Alibaba Cloud MaxCompute

Best Practices

Issue: 20181211

Legal disclaimer

Generic conventions

Table -1: Style conventions

Style	Description	Example
⛔	This warning information 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.
⚠	This warning information 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 restore business.
📋	This indicates warning information, supplementary instructions, and other content that the user must understand.	📋 **Note:** Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	📋 **Note:** You can use **Ctrl** + **A** to select all files.
>	Multi-level menu cascade.	**Settings** > **Network** > **Set network type**
Bold	It is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
`Italics`	It is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *`[-all|-t]`*
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *`{stand | slave}`*

Contents

1 Basic differences with standard SQL and solutions

This article describes problems that occur frequently when users who are familiar with the relational database SQL use MaxCompute SQL. For specific MaxCompute SQL syntax, see *SQL Overview*.

Basic differences of MaxCompute SQL

Scenarios

- Transactions are not supported. (The commit and rollback actions are not supported. We recommend using the codes that have idempotence and support re-running. We also recommend using Insert Overwrite, instead of Insert Into, to write data.)
- Indexes and primary and foreign key constraints are not supported.
- Auto increment fields and default values are not supported. If the default value exists, assign the value when writing the data.

Table Partitioning

- A table supports 60,000 partitions.
- Otherwise an error is returned. If the partitions are list partitions and the query performs filtering based only on list partitions, an error may be returned if the total number of partitions exceeds 10,000.

Precision

- The Double type has precision. Therefore, we do not recommend that you directly join two Double fields using an equal sign (=). We recommend deducting one number from the other, and consider the two numbers as the same if their difference is smaller than a preset value, for example, `abs(a1- a2)< 0.000000001`.
- Currently, MaxCompute SQL supports the high-precision type Decimal. If you have higher precision requirements, you can store the data as the string type and use UDF to implement such calculation.

Data type conversion

- If two different field types need to be joined, to prevent unexpected errors, we recommend first converting the types before joining them. This also helps code maintenance.

- For implicit conversions of dates and strings, to input a string to a function that requires entering a date, you can convert between the string and date according to the `yyyy-mm-dd hh:mm:ss` format. For other formats, see *Date functions > TO_DATE*.

DDL difference and solutions

Table structure

- The partition column name cannot be changed. Only the value corresponding to the partition column can be changed.
- You can add columns but cannot delete columns and modify the data type of columns. You can add columns like this : `ALTER TABLE table_name ADD COLUMNS (col_name1 type1, col_name2 type2…)`

  If you must delete columns or modify the column data types, here is the most secure way:

  1. Create a new table. For example: `CREATE TABLE new_table_name as SELECT c1, c2,c3 FROM table_name;`
  2. Delete your original table and rename your new table as the original one. For example: `ALTER TABLE new_table_name as RENAME TO table_name;`Then you need to insert your data manually into your new table.

DML difference and solutions

INSERT

- The most intuitive syntax difference is that insert into/overwrite is followed by the keyword **Table**.
- The field mapping of the data inserted into tables is not based on the alias of Select, but on the order of the fields of Select or the fields in the table.

UPDATE/DELETE

- Currently, the Update/Delete statements are not supported.
- If you want to perform Update, you may need to import source partition/table data into the new partition/table, and the corresponding update logic must be executed during the import process.
- If you want to perform Delete, you can drop the table to delete data. For non partition table, you can empty the table data through the `TRUNCATE TABLE table_name;` statement. For partition table, you can delete partitions by `ALTER TABLE table_name DROP IF EXISTS PARTITION` (partition name='specific partition value'). You can also use `insert overwrite` statement to update or delete data.

SELECT

- The Select field in the Group by query is either the group field of Group By or the aggregate function to be used. From the logic perspective, if a non-group column and Group By Key have multiple records of data, the data cannot be displayed without the aggregate function.
- MaxCompute does not support Group by cube (group by with rollup). However, you can use the union clause to simulate Group by cube, for example, `select k1, null, sum(v) from t group by k1 union all select k1, k2, sum(v) from t group by k1 , k2;`.

Subquery

Subqueries must have aliases. We recommend that all queries have aliases.

IN/NOT IN

- The data volume in the subquery following In/Not In and Exist/Not Exist cannot exceed 1000. To solve the problem. If the uniqueness of the results returned by the subquery is guaranteed in the business aspect, you can consider removing Distinct to improve the query performance.

10,000 results returned by MaxCompute SQL

- MaxCompute limits the number of data records returned by the separately executed Select clause. For the specific configurations, see *Other operations*. The maximum data number is set to 10,000. If the number of data records to be queried is large.

MAPJOIN

- The Join clause does not support Cartesian products. The Join clause must use the on clause to set the joining conditions. If you want to create broadcast tables for small tables, you must use MapJoin Hint. For more information.

ORDER BY

- **Order By must be followed by Limit n**. If you want to sort a large number of data records, or even sort the full table, you can set N to a large number. However, use this method with caution. Because the system does not have advantages of distributed systems, the performance may be affected. For more information.

UNION ALL

- The data types, quantities, and names of all columns involved in the UNION ALL calculation must be consistent. Otherwise, an exception is returned.
- The UNION ALL query must be nested by a subquery.

2 Join on condition in MaxCompute(ODPS) SQL

One of the most common operations in MaxCompute(ODPS) SQL is join.

Overview

Currently MaxCompute offers several join types:

Type	Meaning
Inner join	Output data that matches the criteria of the Association
Left join	Outputs all records for the left table, and for the right table that matches the associated data, outputs the right table, there is no match, and the right table supplements null.
Right join	Outputs all records of the right table, for which the left table matches the associated data, for which the left table is output, no match, left table to fill in null.
Full join	Outputs all records for the left and right tables , for data that is not associated with, a null is added on the other side that is not associated.
Left Semi Join	For a single piece of data in the left table, if the right table has rows that match the criteria of the Association, the left table is output.
Left Anti Join	For a single piece of data in the left table, if for all rows in the right table, there is no data that matches the criteria of the Association, and the left table is output.

Note:
User Defined Join specifies both input streams, and you can implements the logic of the join yourself, which is not discussed here.

Depending on the scenario, the user can use different Join types to implement the corresponding Association operation. But in the actual use process, it is often not clear to users that the filtering criteria are different in join on statements or in where, or think they're doing the same thing, for example, in a production environment, users can often be seen writing:

```
A (LEFT/RIGHT/FULL/LEFT SEMI/LEFT ANTI) JOIN B
```

```
ON a.key = b.key and A.ds='20180101' and B.ds='20180101';
```

The intention of the user here is to get the data for a partition in A and B for the join operation., that is:

```
(Select * from a where DS = '20180101 ')
(LEFT/RIGHT/FULL/LEFT SEMI/LEFT ANTI)  JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key
```

However, for different Join types, the two may not be equivalent, not only can not push the partition conditions, results In a full table scan, and it can cause correctness problems. Here is a brief analysis of the filter conditions in:

1. Where condition of subquery
2. JOIN ON condition
3. Where condition after JOIN ON

The similarities and differences.

Principle

Let's start with the computed order of a join and a where condition,:

```
(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}
```

For example, the order of calculation is

1. Subquery`{subquery_where_condition}`.
2. The condition of the `{on_condition}` For the join.
3. The calculation of the join result collection, `{where_condition}`.

For different Join types, filter statements are placed in `{subquery_where_condition}`,`{on_condition}`, and `{where_condition}`, sometimes the results are consistent, and sometimes the results are inconsistent. The following discussion takes place:

Experiment

1. Prepare

First construct table:

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

key	ds
1	20180101
2	20180101
2	20180102

Table B:

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

key	ds
1	20180101
3	20180101
2	20180102

Then their product of Descartes is:

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

2. Inner Join

Conclusion: the filter conditions are equivalent in`{subquery_where_condition}`, `{on_condition}`, and `{where_condition}`.

The processing logic of inner join is to product the left and right tables to the Descartes, then select the traveling line output that meets the on expression.

a. In the first case, the subquery is filtered:

```
Select a. *, B .*
From
(Select * from a where DS = '20180101 ')
JOIN
(SELECT * FROM B WHERE ds='20180101') B
ON a.key = b.key;
```

It's very simple, and there's only one result:

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

b. In the second case, the JOIN condition is filtered:

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

There are nine results of Descartes, and there is only one result to satisfy the condition of on.

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

c. In the third case, the where condition filter after JOIN:

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

For example, there are nine results of Descartes that meet the on `a.key = b.key` there are 3 results for key, respectively:

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

filter this result again`A.ds='20180101' and B.ds='20180101'`, results in only 1 Article.

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

As you can see, three different results have been obtained by placing the filter conditions in three different places.

3. Left join

Conclusion: The filtering conditions are not necessarily equivalent in `{subquery_where_condition}`, `{on_condition}`, and `{where_condition}`.

For the filter criteria for the left table, the ones placed in `{subquery_where_condition}` and `{where_condition}` are equivalent.

For the filter criteria for the right table, the ones placed in `{subquery_where_condition}` and `{on_condition}` are equivalent.

The processing logic of left join is to make the left and right tables a Descartes product, then for the moving line output that satisfies the on expression, for the rows in the left table that do not meet the on expression, the left table is output, and the right table supplements null.

a. In the first case, the subquery is filtered:

```
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;
```

After filtering, there are two on the left and one on the right and two on the results:

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

b. In the second case, the JOIN condition is filtered:

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

There are nine results of Descartes, and only one result to satisfy the condition of on, the left table is null for the remaining two outputs of the Left table.

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

c. In the third case, the where condition filter after JOIN:

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

For example, there are nine results of Descartes that meet the ON `a.key = b.key` there are 3 results for key, respectively:

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

filter this result again`A.ds='20180101' and B.ds='20180101'`, results in only 1 Article.

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

As you can see, three different results have been obtained by placing the filter conditions in three different places.

4. Right join

The right join and left join are similar, just the difference between the left and right tables. Conclusion: The filtering conditions are not necessarily equivalent in `{subquery_where_condition}`, `{on_condition}`, and `{where_condition}`. For the filter criteria for the right table, the ones placed in `{subquery_where_condition}` and `{where_condition}` are equivalent. For the filter criteria for the left table, the ones placed in `{subquery_where_condition}` and `{on_condition}`are equivalent.

5. Full join

Conclusion: The filter conditions are written in `{subquery_where_condition}`, `{ on_condition}`, and `{where_condition}` are not equivalent.

The processing logic of full join is to make the left and right tables a Descartes product, then for the moving line output that satisfies the on expression, for the rows that do not meet the on expression in the tables on both sides, outputs a table with data, with null on the other side.

a. In the first case, the subquery is filtered:

```
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;
```

After filtering, there are two on the left and two on the right, and three on the right:

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

b. In the second case, the JOIN condition is filtered:

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

There are nine results of Descartes, and only one result to satisfy the condition of on, the left table is null for the remaining two outputs of the Left table. The remaining two outputs of the right table, the right table, and the left table, fill in null.

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

c. In the third case, the where condition filter after JOIN:

```
SELECT A.*, B.*
FROM A FULL JOIN B:
```

```
ON a.key = b.key
WHERE A.ds='20180101' and B.ds='20180101';
```

For example, there are nine results of Descartes that meet the on `a.key = b.key` there are 3 results for key, respectively:

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

Then the data on the other side of the JOIN is output, and NULL is added to the other side and the result is:

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

filter this result again`A.ds='20180101' and B.ds='20180101'`, results in only 1 Article.

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

As you can see, like LEFT JOIN, there are three different results.

6. Left Semi Join

Conclusion: The filter conditions are written in `{subquery_where_condition}`, `{on_condition}`, and `{where_condition}` are not equivalent.

The processing logic for LEFT SEMI Join is for each record in the left table, all go to the right table to match, and if the match succeeds, the left table is output. What you need to note here is that only the left table is output, therefore, the filter condition to the right cannot be written in the where condition after the JOIN. Left semi join is commonly used to implement the semantics of exists:

a. In the first case, the subquery is filtered:

```
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;
```

After filtering, there are two on the left and the right, which eventually fit`a.key = b.key` there is only one of the key:

a.key	a.ds
1	20180101

b. In the second case, the JOIN condition is filtered:

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

For the three records on the left, there is also only one result that meets the on condition.

a.key	a.ds
1	20180101

c. In the third case, the where condition filter after JOIN:

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

There is one on that can meet the on condition on the left:

a.key	a.ds
1	20180101

A. filter this result again`A.ds='20180101'`, results remain 1 Article:

a.key	a.ds
1	20180101

As you can see, the left semi join and inner join are similar, no matter where the filter conditions are placed, the results were consistent.

7. Left Anti Join

Conclusion: The filter conditions are written in `{subquery_where_condition}`, `{on_condition}`, and `{where_condition}` are not equivalent.

For the filter criteria for the left table, the ones placed in `{subquery_where_condition}` and `{where_condition}` are equivalent.

For the filter criteria for the right table, the ones placed in `{subquery_where_condition}` and `{on_condition}` are equivalent, the right table expression cannot be placed in `{where_condition}`.

The processing logic for left anti join is for each record in the left table, all go to match the right table, and if none of the records on the right table are matched successfully, the left table is output. Similarly, since only the left table is output, therefore, the filter condition to the right cannot be written in the where condition after the join. LEFT SEMI JOIN are often used to implement the semantics of not exists.

a. In the first case, the subquery is filtered:

```
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;
```

After filtering, there are two on the left, two on the right and one on the results.

a.key	a.ds
2	20180101

b. In the second case, the JOIN condition is filtered:

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

For the three records on the left, only the first one has the result of satisfying the ON condition, so output the remaining two records.

a.key	a.ds
2	20180101
2	20180102

c. In the third case, the WHERE condition filter after JOIN:

```
SELECT A.*
```

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

There are two on conditions that can be passed ON the left:

a.key	a.ds
2	20180101
2	20180102

The result is filtered again for a `A.ds='20180101'` and the result is 1.

a.key	a.ds
2	20180101

As you can see, in LEFT ANTI JOIN, the filter condition is placed in the JOIN ON condition and in the where condition before and after, and the result is different.

The above is a simple test of several different writing methods for a common scenario , without a specific deduction process, it will be more complex for scenarios involving expressions that are not equivalent to each other, interested students can try to derive them themselves.

Online status

These results are derived from SQL standard semantic patterns. Some users will find that the results of the same statements in the online environment do not match expectations, this is due to some historical reasons and compatibility considerations. In the implementation of Outer Join, a flag is set at the project level, called `odps.sql.outerjoin.supports.filters`, if set to false, indicates that the on condition of Outer Join does not support filtering conditions, writing `{on_condition}` will be treated as if it were written in `{subquery_where_condition}`, which is a non-standard behavior. Some users switch between two projects, it is caused by this that the same SQL runs differently in both projects.

It is hoped that everyone can write SQL according to the standard SQL semantics, in this way, you can ensure the portability of subsequent SQL.

View Project settings you can find the corresponding project and view the properties in the `project administration` in *http://adminconsole.odps.aliyun-inc.com/inn.view*.

Conclusion

The semantics of the filter conditions in different locations may vary greatly for the user, if you are simply filtering the data and then joining, you can simply remember the following points.

1. The inner join/left semi join can be written on both sides of the expression.
2. Left join/left anti join the filter criteria for the left table are to be put in `{subquery_where_condition}`or `{where_condition}`, the filter criteria for the right table are to be placed in either `{subquery_where_condition}` or `{on_condition}`.
3. Right join is opposite to left join, the filter criteria for the right table are to be placed either `{subquery_where_condition}` or `{where_condition}`, the filter criteria for the left table are to be placed on `{subquery_where_condition}` or `{on_condition}`.
4. Full outer join can only be placed in`{subquery_where_condition}`.

Of course, if you still think the rules are complicated, the best way to do this is to write the filter criteria to the subquery every time.

3 Export SQL operation result

This article provides examples to illustrate how to download the MaxCompute SQL computing results by using several methods.

Note:
The Java SDK is used as an example throughout this article.

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

- If the data volume is small, use *SQL Task* to list all query results.
- If you want to export the results of a specific table or partition, use *Tunnel*.
- If the SQL statements are complex, use Tunnel and SQL Task in combination.
- *DataWorks* allows you to conveniently run SQL statements and *Synchronize data*. It supports regular scheduling and task dependency configuration.DatadataDatadependency
- The Open Source Tool **DataX**helps you easily export data from maxcompute to the target data source.

Use SQL Task to export data

SQL Task is the interface where the SDK calls maxcompute SQL directly, you can easily run SQL and get its return results.

`SQLTask.getResult(i);` returns a list which can be iterated cyclically to obtain the complete SQL computing results. However, there is a flaw in this method. For more information, see the `SetProject READ_TABLE_MAX_ROW` maid feature mentioned in *other actions*.

Currently, you can adjust the maximum number of data records that the SELECT statement returns to the client up to **10,000**. If you run the SELECT statement on a client or using SQL Task, the query results are appended with Limit N. Limit N does not apply to the CREATE TABLE XX AS SELECT statement or in the case that the results are solidified in a specific table through INSERT INTO/OVERWRITE TABLE.

Use Tunnel to export data

If you need to export a query that results in the entire contents of a table (or a specific partition) all of the content), you can do this through tunnel, see *command-line* tools for details, and the *tunnel SDK* written based on the SDK.

An example is provided to illustrate how to export data by using the Tunnel command line. You can compile the Tunnel SDK only when data cannot be exported using some command lines. For more information, see *Batch data 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 SQL Task and Tunnel to export data

SQL Task cannot export more than 10,000 records, whereas Tunnel can. You can use them in combination You can use them in combination to export data.

The sample code is as follows:

```
private static final String accessId = "userAccessId";
private static final String accessKey = "userAccessKey";
private static final String endPoint = "http://service.odps.aliyun.com
/api";
private static final String project = "userProject";
private static final String sql = "userSQL";
private static final String table = "Tmp_" + UUID.randomUUID().
toString().replace("-", "_");//The name of the temporary table is a
random string.
private static final Odps odps = getOdps();
public static void main(String[] args) {
System.out.println(table);
runSql();
tunnel();
}
/*
* Download the results returned by SQL Task.
* */
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 record.
* If the data volume is small, you can print and copy the data
directly.  In reality, you can use Java.io to write the data to a
local file or a remote data storage.
* */
private static void consumeRecord(Record record, TableSchema schema) {
System.out.println(record.getString("username")+","+record.getBigint("
cnt"));
}
/*
* Run an SQL statement to save the query results to a temporary table
. The saved results can be downloaded using Tunnel.
* The lifecycle of the saved data is 1 day. The data does not occupy
much storage space even when an error occurs while you delete 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 the MaxCompute (formerly
ODPS) instance.
* */
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 export data using synchronization

Using the preceding method, you can save the downloaded data. Other methods are required to create the data and implement the scheduling dependency between data creation and storage.

DataWorks allows you to *Configure a data synchronization task* and configure *Periodic running* and *Dependency among multiple tasks* to complete the process from data creation to storage.

An example is provided to illustrate how to use Data IDE to run SQL statements and configure a data synchronization task to create and export data.

Procedure

1. Create a workflow with an SQL node and a data synchronization node. Connect the two nodes and configure an inter-node dependency, with the SQL node as the data production node and the data synchronization node as the data export node.
2. Configure the SQL node.

   Note:
   Run an SQL statement to create a table before you configure synchronization. If no table exists, the synchronization task cannot be configured.
3. Perform the following to configure the data synchronization task.

   a. Select a Source.
   b. Select a Target.
   c. Map fields.
   d. Control the tunnel.
   e. Preview and Save.
4. After workflow scheduling is configured, save and submit the workflow. Click **Test Run**. If you do not configure workflow scheduling, you can use the default scheduling configuration directly. View the running log on data synchronization as as in the following figure.

```
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 task time : 11s
Average data per task : 31.36 KB/s-
Write speed : 1,668 rec/s
Read records : 16,689
Failed read-write attempts : 0
```
5. Run an SQL statement to view the data synchronization results.

4 Demo: Modify incompatible SQL

The MaxCompute development team has completed the grayscale upgrade of MaxCompute 2.0 recently. The new version fully embraces open source ecosystems, supports more languages and functions, and enables faster operation. It also implements more rigorous syntax inspection, so that errors may be returned for some less rigorous syntax cases that can run normally in the earlier editor.

To enable smooth grayscale upgrade of MaxCompute 2.0, the MaxCompute framework supports rollback. If a task of MaxCompute 2.0 fails, it will be executed in MaxCompute 1.0. The rollback increases the E2E latency of the task. We recommend that you set `set odps.sql.planner.mode=lot;` to manually disable the rollback function before submitting jobs, to avoid the impact resulting from the changes made to the MaxCompute rollback policy.

The MaxCompute team notifies the owners of problematic tasks by email or DingTalk based on the online rollback condition. Modify your SQL tasks immediately; otherwise, the tasks may fail. Check the following errors against your tasks to avoid task failure in case of missed notifications.

The following lists the syntaxes for which MaxCompute 2.0 may return errors.

group.by.with.star

SELECT * … GROUP BY… syntax is problematic.

In the earlier version of MaxCompute, `select * from group by  key` is supported even when the columns that match * are not included in the GROUP BY key. Compatible with Hive, MaxCompute 2.0 prohibits this syntax unless the GROUP BY list is a column in all source tables. Examples:

Scenario 1: **The GROUP BY key does not include all columns**.

Incorrect syntax:

```
SELECT * FROM t GROUP BY key;
```

Error message:

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

Correct syntax:

```
SELECT DISTINCT key FROM t;
```

Scenario 2: **The GROUP BY key includes all columns**.

Not recommended syntax:

```
SELECT * FROM t GROUP BY key, value; -- t has columns key and value
```

Though MaxCompute 2.0 does not return an error, we recommend that you modify the syntax as follows:

```
SELECT DISTINCT key, value FROM t;
```

bad.escape

The escape sequence is incorrect.

According to the MaxCompute documentation, in string literal, every ASCII character ranging from 0 to 127 must be written in the format of a backslash followed by three octal numbers. For example, 0 is written as \001, and 1 is written as \002. Currently, \01 and \0001 are considered as \001.

It may bring confusions to new users. For example, “\000” + “1” cannot written as “\0001”. For users who migrate data from other systems to MaxCompute, incorrect data may be generated.

Note:
An error may be returned if numbers are appended to `\0 00`, for example, `\0001 - \0009 or \00001`.

MaxCompute 2.0 solves this problem. The script owner must correct the sequence, for example:

Incorrect 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
-
```

Correct syntax:

```
SELECT split(key, "\001"), value like "\001" FROM t;
```

column.repeated.in.creation

Duplicate column names are detected during execution of the CREATE TABLE statement.

