ALIBABA CLOUD

阿里云

表格存储Tablestore

常见问题

文档版本：20220328

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.一般性问题

1.1. 表格存储和传统关系型数据库（例如MySQL、SQL Server）有什么区别？

表格存储是阿里云自研的多模型结构化数据存储，提供海量结构化数据存储以及快速的查询和分析服务，与传统关系型数据库（RDBMS，例如MySQL、SQL Server）在数据模型和技术实现上都有较大的区别。

表格存储和传统关系型数据库的主要区别如下：

- 相对于传统关系型数据库的软硬件部署和维护，使用表格存储只需开通服务并按照实际占用的资源（存储和读/写吞吐量）进行付费即可。
- 相对于传统关系型数据库的视图、索引、事务等功能，表格存储提供全局二级索引、多元索引、局部事务等功能，同时具有更好的规模扩展性，能够支持更大的数据规模（百TB级别）和并发访问（单表10万QPS）。
- 相对于传统关系型数据库的SQL语句支持，表格存储提供标准的HTTP RESTful API和多种语言SDK。
- 相对于传统关系型数据库严格的SCHEMA，表格存储的表是稀疏的，每一行可以有不同的列，可以动态增加或者减少属性列，创建表时无需为表的属性列定义严格的SCHEMA。

1.2. 表格存储建表注意事项

表格存储支持半结构化的表，即建表时只需要指定主键列（1至4列），不需要在创建表的时候指定属性列。

表格存储表中包含的属性列个数无限制，且每一行数据可以拥有不同数量不同类型的属性列。在应用程序写入数据时，表格存储需要应用程序指定数据所有列（主键列及属性列）的列名和列值。

如何理解建表时主键（Primary Key）的第一列为分区键（Partition Key）？

主键的第一列为分区键，可以理解为当表的数据量达到一个设定值时，表格存储会根据分区键列值的范围来进行分区的操作，通过分区来达到数据访问负载均衡的目的。

建表时，表内的数据默认拥有一个分区，即该表的所有数据在一个数据分区上。当表拥有多个分区时，每个分区所存储的数据对应的是该表分区键列值某个范围内所有的数据。所有的分区键列值范围是按照其列值自然序切分的，即按照Integer或String（主键列数据类型）的自然序切分。

除了会影响到数据访问的性能，数据的分区也会影响到您预留读/写吞吐量的使用率。当表拥有多个分区时，该表的预留读/写吞吐量会按照一定的比例预分配到各个分区上。

如何设定一个好的分区键？

建表时，分区键的选择是很重要的，会影响当表数据量很大时访问的性能。应用程序在选择分区键时，应该遵循以下基本原则：

- 不要使用拥有固定值或取值范围较小的属性，如客户性别（Male/Female）。
- 尽量避免使用按自然序排序后会有明显访问热点的属性，如在查询最新数据场景中使用时间戳（TimeStamp）作为分区键。
- 尽量使用按自然序排序后访问热点比较分散的属性，如UserID。

如果无法预估访问热点，该怎么做？

建议在写入分区键之前，根据应用程序的特点进行一次哈希（Hash）。如在写入一行数据时，将UserID通过简单的哈希算法生成一个哈希值，然后在哈希值后拼接UserID作为分区键的值存入表格存储的表。通过这种轻量级的操作可以有效地解决部分访问热点问题。但是需要特别注意的是，由于分区键的值是由哈希值和实际值拼接的，应用程序将无法使用分区键进行范围读取的操作（getRange）。

为什么一个账户下会有表个数的限制？

每个表格存储用户可以创建10个实例，每个实例可以创建64个表，即表格存储限制在一个账户下最多可以创建640个表。

放开表个数的需求一般有以下几种情况：

- 数据量大、访问性能要求高

  不同于传统的SQL数据库（例如MySQL） 解决海量数据访问需求的方法是分库分表，表格存储作为分布式实现方式很好地解决了数据量及访问延迟的瓶颈。

  您可以将结构化或半结构化的数据存在一张稀疏的大表中，不用担忧数据量过大后的访问的性能问题。

- 应用的快速增长

  除了数据本身及访问量的增长，您可能使用表格存储为您的客户（如第三方伙伴、供应商等）提供服务。以为供应商提供服务为例，您有了一套基于表格存储的解决方案后，每加一个供应商就部署一组表格存储的表。这样，表的个数很快达到上限。如果您不断提高表个数的上限，会造成运维成本的不可控，也增加了后续全局数据分析的难度。

  建议在使用表格存储时打破传统思想，使用大表的概念将同类型海量结构化及半结构化数据存在一张表上。

- 表格存储服务本身的考虑

  基于表格存储分布式的实现，表的个数也成为了表格存储本身的一个资源属性。可以理解为在表格存储集群规模一定的情况下，表的个数是有一个最大值的。当然，表格存储的扩展能力可以有效地解决表个数的限制，但从表格存储服务本身资源可控性角度来看，表格存储设定了单个账户表个数的限制。

如果您仍然需要提高一个账号下表个数的限制，请提交工单。

1.3. 表、列和实例的命名规范

了解表、列和实例的命名规范。

表和列名称

表和列的命名规范如下：

- 只能由英文字母、数字和下划线（_）组成。
- 大小写敏感。
- 首字符不能为数字。
- 名称长度为1~255个字符。

实例名称

实例名称在一个地域内必须唯一，不同地域内的实例名称可以相同。

实例的命名规范如下：

- 只能由英文字母、数字和短划线（-）组成。
- 大小写不敏感。

- 首字符必须为英文字母且末尾字符不能为短划线（-）。
- 名称长度为3 Byte~16 Byte。

1.4. 如何理解主键、数据分区和数据分区键

主键

表中的每一行由主键（PK）唯一确定。您在创建表的时候必须指定组成主键的列，这些列称为主键列。主键列必须有值。您必须确保主键列的值的组合能够唯一地确定一行。在后续使用的过程中，主键列的类型不能改变。

数据分区和分区键

表格存储会自动把表分成不同的数据分区，以达到对其存储数据的负载均衡。数据分区的划分粒度为主键的第一列，该列即为数据分区键。

拥有相同数据分区键的行必然在同一个数据分区中。表格存储能够保证对具有同一数据分区键的数据进行更改操作的一致性。

下图是一个电子邮件系统的邮件表的一部分。该表的主键和分区键信息如下。

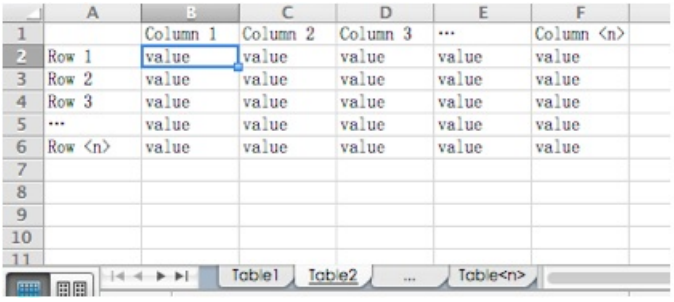
- 列UserID、ReceiveTime、FromAddr分别表示邮件用户的ID、接收时间、发送人，这些列为主键列，唯一确定一封邮件。其中UserID列为数据分区键。
- 列ToAddr、MailSize、Subject、Read分别表示收件人、邮件大小、邮件主题和邮件是否已读，这些为普通的列，存储邮件的相关信息。
- 图中表格存储把UserID为U0001和U0002的用户信息划在一个数据分区中，而把UserID为U0003和U0004的用户信息划分在另一个数据分区中。

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. 表格存储中表、行、列、值和电子表格的类比

表格存储中的表存储着用户的结构化数据。用户可以在表中查询、插入、修改和删除数据。一个用户可以拥有多个表。数据在表中以行、列、值的形式来组织。



上图展示了表格存储中表及其它概念与电子表格的类比：

- **表**：类似电子表格中底端的标签，不同的标签对应到不同的表。
- **行**：类似电子表格中的行。每一行由列和值组合。

- **列**：类似电子表格中的列。位于同一列的数据具有相同的类别属性。
- **值**：类似电子表格中的单元格。表示某一行在特定列上的值。与电子表格不同的是，表格存储允许某些列没有值，如果某些列没有值，则不占用存储空间。值的类型可以为STRING、INTEGER、BOOLEAN、DOUBLE、BINARY，如果该列为主键列，则值的类型只能是STRING、INTEGER或BINARY。

1.6. 用户验证

表格存储通过对称签名的方法来验证某个请求是其拥有者发送，以及应答是表格存储所发送。

当用户在阿里云注册账号之后，可以在AccessKey控制台上创建AccessKey。AccessKey ID用来标识用户，AccessKey Secret是对请求和响应进行签名和验证的密钥。AccessKeySecret需要保密。关于创建AccessKey的具体操作，请参见获取AccessKey。

