ALIBABA CLOUD

Alibaba Cloud

MaxCompute

SDK Reference

Document Version: 20201109

Alibaba Cloud

Legal disclaimer

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

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

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

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

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

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

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

Document conventions

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

Table of Contents

1.Java SDK ---- 05
1.1. Java SDK ---- 05
1.2. Java SDK examples ---- 12
1.2.1. Run security-related commands ---- 13
1.2.2. Instance Logview ---- 15
1.2.3. Return error logs ---- 16
1.2.4. Set the SQL flag ---- 17
1.2.5. Use SQLTask and Tunnel to export a large amount of ... ---- 18
2.Python SDK ---- 22
2.1. Overview ---- 22
2.2. SDK for Python ---- 24

1.Java SDK

1.1. Java SDK

This article introduces most commonly used MaxCompute core interfaces. For more information, see SDK Java Doc.

You can configure the version of the new SDK through maven management. The configuration information of Maven is as follows,The latest version can be searched for odps-sdk-core at any time at search.maven.org.

```
<dependency>
 <groupId>com.aliyun.odps</groupId>
 <artifactId>odps-sdk-core</artifactId>
 <version>0.26.2-public</version>
</dependency>
```

> **Note** 0.27.2-public version and above support MaxCompute 2.0 New data type.

The overall information of the SDK package provided by MaxCompute is shown in the following table:

Package Name	Description
odps-sdk-core	The basic functions of MaxCompute, such as the operation of tables, Project, and Tunnel, are all included in this package.
odps-sdk-commons	Some Util packages.
odps-sdk-udf	Main interface of UDF.
odps-sdk-mapred	MapReduce Java SDK.
odps-sdk-graph	Graph Java SDK, the keyword. used to search is “odps-sdk-graph” .

AliyunAccount

AlibabaCloudAccount. The primary account created with Alibaba Cloud. It generally has an AccessKey that comprises of an AccessKeyId and an AccessKeySecret, used to initialize MaxCompute.

MaxCompute

It is the entry of MaxCompute SDK. You can get set of all objects under the project shell by such endpoint, including Projects, Tables, Resources, Functions, and Instances.

> **Note** MaxCompute was formerly called ODPS, so the portal class is still named as ODPS in the current SDK version.

User can construct MaxCompute object by entering the AliyunAccount instance. The code example is shown as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Table t : odps.tables()) {
  ....
}
```

Tunnel

The MaxCompute Tunnel data channel is written based on the Tunnel SDK, and you can upload or download data to MaxCompute through the Tunnel. For more information, seeTunnel SDK.At present, Tunnel supports only tables (excluding views View) and uploading and downloading data.

MapReduce

SeeMapReduce SDK for more information.

Projects

It is the set of all projects in MaxCompute. The element of this set is Projects. The code example is shown as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Project p = odps.projects().get("my_exists");
p.reload();
Map<String, String> properties = prj.getProperties();
...
```

Project

It refers to the description of project and corresponding project, and can be acquired from Projects.

SQLTask

It refers to an interface to run and process SQL task. SQL can run directly through the interface 'run'.(**Note: only one SQL statement can be submitted at a time.**)

The run interface returns the Instance instance and obtains the SQL running status and result through Instance.

Example:

```
import java.util.List;
import com.aliyun.odps.Instance;
mport com.aliyun.odps.Odps;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.task.SQLTask;
public class testSql {
private static final String accessId = "";
private static final String accessKey = "";
private static final String endPoint = "http://service.odps.aliyun.com/api";
private static final String project = "";
private static final String sql = "select category from iris;";
public static void
main(String[] args) {
 Account account = new AliyunAccount(accessId, accessKey);
  Odps odps = new Odps(account);
  odps.setEndpoint(endPoint);
  odps.setDefaultProject(project);
  Instance i;
 try {
   i = SQLTask.run(odps, sql);
   i.waitForSuccess();
   List<Record> records = SQLTask.getResult(i);
   for(Record r:records){
     System.out.println(r.get(0).toString());
   }
 } catch (OdpsException e) {
   e.printStackTrace();
 }
}
}
```

> **Note** To create a table, use SQLTask interface instead of the interface Table. You must introduce the statement of Table Operation into SQLTask.

Instances

This class refers to the set of all (instances) in MaxCompute and the element of this set is Instance. The code example is shown as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Instance i : odps.instances()) {
  ....
}
```

Instance

It refers to the description of instance and corresponding instance,and can be acquired from Instances. The code example is as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Instance instance= odps.instances().get("instance id");
Date startTime = instance.getStartTime();
Date endTime = instance.getEndTime();
...
Status instanceStatus = instance.getStatus();
String instanceStatusStr = null;
if (instanceStatus == Status.TERMINATED) {
 instanceStatusStr = TaskStatus.Status.SUCCESS.toString();
 Map<String, TaskStatus> taskStatus = instance.getTaskStatus();
 for (Entry<String, TaskStatus> status : taskStatus.entrySet()) {
  if (status.getValue().getStatus() ! = TaskStatus.Status.SUCCESS) {
   instanceStatusStr = status.getValue().getStatus().toString();
   break;
  }
 }
} else {
 instanceStatusStr = instanceStatus.toString();
}
...
TaskSummary summary = instance.getTaskSummary("task name");
String s = summary.getSummaryText();
```

