Table Store

Product Introduction

Product Introduction

What is Table Store

Table Store is a NoSQL database service built on Alibaba Cloud’ s Apsara distributed operating system that can store and access large volumes of structured data in real time by allowing you to:

Organize data into instances and tables that can seamlessly scale using data partitioning and load balancing.

Protect applications from faults and errors that may occur on the underlying hardware platform, providing fast recovery capability and high service availability.

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

When using Table Store, you only pay for the resources you reserve and use. Services including cluster resizing, upgrades, and maintenance of database software and hardware are managed free of charge.

Benefits

Table Store provides the following benefits:

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 business requirements and data access conditions. Table Store

schedules and reserves resources, based on the table’ s reserved read/write throughput, to minimize the resource usage costs. You can dynamically adjust the table’ s reserved read/write throughput based on the application.

Unlimited capacity

The amount of data stored in Table Store tables is unlimited. If a table size increases, Table Store adjusts the data partitions for immediate storage space allocation to the resized table.

Data reliability

Table Store stores multiple data copies across different servers in different racks. If a failure occurs, backup servers with copied data immediately restore services, resulting in zero data loss.

High availability

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

Ease of management

Table Store automatically manages complex tasks, such as the management of data partitions, software upgrades, hardware upgrades, configuration updates, and cluster resizing, allowing you to focus on growing your business.

Secure access platform

Table Store performs identity authentication for each application request, preventing unauthorized data access and ensuring data access security.

High consistency

Table Store guarantees 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, and the value types in columns of the same name but different rows, can be varied. Table Store supports multiple data types, such as Integer, Boolean, Double, String, and Binary.

Multi-tenancy mechanism

Table Store uses a shared storage mechanism, which allows multiple instances of different users to share the same cluster resource. Table Store uses the data partition as the smallest unit , which supports the load balancing mechanism at the data partition level to isolate the impact between different instances.

Pay-As-You-Go

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

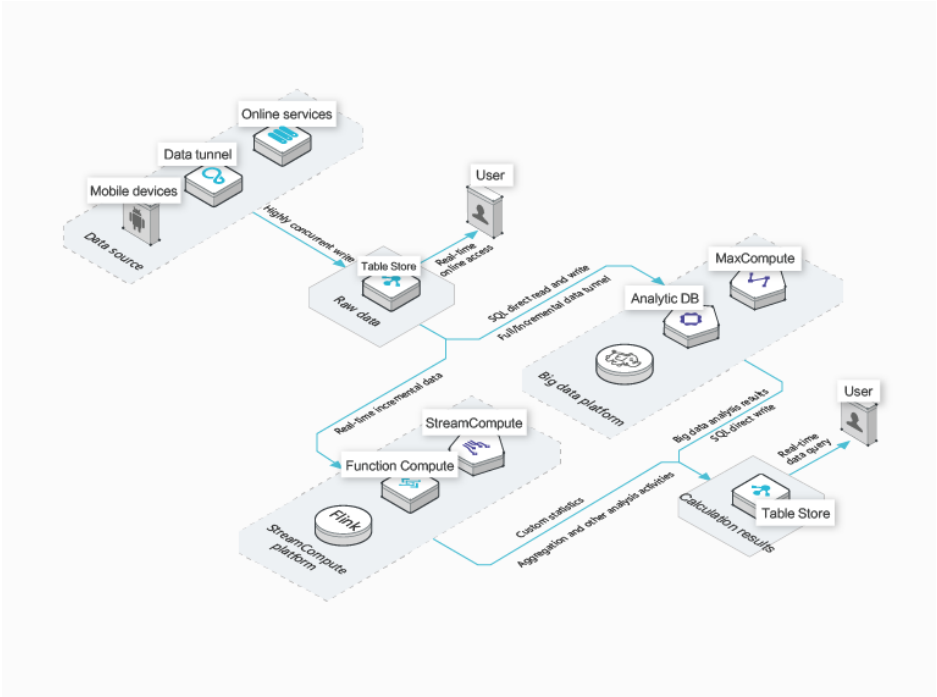
Monitoring integration

The Table Store console provides real-time monitoring information, including the request number per second and the average response latency.

