表格存储

产品简介

产品简介

什么是表格存储

表格存储（Table Store）是构建在阿里云飞天分布式系统之上的 NoSQL 数据存储服务，提供海量结构化数据的存储和实时访问。

表格存储以实例和表的形式组织数据，通过数据分片和负载均衡技术，达到规模的无缝扩展。

表格存储向应用程序屏蔽底层硬件平台的故障和错误，能自动从各类错误中快速恢复，提供非常高的服务可用性。

表格存储管理的数据全部存储在 SSD 中并具有多个备份，提供了快速的访问性能和极高的数据可靠性。

您在使用表格存储服务时，只需按照预留和实际使用的资源进行付费，无需关心数据库的软硬件升级维护、集群缩容扩容等复杂问题。

相关概念

数据模型

表格存储的**数据模型**概念包括表、行、主键和属性。其中，表是行的集合，行由**主键和属性**组成。

数据生命周期

数据生命周期（Time To Live，简称 TTL）是数据表的一个属性，即数据的存活时间，单位为秒。表格存储会在后台对超过存活时间的数据进行清理，以减少您的数据存储空间，降低存储成本。

地域

地域（Region）是指物理的数据中心，表格存储服务会部署在多个阿里云地域中，您可以根据自身的业务需求选择不同地域的表格存储服务。详情请查看**表格存储已经开通的 Region**。

读/写吞吐量

读/写吞吐量的单位为读服务能力单元和写服务能力单元，简称 CU （ Capacity Unit ），是数据读写操作的最小计费单位。此外，读/写吞吐量分为预留读/写吞吐量和按量读/写吞吐量两种。

相关服务

您把数据存储到表格存储以后，可以使用阿里云提供的其他产品和服务对其进行相关操作：

如何将表格存储中的全量数据导出以及增量同步到 OSS，请参考数据通道 OSS 概述。

如何在同一个云账号下实现表格存储和 MaxCompute 之间的无缝连接，请参考使用 MaxCompute 访问表格存储。

如何使用函数计算对表格存储增量数据进行实时计算，请参考使用函数计算。

如何将表格存储中的全量数据通过开源工具 DataX导出到 Elasticsearch 中，请参考全量导出（ DataX ）。

使用表格存储

阿里云提供了Web服务页面方便您管理表格存储。您可以登录表格存储管理控制台操作表格存储实例。

阿里云也提供了API和SDK接口方便您灵活地管理表格存储。请参见表格存储 API 参考和表格存储 SDK 参考。

表格存储定价

表格存储计量项包括数据存储量、预留读/写吞吐量、按量读/写吞吐量和外网下行流量。更多信息，请参见计量项和计费说明。

表格存储及相关资源的价格信息，请参考云产品定价页。

产品优势

表格存储是一种完全托管的 NoSQL 数据库服务，提供快速而可预测的性能，能够实现无缝扩展。使用表格存储，您可以免除操作和扩展分布式数据库的管理工作负担，因而无需担心硬件预置、设置和配置、复制、软件修补等问题。此外，表格存储还包含以下优势：