Tables

This class refers to the set of all tables in MaxCompute. The element of this set is Table. The code example is shown as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Table t : odps.tables()) {
  ....
}
```

Table

It refers to the description of table and corresponding table and can be acquired through Tables. The code example is shown as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Table t = odps.tables().get("table name");
t.reload();
Partition part = t.getPartition(new PartitionSpec(tableSpec[1]));
part.reload();
...
```

Resources

The class refers to the set of all resources in MaxCompute. The element of this set is Resource. The code example is as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Resource r : odps.resources()) {
  ....
}
```

Resource

It refers to the resource description and the corresponding resource and can be acquired through Resources. The code example is as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Resource r = odps.resources().get("resource name");
r.reload();
if (r.getType() == Resource.Type.TABLE) {
 TableResource tr = new TableResource(r);
 String tableSource = tr.getSourceTable().getProject() + "."
   + tr.getSourceTable().getName();
 if (tr.getSourceTablePartition() ! = null) {
  tableSource += " partition(" + tr.getSourceTablePartition().toString()
    + ")";
 }
 ....
}
```

File resource creation example is as follows:

```
String projectName = "my_porject";
String source = "my_local_file.txt";
File file = new File(source);
InputStream is = new FileInputStream(file);
FileResource resource = new FileResource();
String name = file.getName();
resource.setName(name);
odps.resources().create(projectName, resource, is);
```

Table resource creation example is as follows:

```
TableResource resource = new TableResource(tableName, tablePrj, partitionSpec);
//resource.setName(INVALID_USER_TABLE);
resource.setName("table_resource_name");
odps.resources().update(projectName, resource);
```

Functions

This class refers to the set of all functions in MaxCompute. The element of this set is Function. An example is as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Function f : odps.functions()) {
  ....
}
```

Function

It refers to the function description and corresponding function and can be acquired through Functions. The code example is as follows:

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Function f = odps.functions().get("function name");
List<Resource> resources = f.getResources();
```

Function creation example:

```
String resources = "xxx:xxx";
String classType = "com.aliyun.odps.mapred.open.example.WordCount";
ArrayList<String> resourceList = new ArrayList<String>();
for (String r : resources.split(":")) {
 resourceList.add(r);
}
Function func = new Function();
func.setName(name);
func.setClassType(classType);
func.setResources(resourceList);
odps.functions().create(projectName, func);
```

1.2. Java SDK examples

1.2.1. Run security-related commands

This topic describes how to use the SecurityManager.runQuery() method of MaxCompute SDK for Java to run security-related commands on the MaxCompute client.

run security-related commands SecurityManager.runQuery()

Prerequisites

The following operations are completed:

- Install the IntelliJ IDEA development tool. For more information, see Install IntelliJ IDEA.
- Create a MaxCompute Studio project and connect it to your MaxCompute project. For more information, see Manage project connections.
- Add project dependencies in MaxCompute Studio.

  The SecurityManager class is contained in the odps-sdk-core package. Therefore, you must add project dependencies by using the following configuration:

```
<dependency>
 <groupId>com.aliyun.odps</groupId>
 <artifactId>odps-sdk-core</artifactId>
 <version>0.29.11-oversea-public</version>
</dependency>
```

Context

You can run security-related commands by using one of the following methods:

- Use the MaxCompute client. For more information, see the instructions in Target users and Security configuration of a project.

  The commands that start with any of the following keywords are the security-related commands of MaxCompute:

```
GRANT/REVOKE ...
SHOW GRANTS/ACL/PACKAGE/LABEL/ROLE/PRINCIPALS
SHOW PRIV/PRIVILEGES
LIST/ADD/REOVE USERS/ROLES/TRUSTEDPROJECTS
DROP/CREATE ROLE
CLEAR EXPIRED GRANTS
DESC/DESCRIBE ROLE/PACKAGE
CREATE/DELETE/DROP PACKAGE
ADD ... TO PACKAGE
REMOVE ... FROM PACKAGE
ALLOW/DISALLOW PROJECT
INSTALL/UNINSTALL PACKAGE
LIST/ADD/REMOVE ACCOUNTPROVIDERS
SET LABLE ...
```

- Use the `SecurityManager.runQuery()` method of MaxCompute SDK for Java. For more information, see the MaxCompute SDK for Java documentation.

> **Note** The security-related commands of MaxCompute are not SQL commands. Therefore, you cannot create instances to run security-related commands by using SQLTask.

