ALIBABA CLOUD

Alibaba Cloud

Tablestore

FAQ

Document Version: 20201023

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.General FAQ

1.1. What are the differences between Tablestore and traditional database services such as MySQL and SQL Server?

Tablestore is a NoSQL database service. It is built on cloud computing technologies to store and manage distributed structured and semi-structured data. Tablestore and traditional database services (RDBMS such as MySQL and SQL Server) differ in data models and technologies that are implemented.

Data models of Tablestore are based on two-dimensional tables. Tablestore uses rows and columns. However, Tablestore does not use the tabular schema of rows and columns that are found in most traditional database systems. Instead, Tablestore uses a storage model that is optimized for the specific requirements of the type of data being stored. Columns in each row can be different. You can add or remove attribute columns as needed.

Traditional database services provide a variety of features such as index, transaction, and various SQL statements. Tablestore is superior in scalability to support large amounts of data (several hundred TB) and higher concurrency level of access (100,000 QPS per table).

In programming, Tablestore provides a unified HTTP RESTful API that does not support SQL syntax. You are charged for the actual resources you use such as storage resources and read/write capacity units (CUs) when you use Tablestore.

If your problem persists, contact after-sales support personnel.

1.2. What do I need to specify when I create a table?

Tablestore can store semi-structured data. When you create a table, you need only to specify one to four primary key columns. Attribute columns are optional.

Each table can contain an unlimited number of attribute columns. Rows in each table can contain attribute columns that differ in column quantity and supported data types. When you use an application to write data, Tablestore requires that the application have the names of primary key columns and attribute columns and the values in all these columns specified.

What is partition key?

The first primary key column is called the partition key. When the size of data in a table reaches a specified value, Tablestore distributes data to different partitions based on the range of values in the partition key to implement load balancing

By default, data in a table is distributed to one partition when the table is created. When multiple partitions are created for the table, data distributed to a partition shares the same specified range of the values in the partition key. The range of values in a partition key is sorted by the system based on different data types of the values such as INTEGER and STRING.

The number of partitions affects performance when you access data and the utilization of reserved read/write throughput. When a table contains multiple partitions, the reserved read/write throughput is pre-distributed to each partition based on a proportion.

How can I select a partition key?

When you create a table, select a partition key based on the following rules to ensure access performance if the table stores large amounts of data.

- Do not use primary key columns whose range of values is narrow, such as customer gender Male or Female.
- Do not use primary key columns whose values may be frequently accessed. For example, the primary key column that contains timestamps is used as the partition key.
- We recommend that you use primary key columns that contain data that is less frequently accessed as the partition key. For example, you can select the UserID column.

What do I do if I cannot predict which data will be frequently accessed when I select a partition key?

We recommend that you calculate the hash value of data in a column before you write data to a partition key. For example, when you write a row of data, you can use simple algorithms to calculate the hash value of characters from a user ID, concatenate the hash value and the actual value, and store the concatenation result of the user ID in the table. You can use this lightweight operation to distribute access. Note that a value in a partition key is concatenated by a hash value and actual value. You cannot perform the GetRange operation based on the partition key.

Why Tablestore limits the number of tables that an account can have?

Each Tablestore account can create a maximum of 10 instances. Each instance can have 64 tables created. In other words, a maximum of 640 tables can be created in an account.

The following situations are considered to increase the number of tables:

- Large amounts of data and high access performance

  Unlike traditional SQL-based database services such as MySQL that addresses access to large amounts of data by using sharding, Tablestore resolves these bottlenecks by implementing a distributed solution.

  You can store structured or semi-structured data in a large sparse table without having to worry about performance issues when the amount of data to access is extremely large.

- Rapid business increase

  In addition to the increase based on existing data and increase of visits, you may use Tablestore to provide services for your customers such as the third-party partners and providers. To provide services for the providers, you can deploy a table for each provider after you activate Tablestore. The number of tables may quickly reach the upper limit. The constant increase of the upper limit can cause maintenance costs uncontrollable and make global data analysis more difficult.

  We recommend that you use Tablestore in a nontraditional way and store large amounts of structured and semi-structured data in a table.

- Considerations for Tablestore

The distributed mode of Tablestore makes the number of tables a resource attribute of Tablestore. When the scale of the Tablestore cluster is specified, Tablestore imposes no limits on the maximum number of tables. The scalability of Tablestore allows you to increase the upper limit. Tablestore specifies the upper limit of tables in an account in terms of controllability of Tablestore resources.

To increase the upper limit of tables in an account, submit a ticket.

1.3. What are the naming conventions of tables, columns, and instances?

This topic describes the naming conventions of Tablestore tables, columns, and instances.

Names of tables and columns

Conventions:

- The name can contain only letters, digits, and underscores (_).
- The name is case-sensitive.
- The name cannot start with a digit.
- The name must be 1 to 255 characters in length.

Names of instances

The name of an instance must be unique within a region. The name can be the same across different regions.

Conventions:

- The name can contain only letters, digits, and hyphens (-).
- The name is case-insensitive.
- The name must start with a letter and cannot end with a hyphen (-).
- The name must be 3 to 16 bytes in length.

1.4. What are primary key, partition, and partition key?

Primary key

Each row is uniquely identified by the primary key. You must specify columns that compose a primary key when you create a table. These columns are called primary key columns. Each primary key column must contain values. You must ensure that the combination of values of primary key columns can identify a row. The data types of values in primary key columns cannot be modified after the data types are specified.

Partition and partition key

Tablestore automatically splits tables into different partitions to balance data loads. The first column of a primary key is called the partition key.

Data that shares the same partition keys is distributed to the same partition. You can use the partition keys to modify data in the same partition, which ensures consistency.