用户以个人身份向表格存储发送请求时，需要包括请求明文、AccessKey ID、使用AccessKey Secret对请求明文中的信息部分签名产生的验证码。

表格存储收到请求后，会通过AccessKey ID找到相应的AccessKey Secret，以同样的方式签名明文中的信息。如果计算出来的验证码和提供的验证码相同，则认为是有效的用户发送的请求。

验证表格存储的应答时需要用户使用相同的方式进行计算，如果计算的验证码和提供的验证码一致，则用户可以认为应答是有效的表格存储应答。

1.7. 如何获取AccessKey

您可以为阿里云账号（主账号）和RAM用户创建一个访问密钥（AccessKey）。在调用阿里云API时您需要使用AccessKey完成身份验证。

背景信息

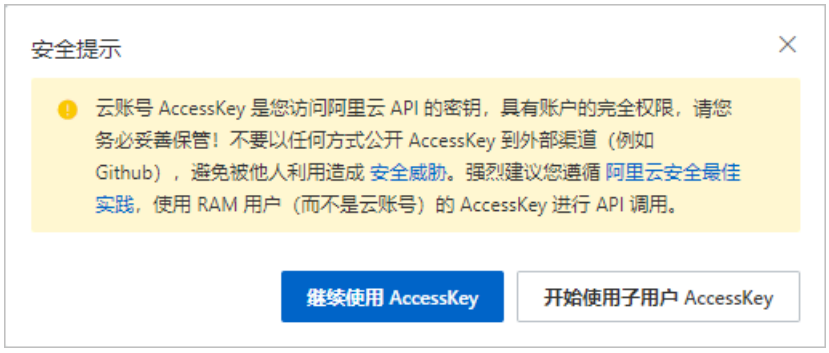
AccessKey包括AccessKey ID和AccessKey Secret。

- AccessKey ID：用于标识用户。
- AccessKey Secret：用于验证用户的密钥。AccessKey Secret必须保密。

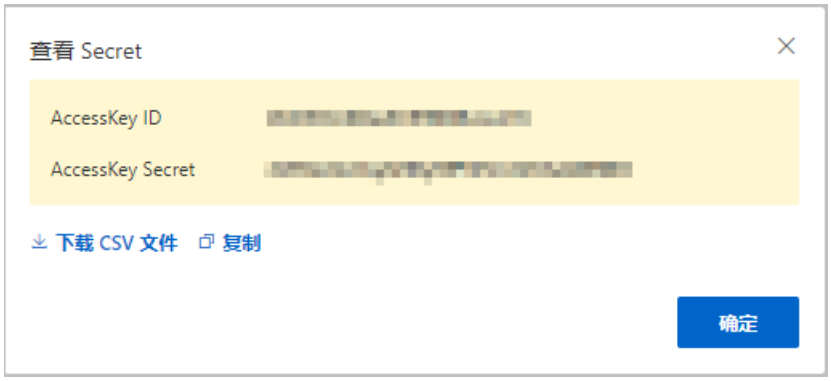
> **警告** 阿里云账号AccessKey泄露会威胁您所有资源的安全。建议您使用RAM用户AccessKey进行操作，可以有效降低AccessKey泄露的风险。

操作步骤

1. 使用阿里云账号登录控制台。
2. 将鼠标置于页面右上方的账号图标，单击**AccessKey管理**。
3. 在**安全提示**对话框，选择使用阿里云账号AccessKey或RAM用户AccessKey。



- 使用阿里云账号AccessKey
  a. 单击**继续使用AccessKey**。
  b. 在**AccessKey**页面，单击**创建AccessKey**。
  c. 获取验证码，单击**确定**。
  d. 在**查看Secret**对话框，查看AccessKey ID和AccessKey Secret。可以单击**下载CSV文件**，下载AccessKey信息。或者单击**复制**，复制AccessKey信息。



- 使用RAM用户AccessKey
  a. 单击**开始使用子用户AccessKey**。
  b. 系统自动跳转到RAM控制台的**用户**页面，找到需要获取AccessKey的RAM用户。

> **说明** 如果没有RAM用户，请先创建RAM用户，详情请参见创建RAM用户。

  c. 单击用户登录名称。
  d. 在**认证管理**页签下的**用户AccessKey**区域，单击**创建AccessKey**。
  e. 获取验证码，单击**确定**。

f. 在**查看Secret**页面，查看AccessKey ID和AccessKey Secret。可以单击**下载CSV文件**，下载AccessKey信息。或者单击**复制**，复制AccessKey信息。

> **说明**
>
> - RAM用户的AccessKey Secret只在创建时显示，不提供查询，请妥善保管。
> - 若AccessKey泄露或丢失，则需要创建新的AccessKey，最多允许为每个RAM用户创建2个AccessKey。

1.8. AccessKey权限控制

表格存储目前支持通过AccessKey ID和AccessKey Secret对用户身份进行认证，认证通过的用户对表格存储内的资源访问不受限。

一个用户的帐户下可以有多组不同的AccessKey ID和AccessKey Secret对，同一个帐户使用不同AccessKey ID和AccessKey Secret看到的表格存储资源是相同的。

如果您使用RAM用户，则RAM用户需要被主账号授予权限，且RAM用户只能访问被授权的资源。配置RAM用户权限的具体操作，请参见配置用户权限。

1.9. 存储数据量对查询速度有影响吗？

对于单行查询和范围查询，查询的速度不在于数据量有多少。

表格存储作为NoSQL数据库，其数据量可以随集群的规模线性扩展，并且对单行和范围查询的速度不会有任何影响。即使数据规模达到亿级或者百亿级，查询速度都不会变。

在高性能实例（底层是SSD）上，单行查询的速度是毫秒级别，如果单行数据量比较小，查询速度一般在10毫秒以内。

关于查询API的详细信息，请参考GetRow、GetRange和BatchGetRow。

1.10. 如何查看表的总行数

通过使用GetRange接口、多元索引或者数据湖分析的方法获取表的总行数。

- 使用GetRange接口

  使用GetRange接口对表中的行数进行计数，由于并发及性能较低，因此计数较慢。

- 使用多元索引

> **说明** 使用多元索引功能，需要先创建多元索引。

- 使用多元索引的全匹配查询接口，毫秒级返回表中总行数。
- 使用多元索引的统计聚合接口，毫秒级返回表中总行数。

- 使用数据湖分析

> **说明** 数据湖分析DLA（Data Lake Analytics）是无服务器化的云上交互式查询分析服务，可以通过标准JDBC直接对表格存储的数据进行查询和分析。

通过数据湖分析，您可以使用SQL语句对表格存储中的数据进行统计分析，减少多个数据源之间相互同步的问题并节省成本。

开通数据湖分析后，建立数据湖分析和表格存储的实例以及实例中数据表的映射，然后执行SQL语句select count(*) from table获取表的总行数，详情请参见使用DLA服务。

在执行SQL语句时，分析引擎会启动多个任务并发处理，所以计数较快。典型场景下（受行大小和表设计模式是否合理等影响），1000万行数据计数时间为10秒。

1.11. 报错OTSErrorMsg: Disallow read index table in building base state

问题现象

读正在构造存量数据的索引表时出现如下报错。

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

问题分析

全局二级索引的存量构造需要对表中的存量数据进行读取，然后同步到索引表。在存量数据同步完成之前，不允许读索引表，当存量数据同步完成，可以正常读取索引表。存量数据同步时间与主表的数据量大小相关。

1.12. 如何突破批量写200条的限制

您可以使用TableStoreWriter异步接口进行批量写入，详情参见使用TableStoreWriter进行高并发、高吞吐的数据写入

1.13. 表格存储是否支持类似关系数据库的in和between...and查询

当需要进行in查询时，请根据实际选择合适的方式。

- 方式一：使用多元索引的多词精确查询（TermsQuery）功能，支持任意列的多词精确查询，完全等价于MySQL的in查询，且性能更好。
- 方式二：使用批量读（BatchGetRow）功能，只支持完整主键的精确查询。

当需要进行between...and查询时，请根据实际选择合适的方式。

- 方式一：使用多元索引的范围查询（RangeQuery）功能，支持任意列的范围查询。

- 方式二：使用范围读（GetRange）功能，只支持完整主键或者主键前缀的范围查询。

1.14. 如何查看表的数据存储量

介绍如何查看表的数据存储量。

操作步骤

- 通过控制台在数据表的**基本详情**页签，可以查看表大小，即表的数据存储量。

- 通过控制台在数据表的**监控指标**页签，选择指标分组为**表大小**，可以查看表大小，即表的数据存储量。

>

1.15. 如何清空数据表中数据

通过删除数据表后重建表结构或者获取主键信息后删除数据均可清空数据表中数据。

- 删除数据表后重建表结构

  使用SDK调用DeleteTable接口或者通过控制台删除数据表。删除数据表后，需要重建表结构。

  > 说明
  > - 如果数据表上有索引，删除数据表前，必须删除数据表上的索引。
  > - 删除数据表上的索引后，如果仍需使用，则创建数据表后还需要重建索引结构。