Procedure

1. Run the following command to create a test table named test_label:

```
CREATE TABLE IF NOT EXISTS test_label(
 key STRING,
 value BIGINT
);
```

2. Run the following Java code in MaxCompute Studio to run the **set label** command on the test_label table. For more information about how to use MaxCompute Studio, see Overview.

```
import com.aliyun.odps.Column;
import com.aliyun.odps.Odps;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.OdpsType;
import com.aliyun.odps.TableSchema;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.security.SecurityManager;
public class test {
 public static void main(String [] args) throws OdpsException {
  try {
   // init odps
   Account account = new AliyunAccount("<your_accessid>", "<your_accesskey>");
   Odps odps = new Odps(account);
   odps.setEndpoint("http://service-corp.odps.aliyun-inc.com/api");
   odps.setDefaultProject("<your_project>");
   // set label 2 to table columns
   SecurityManager securityManager = odps.projects().get().getSecurityManager();
   String res = securityManager.runQuery("SET LABEL 2 TO TABLE test_label(key, value);", false);
   System.out.println(res);
  } catch (OdpsException e) {
   e.printStackTrace();
  }
 }
}
```

3. View the result.

4. Verify the result. After the Java code is run, run the `desc test_label` command on the MaxCompute client. In this example, the result in the following figure shows that the **set label** command has taken effect on the test_label table.

> **Note** For more information about security-related commands in MaxCompute, see Authorize users, Column-level access control, Security configurations, and Resource sharing across projects based on package.

1.2.2. Instance Logview

This topic describes how to use MaxCompute SDK for Java to generate a Logview URL for an instance. You can use the Logview URL of an instance to quickly locate problems.

Instance Logview

Background information

Background information

You can view and debug submitted MaxCompute jobs in Logview. For more information, see Use Logview to view job information.

MaxCompute SDK for Java provides the Logview interface. For more information, see MaxCompute SDK for Java.

Description

```
Instance i = odps.instances().get("<instance_id>");
String logviewUrl = odps.logview().generateLogView(i, 7 * 24);
```

Parameters:

- Instance: the name of the instance that you want to view.
- hours: the timeout period. Unit: hours. For example, set the value to 7*24.

> **Note** You can also run the **wait< instance_id>** command on the MaxCompute client to obtain the Logview URL of an instance.

1.2.3. Return error logs

This topic describes how to use the MaxCompute Java SDK to return error logs.

The MaxCompute Java SDK provides the abstract class RetryLogger. For more information, see the MaxCompute Java SDK documentation.

```
public static abstract class RetryLogger {
/**
 * The callback function that enables RestClient to retry if an error occurs.
 * @param e
 *    The error.
 * @param retryCount
 *    The number of retries.
 * @param retrySleepTime
 *    The retry time required next time.
 */
public abstract void onRetryLog(Throwable e, long retryCount, long retrySleepTime);
}
```

You only need to create a RetryLogger subclass, and then use `odps.getRestClient().setRetryLogger(new UserRetryLogger());` to return logs when initializing MaxCompute objects.

Typical example

```
// init odps
odps.getRestClient().setRetryLogger(new UserRetryLogger());
// your retry logger
public class UserRetryLogger extends RetryLogger {
 @Override
 public void onRetryLog(Throwable e, long retryCount, long sleepTime) {
  if (e ! = null && e instanceof OdpsException) {
   String requestId = ((OdpsException) e).getRequestId();
        if (requestId ! = null) {
            System.err.println(String.format(
       "Warning: ODPS request failed, requestID:%s, retryCount:%d, will retry in %d seconds.", requestId, retr
yCount, sleepTime));
     return;
   }
  }
  System.err.println(String.format(
     "Warning: ODPS request failed:%s, retryCount:%d, will retry in %d seconds.", e.getMessage(), retryCount
, sleepTime));
 }
}
```

1.2.4. Set the SQL flag

This topic describes how to use the MaxCompute Java SDK to set the SQL flag.

When using DataWorks or MaxCompute Console to submit SQL statements, you need to set the SQL flag. If you want to use a new data type of MaxCompute and enable it at the session level, add the statement `set odps.sql.type.system.odps2=true;` to set the SQL flag before the SQL query statement that involves the new data type.

When using the SDK to submit SQL statements, do not simply insert the statement used to set the SQL flag into SQL query statements. For example, if you use the Java SDK, you can set the SQL flag as follows:

```
// Create an SQLTask object.
SQLTask task = new SQLTask();
task.setName("foobar");
task.setQuery("select ...") ;
// Set the SQL flag.
Map<String, String> settings = new HashMap<>();
settings.put("odps.sql.type.system.odps2", "true");
... // Set other flags.
task.setProperty("settings", new JSONObject(settings).toString()); // Set the settings property to the JSON string corresponding to other flags.
Instance instance = odps.instances().create(task); // Run the SQL statement.
```

1.2.5. Use SQLTask and Tunnel to export a large amount of data

This topic describes how to use SQLTask and Tunnel to export a large amount of data.

SQLTask

SQLTask is an SQL class of the MaxCompute Java SDK. By using SQLTask, you can easily run SQL statements and obtain the returned results. For more information about SQLTask, see Java SDK.