The following figure shows a part of a mailing list from an email system. Information about the primary key and partition key:

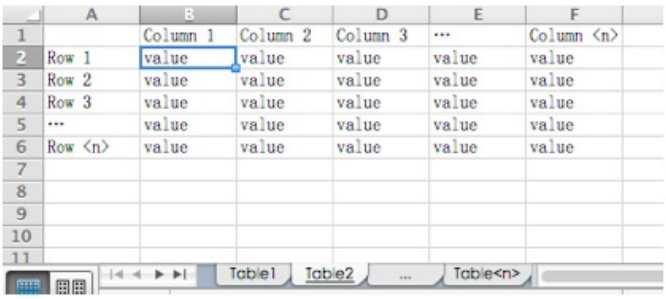
- UserID, ReceiveTime, and FromAddr are primary key columns that uniquely identify an email. UserID is the partition key.
- ToAddr, MailSize, Subject, and Read are attribute columns that are used to store information related to emails.
- Data whose user ID is U0001 and U0002 is distributed to a partition. Data whose user ID is U0003 and U0004 is distributed to another partition.

UserID	ReceiveTime	FromAddr	ToAddr	MailSize	Subject	Read
U0001	1998-1-1	eric@demo.com	bob@demo.com	10000	Hello	Y
U0001	2011-10-20	alice@demo.com	bob@demo.com; vivian@demo.com	15000	Fw: Greetings	Y
U0001	2011-10-21	alice@demo.com	team1@demo.com	8900	Review	N
U0001	2011-10-24	lucy@demo.com	vteam@demo.com	500	Meeting Request	N
U0001	2011-11-9	alice@demo.com	bob@demo.com; team@demo.com; robin@demo.com	1250	Re: Review	N
U0001	2011-11-11	alice@demo.com	team@demo.com	3300	Re: Review	Y
U0002	1999-12-1	bill@demo.com	tom@demo.com; windy@demo.co; bill@demo.com	4300	Re: Hello	Y
U0002	2010-3-18	windy@demo.com	tom@demo.com	500	Meeting Request	Y
U0003	2010-11-11	robin@demo.com	vteam@demo.com	21000	Have a nice day	Y
U0003	2010-12-10	bob@demo.com	vteam@demo.co; alice@demo.com	10000	Re: help	N
U0004	1999-6-29	vivian@demo.com	vivian@demo.com	50	Test	N

数据分片键	主键	列	数据分片

1.5. What are the differences between Tablestore and spreadsheets in tables, rows, columns, and values?

Tablestore tables store structured data. You can query, insert, modify, and delete data in the tables. One user can create multiple tables. Data in the tables is organized in forms of columns, rows, and values.



The preceding figure compares concepts such as table between Tablestore and spreadsheets.

- **table**: similar to sheets in spreadsheets. Each sheet corresponds to a table.
- **row**: similar to rows in spreadsheets. Each row consists of column names and values in the columns.
- **column**: similar to the column in spreadsheets. All data in a column shares the same dimension attributes.
- **value**: similar to the cell in spreadsheets. Each value indicates the value of a column in a specific row. Unlike spreadsheets, Tablestore can contain columns that contain null values. The following data types are supported for values: STRING, INTEGER, BOOLEAN, DOUBLE, and BINARY. Data types of values in primary key columns can only be STRING, INTEGER, or BINARY.

1.6. How does Tablestore verify a user?

Tablestore uses symmetric signatures to verify whether a request is sent by the resource owner and whether the response is sent by Tablestore.

After you create an Alibaba Cloud account, you can create an AccessKey pair in the User Management Console. For more information, see . An AccessKey ID identifies a user. An AccessKey secret is used to sign and verify requests and responses. Keep the AccessKey secret secure.

When you send a request, you must include the request plaintext, AccessKey ID, and the verification code generated based on the signature calculated by using AccessKey secret and request plaintext.

After Tablestore receives the request, Tablestore uses the AccessKey ID to obtain the corresponding AccessKey secret and signs the plaintext by using the same method. If the calculated verification code is the same as what is provided by the client, the request is valid.

The client uses the same method to verify whether the calculated verification code is the same as what is provided by the server. If the verification codes are the same, the response is valid.

1.7. How can I obtain an AccessKey ID and AccessKey secret?

An AccessKey pair is similar to the username and password. An AccessKey pair is used to call an operation of a cloud service, whereas the username and password are used to log on to the console. If you do not need to call an operation, an AccessKey pair is not required.

Create an AccessKey for a RAM user account

1. Log on to the RAM console using your Alibaba Cloud account.
2. If you have not created a RAM user, in the left-side navigation pane, select **Users** and click

Create User. Skip this step if you have created a RAM user.

3. In the left-side navigation pane, click **Users**. On the displayed page, click the target user name to access the user details page.
4. In the User Access Key area, click **Create Access Key**.
5. After completing the phone verification, on the **Create User Access Key** page, expand **Access Key Details** to view the AcessKeyId and AccessKeySecret. Click **Save Access Key Information**.

   > **Notice** After the AccessKey is created, it cannot be viewed in the console. You must save your AccessKey properly and keep it confidential.

6. Click **Authorize** on the right of the RAM user to grant permissions. For example, you can grant the **AliyunOSSFullAccess** permission to the RAM user for OSS management.

1.8. How does Tablestore authenticate user identities?

Only an AccessKey pair that consists of an AccessKey ID and AccessKey secret can be used to authenticate a user identity. The user can access any resources in your account if they pass the authentication.

An account may have different AccessKey pairs, but Tablestore resources available to the account remain the unchanged regardless of which AccessKey pair is used.

