;
```

- Example 6: Query the names and the sorting order of the first three employees who have the highest salaries.

  The following code shows how to use the TOP N clause in the SQL statement for this scenario:

```
SELECT *
FROM (
  SELECT deptno
    , ename
    , sal
    , ROW_NUMBER() OVER (PARTITION BY deptno ORDER BY sal DESC) AS nums
  FROM emp
) emp1
WHERE emp1.nums < 4;
```

- Example 7: Query the number of employees in each department and the proportion of clerks in these departments.

```
SELECT deptno
    , COUNT(empno) AS cnt
    , ROUND(SUM(CASE
      WHEN job = 'CLERK' THEN 1
      ELSE 0
    END) / COUNT(empno), 2) AS rate
FROM `EMP`
GROUP BY deptno;
```

Notes

- When you use the GROUP BY clause, the SELECT list can only consist of aggregate functions or columns that are part of the GROUP BY clause.
- ORDER BY must be followed by LIMIT N.
- The SELECT expression does not support subqueries. To use subqueries, you can rewrite the code to include a JOIN clause.
- The JOIN clause does not support Cartesian projects. You can replace the JOIN clause with MAPJOIN.
- UNION ALL must be replaced with subqueries.
- The subquery that is specified in the IN or NOT IN clause must contain only one column and return a maximum of 1,000 rows. Otherwise, use the JOIN clause.

1.2. Rewrite incompatible SQL statements

This topic describes how to modify SQL statements that are incompatible with MaxCompute V2.0.

Background information

MaxCompute V2.0 fully embraces open source ecosystems, supports more programming languages and features, and provides higher performance. It also inspects syntax more rigorously. As a result, errors may be returned for some statements that use less rigorous syntax and are successfully executed in the earlier versions.

To enable the smooth canary upgrade to MaxCompute V2.0, the MaxCompute framework supports rollback. If MaxCompute V2.0 fails to execute a job, MaxCompute V1.0 will execute the job instead. The rollback increases the latency of the job. Before you submit jobs, we recommend that you configure `set odps.sql.planner.mode=lot;` to manually disable the rollback feature. This prevents the impacts from the modification on the MaxCompute rollback policy.

The MaxCompute team notifies the owners of the jobs for which the required SQL statements cannot be executed by email or DingTalk based on the online rollback condition. The job owners must modify the SQL statements for the jobs at the earliest opportunity. Otherwise, the jobs may fail.

group.by.with.star

This statement is equivalent to the `select * …group by…` statement.

- In MaxCompute V2.0, all the columns of a source table must be included in the GROUP BY clause. Otherwise, an error is returned.
- In the earlier version of MaxCompute, `select * from group by key` is supported even if not all columns of a source table are included in the GROUP BY clause.

Examples

- Scenario 1: The GROUP BY key does not include all columns.
  - Invalid syntax:

    ```
    select * from t group by key;
    ```

  - Error message:

    ```
    FAILED: ODPS-0130071:[1,8] Semantic analysis exception - column reference t.value shoul
    d appear in GROUP BY key
    ```

  - Valid syntax:

    ```
    select distinct key from t;
    ```

- Scenario 2: The GROUP BY key includes all columns.
  - We recommend that you do no use the following syntax:

    ```
    select * from t group by key, value; -- t has columns key and value
    ```

  - Even if the preceding syntax causes no errors in MaxCompute V2.0, we recommend that you use the following syntax:

    ```
    select distinct key, value from t;
    ```

bad.escape

The escape sequence is invalid.

MaxCompute defines that, in string literal, each ASCII character that ranges from 0 to 127 must be written in the format of a backslash (\) followed by three octal digits. For example, 0 is written as \001, and 1 is written as \002. However, \01 and \0001 are processed as \001.

This method confuses new users. For example, "\0001" cannot be processed as "\000"+"1". For users who migrate data from other systems to MaxCompute, invalid data may be generated.

> **Note** If numbers are appended to `\000`, such as numbers in the range of `\0001 to \0009` or the number `\00001`, an error may be returned.

MaxCompute V2.0 corrects the sequences in scripts to handle this issue.

- Invalid syntax:

```
select split(key, "\01"), value like "\0001" from t;
```

- Error message:

```
FAILED: ODPS-0130161:[1,19] Parse exception - unexpected escape sequence: 01
ODPS-0130161:[1,38] Parse exception - unexpected escape sequence: 0001
```

- Valid syntax:

```
select split(key, "\001"), value like "\001" from t;
```

column.repeated.in.creation

If duplicate column names are detected when the CREATE TABLE statement is executed, MaxCompute V2.0 returns an error.

Examples

- Invalid syntax:

```
create table t (a BIGINT, b BIGINT, a BIGINT);
```

- Error message:

```
FAILED: ODPS-0130071:[1,37] Semantic analysis exception - column repeated in creation: a
```

- Valid syntax:

```
create table t (a BIGINT, b BIGINT);
```

string.join.double

You want to join the values of the STRING type with those of the DOUBLE type.

- In the early version of MaxCompute, the values of the STRING and DOUBLE types are converted into the BIGINT type. This causes precision loss. For example, 1.1 = "1" in a JOIN condition is considered equal.
- In MaxCompute V2.0, the values of the STRING and DOUBLE types are converted into the DOUBLE type because MaxCompute V2.0 is compatible with Hive.

Examples

- Syntax that is not recommended:

```
select * from t1 join t2 on t1.double_value = t2.string_value;
```

- Warning information:

```
WARNING:[1,48]  implicit conversion from STRING to DOUBLE, potential data loss, use CAST
function to suppress
```

- Recommended syntax:

```
select * from t1 join t2 on t.double_value = cast(t2.string_value as double);
```

window.ref.prev.window.alias

Window functions reference the aliases of other window functions in the SELECT clause of the same level.

Examples

- Assume that rn does not exist in t1. Invalid syntax:

```
select row_number() over (partition by c1 order by c1) rn,
row_number() over (partition by c1 order by rn) rn2
from t1;
```

- Error message:

```
FAILED: ODPS-0130071:[2,45] Semantic analysis exception - column rn cannot be resolved
```

- Valid syntax:

```
select row_number() over (partition by c1 order by rn) rn2
from
(select c1, row_number() over (partition by c1 order by c1) rn
from t1
) tmp;
```

select.invalid.token.after.star

The SELECT clause allows you to use an asterisk (*) to select all the columns of a table. However, the asterisk cannot be followed by `aliases` even if the asterisk specifies only one column. The new editor returns errors for similar syntax.

Examples

- Invalid syntax:

```
select * as alias from table_test;
```

- Error message:

```
FAILED: ODPS-0130161:[1,10] Parse exception - invalid token 'as'
```

- Valid syntax:

```
select * from table_test;
```

agg.having.ref.prev.agg.alias

If HAVING exists, the SELECT clause can reference aggregate function aliases.

Examples

- Invalid syntax:

```
select count(c1) cnt,
sum(c1) / cnt avg
from t1
group by c2
having cnt > 1;
```

- Error message:

```
FAILED: ODPS-0130071:[2,11] Semantic analysis exception - column cnt cannot be resolved
ODPS-0130071:[2,11] Semantic analysis exception - column reference cnt should appear in G
ROUP BY key
```

s and cnt do not exist in source table t1. However, the early version of MaxCompute does not return an error because HAVING exists. In MaxCompute V2.0, the error message `column cannot be resolve` is returned.

- Valid syntax:

```
select cnt, s, s/cnt avg
from
(
select count(c1) cnt,
sum(c1) s
from t1
group by c2
having count(c1) > 1
) tmp;
```

order.by.no.limit

In MaxCompute, the `ORDER BY` clause must be followed by a `LIMIT` clause to limit the number of data records. `ORDER BY` is used to sort all data records. If ORDER BY is not followed by a `LIMIT` clause, the execution performance is low.

Examples

- Invalid syntax:

```
select * from (select *
from (select cast(login_user_cnt as int) as uv, '3' as shuzi
from test_login_cnt where type = 'device' and type_name = 'mobile') v
order by v.uv desc) v
order by v.shuzi limit 20;
```

- Error message:

```
FAILED: ODPS-0130071:[4,1] Semantic analysis exception - ORDER BY must be used with a LIM
IT clause
```

Add a `LIMIT` clause to the subquery `order by v.uv desc`.

In MaxCompute V1.0, view checks are not rigorous. For example, a view is created in a project which does not require a check on the `LIMIT` clause. odps.sql.validate.orderby.limit=false indicates that the project does not require a check on the LIMIT clause.

```
create view table_view as select id from table_view order by id;
```

Execute the following statement to access the view:

```
select * from table_view;
```

MaxCompute V1.0 does not return an error, whereas MaxCompute V2.0 returns the following error:

```
FAILED: ODPS-0130071:[1,15] Semantic analysis exception - while resolving view xdj.xdj_view
_limit - ORDER BY must be used with a LIMIT clause
```

generated.column.name.multi.window

Automatically generated `aliases` are used.

In the early version of MaxCompute, an alias is automatically generated for each expression of a SELECT statement. The alias is displayed on the MaxCompute client. However, the early version of MaxCompute does not guarantee that the alias generation rule is correct or remains unchanged. We recommend that you do not use automatically generated aliases.

MaxCompute V2.0 warns you against the use of automatically generated aliases. However, MaxCompute V2.0 does not prohibit the use of automatically generated aliases to avoid adverse impacts.

In some cases, known changes are made to the alias generation rules in the different versions of MaxCompute. Some online jobs depend on automatically generated aliases. These jobs may fail when MaxCompute is being upgraded or rolled back. If you encounter these issues, modify your queries and explicitly specify the aliases of the columns.

Examples

- Syntax that is not recommended:

  ```
  select _c0 from (select count(*) from table_name) t;
  ```

- Recommended syntax:

  ```
  select c from (select count(*) c from table_name) t;
  ```

non.boolean.filter

Non-BOOLEAN filter conditions are used.

MaxCompute prohibits implicit conversions between the BOOLEAN type and other data types. However, the early version of MaxCompute allows the use of BIGINT filter conditions in some cases. MaxCompute V2.0 prohibits the use of BIGINT filter conditions. If your scripts have BIGINT filter conditions, modify them at the earliest opportunity. Examples:

Invalid syntax:

```
select id, count(*) from table_name group by id having id;
```

Error message:

```
FAILED: ODPS-0130071:[1,50] Semantic analysis exception - expect a BOOLEAN expression
```

Valid syntax:

```
select id, count(*) from table_name group by id having id <> 0;
```

post.select.ambiguous

The ORDER BY, CLUSTER BY, DISTRIBUTE BY, and SORT BY clauses reference columns with conflicting names.

In the early version of MaxCompute, the system automatically selects the last column in a SELECT clause as the operation object. However, MaxCompute V2.0 reports an error in this case. Modify your queries at the earliest opportunity. Examples:

Invalid syntax:

```
select a, b as a from t order by a limit 10;
```

Error message:

```
FAILED: ODPS-0130071:[1,34] Semantic analysis exception - a is ambiguous, can be both t.a o
r null.a
```

Valid syntax:

```
select a as c, b as a from t order by a limit 10;
```

The change covers the statements that have conflicting column names but have the same syntax. Even though no ambiguity is caused, the system returns an error to warn you against these statements. We recommend that you modify relevant statements.

duplicated.partition.column

Partitions with the same name are specified in a query.

In the early version of MaxCompute, no error is returned if two partition keys with the same name are specified. The latter partition key overwrites the former partition. This causes confusion. MaxCompute V2.0 returns an error in this case. Examples:

Invalid syntax 1:

```
insert overwrite table partition (ds = '1', ds = '2')select ... ;
```

`ds = '1'` is ignored during execution.

Valid syntax:

```
insert overwrite table partition (ds = '2')select ... ;
```

Invalid syntax 2:

```
create table t (a bigint, ds string) partitioned by (ds string);
```

Valid syntax:

```
create table t (a bigint) partitioned by (ds string);
```

order.by.col.ambiguous

The ORDER BY clause references the duplicate aliases in a SELECT clause.

Invalid syntax:

```
select id, id
from table_test
order by id;
```

Valid syntax:

```
select id, id id2
from table_name
order by id;
```

Remove the duplicate aliases before the ORDER BY clause can reference them.

in.subquery.without.result

If colx in subquery does not return results, colx does not exist in the source table.

Invalid syntax:

```
select * from table_name
where not_exist_col in (select id from table_name limit 0);
```

Error message:

```
FAILED: ODPS-0130071:[2,7] Semantic analysis exception - column not_exist_col cannot be res
olved
```

ctas.if.not.exists

The syntax of a destination table is invalid.

If the destination table exists, the early version of MaxCompute does not check the syntax. However, MaxCompute V2.0 checks the syntax. As a result, a large number of errors may be returned. Examples:

Invalid syntax:

```
create table if not exists table_name
as
select * from not_exist_table;
```

Error message:

```
FAILED: ODPS-0130131:[1,50] Table not found - table meta_dev.not_exist_table cannot be reso
lved
```

worker.restart.instance.timeout

In the early version of MaxCompute, each time a UDF generates a record, a write operation is triggered on Apsara Distributed File System, and a heartbeat packet is sent to Job Scheduler. If the UDF does not generate records for 10 minutes, the following error is returned:

```
FAILED: ODPS-0123144: Fuxi job failed - WorkerRestart errCode:252,errMsg:kInstanceMonitorTi
meout, usually caused by bad udf performance.
```

The runtime framework of MaxCompute V2.0 supports vectoring to process multiple rows of a column at a time. This makes execution more efficient. If multiple records are processed at a time and no heartbeat packets are sent to Job Scheduler within the specific period, vectoring may cause normal statements to time out. The interval between two output records cannot exceed 10 minutes.

If a timeout error occurs, we recommend that you first check the performance of UDFs. It requires several seconds to process each record. If UDFs cannot be optimized, you can manually set batch.rowcount to handle this issue. The default value of batch.rowcount is 1024.

```
set odps.sql.executionengine.batch.rowcount=16;
```

divide.nan.or.overflow

The early version of MaxCompute does not support division constant folding.

The following code shows the physical execution plan in the early version of MaxCompute:

```
explain
select if(false, 0/0, 1.0)
from table_name;
in task M1_Stg1:
    Data source: meta_dev.table_name
    TS: alias: table_name
      SEL: If(False, Divide(UDFToDouble(0), UDFToDouble(0)), 1.0)
        FS: output: None
```

The IF and DIVIDE functions are retained. During execution, the first parameter of IF is set to False, and the expression of DIVIDE is not evaluated. Divide-by-zero errors do not occur.

However, MaxCompute V2.0 supports division constant folding. As a result, an error is returned. Examples:

Invalid syntax:

```
select IF(FALSE, 0/0, 1.0)
from table_name;
```

Error message:

```
FAILED: ODPS-0130071:[1,19] Semantic analysis exception - encounter runtime exception while
evaluating function /, detailed message: DIVIDE func result NaN, two params are 0.000000 an
d 0.000000
```

An overflow error may also occur. Examples:

Invalid syntax:

```
select if(false, 1/0, 1.0)
from table_name;
```

Error message:

```
FAILED: ODPS-0130071:[1,19] Semantic analysis exception - encounter runtime exception while
evaluating function /, detailed message: DIVIDE func result overflow, two params are 1.0000
00 and 0.000000
```

Valid syntax:

We recommend that you remove /0 and use valid constants.

A similar issue occurs in the constant folding for CASE WHEN, such as CASE WHEN TRUE THEN 0 ELSE 0/0. During constant folding in MaxCompute V2.0, all subexpressions are evaluated, which causes divide-by-zero errors.

CASE WHEN may involve more complex optimization scenarios. Example:

```
select case when key = 0 then 0 else 1/key end
from (
select 0 as key from src
union all
select key from src) r;
```

The optimizer pushes down the division operation to subqueries. The following code shows a similar conversion:

```
M (
select case when 0 = 0 then 0 else 1/0 end c1 from src
UNION ALL
select case when key = 0 then 0 else 1/key end c1 from src) r;
```

Error message:

```
FAILED: ODPS-0130071:[0,0] Semantic analysis exception - physical plan generation failed: j
ava.lang.ArithmeticException: DIVIDE func result overflow, two params are 1.000000 and 0.00
0000
```

An error is returned for the constant folding in the first clause of UNION ALL. We recommend that you move CASE WHEN in the SQL statement to subqueries and remove useless CASE WHEN statements and /0.

```
select c1 end
from (
select 0 c1 end from src
union all
select case when key = 0 then 0 else 1/key end) r;
```

small.table.exceeds.mem.limit

The early version of MaxCompute supports multi-way join optimization. Multiple JOIN operations with the same join key are merged for execution in the same Fuxi task, such as J4_1_2_3_Stg1 in this example.

```
explain
select t1.*
from t1 join t2 on t1.c1 = t2.c1
join t3 on t1.c1 = t3.c1;
```

The following code shows the physical execution plan in the early version of MaxCompute:

```
In Job job0:
root Tasks: M1_Stg1, M2_Stg1, M3_Stg1
J4_1_2_3_Stg1 depends on: M1_Stg1, M2_Stg1, M3_Stg1
In Task M1_Stg1:
    Data source: meta_dev.t1
In Task M2_Stg1:
    Data source: meta_dev.t2
In Task M3_Stg1:
    Data source: meta_dev.t3
In Task J4_1_2_3_Stg1:
    JOIN: t1 INNER JOIN unknown INNER JOIN unknown
        SEL: t1._col0, t1._col1, t1._col2
            FS: output: None
```

If MAPJOIN hints are added, the physical execution plan in the early version of MaxCompute remains unchanged. In the early version of MaxCompute, multi-way join optimization is preferentially used, and user-defined MAPJOIN hints can be ignored.

```
explain
select /* +mapjoin(t1) */ t1.*
from t1 join t2 on t1.c1 = t2.c1
join t3 on t1.c1 = t3.c1;
```

The preceding physical execution plan in the early version of MaxCompute is applied.

The optimizer of MaxCompute V2.0 preferentially uses user-defined MAPJOIN hints. In this example, if t1 is a large table, an error similar to the following one is returned:

```
FAILED: ODPS-0010000:System internal error - SQL Runtime Internal Error: Hash Join Cursor H
ashJoin_REL… small table exceeds, memory limit(MB) 640, fixed memory used …, variable memor
y used …
```

In this case, if MAPJOIN is not required, we recommend that you remove MAPJOIN hints.

sigkill.oom

sigkill.oom has the same issue as small.table.exceeds.mem.limit. If you specify MAPJOIN hints and the sizes of small tables are large, multiple JOIN statements may be optimized by using multi-way joins in the early version of MaxCompute. As a result, the statements are successfully executed in the early version of MaxCompute. However, in MaxCompute V2.0, some users may use `odps.sql.mapjoin.memory.max` to prevent small tables from exceeding the size limit. Each MaxCompute worker has a memory limit. If the sizes of small tables are large, MaxCompute workers may be terminated because the memory limit is exceeded. If this happens, an error similar to the following one is returned:

```
Fuxi job failed - WorkerRestart errCode:9,errMsg:SigKill(OOM), usually caused by OOM(out of memory).
```

We recommend that you remove MAPJOIN hints and use multi-way joins.

wm_concat.first.argument.const

Based on the WM_CONCAT function described in Aggregate functions, the first parameter of WM_CONCAT must be a constant. However, the early version of MaxCompute does not have rigorous check standards. For example, if the source table has no data, no error is returned even if the first parameter of WM_CONCAT is ColumnReference.

```
Function declaration:
string wm_concat(string separator, string str)
Parameters:
separator: the delimiter, which is a constant of the STRING type. Delimiters of other types or non-constant delimiters result in exceptions.
```

MaxCompute V2.0 checks the validity of parameters during the planning stage. If the first parameter of WM_CONCAT is not a constant, an error is returned. Examples:

Invalid syntax:

```
select wm_concat(value, ',') FROM src group by value;
```

Error message:

```
FAILED: ODPS-0130071:[0,0] Semantic analysis exception - physical plan generation failed: com.aliyun.odps.lot.cbo.validator.AggregateCallValidator$AggregateCallValidationException: Invalid argument type - The first argument of WM_CONCAT must be constant string.
```

pt.implicit.convertion.failed

srcpt is a partitioned table that has two partitions.

```
create table srcpt(key STRING, value STRING) partitioned by (pt STRING);
alter table srcpt add partition (pt='pt1');
alter table srcpt add partition (pt='pt2');
```

In the preceding SQL statements, the constants of the INT type in the pt columns of the STRING type are converted into those of the DOUBLE type for comparison. Even if `odps.sql.udf.strict.mode=true` is configured for the project, the early version of MaxCompute does not return an error and it filters out all pt columns. However, in MaxCompute V2.0, an error is returned. Examples:

Invalid syntax:

```
select key from srcpt where pt in (1, 2);
```

Error message:

```
FAILED: ODPS-0130071:[0,0] Semantic analysis exception - physical plan generation failed: j
ava.lang.NumberFormatException: ODPS-0123091:Illegal type cast - In function cast, value 'p
t1' cannot be casted from String to Double.
```

We recommend that you do not compare the values in the partition key columns of the STRING and INT constants. If such comparison is required, convert the INT constants into the STRING type.

having.use.select.alias

SQL specifications define that the GROUP BY and HAVING clauses precede a SELECT clause. Therefore, the column alias generated by the SELECT clause cannot be used in the HAVING clause.

Examples

- Invalid syntax:

  ```
  select id id2 from table_name group by id having id2 > 0;
  ```

- Error message:

  ```
  FAILED: ODPS-0130071:[1,44] Semantic analysis exception - column id2 cannot be resolvedOD
  PS-0130071:[1,44] Semantic analysis exception - column reference id2 should appear in GRO
  UP BY key
  ```

  id2 is the column alias generated by the SELECT clause and cannot be used in the HAVING clause.

dynamic.pt.to.static

In MaxCompute V2.0, dynamic partitions may be converted into static partitions by the optimizer.

Examples

```
insert overwrite table srcpt partition(pt) select id, 'pt1' from table_name;
```

The preceding statement is converted into the following statement:

```
insert overwrite table srcpt partition(pt='pt1') select id from table_name;
```

If the specified partition value is invalid, such as '${bizdate}', MaxCompute V2.0 returns an error during syntax checks. For more information, see Partition.

Invalid syntax:

```
insert overwrite table srcpt partition(pt) select id, '${bizdate}' from table_name limit 0;
```

Error message:

```
FAILED: ODPS-0130071:[1,24] Semantic analysis exception - wrong columns count 2 in data sou
rce, requires 3 columns (includes dynamic partitions if any)
```

In the early version of MaxCompute, no results are returned by the SQL statements due to LIMIT 0, and no dynamic partitions are created. As a result, no error is returned.

lot.not.in.subquery

Processing of NULL values in the IN subquery.

In a standard SQL IN operation, if the value list contains a NULL value, the return value may be NULL or true, but cannot be false. For example, 1 in (null, 1, 2, 3) returns true, 1 in (null, 2, 3) returns NULL, and null in (null, 1, 2, 3) returns NULL. Likewise, for the NOT IN operation, if the value list contains a NULL value, the return value may be false or NULL, but cannot be true.

MaxCompute V2.0 processes NULL values by using standard execution rules. If you receive a notification for this issue, check whether the subqueries in the IN operation have a NULL value and whether the related execution meets your expectations. If the related execution does not meet your expectations, modify the queries.

Examples

```
select * from t where c not in (select accepted from c_list);
```

If the accepted column does not contain NULL values, ignore this issue. If the accepted column contains NULL values, `c not in (select accepted from c_list)` returns true in the early version of MaxCompute and NULL in MaxCompute V2.0.

- Valid syntax:

```
select * from t where c not in (select accepted from c_list where accepted is not null)
```

1.3. Export SQL execution results

This topic describes how to export SQL execution results in MaxCompute.

> **Note** This topic provides examples based on Alibaba Cloud MaxCompute SDK for Java.

Overview

You can use the following methods to export the execution results of SQL statements:

- If the amount of data is small, use SQLTask to obtain all query results.
- If you want to export the query results of a table or a partition, use Tunnel.
- If the SQL statements are complex, use Tunnel and SQLTask to export the query results.
- Use DataWorks to execute SQL statements, synchronize data, perform timed scheduling, and configure task dependencies.
- Use the open source tool DataX to export data from MaxCompute to specified destination data

sources.

Use SQLTask to export data

SQLTask uses Alibaba Cloud MaxCompute SDK to call SQLTask.getResult(i) to execute SQL statements and obtain the query results. For more information, see SQLTask.

When you use SQLTask, note the following rules:

- SQLTask.getResult(i) is used to export the results of SELECT statements. You cannot use it to export the execution results of other MaxCompute SQL statements such as **SHOW TABLES**.
- You can use READ_TABLE_MAX_ROW to specify the maximum number of data records that the SELECT statement returns to a client. For more information, see Project operations.
- The SELECT statement returns a maximum of 10,000 data records to a client. You can execute the SELECT statement on a client such as SQLTask. This is equivalent to appending a LIMIT N clause to the SELECT statement.

  This rule does not apply if you execute the **CREATE TABLE XX AS SELECT** or **INSERT INTO/OVERWRITE TABLE** statement to solidify the results into a specified table.

Use Tunnel to export data

If a query returns all data of a table or a partition, you can use Tunnel to export the data. For more information, see Tunnel commands and MaxCompute Tunnel overview.

The following example shows how to run a Tunnel command to export data. If the Tunnel command cannot be used to export data, you can compile the Tunnel SDK to export data. For more information, see MaxCompute Tunnel overview.

```
tunnel d wc_out c:\wc_out.dat;
2016-12-16 19:32:08 - new session: 201612161932082d3c9b0a012f68e7 total lines: 3
2016-12-16 19:32:08 - file [0]: [0, 3), c:\wc_out.dat
downloading 3 records into 1 file
2016-12-16 19:32:08 - file [0] start
2016-12-16 19:32:08 - file [0] OK. total: 21 bytes
download OK
```

Use SQLTask and Tunnel to export data

SQLTask cannot be used to process more than 10,000 data records, whereas Tunnel can. These two methods complement each other. You can use SQLTask and Tunnel together to export more than 10,000 data records.

The following sample code provides an example to show how to use SQLTask and Tunnel to export data:

```
private static final String accessId = "userAccessId";
    private static final String accessKey = "userAccessKey";
    private static final String endPoint = "http://service.cn-shanghai.maxcompute.aliyun.co
m/api";
    private static final String project = "userProject";
    private static final String sql = "userSQL";
    private static final String table = "Tmp_" + UUID.randomUUID().toString().replace("-",
"_");// Use a random string as the name of the temporary table.
    private static final Odps odps = getOdps();
    public static void main(String[] args) {
```

```
        System.out.println(table);
        runSql();
        tunnel();
    }
    /*
     * Download the results that are returned by SQLTask.
     * */
    private static void tunnel() {
        TableTunnel tunnel = new TableTunnel(odps);
        try {
            DownloadSession downloadSession = tunnel.createDownloadSession(
                    project, table);
            System.out.println("Session Status is : "
                    + downloadSession.getStatus().toString());
            long count = downloadSession.getRecordCount();
            System.out.println("RecordCount is: " + count);
            RecordReader recordReader = downloadSession.openRecordReader(0,
                    count);
            Record record;
            while ((record = recordReader.read()) != null) {
                consumeRecord(record, downloadSession.getSchema());
            }
            recordReader.close();
        } catch (TunnelException e) {
            e.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    /*
     * Save the data.
     * If the amount of data is small, you can directly copy the data from the output. You
can also use Java.io to write the data to a local file or a remote storage system to save t
he data.
     * */
    private static void consumeRecord(Record record, TableSchema schema) {
        System.out.println(record.getString("username")+","+record.getBigint("cnt"));
    }
    /*
     * Execute an SQL statement to save the query results to a temporary table.
     * The time-to-live (TTL) of the saved data is one day. Saved data does not consume muc
h storage space. The storage of the system is not affected even if an error occurs when the
system deletes the data.
     * */
    private static void runSql() {
        Instance i;
        StringBuilder sb = new StringBuilder("Create Table ").append(table)
                .append(" lifecycle 1 as ").append(sql);
        try {
            System.out.println(sb.toString());
            i = SQLTask.run(getOdps(), sb.toString());
            i.waitForSuccess();
        } catch (OdpsException e) {
            e.printStackTrace();
```

```
        }
    }
    /*
     * Initialize the connection information of MaxCompute.
     * */
    private static Odps getOdps() {
        Account account = new AliyunAccount(accessId, accessKey);
        Odps odps = new Odps(account);
        odps.setEndpoint(endPoint);
        odps.setDefaultProject(project);
        return odps;
    }
```

Use DataWorks to synchronize and export data

DataWorks allows you to execute SQL statements and configure data synchronization tasks to generate and export data.

1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Workspaces**.
3. On the Workspaces page, find the workspace that you want to manage and click **Data Analytics** in the Actions column.
4. Create a business process.
   i. On the Data Analytics page, right-click **Business process** and select **New business process**.
   ii. Enter a name in the **Business Name** field.
   iii. Click **New**.
5. Create an ODPS SQL node.
   i. Right-click the business process and choose **New > MaxCompute > ODPS SQL**.
   ii. Enter runsql in the **Node name** field and click **Submit**.
   iii. Configure the ODPS SQL node and click the **Save** icon.
6. Create a data synchronization node.
   i. Right-click the business process and choose **New > Data Integration > Offline synchronization**.
   ii. Enter sync2mysql in the **Node name** field and click **Submit**.
   iii. Specify a data source and a data destination.
   iv. Configure the mapping between columns in the source and destination tables.
   v. Configure channel control.
   vi. Click the **Save** icon.
7. Configure a dependency between the data synchronization node and the ODPS SQL node. Configure the ODPS SQL node as the output node and the data synchronization node as the export node.
8. Configure workflow scheduling or use the default settings. Then, click the **Run** icon. The following information shows the operational log for data synchronization:

```
2016-12-17 23:43:46.394 [job-15598025] INFO JobContainer -
Task start time: 2016-12-17 23:43:34
Task end time: 2016-12-17 23:43:46
Total execution time: 11s
Average amounts of data per task: 31.36 KB/s
Write speed: 1,668 rec/s
Read records: 16,689
Failed read and write attempts: 0
```

9. Execute the following SQL statement to query the data synchronization results:

```
select count(*) from result_in_db;
```

1.4. Check whether partition pruning is effective

This topic describes how to check whether partition pruning is effective.

Background information

A MaxCompute partitioned table is a table with partitions. You can specify one or more columns as the partition key to create a partitioned table. If you have specified the name of a partition that you want to access, MaxCompute reads data only from that partition and does not scan the entire table. This reduces costs and improves efficiency.

Partition pruning allows you to specify filter conditions for partition key columns. This way, MaxCompute reads data only from the partitions that meet the filter conditions that you have specified in SQL statements. This avoids the errors and waste of resources that are caused by full table scans. However, partition pruning may not take effect sometimes.

This topic describes partition pruning from the following aspects:

- Check whether partition pruning is effective
- Scenarios where partition pruning does not take effect

Check whether partition pruning is effective

To check whether partition pruning is effective for a query, execute the EXPLAIN statement to view the execution plan of the query.

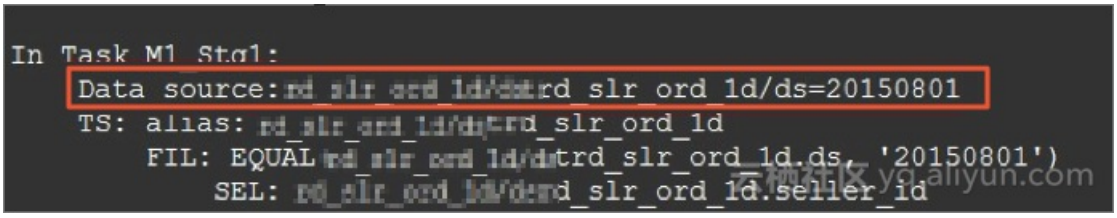
- For a query where partition pruning does not take effect:

  ```
  explain
  select seller_id
  from xxxxx_trd_slr_ord_1d
  where ds=rand();
  ```

  The execution plan indicates that all the 1,344 partitions of Table xxxxx_trd_slr_ord_1d are read.

- For a query where partition pruning is effective:

```
explain
select seller_id
from xxxxx_trd_slr_ord_1d
where ds='20150801';
```



The execution plan indicates that only Partition 20150801 of Table xxxxx_trd_slr_ord_1d is read.

Scenarios where partition pruning does not take effect

- Improper use of UDFs

  If you use user-defined functions (UDFs) or specific built-in functions to specify partitions, partition pruning may not take effect. In this case, we recommend that you execute the EXPLAIN statement to check whether partition pruning is effective.

  ```
  explain
  select ...
  from xxxxx_base2_brd_ind_cw
  where ds = concat(SPLIT_PART(bi_week_dim(' ${bdp.system.bizdate}'), ',', 1), SPLIT_PART(b
  i_week_dim(' ${bdp.system.bizdate}'), ',', 2))
  ```

  > **Note** For more information about UDF-based partition pruning, see the "WHERE" section in WHERE clause (where_condition).

- Improper use of joins

  When you join tables, pay attention to the following rules:

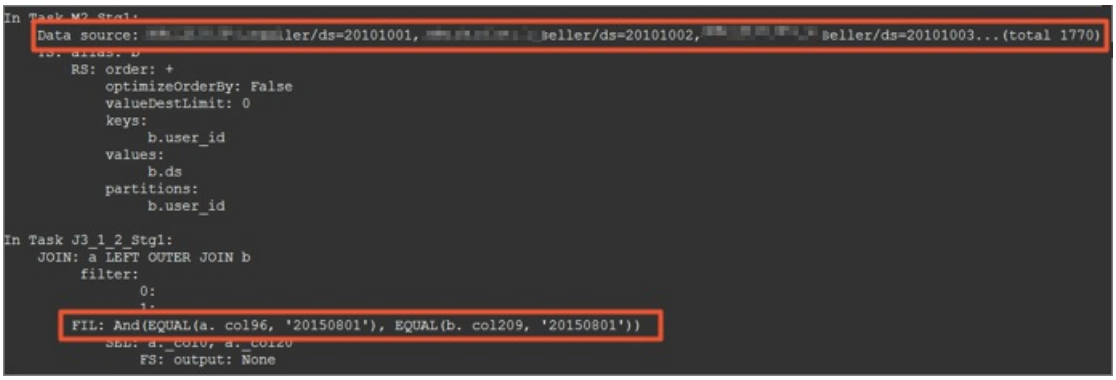
  - If partition pruning conditions are specified in the WHERE clause, partition pruning is effective.
  - If partition pruning conditions are specified in the ON clause, partition pruning is effective for the secondary table, but not the primary table.

  The following examples describe how partition pruning works when three different types of JOIN operations are performed:

  - LEFT OUTER JOIN

- For a query where partition pruning conditions are specified in the ON clause:

```
set odps.sql.allow.fullscan=true;
explain
select a.seller_id
    ,a.pay_ord_pbt_1d_001
from xxxxx_trd_slr_ord_1d a
left outer join
     xxxxx_seller b
on a.seller_id=b.user_id
and a.ds='20150801'
and b.ds='20150801';
```



The execution plan indicates that partition pruning is effective for the right table, but not the left table.

- For a query where partition pruning conditions are specified in the WHERE clause:

```
set odps.sql.allow.fullscan=true;
explain
select a.seller_id
    ,a.pay_ord_pbt_1d_001
from xxxxx_trd_slr_ord_1d a
left outer join
    xxxxx_seller b
on a.seller_id=b.user_id
where a.ds='20150801'
and b.ds='20150801';
```

```
1 Task M2_Stg1:
    Data source: [illegible]eller/ds=20150801
    TS: alias: b
        FIL: EQUAL(b.ds, '20150801')
            RS: order: +
                optimizeOrderBy: False
                valueDestLimit: 0
                keys:
                    b.user_id
                values:
                partitions:
                    b.user_id

In Task J3_1_2_Stg1:
    JOIN: a LEFT OUTER JOIN unknown
         filter:
                0:
                1:
        SEL: a._col0, a._col20
            FS: output: None

In Task M1_Stg1:
    Data source: [illegible]trd_slr_ord_1d/ds=20150801
    TS: alias: a
```

The execution plan indicates that partition pruning is effective for both tables.

- RIGHT OUTER JOIN

  A RIGHT OUTER JOIN operation is similar to a LEFT OUTER JOIN operation. If partition pruning conditions are specified in the ON clause, partition pruning is effective only for the left table, but not the right table. If partition pruning conditions are specified in the WHERE clause, partition pruning is effective for both tables.

- FULL OUTER JOIN

  Partition pruning is effective only when partition pruning conditions are specified in the WHERE clause, but not the ON clause.

Impact and consideration

- If partition pruning does not take effect, the query performance can be greatly deteriorated. This

issue can be hardly discovered. We recommend that you check whether partition pruning is effective before you commit the code.

- To use UDFs for partition pruning, you must modify the classes of the UDFs or add `set odps.sql.udf.ppr.deterministic = true;` before the SQL statements to execute. For more information, see WHERE clause (where_condition).

1.5. Query the first N data records of each group

This topic describes how to group data records and query the first N data records.

Sample data

empno	ename	job	sal
7369	SMITH	CLERK	800.0
7876	SMITH	CLERK	1100.0
7900	JAMES	CLERK	950.0
7934	MILLER	CLERK	1300.0
7499	ALLEN	SALESMAN	1600.0
7654	MARTIN	SALESMAN	1250.0
7844	TURNER	SALESMAN	1500.0
7521	WARD	SALESMAN	1250.0

Implementation

You can use one of the following methods to query the first N data records of each group:

- Query the row ID of each record and use the WHERE clause to filter the records.

```
SELECT * FROM (
  SELECT empno
  , ename
  , sal
  , job
  , ROW_NUMBER() OVER (PARTITION BY job ORDER BY sal) AS rn
  FROM emp
) tmp
WHERE rn < 10;
```

- Use the SPLIT function.

  For more information, see the last example in MaxCompute learning plan. This method can be used to determine the sequence number of a data record. If the sequence number is greater than the specified number, such as 10, the data records that remain are no longer processed. This improves computing efficiency.

1.6. Merge multiple rows of data into one row

This topic describes how to use SQL statements to merge multiple rows of data into one row.

Sample data

class	gender	name
1	M	LiLei
1	F	HanMM
1	M	Jim
1	F	HanMM
2	F	Kate
2	M	Peter

Examples

- Example 1: Execute the following statement to merge the rows whose values in the class column are the same into one row based on the values in the name column and deduplicate the values in the name column. You can implement the deduplication by using nested subqueries.

```
SELECT class, wm_concat(distinct ',', name) FROM students GROUP BY class;
```

> **Note** The `wm_concat` function is used to aggregate data. For more information, see Aggregate functions.

The following result is returned.

class	names
1	LiLei,HanMM,Jim
2	Kate,Peter

- Example 2: Execute the following statement to collect statistics on the numbers of males and females based on the values in the class column:

```
SELECT
class
,SUM(CASE WHEN gender = 'M' THEN 1 ELSE 0 END) AS cnt_m
,SUM(CASE WHEN gender = 'F' THEN 1 ELSE 0 END) AS cnt_f
FROM students
GROUP BY class;
```

The following result is returned.

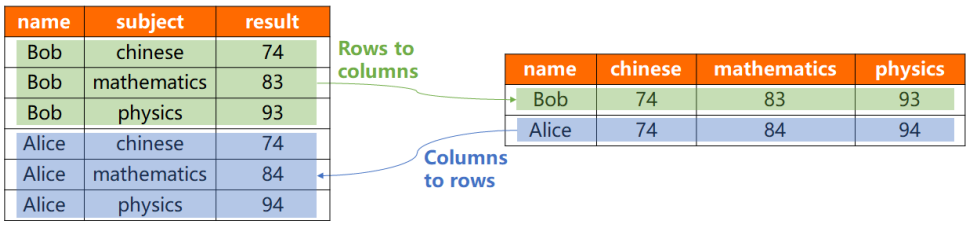
class	cnt_m	cnt_f
1	2	2
2	1	1

1.7. Transpose rows to columns or columns to rows

This topic describes how to use SQL statements to transpose rows to columns or columns to rows.

Background information

The following figure shows the effect of transposing rows to columns or columns to rows.



- Rows to columns

  Transpose multiple rows to one row, or transpose one column to multiple columns.

- Columns to rows

  Transpose one row to multiple rows, or transpose multiple columns to one column.

Sample data

Sample source data is provided for you to better understand the examples of transposing rows to columns or columns to rows.

- Create a source table and insert data into the source table. The table is used to transpose **rows to columns**. Sample statements:

```
create table rowtocolumn (name string, subject string, result bigint);
insert into table rowtocolumn values
('Bob' , 'chinese' , 74),
('Bob' , 'mathematics' , 83),
('Bob' , 'physics' , 93),
('Alice' , 'chinese' , 74),
('Alice' , 'mathematics' , 84),
('Alice' , 'physics' , 94),
```

- Create a source table and insert data into the source table. The table is used to transpose **columns to rows**. Sample statements:

```
create table columntorow (name string, chinese bigint, mathematics bigint, physics bigint
);
insert into table columntorow values
('Bob' , 74, 83, 93),
('Alice' , 74, 84, 94);
```

Examples of transposing rows to columns

You can use one of the following methods to transpose rows to columns:

- Method 1: Use the `CASE WHEN` expression to extract the values of each subject as separate columns. Sample statement:

  ```
  select name as name,
         max(case subject when 'chinese' then result end) as chinese,
         max(case subject when 'mathematics' then result end) as mathematics,
         max(case subject when 'physics' then result end) as physics
  from rowtocolumn
  group by name;
  ```

  The following result is returned:

  ```
  +--------+------------+------------+------------+
   name    chinese       mathematics      physics
  +--------+------------+------------+------------+
   Bob    74         83         93
   Alice    74         84         94
  +--------+------------+------------+------------+
  ```

- Method 2: Use built-in functions to transpose rows to columns. Merge the values of the subject and result columns into one column by using the CONCAT and WM_CONCAT functions. Then, parse the values of the subject column as separate columns by using the KEYVALUE function. Sample statement:

  ```
  select name as name,
         keyvalue(subject, chinese') as chinese,
         keyvalue(subject, 'mathematics') as mathematics,
         keyvalue(subject, 'physics') as physics
  from(
       select name, wm_concat(';',concat(subject,':',result))as subject
       from rowtocolumn
       group by name);
  ```

  The following result is returned:

  ```
  +--------+------------+------------+------------+
   name    chinese       mathematics      physics
  +--------+------------+------------+------------+
   Bob    74         83         93
   Alice    74         84         94
  +--------+------------+------------+------------+
  ```

Examples of transposing columns to rows

You can use one of the following methods to transpose columns to rows:

- Method 1: Use the `UNION ALL` clause to combine the values in chinese, mathematics, and physics columns into one column. Sample statements:

```
-- Remove the limit on the simultaneous execution of the ORDER BY and LIMIT clauses. This
way, you can use ORDER BY to sort the results by name.
set odps.sql.validate.orderby.limit=false;
-- Transpose columns to rows.
select name as name, subject as subject, result as result
from(
     select name, 'chinese' as subject, chinese as result from columntorow
     union all
     Choose name, 'mathematics' as subject, mathematics as result from columntorow
     union all
     select name, 'physics' as subject, physics as result from columntorow)
order by name;
```

  The following result is returned:

```
+--------+--------+------------+
 name    subject    result
+--------+--------+------------+
 Bob    chinese    74
 Bob    mathematics    83
 Bob    physics    93
 Alice   chinese    74
 Alice    mathematics    84
 Alice    physics    94
+--------+--------+------------+
```

- Method 2: Use built-in functions to transpose columns to rows. Concatenate the column name of each subject and the values in each column by using the CONCAT function. Then, split the concatenated values into the subject and result columns as separate columns by using the TRANS_ARRAY and SPLIT_PART functions. Sample statement:

```
select name as name,
       split_part(subject,':',1) as subject,
       split_part(subject,':',2) as result
from(
       select trans_array(1,';',name,subject) as (name,subject)
       from(
            select name,
        concat('chinese',':',chinese,';', 'mathematics ',':',mathematics,';', 'physics','
:',physics) as subject
            from columntorow)tt)tx;
```

  The following result is returned:

```
+--------+--------+------------+
 name    subject    result
+--------+--------+------------+
 Bob    chinese    74
 Bob    mathematics    83
 Bob    physics    93
 Alice    chinese    74
 Alice    mathematics    84
 Alice    physics    94
+--------+--------+------------+
```

1.8. JOIN operations in MaxCompute SQL

This topic describes the JOIN operations that MaxCompute SQL supports.

Overview

The following table describes the JOIN operations that MaxCompute SQL supports.

Operation	Description
INNER JOIN	Returns the rows that have matching column values in both the left table and the right table based on the join condition.
LEFT JOIN	Returns all the rows from the left table and matched rows from the right table based on the join condition. If a row in the left table has no matching rows in the right table, NULL values are returned in the columns from the right table in the result set.
RIGHT JOIN	Returns all the rows from the right table and matched rows from the left table based on the join condition. If a row in the right table has no matching rows in the left table, NULL values are returned in the columns from the left table in the result set.
FULL JOIN	Returns all the rows in both the left table and the right table whether the join condition is met or not. In the result set, NULL values are returned in the columns from the table that lacks a matching row in the other table.
LEFT SEMI JOIN	Returns only the rows in the left table that have a matching row in the right table.
LEFT ANTI JOIN	Returns only the rows in the left table that have no matching rows in the right table.

The ON clause and the WHERE clause can be used in the same SQL statement. For example, consider the following SQL statement:

```
(SELECT * FROM A WHERE {subquery_where_condition} A) A
JOIN
(SELECT * FROM B WHERE {subquery_where_condition} B) B
ON {on_condition}
WHERE {where_condition}
```

The conditions in the preceding SQL statement are evaluated in the following order:

1. The `{subquery_where_condition}` condition in the WHERE clause of the subqueries
2. The `{on_condition}` condition in the ON clause
3. The `{where_condition}` condition in the WHERE clause after the JOIN clause

Therefore, a JOIN operation may return different results, depending on whether the filter conditions are specified in `{subquery_where_condition}`, `{on_condition}`, or `{where_condition}`. For more information, see Case-by-case study.

Test tables

- Table A

  Execute the following statement to create Table A:

  ```
  CREATE TABLE A AS SELECT * FROM VALUES (1, 20180101),(2, 20180101),(2, 20180102) t (key, ds);
  ```

  Table A has the following three rows and is used as the left table for all JOIN operations in this topic.

key	ds
1	20180101
2	20180101
2	20180102

- Table B

  Execute the following statement to create Table B:

  ```
  CREATE TABLE B AS SELECT * FROM VALUES (1, 20180101),(3, 20180101),(2, 20180102) t (key, ds);
  ```

  Table B has the following three rows and is used as the right table for all JOIN operations in this topic.

key	ds
1	20180101
3	20180101
2	20180102

- Cartesian product of Table A and Table B

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
1	20180101	3	20180101
1	20180101	2	20180102
2	20180101	1	20180101
2	20180101	3	20180101
2	20180101	2	20180102
2	20180102	1	20180101
2	20180102	3	20180101
2	20180102	2	20180102

Case-by-case study

- INNER JOIN

  An INNER JOIN operation first takes the Cartesian product of the rows in Table A and Table B and returns rows that have matching column values in Table A and Table B based on the ON clause.

  Conclusion: An INNER JOIN operation returns the same results independently of whether the filter conditions are specified in `{subquery_where_condition}`, `{on_condition}`, or `{where_condition}`.

  - Case 1: Specify the filter conditions in the `{subquery_where_condition}` clause, as shown in the following statement:

    ```
    SELECT A.*, B.*
    FROM
    (SELECT * FROM A WHERE ds='20180101') A
    JOIN
    (SELECT * FROM B WHERE ds='20180101') B
    ON a.key = b.key;
    ```

    The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101

- Case 2: Specify the filter conditions in the `{on_condition}` clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM A JOIN B
ON a.key = b.key and A.ds='20180101' and B.ds='20180101';
```

The Cartesian product of Table A and Table B contains nine rows, of which only one meets the join condition. The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101

- Case 3: Specify the filter conditions in the WHERE clause after the ON clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM A JOIN B
ON a.key = b.key
WHERE A.ds='20180101' and B.ds='20180101';
```

The Cartesian product of Table A and Table B contains nine rows, of which only three meet the join condition. The following table lists the result set.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180102	2	20180102
2	20180101	2	20180102

The query processor then filters the preceding result set based on the `A.ds='20180101' and B.ds='20180101'` filter condition. The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101

- LEFT JOIN

A LEFT JOIN operation first takes the Cartesian product of the rows in Table A and Table B and returns all the rows of Table A and rows in Table B that meet the join condition. If the join condition finds no matching rows in Table B for a row in Table A, the row in Table A is returned in the result set with NULL values in each column from Table B.

Conclusion: A LEFT JOIN operation may return different results, depending on whether the filter conditions are specified in `{subquery_where_condition}`, `{on_condition}`, or `{where_condition}`:

- The operation returns the same results, regardless of whether the filter condition for Table A is specified in `{subquery_where_condition}` or `{where_condition}`.

- The operation returns the same results, regardless of whether the filter condition for Table B is specified in `{subquery_where_condition}` or `{on_condition}`.
- Case 1: Specify the filter conditions in the `{subquery_where_condition}` clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM
(SELECT * FROM A WHERE ds='20180101') A
LEFT JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key;
```

The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180101	NULL	NULL

- Case 2: Specify the filter conditions in the `{on_condition}` clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM A LEFT JOIN B
ON a.key = b.key and A.ds='20180101' and B.ds='20180101';
```

The Cartesian product of Table A and Table B contains nine rows, of which only one meets the join condition. The other two rows in Table A do not have matching rows in Table B. Therefore, NULL values are returned in the columns from Table B for the two rows in Table A. The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180101	NULL	NULL
2	20180102	NULL	NULL

- Case 3: Specify the filter conditions in the WHERE clause after the ON clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM A LEFT JOIN B
ON a.key = b.key
WHERE A.ds='20180101' and B.ds='20180101';
```

The Cartesian product of Table A and Table B contains nine rows, of which only three meet the join condition. The following table lists the result set.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180101	2	20180102
2	20180102	2	20180102

The query processor then filters the preceding result set based on the `A.ds='20180101' and B.ds='20180101'` filter condition. The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101

- RIGHT JOIN

A RIGHT JOIN operation is similar to a LEFT JOIN operation, except that the two tables are used in a reversed manner. A RIGHT JOIN operation returns all the rows of Table B and rows in Table A that meet the join condition.

- Conclusion: A RIGHT JOIN operation may return different results, depending on whether the filter conditions are specified in `{subquery_where_condition}`, `{on_condition}`, or `{where_condition}`.
- The operation returns the same results, regardless of whether the filter condition for Table B is specified in `{subquery_where_condition}` or `{where_condition}`.
- The operation returns the same results, regardless of whether the filter condition for Table A is specified in `{subquery_where_condition}` or `{on_condition}`.

- FULL JOIN

A FULL JOIN operation takes the Cartesian product of the rows in Table A and Table B and returns all the rows in Table A and Table B, whether the join condition is met or not. In the result set, NULL values are returned in the columns from the table that lacks a matching row in the other table.

Conclusion: A FULL JOIN operation may return different results, depending on whether the filter conditions are specified in `{subquery_where_condition}`, `{on_condition}`, or `{where_condition}`.

- Case 1: Specify the filter conditions in the `{subquery_where_condition}` clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM
(SELECT * FROM A WHERE ds='20180101') A
FULL JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key;
```

The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180101	NULL	NULL
NULL	NULL	3	20180101

- Case 2: Specify the filter conditions in the `{on_condition}` clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM A FULL JOIN B
ON a.key = b.key and A.ds='20180101' and B.ds='20180101';
```

The Cartesian product of Table A and Table B contains nine rows, of which only one meets the join condition. In the result set, for the two rows in Table A that match no rows in Table B, NULL values are returned in the columns from Table B. For the two rows in Table B that match no rows in Table A, NULL values are returned in the columns from Table A. The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180101	NULL	NULL
2	20180102	NULL	NULL
NULL	NULL	3	20180101
NULL	NULL	2	20180102

- Case 3: Specify the filter conditions in the WHERE clause after the ON clause, as shown in the following statement:

```
SELECT A.*, B.*
FROM A FULL JOIN B
ON a.key = b.key
WHERE A.ds='20180101' and B.ds='20180101';
```

The Cartesian product of Table A and Table B contains nine rows, of which only three meet the join condition.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180101	2	20180102
2	20180102	2	20180102

The row in Table B that has no matching rows in Table A is returned in the result set, with NULL values in the columns from Table A for that row. The following table lists the result set.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101
2	20180101	2	20180102
2	20180102	2	20180102
NULL	NULL	3	20180101

The query processor then filters the preceding result set based on the `A.ds='20180101' and B.ds='20180101'` filter condition. The following table lists the results that the preceding statement returns.

a.key	a.ds	b.key	b.ds
1	20180101	1	20180101

- LEFT SEMI JOIN

  A LEFT SEMI JOIN operation returns only the rows in Table A that have a matching row in Table B. A LEFT SEMI JOIN operation does not return rows from Table B. Therefore, you cannot specify a filter condition for Table B in the WHERE clause after the ON clause.

  Conclusion: A LEFT SEMI JOIN operation returns the same results independently of whether the filter conditions are specified in `{subquery_where_condition}`, `{on_condition}`, or `{where_condition}`.

- Case 1: Specify the filter conditions in the {subquery_where_condition} clause, as shown in the following statement:

```
SELECT A.*
FROM
(SELECT * FROM A WHERE ds='20180101') A
LEFT SEMI JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key;
```

The following table lists the results that the preceding statement returns.

a.key	a.ds
1	20180101

- Case 2: Specify the filter conditions in the `{on_condition}` clause, as shown in the following statement:

```
SELECT A.*
FROM A LEFT SEMI JOIN B
ON a.key = b.key and A.ds='20180101' and B.ds='20180101';
```

The following table lists the results that the preceding statement returns.

a.key	a.ds
1	20180101

- Case 3: Specify the filter conditions in the WHERE clause after the ON clause, as shown in the following statement:

```
SELECT A.*
FROM A LEFT SEMI JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key
WHERE A.ds='20180101';
```

The following table lists the result set.

a.key	a.ds
1	20180101

The query processor then filters the preceding result set based on the `A.ds='20180101'` filter condition. The following table lists the results that the preceding statement returns.

a.key	a.ds
1	20180101

- LEFT ANTI JOIN

A LEFT ANTI JOIN operation returns only the rows in Table A that have no matching rows in Table B. A LEFT ANTI JOIN operation does not return rows from Table B. Therefore, you cannot specify a filter condition for Table B in the WHERE clause after the ON clause. A LEFT ANTI JOIN operation is usually used to replace the NOT EXISTS syntax.

Conclusion: A LEFT ANTI JOIN operation may return different results, depending on whether the filter conditions are specified in `{subquery_where_condition}`, `{on_condition}`, or `{where_condition}`.

- The operation returns the same results, regardless of whether the filter condition for Table A is specified in `{subquery_where_condition}` or `{where_condition}`.
- The operation returns the same results, regardless of whether the filter condition for Table B is specified in `{subquery_where_condition}` or `{on_condition}`.
- Case 1: Specify the filter conditions in the `{subquery_where_condition}` clause, as shown in the following statement:

```
SELECT A.*
FROM
(SELECT * FROM A WHERE ds='20180101') A
LEFT ANTI JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key;
```

The following table lists the results that the preceding statement returns.

a.key	a.ds
2	20180101

- Case 2: Specify the filter conditions in the `{on_condition}` clause, as shown in the following statement:

```
SELECT A.*
FROM A LEFT ANTI JOIN B
ON a.key = b.key and A.ds='20180101' and B.ds='20180101';
```

The following table lists the results that the preceding statement returns.

a.key	a.ds
2	20180101
2	20180102

- Case 3: Specify the filter conditions in the WHERE clause after the ON clause, as shown in the following statement:

```
SELECT A.*
FROM A LEFT ANTI JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key
WHERE A.ds='20180101';
```

The following table lists the result set.

a.key	a.ds
2	20180101
2	20180102

The query processor then filters the preceding result set based on the `A.ds='20180101'` filter condition. The following table lists the results that the preceding statement returns.

a.key	a.ds
2	20180101

Usage notes

- For an INNER JOIN operation or a LEFT SEMI JOIN operation, an SQL statement returns the same results, regardless of where you specify filter conditions for the left table and the right table.
- For a LEFT JOIN operation or a LEFT ANTI JOIN operation, the filter condition for the left table functions the same whether it is specified in `{subquery_where_condition}` or `{where_condition}`. The filter condition for the right table functions the same whether it is specified in `{subquery_where_condition}` or `{on_condition}`.
- For a RIGHT JOIN operation, the filter condition for the left table functions the same whether it is specified in `{subquery_where_condition}` or `{on_condition}`. The filter condition for the right table functions the same whether it is specified in `{subquery_where_condition}` or `{where_condition}`.
- For a FULL OUTER JOIN operation, filter conditions can be specified only in `{subquery_where_condition}`.

2.Data migration

2.1. Overview

This topic describes the best practices for data migration, including migrating business data or log data from other business platforms to MaxCompute or migrating data from MaxCompute to other business platforms.

Background information

Traditional relational databases are not suitable for processing a large amount of data. If you have a large amount of data stored in a traditional relational database, you can migrate the data to MaxCompute.

MaxCompute provides a comprehensive set of data migration solutions and a variety of classic distributed computing models, allowing you to store a large amount of data and compute data fast. By using MaxCompute, you can efficiently save costs for your enterprise.

DataWorks provides comprehensive features for MaxCompute, such as data integration, data analytics, data management, and data administration. Among these features, data integration enables stable, efficient, and scalable data synchronization.

Best practices

- Migrate business data from other business platforms to MaxCompute
  - Migrate data across DataWorks workspaces. For more information, see Migrate data across DataWorks workspaces.
  - Migrate data from Hadoop to MaxCompute. For more information, see Best practices of migrating data from Hadoop to MaxCompute. For more information about the issues that you may encounter during data and script migration and the solutions, see Practices of migrating data from a user-created Hadoop cluster to MaxCompute.
  - Migrate data from Oracle to MaxCompute. For more information, see Migrate data from Oracle to MaxCompute.
  - Migrate data from a Kafka cluster to MaxCompute. For more information, see Migrate data from a Kafka cluster to MaxCompute.
  - Migrate data from an Elasticsearch cluster to MaxCompute. For more information, see Migrate data from an Elasticsearch cluster to MaxCompute.
  - Migrate data from RDS to MaxCompute. For more information, see Migrate data from RDS to MaxCompute to implement dynamic partitioning.
  - Migrate JSON data from Object Storage Service (OSS) to MaxCompute. For more information, see Migrate JSON data from OSS to MaxCompute.
  - Migrate JSON data from MongoDB to MaxCompute. For more information, see Migrate JSON data from MongoDB to MaxCompute.
  - Migrate data from a user-created MySQL database on an Elastic Compute Service (ECS) instance to MaxCompute. For more information, see Migrate data from a user-created MySQL database on an ECS instance to MaxCompute.
- Migrate log data from other business platforms to MaxCompute

  - Use Tunnel to migrate log data to MaxCompute. For more information, see Use Tunnel to upload log data to MaxCompute.
  - Use DataHub to migrate log data to MaxCompute. For more information, see Use DataHub to migrate log data to MaxCompute.
  - Use DataWorks to migrate log data to MaxCompute. For more information, see Use DataWorks Data Integration to migrate log data to MaxCompute.
- Migrate data from MaxCompute to other business platforms
  - Migrate data from MaxCompute to OSS. For more information, see Migrate data from MaxCompute to OSS.
  - Migrate data from MaxCompute to Tablestore. For more information, see Migrate data from MaxCompute to Tablestore.

After the business data and log data are processed by MaxCompute, you can use Quick BI to present the data processing results in a visualized manner. For more information, see Best practices of using MaxCompute to process data and Quick BI to present the data processing results.

2.2. Migrate data across DataWorks workspaces

This topic describes how to migrate data across DataWorks workspaces in the same region.

Prerequisites

All the steps in the tutorial Build an online operation analysis platform are completed. For more information, see Business scenarios and development process.

Context

This topic uses the bigdata_DOC workspace created in the tutorial Build an online operation analysis platform as the source workspace. You need to create a destination workspace to store the tables, resources, configurations, and data synchronized from the source workspace.