The `SQLTask.getResult(i)` method returns a list. When running an SQL statement, you can use this method repeatedly to obtain the complete SQL computation result. However, this method has limits. Currently, the SELECT statement can return a maximum of 10,000 data entries to MaxCompute Console. For more information, see the valid values of READ_TABLE_MAX_ROW in Project operations. That is, if you run the SELECT statement in MaxCompute Console or by using SQLTask, you limit the number of data entries to return in each query result to 10,000. If you use the **CREATE TABLE XX AS SELECT** or **INSERT INTO/OVERWRITE TABLE** statement to insert the query result into a table, the number of data entries to return is not limited.

Tunnel

If the query result to be exported is all data in a table or a specific partition, you can use the Tunnel command-line tool to export all data at a time. MaxCompute provides the Tunnel command-line tool and Tunnel SDK. For more information, see Tunnel commands and MaxCompute Tunnel overview.

For example, use the Tunnel command-line tool to export data as follows:

```
>tunnel d wc_out c:\wc_out.dat;
2017-12-16 19:32:08 - new session: 201712161932082dxxxxx total lines: 3
2017-12-16 19:32:08 - file [0]: [0, 3), c:\wc_out.dat
downloading 3 records into 1 file
2017-12-16 19:32:08 - file [0] start
2017-12-16 19:32:08 - file [0] OK. total: 21 bytes
download OK
```

Export more than 10,000 data entries by using SQLTask and Tunnel

SQLTask cannot process more than 10,000 data entries, but Tunnel can. Therefore, you can use SQLTask and Tunnel together to export a large amount of data. Use the following sample code:

```
private static final String accessId = "userAccessId";
private static final String accessKey = "userAccessKey";
private static final String endPoint = "http://service.cn-shanghai.maxcompute.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("-", "_");// Use a random st
ring as the name of a temporary table for storing the exported data.
private static final Odps odps = getOdps();
public static void main(String[] args) {
  System.out.println(table);
  runSql();
  tunnel();
}
/*
   * Download the query result returned by using SQLTask.
   * */
private static void tunnel() {
  TableTunnel tunnel = new TableTunnel(odps);
  try {
    DownloadSession downloadSession = tunnel.createDownloadSession(project, table);
    System.out.println("Session Status is : "+ downloadSession.getStatus().toString());
    long count = downloadSession.getRecordCount();
    System.out.println("RecordCount is: " + count);
    RecordReader recordReader = downloadSession.openRecordReader(0, count);
    Record record;
    while ((record = recordReader.read()) ! = null) {
      consumeRecord(record, downloadSession.getSchema());
    }
    recordReader.close();
```

```
      recordReader.close();
    } catch (TunnelException e) {
      e.printStackTrace();
    } catch (IOException e1) {
      e1.printStackTrace();
    }
  }
  /*
   * Save the data.
   * If only a small amount of data is returned, you can display the query result and copy the data. In actual s
cenarios, you can also use the java.io package to write data to a local file or a file on the remote server to sav
e the query result.
   * */
  private static void consumeRecord(Record record, TableSchema schema) {
    System.out.println(record.getString("username")+","+record.getBigint("cnt"));
  }
  /*
   * Run the SQL statement and save the query result as a temporary table so that you can use Tunnel to do
wnload data.
   * Set the lifecycle of the saved data to only one day. If the data fails to be deleted, it does not occupy too
much storage space.
   * */
  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 project connection of MaxCompute.
   * */
  private static Odps getOdps() {
    Account account = new AliyunAccount(accessId, accessKey);
    Odps odps = new Odps(account);
    odps.setEndpoint(endPoint);
```

```
    odps.setDefaultProject(project);
    return odps;
}
```

2.Python SDK

2.1. Overview

This topic introduces Python on MaxCompute (PyODPS) and its common usage.

Python SDK

Background information

PyODPS is the MaxCompute SDK for Python. PyODPS supports the DataFrame framework and basic operations on MaxCompute objects. You can use PyODPS to analyze data in MaxCompute.

PyODPS supports Python 2.6 and later, and Python 3.

Read the following documentation to get familiar with PyODPS:

- For more information about PyODPS, see PyODPS: ODPS Python SDK and data analysis framework and PyODPS-related articles.
- For more information about how to download *PyODPS*, visit GitHub.
- For more information about how to install PyODPS, see PyODPS installation instructions.
- For more information about how to develop PyODPS, see PyODPS developer guide.

If you have the interest to help build the PyODPS ecosystem, you can perform the following operations:

- Cooperate in writing PyODPS documentation.
- Develop PyODPS code at GitHub.
- Join the DingTalk group for technical communication. To join, find and enter group number 11701793.

Initialization

Before you use PyODPS, run the following command to initialize a connection to MaxCompute by using your Alibaba Cloud account:

```
from odps import ODPS
odps = ODPS('**your-access-id**', '**your-secret-access-key**', '**your-default-project**',endpoint='**you
r-end-point**')
```

Parameter description:

- your-access-id: the AccessKey ID of your Alibaba Cloud account.
- your-secret-access-key: the AccessKey secret of your Alibaba Cloud account.
- your-default-project: the name of the project.
- your-end-point: the endpoint of the region where your MaxCompute project resides. For more information, see Configure endpoints.