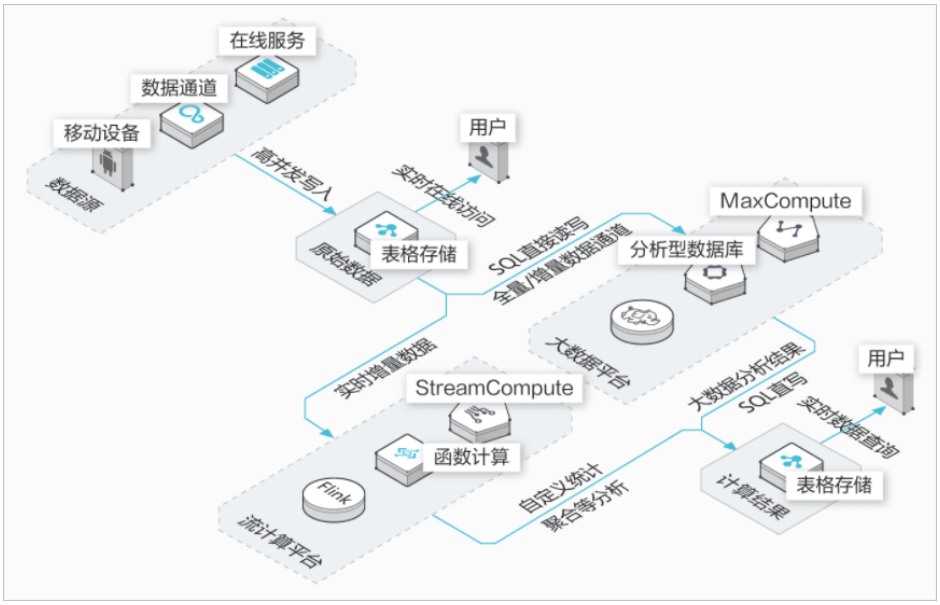
优势项	描述

扩展性	- 动态调整预留读/写吞吐量 在创建表的时候，应用程序可以根据业务访问的情况来配置预留读/写吞吐量。表格存储根据表的预留读/写吞吐量进行资源的调度和预留，从而获得更低的资源使用成本。在使用过程中，还可以根据应用程序情况动态修改预留读/写吞吐量。 - - 无限容量 表格存储中表的数据量没有上限，随着表数据量的不断增大，表格存储会进行数据分区的调整从而为该表配置更多的存储
数据可靠性	表格存储将数据的多个备份存储在不同机架的不同机器上，并会在备份失效时进行快速恢复，提供了极高的数据可靠性，数据可靠性为99.99999999%（10个9）。
高可用性	通过自动的故障检测和数据迁移，表格存储对应用程序屏蔽了机器和网络的硬件故障，提供了高可用性，服务可用性为99.9%。
管理便捷	应用程序无需关心数据分区的管理、软硬件升级、配置更新、集群扩容等繁琐的运维任务。
访问安全性	表格存储提供多种权限管理机制，并对应用的每一次请求都进行身份认证和鉴权，以防止未授权的数据访问，确保数据访问的安全性。
强一致性	表格存储保证数据写入强一致，并保证数据 3 副本均写入磁盘且数据最终将在所有存储位置中保持一致。写操作一旦返回成功，应用程序就能立即读到最新的数据。
灵活的数据模型	表格存储的表无固定格式要求，每行的列数及不同行同名列的类型可以不相同，支持多种数据类型，如 Integer、Boolean、Double、String 和 Binary。
多租户机制	表格存储使用共享存储机制，支持不同用户的多个实例共享同一集群资源，以数据分区为最小服务单位，支持以数据分区级别的负载均衡机制来隔离不同实例之间的影响。
按量付费	表格存储根据用户预留和实际使用的资源进行收费，起步门槛低。
监控集成	您可以从表格存储控制台实时获取每秒请求数、平均响应延时等监控信息。