Procedure

1. Create a destination workspace.
   i. Log on to the DataWorks console. In the left-side navigation pane, click **Workspaces**.
   ii. On the Workspaces page that appears, select the **China (Hangzhou)** region in the upper-left corner and click **Create Workspace**.
   iii. In the **Create Workspace** pane that appears, set parameters in the **Basic Settings** step and click **Next**.

Section	Parameter	Description
	Workspace Name	The name of the workspace. The name must be 3 to 23 characters in length and can contain letters, underscores (_), and digits. The name must start with a letter.

Section	Parameter	Description
Basic Information	Display Name	The display name of the workspace. The display name can be a maximum of 23 characters in length. It can contain letters, underscores (_), and digits and must start with a letter.
	Mode	The mode of the workspace. Valid values: **Basic Mode (Production Environment Only)** and **Standard Mode (Development and Production Environments)**. ■ **Basic Mode (Production Environment Only)**: A workspace in basic mode is associated with only one MaxCompute project. Workspaces in basic mode do not isolate the development environment from the production environment. In these workspaces, you can perform only basic data development and cannot strictly control the data development process and the permissions on tables. ■ **Standard Mode (Development and Production Environments)**: A workspace in standard mode is associated with two MaxCompute projects. One serves as the development environment, and the other serves as the production environment. Workspaces in standard mode allow you to develop code in a standard way and strictly control the permissions on tables. These workspaces impose limits on table operations in the production environment for data security. For more information, see Basic mode and standard mode.
	Description	The description of the workspace.
Advanced Settings	Download SELECT Query Result	Specifies whether the query results that are returned by SELECT statements in DataStudio can be downloaded. If you turn off this switch, the query results cannot be downloaded. You can change the setting of this parameter for the workspace in the Workspace Settings panel after the workspace is created. For more information, see Configure security settings.

The source workspace bigdata_DOC is in the basic mode. For convenience, set Mode to **Basic Mode (Production Environment Only)** in the Basic Settings step when you create a destination workspace.

Set Workspace Name to a globally unique name. We recommend that you use a name that is easy to distinguish. In this example, set Workspace Name to clone_test_doc.

iv. In the Select Engines and Services step, select the MaxCompute check box and Pay-As-You-Go in the **Compute Engines** section and click **Next**.

v. In the **Engine Details** step, set the required parameters and click **Create Workspace**.

Compute engine	Parameter	Description
MaxCompute	**Instance Display Name**	The display name of the compute engine instance. The display name must be 3 to 27 characters in length, and can contain only letters, underscores (_), and digits. It must start with a letter.
	MaxCompute Project Name	The name of the MaxCompute project. By default, the name is the same as that of the DataWorks workspace.
	Account for Accessing MaxCompute	The identity used to access the MaxCompute project. For the development environment, the value is fixed to **Task owner**. For the production environment, the valid values are **Alibaba Cloud primary account** and **Alibaba Cloud sub-account**.
	Resource Group	The quotas of computing resources and disk spaces for the compute engine instance.

2. Clone node configurations and resources across workspaces.

   You can use the **cross-workspace cloning** feature of DataWorks to clone the node configurations and resources from the bigdata_DOC workspace to the clone_test_doc workspace. For more information, see Clone nodes across workspaces.

   > Note
   > - The cross-workspace cloning feature cannot clone table schemas or data.
   > - The cross-workspace cloning feature cannot clone combined nodes. If the destination workspace needs to use the combined nodes that exist in the source workspace, you need to manually create the combined nodes in the destination workspace.

   i. Go to the bigdata_DOC workspace and click **Cross-project cloning** in the upper-right corner. The Create Clone Task page appears.

ii. Set **Target Workspace** to clone_test_doc and **Workflow** to Workshop that needs to be cloned. Select all the nodes in the workflow and click **Add to List**. Click **To-Be-Cloned Node List** in the upper-right corner.

iii. In the Nodes to Clone pane that appears, click **Clone All**. The selected nodes are cloned to the clone_test_doc workspace.

iv. Go to the destination workspace and check whether the nodes are cloned.

3. Create tables.

   The cross-workspace cloning feature cannot clone table schemas. Therefore, you need to manually create required tables in the destination workspace.

   - For non-partitioned tables, we recommend that you use the following SQL statement to synchronize the table schema from the source workspace:

     ```
     create table table_name as select * from Source workspace. Table name;
     ```

   - For partitioned tables, we recommend that you use the following SQL statement to synchronize the table schema from the source workspace:

     ```
     create table table_name partitioned by (Partition key column string);
     ```

   Commit the tables to the production environment. For more information, see Create tables.

4. Synchronize data.

   The cross-workspace cloning feature cannot clone data from the source workspace to the destination workspace. You need to manually synchronize required data to the destination workspace. To synchronize the data of the rpt_user_trace_log table from the source workspace to the destination workspace, follow these steps:

   i. Create a connection.

      a. Go to the **Data Integration** page and click **Connection** in the left-side navigation pane.

      b. On the **Data Source** page that appears, click **Add a Connection** in the upper-right corner. In the Add Connection dialog box that appears, select **MaxCompute(ODPS)** in the Big Data Storage section.

      c. In the Add MaxCompute(ODPS) Connection dialog box that appears, set **Connection Name**, **MaxCompute Project Name**, **AccessKey ID**, and **AccessKey Secret**, and click **Complete**. For more information, see Add a MaxCompute data source.

   ii. Create a batch sync node.

      a. Go to the **DataStudio** page, click the Data Analytics tab, and then click Workshop under **Business Flow**. Right-click **Data Integration** and choose **Create > Batch Synchronization** to create a batch sync node.

      b. On the configuration tab of the batch sync node, set the required parameters. In this example, set Connection under Source to bigdata_DOC and that under Target to **odps_first**. Set Table to rpt_user_trace_log. After the configuration is completed, click the **Properties** tab in the right-side navigation pane.

      c. Click **Use Root Node** in the Dependencies section and commit the batch sync node.

iii. Generate retroactive data for the batch sync node.

a. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.

b. On the page that appears, choose **Cycle Task Maintenance > Cycle Task** in the left-side navigation pane.

c. On the page that appears, find the batch sync node you created in the node list and click the node name. On the canvas that appears on the right, right-click the batch sync node and choose **Run > Current Node Retroactively**.

d. In the Patch Data dialog box that appears, set the required parameters. In this example, set Data Timestamp to Jun 11, 2019 - Jun 17, 2019 to synchronize data from multiple partitions. Click **OK**.

e. On the **Patch Data** page that appears, check the running status of the retroactive instances that are generated. If **Successful** appears in the **STATUS** column of a retroactive instance, the instance is run and the corresponding data is synchronized.

iv. Verify the data synchronization.

On the Data Analytics tab of the DataStudio page, right-click the Workshop workflow under Business Flow and choose **Create > MaxCompute > ODPS SQL** to create an ODPS SQL node. On the configuration tab of the ODPS SQL node, run the following SQL statement to check whether data is synchronized to the destination workspace:

```
select * from rpt_user_trace_log where dt BETWEEN '20190611' and '20190617';
```

2.3. Synchronize data from Hadoop to MaxCompute

This topic describes how to use the data synchronization feature of DataWorks to synchronize data from Hadoop Distributed File System (HDFS) to MaxCompute. Data synchronization between MaxCompute and Hadoop or Spark is supported.

Prerequisites

- MaxCompute is activated. A MaxCompute project is created.

  In this example, a project named bigdata_DOC in the China (Hangzhou) region is used. For more information, see Activate MaxCompute and DataWorks.

- A Hadoop cluster is created.

  Before you synchronize data, you must make sure that your Hadoop cluster can work as expected. In this example, Alibaba Cloud E-MapReduce (EMR) is used to create the Hadoop cluster. For more information, see Create a cluster.

  In this example, the following configurations are used for the EMR Hadoop cluster:

  - EMR version: EMR V3.11.0
  - Cluster type: Hadoop
  - Software: HDFS 2.7.2, YARN 2.7.2, Hive 2.3.3, Ganglia 3.7.2, Spark 2.2.1, Hue 4.1.0, Zeppelin 0.7.3, Tez 0.9.1, Sqoop 1.4.6, Pig 0.14.0, ApacheDS 2.0.0, and Knox 0.13.0

The EMR Hadoop cluster is a non-high availability (HA) cluster that is deployed on the classic network in the China (Hangzhou) region. A public IP address and a private IP address are configured for the Elastic Compute Service (ECS) instance in the master node group of the EMR Hadoop cluster.

Step 1: Prepare test data

1. Create test data in the EMR Hadoop cluster.
   i. Log on to the EMR console by using your Alibaba Cloud account.
   ii. In the EMR console, click the Data Platform tab. On the Data Platform tab, find the desired project and create a job named doc in the project. In the job that you created, execute a table creation statement to create a table. In this example, the following statement is used to create a table named hive_doc_good_sale in the EMR Hadoop cluster. For more information about how to create an EMR job, see Edit jobs.

```
CREATE TABLE IF NOT
EXISTS hive_doc_good_sale(
   create_time timestamp,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   )
   PARTITIONED BY (pt string) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' lines terminated by '\n';
```

   iii. Click **Run** in the upper-right corner of the code editor on the Data Platform tab. If the `Query executed successfully` message appears, the table hive_doc_good_sale is created in the EMR Hadoop cluster.

   Create a table

   iv. Insert test data into the table. You can import test data from Object Storage Service (OSS) or other data sources to the table. You can also manually insert test data into the table. In this example, the following statement is used to manually insert test data into the table:

```
insert into
hive_doc_good_sale PARTITION(pt =1 ) values('2018-08-21','Coat','Brand A','lilei',3
,500.6,7),('2018-08-22','Fresh food','Brand B','lilei',1,303,8),('2018-08-22','Coat
','Brand C','hanmeimei',2,510,2),(2018-08-22,'Bathroom product','Brand A','hanmeime
i',1,442.5,1),('2018-08-22','Fresh food','Brand D','hanmeimei',2,234,3),('2018-08-2
3','Coat','Brand B','jimmy',9,2000,7),('2018-08-23','Fresh food','Brand A','jimmy',
5,45.1,5),('2018-08-23','Coat','Brand E','jimmy',5,100.2,4),('2018-08-24','Fresh fo
od','Brand G','peiqi',10,5560,7),('2018-08-24','Bathroom product','Brand F','peiqi'
,1,445.6,2),('2018-08-24','Coat','Brand A','ray',3,777,3),('2018-08-24','Bathroom p
roduct','Brand G','ray',3,122,3),('2018-08-24','Coat','Brand C','ray',1,62,7) ;
```

   v. After you insert the data into the table, execute the `select * from hive_doc_good_sale where pt =1;` statement to check whether the data exists in the table that you created in the EMR Hadoop cluster.

   Check the inserted data

2. Create a MaxCompute table in the DataWorks console.

i.

ii.

iii.

iv.

v.

vi.

vii. In the **DDL Statement** dialog box, enter the following table creation statement and click **Generate Table Schema**. In the Confirm message, click OK. In this example, the following table creation statement is used to create a MaxCompute table named hive_doc_good_sale:

```
CREATE TABLE IF NOT EXISTS hive_doc_good_sale(
   create_time string,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   )
   PARTITIONED BY (pt string) ;
```

When you create the table, you must consider the mappings between Hive data types and MaxCompute data types. For more information about the mappings, see Data type mappings.

You can also use the MaxCompute client odpscmd to create a MaxCompute table. For more information about how to install and configure the MaxCompute client, see Install and configure the MaxCompute client.

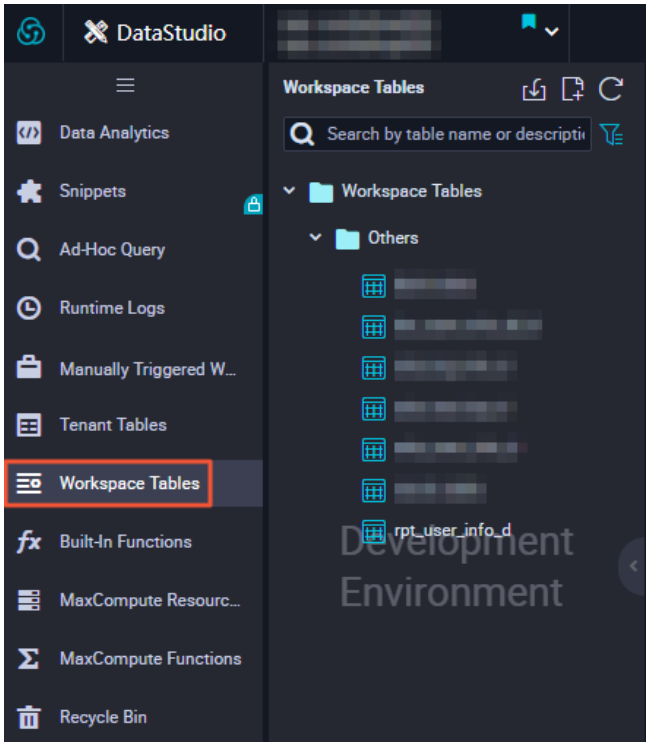
```
odps@ bigdata_DOC>CREATE TABLE IF NOT EXISTS hive_doc_good_sale(
   create_time timestamp,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   )
   PARTITIONED BY (pt string) ;
                >               >               >               >
        >               >               >               >
 >
ID = 20180906110540873gev1bpim
OK
odps@ bigdata_DOC>drop table hive_doc_good_sale;
Confirm to "drop table hive_doc_good_sale;" (yes/no)? yes

ID = 20180906110825180gxh66292
```

Note If you need to resolve compatibility issues between Hive data types and MaxCompute data types, we recommend that you run the following commands on the MaxCompute client:

```
set odps.sql.type.system.odps2=true;
set odps.sql.hive.compatible=true;
```

viii. Click **Commit to Production Environment**. The table is created.

ix. In the left-side navigation pane of the DataStudio page, click **Workspace Tables**. In the Workspace Tables pane, view the MaxCompute table that you created.



Step 2: Synchronize data

1. Create a custom resource group.

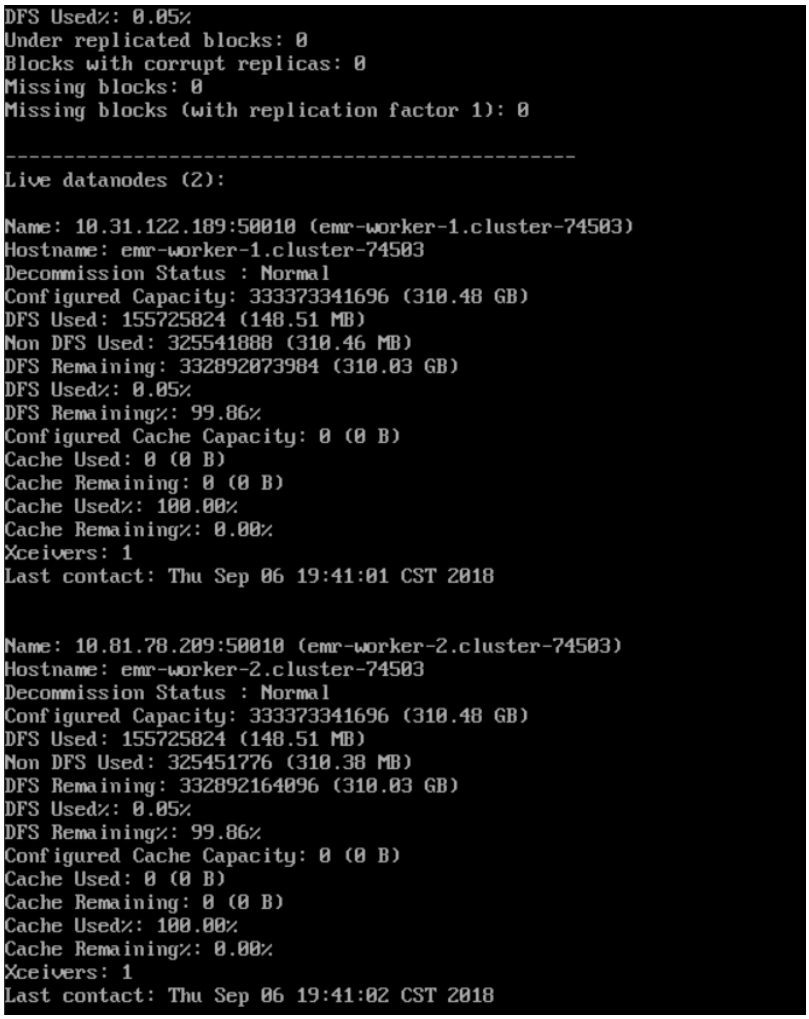
   In most cases, the network where a MaxCompute project resides is inaccessible to a data node in a Hadoop cluster. To resolve the connectivity issue, you can create a custom resource group to run your DataWorks synchronization node on the master node of the Hadoop cluster. In most cases, the master node and data nodes in a Hadoop cluster are connected.

   i. View information about the data nodes of the EMR Hadoop cluster.

      a. Log on to the EMR console. Click the **Cluster Management** tab.

      b. On the Cluster Management tab, find the EMR Hadoop cluster that you created and click the name of the cluster. On the Clusters and Services page, click **Instances** in the left-side navigation pane. On the Instances page, view information about the data nodes of the EMR Hadoop cluster.

You can also click the ECS instance ID of the master node to go to the Instance Details tab of the ECS instance in the ECS console. In the Basic Information section of the Instance Details tab, click Connect to log on to the ECS instance and run the `hadoop dfsadmin -report` command to view the information about the data nodes.



> **Note** In this example, each data node has only a private IP address and cannot communicate with the default resource group of DataWorks. Therefore, you must create a custom resource group to run your DataWorks synchronization node on the master node.

ii. Create a custom resource group.

a. Log on to the DataWorks console. In the left-side navigation pane, click Workspaces. On the Workspaces page, find the workspace in which you want to create a custom resource group and click **Data Integration** in the Actions column. On the Data Integration page, click **Custom Resource Group** in the left-side navigation pane. On the Custom Resource Groups page, click **Add Resource Group** in the upper-right corner.

> **Note** You can perform this step to create a custom resource group only when you use DataWorks Professional Edition or a more advanced edition.

b. When you add a server, enter information such as the UUID of the ECS instance and the server IP address. If the network type is classic network, enter the hostname. If the network type is virtual private cloud (VPC), enter the UUID of the ECS instance. You can add scheduling resources whose network type is classic network in DataWorks V2.0 only in the China (Shanghai) region. In other regions, you must add scheduling resources whose network type is VPC regardless of the network type of your ECS instances.

For the server IP address, enter the public IP address of the master node because the private IP address may be unreachable. To query the UUID of the ECS instance, log on to the master node and run the dmidecode grep UUID command. You can also use this command to query the UUID of the ECS instance even if your Hadoop cluster is not created by using EMR.

c. After you add the server, you must make sure that the master node and DataWorks are connected. If you add an ECS instance, you must configure a security group for the instance.

- If you use a private IP address, add the private IP address to the security group of the ECS instance. For more information, see Configure a security group for an ECS instance where a self-managed data store resides.
- If you use a public IP address, configure the Internet inbound and outbound rules in the security group of the ECS instance. In this example, all ports are specified in the configured inbound rules to allow traffic from the Internet. In actual scenarios, we recommend that you configure specific security group rules for security purposes.

  Inbound and outbound rules

d. After you complete the preceding steps, install an agent for the custom resource group as prompted. If the status of the ECS instance is **Available**, the custom resource group is created.

View the status of the ECS instance

If the status of the ECS instance is **Unavailable**, log on to the master node and run the `tail -f/home/admin/alisatasknode/logs/heartbeat.log` command to check whether the heartbeat packets between DataWorks and the master node timed out.

View heartbeat packets

2. Add data sources.

After you create a workspace in DataWorks and associate a MaxCompute compute engine instance with the workspace, DataWorks creates the default MaxCompute data source odps_first. In this

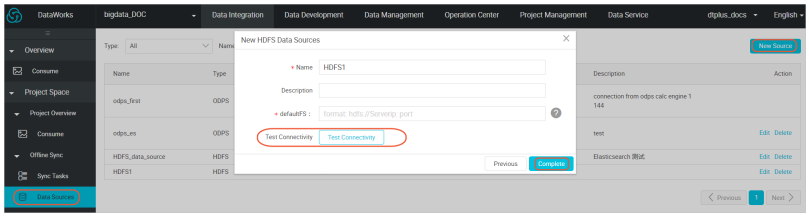
example, the default MaxCompute data source is used. Therefore, you need to add only a Hadoop data source. For more information about how to add a Hadoop data source, see Add an HDFS data source.

i. On the Data Integration page of the DataWorks console, click **Data Source** in the left-side navigation pane.

ii. On the **Data Source** page, click **Add data source** in the upper-right corner.

iii. In the **Add data source** dialog box, click **HDFS** in the Semi-structuredstorage section.

iv. In the Add HDFS data source dialog box, configure the parameters.



Parameter	Description
Data Source Name	The name of the data source. The name can contain letters, digits, and underscores (_) and must start with a letter.
Data Source Description	The description of the data source. The description cannot exceed 80 characters in length.
Environment	The environment in which the data source is used. Valid values: **Development** and **Production**. **Note** This parameter is displayed only when the workspace is in standard mode.
DefaultFS	The address of the NameNode in HDFS. If the EMR Hadoop cluster is in HA mode, the address is `hdfs://IP address of the emr-header-1 node:8020`. If the EMR Hadoop cluster is in non-HA mode, the address is `hdfs://IP address of the emr-header-1 node:9000`. In this example, the emr-header-1 node is connected to DataWorks over the Internet. Therefore, enter the public IP address and allow traffic from the Internet.

v. Click **Test Connectivity**.

vi. If the connectivity test is successful, click **Complete**.

Note If the network type of the EMR Hadoop cluster is VPC, the connectivity test is not supported.

3. Create and configure a data synchronization node.

i.

ii.

iii.

iv. In the **Confirm** message, click **OK** to switch to the code editor.

v. Click the **Apply Template** icon in the top toolbar.

Apply Template icon

vi. In the **Apply Template** dialog box, configure the **Source Connection Type**, **Connection**, **Target Connection Type**, and **Connection** parameters and click **OK**.

Apply Template dialog box

vii. After the template is applied, the basic settings of HDFS Reader are configured.

You can further configure the data source and source table for HDFS Reader based on your business requirements. In this example, the following script is used. For more information, see HDFS Reader.

```
{
  "configuration": {
    "reader": {
      "plugin": "hdfs",
      "parameter": {
        "path": "/user/hive/warehouse/hive_doc_good_sale/",
        "datasource": "HDFS1",
        "column": [
          {
            "index": 0,
            "type": "string"
          },
          {
            "index": 1,
            "type": "string"
          },
          {
            "index": 2,
            "type": "string"
          },
          {
            "index": 3,
            "type": "string"
          },
          {
            "index": 4,
            "type": "long"
          },
          {
            "index": 5,
            "type": "double"
          },
          {
            "index": 6,
            "type": "long"
          }
        ],
        "defaultFS": "hdfs://47.100.XX.XXX:9000",
        "fieldDelimiter": ",",
```

```
        "encoding": "UTF-8",
        "fileType": "text"
      }
    },
    "writer": {
      "plugin": "odps",
      "parameter": {
        "partition": "pt=1",
        "truncate": false,
        "datasource": "odps_first",
        "column": [
          "create_time",
          "category",
          "brand",
          "buyer_id",
          "trans_num",
          "trans_amount",
          "click_cnt"
        ],
        "table": "hive_doc_good_sale"
      }
    },
    "setting": {
      "errorLimit": {
        "record": "1000"
      },
      "speed": {
        "throttle": false,
        "concurrent": 1,
        "mbps": "1",
      }
    }
  },
  "type": "job",
  "version": "1.0"
}
```

In the preceding script, the path parameter specifies the directory where the source data is stored in the EMR Hadoop cluster. You can log on to the master node and run the `hdfs dfs -ls /user/hive/warehouse/hive_doc_good_sale` command to check the directory. For a partitioned table, the data synchronization feature of DataWorks can automatically recurse to the partition where the data is stored.

```
[root@emr-header-1 logs]# hdfs dfs -ls /user/hive/warehouse/hive_doc_good_sale/
Found 1 items
drwxr-x--x   - hive hadoop          0 2018-09-03 17:46 /user/hive/warehouse/hive_doc_good_sale/pt=1
```

viii. After the configuration is complete, click the **Run** icon in the top toolbar. If a message indicating that the synchronization node is successfully run appears, the data is synchronized. If a message indicating that the synchronization node failed to be run appears, check logs for troubleshooting.

Step 3: View the result

1. In the left-side navigation pane of the **DataStudio** page, click **Ad Hoc Query**.
2. In the Ad Hoc Query pane, create an **ODPS SQL** node based on the instructions in the following

figure.

ODPS SQL

3. In the code editor of the created ODPS SQL node, write and execute an SQL statement to view the data that is synchronized to the hive_doc_good_sale table.

   Sample statement:

```
-- Check whether the data is synchronized to MaxCompute.
select * from hive_doc_good_sale where pt=1;
```

> **Note** You can also run the `select * FROM hive_doc_good_sale where pt =1;` command by using the MaxCompute client to query the synchronized data.

If you want to synchronize data from MaxCompute to Hadoop, you can also perform the preceding steps. However, you must exchange the reader and writer in the preceding script. You can use the following script to synchronize data from MaxCompute to Hadoop:

```
{
  "configuration": {
    "reader": {
      "plugin": "odps",
      "parameter": {
      "partition": "pt=1",
      "isCompress": false,
      "datasource": "odps_first",
      "column": [
        "create_time",
        "category",
        "brand",
      "buyer_id",
      "trans_num",
      "trans_amount",
      "click_cnt"
    ],
    "table": "hive_doc_good_sale"
    }
  },
  "writer": {
    "plugin": "hdfs",
    "parameter": {
    "path": "/user/hive/warehouse/hive_doc_good_sale",
    "fileName": "pt=1",
    "datasource": "HDFS_data_source",
    "column": [
      {
        "name": "create_time",
        "type": "string"
      },
      {
        "name": "category",
        "type": "string"
      },
      {
```

```
          "name": "brand",
          "type": "string"
        },
        {
          "name": "buyer_id",
          "type": "string"
        },
        {
          "name": "trans_num",
          "type": "BIGINT"
        },
        {
          "name": "trans_amount",
          "type": "DOUBLE"
        },
        {
          "name": "click_cnt",
          "type": "BIGINT"
        }
      ],
      "defaultFS": "hdfs://47.100.XX.XX:9000",
      "writeMode": "append",
      "fieldDelimiter": ",",
      "encoding": "UTF-8",
      "fileType": "text"
      }
    },
    "setting": {
      "errorLimit": {
        "record": "1000"
    },
    "speed": {
      "throttle": false,
      "concurrent": 1,
      "mbps": "1",
    }
    }
},
"type": "job",
"version": "1.0"
}
```

Note Before you run a synchronization node to synchronize data from MaxCompute to Hadoop, you must configure the Hadoop cluster. For more information, see HDFS Writer. After the synchronization node is run, you can copy the file that is synchronized.

2.4. Best practice to migrate data from Oracle to MaxCompute

This topic describes how to use the data integration feature of DataWorks to migrate data from Oracle to MaxCompute.

Prerequisites

- The DataWorks environment is ready.
  i. Activate MaxCompute and DataWorks.
  ii. Create a workspace. In this example, a workspace in basic mode is used.
  iii. A workflow is created in the DataWorks console. For more information, see Create a workflow.
- The Oracle database is ready.

  In this example, the Oracle database is installed on an Elastic Compute Service (ECS) instance. To enable network communication, you must configure a public IP address for the ECS instance. In addition, you must configure a security group rule for the ECS instance to ensure that the common port 1521 of the Oracle database is accessible. For more information about how to configure a security group rule for an ECS instance, see Modify security group rules.

  In this example, the type of the ECS instance is **ecs.c5.xlarge**. The ECS instance resides in a virtual private cloud (VPC) in the China (Hangzhou) region.

Context

In this example, DataWorks Oracle Reader is used to read test data from the Oracle database. For more information, see Oracle Reader.

Prepare test data in the Oracle database

1. In the Oracle database, create the DTSTEST.GOOD_SALE table that contains the CREATE_TIME, CATEGORY, BRAND, BUYER_ID, TRANAS_NUM, TRANS_AMOUNT, and CLICK_CNT columns.
2. Insert test data to the DTSTEST.GOOD_SALE table. In this example, the following statements are executed to insert test data:

```
insert into good_sale values('28-December-19','Kitchenware','Brand A','hanmeimei','6','
80.6','4');
insert into good_sale values('21-December-19','Fresh food','Brand B','lilei','7','440.6
','5');
insert into good_sale values('29-December-19','Clothing','Brand C','lily','12','351.9',
'9');
commit;
```

3. After data insertion, execute the following statement to view the data in the table:

```
select * from good_sale;
```

Use DataWorks to migrate data from the Oracle database to MaxCompute

1. Go to the **DataStudio** page.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. Select the region where the required workspace resides. Find the required workspace and click **Data Analytics**.

2. On the **DataStudio** page, create a destination table to receive data migrated from the Oracle database.
   i.
   ii.
   iii.
   iv. In the **DDL Statement** dialog box, enter the following statement and click **Generate Table Schema**:

   ```
   CREATE TABLE good_sale
   (
       create_time     string,
       category        string,
       brand           string,
       buyer_id        string,
       trans_num       bigint,
       trans_amount    double,
       click_cnt       bigint
   ) ;
   ```

   When you create the MaxCompute table, make sure that the data types of the MaxCompute table match those of the Oracle table. For more information about the data types supported by Oracle Reader, see Data types.
   v.
3. Create an Oracle connection. For more information, see Add an Oracle data source.
4. Create a batch sync node.
   i.
   ii.
   iii. After you create the batch sync node, set the **Connection** parameter to the created Oracle connection and the **Table** parameter to the Oracle table that you have created. Click **Map Fields with the Same Name**. Use the default values for other parameters.
   iv.
   v.

Verify the result

1.
2.
3. On the configuration tab of the ODPS SQL node, enter the following statement:

   ```
   -- Check whether the data is written to MaxCompute.
   select * from good_sale;
   ```

4.
5.

2.5. Migrate data from Kafka to MaxCompute

This topic describes how to use DataWorks Data Integration to migrate data from a Kafka cluster to MaxCompute.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute and DataWorks.
- A workflow is created in DataWorks. In this example, a DataWorks workspace in basic mode is used. For more information, see Create a workflow.
- A Kafka cluster is created.

  Before data migration, make sure that your Kafka cluster works as expected. In this example, Alibaba Cloud E-MapReduce (EMR) is used to automatically create a Kafka cluster. For more information, see Kafka quick start.

  In this example, the following version of EMR Kafka is used:

  - EMR version: V3.12.1
  - Cluster type: Kafka
  - Software: Ganglia 3.7.2, ZooKeeper 3.4.12, Kafka 2.11-1.0.1, and Kafka Manager 1.3.3.16

  The Kafka cluster is deployed in a virtual private cloud (VPC) in the China (Hangzhou) region. The Elastic Compute Service (ECS) instances in the primary instance group of the Kafka cluster are configured with public and private IP addresses.

Context

Kafka is distributed middleware that is used to publish and subscribe to messages. Kafka is widely used because of its high performance and high throughput. Kafka can process millions of messages per second. Kafka is applicable to streaming data processing, and is used in scenarios such as user behavior tracing and log collection.

A typical Kafka cluster contains several producers, brokers, consumers, and a ZooKeeper cluster. A Kafka cluster uses ZooKeeper to manage configurations and coordinate services in the cluster.

A topic is the most commonly used collection of messages in a Kafka cluster, and is a logical concept for message storage. Topics are not stored on physical disks. Instead, messages in each topic are stored on the disks of each cluster node by partition. Multiple producers can publish messages to a topic, and multiple consumers can subscribe to messages in a topic.

When a message is stored to a partition, the message is allocated an offset. The offset is the unique ID of the message in the partition. The offsets of messages in each partition start from 0.

Step 1: Prepare Kafka data

You must prepare test data in the Kafka cluster. Configure a security group rule for the header node of the EMR cluster to allow requests on TCP ports 22 and 9092. This way, you can log on to the header node of the EMR cluster and MaxCompute and DataWorks can communicate with the header node.

1. Log on to the header node of the EMR cluster.
   i. Log on to the EMR console.

ii. In the top navigation bar, click **Cluster Management**.

iii. On the page that appears, find the cluster for which you want to prepare test data and go to the details page of the cluster.

iv. On the details page of the cluster, click Instances. Find the IP address of the header node of the E-MapReduce cluster and use the IP address to remotely log on to the header node by using Secure Shell (SSH).

2. Create a test topic.

   Run the following command to create a test topic named testkafka:

```
[root@emr-header-1 ~]# kafka-topics.sh --zookeeper emr-header-1:2181/kafka-1.0.1 --part
itions 10 --replication-factor 3 --topic testkafka  --create
Created topic "testkafka".
```

3. Write test data.

   Run the following command to simulate a producer to write data to the testkafka topic. Kafka is used to process streaming data. You can continuously write data to the topic. To ensure that test results are valid, we recommend that you write more than 10 records.

```
[root@emr-header-1 ~]# kafka-console-producer.sh --broker-list emr-header-1:9092 --topi
c testkafka
>123
>abc
>
```

   To simulate a consumer to check whether data is written to Kafka, open another SSH window and run the following command. If the data that is written appears, the data is written to the topic.

```
[root@emr-header-1 ~]# kafka-console-consumer.sh --bootstrap-server emr-header-1:9092 -
-topic testkafka --from-beginning
123
abc
```

Step 2: Create a destination table in DataWorks

Create a destination table in DataWorks to receive data from Kafka.

1.

2.

3.

4. Click **DDL Statement**. In the **DDL Statement** dialog box, enter the following CREATE TABLE statement and click **Generate Table Schema**:

```
CREATE TABLE testkafka
(
 key             string,
 value           string,
 partition1      string,
 timestamp1      string,
 offset          string,
 t123            string,
 event_id        string,
 tag             string
) ;
```

Each column in the statement corresponds to a default column of Kafka Reader that is provided by DataWorks Data Integration.

- __key__: the key of the message.
- __value__: the complete content of the message.
- __partition__: the partition where the message resides.
- __headers__: the header of the message.
- __offset__: the offset of the message.
- __timestamp__: the timestamp of the message.

You can customize a column. For more information, see Kafka Reader.

5.

Step 3: Synchronize the data

1. Create an exclusive resource group for Data Integration.

   The Kafka plug-in cannot run on the default resource group of DataWorks as expected. You must use an exclusive resource group for Data Integration to synchronize data. For more information, see Create and use an exclusive resource group for Data Integration.

2.
3.
4.
5. Configure the script. In this example, enter the following code:

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "kafka",
            "parameter": {
                "server": "47.xxx.xxx.xxx:9092",
                "kafkaConfig": {
                    "group.id": "console-consumer-83505"
                },
                "valueType": "ByteArray",
                "column": [
                    "__key__",
                    "__value__",
```

```
                    "__partition__",
                    "__timestamp__",
                    "__offset__",
                    "'123'",
                    "event_id",
                    "tag.desc"
                ],
                "topic": "testkafka",
                "keyType": "ByteArray",
                "waitTime": "10",
                "beginOffset": "0",
                "endOffset": "3"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "truncate": true,
                "compress": false,
                "datasource": "odps_first",
                "column": [
                    "key",
                    "value",
                    "partition1",
                    "timestamp1",
                    "offset",
                    "t123",
                    "event_id",
                    "tag"
                ],
                "emptyAsNull": false,
                "table": "testkafka"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "speed": {
            "throttle": false
```

```
            "throttle": false,
            "concurrent": 1,
        }
    }
}
```

To view the values of the group.id parameter and the names of consumer groups, run the **kafka-consumer-groups.sh --bootstrap-server emr-header-1:9092 --list** command on the header node.

```
[root@emr-header-1 ~]#  kafka-consumer-groups.sh  --bootstrap-server emr-header-1:9092
--list
Note: This will not show information about old Zookeeper-based consumers.
_emr-client-metrics-handler-group
console-consumer-69493
console-consumer-83505
console-consumer-21030
console-consumer-45322
console-consumer-14773
```

In this example, console-consumer-83505 is used. Run the **kafka-consumer-groups.sh --bootstrap-server emr-header-1:9092 --describe --group console-consumer-83505** command on the header node to obtain the values of the beginOffset and endOffset parameters.

```
[root@emr-header-1 ~]# kafka-consumer-groups.sh --bootstrap-server emr-header-1:9092 --
describe --group console-consumer-83505
Note: This will not show information about old Zookeeper-based consumers.
Consumer group 'console-consumer-83505' has no active members.
TOPIC                          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CO
NSUMER-ID                                       HOST                           CLIENT-I
D
testkafka                      6          0               0               0          -
-                              -
test                           6          3               3               0          -
-                              -
testkafka                      0          0               0               0          -
-                              -
testkafka                      1          1               1               0          -
-                              -
testkafka                      5          0               0               0          -
-                              -
```

6. Configure a resource group for scheduling.
   i. On the node configuration tab, click the **Properties** tab in the right-side navigation pane.
   ii. In the **Resource Group** section, set the **Resource Group** parameter to the exclusive resource group for Data Integration that you have created.

   > **Note** Assume that you want to write Kafka data to MaxCompute at a regular interval, for example, on an hourly basis. You can use the beginDateTime and endDateTime parameters to set the interval for data reading to 1 hour. Then, the data integration node is scheduled to run once per hour. For more information, see Kafka Reader.

7.

8.

What's next

You can create a data development job and run SQL statements to check whether the data has been synchronized from Message Queue for Apache Kafka to the current table. This topic uses the `select * from testkafka` statement as an example. Specific steps are as follows:

1. In the left-side navigation pane, choose **Data Development > Business Flow**.
2. Right-click and choose **Data Development > Create Data Development Node ID > ODPS SQL**.
3. In the **Create Node** dialog box, enter the node name, and then click **Submit**.
4. On the page of the created node, enter `select * from testkafka` and then click the **Run** icon.

2.6. Migrate data from Elasticsearch to MaxCompute

This topic describes how to use the data synchronization feature of DataWorks to migrate data from an Alibaba Cloud Elasticsearch cluster to MaxCompute.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute and DataWorks.
- DataWorks is activated.
- A workflow is created in DataWorks. In this example, a DataWorks workspace in basic mode is used. For more information, see Create a workflow.
- An Alibaba Cloud Elasticsearch cluster is created.

  Before you migrate data, make sure that your Alibaba Cloud Elasticsearch cluster works as expected. For more information about how to create an Alibaba Cloud Elasticsearch cluster, see Quick start.

  An Alibaba Cloud Elasticsearch cluster with the following configurations is used in this example:

  - Region: China (Shanghai)
  - Zone: Zone B
  - Version: Elasticsearch 5.5.3 with Commercial Feature

Context

Elasticsearch is a Lucene-based search server. It provides a distributed multi-tenant search engine that supports full-text search. Elasticsearch is an open source service that complies with the Apache open standards. It is a mainstream enterprise-class search engine.

Alibaba Cloud Elasticsearch includes Elasticsearch 5.5.3 with Commercial Feature, Elasticsearch 6.3.2 with Commercial Feature, and Elasticsearch 6.7.0 with Commercial Feature. It also contains the commercial X-Pack plug-in. You can use Alibaba Cloud Elasticsearch in scenarios such as data analysis and search. Based on open source Elasticsearch, Alibaba Cloud Elasticsearch provides enterprise-class access control, security monitoring and alerting, and automatic reporting.

Procedure

1. Create a source table in Elasticsearch. For more information, see Use DataWorks to synchronize data from MaxCompute to an Alibaba Cloud Elasticsearch cluster.

2. Create a destination table in MaxCompute.
   i.
   ii.
   iii.
   iv.
   v. In the **DDL Statement** dialog box, enter the following CREATE TABLE statement and click **Generate Table Schema**:

```
create table elastic2mc_bankdata
(
age             string,
job             string,
marital         string,
education       string,
default         string,
housing         string,
loan            string,
contact         string,
month           string,
day of week     string
);
```

   vi.
3. Synchronize data.
   i.
   ii.
   iii.
   iv.
   v.
   vi. Configure the script.

   In this example, enter the following code. For more information about the code description, see Elasticsearch Reader.

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "elasticsearch",
            "parameter": {
                "retryCount": 3,
                "column": [
                    "age",
                    "job",
                    "marital",
                    "education",
                    "default",
                    "housing",
                    "loan",
                    "contact",
```

```
                    "month",
                    "day_of_week",
                    "duration",
                    "campaign",
                    "pdays",
                    "previous",
                    "poutcome",
                    "emp_var_rate",
                    "cons_price_idx",
                    "cons_conf_idx",
                    "euribor3m",
                    "nr_employed",
                    "y"
                ],
                "scroll": "1m",
                "index": "es_index",
                "pageSize": 1,
                "sort": {
                    "age": "asc"
},
                "type": "elasticsearch",
                "connTimeOut": 1000,
                "retrySleepTime": 1000,
                "endpoint": "http://es-cn-xxxx.xxxx.xxxx.xxxx.com:9200",
                "password": "xxxx",
                "search": {
                    "match_all": {}
                },
                "readTimeOut": 5000,
                "username": "xxxx"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "truncate": true,
                "compress": false,
                "datasource": "odps_first",
                "column": [
                    "age",
                    "job",
                    "marital",
                    "education",
                    "default",
                    "housing",
                    "loan",
                    "contact",
                    "month",
                    "day_of_week",
                    "duration",
                    "campaign",
```

```
                    "pdays",
                    "previous",
                    "poutcome",
                    "emp_var_rate",
                    "cons_price_idx",
                    "cons_conf_idx",
                    "euribor3m",
                    "nr_employed",
                    "y"
                ],
                "emptyAsNull": false,
                "table": "elastic2mc_bankdata"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
            "throttle": false,
            "concurrent": 1,
            "dmu": 1
        }
    }
}
```

> **Note** On the **Basic Information** page of the created Alibaba Cloud Elasticsearch cluster, you can view the public IP address and port number in the Public Network Access and Public Network Port fields.

vii. Click the icon to run the code.

viii. You can view the running result on the **Runtime Log** tab.

4. View the result.

i.

ii.

iii. On the configuration tab of the ODPS SQL node, enter the following statement:

```
SELECT * FROM elastic2mc_bankdata;
```

iv.

v.

2.7. Migrate JSON-formatted data from MongoDB to MaxCompute

This topic describes how to use the Data Integration service of DataWorks to migrate JSON-formatted fields from MongoDB to MaxCompute.

Prerequisites

- MaxCompute is activated. For more information, see Activate MaxCompute and DataWorks.
- DataWorks is activated.
- A workflow is created in DataWorks. In this example, a DataWorks workspace in basic mode is used. For more information, see Create a workflow.

Prepare test data in MongoDB

1. Prepare an account.

   Create a user in your database to prepare information for creating a connection in DataWorks. In this example, run the following command:

   ```
   db.createUser({user:"bookuser",pwd:"123456",roles:["root"]})
   ```

   The username is bookuser, the password is 123456, and the permission is root.

2. Prepare data.

   Upload the data to the MongoDB database. In this example, an ApsaraDB for MongoDB instance in a virtual private cloud (VPC) is used. You must apply for a public endpoint for the ApsaraDB for MongoDB instance to communicate with the default resource group of DataWorks. The following test data is uploaded:

```
{
    "store": {
        "book": [
            {
                "category": "reference",
                "author": "Nigel Rees",
                "title": "Sayings of the Century",
                "price": 8.95
                },
            {
                "category": "fiction",
                "author": "Evelyn Waugh",
                "title": "Sword of Honour",
                "price": 12.99
                },
            {
                "category": "fiction",
                "author": "J. R. R. Tolkien",
                "title": "The Lord of the Rings",
                "isbn": "0-395-19395-8",
                "price": 22.99
                }
                    ],
        "bicycle": {
            "color": "red",
            "price": 19.95
                }
                    },
        "expensive": 10
            }
```

3. Log on to the MongoDB database in the Data Management (DMS) console. In this example, the name of the database is admin, and the name of the collection is userlog. You can run the following command to view the uploaded data:

```
db.userlog.find().limit(10)
```

Migrate JSON-formatted data from MongoDB to MaxCompute by using DataWorks

1.
2. Create a destination table in DataWorks. This table is used to store the data that is migrated from MongoDB.
   i.
   ii.
   iii.
   iv. In the **DDL Statement** dialog box, enter the following statement and click **Generate Table Schema**:

```
create table mqdata (mqdata string);
```

v. Click **Commit to Production Environment**.

3. Create a MongoDB connection. For more information, see Add a MongoDB data source.
4. Create a batch sync node.
   i.
   ii.
   iii.
   iv.
   v.
   vi. Enter the following script:

```
{
    "type": "job",
    "steps": [
    {
        "stepType": "mongodb",
        "parameter": {
            "datasource": "mongodb_userlog", // The name of the connection.
            "column": [
                {
                "name": "store.bicycle.color", // The path of the JSON-formatted fi
eld. In this example, the color field is extracted.
                "type": "document.String" // For fields other than top-level fields
, the data type of such a field is the type that is finally obtained. If the specif
ied JSON-formatted field is a top-level field, such as the expensive field in this
example, enter string.
                }
              ],
            "collectionName": "userlog" // The name of the collection.
            },
        "name": "Reader",
        "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
            "partition": "",
            "isCompress": false,
            "truncate": true,
            "datasource": "odps_first",
            "column": [
            "mqdata" // The name of the column in the MaxCompute table.
            ],
            "emptyAsNull": false,
            "table": "mqdata"
            },
            "name": "Writer",
            "category": "writer"
            }
            ],
            "version": "2.0",
            "order": {
```

```
            "hops": [
            {
            "from": "Reader",
            "to": "Writer"
            }
            ]
            },
            "setting": {
            "errorLimit": {
            "record": ""
            },
            "speed": {
            "concurrent": 2,
            "throttle": false,
            }
            }
        }
```

vii.

viii.

Verify the result

1.
2.
3. On the configuration tab of the ODPS SQL node, enter the following statement:

```
SELECT * from mqdata;
```

4.
5.

2.8. Migrate data from ApsaraDB RDS to MaxCompute based on dynamic partitioning

This topic describes how to use the data integration and data synchronization features of DataWorks to migrate data from ApsaraDB RDS to MaxCompute based on dynamic partitioning.

Prerequisites

- The DataWorks environment is ready.
  i. MaxCompute is activated. For more information, see Activate MaxCompute and DataWorks.
  ii. DataWorks is activated. To activate DataWorks, go to the DataWorks buy page.
  iii. A workflow is created in the DataWorks console. In this example, a workflow is created in a DataWorks workspace in basic mode. For more information, see Create a workflow.
- Connections to the source and destination data stores are created.
  - A MySQL connection is created as the source connection. For more information, see Add a MySQL

data source.

- A MaxCompute connection is created as the destination connection. For more information, see Add a MaxCompute data source.

Migrate data from ApsaraDB RDS to MaxCompute based on dynamic partitioning

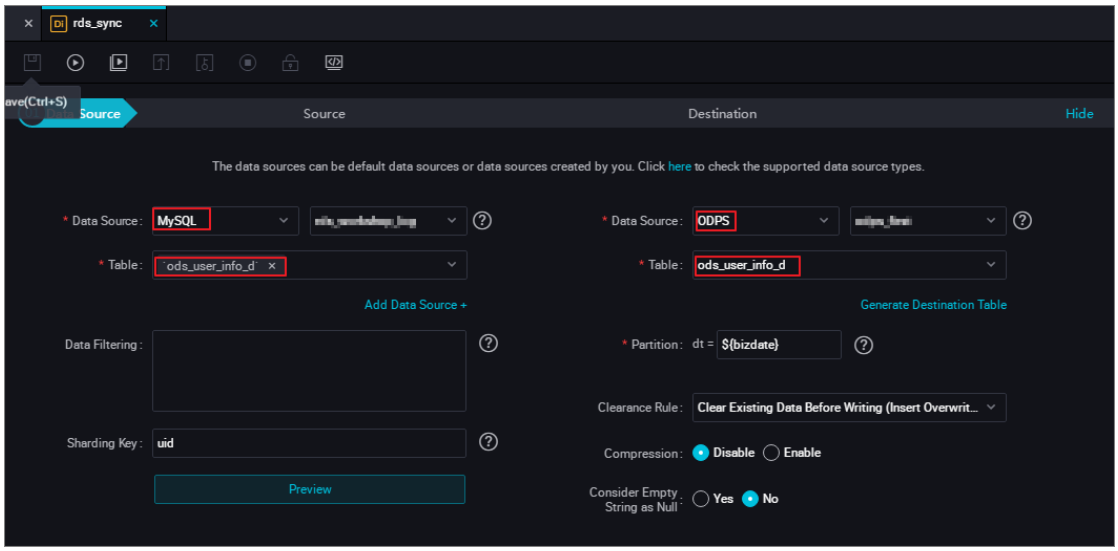
After the preceding preparations are made, configure a sync node to migrate data from ApsaraDB RDS to MaxCompute based on dynamic partitioning every day at a scheduled time. For more information about how to configure a sync node, see Overview.

1.
2. Create a destination table in MaxCompute.
   i.
   ii.
   iii.
   iv.
   v.
   vi. In the **DDL Statement** dialog box, enter the following statement and click **Generate Table Schema**:

   ```
   CREATE TABLE IF NOT EXISTS ods_user_info_d (
   uid STRING COMMENT 'User ID',
   gender STRING COMMENT 'Gender',
   age_range STRING COMMENT 'Age range',
   zodiac STRING COMMENT 'Zodiac sign'
   )
   PARTITIONED BY (
   dt STRING
   );
   ```

   vii.
3. Create a batch sync node.
   i.
   ii.

iii. Configure the source and destination for the batch sync node.



4. Configure the partition parameter.

i. In the right-side navigation pane of the node configuration tab, click the **Properties** tab.

ii. In the **General** section, set the **Arguments** parameter. The default value is `${bizdate}` in the format of yyyymmdd.

> **Note** The value of the **Arguments** parameter in the General section on the Properties tab is the same as that of the **Partition Key Column** parameter in the **Target** section on the node configuration page. When the sync node is scheduled and run, the value of the partition parameter of the destination table is replaced with the date that is one day before the node is run, which is known as the data timestamp. By default, the data generated on the day before the node is run is migrated. To use the date when the node is run as the value of the partition parameter of the destination table, you must customize the partition parameter.

You can specify a date in one of the following formats for the partition parameter:

- N years later: `$[add_months(yyyymmdd,12*N)]`
- N years ago: `$[add_months(yyyymmdd,-12*N)]`
- N months ago: `$[add_months(yyyymmdd,-N)]`
- N weeks later: `$[yyyymmdd+7*N]`
- N months later: `$[add_months(yyyymmdd,N)]`
- N weeks ago: `$[yyyymmdd-7*N]`
- N days later: `$[yyyymmdd+N]`
- N days ago: `$[yyyymmdd-N]`
- N hours later: `$[hh24miss+N/24]`
- N hours ago: `$[hh24miss-N/24]`
- N minutes later: `$[hh24miss+N/24/60]`
- N minutes ago: `$[hh24miss-N/24/60]`

> **Note**
>
> - Keep the value calculation formula in brackets []. For example, `key1=$[yyyy-mm-dd]`.
> - The default unit of the calculation result is day. For example, `$[hh24miss-N/24/60]` refers to the calculation result of `(yyyymmddhh24miss - (N/24/60 × 1 day))`. The format hh24miss is used to align the value.
> - The unit of add_months is month. For example, `$[add_months(yyyymmdd,12 N)-M/24/60]` refers to the calculation result of `(yyyymmddhh24miss - (12 × N × 1 month)) - (M/24/60 × 1 day)`. The format `yyyymmdd` is used to align the value.

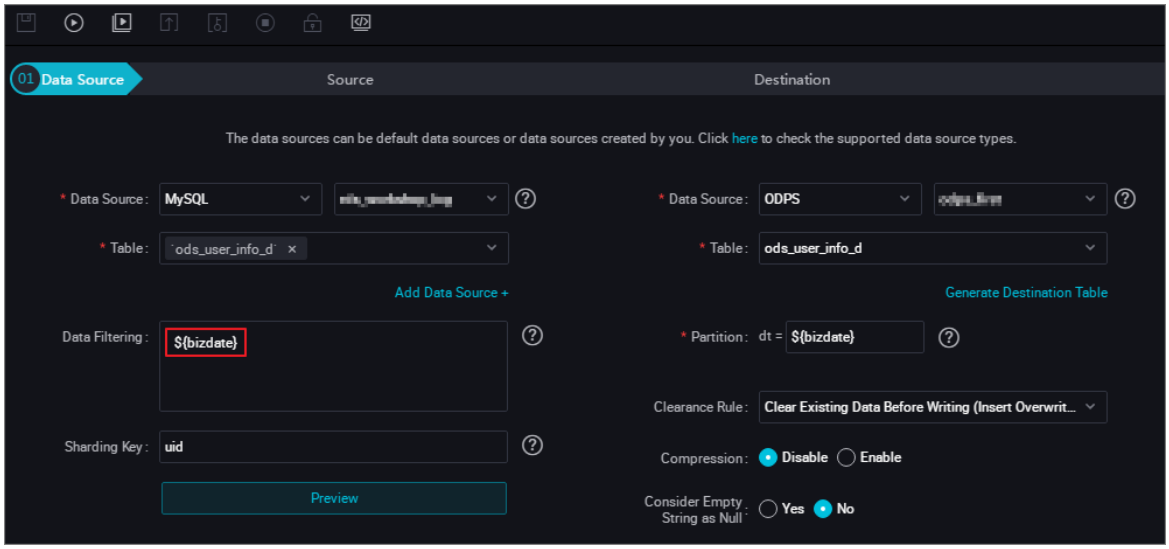
5.
6.

Generate retroactive data

If you have a large amount of historical data in ApsaraDB RDS that is generated before the node is run, all historical data needs to be automatically migrated to MaxCompute and the partitions need to be automatically created. To generate retroactive data for the current sync node, you can use the Patch Data feature of DataWorks.

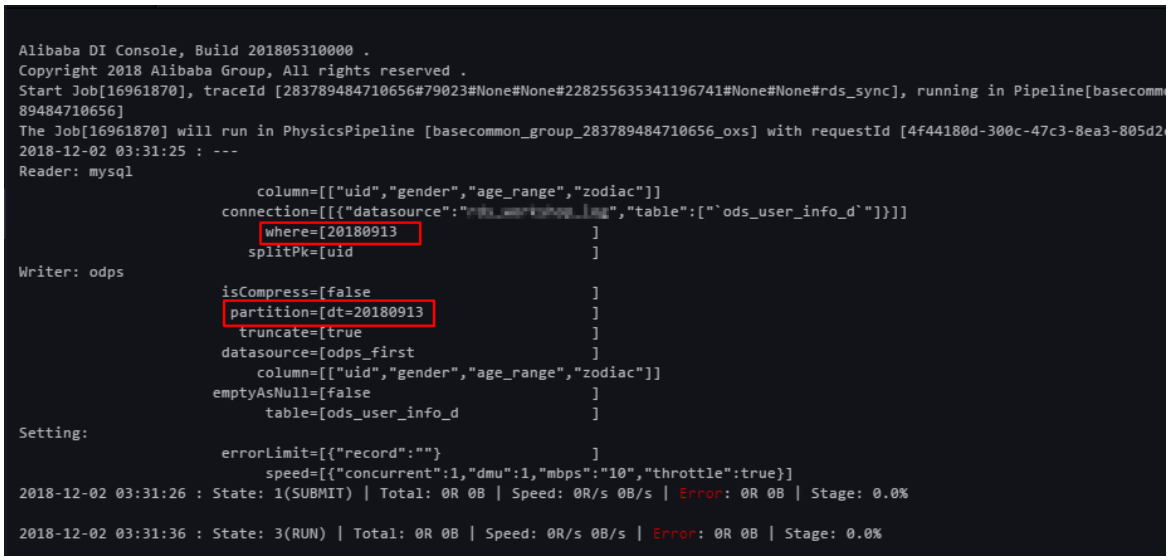
1. Filter historical data in ApsaraDB RDS by date.

   You can set the **Filter** parameter in the **Source** section to filter data in ApsaraDB RDS.



2. Generate retroactive data for the node. For more information, see Perform retroactive data generation and view retroactive data generation instances.
3. View the process of extracting data from ApsaraDB RDS on the Run Log tab.

   The logs indicate that Partition 20180913 is automatically created in MaxCompute.



4. Verify the execution result. Execute the following statement on the MaxCompute client to check whether the data is written to MaxCompute:

```
SELECT count(*) from ods_user_info_d where dt = 20180913;
```

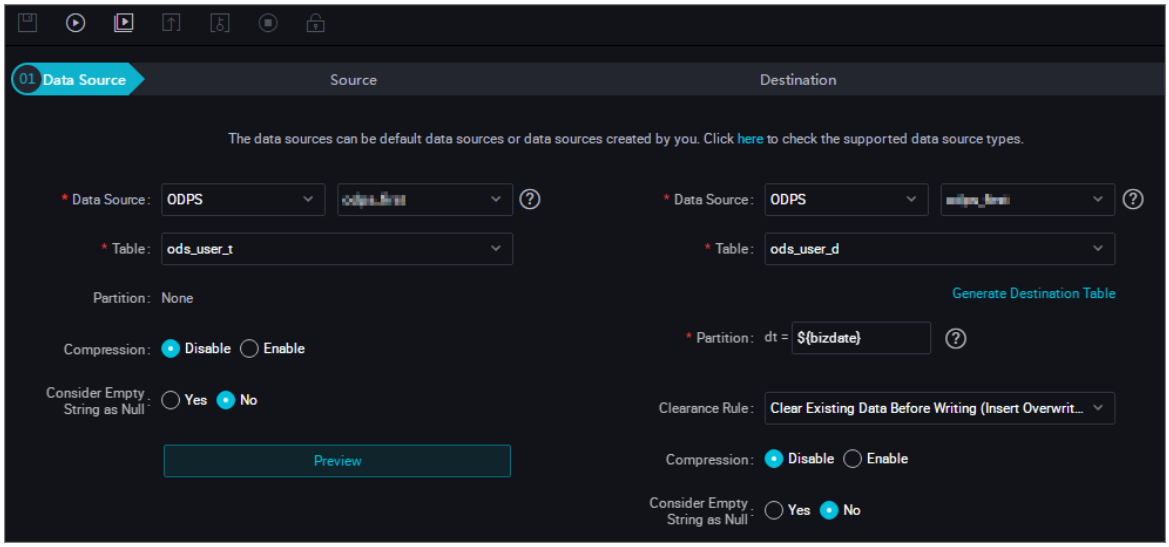
Use hash functions to create partitions based on non-date fields

If you have a large amount of data or full data is migrated to partitions based on a non-date field for the first time, the partitions cannot be automatically created during the migration. In this case, you can map the values in a field in the source table to a corresponding partition in MaxCompute by using a hash function.

1. Create an SQL script node. Execute the following statements to create a temporary table in MaxCompute and migrate data to the table:

```
drop table if exists ods_user_t;
CREATE TABLE ods_user_t (
        dt STRING,
        uid STRING,
        gender STRING,
        age_range STRING,
        zodiac STRING);
-- Create a temporary table named ods_user_t and write data from Table ods_user_info_d
to Table ods_user_t.
insert overwrite table ods_user_t select dt,uid,gender,age_range,zodiac from ods_user_i
nfo_d;
```

2. Create a sync node named mysql_to_odps to migrate full data from ApsaraDB RDS to MaxCompute. Partitioning is not required.