To access Tablestore as a RAM user, the RAM user must be authorized by an Alibaba Cloud account to access specified resources. For more information about RAM users, see RAM and STS.

1.9. Does the amount of data affect the query speed?

No. The performance of single-row query and range query is not affected by the amount of data.

Tablestore is a NoSQL database service. The size of data stored in Tablestore tables can be linearly scaled up to the cluster scale without affecting the speed of single-row query and range query. The query speed is not affected even if the count of rows reaches several hundred million or tens of billions.

The time to query a high-performance instance (SSD-based) is within several milliseconds. If the size of a single row is small, the time for the query is within 10 ms.

For more information about query-related operations, see GetRow, GetRange, and BatchGetRow.

1.10. How do I view the total count of rows in a table?

You can obtain the total count of rows in a table by calling the GetRange operation, using search index, or using Data Lake Analytics (DLA).

- Call the GetRange operation

Call the GetRange operation to calculate the count of rows in a table. High-concurrency levels and poor performance may result in latencies when you call this operation.

- Use search index

  > **Note** To use search index, you must CreateSearchIndex.

  - Call MatchAllQuery of search index to return the total count of rows in a table within several milliseconds. For more information, see Match all query.
  - Call aggregation of search index to return the total count of rows in a table within several milliseconds. For more information, see Aggregation.

- Use DLA

  > **Note** DLA is a serverless, in-cloud interactive query and analysis service. You can call the standard JDBC operation to query and analyze data in Tablestore.

  You can execute SQL statements to collect statistics for and analyze data in Tablestore. This feature prevents mutual synchronization between multiple data sources and minimizes costs.

  After you enable DLA, you must create an instance for DLA and Tablestore, and map the instance to the table. You can use select count(*) from table to query the count of rows in the table.

  When you execute SQL statements, DLA starts multiple tasks to calculate the count of rows, which improves query efficiency. In typical scenarios, it takes about 10 seconds to count 10 million rows of data depending on the size of rows and design of a table.

1.11. What do I do if "OTSErrorMsg: Disallow read index table in building base state" is displayed?

Problem description

Similar information is displayed when I read data from an index table for which existing data is being read from the base table:

```
OTSErrorMsg: Disallow read index table in building base state
```

Cause

Existing data is read from the base table and synchronized to the index table. Data cannot be read from the index table before existing data has been synchronized. You can read data from the index table only after existing data is synchronized to the index table. The time used to synchronize existing data is related to the size of data in the base table.

1.12. Does Tablestore support queries based on the IN and BETWEEN operators?

Yes. You can implement queries similar to the IN operator-based queries based on the following conditions:

- If you can specify complete primary key columns, call the BatchGetRow operation. For more information, see BatchGetRow.
- If you cannot specify the complete primary key columns or attribute columns, use terms query of search index. For more information, see TermsQuery.

You can implement queries similar to the BETWEEN operator-based queries based on the following conditions:

- If you can specify complete primary key columns, call the GetRow operation. For more information, see GetRange.
- If you cannot specify the complete primary key columns or attribute columns, use range query of search index. For more information, see Range query.

1.13. How do I view the size of data stored in a table?

This topic describes how to view the size of data stored in a table.

Procedure

- Log on to the Tablestore console. Go to the **Details** tab of a table. View the size of data stored in the table in the Description section.

- Log on to the Tablestore console. Go to the **Monitoring Indicators** tab of a table. In the Catergories section, click **Table Size** to view the size of data stored in the table.

>

1.14. How do I delete all data from a table?

You can delete all data from a table by using one of the following methods: 1. Delete the table and create a table that has the same name. 2. Obtain information about the primary key of the table. Then, delete the data in the table.

- Delete a table and create a table of the same schema

  To delete a table, you can use Tablestore SDKs to call the DeleteTable operation. You can also delete a table in the Tablestore console. After you delete the table, you must create a table that has the same schema.

  > **Note**
  > - If the table has indexes, you must delete the indexes before you delete the table.
  > - After the indexes of the table are deleted, you must create indexes for the created table if you want to use indexes.

- Obtain the primary keys of all rows in the table to delete data from the table

Obtain the primary keys of all rows in a table by using Tablestore SDK to call the GetRange operation. And then call the BatchWriteRow or DeleteRow operation to delete all data from the table.

1.15. How do I increase the limit on the rows of data returned by calling the Search operation to 1,000?

This topic describes how to increase the limit on the rows of data returned by calling the Search operation to 1,000.

To increase the number of results returned by calling the Search operation in a query, if only data in search indexes is queried, the limit is automatically increased to 1,000. If data in tables must be queried, the limit is 100.

Procedure

You can use the following operations to increase the limit on the rows of data returned by calling the Search operation to 1,000:

1. When you create a search index, you must set the value of store to true for the specified column.
   - By default, the value of store is true when you create a search index in the console. You do not need to set this parameter.
   - When you use SDKs to create a search index, you can set the store parameter in the FieldSchema parameter to true for the specified column.
2. When you call the Search operation to query data, you must set the ColumnsToGet parameter of SearchRequest.

   The ColumnsToGet parameter returns only columns for which the store parameter is set. Columns that of ARRAY, GEOPOINT, or NEST are excluded. The limit is automatically increased to 1,000.

   > **Note** If the ColumnsToGet parameter includes the columns that of ARRAY, GEOPOINT, or NEST, the table is still queried when you call the Search operation to query data. The limit is 100.