After you initialize the connection, you can use PyODPS to manage MaxCompute objects, such as tables, resources, and functions.

Method description

The following table lists the available basic methods that PyODPS provides for you to use on MaxCompute objects.

Item	Method	Description
Project	get_project(project_name)	Obtains a project.
	exist_project(project_name)	Checks whether a project exists.
Table	list_tables()	Lists all tables in a project.
	exist_table(table_name)	Checks whether a table exists.
	get_table(table_name,project=project_name)	Obtains a specified table in the current project or another specified project.
	create_table()	Creates a table.
	read_table()	Reads data from a table.
	write_table()	Writes data to a table.
	delete_table()	Deletes a table.
Table partition	exist_partition()	Checks whether a partition exists.
	get_partition()	Obtains a partition.
	create_partition()	Creates a partition.
	delete_partition()	Deletes a partition.
SQL	execute_sql()/run_sql()	Executes an SQL statement.
	open_reader()	Reads execution results.
Instance	list_instances()	Lists all instances in a project.
	exist_instance()	Checks whether an instance exists.
	get_instance()	Obtains an instance.
	stop_instance()	Terminates an instance.
Resource	create_resource()	Creates a resource.
	open_resource()	Opens a resource.
	get_resource()	Obtains a resource.
	list_resources()	Lists all resources.

Item	Method	Description
	exist_resource()	Checks whether a resource exists.
	delete_resource()	Deletes a resource.
Function	create_function()	Creates a function.
	delete_function()	Deletes a function.
Data upload or download channel	create_upload_session()	Creates a session that uploads data.
	create_download_session()	Creates a session that downloads data.

2.2. SDK for Python

This topic describes how to use MaxCompute SDK for Python to create, view, and delete a table. MaxCompute SDK for Python is also called PyODPS.

Obtain a PyODPS project

A project is a basic operation unit in MaxCompute. For more information, see Project.

You can perform the following basic operations for a project:

- Obtain a project: You can use the `get_project` method of a MaxCompute entry object to obtain a project. The PyODPS node in DataWorks contains a global variable `odps` or `o`, which is the MaxCompute entry. You do not need to manually define the MaxCompute entry.

  ```
  project = o.get_project('project_name')  # If you specify a project, data of the specified project is returned.
  project = o.get_project()                # If you do not specify a project, data of the current project is returned.
  ```

  project_name is required when you use the get_project method.
- Verify that the project exists: Use the `exist_project` method to check whether the project exists.

Perform basic operations on tables

You can perform the following basic operations on tables:

- Use the `list_tables` method of the entry object to query all tables in a project.

  ```
  # Query all tables in a project.
  for table in odps.list_tables():
  ```
- Use the `exist_table` method of the entry object to check whether a specific table exists. Then, use the `get_table` method to obtain the table.

```
t = odps.get_table('table_name')
t.schema
odps.Schema {
 c_int_a           bigint
 c_int_b           bigint
 c_double_a        double
 c_double_b        double
 c_string_a        string
 c_string_b        string
 c_bool_a          boolean
 c_bool_b          boolean
 c_datetime_a      datetime
 c_datetime_b      datetime
}
t.lifecycle
-1
print(t.creation_time)
2014-05-15 14:58:43
t.is_virtual_view
False
t.size
1408
t.schema.columns
[<column c_int_a, type bigint>,
 <column c_int_b, type bigint>,
 <column c_double_a, type double>,
 <column c_double_b, type double>,
 <column c_string_a, type string>,
 <column c_string_b, type string>,
 <column c_bool_a, type boolean>,
 <column c_bool_b, type boolean>,
 <column c_datetime_a, type datetime>,
 <column c_datetime_b, type datetime>]
```

Create a table schema

You can use one of the following methods to create a table schema:

- Create a schema based on table columns and optional partitions.

```
from odps.models import Schema, Column, Partition
columns = [Column(name='num', type='bigint', comment='the column'),
        Column(name='num2', type='double', comment='the column2')]
partitions = [Partition(name='pt', type='string', comment='the partition')]
schema = Schema(columns=columns, partitions=partitions)
schema.columns
[<column num, type bigint>,
 <column num2, type double>,
 <partition pt, type string>]
schema.partitions
[<partition pt, type string>]
schema.names  # Obtain the names of non-partitioned fields.
['num', 'num2']
schema.types  # Obtain the data types of non-partitioned fields.
[bigint, double]
```

- Create a schema by calling the `Schema.from_lists()` method. This method is more convenient, but you cannot directly set comments for columns and partitions.

```
schema = Schema.from_lists(['num', 'num2'], ['bigint', 'double'], ['pt'], ['string'])
schema.columns
[<column num, type bigint>,
 <column num2, type double>,
 <partition pt, type string>]
```

Create a table

You can call the `create_table()` method to create a table by using one of the following methods:

- Use a table schema to create a table.

```
table = o.create_table('my_new_table', schema)
table = o.create_table('my_new_table', schema, if_not_exists=True)  # Create the table only if no table wit
h the same name exists.
table = o.create_table('my_new_table', schema, lifecycle=7)  # Set the lifecycle of the table.
```