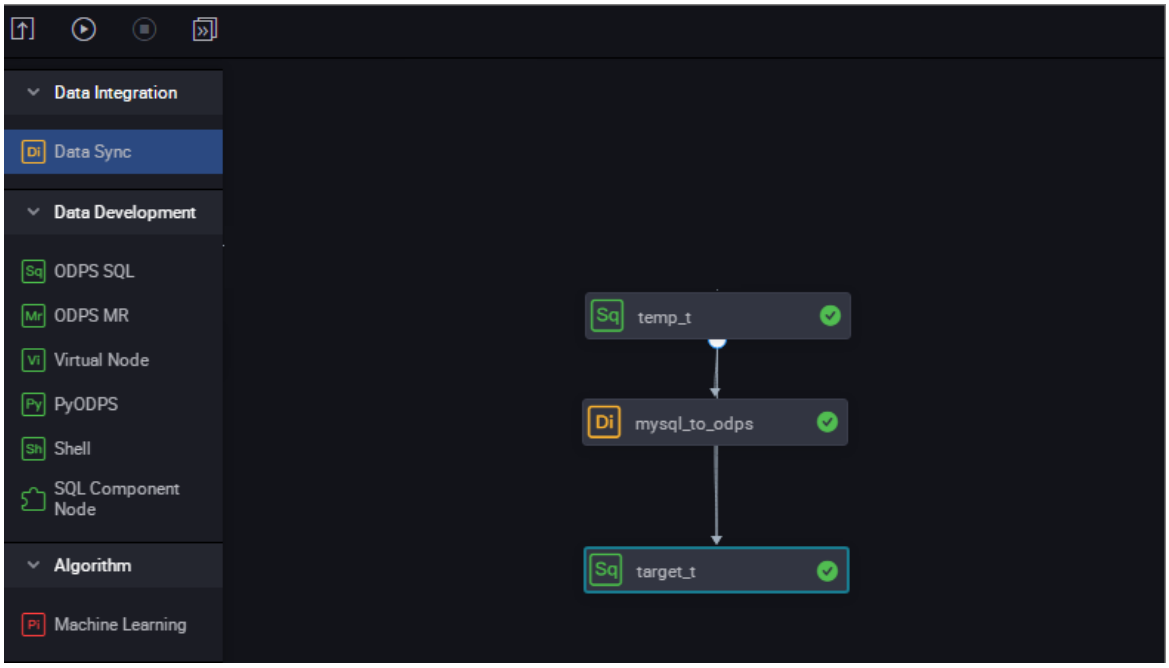


3. Execute the following SQL statements to migrate data from Table ods_user_t to Table ods_user_d based on dynamic partitioning:

```
drop table if exists ods_user_d;
// Create a MaxCompute partitioned table named ods_user_d, which is the destination tab
le.
    CREATE TABLE ods_user_d (
    uid STRING,
        gender STRING,
        age_range STRING,
        zodiac STRING
)
PARTITIONED BY (
    dt STRING
);
// Create dynamic partitions for Table ods_user_d based on the dt field in Table ods_us
er_t. In Table ods_user_d, a partition is automatically created for each unique value i
n the dt field in Table ods_user_t.
// For example, if the value of the dt field is 20181025 in some rows of Table ods_user
_t, Partition dt=20181025 is created in Table ods_user_d.
// The following SQL statement is used to migrate data from Table ods_user_t to Table o
ds_user_d based on dynamic partitioning.
// The dt field is specified in the SELECT clause. This indicates that the partitions a
re automatically created based on this field.
insert overwrite table ods_user_d partition(dt)select dt,uid,gender,age_range,zodiac fr
om ods_user_t;
// After data migration is complete, you may drop the temporary table to release storag
e space.
drop table if exists ods_user_t;
```

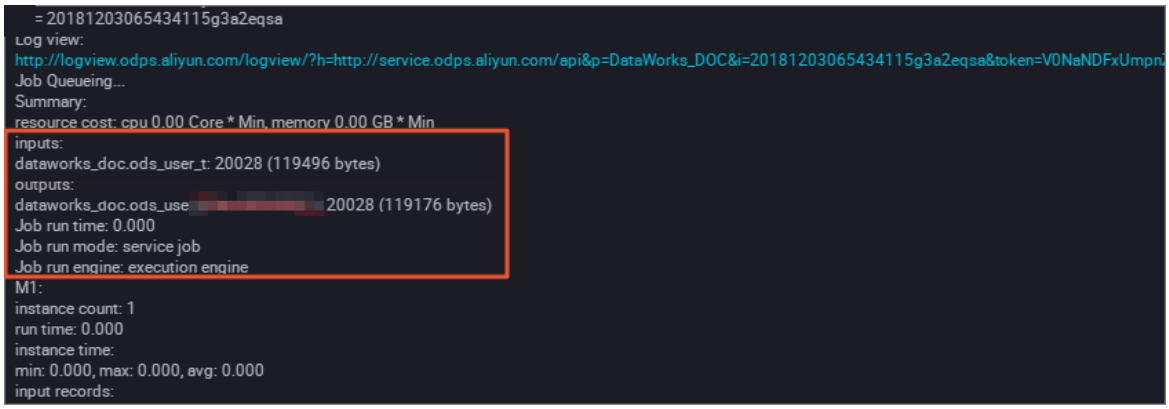
You can use SQL statements to migrate data in MaxCompute. For more information about SQL statements, see Use partitioned tables in MaxCompute.

4. Configure the three nodes to form a workflow to run these nodes sequentially, as shown in the following figure.



5. View the execution process. The last node represents the process of dynamic partitioning, as

shown in the following figure.



6. Verify the execution result. Execute the following statement on the MaxCompute client to check whether the data is written to MaxCompute:

```
SELECT count(*) from ods_user_d where dt = 20180913;
```

2.9. Migrate JSON data from OSS to MaxCompute

This topic describes how to use the data integration feature of DataWorks to migrate JSON data from Object Storage Service (OSS) to MaxCompute and use the GET_JSON_OBJECT function to extract JSON objects.

Prerequisites

- MaxCompute is activated.
- DataWorks is activated.
- A workflow is created in the DataWorks console. In this example, a workflow is created in a DataWorks workspace in basic mode. For more information, see Create a workflow.
- A TXT file that contains JSON data is uploaded to an OSS bucket. In this example, the OSS bucket is in the China (Shanghai) region. The TXT file contains the following JSON data:

```
{
    "store": {
        "book": [
             {
                "category": "reference",
                "author": "Nigel Rees",
                "title": "Sayings of the Century",
                "price": 8.95
             },
             {
                "category": "fiction",
                "author": "Evelyn Waugh",
                "title": "Sword of Honour",
                "price": 12.99
             },
             {
                 "category": "fiction",
                 "author": "J. R. R. Tolkien",
                 "title": "The Lord of the Rings",
                 "isbn": "0-395-19395-8",
                 "price": 22.99
             }
          ],
          "bicycle": {
              "color": "red",
              "price": 19.95
          }
    },
    "expensive": 10
}
```

Migrate JSON data from OSS to MaxCompute

1. Add an OSS connection. For more information, see Add an OSS data source.
2. Create a table in DataWorks to store the JSON data to be migrated from OSS.
   i.
   ii.
   iii.
   iv. In the **DDL Statement** dialog box, enter the following statement and click **Generate Table Schema**:

   ```
   create table mqdata (mq_data string);
   ```

   v.
3. Create a batch synchronization node.
   i.
   ii.
   iii.
   iv.

v.

vi. Modify JSON code and click the icon.

Sample code:

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "oss",
            "parameter": {
                "fieldDelimiterOrigin": "^",
                "nullFormat": "",
                "compress": "",
                "datasource": "OSS_userlog",
                "column": [
                    {
                        "name": 0,
                        "type": "string",
                        "index": 0
                    }
                ],
                "skipHeader": "false",
                "encoding": "UTF-8",
                "fieldDelimiter": "^",
                "fileFormat": "binary",
                "object": [
                    "applog.txt"
                ]
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "isCompress": false,
                "truncate": true,
                "datasource": "odps_first",
                "column": [
                    "mqdata"
                ],
                "emptyAsNull": false,
                "table": "mqdata"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
```

```
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": ""
        },
        "speed": {
            "concurrent": 2,
            "throttle": false,
        }
    }
}
```

Use the GET_JSON_OBJECT function to extract JSON objects

1. Create an ODPS SQL node.
   i.
   ii.
   iii. On the configuration tab of the ODPS SQL node, enter the following statements:

```
--Query data in the mqdata table.
SELECT * from mqdata;
--Obtain the value of the expensive field.
SELECT GET_JSON_OBJECT(mqdata.MQdata,'$.expensive') FROM mqdata;
```

   iv.
   v.

2.10. Migrate data from MaxCompute to Tablestore

This topic describes how to migrate data from MaxCompute to Tablestore.

Prerequisites

Procedure

1. Create a table in the DataWorks console.
   i.
   ii.
   iii.
   iv.
   v.
   vi.

vii. In the **DDL Statement** dialog box, enter the following statement and click **Generate Table Schema**:

```
create table Transs
(name    string,
id    bigint,
gender    string);
```

viii.

2. Import data to table Transs.

i.

ii.

iii. In the dialog box that appears, set **Select Data Import Method** to **Upload Local File** and click **Browse** next to **Select File**. Select the local file that you want to import. Then, specify other parameters.

Example:

```
qwe,145,F
asd,256,F
xzc,345,M
rgth,234,F
ert,456,F
dfg,12,M
tyj,4,M
bfg,245,M
nrtjeryj,15,F
rwh,2344,M
trh,387,F
srjeyj,67,M
saerh,567,M
```

iv.

v.

vi.

3. Create a table in the Tablestore console.

i. Log on to the Tablestore console and create an instance. For more information, see Create instances.

ii. Create a table named Trans. For more information, see Create tables.

4. Add data sources in the DataWorks console.

i.

ii.

iii.

iv.

v. In the upper-right corner, click **New data source**. In the dialog box that appears, click **MaxCompute(ODPS)**.

vi. In the **Add MaxCompute(ODPS) data source** dialog box, specify the required parameters and click **Complete**. For more information, see Add a MaxCompute data source.

vii. Add Tablestore as a data source. For more information, see Add a Tablestore data source.

5. Configure MaxCompute as the reader and Tablestore as the writer.

i.

ii.

iii.

iv.

v.

vi. Modify JSON code and click the icon.

Sample code:

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "odps",
            "parameter": {
                "partition": [],
                "datasource": "odps_first",
                "column": [
                    "name",
                    "id",
                    "gender"
                ],
                "table": "Transs"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "ots",
            "parameter": {
                "datasource": "Transs",
                "column": [
                    {
                        "name": "Gender",
                        "type": "STRING"
                    }
                ],
                "writeMode": "UpdateRow",
                "table": "Trans",
                "primaryKey": [
                    {
                        "name": "Name",
                        "type": "STRING"
                    },
                    {
                        "name": "ID",
                        "type": "INT"
                    }
                ]
```

```
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
            "throttle": false,
            "concurrent": 1,
            "dmu": 1
        }
    }
}
```

6. View the data of the newly created table in the Tablestore console.
   i. Log on to the Tablestore console.
   ii. In the left-side navigation pane, click **All Instances**.
   iii. On the page that appears, find the target instance and click the instance name to go to the **Instance Management** page. In the **Tables** section, click the name of the table whose data you want to view.
   iv. On the page that appears, click the **Data Editor** tab to view the data.

2.11. Migrate data from MaxCompute to OSS

This topic describes how to use the data synchronization feature of DataWorks to migrate data from MaxCompute to Object Storage Service (OSS).

Prerequisites

Procedure

1. Create a table in the DataWorks console.
   i.
   ii.
   iii.
   iv.

v.

vi.

vii. In the **DDL Statement** dialog box, enter the following statement and click **Generate Table Schema**:

```
create table Transs
(name    string,
id    string,
gender    string);
```

viii.

2. Import data to table Transs.

i.

ii.

iii. In the dialog box that appears, set **Select Data Import Method** to **Upload Local File** and click **Browse** next to **Select File**. Select the local file that you want to import. Then, specify other parameters.

Example:

```
qwe,145,F
asd,256,F
xzc,345,M
rgth,234,F
ert,456,F
dfg,12,M
tyj,4,M
bfg,245,M
nrtjeryj,15,F
rwh,2344,M
trh,387,F
srjeyj,67,M
saerh,567,M
```

iv.

v.

vi.

3. Create a table in the OSS console.

i. Log on to the OSS console and create a bucket. For more information, see Create buckets.

ii. Upload the *qwee.csv* file to OSS. For more information, see Upload objects.

> **Note** Make sure that fields in the *qwee.csv* file are exactly the same as those in the Transs table.

4. Add data sources in the DataWorks console.

i.

ii.

iii.

iv. In the left-side navigation pane of the page that appears, click **Connection**. The **Data Source** page appears.

v. In the upper-right corner, click **New data source**. In the dialog box that appears, click **MaxCompute(ODPS)**.

vi. In the **Add MaxCompute(ODPS) data source** dialog box, specify the required parameters and click **Complete**. For more information, see Add a MaxCompute data source.

vii. Add OSS as a data source. For more information, see Add an OSS data source.

5. Configure MaxCompute as the reader and OSS as the writer.

i.

ii.

iii.

iv.

v.

vi. Modify JSON code and click the icon.

Sample code:

```
{
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    },
    "setting":{
        "errorLimit":{
            "record":"0"
        },
        "speed":{
            "concurrent":1,
            "dmu":1,
            "throttle":false
        }
    },
    "steps":[
        {
            "category":"reader",
            "name":"Reader",
            "parameter":{
                "column":[
                    "name",
                    "id",
                    "gender"
                ],
                "datasource":"odps_first",
                "partition":[],
                "table":"Transs"
            },
```

```
            "stepType":"odps"
        },
        {
            "category":"writer",
            "name":"Writer",
            "parameter":{
                "datasource":"Trans",
                "dateFormat":"yyyy-MM-dd HH:mm:ss",
                "encoding":"UTF-8",
                "fieldDelimiter":",",
                "fileFormat":"csv",
                "nullFormat":"null",
                "object":"qweee.csv",
                "writeMode":"truncate"
            },
            "stepType":"oss"
        }
    ],
    "type":"job",
    "version":"2.0"
}
```

6. View the data of the newly created table in the OSS console. For more information, see Download objects.

2.12. Migrate data from a user-created MySQL database on an ECS instance to MaxCompute

This topic describes how to use an exclusive resource group for Data Integration to migrate data from a user-created MySQL database on an Elastic Compute Service (ECS) instance to MaxCompute.

Prerequisites

- An ECS instance is purchased and bound to a virtual private cloud (VPC) but not the classic network. A MySQL database that stores test data is deployed on the ECS instance. An account used to connect to the database is created. In this example, use the following statements to create a table in the MySQL database and insert test data to the table:

```
CREATE TABLE IF NOT EXISTS good_sale(
   create_time timestamp,
   category varchar(20),
   brand varchar(20),
   buyer_id varchar(20),
   trans_num varchar(20),
   trans_amount DOUBLE,
   click_cnt varchar(20)
   );
insert into good_sale values('2018-08-21','coat','brandA','lilei',3,500.6,7),
('2018-08-22','food','brandB','lilei',1,303,8),
('2018-08-22','coat','brandC','hanmeimei',2,510,2),
('2018-08-22','bath','brandA','hanmeimei',1,442.5,1),
('2018-08-22','food','brandD','hanmeimei',2,234,3),
('2018-08-23','coat','brandB','jimmy',9,2000,7),
('2018-08-23','food','brandA','jimmy',5,45.1,5),
('2018-08-23','coat','brandE','jimmy',5,100.2,4),
('2018-08-24','food','brandG','peiqi',10,5560,7),
('2018-08-24','bath','brandF','peiqi',1,445.6,2),
('2018-08-24','coat','brandA','ray',3,777,3),
('2018-08-24','bath','brandG','ray',3,122,3),
('2018-08-24','coat','brandC','ray',1,62,7) ;
```

- The private IP address, VPC, and vSwitch of your ECS instance are noted.
- A security group rule is added for the ECS instance to allow access requests on the port used by the MySQL database. By default, the MySQL database uses port 3306. For more information, see Add a security group rule. The name of the security group is noted.
- A DataWorks workspace is created. In this example, create a DataWorks workspace that is in basic mode and uses a MaxCompute compute engine. Make sure that the created DataWorks workspace belongs to the same region as the ECS instance. For more information about how to create a workspace, see Create a workspace.
- An exclusive resource group for Data Integration is purchased and bound to the VPC where the ECS instance resides. The exclusive resource group and the ECS instance are in the same zone. For more information, see Create and use an exclusive resource group for Data Integration. After the exclusive resource group is bound to the VPC, you can view information about the exclusive resource group on the **Resource Groups** page.
- Check whether the VPC, vSwitch, and security group of the exclusive resource group are the same as those of the ECS instance.

Context

An exclusive resource group can transmit your data in a fast and stable manner. Make sure that the exclusive resource group for Data Integration belongs to the same zone in the same region as the data store that needs to be accessed. Make sure that the exclusive resource group for Data Integration belongs to the same region as the DataWorks workspace. In this example, the data store that needs to be accessed is a user-created MySQL database on an ECS instance.

Procedure

1. Create a connection to the MySQL database in the DataWorks console.
   i. Log on to the DataWorks console by using your Alibaba Cloud account.

ii. On the **Workspaces** page, find the required workspace and click **Data Integration**.

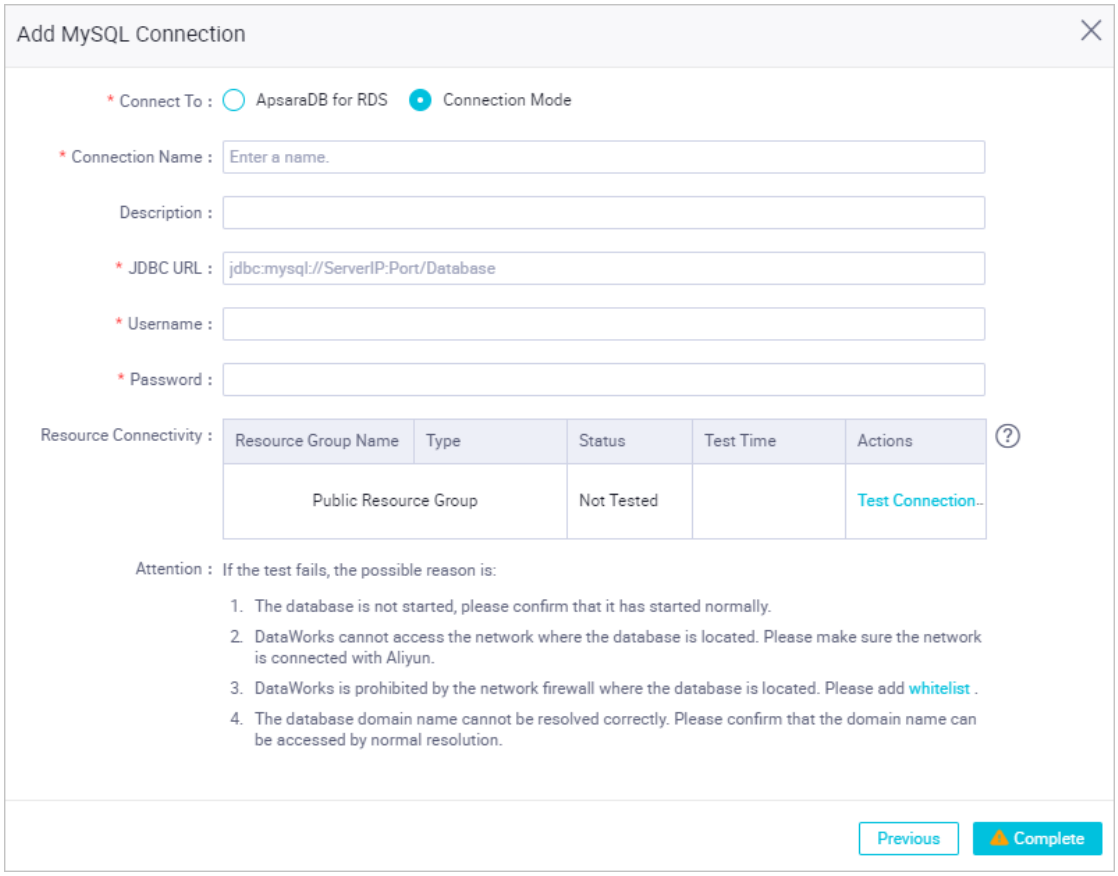
iii. In the left-side navigation pane, click **Connection**.

iv. On the **Data Source** page, click **New data source** in the upper-right corner.

v. In the **Add data source** dialog box, select **MySQL**.

vi. In the **Add MySQL data source** dialog box, set the parameters. For more information, see Add a MySQL data source.

For example, set the Data source type parameter to **Connection string mode**. Use the private IP address of the ECS instance and the default port number 3306 of the MySQL database when you specify the Java Database Connectivity (JDBC) URL.



> **Note** DataWorks cannot test the connectivity of a user-created MySQL database in a VPC. Therefore, it is normal that a connectivity test fails.

vii. Find the required resource group and click **Test connectivity**.

During data synchronization, a sync node uses only one resource group. You must test the connectivity of all the resource groups for Data Integration on which your sync nodes will be run and make sure that the resource groups can connect to the data store. This ensures that your sync nodes can be run as expected. For more information, see Select a network connectivity solution.

viii. After the connection passes the connectivity test, click **Complete**.

2. Create a MaxCompute table.

You must create a table in DataWorks to receive test data from the MySQL database.

i. Click the icon in the upper-left corner and choose **All Products > DataStudio**.

ii. Create a workflow. For more information, see Create a workflow.

iii. Right-click the created workflow and choose **Create > MaxCompute > Table**.

iv. Enter a name for your MaxCompute table. In this example, set the Table Name parameter to good_sale, which is the same as the name of the table in the MySQL database. Click **DDL Statement**, enter the table creation statement, and then click **Generate Table Schema**.

In this example, enter the following table creation statement. Pay attention to data type conversion.

```
CREATE TABLE IF NOT EXISTS good_sale(
   create_time string,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   );
```

v. Set the **Display Name** parameter and click **Commit to Production Environment**. The MaxCompute table named good_sale is created.

3. Configure a data integration node.

i. Right-click the workflow you just created and choose **Create > Data Integration > Batch Synchronization** to create a data integration node.

ii. Set the Connection parameter under Source to the created MySQL connection and the Connection parameter under Target to odps_first. Click the **Switch to Code Editor** icon to switch to the code editor.

If you cannot set the **Table** parameter under Source or an error is returned when you attempt to switch to the code editor, ignore the issue.

iii. Click the **Resource Group configuration** tab in the right-side navigation pane and select an exclusive resource group that you have purchased.

If you do not select the exclusive resource group as the resource group for Data Integration of your node, the node may fail to be run.

iv. Enter the following code for the data integration node:

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "mysql",
            "parameter": {
                "column": [// The columns in the source table.
                    "create_time",
                    "category",
                    "brand",
                    "buyer_id",
                    "trans_num",
                    "trans_amount",
                    "click_cnt"
```

```
                ],
                "connection": [
                    {
                        "datasource": "shuai",// The source connection.
                        "table": [
                            "good_sale"// The name of the table in the source datab
ase. The name must be enclosed in brackets [].
                        ]
                    }
                ],
                "where": "",
                "splitPk": "",
                "encoding": "UTF-8"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "truncate": true,
                "datasource": "odps_first",// The destination connection.
                "column": [// The columns in the destination table.
                    "create_time",
                    "category",
                    "brand",
                    "buyer_id",
                    "trans_num",
                    "trans_amount",
                    "click_cnt"
                ],
                "emptyAsNull": false,
                "table": "good_sale"// The name of the destination table.
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "version": "2.0",
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    },
    "setting": {
        "errorLimit": {
            "record": "0"
        },
        "speed": {
            "throttle": false,
```

```
            "concurrent": 2
        }
    }
}
```

v. Click the Run icon. You can view the **Runtime Log** tab in the lower part of the page to check whether the test data is synchronized to MaxCompute.

Result

To query data in the MaxCompute table, create an **ODPS SQL** node.Enter the statement `select * from good_sale ;` , and click the **Run** icon. If the test data appears, it is synchronized to the MaxCompute table.

2.13. Migrate data from Amazon Redshift to MaxCompute

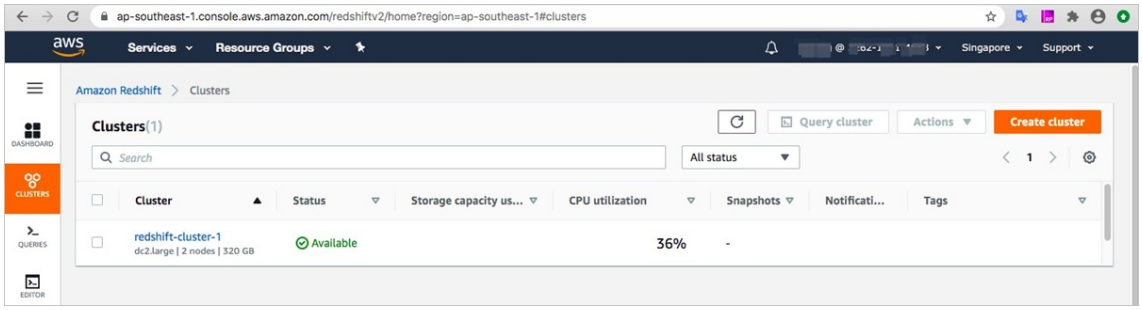
This topic describes how to migrate data from Amazon Redshift to MaxCompute over the Internet.

Prerequisites

- Create an Amazon Redshift cluster and prepare data for migration.

  For more information about how to create an Amazon Redshift cluster, see Amazon Redshift Cluster Management Guide.

  i. Create an Amazon Redshift cluster. If you already have an Amazon Redshift cluster, skip this step.

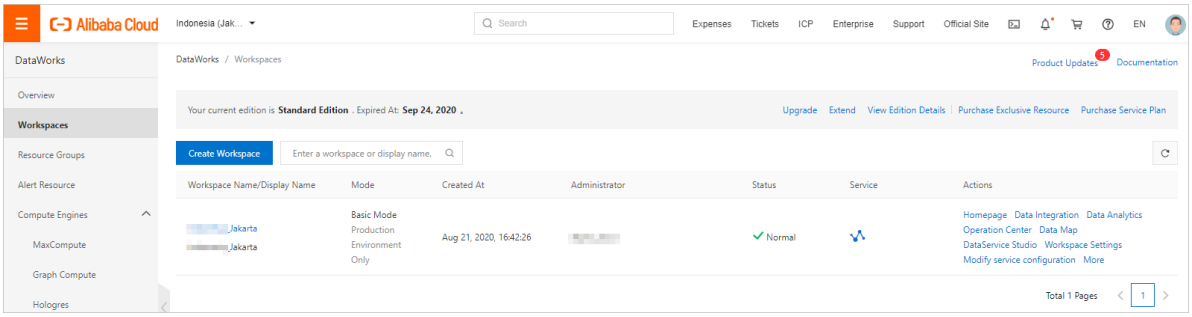
  ii. Prepare the data that you want to migrate in the Amazon Redshift cluster.

  In this example, a TPC-H dataset is available in public schema. The dataset uses the MaxCompute V2.0 data types and the Decimal 2.0 data type.

- Prepare a MaxCompute project.

  For more information, see Prepare.

In this example, a MaxCompute project is created as the migration destination in the **Singapore (Singapore)** region. The project is created in MaxCompute V2.0 because the TPC-H dataset uses the MaxCompute V2.0 data types and the Decimal V2.0 data type.

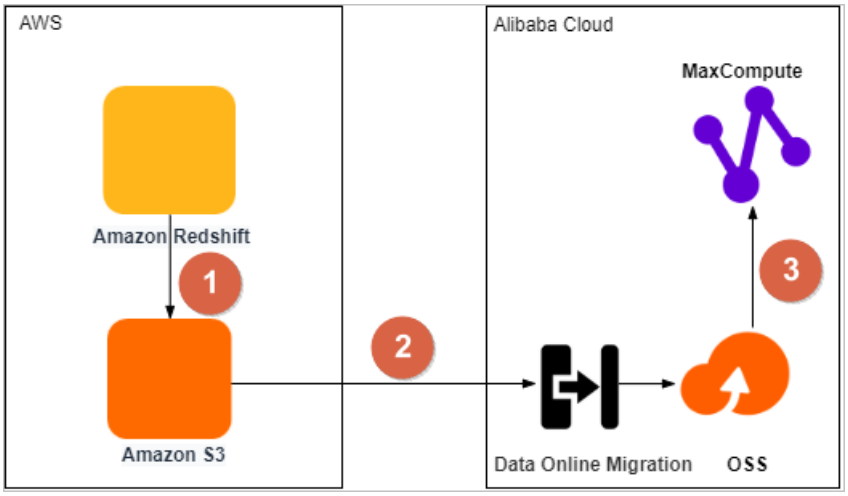


- Activate Alibaba Cloud Object Storage Service (OSS).

  For more information, see 开通OSS服务.

Context

The following figure shows the process to migrate data from Amazon Redshift to MaxCompute.



No.	Description
①	Unload the data from Amazon Redshift to a data lake on Amazon Simple Storage Service (S3).
②	Migrate the data from Amazon S3 to an OSS bucket by using the **Data Online Migration** service of OSS.
③	Migrate the data from the OSS bucket to a MaxCompute project in the same region, and then verify the integrity and accuracy of the migrated data.

Step 1: Unload data from Amazon Redshift to Amazon S3

Amazon Redshift supports authentication based on Identity and Access Management (IAM) roles and temporary security credentials (AccessKey pairs). You can use the UNLOAD command of Amazon Redshift to unload data to Amazon S3 based on these two authentication methods. For more information, see Unloading data.

The syntax of the UNLOAD command varies based on the authentication method.

- UNLOAD command based on an IAM role

```
-- Run the UNLOAD command to unload data from the customer table to Amazon S3.
UNLOAD ('SELECT * FROM customer')
TO 's3://bucket_name/unload_from_redshift/customer/customer_' -- The Amazon S3 bucket.
IAM_ROLE 'arn:aws:iam::****:role/MyRedshiftRole'; -- The Alibaba Cloud Resource Name (ARN
) of the IAM role.
```

- UNLOAD command based on an AccessKey pair

```
-- Run the UNLOAD command to unload data in the customer table to Amazon S3.
UNLOAD ('SELECT * FROM customer')
TO 's3://bucket_name/unload_from_redshift/customer/customer_' -- The Amazon S3 bucket.
Access_Key_id '<access-key-id>' -- The AccessKey ID of the IAM user.
Secret_Access_Key '<secret-access-key>' -- The AccessKey secret of the IAM user.
Session_Token '<temporary-token>'; -- The temporary access token of the IAM user.
```

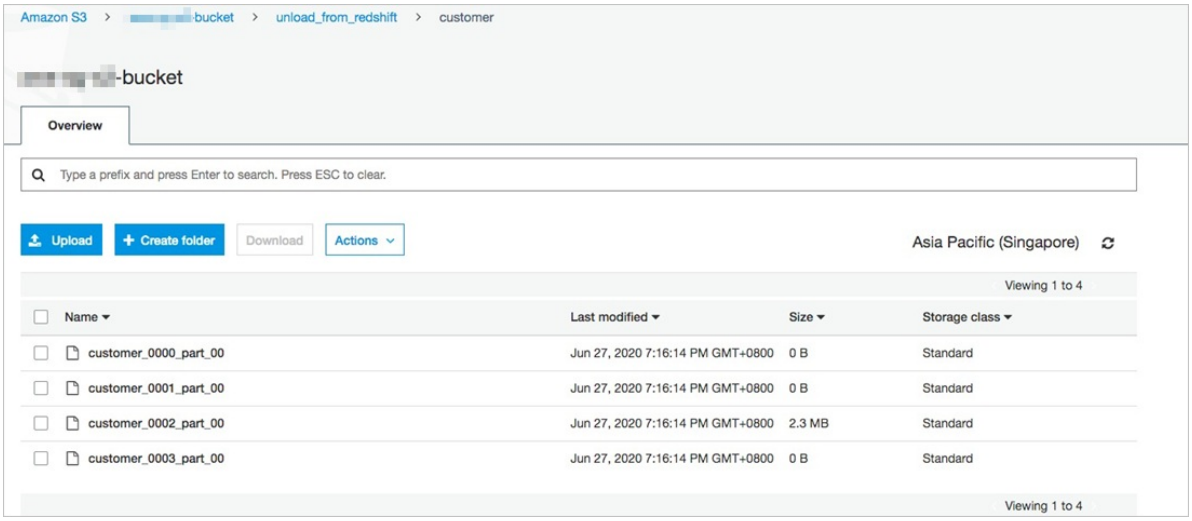
The UNLOAD command allows you to unload data in one of the following formats:

- Default format

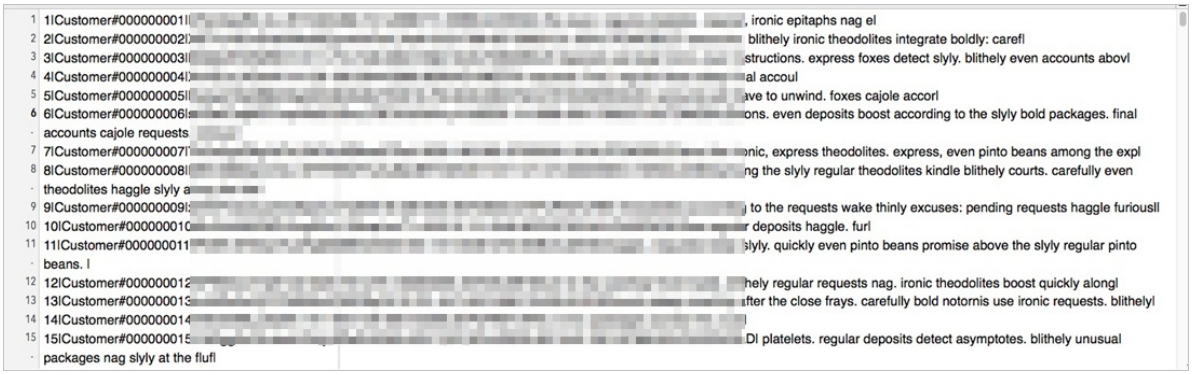
  The following sample command shows how to unload data in the default format:

```
UNLOAD ('SELECT * FROM customer')
TO 's3://bucket_name/unload_from_redshift/customer/customer_'
IAM_ROLE 'arn:aws:iam::****:role/redshift_s3_role';
```

  After the command is run, data is unloaded to text files in which values are separated by vertical bars (|). You can log on to the Amazon S3 console and view the unloaded text files in the specified bucket.



  The following figure shows the unloaded text files in the default format.

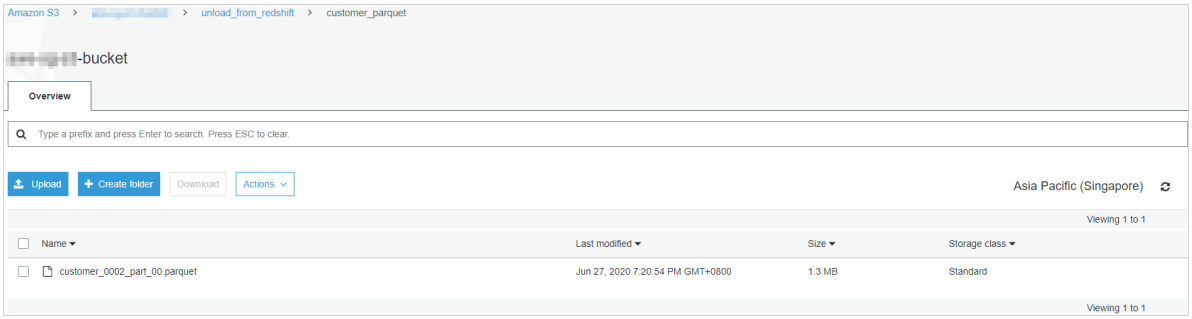


- Apache Parquet format

  Data unloaded in the Apache Parquet format can be directly read by other engines. The following sample command shows how to unload data in the Apache Parquet format:

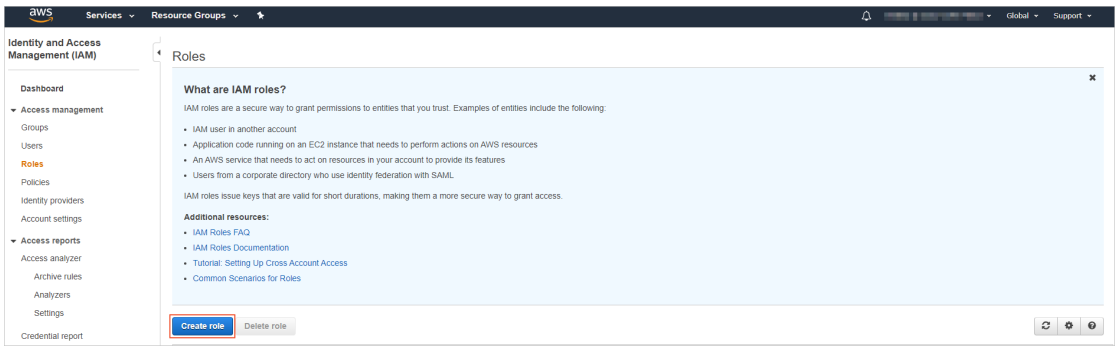
```
UNLOAD ('SELECT * FROM customer')
TO 's3://bucket_name/unload_from_redshift/customer_parquet/customer_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
```

  After the command is run, you can view the unloaded Parquet files in the specified bucket. Parquet files are smaller than text files and have a higher data compression ratio.

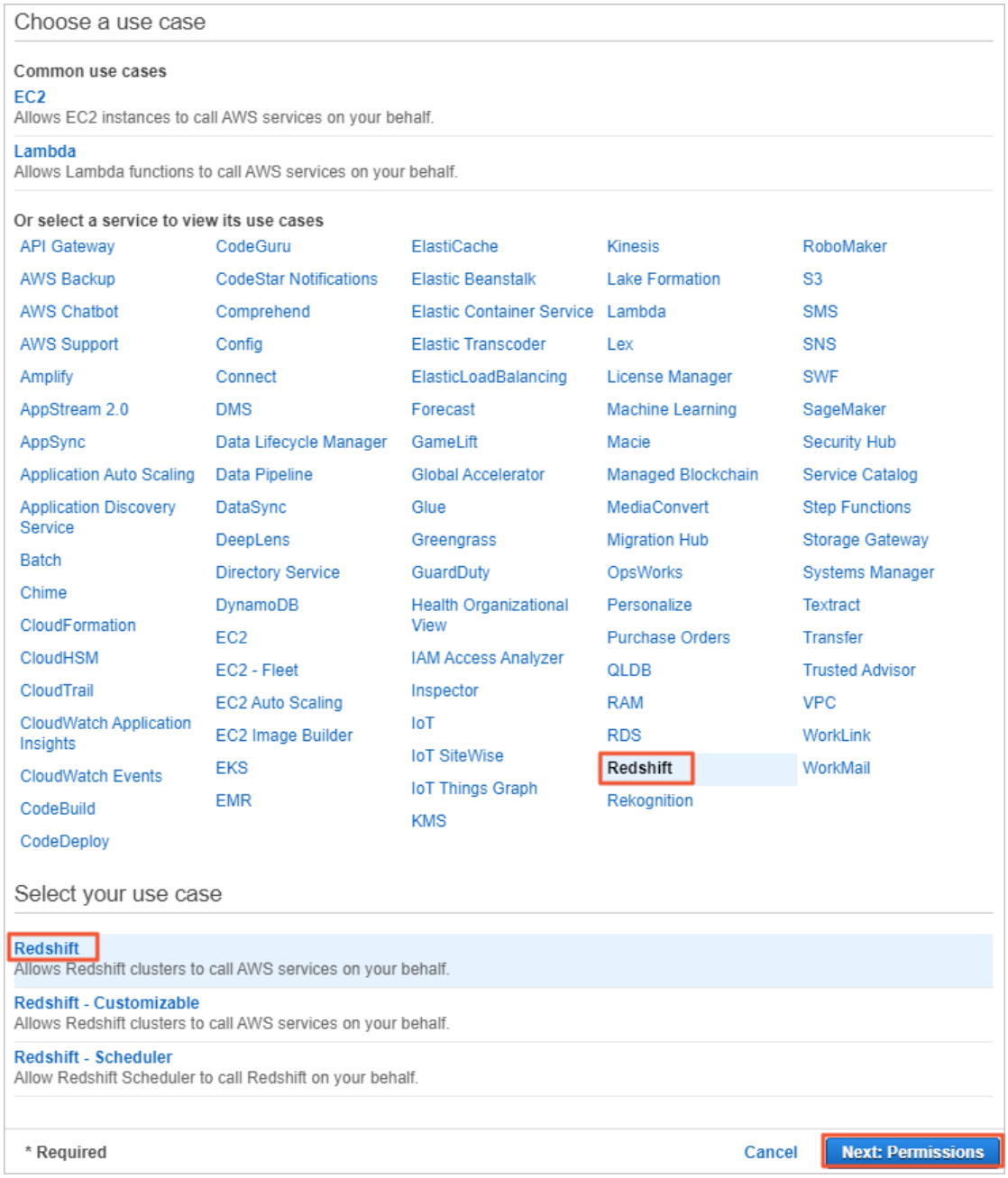


This section describes how to authenticate requests based on IAM roles and unload data in the Apache Parquet format.

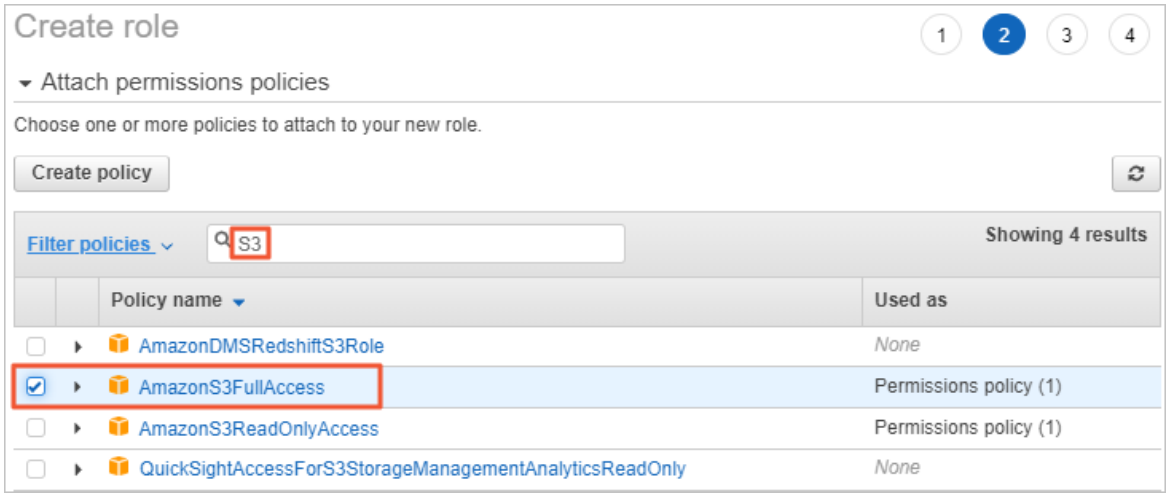
1. Create an IAM role for Amazon Redshift.
   i. Log on to the IAM console. In the left-side navigation pane, choose Access Management > Roles. On the Roles page, click **Create role**.



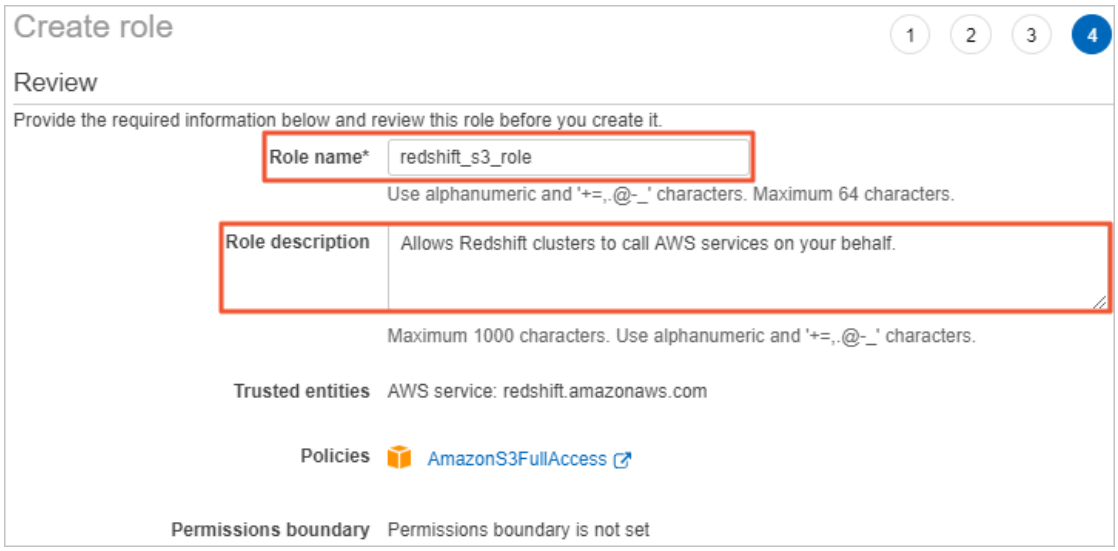
ii. In the **Common use cases** section of the **Create role** page, click **Redshift**. In the **Select your use case** section, click **Redshift - Customizable**, and then click **Next: Permissions**.



2. Add an IAM policy that grants the read and write permissions on Amazon S3. In the **Attach permissions policies** section of the **Create role** page, enter **S3**, select **AmazonS3FullAccess**, and then click **Next: Tags**.

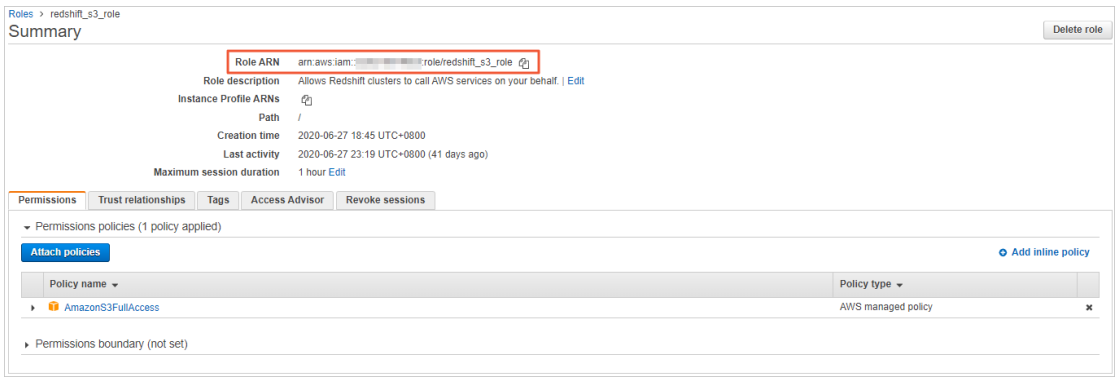


3. Assign a name to the IAM role and complete the IAM role creation.
   i. Click **Next: Review**. In the **Review** section of the **Create role** page, specify **Role name** and **Role description**, and click **Create role**. The IAM role is then created.



   ii. Go to the IAM console, and enter **redshift_s3_role** in the search box to search for the role. Then, click the role name **redshift_s3_role**, and copy the value of **Role ARN**.

   When you run the UNLOAD command to unload data, you must provide the **Role ARN** value to access Amazon S3.



4. Associate the created IAM role with the Amazon Redshift cluster to authorize the cluster to access Amazon S3.

i. Log on to the Amazon Redshift console. In the upper-right corner, select **Asia Pacific (Singapore)** from the drop-down list.

ii. In the left-side navigation pane, click **CLUSTERS**, find the created Amazon Redshift cluster, click **Actions**, and then click **Manage IAM roles**.

iii. On the **Manage IAM roles** page, click the icon next to the search box, and select **redshift_s3_role**. Click **Add IAM role** > **Done** to associate the **redshift_s3_role** role with the Amazon Redshift cluster.

5. Unload data from Amazon Redshift to Amazon S3.

i. Go to the Amazon Redshift console.

ii. In the left-side navigation pane, click **EDITOR**. Run the UNLOAD command to upload data from Amazon Redshift to each destination bucket on Amazon S3 in the Apache Parquet format.

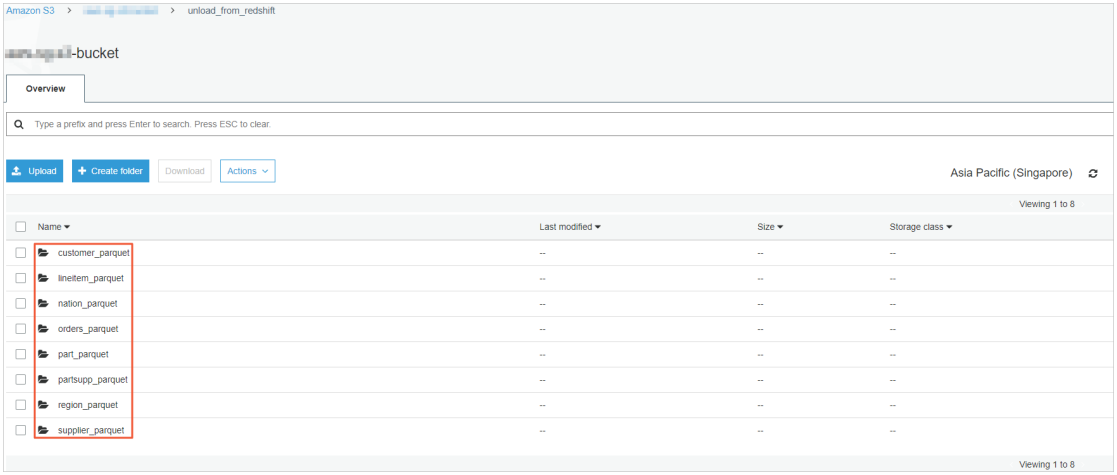
The following sample command shows how to unload data from Amazon Redshift to Amazon S3:

```
UNLOAD ('SELECT * FROM customer')
TO 's3://bucket_name/unload_from_redshift/customer_parquet/customer_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
UNLOAD ('SELECT * FROM orders')
TO 's3://bucket_name/unload_from_redshift/orders_parquet/orders_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
UNLOAD ('SELECT * FROM lineitem')
TO 's3://bucket_name/unload_from_redshift/lineitem_parquet/lineitem_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
UNLOAD ('SELECT * FROM nation')
TO 's3://bucket_name/unload_from_redshift/nation_parquet/nation_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
UNLOAD ('SELECT * FROM part')
TO 's3://bucket_name/unload_from_redshift/part_parquet/part_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
UNLOAD ('SELECT * FROM partsupp')
TO 's3://bucket_name/unload_from_redshift/partsupp_parquet/partsupp_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
UNLOAD ('SELECT * FROM region')
TO 's3://bucket_name/unload_from_redshift/region_parquet/region_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
UNLOAD ('SELECT * FROM supplier')
TO 's3://bucket_name/unload_from_redshift/supplier_parquet/supplier_'
FORMAT AS PARQUET
IAM_ROLE 'arn:aws:iam::xxxx:role/redshift_s3_role';
```

> **Note** You can submit multiple UNLOAD commands at a time in **EDITOR**.

iii. Log on to the Amazon S3 console and check the unloaded data in the directory of each destination bucket on Amazon S3.

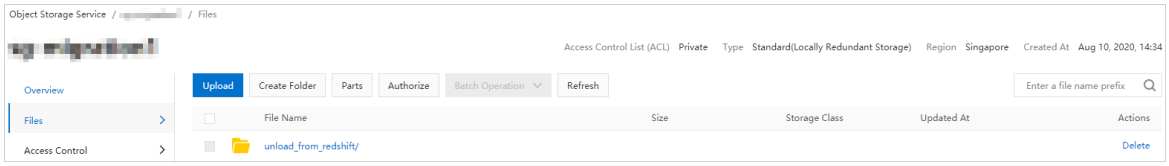
The unloaded data is available in the Apache Parquet format.



Step 2: Migrate the unloaded data from Amazon S3 to OSS

In MaxCompute, you can use the **Data Online Migration** service of OSS to migrate data from Amazon S3 to OSS. For more information, see Migrate data from Amazon Simple Storage Service (Amazon S3) to OSS. The **Data Online Migration** service is in public preview. Before you use this service, you must submit a ticket to contact the Customer Service to activate the service.

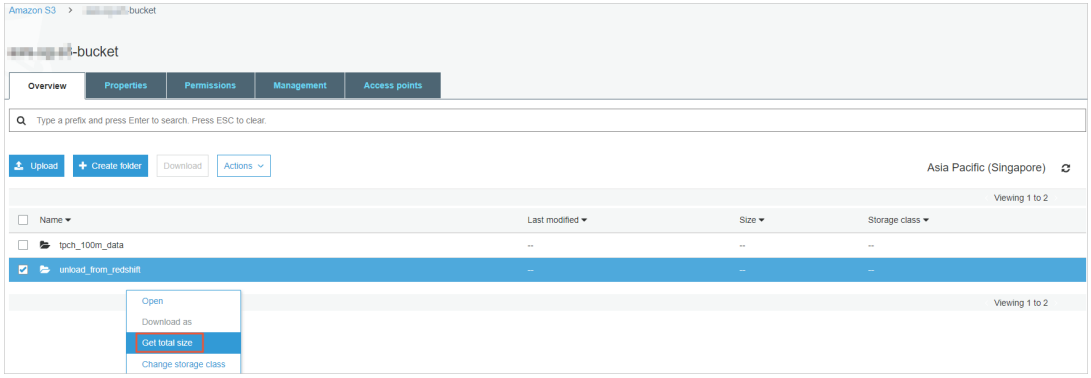
1. Log on to the OSS console, and create a bucket to save the migrated data. For more information, see Create buckets.



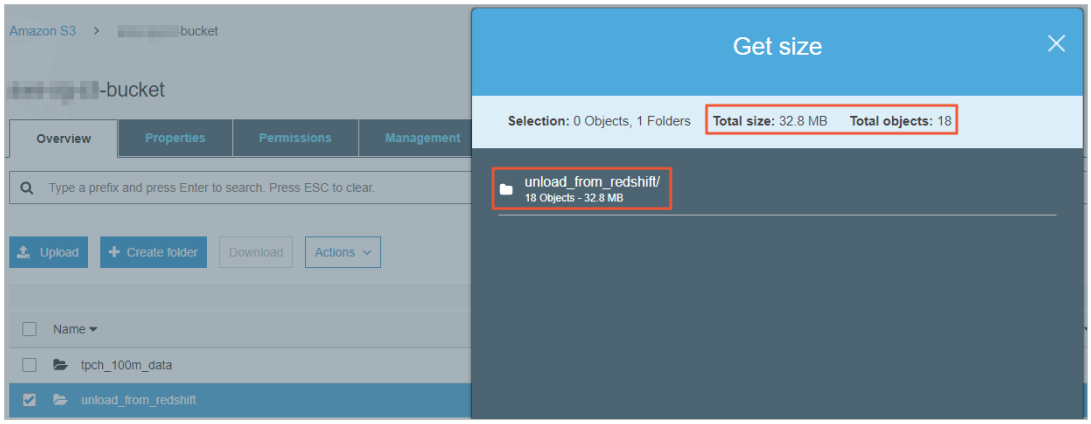
2. Create a Resource Access Management (RAM) user and grant relevant permissions to the RAM user.
   i. Log on to the RAM console and create a RAM user. For more information, see Create a RAM user.
   ii. Find the RAM user that you created, and click **Add Permissions** in the Actions column. On the page that appears, select **AliyunOSSFullAccess** and **AliyunMGWFullAccess**, and click **OK** and complete. The AliyunOSSFullAccess policy authorizes the RAM user to read data from and write data to OSS buckets. The AliyunMGWFullAccess policy authorizes the RAM user to perform online migration jobs.
   iii. In the left-side navigation pane, click **Overview**. In the **Account Management** section of the **Overview** page, click the link under **RAM user logon**, and use the credentials of the RAM user to log on to the Alibaba Cloud Management Console.
3. On the Amazon Web Services (AWS) platform, create an IAM user who uses the programmatic access method to access Amazon S3.
   i. Log on to the Amazon S3 console.

ii. Right-click the exported folder and select **Get total size** to obtain the total size of the folder and the number of files in the folder.

- Obtain the total size.

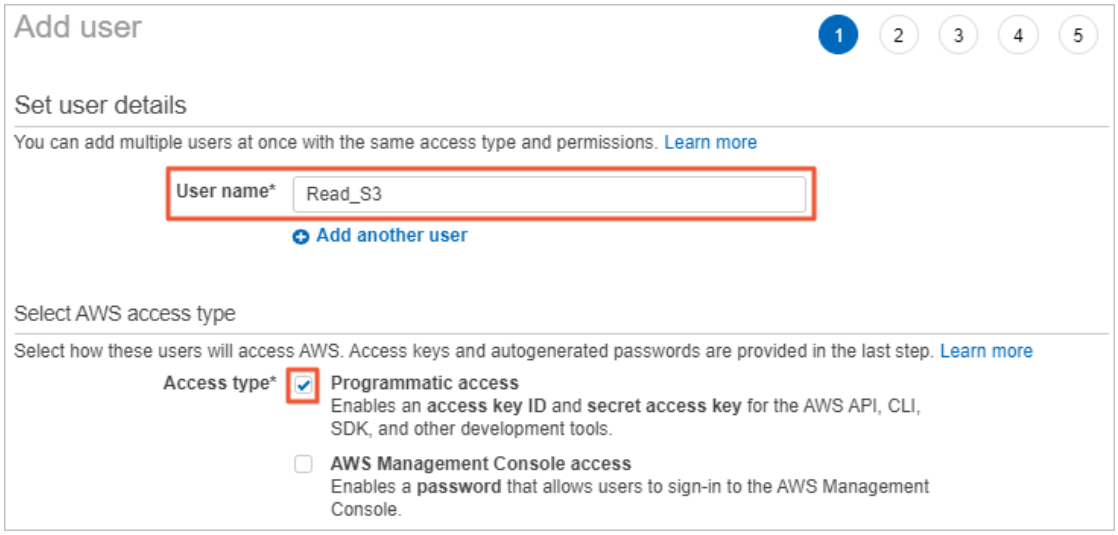


- Obtain the total size of the folder and the number of files in the folder.

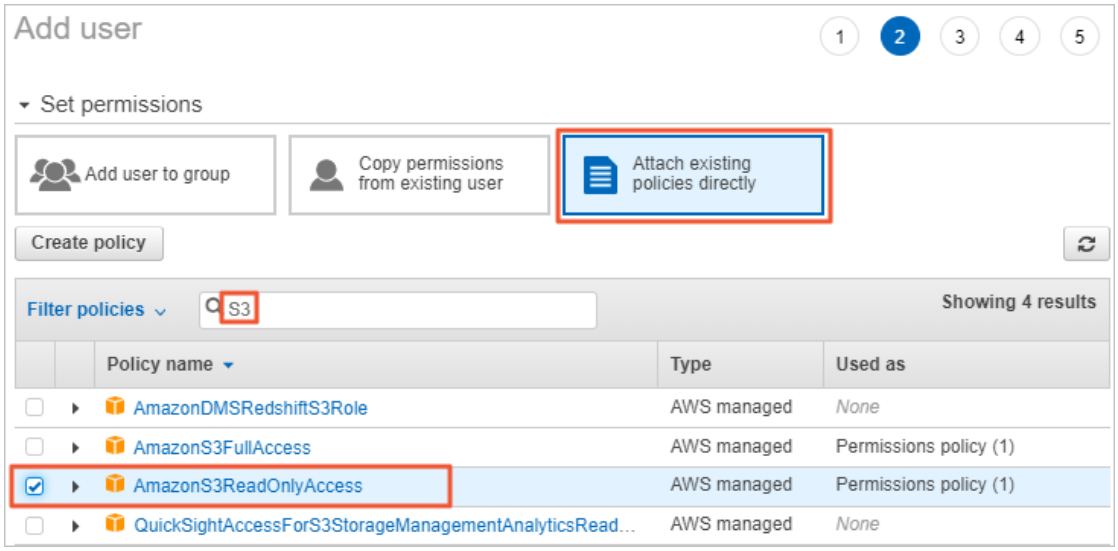


iii. Log on to the IAM console and click **Add user**.

iv. On the **Add user** page, specify the **User name**. In the **Select AWS access type** section, select **Programmatic access** and then click **Next: Permissions**.

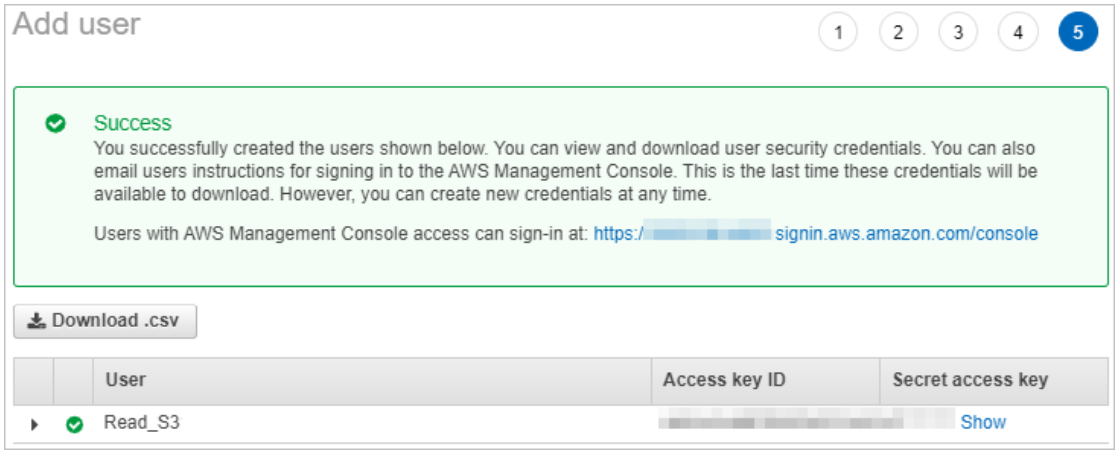


v. On the **Add user** page, click **Attach existing policies directly**. Enter **S3** in the search box, select the **AmazonS3ReadOnlyAccess** policy, and then click **Next: Tags**.



vi. Click **Next: Review > Create user**. The IAM user is created. Obtain the AccessKey pair.

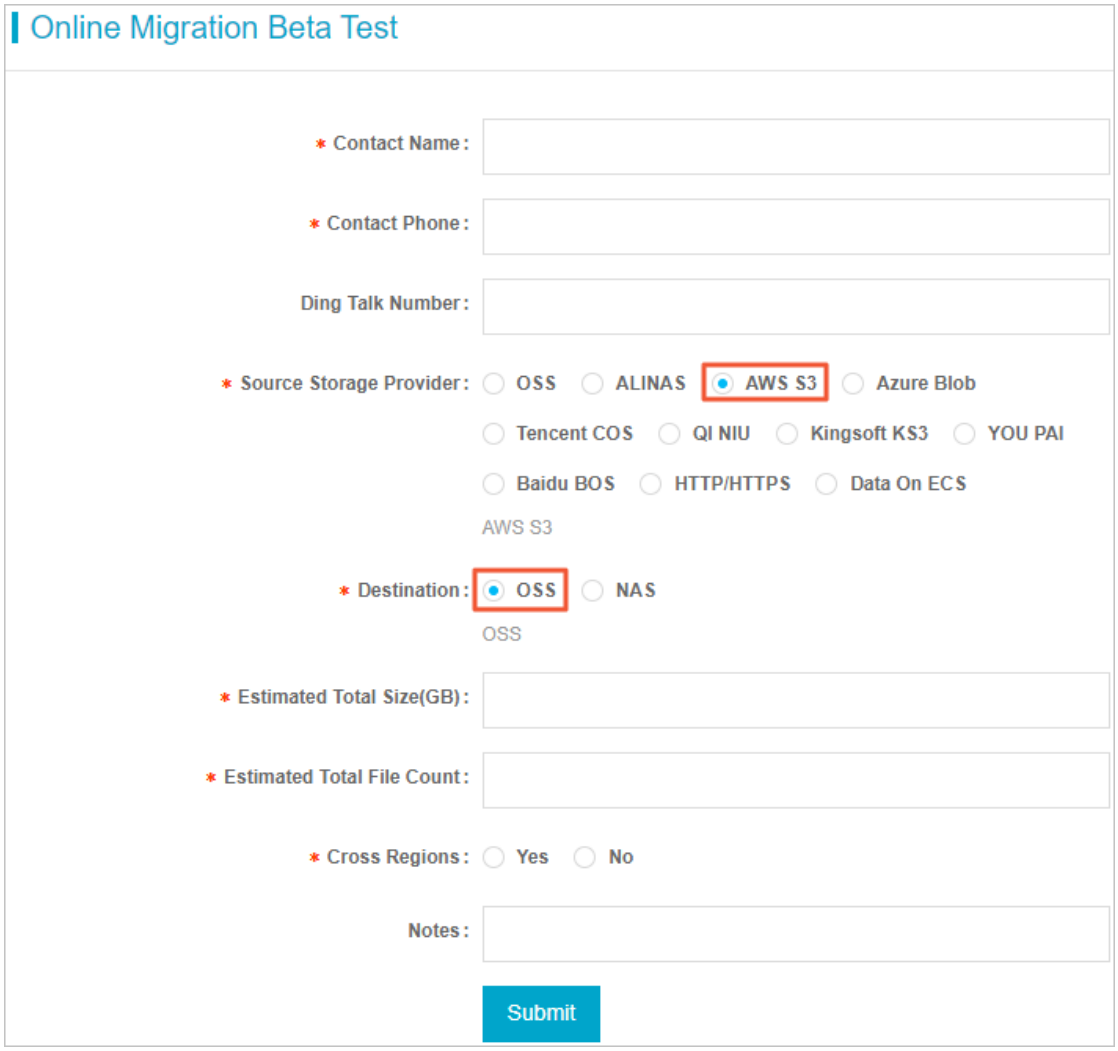
If you create an online migration job, you must provide this AccessKey pair.



4. Create a source data address and a destination data address for online migration.

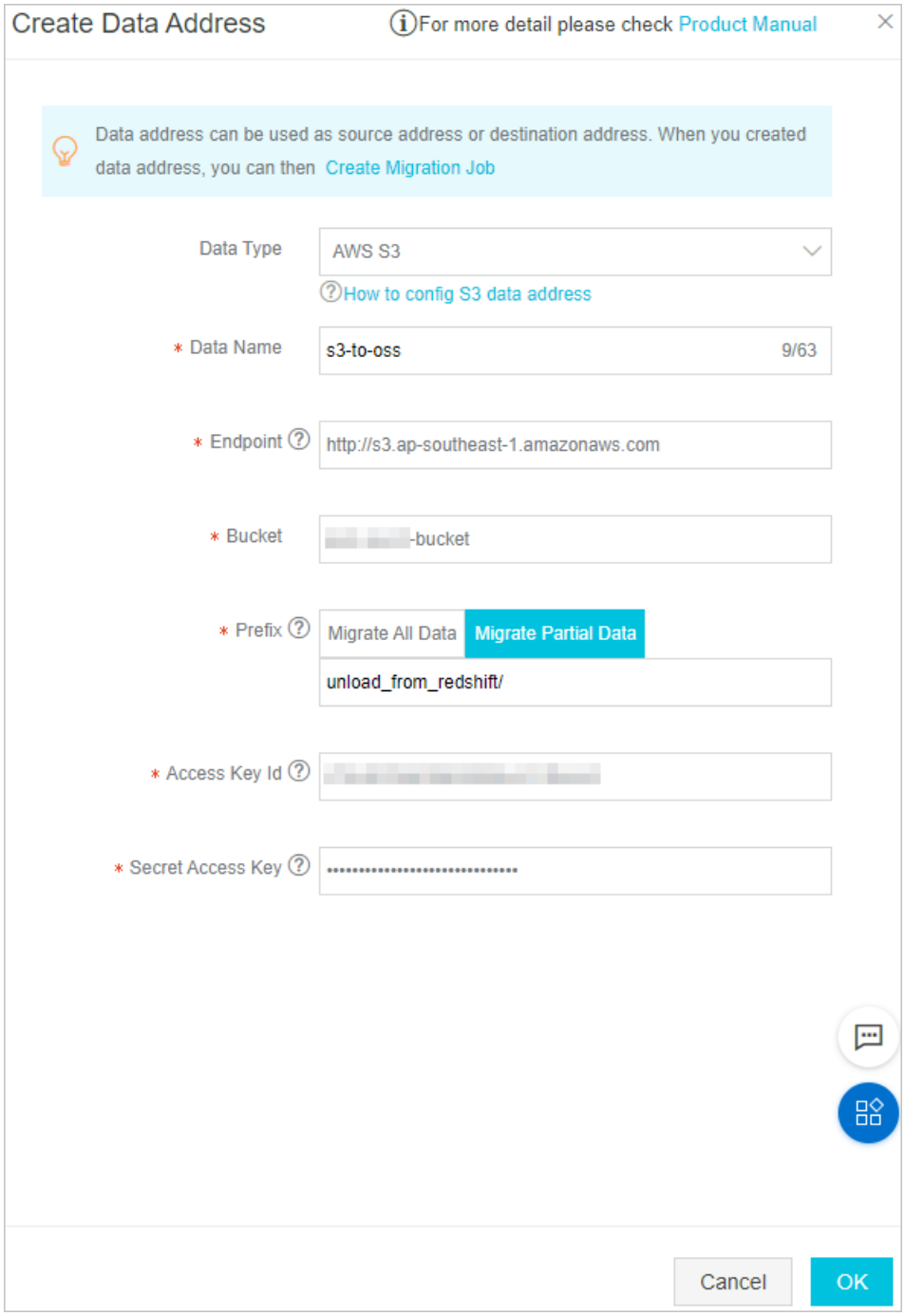
i. Log on to the Alibaba Cloud Data Transport console. In the left-side navigation pane, click **Data Address**.

ii. (Optional)If you have not activated Data Online Migration, click **Application** in the dialog box that appears. On the **Online Migration Beta Test** page, specify the required information and click **Submit**.



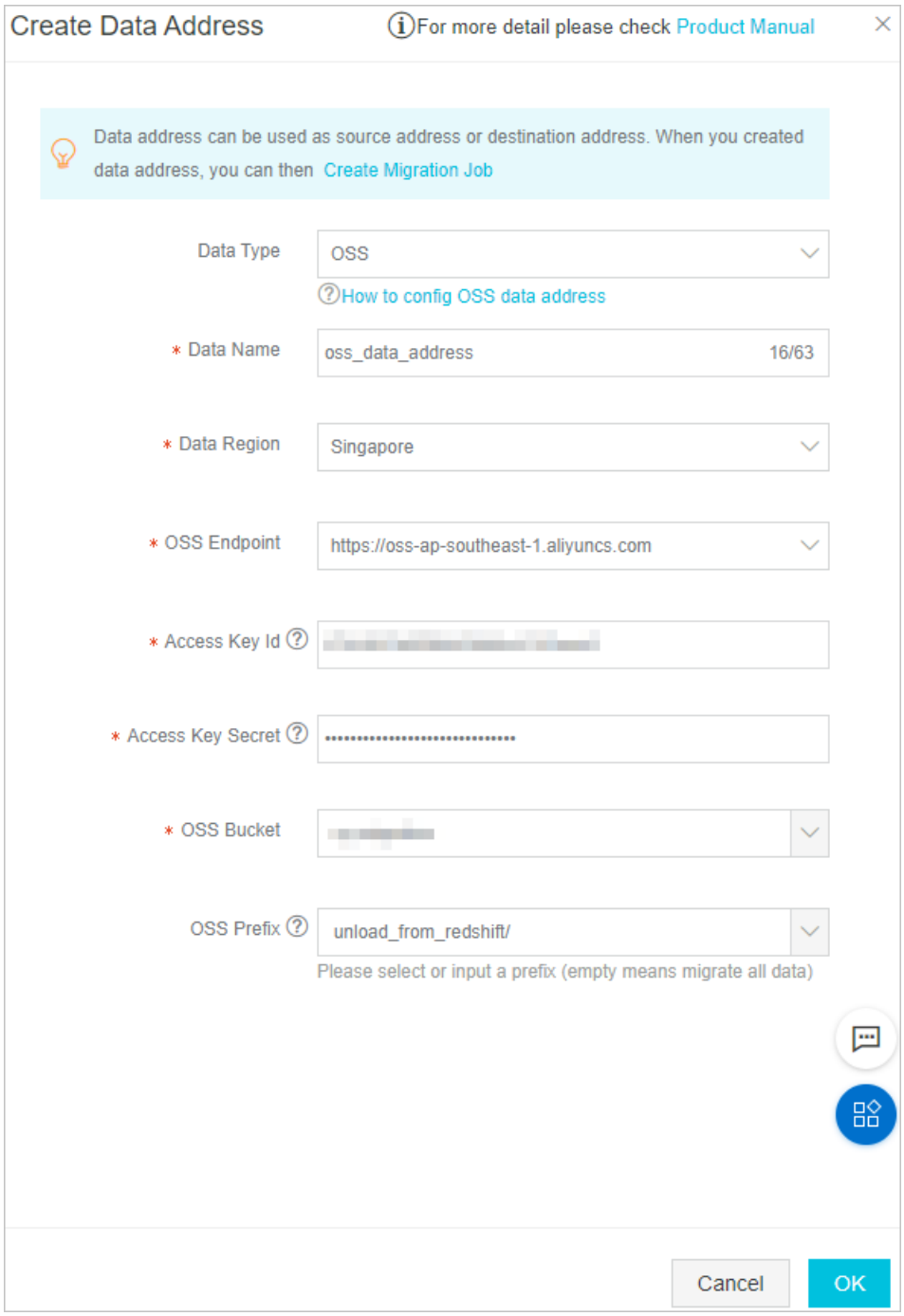
iii. On the **Data Address** page, click **Create Data Address**. In the Create Data Address panel, set the required parameters and click **OK**. For more information about the required parameters, see Migrate data.

- Source data address



Note In the **Access Key Id** and **Access Key Secret** fields, enter the AccessKey ID and AccessKey secret of the IAM user.

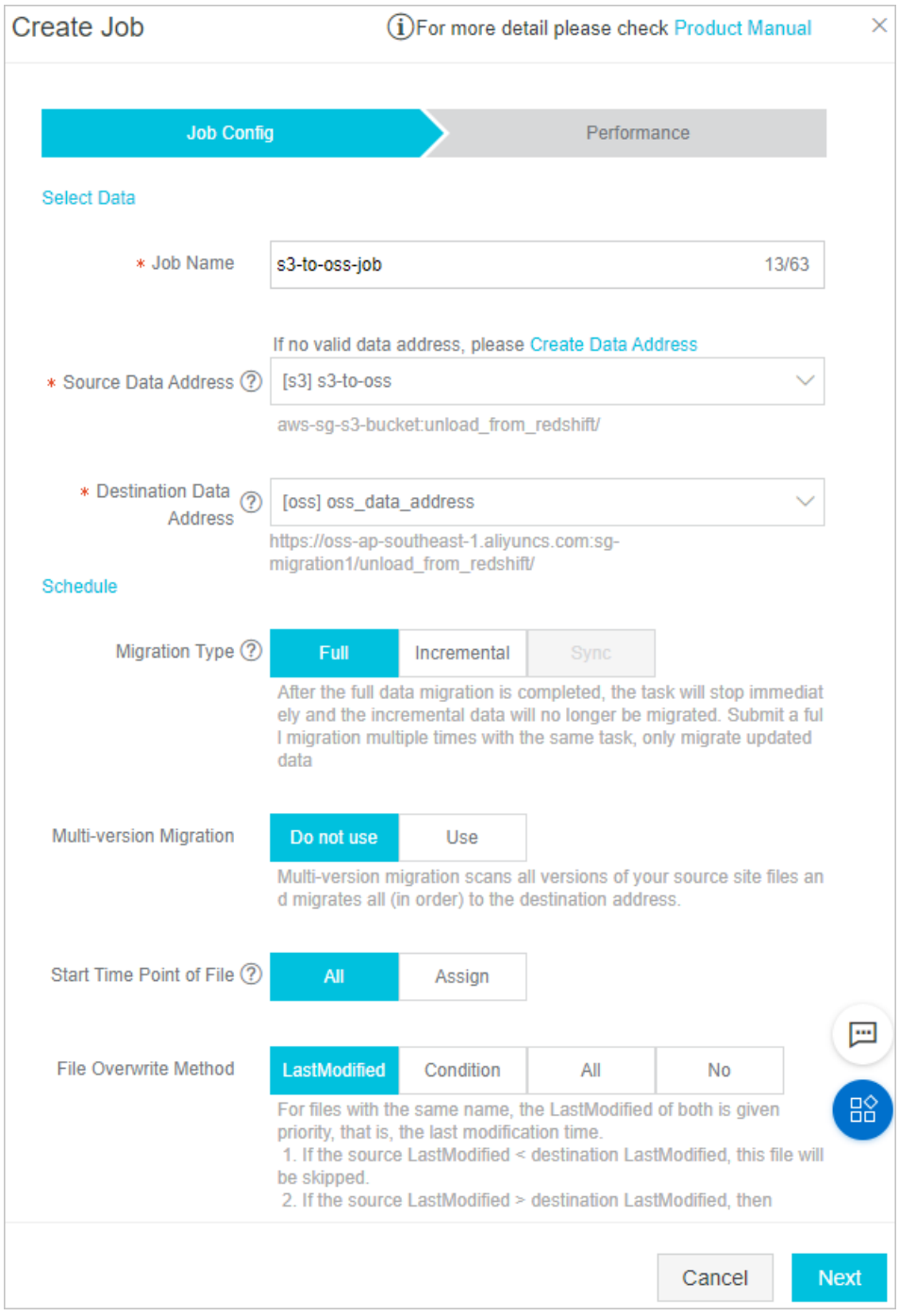
- Destination data address



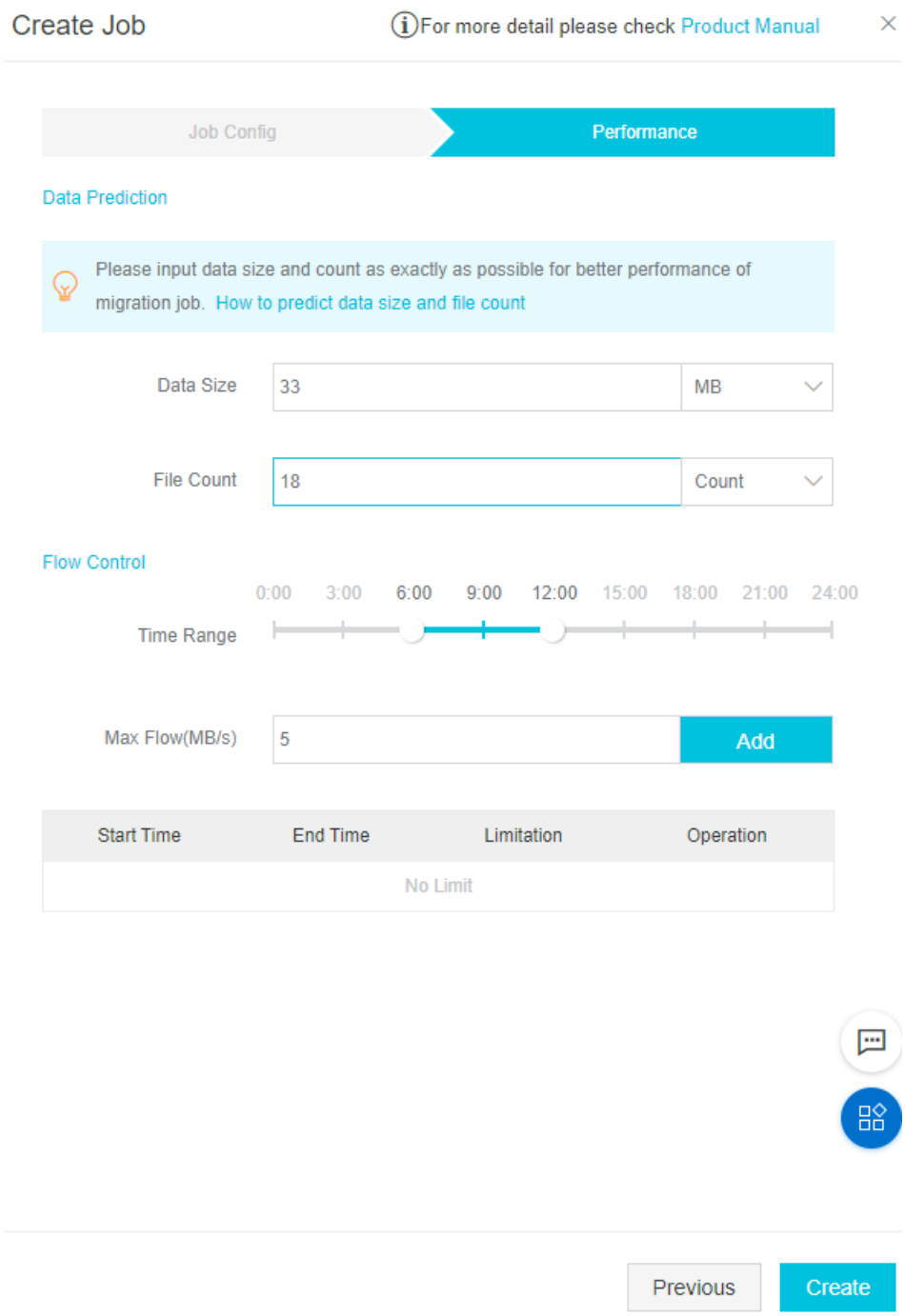
> **Note** In the **Access Key Id** and **Access Key Secret** fields, enter the AccessKey ID and the AccessKey secret of the RAM user.

5. Create an online migration job.
   i. In the left-side navigation pane, click **Migration Jobs**.
   ii. On the **File Sync Management** page, click **Create Job**. In the Create Job wizard, set the required parameters and click **Create**. For more information about the required parameters, see Migrate data.

- Job Config

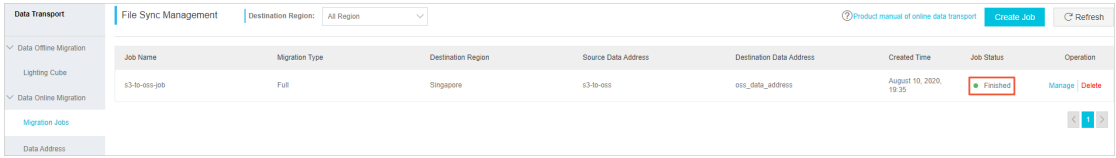


- Performance

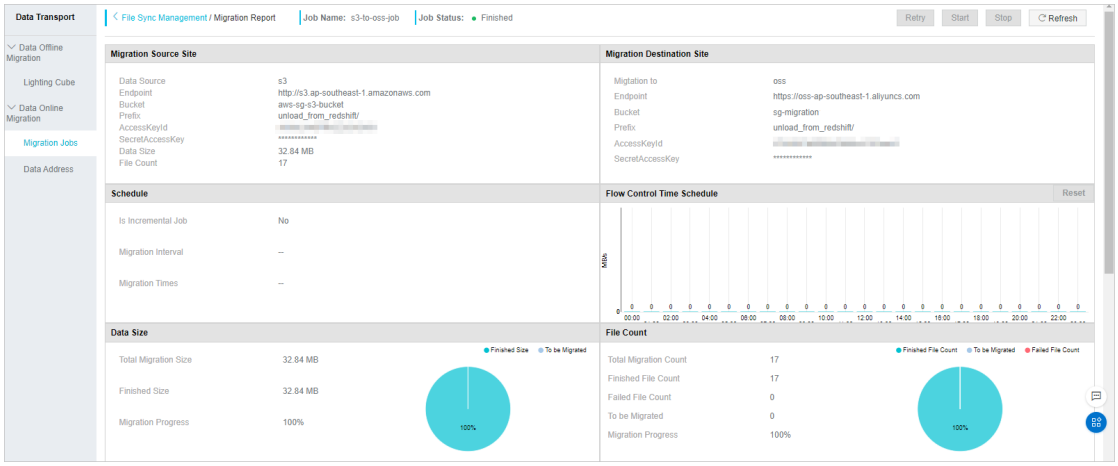


Note In the **Data Size** and **File Count** fields, enter the size and the number of files that you want to migrate from Amazon S3.

iii. The migration job that you created is automatically run. If **Finished** is displayed in the **Job Status** column, the migration job is complete.

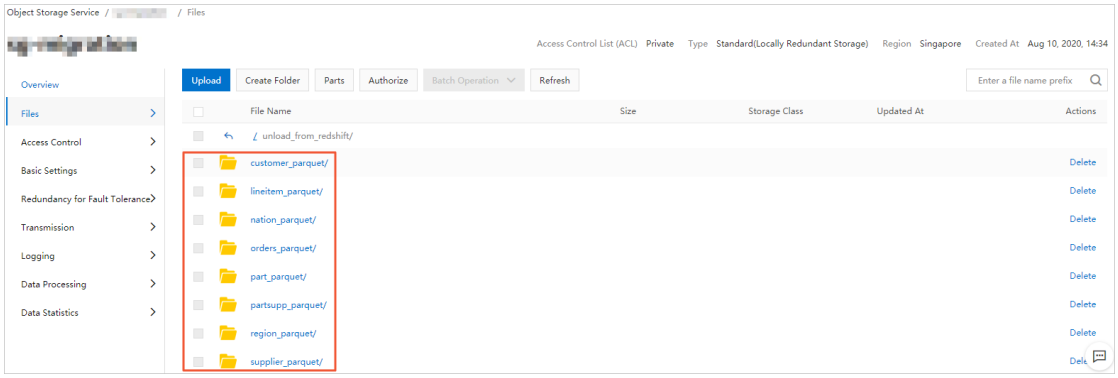


iv. In the Operation column of the migration job, click **Manage** to view the migration report and confirm that all the data is migrated.



v. Log on to the OSS console.

vi. In the left-side navigation pane, click **Buckets**. On the **Buckets** page, click the created bucket. In the left-side navigation pane of the bucket details page, click **Files** to view the migration results.



Step 3: Migrate data from the OSS bucket to the MaxCompute project in the same region

You can run the LOAD command of MaxCompute to migrate data from an OSS bucket to a MaxCompute project in the same region.

The LOAD command supports Security Token Service (STS) and AccessKey for authentication. If you use AccessKey for authentication, you must provide the AccessKey ID and AccessKey secret of your account in plaintext. STS authentication is highly secure because it does not expose the AccessKey pair. In this section, STS authentication is used as an example to show how to migrate data.

1. On the Ad-Hoc Query tab of DataWorks or the MaxCompute client odpscmd, execute the DDL statements to create tables to store the migrated data in MaxCompute. The DDL statements that you execute must be the same as those executed in the Amazon Redshift cluster.

   For more information about ad hoc queries, see Use the ad-hoc query feature to execute SQL statements (optional). The following sample commands show how to create tables:

```
CREATE TABLE customer(
C_CustKey int ,
C_Name varchar(64) ,
C_Address varchar(64) ,
```

```
C_NationKey int ,
C_Phone varchar(64) ,
C_AcctBal decimal(13, 2) ,
C_MktSegment varchar(64) ,
C_Comment varchar(120) ,
skip varchar(64)
);
CREATE TABLE lineitem(
L_OrderKey int ,
L_PartKey int ,
L_SuppKey int ,
L_LineNumber int ,
L_Quantity int ,
L_ExtendedPrice decimal(13, 2) ,
L_Discount decimal(13, 2) ,
L_Tax decimal(13, 2) ,
L_ReturnFlag varchar(64) ,
L_LineStatus varchar(64) ,
L_ShipDate timestamp ,
L_CommitDate timestamp ,
L_ReceiptDate timestamp ,
L_ShipInstruct varchar(64) ,
L_ShipMode varchar(64) ,
L_Comment varchar(64) ,
skip varchar(64)
);
CREATE TABLE nation(
N_NationKey int ,
N_Name varchar(64) ,
N_RegionKey int ,
N_Comment varchar(160) ,
skip varchar(64)
);
CREATE TABLE orders(
O_OrderKey int ,
O_CustKey int ,
O_OrderStatus varchar(64) ,
O_TotalPrice decimal(13, 2) ,
O_OrderDate timestamp ,
O_OrderPriority varchar(15) ,
O_Clerk varchar(64) ,
O_ShipPriority int ,
O_Comment varchar(80) ,
skip varchar(64)
);
CREATE TABLE part(
P_PartKey int ,
P_Name varchar(64) ,
P_Mfgr varchar(64) ,
P_Brand varchar(64) ,
P_Type varchar(64) ,
P_Size int ,
P_Container varchar(64) ,
P_RetailPrice decimal(13, 2) ,
```

```
P_Comment varchar(64) ,
skip varchar(64)
);
CREATE TABLE partsupp(
PS_PartKey int ,
PS_SuppKey int ,
PS_AvailQty int ,
PS_SupplyCost decimal(13, 2) ,
PS_Comment varchar(200) ,
skip varchar(64)
);
CREATE TABLE region(
R_RegionKey int ,
R_Name varchar(64) ,
R_Comment varchar(160) ,
skip varchar(64)
);
CREATE TABLE supplier(
S_SuppKey int ,
S_Name varchar(64) ,
S_Address varchar(64) ,
S_NationKey int ,
S_Phone varchar(18) ,
S_AcctBal decimal(13, 2) ,
S_Comment varchar(105) ,
skip varchar(64)
);
```

In this example, the project uses the MaxCompute V2.0 data types because the TPC-H dataset uses the MaxCompute V2.0 data types and the Decimal 2.0 data type. If you want to configure the project to use the MaxCompute V2.0 data types and the Decimal 2.0 data type, add the following commands at the beginning of the CREATE TABLE statements:

```
setproject odps.sql.type.system.odps2=true;
setproject odps.sql.decimal.odps2=true;
```

2. Create a RAM role that has the OSS access permissions and assign the RAM role to the RAM user. For more information, see STS authorization.
3. Run the LOAD command multiple times to load all data from OSS to the MaxCompute tables that you created, and execute the SELECT statement to query and verify the imported data. For more information about the LOAD command, see LOAD.

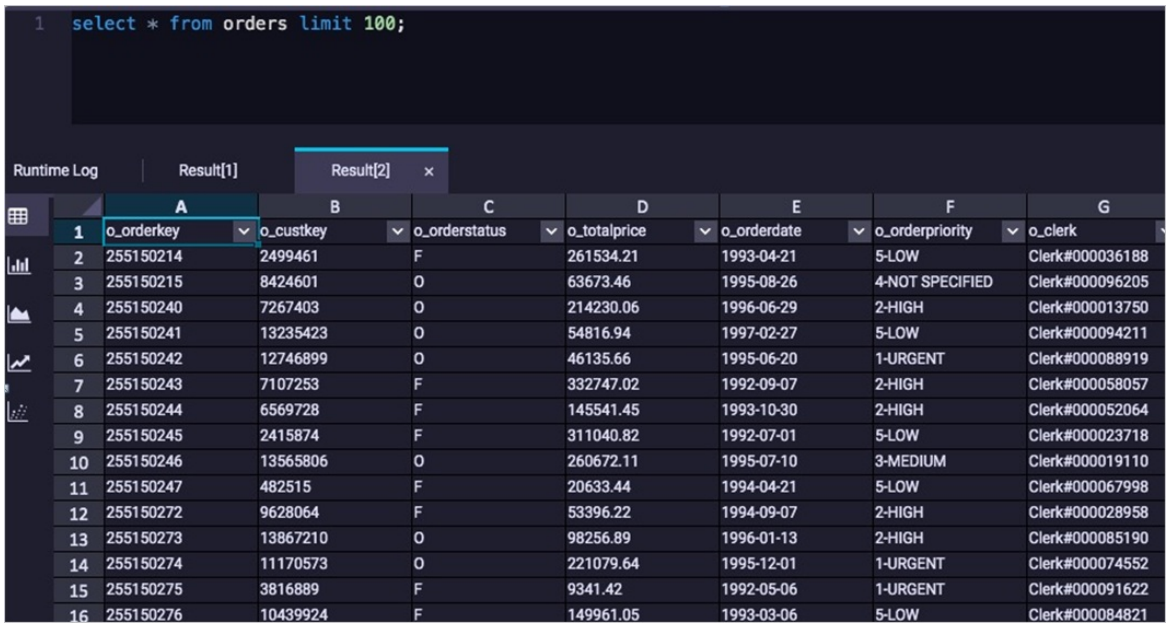
```
LOAD OVERWRITE TABLE orders
FROM LOCATION 'oss://endpoint/oss_bucket_name/unload_from_redshift/orders_parquet/' --
The endpoint of the OSS bucket.
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
WITH SERDEPROPERTIES ('odps.properties.rolearn'='acs:ram::xxx:role/xxx_role')
STORED AS PARQUET;
```

> **Note** If the data import fails, submit a ticket to contact the MaxCompute team.

Execute the following statement to query and verify the imported data:

```
SELECT * FROM orders limit 100;
```

The statement returns the following output.



```
1  select * from orders limit 100;
```

Runtime Log | Result[1] | Result[2]

	A	B	C	D	E	F	G
1	o_orderkey	o_custkey	o_orderstatus	o_totalprice	o_orderdate	o_orderpriority	o_clerk
2	255150214	2499461	F	261534.21	1993-04-21	5-LOW	Clerk#000036188
3	255150215	8424601	O	63673.46	1995-08-26	4-NOT SPECIFIED	Clerk#000096205
4	255150240	7267403	O	214230.06	1996-06-29	2-HIGH	Clerk#000013750
5	255150241	13235423	O	54816.94	1997-02-27	5-LOW	Clerk#000094211
6	255150242	12746899	O	46135.66	1995-06-20	1-URGENT	Clerk#000088919
7	255150243	7107253	F	332747.02	1992-09-07	2-HIGH	Clerk#000058057
8	255150244	6569728	F	145541.45	1993-10-30	2-HIGH	Clerk#000052064
9	255150245	2415874	F	311040.82	1992-07-01	5-LOW	Clerk#000023718
10	255150246	13565806	O	260672.11	1995-07-10	3-MEDIUM	Clerk#000019110
11	255150247	482515	F	20633.44	1994-04-21	5-LOW	Clerk#000067998
12	255150272	9628064	F	53396.22	1994-09-07	2-HIGH	Clerk#000028958
13	255150273	13867210	O	98256.89	1996-01-13	2-HIGH	Clerk#000085190
14	255150274	11170573	O	221079.64	1995-12-01	1-URGENT	Clerk#000074552
15	255150275	3816889	F	9341.42	1992-05-06	1-URGENT	Clerk#000091622
16	255150276	10439924	F	149961.05	1993-03-06	5-LOW	Clerk#000084821

4. Verify that the data migrated to MaxCompute is the same as the data in Amazon Redshift. This verification is based on the number of tables, the number of rows, and the query results of typical jobs.

   i. Log on to the Amazon Redshift console. In the upper-right corner, select **Asia Pacific (Singapore)** from the drop-down list. In the left-side navigation pane, click **EDITOR**. Execute the following statement to query data:

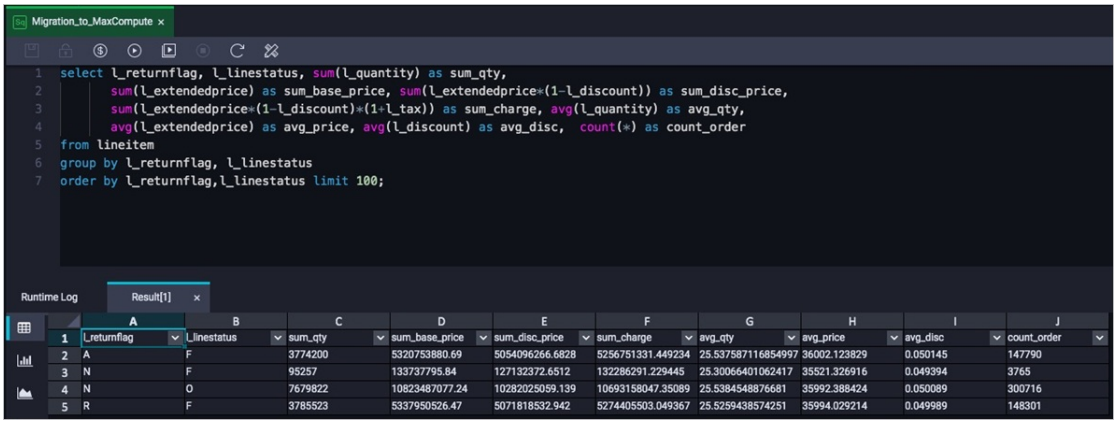
```
SELECT l_returnflag, l_linestatus, SUM(l_quantity) as sum_qty,
SUM(l_extendedprice) AS sum_base_price, SUM(l_extendedprice*(1-l_discount)) AS sum_
disc_price,
SUM(l_extendedprice*(1-l_discount)*(1+l_tax)) AS sum_charge, AVG(l_quantity) AS avg
_qty,
AVG(l_extendedprice) AS avg_price, AVG(l_discount) AS avg_disc,  COUNT(*) AS count_
order
FROM lineitem
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag,l_linestatus;
```

   The statement returns the following output.

l_returnflag	l_linestatus	sum_qty	sum_base_price	sum_disc_price	sum_charge	avg_qty
A	F	3774200	5320753880.69	5054096266.6828	5256751331.449234	25
N	F	95257	133737795.84	127132372.6512	132286291.229445	25
N	O	7679822	10823487077.24	10282025059.1390	10693158047.350890	25
R	F	3785523	5337950526.47	5071818532.9420	5274405503.049367	25

ii. On the Ad-Hoc Query tab of DataWorks or the MaxCompute client (odpscmd), execute the preceding statement and check whether the returned results are consistent with the data that is queried from the Amazon Redshift cluster.

The following output is returned.



2.14. Migrate data from BigQuery to MaxCompute

This topic describes how to migrate data from BigQuery on Google Cloud Platform (GCP) to Alibaba Cloud MaxCompute over the Internet.

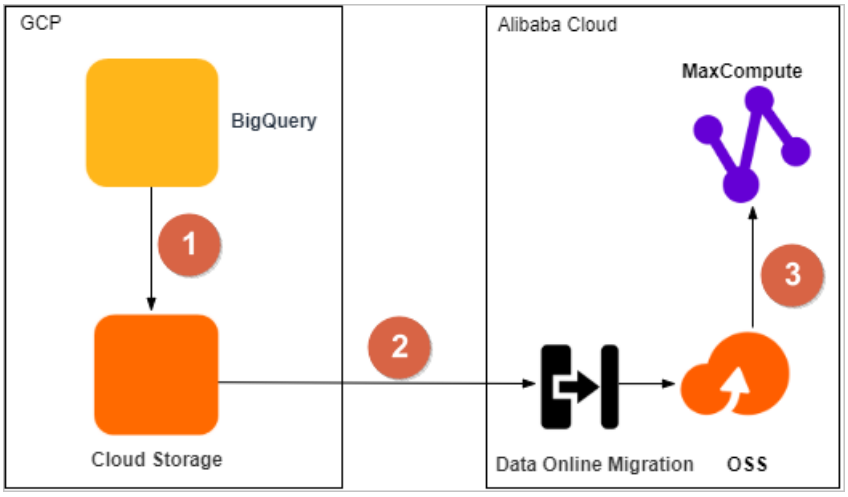
Prerequisites

Category	Platform	Requirement	Reference
Environment and data	Google Cloud Platform	• The Google BigQuery service is activated, and the environment and datasets for migration are prepared. • The Google Cloud Storage service is activated and a bucket is created.	If you do not have the relevant environment and datasets, see the following references for preparation: • BigQuery: Quickstarts and Creating datasets • Google Cloud Storage: Quickstart: Using the Console and Creating storage buckets.

Category	Platform	Requirement	Reference
	Alibaba Cloud	• The MaxCompute and DataWorks services are activated and a project is created. In this example, a MaxCompute project in the **Indonesia (Jakarta)** region is created as the migration destination. • Object Storage Service (OSS) is activated and a bucket is created. • The Data Online Migration service of OSS is activated.	If you do not have the relevant environment, see the following references for preparation: • MaxCompute and DataWorks: Prepare and Create a MaxCompute project. • OSS: 开通OSS服务 and Create buckets. • Data Online Migration: Submit a ticket or Apply for the service online.
Account	Google Cloud Platform	An Identity and Access Management (IAM) user is created and granted the permissions to access Google Cloud Storage.	IAM permissions for JSON methods
	Alibaba Cloud	A Resource Access Management (RAM) user and a RAM role are created. The RAM user is granted the read and write permissions on OSS buckets and the online migration permissions.	Create a RAM user and STS authorization
Region	Google Cloud Platform	N/A	N/A
	Alibaba Cloud	The OSS bucket and the MaxCompute project are in the same region.	N/A

Context

The following figure shows the process to migrate datasets from BigQuery to Alibaba Cloud MaxCompute.

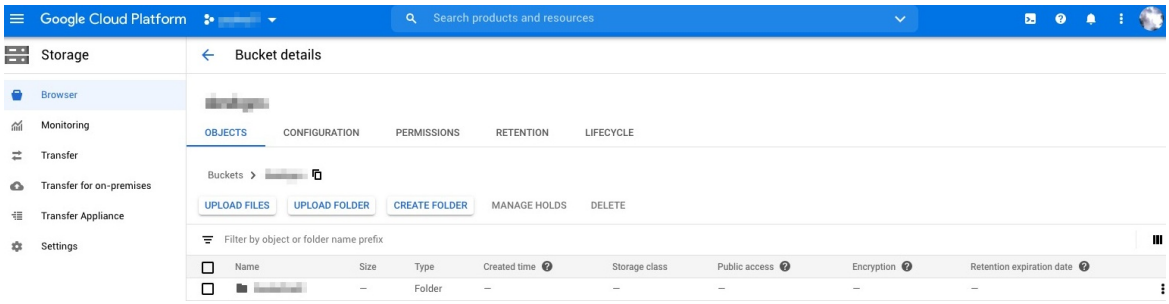


No.	Description
①	Export datasets from BigQuery to Google Cloud Storage.
②	Migrate data from Google Cloud Storage to an OSS bucket by using the **Data Online Migration** service of OSS.
③	Migrate data from the OSS bucket to a MaxCompute project in the same region, and then verify the integrity and accuracy of the migrated data.

Step 1: Export datasets from BigQuery to Google Cloud Storage

Use the bq command-line tool to run the `bq extract` command to export datasets from BigQuery to Google Cloud Storage.

1. Log on to Google Cloud Platform, and create a bucket for the data you want to migrate. For more information, see Creating storage buckets.



2. Use the bq command-line tool to query the Data Definition Language (DDL) scripts of tables in the TPC-DS datasets and download the scripts to an on-premises device. For more information, see Getting table metadata using INFORMATION_SCHEMA.

   BigQuery does not support commands such as `show create table` to query DDL scripts of tables. BigQuery allows you to use built-in user-defined functions (UDFs) to query the DDL scripts of the tables in a dataset. The following code shows examples of DDL scripts.

3. Use the bq command-line tool to run the `bq extract` command to export tables in BigQuery datasets to the destination bucket of Google Cloud Storage. For more information about the operations, formats of exported data, and compression types, see Exporting table data.

   The following code shows a sample extract command:

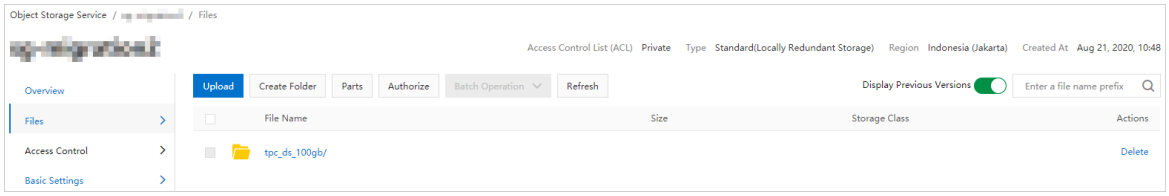
```
bq extract
--destination_format AVRO
--compression SNAPPY
tpcds_100gb.web_site
gs://bucket_name/web_site/web_site-*.avro.snappy;
```

4. View the bucket and check the data export result.

Step 2: Migrate the exported data from Google Cloud Storage to OSS

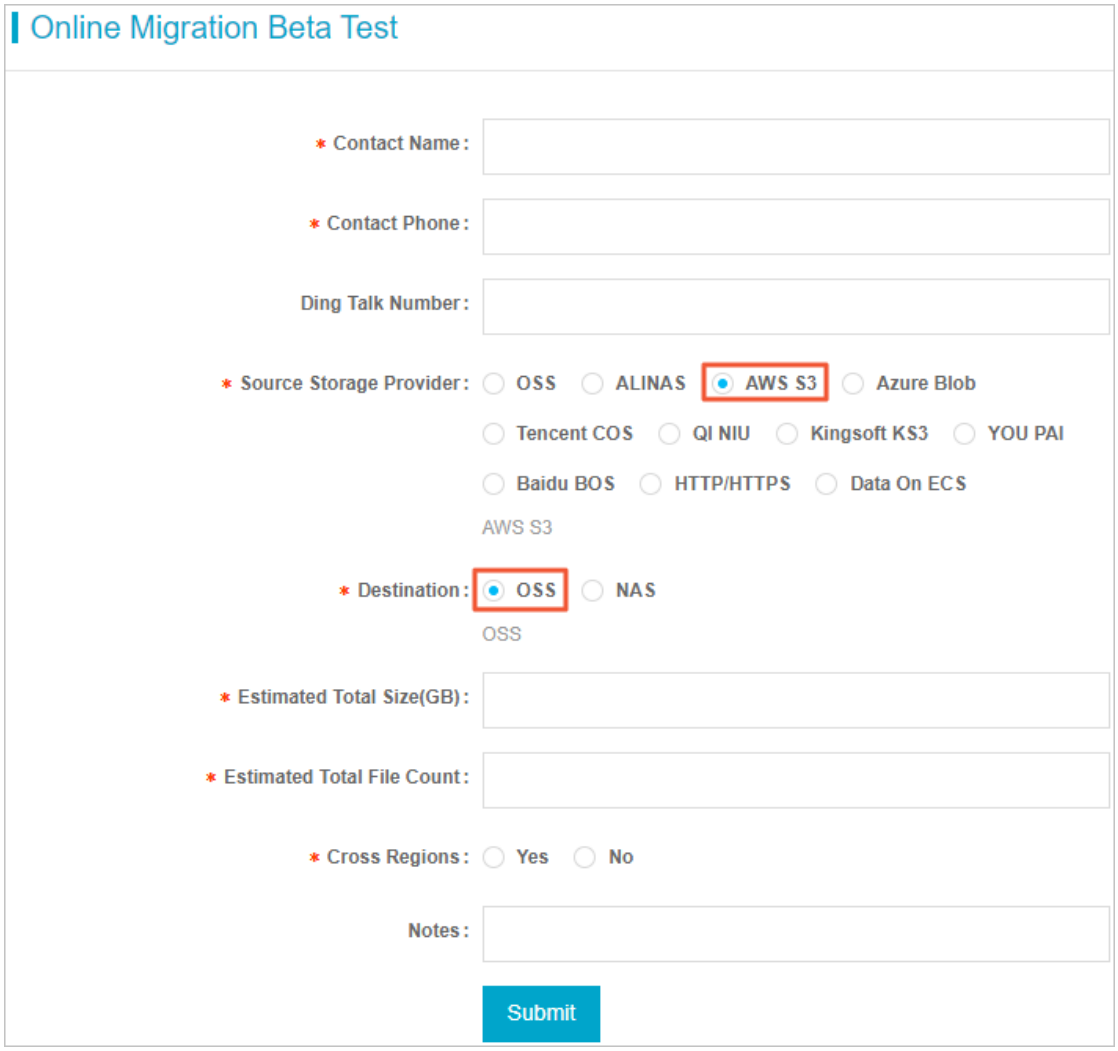
You can use the **Data Online Migration** service to migrate data from Google Cloud Storage to OSS. For more information, see Migrate data from Google Cloud Platform to OSS. The **Data Online Migration** service is in public preview. Before you use the service, you must submit a ticket to contact the Customer Service to activate the service.

1. Estimate the size and the number of files that you want to migrate. You can query the data size in the bucket of Google Cloud Storage by using the gsutil tool or checking the storage logs. For more information, see Getting bucket information.
2. (Optional)If you do not have a bucket in OSS, log on to the OSS console and create a bucket to store the migrated data. For more information, see Create buckets.



3. (Optional)If you do not have a RAM user, create a RAM user and grant relevant permissions to the RAM user.
   i. Log on to the RAM console and create a RAM user. For more information, see Create a RAM user.
   ii. Find the newly created RAM user, and click **Add Permissions** in the Actions column. On the page that appears, select **AliyunOSSFullAccess** and **AliyunMGWFullAccess**, and click **OK** > **Complete**. The AliyunOSSFullAccess permission authorizes the RAM user to read and write OSS buckets. The AliyunMGWFullAccess permission authorizes the RAM user to perform online migration jobs.
   iii. In the left-side navigation pane, click **Overview**. In the **Account Management** section of the **Overview** page, click the link under **RAM user logon**, and use the credentials of the RAM user to log on to the Alibaba Cloud Management Console.
4. On Google Cloud Platform, create a user who uses the programmatic access method to access Google Cloud Storage. For more information, see IAM permissions for JSON methods.
   i. Log on to the IAM & Admin console, and find a user who has permissions to access BigQuery. In the **Actions** column, click > **Create key**.
   ii. In the dialog box that appears, select **JSON**, and click **CREATE**. Save the JSON file to an on-premises device and click **CLOSE**.
   iii. In the **Create service account** wizard, click **Select a role**, and choose **Cloud Storage** > **Storage Admin** to authorize the IAM user to access Google Cloud Storage.
5. Create a source data address and a destination data address for online data migration.
   i. Log on to the Alibaba Cloud Data Transport console. In the left-side navigation pane, click **Data Address**.

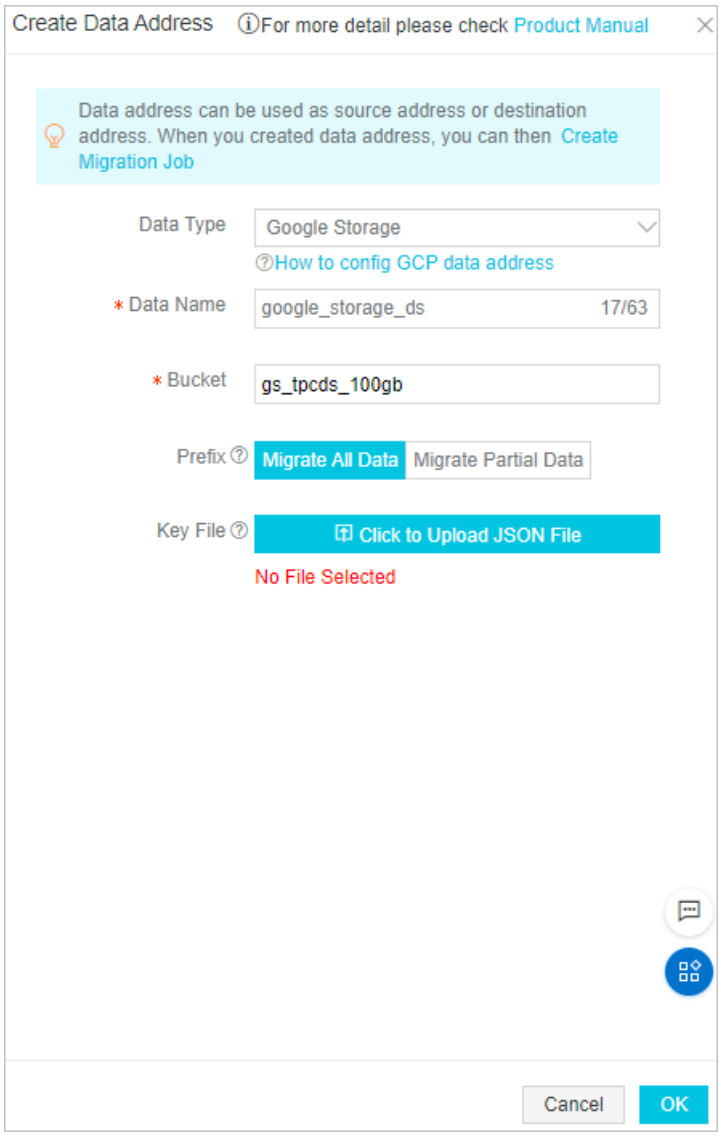
ii. (Optional)If you have not activated the Data Online Migration service, click **Application** in the dialog box that appears. On the **Online Migration Beta Test** page, specify the required information and click **Submit**.



> **Note** On the **Online Migration Beta Test** page, if the **Source Storage Provider** options do not include Google Cloud Platform, select a source storage provider and specify the actual source storage provider in the **Notes** field.

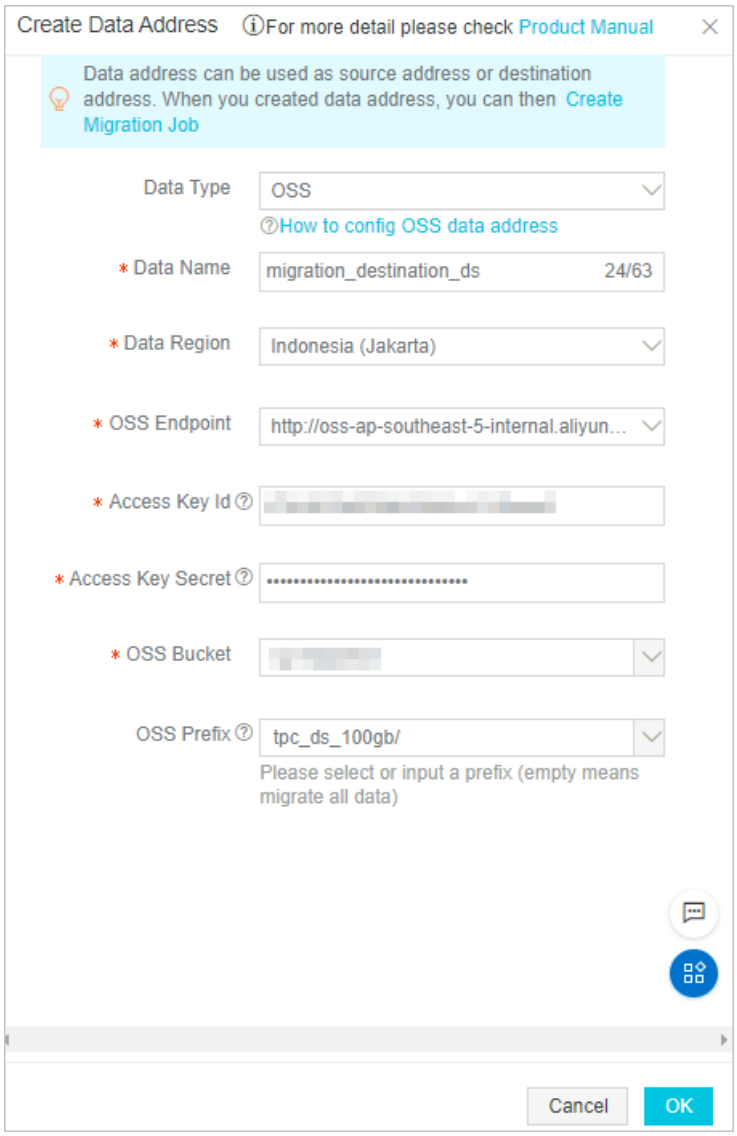
iii. On the **Data Address** page, click **Create Data Address**. In the Create Data Address dialog box, set the required parameters and click **OK**. For more information about the parameters, see Migrate data.

- Source data address



> **Note** For the **Key File** field, upload the JSON file that is downloaded in Step 4.

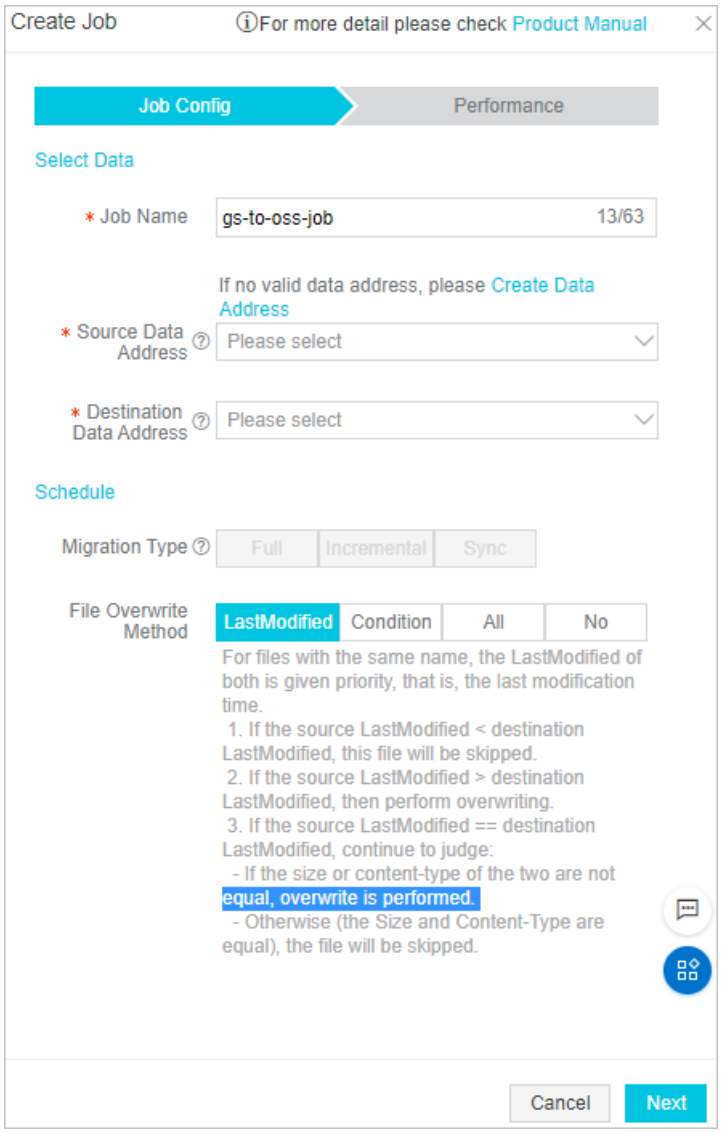
- Destination data address



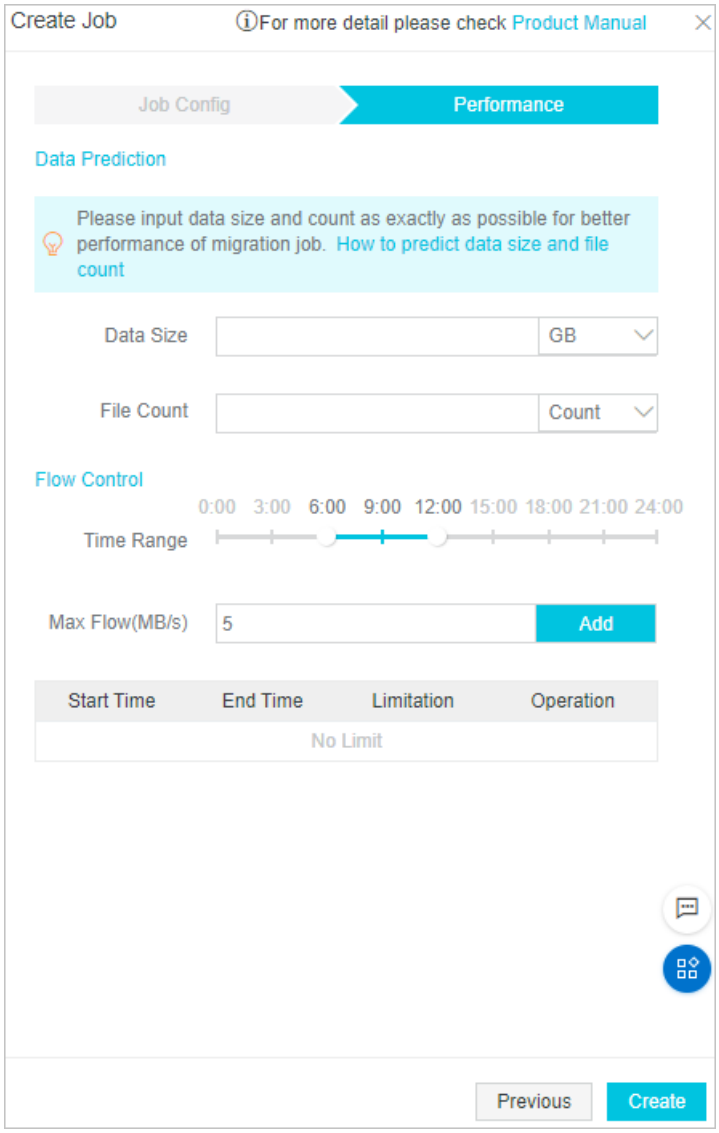
> **Note** In the **Access Key Id** and **Access Key Secret** fields, enter the AccessKey ID and the AccessKey secret of the RAM user.

6. Create an online migration job.
   i. In the left-side navigation pane, click **Migration Jobs**.
   ii. On the **File Sync Management** page, click **Create Job**. In the Create Job wizard, set the required parameters and click **Create**. For more information about the parameters, see Migrate data.

- Job Config



- Performance



Note In the **Data Size** and **File Count** fields, enter the size and the number of files that were migrated from Google Cloud Platform.

iii. The created migration job is automatically run. If **Finished** is displayed in the **Job Status** column, the migration job is complete.

iv. In the Operation column of the migration job, click **Manage** to view the migration report and confirm that all data is migrated.

v. Log on to the OSS console.

vi. In the left-side navigation pane, click **Buckets**. On the **Buckets** page, click the created bucket. In the left-side navigation pane of the bucket details page, choose Files > **Files** to view the migration results.

Step 3: Migrate data from the OSS bucket to a MaxCompute project in the same region

You can execute the LOAD statement of MaxCompute to migrate data from an OSS bucket to a MaxCompute project in the same region.

The LOAD statement supports Security Token Service (STS) and AccessKey for authentication. If you use AccessKey for authentication, you must provide the AccessKey ID and AccessKey secret of your account in plaintext. STS authentication is highly secure because it does not expose the AccessKey information. In this section, STS authentication is used as an example to show how to migrate data.

1. On the Ad-Hoc Query tab of DataWorks or the MaxCompute client odpscmd, modify the DDL scripts of the tables in the BigQuery datasets, specify the MaxCompute data types, and create a destination table that stores the migrated data in MaxCompute.

   For more information about ad hoc queries, see Use the ad-hoc query feature to execute SQL statements (optional). The following code shows a configuration example:

```
CREATE OR REPLACE TABLE
`****.tpcds_100gb.web_site`
(
  web_site_sk INT64,
  web_site_id STRING,
  web_rec_start_date STRING,
  web_rec_end_date STRING,
  web_name STRING,
  web_open_date_sk INT64,
  web_close_date_sk INT64,
  web_class STRING,
  web_manager STRING,
  web_mkt_id INT64,
  web_mkt_class STRING,
  web_mkt_desc STRING,
  web_market_manager STRING,
  web_company_id INT64,
  web_company_name STRING,
  web_street_number STRING,
  web_street_name STRING,
  web_street_type STRING,
  web_suite_number STRING,
  web_city STRING,
  web_county STRING,
  web_state STRING,
  web_zip STRING,
  web_country STRING,
  web_gmt_offset FLOAT64,
  web_tax_percentage FLOAT64
)
Modify the INT64 and
FLOAT64 fields to obtain the following DDL script:
CREATE
TABLE IF NOT EXISTS <your_maxcompute_project>.web_site_load
(
web_site_sk BIGINT,
web_site_id STRING,
web_rec_start_date STRING,
web_rec_end_date STRING,
web_name STRING,
web_open_date_sk BIGINT,
web_close_date_sk BIGINT,
web_class STRING,
```

```
web_manager STRING,
web_mkt_id BIGINT,
web_mkt_class STRING,
web_mkt_desc STRING,
web_market_manager STRING,
web_company_id BIGINT,
web_company_name STRING,
web_street_number STRING,
web_street_name STRING,`
web_street_type STRING,
web_suite_number STRING,
web_city STRING,
web_county STRING,
web_state STRING,
web_zip STRING,
web_country STRING,
web_gmt_offset DOUBLE,
web_tax_percentage DOUBLE
);
```

The following table describes the mapping between BigQuery data types and MaxCompute data types.

BigQuery data type	MaxCompute data type
INT64	BIGINT
FLOAT64	DOUBLE
NUMERIC	DECIMAL and DOUBLE
BOOL	BOOLEAN
STRING	STRING
BYTES	VARCHAR
DATE	DATE
DATETIME	DATETIME
TIME	DATETIME
TIMESTAMP	TIMESTAMP
STRUCT	STRUCT
GEOGRAPHY	STRING

2. (Optional)If you do not have a RAM role, create a RAM role that has the OSS access permissions and assign the role to the RAM user. For more information, see STS authorization.
3. Execute the LOAD statement to load all data from the OSS bucket to the MaxCompute table, and execute the SELECT statement to query and verify the imported data. You can only load one table at a time. To load multiple tables, you must execute the LOAD statement multiple times. For more

information about the LOAD statement, see LOAD.

