MaxCompute

Product Introduction

Product Introduction

What is MaxCompute

MaxCompute is a big data processing platform that processes and stores massive batch structural data to provide effective data warehousing solutions and big data modeling. MaxCompute supports a variety of classic distributed computing models that enable you to solve massive data calculation problems while reducing business costs, and maintaining data security.

MaxCompute seamlessly integrates with DataWorks, which provides one-stop data synchronization, task development, data workflow development, data operation and maintenance, and data management for MaxCompute. For more information, see DataWorks.

Benefits of MaxCompute

Large-scale computing and storage

MaxCompute is suitable for storage and computing large volumes of data (up to PB-level).

Multiple computation models

MaxCompute supports data processing methods based on SQL, MapReduce, Graph, MPI iteration algorithm, and other programming models.

Strong data security

MaxCompute supports all offline business analysis of Alibaba Group with robust multi-layer sandbox protection and monitoring.

Low-cost

MaxCompute can help reduce procurement costs by 20%-30% compared with on-premises private cloud models.

Function

MaxCompute Tunnel

Supports large volumes of historical data channels

Tunnel provides high concurrency data upload and download services. You can use Tunnel to import TB/PB level data from various heterogeneous data sources into MaxCompute, or export data from MaxCompute. As the unified channel for MaxCompute data transmission, Tunnel provides stable and high- throughput services. Tunnel provides RESTful APIs and a Java SDK to facilitate programming.

Real-time and incremental data channels

For real-time data upload scenarios, MaxCompute provides DataHub services with low latency and convenient usage. It is especially suitable for incremental data import. DataHub also supports a variety of data transmission plug-ins, such as Logstash, Flume, Fluentd, Sqoop.

Computing and analysis tasks

MaxCompute provides multiple computing models.

SQL: In MaxCompute, data is stored in tables. MaxCompute provides an SQL query function for the external interface. You can operate MaxCompute similarly to a traditional database software but with the ability to process PB-level data.

Notes:

MaxCompute SQL does not support transactions, index, or UPDATE/DELETE operations.

MaxCompute SQL syntax differs from Oracle and MySQL, notably, you cannot seamlessly migrate SQL statements of other databases into MaxCompute. For more information, see SQL syntax.

After you submit MaxCompute jobs, the jobs can be queued and scheduled for execution. MaxCompute SQL can complete queries at the second- to millisecond-level.

UDF: A user-defined function. MaxCompute provides numerous built-in functions to meet your computing needs, while also supporting the creation of custom functions.

MapReduce: MapReduce is a Java MapReduce programming model provided by MaxCompute and uses the Java programming interface. It simplifies the development process, however, users are recommended to have a basic understanding of the concept of distribution, and relevant programming experience, before using MapReduce.

Graph: Graph in MaxCompute is a processing framework designed for iterative graph computing. Graph jobs use graphs to build models. Graphs are composed of vertices and edges. Vertices and edges contain values. After performing iterative graph editing and evolution, you can get the final result. Typical applications include PageRank, SSSP algorithm, and K-Means algorithm.

SDK

A convenient toolkit provided for developers. For more information, see MaxCompute SDK.

Security

MaxCompute provides powerful security services that fully protects user data. For more information about each function model, see MaxCompute Security Manual.

History

Date	Description
2010	Named ODPS, the service is released as an operational component of Alibaba Group’ s Ant Financial
2013	ODPS is released for beta testing
2013	ODPS v1.0 is released as a commercially available service. A single cluster contains 5,000 servers, with support for multi-level clusters available.
2016	Renamed to MaxCompute, v2.0 is released as a commercially available service.

Definition

Project

Project is the basic unit of operation in MaxCompute. It is similar to the concept of Database or Schema in traditional databases, and sets the boundary for MaxCompute multi-user isolation and access control. You can have multiple project permissions at the same time and, by granting relevant authorization, users can access the objects of another project in their own project, such as Table, Resource, Function and Instance.

To enter a project (in this example, 'my project'), run the **Use Project** command, as follows:

```
use my_project  -- Enter a project named 'my_project'.
```

After running the preceding command, you can enter a project named **my project** and all objects in this project can be operated. **Use Project** is a command provided by the MaxCompute client. For more commands, see Common Commands.

Table

A table is the data storage unit in MaxCompute. A table is a two-dimensional data structure composed of rows and columns. Each row represents a record, and each column represents a field with the same data type. One record can contain one or more columns. The column name and data type comprise the schema of a table.

The operating objects (input, output) of various computing tasks in MaxCompute are tables. You can create a table, delete a table, and import data into a table.

MaxCompute v2.0 supports two types of tables: internal tables and external tables.