MaxCompute 2.0 returns an error when duplicate column names are detected when the CREATE TABLE statement is executed. For example:

Incorrect 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
```

Correct syntax:

```
CREATE TABLE t (a BIGINT, b BIGINT);
```

string.join.double

The data on both sides of the equal sign of a Join condition belongs to the String and Double types.

In the old version of MaxCompute, the String and Double type data is converted to the Bigint type at the cost of precision. 1.1 = “1” in a Join condition is considered as equal But compatible with Hive, MaxCompute 2.0 converts the String and Double type data to the Double type.

Syntax that is not recommended:

```
SELECT * FROM t1 JOIN t2 ON t1.double_value = t2.string_value;
```

Warning:

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

Recommended correction:

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

You can also convert the data as needed.

window.ref.prev.window.alias

Window functions reference other aliases in the select list of the same level.

Examples:

With rn absent from t1, the incorrect syntax is as follows:

```

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
```

Correct 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

Select * is followed by an alias.

The select list allows the use of * to select all the columns of a table, but * cannot be followed by any alias even if the expanded * has only one column. The new editor returns errors for similar syntaxes. For example:

Incorrect syntax:

```
select * as alias from dual;
```

Error message:

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

Correct syntax:

```
select * from dual;
```

agg.having.ref.prev.agg.alias

The select list is preceded by an aggregate function alias when HAVING exists. Examples:

Incorrect 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 GROUP BY key
```

s and cnt do not exist in source table t1, but MaxCompute of the old version does not return an error because HAVING exists. In MaxCompute 2.0, the prompt “column cannot be resolved” appears and an error is returned.

Correct 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

ORDER BY is not followed by the LIMIT statement.

By default, MaxCompute requires that ORDER BY be followed by the LIMIT statement to limit the number of data records. Because ORDER BY is used for full data sorting, the execution performance is low without the LIMIT statement. Examples:

Incorrect 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 LIMIT clause
```

Correct syntax:

Add the LIMIT statement to the subquery “order by v.uv desc”.

In MaxCompute 1.0, the view inspection is not strict. For example, a view is created in a project which does not require LIMIT statement check (odps.sql.validate.orderby.limit=false).

```
CREATE VIEW dual_view AS SELECT id FROM dual ORDER BY id;
```

When you run the following statement to access the view:

```
SELECT * FROM dual_view;
```

MaxCompute 1.0 does not return an error, whereas MaxCompute 2.0 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 earlier version of MaxCompute, an alias is auto generated for every expression of the SELECT statement. The alias is displayed on the console. However, the earlier version does not guarantee that the alias generation rule is correct or remains unchanged. Therefore, we do not recommend that you use auto generated aliases.

In MaxCompute 2.0, a warning is given for use of auto generated aliases, but due to breadth of involvement such use is not prohibited for the moment.

In some cases, known changes are made to the alias generation rules in the different versions of MaxCompute. Some online jobs depend on the auto generated aliases. Queries may fail when MaxCompute performs version upgrade or rollback. If you have such problems, modify queries and explicitly specify the alias of the column of interest. Examples:

Not recommended syntax:

```
SELECT _c0 FROM (SELECT count(*) FROM dual) t;
```

Recommended correction:

```
SELECT c FROM (SELECT count(*) c FROM dual) t;
```

non.boolean.filter

Non-Boolean filter conditions are used.

MaxCompute prohibits the implicit conversion between the Boolean type and other data types . However, the earlier version of MaxCompute allows the use of Bigint type filter conditions in some cases. MaxCompute 2.0 prohibits the use of Bigint type filter conditions. If your scripts have Bigint type filter conditions, modify them promptly. Examples:

Incorrect syntax:

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

Error message:

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

Correct syntax:

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

post.select.ambiguous

The GROUP BY, CLUSTER BY, DISTRIBUTE BY, and SORT BY statements reference columns with conflicting names.

In the old version of MaxCompute, by default, the system selects the last column of the select list as the operation object. MaxCompute 2.0 reports an error in this case. Make relevant modification timely. Examples:

Incorrect 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 or null.a
```

Correct syntax:

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

The pushed change covers the statements with conflicting column names but the same syntax. Although there is no ambiguity, an error is often returned for these statements and a warning is triggered. We recommend that you make relevant modification.

duplicated.partition.column

Partitions with the same name are specified in a query.

In the earlier version of MaxCompute, no error is returned when two partition keys with the same name are specified. The latter partition key overwrites the former one. It causes confusion. MaxCompute 2.0 returns an error in this case.

Incorrect syntax:

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

In fact, ds = ‘1’ is ignored during execution.

Correct syntax:

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

Incorrect syntax:

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

Correct syntax:

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

order.by.col.ambiguous

The ORDER BY clause references the duplicate aliases in the select list.

Incorrect syntax:

```
SELECT id, id
FROM dual
ORDER BY id;
```

Correct syntax:

```
SELECT id, id id2
FROM dual
ORDER BY id;
```

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

in.subquery.without.result

colx does not exist in the source table if colx in subquery does not return any results.

Incorrect syntax:

```
SELECT * FROM dual
WHERE not_exist_col IN (SELECT id FROM dual LIMIT 0);
```

Error message:

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

ctas.if.not.exists

The syntax of the target table is incorrect.

If the target table exists, the earlier version of MaxCompute does not check the syntax, whereas MaxCompute 2.0 does. Many errors may return in this case, for example:

Incorrect syntax:

```
CREATE TABLE IF NOT EXISTS dual
AS
```

```
SELECT * FROM not_exist_table;
```

Error message:

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

worker.restart.instance.timeout

In the earlier version of MaxCompute, every time a UDF outputs a record, a write operation is triggered on the distributed file system, and a heartbeat packet is sent to Fuxi. If the UDF does not output any records for 10 minutes, the following error is reported:

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

The runtime framework of MaxCompute 2.0 supports vectoring. It processes multiple rows of a column at a time to improve the execution efficiency. Vectoring may cause the normal statements to time out in the case that a heartbeat packet is not sent to Fuxi within the specified time while multiple records are processed at a time. The interval between two output records does not exceed 10 minutes.

If a time-out error occurs, we recommend that you first check the UDF performance. It takes several seconds to process each record. If the UDF performance cannot be optimized, as a workaround, you can set the value of "batch row" manually. The default value is 1024.

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

divide.nan.or.overflow

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

In the old version of MaxCompute, the physical execution plan for a statement is as follows:

```

EXPLAIN
Select if (false, 0/0, 1.0)
FROM dual;
In Task M1_Stg1:
    Data source: meta_dev.dual
    TS: alias: dual
      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 the second parameter Divide is not evaluated. Division by zero is normal.

MaxCompute 2.0 supports division constant folding. An error is returned. As shown in the following:

Incorrect syntax:

```
SELECT IF(FALSE, 0/0, 1.0)
FROM dual;
```

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 and 0.000000
```

An overflow error may occur besides nan. For example:

Incorrect syntax:

```
SELECT IF(FALSE, 1/0, 1.0)
FROM dual;
```

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 1.000000 and 0.000000
```

Correct syntax:

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

A similar problem occurs in the constant folding of CASE WHEN, such as CASE WHEN TRUE THEN 0 ELSE 0/0. During constant folding in MaxCompute 2.0, all subexpressions are evaluated , causing incorrect division by zero.

CASE WHEN may involve more complex optimization scenarios, for 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. A similar conversion is as follows :

```
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: java.lang.ArithmeticException: DIVIDE func result
overflow, two params are 1.000000 and 0.000000
```

An error is returned for the constant folding of the first clause of UNION ALL. We recommend that you transfer CASE WHEN in SQL to subqueries and eliminate 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 earlier version of MaxCompute supports Multi-way Join optimization. Multiple JOIN statements with the same Join key are merged for execution in the same Fuxi task, such as J4_1_2_3_Stg1 in the following example.

```

EXPLAIN
SELECT t1. *
FROM t1 JOIN t2 ON t1.c1 = t2.c1
JOIN t3 ON t1.c1 = t3.c1;
```

The earlier version of MaxCompute has the following physical execution plan:

```

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
```

The earlier version of MaxCompute still keeps the physical execution plan when MapJoin Hints are added, and gives priority to applications in Multi-way Join optimization. It may ignore the user -specified MapJoin Hint.

```
EXPLAIN
SELECT /*+mapjoin(t1)*/ t1. *
FROM t1 JOIN t2 ON t1.c1 = t2.c1
JOIN t3 ON t1.c1 = t3.c1;
```

The physical execution plan of the earlier version of MaxCompute is the same as the preceding one.

The optimizer of MaxCompute 2.0 gives priority to the user-specified MapJoin Hint. In the preceding example, if the value of t1 is relatively higher, the following error is returned:

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

We recommend that you remove the MapJoin Hint if MapJoin is not your expected behavior.

sigkill.oom

Like small.table.exceeds.mem.limit, if you specify a MapJoin Hint and your small tables are of a relatively larger size, in the earlier version of MaxCompute, multiple JOIN statements may be optimized by Multi-way Join and can be successfully executed. However, in MaxCompute 2.0, some users may set odps.sql.mapjoin.memory.max to prevent small tables from exceeding the size limit. Each MaxCompute worker has a fixed memory limit. If small tables are of a large size, MaxCompute workers may be killed because the memory limit exceeds. The error is similar to the following:

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

We recommend that you remove MapJoin Hint and use Multi-way Join.

wm_concat.first.argument.const

According to the *Aggregate function* document, the first parameter of WM_CONCAT must be a constant. The old version of MaxCompute does not have strict check standards. For example, when the source table has no data, no error is returned even if the first parameter of WM_CONCAT is ColumnReference.

```
The function statement is as follows:
string wm_concat(string separator, string str)
Description of parameters:
Separator: String-type constant.  Constants of other types or non-
constants can cause exceptions.
```

MaxCompute 2.0 checks the validity of parameters during the planning phase. An error is returned if the first parameter of WM_CONCAT is not a constant. Examples:

Incorrect 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 partition table with 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');
```

For the preceding SQL statements, the constants of the IN INT type in the pt columns of the String type are converted to the Double type for comparison. Even if odps.sql.udf.strict.mode is set to "true" in the project, the old version of MaxCompute does not return an error and it filters out all pt columns, whereas in MaxCompute 2.0, an error is returned. Examples:

Incorrect syntax:

```
SELECT key FROM srcpt WHERE pt IN (1, 2);
```

Error message:

```
FAILED: ODPS-0130071:[0,0] Semantic analysis exception - physical
plan generation failed: java.lang.NumberFormatException: ODPS-0123091
```

```
:Illegal type cast - In function cast, value 'pt1' cannot be casted
from String to Double.
```

We recommend that you avoid comparing String type partition columns and INT-type constants and convert INT-type constants to the String type.

having.use.select.alias

SQL specifies that the GROUP BY+HAVING clause precedes the SELECT clause. Therefore, the column alias generated by the SELECT clause cannot be used in the HAVING clause. For example:

Incorrect syntax:

```
SELECT id id2 FROM DUAL GROUP BY id HAVING id2 > 0;
```

Error message:

```
FAILED: ODPS-0130071:[1,44] Semantic analysis exception - column id2
 cannot be resolvedODPS-0130071:[1,44] Semantic analysis exception -
column reference id2 should appear in GROUP 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 2.0, dynamic partitions may be converted to static ones by the optimizer. For example:

```
INSERT OVERWRITE TABLE srcpt PARTITION(pt) SELECT id, 'pt1' FROM dual;
```

is converted to:

```
INSERT OVERWRITE TABLE srcpt PARTITION(pt='pt1') SELECT id FROM dual;
```

If the specified partition value is invalid (for example, '${bizdate}' is used), MaxCompute 2.0 returns an error during syntax check. For more information, For more information, see *Partition*.

Incorrect syntax:

```
INSERT OVERWRITE TABLE srcpt PARTITION(pt) SELECT id, '${bizdate}'
FROM dual LIMIT 0;
```

Error message:

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

In the earlier version of MaxCompute, with LIMIT 0, no results are returned by the SQL statements , and no dynamic partitions are created. In this case, no error is returned.

lot.not.in.subquery

Processing of the null value of In subquery.

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

MaxCompute 2.0 performs processing with a standard behavior. If you receive a notification on this problem, check your queries to determine whether the subqueries in the IN operation have a null value and whether the related behavior meets your expectation. If not, make relevant changes . Examples:

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

Ignore this problem if "accepted" does not have null values. If a null value exists, c not in (select accepted from c_list) returns the value "true" in the earlier version, but returns a null value in the new version.

Correct syntax:

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

5 Computing optimization program for long period indicators

Experiment background

E-commerce companies such as TaoBao have all kinds of perspectives and procedures regarding user data analysis. E-commerce data warehouse and business analysis often require the calculation of various indicators such as the number of visitors in last few days, buyers, and regular customers.

The calculation of these indicators is based on the data that has accumulated on the e-commerce platform or the online store over a period. This article assumes that the data for calculation is stored in MaxCompute.

In general, these indicators are calculated using the log detail table. The following code calculates the number of visitors of the product in the past 30 days:

```
select item_id --Item ID
  ,count(distinct visitor_id) as ipv_uv_1d_001
from Log detail table that stores the data about the visits to the
item
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 are the scheduling variables of DataWorks. They are only applicable to the scheduling tasks of DataWorks. This condition applies in the rest of this article.

The preceding code has a serious problem when many logs are generated every day. Massive **Map Instance** are required and may exceed the **99999** quantity limit, affecting the execution of the map task and subsequent operations.

The need for massive map instances is owing to the large volume of log data per day, not to mention the log data generated over a period of 30 days. The SELECT operation requires more map instances than permitted by the upper limit, causing abnormal code execution.

Tutorial goal

The impact on performance resulting from the calculation of long-period indicators can be minimized in either of the following ways:

- Reduce the data volume to avoid a summary of data from multiple days.
- Create a temporary table and make a summary of 1D data to remove duplicate data and thus reduce the data volume on a daily basis.

Tutorial scheme

Procedure

1. Create a medium table and make a daily summary.

   In the preceding example, you can create a medium table with a granularity of item_id+ visitor_id for daily summary. Record this table as A. For example:

```
insert overwrite table mds_itm_vsr_xx(ds='${bdp.system.bizdate} ')
select item_id,visitor_id,count(1) as pv
  from
   (
  select item_id,visitor_id
  from Log detail table that stores the data about the visits to the
 item
  where ds =${bdp.system.bizdate}
  group by item_id,visitor_id
   ) a;
```

2. Calculate the data from multiple days, and use the medium table to make a summary.

   Make a 30-day summary on Table A, as shown in the following code:

```
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'),'yyyymmdd')
  group by item_id;
```

Impact and consideration

In the preceding method, the detailed data of daily access logs is deduplicated to reduce the data volume while improving the performance. A weakness of this method is the need to read data from N partitions for every calculation of multi-day data.

To avoid repeated data reads, you can compress the data in the N partitions into one partition which contains all historical data.

To do this, you can **calculate long-period indicators in an incremental and accumulative way**.

Use cases

Calculate the number of regular customers of the store for the past one day. A regular customer is defined as a buyer who has purchased items at the store over a period, such as the past 30 days.

The number of regular buyers is calculated as follows:

```
select item_id ----Item ID
        ,buyer_id as old_buyer_id
from Detail table that stores the data about buyers and purchased
items
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:

- Maintain a table as table A to record the relationship between buyers and purchased items . Specifically, record the first purchase period, the last purchase period, the total number of purchased items, and the total amount of the purchases in this table.
- Update the data of Table A every day based on the payment detail log for the past one day.
- To determine whether a person is a regular buyer, check whether the last purchase time is within the past 30 days. This deduplicates data relationship pairs and reduces the volume of input data.

6 Optimize long tail computing

Long tail is a common and difficult problem in distributed computing. The workload of each node is different when data is distributed unevenly. The whole task does not end until the work on the slowest node is complete.

To address this problem, you can distribute a piece of task to multiple workers for execution, rather than designate a worker to run the heaviest task. This article describes how to address typical cases of long tail.

Join long tail

Cause:

When the Key of a Join statement has a large amount of data, a long tail occurs.

Solution:

You can solve the problem in three steps:

- Verify that one or both of the tables are not small tables. If one table is large and the other small, you can use the mapjoin statement to cache the small table. For the syntax and relevant description, see *Introduction to the SELECT Syntax*. If the job is a MapReduce job, you can use the resource table function to cache the small table.
- If both tables are relatively large, reduce duplicated data as much as possible.
- If the problem persists, consider service optimization to avoid the calculation of Cartesian product on the two keys with a large data volume.
- Small table leftjoin large table, odps direct leftjoin is slow. At this point, you can first small the table and the large table mapjoin, so that you can get the intersection between the small table and the big table, and this intermediate table must not be greater than the large table (as long as there is not a large key tilt not very large). The small table then performs a lettjoin with this intermediate table, and the effect is equal to the larger table of the small leftjoin.

Group By long tail

Cause

When a Key of the Group By statement has a large amount of computations, a long tail occurs.

Solution:

You can solve the problem in either of the following ways:

- Rewrite the SQL statement and add random numbers to split the long key. As shown in the following:

  ```
  SELECT Key,COUNT(*) AS Cnt FROM TableName GROUP BY Key;
  ```

  Provided that the Combiner is skipped, data is shuffled from the M node to the R node. Then, the R node performs the Count operation. The execution plan is M > R. If you want to redistribu te work to the key with the long tail, modify the statement as follows:

  ```
  -- Assume that the key with the long tail 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 is changed to M > R > R. The execution lengthens, but the consumed time may reduce because the long-tail key is processed in two steps.

  Note:

  If you use the preceding method to add an R execution step for solving a long tail problem that is not serious, the consumed time may increase.

- Set system parameters as follows:

  ```
  set odps.sql.groupby.skewindata=true.
  ```

  Parameter setting is a universal optimization method, but the results are not satisfying because this method does not consider specific services. You can rewrite the SQL statement in a more efficient manner based on the actual data.

Distinct long tail

When a long tail occurs in a Distinct statement, the Key splitting method does not apply. You can consider other methods.

Solution:

```
--Original SQL statement, with the null UID skipped
SELECT COUNT(uid) AS Pv
```

```
    , COUNT(DISTINCT uid) AS Uv
FROM UserLog;
```

Rewrite the statement as follows:

```
SELECT SUM(PV) AS Pv
    , COUNT(*) AS UV
FROM (
    SELECT COUNT(*) AS Pv
      , uid
    FROM UserLog
    GROUP BY uid
) a;
```

The Distinct statement is rewritten to a Count statement to relieve the computing pressure on a single Reducer. This solution also enables Group By statement optimization and Combiner execution, greatly improving the performance.

Long tail of a dynamic partition

Cause:

- To sort the data of small files, the dynamic partition function starts a Reduce task in the final stage of execution. A long tail occurs when the data written by the dynamic partition function is skewed.
- Misuse of the dynamic partition function often results in long tails.

Solution:

If the target partition for data write is determined, you can specify this partition during the Insert operation, instead of using the dynamic partition function.

Remove long tails using the Combiner

For MapReduce jobs, it is a common practice to use the Combiner to remove long tails. This practice has been mentioned in the WordCount example. The Combiner can reduce the volume of data shuffle from the Mapper to the Reducer, thus sufficiently reduce the overhead of network transfer. The optimization is automatically done for MaxCompute SQL statements.

Note:

The Combiner only supports optimization for Map. You must make sure that the results of Combiner execution are the same. For example, in WordCount, the results are the same for two (KEY,1)s and one (KEY,2). For example, in average value calculation, (KEY,1) and (KEY,2) cannot be directly merged into (KEY,1.5) in the Combiner.

Remove long tails using system optimization

In addition to the Local Combiner, you can use the optimization method provided by MaxCompute to remove long tails. For example, the following content (+N backups) is logged during the task running process:

```
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%]
```

There total number of Reducers is 1,047, of which 1,046 are complete. The last one is incomplete. After the system identifies this condition, it starts a new Reducer to run the same data. Then, the system takes the data of the Reducer which completes first and combines the data to the result set.

Remove long tails using service optimization

The preceding optimization methods cannot solve all problems. You can analyze your services to find a better solution,the example is as follows:

- A large amount of noisy data may exist in reality. For example, you want to check the behavior data in the access record of each user by visitor ID. You must remove crawler data first, even though crawlers become increasingly difficult to identify; otherwise, the crawler data may easily cause a long tail during the calculation process. A similar case is when you associate data by a specific xxid, you must check whether the field for association is null.
- Special services exist. For example, ISV operation records differ greatly from common user records in terms of data volume and behavior. You can use a special method to analyze and handle the ISV operation records for major accounts independently.
- When data distribution is uneven, do not use a constant field as the Distribute by field for full sorting.

7 Learn to write SQL statements quickly

This article introduces you to maxcompute SQL through the way of course practice, allows you to quickly understand how SQL is written, and you know the difference between maxcompute SQL and standard SQL, please read in conjunction with the *MaxCompute SQL base documentation*.

Prepare a dataset

In the example the Emp/Dept table is used as the dataset. For the convenience of everybody's operation, provides the relevant maxcompute build table statements and data files (*emp table data files*, *dept table data file*), where you can create a table and upload the data yourself on the maxcompute project.

The DDL statements for creating an emp table are as follows:

```
CREATE TABLE IF NOT EXISTS emp (
  EMPNO string,
  ENAME string ,
  JOB string ,
  MGR bigint ,
  HIREDATE datetime ,
  SAL double ,
  COMM double ,
  DEPTNO bigint );
```

The DDL statements for creating a Dept table are as follows:

```
CREATE TABLE IF NOT EXISTS dept (
  DEPTNO bigint ,
  DNAME string,
  LOC string);
```

SQL操作

Notes for SQL beginners

- If you use Group by, the Select part must be either group items or aggregate functions.
- Order by must be followed by Limit n.
- The Select expression does not support subqueries. You can rewrite the code to the Join clause to use subqueries.
- The Join clause does not support Cartesian projects and usage of MapJoin.
- Union all must use the format of subqueries.
- The In/Not in statements can correspond to only one column of subquery, and the number of rows of the returned results cannot exceed 1,000. Otherwise, rewrite the statements using Join.