- Create a table by specifying the names and data types of the fields to be contained in the table.

```
# Create a non-partitioned table.
table = o.create_table('my_new_table', 'num bigint, num2 double', if_not_exists=True)
# Create a partitioned table with the specified table columns and partition fields.
table = o.create_table('my_new_table', ('num bigint, num2 double', 'pt string'), if_not_exists=True)
```

By default, when you create a table, you can only use the BIGINT, DOUBLE, DECIMAL, STRING, DATETIME, BOOLEAN, MAP, and ARRAY data types. If you need to use other data types such as TINYINT and STRUCT, you must set `options.sql.use_odps2_extension` to True. Example:

```
from odps import options
options.sql.use_odps2_extension = True
table = o.create_table('my_new_table', 'cat smallint, content struct<title:varchar(100), body:string>')
```

Delete a table

You can call the `delete_table()` method to delete an existing table.

```
o.delete_table('my_table_name', if_exists=True)  # Delete a table only if the table exists.
t.drop()  # Call the drop() method to delete a table if the table exists.
```

Manage table partitions

- Check whether a table is partitioned.

```
if table.schema.partitions:
 print('Table %s is partitioned.' % table.name)
```

- Iterate over all the partitions in a table.

```
for partition in table.partitions:
    print(partition.name)
for partition in table.iterate_partitions(spec='pt=test'):
# Iterate over level-2 partitions.
```

- Check whether a partition exists.

```
table.exist_partition('pt=test,sub=2015')
```

- Obtain a partition.

```
partition = table.get_partition('pt=test')
 print(partition.creation_time)
2015-11-18 22:22:27
 partition.size
0
```

- Create a partition.

```
t.create_partition('pt=test', if_not_exists=True)  # Create a partition only if no partition with the same na
me exists.
```

- Delete a partition.

```
t.delete_partition('pt=test', if_exists=True)  # Delete a partition only if the partition exists.
partition.drop()  # Call the drop() method to delete a partition if the partition exists.
```

PyODPS SQL

PyODPS supports MaxCompute SQL queries and provides methods to obtain query results.

- Execute SQL statements.

  You can use the `execute_sql('statement')` and `run_sql('statement')` methods of the entry object to execute SQL statements. For more information about return values, see Task instance.

  ```
  odps.execute_sql('select * from table_name')  # Execute SQL statements in synchronous mode. Threads a
  re blocked until the execution of the SQL statements is completed.
  instance = odps.run_sql('select * from table_name')  # Execute SQL statements in asynchronous mode.
  instance.wait_for_success() # Threads are blocked until the execution of the SQL statements is complete
  d.
  ```

- Obtain SQL query results.

  You can perform the `open_reader` operation on the instance on which SQL statements are executed to read the SQL query results. When the query results are being read, the following situations may occur:

  - The SQL statements return structured data.

    ```
    with o.execute_sql('select * from table_name').open_reader() as reader:
      for record in reader:
      # Process each record.
    ```

  - If the `DESC` command is executed, you can use `reader.raw` to obtain the raw SQL query results.

    ```
    with o.execute_sql('desc table_name').open_reader() as reader:
    print(reader.raw)
    ```

PyODPS resources

Typically, MaxCompute resources are used in UDFs and MapReduce. You can use the following methods to perform basic operations on the resources:

- `list_resources`: lists all resources in a project.
- `exist_resource`: checks whether a resource exists.
- `delete_resource`: deletes a resource. You can also use the Resource object to call the `drop` method to delete a resource.
- `create_resource`: creates a resource.
- `open_resource`: reads a resource.

PyODPS supports two types of resources: file and table.

- File resources

File resources include `FILE`, `PY`, `JAR`, and `ARCHIVE` files.

Common operations on file resources:

- Create a file resource

  You can specify a resource name, file type, and file-like object or string to call the `create_resource` method to create a file resource.

```
resource = o.create_resource('test_file_resource', 'file', file_obj=open('/to/path/file')) # Use a file-like object to create a file resource.
resource = o.create_resource('test_py_resource', 'py', file_obj='import this') # Use a string to create a file resource.
```

- Read and modify a file resource

  You can use one of the following methods to open a resource:

  - Call the `open` method for a file resource to open it.
  - Call the `open_resource` method at the MaxCompute entry to open a file resource.