Examples

In this example, Tablestore SDK for Java is used to describe how to set the ColumnsToGet parameter. The configuration method of ColumnsToGet is similar to that of SDKs in other languages. You need to modify only the ColumnsToGet parameter in SearchRequest.

```
SearchQuery searchQuery = new SearchQuery();
searchQuery.setQuery(new MatchQuery());
searchQuery.setLimit(1000);

SearchRequest searchRequest = new SearchRequest(tableName, indexName, searchQuery);
ColumnsToGet columnsToGet = new ColumnsToGet();
columnsToGet.setReturnAll(false);
columnsToGet.setColumns(Arrays.asList("field_1", "field_2", "field_3")); // Set the name of the returned column. The data type of the returned column cannot be ARRAY, GEOPOINT, or NEST. Otherwise, the table is queried.
searchRequest.setColumnsToGet(columnsToGet);
SearchResponse response = client.search(searchRequest);

// In Tablestore SDK for Java V5.6.1 and later, you can set the returnAllColumnsFromIndex parameter in ColumnsToGet to query the attribute column in all search indexes.

ColumnsToGet columnsToGet = new ColumnsToGet();
columnsToGet.setReturnAllFromIndex(true);
searchRequest.setColumnsToGet(columnsToGet);
SearchResponse response = client.search(searchRequest);
```

1.16. Can I query joined Tablestore tables?

No. Tablestore does not support queries based on joined tables. You can combine Tablestore with Data Lake Analytics (DLA) and use standard SQL statements that are compatible with most SQL syntax of MySQL 5.7 to analyze data in Tablestore. For more information, see DLA-based analysis of data in Tablestore by using SQL statements.

2.FAQ about API/SDK

2.1. What do I do if SocketTimeoutException is reported when I use Tablestore SDK for Java?

Problem description

When the difference between the time Tablestore SDK for Java receives data and the time Tablestore SDK for Java sends data exceeds the socket timeout value, Tablestore SDK for Java returns SocketTimeoutException. The socket timeout period includes the period for network transmission between the application and server, which is calculated from the time the application sends a request to when server processes the request and when the application receives the response. You can customize the socket timeout value when you create an OTSClient instance. By default, the socket timeout value is 15s.

Solution

Solutions are provided based on the following possible causes:

- Network connection failure

  If SocketTimeoutException is returned for all requests, the possible cause is network connection failures. You can run the ping or curl command to test the connectivity of the network.

  ```
  ping aaaa.cn-hangzhou.ots.aliyuncs.com
  curl aaaa.cn-hangzhou.ots.aliyuncs.com
  ```

  In most cases, the curl command returns a similar output:

  ```
  <? xml version="1.0" encoding="UTF-8"? >
  <Error><Code>OTSUnsupportOperation</Code><Message>Unsupported operation: ". </Message><R
  equestID>00054ec5-822c-8964-adaf-990a07a4d0c9</RequestID><HostID>MTAuMTUzLjE3NS4xNzM=</
  HostID></Error>
  ```

  If the connection fails, a possible cause is that an internal endpoint is used when the request is not sent by an ECS instance.

- Server-side latency that exceeds the socket timeout value specified in Tablestore SDK for Java

  The period for the Tablestore server to process a request does not exceed 15 seconds because the default maximum overtime period for the server is about 10 seconds. If the actual period exceeds 10 seconds, OTSTimeout is returned to the client.

  However, if a socket timeout value such as 2 is specified in Tablestore SDK for Java, Tablestore SDK for Java returns SocketTimeoutException.

- Network transmission latency

  SocketTimeoutException may also be returned if the overall latency is extended by network transmission instead of server-side processing. Check whether traffic and bandwidth usage as well as the packet retransmission rate are high.

- Frequent full GC

  SocketTimeoutException may be returned when the program runs under heavy load due to frequent full GC. The possible cause is that when full GC occurs, the request cannot be sent or the response cannot be received. SocketTimeoutException is returned when the actual timeout period exceeds the socket timeout value specified in Tablestore SDK for Java.

  If this error occurs, you must use a tool to analyze the GC status of the process to resolve frequent GC.

2.2. FAQ about the Tablestore SDK for Java logging framework

Which logging framework does Tablestore SDK for Java use?

Tablestore SDK for Java uses Simple Logging Facade for Java (SLF4J) that depends on **Log4j 2** as the dependency to implement the logging framework.

How can I replace a logging framework?

You can remove the declaration of the dependency on **Log4j 2** from the dependency of Tablestore SDK for Java. SLF4J automatically searches your applications for a logging framework to use the framework as the dependency.

```
<dependency>
 <groupId>com.aliyun.openservices</groupId>
 <artifactId>ots-public</artifactId>
 <version>2.2.4</version>
 <exclusions>
       <exclusion>
          <groupId>org.apache.logging.log4j</groupId>
          <artifactId>log4j-api</artifactId>
       </exclusion>
       <exclusion>
          <groupId>org.apache.logging.log4j</groupId>
          <artifactId>log4j-core</artifactId>
       </exclusion>
       <exclusion>
          <groupId>org.apache.logging.log4j</groupId>
          <artifactId>log4j-slf4j-impl</artifactId>
       </exclusion>
 </exclusions>
</dependency>
```

2.3. What can I do if errors related to data types supported by primary key columns occur?

Problem description

```
Caused by: [ErrorCode]:OTSInvalidPK, [Message]:Validate PK type fail. Input: VT_STRING, Meta: VT_BLOB. [RequestId]:00055f43-3d31-012b-62c3-980a3eefe39e, [TraceId]:02822839-3b5b-af35-409a-cf68841239fa, [HttpStatus:]400
```

Cause

When the table is created, the data type supported by the primary key columns is BINARY. However, the data type of your data is STRING.

Solution

Ensure that your data is of the correct type.