Compile SQL programs to solve problems

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

To avoid a situation where the amount of data is too large, you often encounter 6th points in the **problem point** , you need to override using join. For example:

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

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

This example is a typical scenario of MapJoin, as shown in the following code:

```
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: List the name and the immediate superior's name of all employees.

Use non-equi join, as shown in the following code:

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

Example 4: List all jobs of which salaries are higher than 1500 yuan.

Use Having, as shown in the following code:

```
SELECT emp. 'JOB'
    , MIN(emp.sal) AS sal
FROM 'emp'
GROUP BY emp. 'JOB'
```

```
HAVING MIN(emp.sal) > 1500;
```

Example 5: List the number of employees of each department, and their average salary and average service year.

You can use many built-in functions for time processing, as shown in the following code:

```
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: List the names of first three employees who have the highest salaries and their ranks (Top n is frequently used).

The SQL statements are as follows:

```
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;
```

Title 7: write down the number of people in each department in one SQL, clerk (clerk) the total number of people in the department accounted.

The SQL statement is as follows:

```
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;
```

8 Migrate data from Hadoop to MaxCompute

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

Prepare the environment

1. Build a Hadoop cluster.

   Before data migration, you must ensure that your Hadoop cluster works properly. You can use Alibaba Cloud E-MapReduce to automatically build a Hadoop cluster.

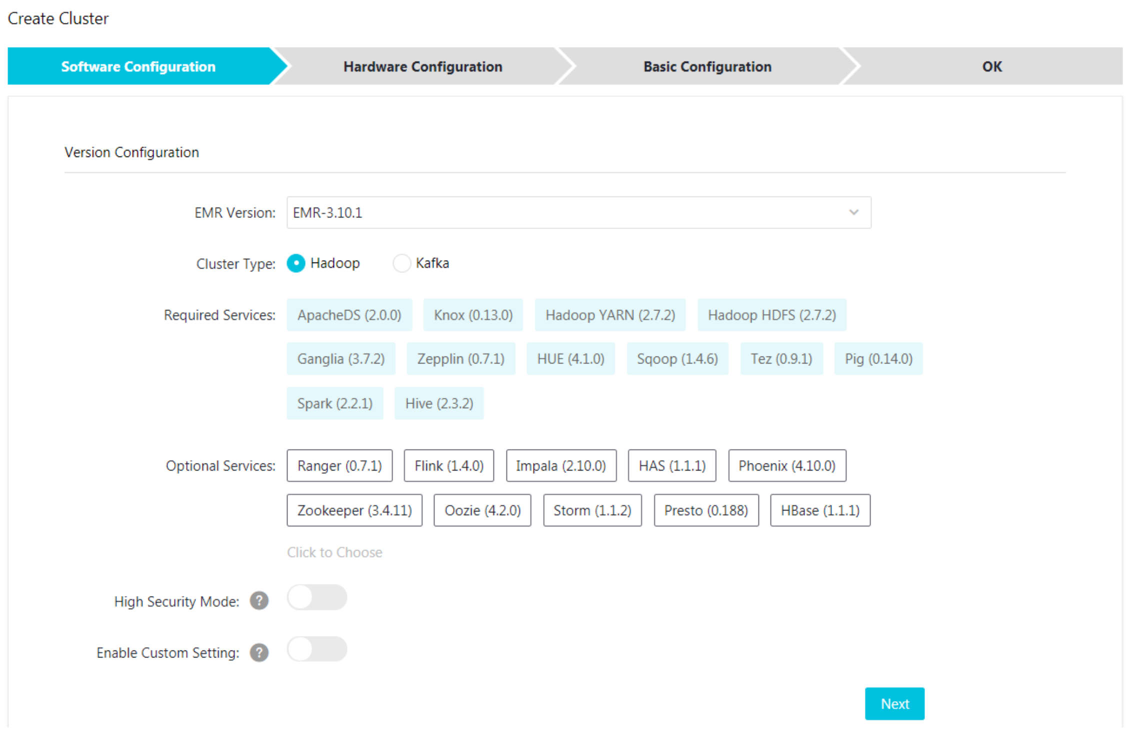
   The version information of E-MapReduce Hadoop is as follows:

   E-MapReduce version: EMR-3.10.1 or 3.11.0

   Cluster type: Hadoop

   Software(for EMR-3.11.0): HDFS2.7.2 / YARN2.7.2 / Hive2.3.3 / Ganglia3.7.2 / Spark2.2.1 / HUE4.1.0 / Zeppelin0.7.3 / Tez0.9.1 / Sqoop1.4.6 / Pig0.14.0 / ApacheDS2.0.0 / Knox0.13.0

   The network type of the Hadoop cluster is classic. The region is China East 1 (Hangzhou). The ECS compute resource of the master instance group is configured with an Internet IP address and an intranet IP address. The high availability mode is set to No (a non-HA mode). The following figure shows the configuration for EMR-3.10.1.



Create Cluster

Software Configuration | Hardware Configuration | Basic Configuration | OK

Version Configuration

EMR Version: EMR-3.10.1

Cluster Type: [✓ Hadoop] Kafka

Required Services: ApacheDS (2.0.0) Knox (0.13.0) Hadoop YARN (2.7.2) Hadoop HDFS (2.7.2) Ganglia (3.7.2) Zepplin (0.7.1) HUE (4.1.0) Sqoop (1.4.6) Tez (0.9.1) Pig (0.14.0) Spark (2.2.1) Hive (2.3.2)

Optional Services: Ranger (0.7.1) Flink (1.4.0) Impala (2.10.0) HAS (1.1.1) Phoenix (4.10.0) Zookeeper (3.4.11) Oozie (4.2.0) Storm (1.1.2) Presto (0.188) HBase (1.1.1)

Click to Choose

High Security Mode: ?

Enable Custom Setting: ?

Next

2. MaxCompute

For more information, see *Activate MaxCompute*.

Activate MaxCompute and create a project. In this topic, create a project named bigdata_DOC in China East 1 (Hangzhou) and enable the related DataWorks services for this project.

Prepare data

1. Create test data on the Hadoop cluster.

   In the E-MapReduce console, go to the Hadoop cluster page and use **Notebook** to create a notebook task. The table creation Hive statements in this example are as follows:

```
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'
```

   Click **run**. The test table `hive_doc_good_sale` is then successfully created on the E-MapReduce Hadoop cluster.

   Insert the test data. You can select data from OSS or other data sources, or manually insert a small amount of test data. The following data can be manually inserted:

```
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,'Toiletries','Brand A','hanmeimei',1,442.5,1),('2018-08
-22','Fresh food','Brand D','hanmeimei',2,234,3),('2018-08-23','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 food','Brand G','peiqi',10,5560,7),('2018-08-
24','Sanitary ware','Brand F','peiqi',1,445.6,2),('2018-08-24','Coat
','Brand A','ray',3,777,3),('2018-08-24','Sanitary ware','Brand G','
ray',3,122,3),('2018-08-24','Coat','Brand C','ray',1,62,7) ;
```

After inserting the data, you can use the `select * from hive_doc_good_sale where pt =1;` statement to check whether the data exists in the Hadoop cluster table for migration.

2. Use DataWorks to create a destination table.

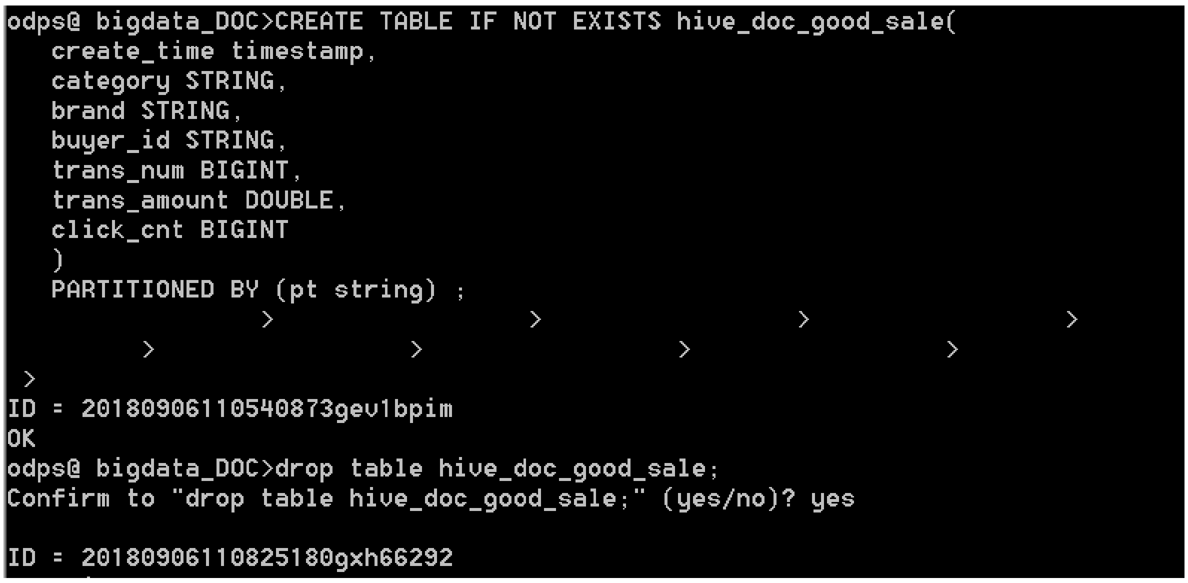
   In the DataWorks console, click the MaxCompute project, and choose **Data Development** > **New** > **Create Table**.

   In the displayed window, enter the following table creation SQL statements:

```
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 creating a table, pay attention to the mapping between the Hive data type and the MaxCompute data type.

   The data synchronization feature of DataWorks does not support **`timestamp`** data. Therefore, in the DataWorks table creation statements, create_time is set to a string value. You can also use the odpscmd Command Line (CLI) tool to create a table. For more information about how to install and configure the tool, see *Install and configure a client*. The table creation process is as follows.



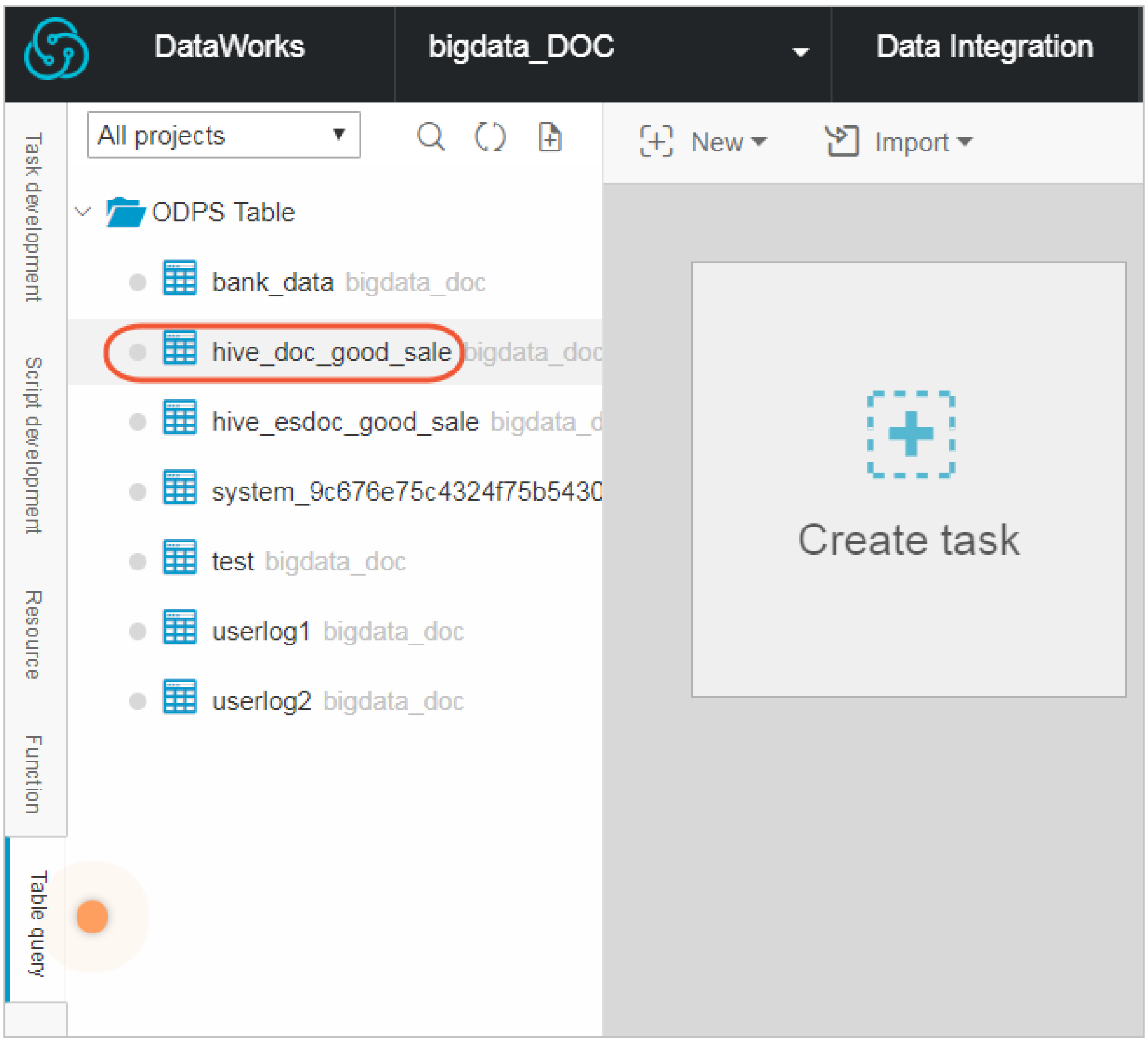
```
odps@ bigdata_DOC>CREATE TABLE IF NOT EXISTS hive_doc_good_sale(
   create_time timestamp,
   category STRING,
   brand STRING,
   buyer_id STRING,
   trans_num BIGINT,
   trans_amount DOUBLE,
   click_cnt BIGINT
   )
   PARTITIONED BY (pt string) ;
                  >                  >        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

Considering the compatibility of the Hive and MaxCompute data types, we recommend that you run the following command on the **odpscmd** client:

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

After the table is created, you can choose **Data Development** > **Table Query** in the DataWorks console to view the table created in MaxCompute, as shown in the following figure.



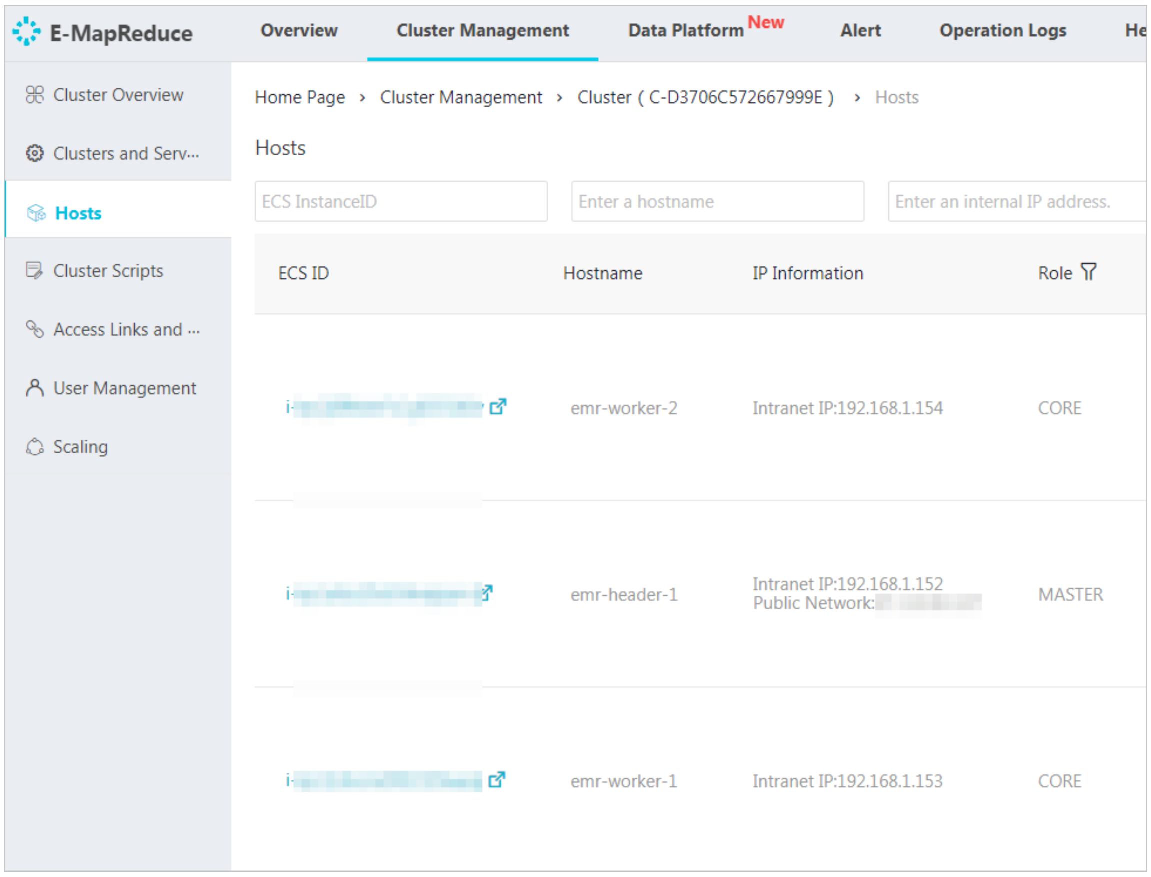
Synchronize data

1. Create a custom resource group.

   In most cases, the network between the project data node of MaxCompute and the data node of the Hadoop cluster is not connected. You can customize a resource group to execute the synchronization task of DataWorks on the master node of the Hadoop cluster. (In general, the network between the master node and the data node on the Hadoop cluster is connected).

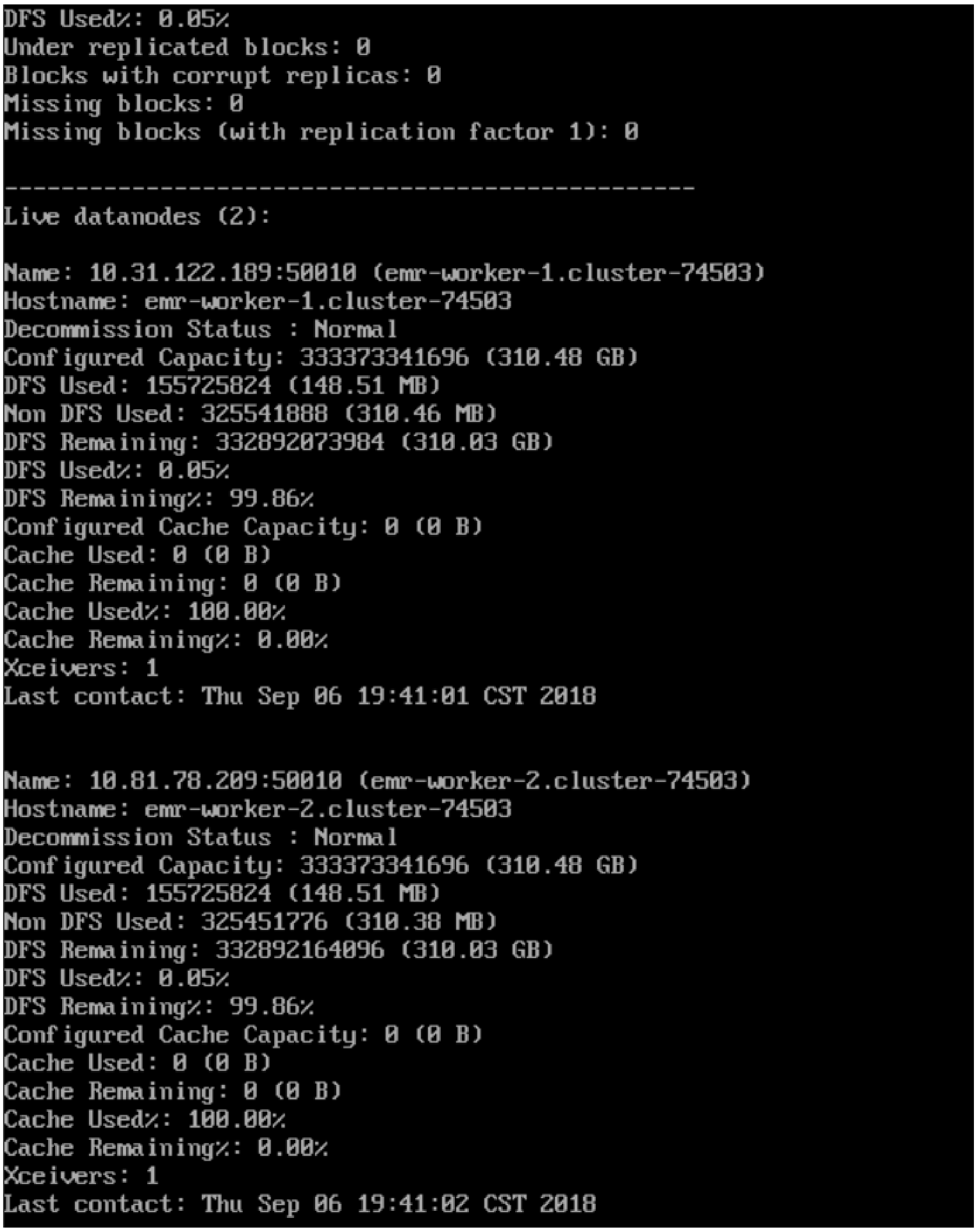
a. View the data node of the Hadoop cluster.

On the home page of the E-MapReduce console, choose Cluster Management > Cluster > Hosts. You can view the data node of the Hadoop cluster. As shown in the following figure, the host name of the master node on the E-MapReduce Hadoop cluster (non-HA mode) is emr-header-1, and the host name of the data node is emr-worker-X.



ECS ID	Hostname	IP Information	Role
i-	emr-worker-2	Intranet IP:192.168.1.154	CORE
i-	emr-header-1	Intranet IP:192.168.1.152 Public Network:	MASTER
i-	emr-worker-1	Intranet IP:192.168.1.153	CORE

You can also click the ECS ID of the master node, click Connect on the displayed ECS details page, and run the hadoop dfsadmin –report command to view the data node, as shown in the following figure.

