Table Store

Product Introduction

Product Introduction

Overview

Table Store is a NoSQL database service built on Alibaba Cloud’ s Apsara distributed file system. It enables you to store and access massive structured data in real-time.

Table Store organizes data into instances and tables capable of seamless scaling through data partitioning and load balancing.

Table Store shields applications from faults and errors occurring on the underlying hardware platform, providing fast recovery capability and high service availability.

Table Store manages data with multiple backups using solid state disks (SSDs), enabling quick data access and high data reliability.

When using Table Store, you only need to pay for the resources you reserve and use, without handling complex issues such as cluster resizing, upgrades and maintenance of the database software and hardware.

Advantages

Table Store provides the following advantages:

Scalability

Dynamic adjustment of reserved read/write throughput

When creating a table, you can configure the reserved read/write throughput for an application based on the business access conditions. Table Store schedules and reserves resources based on the table’ s reserved read/write throughput, to minimize the resource usage costs. While using it, Table Store can dynamically adjust the table’ s reserved read/write throughput based on the application

situation.

Unlimited capacity

There is no limit to the amount of data stored in the tables of Table Store. As the size of a table increases, Table Store will adjust the data partitions, allocating more storage space to the table.

Data reliability

Table Store stores multiple data copies on different servers in different racks. When a backup fails, it is quickly restored. Table Store thereby provides excellent data reliability.

High availability

Through the automatic failure detection and data migration, Table Store shields applications from machine and network-related hardware faults to deliver high availability.

Ease of management

Applications do not require tedious and complex operation and maintenance tasks, such as the management of data partitions, software/hardware upgrades, configuration updates and cluster resizing.

Access security

Table Store performs identity authentication for each application request to prevent unauthorized data access and ensure data access security.

High consistency

Table Store ensures high consistency of writing data. Once a successful result is returned for a write operation, applications can read the latest data.

Flexible data models

Table Store tables do not require a fixed format. The column numbers of each row can be different. The value types in the columns with the same name of different rows can also be different. Table Store supports multiple data types, such as Integer, Boolean, Double, String, and Binary.

Pay-As-You-Go

Table Store charges fees based on the actual resources you have reserved and used.

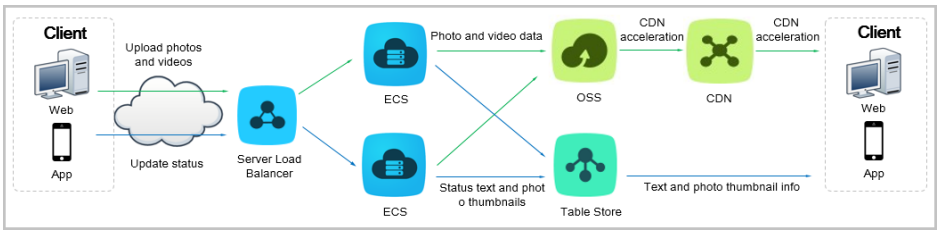
Monitoring integration

You can log on to the Table Store console and obtain the real-time monitoring information including the requests number per second and the average response latency.

Application scenarios

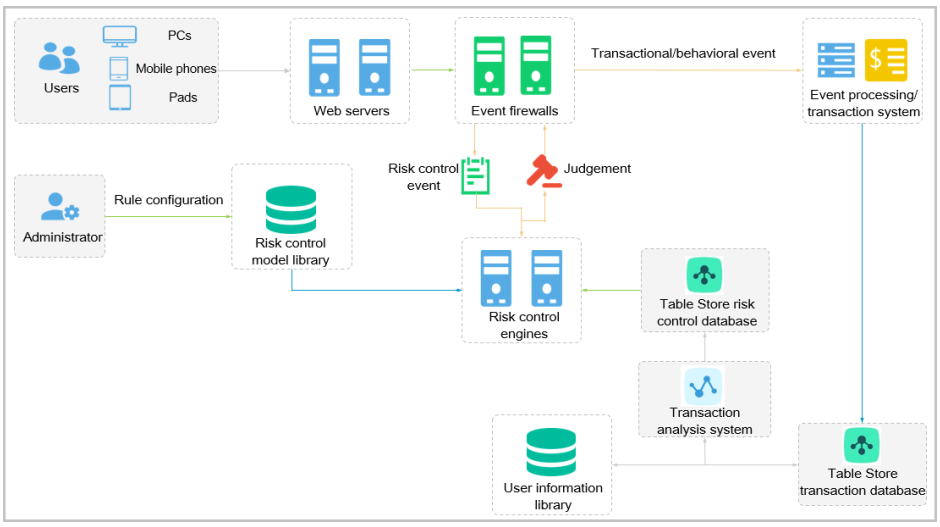
Mobile social media

Table Store stores massive volumes of social information produced by interactions between people, including chats, comments, threads, and likes. At relatively low cost, table store meets the needs of applications that feature significant traffic fluctuations and high concurrency while requiring low latency. This service stores images and videos on OSS and, with CDN acceleration, provides an optimal user experience.