```
LOAD OVERWRITE TABLE web_site
FROM  LOCATION 'oss://oss-<your_region_id>-internal.aliyuncs.com/bucket_name/tpc_ds_100
gb/web_site/' --The endpoint of the OSS bucket
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
WITH SERDEPROPERTIES ('odps.properties.rolearn'='<Your_RoleARN>','mcfed.parquet.compres
sion'='SNAPPY')
STORED AS AVRO;
```

> **Note** If the data import fails, submit a ticket to contact the MaxCompute team.

Execute the following statement to query and verify the imported data:

```
SELECT * FROM web_site;
```

The statement returns the following output.

	web_site_sk	web_site_id	web_rec_start_date	web_rec_end_date	web_name	web_open_date_sk	web_close_date_sk	web_class	web_manager	web_mkt_id	web_mkt_class	web_mkt_desc	web_market_manager	web_company_id	web_company_na
1	16	AAAAA...	1997-08-16	1999-08-16	site_2	2450679	2447029	Unkno...	Herbert H...	6	Hard new ...	Basic othe...	Albert Leung	6	cally
2	17	AAAAA...	1999-08-17	2001-08-15	site_2	2450679	2447029	Unkno...	Charles C...	5	New, differ...	Basic othe...	Keith Frazier	6	cally
3	18	AAAAA...	2001-08-16	\N	site_2	2450679	2447029	Unkno...	Charles C...	5	Broad, ne...	Basic othe...	Keith Frazier	6	cally
4	1	AAAAA...	1997-08-16	\N	site_0	2450807	\N	Unkno...	Ronald Sh...	4	Grey lines ...	Well simila...	Joe George	6	cally
5	2	AAAAA...	1997-08-16	2000-08-15	site_0	2450798	2443498	Unkno...	Tommy J...	6	Completely...	Lucky pas...	David Myers	4	ese
6	3	AAAAA...	2000-08-16	\N	site_0	2450798	2443498	Unkno...	Tommy J...	3	Completely...	Particular, ...	David Myers	4	ese
7	19	AAAAA...	1997-08-16	\N	site_3	2450654	\N	Unkno...	Jimmy Pope	2	Leaders m...	Home new...	Eldon Snow	5	anti
8	4	AAAAA...	1997-08-16	1999-08-16	site_0	2450781	2447131	Unkno...	Harold Wi...	5	As strong ...	Deeply sm...	James Harris	5	anti
9	5	AAAAA...	1999-08-17	2001-08-15	site_0	2450781	2447131	Unkno...	Harold Wi...	5	Wide, final ...	Deeply sm...	Edward George	1	ought
10	6	AAAAA...	2001-08-16	\N	site_0	2450781	2447131	Unkno...	Harold Wi...	5	Wide, final ...	Thin roles ...	John Sheppard	3	ought
11	20	AAAAA...	1997-08-16	2000-08-15	site_3	2450645	2446995	Unkno...	Peter Cas...	4	Just popul...	Formerly l...	Kelvin Lynch	3	pri
12	21	AAAAA...	2000-08-16	\N	site_3	2450645	2446995	Unkno...	Adam Sto...	5	As private ...	Formerly l...	Jarvis Allen	4	pri
13	22	AAAAA...	1997-08-16	1999-08-16	site_3	2450628	2443328	Unkno...	Lewis Wolf	5	Severe, us...	Individual ...	James Bernard	2	able
14	23	AAAAA...	1999-08-17	2001-08-15	site_3	2450628	2443328	Unkno...	Jason Silva	4	Severe, us...	Individual ...	Jeffrey Martin	2	able
15	24	AAAAA...	2001-08-16	\N	site_3	2450628	2443328	Unkno...	John Ward	4	Severe, us...	Individual ...	Philip Sampson	2	able
16	7	AAAAA...	1997-08-16	\N	site_1	2450756	\N	Unkno...	Moses Hi...	4	New home...	Enough s...	William Reyes	4	ese
17	8	AAAAA...	1997-08-16	2000-08-15	site_1	2450747	2447097	Unkno...	Charles P...	1	Only, temp...	Physical wi...	Gerald Craft	5	anti
18	9	AAAAA...	2000-08-16	\N	site_1	2450747	2447097	Unkno...	Marshall ...	6	Labour car...	Physical wi...	Scott Bryant	1	anti
19	10	AAAAA...	1997-08-16	1999-08-16	site_1	2450730	2443430	Unkno...	Dwight A...	3	New, impo...	Companie...	Frank Cooper	2	able
20	11	AAAAA...	1999-08-17	2001-08-15	site_1	2450730	2443430	Unkno...	Dwight A...	3	Right effor...	Companie...	Casey Banks	2	able
21	12	AAAAA...	2001-08-16	\N	site_1	2450730	2443430	Unkno...	Richard F...	3	Right effor...	Acres see ...	Casey Banks	2	able
22	13	AAAAA...	1997-08-16	\N	site_2	2450705	\N	Unkno...	John Tho...	2	About rura...	Political af...	James Brewer	5	anti
23	14	AAAAA...	1997-08-16	2000-08-15	site_2	2450696	2443396	Unkno...	Robert Ar...	5	Necessary,...	Associatio...	Gilbert Chapman	5	anti
24	15	AAAAA...	2000-08-16	\N	site_2	2450696	2443396	Unkno...	James Au...	5	Necessary,...	Associatio...	Zachery Oneil	5	anti

4. Verify that the data migrated to MaxCompute is the same as the data in BigQuery. This verification is based on the number of tables, the number of rows, and the query results of typical jobs.

2.15. Migrate log data to MaxCompute

2.15.1. Overview

Enterprises develop their businesses based on real-time log data. The data includes that for Elastic Compute Service (ECS) instances, containers, mobile terminals, open-source software, website services, and JavaScript. This topic describes how to migrate log data to MaxCompute by using Tunnel, DataHub, Log Service, and DataWorks Data Integration.

Migration method	Description	Scenario
Tunnel	Use Tunnel in MaxCompute to upload log data to MaxCompute. For more information, see Use Tunnel to upload log data to MaxCompute.	This method is used to upload large volumes of offline data to MaxCompute tables. Tunnel is suitable for offline computing.

Migration method	Description	Scenario
DataHub	Use DataHub to migrate data to MaxCompute. DataHub DataConnector synchronizes streaming data from DataHub to MaxCompute. You only need to write data to DataHub and configure the data synchronization feature in DataHub. Then, you can use the data in MaxCompute. For more information, see Use DataHub to migrate log data to MaxCompute.	This method is mainly used for public preview and development. DataHub is used to upload data in real time. It is suitable for stream processing. After data is uploaded to DataHub, the data is stored in a table for real-time processing. DataHub executes scheduled tasks to synchronize the data to a MaxCompute table within a few minutes for offline computing.
DataWorks Data Integration	Configure batch synchronization nodes and synchronization tasks in DataWorks Data Integration to synchronize data to MaxCompute. For more information, see Use DataWorks Data Integration to migrate log data to MaxCompute.	This method is used after you configure batch synchronization tasks in DataWorks Data Integration. The tasks are executed on a regular basis.

2.15.2. Use Tunnel to upload log data to MaxCompute

This topic describes how to use Tunnel to upload log data to MaxCompute.

Prerequisites

- The odpscmd client is installed. For more information, see Install and configure the MaxCompute client.
- Log data is stored in a local directory.loghub.csv is used as an example in this topic.

Context

Tunnel is a tool that can be used to upload large volumes of data to MaxCompute at a time. It is suitable for offline computing. For more information, see Usage notes.

Procedure

1. On the odpscmd client, run the following commands to create a table named *loghub* that is used to store the uploaded data:

```
Enable the new data types supported by MaxCompute V2.0. Commit the following command wi
th the SQL statement that is used to create the table:
set odps.sql.type.system.odps2=true;
-- Create a table named loghub.
CREATE TABLE loghub
(
client_ip    STRING ,
receive_time STRING ,
topic STRING,
id STRING,
name VARCHAR(32),
salenum STRING
);
```

2. Run the following command to upload log data to MaxCompute:

```
Tunnel u D:\loghub.csv loghub;
```

where,

- D:\loghub.csv: specifies the path where the log data file is stored.
- loghub: specifies the name of the MaxCompute table that is used to store log data.

> **Note** Wildcards or regular expressions are not supported for Tunnel-based data uploads.

3. Execute the following statement to check whether the data is uploaded to the table:

```
SELECT * FROM loghub;
```

If the result shown in the following figure is displayed, the data is uploaded.

2.15.3. Use DataHub to migrate log data to MaxCompute

This topic describes how to use DataHub to migrate log data to MaxCompute.

Prerequisites

The following permissions are granted to the account authorized to access MaxCompute:

- CreateInstance permission on MaxCompute projects

- Permissions to view, modify, and update MaxCompute tables

For more information, see Permissions.

Context

DataHub is a platform that is designed to process streaming data. After data is uploaded to DataHub, the data is stored in a table for real-time processing. DataHub executes scheduled tasks within five minutes to synchronize the data to a MaxCompute table for offline computing.

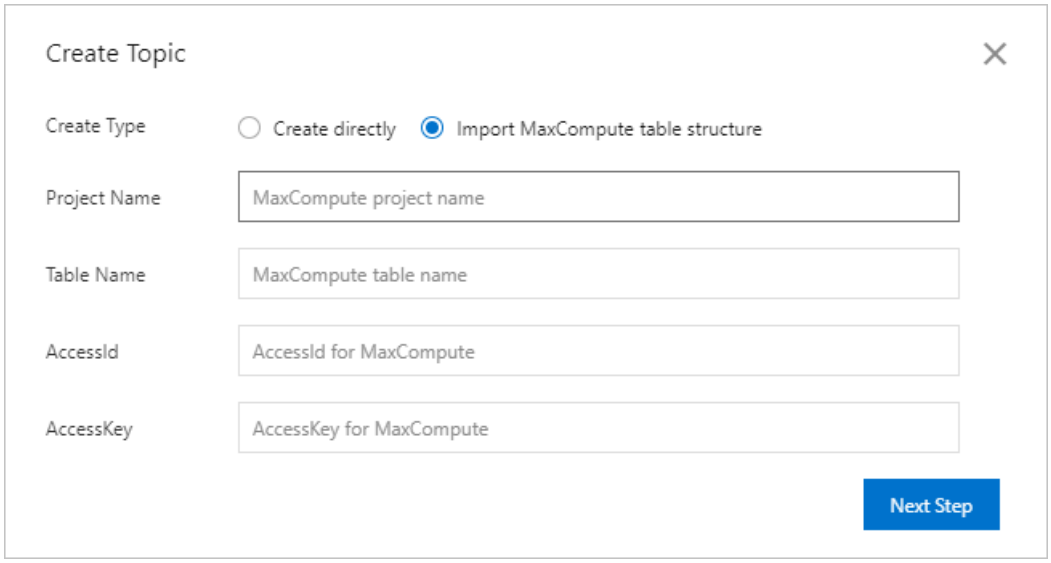
To periodically archive streaming data in DataHub to MaxCompute, you only need to create and configure a DataConnector.

Procedure

1. On the odpscmd client, create a table that is used to store the data synchronized from DataHub. Example:

```
CREATE TABLE test(f1 string, f2 string, f3 double) partitioned by (ds string);
```

2. Create a project in the DataHub console.
   i. Log on to the DataHub console. In the upper-left corner, select a region.
   ii. In the left-side navigation pane, click **Project Manager**.
   iii. In the upper-right corner of the **Project List** page, click **Create Project**.
   iv. In the **Create Project** dialog box, specify **Name** and **Comment**, and click **Create**.
3. Create a topic.
   i. On the **Project List** page, find the project for which you want to create a topic and click **View** in the Operate column.
   ii. In the upper-right corner of the project details page, click **Create Topic**. In the **Create Topic** dialog box, configure the parameters.

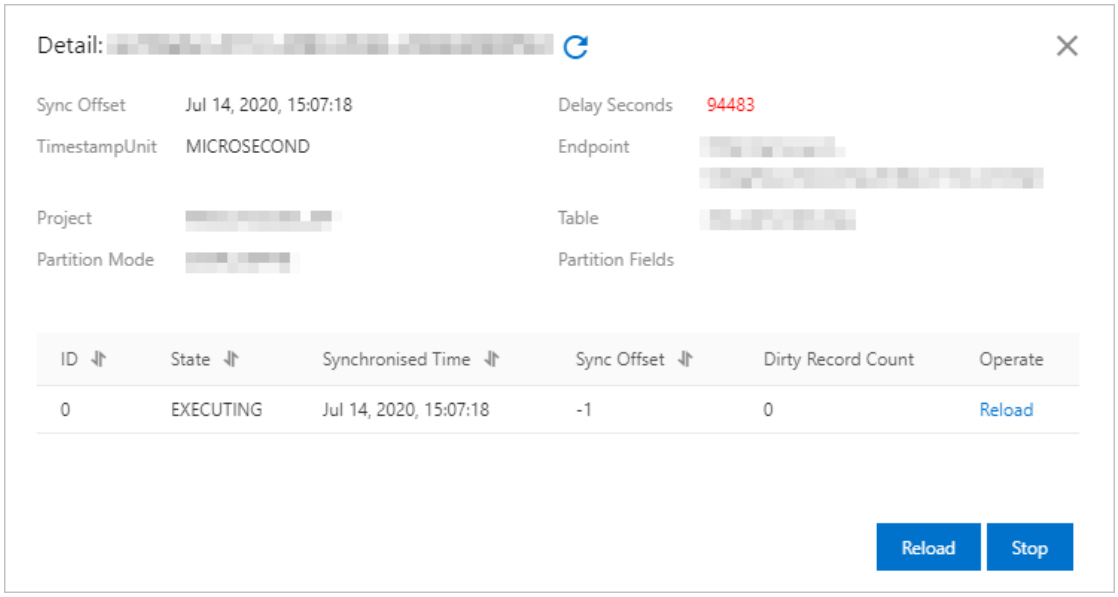


iii. Click **Next Step** to complete topic configurations.

> **Note**
>
> - **Schema** corresponds to a MaxCompute table. The field names, data types, and field sequence specified by Schema must be consistent with those of the MaxCompute table. You can create a DataConnector only if the three conditions are met.
> - You are allowed to migrate the topics of the **TUPLE** and **BLOB** types to MaxCompute tables.
> - A maximum of 20 topics can be created by default. If you require more topics, submit a ticket.
> - The owner of a DataHub topic or the Creator account has the permissions to manage a DataConnector. For example, you can create or delete a DataConnector.

4. Write data to the newly created topic.
   i. Click **View** in the Operate column of the newly created topic.
   ii. On the topic details page, click **Connector**.
   iii. In the **Create connector** dialog box, click **MaxCompute**, configure the parameters, and then click **Create**.
5. View DataConnector details.
   i. In the left-side navigation pane, click **Project Manager**.
   ii. On the **Project List** page, find the project that you want to view its DataConnector details and click **View** in the Operate column.
   iii. On the **Topic List** tab, find the topic of the project and click **View** in the Operate column.
   iv. On the topic details page, click the **Connector** tab to view the created DataConnector.
   v. Click **View** to view DataConnector details.

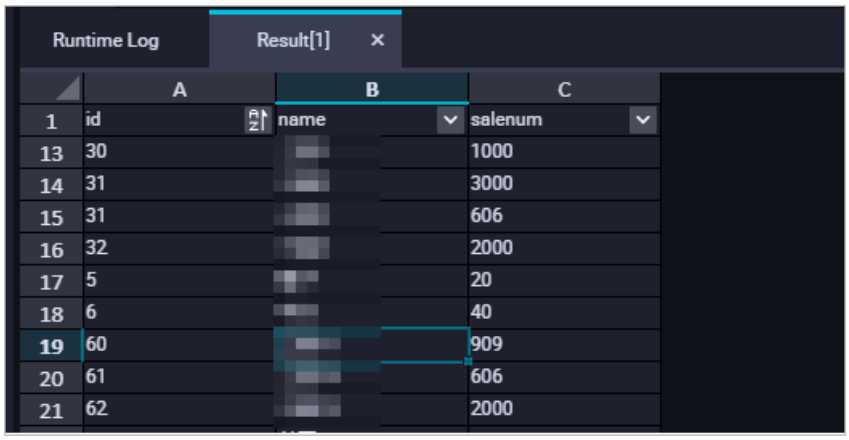
   By default, DataHub migrates data to MaxCompute tables at five-minute intervals or when the amount of data reaches 60 MB. **Sync Offset** indicates the number of migrated data entries.



6. Execute the following statement to check whether the log data is migrated to MaxCompute:

```
SELECT * FROM test;
```

If the result shown in the following figure is displayed, the log data is migrated to MaxCompute.



2.15.4. Use DataWorks Data Integration to migrate log data to MaxCompute

This topic describes how to use DataWorks Data Integration to synchronize LogHub data to MaxCompute.

Context

3.Data development

3.1. Convert data types among STRING, TIMESTAMP, and DATETIME

This topic describes how to convert data types among STRING, TIMESTAMP, or DATETIME. This topic provides multiple date conversion methods that you can use to improve your business efficiency.

Common scenarios of data type conversions for date values:

- STRING to TIMESTAMP
- STRING to DATETIME
- TIMESTAMP to STRING
- TIMESTAMP to DATETIME
- DATETIME to TIMESTAMP
- DATETIME to STRING

STRING to TIMESTAMP

- Scenarios

  Convert a date value of the STRING type to the TIMESTAMP type. The date value of the TIMESTAMP type is in the `yyyy-mm-dd hh:mi:ss.ff3` format.

- Conversion methods

  Use the CAST function.

- Limits

  Date values of the STRING type must be at least accurate to the second and must be specified in the `yyyy-mm-dd hh:mi:ss` format.

- Examples
  - Example 1: Use the CAST function to convert the string `2009-07-01 16:09:00` to the TIMESTAMP type. Sample statement:

    ```
    -- The return value is 2009-07-01 16:09:00.000.
    select cast('2009-07-01 16:09:00' as timestamp);
    ```

  - Example 2: **Incorrect** usage of the CAST function

    ```
    -- The return value is NULL because the input data value is invalid. The date value mus
    t be in the yyyy-mm-dd hh:mi:ss format.
    select cast('2009-07-01' as timestamp);
    ```

STRING to DATETIME

- Scenarios

  Convert a date value of the STRING type to the DATETIME type. The date value of the DATETIME type is in the `yyyy-mm-dd hh:mi:ss` format.

- Conversion methods

- Method 1: Use the CAST function.
- Method 2: Use the TO_DATE function.

- Limits
  - If you use the CAST function, the date value of the STRING type must be specified in the `yyyy-mm-dd hh:mi:ss` format.
  - If you use the TO_DATE function, you must set the value of the format parameter to `yyyy-mm-dd hh:mi:ss`.

- Examples
  - Example 1: Use the CAST function to convert the string `2009-07-01 16:09:00` to the DATETIME type. Sample statement:

```
-- The return value is 2009-07-01 16:09:00.
select cast('2009-07-01 16:09:00' as datetime);
```

  - Example 2: Use the TO_DATE function and specify the format parameter to convert the string `2009-07-01 16:09:00` to the DATETIME type. Sample statement:

```
-- The return value is 2009-07-01 16:09:00.
select to_date('2009-07-01 16:09:00','yyyy-mm-dd hh:mi:ss');
```

  - Example 3: **Incorrect** usage of the CAST function

```
-- The return value is NULL because the input data value is invalid. The date value mus
t be in the yyyy-mm-dd hh:mi:ss format.
select cast('2009-07-01' as datetime);
```

  - Example 4: **Incorrect** usage of the TO_DATE function

```
-- The return value is NULL because the input data value is invalid. The date value mus
t be in the yyyy-mm-dd hh:mi:ss format.
select to_date('2009-07-01','yyyy-mm-dd hh:mi:ss');
```

TIMESTAMP to STRING

- Scenarios

  Convert a date value of the TIMESTAMP type to the STRING type. The value of the TIMESTAMP type is in the `yyyy-mm-dd hh:mi:ss.ff3` format.

- Conversion methods
  - Method 1: Use the CAST function.
  - Method 2: Use the TO_CHAR function. The format of the value after the conversion is specified by the format parameter.

- Examples
  - Example 1: Use the CAST function to convert the TIMESTAMP value `2009-07-01 16:09:00` to the STRING type. To construct data of the TIMESTAMP type, you must use the CAST function twice. Sample statement:

```
-- The return value is 2009-07-01 16:09:00.
select cast(cast('2009-07-01 16:09:00' as timestamp) as string);
```

- Example 2: Use the TO_CHAR function to convert the TIMESTAMP value `2009-07-01 16:09:00` to the STRING type. To construct data of the TIMESTAMP type, you must use the CAST function once. Sample statement:

```
-- The return value is 2009-07-01 16:09:00.
select to_char(cast('2009-07-01 16:09:00' as timestamp),'yyyy-mm-dd hh:mi:ss');
```

TIMESTAMP to DATETIME

- Scenarios

  Convert a date value of the TIMESTAMP type to the DATETIME type. Before the conversion, the date value of the TIMESTAMP type is in the `yyyy-mm-dd hh:mi:ss.ff3` format. After the conversion, the date value of the DATETIME type is in the `yyyy-mm-dd hh:mi:ss` format.

- Conversion methods
  - Method 1: Use the CAST function.
  - Method 2: Use the TO_DATE function.

- Limits

  If you use the TO_DATE function, you must set the value of the format parameter to `yyyy-mm-dd hh:mi:ss`.

- Examples
  - Example 1: Use the CAST function to convert the TIMESTAMP value `2009-07-01 16:09:00` to the DATETIME type. To construct data of the TIMESTAMP type, you must use the CAST function twice. Sample statement:

```
-- The return value is 2009-07-01 16:09:00.
select cast(cast('2009-07-01 16:09:00' as timestamp) as datetime);
```

  - Example 2: Use the TO_DATE function and specify the format parameter to convert the TIMESTAMP value `2009-07-01 16:09:00` to the DATETIME type. To construct data of the TIMESTAMP type, you must use the CAST function once. Sample statement:

```
-- The return value is 2009-07-01 16:09:00.
select to_date(cast('2009-07-01 16:09:00' as timestamp),'yyyy-mm-dd hh:mi:ss');
```

DATETIME to TIMESTAMP

- Scenarios

  Convert a date value of the DATETIME type to the TIMESTAMP type. Before the conversion, the date value of the DATETIME type is in the `yyyy-mm-dd hh:mi:ss` format. After the conversion, the date value of the TIMESTAMP type is in the `yyyy-mm-dd hh:mi:ss.ff3` format.

- Conversion methods

  Use the CAST function.

- Examples

  Use the CAST function to convert a DATETIME value to the TIMESTAMP type. To construct data of the DATETIME type, you must use the GETDATE function once. Sample statement:

```
-- The return value is 2021-10-14 10:21:47.939.
select cast(getdate() as timestamp);
```

DATETIME to STRING

- Scenarios

  Convert a date value of the DATETIME type to the STRING type. The date value of the DATETIME type is in the `yyyy-mm-dd hh:mi:ss` format.

- Conversion methods
  - Method 1: Use the CAST function.
  - Method 2: Use the TO_CHAR function. The format of the value after the conversion is specified by the format parameter.

- Examples
  - Example 1: Use the CAST function to convert a DATETIME value to the STRING type. To construct data of the DATETIME type, you must use the GETDATE function once. Sample statement:

    ```
    -- The return value is 2021-10-14 10:21:47.
    select cast(getdate() as string);
    ```

  - Example 2: Use the TO_CHAR function to convert a DATETIME value to the STRING type in the specified format. To construct data of the DATETIME type, you must use the GETDATE function once. Sample statements:

    ```
    -- The return value is 2021-10-14 10:21:47.
    select to_char (getdate(),'yyyy-mm-dd hh:mi:ss');
    -- The return value is 2021-10-14.
    select to_char (getdate(),'yyyy-mm-dd');
    -- The return value is 2021.
    select to_char (getdate(),'yyyy');
    ```

3.2. Use a MaxCompute UDF to convert IPv4 or IPv6 addresses into geolocations

The development of big data platforms allows you to process multiple types of unstructured and semi-structured data. For example, you can convert IP addresses into geolocations. This topic describes how to use a MaxCompute user-defined function (UDF) to convert IPv4 or IPv6 addresses into geolocations.

Prerequisites

Make sure that the following requirements are met:

- The MaxCompute client is installed.

  For more information about how to install and configure the MaxCompute client, see Install and configure the MaxCompute client.

- MaxCompute Studio is installed and connected to a MaxCompute project. A MaxCompute Java

module is created.

For more information, see Install MaxCompute Studio, Manage project connections, and Create a MaxCompute Java module.

Context

To convert IPv4 or IPv6 addresses into geolocations, you must download the IP address library file that includes the IP addresses, and upload the file to the MaxCompute project as a resource. After you develop and create a MaxCompute UDF based on the IP address library file, you can call the UDF in SQL statements to convert IP addresses into geolocations.

Usage notes

The IP address library file provided in this topic is for reference only. You must maintain the IP address library file based on your business requirements.

Procedure

To convert IPv4 or IPv6 addresses into geolocations by using a MaxCompute UDF, perform the following steps:

1. Step 1: Upload an IP address library file

   Upload an IP address library file as a resource to your MaxCompute project. The resource is used when you create a MaxCompute UDF in subsequent steps.

2. Step 2: Write a MaxCompute UDF

   Write a MaxCompute UDF by using IntelliJ IDEA.

3. Step 3: Create the MaxCompute UDF

   Create the MaxCompute UDF.

4. Step 4: Call the MaxCompute UDF to convert an IP address into a geolocation

   Call the MaxCompute UDF that you created in an SQL statement to convert IP addresses into geolocations.

Step 1: Upload an IP address library file

1. Download an IP address library file to your on-premise machine, decompress the file to obtain the ipv4.txt and ipv6.txt files, and then place the files in the installation directory of the MaxCompute client, `...\odpscmd_public\bin`.

   The IP address library file provided in this topic is for reference only. You must maintain the IP address library file based on your business requirements.

2. Start the MaxCompute client and go to the MaxCompute project to which you want to upload the ipv4.txt and ipv6.txt files.

3. Run the `add file` command to upload the two files as file resources to the MaxCompute project.

   Sample commands:

   ```
   add file ipv4.txt -f;
   add file ipv6.txt -f;
   ```

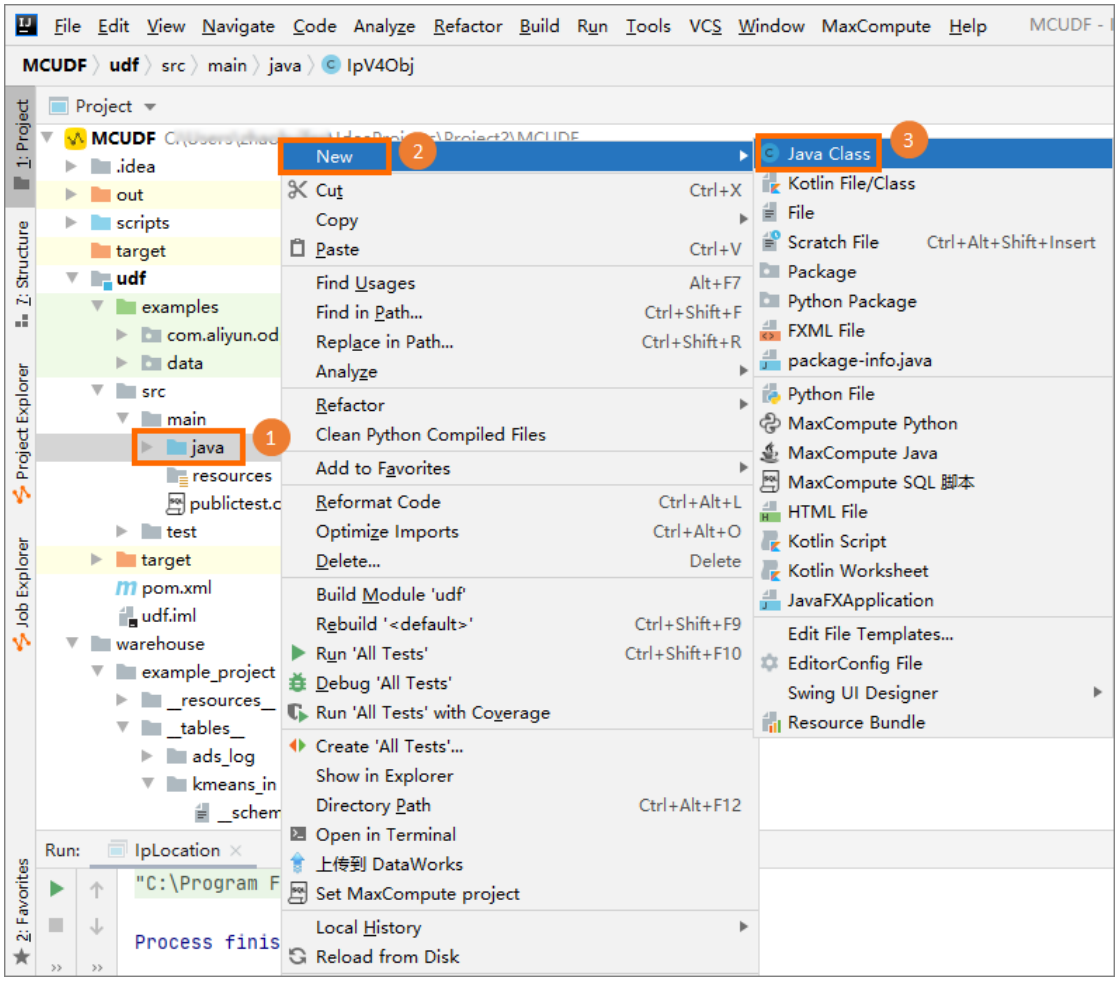
   For more information about how to add resources, see Add resources.

Step 2: Write a MaxCompute UDF

1. Create a Java class.

   The Java class is used for writing a MaxCompute UDF in the next substep.

   i. Start IntelliJ IDEA. In the left-side navigation pane of the **Project** tab, choose **src > main > java**, right-click java, and then choose **New > Java Class**.



   ii. In the **New Java Class** dialog box, enter a class name, press Enter, and then enter the code in the code editor.

   You must create three Java classes. The following sections show the names and code of these classes. You can reuse the code without modification.

   - **IpUtils**

```
package com.aliyun.odps.udf.utils;
import java.math.BigInteger;
import java.net.Inet4Address;
import java.net.Inet6Address;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;
public class IpUtils {
    /**
     * Convert the data type of IP addresses from STRING to LONG.
     *
```

```
     * @param ipInString
     * IP addresses of the STRING type.
     * @return Return the IP addresses of the LONG type.
     */
    public static long StringToLong(String ipInString) {
        ipInString = ipInString.replace(" ", "");
        byte[] bytes;
        if (ipInString.contains(":"))
            bytes = ipv6ToBytes(ipInString);
        else
            bytes = ipv4ToBytes(ipInString);
        BigInteger bigInt = new BigInteger(bytes);
//        System.out.println(bigInt.toString());
        return bigInt.longValue();
    }
    /**
     * Convert the data type of IP addresses from STRING to LONG.
     *
     * @param ipInString
     * IP addresses of the STRING type.
     * @return Return the IP addresses of the STRING type that is converted from
BIGINT.
     */
    public static String StringToBigIntString(String ipInString) {
        ipInString = ipInString.replace(" ", "");
        byte[] bytes;
        if (ipInString.contains(":"))
            bytes = ipv6ToBytes(ipInString);
        else
            bytes = ipv4ToBytes(ipInString);
        BigInteger bigInt = new BigInteger(bytes);
        return bigInt.toString();
    }
    /**
     * Convert the data type of IP addresses from BIGINT to STRING.
     *
     * @param ipInBigInt
     * IP addresses of the BIGINT type.
     * @return Return the IP addresses of the STRING type.
     */
    public static String BigIntToString(BigInteger ipInBigInt) {
        byte[] bytes = ipInBigInt.toByteArray();
        byte[] unsignedBytes = Arrays.copyOfRange(bytes, 1, bytes.length);
        // Remove the sign bit.
        try {
            String ip = InetAddress.getByAddress(unsignedBytes).toString();
            return ip.substring(ip.indexOf('/') + 1).trim();
        } catch (UnknownHostException e) {
            throw new RuntimeException(e);
        }
    }
    /**
     * Convert the data type of IPv6 addresses into signed byte 17.
     */
```

```
    private static byte[] ipv6ToBytes(String ipv6) {
        byte[] ret = new byte[17];
        ret[0] = 0;
        int ib = 16;
        boolean comFlag=false;// IPv4/IPv6 flag.
        if (ipv6.startsWith(":"))// Remove the colon (:) from the start of IPv6 a
ddresses.
            ipv6 = ipv6.substring(1);
        String groups[] = ipv6.split(":");
        for (int ig=groups.length - 1; ig > -1; ig--) {// Reverse scan.
            if (groups[ig].contains(".")) {
                // Both IPv4 and IPv6 addresses exist.
                byte[] temp = ipv4ToBytes(groups[ig]);
                ret[ib--] = temp[4];
                ret[ib--] = temp[3];
                ret[ib--] = temp[2];
                ret[ib--] = temp[1];
                comFlag = true;
            } else if ("".equals(groups[ig])) {
                // Zero-length compression. Calculate the number of missing group
s.
                int zlg = 9 - (groups.length + (comFlag ? 1 : 0));
                while (zlg-- > 0) {// Set these groups to 0.
                    ret[ib--] = 0;
                    ret[ib--] = 0;
                }
            } else {
                int temp = Integer.parseInt(groups[ig], 16);
                ret[ib--] = (byte) temp;
                ret[ib--] = (byte) (temp >> 8);
            }
        }
        return ret;
    }
    /**
     * Convert the data type of IPv4 addresses into signed byte 5.
     */
    private static byte[] ipv4ToBytes(String ipv4) {
        byte[] ret = new byte[5];
        ret[0] = 0;
        // Find the positions of the periods (.) in the IP addresses of the STRIN
G type.
        int position1 = ipv4.indexOf(".");
        int position2 = ipv4.indexOf(".", position1 + 1);
        int position3 = ipv4.indexOf(".", position2 + 1);
        // Convert the IP addresses of the STRING type between periods (.) into I
NTEGER.
        ret[1] = (byte) Integer.parseInt(ipv4.substring(0, position1));
        ret[2] = (byte) Integer.parseInt(ipv4.substring(position1 + 1,
                position2));
        ret[3] = (byte) Integer.parseInt(ipv4.substring(position2 + 1,
                position3));
        ret[4] = (byte) Integer.parseInt(ipv4.substring(position3 + 1));
        return ret;
```

```
    }
    /**
     * @param ipAdress IPv4 or IPv6 addresses of the STRING type.
     * @return 4:IPv4, 6:IPv6, 0: Invalid IP addresses.
     * @throws Exception
     */
    public static int isIpV4OrV6(String ipAdress) throws Exception {
        InetAddress address = InetAddress.getByName(ipAdress);
        if (address instanceof Inet4Address)
            return 4;
        else if (address instanceof Inet6Address)
            return 6;
        return 0;
    }
    /*
     * Check whether the IP address belongs to a specific IP section.
     *
     * ipSection The IP sections that are separated by hyphens (-).
     *
     * The IP address to check.
     */
    public static boolean ipExistsInRange(String ip, String ipSection) {
        ipSection = ipSection.trim();
        ip = ip.trim();
        int idx = ipSection.indexOf('-');
        String beginIP = ipSection.substring(0, idx);
        String endIP = ipSection.substring(idx + 1);
        return getIp2long(beginIP) <= getIp2long(ip)
                && getIp2long(ip) <= getIp2long(endIP);
    }
    public static long getIp2long(String ip) {
        ip = ip.trim();
        String[] ips = ip.split("\\.");
        long ip2long = 0L;
        for (int i = 0; i < 4; ++i) {
            ip2long = ip2long << 8 | Integer.parseInt(ips[i]);
        }
        return ip2long;
    }
    public static long getIp2long2(String ip) {
        ip = ip.trim();
        String[] ips = ip.split("\\.");
        long ip1 = Integer.parseInt(ips[0]);
        long ip2 = Integer.parseInt(ips[1]);
        long ip3 = Integer.parseInt(ips[2]);
        long ip4 = Integer.parseInt(ips[3]);
        long ip2long = 1L * ip1 * 256 * 256 * 256 + ip2 * 256 * 256 + ip3 * 256
                + ip4;
        return ip2long;
    }
    public static void main(String[] args) {
        System.out.println(StringToLong("2002:7af3:f3be:ffff:ffff:ffff:ffff:ffff"
));
        System.out.println(StringToLong("54.38.72.63"));
    }
```

```
    }
    private class Invalid{
        private Invalid()
        {
        }
    }
}
```

- IpV4Obj

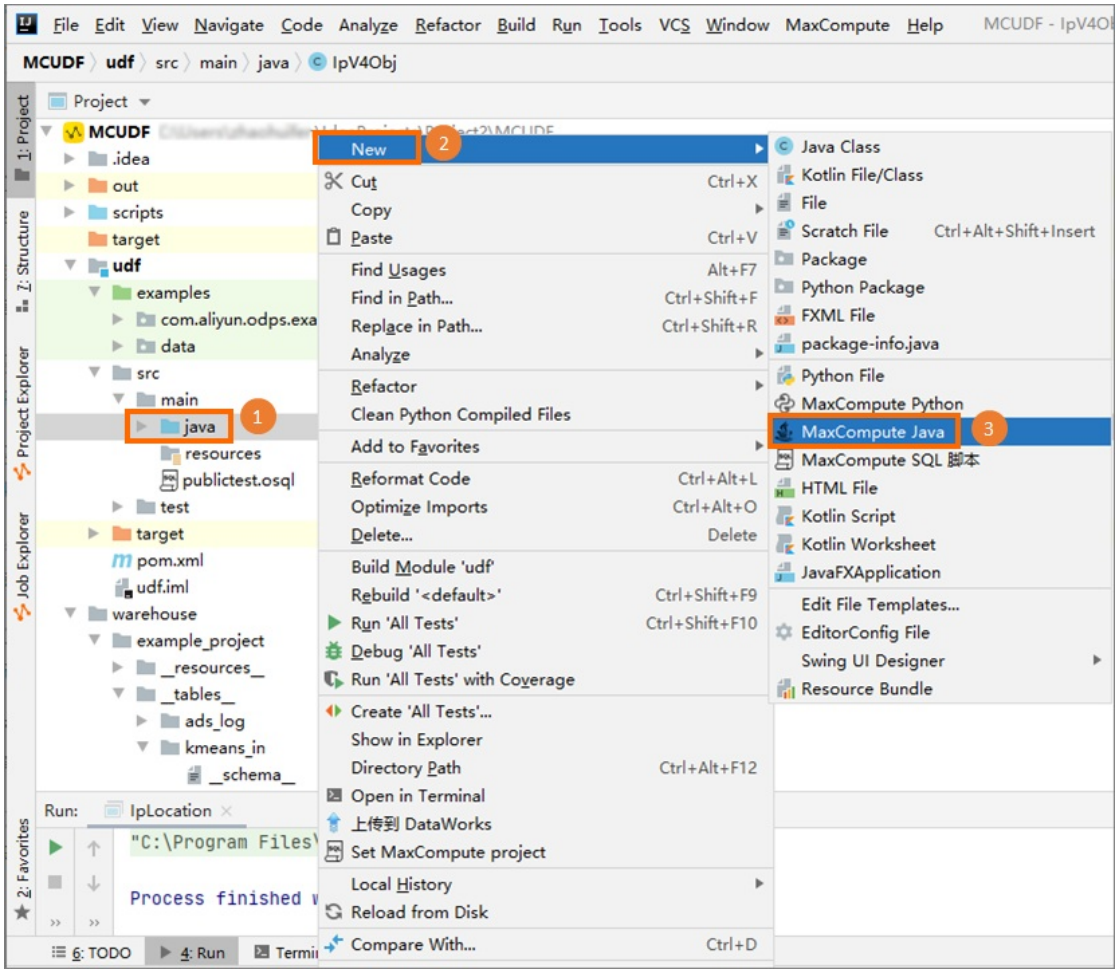
```
package com.aliyun.odps.udf.objects;
public class IpV4Obj {
    public long startIp ;
    public long endIp ;
    public String city;
    public String province;
    public IpV4Obj(long startIp, long endIp, String city, String province) {
        this.startIp = startIp;
        this.endIp = endIp;
        this.city = city;
        this.province = province;
    }
    @Override
    public String toString() {
        return "IpV4Obj{" +
                "startIp=" + startIp +
                ", endIp=" + endIp +
                ", city='" + city + '\'' +
                ", province='" + province + '\'' +
                '}';
    }
    public void setStartIp(long startIp) {
        this.startIp = startIp;
    }
    public void setEndIp(long endIp) {
        this.endIp = endIp;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public void setProvince(String province) {
        this.province = province;
    }
    public long getStartIp() {
        return startIp;
    }
    public long getEndIp() {
        return endIp;
    }
    public String getCity() {
        return city;
    }
    public String getProvince() {
        return province;
    }
}
```

- IpV6Obj

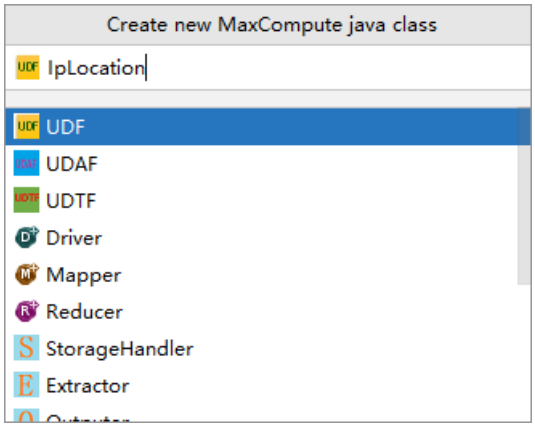
```
package com.aliyun.odps.udf.objects;
public class IpV6Obj {
    public String startIp ;
    public String endIp ;
    public String city;
    public String province;
    public String getStartIp() {
        return startIp;
    }
    @Override
    public String toString() {
        return "IpV6Obj{" +
                "startIp='" + startIp + '\'' +
                ", endIp='" + endIp + '\'' +
                ", city='" + city + '\'' +
                ", province='" + province + '\'' +
                '}';
    }
    public IpV6Obj(String startIp, String endIp, String city, String province) {
        this.startIp = startIp;
        this.endIp = endIp;
        this.city = city;
        this.province = province;
    }
    public void setStartIp(String startIp) {
        this.startIp = startIp;
    }
    public String getEndIp() {
        return endIp;
    }
    public void setEndIp(String endIp) {
        this.endIp = endIp;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getProvince() {
        return province;
    }
    public void setProvince(String province) {
        this.province = province;
    }
}
```

2. Write a MaxCompute UDF.

i. In the left-side navigation pane of the **Project** tab, choose **src > main > java**, right-click java, and then choose **New > MaxCompute Java**.



ii. In the **Create new MaxCompute java class** dialog box, click **UDF** and enter a class name in the **Name** field. Then, press Enter and enter the code in the code editor.



The following code shows how to write a UDF based on a Java class named IpLocation. You can reuse the code without modification.

```
package com.aliyun.odps.udf.udfFunction;
import com.aliyun.odps.udf.ExecutionContext;
import com.aliyun.odps.udf.UDF;
import com.aliyun.odps.udf.UDFException;
```

```
import com.aliyun.odps.udf.utils.IpUtils;
import com.aliyun.odps.udf.objects.IpV4Obj;
import com.aliyun.odps.udf.objects.IpV6Obj;
import java.io.*;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;
public class IpLocation extends UDF {
    public static IpV4Obj[] ipV4ObjsArray;
    public static IpV6Obj[] ipV6ObjsArray;
    public IpLocation() {
        super();
    }
    @Override
    public void setup(ExecutionContext ctx) throws UDFException, IOException {
        //IPV4
        if(ipV4ObjsArray==null)
        {
            BufferedInputStream bufferedInputStream = ctx.readResourceFileAsStream(
"ipv4.txt");
            BufferedReader br = new BufferedReader(new InputStreamReader(bufferedIn
putStream));
            ArrayList<IpV4Obj> ipV4ObjArrayList=new ArrayList<>();
            String line = null;
            while ((line = br.readLine()) != null) {
                String[] f = line.split("\\|", -1);
                if(f.length>=5)
                {
                    long startIp = IpUtils.StringToLong(f[0]);
                    long endIp = IpUtils.StringToLong(f[1]);
                    String city=f[3];
                    String province=f[4];
                    IpV4Obj ipV4Obj = new IpV4Obj(startIp, endIp, city, province);
                    ipV4ObjArrayList.add(ipV4Obj);
                }
            }
            br.close();
            List<IpV4Obj> collect = ipV4ObjArrayList.stream().sorted(Comparator.com
paring(IpV4Obj::getStartIp)).collect(Collectors.toList());
            ArrayList<IpV4Obj> basicIpV4DataList=(ArrayList)collect;
            IpV4Obj[] ipV4Objs = new IpV4Obj[basicIpV4DataList.size()];
            ipV4ObjsArray = basicIpV4DataList.toArray(ipV4Objs);
        }
        //IPV6
        if(ipV6ObjsArray==null)
        {
            BufferedInputStream bufferedInputStream = ctx.readResourceFileAsStream(
"ipv6.txt");
            BufferedReader br = new BufferedReader(new InputStreamReader(bufferedIn
putStream));
            ArrayList<IpV6Obj> ipV6ObjArrayList=new ArrayList<>();
            String line = null;
            while ((line = br.readLine()) != null) {
```