应用场景

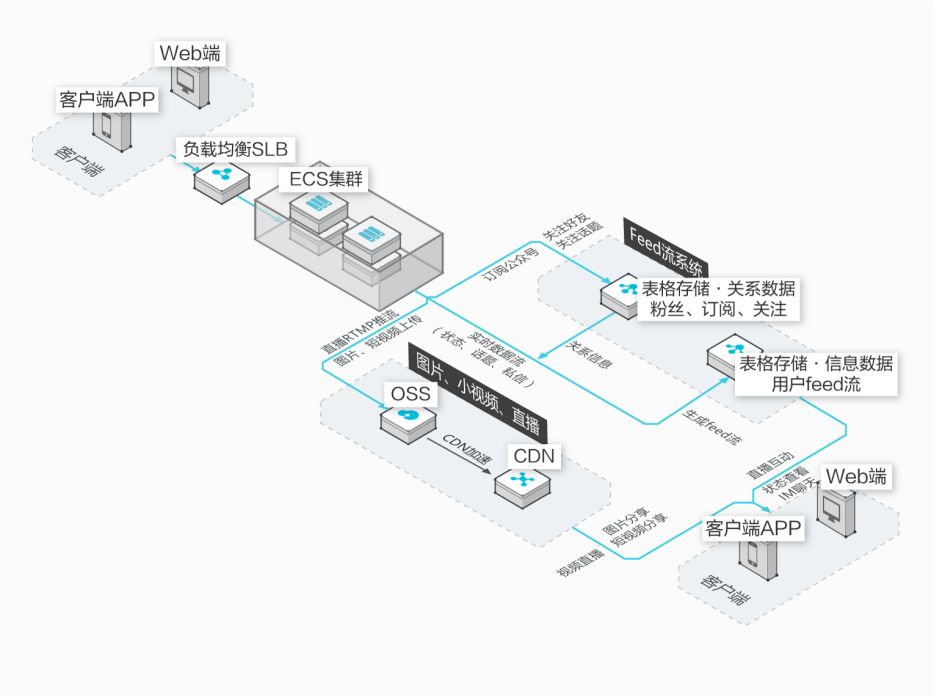
大数据存储与分析

表格存储提供低成本、高并发、低延时的海量数据存储与在线访问，提供增量以及全量数据通道，并支持 MaxCompute 等大数据分析平台的 SQL 直读直写。高效的增量流式读接口让数据轻松完成实时流计算。



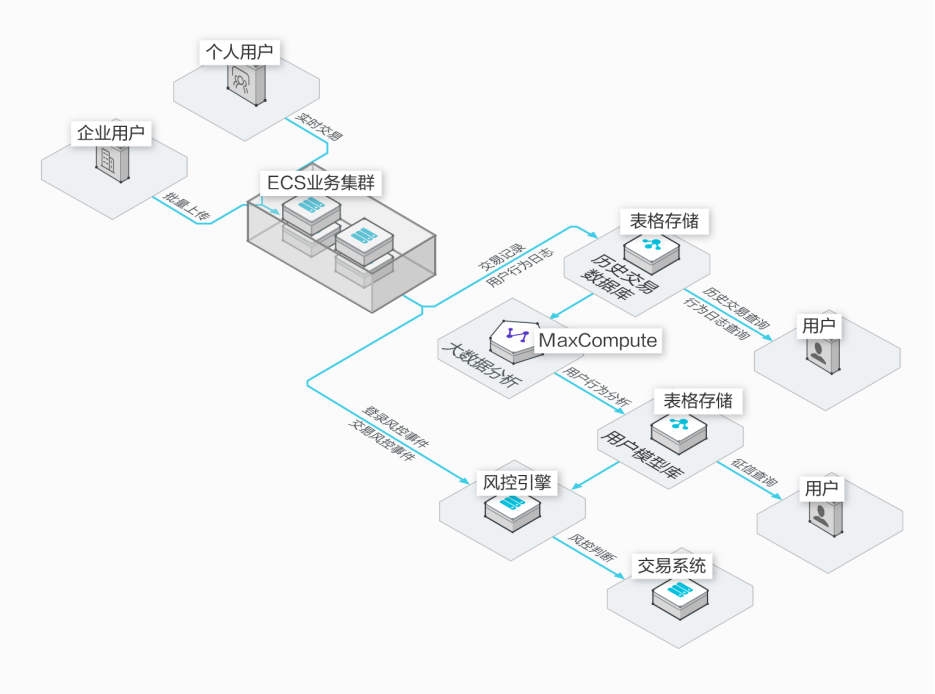
社交 Feed 流

表格存储能够存储人与人之间产生的大量社交信息，包括 IM 聊天，以及评论、跟帖和点赞等 Feed 流信息。表格存储的弹性资源按量付费，能够以较低的成本满足访问波动明显、高并发、低延时的需要。图片与视频存储在 OSS 上通过 CDN 加速带来最优的用户体验。