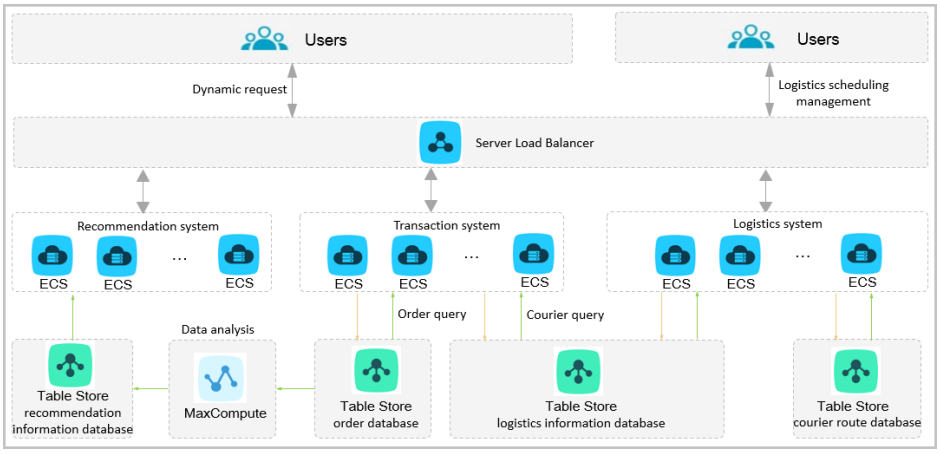
Financial risk control

The low latency, high concurrency, elastic resources, and Pay-As-You-Go billing method of this service allow your risk control system to operate in optimal condition, so you can firmly control transaction risks. Meanwhile, the flexible data structure allows your business model to rapidly evolve to meet market demands.



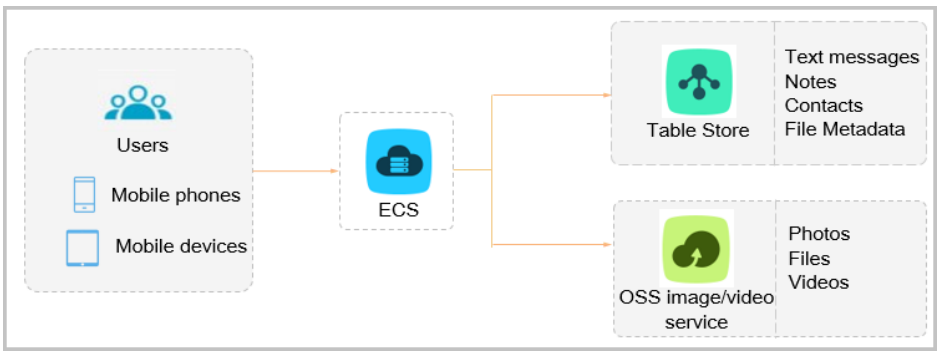
E-commerce logistics

Using Table Store means you do not have to worry about the large volumes of transaction orders and logistics tracking information. The elastic resources and Pay-As-You-Go billing method allow you to easily cope with all your holiday promotions. Supported by MaxCompute, this service makes precision marketing simple. With pickup addresses and courier route tracking, you can optimize delivery routes to increase efficiency.