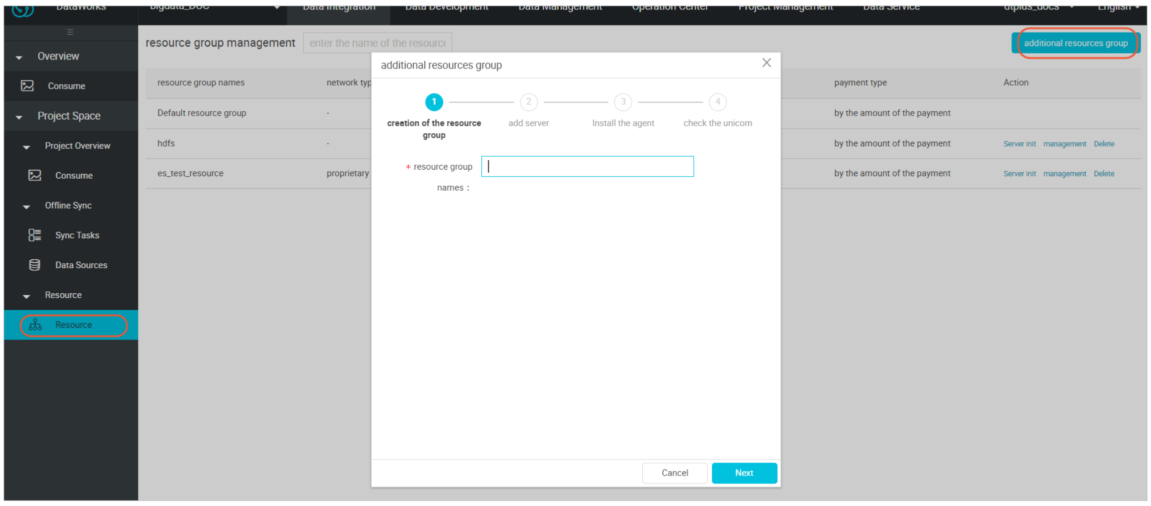


```
DFS Used%: 0.05%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-------------------------------------------------
Live datanodes (2):

Name: 10.31.122.189:50010 (emr-worker-1.cluster-74503)
Hostname: emr-worker-1.cluster-74503
Decommission Status : Normal
Configured Capacity: 333373341696 (310.48 GB)
DFS Used: 155725824 (148.51 MB)
Non DFS Used: 325541888 (310.46 MB)
DFS Remaining: 332892073984 (310.03 GB)
DFS Used%: 0.05%
DFS Remaining%: 99.86%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Sep 06 19:41:01 CST 2018


Name: 10.81.78.209:50010 (emr-worker-2.cluster-74503)
Hostname: emr-worker-2.cluster-74503
Decommission Status : Normal
Configured Capacity: 333373341696 (310.48 GB)
DFS Used: 155725824 (148.51 MB)
Non DFS Used: 325451776 (310.38 MB)
DFS Remaining: 332892164096 (310.03 GB)
DFS Used%: 0.05%
DFS Remaining%: 99.86%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu Sep 06 19:41:02 CST 2018
```

As shown in the preceding figure, the data node has only an intranet address and cannot communicate with the default resource group of DataWorks. Therefore, you need to customize a resource group and set the master node to a node that executes the synchronization task of DataWorks.

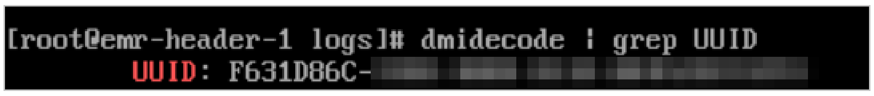
b. Create a custom resource group.

In the DataWorks console, go to the Data Integration page, select Resource Group, and click New Resource Groups, as shown in the following figure. For more information about how to customize a resource group, see *Add scheduling resources*.



resource group names	network typ	payment type	Action
Default resource group	-	by the amount of the payment	
hdfs	-	by the amount of the payment	Server init management Delete
es_test_resource	proprietary	by the amount of the payment	Server init management Delete

When you add a server, you need to enter the information such as the ECS UUID and machine IP address. (For a classic network, enter the sever name. For a VPC network, enter the server UUID. You can add scheduling resources only for classic networks in China East 2 (Shanghai) in DataWorks V2.0. In other regions, select the VPC network type when you add a scheduling resource group regardless of whether your network type is classic or VPC.) Set the machine IP address to the Internet IP address of the master node because the intranet IP address might be inaccessible. The ECS UUID must be connected to the master node management terminal, which can be obtained by running the dmidecode | grep UUID command, as shown in the following figure. (You can use the same method if your Hadoop cluster is not built on E-MapReduce.)



```
[root@emr-header-1 logs]# dmidecode | grep UUID
        UUID: F631D86C-[redacted]
```

After adding the server, make sure that the network between the master node and DataWorks is connected. If you are using an ECS server, you must set the server security group. If you are using an intranet IP address for communication, set the server security group. For more information, see *Adding security groups*.

If you are using an Internet IP address, you can directly set the Internet ingress and egress under **Security Group Rules**(In practical application scenarios, we recommend that you set detailed bypass rules for your data security.)

After completing the preceding steps, install the custom resource group agent as prompted. If the state is **available**, the custom resource group is added successfully.

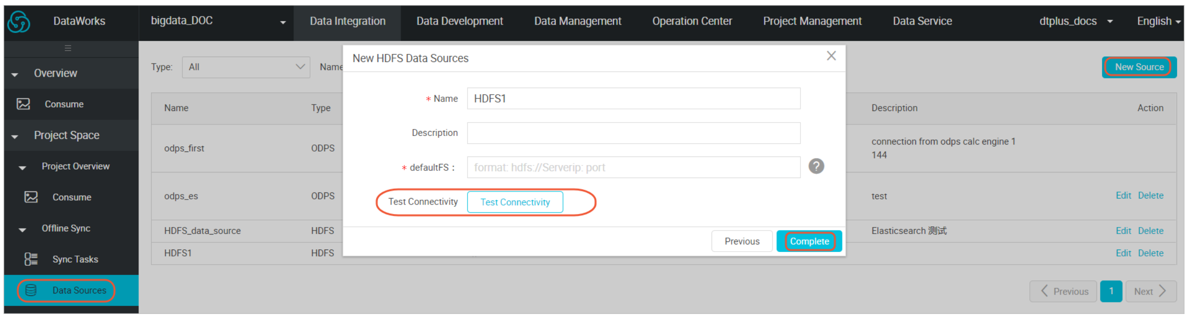
If the state is **unavailable**, you can log on to the master node, and run the `tail -f/home/admin/alisatasknode/logs/heartbeat.log` command to check whether the heartbeat message between DataWorks and the master node has timed out, as shown in the following figure.

2. Create a data source.

   For more information about how to create a data source in DataWorks, see *Configuring Data Source*.

   After you create a project in DataWorks, the data source is set to **`odps_first`** by default. Therefore, you only need to add a Hadoop cluster data source. To do so, perform the following steps: On the Data Integration page of DataWorks, choose **Data Source** > **New Source**, and select **HDFS**.

   In the displayed window, enter the data source name and defaultFS. If the E-MapReduce Hadoop cluster is an HA cluster, the address is IP:8020 of hdfs://emr-header-1. If the E-MapReduce Hadoop cluster is a non-HA cluster, the address is IP:9000 of hdfs://emr-header-1. In this topic, emr-header-1 is connected to DataWorks through the Internet. Therefore, enter the Internet IP address and open the security group.



New HDFS Data Sources

* Name	HDFS1
Description	
* defaultFS :	format: hdfs://Serverip: port

Type: All

Name	Type	Description	Action
odps_first	ODPS	connection from odps calc engine 1 144	
odps_es	ODPS	test	Edit Delete
HDFS_data_source	HDFS	Elasticsearch 测试	Edit Delete
HDFS1	HDFS		Edit Delete

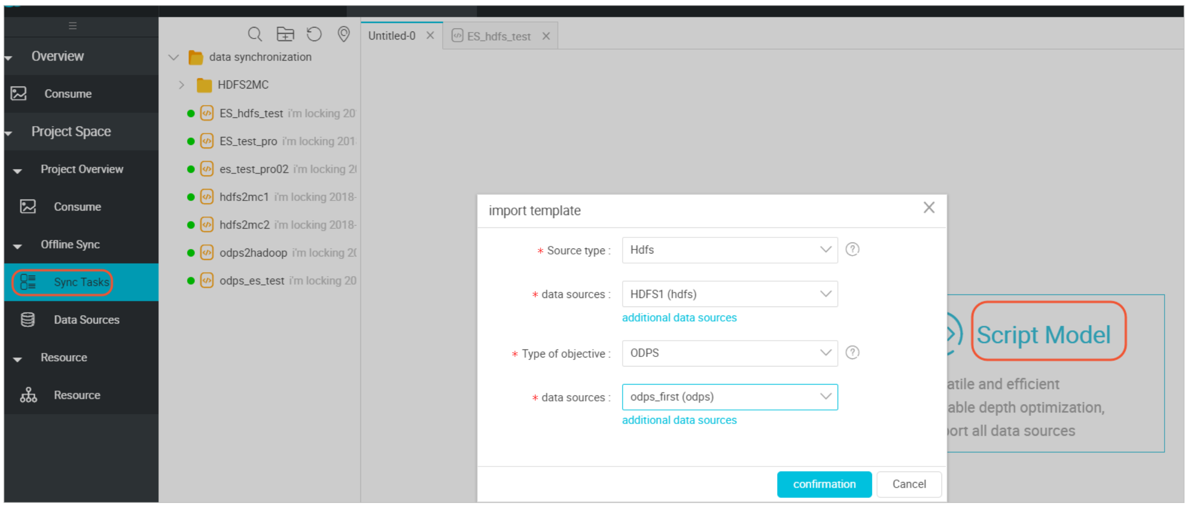
   After the configuration is completed, click Test Connectivity. If **Test connectivity successfully** is displayed, the data source is added successfully.

   Note:

   If the network type of the E-MapReduce Hadoop cluster is VPC, the connectivity test is not supported.

3. Configure the data synchronization task.

On the Data Integration page of DataWorks, click **Sync Tasks** and create a script mode. In the displayed window, select a data source, as shown in the following figure.



import template

* Source type : Hdfs
* data sources : HDFS1 (hdfs)
additional data sources
* Type of objective : ODPS
* data sources : odps_first (odps)
additional data sources

confirmation Cancel

After the template is imported, the synchronization task is converted to the script mode. For more information, see *Script Mode*.

When you configure the data synchronization task script, the data types of the DataWorks synchronization task and the Hive table are as follows.

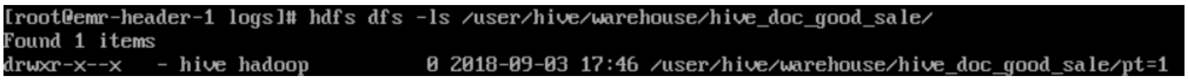
Data type in the Hivetable	Data type in DataX / DataWorks
TINYINT,SMALLINT,INT,BIGINT	Long
FLOAT,DOUBLE,DECIMAL	Double
String,CHAR,VARCHAR	String
BOOLEAN	Boolean
Date,TIMESTAMP	Date
Binary	Binary

The code details are as follows:

```
{
  "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://121.199.11.138:9000",
          "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",
          "dmu": 1
        }
      }
    },
    "type": "job",
    "version": "1.0"
```

```
}
```

The path parameter indicates the place where the data is stored in the 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 confirm the place. For a partition table, you do not need to specify the partitions. The data synchronization feature of DataWorks can automatically recurse to the partition path, as shown in the following figure.



```
[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
```

After the configuration is completed, click **Run**. If a message is displayed indicating that the task is executed successfully, the synchronization task is completed. If a message is displayed indicating that the task failed to be executed, copy the logs for further troubleshooting.

Verify the results

In the DataWorks console, choose **Data Development** > **Table Query** and select the **`hive_doc_good_sal`**e table. You can check whether the Hive data has been synchronized to MaxCompute. You can also create a table query task, enter the `select * FROM hive_doc_good_sale where pt =1;` script in the task, and click Run to query the results.

You can also enter `select * FROM hive_doc_good_sale where pt =1;` in the **odpscmd** CLI tool to query the table results.

Migrate data from MaxCompute to Hadoop

To migrate data from MaxCompute to Hadoop, perform the preceding steps but exchange the reader and writer objects in the synchronization script. The following is an example:

```
{
  "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://47.99.162.100:9000",
     "writeMode": "append",
     "fieldDelimiter": ",",
     "encoding": "UTF-8",
     "fileType": "text"
     }
   },
   "setting": {
     "errorLimit": {
       "record": "1000"
   },
   "speed": {
     "throttle": false,
     "concurrent": 1,
     "mbps": "1",
     "dmu": 1
   }
   }
 },
 "type": "job",
 "version": "1.0"
```

```
}
```

Before executing the preceding synchronization task, you must set the Hadoop cluster. For more information, see *Configure HDFS Writer*. After executing the synchronization task, you need to manually copy the synchronized files.

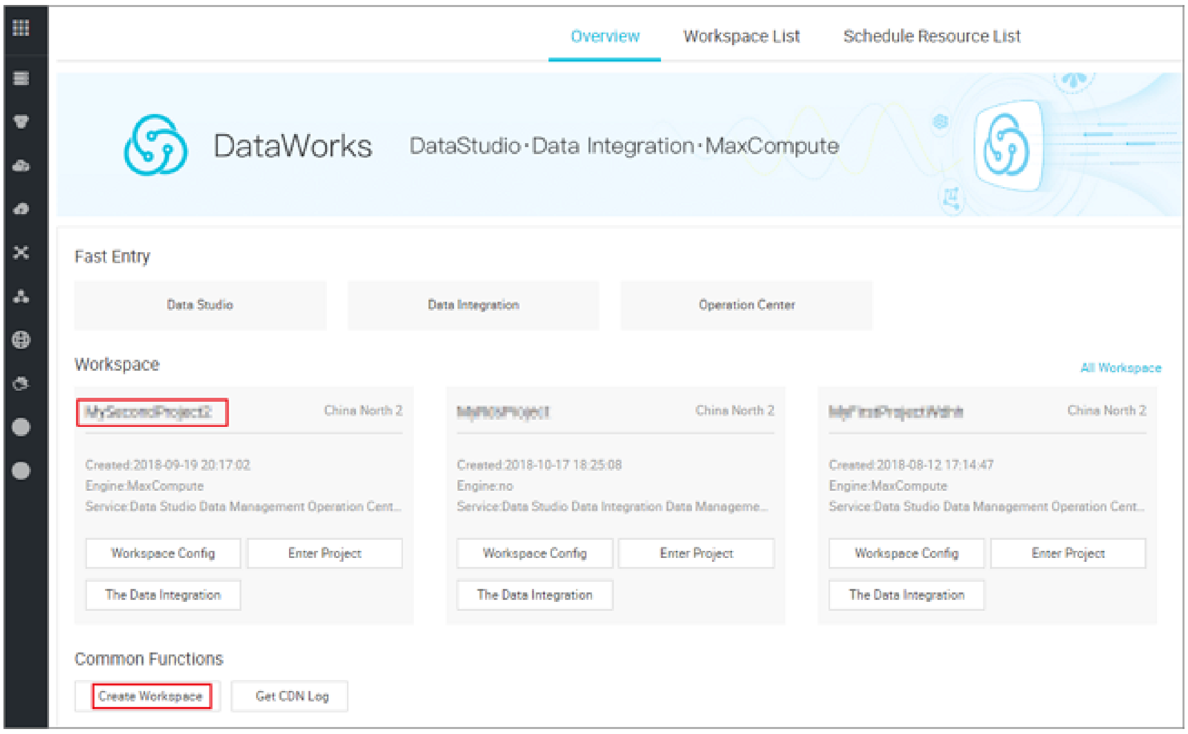
9 Migrate data from RDS to MaxCompute to implement dynamic partitioning

This topic describes how to use the data synchronization feature of DataWorks to automatically create partitions and dynamically migrate data from RDS to MaxCompute.

Preparations

1. MaxCompute

   Activate MaxCompute and create a workspace. In this example, create a workspace in China North 2 (Beijing) and enable related DataWorks services for this workspace, as shown in the following figure.



> **Note:**
> If you are using DataWorks for the first time, you need to complete the operations described in *Preparation*. For example, you need to get your account ready, set the workspace role, and configure the workspace. For more information about how to activate MaxCompute, see *Activate MaxCompute*. After you have completed the DataWorks preparation, log on to the DataWorks console, select the corresponding workspace, and click **Enter Project** for data development.

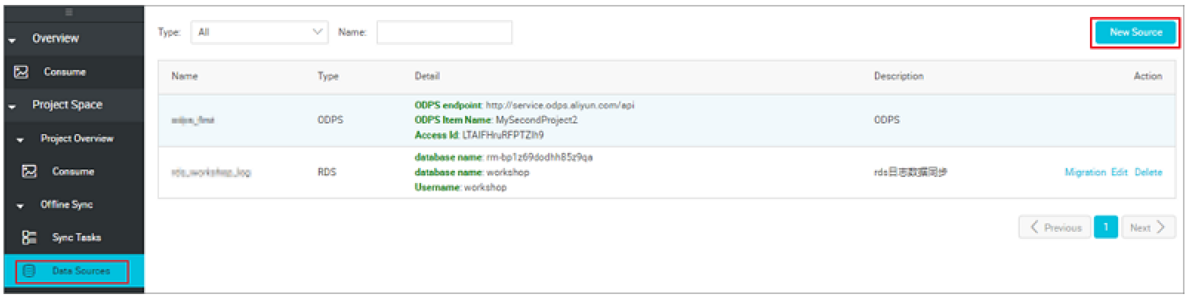
2. Add data sources.

a. Add the RDS data source.

Log on the DataWorks console, select *Data Integration*, and add the *MySQL data source*.

b. Add the ODPS data source. For more information, see *Configure MaxCompute data source*.

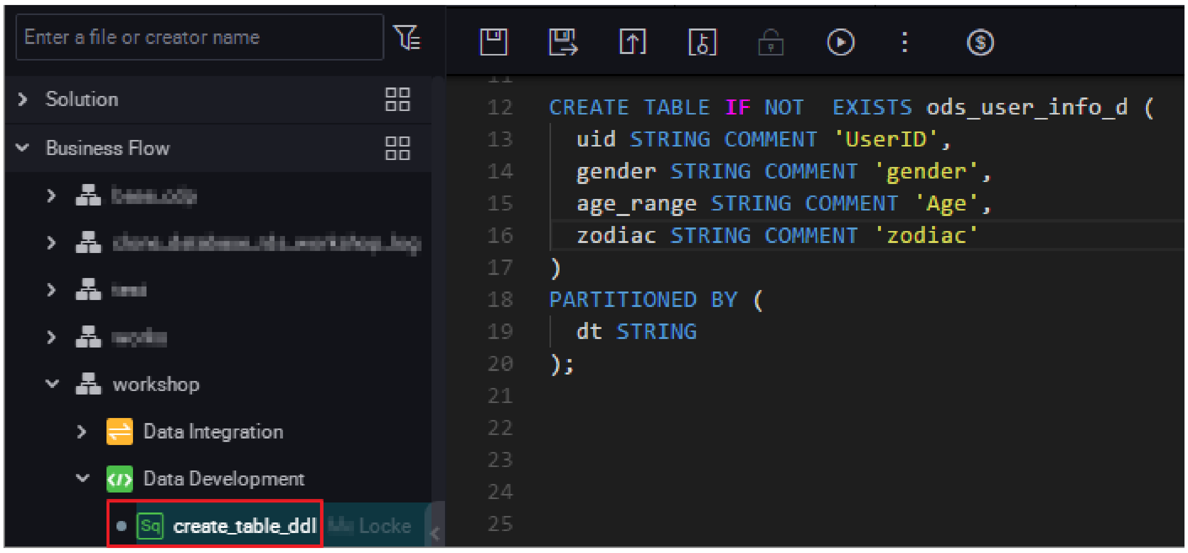
Verify that the configuration is completed, as shown in the following figure.



Name	Type	Detail	Description	Action
[redacted]	ODPS	**ODPS endpoint**: http://service.odps.aliyun.com/api **ODPS Item Name**: MySecondProject2 **Access Id**: LTAIFHruRFPTZh9	ODPS	
[redacted]	RDS	**database name**: rm-bp1z69dodhh85z9qa **database name**: workshop **Username**: workshop	rds日志数据同步	Migration Edit Delete

3. Make sure that a table is available in the destination ODPS database.

In the ODPS database, create a destination table named ods_user_info_d. This table corresponds to a table in RDS. Under **Data Development**, right-click **New ODPS SQL node**, create a node named create_table_ddl, and enter the table creation statements, as shown in the following figure.