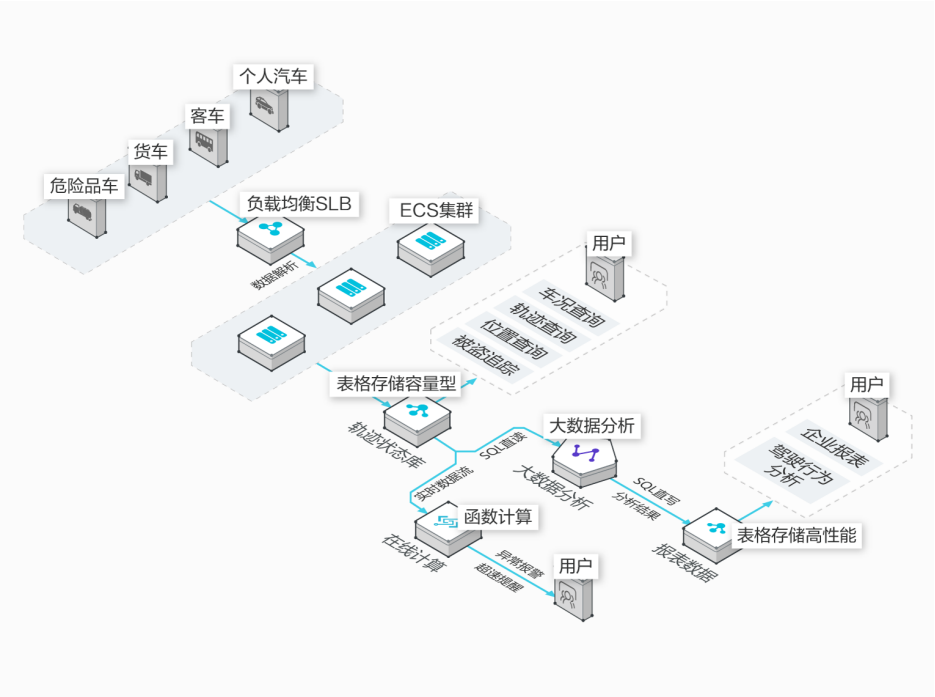
金融风控

低延时、高并发，弹性资源按量付费让您的风控系统永远工作在最佳状态，牢牢控制交易风险。灵活的数据结构能够让业务模式跟随市场需求快速迭代。



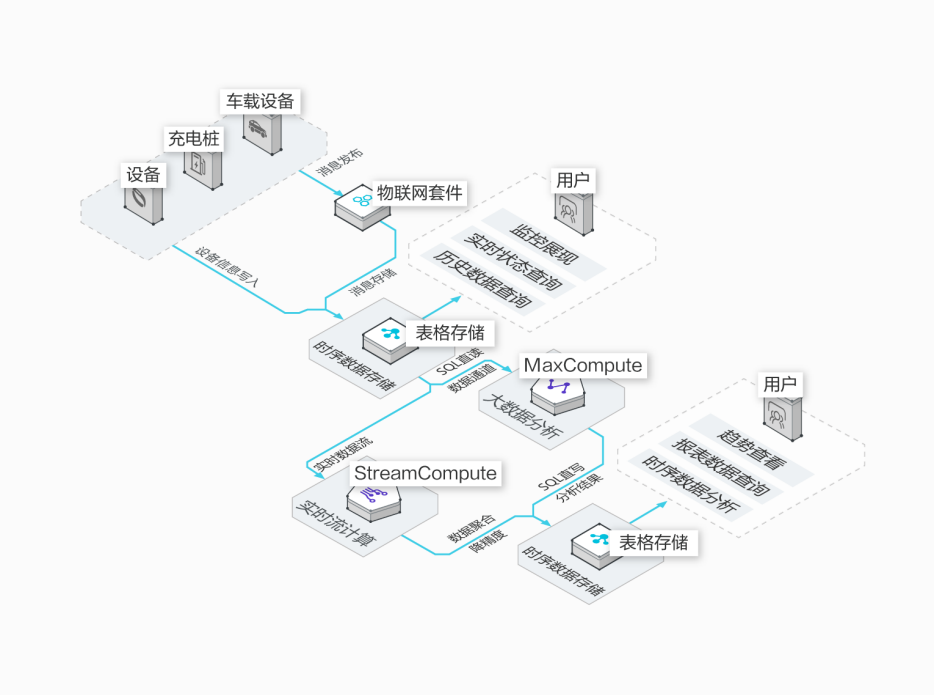
车联网数据

单表支撑 PB 级数据量，无需分库分表，简化业务逻辑，schemafree 数据模型轻松接入不同车辆设备的监控数据。与多种大数据分析平台、实时计算服务等无缝结合，轻松完成实时在线查询以及业务报表分析。