```
                String[] f = line.split("\\|", -1);
                if(f.length>=5)
                {
                    String startIp = IpUtils.StringToBigIntString(f[0]);
                    String endIp = IpUtils.StringToBigIntString(f[1]);
                    String city=f[3];
                    String province=f[4];
                    IpV6Obj ipV6Obj = new IpV6Obj(startIp, endIp, city, province);
                    ipV6ObjArrayList.add(ipV6Obj);
                }
            }
            br.close();
            List<IpV6Obj> collect = ipV6ObjArrayList.stream().sorted(Comparator.com
paring(IpV6Obj::getStartIp)).collect(Collectors.toList());
            ArrayList<IpV6Obj> basicIpV6DataList=(ArrayList)collect;
            IpV6Obj[] ipV6Objs = new IpV6Obj[basicIpV6DataList.size()];
            ipV6ObjsArray = basicIpV6DataList.toArray(ipV6Objs);
        }
    }
    public String evaluate(String ip){
        if(ip==null||ip.trim().isEmpty()||!(ip.contains(".")||ip.contains(":")))
        {
            return null;
        }
        int ipV4OrV6=0;
        try {
            ipV4OrV6= IpUtils.isIpV4OrV6(ip);
        } catch (Exception e) {
            return null;
        }
        // IPv4 addresses are used.
        if(ipV4OrV6==4)
        {
            int i = binarySearch(ipV4ObjsArray, IpUtils.StringToLong(ip));
            if(i>=0)
            {
                IpV4Obj ipV4Obj = ipV4ObjsArray[i];
                return ipV4Obj.city+","+ipV4Obj.province;
            }else{
                return null;
            }
        } else if(ipV4OrV6==6)// IPv6 addresses are used.
        {
            int i = binarySearchIPV6(ipV6ObjsArray, IpUtils.StringToBigIntString(ip
));
            if(i>=0)
            {
                IpV6Obj ipV6Obj = ipV6ObjsArray[i];
                return ipV6Obj.city+","+ipV6Obj.province;
            }else{
                return null;
            }
        } else{// IP addresses are invalid.
            return null;
        }
```

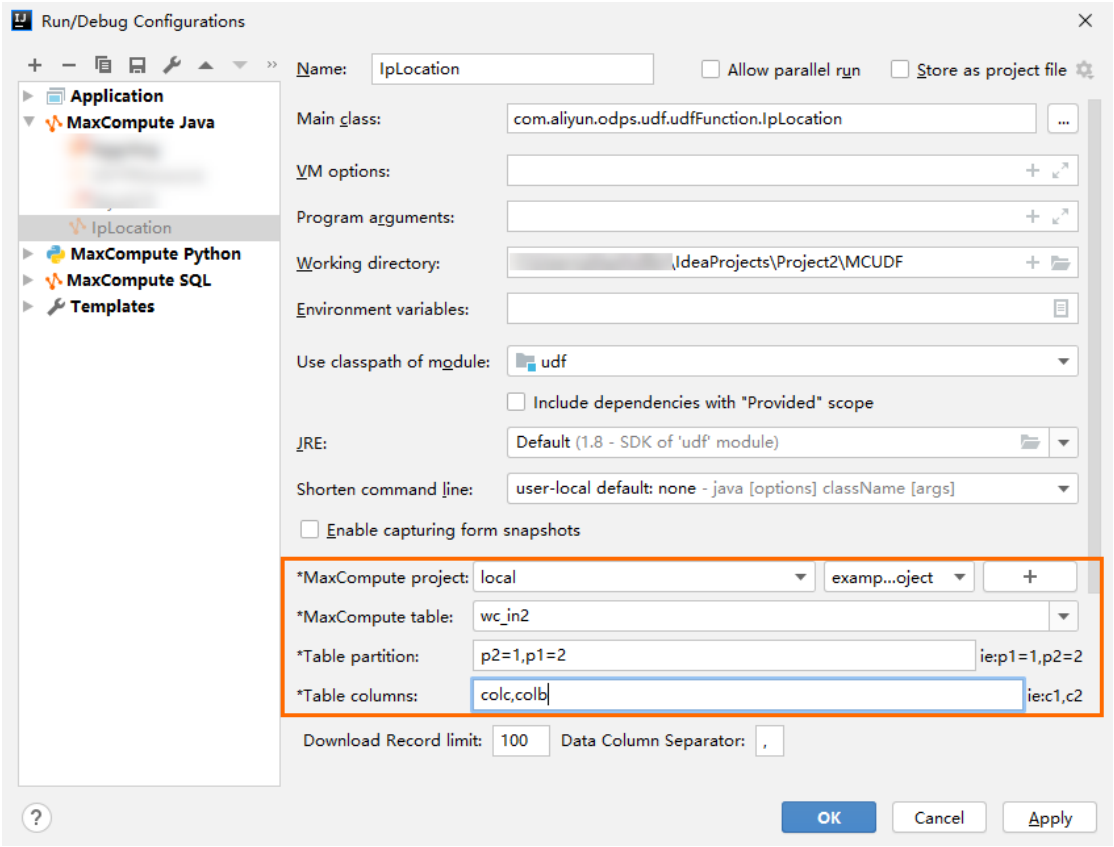
```
    }
    @Override
    public void close() throws UDFException, IOException {
        super.close();
    }
    private static int binarySearch(IpV4Obj[] array,long ip){
        int low=0;
        int hight=array.length-1;
        while (low<=hight)
        {
            int middle=(low+hight)/2;
            if((ip>=array[middle].startIp)&&(ip<=array[middle].endIp))
            {
                return middle;
            }
            if (ip < array[middle].startIp)
                hight = middle - 1;
            else {
                low = middle + 1;
            }
        }
        return -1;
    }
    private static int binarySearchIPV6(IpV6Obj[] array,String ip){
        int low=0;
        int hight=array.length-1;
        while (low<=hight)
        {
            int middle=(low+hight)/2;
            if((ip.compareTo(array[middle].startIp)>=0)&&(ip.compareTo(array[middle
].endIp)<=0))
            {
                return middle;
            }
            if (ip.compareTo(array[middle].startIp) < 0)
                hight = middle - 1;
            else {
                low = middle + 1;
            }
        }
        return -1;
    }
    private class Invalid{
        private Invalid()
        {
        }
    }
}
```

3. Debug the MaxCompute UDF to check whether the code is run as expected.

   For more information about how to debug UDFs, see Perform a local run to debug the UDF.

   i. Right-click the MaxCompute UDF script that you wrote and select **Run**.

ii. In the **Run/Debug Configurations** dialog box, configure the required parameters and click **OK**, as shown in the following figure.

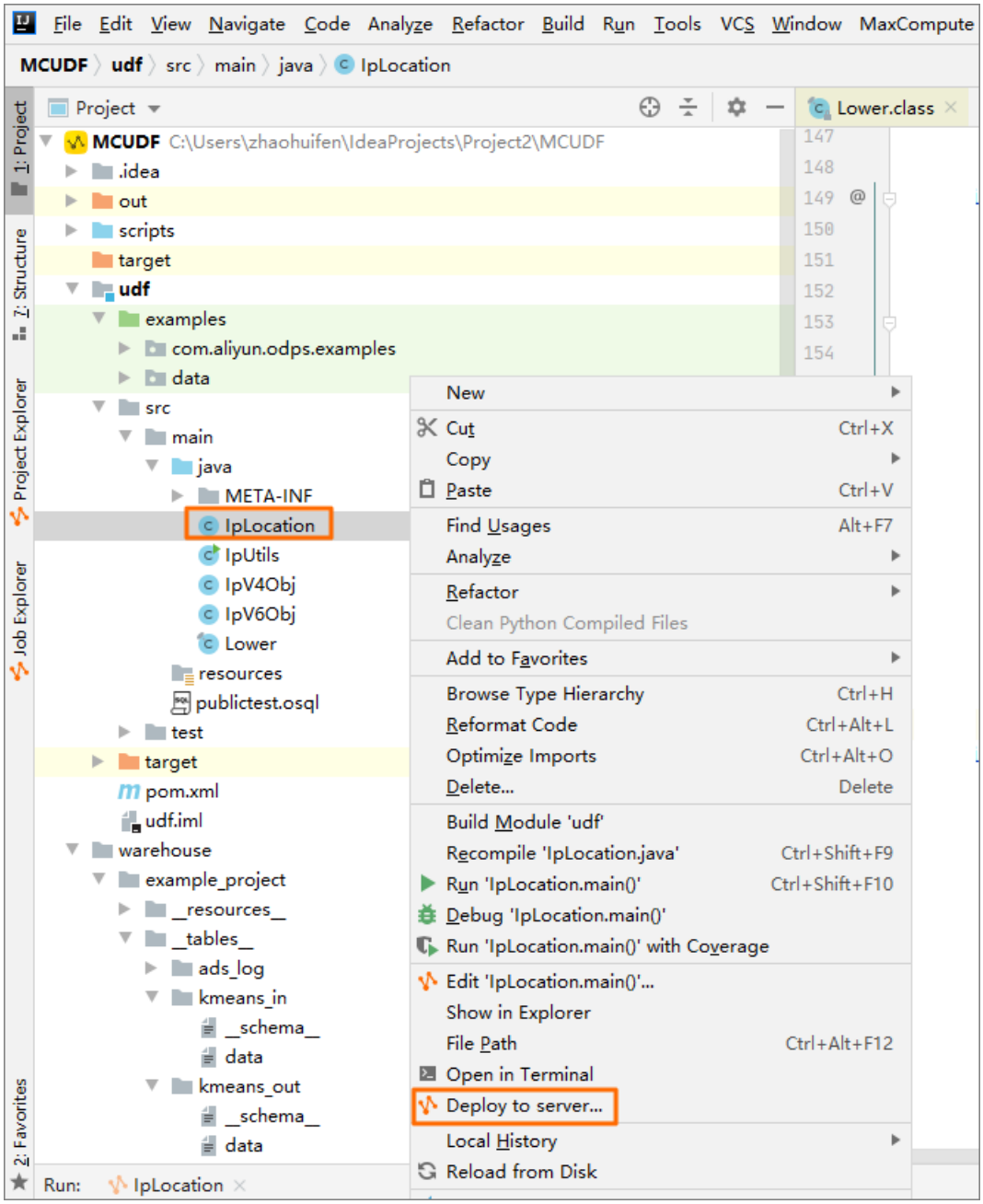


If no error is returned, the code is run successfully. You can proceed with subsequent steps. If an error is reported, you can perform troubleshooting based on the error information displayed on IntelliJ IDEA.

> **Note** The parameter settings in the preceding figure are provided for reference.

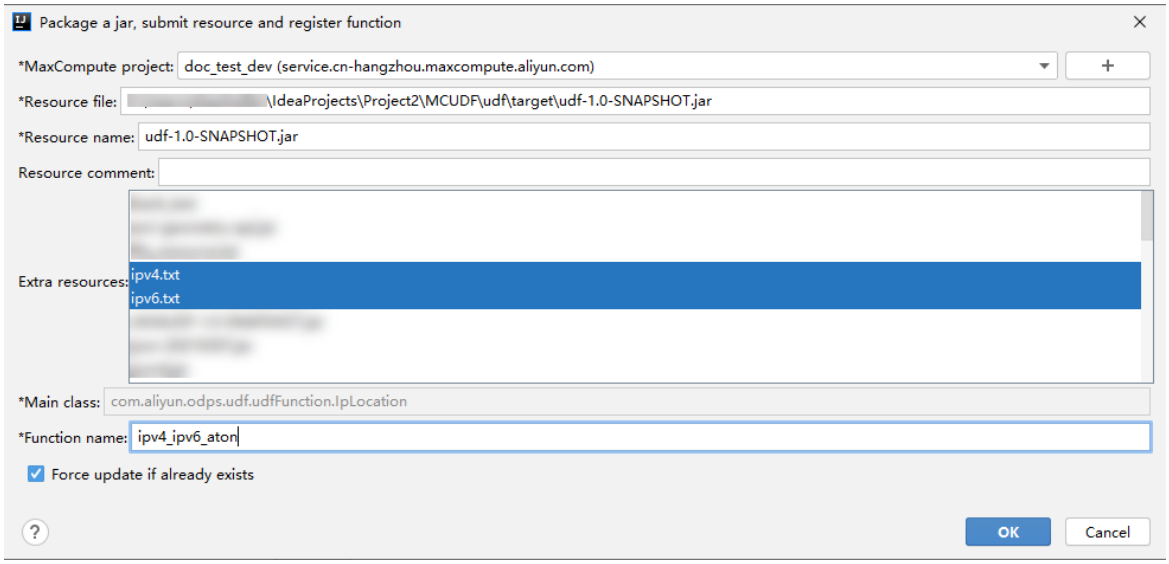
Step 3: Create the MaxCompute UDF

1. Right-click the MaxCompute UDF script that you compiled and select **Deploy to server...**.



2. In the **Package a jar, submit resource and register function** dialog box, configure the parameters.

   For more information about the parameters, see Package a Java program, upload the package, and create a MaxCompute UDF.



Extra resources: You must select the IP address library files ipv4.txt and ipv6.txt that you uploaded in Step 1. In this topic, the created function is named ipv4_ipv6_aton.

Step 4: Call the MaxCompute UDF to convert an IP address into a geolocation

1. Start the MaxCompute client.
2. You can execute an SQL SELECT statement to call the MaxCompute UDF to convert an IPv4 or IPv6 address into a geolocation.

   Sample statements:

   - Convert an IPv4 address into a geolocation

     ```
     select ipv4_ipv6_aton('116.11.34.15');
     ```

     The following result is returned:

     ```
     Beihai, Guangxi Zhuang Autonomous Region
     ```

   - Convert an IPv6 address into a geolocation

     ```
     select ipv4_ipv6_aton('2001:0250:080b:0:0:0:0:0');
     ```

     The following result is returned:

     ```
     Baoding, Hebei Province
     ```

3.3. Use IntelliJ IDEA to develop a Java UDF

IntelliJ IDEA is an integrated development environment (IDE) that is written in Java. IntelliJ IDEA helps you develop Java programs. This topic describes how to use MaxCompute Studio to develop a user-defined function (UDF) that is used to convert uppercase letters into lowercase letters. MaxCompute Studio is a plug-in that is developed based on IntelliJ IDEA.

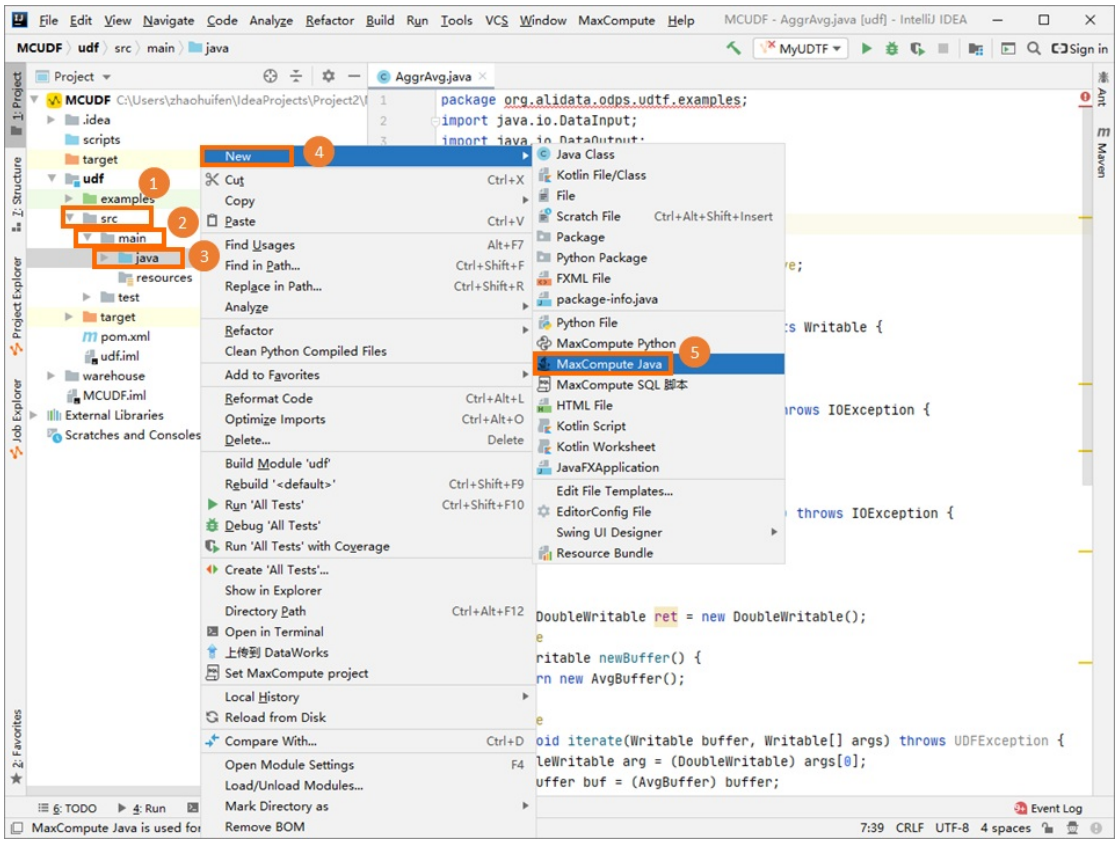
Prerequisites

Make sure that the following operations are performed:

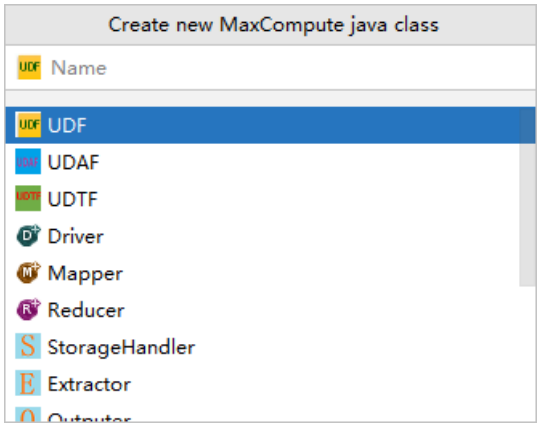
1. Install MaxCompute Studio.
2. Establish a connection to a MaxCompute project.
3. Create a MaxCompute Java module.

Procedure

1. Write a UDF in Java
   i. In the left-side navigation pane of the **Project** tab, choose **src > main > java**, right-click java, and then choose **New > MaxCompute Java**.

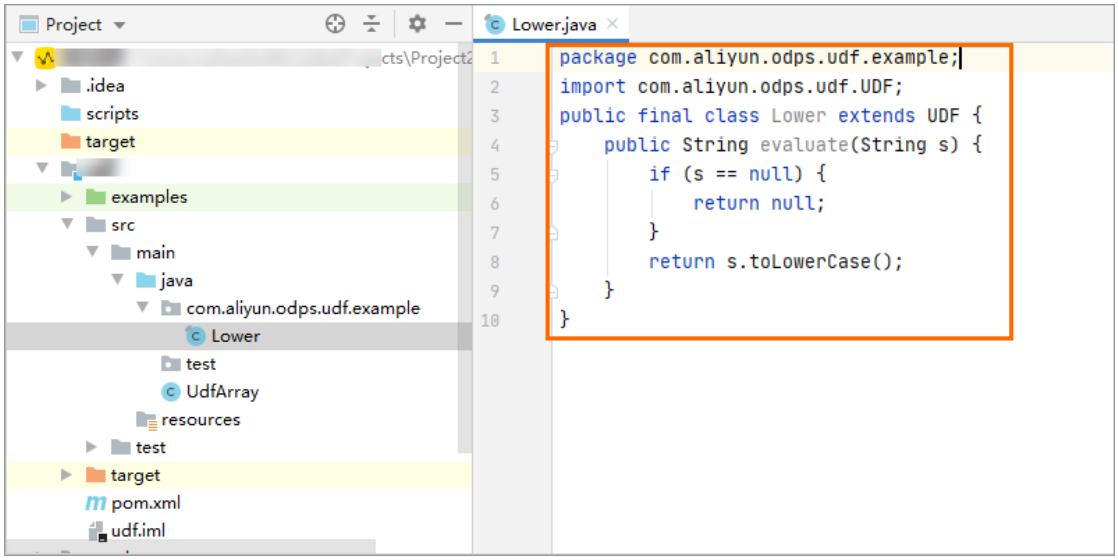


ii. In the **Create new MaxCompute java class** dialog box, click **UDF**, enter a class name in the **Name** field, and then press Enter. In this example, the class is named Lower.



Name: the name of the MaxCompute Java class. If no package is created, enter packagename.classname. The system automatically generates a package.

iii. Write code in the code editor.



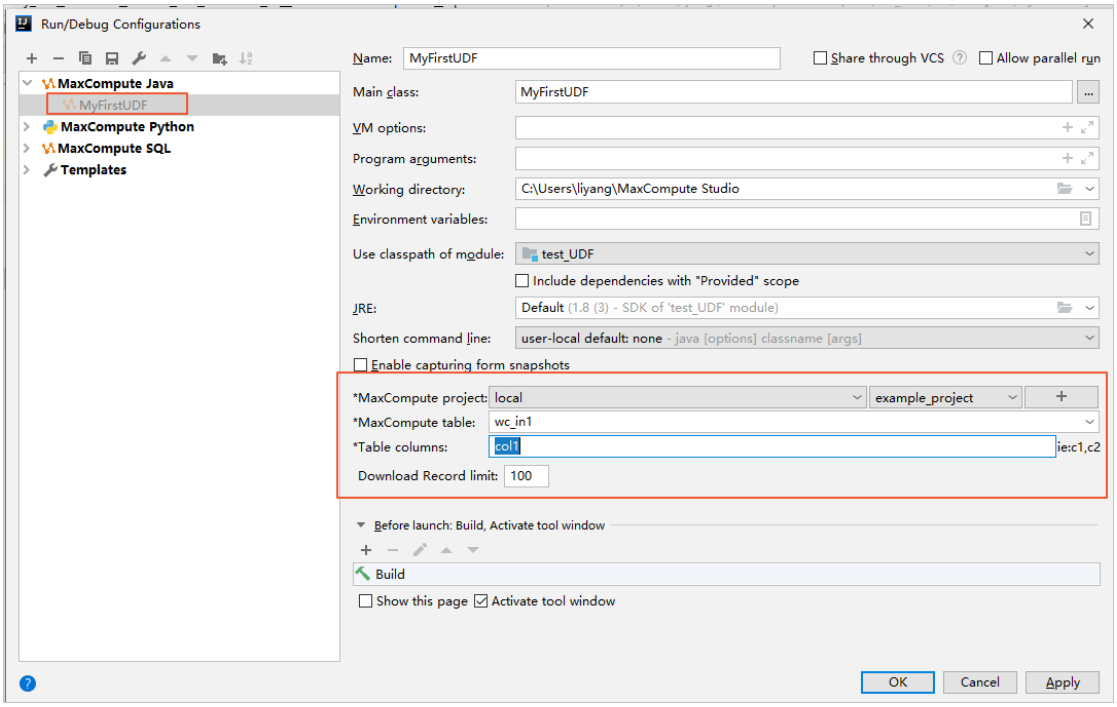
Sample code:

```
package <packagename>;
import com.aliyun.odps.udf.UDF;
public final class Lower extends UDF {
    public String evaluate(String s) {
        if (s == null) {
           return null;
        }
           return s.toLowerCase();
    }
}
```

2. Debug the UDF to check whether the code is run as expected.

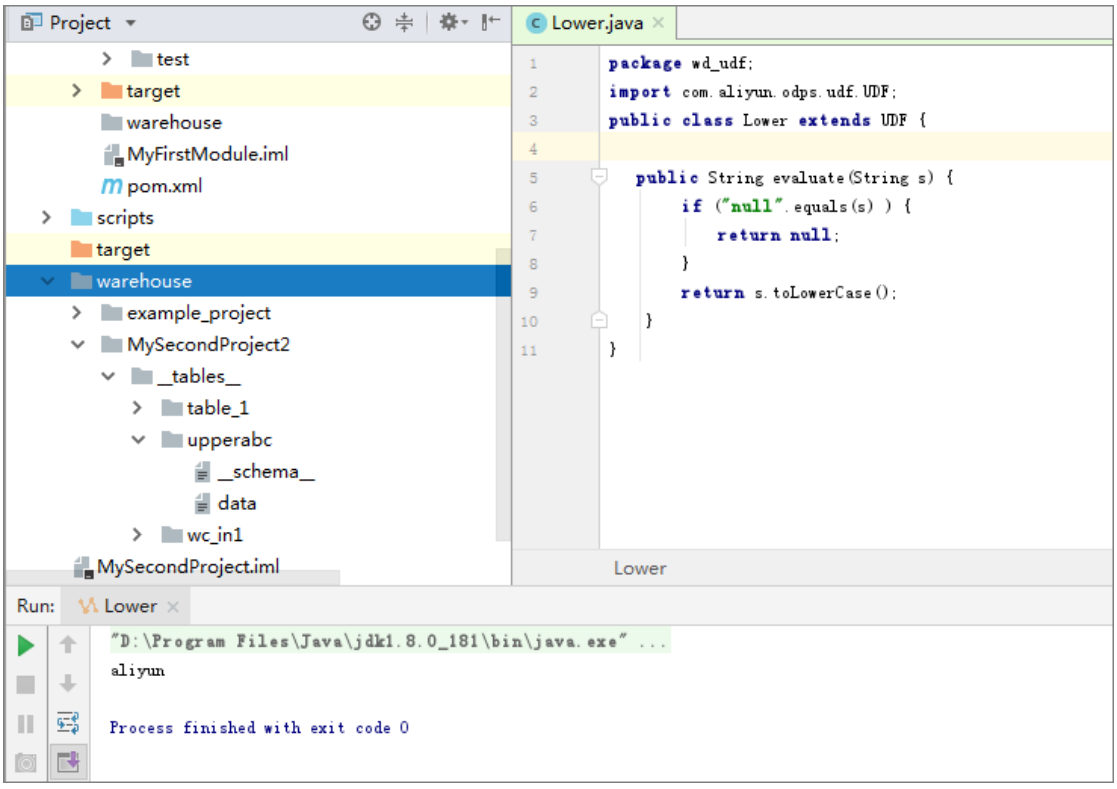
i. In the *java* directory, right-click the Java script that you wrote and select **Run**.

ii. In the **Run/Debug Configurations** dialog box, configure the required parameters.

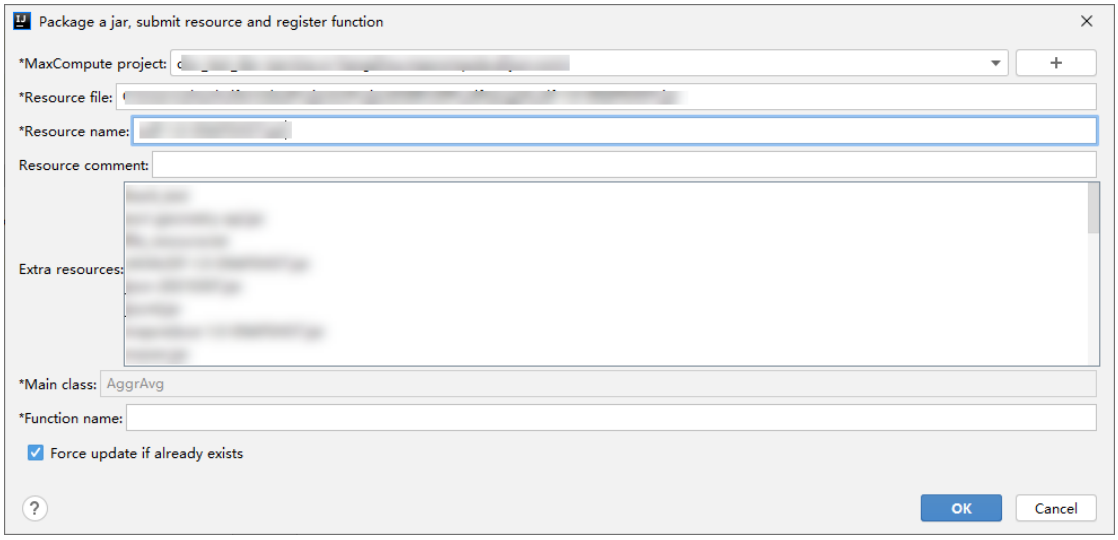


- MaxCompute project: the MaxCompute project in which the UDF runs. To perform a local run, select **local** from the drop-down list.
- MaxCompute table: the name of the MaxCompute table in which the UDF runs.
- Table columns: the columns in the MaxCompute table in which the UDF runs.

iii. Click **OK**. The following figure shows the return result.



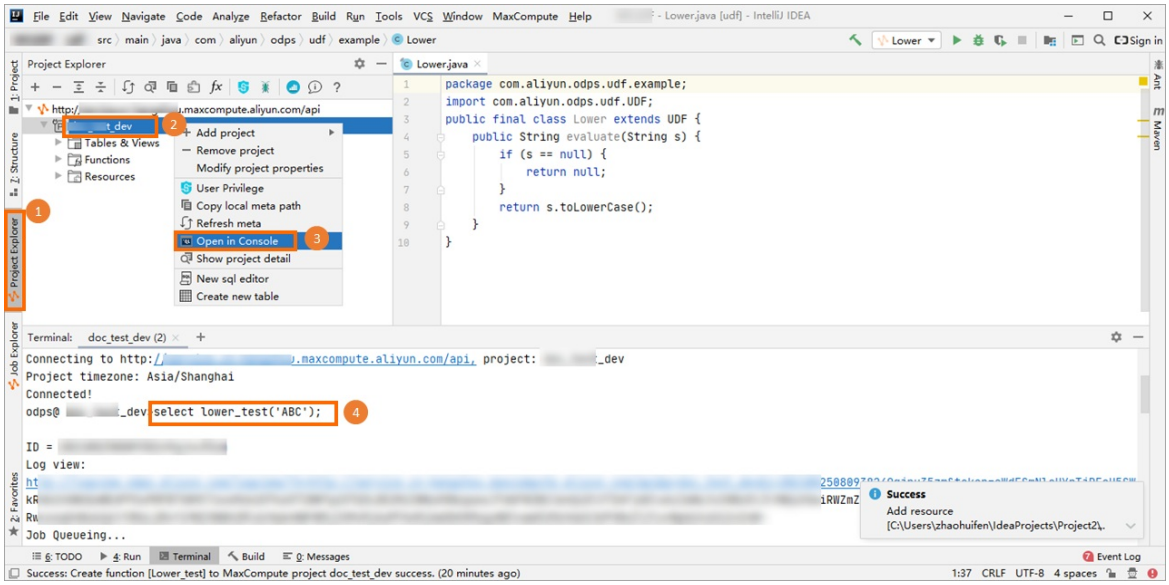
3. Create a MaxCompute UDF.
   i. Right-click the UDF Java file and select **Deploy to server...**.
   ii. In the **Package a jar, submit resource and register function** dialog box, configure the parameters.



   - **MaxCompute project**: the name of the MaxCompute project to which the UDF belongs. Retain the default value, which indicates that the connection to the MaxCompute project is established when you write the UDF.
   - **Resource file**: the path of the resource file on which the UDF depends. Retain the default value.
   - **Resource name**: the name of the resource on which the UDF depends. Retain the default value.
   - **Function name**: the name of the UDF that you want to create. This name is used in the SQL statements that are used to call the UDF. Example: Lower_test.

   iii. Click **OK**.
4. Call the UDF.

   In the left-side navigation pane, click the **Project Explore** tab. Right-click the MaxCompute project to which the UDF belongs, select **Open in Console**, enter the SQL statement that is used to call the UDF, and then press Enter to execute the SQL statement.



Sample statement:

```
select Lower_test('ALIYUN');
```

The following figure shows the result that the preceding statement returns. The result indicates that the Java UDF Lower_test runs as expected.

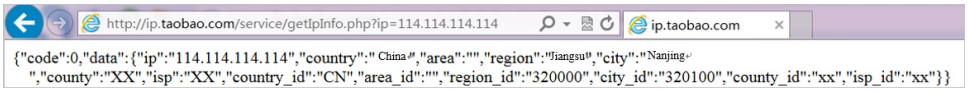
3.4. Use MaxCompute to query geolocations of IP addresses

This topic describes how to query geolocations of IP addresses by using MaxCompute. To query geolocations of IP addresses, you must download an IP address geolocation library, upload the library to MaxCompute, create a user-defined function (UDF), and then execute an SQL statement.

Prerequisites

Context

To query the geolocation of an IP address, you can send an HTTP request to call the API provided by the IP address geolocation library of Taobao. After the API is called, a string that indicates the geolocation of the IP address is returned. The following figure shows an example of a returned string.



You cannot send HTTP requests in MaxCompute. You can query geolocations of IP addresses in MaxCompute by using one of the following methods:

- Execute SQL statements to download data in the IP address geolocation library to your on-premises machine. Then, send HTTP requests to query the geolocation information.

> **Note** This method is inefficient. The query frequency must be less than 10 queries per second (QPS). Otherwise, query requests are rejected by the IP address geolocation library of Taobao.

- Download the IP address geolocation library to your on-premises machine. Then, query the geolocation information in the library.

  > **Note** This method is inefficient and is not suitable for scenarios in which data is analyzed by using data warehouses.

- Maintain an IP address geolocation library and upload it to MaxCompute on a regular basis. Then, query geolocations of IP addresses in the IP address geolocation library.

  > **Note** This method is efficient. You must maintain the IP address geolocation library on a regular basis.

Download an IP address geolocation library

1. Obtain an IP address geolocation library. In this example, the sample IP address geolocation library ipdata.txt.utf8 is used. This IP address geolocation library is a library demo in the UTF-8 format.
2. Download the ipdata.txt.utf8 file. The following figure shows the data in the file.

   ```
   0,16777215,"0.0.0.0","0.255.255","","Intranet IP","Intranet IP","Intranet IP"
   16777216,16777471,"1.0.0.0","1.0.0.255","Australia","","","",""
   16777472,16778239,"1.0.1.0","1.0.3.255","China","Fujian","Fuzhou","Telecom"
   ```

   The following content describes the data in the sample IP address geolocation library.

   - The data is in the UTF-8 format.
   - The first two strings in a data record are the start IP address and the end IP address of an IP address range, in the decimal integer literal format. The third and fourth strings are equivalent to the first two strings, but are expressed in dotted decimal notation. The decimal integer literal format helps you check whether an IP address is within a specific IP address range.

   > **Note** You can also use your own IP address geolocation library.

Upload the IP address geolocation library

1. Execute the following statements on the MaxCompute client to create a table named ipresource. This table is used to store geolocation data of IP addresses.

```
DROP TABLE IF EXISTS ipresource ;
CREATE TABLE IF NOT EXISTS ipresource
(
    start_ip BIGINT
    ,end_ip BIGINT
    ,start_ip_arg string
    ,end_ip_arg string
    ,country STRING
    ,area STRING
    ,city STRING
    ,county STRING
    ,isp STRING
);
```

2. Run the following Tunnel command to upload data in the ipdata.txt.utf8 file to the ipresource table:

```
odps@ workshop_demo>tunnel upload D:/ipdata.txt.utf8 ipresource;
```

In the preceding command, *D:/ipdata.txt.utf8* is the on-premises path of the ipdata.txt.utf8 file. For more information about the command, see Tunnel commands.

You can execute the following statement to check whether the data in the file is uploaded:

```
-- Query the number of data records in the ipresource table.
select count(*) from ipresource;
```

3. Execute the following SQL statement to obtain the first 10 data records in the ipresource table:

```
select * from ipresource limit 10;
```

The following result is returned.

```
Job Queueing...
+------------+------------+--------------+------------+---------+------+--------+--------+-------+
| start_ip   | end_ip     | start_ip_arg | end_ip_arg | country | area | city   | county | isp   |
+------------+------------+--------------+------------+---------+------+--------+--------+-------+
| 3395369026 | 3395369026 | "202.97.56.66" | "202.97.56.66" | "China" | "Hunan"    | "Changsha" | ""  | "Telecom" |
| 3395369027 | 3395369028 | "202.97.56.67" | "202.97.56.68" | "China" | "Heilongjiang" | ""     | ""  | "Telecom" |
| 3395369029 | 3395369029 | "202.97.56.69" | "202.97.56.69" | "China" | "Anhui"    | "Hefei"    | ""  | "Telecom" |
| 3395369030 | 3395369030 | "202.97.56.70" | "202.97.56.70" | "China" | "Hunan"    | "Changsha" | ""  | "Telecom" |
| 3395369031 | 3395369033 | "202.97.56.71" | "202.97.56.73" | "China" | "Heilongjiang" | ""     | ""  | "Telecom" |
| 3395369034 | 3395369034 | "202.97.56.74" | "202.97.56.74" | "China" | "Hunan"    | "Changsha" | ""  | "Telecom" |
| 3395369035 | 3395369036 | "202.97.56.75" | "202.97.56.76" | "China" | "Heilongjiang" | ""     | ""  | "Telecom" |
| 3395369037 | 3395369037 | "202.97.56.77" | "202.97.56.77" | "China" | "Jiangsu"  | "Nanjing"  | ""  | "Telecom" |
| 3395369038 | 3395369038 | "202.97.56.78" | "202.97.56.78" | "China" | "Hunan"    | "Changsha" | ""  | "Telecom" |
| 3395369039 | 3395369040 | "202.97.56.79" | "202.97.56.80" | "China" | "Heilongjiang" | ""     | ""  | "Telecom" |
+------------+------------+--------------+------------+---------+------+--------+--------+-------+
```

Create a UDF

1.

2. Create a Python resource.

   i.

   ii. In the **Create Resource** dialog box, enter a **resource name**, select **Upload to MaxCompute**, and then click **Create**.

iii. Enter the following code in the Python resource and click the icon.

```
from odps.udf import annotate
@annotate("string->bigint")
class ipint(object):
    def evaluate(self, ip):
        try:
            return reduce(lambda x, y: (x << 8) + y, map(int, ip.split('.')))
        except:
            return 0
```

iv. Click the icon.

3. Create a function.

i. Right-click the workflow that you created and choose **Create > MaxCompute > Function**.

ii. In the **Create Function** dialog box, enter a **function name**, and click **Create**.

> **Note** If multiple MaxCompute engines are bound to the workspace, select one of the engines from the **Engine Instance** drop-down list.

iii. On the configuration tab of the function, configure the parameters.

Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Operation Functions**, **Aggregate Functions**, **String Processing Functions**, **Date Functions**, **Window Functions**, and **Other Functions**.
Engine Instance MaxCompute	A default value is displayed and cannot be changed.
Function Name	The name of the function. You can use the function name to reference the function in SQL statements. The function name must be globally unique and cannot be changed after the function is created.
Owner	The value of this parameter is automatically displayed.

Parameter	Description
Class Name	Required. The name of the class that implements the function. **Note** If the resource type is Python, enter the class name in the Python resource name.Class name format. Do not include the .py extension in the resource name.
Resources	Required. The list of resources. You can search for existing resources in the workspace in fuzzy match mode. Separate multiple resources with commas (,).
Description	The description of the function.
Expression Syntax	The syntax of the function. Example: `test`.
Parameter Description	The description of the supported data types of input and output parameters.
Return Value	Optional. The value to return. Example: 1.
Example	Optional. The example of the function.

4. Click the icon in the toolbar.
5. Commit the function.
   i. Click the icon in the toolbar to commit the function.
   ii. In the **Commit Node** dialog box, enter your comments in the **Change description** field.
   iii. Click **OK**.

Use the UDF that you created in an SQL statement to query the geolocation of an IP address

1.
2.
3. On the configuration tab of the ODPS SQL node, enter the following statement:

```
select * from ipresource
WHERE ipint('1.2.24.2') >= start_ip
AND ipint('1.2.24.2') <= end_ip
```

4.
5.

3.5. Resolve the issue that you cannot upload files that exceed 10 MB to DataWorks

The JAR package or resource file that is used in a MapReduce job often exceeds 10 MB. However, the maximum size of a file that you can upload to DataWorks is 10 MB. This topic describes how to resolve this issue to schedule a MapReduce job where a JAR package or resource file that exceeds 10 MB is used.

Prerequisites

The MaxCompute client is installed. For more information, see Install and configure the MaxCompute client.

Procedure

1. Run the following command on the MaxCompute client to upload a JAR package that exceeds 10 MB:

```
--Upload a JAR package.
add jar C:\test_mr\test_mr.jar -f;
```

2. Resources that you upload on the MaxCompute client are not displayed on the DataStudio page of the DataWorks console. You must run the following command to check whether the resource is uploaded:

```
--View resources.
list resources;
```

3. Reduce the size of the JAR package. DataWorks runs a MapReduce job on the computer where the MaxCompute client resides. Therefore, you can submit only the Main function to DataWorks to run a MapReduce job.

```
jar
-resources test_mr.jar,test_ab.jar --A file can be referenced after it is registered on
the MaxCompute client.
-classpath test_mr.jar --Reduce the size of a JAR package by using the following method
: Submit only the Mapper and Reducer that contain the Main function on the gateway. You
do not need to submit third-party dependencies. You can store the resources in the wc_i
n directory of the MaxCompute client.
```

3.6. Grant a specified user the access permissions on a specific UDF

This topic describes how to grant specified users the access permissions on specific resources, such as tables and user-defined functions (UDFs). This operation involves data encryption and decryption algorithms and is related to data security.

Prerequisites

The MaxCompute client is installed. For more information, see Install and configure the MaxCompute

client.

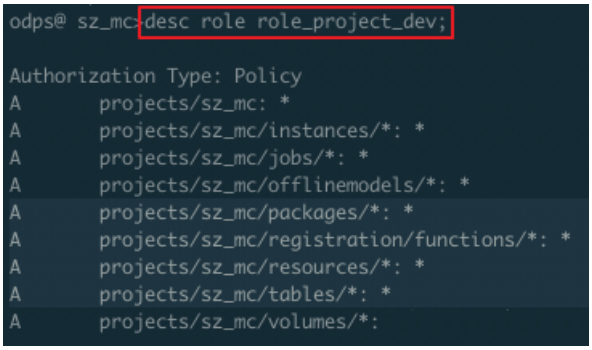
Context

You can use one of the following methods to manage the access permissions of users:

- Use packages to achieve fine-grained access control.

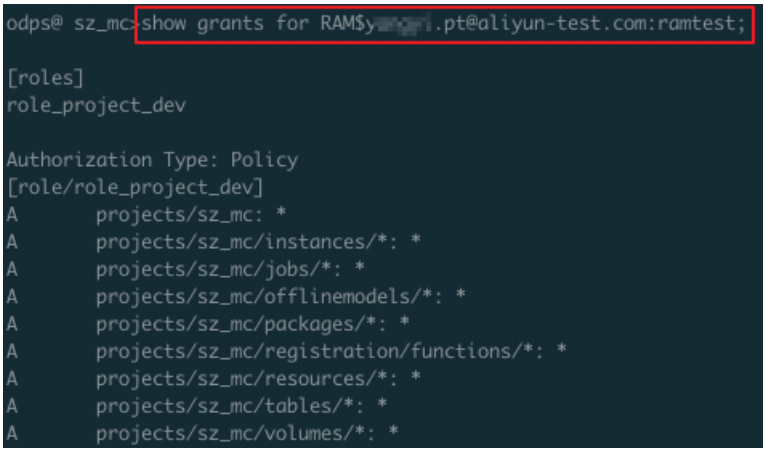
  This method is used for data sharing and resource authorization across projects. After you assign the developer role to a user by using a package, the user has full permissions on all objects in the package. This may cause uncontrollable risks. For more information, see Cross-project resource access based on packages.

  - The following figure shows the permissions of the developer role that is defined in DataWorks.

    By default, the developer role has full permissions on all packages, functions, resources, and tables in a workspace. This does not meet the requirements for permission management.

  - The following figure shows the permissions that are granted to a RAM user that is assigned the developer role in DataWorks.

  You cannot grant a specified user the access permissions on a specific UDF by using package-based authorization or by assigning the developer role in DataWorks to the user. For example, if you assign the developer role to the RAM user named `RAM$xxxxx.pt@aliyun-test.com:ramtest`, the RAM user has full permissions on all objects in the current workspace. For more information, see Authorize users.

- Create a role in the DataWorks console to manage access permissions.

  On the **MaxCompute Management** page in the DataWorks console, you can manage the access permissions of custom user roles. On this page, you can grant permissions on a table or a project. You cannot grant permissions on resources or UDFs.

> **Note** For more information about MaxCompute projects for DataWorks workspaces, see Configure MaxCompute.

- Use a role policy and a project policy.

  Role and project policies allow you to grant a specified user the permissions on specific resources.

  > **Note** To ensure security, we recommend that you verify role and project policies in a test workspace.

You can use a role policy and a project policy to grant access permissions on a specific UDF to a specified user.

- To prevent a user from accessing a specific resource in a workspace, assign the developer role to the user in the DataWorks console and configure a role policy for the user to deny access requests for the resource on the MaxCompute client.
- To allow a user to access a specific resource, assign the developer role to the user in the DataWorks console and configure a project policy for the user to allow access requests for the resource on the MaxCompute client.

Procedure

1. Create a role that has no permission to access a UDF named getregion by default.
   i. On the MaxCompute client, run the following command to create a role named denyudfrole:

   ```
   create role denyudfrole;
   ```

   ii. Create a role policy file that contains the following content:

   ```
   {
   "Version": "1", "Statement"
   [{
   "Effect":"Deny",
   "Action":["odps:Read","odps:List"],
   "Resource":"acs:odps:*:projects/sz_mc/resources/getaddr.jar"
   },
   {
   "Effect":"Deny",
   "Action":["odps:Read","odps:List"],
   "Resource":"acs:odps:*:projects/sz_mc/registration/functions/getregion"
   }
    ] }
   ```

   iii. Configure the storage path for the role policy file.

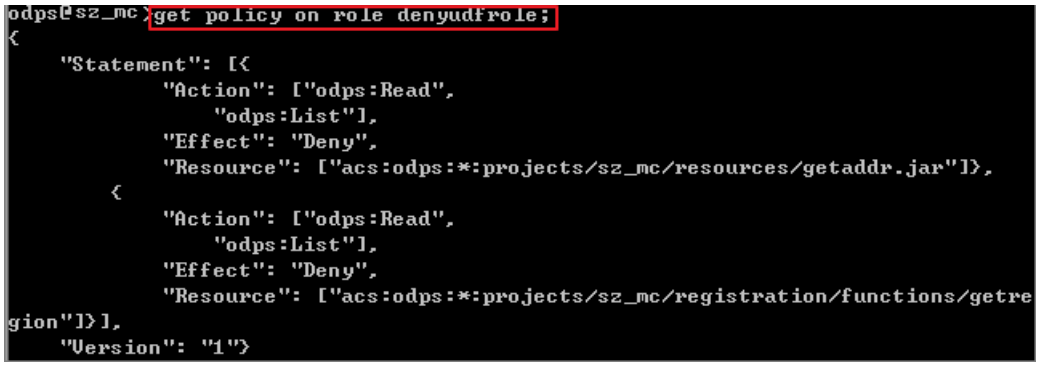
   On the MaxCompute client, run the following command:

   ```
   put policy /Users/yangyi/Desktop/role_policy.json on role denyudfrole;
   ```

iv. On the MaxCompute client, run the following command to view the role policy:

```
get policy on role denyudfrole;
```

The following result is returned:

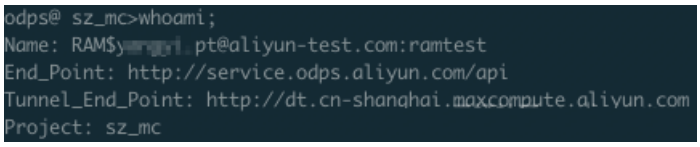


v. On the MaxCompute client, run the following command to assign the denyudfrole role to a RAM user:

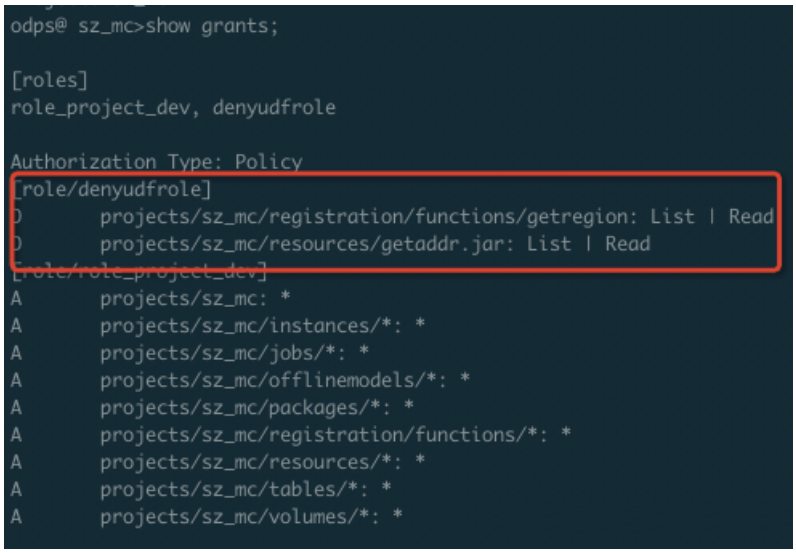
```
grant denyudfrole to RAM$xxxx.pt@aliyun-test.com:ramtest;
```

2. Check whether the denyudfrole role is created.

i. Log on to the MaxCompute client as the RAM user to which the denyudfrole role is assigned. Then, run the `whoami;` command to check the current logon user.

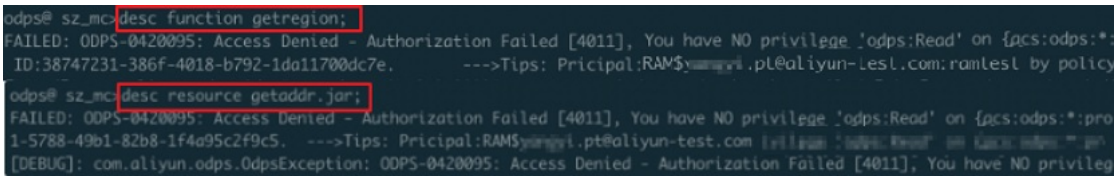


ii. Run the `show grants;` command to check the permissions of the current logon user.



The result indicates that the RAM user has the following two roles: role_project_dev and denyudfrole. role_project_dev is the default developer role in DataWorks.

iii. Check the permissions of the RAM user on the getregion UDF and its dependencies.



The result indicates that the RAM user with the developer role in DataWorks does not have read permissions on the getregion UDF. You can perform the next step to configure a project policy to ensure that only a specified RAM user can access the UDF.

3. Configure a project policy.

i. Create a project policy file that contains the following content:

```
{
"Version": "1", "Statement":
[{
"Effect":"Allow",
"Principal":"RAM$yangyi.pt@aliyun-test.com:yangyitest",
"Action":["odps:Read","odps:List","odps:Select"],
"Resource":"acs:odps:*:projects/sz_mc/resources/getaddr.jar"
},
{
"Effect":"Allow",
 "Principal":"RAM$xxxx.pt@aliyun-test.com:yangyitest",
"Action":["odps:Read","odps:List","odps:Select"],
"Resource":"acs:odps:*:projects/sz_mc/registration/functions/getregion"
}] }
```

ii. Configure the storage path for the project policy file.

On the MaxCompute client, run the following command:

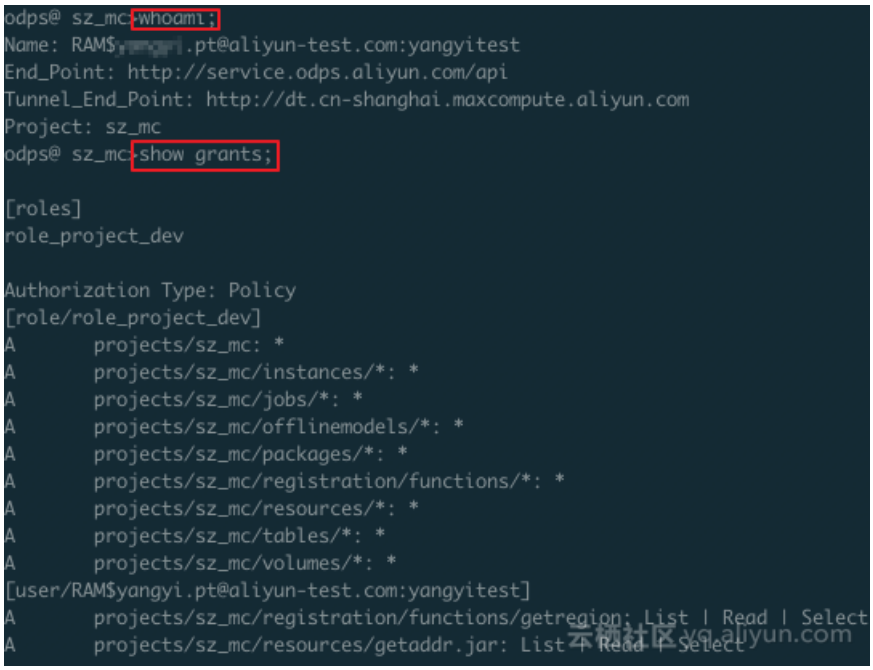
```
put policy /Users/yangyi/Desktop/project_policy.json;
```

iii. On the MaxCompute client, run the following command to view the project policy:

```
get policy;
```

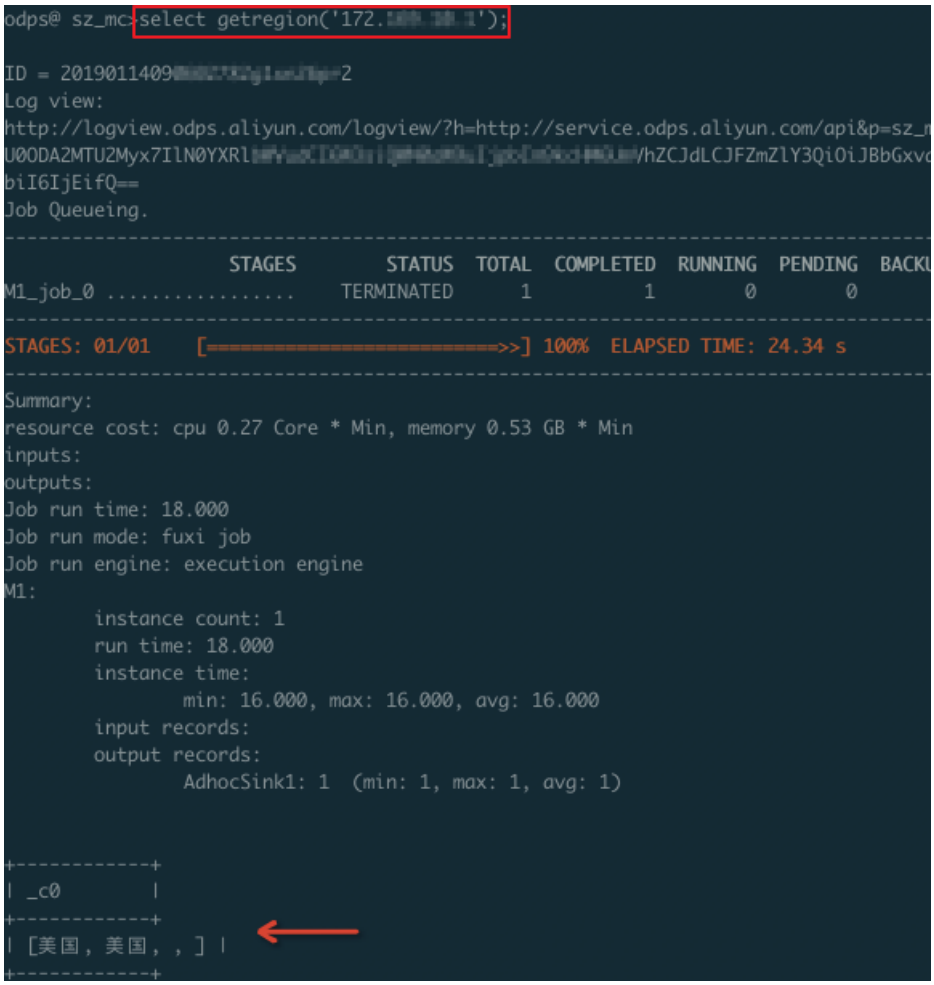
The following result is returned:

iv. Run the `whoami;` command to check the current logon user. Then, run the `show grants;` command to check the permissions of the user.



v. Run an SQL job to check whether only the specified RAM user can access the specific UDF and its dependencies.

- The following result indicates that the specified RAM user can access the specific UDF:



- The following result indicates that the specified RAM user can access the dependencies of the UDF:

3.7. Use a PyODPS node to segment Chinese text based on Jieba

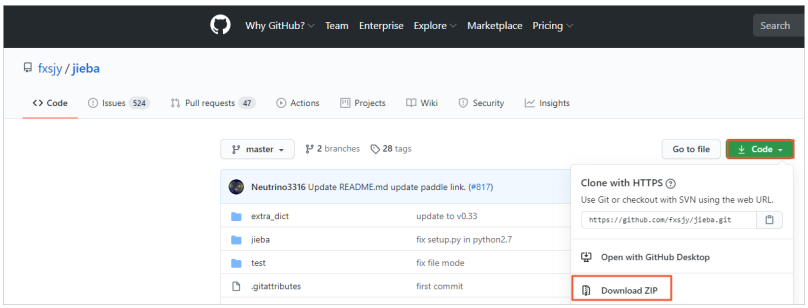
This topic describes how to use a PyODPS node in DataWorks to segment Chinese text based on the open source segmentation tool Jieba and write the segmented words and phrases to a new table. This topic also describes how to use closure functions to segment Chinese text based on a custom dictionary.

Prerequisites

- A DataWorks workspace is created. In this example, a workspace in **basic mode** is used. The workspace is associated with multiple MaxCompute compute engines. For more information, see

Create a workspace.

- The open source Jieba package is downloaded from GitHub.



Context

PyODPS nodes are integrated with MaxCompute SDK for Python. You can directly edit Python code and use MaxCompute SDK for Python on PyODPS nodes of DataWorks. For more information about PyODPS nodes, see Create a PyODPS 2 node.

This topic describes how to use a PyODPS node to segment Chinese text based on Jieba.

- Use open source packages to segment Chinese text based on Jieba
- Use custom dictionaries to segment Chinese text based on Jieba

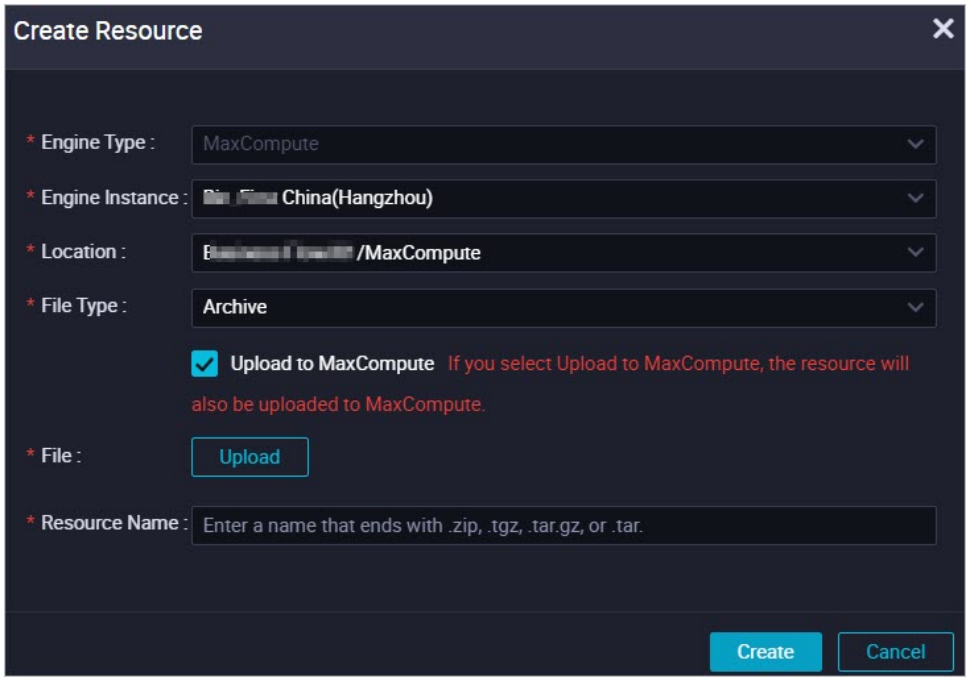
> **Notice** Sample code in this topic is for reference only. We recommend that you do not use the code in your production environment.

Use open source packages to segment Chinese text based on Jieba

1. Create a workflow.
   i. Log on to the DataWorks console.
   ii. In the left-side navigation pane, click **Workspaces**.
   iii. Select the region where the workspace resides, find the workspace, and then click **Data Development** in the Actions column.
   iv. Move the pointer over the icon and click **Workflow**.
   v. In the **Create Workflow** dialog box, specify the **Workflow Name** and **Description** parameters. Then, click **Create**.

   > **Notice** The workflow name must be 1 to 128 characters in length, and can contain letters, digits, underscores (_), and periods (.).

2. Upload the jieba-master.zip package.
   i. Click the workflow that you created, expand **MaxCompute**, right-click **Resource**, and then choose **Create > Archive**.
   ii. In the **Create Resource** dialog box, configure the parameters and click **Create**. The following table describes the parameters.



Parameter	Description
Engine Type	Select the compute engine where the resource resides from the drop-down list. **Note** If only one instance is bound to your workspace, this parameter is not displayed.
Engine Instance	The name of the MaxCompute engine to which the task is bound.
Location	The folder that is used to store the resource. The default value is the path of the current folder. You can modify the path based on your business requirements.
File Type	Select **Archive** from the File Type drop-down list. **Note** If the resource package has been uploaded to the MaxCompute client, clear **Upload to MaxCompute**. Otherwise, an error occurs during the upload process.
File	Click **Upload**, select the downloaded file named jieb-master.zip from your on-premises machine, and then click **Open**.

Parameter	Description
Resource Name	The name of the resource. The resource name can be different from the name of the file that you uploaded but must comply with the following conventions: ■ The resource name can contain only letters, digits, periods (.), underscores (_), and hyphens (-). ■ If you select **Archive** from the File Type drop-down list, the extension of the resource name must be the same as that of the file name. The extension can be .zip, .tgz, .tar.gz, or .tar.

iii. In the toolbar, click the icon.

iv. In the **Commit** dialog box, specify the **Change description** parameter and click **Commit**.

3. Create a table that is used to store test data.

i. Click the workflow that you created, expand **MaxCompute**, right-click **Table**, and then select **Create Table**.

ii. In the **Create Table** dialog box, specify the **Table Name** parameter and click **Create**.

> **Note** In this example, the table name is **jieba_test**.

iii. Click **DDL Statement** and enter the following DDL statement to create a table:

```
CREATE TABLE jieba_test (
    'chinese' string,
    `content` string
);
```

> **Note** The table in this example contains two columns. You can segment text in one column during data development.

iv. In the Confirm message, click **OK**.

v. In the **General** section, specify the **Display Name** parameter for the table. Click **Commit to Production Environment**.

vi. In the **Commit to Production Environment** dialog box, select **I am aware of the risk and confirm the commissions** and click **OK**.

4. Use the same method to create a table named **jieba_result**. This table is used to store the test result. Sample DDL statement:

```
CREATE TABLE jieba_result (
    `chinese` string
);
```

> **Note** In this example, only the text in the chinese column of the test data is segmented. Therefore, the result table contains only one column.

5. Click Test Data to download the test data.

6. Upload test data.
   i. Click the icon on the **DataStudio** page.
   ii. In the **Data Import Wizard** dialog box, enter the name of the test table jieba_test to which you want to import data, select the table, and then click **Next**.
   iii. Click **Browse**, upload the *jieba_test.csv* file from your on-premises machine, and then click **Next**.
   iv. Select **By Name** and click **Import Data**.
7. Create a PyODPS 2 node.
   i. Click the workflow, expand **MaxCompute**, right-click **Data Analytics**, and then choose **Create > PyODPS 2**.
   ii. In the **Create Node** dialog box, specify the **Node Name** and **Location** parameters and click **Commit**.

   > **Note**
   > - The node name must be 1 to 128 characters in length, and can contain letters, digits, underscores (_), and periods (.).
   > - In this example, the **Node Name** parameter is set to word_split.

   iii. On the configuration tab of the node, enter the following PyODPS code:

```
def test(input_var):
    import jieba
    import sys
    reload(sys)
    sys.setdefaultencoding('utf-8')
    result=jieba.cut(input_var, cut_all=False)
    return "/ ".join(result)
hints = {
    'odps.isolation.session.enable': True
}
libraries =['jieba-master.zip']  # Reference the jieba-master.zip package.
iris = o.get_table('jieba_test').to_df()  # Reference the data in the jieba_test ta
ble.
example = iris.chinese.map(test).execute(hints=hints, libraries=libraries)
print(example)  # Display the text segmentation result. The result is of the MAP ty
pe.
abci=list(example)   # Convert the text segmentation result into the LIST type.
i = 0
for i in range(i,len(abci)):
    pq=str(abci[i])
    o.write_table('jieba_result',[pq])  # Write the data records to the jieba_resul
t table one by one.
    i+=1
else:
    print("done")
```

   iv. Click the icon in the toolbar to save the code.

v. Click the icon in the toolbar. In the **Parameters** dialog box, select a resource group from the **Resource Group** drop-down list and click **OK**.

> **Note** For more information about **resource groups for scheduling**, see Overview.

vi. View the execution result of the Jieba segmentation program on the **Runtime Log** tab in the lower part of the page.

8. Create and run an ODPS SQL node.

i. Click the workflow, expand **MaxCompute**, right-click **Data Analytics**, and then choose **Create > ODPS SQL**.

ii. In the **Create Node** dialog box, specify the **Node Name** and **Location** parameters and click **Commit**.

> **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.).

iii. On the configuration tab of the node, enter the following SQL statement:

```
select * from jieba_result;
```

iv. Click the icon in the toolbar to save the code.

v. Click the icon in the toolbar. In the **Parameters** dialog box, select a resource group from the **Resource Group** drop-down list and click **OK**.

> **Note** For more information about **resource groups for scheduling**, see Overview.

vi. In the **Expense Estimate** dialog box, check the **estimated cost** and click **Run**.

vii. View the execution result on the **Runtime Log** tab in the lower part of the page.

Use custom dictionaries to segment Chinese text based on Jieba

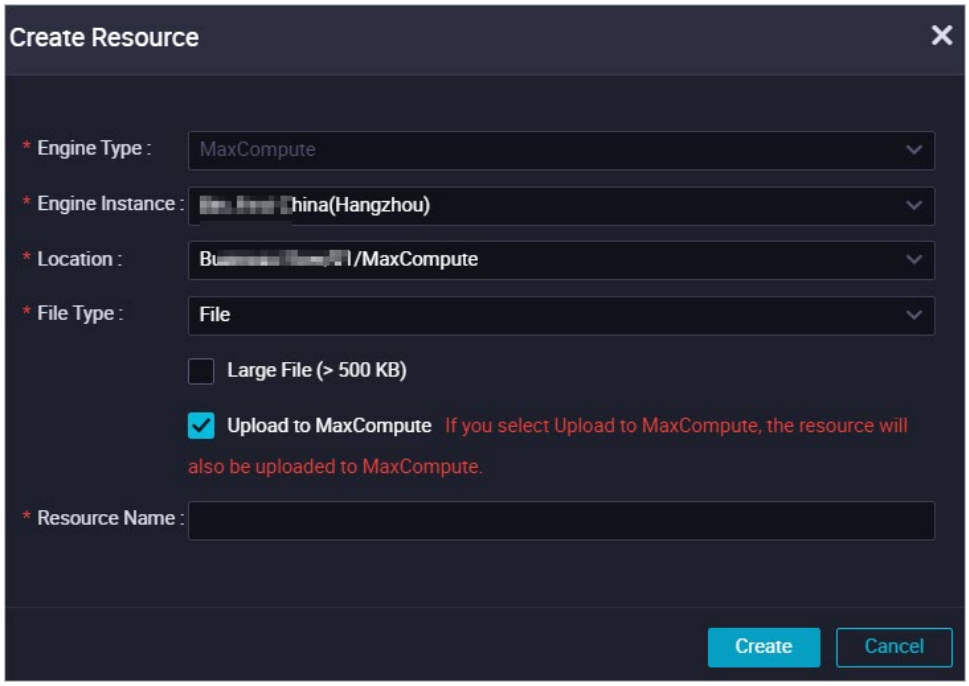
If the dictionary of the Jieba tool does not meet your requirements, you can use a custom dictionary.

You can use a PyODPS user-defined function (UDF) to read table or file resources that are uploaded to MaxCompute. In this case, you must write the UDF as a closure function or a callable class. If you need to reference complex UDFs, you can create a MaxCompute function in DataWorks. For more information, see Register a MaxCompute function.

In this topic, a closure function is used to reference the custom dictionary file key_words.txt that is uploaded to MaxCompute.

> **Note** In this example, the custom dictionary file name is key_words.txt.

1. Click the workflow, expand **MaxCompute**, right-click **Resource**, and then choose **Create > File**.
2. In the **Create Resource** dialog box, configure the parameters and click **Create**. The following table describes the parameters.



Parameter	Description
Engine Type	Select the compute engine where the resource resides from the drop-down list. **Note** If only one instance is bound to your workspace, this parameter is not displayed.
Engine Instance	The name of the MaxCompute engine to which the task is bound.
Location	The folder that is used to store the resource. The default value is the path of the current folder. You can modify the path based on your business requirements.
File Type	Select **File** from the File Type drop-down list. **Note** If you want to upload a dictionary file from the on-premises machine to DataWorks, the file must be encoded in UTF-8.
Upload	Click **Upload**, select the key_words.txt file from your on-premises machine, and then click **Open**.
Resource Name	The name of the resource. The resource name can contain only letters, digits, periods (.), underscores (_), and hyphens (-).

3. On the configuration tab of the key_words.txt resource, enter the content of the custom dictionary. Dictionary format:
   - Each word occupies one line.
   - Each line contains the following parts in sequence: word, frequency, and parts of speech. The

frequency and part of speech are optional. Separate every two parts with a space. The order of the three parts cannot be adjusted.

4. Click the icon in the toolbar to commit the resource.

5. Create a **PyODPS 2** node and enter the following code:

```
def test(resources):
    import jieba
    import sys
    reload(sys)
    sys.setdefaultencoding('utf-8')
    fileobj = resources[0]
    def h(input_var):# Use the nested function h() to load the dictionary and segment t
ext.
        import jieba
        jieba.load_userdict(fileobj)
        result=jieba.cut(input_var, cut_all=False)
        return "/ ".join(result)
    return h
hints = {
    'odps.isolation.session.enable': True
}
libraries =['jieba-master.zip']  # Reference the jieba-master.zip package.
iris = o.get_table('jieba_test').to_df()  # Reference the data in the jieba_test table.