2.4. What do I do if the Validate PK size fail exception occurs when I use SDKs?

This topic describes the problem description, cause, and solution of the Validate PK size fail exception when you use SDKs.

Problem description

The following exception occurs when you use SDKs to update data:

```
Caused by: [ErrorCode]:OTSInvalidPK, [Message]:Validate PK size fail
```

Cause

The number of primary keys configured is inconsistent with the actual number of primary keys of the table.

Solution

The number of primary keys configured must be consistent with the actual number of primary keys of the table.

2.5. What do I do if "java.lang.IllegalStateException: Request cannot be executed; I/O reactor status: STOPPED" is displayed when I use Tablestore SDK for Java?

Problem description

A similar output is displayed when I use Tablestore SDK for Java:

```
java.lang.IllegalStateException: Request cannot be executed; I/O reactor status: STOPPED
```

Cause

shutDown is called to shut down the OTSClient instance and the internal I/O reactor. If the OTSClient instance is called to read and write data, the error occurs.

Solution

Check whether the OTSClient instance is shut down.

2.6. What do I do if Invalid date format is reported when I use Tablestore SDK for Java?

Problem description

Runtime environment: JDK 8

A similar output is displayed when I use Tablestore SDK for Java:

```
[Error Code]:OTSParameterInvalid, [Message]:Invalid date format: Wed, 18 May 2016 08:32:51 +00:00.
```

Cause

The version of Joda-Time on which Classpath depends is earlier than the required version. Similar errors occur when you run Joda-Time of an early version in JDK 8.

Solution

Update the ots-public version to 2.2.4. If your project also depends on the Joda-Time library, update the version of ots-public to 2.9.

2.7. What do I do if the "java.lang.UnsupportedOperationException: This is supposed to be overridden by subclassed" exception occurs when I use Tablestore SDK for Java?

This topic describes the problem description, cause, and solution of the "java.lang.UnsupportedOperationException: This is supposed to be overridden by subclassed" exception when you use Tablestore SDK for Java.

Problem description

A similar output is displayed when I use Tablestore SDK for Java:

```
Caused by: java.lang.UnsupportedOperationException: This is supposed to be overridden by subclasse
d
```

Cause

Tablestore SDK V2.4.1 depends on Protobuf library 2.4.1 and HttpAsyncClient 4.0.2, which may conflict with the same built-in library of your application.

Solution

Add the following dependencies to the pom.xml file in the Maven project:

> **Note** classifier is jar-with-dependencies, which packages the HttpClient and Protobuf dependencies by using rename package to remove the dependency on HttpClient and Protobuf.

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>tablestore</artifactId>
    <version>Your current version</version>
    <classifier>jar-with-dependencies</classifier>
    <exclusions>
      <exclusion>
        <groupId>com.google.protobuf</groupId>
        <artifactId>protobuf-java</artifactId>
      </exclusion>
      <exclusion>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpasyncclient</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
```

2.8. What do I do if OTSUnsupportOperation is reported when I use Tablestore SDK?

Problem description

A similar output is displayed when I call `syncClient.createTable(request)`:

```
Caused by: [ErrorCode]:OTSUnsupportOperation, [Message]:Unsupported operation: 'CreateTable'.
```

Cause

Tablestore SDK later than V4.0.0 is used to access tables created by using Tablestore SDK V2.x.x or earlier.

Solution

Use Tablestore SDK V2.x.x.

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>ots-public</artifactId>
    <version>2.2.5</version>
</dependency>
```

2.9. What do I do if OTSParameterInvalid is reported when I use BatchWriteRow to submit 100 data entries at a time?

Problem description

A similar output is displayed when I use BatchWriteRow to submit 100 data entries at a time:

```
ErrorCode: OTSParameterInvalid, ErrorMessage: The input parameter is invalid.
```

Cause

The possible cause is that one batch operation cannot be repeatedly performed on the same row. If the operation is repeatedly performed on the same row, an error occurs.

Solution

Change 100 to 1 to submit one data entry each time. Keep other code unchanged.

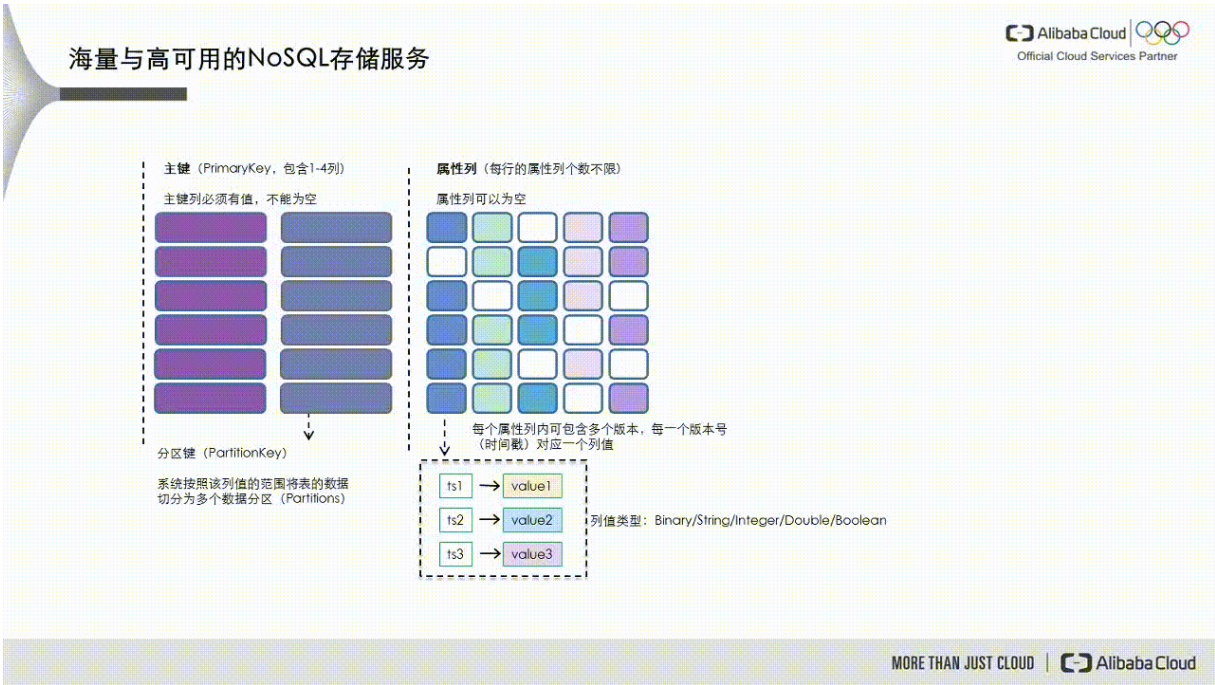
2.10. Why do 5xx-related errors occur when I use Tablestore?