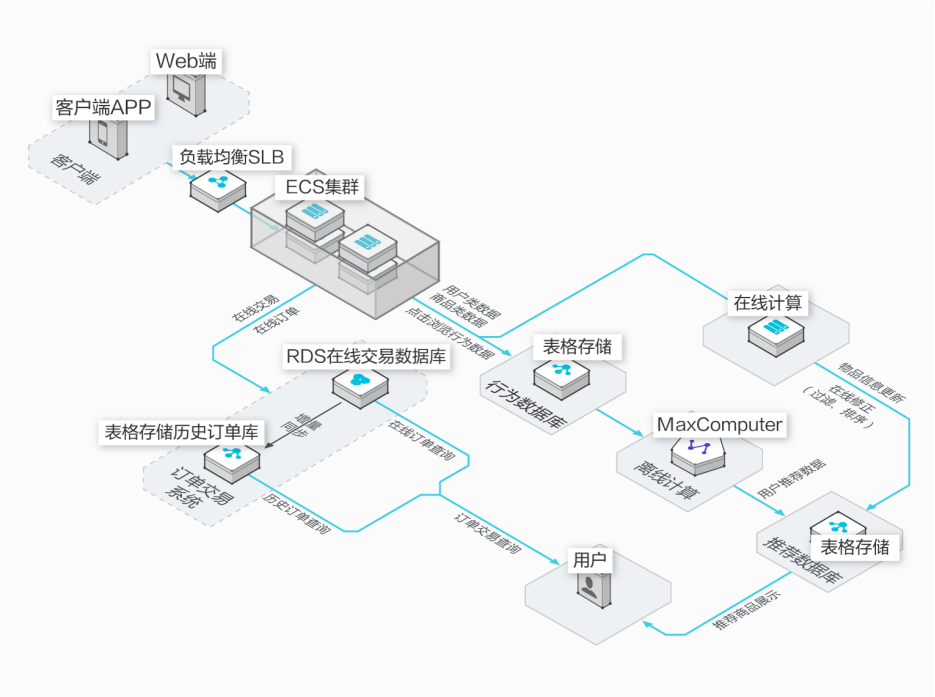
物联网时序数据

单表 PB 级数据规模及千万级 QPS 让表格存储轻松满足 IoT 设备、监控系统等时序数据的存储需求，大数据分析 SQL 直读以及高效的增量流式读接口让数据轻松完成离线分析与实时流计算。



电商推荐

使用表格存储让您无需担心大量历史交易订单的数据规模与访问性能，配合大数据计算服务，轻松实现精准营销，弹性资源、按量付费，让您从容应对所有用户在线高峰时刻。



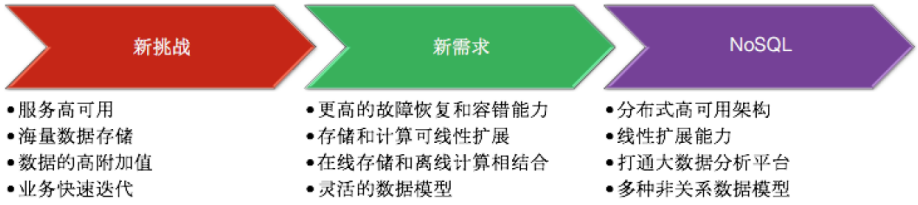
名词解释

Table Store 数据模型 - Wide Column 和 Timeline

前言

Table Store 是阿里云自研的一款分布式 NoSQL 数据库，提到 NoSQL 数据库，现在对很多应用研发而言都已不再陌生。当前很多应用系统底层不再仅依赖于关系型数据库，而是会根据不同的业务场景来选择不同类型的数据库，例如缓存型 KeyValue 数据存储在 Redis、文档型数据存储在 MongoDB、图数据会存储在 Neo4J 等。

传统的关系型数据库很难承载如此海量的数据，需要一种具备高扩展能力的分布式数据库。但基于传统的关系数据模型，实现高可用和可扩展的分布式数据库非常困难。如果能打破关系模型，以一种更简单的数据模型对数据建模、弱化事务和约束、以高可用和可扩展、能更好地满足业务需求为设计理念，基于这样的理念推动了 NoSQL 的发展。