- 获取主键信息后删除数据

  使用SDK调用GetRange接口获取主键信息，再调用BatchWriteRow或者DeleteRow接口删除数据。

1.16. 如何将多元索引Search接口查询数据的limit提高到1000

介绍将多元索引Search接口查询数据的limit提高到1000的方法。

为了提高使用多元索引Search接口单次查询的返回结果数，当查询数据时只查询多元索引中的数据没有反查数据表时，则limit限制自适应提高到1000，如果查询数据时需要反查数据表，则limit限制为100。

操作步骤

通过如下操作可以将多元索引Search接口查询数据的limit提高到1000。

1. 创建多元索引时，将指定列的附加存储设置为true。
   - 当通过控制台创建多元索引时，附加存储默认为true，无需设置。
   - 当通过SDK创建多元索引时，将指定列的“FieldSchema”参数中的“store”参数设置为true。
2. 通过多元索引Search接口查询数据时，设置SearchRequest的ColumnsToGet参数。

   ColumnsToGet参数中仅返回在多元索引中建立了索引的列，且不能包含“数组类型”、“geo地理位置类型”和“nested嵌套类型”的列，此时limit限制自适应提高到1000。

   > 说明 如果ColumnsToGet参数中包含“数组类型”、“geo地理位置类型”或“nested嵌套类型”的列，通过多元索引Search接口查询数据时仍会触发反查数据表，则limit限制为100。

示例

此处以Java SDK为例介绍如何设置ColumnsToGet参数，其他语言的SDK实现类似，只需修改SearchRequest中的ColumnsToGet参数即可。

```
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"));  //设置返回列的名称，列的数据类型不能为数组类型、geo地理位置类型和nested嵌套类型，否则会反查数据表。
searchRequest.setColumnsToGet(columnsToGet);
SearchResponse response = client.search(searchRequest);
//java-sdk-5.6.1及之后版本支持在ColumnsToGet中设置returnAllColumnsFromIndex参数，获取多元索引中的所有属性列。
ColumnsToGet columnsToGet = new ColumnsToGet();
columnsToGet.setReturnAllFromIndex(true);
searchRequest.setColumnsToGet(columnsToGet);
SearchResponse response = client.search(searchRequest);
```

1.17. 两表关联怎么查询

表格存储不支持两表关联查询，但您可以将表格存储连通DLA服务，然后使用通用的SQL语言（兼容MySQL5.7绝大部分查询语法）对表格存储进行灵活的数据分析。具体参见DLA文档。

1.18. 使用Tablestore（OTS）Reader同步全量数据时如何配置Split

当使用Tablestore（OTS）Reader同步全量数据过程中出现同步速度较慢问题时的现象、原因和解决方案。

现象

使用Tablestore（OTS）Reader同步全量数据过程中出现同步速度较慢问题。同步脚本的配置如下：

```
"reader": {
  "plugin": "ots",
  "parameter": {
    "datasource": "",
    "table": "",
    "column": [],
    "range": {
      "begin": [
        {
          "type": "INF_MIN"
        }
      ],
      "end": [
        {
          "type": "INF_MAX"
        }
      ]
    }
  }
}
```

原因

全量数据非常大且未在同步脚本中配置Split，此时同步任务为单并发拉取数据，会影响数据的同步速度。

解决方案

当全量数据非常大时，请在同步脚本中配置Split。具体操作如下：

1. 通过如下任意方式获取Split位点，请根据实际选择。
   - 使用SDK调用ComputeSplitPointsBySize接口。具体操作，请参见指定大小计算分片。

     返回示例如下：

     ```
     LowerBound:pkname1:INF_MIN, pkname2:INF_MIN
     UpperBound:pkname1:cbcf23c8cdf831261f5b3c052db3479e, pkname2:INF_MIN
     LowerBound:pkname1:cbcf23c8cdf831261f5b3c052db3479e, pkname2:INF_MIN
     UpperBound:pkname1:INF_MAX, pkname2:INF_MAX
     ```

   - 下载Tablestore CLI工具后，使用**points -s splitSize -t tablename**命令获取。具体操作，请参见命令行工具。

     > **说明** splitSize以100 MB为单位。当数据量较少时，无需配置Split位点；当数据量较多时，建议根据同步环境所能支持的最大并发度合理配置splitSize。

     返回示例如下：

     ```
     [
         {
             "LowerBound": {
                 "PrimaryKeys": [
                     {
                         "ColumnName": "pkname1",
                         "Value": null,
                         "PrimaryKeyOption": 2
                     },
                     {
                         "ColumnName": "pkname2",
                         "Value": null,
                         "PrimaryKeyOption": 2
                     }
                 ]
             },
             "UpperBound": {
                 "PrimaryKeys": [
                     {
                         "ColumnName": "pkname1",
                         "Value": "cbcf23c8cdf831261f5b3c052db3479e\u0000",
                         "PrimaryKeyOption": 0
                     },
                     {
                         "ColumnName": "pkname2",
                         "Value": null,
                         "PrimaryKeyOption": 2
                     }
                 ]
     ```

```
        },
        "Location": "80310717938EDF503FB1E26F70710391"
    },
    {
        "LowerBound": {
            "PrimaryKeys": [
                {
                    "ColumnName": "pkname1",
                    "Value": "cbcf23c8cdf831261f5b3c052db3479e\u0000",
                    "PrimaryKeyOption": 0
                },
                {
                    "ColumnName": "pkname2",
                    "Value": null,
                    "PrimaryKeyOption": 2
                }
            ]
        },
        "UpperBound": {
            "PrimaryKeys": [
                {
                    "ColumnName": "pkname1",
                    "Value": null,
                    "PrimaryKeyOption": 3
                },
                {
                    "ColumnName": "pkname2",
                    "Value": null,
                    "PrimaryKeyOption": 3
                }
            ]
        },
        "Location": "80310717938EDF503FB1E26F70710391"
    }
]
```

找到返回示例中第一列主键（即分区键）的Value，例如第一个LowerBound的pkname1 = null，第一个UpperBound的pkname1 = "cbcf23c8cdf831261f5b3c052db3479e\u0000"，第二个LowerBound的pkname1 = "cbcf23c8cdf831261f5b3c052db3479e\u0000"，第二个UpperBound的pkname1 = null，则全量同步脚本中的split配置如下：

```
"split" : [
   {
       "type":"STRING",
       "value":"cbcf23c8cdf831261f5b3c052db3479e\u0000"
   }
]
```

使用上述配置时，表格存储会将全量数据分为(INF_MIN,cbcf23c8cdf831261f5b3c052db3479e\u0000)和[cbcf23c8cdf831261f5b3c052db3479e\u0000,INF_MAX)两个区间进行数据并发拉取，提高同步速度。

2. 在同步脚本中配置Split位点。同步脚本示例如下：

```
"range": {
      "begin": [
        {
          "type": "INF_MIN"
        }
      ],
      "end": [
        {
          "type": "INF_MAX"
        }
      ],
      "split": [
        {
          "type": "STRING",
          "value": "splitPoint1"
        },
        {
          "type": "STRING",
          "value": "splitPoint2"
        },
        {
          "type": "STRING",
          "value": "splitPoint3"
        }
      ]
}
```

配置Split位点后，如果同步速度仍未提升，请提交工单或者加入钉钉群23307953（表格存储技术交流群-2）联系表格存储技术支持进行处理。

1.19. 表格存储支持跨地域数据迁移吗？

表格存储支持跨地域数据迁移，您可以使用DataWorks或者通道服务进行数据迁移。

当进行跨地域数据迁移时，请确保执行调度任务的机器与源实例以及目标实例的网络连通。您可以通过如下方式测试网络连通性。

- 当使用DataWorks迁移数据时，分别新建OTSReader和OTSWriter的数据源，测试数据源的网络连通性。
- 当使用通道服务迁移数据时，在连接通道的客户端机器上分别curl源实例的Endpoint和目标实例的Endpoint，测试网络连通性。如果测试结果中均返回Unsupported Operation信息，则说明网络连通。

更多信息，请参见迁移工具。

1.20. 如何批量删除数据

查询到待删除数据的主键信息后，您可以通过调用BatchWriteRow接口批量删除数据。

表格存储支持根据主键信息批量删除数据。具体步骤如下：

1. 根据实际选择合适的方式查询待删除数据的主键信息。
   - 如果要删除指定主键范围内的数据，请调用GetRange接口，查询指定主键范围内的数据，并获取待删除数据的主键信息。具体操作，请参见范围读（GetRange）。
   - 如果要删除满足指定条件的数据，请创建多元索引后，使用多元索引查询满足指定条件的数据，并获取待删除数据的主键信息。具体操作，请参见创建多元索引和使用SDK。