  The opened object is a file-like object. The opening modes of file resources are similar to the `open` method predefined in Python. The following example demonstrates the opening modes of file resources:

```
with resource.open('r') as fp:  # Open the specified file in read mode.
    content = fp.read()  # Read all content.
    fp.seek(0)  # Return to the beginning of the file.
    lines = fp.readlines()  # Read multiple lines.
    fp.write('Hello World')  # Error. Data cannot be written to a file in read mode.


with o.open_resource('test_file_resource', mode='r+') as fp:  # Open the file in read/write mode.
    fp.read()
    fp.tell()  # Locate the current position.
    fp.seek(10)
    fp.truncate()  # Truncate the file to the specified length.
    fp.writelines(['Hello\n', 'World\n'])  # Write multiple lines to the file.
    fp.write('Hello World')
    fp.flush()  # Manually call the method to submit the update to MaxCompute.
```

  PyODPS supports the following opening modes:

  - `r`: read mode. The file can be opened but data cannot be written to it.
  - `w`: write mode. Data can be written to the file, but data in the file cannot be read. If a file is opened in write mode, the file content is cleared first.
  - `a`: append mode. Data can be added to the end of the file.
  - `r+`: read/write mode. You can read data from and write data to the file.
  - `w+`: This mode is similar to the `r+` mode. The only difference is that the file content is cleared first.
  - `a+`: This mode is similar to the `r+` mode. The only difference is that data can be written only to the end of the file.

  PyODPS also supports the following binary opening modes for some file resources, such as compressed files:

  - `rb`: binary read mode.
  - `r+b`: binary read/write mode.

- Table resources

- Create a table resource

```
o.create_resource('test_table_resource', 'table', table_name='my_table', partition='pt=test')
```

- Update a table resource

```
table_resource = o.get_resource('test_table_resource')
table_resource.update(partition='pt=test2', project_name='my_project2')
```

- Obtain a table and a partition

```
table_resource = o.get_resource('test_table_resource')
table = table_resource.table
print(table.name)
partition = table_resource.partition
print(partition.spec)
```

- Read and write data

```
table_resource = o.get_resource('test_table_resource')
with table_resource.open_writer() as writer:
  writer.write([0, 'aaaa'])
  writer.write([1, 'bbbbb'])
  with table_resource.open_reader() as reader:
    for rec in reader:
      print(rec)
```

DataFrame

PyODPS provides a pandas-like API called PyODPS DataFrame. This API makes full use of the computing power of MaxCompute. For more information, see DataFrame.

Assume that the following tables exist: `pyodps_ml_100k_movies` that contains movie-related data, `pyodps_ml_100k_users` that contains user-related data, and `pyodps_ml_100k_ratings` that contains rating-related data.

1. Create an entry object of MaxCompute.

```
# Create an entry object of MaxCompute.
o = ODPS('**your-access-id**', '**your-secret-access-key**',project='**your-project**', endpoint='**your-end-point**')
```

2. Call a table object and create the DataFrame object named users.

```
from odps.df import DataFrame
users = DataFrame(o.get_table('pyodps_ml_100k_users'))
```

3. Perform the following operations on the DataFrame object:
   - View the fields of DataFrame and the types of these fields based on the dtypes attribute.

```
users.dtypes
```

- Use the head method to obtain the first N data records for a quick data preview.

```
users.head(10)
```

The following table lists the returned results.

-	user_id	age	sex	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
7	8	36	M	administrator	05201
8	9	29	M	student	01002
9	10	53	M	lawyer	90703

- Filter fields.
  - Filter some fields.

```
users[['user_id', 'age']].head(5)
```

The following table lists the returned results.

-	user_id	age
0	1	24
1	2	53
2	3	23
3	4	24
4	5	33

- Exclude some fields. Example:

```
>>> users.exclude('zip_code', 'age').head(5)
```

The following table lists the returned results.

-	user_id	sex	occupation
0	1	M	technician
1	2	F	other
2	3	M	writer
3	4	M	technician
4	5	F	other

- When you exclude some fields, you may want to obtain new columns based on computation. For example, add the sex_bool attribute and set it to True if sex is M. Otherwise, set the sex_bool attribute to False. Example:

```
>>> users.select(users.exclude('zip_code', 'sex'), sex_bool=users.sex == 'M').head(5)
```

The following table lists the returned results.

-	user_id	age	occupation	sex_bool
0	1	24	technician	True
1	2	53	other	False
2	3	23	writer	True
3	4	24	technician	True
4	5	33	other	False

- Query the number of users aged from 20 to 25. Example:

```
>>> users.age.between(20, 25).count().rename('count')
943
```

- Query the numbers of male and female users.

```
>>> users.groupby(users.sex).count()
```

The following table lists the returned results.

-	sex	count
0	F	273

-	sex	count
1	M	670

- To divide users by occupation, obtain the top 10 occupations with the most population, and sort the occupations in descending order.

```
>>> df = users.groupby('occupation').agg(count=users['occupation'].count())
>>> df.sort(df['count'], ascending=False)[:10]
```

The following table lists the returned results.

-	occupation	count
0	student	196
1	other	105
2	educator	95
3	administrator	79
4	engineer	67
5	programmer	66
6	librarian	51
7	writer	45
8	executive	32
9	scientist	31

The DataFrame API provides the value_counts method for the same purpose.

```
>>> users.occupation.value_counts()[:10]
```

The following table lists the returned results.