NoSQL 具有以下特征：

多数据模型

为满足不同数据的需求，提供了多数据模型供您选择，例如 KeyValue、Document、Wide Column、Graph 以及 Time Series 等。NoSQL数据库打破了关系模型的约束，选择了多元的发展方向。多数据模型更贴近不同场景的实际需求。

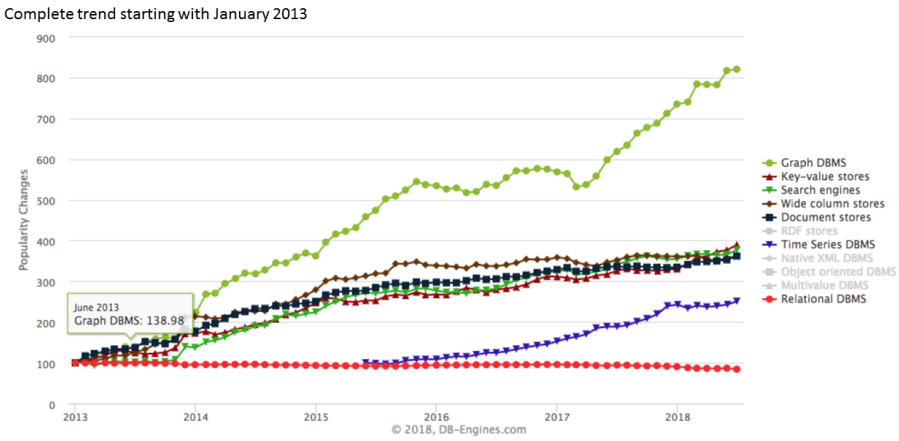
高并发、低延迟

NoSQL 的设计目标是面向在线业务提供高并发、低延迟的访问。

高可扩展

为应对爆发的数据量增长，可扩展是核心的设计目标之一。

DB-Engines 旨在收集和介绍数据库管理系统（DBMS）方面的信息。下面展示了 DB-Engines 收集到的各类 NoSQL 数据库从 2013 年到 2018 年的发展趋势。

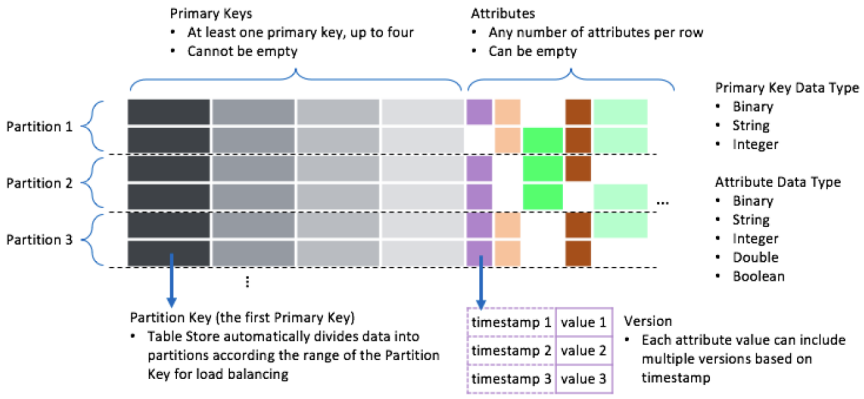


从 DB-Engines 的发展趋势来看，各类 NoSQL 数据库在近几年都处于一个蓬勃发展的状态。阿里云 Table

Store 作为一款分布式 NoSQL 数据库，在数据模型上选择了多模型的架构，同时支持 Wide Column 和 Timeline。

Wide Column 模型由 Bigtable 提出，后被其他同类型系统广泛应用的一种经典模型。目前绝大部分半结构化、结构化数据都存储在该模型系统中。除了 Wide Column 模型外，我们推出了另一种全新的数据模型 Timeline。Timeline 模型是一种用于消息数据的新一代模型，适用于 IM、Feeds 和物联网设备消息下推等消息系统中消息的存储和同步，目前已被广泛使用。接下来，我们详细了解下这两种模型。