2. 调用BatchWriteRow接口，根据主键信息批量删除数据。具体操作，请参见批量写（BatchWriteRow）。

1.21. OTSStreamReader常见问题

本文介绍了OTSStreamReader运行时可能出现的问题，请根据实际问题排查处理错误。

OTSStreamReader运行时出现“Must set date or time range millis or time range string, please check your config”错误

- 问题现象

  OTSStreamReader运行时出现“Must set date or time range millis or time range string, please check your config”错误，如下图所示。

```
com.alibaba.datax.common.exception.DataXException: [ code:OTSStreamReaderError, messageOTS Stream Reader Error] - com.alibaba.datax.plugin.reader
.otsstreamreader.internal.OTSStreamReaderException: Must set date or time range millis or time range string, please check your config. - com.alib
aba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReaderException: Must set date or time range millis or time range string, please check
your config.
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.config.OTSStreamReaderConfig.load(OTSStreamReaderConfig.java:201)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReader$Job.init(OTSStreamReader.java:30)
        at com.alibaba.datax.core.job.JobContainer.initJobReader(JobContainer.java:1085)
        at com.alibaba.datax.core.job.JobContainer.init(JobContainer.java:497)
        at com.alibaba.datax.core.job.JobContainer.start(JobContainer.java:220)
        at com.alibaba.datax.core.Engine.start(Engine.java:100)
```

- 可能原因

  OTSStreamReader配置脚本中缺少增量数据时间范围的配置。

- 解决方案

  请通过以下方式配置增量数据范围。

  - 通过parameter.date参数配置导出数据的日期。
  - 通过parameter.startTimestampMillis和parameter.endTimestampMillis参数分别配置开始导出的时间点和结束导出的时间点。

  关于增量数据范围配置的更多信息，请参见增量同步（脚本模式）。

OTSStreamReader运行时出现“The stream of data table is not enabled”错误

- 问题现象

  OTSStreamReader运行时出现“The stream of data table is not enabled”错误，如下图所示。

```
[ code:OTSStreamReaderError, messageOTS Stream Reader Error] - com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReaderException:
 The stream of data table is not enabled. - com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReaderException: The stream of data
 table is not enabled.
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.core.OTSStreamReaderChecker.checkStreamEnabledAndTimeRangeOK(OTSStreamReaderC
hecker.java:49)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReaderMasterProxy.init(OTSStreamReaderMasterProxy.java:38)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReader$Job.init(OTSStreamReader.java:31)
        at com.alibaba.datax.core.job.JobContainer.initJobReader(JobContainer.java:1085)
        at com.alibaba.datax.core.job.JobContainer.init(JobContainer.java:497)
        at com.alibaba.datax.core.job.JobContainer.start(JobContainer.java:220)
        at com.alibaba.datax.core.Engine.start(Engine.java:100)
        at com.alibaba.datax.core.Engine.entry(Engine.java:321)
        at com.alibaba.datax.core.Engine.main(Engine.java:354)
```

- 可能原因

  OTSStreamReader中配置的表格存储数据表（paramter.table）未开启Stream。

- 解决方案

  请通过以下方式为OTSStreamReader中配置的表格存储数据表开启Stream。

- 登录表格存储控制台后，在目标数据表的**表管理**页面中**实时消费通道**页签，开启Stream。

- 通过不同SDK调用UpdateTable接口修改数据表属性。更多信息，请参见UpdateTable。

OTSStreamReader运行时出现“As expiration time is xx, so the start timestamp must greater than xx”错误

- 问题现象

  OTSStreamReader运行时出现“As expiration time is xx, so the start timestamp must greater than xx”错误，如下图所示。

```
com.alibaba.datax.common.exception.DataXException: [ code:OTSStreamReaderError, messageOTS Stream Reader Error] - com.alibaba.datax.plugin.reader
.otsstreamreader.internal.OTSStreamReaderException: As expiration time is 25200000, so the start timestamp must greater than 2021-09-25T20:09Z(16
32600562281) - com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReaderException: As expiration time is 25200000, so the start ti
mestamp must greater than 2021-09-25T20:09Z(1632600562281)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.core.OTSStreamReaderChecker.checkStreamEnabledAndTimeRangeOK(OTSStreamReaderC
hecker.java:60)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReaderMasterProxy.init(OTSStreamReaderMasterProxy.java:38)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReader$Job.init(OTSStreamReader.java:31)
        at com.alibaba.datax.core.job.JobContainer.initJobReader(JobContainer.java:1085)
        at com.alibaba.datax.core.job.JobContainer.init(JobContainer.java:497)
        at com.alibaba.datax.core.job.JobContainer.start(JobContainer.java:220)
        at com.alibaba.datax.core.Engine.start(Engine.java:100)
        at com.alibaba.datax.core.Engine.entry(Engine.java:321)
```

- 可能原因

  OTSStreamReader读取的增量日志存在过期时长，startTime参数配置错误。

- 解决方案

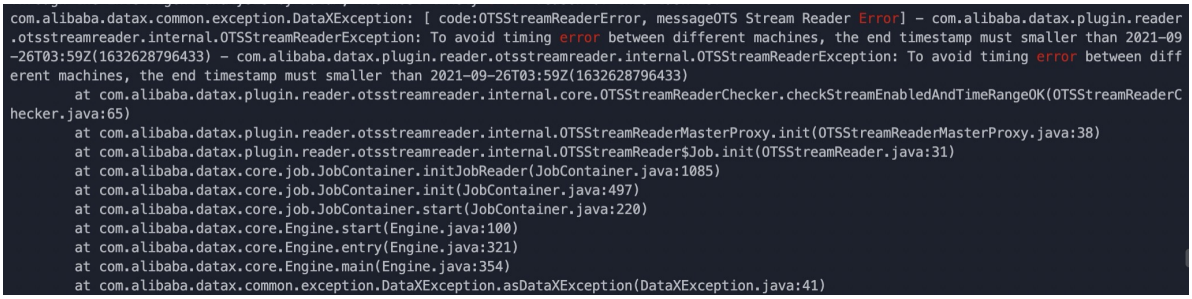
  同步脚本中配置的startTime必须大于 `任务启动时间-日志过期时长+10分钟` 。

  请登录表格存储控制台后，在目标数据表的**表管理**页面中**基本详情**页签，查看日志过期时长。

OTSStreamReader运行时出现“To avoid timing error between different machines, the end timestamp must smaller than xx”错误

- 问题现象

OTSStreamReader运行时出现“To avoid timing error between different machines, the end timestamp must smaller than xx”错误，如下图所示。



- 可能原因

  endTime参数配置错误。

- 解决方案

  同步脚本中配置的endTime必须小于 `任务启动时间 - 5分钟` 。

OTSStreamReader运行时出现“配置中的源表的列个数和目的端表不一致，源表中您配置的列数是:xx大于目的端的列数是:xx ”错误

- 问题现象

  OTSStreamReader运行时出现“配置中的源表的列个数和目的端表不一致，源表中您配置的列数是:xx 大于目的端的列数是:xx ”错误，如下图所示。

- 可能原因
  - 当在同步脚本中未配置 `"mode": "single_version_and_update_only"` （即为行模式同步）时，OTSStreamReader中paramter.columns的配置会失效，实际会按照 `pk,colname,version,colvalue,optype` 结构来同步数据。

    例如表格存储数据表的结构为primary key: id,name，column name: col1,col2。如果Writer中配置为 `paramter.columns = { id,name,col1,col2 }` ，则实际同步的结构为 `schema = { id,name,colname,version,colvalue,optype }` 。此时会出现“源表中您配置的列数是：6 大于目的端的列数是：4”错误。
  - 当在同步脚本中配置了 `"mode": "single_version_and_update_only"` （即为列模式同步）时，请检查OTSStreamReader中的paramter.columns和Writer中的paramter.columns配置列个数是否不一致。

- 解决方案

  在OTSStreamReader的paramter中增加 `"mode": "single_version_and_update_only"` 配置，并确保OTSStreamReader中的paramter.columns和Writer中的paramter.columns配置列个数一致。

OTSStreamReader运行时出现“The item of column must be map object”错误

- 问题现象

  OTSStreamReader运行时出现“The item of column must be map object”错误，示例如下：

```
[ code:OTSStreamReaderError, messageOTS Stream Reader Error] - com.alibaba.datax.plugin.r
eader.otsstreamreader.internal.OTSStreamReaderException: Parse column fail, please check
your config. - com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReaderEx
ception: Parse column fail, please check your config.
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.config.OTSStreamReade
rConfig.parseConfigForSingleVersionAndUpdateOnlyMode(OTSStreamReaderConfig.java:178)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.config.OTSStreamReade
rConfig.load(OTSStreamReaderConfig.java:267)
        at com.alibaba.datax.plugin.reader.otsstreamreader.internal.OTSStreamReader$Job.i
nit(OTSStreamReader.java:30)
        at com.alibaba.datax.core.job.JobContainer.initJobReader(JobContainer.java:1083)
        at com.alibaba.datax.core.job.JobContainer.init(JobContainer.java:497)
        at com.alibaba.datax.core.job.JobContainer.start(JobContainer.java:220)
        at com.alibaba.datax.core.Engine.start(Engine.java:100)
        at com.alibaba.datax.core.Engine.entry(Engine.java:318)
        at com.alibaba.datax.core.Engine.main(Engine.java:351)
Caused by: java.lang.IllegalArgumentException: The item of column must be map object, ple
ase check your input.
```