-	occupation	count
0	student	196
1	other	105
2	educator	95
3	administrator	79
4	engineer	67
5	programmer	66

-	occupation	count
6	librarian	51
7	writer	45
8	executive	32
9	scientist	31

- Show data in a more intuitive graph.

```
%matplotlib inline
```

- Visualize data on a horizontal bar chart.

```
users['occupation'].value_counts().plot(kind='barh', x='occupation', ylabel='prefession')
```

- Visualize data on a histogram. Divide users into 30 groups by age and view the histogram of age distribution. Example:

```
>>> users.age.hist(bins=30, title="Distribution of users' ages", xlabel='age', ylabel='count of users')
```

- Join three tables to obtain a new table and save it.

```
movies = DataFrame(o.get_table('pyodps_ml_100k_movies'))
ratings = DataFrame(o.get_table('pyodps_ml_100k_ratings'))
o.delete_table('pyodps_ml_100k_lens', if_exists=True)
lens = movies.join(ratings).join(users).persist('pyodps_ml_100k_lens')
lens.dtypes
```

In this example, the following information is returned:

```
odps.Schema {
 movie_id                int64
 title                   string
 release_date            string
 video_release_date      string
 imdb_url                string
 user_id                 int64
 rating                  int64
 unix_timestamp          int64
 age                     int64
 sex                     string
 occupation              string
 zip_code                string
}
```

- Divide ages from 0 to 80 into eight age groups.

```
labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']
cut_lens = lens[lens, lens.age.cut(range(0, 81, 10), right=False, labels=labels).rename('age_group')]
```

- View the first 10 ages with only one user.

```
cut_lens['age_group', 'age'].distinct()[:10]
```

The following table lists the returned results.

-	age_group	age
0	0-9	7
1	10-19	10
2	10-19	11
3	10-19	13
4	10-19	14
5	10-19	15
6	10-19	16
7	10-19	17
8	10-19	18
9	10-19	19

- View the total rating and average rating of users in each age group. Example:

```
cut_lens.groupby('age_group').agg(cut_lens.rating.count().rename('total_rating'), cut_lens.rating.mean().rename('average_rating'))
```

The following table lists the returned results.

-	age_group	average_rating	total_rating
0	0-9	3.767442	43
1	10-19	3.486126	8181
2	20-29	3.467333	39535
3	30-39	3.554444	25696
4	40-49	3.591772	15021
5	50-59	3.635800	8704
6	60-69	3.648875	2623
7	70-79	3.649746	197

Configuration

PyODPS provides a series of configuration options, which can be obtained by using `odps.options` The following tables describe configurable MaxCompute options.

- General configurations

Option	Description	Default value
end_point	MaxCompute Endpoint	None
default_project	The default project.	None
log_view_host	The Logview host.	None
log_view_hours	The retention time of Logview. Unit: hours.	24
local_timezone	The time zone used. True indicates the local time, and False indicates UTC. The time zone of pytz can also be used.	1
lifecycle	The lifecycle of all tables.	None
temp_lifecycle	The lifecycle of temporary tables.	1
biz_id	The ID of the user.	None

Option	Description	Default value
verbose	Specifies whether to print logs.	False
verbose_log	The log receiver.	None
chunk_size	The size of the write buffer.	1496
retry_times	The maximum number of request retries.	4
pool_connections	The number of cached connections in the connection pool.	10
pool_maxsize	The maximum capacity of the connection pool.	10
connect_timeout	The time to wait before the connection times out.	5
read_timeout	The time to wait before the read operation times out.	120
completion_size	The maximum number of object completion listing items.	10
notebook_repr_widget	Specifies whether to use interactive graphs.	True
sql.settings	MaxCompute SQL runs global hints.	None
sql.use_odps2_extension	Specifies whether to enable MaxCompute 2.0 language extension.	False

- Data upload or download configurations

Option	Description	Default value
tunnel.endpoint	Tunnel Endpoint	None
tunnel.use_instance_tunnel	Specifies whether to use Instance Tunnel to obtain execution results.	True
tunnel.limited_instance_tunnel	Specifies whether to limit the number of data records obtained by using Instance Tunnel.	True

Option	Description	Default value
tunnel.string_as_binary	Specifies whether to use bytes instead of unicode for data of the STRING type.	False

- DataFrame configuration

Option	Description	Default value
interactive	Specifies whether DataFrame is used in an interactive environment.	Depends on the measured value
df.analyze	Specifies whether to enable non-MaxCompute built-in functions.	True
df.optimize	Specifies whether to enable full DataFrame optimization.	True
df.optimizes.pp	Specifies whether to enable DataFrame predicate push optimization.	True
df.optimizes.cp	Specifies whether to enable DataFrame column pruning optimization.	True
df.optimizes.tunnel	Specifies whether to enable DataFrame tunnel optimization.	True
df.quote	Specifies whether to use a pair of grave accents (``) to mark fields and table names in the backend of MaxCompute SQL.	True
df.libraries	The resource name of the third-party library that is used for DataFrame operation.	None

- PyODPS ML configuration

Option	Description	Default value
ml.xflow_project	The default XFlow project name.	algo_public
ml.use_model_transfer	Specifies whether to use ModelTransfer to obtain the PMML model.	True
ml.model_volume	The volume name used when ModelTransfer is used.	pyodps_volume