Wide Column 模型



Wide Column 模型对比关系模型的区别如下：

Wide Column 模型具有以下特征：三维结构（行、列和时间）、schema-free、宽行、多版本数据以及生命周期管理。

关系模型的特征如下：二维（行、列）以及固定的Schema。

Wide Column 模型由以下几个部分组成：

主键（Primary Key）：每一行都会有主键，主键会由多列（1-4列）构成，主键的定义是固定Schema，主键主要用于唯一区分一行数据。

分区键（Partition Key）：主键的第一列称为分区键，分区键用于对表进范围分区。每个分区会分布式的调度到不同的机器上进行服务。在同一个分区键内，我们提供跨行事务。

属性列（Attribute Column）：一行中除主键列外，其余都是属性列。属性列会对应多个值，不同值对应不同的版本，一行可存储不限个数个属性列。

版本（Version）：每一个值对应不同的版本，版本的值是一个时间戳，用于定义数据的生命周期。

数据类型（Data Type）：Table Store 支持多种数据类型，包含 String、Binary、Double、

Integer 和 Boolean。

生命周期（Time To Live）：每个表可定义数据生命周期。例如生命周期配置为一个月，则该表数据中在一个月之前写入的数据就会被自动清理。数据的写入时间由版本来决定，版本一般由服务端在数据写入时根据服务器时间来定，也可由应用指定。

最大版本数（Max Versions）：每个表可定义每一列最多保存的版本数，用于控制一列的版本的个数。当一个属性列的版本个数超过 Max Versions 时，最早的版本将被异步删除。

同时 Wide Column 模型在数据操作层面，提供两类数据访问 API：Data API 和 Stream API。

Data API

关于 Data API 的详细文档，请参考**表格存储的数据操作**。Data API 是标准的数据 API，提供数据的在线读写，包括：

PutRow：新插入一行，如果存在相同行，则覆盖。

UpdateRow：更新一行，可增加、删除一行中的属性列，或者更新已经存在的属性列的值。如果该行不存在，则新插入一行。

DeleteRow：删除一行。

BatchWriteRow：批量更新多张表的多行数据。

GetRow：读取一行数据。

GetRange：范围扫描数据，可正序扫描或者逆序扫描。

BatchGetRow：批量查询多张表的多行数据。

Stream API

在关系模型数据库中没有对数据库内增量数据定义标准的 API，但是在传统关系数据库的很多应用场景中，增量数据（Binlog）的用途不可忽视。Table Store将增量数据的能力挖掘出来后可以在技术架构上实现：

- 异构数据源复制：MySQL 数据增量同步到 NoSQL，做冷数据存储。
- 对接流计算：可实时对MySQL内数据做统计，做一些大屏类应用。
- 对接搜索系统：可将搜索系统扩展为 MySQL 的二级索引，增强 MySQL 内数据的检索能力。

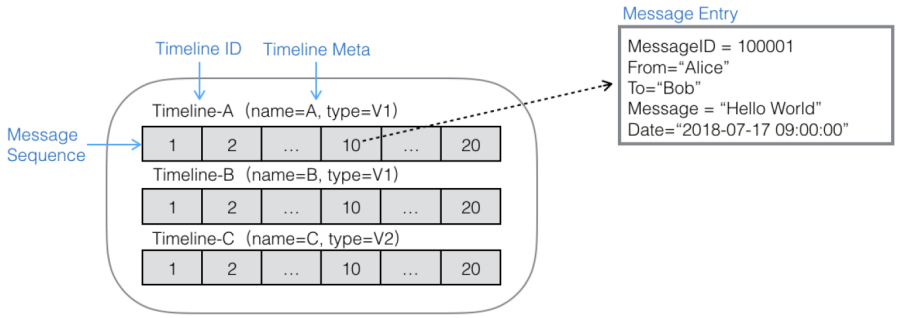
Table Store挖掘了增量数据存在的价值，并提供了标准的**Stream API**。Steam API提供的接口如下：

- ListStream：获取表的 Stream 信息，如 StreamID。
- DescribeStream：获取 Stream 的详细信息，如 Shard 列表、Shard 结构树等。
- GetShardIterator：获取 Shard 当前增量数据的 Iterator。
- GetStreamRecord：根据 ShardIterator 获取 Shard 内的增量数据。