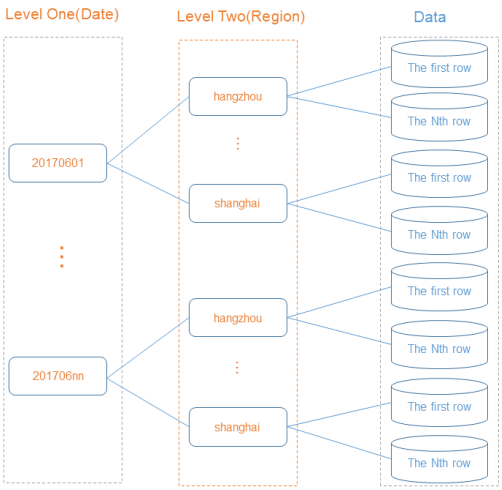
> For internal tables, all data is stored in MaxCompute tables, and the columns in the table can be any of the data types supported by MaxCompute.

For external tables, data is not stored in MaxCompute. Instead, table data can be stored in OSS or Table Store. MaxCompute only records meta information of the table. You can use MaxCompute' s external table to process unstructured data on OSS or Table Store, such as video, audio, genetics, meteorological, and geographic information.

Partition

To improve MaxCompute' s processing efficiency, you can specify a partition when creating a table. Specifically, several fields in the table can be specified as partition columns. A partition is comparable in terms of functionality to a directory under a file system.

In MaxCompute each value of a partition column is used as a partition. You can specify multiple fields of the table as a partition whereby they then function similarly to multi-level directories. If the partitions to be accessed are specified when you use data, then only corresponding partitions are read and a full table scan is avoided, improving processing efficiency while reducing costs.



Partition types

Currently, MaxCompute supports the following partition types: TINYINT, SMALLINT, INT, BIGINT, VARCHAR, and STRING.

Note:

In MaxCompute versions earlier than 2.0, only STRING partition type is supported. Although the partition type can be specified as BIGINT, it is still handled as STRING, and only the schema of the table is indicated as a BIGINT type.

An example is as follows:

```
create table parttest (a bigint) partitioned by (pt bigint);
insert into parttest partition(pt) select 1, 2 from dual;
insert into parttest partition(pt) select 1, 10 from dual;
select * from parttest where pt >= 2;
```

After the execution, the returned result is only one line, because 10 was treated as a STRING and compared with 2, meaning no result can be returned.

Restrictions

When using a partition, the following restrictions apply:

- The maximum number of partition levels for a single table is 6 levels.
- The maximum number of single table partitions is 60,000.
- The maximum number of query partitions for a query is 10,000.

For example, to create a two-level partition table with the date as the level one partition and the region as the level two partition:

```
create table src (key string, value bigint) partitioned by (pt string,region string);
```

When querying, use the partition column as a filter condition in the **WHERE** condition filter:

```
select * from src where pt='20170601' and region='hangzhou';      -- This example is the correct method of using WHERE conditional filter. When MaxCompute generates a query plan, only data of the region 'hangzhou'  under the '20170601' partition is accessed.
select * from src where pt = 20170601; -- This example is an incorrect method of using the WHERE conditional filter. In this example, the effectiveness of the partition filter cannot be guaranteed. Pt is a STRING type. When the STRING type is compared with BIGINT type (20170601), MaxCompute converts both to DOUBLE type, and loss of precision occurs.
```

Some SQL operations on the partitions are less efficient and may cause higher billing, for example, using dynamic partition.

For some MaxCompute commands, when performing operations on partitioned and non-partitioned tables, the syntax is different. For more information, see DDL SQL and DML SQL.

Data type

Basic data types supported by MaxCompute2.0 are listed in the following table. Columns in a MaxCompute table must be any of the listed types. New types include TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, and BINARY data type.

Notes:

If data type such as TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, or BINARY are involved when running an SQL command, the set command set odps.sql.type.system.odps2=true; must be added before the SQL command. The set command and SQL command are then submitted simultaneously. If INT type is involved, and the set command is not added, the INT type is converted to BIGINT, which is 64 bits.

Type	New	Constant	Description
BIGINT	No	100000000000L, -1L	64-bit signed integer, range -263 + 1 to 263 - 1
STRING	No	“abc”,’bcd’,”alibaba” ‘inc’ (Note3)	A single string length can be up to 8M
BOOLEAN	No	TRUE,FALSE	True/False, Boolean type
DOUBLE	No	3.1415926 1E+7	64-bit binary floating point
DATETIME	No	DATETIME ‘2017-11-11 00:00:00’	0001-01-01 00:00:00 ~ 9999-12-31 23:59:59, Date type, use UTC+8 as the standard time system
TINYINT	Yes	Y,-127Y	8-bit signed integer, range -128 to 127
SMALLINT	Yes	32767S, -100S	16-bit signed integer, range -32768 to 32767
INT	Yes	1000,-15645787 (Note1)	32-bit signed integer-231 to 231 - 1
FLOAT	Yes	None	32-bit binary floating point
VARCHAR	Yes	None (Note2)	Variable-length character type, n is

			the length, and the range is 1 to 65535.
BINARY	Yes	None	Binary data type, a single string length can be up to 8M
TIMESTAMP	Yes	TIMESTAMP '2017-11-11 00:00:00.123456789'	It is independent of the time zone and ranges from January 1st 0000 to December 31, 9999 23.59:59.999999999, and is accurate to nanosecond-level.