Scenarios

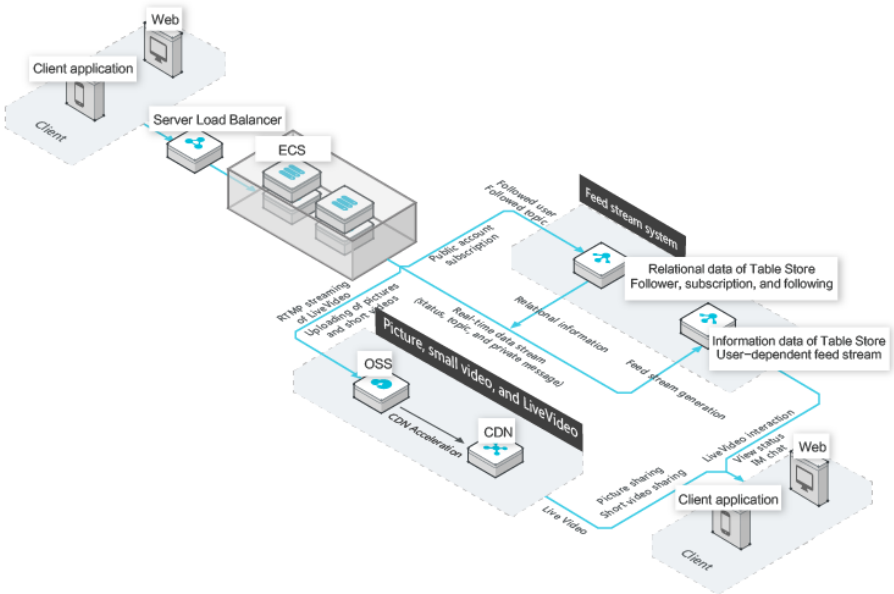
Big data storage and analysis

Table Store provides low-cost, low-latency, and high-concurrency storage and online access of high volumes of data. In addition, Table Store provides incremental and full data tunnels, and also SQL direct read and write on big data analysis platforms, such as MaxCompute. An efficient incremental streaming read interface is provided for easy computing of real-time data streams.



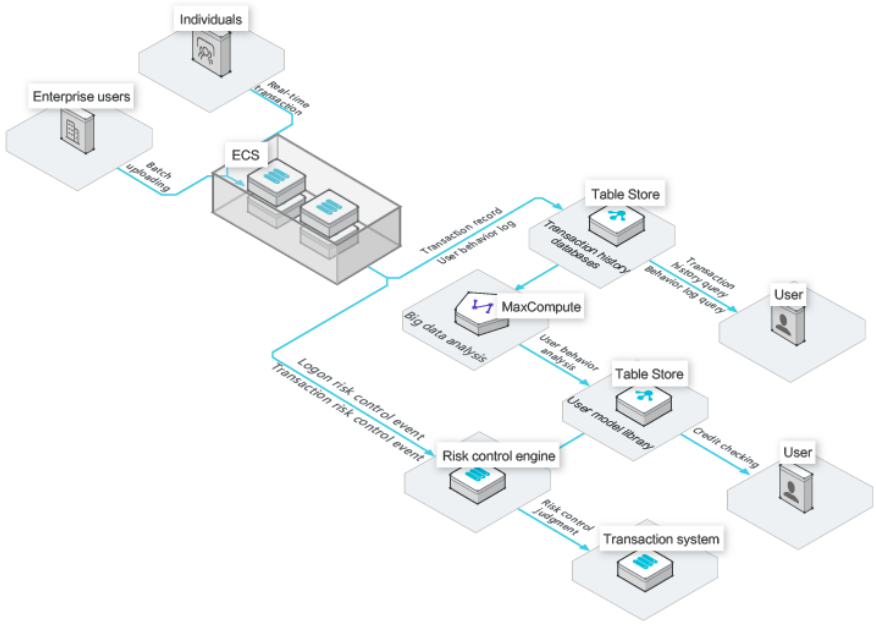
Social feed stream storage

Table Store can store high volumes of social information produced by interactions between people, including instant messaging (IM) chats, comments, and threads. The elastic resources stored in Table Store are billed in Pay-As-You-Go method. At a very competitive cost, Table Store can meet the needs of applications that feature significant traffic fluctuations and high concurrency when low latency is required. Table Store stores images and videos on OSS and, with CDN acceleration, provides an optimal user experience.