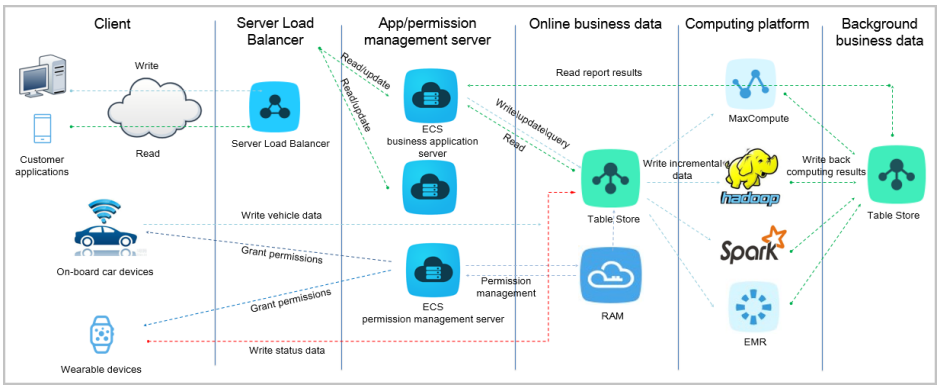
Cloud storage solution

Table Store offers you powerful security and writing reliability for data, including for large volumes of contacts, text messages, call logs, memos, and other structured data, as well as metadata for images, videos, and documents.



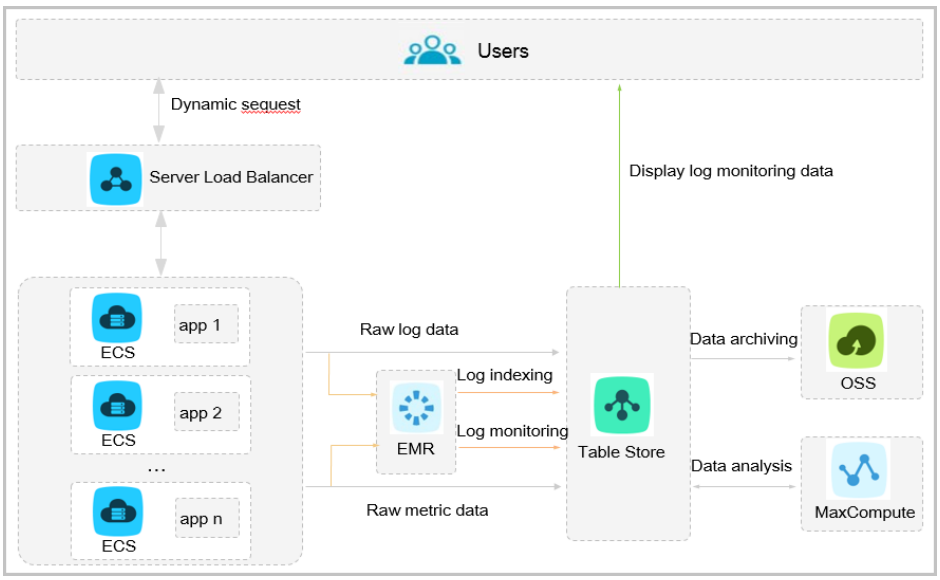
IoT big data

IoT devices generate a continuous stream of status data. RAM’ s Security Token Service (STS) allows you to take full advantage of the high concurrency of Table Store and significantly reduce your application server deployment costs. Table Store’ s high performance keeps you informed of the status of each IoT device.



Log monitoring

You can write application monitoring and log information to Table Store. This service provides an online log query function. Relying on MaxCompute’ s offline data processing service, you can analyze monitoring data and logs to discover valuable data.

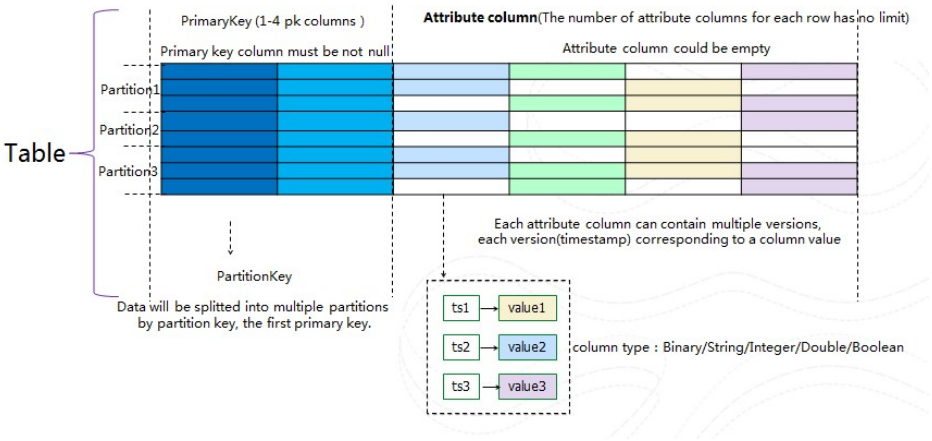


Glossary

Data model

Overview

The data model of Table Store is described by Table, Row, Primary Key and Attribute, as shown in the following picture:



A table is a set of rows, and a row consists of the Primary Key and Attribute.