All data types in the preceding table can be NULL.

Notes:

For INT constant, if the range of INT is exceeded, INT is converted into BIGINT; if the range of BIGINT is exceeded, it is converted into DOUBLE. In MaxCompute versions earlier than 2.0, all INT types in SQL script are converted to BIGINT , for example:

```
create table a_bigint_table(a int);
select cast(id as int) from mytable;
```

To be compatible with earlier MaxCompute versions, MaxCompute 2.0 retains this conversion without setting odps.sql.type.system.odps2 as true, however, a warning is triggered when INT is treated as BIGINT. In this case, we recommend that you change an Int to a Bigint to avoid confusion.

VARCHAR constants can be expressed by STRING constants of implicit transformation.

STRING constants support connections, for example, 'abc' 'xyz' is parsed as 'abcxyz', and different parts can be written on different lines.

Complex Data Types

Complex data types that MaxCompute supports are listed in the following table.

Note:

If a complex data type is involved when you run a SQL command, the set command set odps.sql.type.system.odps2=true; must be added before the SQL command. The set command

and SQL command are then submitted simultaneously.

Type	Definition method	Construction method

ARRAY	array< int >;
	array< struct< a:int,
	b:string >>
		array(1, 2, 3);
		array(array(1, 2);
		array(3, 4))

MAP	map< string, string >; map< smallint, array< string>>	map(“k1” , “v1” , “k2” , “v2”); map(1S, array(‘a’ , ‘b’), 2S, array(‘x’ , ‘y))
STRUCT	struct< x:int, y:int>; struct< field1:bigint, field2:array< int>, field3:map< int, int>>	named_struct(‘x’ , 1, ‘y’ , 2); named_struct(‘field1’ , 100L, ‘field2’ , array(1, 2), ‘field3’ , map(1, 100, 2, 200)

Lifecycle

The lifecycle of a MaxCompute table or partition is measured from the last update time. If the table or partition remains unchanged after a specified time, MaxCompute automatically recycles it. The **specified time** indicates the lifecycle.

Lifecycle units: days, positive integers only.

When a lifecycle is specified for a non-partition table, the lifecycle is counted from the last time the table data was modified (LastDataModifiedTime). If table data has not been changed, MaxCompute recycles the table automatically without manual operation (similar to the drop table operation).

Note:

Lifecycle scanning is started at a scheduled time every day, and entire partitions are scanned. If the partition remains unchanged after its lifecycle, MaxCompute automatically recycles it.

When a lifecycle is specified for a non-partition table, the lifecycle is counted from the last time the table data was modified (LastDataModifiedTime). If table data has not been changed, MaxCompute recycles the table automatically without manual operation (similar to

the drop table operation).

> Note:
>
> Lifecycle scanning is started at a scheduled time every day, and entire partitions are scanned. If the partition remains unchanged after its lifecycle, MaxCompute automatically recycles it.

When a lifecycle is specified for a partition table, MaxCompute determines whether to recycle the partition based on its LastDataModifiedTime. Unlike non-partition tables, a partition table cannot be deleted even when all its partitions have been recycled.

You can set the lifecycle of tables, but not of partitions. The lifecycle of a table can be specified during table creation.

If no lifecycle is specified, according to lifecycle rules the table, or partition, cannot be automatically recycled by MaxCompute.

For more information on specifying or modifying lifecycle during table creation, and modifying a table’ s LastDataModifiedTime, see DDL documentation.

Resources

Resources is a unique concept of MaxCompute. To use user-defined functions (for more information, see UDF), or MapReduce, you must use resources to accomplish tasks.

SQL UDF: After writing a UDF, you must compile it as a Jar package and upload the package to MaxCompute as a resource. Then, when you run this UDF, MaxCompute automatically downloads its corresponding JAR package to obtain the written code. The JAR package is one type of MaxCompute resource.

MapReduce: After writing a MapReduce program, you must compile it as a Jar package and upload the package to MaxCompute as a resource. Then, when running a MapReduce job, the MapReduce framework automatically downloads the corresponding JAR package and obtain the written code. You can upload text files and MaxCompute tables to MaxCompute as different types of resources. Then, you can read or use these resources when running UDF or MapReduce.