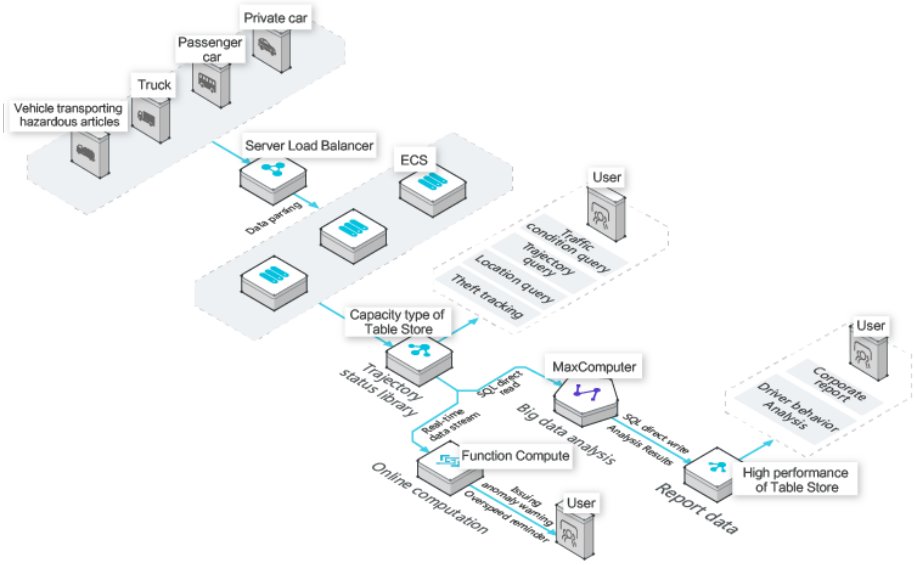
Financial risk control

The advantages of Table Store such as low latency, high concurrency, and Pay-As-You-Go billing of elastic resources combine to optimize the financial risk control system, allowing you to strictly limit transaction risks. Flexible data structures enable fast iteration of business models as market needs shift.



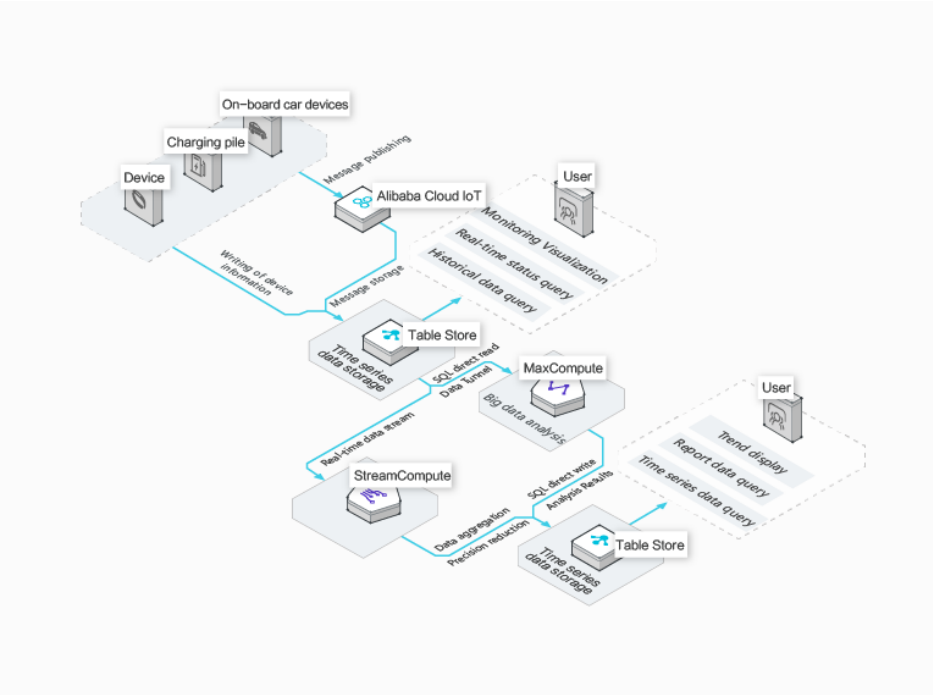
IoV data storage

A single table can store petabytes of data without distributing data in separate databases and tables, which simplifies the business logic. The schema-free data model enables easy access to the monitoring data of different vehicle-mounted devices. Table Store can be seamlessly integrated with multiple big data analysis platforms and real-time computing services for ease of real-time online query and business report analysis.