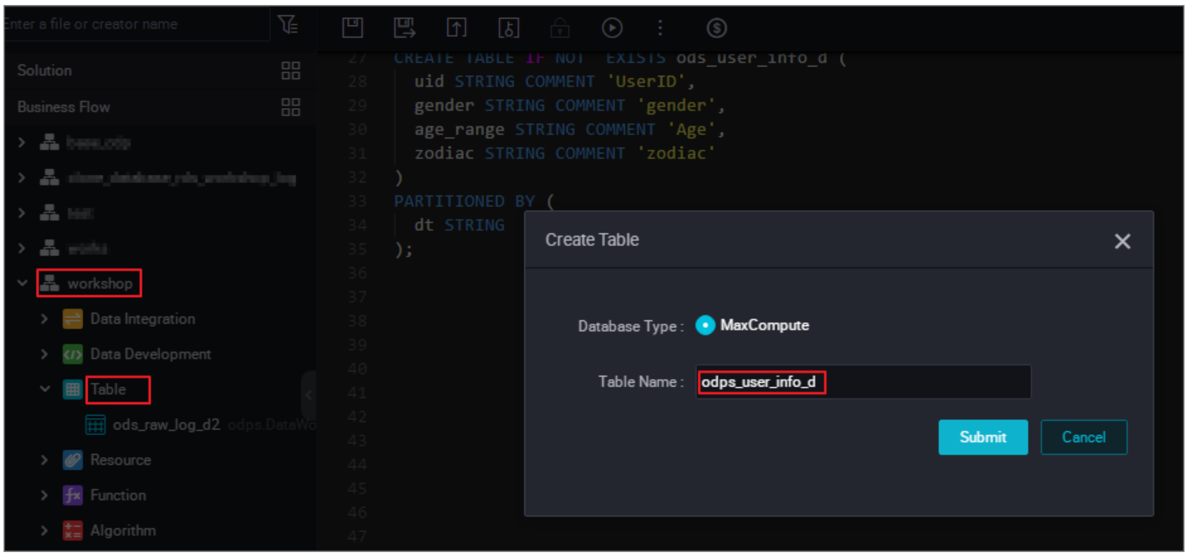


```
CREATE TABLE IF NOT  EXISTS ods_user_info_d (
  uid STRING COMMENT 'UserID',
  gender STRING COMMENT 'gender',
  age_range STRING COMMENT 'Age',
  zodiac STRING COMMENT 'zodiac'
)
PARTITIONED BY (
  dt STRING
);
```

The SQL statements are as follows:

```
CREATE TABLE IF NOT EXISTS ods_user_info_d (
uid STRING COMMENT 'UserID',
gender STRING COMMENT 'gender',
age_range STRING COMMENT 'Age',
zodiac STRING COMMENT 'zodiac'
)
PARTITIONED BY (
dt STRING
);
```

You can also select **Create Table** under **Business Flow** > **Table**, as shown in the following figure.



```
CREATE TABLE IF NOT  EXISTS ods_user_info_d (
  uid STRING COMMENT 'UserID',
  gender STRING COMMENT 'gender',
  age_range STRING COMMENT 'Age',
  zodiac STRING COMMENT 'zodiac'
)
PARTITIONED BY (
  dt STRING
);
```

Create Table

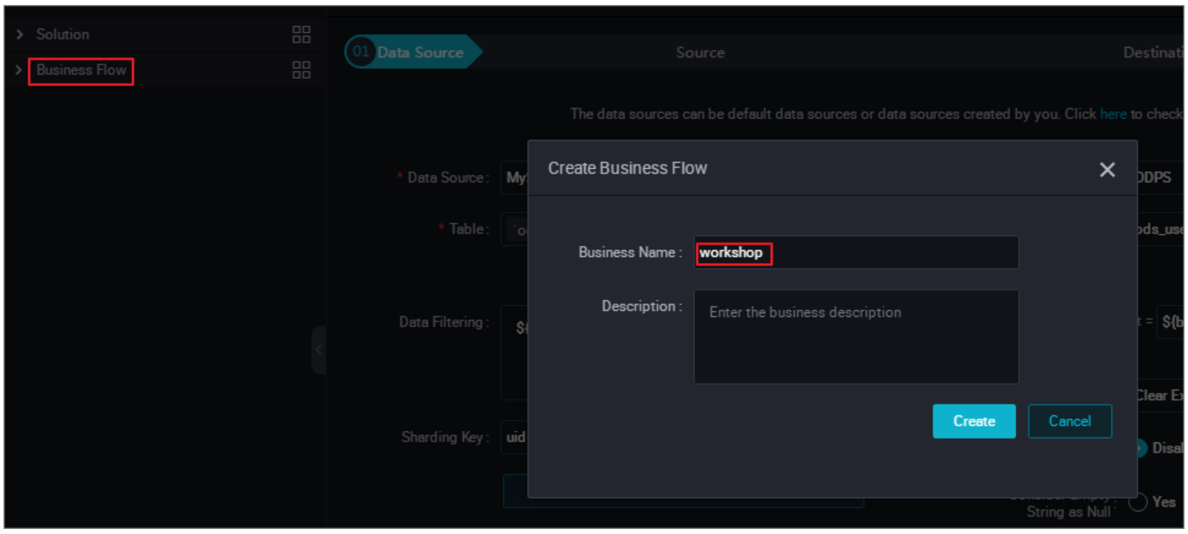
Database Type : MaxCompute

Table Name : odps_user_info_d

For more information, see *Create a table and upload data*.

4. Create a business flow.

   Choose **Data Development** > **Business Flow** and click **Create Business Flow** to create a workshop, as shown in the following figure.



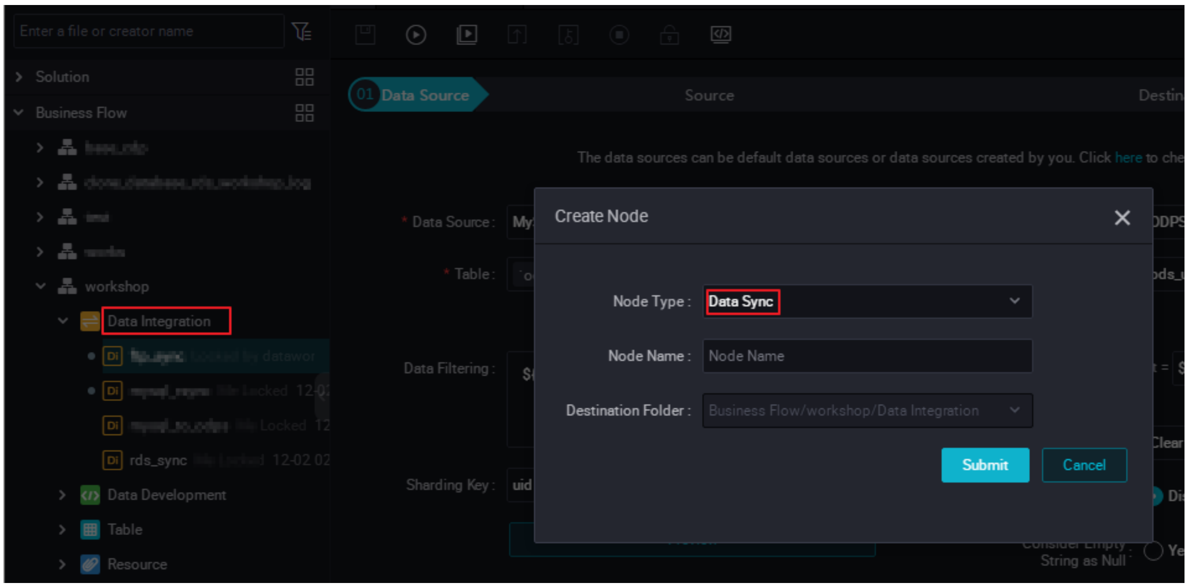
Create Business Flow

Business Name : workshop

Description : Enter the business description

5. Create and configure a synchronization task node.

   Create a synchronization node named rds_sync under the workshop business flow, as shown in the following figure.



Node Type : Data Sync

Node Name : Node Name

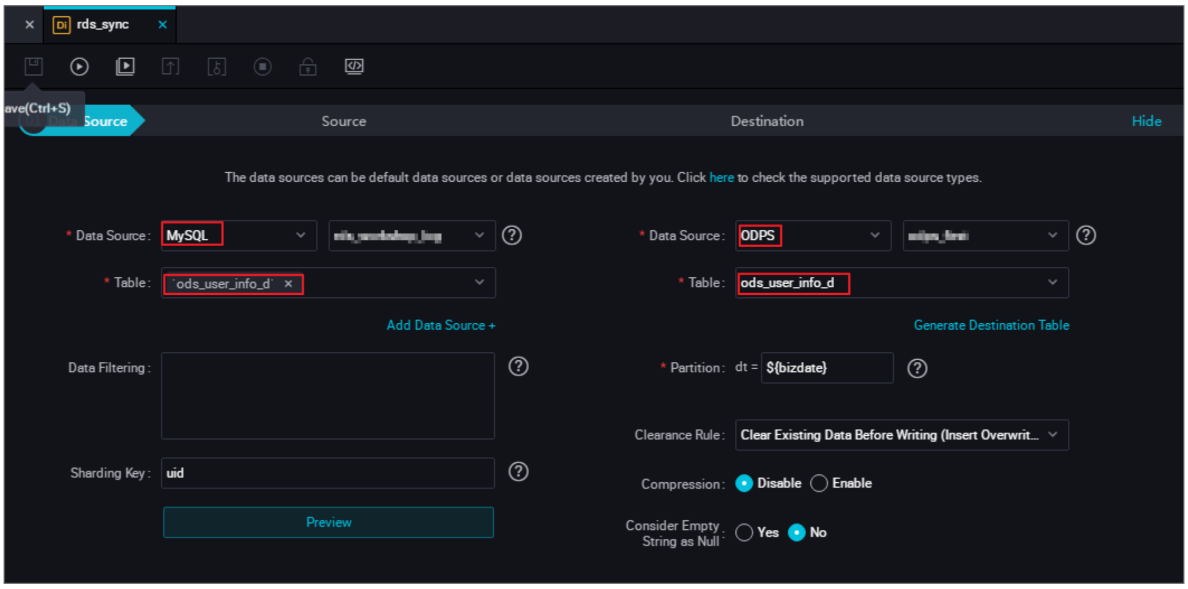
Destination Folder : Business Flow/workshop/Data Integration

For more information about how to configure a data synchronization task, see *DataWorks data development and O&M*.

Automatic creation of partitions

After the preparations are completed, the data in RDS needs to be synchronized to MaxCompute every day, so that the date-based partition can be automatically created.

1. Select the data source and data destination, as shown in the following figure.

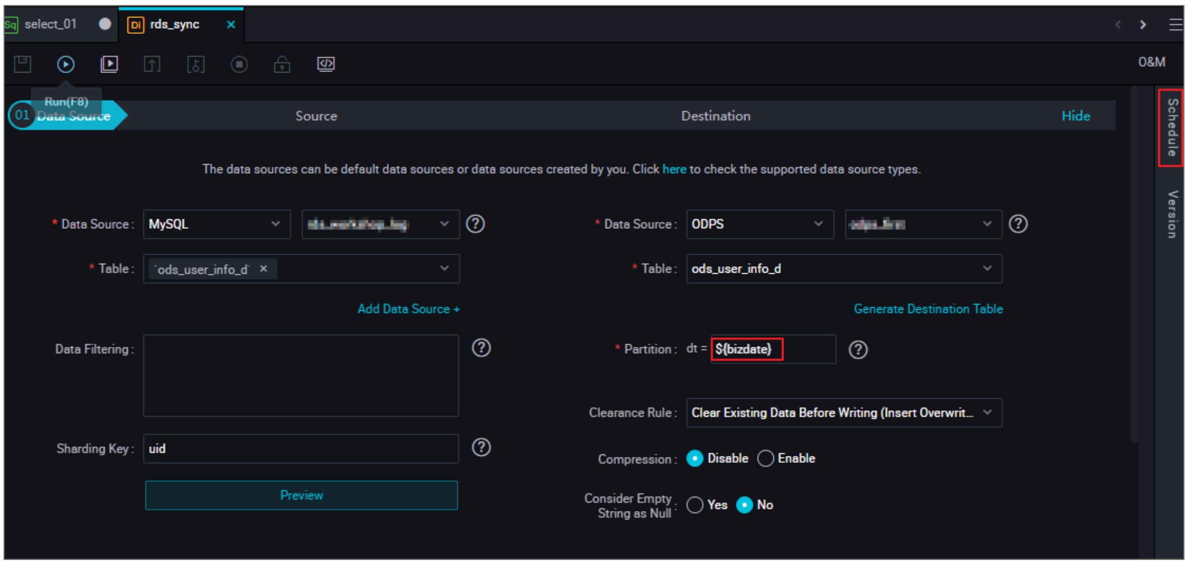


The data sources can be default data sources or data sources created by you. Click here to check the supported data source types.

Source:
- * Data Source : MySQL [redacted]
- * Table : `ods_user_info_d`
- Data Filtering :
- Sharding Key : uid

Destination:
- * Data Source : ODPS [redacted]
- * Table : ods_user_info_d
- * Partition : dt = ${bizdate}
- Clearance Rule : Clear Existing Data Before Writing (Insert Overwrit...
- Compression : [✓ Disable] Enable
- Consider Empty String as Null : Yes [✓ No]

2. Set the parameters, as shown in the following figure.

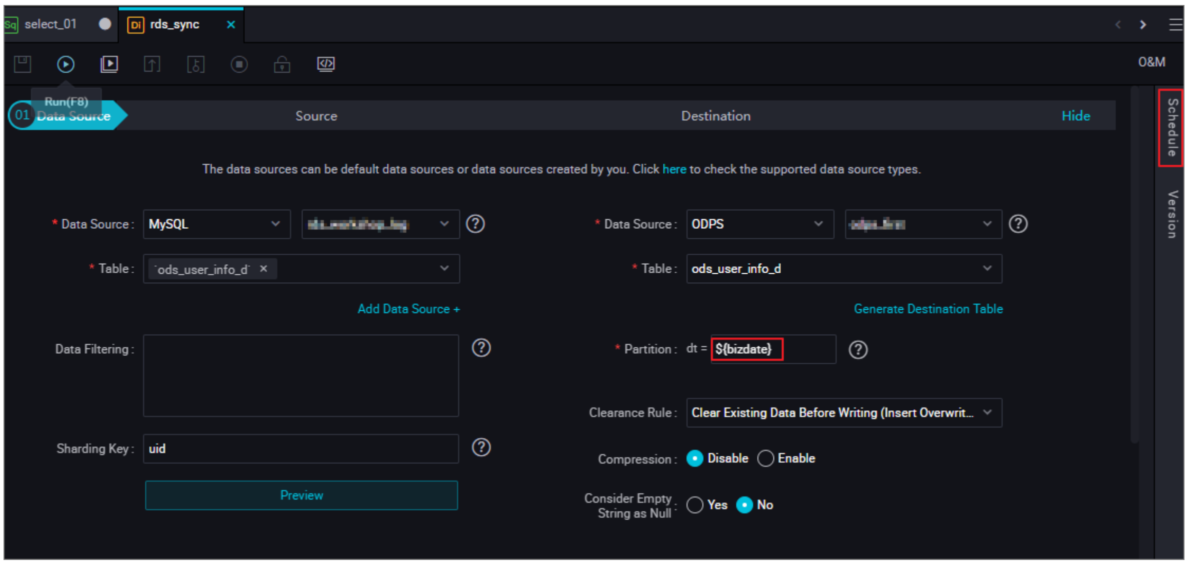


Source | Destination

The data sources can be default data sources or data sources created by you. Click here to check the supported data source types.

* Data Source :	MySQL [redacted]
* Table :	`ods_user_info_d`
Data Filtering :	
Sharding Key :	uid
* Data Source :	ODPS [redacted]
* Table :	ods_user_info_d
* Partition :	dt = ${bizdate}
Clearance Rule :	Clear Existing Data Before Writing (Insert Overwrit...
Compression :	[✓ Disable] Enable
Consider Empty String as Null :	Yes [✓ No]

The system time parameter `${bizdate}` is generally used by default. The format is yyyymmdd. This parameter corresponds to the **Partition** parameter under **Destination**. The partition date is called business date. In most cases, users process the business data generated in the previous day. Therefore, when the data synchronization task is scheduled and executed, the partition date is automatically replaced with the date one day before the task execution date. To use the task execution date as the partition value (partition date), you must select **Schedule** to customize the parameter, as shown in the following figure.



Source | Destination

The data sources can be default data sources or data sources created by you. Click here to check the supported data source types.

* Data Source :	MySQL [redacted]
* Table :	`ods_user_info_d`
Data Filtering :	
Sharding Key :	uid
* Data Source :	ODPS [redacted]
* Table :	ods_user_info_d
* Partition :	dt = ${bizdate}
Clearance Rule :	Clear Existing Data Before Writing (Insert Overwrit...
Compression :	[✓ Disable] Enable
Consider Empty String as Null :	Yes [✓ No]

The custom parameter can be configured in different formats. By default, the real-time date is displayed. You can select a date and a format as needed. The custom parameter can be set in one of the following formats:

N years later: `$[add_months(yyyymmdd,12*N)]`

N years ago: `$[add_months(yyyymmdd,-12*N)]`

N months later: `$[add_months(yyyymmdd,N)]`

N months ago: `$[add_months(yyyymmdd,-N)]`

N weeks later: `$[yyyymmdd+7*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]`

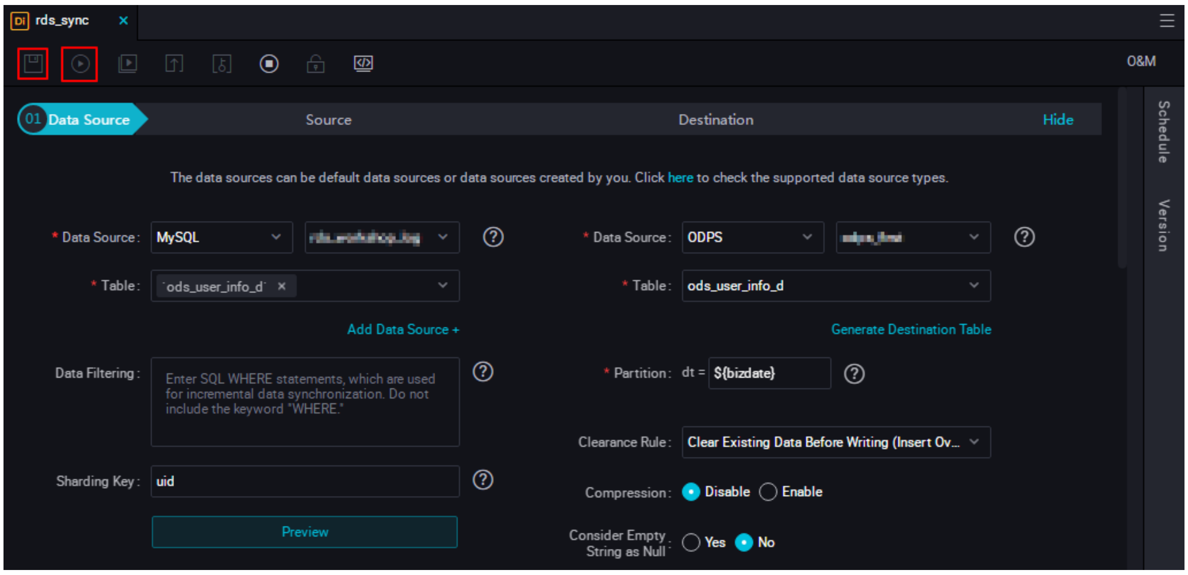
Note:

- You need to use brackets ([]) to edit the value calculation formula of the custom parameter, for example, `key1=$[yyyy-mm-dd]`.
- The default calculation unit of the custom parameter is day. For example, `$[hh24miss-N/24/60]` indicates the calculation result of `(yyyymmddhh24miss-(N/24/60 * 1 day))`. The hour, minute, and second are in the format of hh24miss.
- The calculation unit of add_months is month. For example, `$[add_months(yyyymmdd,12 N)-M/24/60]` indicates the calculation result of `(yyyymmddhh24miss-(12 N 1 month))-(M/24/60 1 day)`. The year, month, and day are in the format of `yyyymmdd`.

For more information, see *Parameter configuration*.

3. Perform the test run.

   Click **Save** to save all configurations, and click **Run**, as shown in the following figure.



rds_sync ×

O&M

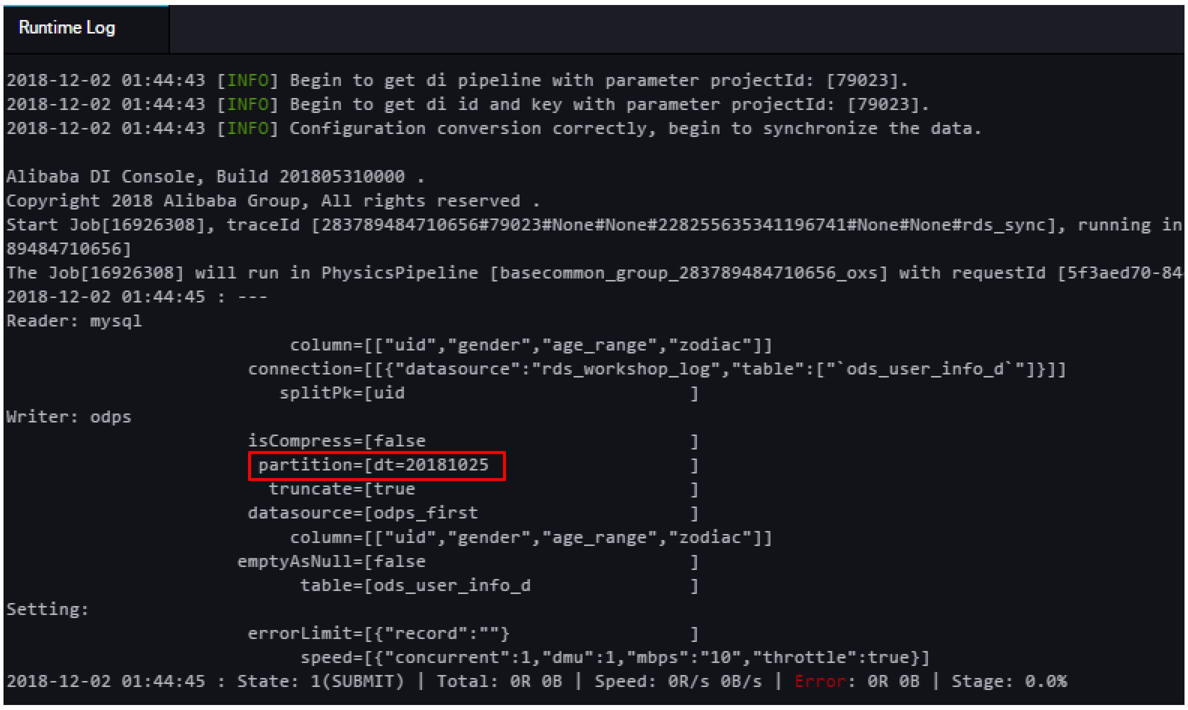
01 Data Source | Source | Destination | Hide

Schedule | Version

The data sources can be default data sources or data sources created by you. Click here to check the supported data source types.