MaxCompute provides interfaces for you to read and use resources. For more information, see Use Resourse Example and UDTF Usage .

> **Note:**
>
> For more information about the limitations of resource reading by MaxCompute's user-defined function (UDF) or MapReduce function, see Application Restriction.

Types of MaxCompute resources include:

File type

Table type, which are tables in MaxCompute

> **Note:**
>
> Currently, only BIGINT, DOUBLE, STRING, DATETIME, and BOOLEAN fields are supported in tables referenced by MapReduce.

Jar type, which is compiled Java JAR packages

Archive type, which is the compression type, and is determined by the resource name suffix. Supported compression types include: .zip/.tgz/.tar.gz/.tar/jar

For more information about resources, see Add Resource, Drop Resource, List Resources and Describe Resource.

Function

MaxCompute provides SQL computing capabilities. In MaxCompute SQL, you can use the system's built-in functions to perform certain computing and counting tasks. However, if such SQL functions do not meet your requirements, you can use the Java programming interface provided by MaxCompute to develop user-defined functions (for more information, see UDFs). UDFs can be divided into scalar valued functions, user-defined aggregate functions (UDAFs), and user-defined tables functions (UDTFs). UDFs are used in the same way as the built-in function, that is, you specify the UDF name, and input relevant parameters in SQL.

After writing the code for a UDF, you must compile the code into a JAR package and upload this package to MaxCompute. Then, you can register the UDF in MaxCompute.

For more information, see Function introduction.

Task

A task is a basic computing unit of MaxCompute. Computing tasks such as SQL DML and MapReduce functions are completed by task.

For most user-submitted tasks, MaxCompute first analyzes them and then generates a task execution plan. The execution plan is composed of multiple execution stages that are dependent on each other.

Currently, the execution plan is displayed as a directed acyclic graph. The vertex in the graph designates the execution phase, while edges of the graph indicate the dependence of each execution phase. MaxCompute follows the dependency of execution plan to run each phase.

In an execution stage, multiple processes, known as Workers, complete the computing work. Different Workers handle different data, but the execution logic is the same. Computational tasks are executed directly in MaxCompute instances, for example, Status Instance and Kill Instance.

For MaxCompute tasks that are not computational tasks, such as DDL statement in SQL, these tasks can only read and modify the metadata information in MaxCompute. This means that no execution plan can be analyzed and generated from the task.

> **Note:**
>
> Not all the requests are converted into tasks in MaxCompute, for example, the operations of Project, Resource, UDF and Instance can be completed without MaxCompute tasks.

Instance

In MaxCompute, most tasks are initiated in MaxCompute instances. MaxCompute instances can be in one of two phases: Running and Terminated. The status of the running phase is 'Running', while the status of the Terminated phase can be 'Success', 'Failed' or 'Canceled'. You can query or change the status using the instance ID assigned by MaxCompute. For example:

```
status <instance_id>;          --View the status of a certain instance.
kill <instance_id>; --Stop an instance and set its status as 'Canceled'.
wait <instance_id>; --View the running logs of a certain instance.
```

Reading guidance

The following sections detail recommended readings for first time users, data analysts, project owner/administrators, and developers.

For first time users

MaxCompute Summary — Introduces MaxCompute, including its main function modules.

Quick Start — Provides a step-by-step guide including how to apply for an account, how to install the client, how to create a table, how to authorize a user, how to export/import data, how to run SQL tasks, how to run UDF, and how to run Mapreduce programs.

Basic Introduction — Details key terms and frequently used commands of MaxCompute.

Tools — Lists all key tools used in MaxCompute (also, see MaxCompute Client).

For data analysts

- MaxCompute SQL: Query and analyze massive data that stored on MaxCompute.

For developers

MapReduce: Explains the MapReduce programming interface. You can use the Java API, which is provided by MapReduce, to write MapReduce program for processing data in MaxCompute.

Graph: Provides a set of frameworks for iterative graph computing.

Eclipse Plugin: Facilitates users to use the Java SDK of MapReduce, UDF, and Graph.

Tunnel: Facilitates users to use the Tunnel service to upload batch offline data to MaxCompute, or download batch offline data from MaxCompute.

SDK:

- Java SDK: Provides developers with Java interfaces.
- Python SDK: Provides developers with Python interfaces.

For project owners/administrators

Security: Explains how to grant privileges to a user, how to share resource span projects, how to set project protection, and how to grant privilege by policy, and more.

Billing: Details the pricing of MaxCompute.

Commands that only the project owner can use. For example, the SetProject operation in Others of *Common Commands*.