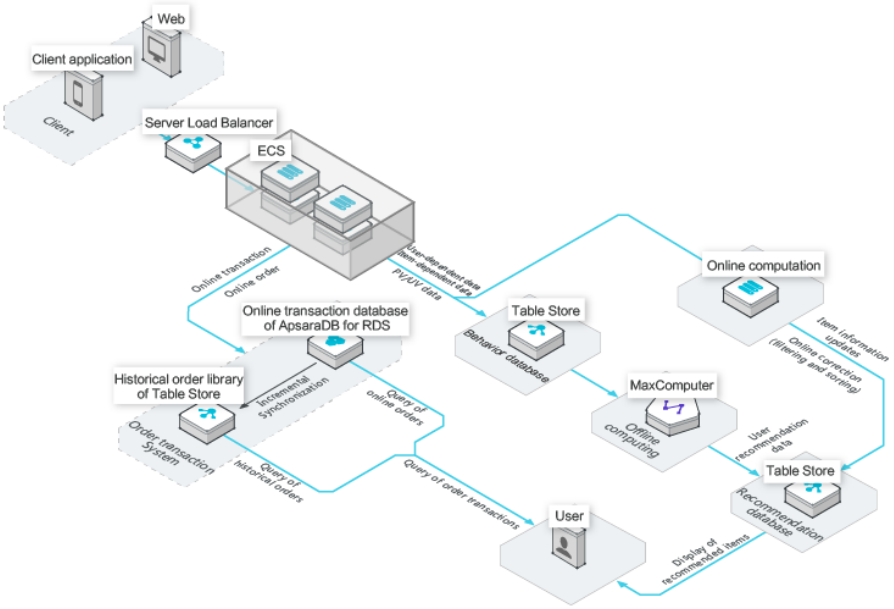
IoT time series data storage

With a single table capable of storing petabytes of data and processing thousands of queries per second (QPS), Table Store makes it easy to store the time series data of IoT devices and monitoring systems. The big data analysis SQL direct read function and the efficient incremental streaming read interface provide an easy way of offline data analysis and real-time streaming computing.



E-commerce recommendation

Table Store makes it possible for you to deal with data volumes and access performance with ease when handling a large number of historical transaction orders. Combined with MaxCompute, Table Store enables precision marketing, elastic resource storage, and Pay-As-You-Go billing. Table Store allows you to easily manage peak hours when a large majority of customers go online.

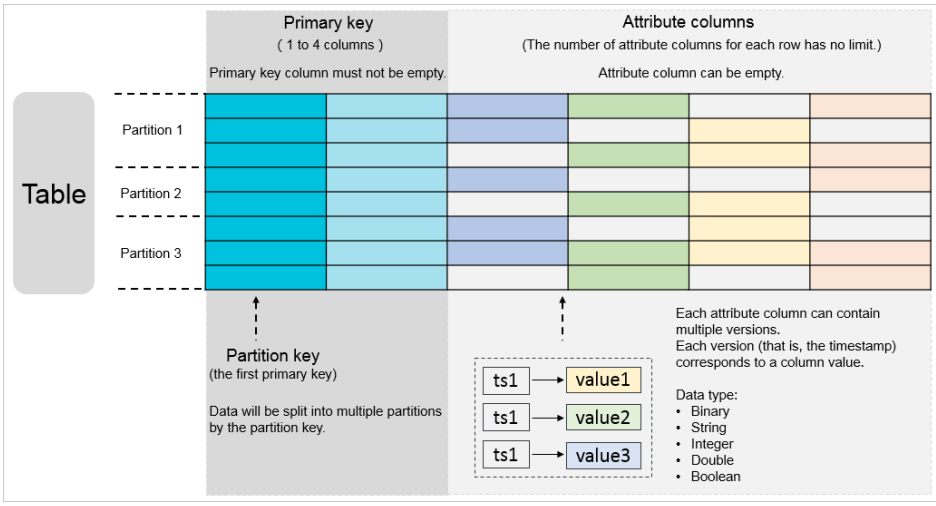


Terms

Data model

Overview

The data model of Table Store is defined by Table, Row, Primary Key, and Attribute, as shown in the following diagram.