Source		Destination	
* Data Source:	MySQL [redacted]	* Data Source:	ODPS [redacted]
* Table:	`ods_user_info_d` ×	* Table:	ods_user_info_d
	Add Data Source +		Generate Destination Table
Data Filtering:	Enter SQL WHERE statements, which are used for incremental data synchronization. Do not include the keyword "WHERE."	* Partition:	dt = ${bizdate}
		Clearance Rule:	Clear Existing Data Before Writing (Insert Ov...
Sharding Key:	uid	Compression:	[✓ Disable] Enable
	Preview	Consider Empty String as Null:	Yes [✓ No]

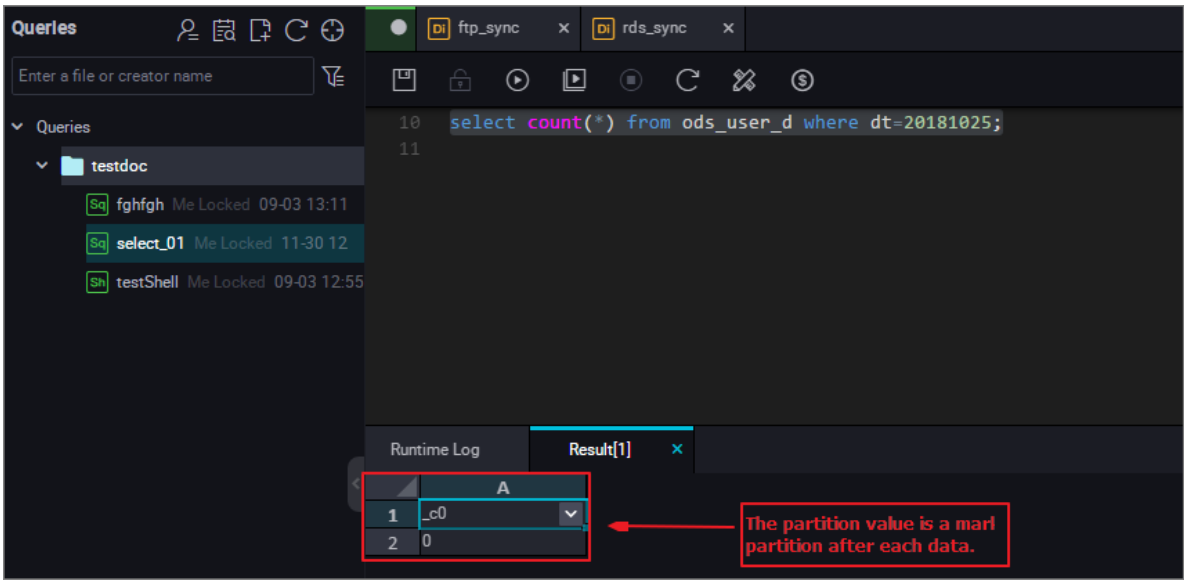
View the running log, as shown in the following figure.



Runtime Log

```
2018-12-02 01:44:43 [INFO] Begin to get di pipeline with parameter projectId: [79023].
2018-12-02 01:44:43 [INFO] Begin to get di id and key with parameter projectId: [79023].
2018-12-02 01:44:43 [INFO] Configuration conversion correctly, begin to synchronize the data.

Alibaba DI Console, Build 201805310000 .
Copyright 2018 Alibaba Group, All rights reserved .
Start Job[16926308], traceId [283789484710656#79023#None#None#228255635341196741#None#None#rds_sync], running in
89484710656]
The Job[16926308] will run in PhysicsPipeline [basecommon_group_283789484710656_oxs] with requestId [5f3aed70-84
2018-12-02 01:44:45 : ---
Reader: mysql
                           column=[["uid","gender","age_range","zodiac"]]
                       connection=[[{"datasource":"rds_workshop_log","table":["`ods_user_info_d`"]}]]
                          splitPk=[uid                              ]
Writer: odps
                       isCompress=[false                            ]
                        partition=[dt=20181025                      ]
                         truncate=[true                             ]
                       datasource=[odps_first                       ]
                           column=[["uid","gender","age_range","zodiac"]]
                      emptyAsNull=[false                            ]
                            table=[ods_user_info_d                  ]
Setting:
                       errorLimit=[{"record":""}                    ]
                            speed=[{"concurrent":1,"dmu":1,"mbps":"10","throttle":true}]
2018-12-02 01:44:45 : State: 1(SUBMIT) | Total: 0R 0B | Speed: 0R/s 0B/s | Error: 0R 0B | Stage: 0.0%
```

In the sample log shown in the preceding figure, you can find that the partition value in MaxCompute (whose printed name is ODPS) information is dt=20181025. This indicates that the partition value is automatically replaced. Verify that the data is successfully migrated to the ODPS table, as shown in the following figure.



```
select count(*) from ods_user_d where dt=20181025;
```

	A
1	_c0
2	0

The partition value is a marl partition after each data.

Queries
Enter a file or creator name
Queries
testdoc
fghfgh Me Locked 09-03 13:11
select_01 Me Locked 11-30 12
testShell Me Locked 09-03 12:55
ftp_sync
rds_sync
Runtime Log
Result[1]

Note:

In MaxCompute 2.0, parameter settings are required for partition table query. Full query is not supported. The SQL statements are as follows:

```
--Check whether the data is successfully written to MaxCompute.
select count(*)from ods_user_info_d where dt=business date;
```

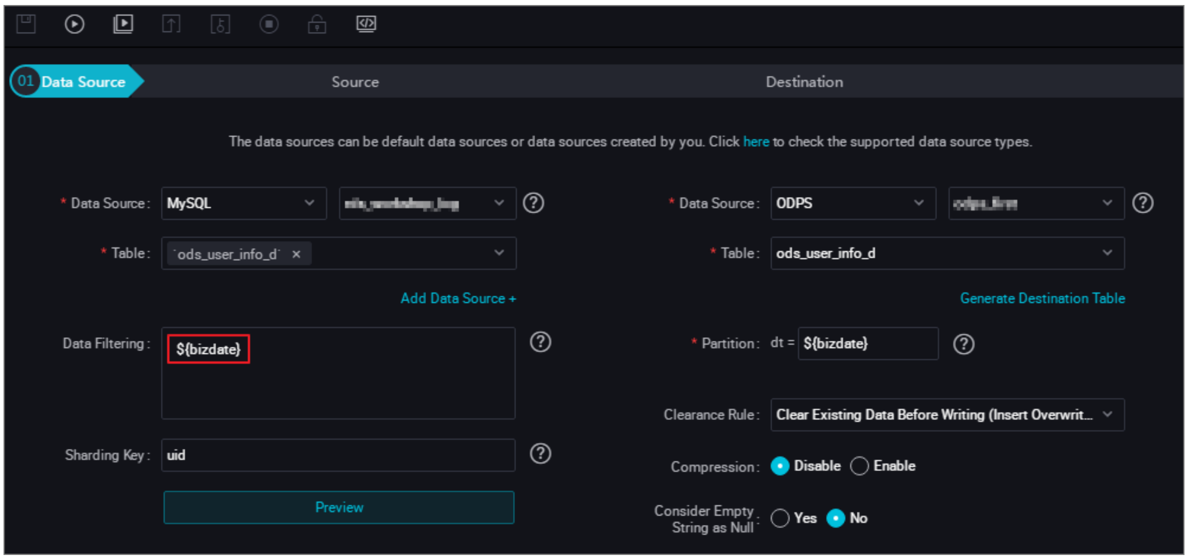
For more information about the **SELECT** command, see *Select operation*.

Now you can see that the data has been migrated to the ODPS table and a partition value has been successfully created. Then, when the task is executed at scheduled time, the data in RDS is automatically synchronized to the date-based partition in MaxCompute.

Data patching

If you have many historical data that is generated before the execution date, and you want to implement automatic synchronization and partitioning, you can log on to the DataWorks console, click **O&M**, select the data synchronization node, and click **Patch Data**.

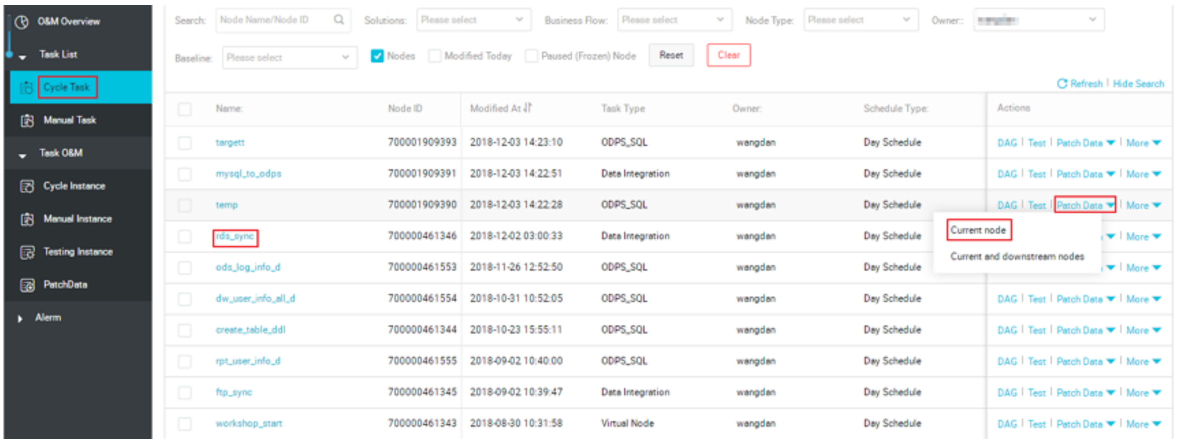
1. Filter the historical data in RDS by date. For example, filter the historical data generated on 2018-09-13, so that the data can be automatically synchronized to the 20180825 partition in MaxCompute. You can use a WHERE clause to filter data in RDS, as shown in the following figure.



The data sources can be default data sources or data sources created by you. Click here to check the supported data source types.

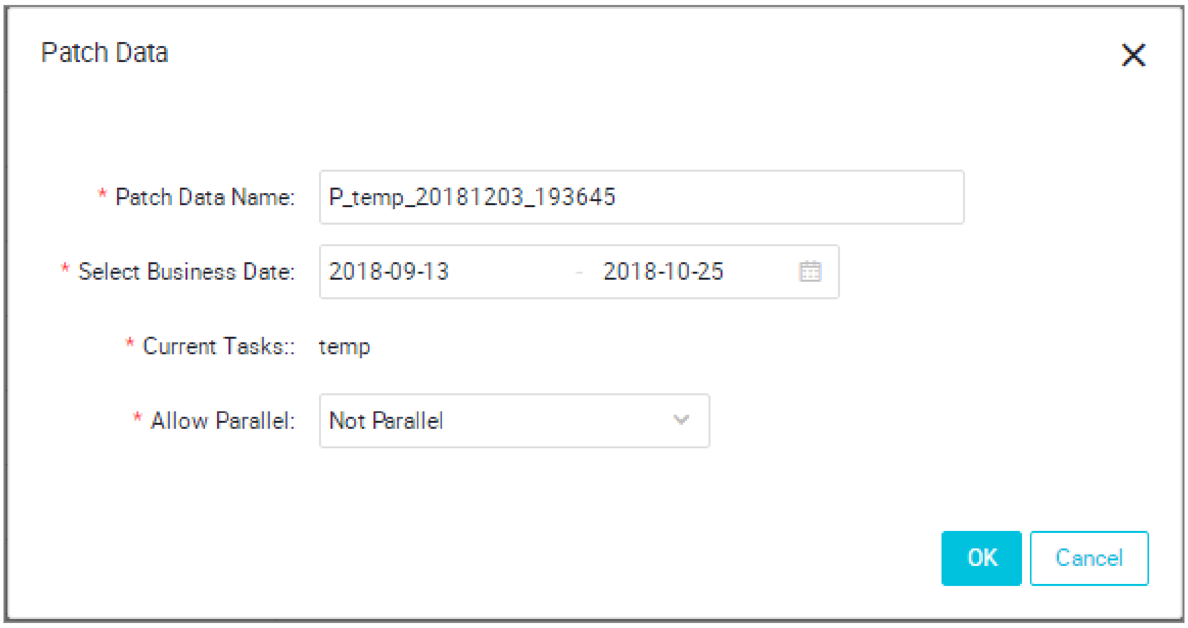
* Data Source:	MySQL, [redacted]
* Table:	`ods_user_info_d`
Data Filtering:	${bizdate}
Sharding Key:	uid
* Data Source:	ODPS, [redacted]
* Table:	ods_user_info_d
* Partition:	dt = ${bizdate}
Clearance Rule:	Clear Existing Data Before Writing (Insert Overwrit...
Compression:	[✓ Disable] Enable
Consider Empty String as Null:	Yes [✓ No]

2. Perform data patching. Choose **Save** > **Submit**. After the data is submitted, choose **O&M** > **Task List** > **Cycle Task**, select the rds_sync node, and choose **Patch Data** > **Current node**, as shown in the following figure.



Name:	Node ID	Modified At	Task Type	Owner:	Schedule Type:	Actions
targett	700001909393	2018-12-03 14:23:10	ODPS_SQL	wangdan	Day Schedule	DAG | Test | Patch Data | More
mysql_to_odps	700001909391	2018-12-03 14:22:51	Data Integration	wangdan	Day Schedule	DAG | Test | Patch Data | More
temp	700001909390	2018-12-03 14:22:28	ODPS_SQL	wangdan	Day Schedule	DAG | Test | Patch Data | More
rds_sync	700000461346	2018-12-02 03:00:33	Data Integration	wangdan	Day Schedule	More
ods_log_info_d	700000461553	2018-11-26 12:52:50	ODPS_SQL	wangdan	Day Schedule	More
dw_user_info_all_d	700000461554	2018-10-31 10:52:05	ODPS_SQL	wangdan	Day Schedule	DAG | Test | Patch Data | More
create_table_ddl	700000461344	2018-10-23 15:55:11	ODPS_SQL	wangdan	Day Schedule	DAG | Test | Patch Data | More
rpt_user_info_d	700000461555	2018-09-02 10:40:00	ODPS_SQL	wangdan	Day Schedule	DAG | Test | Patch Data | More
ftp_sync	700000461345	2018-09-02 10:39:47	Data Integration	wangdan	Day Schedule	DAG | Test | Patch Data | More
workshop_start	700000461343	2018-08-30 10:31:58	Virtual Node	wangdan	Day Schedule	DAG | Test | Patch Data | More

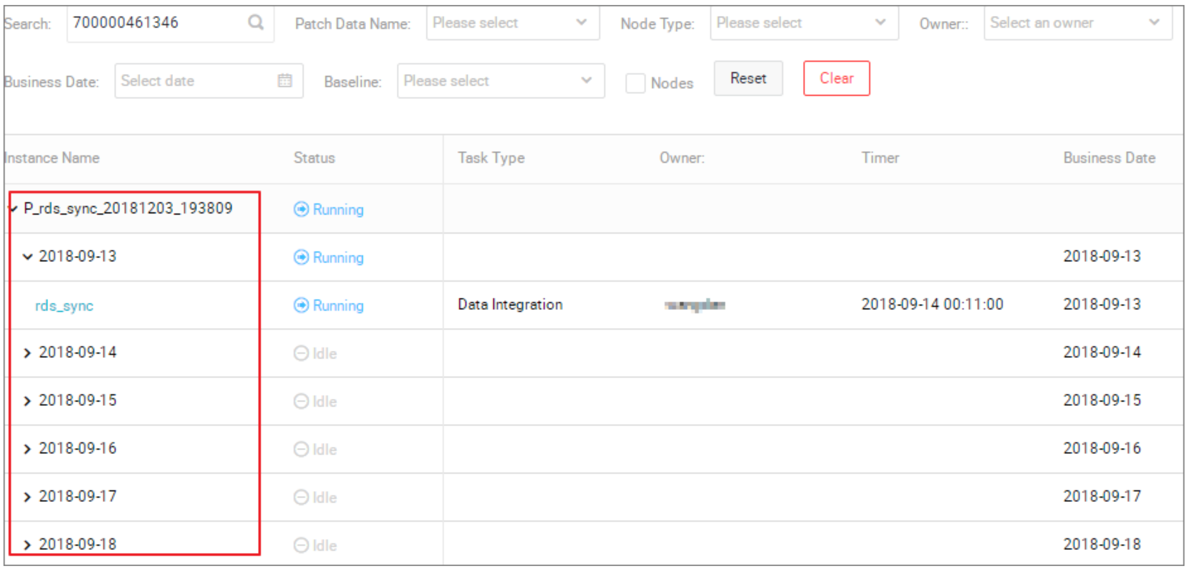
3. On the displayed page, select the business date, as shown in the following figure.



Patch Data

* Patch Data Name:	P_temp_20181203_193645
* Select Business Date:	2018-09-13 - 2018-10-25
* Current Tasks::	temp
* Allow Parallel:	Not Parallel

4. Click **OK**. Multiple synchronization task instances are generated at the same time and executed in sequence, as shown in the following figure.

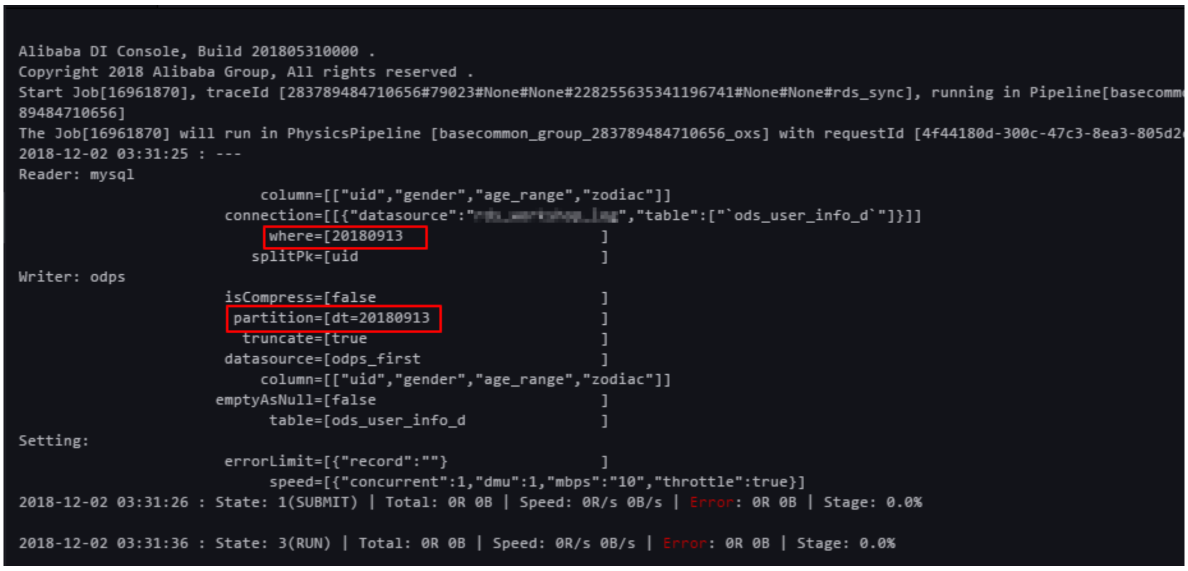


Search: 700000461346　Patch Data Name: Please select　Node Type: Please select　Owner:: Select an owner

Business Date: Select date　Baseline: Please select　Nodes

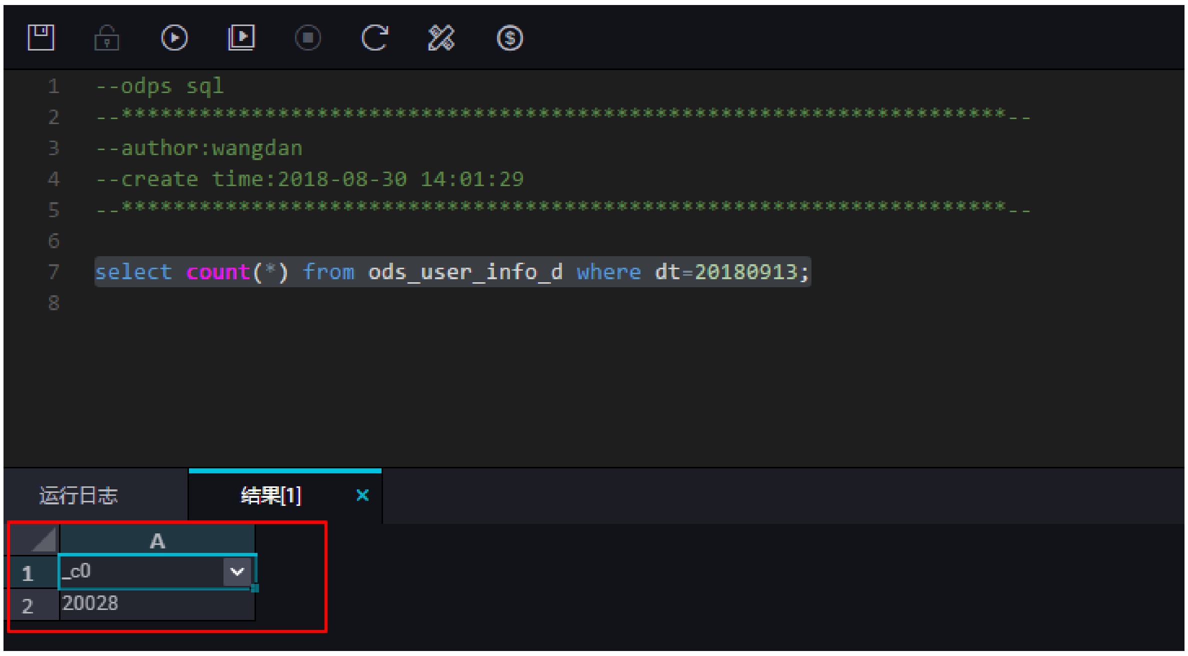
Instance Name	Status	Task Type	Owner:	Timer	Business Date
P_rds_sync_20181203_193809	Running				
2018-09-13	Running				2018-09-13
rds_sync	Running	Data Integration	[redacted]	2018-09-14 00:11:00	2018-09-13
2018-09-14	Idle				2018-09-14
2018-09-15	Idle				2018-09-15
2018-09-16	Idle				2018-09-16
2018-09-17	Idle				2018-09-17
2018-09-18	Idle				2018-09-18

5. View the running log. You can see the process of extracting data from RDS. A partition has been automatically created in MaxCompute, as shown in the following figure.



```
Alibaba DI Console, Build 201805310000 .
Copyright 2018 Alibaba Group, All rights reserved .
Start Job[16961870], traceId [283789484710656#79023#None#None#228255635341196741#None#None#rds_sync], running in Pipeline[basecomm
89484710656]
The Job[16961870] will run in PhysicsPipeline [basecommon_group_283789484710656_oxs] with requestId [4f44180d-300c-47c3-8ea3-805d2
2018-12-02 03:31:25 : ---
Reader: mysql
                           column=[["uid","gender","age_range","zodiac"]]
                       connection=[[{"datasource":"[redacted]","table":["`ods_user_info_d`"]}]]
                            where=[20180913                             ]
                          splitPk=[uid                                  ]
Writer: odps
                       isCompress=[false                                ]
                        partition=[dt=20180913                          ]
                         truncate=[true                                 ]
                       datasource=[odps_first                           ]
                           column=[["uid","gender","age_range","zodiac"]]
                      emptyAsNull=[false                                ]
                            table=[ods_user_info_d                      ]
Setting:
                       errorLimit=[{"record":""}                        ]
                            speed=[{"concurrent":1,"dmu":1,"mbps":"10","throttle":true}]
2018-12-02 03:31:26 : State: 1(SUBMIT) | Total: 0R 0B | Speed: 0R/s 0B/s | Error: 0R 0B | Stage: 0.0%

2018-12-02 03:31:36 : State: 3(RUN) | Total: 0R 0B | Speed: 0R/s 0B/s | Error: 0R 0B | Stage: 0.0%
```