file_object = o.get_resource('key_words.txt') # Use the get_resource() function to refe
rence the MaxCompute resource.
example = iris.chinese.map(test, resources=[file_object]).execute(hints=hints, librarie
s=libraries) # Call the map function to transfer the resources parameter.
print(example)  # Display the text segmentation result. The result is of the MAP type.
abci=list(example)   # Convert the text segmentation result into the List type.
for i in range(i,len(abci)):
    pq=str(abci[i])
    o.write_table('jieba_result',[pq])  # Write the data records to the jieba_result ta
ble one by one.
    i+=1
else:
    print("done")
```

6. Run the code and compare the results before and after the custom dictionary is referenced.

3.8. Use a PyODPS node to download data to a local directory for processing or to process data online

This topic describes how to use a PyODPS node to download data to a local directory for processing or to process data online.

Background information

PyODPS provides multiple methods to download data to a local directory. You can download data to a local directory for processing and then upload the data to MaxCompute. However, local data processing is inefficient because the massively parallel processing capability of MaxCompute cannot be used if you download data to a local directory. If the data volume is greater than 10 MB, we recommend that you do not download data to a local directory for processing. You can use one of the following methods to download data to a local directory:

- Use the head, tail, or to_pandas method. In most cases, use the `head` or `tail` method to obtain small volumes of data. If you want to obtain large volumes of data, use the persist method to store data in a MaxCompute table. For more information, see Execution.
- Use the open_reader method. You can execute open_reader on a table or an SQL instance to obtain the data. If you need to process large volumes of data, we recommend that you use PyODPS DataFrame or MaxCompute SQL. A PyODPS DataFrame object is created based on a MaxCompute table. This method provides higher efficiency than local data processing.

Sample code

Convert a JSON string to multiple rows. Each row consists of a key and its value.

- For local testing, use the `head` method to obtain small volumes of data

```
In [12]: df.head(2)
                json
0  {"a": 1, "b": 2}
1  {"c": 4, "b": 3}
In [14]: from odps.df import output
In [16]: @output(['k', 'v'], ['string', 'int'])
    ...: def h(row):
    ...:     import json
    ...:     for k, v in json.loads(row.json).items():
    ...:         yield k, v
    ...:
In [21]: df.apply(h, axis=1).head(4)
   k  v
0  a  1
1  b  2
2  c  4
3  b  3
```

- For online production, use the `persist` method to store large volumes of data in a MaxCompute table

```
In [14]: from odps.df import output
In [16]: @output(['k', 'v'], ['string', 'int'])
    ...: def h(row):
    ...:     import json
    ...:     for k, v in json.loads(row.json).items():
    ...:         yield k, v
    ...:
In [21]: df.apply(h, axis=1).persist('my_table')
```

4.Compute optimization

4.1. Optimize SQL statements

This topic describes common scenarios where you can optimize SQL statements to achieve better performance.

Reduce impacts of data skew

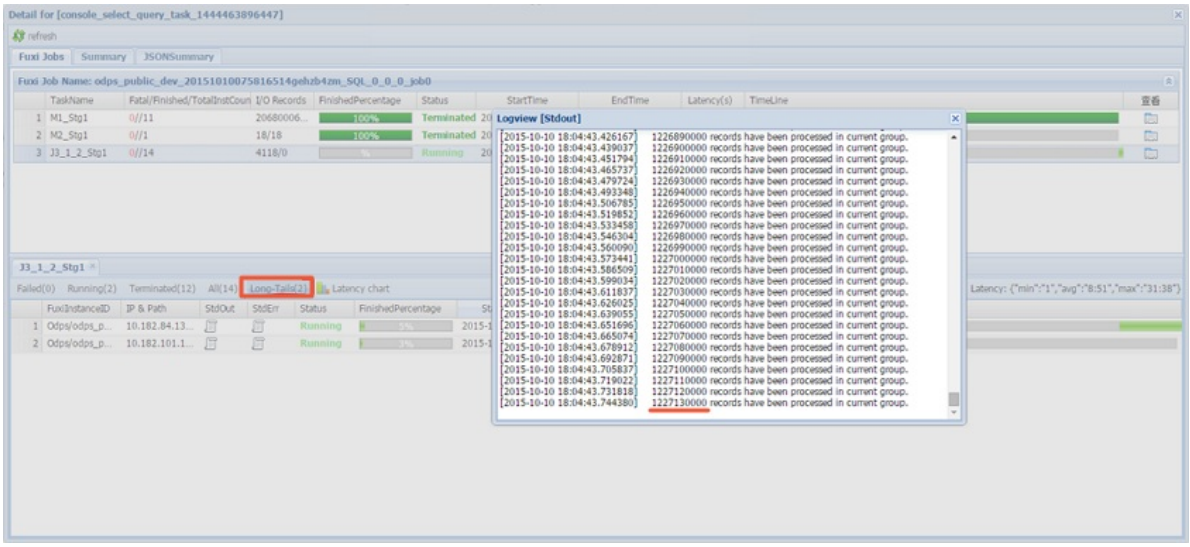
Data skew can lead to an extreme imbalance of work. When data is skewed, some workers need to process larger amounts of data than the others. As a result, these workers take much longer to complete. This prolongs the overall time used to process data and may lead to latency.

- **Skewed joins**

  An imbalance of work may occur when you join tables based on a key that is not evenly distributed. For example, execute the following statement to join a large table named A and a small table named B:

  ```
  select * from A join B on A.value= B.value;
  ```

  Copy the Logview URL of the query and open it in a browser to go to the Logview page. Double-click the Job Scheduler job that performs the JOIN operation. On the Long-tails tab, you can see that long tails exist, as shown in the following figure. This indicates that data is skewed.

  To optimize the preceding statement, you can use one of the following methods:

  - Use a MAPJOIN statement. Table B is a small table which does not exceed 512 MB in size. In this case, you can replace the preceding statement with the following statement:

    ```
    select /*+ MAPJOIN(B) */ * from A join B on A.value= B.value;
    ```

- Handle the skewed key separately. If data skew occurs because a large number of null key values exist in both tables, you must filter out these null values or generate random numbers to replace them before you perform the JOIN operation. For example, you can replace the preceding statement with the following statement:

  ```
  select * from A join B
  on case when A.value is null then concat('value',rand() ) else A.value end = B.value;
  ```

  The following example describes how to identify the key values that cause data skew:

  ```
  -- Data skew leads to an imbalance of work when the following statement is executed:
  select * from a join b on a.key=b.key;
  -- Execute the following statement to view the distribution of key values and identify
  the key values that cause data skew:
  select left.key, left.cnt * right.cnt from
  (select key, count(*) as cnt from a group by key) left
  join
  (select key, count(*) as cnt from b group by key) right
  on left.key=right.key;
  ```

- **Skewed GROUP BY operations**

  An imbalance of work may occur when you perform a GROUP BY operation based on a key that is not evenly distributed.

  Assume that Table A has two fields, which are Key and Value. The table contains a large amount of data and the values of the Key field are not evenly distributed. Execute the following statement to perform a GROUP BY operation on the table:

  ```
  select key,count(value) from A group by key;
  ```

  When the amount of the data in the table is large enough, you may find long trails on the Logview page of the query. To resolve the issue, add `set odps.sql.groupby.skewindata=true` before the preceding statement to enable anti-skew before the query is performed.

- **Skewed reduce tasks during improper use of dynamic partitioning**

  If you use dynamic partitioning in MaxCompute, one or more reduce tasks are assigned to each partition to aggregate data by partition. This brings the following benefits:

  - Reduce the number of small files generated by MaxCompute and improve the efficiency of processing.
  - Avoid high memory usage when a worker needs to write many files to a partition.

  If the data to be written to partitions is skewed, long tails may occur during the reduce stage. Each partition can be assigned a maximum of 10 map tasks. If a larger amount of data is to be written to a partition than the other partitions, long tails may occur. If you can determine the partition to which data is to be written, we recommend that you do not use dynamic partitioning. For example, long tails may occur if you execute the following statement to write data from a specific partition in a table to another table:

  ```
  insert overwrite table A2 partition(dt) select split_part(value,'\t',1) as field1, split_
  part(value,'\t',2) as field2, dt from A where dt='20151010';
  ```

  In this case, you can replace the preceding statement with the following statement:

```
insert overwrite table A2 partition(dt='20151010')
select
split_part(value,'\t',1) as field1,
split_part(value,'\t',2) as field2
from A
where dt='20151010';
```

For more information about how to reduce impacts of data skew, see Long-tail computing optimization.

Optimize window functions

If window functions are used in SQL statements, a reduce task is assigned to each window function. A large number of window functions consume a large amount of resources. You can optimize the window functions that meet both of the following conditions:

- The OVER clause which defines how to partition and sort rows in a table must be the same.
- Multiple window functions must be executed at the same level of nesting in an SQL statement.

The window functions that meet the preceding conditions are merged to be executed by one reduce task. The following SQL statement provides an example:

```
select
rank()over(partition by A order by B desc) as rank,
row_number()over(partition by A order by B desc) as row_num
from MyTable;
```

Optimize subqueries

The following statement contains a subquery:

```
SELECT * FROM table_a a WHERE a.col1 IN (SELECT col1 FROM table_b b WHERE xxx);
```

If the subquery on the table_b table returns more than 1,000 values from the col1 column, the system reports the following error: `records returned from subquery exceeded limit of 1000`. In this case, you can replace the preceding statement with the following statement:

```
SELECT a. * FROM table_a a JOIN (SELECT DISTINCT col1 FROM table_b b WHERE xxx) c ON (a.col
1 = c.col1)
```

> **Note**
>
> - If the DISTINCT keyword is not used, the subquery result table c may contain duplicate values in the col1 column. In this case, the query on the a table returns more results.
> - If the DISTINCT keyword is used, only one worker is assigned to perform the subquery. If the subquery involves a large amount of data, the whole query slows down.
> - If you are sure that the values that meet the subquery conditions in the col1 column are unique, you can delete the DISTINCT keyword to improve the query performance.

Optimize joins

When you join two tables, we recommend that you use the WHERE clause based on the following rules:

- Specify the partition limits of the primary table in the WHERE clause. We recommend that you define a subquery for the primary table to obtain the required data first.
- Write the WHERE clause of the primary table at the end of the statement.
- Specify the partition limits of the secondary table in the ON clause or a subquery.

Examples:

```
select * from A join (select * from B where dt=20150301)B on B.id=A.id where A.dt=20150301;
select * from A join B on B.id=A.id where B.dt=20150301; -- We recommend that you do not us
e this statement. The system performs the JOIN operation before it performs partition pruni
ng. This can result in a large amount of data and deteriorate the query performance.
select * from (select * from A where dt=20150301)A join (select * from B where dt=20150301)
B on B.id=A.id;
```

4.2. Optimize JOIN long tails

This topic describes the data skew issue and provides related solutions. This issue may occur when the JOIN statement in MaxCompute SQL is executed.

Background information

When the JOIN statement in MaxCompute SQL is executed, the data with the same join key is sent to and processed on the same instance. If a key contains a large amount of data, the instance takes a longer time to process the data than other instances. Long tails exist if the execution log shows that a few instances in this JOIN task remain in the executing state and other instances are in the completed state.

Long tails caused by data skew are common and significantly prolong task execution. During promotions such as Double 11, severe long tails may occur. For example, page views of large sellers are much more than page views of small sellers. If page view log data is associated with the seller dimension table, data is distributed by seller ID. This causes some instances to process far more data than others. In this case, the task cannot be completed due to a few long tails.

You can resolve long tails from four perspectives:

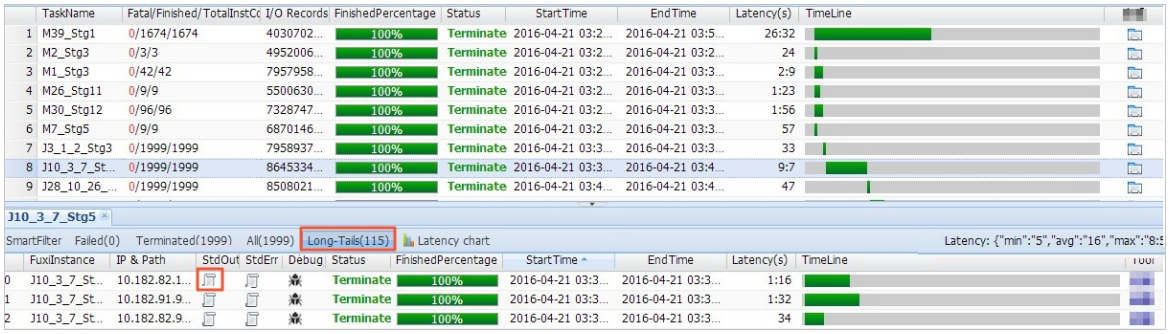
- If you want to join one large table and one small table, you can execute the MAP JOIN statement to cache the small table. For more information about the MAP JOIN statement, see SELECT syntax.
- To join two large tables, deduplicate data first.
- Try to find out the cause for the Cartesian product of two large keys and optimize these keys from the business perspective.
- It takes a long time to directly execute the LEFT JOIN statement for a small table and a large table. In this case, we recommend that you execute the MAP JOIN statement for the small and large tables to generate an intermediate table that contains the intersection of the two tables. This intermediate table is not greater than the large table because the MAP JOIN statement filters out unnecessary data from the large table. Then, execute the LEFT JOIN statement for the small and intermediate tables. The effect of this operation is equivalent to that of executing the LEFT JOIN statement for the small and large tables.

Check data skew

Perform the following steps to check data skew:

1. Open the log file generated on Logview when SQL statements are executed and check the execution details of each Fuxi task. In the following figure, **Long-Tails(115)** indicates that 115

long tails exist.



2. Find your **Fuxi instance** and click the icon in the **StdOut** column to view the size of data read by the instance.

   For example, `Read from 0 num:52743413 size:1389941257` indicates that 1,389,941,257 rows of data are being read when the JOIN statement is executed. If an instance listed in Long-Tails reads far more data than other instances, a long tail occurs due to the large data size.

Common causes and solutions

- MAP JOIN statement: If data skew occurs when the JOIN statement is executed on a large table and a small table, you can execute the MAP JOIN statement to prevent a long tail.

  When you use the MAP JOIN statement, the JOIN operation is performed at the Map side. This prevents data skew caused by uneven key distribution. The MAP JOIN statement is subject to the following limits:

  - The MAP JOIN statement is applicable only when the secondary table is small. A secondary table refers to the right table in the LEFT OUTER JOIN statement or the left table in the RIGHT OUTER JOIN statement.
  - The size of the small table is also limited when the MAP JOIN statement is used. By default, the maximum size is 512 MB after the small table is loaded into the memory. You can execute the following statement to increase the maximum size to 10,000 MB:

    ```
    set odps.sql.mapjoin.memory.max=10000
    ```

  The MAP JOIN statement is easy to use. You can append `/* mapjoin(b) */` to the `SELECT` statement, where b indicates the alias of the small table or the subquery. Example:

  ```
  select   /* mapjoin(b) */
             a.c2
           ,b.c3
  from
           (select   c1
                    ,c2
             from     t1          ) a
  left outer join
           (select   c1
                    ,c3
            from     t2          ) b
  on       a.c1 = b.c1;
  ```

- JOIN long tails caused by hot key values

If hot key values cause a long tail and the MAP JOIN statement cannot be used because no small table is involved, extract hot key values. Hot key values in the primary table are separated from non-hot key values, processed independently, and then joined with non-hot key values. In the following example, the page view log table of the Taobao website is associated with the commodity dimension table.

i. Extract hot key values: Extract the IDs of the commodities whose page views are greater than 50,000 to a temporary table.

```
insert   overwrite table topk_item PARTITION (ds = '${bizdate}')
select   item_id
from
        (select   item_id
                 ,count(1) as cnt
         from     dwd_tb_log_pv_di
         where    ds = '${bizdate}'
         and      url_type = 'ipv'
         and      item_id is not null
         group by item_id
         ) a
where    cnt >= 50000;
```

ii. Extract non-hot key values.

Execute the OUTER JOIN statement to associate the *sdwd_tb_log_pv_di* primary table with the *topk_item* hot key table. Then, apply condition `b1.item_id is null` to extract the log data of non-hot commodities that cannot be associated. In this case, execute the MAP JOIN statement to extract non-hot key values. Then, associate the non-hot key table with the commodity dimension table. No long tails occur because hot key values have been removed.

```
select   ...
from
        (select   *
         from     dim_tb_itm
         where    ds = '${bizdate}'
         ) a
right outer join
        (select   /* mapjoin(b1) */
                  b2.*
         from
                 (select   item_id
                  from     topk_item
                  where    ds = '${bizdate}'
                  ) b1
         right outer join
                 (select   *
                  from     dwd_tb_log_pv_di
                  where    ds = '${bizdate}'
                  and      url_type = 'ipv'
                  ) b2
         on       b1.item_id = coalesce(b2.item_id,concat("tbcdm",rand())
         where    b1.item_id is null
         ) l
on       a.item_id = coalesce(l.item_id,concat("tbcdm",rand());
```

iii. Extract hot key values.

Execute the INNER JOIN statement to associate the *sdwd_tb_log_pv_di* primary table with the *topk_item* hot key table. In this case, execute the MAP JOIN statement to extract the log data of hot commodities. Execute the INNER JOIN statement to associate the *dim_tb_itm* commodity dimension table with the *topk_item* hot key table to extract the data of the hot commodity dimension table. Execute the OUTER JOIN statement to associate the log data with the data of the dimension table. The dimension table contains a small amount of data and MAP JOIN can be used to prevent long tails.

```
select   /* mapjoin(a) */
         ...
from
        (select   /* mapjoin(b1) */
                  b2.*
         from
                 (select   item_id
                  from     topk_item
                  where    ds = '${bizdate}'
                  )b1
         join
                 (select   *
                  from     dwd_tb_log_pv_di
                  where    ds = '${bizdate}'
                  and      url_type = 'ipv'
                  and      item_id is not null
                  ) b2
         on       (b1.item_id = b2.item_id)
         ) l
left outer join
        (select   /* mapjoin(a1) */
                  a2.*
         from
                 (select   item_id
                  from     topk_item
                  where    ds = '${bizdate}'
                  ) a1
         join
                 (select   *
                  from     dim_tb_itm
                  where    ds = '${bizdate}'
                  ) a2
         on       (a1.item_id = a2.item_id)
         ) a
on       a.item_id = l.item_id;
```

iv. Execute the `UNION ALL` statement to merge the data obtained in Substeps ii and iii to generate complete log data, with commodity information associated.

- Set the odps.sql.skewjoin parameter to resolve long tails.

  This is a simple solution. However, you must modify code and execute the statements again if skewed key values change. In addition, value changes cannot be predicted. If many skewed key values exist, it is inconvenient for you to configure them in parameters. In this case, you can split code or specify parameters as required. Perform the following steps to set the odps.sql.skewjoin parameter:

i. Set the odps.sql.skewjoin parameter to true.

```
set odps.sql.skewjoin=true
```

ii. Set a skewed key and its value.

```
set odps.sql.skewinfo=skewed_src:(skewed_key) [("skewed_value")]
```

skewed_key indicates the skewed column and skewed_value indicates the skewed value of this column.

- Use SKEWJOIN HINT to avoid skewed hot key values. For more information about SKEWJOIN HINT, see SKEWJOIN HINT.

Procedure

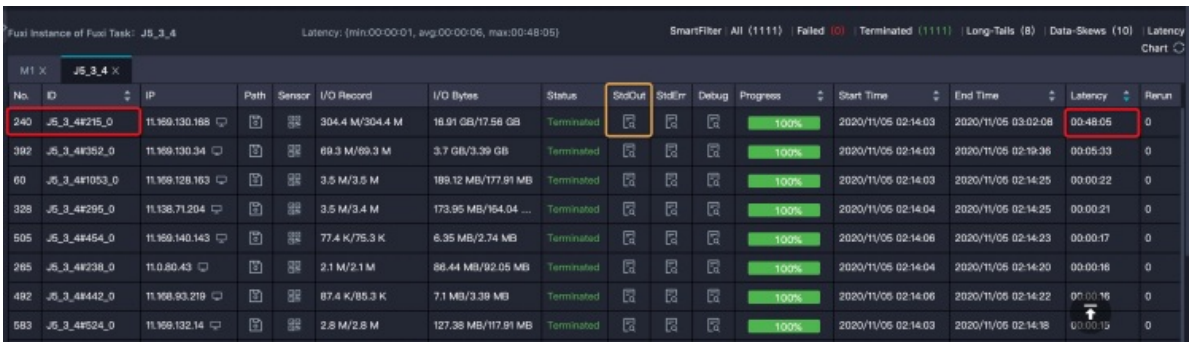
```
-- Method 1: Include the alias of the table in SKEWJOIN HINT.
select /*+ skewjoin(a) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c1 = b.c1;
-- Method 2: Include the table name and possibly skewed columns in SKEWJOIN HINT. In the following statement, the c0 and c1 columns of table a are skewed columns.
select /*+ skewjoin(a(c0, c1)) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c1 = b.c1 and a.c2 = b.c2;
-- Method 3: Include the table name, columns, and skewed hot key values in SKEWJOIN HINT. If skewed key values are of the STRING type, enclose each value with double quotation marks. In the following statement, (a.c0=1 and a.c1="2") and (a.c0=3 and a.c1="4") contain skewed hot key values.
select /*+ skewjoin(a(c0, c1)((1, "2"), (3, "4"))) */ * from T0 a join T1 b on a.c0 = b.c0 and a.c1 = b.c1 and a.c2 = b.c2;
```

> **Note** Method 3 is more efficient than Method 1 and Method 2.

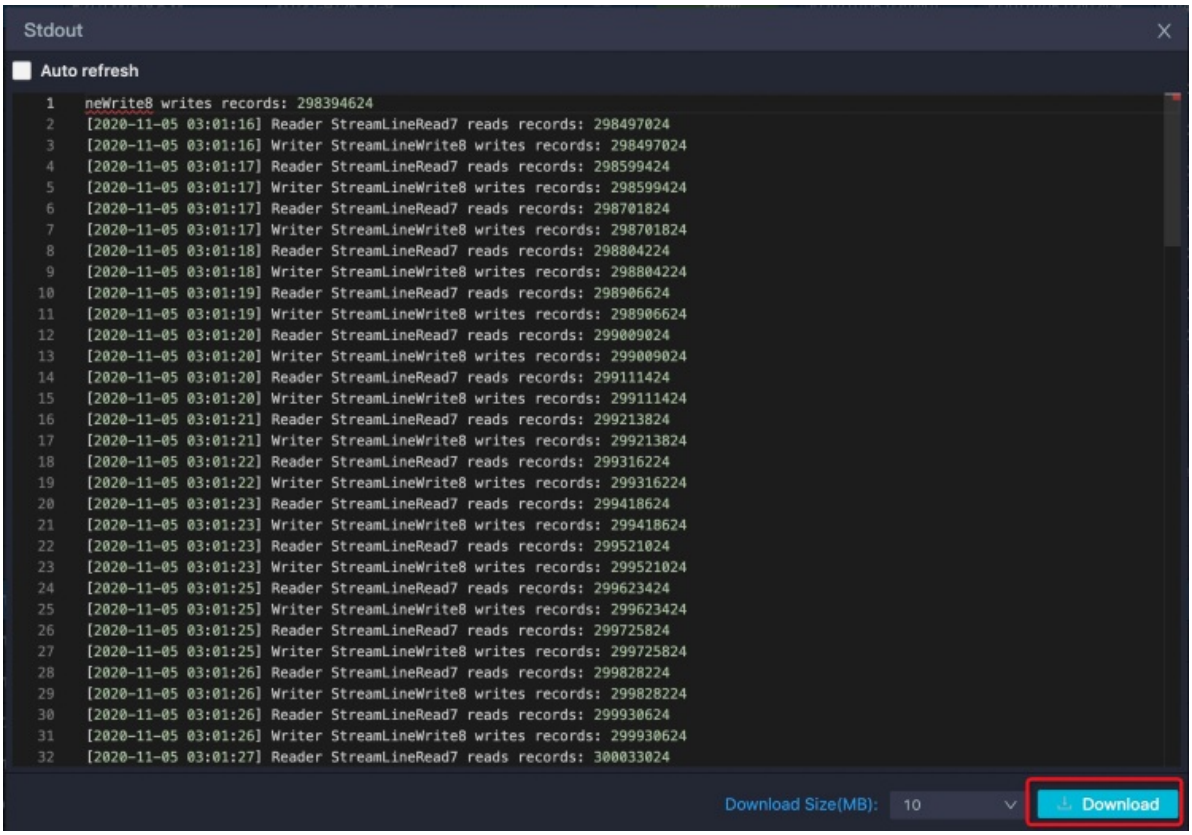
Identify the JOIN statement that causes data skew

In the following snapshot captured on Logview, J5_3_4 is the Fuxi task that took the longest time to execute.

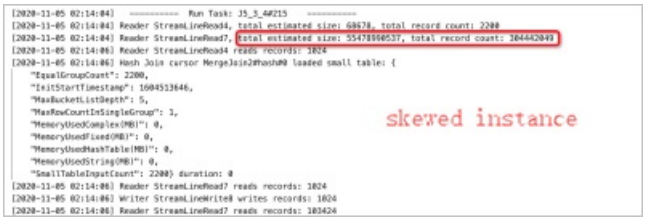
Click the J5_3_4 task and query the instances of this task on the tab that appears. The query results show that the J5_3_4#215_0 instance took the longest time to execute and its I/O records and I/O bytes are much more than those of other instances.



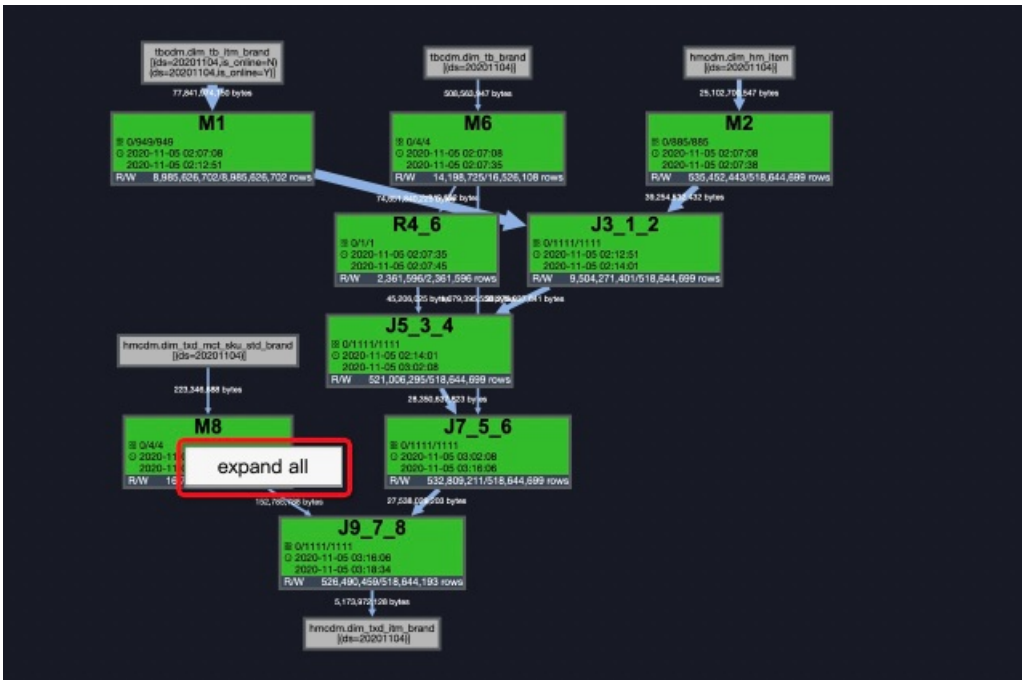
In this case, you can find that data skew occurs on the J5_3_4#215_0 instance. The JOIN statement that causes data skew needs to be further determined. Find the skewed instance, and click the icon in the **StdOut** column. Find a non-skewed instance, and click the icon in the **StdOut** column. The content in the **StdOut** column cannot be completely displayed. You can click **Download** and view the complete information.

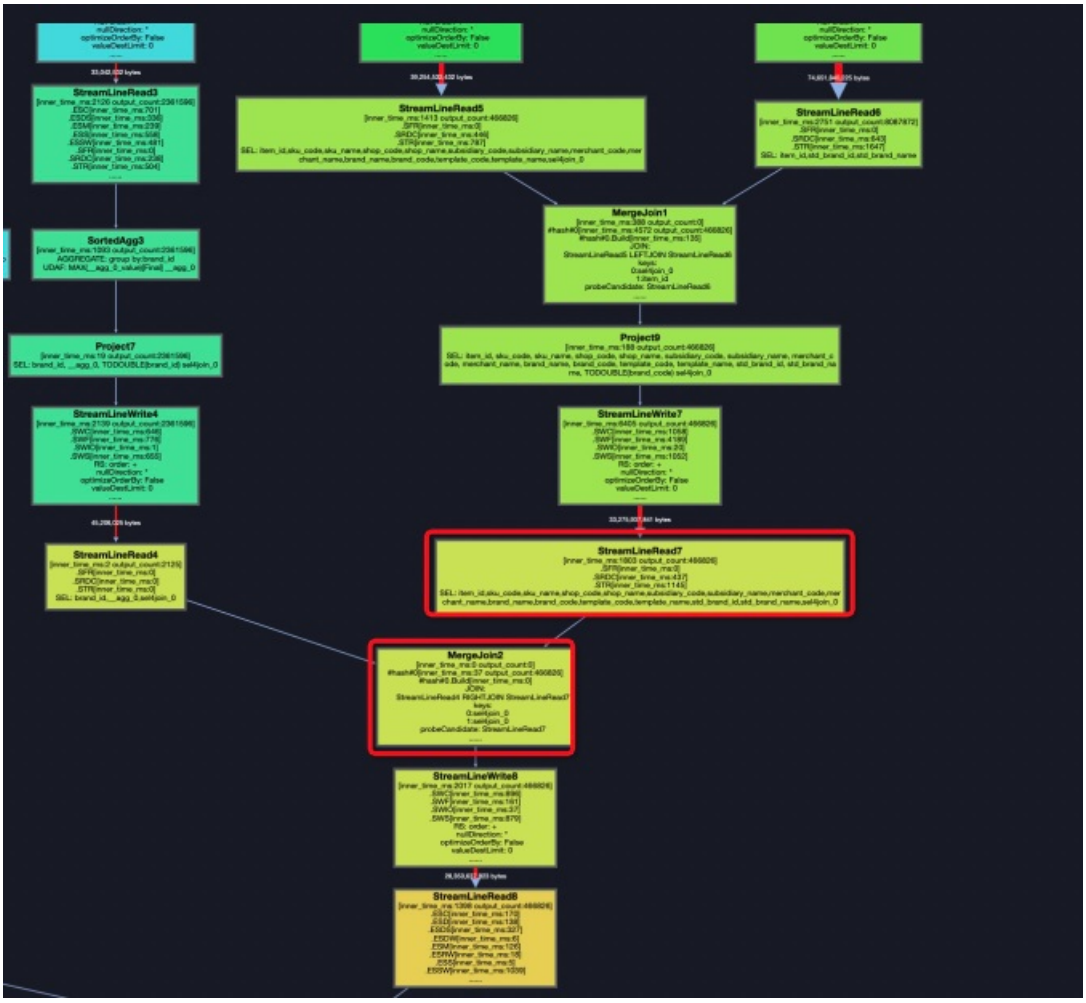


In the following figures, you can find that the value of record count in StreamLineRead7 of the skewed instance is much greater than the value of record count of the non-skewed instance. Therefore, data skew occurs when data in StreamLineWrite7 and SreamLineRead7 is shuffled.

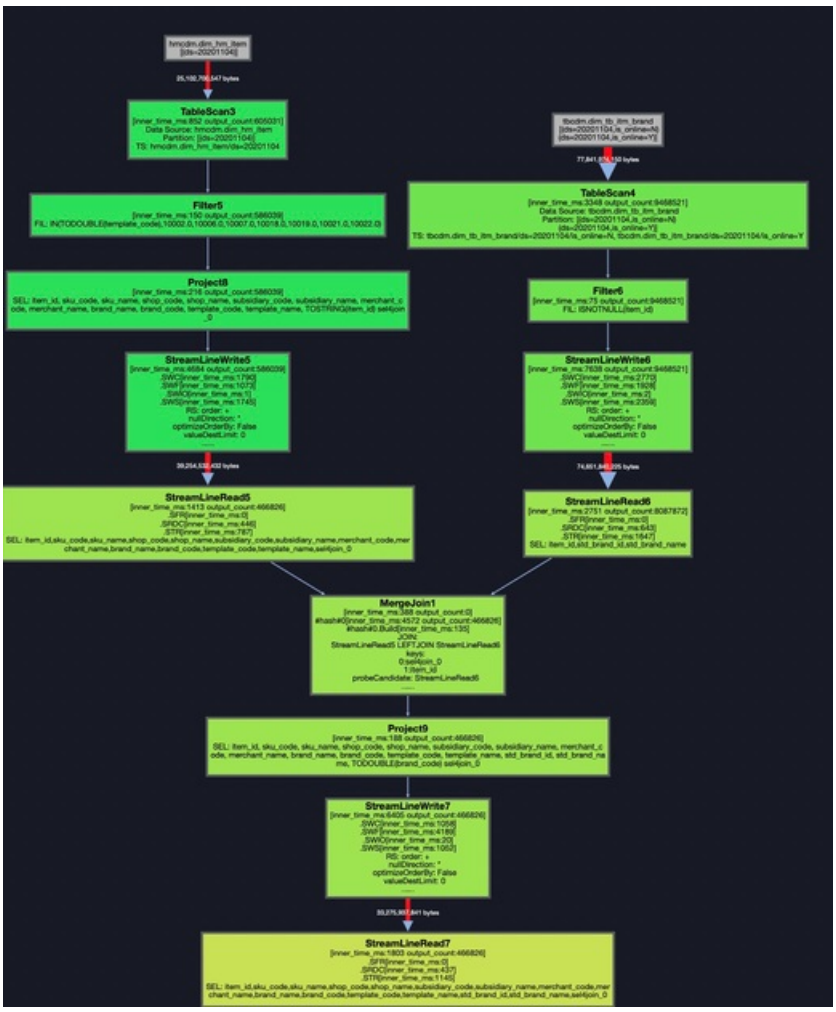


On the DAG page, right-click the skewed instance and select **expand all** to find StreamLineWrite7 and StreamLineRead7.




StreamLineRead3
StreamLineRead5
StreamLineRead6
SortedAgg3
MergeJoin1
Project7
Project9
StreamLineWrite4
StreamLineWrite7
StreamLineRead4
StreamLineRead7
MergeJoin2
StreamLineWrite8
StreamLineRead8

You can find that data skew occurs on StreamLineRead7 in MergeJoin2. MergeJoin2 is generated after the dim_hm_item and dim_tb_itm_brand tables are joined and then the joined table and the dim_tb_brand table are joined.



Use these table names to find the skewed table. The result shows that data skew occurs when the LEFT OUTER JOIN statement is executed and the t1 table is skewed. You can add `/*+ skewjoin(t1) */` to the SQL statement to resolve the data skew issue.

```
        t1.sku_code,
        t1.sku_name,
        t1.brand_code,
        t1.brand_name,
        t2.std_brand_id,
        t2.std_brand_name
    from hmcdm.dim_hm_item t1
    left outer join tbcdm.dim_tb_itm_brand t2
    on cast(t1.item_id as string)=cast(t2.item_id as string)
    and t2.ds=max_pt('tbcdm.dim_tb_itm_brand')
    where t1.ds='20201104'
    and t1.template_code in (
        10002,
        10006,
        10007,
        10018,
        10019,
        10021,
        10022
    )
) t1
left outer join(
    select brand_id as std_brand_id, max(brand_name) as std_brand_name
    from tbcdm.dim_tb_brand
    where ds=max_pt('tbcdm.dim_tb_brand')
    and is_standard_brand='Y'
    and cast(brand_id as bigint)>0
    group by brand_id
) t21
on t1.brand_code=t21.std_brand_id
left outer join(
    select
```

4.3. Long-tail computing optimization

Long tails may occur in JOIN operations and other computing jobs. This topic describes the scenarios in which long tails occur and the solutions.

Long tails are one of the common issues in distributed computing. The main cause of a long tail is uneven data distribution. As a result, the workloads of individual nodes differ. The entire job can be completed only after the slowest node processes all its data.

To prevent one worker from running a large number of jobs, the jobs must be distributed to multiple workers.

GROUP BY long tail

Causes

The computing workloads for the key of a GROUP BY clause are heavy.

Solution

You can use one of the following methods to handle this issue:

- Rewrite the SQL statement and add random numbers to split the key. Example:

```
SELECT Key,COUNT(*) AS Cnt FROM TableName GROUP BY Key;
```

Regardless of combiners, a mapper shuffles data to a reducer, and the reducer performs the count operation. The execution plan is in the following sequence: Mapper > Reducer. However, if the jobs of the long-tailed key are distributed again, use the following statement:

```
-- Assume that the long-tailed key is KEY001.
SELECT a.Key
  , SUM(a.Cnt) AS Cnt
FROM (
  SELECT Key
    , COUNT(*) AS Cnt
FROM TableName
GROUP BY Key,
  CASE
    WHEN Key = 'KEY001' THEN Hash(Random()) % 50
    ELSE 0
   END
) a
GROUP BY a.Key;
```

The execution plan for this statement is in the following sequence: Mapper > Reducer > Reducer. Although more steps are required for the execution, the jobs of the long-tailed key are processed in two steps, and the time required may be shorter.

> **Note** If you use this method to add a reducer execution step to handle a long tail that has slight impacts on your jobs, the time required may be longer.

- Specify system parameters.

```
set odps.sql.groupby.skewindata=true
```

This configuration is used for general optimization instead of business-specific optimization. Therefore, the optimization effect may not be optimal. You can rewrite SQL statements in a more efficient way based on your data.

DISTINCT long tail

If a long tail occurs for the DISTINCT keyword, the key splitting method does not apply. In this case, you must seek for other methods.

Solution

```
-- The original SQL statement, regardless of the case where uid is not specified.
SELECT COUNT(uid) AS Pv
    , COUNT(DISTINCT uid) AS Uv
FROM UserLog;
```

The preceding statement can be rewritten into the following statement:

```
SELECT SUM(PV) AS Pv
    , COUNT(*) AS UV
FROM (
    SELECT COUNT(*) AS Pv
      , uid
    FROM UserLog
    GROUP BY uid
) a;
```

This method is to change DISTINCT to COUNT. This way, the computing workloads are distributed to different reducers. After you rewrite the statement, you can use the optimization method for GROUP BY, and the combiner is involved in the computation. This greatly improves the performance.

Long tail of dynamic partitions

Causes

- To sort the data of small files, the dynamic partition feature starts a reducer at the final stage of execution. If data written by using the dynamic partition feature is skewed, a long tail occurs.
- In general, the incorrect use of the dynamic partition feature causes long tails.

Solution

If you are sure about the partition to which data is written, you can specify the partition before you insert the data instead of using dynamic partitions.

Use combiners to handle long tails

Combiners are frequently used to handle long tails in MapReduce jobs. Combiners can be used to reduce the amount of data that needs to be shuffled from mappers to reducers. This greatly reduces the overheads of network transmission. This optimization is automatically implemented in MaxCompute SQL.

> **Note** Combiners only optimize execution in the map stages. Make sure that the results of the execution during which combiners are used are the same as those of the execution during which combiners are not used. WordCount is used in this example. The result of passing `(KEY,1)` twice is the same as that of passing `(KEY,2)` once. For more information, see WordCount. However, when you calculate the average value, you cannot use a combiner to directly combine `(KEY,1)` and `(KEY,2)` to obtain `(KEY,1.5)`.

Optimize the system to handle long tails

In addition to combiners, MaxCompute is also optimized. For example, the following logs (+N backups) are generated during the running of a job.

```
M1_Stg1_job0:0/521/521[100%] M2_Stg1_job0:0/1/1[100%] J9_1_2_Stg5_job0:0/523/523[100%] J3_1
_2_Stg1_job0:0/523/523[100%] R6_3_9_Stg2_job0:1/1046/1047[100%]
M1_Stg1_job0:0/521/521[100%] M2_Stg1_job0:0/1/1[100%] J9_1_2_Stg5_job0:0/523/523[100%] J3_1
_2_Stg1_job0:0/523/523[100%] R6_3_9_Stg2_job0:1/1046/1047[100%]
M1_Stg1_job0:0/521/521[100%] M2_Stg1_job0:0/1/1[100%] J9_1_2_Stg5_job0:0/523/523[100%] J3_1
_2_Stg1_job0:0/523/523[100%] R6_3_9_Stg2_job0:1/1046/1047(+1 backups)[100%]
M1_Stg1_job0:0/521/521[100%] M2_Stg1_job0:0/1/1[100%] J9_1_2_Stg5_job0:0/523/523[100%] J3_1
_2_Stg1_job0:0/523/523[100%] R6_3_9_Stg2_job0:1/1046/1047(+1 backups)[100%]
```

A total of 1,047 reducers are used. Among these reducers, 1,046 reducers have completed their calculations, but the last one has not. After MaxCompute detects this issue, it automatically starts a new reducer, calculates the same data, and then aggregates the results of the reducer that completed the calculation earlier to the final result set.

Optimize business logic to handle long tails

The aforementioned optimization methods cannot handle all the long tails. In some cases, you must optimize your business logic to handle long tails.

- A large amount of noisy data may exist in calculations. For example, you need to calculate the data based on visitor IDs to check the access records of each user. In this case, you must filter out crawler data. Otherwise, a long tail may occur due to the crawler data during calculation. It is increasingly difficult to identify crawler data. Similarly, if you want to use the xxid field for associations, you must check whether the associated field is empty.
- Long tails may occur in some special business scenarios. For example, the operation records of independent software vendors (ISVs) are greatly different from those of individuals in terms of the amount of data and behavior. In this case, you must use specific analysis methods to handle the issues of important customers.
- If data is unevenly distributed, we recommend that you do not use constants as the key of DISTRIBUTE BY to sort all the data records.

4.4. Optimize the calculation for long-period metrics

This topic describes how to optimize the calculation for long-period metrics.

Background information

When e-commerce companies build data warehouses or analyze their business, they often need to calculate metrics such as the numbers of visitors, buyers, and regular buyers in a period of time. These metrics are calculated based on the data that is accumulated over the period of time.

In general, these metrics are calculated based on the data in log tables. For example, you can execute the following statement to calculate the number of visitors for each item in the last 30 days:

```
select item_id -- The field that indicates the item ID.
  ,count(distinct visitor_id) as ipv_uv_1d_001
from vistor_item_detail_log
where ds <= ${bdp.system.bizdate}
and ds >=to_char(dateadd(to_date(${bdp.system.bizdate},'yyyymmdd'),-29,'dd'),'yyyymmdd')
 group by item_id;
```

> **Note** All the variables in the code samples in this topic are scheduling variables in DataWorks. Therefore, the code samples in this topic are applicable only to scheduling nodes in DataWorks.

If a large amount of log data is generated every day, the preceding SELECT statement requires a large number of map tasks. If more than 99,999 map tasks are required, the map tasks fail.

Objective

Calculate long-period metrics with minimal impact on the query performance.

The amount of the data accumulated over a long period of time is huge. If the system calculates metrics based on the data, the query performance is deteriorated. We recommend that you create an intermediate table that is used to summarize the data generated every day. This can remove duplicate data records and reduce the amount of data to be queried.

Solution

1. Create an intermediate table to summarize the data generated every day.

   In this example, you can create an intermediate table based on the data in the item_id and visitor_id fields. The following code provides an example:

```
insert overwrite table mds_itm_vsr_xx(ds='${bdp.system.bizdate} ')
select item_id,visitor_id,count(1) as pv
  from
  (
  select item_id,visitor_id
  from vistor_item_detail_log
  where ds =${bdp.system.bizdate}
  group by item_id,visitor_id
  ) a;
```

2. Summarize the data that is accumulated over a long period of time from the intermediate table.

   The following code calculates the number of visitors for each item in the last 30 days:

```
select item_id
        ,count(distinct visitor_id) as uv
        ,sum(pv) as pv
  from mds_itm_vsr_xx
  where ds <= '${bdp.system.bizdate} '
  and ds >= to_char(dateadd(to_date('${bdp.system.bizdate} ','yyyymmdd'),-29,'dd'),'yyy
ymmdd')
  group by item_id;
```

Impact and consideration

In the preceding solution, the log data is deduplicated on a daily basis. This can reduce the amount of data for calculation and improve the query performance. However, the system needs to read data from multiple partitions for every calculation on the data that is accumulated over a long period of time.

To resolve this issue, you can merge data from multiple partitions into one partition, which contains all historical data. This way, you can accumulate data in an incremental manner and calculate long-period metrics based on data in one partition.

Scenarios

Calculate the number of the regular buyers in the last day. A regular buyer is a buyer who made a purchase in a specific period of time, for example, in the last 30 days.

The following code calculates the number of the regular buyers in a period of time:

```
select item_id -- The field that indicates the item ID.
        ,buyer_id as old_buyer_id
from buyer_item_detail_log
where ds < ${bdp.system.bizdate}
and ds >=to_char(dateadd(to_date(${bdp.system.bizdate},'yyyymmdd'),-29,'dd'),'yyyymmdd')
group by item_id
        ,buyer_id;
```

Improvement:

- Create and maintain a dimension table. This table is used to record the relationship between buyers and purchased items, such as the first purchase time, the last purchase time, the total number of purchased items, and the total amount of the purchases.
- Update the data in the dimension table every day with the data in the billing logs of the last day.
- To determine whether a buyer is a regular buyer, check whether the last purchase time of the buyer is within the last 30 days. This deduplicates data mappings and reduces the amount of data for calculation.

5.Job diagnostics

5.1. Use Logview to diagnose jobs that run slowly

In most cases, enterprises need job results to be generated earlier than expected. This way, they can make business development decisions based on the results at the earliest opportunity. In this case, job developers must pay attention to the job status to identify and optimize the jobs that run slowly. You can use Logview of MaxCompute to diagnose jobs that run slowly. This topic provides the causes for which jobs run slowly and the related solutions. This topic also describes how to view information about the jobs that run slowly.

Background information

Logview of MaxCompute records all logs of jobs and provides guidance for you to view and debug jobs. You can obtain the Logview URL below **Log view** in the job result. MaxCompute provides two versions of Logview. We recommend that you use Logview V2.0 because it provides faster page loading and a better design style. For more information about Logview V2.0, see Logview V2.0.

Common causes for which jobs run slowly:

- Insufficient CUs

  If the MaxCompute project uses the subscription billing method and a large number of jobs are submitted or a large number of small files are generated within a specific period of time, all the purchased compute units (CUs) are occupied and the jobs become queued.

- Data skew

  If a large amount of data is processed or some jobs are dedicated for some special data, long tails may occur even if most jobs are completed.

- Inefficient code logic

  If the SQL or user-defined function (UDF) logic is inefficient or parameter settings are not optimal, a Fuxi task may run for a long period of time. However, the time for which each Fuxi instance runs is almost the same. For more information about the relationships between jobs, Fuxi tasks, and Fuxi instances, see Job details section.

Insufficient CUs

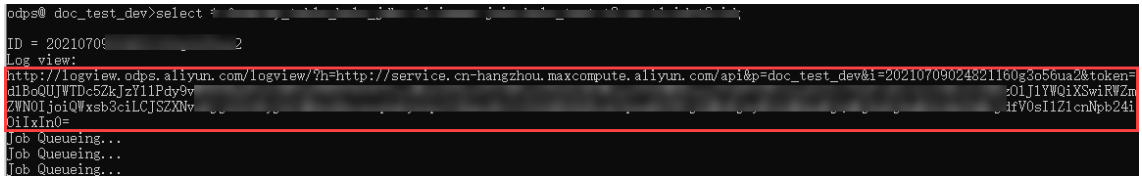
Problem description

If the CUs are insufficient, the following issues may occur after you submit a job:

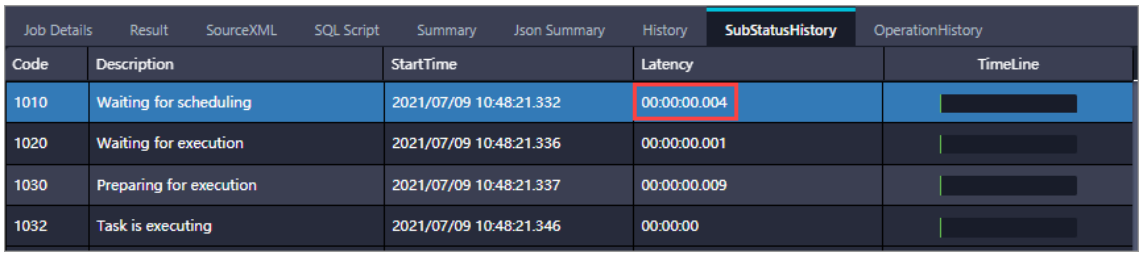
- Issue 1: `Job Queueing...` is displayed.

  The job may be queued because other jobs occupy the resources of the resource group. You can perform the following steps to view the duration for which the job is queued:

  i. Obtain the Logview URL in the job result and open the URL in the browser.



ii. On the **SubStatusHistory** tab of Logview, find Waiting for scheduling in the **Description** column and view the value in the **Latency** column. The value indicates the duration for which the job is queued.



Code	Description	StartTime	Latency	TimeLine
1010	Waiting for scheduling	2021/07/09 10:48:21.332	00:00:00.004	
1020	Waiting for execution	2021/07/09 10:48:21.336	00:00:00.001	
1030	Preparing for execution	2021/07/09 10:48:21.337	00:00:00.009	
1032	Task is executing	2021/07/09 10:48:21.346	00:00:00	

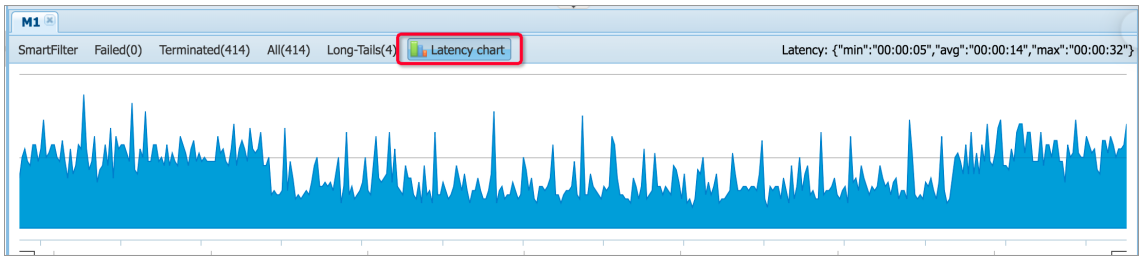
- Issue 2: The job runs slowly.

  After a job is submitted, a large number of CUs are required. However, the resource group cannot start all Fuxi instances at the same time. As a result, the job runs slowly. You can perform the following steps to view the job status:

  i. Obtain the Logview URL in the job result and open the URL in a browser.

  ii. In the Fuxi Instance section of the **Job Details** tab, click **Latency chart** to view the job status diagram.

  The following figure shows the status of a job that has sufficient resources. The lower blue part in the diagram remains at approximately the same height, which indicates that all Fuxi instances of the job start at approximately the same time.



  The following figure shows the status of a job that does not have sufficient resources. The diagram shows an upward trend, which indicates that the Fuxi instances of the job are gradually scheduled.

Causes

To locate the causes of the preceding issues, perform the following steps:

1. Go to MaxCompute Management.

2. In the left-side navigation pane, click **Quotas**.

3. In the **Subscription Quota Groups** section, click the quota group that corresponds to the MaxCompute project.
4. In the **Usage Trend of Reserved CUs** chart on the **Resource Consumption** tab, click the point at which the CU usage is the highest and record the point in time.
5. In the left-side navigation pane, click **Jobs**. On the right part of the page, click the **Job Management** tab.
6. On the Job Management tab, configure **Time Range** based on the point in time that you recorded, select **Running** from the **Job Status** drop-down list, and then click **OK**.
7. In the job list, click the icon next to **CPU Utilization (%)** to sort jobs by CPU utilization in descending order.
   - If the CPU utilization of a job is excessively high, click Logview in the **Actions** column and view **I/O Bytes** in the Fuxi Instance section. If **I/O Bytes** is only 1 MB or tens of KB and multiple Fuxi instances are running in the job, a large number of small files are generated when the job is run. In this case, you need to merge the small files or adjust the parallelism.
   - If the values of CPU Utilization (%) are almost the same, multiple large jobs are submitted at the same time and the jobs consume all CUs. In this case, you must purchase additional CUs or use pay-as-you-go resources to run jobs.

Solutions

- Merge small files.
- Adjust the parallelism.

  The parallelism of MaxCompute jobs is automatically estimated based on the amount of input data and the job complexity. In most cases, you do not need to manually adjust the parallelism. If you adjust the parallelism to a higher value, the job processing speed increases. However, subscription resource groups may be fully occupied. In this case, jobs are queued to wait for resources and therefore run slowly. You can configure the odps.stage.mapper.split.size, odps.stage.reducer.num, odps.stage.joiner.num, or odps.stage.num parameter to adjust the parallelism. For more information, see SET operations.
- Purchase CUs.

  For more information about how to purchase CUs, see Upgrade resource configurations.
- Use pay-as-you-go resources.

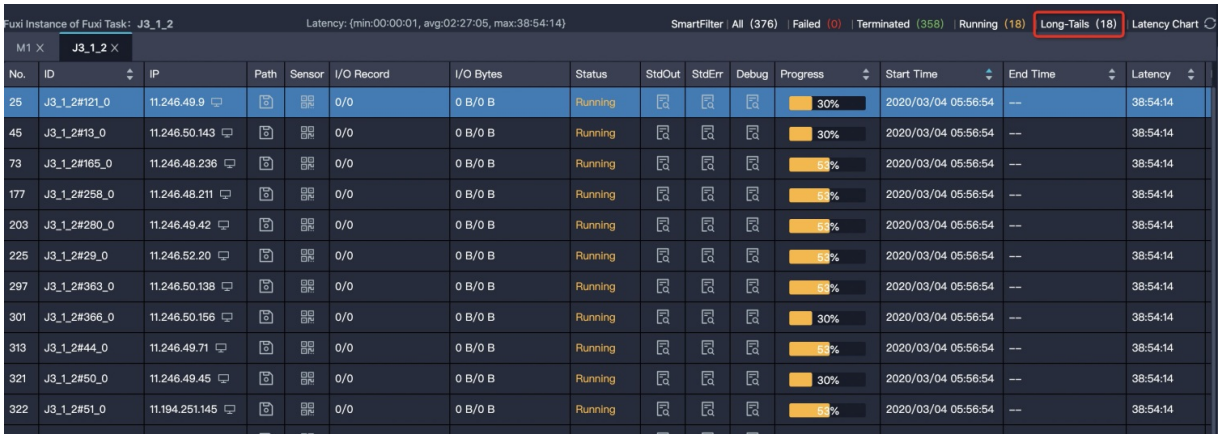
  Purchase pay-as-you-go resources and use MaxCompute Management to allow subscription projects to use the pay-as-you-go resources.

Data skew

Problem description

Some Fuxi instances in a Fuxi task continue to run even if most Fuxi instances of the Fuxi task stopped. As a result, long tails occur.

In the Fuxi Instance section of the **Job Details** tab of Logview, you can click **Long-Tails** to view the Fuxi instances that have a long tail.



No.	ID	IP	Path	Sensor	I/O Record	I/O Bytes	Status	StdOut	StdErr	Debug	Progress	Start Time	End Time	Latency
25	J3_1_2#121_0	11.246.49.9			0/0	0 B/0 B	Running				30%	2020/03/04 05:56:54	--	38:54:14
45	J3_1_2#13_0	11.246.50.143			0/0	0 B/0 B	Running				30%	2020/03/04 05:56:54	--	38:54:14
73	J3_1_2#165_0	11.246.48.236			0/0	0 B/0 B	Running				53%	2020/03/04 05:56:54	--	38:54:14
177	J3_1_2#258_0	11.246.48.211			0/0	0 B/0 B	Running				53%	2020/03/04 05:56:54	--	38:54:14
203	J3_1_2#280_0	11.246.49.42			0/0	0 B/0 B	Running				53%	2020/03/04 05:56:54	--	38:54:14
225	J3_1_2#29_0	11.246.52.20			0/0	0 B/0 B	Running				53%	2020/03/04 05:56:54	--	38:54:14
297	J3_1_2#363_0	11.246.50.138			0/0	0 B/0 B	Running				53%	2020/03/04 05:56:54	--	38:54:14
301	J3_1_2#366_0	11.246.50.156			0/0	0 B/0 B	Running				30%	2020/03/04 05:56:54	--	38:54:14
313	J3_1_2#44_0	11.246.49.71			0/0	0 B/0 B	Running				53%	2020/03/04 05:56:54	--	38:54:14
321	J3_1_2#50_0	11.246.49.45			0/0	0 B/0 B	Running				30%	2020/03/04 05:56:54	--	38:54:14
322	J3_1_2#51_0	11.194.251.145			0/0	0 B/0 B	Running				53%	2020/03/04 05:56:54	--	38:54:14

Cause

The Fuxi instances that continue to run process large amounts of data or are dedicated for special data.

Solution

For more information about how to resolve data skew, see Reduce impacts of data skew.

Inefficient code logic

Problem description

If the code logic is inefficient, the following issues may occur after you submit a job:

- Issue 1: Data expansion occurs. The amount of output data of a Fuxi task is significantly greater than the amount of input data.

  You can view **I/O Record** and **I/O Bytes** in the Fuxi Task section to check the amounts of input and output data of a Fuxi task. In the following figure, 1 GB of data is changed to 1 TB after the data is processed. One Fuxi instance processes 1 TB of data, which reduces data processing efficiency.

  SQL_0_0_0_job_0 | SQL_0_0_0_merge

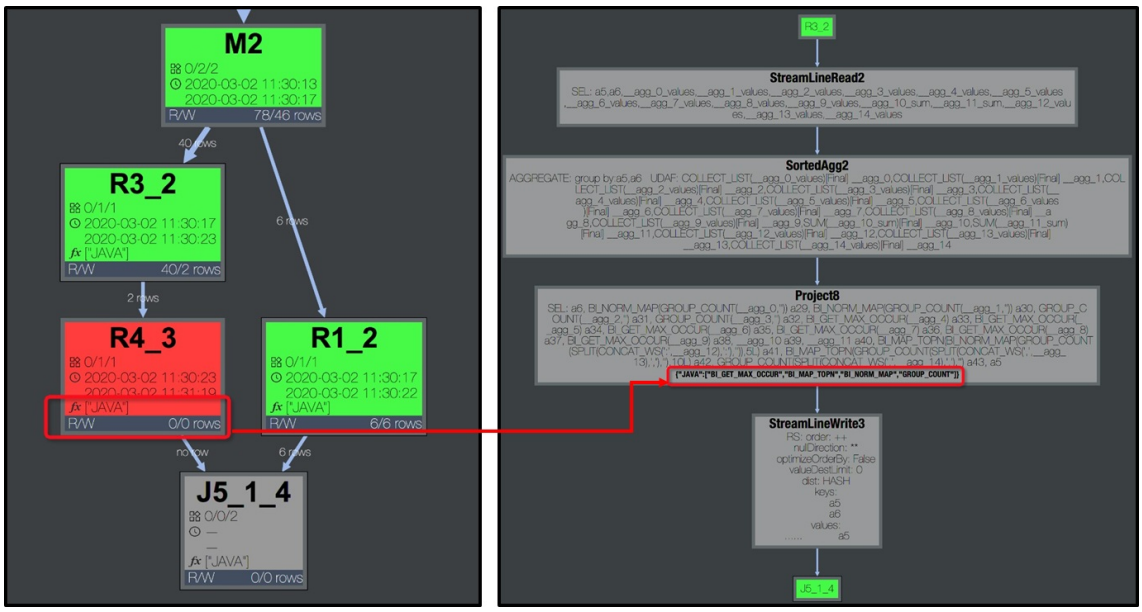
Fuxi Task	Failed/Terminated/ALL	I/O Record	I/O Bytes	Status	Sensor
M1	0/10,054/10,054	793.6 M/1.3 T	1.95 TB/14.88 TB	Terminated	
R2_1	0/4,000/4,000(+2 backups)	1.3 T/8.8 G	14.88 TB/189.14 GB	Terminated	
R3_2	0/4,000/4,000	8.8 G/5.6 M	189.14 GB/113.22 GB	Terminated	

- Issue 2: The UDF execution efficiency is low.

  A Fuxi task runs slowly, and the Fuxi task has UDFs. When a timeout error occurs on a UDF, the error `Fuxi job failed - WorkerRestart errCode:252,errMsg:kInstanceMonitorTimeout, usually caused by bad udf performance` is returned. You can perform the following steps to view the location and execution speed of the UDF:

  i. Obtain the Logview URL in the job result and open the URL in a browser.

  ii. In the progress chart, double-click the Fuxi Task that runs slowly or fails to run. In the operator graph, view the location of the UDF, as shown in the following figure.



iii. In the Fuxi Instance section, click StdOut to view the execution speed of the UDF.

In normal cases, the value of `Speed(records/s)` indicates that millions or hundreds of thousands of records are processed per second.

Causes

- Issue 1: The business processing logic causes data expansion. In this case, check whether the business logic meets your business requirements.
- Issue 2: The UDF code logic does not meet your business requirements. In this case, adjust the code logic.

Solutions

- Issue 1: Check whether the business logic has a defect. If the logic has a defect, modify the code. If the logic does not have a defect, configure the odps.stage.mapper.split.size, odps.stage.reducer.num, odps.stage.joiner.num, or odps.stage.num parameter to adjust the parallelism. For more information, see SET operations.
- Issue 2: Check and modify the UDF code logic. We recommend that you preferentially use built-in functions. If built-in functions cannot meet your business requirements, use UDFs. For more information about built-in functions, see Built-in functions.

6.Cost optimization

6.1. Overview

This topic describes the process of cost optimization.

Enterprises must continually optimize their costs on MaxCompute in response to the changes in big data. You can reference the following process for cost optimization:

1. Before you use MaxCompute, make sure that you fully understand the billing methods, accurately estimate the resources that you require, and then select an appropriate billing method. For more information, see Select a billing method.
2. To reduce costs when you use MaxCompute, optimize the resources that are used for data computing, storage, uploads, and downloads. For more information, see Optimize computing costs, Optimize storage costs, and Optimize the costs of data uploads and downloads.
3. View your bills in a timely manner. Analyze any exceptions in the bills and perform optimization. For more information, see Manage costs.

6.2. Select a billing method

This topic describes how to select a cost-effective billing method.

Billing methods

MaxCompute supports the following billing methods:

- Subscription: Computing resources are charged on a monthly or annual basis. Storage and download resources are charged on a pay-as-you-go basis.
- Pay-as-you-go: Storage, computing, and download resources are all charged on a pay-as-you-go basis.

For more information, see Billing method. You can select a billing method with the help of Total Cost of Ownership (TCO) tools and the best practices of cost estimation.

TCO tools

You can use the following TCO tools to estimate costs:

- MaxCompute price calculator: This tool is suitable for the subscription billing method. To calculate the monthly cost, enter the required computing resources and the volumes of the data you want to upload and download.
- CostSQL: This tool is suitable for the pay-as-you-go billing method.
  - You can run the cost sql command to estimate the cost of an SQL job before you execute the SQL job in a production environment. For more information, see Cost estimation.
  - If you use IntelliJ IDEA, you can submit SQL scripts for automatic cost estimation. For more information, see Develop and submit an SQL script.
  - If you use DataWorks, you can also estimate costs.

> Note
>
> - The costs of some SQL jobs cannot be estimated, such as SQL jobs that involve external tables.
> - The actual costs are subject to final bills.

Best practices of cost estimation

This section provides some cost estimation examples and tips for your reference. You can select a cost-effective billing method based on the information.

- **Billing methods for 1 TB of data**

  The following table describes estimated costs for reference.

Billing method	Business scenario	Response speed	Estimated cost per month
Subscription	Compute-intensive	Within a few minutes	3768 USD
	Storage-intensive	Within a few hours	1177.5 USD
Pay-as-you-go	1413 USD (The cost is estimated with the SQL complexity of 1 and the execution frequency of once per day.)		

  If you select the subscription billing method, the costs vary depending on your business type:

  - Compute-intensive scenario: In this scenario, a large number of CPU resources are required. 160 compute units are used to process 1 TB of data. The system responds to a request within a few minutes. The estimated cost is 3768 USD per month.
  - Storage-intensive scenario: If your jobs are not sensitive to the response speed, we recommend that you purchase a storage plan. About 50 compute units are used to process 1 TB of data. The system responds to a request within a few hours. The estimated cost is 1177.5 USD per month.

  If you select the pay-as-you-go billing method, the cost for the computing resources that are used to process 1 TB of data once is about 47.1 USD per day and 1413 USD per month. The prerequisites are that the SQL complexity is 1 and the data is processed once per day. If the data is processed multiple times per day, the cost is multiplied.

  When you migrate data to the cloud for the first time, we recommend that you select the pay-as-you-go billing method first. Perform a Proof of Concept (POC) test to calculate the approximate number of workers used for your jobs. Then, calculate the number of compute units that you need to purchase based on the number of workers.

- **Billing methods for Hadoop users to migrate data to the cloud**

  Assume that a Hadoop cluster has one controller node and five compute nodes. Each node has 32 cores, equivalent to 32 CPUs. The total number of CPUs for the compute nodes is 160. The estimated cost of the cluster is 3768 USD per month with no discounts or promotional offers applied.

  MaxCompute does not require any controller nodes. The performance of MaxCompute is 80% higher than Hive. It frees you from operations and maintenance (O&M), which also reduces costs.

- **Mixed billing methods**

- Subscription billing method for production businesses, such as hourly extract, transform, load (ETL), and pay-as-you-go billing method for aperiodic jobs or ad hoc queries

  We recommend that you select the subscription billing method for periodic computing jobs that are frequently executed and the pay-as-you-go billing method for aperiodic jobs that are used to process large amounts of data. In pay-as-you-go mode, you can choose not to store data. Instead, you can read data from tables under other accounts. This reduces data storage costs. Authorization is required for computing operations on tables under different accounts. For more information, see Create a project-level role.

- Subscription billing method for aperiodic jobs or ad hoc queries and pay-as-you-go billing method for production businesses, such as daily ETL

  Daily data testing may cause the issue of uncontrollable costs. To avoid this issue, you can add data testing and aperiodic jobs to fixed resource groups. Then, use MaxCompute Management to configure custom development groups and business intelligence (BI) groups. If production jobs are executed only once per day, you can add them to a pay-as-you-go resource group.

Switching between billing methods

If you select the subscription billing method, you can upgrade or degrade the configurations in the following scenarios: The data volume changes, and the purchased resources are insufficient or become idle. For more information, see Upgrade or downgrade configurations.

You can also switch between the subscription and pay-as-you-go billing methods. For more information, see Switch billing methods.

> **Note** Before you switch the billing method from pay-as-you-go to subscription, evaluate the computing performance and cycles of jobs to determine the number of compute units you need to purchase. If the compute units you purchase are insufficient, the computing cycle of a job may be prolonged, and the computing performance may not meet your expectations. If this occurs, you may need to switch the billing method again.

6.3. Optimize computing costs

This topic describes how to optimize SQL and MapReduce jobs to reduce computing costs.

You can estimate computing costs before you execute computing jobs. For more information, see TCO tools. You can also configure alerts for resource consumption to avoid extra costs. If computing costs are high, you can use the methods described in this topic to reduce the costs.

Control the computing costs of SQL jobs

Some SQL jobs that trigger full table scans incur high computing costs. The frequent scheduling of SQL jobs may cause an accumulation of jobs, which also increases computing costs. If an accumulation occurs and the pay-as-you-go billing method is used, jobs are queued and require more resources. As a result, the bill generated the next day is abnormally high. You can use the following methods to control the computing costs of SQL jobs:

- Avoid frequent scheduling. MaxCompute provides a computing service to process large amounts of data at a time. It is different from real-time computing services. If SQL jobs are executed at short intervals, computing frequency is increased. The increased computing frequency and improper execution of SQL jobs cause increases in computing costs. If you require frequent scheduling, use CostSQL to estimate the costs of SQL jobs to avoid extra costs.

- Reduce full table scans. You can use the following methods:
  - Specify the required parameters to disable the full table scan feature. You can disable the feature for a session or project.

    ```
    -- Disable the feature for a session.
    set odps.sql.allow.fullscan=false;
    -- Disable the feature for a project.
    SetProject odps.sql.allow.fullscan=false;
    ```

  - Prune columns. Column pruning allows the system to read data only from the required columns. We recommend that you do not use the `SELECT *` statement, which triggers a full table scan.

    ```
    SELECT a,b FROM T WHERE e < 10;
    ```

    In this statement, the T table contains the `a, b, c, d, and e` columns. However, only the a, b, and e columns are read.

  - Prune partitions. Partition pruning allows you to specify filter conditions for partition key columns. This way, the system reads data only from the required partitions. This avoids the errors and waste of resources caused by full table scans.

    ```
    SELECT a,b FROM T WHERE partitiondate='2017-10-01';
    ```

  - Optimize SQL keywords that incur costs. The keywords include JOIN, GROUP BY, ORDER BY, DISTINCT, and INSERT INTO. You can optimize the keywords based on the following rules:
    - Before a JOIN operation, you must prune partitions. Otherwise, a full table scan may be performed. For more information about scenarios in which partition pruning is invalid, see Scenarios where partition pruning does not take effect.
    - Use UNION ALL instead of FULL OUTER JOIN.

      ```
      SELECT COALESCE(t1.id, t2.id) AS id, SUM(t1.col1) AS col1
       , SUM(t2.col2) AS col2
      FROM (
       SELECT id, col1
       FROM table1
      ) t1
      FULL OUTER JOIN (
       SELECT id, col2
       FROM table2
      ) t2
      ON t1.id = t2.id
      GROUP BY COALESCE(t1.id, t2.id);
      -- Optimized:
      SELECT t.id, SUM(t.col1) AS col1, SUM(t.col2) AS col2
      FROM (
       SELECT id, col1, 0 AS col2
       FROM table1
       UNION ALL
       SELECT id, 0 AS col1, col2
       FROM table2
      ) t
      GROUP BY t.id;
      ```

- Try not to include GROUP BY in UNION ALL. Use GROUP BY outside UNION ALL.