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 primary key columns with the same number and name. However, the number, name, and data type of attribute columns within the rows can vary.

Each attribute column can contain multiple versions, and each version (that is, the 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. 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 data versions in an attribute column exceeds the value of Max Versions, the earliest versions are deleted asynchronously.

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

Data that exceeds the value of Max Versions is cleared asynchronously. When dynamically modifying the value of Max Versions, the following conditions may apply:

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, data that exceeds the new value is deleted asynchronously.

If you increase the value of Max Versions, data that had exceeded the previous Max Version limit, and has not been deleted, can be accessed again.

Time To Live

Time To Live (TTL) is a data table attribute measured in seconds that indicates the validity period of data. To save data storage space and reduce storage costs, Table Store automatically clears any data

that exceeds TTL. TTL usage is described as follows:

Set TTL during table creation. If you do not need a data expiration date, set TTL to **–1**.

After table creation, you can use the UpdateTable operation to dynamically change the value of TTL.

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

Expired data is cleared asynchronously. When dynamically modifying the value of TTL, the following conditions may apply:

Data with a date 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, data with a date that exceeds the later value is deleted asynchronously.

If you increase the value of TTL, data that had exceeded the previous TTL value, and has not been deleted, can be accessed again.

Max Version Offset

Max Version Offset is a data table attribute that is measured in seconds. To prevent the writing of unexpected data, the server checks the attribute columns' versions when processing writing requests. Writing data to a specified row fails if the row has an attribute column in which:

Its version is earlier than the current writing time minus the value of Max Version Offset.

Its version is later than or equal to the current writing time plus the value of Max Version Offset.

The valid version range for attribute columns is **[Data written time – the value of Max Version Offset, Data written time + the value of Max Version Offset)**. 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. Versions of the attribute

columns (expressed in milliseconds) must, after being divided by 1000 and converted to seconds, fall into the valid version range.

For example, if the valid version range of a data table is 86400 (one day), then at 2016-07-21 00:00:00 UTC, only 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.

Max Version Offset 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 date and time.

If you do not set the value of Max Version Offset when creating a data table, the data table uses the default value **86400**.

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

Primary key and attribute

Primary key

A primary key is the unique identifier of each row in a table. It consists of one to four columns.

When a table is created, the primary key must be defined. To define a primary key, the column name and data type of each primary key column, and the fixed sequence of the primary key columns, must be provided.

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

Partition key

The first primary key column is also called a partition key. Table Store checks the range where the partition key value 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 value belong to the same partition. A partition may contain multiple partition key values. The partition key value is the smallest partition unit. The data of the same partition key value 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 value does not exceed 10 GB.

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

Attribute

Data of rows are stored in the attribute columns. The number of attribute columns for each row is unlimited.

Version

When writing data, you can specify the attribute columns' versions. 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 01/01/1970 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 Max Versions value.

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

Note:

- During TTL comparison and Max Version Offset calculation, versions in milliseconds must be converted to seconds by dividing by 1000.
- If versions are determined by the server, data is cleared after the time (in seconds) indicated when TTL has elapsed since the data was written.
- To prevent invalid written data, the system denies the writing of expired data. For example, at 21/07/2016 00:00:00, the data in which the version is earlier than 1468944000000 (20/07/2016 00:00:00 UTC) cannot be written to a data table where the TTL is 86400.
- To prevent writing errors, the system requires that the attribute column version to which the data is written is within range of [Data written time – The value of Max Version Offset, Data written time + The value of Max Version Offset).

Column naming conventions

The primary key and attribute columns follow the same naming conventions, in which:

Each name can be 1 to 255 characters in length.

Only uppercase letters, lowercase letters, digits, and underscores (_) are permitted.

The first character must be an uppercase letter, lowercase letter, or underscore (_).

All characters are case sensitive.