- 可能原因

  OTSStreamReader中parameter.column配置的格式错误。

- 解决方案

  请确保parameter.column的配置正确。配置示例如下：

  - 错误示例

    ```
    "column":[
         "col1",
         "col2"
    ]
    ```

  - 正确示例

    ```
    "column":[
         {
              "name" : "col1"
         },
         {
               "name" : "col2"
         }
    ]
    ```

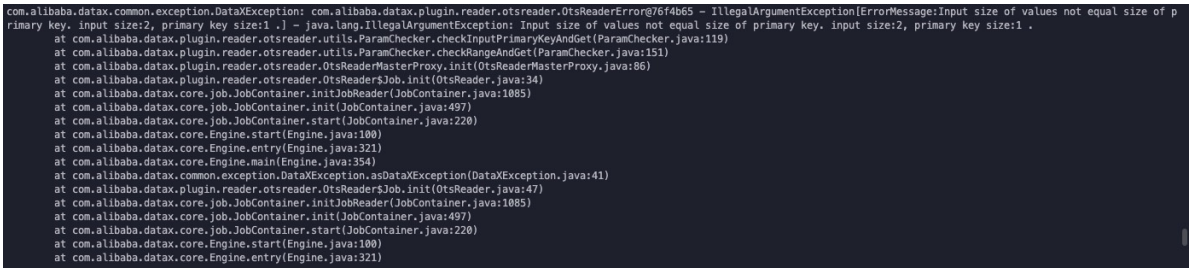
1.22. OTSReader常见问题

本文介绍了OTSReader运行时可能出现的问题，请根据实际问题排查处理错误。

OTSReader运行时出现“Input size of values not equal size of primary key. input size: xx, primary key size: xx”错误

- 问题现象

OTSReader运行时出现“Input size of values not equal size of primary key. input size: xx, primary key size: xx”错误，如下图所示。



- 可能原因

  OTSReader脚本中配置的主键个数与Tablestore数据表中的主键个数不一致。

- 解决方案

  首先查看读取的Tablestore数据表Schema，确认主键个数，然后确保OTSReader脚本中配置的主键个数与Tablestore数据表中的主键个数一致。

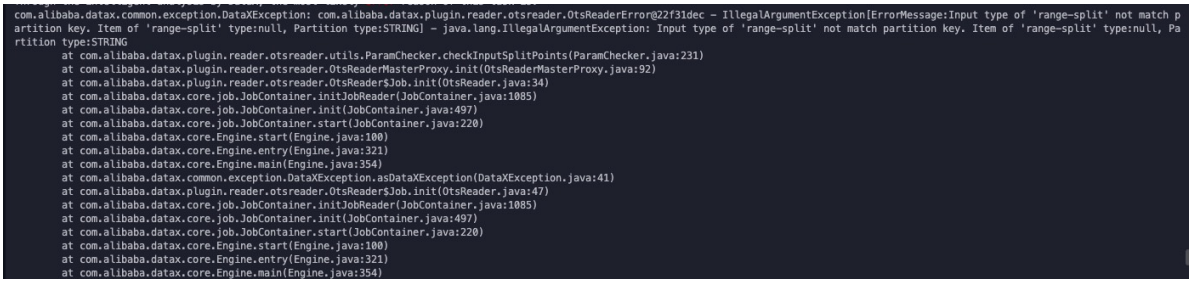
  假设Tablestore数据表有两个主键列，以全表同步为例，则OTSReader中应该配置的range范围如下：

```
"range": {
  "begin": [
    {
      "type": "INF_MIN"
    },
    {
      "type": "INF_MIN"
    }
  ],
  "end": [
    {
      "type": "INF_MAX"
    },
    {
      "type": "INF_MAX"
    }
  ]
}
```

OTSReader运行时出现“Input type of 'range-split' not match partition key. Item of 'range-split' type:xx, Partition type:xx”错误

- 问题现象

OTSReader运行时出现“Input type of 'range-split' not match partition key. Item of 'range-split' type:xx, Partition type:xx”错误，如下图所示。



- 可能原因

  OTSReader脚本中Split配置错误。不支持配置Split为如下格式JSON：

```
"split": [
  {
    "type": "INF_MIN"
  },
  {
  "type": "INF_MAX"
  }
]
```

- 解决方案
  - 删除OTSReader脚本中的Split配置。
  - 重新配置Split。具体操作，请参见使用Tablestore（OTS）Reader同步全量数据时如何配置Split。

OTSReader运行时出现“Invalid 'column', Can not parse Object to 'OTSColumn', item of list is not a map”错误

- 问题现象

  OTSReader运行时出现“Invalid 'column', Can not parse Object to 'OTSColumn', item of list is not a map”错误，如下图所示。

- 可能原因

  OTSReader脚本中的column配置错误。错误配置示例如下：

```
"column": [
  "id",
  "name",
  "age"
]
```

- 解决方案

请确保OTSReader脚本中的column配置正确。正确配置示例如下：

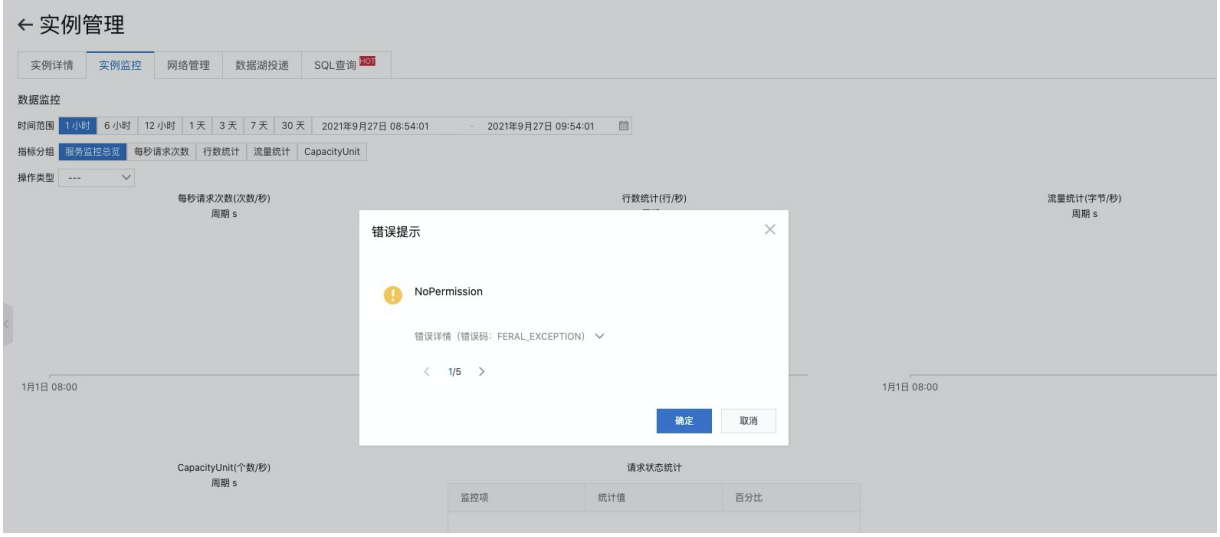
```
"column": [
  {
    "name": "id"
  },
  {
    "name": "name"
  },
  {
    "name": "age"
  }
]
```

1.23. 通过控制台查看监控数据时出现NoPermission错误

通过控制台查看监控数据时出现NoPermission错误，请为RAM用户授予云监控相关权限。

问题现象

使用RAM用户登录表格存储控制台查看实例监控或者表监控数据时，出现NoPermission错误，如下图所示。



可能原因

RAM用户无查看云监控的相关权限。

解决方案

使用阿里云账号登录访问控制控制台后，为RAM用户授予AliyunCloudMonitorFullAccess（管理云监控权限）或者AliyunCloudMonitorReadOnlyAccess（只读访问云监控权限）权限。具体操作，请参见为RAM用户授权。

1.24. 使用GetRange接口查询数据时返回指定范围外的数据

当使用GetRange接口查询数据时返回指定范围外的数据时，您可以通过创建二级索引进行查询。

问题现象

在控制台范围查询、SDK范围查询、OTS外部表查询、OTSReader数据同步等场景中，使用GetRange接口查询数据时，返回的结果与查询条件中设置的范围不一致。