```
SELECT t.id, SUM(t.val) AS val
FROM (
 SELECT id, SUM(col3) AS val
 FROM table3
 GROUP BY id
 UNION ALL
 SELECT id, SUM(col4) AS val
 FROM table4
 GROUP BY id
) t
GROUP BY t.id;
Optimized:---------------------------
SELECT t.id, SUM(t.val) AS val
FROM (
 SELECT id, col3 AS val
 FROM table3
 UNION ALL
 SELECT id, col4 AS val
 FROM table4
) t
GROUP BY t.id;
```

- To sort temporarily exported data, sort the data by using tools such as EXCEL instead of ORDER BY.
- Try not to use DISTINCT. Use GROUP BY instead.

```
SELECT COUNT(DISTINCT id) AS cnt
FROM table1;
Optimized:---------------------------
SELECT COUNT(1) AS cnt
FROM (
 SELECT id
 FROM table1
 GROUP BY id
) t;
```

- Try not to use INSERT INTO to write data. Add a partition field instead. This reduces SQL complexity and saves computing costs.

- Try not to execute SQL statements to view table data. You can use the table preview feature to view table data. This method does not consume computing resources. If you use DataWorks, you can preview a table and query details about the table on the **Data Map** page. For more information, see View the details of a table. If you use MaxCompute Studio, double-click a table to preview its data.
- Select an appropriate tool for data computing. MaxCompute responds to a query within minutes. It is not suitable for frontend queries. Computing results are synchronized to an external storage system. Most users use relational databases to store results. We recommend that you use MaxCompute for lightweight computing jobs and relational databases, such as ApsaraDB for RDS, for frontend queries. Frontend queries require the real-time generation of query results. If the query results are displayed in the frontend, no conditional clauses are executed on the data. The data is not aggregated or associated with dictionaries. The queries do not even include the WHERE clause.

Control the computing costs of MapReduce jobs

You can use the following methods to control the computing costs of MapReduce jobs:

- Configure the required settings
  - Split size

    The default split size for a mapper is 256 MB. The split size determines the number of mappers. If your code logic for a mapper is time-consuming, you can use `JobConf#setSplitSize` to reduce the split size. You must configure an appropriate split size. Otherwise, excessive computing resources are required.
  - MapReduce Reduce Instance

    By default, the number of reducers that are used to complete a job is one fourth of the number of mappers. You can set the number of the reducers to a value that ranges from 0 to 2,000. More reducers require more computing resources, which increases costs. You must appropriately configure the number of reducers.
- Reduce the number of MapReduce jobs

  If multiple MapReduce jobs are correlated and the output of a job is the input of the next job, we recommend that you use the pipeline mode. The pipeline mode allows you to merge multiple serial MapReduce jobs into a single job. This reduces redundant disk I/O operations caused by intermediate tables and improves performance. This also simplifies job scheduling and enhances process maintenance efficiency. For more information, see Pipeline examples.
- Prune the columns of input tables

  For input tables that contain a large number of columns, only a few columns are processed by a mapper. When you add an input table, you can specify the columns to reduce the amount of data that needs to be read. For example, to process data in the c1 and c2 columns, use the following configuration:

  ```
  InputUtils.addTable(TableInfo.builder().tableName("wc_in").cols(new String[]{"c1","c2"}).
  build(), job);
  ```

  After the configuration, the mapper reads data only from the c1 and c2 columns. This does not affect the data that is obtained based on column names. However, this may affect the data that is obtained based on subscripts.
- Avoid the duplicate reads of resources

  We recommend that you read resources in the setup stage. This avoids performance loss caused by duplicate resource reads. You can read resources for up to 64 times. For information about, see Resource usage example.
- Reduce the overheads of object construction

  Java objects are used in each map or reduce stage. You can construct Java objects in the setup stage instead of the map or reduce stage. This reduces the overheads of object construction.

```
{
    ...
    Record word;
    Record one;
    public void setup(TaskContext context) throws IOException {
      // Create a Java object in the setup stage. This avoids the repeated creation of Ja
va objects in each map stage.
      word = context.createMapOutputKeyRecord();
      one = context.createMapOutputValueRecord();
      one.set(new Object[]{1L});
    }
    ...
}
```

- Use a combiner in the proper manner

  If the output of a map task contains multiple duplicate keys, you can use a combiner to merge these keys. This reduces transmission bandwidth and shuffling overheads. If the output of a map task does not contain multiple duplicate keys, using a combiner may incur extra overheads. A combiner implements a reducer interface. The following code defines the combiner in a WordCount program:

```
/**
   * A combiner class that combines map output by sum them.
   */
  public static class SumCombiner extends ReducerBase {
    private Record count;
    @Override
    public void setup(TaskContext context) throws IOException {
      count = context.createMapOutputValueRecord();
    }
    @Override
    public void reduce(Record key, Iterator<Record> values, TaskContext context)
        throws IOException {
      long c = 0;
      while (values.hasNext()) {
        Record val = values.next();
        c += (Long) val.get(0);
      }
      count.set(0, c);
      context.write(key, count);
    }
  }
```

- Appropriately select partition key columns or customize a partitioner

  You can use `JobConf#setPartitionColumns` to specify partition key columns. The default partition key columns are defined in the key schema. If you use this method, data is transferred to reducers according to the hash values of the specified columns. This avoids long-tail issues caused by data skew. You can also customize a partitioner if necessary. The following code shows how to customize a partitioner:

```
import com.aliyun.odps.mapred.Partitioner;
public static class MyPartitioner extends Partitioner {
@Override
public int getPartition(Record key, Record value, int numPartitions) {
  // numPartitions indicates the number of reducers.
  // This function is used to determine the reducers to which the keys of map tasks are t
ransferred.
  String k = key.get(0).toString();
  return k.length() % numPartitions;
}
}
```

Configure the following settings in *jobconf*:

```
jobconf.setPartitionerClass(MyPartitioner.class)
```

Specify the number of reducers in *jobconf*.

```
jobconf.setNumReduceTasks(num)
```

- Configure JVM memory parameters as required

  The large memory of a MapReduce job increases computing costs. We recommend that you configure one CPU core and 4 GB of memory for a MapReduce job and set `odps.stage.reducer.jvm.mem` to 4006 for a reducer. A large CPU core-to-memory ratio (greater than 1:4) also increases computing costs.

6.4. Optimize storage costs

This topic describes how to optimize storage costs in terms of data partitions, table lifecycles, and the periodic deletion of deprecated tables.

You can perform the following operations to optimize storage costs:

- Properly configure data partitions.
- Configure reasonable lifecycles for tables.
- Periodically delete deprecated tables.

Properly configure data partitions

In MaxCompute, each value of a partition key column is called a partition. You can group multiple fields of a table in a single partition to create a multi-level partition. Multi-level partitions are similar to multi-level directories. If you specify the name of a partition you want to access, the system reads data only from that partition and does not scan the entire table. This reduces costs and improves efficiency.

- If the minimum period for data collection is one day, we recommend that you use the date field as a partition field. The system migrates data to the specified partitions every day. Then, it reads the data from the specified partitions for subsequent operations.
- If the minimum period for data collection is one hour, we recommend that you use the combination of the date and hour fields as a partition field. The system migrates data to the specified partitions every hour. Then, it reads the data from the specified partitions for subsequent operations. If data that is collected on an hourly basis is partitioned based on dates, data in each partition is appended every hour. As a result, the system reads large amounts of unnecessary data, which increases storage costs.

You can use partition fields based on your business needs. In addition to the date and time fields, you can use other fields that have a relatively fixed number of enumerated values, such as channel, country, or province. Alternatively, you can use a combination of time and other fields as a partition field. We recommend that you specify two levels of partitions in a table. Each table supports a maximum of 60,000 partitions.

Configure reasonable lifecycles for tables

When you create a table, you can configure its lifecycle based on data usage. MaxCompute deletes data that exceeds the lifecycle threshold in a timely manner. This saves storage space.

For example, you can execute the following statement to create a table with the lifecycle of 100 days. If the last modification of the table or a partition occurred more than 100 days ago, MaxCompute deletes the table or partition.

```
CREATE TABLE test3 (key boolean) PARTITIONED BY (pt string, ds string) LIFECYCLE 100;
```

The lifecycle takes a partition as the smallest unit. If some partitions in a partitioned table reach the lifecycle threshold, these partitions are deleted. Partitions that do not reach the lifecycle threshold are not affected.

You can execute the following statement to modify the lifecycle settings for an existing table. For more information, see Lifecycle management operations.

```
ALTER TABLE table_name SET lifecycle days;
```

Periodically delete deprecated tables

We recommend that you periodically delete deprecated tables that are not accessed for a long period of time. The following tables are considered deprecated tables:

- Tables that are not accessed within the last three months
- Non-partitioned tables that are not accessed within the last month
- Tables that do not consume storage resources

6.5. Optimize the costs of data uploads and downloads

This topic describes how to optimize the synchronization costs incurred by data uploads and downloads.

- Use the classic network or a Virtual Private Cloud (VPC)

  You can use an internal network, such as the classic network or VPC, to upload or download data at no cost. For more information about how to configure networks, see Endpoints.

- Use Elastic Compute Service (ECS) to download resources

  If you create a subscription ECS instance, you can use a data synchronization tool such as Tunnel to synchronize data from MaxCompute to the ECS instance. Then, download the data to your local directory. For more information, see Export SQL execution results.

- Optimize Tunnel-based file uploads

Separate uploads of small files consume too many computing resources. We recommend that you upload a large number of small files at a time. For example, if you call Tunnel SDK, we recommend that you upload files when the cache of the files reaches 64 MB.

- Estimate the VPC bandwidth

  If you want to synchronize data from your on-premises data center to MaxCompute by using a physical connection, you must estimate the bandwidth and costs of data synchronization. For example, if you migrate 50 TB of data to MaxCompute, the estimated bandwidth for one day is 5 Gbit/s. The estimated bandwidth is calculated by using the following formula:

  ```
  50 × 1024 × 8/(24 × 3600) = 4.7 Gbit/s
  ```

6.6. Manage costs

This topic describes how to track resource consumption, optimize resource usage, and reduce costs.

You can use the following items to manage costs:

- Bill details: You can view bill details on the Billing Management page of the Alibaba Cloud Management Console.
- Usage records: Each usage record contains the complexity and metering information of an SQL statement, as well as details about daily storage and download traffic.
- Command-line interface (CLI): You can use a CLI to reproduce operation scenarios and determine the causes of high costs incurred by SQL statements.

Bill details

We recommend that you regularly view your bills to optimize costs in a timely manner. You can view bill details in the Alibaba Cloud Management Console. If you select the subscription billing method, bills are generated at 12:00 the next day. If you select the pay-as-you-go billing method, bills are generated at 09:00 the next day. For more information, see View billing details.

Usage records

If the bill amount of a project reaches thousands of dollars on a given day and is a multiple of the normal bill amount, you must view the bill details. You can download usage records to view details about exception records. For more information, see View billing details.

Metering information about a storage fee is pushed every hour. To calculate a storage fee, obtain the total number of bytes and calculate the average value over a 24-hour period. Then, use the tiered pricing method to obtain the storage fee.

The calculation of metering information depends on the end time of each task. If a task is complete in the early morning of the next day after it starts, the metering information of this task is included in the calculation for the day the task is complete.

You are not charged for the resources that are used to download data over an internal network, such as the classic network. The resources that are used to upload data are also free of charge. You are charged only for the resources that are used to download data over the Internet.

CLI

If an abnormal SQL statement is detected, you can use a CLI to reproduce the operation scenario.

- You can check usage records or run the `show p;` command to obtain the ID of the instance on which abnormal data is detected. Then, run the `wait InstanceId` command to obtain the Logview

URL of the instance. The logs of the SQL statement are displayed in Logview. You can view the logs to determine the causes of high costs.

> **Note** You can obtain information generated only in the last seven days in Logview.

- You can also run the `desc instance instid` command to show information about the SQL statement in the console.

6.7. Command reference

This topic describes the commonly used commands and whether these commands are free of charge.

Command	Description	Charged?	Example
TUNNEL DOWNLOAD	Download data over the classic network.	No	`TUNNEL DOWNLOAD table_name e:/table_name.txt;` Configure the endpoint of MaxCompute in the classic network: `http://dt.cn-shanghai.maxcompute.aliyun-inc.com` . For more information, see Endpoints.

TUNNEL DOWNLOAD	Download data over the Internet.	Yes	`TUNNEL DOWNLOAD table_name e:/table_name.txt;`
			`Configure the public endpoint of MaxCompute: http://dt.cn-shanghai.maxcompute.aliyun.com.`
			For more information, see Endpoints.

TUNNEL UPLOAD	Upload data.	No	`TUNNEL UPLOAD e:/table_name.txt table_name;`
COST SQL	Estimate costs.	No	`COST SQL SELECT * FROM table_name;`

INSERT OVERWRITE...SELECT	Update data.	Yes	`INSERT OVERWRITE TABLE table_name PARTITION (sale_date='20180122') SELECT shop_name, customer_id, total_price FROM sale_detail;`
DESC TABLE	Query table information.	No	`DESC table_name;`
DROP TABLE	Delete a table.	No	`DROP TABLE if exists table_name;`
CREATE TABLE	Create a table.	No	`CREATE TABLE if not exists table_name (key string ,value bigint) PARTITIONED BY(p string);`
CREATE TABLE...SELECT	Create a table.	Yes	`CREATE TABLE if not exists table_name AS SELECT * FROM a_tab;`
INSERT INTO TABLE...VALUES	Quickly insert constant data.	No	`INSERT INTO TABLE table_name partition (p)(key,p) VALUES ('d','20170101'), ('e','20170101'), ('f','20170101');`
INSERT INTO TABLE...SELECT	Insert data.	Yes	`INSERT INTO TABLE table_name SELECT shop_name, customer_id, total_price FROM sale_detail;`
SELECT UDF [NOT COUNT or All] FROM TABLE	Query table data.	Yes	`SELECT sum(a) FROM table_name;`
SET FLAG	Configure session settings.	No	`SET odps.sql.allow.fullscan=true;`
JAR MR	Execute a MapReduce job.	Yes	`JAR -l com.aliyun.odps.mapred.example.WordCount wc_in wc_out`

ADD JAR/FILE/ARCHIVE/TABLE	Add a resource.	No	`ADD jar data\resources\mapreduce-examples.jar -f;`
DROP JAR/FILE/ARCHIVE/TABLE	Remove a resource.	No	`DROP RESOURCE sale.res`
LIST RESOURCES	Query resources.	No	`LIST RESOURCES;`
GET RESOURCES	Download resources.	No	`GET RESOURCES odps-udf-examples.jar d:\;`
CREATE FUNCTIONS	Create a function.	No	`CREATE FUNCTION test_lower ;`
DROP FUNCTIONS	Delete a function.	No	`DROP FUNCTION test_lower;`
LIST FUNCTIONS	Query functions.	No	`LIST FUNCTIONS;`
ALTER TABLE...DROP PARTITION	Delete a partition from a table.	No	`ALTER TABLE user DROP if exists partition(region='hangzhou',dt='20150923');`
TRUNCATE TABLE	Remove all records from a non-partitioned table.	No	`TRUNCATE TABLE table_name;`
CREATE EXTERNAL TABLE	Create an external table.	No	`CREATE EXTERNAL TABLE IF NOT EXISTS ambulance_data_csv_external…LOCATION 'oss://oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/Demo/'`
SELECT [EXTERNAL] TABLE	Read an external table.	Yes	`SELECT recordId, patientId, direction FROM ambulance_data_csv_external WHERE patientId > 25;`
SHOW TBALES	Query all tables in the current project.	No	`SHOW TABLES;`

SHOW PARTITIONS table_name	Query all partitions of a table.	No	`SHOW PARTITIONS <table_name>`
SHOW INSTANCE/SHOW P	Show information about the instance that the current user creates.	No	`SHOW INSTANCES/SHOW P`
WAIT INSTANCE	Return the Logview URL of the specified instance.	No	`WAIT 20131225123302267gk3u6k4y2`
STATUS INSTANCE	Return the status of the specified instance.	No	`STATUS 20131225123302267gk3u6k4y2`
KILL INSTANCE	Stop the specified instance.	No	`KILL 20131225123302267gk3u6k4y2`

6.8. Analyze MaxCompute bills

If you want to know the distribution of your expenses or prevent MaxCompute costs from increasing beyond expectations, you can obtain and analyze your MaxCompute bills. The analysis helps maximize resource utilization and reduce costs. This topic describes how to analyze the cost distribution of MaxCompute based on usage records of bills.

Background information

MaxCompute is a big data analytics platform. Computing resources of MaxCompute support two types of billing methods: subscription and pay-as-you-go. You are charged based on MaxCompute projects on a daily basis, and daily bills are generated before 06:00 of the next day. For more information about billable items and billing methods of MaxCompute, see Billing method.

Alibaba Cloud provides information about the MaxCompute bill fluctuations (cost increases in most cases) during data development or before a version release of MaxCompute. You can analyze the bill fluctuations and optimize jobs in your MaxCompute projects based on the analysis results. You can download the usage records of all commercial services on the Billing Management page in the Alibaba Cloud Management Console. For more information about how to obtain and download bills, see View billing details.

Upload usage records of bills to MaxCompute

1. Create a MaxCompute table that is named maxcomputefee on the MaxCompute client odpscmd. Sample statements:

```
DROP TABLE IF EXISTS maxcomputefee;
CREATE TABLE IF NOT EXISTS maxcomputefee
(
    projectid STRING COMMENT 'ProjectId'
    ,feeid STRING COMMENT 'MeteringId'
    ,type STRING COMMENT 'MeteringType, such as Storage, ComputationSQL, or DownloadEx'
    ,storage BIGINT COMMENT 'Storage'
    ,endtime STRING COMMENT 'EndTime'
    ,computationsqlinput BIGINT COMMENT 'SQLInput(Byte)'
    ,computationsqlcomplexity DOUBLE COMMENT 'SQLComplexity'
    ,uploadex BIGINT COMMENT 'UploadEx'
    ,download BIGINT COMMENT 'DownloadEx(Byte)'
    ,cu_usage DOUBLE COMMENT 'MRCompute(Core*Second)'
    ,input_ots BIGINT COMMENT 'InputOTS(Byte)'
    ,input_oss BIGINT COMMENT 'InputOSS(Byte)'
    ,starttime STRING COMMENT 'StartTime'
    ,source_type STRING COMMENT 'SpecificationType'
    ,source_id STRING COMMENT 'DataWorksNodeID'
);
```

Fields of usage records:

- ProjectId: a MaxCompute project of your Alibaba Cloud account or a MaxCompute project of the Alibaba Cloud account to which the current RAM user belongs.
- MeteringId: a billing ID, which indicates the ID of a storage task, an SQL computing task, an upload task, or a download task. The ID of an SQL computing task is specified by InstanceId, and the ID of an upload or download task is specified by Tunnel SessionId.
- MeteringType: a billing type. Valid values: Storage, ComputationSql, UploadIn, UploadEx, DownloadIn, and DownloadEx.
- Storage: the amount of data that is read per hour. Unit: bytes.
- StartTime or EndTime: the time when a job started to run or the time when a job stopped. Only storage data is obtained on an hourly basis.
- SQLInput(Byte): the SQL computation item. This field specifies the amount of input data each time an SQL statement is executed. Unit: bytes.
- SQLComplexity: the complexity of SQL statements. This field is one of the SQL billing factors.
- UploadEx or DownloadEx(Byte): the amount of data that is uploaded or downloaded over the Internet. Unit: bytes.
- MRCompute(Core*Second): the billable hours of a MapReduce job or a Spark job, which are calculated by using the following formula: `Number of cores × Number of seconds`. After calculation, you must convert the calculation result into billable hours.
- InputOTS(Byte) or InputOSS(Byte): the amount of data that is read from Tablestore or OSS by using external tables. Unit: bytes. These fields are used when fees for external tables are generated.

2. Run Tunnel commands to upload usage records of bills.

   To upload a CSV file that contains usage records of bills to MaxCompute, you must make sure that the number and data types of columns in the CSV file are the same as the number and data types of columns in the maxcomputefee table. Otherwise, the data upload fails.

```
tunnel upload ODPS_2019-01-12_2019-01-14.csv maxcomputefee -c "UTF-8" -h "true" -dfp "y
yyy-MM-dd HH:mm:ss";
```

```
odps@ sz_mc>tunnel upload /Users/yangyi/Desktop/ODPS_2019-01-12_2019-01-14.csv maxcomputefee -c "UTF-8" -h "true" -dfp "yyyy-MM-dd HH:mm:ss";
Upload session: 201901151930039d37dc0b00015e67
Start upload:/Users/yangyi/Desktop/ODPS_2019-01-12_2019-01-14.csv
Using \n to split records
Upload in strict schema mode: true
Total bytes:423234       Split input to 1 blocks
2019-01-15 19:30:03     scan block: '1'
2019-01-15 19:30:03     scan block complete, blockid=1
2019-01-15 19:30:03     upload block: '1'
2019-01-15 19:30:04     upload block complete, blockid=1
upload complete, average speed is 206.7 KB/s
OK
```

> **Note**
>
> - For more information about the configurations of Tunnel commands, see Tunnel commands.
> - You can also upload usage records of bills by using the data import feature of DataWorks. For more information, see Import data by using Data Integration.

3. Execute the following statement to check whether all usage records are uploaded:

```
SELECT * FROM maxcomputefee limit 10;
```

```
odps@ sz_mc>select * from maxcomputefee limit 10;

ID = 201901
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.odps.[redacted]TGl0dE5VPSxPRFBTX09CTzoxNDI3NDgwMzM4OTYzMTQ0LDE1[redacted]
wNzIseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXS[redacted]cy8yMDE5MDExNjAyMjExMjYwNGc4cTIwMzkyIl19XSwiVmVyc2lvbi
Q==
Job Queueing.
+------------+------------+------------+------------+------------+------------+--------------------+--------------------------+------------+------------+------------+------------+------------+
| projectid  | feeid      | type       | starttime  | storage    | endtime    | computationsqlinput | computationsqlcomplexity | uploadex   | download   | cu_usage   | input_ots  | input_oss  |
+------------+------------+------------+------------+------------+------------+--------------------+--------------------------+------------+------------+------------+------------+------------+
|[redacted]| 13911887[redacted]| Storage    | 2019-01-12 00:08:47 | 4940136534555 | 2019-01-12 01:08:47 | NULL               | NULL                     | NULL       | NULL       |
L[redacted]| NULL
|[redacted]| 20190[redacted]| ComputationSql | 2019-01-12 00:15:34 | NULL       | 2019-01-12 00:15:36 | 11533312           | 1.0                      | NULL       | NULL       |
```

Use SQL statements to analyze usage records of bills

1. Analyze the costs of SQL jobs. MaxCompute SQL meets the business requirements of 95% of cloud users. The generated fees also occupy a large proportion of MaxCompute bills.

> **Note** Costs of an SQL job = Amount of input data × Complexity of SQL statements × Unit price (USD 0.0438/GB)

```
-- Sort SQL jobs based on sqlmoney to analyze the costs of SQL jobs.
SELECT  to_char(endtime,'yyyymmdd') as ds,feeid as instanceid
        ,projectid
        ,computationsqlcomplexity  -- SQL complexity
        ,SUM((computationsqlinput / 1024 / 1024 / 1024)) as computationsqlinput  -- Amo
unt of input data (GB)
        ,SUM((computationsqlinput / 1024 / 1024 / 1024)) * computationsqlcomplexity * 0
.0438 AS sqlmoney
FROM    maxcomputefee
WHERE   TYPE = 'ComputationSql'
AND to_char(endtime,'yyyymmdd') >= '20190112'
GROUP BY to_char(endtime,'yyyymmdd'),feeid
         ,projectid
         ,computationsqlcomplexity
ORDER BY sqlmoney DESC
LIMIT   10000
;
```

The following conclusions can be drawn from the execution result:

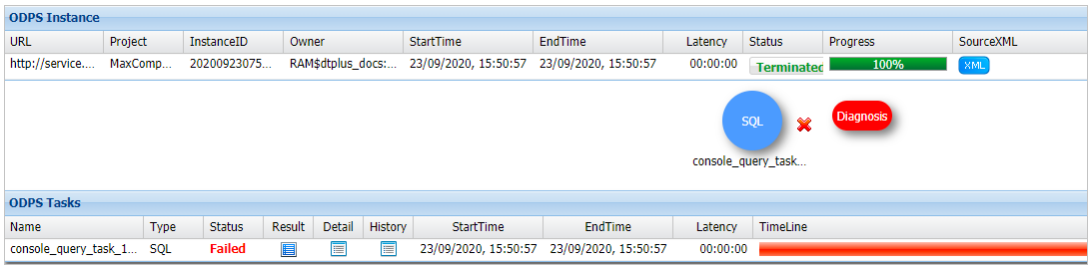
- To reduce the costs of large jobs, you can reduce the amount of data that you want to read and the complexity of SQL statements.
- You can summarize daily data based on the ds field and analyze the trend of the costs of SQL jobs in a specified period of time. For example, you can create a line chart in an EXCEL file or by using tools, such as Quick BI, to display the trend.
- You can perform the following steps to locate the node that you want to optimize based on the execution result:
  a. Obtain the ID of a job instance.

     Run the `wait InstanceId;` command on the MaxCompute client odpscmd or in the DataWorks console to view the information about a specific job and the related SQL statements.

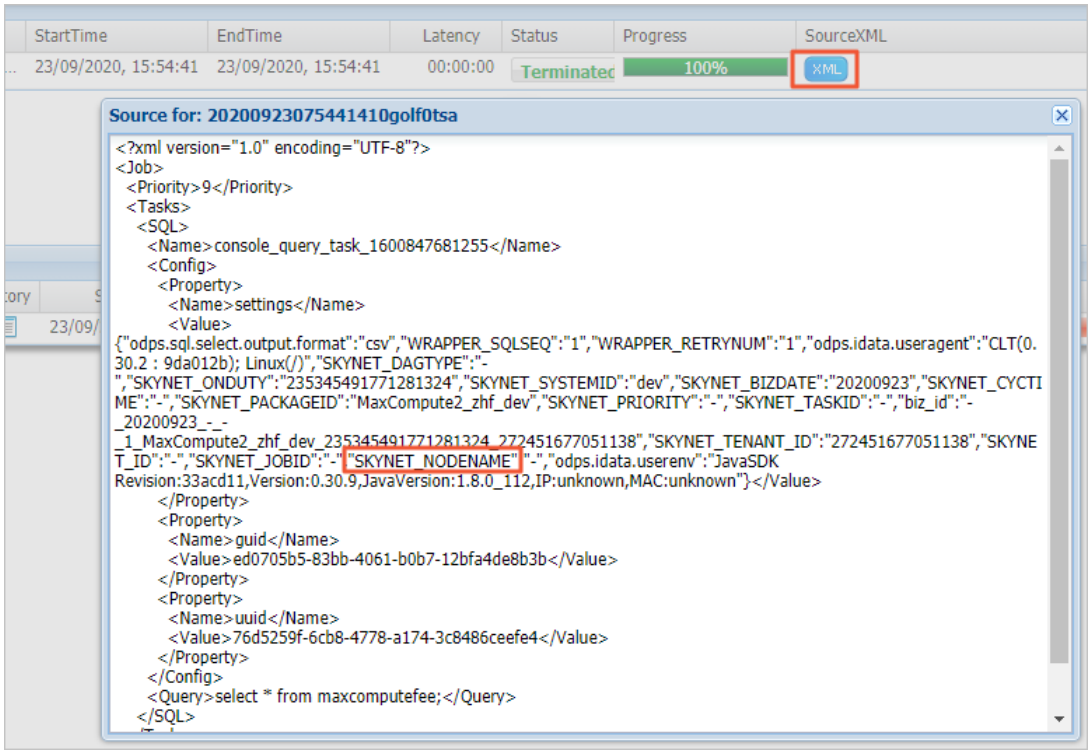
  b. Enter the returned Logview URL in a web browser and press Enter to view the information about the SQL job.

     For more information about how to use Logview to view information about jobs, see Use Logview to view job information.

c. Obtain the name of the DataWorks node from Logview.

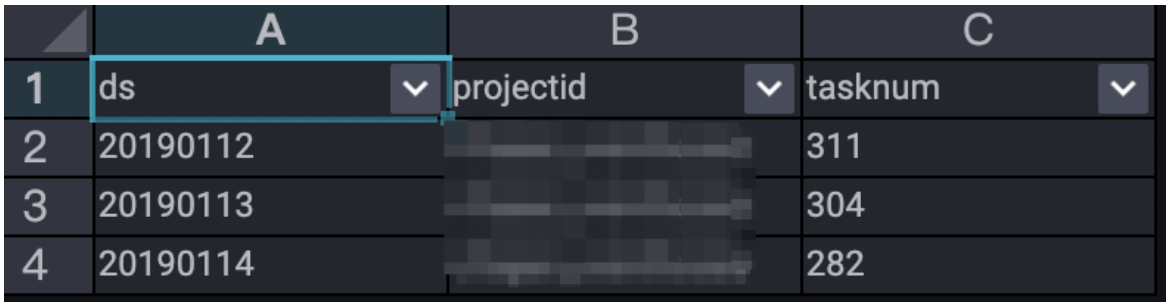
In Logview, find the job whose information you want to view and click XML in the SourceXML column to view the job details. In the following figure, SKYNET_NODENAME indicates the name of the DataWorks node. This parameter is displayed only for the jobs that are run by the scheduling system. This parameter is left empty for ad hoc queries. After you obtain the node name, you can quickly locate the node in the DataWorks console to optimize the node or view the node owner.



2. Analyze the trend of the number of jobs. In most cases, a surge in the number of jobs due to repeated operations or invalid settings of scheduling attributes results in cost increases.

```
-- Analyze the trend of the number of jobs.
SELECT  TO_CHAR(endtime,'yyyymmdd') AS ds
        ,projectid
        ,COUNT(*) AS tasknum
FROM    maxcomputefee
WHERE   TYPE = 'ComputationSql'
AND     TO_CHAR(endtime,'yyyymmdd') >= '20190112'
GROUP BY TO_CHAR(endtime,'yyyymmdd')
         ,projectid
ORDER BY tasknum DESC
LIMIT   10000
;
```

The following figure shows the execution result.



	A	B	C
1	ds	projectid	tasknum
2	20190112		311
3	20190113		304
4	20190114		282

The execution result shows the trend of the number of jobs that were submitted to MaxCompute and were successfully run from January 12, 2019 to January 14, 2019.

3. Analyze storage costs.

```
-- Analyze storage costs.
SELECT  t.ds
        ,t.projectid
        ,t.storage
        ,CASE    WHEN t.storage < 0.5 THEN 0.01
                 WHEN t.storage >= 0.5 AND t.storage <= 10240 THEN t.storage*0.0072
                 WHEN t.storage > 10240 AND t.storage <= 102400 THEN (10240*0.0072+(t.s
torage-10240)*0.006)
                 WHEN t.storage > 102400  THEN (10240*0.0072+(102400-10240)*0.006+(t.st
orage-102400)*0.004)
        END storage_fee
FROM    (
            SELECT  to_char(starttime,'yyyymmdd') as ds
                    ,projectid
                    ,SUM(storage/1024/1024/1024)/24 AS storage
            FROM    maxcomputefee
            WHERE   TYPE = 'Storage'
            and to_char(starttime,'yyyymmdd') >= '20190112'
            GROUP BY to_char(starttime,'yyyymmdd')
                     ,projectid
        ) t
ORDER BY storage_fee DESC
;
```

The following figure shows the execution result.

	A	B	C	D
1	ds	projectid	storage	storage_fee
2	20190113		8467.066039182557	73.31215472913347
3	20190112		4527.7404568577185	40.22181983760483
4	20190114		2672.068578289938	24.63417605763548

The following conclusions can be drawn from the execution result:

- Storage costs increased on January 12, 2019 and decreased on January 14, 2019.
- To reduce storage costs, we recommend that you configure a lifecycle for tables and delete unnecessary temporary tables.

4. Analyze download costs.

For Internet-based data downloads or cross-region data downloads in your MaxCompute project, you are charged based on the amount of data that is downloaded.

> **Note** Costs of a download job = Amount of downloaded data × Unit price (USD 0.1166/GB)

```
-- Analyze download costs.
SELECT  TO_CHAR(starttime,'yyyymmdd') AS ds
        ,projectid
        ,SUM((download/1024/1024/1024)*0.1166) AS download_fee
FROM    maxcomputefee
WHERE   type = 'DownloadEx'
AND     TO_CHAR(starttime,'yyyymmdd') >= '20190112'
GROUP BY TO_CHAR(starttime,'yyyymmdd')
         ,projectid
ORDER BY download_fee DESC
;
```

5. Analyze the costs of MapReduce jobs.

> **Note** Computing fees for MapReduce jobs on the day = Total billable hours on the day × Unit price (USD 0.0690/Hour/Job)

```
-- Analyze the costs of MapReduce jobs.
SELECT  TO_CHAR(starttime,'yyyymmdd') AS ds
        ,projectid
        ,(cu_usage/3600)*0.0690 AS mr_fee
FROM    maxcomputefee
WHERE   type = 'MapReduce'
AND     TO_CHAR(starttime,'yyyymmdd') >= '20190112'
GROUP BY TO_CHAR(starttime,'yyyymmdd')
         ,projectid
         ,cu_usage
ORDER BY mr_fee DESC
;
```

6. Analyze the costs of jobs that involve Tablestore external tables or OSS external tables.

> **Note** Computing fees for an SQL job that involves external tables = Amount of input data × Unit price (USD 0.0044/GB).

```
-- Analyze the costs of SQL jobs that involve Tablestore external tables.
SELECT  TO_CHAR(starttime,'yyyymmdd') AS ds
        ,projectid
        ,(input_ots/1024/1024/1024)*1*0.0044 AS ots_fee
FROM    maxcomputefee
WHERE   type = 'ComputationSql'
AND     TO_CHAR(starttime,'yyyymmdd') >= '20190112'
GROUP BY TO_CHAR(starttime,'yyyymmdd')
         ,projectid
         ,input_ots
ORDER BY ots_fee DESC
;
-- Analyze the costs of SQL jobs that involve OSS external tables.
SELECT  TO_CHAR(starttime,'yyyymmdd') AS ds
        ,projectid
        ,(input_oss/1024/1024/1024)*1*0.0044 AS ots_fee
FROM    maxcomputefee
WHERE   type = 'ComputationSql'
AND     TO_CHAR(starttime,'yyyymmdd') >= '20190112'
GROUP BY TO_CHAR(starttime,'yyyymmdd')
         ,projectid
         ,input_oss
ORDER BY ots_fee DESC
;
```

7. Analyze the costs of Spark jobs.

> **Note** Computing fees for Spark jobs on the day = Total billable hours on the day × Unit price (USD 0.1041/Hour/Job)

```
-- Analyze the costs of Spark jobs.
SELECT  TO_CHAR(starttime,'yyyymmdd') AS ds
        ,projectid
        ,(cu_usage/3600)*0.1041 AS mr_fee
FROM    maxcomputefee
WHERE   type = 'spark'
AND     TO_CHAR(starttime,'yyyymmdd') >= '20190112'
GROUP BY TO_CHAR(starttime,'yyyymmdd')
         ,projectid
         ,cu_usage
ORDER BY mr_fee DESC
;
```

7.Security management

7.1. Set a RAM user as the super administrator for a MaxCompute project

This topic describes how to set a RAM user as the super administrator for a MaxCompute project, and provides suggestions on how to manage members and permissions.

Background information

To ensure data security, the Alibaba Cloud account of a project is used only by authorized personnel. Common users can only log on to MaxCompute as RAM users. A project owner must be the Alibaba Cloud account, and some operations can only be performed by the project owner, such as setting a project flag and configuring cross-project resource sharing by using packages. If you use a RAM user, make sure that it has been granted the super administrator role.

The built-in management role Super_Administrator has been added to MaxCompute. This role has permissions on all types of resources in a project and project management permissions. For more information about permissions, see Role planning and management.

A project owner can grant the Super_Administrator role to a RAM user. As a super administrator, the RAM user has the permissions needed to manage the project, such as common project flag setting permissions and permissions on managing all resources.

Authorization methods

We recommend that you grant the Super_Administrator role to a RAM user that has the permissions to create a project. This way, the RAM user can manage both DataWorks workspaces and MaxCompute projects that are associated with these DataWorks workspaces.

> **Note**
>
> - For information about how to authorize a RAM user to create projects, see Grant a RAM user the permissions to perform operations in the DataWorks console.
> - To ensure data security, we recommend that you clarify the responsibilities of owners of RAM users. Make sure that each RAM user belongs to one developer.
> - Only one RAM user can be granted the Super_Administrator role in a project. You can grant the Admin role to other RAM users that require basic management permissions.

After you select a RAM user and use the RAM user to create a project, the project owner is still the Alibaba Cloud account, who can grant the Super_Administrator role to the RAM user in the following ways:

- Grant the Super_Administrator role on the MaxCompute client.

  Assume that user bob@aliyun.com is the owner of the project_a project, and user Allen is a RAM user under bob@aliyun.com.

  i. Run the following commands to grant the Super_Administrator and Admin roles as user

bob@aliyun.com:

```
-- Open project_a.
use project_a;
-- Add the RAM user Allen to project_a.
add user ram$bob@aliyun.com:Allen;
-- Grant the Super_Administrator role to Allen.
grant super_administrator TO ram$bob@aliyun.com:Allen;
-- Grant the Admin role to Allen.
grant admin TO ram$bob@aliyun.com:Allen;
```

ii. Run the following command to view the permissions as the authorized RAM user:

```
show grants;
```

If the Super_Administrator role is in the command output, the authorization succeeded.

- Grant the Super_Administrator role in the DataWorks console.
  i. Log on to the DataWorks console and choose Workspace Management.
  ii. Optional. Add a RAM user as a project member. Skip this step if the RAM user is already a project member.
    a. In the left-side navigation pane, click **User Management**.
    b. In the upper-right corner, click **Add Member**.
    c. In the **Add Member** dialog box, select the members you want to add from the **Accounts to be added** section and click the rightwards arrow to add them to the **Added account** section.

    > **Note** In the note block, click **Refresh** to synchronize the RAM users under the current Alibaba Cloud account to the **Account to be added** section.

    d. Select the required roles and click **Confirm**.
  iii. Grant the Super_Administrator role to the RAM user.
    a. In the left-side navigation pane, click **MaxCompute Management**.
    b. In the navigation tree, click **Custom user roles**.

c. Find the role that you want to grant to the user and click **Member management** in the Operation column. In the Member management dialog box, select the members you want to add from the **Account to be added** section and click the rightwards arrow to add them to the **Added account** section.

d. Click **Confirm**.

iv. Run the following command to view the permissions as the authorized RAM user:

```
show grants;
```

If the Super_Administrator role is in the command output, the authorization succeeded.

Usage notes

- Member management
  - MaxCompute supports the Alibaba Cloud account and RAM users. To ensure data security, we recommend that you only add RAM users under the project owner as project members.

    The Alibaba Cloud account is used to control RAM users, such as revoking or updating their credentials. This ensures data security in the case of personnel transfers and resignations.

    > **Note** If you use DataWorks to manage project members, you can add only RAM users under the project owner as project members.

  - RAM users can be added by the Alibaba Cloud account and the super administrator. If you want to add RAM users to a project as the super administrator, wait until the RAM users are created by the Alibaba Cloud account.
  - We recommend that you only add the users who need to develop data, namely, users who need to run jobs, in the current project as project members. For users who require data interactions, you can use packages to share resources across projects. This reduces the complexity of member management because fewer members are added to the project.
  - If an employee who has a RAM user is transferred to another position or resigns, the RAM user with the Super_Administrator role needs to remove the RAM user of the employee from the project, and then notify the project owner to revoke its credentials. If an employee who has a RAM user with the Super_Administrator role is transferred to another position or resigns, the Alibaba Cloud account must be used to remove the RAM user and revoke its credentials.
- Permission management

- We recommend that you manage permissions by role. Permissions are associated with roles, and roles are associated with users.
- We recommend that you use the principle of least privilege to avoid security risks caused by excessive permissions.
- If you need to use cross-project data, we recommend that you share resources by using packages. In this way, resource providers only need to manage packages, which avoids the extra costs caused by the management of additional members.

> **Note** A RAM user who has been granted the Super_Administrator role has the permissions to query and manage all resources in a project. Therefore, no additional permissions need to be granted to the RAM user.

- Permission auditing

  You can use the view provided by the MaxCompute metadata service to audit permissions. For more information, see Metadata views.

- Cost management

  For more information, see View billing details. RAM users can query the billing details only after the Alibaba Cloud account grants them the permissions to access Billing Management. For information about how to grant permissions, see Grant permissions to a RAM role. The following permissions are required:

  - AliyunBSSFullAccess: the permissions to manage Billing Management.
  - AliyunBSSReadOnlyAccess: the access and read-only permissions on Billing Management.
  - AliyunBSSOrderAccess: the permissions to view, pay for, and cancel orders in Billing Management.

  > **Note** Permissions on Billing Management are independent of the Super_Administrator role of a MaxCompute project. You must grant these permissions separately to the user.

- Resource usage management
  - If you use subscription computing resources of MaxCompute, you can view the usage of computing resources and manage all the computing resources on MaxCompute Management. For more information, see Use MaxCompute Management.
  - If you use pay-as-you-go computing resources of MaxCompute, you can view the usage of computing resources in the view provided by the MaxCompute metadata service. For example, TASKS_HISTORY allows you to view the execution details of audit jobs, such as the time, content, and resource consumption. For more information, see TASKS_HISTORY.

    > **Note** The view provided by the metadata service only retains data generated in the last 15 days. If you need to store data for a longer period of time, we recommend that you regularly read and save the data locally.

7.2. Policy-based permission management for users assigned built-in roles

If a user is assigned a built-in role in MaxCompute, the user has the permissions of the built-in role. For example, if a user is assigned the Development role, the user is granted the operation permissions on tables and resources. In actual business scenarios, you may need to manage the operation permissions of such users in a fine-grained manner. For example, you may need to prohibit the users from deleting important tables. This topic describes how to perform policy-based permission management for users assigned built-in roles.

Prerequisites

The MaxCompute client is installed. For more information, see Install and configure the MaxCompute client.

Context

If a user is assigned a built-in role and you want to manage the permissions of the user in a fine-grained manner, we recommend that you use the policy-based permission management mechanism instead of the access control list (ACL) mechanism. For more information about built-in roles, see Users and roles. For more information about the policy-based permission management mechanism, see Policy-based access control and download control.

The policy-based access control mechanism is used to manage permissions based on roles. This mechanism allows you to grant or revoke operation permissions on project objects, such as tables, for roles. The operations include read and write operations. After you assign a role to a user, the permissions granted to or revoked from the role also take effect on the user. For more information about the GRANT and REVOKE syntax, see Policy-based access control and download control.

Grant permissions by using the policy-based access control mechanism

In the following example, the RAM user Alice is assigned the Development role of a MaxCompute project. You need to prohibit the RAM user Alice from deleting all tables whose names start with tb_. The RAM user Alice belongs to the Alibaba Cloud account Bob@aliyun.com.

This operation can be performed only by the project owner or users assigned the Super_Administrator or Admin role.

1. Start the MaxCompute client.
2. Execute the `CREATE ROLE` statement to create a role named delete_test.

   Sample statement:

   ```
   create role delete_test;
   ```

   For more information about how to create a role, see Role planning and management.
3. Execute the `GRANT` statement to grant the delete_test role the permission that prohibits the role from deleting all tables whose names start with tb_.

   Sample statement:

```
grant drop on table tb_* to role delete_test privilegeproperties("policy" = "true", "allow"="false");
```

For more information about the GRANT syntax, see the "Policy-based access control by using the GRANT statement" section in Policy-based access control and download control.

4. Execute the `GRANT` statement to assign the delete_test role to the RAM user Alice.

   Sample statement:

   ```
   grant delete_test to ram$bob@aliyun.com:Alice;
   ```

   If you do not know the Alibaba Cloud account to which the RAM user belongs, you can execute the `LIST USERS;` statement on the MaxCompute client to obtain the account. For more information about how to assign a role to a user, see Role planning and management.

5. Execute the `SHOW GRANTS` statement to view the permissions of the RAM user Alice.

   Sample statement:

   ```
   show grants for ram$bob@aliyun.com:Alice;
   ```

   The following results are returned:

   ```
   [roles]
   role_project_admin, delete_test                                  -- Alice is assigned th
   e delete_test role.
   Authorization Type: Policy                                       -- The authorization me
   thod is Policy.
   [role/delete_test]
   D       projects/mcproject_name/tables/tb_*: Drop                -- Alice is not allowed
   to delete the tables whose names start with tb_ in the project. D indicates Deny.
   [role/role_project_admin]
   A       projects/mcproject_name: *
   A       projects/mcproject_name/instances/*: *
   A       projects/mcproject_name/jobs/*: *
   A       projects/mcproject_name/offlinemodels/*: *
   A       projects/mcproject_name/packages/*: *
   A       projects/mcproject_name/registration/functions/*: *
   A       projects/mcproject_name/resources/*: *
   A       projects/mcproject_name/tables/*: *
   A       projects/mcproject_name/volumes/*: *
   Authorization Type: ObjectCreator
   AG      projects/mcproject_name/tables/local_test: All
   AG      projects/mcproject_name/tables/mr_multiinout_out1: All
   AG      projects/mcproject_name/tables/mr_multiinout_out2: All
   AG      projects/mcproject_name/tables/ramtest: All
   AG      projects/mcproject_name/tables/wc_in: All
   AG      projects/mcproject_name/tables/wc_in1: All
   AG      projects/mcproject_name/tables/wc_in2: All
   AG      projects/mcproject_name/tables/wc_out: All
   ```

   For more information about how to view user permissions, see Query permissions by using MaxCompute SQL.

6. Log on to the MaxCompute client as Alice and execute the `DROP TABLE` statement to delete the

tables whose names start with tb_.

Sample statement:

```
drop table tb_test;
```

The following results are returned. The results indicate that the permission takes effect. If the tables are deleted, the permission does not take effect. In this case, you must check whether the preceding steps are correctly performed.

```
FAILED: Catalog Service Failed, ErrorCode: 50, Error Message: ODPS-0130013:Authorizatio
n exception - Authorization Failed [4011],
You have NO privilege 'odps:Drop' on {acs:odps:*:projects/mcproject_name/tables/tb_test
}.
Explicitly denied by policy.
Context ID:85efa8e9-40da-4660-bbfd-b503dfa64c0a. --->Tips: Pricipal:RAM$bob@aliyun.com:
Alice; Deny by policy
```

Revoke permissions by using the policy-based access control mechanism

The RAM user Alice is not allowed to delete the tables whose names start with tb_, as described in Grant permissions by using the policy-based access control mechanism. If the tables are no longer required and you want to allow the RAM user Alice to delete the tables, you can revoke the related permission from the RAM user Alice.

This operation can be performed only by the project owner or users assigned the Super_Administrator or Admin role. You can use one of the following methods to revoke the permission from the RAM user Alice based on your business requirements.

- **Revoke the permission that is granted to the role and retain the role**

  Perform the following steps:

  i. Start the MaxCompute client.

  ii. Execute the `REVOKE` statement to revoke the permission that is granted to the delete_test role. This way, the delete_test role is allowed to delete the tables whose names start with tb_.

  Sample statement:

  ```
  revoke drop on table tb_* from role delete_test privilegeproperties("policy" = "true"
  , "allow"="false");
  ```

  For more information about the REVOKE syntax, see the "Policy-based access control by using the GRANT statement" section in Policy-based access control and download control.

  iii. Execute the `SHOW GRANTS` statement to view the permissions of the RAM user Alice. Sample statement:

  ```
  show grants for ram$bob@aliyun.com:Alice;
  ```

  The following results are returned:

```
[roles]
role_project_admin, delete_test                                    -- The delete_test ro
le is retained.
Authorization Type: Policy                                         -- The permission is
revoked.
[role/role_project_admin]
A       projects/mcproject_name: *
A       projects/mcproject_name/instances/*: *
A       projects/mcproject_name/jobs/*: *
A       projects/mcproject_name/offlinemodels/*: *
A       projects/mcproject_name/packages/*: *
A       projects/mcproject_name/registration/functions/*: *
A       projects/mcproject_name/resources/*: *
A       projects/mcproject_name/tables/*: *
A       projects/mcproject_name/volumes/*: *
Authorization Type: ObjectCreator
AG      projects/mcproject_name/tables/local_test: All
AG      projects/mcproject_name/tables/mr_multiinout_out1: All
AG      projects/mcproject_name/tables/mr_multiinout_out2: All
AG      projects/mcproject_name/tables/ramtest: All
AG      projects/mcproject_name/tables/tb_test: All
AG      projects/mcproject_name/tables/wc_in: All
AG      projects/mcproject_name/tables/wc_in1: All
AG      projects/mcproject_name/tables/wc_in2: All
AG      projects/mcproject_name/tables/wc_out: All
```

For more information about how to view user permissions, see Query permissions by using MaxCompute SQL.

iv. Log on to the MaxCompute client as Alice and execute the `DROP TABLE` statement to delete the tables whose names start with tb_.

Sample statement:

```
drop table tb_test;
```

If OK is returned, the permission is revoked.

- **Revoke the role from the user and delete the role if required**

Perform the following steps:

i. Start the MaxCompute client.

ii. Execute the `REVOKE` statement to revoke the delete_test role from Alice.

Sample statement:

```
revoke delete_test from ram$bob@aliyun.com:Alice;
```

For more information about how to revoke a role from a user, see Role planning and management.

iii. Execute the `SHOW GRANTS` statement to view the permissions of the RAM user Alice. Sample statement:

```
show grants for ram$bob@aliyun.com:Alice;
```

The following results are returned:

```
[roles]
role_project_admin                                          -- The delete_test role
is revoked.
Authorization Type: Policy
[role/role_project_admin]
A       projects/mcproject_name: *
A       projects/mcproject_name/instances/*: *
A       projects/mcproject_name/jobs/*: *
A       projects/mcproject_name/offlinemodels/*: *
A       projects/mcproject_name/packages/*: *
A       projects/mcproject_name/registration/functions/*: *
A       projects/mcproject_name/resources/*: *
A       projects/mcproject_name/tables/*: *
A       projects/mcproject_name/volumes/*: *
Authorization Type: ObjectCreator
AG      projects/mcproject_name/tables/local_test: All
AG      projects/mcproject_name/tables/mr_multiinout_out1: All
AG      projects/mcproject_name/tables/mr_multiinout_out2: All
AG      projects/mcproject_name/tables/ramtest: All
AG      projects/mcproject_name/tables/wc_in: All
AG      projects/mcproject_name/tables/wc_in1: All
AG      projects/mcproject_name/tables/wc_in2: All
AG      projects/mcproject_name/tables/wc_out: All
```

iv. Log on to the MaxCompute client as Alice and execute the `DROP TABLE` statement to delete the tables whose names start with tb_.

Sample statement:

```
drop table tb_test;
```

If OK is returned, the permission is revoked.

v. Optional. Execute the `DROP ROLE` statement to delete the delete_test role.

Sample statement:

```
drop role delete_test;
```

If OK is returned, the role is deleted. For more information about how to delete a role, see Role planning and management.