Data types of column values

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

Data type	Definition	Permitted as primary key	Size limitation
String	UTF-8, can be empty	Yes	For a primary key column, not greater than 1 KB. For an attribute column, see Limits.
Integer	64 bit Integer	Yes	8 Bytes.
Double	64 bit Double	No	8 Bytes.
Boolean	True/False	No	1 Byte.
Binary	Can be empty	Yes	For a primary key column, not greater than 1 KB. For an attribute column, see Limits.

Read/write throughput

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.
- One write CU indicates that 4 KB data is written into the table.
- Data smaller than 4 KB during the operation is rounded up to the nearest CU. For example, writing 7.6 KB data consumes two write CUs, and reading 0.1 KB data consumes one read CU.

When applications use an API to perform Table Store read/write operations, the corresponding amount of read/write CUs is 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.

For reserved throughput billing, the reserved throughput value is always used to calculate the hourly fee even if an application consumes less than the specified amount of throughput.

For example, suppose that an application reads 3 KB of data per record and 80 records per second from a table. In this case, the application consumes 80 capacity units per second.

If you set the reserved read throughput to 80 capacity units per second, the hourly fee is calculated by using the following formula: Hourly Fee = 80 reserved read throughput capacity units x Hourly Price for Reserved Read Throughput. It is enough for 288000 (80 x 3600 seconds) reads per hour.

Note

- Reserved read/write throughput can be set to zero.
- When the reserved read/write throughput is greater than zero, Table Store assigns and reserves enough resources for the table according to this configuration to guarantee low resource costs.
- The reserved read/write throughput of a table can be dynamically changed using the UpdateTable operation. For a non-zero reserved read/write throughput, your Table Store service is billed even if no read and write requests are made. To guarantee billing accuracy, Table Store limits the maximum reserved read/write throughput to 5000 CUs per table (neither read throughput nor write throughput can exceed 5000 CUs). If you require more than 5000 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 zero. To access a non-existent table, one additional read CU or one additional write CU is consumed depending on the actual operation.

Applications dynamically modify the reserved read/write throughput configuration of the table through the UpdateTable 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 refresh interval is one second.

In the following example, the reserved read throughput is set to 100 units. T0, T1, and T2 show the reserved read throughput and the additional read throughput that an application consumed in three consecutive seconds:

- T0: The actual read throughput consumption is 120 units. The consumption of the reserved read throughput and the consumption of the additional read throughput are 100 units and 20 units, respectively.
- T1: The actual read throughput consumption is 95 units. The consumption of the reserved read throughput and the consumption of the additional read throughput are 100 units and 0 units, respectively.
- T2: The actual read throughput consumption is 110 units. The consumption of the reserved read throughput and the consumption of the additional read throughput are 100 units and 10 units, respectively.

In the three consecutive seconds, the consumption of the reserved read throughput is 100 units, and the total consumption of the additional read throughput is 30 units.

> **Note:** Table Store uses the average value per hour to calculate the consumption of the reserved throughput and uses the total amount per hour to calculate the consumption of the additional throughput.

For 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 is required to provide sufficient service capability to effectively handle access traffic spikes. For this reason, the unit price of additional read/write throughput is higher than that of reserved read/write throughput. To make sure that low costs are maintained, we recommend that you set an appropriate value of the reserved read/write throughput.

> **Note:** 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 an 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, see Table Store tables and billing methods.

Region

Region refers to a service region of Alibaba Cloud. Table Store is deployed across many service regions. You can select the most suitable region according to your requirements.