The Primary Key and Attribute columns consist of names and values.

All rows in a table must contain the Primary Key columns with the same number and name. But the number, name and data type of the Attribute columns contained in the rows can be various.

Each Attribute column can contain multiple versions and each version (timestamp) corresponds to a column value, which is different from that of a Primary Key column.

> **Note:** Timestamp is the sum of the milliseconds counted from 1970-01-01 00:00:00 UTC to the time when data is written.

Example

The following example illustrates two rows in a table. The **ID** column is the Primary Key column.

ID	Type	ISBN	PageCount	Length
‘4776’	timestamp = 1466676354000, value = ‘Book’	timestamp = 1466676354000, value = ‘123*45678912345’	timestamp = 1466676354000, value = 666	-
‘6555’	timestamp = 1466676354000, value = ‘Music’	-	-	timestamp = 1466676354000, value = ‘400’ ; timestamp = 1466762754000, value = ‘500’

ID is the Primary Key of the given table. The rows with the ID of '4776' and '6555' have different attributes and can be stored in the same table.

The Attribute column **Type** of the row with ID '4776' only has one version. The version is 1466676354000 and the data is 'Book'.

The Attribute column **Length** of the row with ID '6555' has two versions. The data of version 1466676354000 is '400' and the data of version 1466762754000 is '500'.

Max Versions

Max Versions is a data table attribute used to indicate the maximum number of data versions in each Attribute column of a data table. When the number of the data versions in an Attribute column exceeds the value of Max Versions, the earliest versions will be deleted asynchronously.

After creating a data table, you can use the UpdateTable interface to dynamically change the value of Max Versions.

Because the data exceeding the value of Max Versions will be cleared asynchronously, pay attention to the following information when you dynamically modify the value of Max Versions:

- The data that exceeds the value of Max Versions is invalid and cannot be viewed or accessed, even though it is not deleted.
- If you reduce the value of Max Versions, the data that exceeds the new value will be deleted asynchronously.
- If you increase the value of Max Versions, the data that exceeds the old value but is not deleted yet can be accessed again.

Time To Live

Time To Live, shortened to TTL, is a data table attribute measured in seconds. It indicates the validity period of data. To save data storage space and reduce storage costs, Table Store clears the data that exceeds TTL in the background.

- Set TTL when creating a table. If you do not need a data expiration date, set TTL to **–1**.
- After creating a data table, you can use the UpdateTable interface to dynamically change the value of TTL.
- TTL is measured in seconds. For example, to get a data validity period of 30 days, set TTL to 30*24*3600.

For a data table whose TTL is 86400 (one day), all the attribute columns with the versions earlier than 1468944000000 (divided by 1000 and converted to seconds to get 2016-07-20 00:00:00 UTC) will expire at 2016-07-21 00:00:00 UTC and be automatically cleared.

As the expired data will be cleared asynchronously, pay attention to the following information when you dynamically modify the value of TTL:

The date of data that exceeds the value of TTL is invalid and cannot be viewed or accessed, even though it is not deleted.

If you reduce the value of TTL, the date of data that exceeds the later value will be deleted asynchronously.

If you increase the value of TTL, the date of data that exceeds the earlier value but is not deleted yet can be accessed again.

Valid offset of data version

Valid offset of data version is a data table attribute measured in seconds. To prevent from writing the unexpected data, the server checks the versions of attribute columns when processing the writing requests. Writing data to a specified row fails if the row has an attribute column whose version is earlier than the current writing time minus the value of Valid offset of data version, or is later than or equal to the current writing time plus the value of Valid offset of data version.

The valid version range for Attribute columns is **[Data written time – the value of Valid offset of data version, Data written time + the value of Valid offset of data version)**. The data written time is the sum of the seconds counted from 1970-01-01 00:00:00 UTC to the time when data is written. The versions (expressed in milliseconds) of the Attribute columns must fall into the valid version range after the versions are divided by 1000 and converted to seconds.

For example, if the valid version range of a data table is 86,400 (one day), then at 2016-07-21 00:00:00 UTC, only the data with versions later than 1468944000000 (converted to seconds to get 2016-07-20 00:00:00 UTC) but earlier than 1469116800000 (converted to seconds to get 2016-07-22 00:00:00 UTC) can be written to the data table. If a row has an Attribute column whose version is 1468943999000 (converted to seconds to get 2016-07-19 23:59:59 UTC, which is less than a day), the data cannot be written to the row.