可能原因

设置的查询条件不符合最左匹配原则。例如当第一主键列设置 `beginPrimaryKey = INF_MIN` 和 `endPrimaryKey = INF_MAX` ，第二主键列设置 `beginPrimaryKey = 10` 和 `endPrimaryKey = 10` ，则返回的结果为全表数据，而不是第二主键列等于10的行。

> **说明** 最左匹配原则表示最左优先，只有当最左主键的起止范围相同时，右侧主键设置的起止范围才有效；如果最左主键设置的起止范围不同时，右侧主键设置的范围将失效。此处最左主键和右侧主键指表结构中主键定义的先后顺序。更多信息，请参见GetRange范围查询详解。

解决方案

创建二级索引调整主键顺序后，请根据实际场景选择合适的查询方式。

- 如果是控制台范围查询、SDK范围查询场景，请直接使用二级索引进行范围查询。具体操作，请分别参见通过控制台读取数据和通过SDK读取数据。
- 如果是OTS外部表查询场景，请在外部表的建表SQL中配置表名称为二级索引名称。具体操作，请参见OTS外部表。
- 如果是OTSReader数据同步场景，请在数据同步脚本中配置表名称为二级索引名称。具体操作，请参见配置Tablestore（OTS） Reader。

1.25. 使用SDK调用API请求服务端时出现Wait future timeout错误

当出现Wait future timeout错误时，请确保客户端未出现FullGC并且未在请求期间调用client.shutdown()方法，以及排查服务端延迟是否过高。

问题现象

使用SDK调用API请求服务端时出现Wait future timeout错误。

可能原因

从发出请求到返回结果的耗时超过了syncClientWaitFutureTimeoutInMillis的值。
syncClientWaitFutureTimeoutInMillis的默认值为60000，单位为毫秒。

> **说明** 正常情况下，syncClientWaitFutureTimeoutInMillis的值不建议设置的过小。您可以在ClientConfiguration中设置syncClientWaitFutureTimeoutInMillis的值。

出现此问题的可能原因如下：

- 客户端出现FullGC或者在请求期间主动调用了client.shutdown()方法。
- 服务端延迟过高，大于syncClientWaitFutureTimeoutInMillis的值。

解决方案

- 确认客户端中是否出现FullGC或者是否在请求期间主动调用了client.shutdown()方法导致callBack回调线程被关闭。
- 登录表格存储控制台确认服务端延迟是否过高。具体步骤如下：

  在数据表的**表管理**页面，单击**监控指标**页签后，选择表或索引，指定时间范围以及选择指标分组为**平均访问延迟**，即可查看不同操作类型的平均访问延迟信息。

  如果服务端延迟大于syncClientWaitFutureTimeoutInMillis的值，请提交工单联系表格存储技术支持。

2.API/SDK

2.1. Java SDK报错：SocketTimeoutException

现象

当“SDK接收到数据的时间”减去“SDK发送数据的时间”超过SocketTimeout时，SDK会抛出SocketTimeoutException。这段时间内包含了“应用发送请求（包含网络传输）”、“服务端处理”和“应用接收响应（包含网络传输）”。SocketTimeout可以在创建OTSClient时自定义，如果没有设置，默认是15s。

解决方案

出现SocketTimeoutException，可能是以下几种原因：

- 网络不通

  如果全部请求都是SocketTimeoutException，那么首先可能是网络不通，可以通过ping或者curl命令测试是否为网络问题：

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

  正常情况下，curl会返回类似下面的结果：

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

  如果发现是网络不通，可能是在非ECS环境使用了内网的endpoint。

- 服务端处理慢，超过了SDK设置的SocketTimeout值

  一个请求在表格存储服务端的处理时间几乎是不会超过15秒的，因为服务端也会有一个超时时间，大概是10秒，超过这个时间会给客户端返回OTSTimeout错误。

  但是，假设在SDK端自定义了SocketTimeout，例如2秒，那么当服务端执行时间超过2秒时，SDK就会抛出SocketTimeoutException错误。

- 网络传输慢

  如果服务端处理时间并不长，但是网络传输慢，导致整体延迟长，也会导致SocketTimeoutException。检查是否存在流量过高、带宽吃紧、网络重传率高等情况。

- Java进程的GC频繁，经常FullGC，导致SocketTimeoutException

  程序负载高，GC频繁时出现SocketTimeoutException。原因是当发生FullGC的时候，请求发不出去，或者收不到响应，超过了SDK端设置的SocketTimeout，就会抛出SocketTimeoutException。

  这种情况下需要用工具分析进程的GC情况，解决进程频繁GC的问题。

2.2. Java SDK日志库相关问题

表格存储Java SDK使用的是哪个日志库？

表格存储 Java SDK 依赖的是 slf4j，在依赖中默认依赖了 **log4j2** 作为日志实现库。

如何替换日志库？

您只需要在 Java SDK 的依赖中把 **log4j2** 的依赖声明移除即可，slf4j 就会自动在您的应用中寻找依赖的其他实现 slf4j 接口的日志库。

```
<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. 主键类型报错

现象

```
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
```

原因

建表时设置的主键类型为binary，写入数据时主键填了string类型的值。

解决方案

写入数据的类型与建表时设置的主键类型保持一致。

2.4. 使用SDK时出现Validate PK size fail异常

介绍使用SDK时出现Validate PK size fail异常的现象、原因和解决方案。

现象

使用SDK更新数据时出现如下异常：

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

原因

设置的主键个数和数据表的主键个数不一致。

解决方案

设置的主键个数必须和数据表的主键个数一致。

2.5. 使用SDK时出现Validate PK name fail异常

介绍使用SDK时出现Validate PK name fail异常的现象、原因和解决方案。

现象

使用SDK查询数据时出现如下异常：

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

原因

设置的主键名称和数据表的主键名称不一致，或者设置的主键顺序和数据表的主键顺序不一致。

解决方案

设置的主键名称与主键顺序必须和数据表的主键名称与主键顺序一致。

2.6. Java SDK报错：java.lang.IllegalStateException: Request cannot be executed; I/O reactor status: STOPPED

现象

使用Java SDK时出现如下异常：

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

原因

一般是因为OTSClient被调用了shutDown，其内部的I/O reactor均已被关闭。如果此时再调用OTSClient进行读写，则会抛出这个错误。

解决方案

检查OTSClient是否处于shutdown状态。

2.7. Java SDK报错：Invalid date format

现象

执行环境：Java 8

使用表格存储Java SDK时抛出以下异常：

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

原因

Classpath中依赖的Joda-time版本过低，joda-time的低版本在Java 8上会出现此类错误。

解决方案

可以更新到ots-public的最新版本2.2.4来解决这个问题。如果您也依赖了joda-time库，需要提升到2.9。

2.8. 使用Java SDK时出现PB库冲突

本文介绍了使用Java SDK时出现PB库冲突的现象、原因和解决方案。

现象

使用Java SDK时出现如下异常：

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

原因

Java SDK依赖2.4.1版本的Protobuf库和4.0.2版本的httpasyncclient，容易与您的应用程序中自带的相同库冲突。

解决方案

在Maven项目中的pom.xml中添加如下依赖即可。

> **说明** classifier为jar-with-dependencies，它将依赖的HttpClient和Protobuf库都通过rename package的方式打包进去，去除了对HttpClient和Protobuf的依赖。