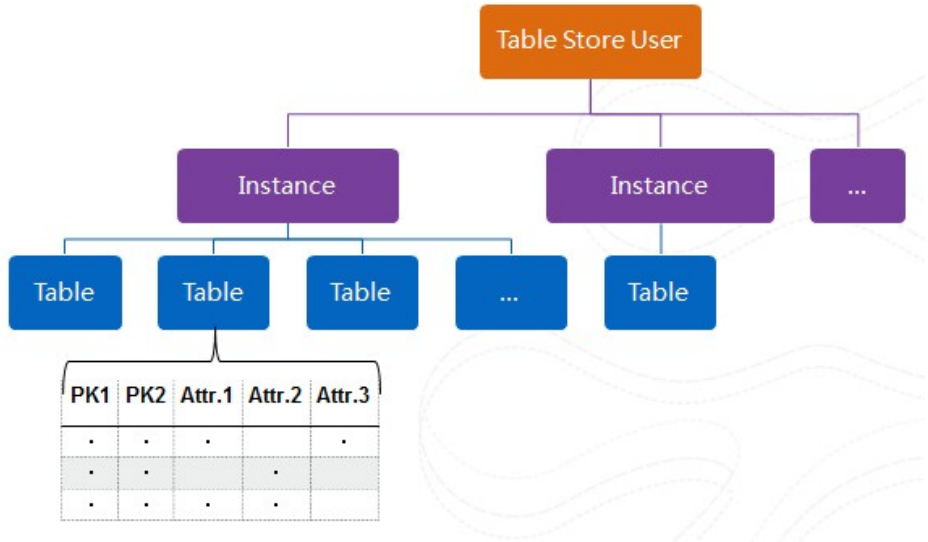
The following table lists the regions supported by Table Store.

Region name	RegionID
China East 1 (Hangzhou)	cn-hangzhou
China East 2 (Shanghai)	cn-shanghai
China North 2 (Beijing)	cn-beijing
China North 3 (Zhangjiakou)	cn-zhangjiakou
China North 5 (Huhehaote)	cn-huhehaote
China South 1 (Shenzhen)	cn-shenzhen
Hong Kong	cn-hongkong
Singapore	ap-southeast-1
US East 1 (Virginia)	us-east-1
US West 1 (Silicon Valley)	us-west-1
Asia Pacific NE 1 (Japan)	ap-northeast-1
Germany 1 (Frankfurt)	eu-central-1
Middle East 1 (Dubai)	me-east-1
Asia Pacific SE 2 (Sydney)	ap-southeast-2
Asia Pacific SE 3 (Kuala Lumpur)	ap-southeast-3
Asia Pacific SE 5 (Jakarta)	ap-southeast-5
Asia Pacific SOU 1 (Mumbai)	ap-south-1

Instance

Overview

An instance is a logical entity in Table Store used to manage tables as a database in a relational database management system (RDBMS). After activating Table Store, create an instance in the Table Store console and then create and manage tables within this instance. An instance is the basic unit in the resource management system of Table Store. Table Store implements access control and resource metering at the instance level.



You can create different instances for multiple businesses to manage their respective tables. You can also create multiple instances for one business based on different development, testing, and production purposes.

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

Name format

The name of each instance is unique within each region. You can create instances of the same names across different service regions. Naming rule for each instance must:

Contain English letters, numbers, and hyphens(-)

Start with English letters

Not end with a hyphen(-)

Be case-insensitive

Be 3 Bytes to 16 Bytes in length

Not contain the words, such as ‘ali’, ‘ay’, ‘ots’, ‘taobao’,and ‘admin’

Instance type

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

Note: An instance type cannot be modified once the instance is created.

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

High-performance instance

High-performance instances support millions of read-write transactions per second (TPS) with 1 ms average latency of read and write operations per row. High-performance instances are suitable for scenarios requiring high read and write performance and concurrency, such as gaming, financial risk control, social networking applications, product recommendation systems, and public opinion sensing.

Capacity instance

Capacity instances provide write throughput and write performance comparable to that of the high-performance instances, but with lower costs. However, the capacity instances do not equal the read performance and concurrency of high-performance instances. The capacity instances are suitable for services with high write frequency but low read frequency, and services with high affordability and reduced performance requirements. This includes access to log monitoring data, Internet of Vehicles data, device data, time sequence data, and logistics data.