If you do not set the value of Valid offset of data version when creating a data table, the data table uses its default value 86400.

After creating a data table, you can use the UpdateTable interface to dynamically change the value.

Valid offset of data version must be set to a non-zero value based on the number of seconds during the period from 1970-01-01 00:00:00 UTC to the current time.

Primary key and attribute

Primary Key

The Primary Key is the unique identifier of each row in a table. It consists of 1 to 4 Attribute columns.

When a table is being created, the Primary Key must be defined. Provide the column name and data type of each Primary Key column with fixed sequence of the Primary Key columns to define a Primary Key.

The data types of the Primary Key column can only be String, Integer and Binary. For the Primary Key column with String or Binary data type, the size of the column value must not exceed 1 KB.

Attribute

The data of rows are stored in the Attribute columns. The number of attribute columns for each row has no limit.

Version

When writing data, you can specify the versions of the Attribute columns. If you do not specify any versions, the server generates the versions of the Attribute columns based on the current timestamp (expressed by the number of milliseconds that have elapsed since 1970-01-01 00:00:00 UTC). You can specify the maximum number of versions per column or the version range to limit the data that can be read from each row.

When the number of versions written to an Attribute column exceeds the value of Max Versions, the data of earlier versions is discarded so that the number of remaining versions is equal to the value of Max Versions.

A version number also indicates the time when data is generated. The version is expressed by the number of milliseconds that have elapsed since 1970-01-01 00:00:00 UTC. For a data table whose TTL is set to 86400 (one day), the data in the Attribute column whose version is 1468944000000 (2016-07-20 00:00:00 UTC) will expire at 2016-07-21 00:00:00 UTC and be automatically deleted.

Note:

- During TTL comparison and Valid offset of data version calculation, versions in milliseconds must be converted to seconds via dividing by 1000.
- In the case that versions (timestamps) are determined by the server, data will be cleared after the time (in seconds) indicated by TTL has elapsed since the data is written.
- To prevent invalid written data, the system denies the writing of expired data. For example, at 2016-07-21 00:00:00, the data whose version is earlier than 1468944000000 (2016-07-20 00:00:00 UTC) cannot be written to a data table whose TTL is 86400.

- To prevent writing errors, the system requires that the version (converted to seconds) of the Attribute column to which the data is written should be within the range of [Data written time – The value of Valid offset of data version, Data written time + The value of Valid offset of data version).

Column naming conventions

The Primary Key and Attribute columns follow the same naming conventions:

The column name can be only constructed by English alphabets, Arabic numbers and underline (_).

The first character must be an English alphabetical letter or the underline (_).

All characters are case sensitive.

The name length is in the range of 1 to 255 characters.

Samples of correct column names:

_id

Message

Samples of incorrect column names:

sn序列号_21

Failure: Chinese characters are not allowed.

5store

Failure: The first character cannot be a number.

shopping(new)

Failure: No other symbols are allowed except the underline (_).

Data types of column values

Table Store supports five data types of column values, as shown in the following table.

Data type	Definition	Whether can be Primary Key	Size limitation
String	UTF-8, could be empty	YES	For Primary Key column, not greater than 1 KB. For Attribute column, refer to Restricted items.
Integer	64 bit Integer	YES	8 Bytes
Double	64 bit Double	NO	8 Bytes
Boolean	True/False	NO	1 Byte
Binary	Could be empty	NO	For Primary Key column, not greater than 1 KB. For Attribute column, refer to Restricted items.

Partition key

The first Primary Key column constituting a Primary Key is also called a partition key. Table Store checks the range where the partition key of each row is located and automatically allocates the data in this row to the corresponding partition and machine to achieve load balancing.

Rows with the same partition key belong to the same partition. A partition may contain multiple partition keys. A partition key is the smallest partition unit. The data under a partition key cannot be split. To avoid a situation where the partition is too large to be split, we recommend that the total data volume of all rows under a single partition key should not exceed 1 GB.

To improve load balancing, Table Store splits and merges partitions according to specific rules. This process is automated without application intervention.

Read/write throughput capacity

The read/write throughput is measured by read/write capacity units (CUs) , which is the smallest billing unit for the data read and write operations.