```
<dependency>
        <groupId>com.aliyun.openservices</groupId>
        <artifactId>tablestore</artifactId>
        <version>替换为您当前使用的版本</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>
```

2.9. 使用SDK出现OTSUnsupportOperation异常

现象

调用 `syncClient.createTable(request)` 时出现如下错误。

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

原因

使用4.0.0之后版本的SDK访问旧版本的表。

解决方案

可以使用2.x.x版本的SDK。

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

2.10. 使用BatchWriteRow一次提交100条数据的时候报OTSParameterInvalid错误

现象

使用BatchWriteRow一次提交100条数据时出现以下错误。

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

原因

因为一次batch操作不能有重复行，如果有重复行，则会报错。

解决方案

将100条改为1条提交一次，其他代码不变，即可提交成功。

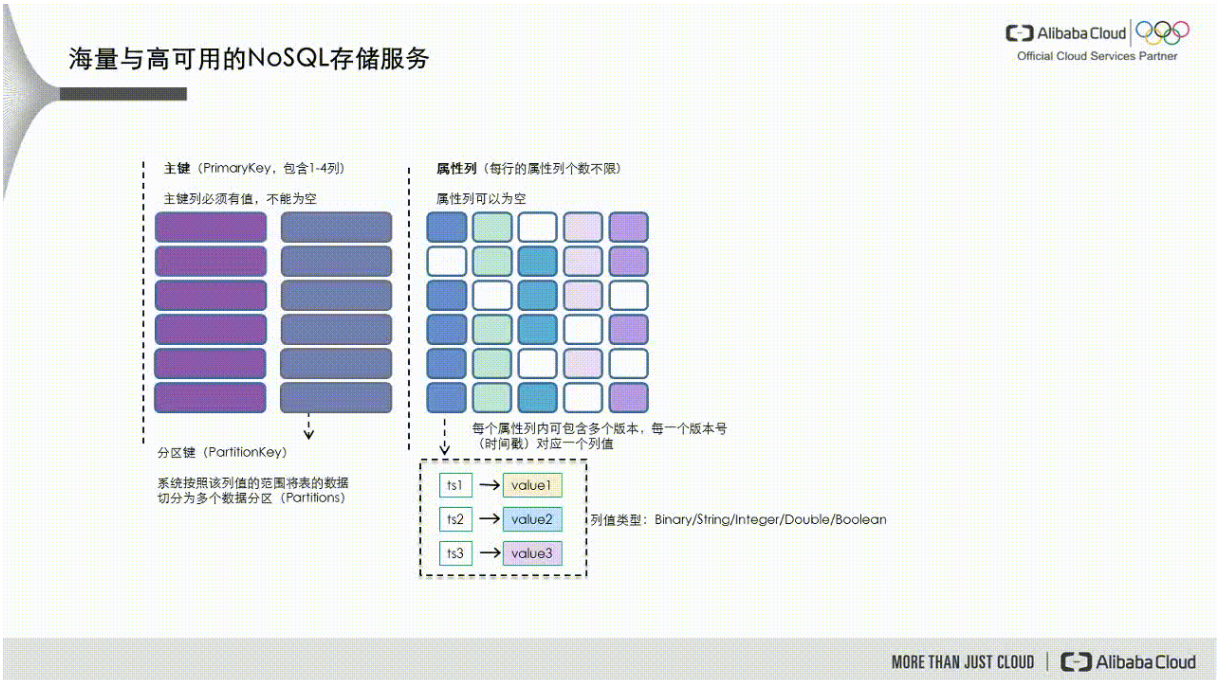
2.11. 为什么使用表格存储过程中会有少量的500错误

不少用户在使用表格存储的过程中偶尔会接到一些500错误，主要错误码如下。

HTTPStatus	ErrorCode	ErrorMsg
503	OTSPartitionUnavailable	The partition is not available.
503	OTSServerUnavailable	Server is not available.
503	OTSServerBusy	Server is busy.
503	OTSTimeout	Operation timeout.

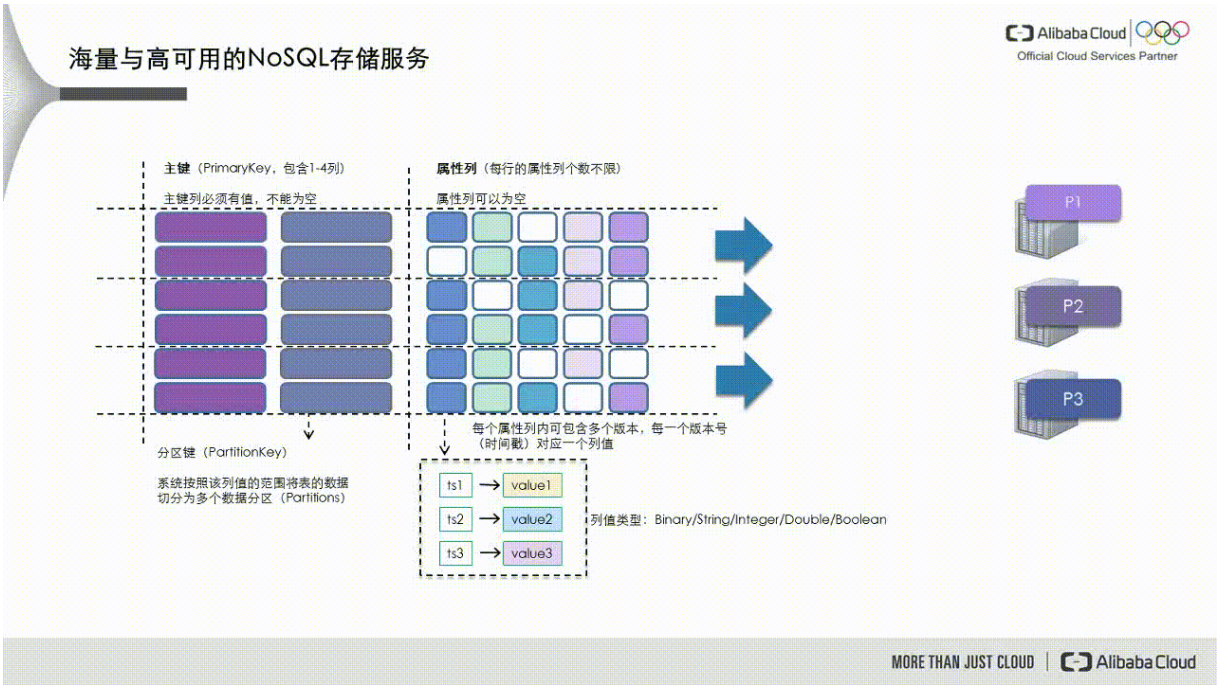
这是由于表格存储是一个纯分布式的NoSQL服务，服务端会根据数据分区的数据量、访问情况做自动的负载均衡，这样就突破了单机服务能力的限制，实现了数据规模和访问并发的无缝扩展。

如下图所示，表格存储会按照第一个主键的顺序，将实际数据划分到不同的数据分区中，不同的数据分区会调度到不同的服务节点提供读写服务。



当某个数据分区的数据量过大，或者访问过热，如下图的数据分区P1，表格存储的动态负载均衡机制能够检测到这种情况的发生，并将数据分区分裂成两个数据分区P1和P5，并将该两个数据分区调度到负载较低的服务节点上。

表格存储使用上述的自动负载均衡机制实现表数据规模和访问并发的自动扩展，全程无需人工介入，当然在数据表新建立时，只有一个数据分区，该表上能够提供的读写并发有限，自动负载均衡机制也有一定的延时性，所以可以直接联系到我们的工程师，预先将数据表划分成多个数据分区。



表格存储使用共享存储的机制，数据分区为逻辑单位，所以在负载均衡的过程中，不会有实际数据的迁移，仅仅是数据表元信息的变更，在元信息变更的过程中，为了保证数据的一致性，涉及到的数据分区会有短暂的不可用时间，**正常情况下影响时间为百毫秒级别，在数据分区负载较大时，可能会持续到秒级别，**在这个时间内对该分区的读写操作就有可能接到上述的错误，一般重试即可解决。在官方的SDK中默认提供了一些重试策略，在初始化Client端时就可以指定重试策略。

同时，表格存储提供的也是标准Restful API协议，由于网络环境的不可控，所有的读写操作也都建议增加重试策略，能够对网络错误等有一定的容错能力。

> **说明** 批量读写操作BatchWriteRow及BatchGetRow读写的数据可能属于多张表或者一张表的多个数据分区，有可能某一个分区正好在分裂，所以整个操作是非原子性的，只能够保证每个单行操作的原子性，该操作返回码为200时仍然需要检查response中的getFailedRows() 是否有失败的单行操作。

2.12. 只设置一个主键，如何获取多行数据？

关于如何查询多行数据，可以使用GetRange接口，具体的代码示例请参见Github。

2.13. 如何实现分页查询

表格存储是一个分布式存储系统，对于查询请求的翻页（分页），有多种方式。本文详细为您介绍如何实现分页查询。

表

如果只有表，没有多元索引，可以通过以下办法翻页：

- 使用next_token翻页：每次GetRange请求的Response中会有一个next_token，将这个next_token设置到下一次请求的Request中即可，这样就能实现连续翻页。
- 使用GetRangeIterator迭代器，通过iterator.next()方法持续获取下一条数据。
- 不支持offset跳页查询，如果业务需要，可以在客户端通过next_token或Iterator模拟。
- 不支持获取整个范围的行数和总页数。

多元索引

如果创建了多元索引，则可以通过以下办法翻页：

- 使用offset + limit方式：可以跳页，但是offset + limit最大值不能超过10000。如果超过，请使用next_token翻页。
- 使用next_token翻页，每次Search请求的Response中会有下一次的next_token，将这个next_token设置到下一次请求的Request中即可，这样就能实现连续翻页。
- 使用SearchIterator迭代器，通过iterator.next()方法持续获取下一条数据。
- 支持获取总行数，总行数除以limit就是总页数，需要在Request中设置getTotalCount为true，该选项打开后会增大资源消耗，所以性能会有所下降。

表示例

下面是一个实现分页读接口的示例代码，提供offset过滤以及读取指定页数的数据。