Note: 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 North 3 (Zhangjiakou)	In development	Support
China North 5 (Huhehaote)	In development	Support
China South 1 (Shenzhen)	Support	Support
Hong Kong	In development	Support
Singapore	Support	In development
US East 1 (Virginia)	Support	In development
US West 1 (Silicon Valley)	Support	In development

Asia Pacific NE 1 (Japan)	In development	Support
Germany 1 (Frankfurt)	In development	Support
Middle East 1 (Dubai)	In development	Support
Asia Pacific SE 2 (Sydney)	In development	Support
Asia Pacific SE 3 (Kuala Lumpur)	In development	Support
Asia Pacific SE 5 (Jakarta)	In development	Support
Asia Pacific SOU 1 (Mumbai)	In development	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:

```
https://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:

```
https://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 for Internet access: https://myInstance.cn-hangzhou.ots.aliyuncs.com
Endpoint for intranet access: https://myInstance.cn-hangzhou.ots-internal.aliyuncs.com
```

Better performance, such as lower response latency and no unnecessary Internet traffic, can be expected through the intranet.

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

```
https://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: https://testVPC-myInstance.cn-hangzhou.vpc.ots.aliyuncs.com
```

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

Stream

Stream is a data table attribute used for real-time analysis of incremental data streams and incremental data synchronization.

You can enable the Stream feature when creating a data table. For more information, see Table Store Stream.

Stream Expiration Time

Stream Expiration Time indicates the duration that the incremental data remains valid. It is set when the Stream feature is enabled.

The expiration time duration is represented in hours and can be set to a maximum of 24 hours. For example, set the expiration time to 24 if you want the data to remain valid for a duration of 24 hours.

Table Store clears the incremental data beyond the expiration time so that expired data cannot be queried. This feature makes available storage space for other incremental data.

You can disable and then re-enable Stream to change the expiration time duration. Once Stream is enabled again, the previous data cannot be queried.

Limits

Item	Limit	Description
Number of instances created	Up to 10 instances	To raise the limit, open a

under an Alibaba Cloud user account		ticket.
Number of tables in an instance	Up to 64 tables	To raise the limit, open a ticket.
Instance name length	3-16 Bytes	Can contain uppercase and lowercase letters, digits, and hyphens. Must begin with a letter, and must not end with a hyphen. Must not contain the words, such as 'ali', 'ay', 'ots', 'taobao' and 'admin'.
Table name length	1-255 Bytes	Can contain uppercase and lowercase letters, digits, and underscores. Must begin with a letter or underscore.
Column name length	1-255 Bytes	Can contain uppercase and lowercase letters, digits, and underscores. Must begin with a letter or underscore.
Number of primary key columns	1-4 columns	Must be at least one column.
Size of string type primary key column values	Up to 1 KB	A single primary key column's string type column value is limited to 1 KB.
Size of string type attribute column values	Up to 2 MB	A single attribute column's string type column value is limited to 2 MB.
Size of binary type primary key column values	Up to 1 KB	A single primary key column's binary type column value is limited to 1 KB.
Size of binary type attribute column values	Up to 2 MB	A single attribute column's binary type column value is limited to 2 MB.
Number of attribute columns in a single row	Unlimited	A single row can contain an unlimited amount of attribute columns.
The number of attribute columns written by one request	Up to 1024 columns	The number of attribute columns written by one PutRow, UpdateRow, or BatchWriteRow request in a single row.
Data size of a single row	Unlimited	The total size of all column names, and column value data, for a single row is unlimited.

Reserved read/write throughput for a single table	Up to 5000	To raise the limit, open a ticket.
Number of columns in a read request' s columns_to_get parameter	Up to 128 columns	The maximum number of columns obtained in a row of data in the read request.
Table-level operation QPS	10	The QPS of a table-level operation on an instance must not exceed 10. For table-level operations, see Table operations.
Number of UpdateTable operations for a single table	Raise: Unlimited Lower: Unlimited	The reserved read/write throughput for each table can be raised or lowered unlimited times within a calendar day (from 00:00:00 to 00:00:00 of the next day in UTC time).
UpdateTable frequency for a single table	Maximum of one update every 2 minutes	The reserved read/write throughput for a single table cannot be adjusted beyond the frequency of once every 2 minutes.
The number of rows read by one BatchGetRow request	Up to 100 rows	N/A
The number of rows written by one BatchWriteRow request	Up to 200 rows	N/A
Data size of one BatchWriteRow request	Up to 4 MB	N/A
Data returned by one GetRange operation	Up to 5000 rows or 4 MB	The data returned by a single operation cannot exceed 5000 rows or 4 MB. Otherwise, the excessive data will be read with a returned token.
The data size of an HTTP Request Body	Up to 5 MB	N/A