Some users may encounter 5xx-related errors when they use Tablestore. The following table lists the HTTP status codes.

HTTP status code	Error code	Error message
503	OTSPartitionUnavailable	The partition is not available.
503	OTSServerUnavailable	Server is not available.
503	OTSServerBusy	Server is busy.
503	OTSTimeout	Operation timeout.

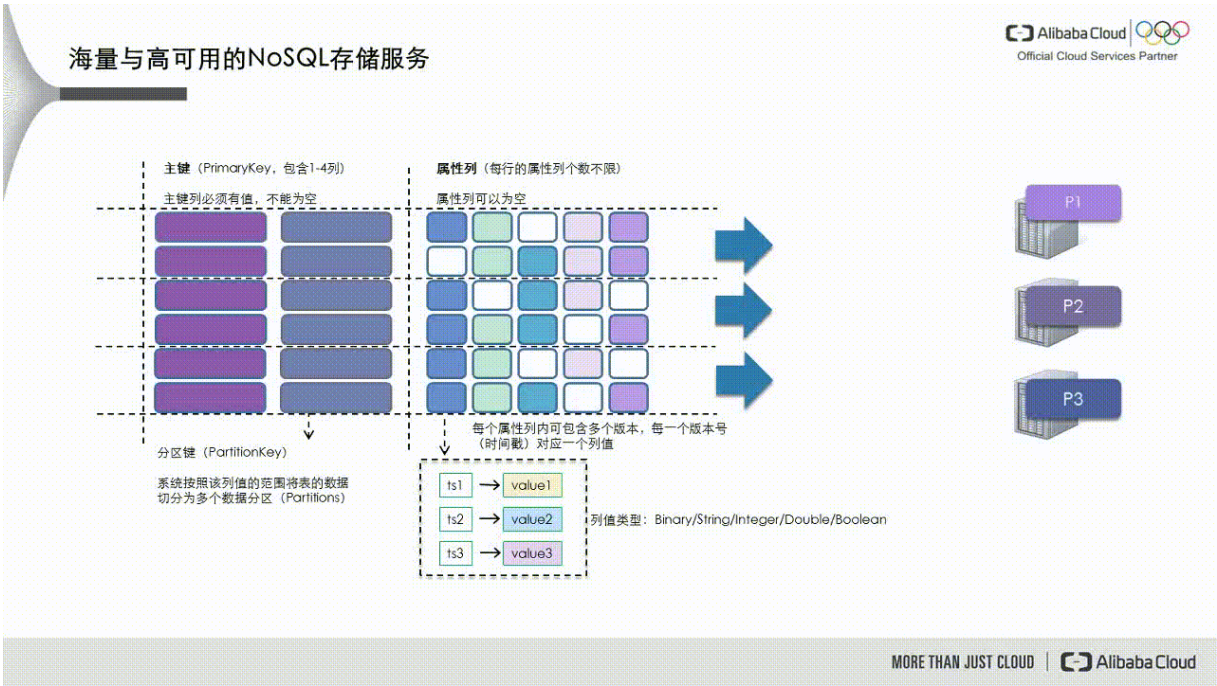
The possible cause is that Tablestore is a distributed NoSQL service. The server implements load balancing based on the amount of and access to data in each partition. This way, data and access can be seamlessly scaled and distributed to multiple servers.

The following figure shows that Tablestore distributes data to different partitions base on the order of the first primary key column. Different partitions are distributed to different racks across servers to provide data reading and writing services.



The dynamic load balancing mechanism of Tablestore can detect whether a partition such as P1 in the following figure stores large amounts of data or processes too frequent access. In this case, the partition is split into two partitions P1 and P5. The two partitions are distributed to the node that supports lower loads.

Tablestore uses the preceding automatic load balancing mechanism to store large amounts of data, process highly concurrent access, and implements automatic scalability without human intervention during the whole process. Only one partition is created when a table is created, which provides limited concurrent read and write capabilities. The automatic load balancing mechanism delivers latencies. To address these issues, contact Alibaba Cloud engineers to split a table into multiple partitions in advance.



Tablestore uses a shared storage mechanism. Partitions are the logical storage unit of Tablestore. When Tablestore implements load balancing, data is not migrated and only metadata of tables is modified. When the metadata changes, partitions may become unavailable for a short period of time to ensure data consistency. **In most cases, this period ranges from several hundred milliseconds to several seconds when the partitions are under heavy load.** The preceding error may occur when read and write operations are performed on the partitions during this period of time. If this error occurs, retry the operations. Tablestore SDK provides default retry policies. You can specify retry policies when you initialize an OTSClient instance.