One read CU indicates that 4 KB data is read from the table, and one write CU indicates that 4 KB data is written into the table. Data smaller than 4 KB during the operation will be rounded up to an integer. For example, writing 7.6 KB data will consume two write CUs, and reading 0.1 KB data will consume one read CU.

When applications perform Table Store read/write operations through an API, the corresponding amount of read/write CUs will be consumed.

Reserved throughput

The reserved read/write throughput is an attribute of a table. When creating a table, the application specifies the read/write throughput reserved for the table. Configuring the reserved read/write throughput does not affect the table’ s access performance and service capability.

The reserved read/write throughput can be set to 0. When the reserved read/write throughput is greater than 0, Table Store assigns and reserves enough resources for the table according to this configuration, so as to ensure lower resource costs. The reserved read/write throughput of a table is dynamically changeable using the UpdateTable request.

With a non-zero reserved read/write throughput, your use of Table Store is still billed even when there are no read and write requests. For this reason, Table Store limits the maximum reserved read/write throughput to 5,000 CUs per table (no more than 5,000 CUs of read throughput and write throughput respectively). If you require more than 5,000 CUs of reserved read/write throughput for a single table, open a ticket to increase the throughput.

The reserved read/write throughput of a non-existent table is regarded as 0. To access a non-existent table will consume one additional read CU or one additional write CU based on the actual operation.

Additional throughput

The additional read/write throughput refers to the portion of the actual consumed read/write throughput that exceeds the reserved read/write throughput. Its statistical period is one second. For example, when the reserved read throughput of a table is set to 100 CUs and the read operation within a second actually consumes 120 CUs, then the additional read throughput consumed within the second is 20 CUs. If the reserved read throughput of a table is set to 0 CU, the read throughput consumed by each read operation of the table is the additional read throughput.

When billing is based on the additional read/write throughput mode, it is difficult to estimate the amount of compute resources that need to be reserved for data tables. Table Store must provide sufficient service capability to deal with the access traffic spikes. For this reason, the unit price of additional read/write throughput is higher than that of the reserved read/write throughput. Set a proper value of the reserved read/write throughput to reduce costs.

> **Notice:** Because it is difficult to accurately reserve resources based on the additional read/write throughput, in extreme situations, Table Store may return an error OTSCapacityUnitExhausted to the application when an access to a single partition key consumes 10,000 CUs per second. In this case, policies such as backoff retry are used to reduce the frequency of access to the table.

For more information, refer to Table Store tables and Billing methods.

Region

Region refers to the service region of Alibaba Cloud. Table Store is deployed in many service regions offered by Alibaba Cloud. Select the Table Store service of different regions according to your requirements.

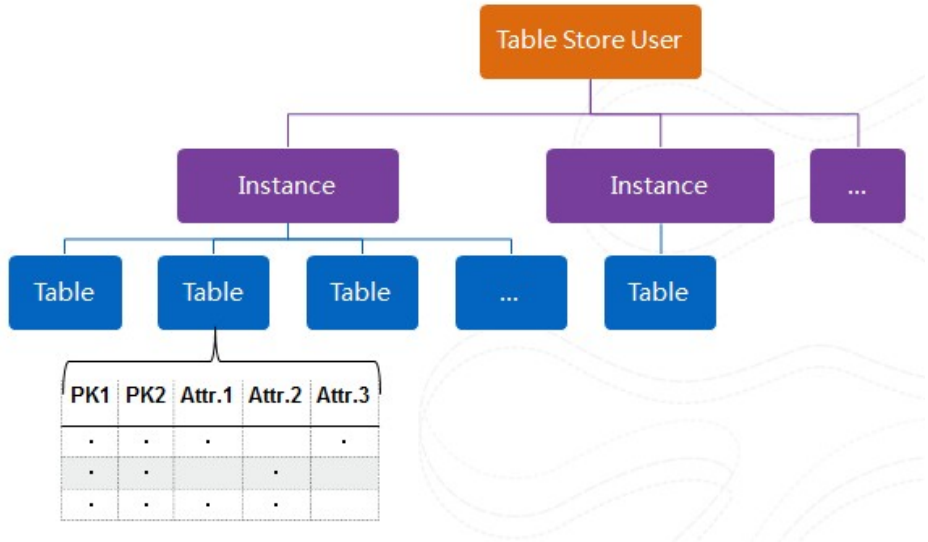
The following table lists the regions supported by Table Store:

Region	Region Name
China East 1 (Hangzhou)	cn-hangzhou
China East 2 (Shanghai)	cn-shanghai
China North 2 (Beijing)	cn-beijing
China South 1 (Shenzhen)	cn-shenzhen
Hong Kong	cn-hongkong
Singapore	ap-southeast-1
US West 1 (Silicon Valley)	us-west-1
Asia Pacific NE 1 (Japan)	ap-northeast-1
Germany 1 (Frankfurt)	eu-central-1
Asia Pacific SE 2 (Sydney)	ap-southeast-2

Instance

Overview

Instance is a logical entity in Table Store to manage tables as database in RDBMS (the relational database management system). Create an instance first in the Table Store console after activating the Table Store service, and then create and manage their tables in this instance. Instance is the basic unit of Table Store’ s resource management. Table Store implements the access control and resource metering at the instance level.



Users usually create different instances for different business to manage the correlative tables. Users also can create different instances for one business based on different development, testing and production purposes.

Table Store allows one Alibaba Cloud account to create 10 instances at most, and at most 64 tables can be created in each instance. If more instances or tables are needed, open a ticket.

Naming conventions

The name of an instance is the unique identifier in a single service region. Users can create instances with the same name in different service regions. The naming conventions of instances are as follows:

- Instance name can be only constructed by English alphabetical letters, Arabic numbers, and dashes (-).
- The first character must be an English letter.
- The last character must not be a dash symbol (-).
- The name length is in the range of 3–16 Bytes.

Instance type

Table Store supports two instance types: high-performance instance and capacity instance.

> **Notice:** The instance type cannot be modified after the instance is created.

The two instance types have the same functions and support petabyte-sized data volumes for a single table, but they differ in costs and application scenarios.

High-performance instance

The high-performance instances support millions of read-write transactions per second (TPS) with a 1 ms average latency of read and write operations per row. The high-performance instances are applicable in scenarios requiring high read and write performance and concurrency, such as gaming, financial risk control, social networking apps, recommendation systems, and public opinion monitoring.

Capacity instance

The capacity instances provide very high write throughput and write performance comparable to that of the high-performance instances but with lower costs. However, the capacity instances are inferior to the high-performance instances in terms of the read performance and concurrency. The capacity instances are applicable to the services with high write frequency but low read frequency, and the services with high affordability but relatively low performance requirements, such as access to log monitoring data, Internet of Vehicles data, device data, time sequence data, and logistic data.

> **Notice:** The capacity instances do not support reserved read/write throughput. All reads and writes are billed based on the additional read/write throughput.

Instance type supported by region

Region	High-performance instance	Capacity instance
China East 1 (Hangzhou)	Support	Support
China East 2 (Shanghai)	Support	Support
China North 2 (Beijing)	Support	Support
China South 1 (Shenzhen)	Support	Support
Hong Kong	Planning	Support
Singapore	Support	Planning
US West 1 (Silicon Valley)	Support	Planning
Asia Pacific NE 1 (Japan)	Planning	Support
Germany 1 (Frankfurt)	Planning	Support
Asia Pacific SE 2 (Sydney)	Planning	Support

Endpoint

Each instance corresponds to an endpoint that is also known as the connection URL. The endpoint needs to be specified before any operations on the tables and data of Table Store.

To access the data in Table Store from the Internet, the endpoint uses the following format:

```
http://instanceName.region.ots.aliyuncs.com
```

To access the data in Table Store from an Alibaba Cloud ECS instance of the same region through the intranet, the endpoint uses the following format:

```
http://instanceName.region.ots-internal.aliyuncs.com
```

For example, to access the Table Store instance in China East 1 (Hangzhou) region, with the instance name of myInstance:

```
Endpoint of the Internet access: http://myInstance.cn-hangzhou.ots.aliyuncs.com
Endpoint of the intranet access: http://myInstance.cn-hangzhou.ots-internal.aliyuncs.com
```

Better performance can be expected on the intranet.

If an application accesses Table Store from an ECS instance in VPC, the endpoint uses the following format:

```
http://vpcName-instanceName.region.vpc.ots.aliyuncs.com
```

For example, the service address used by an application in China East 1 (Hangzhou) region to access the instance named myInstance from a network named testVPC:

```
Endpoint of VPC access: http://testVPC-myInstance.cn-hangzhou.vpc.ots.aliyuncs.com
```

This VPC access address is only used for the access initiated by servers in the testVPC network.