```
    /**
     * 范围查询指定范围内的数据，返回指定页数大小的数据，并能根据offset跳过部分行。
     */
    private static Pair<List<Row>, RowPrimaryKey> readByPage(OTSClient client, String table
Name,
            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;
        // 若查询的数据量很大，则一次请求有可能不会返回所有的数据，需要流式查询所有需要的数据。
        while (limit > 0 && nextStart != null) {
            // 构造GetRange的查询参数。
            // 注意：startPrimaryKey需要设置为上一次读到的位点，从上一次未读完的地方继续往下读，实现
流式的范围查询。
            RangeRowQueryCriteria criteria = new RangeRowQueryCriteria(tableName);
            criteria.setInclusiveStartPrimaryKey(nextStart);
            criteria.setExclusiveEndPrimaryKey(endKey);
            // 需要设置正确的limit，这里期望读出的数据行数最多为完整的一页数据以及需要过滤(offset)的
数据
            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--; // 对于offset之前的数据，需要过滤掉，采用的策略是读出来后在客户端进行过
滤。
                } else {
                    rows.add(row);
                    limit--;
                }
            }
            // 设置下一次查询的起始位点
            nextStart = response.getNextStartPrimaryKey();
        }
        return new Pair<List<Row>, RowPrimaryKey>(rows, nextStart);
    }
```

下面是使用以上接口，顺序地按页读取某个指定范围内的所有数据。

```
  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);
        // 读第一页，从范围的offset=33的行开始读起
        Pair<List<Row>, RowPrimaryKey> result = readByPage(client, tableName, startKey, end
Key, offset, pageSize);
        for (Row row : result.getKey()) {
            System.out.println(row.getColumns());
        }
        System.out.println("Total rows count: " + result.getKey().size());
        // 顺序翻页，读完范围内的所有数据
        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());
        }
    }
```

多元索引示例

详见排序和翻页。

2.14. 如何实现对特定列加一操作

您可以采取以下方式实现加一操作。

```
row = getRow(primary_key, 'col') // 先将该列的值读出来
old_value = row['col'] // 记录该列的旧的值
row['col'] = old_value + 1 // 计算新的值
updateRow(row, condition: row['col'] == old_value) // 写入新的值，写入时必须带条件检查，期望在写入时，当前列还是旧的值，即还没有其他人同时修改这一列
```

2.15. Python SDK ListTable示例

ListTable代码示例

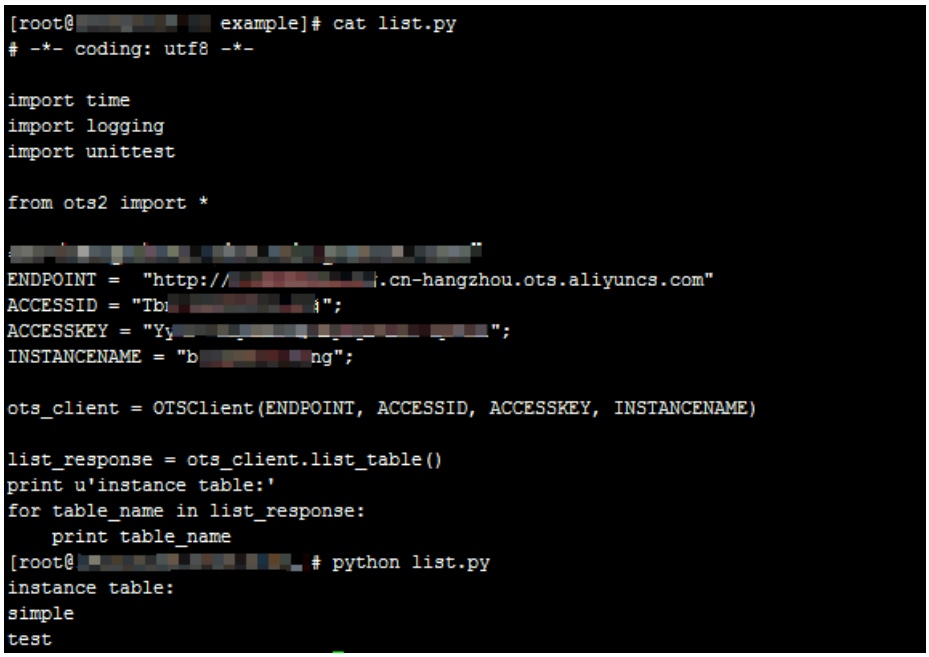
```
import time
import logging
import unittest
from ots2 import *
ENDPOINT = "https://xxx.cn-hangzhou.ots.aliyuncs.com";
ACCESSID = "xxx";
ACCESSKEY = "xxx";
INSTANCENAME = "xxx";
ots_client = OTSClient(ENDPOINT, ACCESSID, ACCESSKEY, INSTANCENAME)
list_response = ots_client.list_table()
print u'instance table:'
for table_name in list_response:
    print table_name
```

> **说明** Python SDK的安装和操作，请参见Python SDK。

Python SDK文档中没有import的提示，如不加import会出现如下提示。

```
Traceback (most recent call last):
  File "listtable.py", line 6, in
    ots_client = OTSClient(ENDPOINT, ACCESSID, ACCESSKEY, INSTANCENAME)
NameError: name 'OTSClient' is not defined
```

添加import运行即可。



2.16. 使用PHP SDK时出现Checksum mismatch异常

介绍使用PHP SDK时出现Checksum mismatch异常的现象、原因和解决方案。

现象

在Windows系统中通过PHP 5.6版本使用表格存储PHP SDK时出现如下异常：

```
Fatal error: Uncaught exception 'AliyunOTSOTSClientException' with message 'Checksum mismat
ch. expected:120,actual:-48'
```

原因

表格存储的整型是64位的，而32位PHP只能用string表示64位的整型，所以暂不支持32位PHP；且Windows系统中PHP 7之前版本的整型不是真正的64位。

解决方案

在Windows系统中使用表格存储PHP SDK时，PHP版本必须使用PHP 7及PHP 7以上的64位版本，强烈建议使用PHP 7以获得最佳性能。

通过phpinfo()查看PHP配置信息中的Architecture类型，可以判断PHP版本是否满足使用要求。

- 当Architecture为X86时，表示PHP版本是32位，需要升级PHP版本到PHP 7及PHP 7以上的64位版本。
- 当Architecture为X64时，表示PHP版本是64位，满足使用要求。

2.17. 使用SDK访问实例时出现Request denied because this instance can only be accessed from the binded VPC异常

介绍使用SDK访问实例时出现Request denied because this instance can only be accessed from the binded VPC异常的现象、原因和解决方案。

现象

使用SDK访问实例时出现如下异常：

```
[ErrorCode]:OTSAuthFailed, [Message]:Request denied because this instance can only be acces
sed from the binded VPC.
```

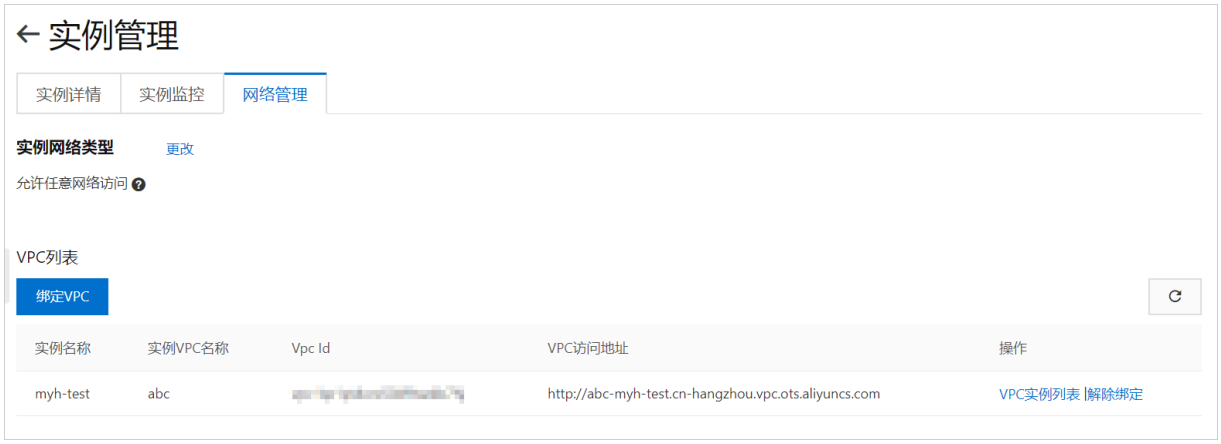
原因

实例绑定VPC后，只能在绑定的VPC中使用VPC访问地址访问该实例。

解决方案

实例绑定VPC后，使用SDK访问实例时，请在绑定的VPC中使用VPC访问地址进行访问。

通过控制台在实例的**网络管理**页签可以获取实例的VPC访问地址，如下图所示。


← 实例管理
实例详情
实例监控
网络管理
实例网络类型
更改
允许任意网络访问
VPC列表
绑定VPC
实例名称
实例VPC名称
Vpc Id
VPC访问地址
操作
myh-test
abc
http://abc-myh-test.cn-hangzhou.vpc.ots.aliyuncs.com
VPC实例列表 |解除绑定