Tablestore uses the API that complies with the standard Restful protocol. Due to the uncontrollability of the network environment, we recommend that you add a retry policy for all reading and writing operations to respond to network errors for fault tolerance.

> **Note** Data read by calling BatchWriteRow and BatchGetRow may be distributed to a thousand tables or multiple partitions of a table. A partition may be being split when operations are called on the table. Therefore, the operations as a whole is not atomic. To ensure that each single-row operation is atomic, check getFailedRows() in the response for failed single-row operations when HTTP status code 200 is returned.

2.11. How can I obtain multiple rows of data if only one primary key column is set?

For more information about how to use GetRange to query multiple rows of data, see GetRange. For more information about specific code examples, visit GitHub.

2.12. How can I paginate query results?

Tablestore is a distributed storage system. You can use multiple methods to paginate query results. This topic describes how to paginate query results in detail.

Paginate query results for tables

If you create only Tablestore tables without creating search indexes, you can use one of the following methods to paginate query results:

- Use next tokens. The response to each request of the GetRange operation contains a next token. The next token can be used in the next request to read remaining data by page.
- Use GetRangeIterator. Use the iterator.next() method to obtain next piece of data.
- offset is not supported to paginate query results.
- The total number of rows and the total number of pages cannot be obtained.

Paginate query results for search indexes

If you create a search index for a table, use the following method to paginate query results:

- Use offset and limit. The sum of the offset and limit values cannot exceed 10000. If the sum of the offset and limit values exceed 10000, use next tokens to paginate query results.
- Use next tokens. The response to each request of the GetRange operation contains a next token. The next token can be used in the next request to read remaining data by page.
- Use SearchIterator. Use the iterator.next() method to obtain next piece of data.
- You can obtain the total number of rows and the total number of pages (Total number of pages = Total number of rows/Value of limit). To obtain the total number of rows, set getTotalCount in the request to true. If getTotalCount is set to true, more resources are consumed and performance may deteriorate.

Examples for tables

The following code provides an example on how to call an operation to paginate query results by setting offset and limit.

```
/**
    * Specify the range of data to query and return specified rows of data by setting the page size and o
ffset.
    */
   private static Pair<List<Row>, RowPrimaryKey> readByPage(OTSClient client, String tableName,
         RowPrimaryKey startKey, RowPrimaryKey endKey, int offset, int pageSize) {
      Preconditions.checkArgument(offset >= 0, "Offset should not be negative.") ;
      Preconditions.checkArgument(pageSize > 0, "Page size should be greater than 0.") ;
      List<Row> rows = new ArrayList<Row>(pageSize);
      int limit = pageSize;
      int skip = offset;
      RowPrimaryKey nextStart = startKey;
      // If the amount of data to query is large, only part of data is returned for one request. To query al
l required data, use multiple range query requests.
      while (limit > 0 && nextStart ! = null) {
         // Create query parameters for GetRange.
         // Note that startPrimaryKey must be set to the position where last reading stops. This way, you
can send multiple range query requests to read remaining data.
```

```
        RangeRowQueryCriteria criteria = new RangeRowQueryCriteria(tableName);
        criteria.setInclusiveStartPrimaryKey(nextStart);
        criteria.setExclusiveEndPrimaryKey(endKey);
        // Set an appropriate limit value. limit specifies the total number of rows to return (a maximum o
f all rows of data on a page) from the offset value.
        criteria.setLimit(skip + limit);
        GetRangeRequest request = new GetRangeRequest();
        request.setRangeRowQueryCriteria(criteria);
        GetRangeResult response = client.getRange(request);
        for (Row row : response.getRows()) {
          if (skip > 0) {
            skip--; // The number of rows of data to filter before the offset value. The data is filtered o
n the client after the data is read.
          } else {
            rows.add(row);
            limit--;
          }
        }
        // Set the position from which to read data next time.
        nextStart = response.getNextStartPrimaryKey();
      }
      return new Pair<List<Row>, RowPrimaryKey>(rows, nextStart);
    }
```

The following code provides an example on how to use the preceding operation to sequentially read all data within a specified range by page:

```
 private static void readByPage(OTSClient client, String tableName) {
     int pageSize = 8;
     int offset = 33;
     RowPrimaryKey startKey = new RowPrimaryKey();
     startKey.addPrimaryKeyColumn(COLUMN_GID_NAME, PrimaryKeyValue.INF_MIN);
     startKey.addPrimaryKeyColumn(COLUMN_UID_NAME, PrimaryKeyValue.INF_MIN);
     RowPrimaryKey endKey = new RowPrimaryKey();
     endKey.addPrimaryKeyColumn(COLUMN_GID_NAME, PrimaryKeyValue.INF_MAX);
     endKey.addPrimaryKeyColumn(COLUMN_UID_NAME, PrimaryKeyValue.INF_MAX);
     // Read the first page from the 33th line.
     Pair<List<Row>, RowPrimaryKey> result = readByPage(client, tableName, startKey, endKey, offset,
pageSize);
     for (Row row : result.getKey()) {
        System.out.println(row.getColumns());
     }
     System.out.println("Total rows count: " + result.getKey().size());
     // Read all data within the range and paginate query results in sequence.
     startKey = result.getValue();
     while (startKey ! = null) {
        System.out.println("============= start read next page ==============");
        result = readByPage(client, tableName, startKey, endKey, 0, pageSize);
        for (Row row : result.getKey()) {
           System.out.println(row.getColumns());
        }
        startKey = result.getValue();
        System.out.println("Total rows count: " + result.getKey().size());
     }
  }
```

Examples for search indexes

For more information, see Sorting and paging.