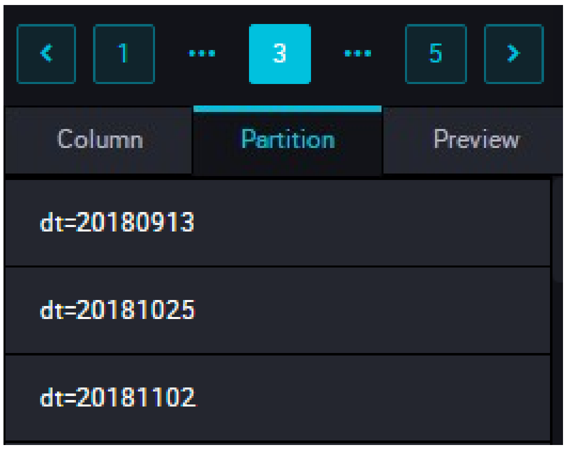
View the results. You can check whether the data is written successfully, whether a partition is created, and whether the data is synchronized to the partition table, as shown in the following figure.



```
--odps sql
--**********************************************************************--
--author:wangdan
--create time:2018-08-30 14:01:29
--**********************************************************************--

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

	A
1	_c0
2	20028

Query the partition information, as shown in the following figure.



dt=20180913

dt=20181025

dt=20181102

Note:

In MaxCompute 2.0, parameter settings are required for partition table query. The partition column needs to be updated to the business date. If the task execution date is 20180717, the business date is 20180716. The SQL statements are as follows:

```
--Check whether the data is successfully written to MaxCompute.
select count(*)from ods_user_info_d where dt=business date;
```

Create a partition by non-date field using hash

If a huge volume of data needs to be processed, or if a full amount of data is partitioned according to a non-date field (such as province) at the first time, a data partition cannot be created automatically during data integration. Therefore, you can use the hash algorithm to save the same values in an RDS field to the corresponding partition in MaxCompute.

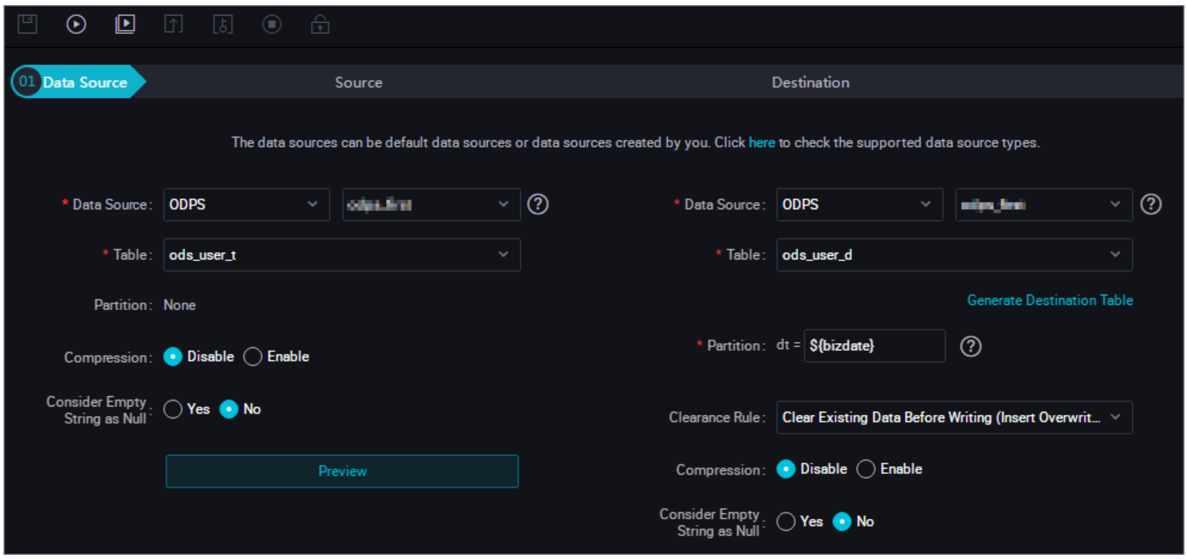
The procedure is as follows:

1. Synchronize the full amount of data to a temporary table in MaxCompute. Create an SQL script node and choose **Run** > **Save** > **Submit**.

   The SQL statements are as follows:

```
drop table if exists ods_user_t;
CREATE TABLE ods_user_t (
        dt STRING,
 uid STRING,
        gender STRING,
        age_range STRING,
        zodiac STRING);
insert overwrite table ods_user_t select dt,uid,gender,age_range,
zodiac from ods_user_info_d;--Save the data in the ODPS table to the
 temporary table.
```

2. Create a synchronization task node named mysql_to_odps to synchronize the full amount of RDS data to MaxCompute without setting the partition, as shown in the following table.



The data sources can be default data sources or data sources created by you. Click here to check the supported data source types.

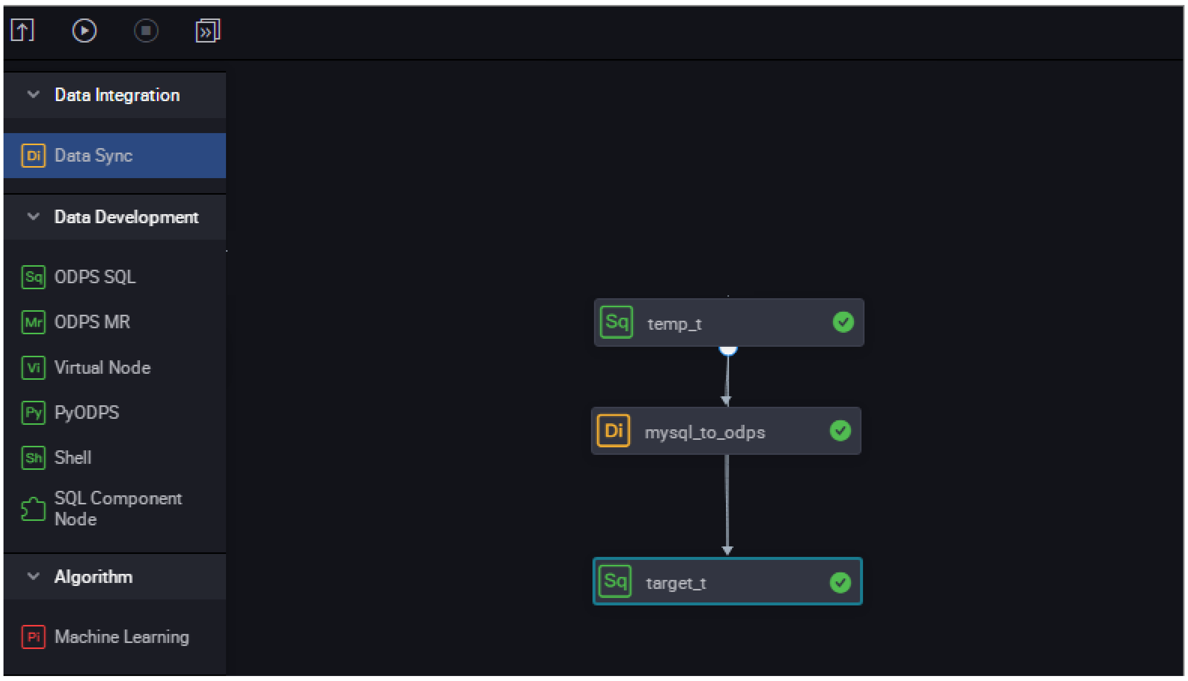
* Data Source:	ODPS [redacted]
* Table:	ods_user_t
Partition:	None
Compression:	[✓ Disable] Enable
Consider Empty String as Null:	Yes [✓ No]

* Data Source:	ODPS [redacted]
* Table:	ods_user_d
* Partition:	dt = ${bizdate}
Clearance Rule:	Clear Existing Data Before Writing (Insert Overwrit...
Compression:	[✓ Disable] Enable
Consider Empty String as Null:	Yes [✓ No]

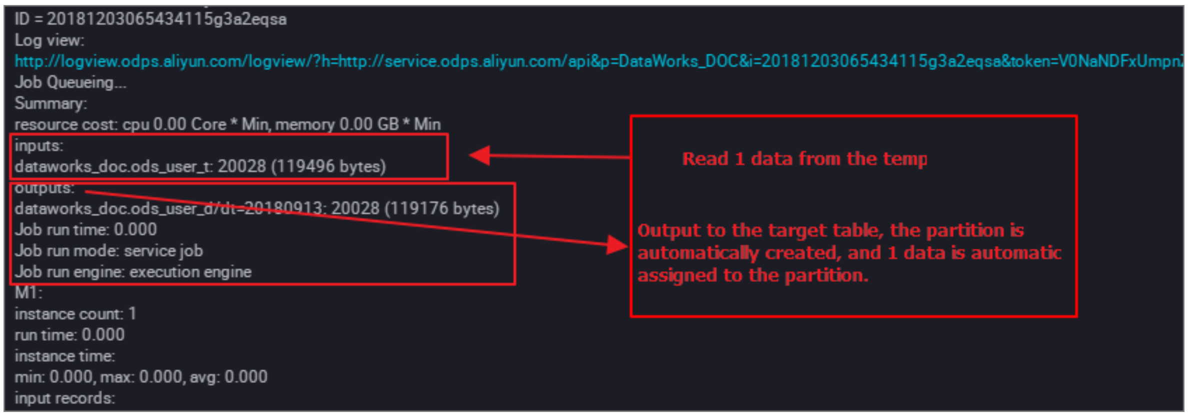
3. Use SQL statements to dynamically create a partition for the destination table. The SQL statements are as follows:

```
drop table if exists ods_user_d;
--Create a partition table (the destination table) in ODPS.
 CREATE TABLE ods_user_d (
 uid STRING,
        gender STRING,
        age_range STRING,
        zodiac STRING
)
PARTITIONED BY (
 dt STRING
);
--Execute the dynamic partitioning SQL statements to automatically
create a partition according to the dt field in the temporary table
. The partition value is automatically created for a data record.
Data records that share the same dt value have the same partition
field value.
--For example, some data records share the value 20180913 in the
dt field. As a result, a partition is automatically created in the
MaxCompute partition table with a partition value of 20181025.
--The dynamic partitioning SQL statements are as follows:
--A date_time field is added in the select field, indicating that a
partition is automatically created according to this field.
insert overwrite table ods_user_d partition(dt)select dt,uid,gender,
age_range,zodiac from ods_user_t;
--After the import is completed, you can delete the temporary table
to reduce excessive storage costs.
drop table if exists ods_user_d;
```

4. Configure the three nodes to form a workflow and execute these nodes in sequence, 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.

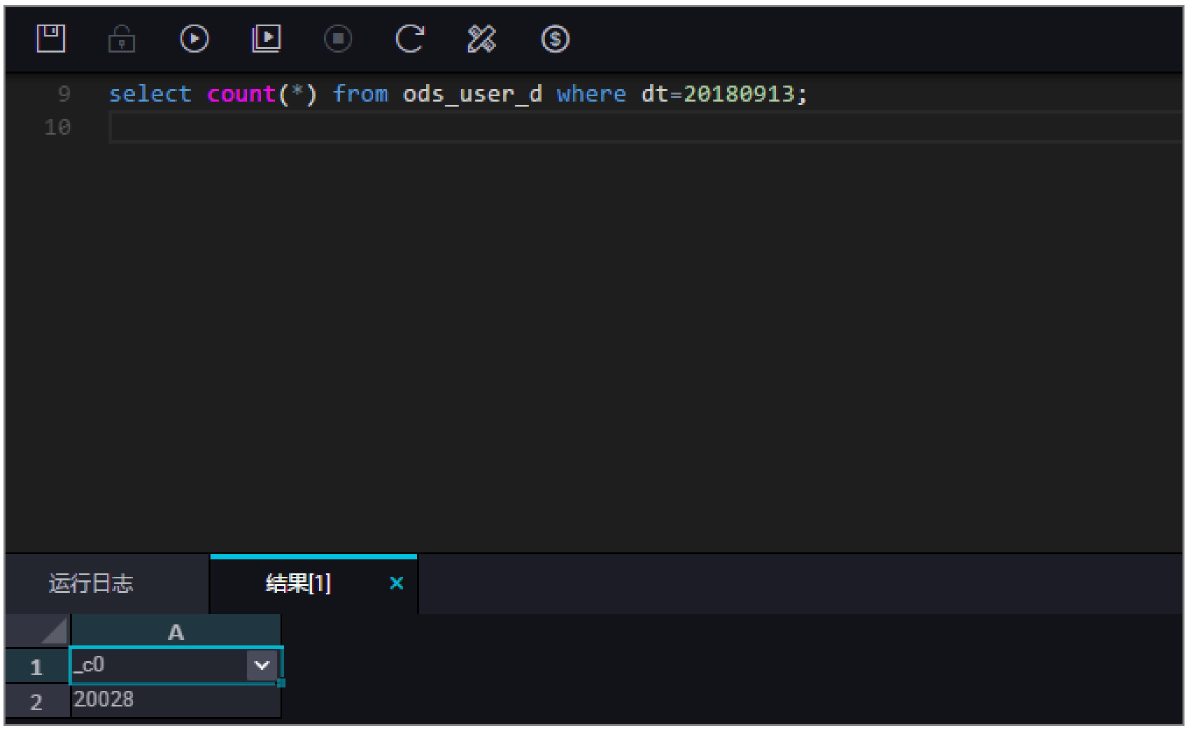


```
ID = 20181203065434115g3a2eqsa
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.odps.aliyun.com/api&p=DataWorks_DOC&i=20181203065434115g3a2eqsa&token=V0NaNDFxUmpn
Job Queueing...
Summary:
resource cost: cpu 0.00 Core * Min, memory 0.00 GB * Min
inputs:
dataworks_doc.ods_user_t: 20028 (119496 bytes)
outputs:
dataworks_doc.ods_user_d/dt=20180913: 20028 (119176 bytes)
Job run time: 0.000
Job run mode: service job
Job run engine: execution engine
M1:
instance count: 1
run time: 0.000
instance time:
min: 0.000, max: 0.000, avg: 0.000
input records:
```

Read 1 data from the temp

Output to the target table, the partition is automatically created, and 1 data is automatic assigned to the partition.

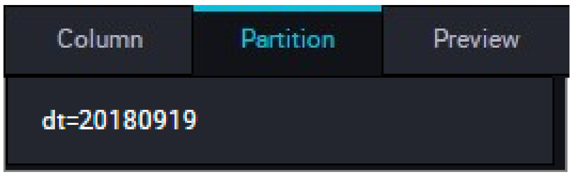
View data. Dynamic partitioning is completed automatically. Data records with the same date are synchronized to the same partition, as shown in the following figure.



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

	A
1	_c0
2	20028

Query the partition information, as shown in the following figure.



dt=20180919

You can follow the preceding steps to name a partition using the province field.

The data synchronization feature of DataWorks supports automatic data operations, including data synchronization, data migration, and data synchronization task scheduling. For more information about scheduling configuration, see *Time attributes* in Scheduling Configuration.

10 Migrate JSON data from OSS to MaxCompute

This topic describes how to use the data integration feature of DataWorks to migrate JSON data from OSS to MaxCompute and use the built-in string function GET_JSON_OBJECT of MaxCompute to extract JSON information.

Upload data to OSS

Convert your JSON file to a TXT file and upload it to OSS. The following is a JSON file example:

```
{
    "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
}
```

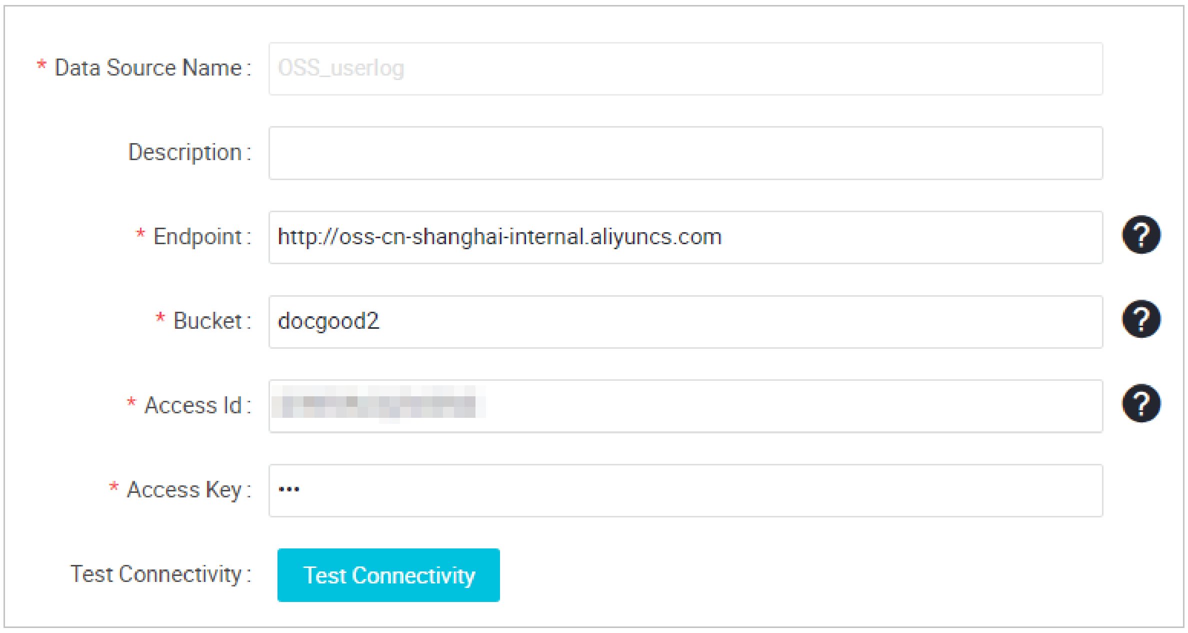
Upload the applog.txt file to OSS. In this example, the OSS bucket is located in China East 2 (Shanghai).

Use DataWorks to import data to MaxCompute

1. Add an OSS data source.

   In the DataWorks console, go to the *Data Integration* page and add an OSS data source. For more information, see *Configure OSS data source*.

Specific parameters are shown in the following figure. Click Finish after the data source connectivity test is successful. The endpoints in this topic include **`http://oss-cn-shanghai.aliyuncs.com`** and **`http://oss-cn-shanghai-internal.aliyuncs.com`**. The OSS and DataWorks projects are located in the same region. Therefore, `http://oss-cn-shanghai-internal.aliyuncs.com` is used. This endpoint is connected through the intranet.



* Data Source Name :	OSS_userlog
Description :	
* Endpoint :	http://oss-cn-shanghai-internal.aliyuncs.com
* Bucket :	docgood2
* Access Id :	[redacted]
* Access Key :	•••
Test Connectivity :	Test Connectivity

2. Create a data synchronization task.

   In the DataWorks console, create a **data synchronization** node. For more information, see *Configure OSS Reader*.At the same time, create a table named mqdata in DataWorks to store the JSON data. For more information, see *Create a table*.You can set the table parameters on the graphical interface. The mqdata table has only one column, which is named MQ data. The data type is string.

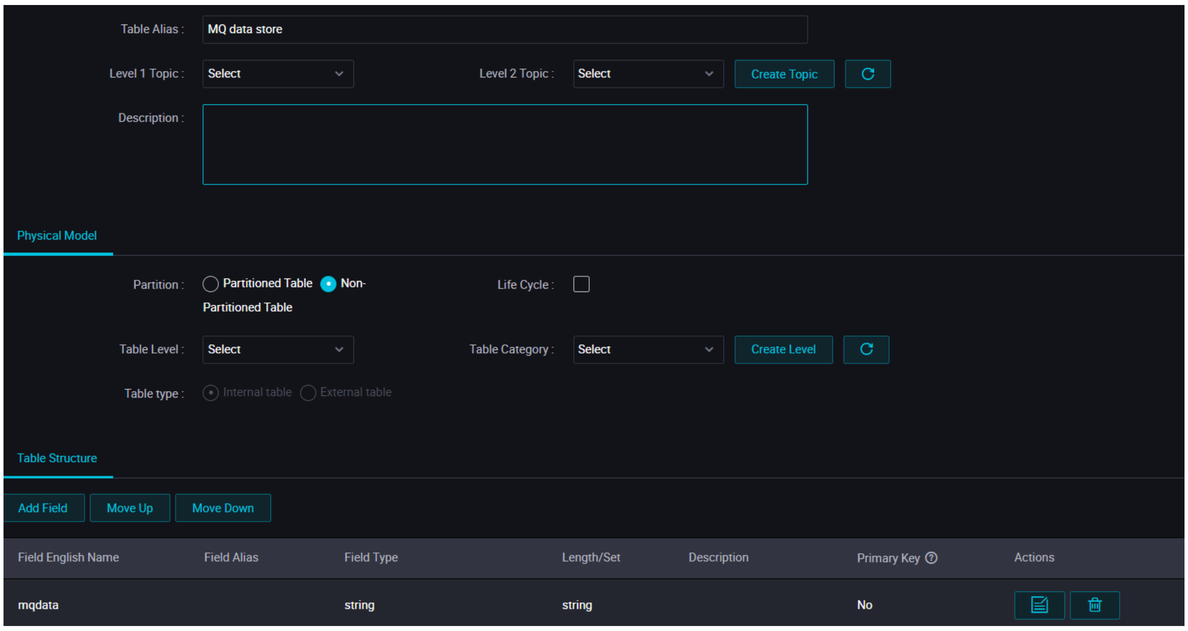


Table Alias :	MQ data store
Level 1 Topic :	Select
Level 2 Topic :	Select
Description :	

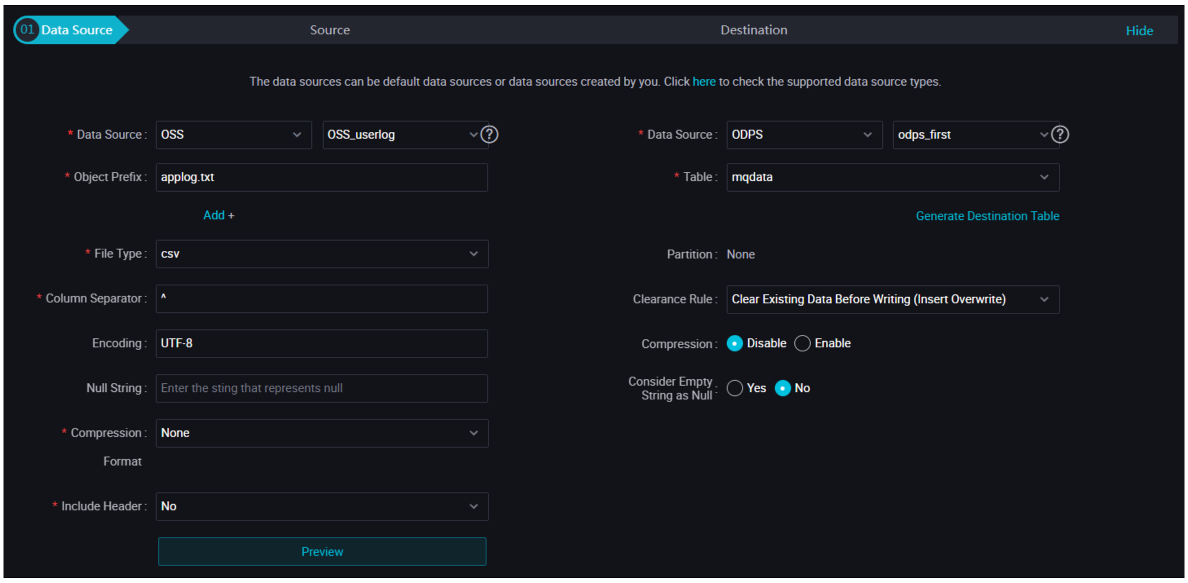
Physical Model

Partition :	Partitioned Table [✓ Non-Partitioned Table]
Life Cycle :	
Table Level :	Select
Table Category :	Select
Table type :	Internal table External table

Table Structure

Field English Name	Field Alias	Field Type	Length/Set	Description	Primary Key	Actions
mqdata		string	string		No	

After creating a table, you can set the data synchronization task parameters on the graphical interface, as shown in the following figure. Set the destination data source to odps_first and the destination table to mqdata. Set the original data source to OSS, and enter the file path and name as the object prefix. Set the column delimiter to caret (^) or any other characters that are not contained in the TXT file. (DataWorks supports multiple column delimiters for the TXT data sources in OSS. Therefore, you can use characters such as %&%#^$$^% to divide the data into a column.)



The data sources can be default data sources or data sources created by you. Click here to check the supported data source types.

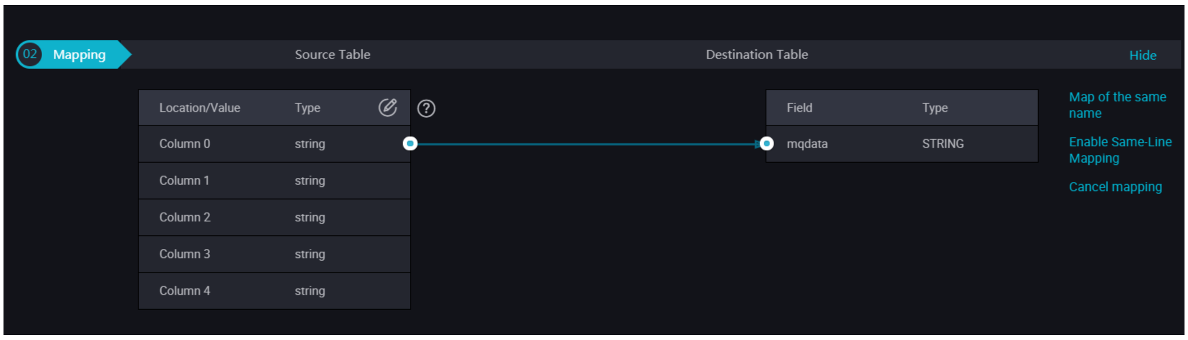
Source:

* Data Source :	OSS / OSS_userlog
* Object Prefix :	applog.txt
* File Type :	csv
* Column Separator :	^
Encoding :	UTF-8
Null String :	Enter the sting that represents null
* Compression Format :	None
* Include Header :	No

Destination:

* Data Source :	ODPS / odps_first
* Table :	mqdata
Partition :	None
Clearance Rule :	Clear Existing Data Before Writing (Insert Overwrite)
Compression :	[✓ Disable] Enable
Consider Empty String as Null :	Yes [✓ No]

Set the mapping method to **same-line mapping**.



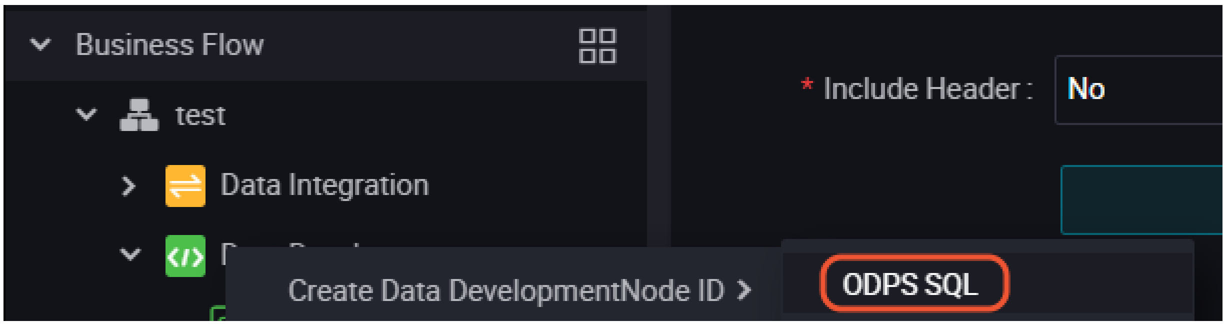
Click the script switching button in the upper-left corner to switch to the script mode. Set fileFormat to `"fileFormat":"binary"`. Then, after the JSON file is synchronized from OSS to MaxCompute, data in the JSON file is saved in the same row. That is, data in the JSON file shares the same field. Use the default values for other parameters. The following is an example of the code in script mode:

```
{
    "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,
                "dmu": 1
            }
        }
    }
```

After completing the preceding configuration, click **run**.

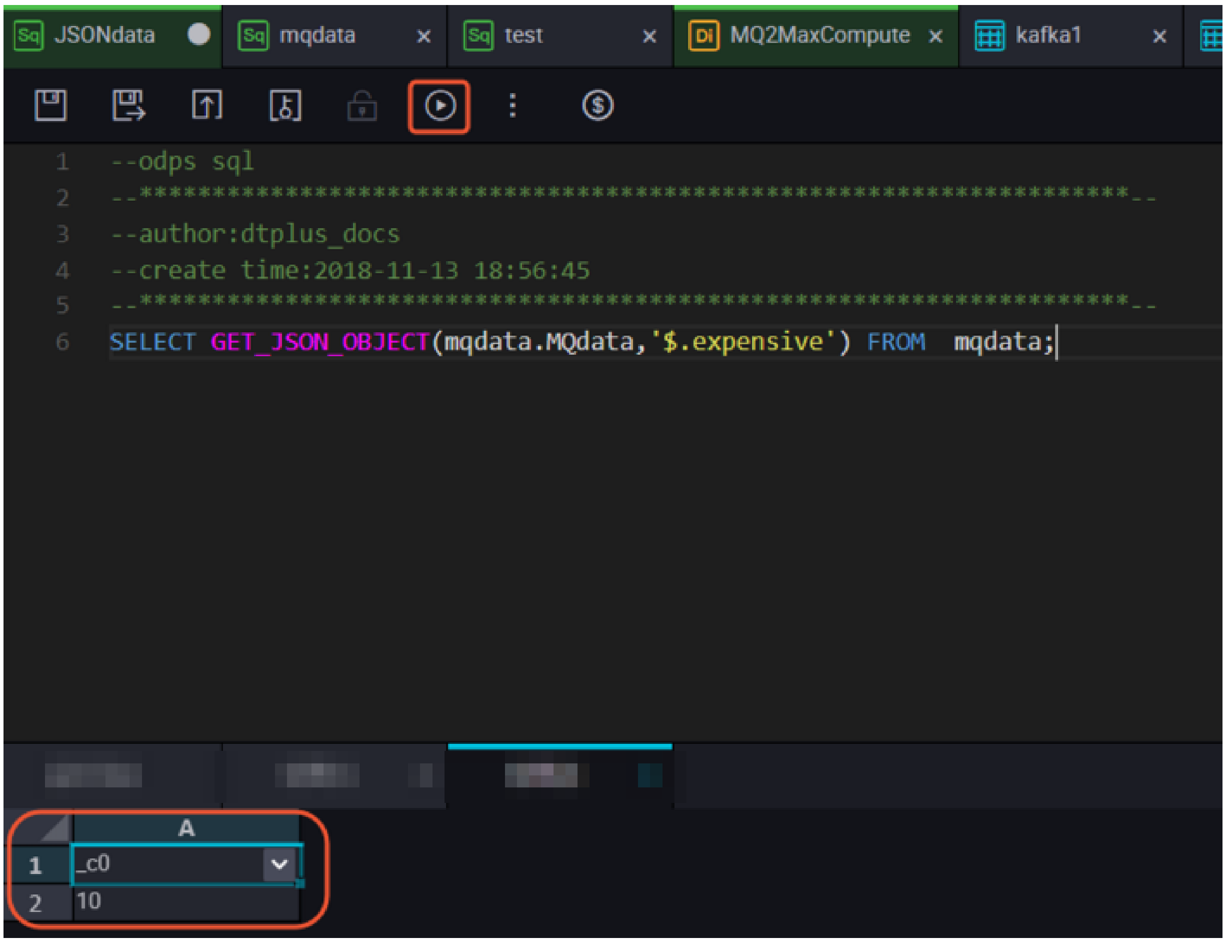
Obtain the JSON field information

Create an ODPS SQL node in your *Business Flow*.



Enter the `SELECT*from mqdata;` statement to view the data in the mqdata table. You can also run the command on the *MaxCompute client* to view the data and perform subsequent steps.

Verify that the data imported to the table is correct. You can then use the built-in string function *GET_JSON_OBJECT* in MaxCompute to obtain the JSON data as needed. Use `SELECT GET_JSON_OBJECT(mqdata.MQdata,'$.expensive') FROM mqdata;` to obtain the value of **`expensive`** in the JSON file. Then, the data is obtained successfully, as shown in the following figure.



```
--odps sql
--**********************************************************************--
--author:dtplus_docs
--create time:2018-11-13 18:56:45
--**********************************************************************--
SELECT GET_JSON_OBJECT(mqdata.MQdata,'$.expensive') FROM  mqdata;
```

	A
1	_c0
2	10

11 Migrate JSON data from MongoDB to MaxCompute

This topic describes how to use the data integration feature of DataWorks to extract JSON fields from MongoDB to MaxCompute.

Prepare data and an account

Upload data to your MongoDB. In this example, Alibaba Cloud ApsaraDB for MongoDB is used. The network type is VPC. (An Internet IP address is required for MongoDB to communicate with the default resource group of DataWorks.) The test data is as follows:

```
{
    "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
}
```

Log on to the MongoDB database. In this example, the database name is **`admin`** and the collection is **`userlog`**. You can run the **`db.userlog.find().limit(10)`** command in the query window to view the uploaded data, as shown in the following figure.

Furthermore, you need to create a user in the database in advance to add data sources in DataWorks. In this example, you can run the `db.createUser({user:"bookuser",pwd:"`

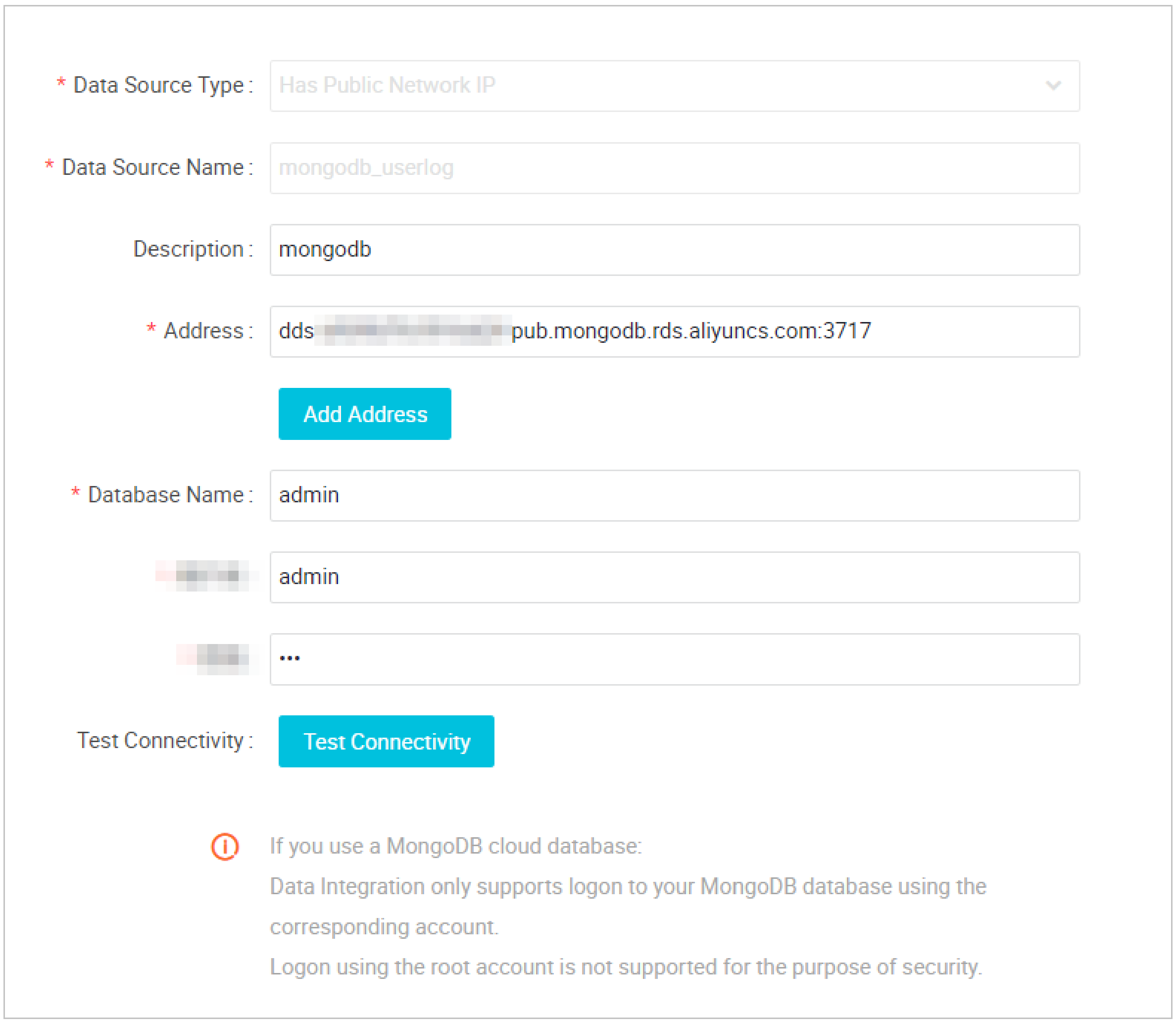
`123456",roles:["root"]})` command to create a user named **`bookuser`**. The password of the user is **`123456`**, and the permission is **`root`**.

Use DataWorks to extract data to MaxCompute

1. Add a MongoDB data source.

   In the DataWorks console, go to the *Data Integration* page and add a *MongoDB* data source.

   The specific parameters are shown in the following figure. Click **Finish** after the data source connectivity test is successful. In this example, the network type of MongoDB is VPC. Therefore, the **Data Source Type** must be set to **Has Public Network IP**.



* Data Source Type :	Has Public Network IP
* Data Source Name :	mongodb_userlog
Description :	mongodb
* Address :	dds[redacted]pub.mongodb.rds.aliyuncs.com:3717
	Add Address
* Database Name :	admin
[redacted]	admin
[redacted]	···
Test Connectivity :	Test Connectivity

If you use a MongoDB cloud database:
Data Integration only supports logon to your MongoDB database using the corresponding account.
Logon using the root account is not supported for the purpose of security.

   To obtain the IP address and the port number, log on to the ApsaraDB for MongDB console and click an instance.

2. Create a data synchronization task.

   In the DataWorks console, create a data synchronization node. For more information, see *Configure OSS Reader*.At the same time, create a table named mqdata in DataWorks to store

JSON data. For more information, see *Create a table*.You can set the table parameters on the graphical interface. The **mqdata** table has only one column, which is named MQ data. The data type is string.

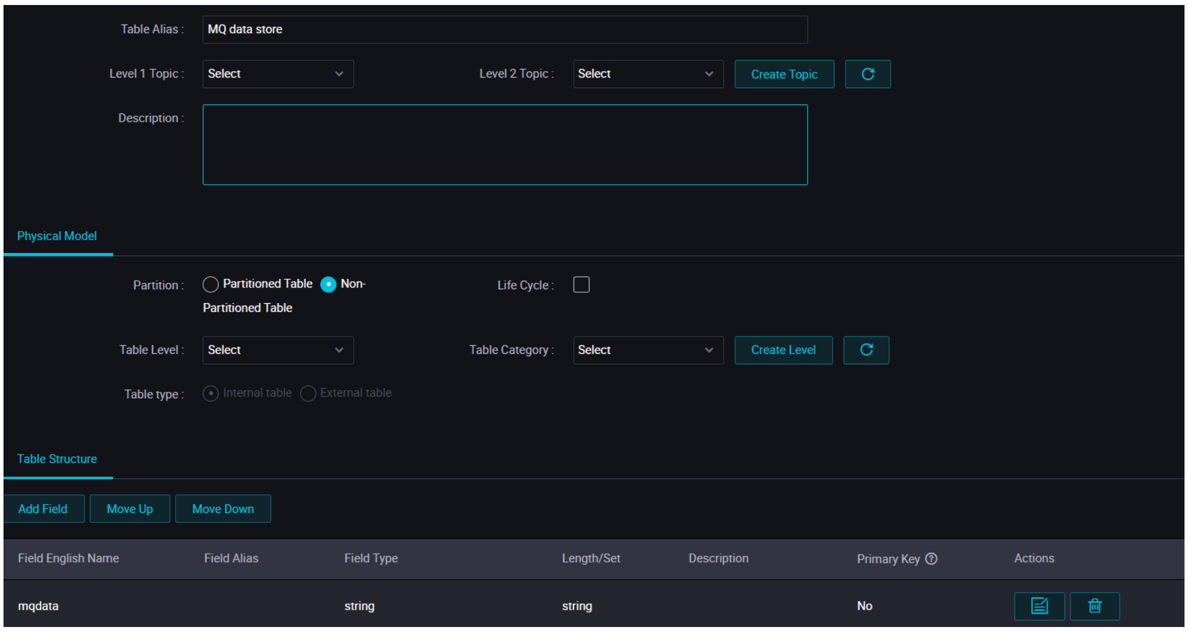


Table Alias : MQ data store

Level 1 Topic : Select — Level 2 Topic : Select — Create Topic

Description :

Physical Model

Partition : Partitioned Table [✓ Non-Partitioned Table] — Life Cycle : ☐

Table Level : Select — Table Category : Select — Create Level

Table type : Internal table External table

Table Structure

Add Field — Move Up — Move Down

Field English Name	Field Alias	Field Type	Length/Set	Description	Primary Key	Actions
mqdata		string	string		No	

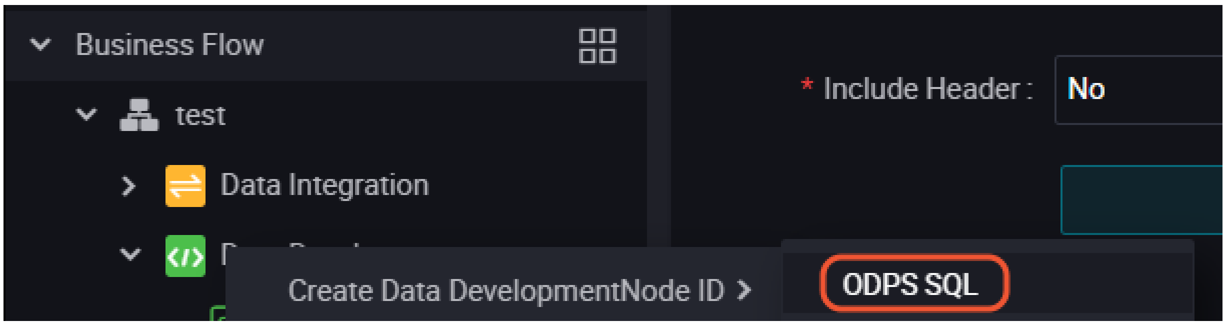
After creating a table, you can set the data synchronization task parameters on the graphical interface. Set the destination data source to odps_first and the destination table to mqdata. Set the original data source to MongoDB, and select mongodb_userlog. After completing the preceding configuration, click **Switch to Script Mode**.The following is an example of the code in script mode:

```
{
    "type": "job",
    "steps": [
        {
            "stepType": "mongodb",
            "parameter": {
                "datasource": "mongodb_userlog",
 //Data source name
                "column": [
                    {
                        "name": "store.bicycle.color", //JSON field
path. In this example, the value of color is extracted.
                        "type": "document.document.string" //The
 number of fields in this line must be the same as that in the
preceding line (the name line). If the JSON field is a level-1 field
, for example, the expensive field in this topic, enter the string.
                    }
                ],
                "collectionName //Collection name": "userlog"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
                "stepType": "odps",
                "parameter": {
                    "partition": "",
                    "isCompress": false,
                    "truncate": true,
                    "datasource": "odps_first",
                    "column": [
                              "mqdata" //Table column name in MaxCompute
                    ],
                    "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,
                "dmu": 1
            }
        }
}
```

After completing the preceding configuration, click **Run**.

Verify the result

Create an ODPS SQL node in your *Business Flow*.



Enter the `SELECT * from mqdata;` statement to view the data in the mqdata table. You can also run the **`SELECT * from mqdata;`** command on the *MaxCompute client* to view the data.