2.13. How can I add one to the values in a specific column?

The following code provides an example on how to add one to the values in a specified column:

```
row = getRow(primary_key, 'col') // Read the values in the column.
old_value = row['col'] // Record the original values in the column.
row['col'] = old_value + 1 // Calculate new values.
updateRow(row, condition: row['col'] == old_value) // Use conditional update when you write new value
s. Condition: The data is written when the original values remain unchanged.
```

2.14. What are the code examples used to perform ListTable in Tablestore SDK for Python?

Examples

```
import time

import logging

import unittest

from ots2 import *

ENDPOINT = "https://xxx.cn-hangzhou.ots.aliyuncs.com";

ACCESSID = "xxx";

ACCESSKEY = "xxx";

INSTANCENAME = "xxx";

ots_client = OTSClient(ENDPOINT, ACCESSID, ACCESSKEY, INSTANCENAME)

list_response = ots_client.list_table()

print u'instance table:'

for table_name in list_response:

    print table_name
```

Note For more information about how to install and use Tablestore SDK for Python, see Preface in Tablestore SDK for Python.

Tablestore SDK for Python does not provide import at the beginning of each snippet for you to run code. If you do not add import to the beginning of each snippet when you run the code, a similar output is displayed:

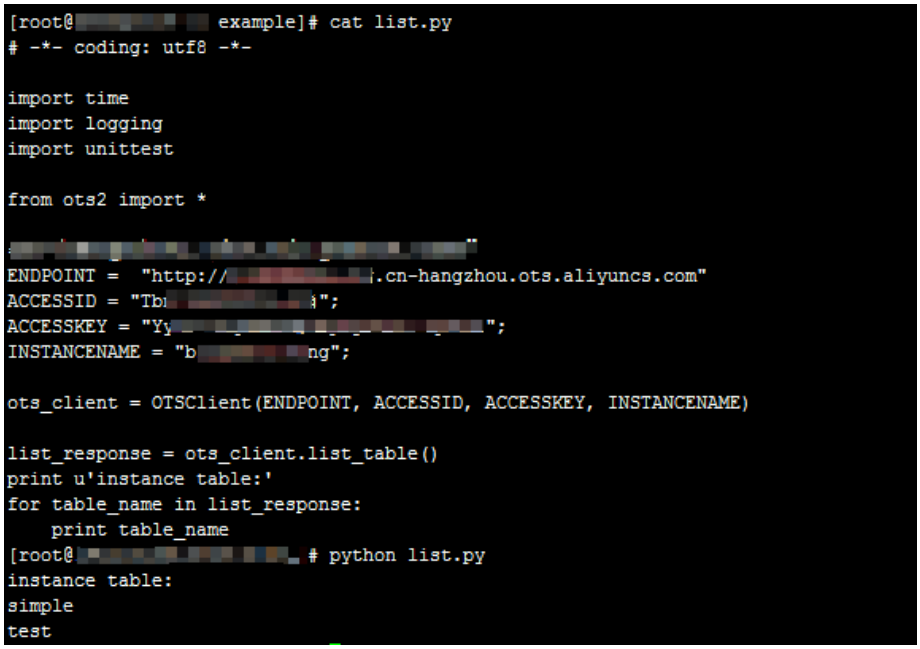
```
Traceback (most recent call last):

  File "listtable.py", line 6, in

    ots_client = OTSClient(ENDPOINT, ACCESSID, ACCESSKEY, INSTANCENAME)

NameError: name 'OTSClient' is not defined
```

Add import to run the code.



2.15. What do I do if Checksum mismatch is reported when I use Tablestore SDK for PHP?

This topic describes the problem description, cause, and solution of the Checksum mismatch exception when you use Tablestore SDK for PHP.

Problem description

The following exception occurs when you use Tablestore SDK for PHP by using PHP 5.6 in Windows:

```
Fatal error: Uncaught exception 'AliyunOTSOTSClientException' with message 'Checksum mismatch. exp
ected:120,actual:-48'
```

Cause

Tablestore uses 64-bit integers. However, PHP versions earlier than PHP 7 in Windows do not fully support 64-bit integers and can only process 64 bit integers as strings. Therefore, Tablestore SDK for PHP does not support PHP versions earlier than PHP 7 in Windows.

Solution

When you use Tablestore SDK for PHP in Windows, ensure that the PHP version is PHP 7 or later. We recommend that you use PHP 7 to obtain the optimal performance.

You can use phpinfo() to check the value of Architecture in PHP configuration information to determine whether your PHP version supports 64-bit integers.

- If the value of Architecture is X86, the current PHP version supports only 32-bit integers. You must upgrade your PHP version to PHP 7 or later.
- If the value of Architecture is X64, the current PHP version supports 64-bit integers and can meet your requirement.

2.16. What do I do if the "Request denied because this instance can only be accessed from the binded VPC" exception occurs when I use Tablestore SDK?

This topic describes the problem description, cause, and solution of the "Request denied because this instance can only be accessed from the binded VPC" exception when you use Tablestore SDK.

Problem description

The following exception occurs when you use SDKs to access an instance:

```
[ErrorCode]:OTSAuthFailed, [Message]:Request denied because this instance can only be accessed fro
m the binded VPC.
```

Cause

After you bind a VPC to a Tablestore instance, you can access the instance only from the address of the bound VPC.

Solution

After you bind a VPC to a Tablestore instance, you must use the address of the bound VPC to access the instance by using SDKs.

Log on to the Tablestore console. Go to the **Network Management** tab to obtain the address of the bound VPC.