Table Store Stream 的实现会比 MySQL Binlog 复杂得多。因为 Table Store 本身是一个分布式的架构，Stream 也是一个分布式的增量数据消费框架。Stream 的数据消费必须保序获取，Stream 的 Shard 与 Table Store 内部表的分区一一对应，且表的分区会存在分裂和合并。

Timeline 模型

Timeline 模型是针对消息数据场景所设计的数据模型，它能满足消息数据场景对消息保序、海量消息存储、实时同步的特殊需求。



说明：将一张大的数据表内的数据抽象为多个 Timeline，能够承载的 Timeline 个数无上限。

Timeline 的构成主要包括：

- Timeline ID：用于唯一标识 Timeline。
- Timeline Meta：Timeline 的元数据，元数据内可包含任意键值对属性。
- Message Sequence：消息队列，承载该 Timeline下的所有消息。

  说明： 消息在队列里有序保存，并且根据写入顺序分配自增的 ID。一个消息队列可承载的消息个数无上限，在消息队列内部可根据消息 ID 随机定位某条消息，并提供正序或者反序扫描。

- Message Entry：消息体，包含消息的具体内容，可以包含任意键值对。

有关 Timeline 模型的起源，请查看现代IM系统中消息推送和存储架构的实现。有关 Timeline 模型的具体应用，请查看Table Store Timeline：轻松构建千万级 IM 和 Feed 流系统。

最大版本数

最大版本数（Max Versions）是数据表的一个属性，表示该数据表中的属性列能够保留多少个版本的数据。当一个属性列的版本个数超过 Max Versions 时，最早的版本将被异步删除。

建表后，您可以通过 UpdateTable 接口动态更改数据表的 Max Versions。

注意：

- 超过 Max Versions 的数据版本为无效数据，即使数据还没有被真正删除，该数据对用户已经不可见，无法读出。
- 当调小 Max Versions 时，如果数据版本个数超过新设的 Max Versions，最早的版本会被系统异步删除。
- 当调大 Max Versions 时，如果以前版本个数超过旧的 Max Versions 但还没有被系统删除的，数据会被重新读出来。

数据生命周期

数据生命周期（Time To Live，简称 TTL）是数据表的一个属性，即数据的存活时间，单位为秒。表格存储会在后台对超过存活时间的数据进行清理，以减少用户的数据存储空间，降低存储成本。

TTL 由用户在建表时进行设置，如果希望数据永不过期，将其设置为 **-1**。

建表后，可以通过 UpdateTable 接口动态更改 TTL。

TTL 的单位为秒，例如期望过期时间为 30 天，TTL 应设置为 2592000（即 30 * 24 * 3600）。

假设数据表的 TTL 设置为 86400 （一天），在 2016-07-21 00:00:00 UTC 时，该数据表上所有版本号小于 1468944000000 （除以 1000 换算成秒之后即 2016-07-20 00:00:00 UTC）的属性列都将过期，系统会自动清理这些过期的数据。

注意：

- 超过 TTL 的过期数据为无效数据，即使数据还没有被真正删除，该数据对用户已经不可见，无法读出。
- 当调小 TTL 时，可能会有数据因为 TTL 变小而过期，这部分数据会被系统异步删除。
- 当调大 TTL 时，如果有版本号在上个 TTL 之外的数据还没有被系统删除，数据会被重新读出。

有效版本偏差

有效版本偏差（Max Version Offset）是数据表的一个属性，单位为秒。

为了防止非期望的写入，服务端在处理写请求时会对属性列的版本号进行检查。当版本号小于当前写入时间减去 Max Version Offset，或者大于等于当前写入时间加上 Max Version Offset 的值时，该行数据写入失败。

属性列的有效版本范围为：**[数据写入时间 - 有效版本偏差，数据写入时间 + 有效版本偏差)**。数据写入时间为 1970-01-01 00:00:00 UTC 时间到当前写入时间的秒数。属性列版本号为毫秒，其除以 1000 换算成秒之后必须属于这个范围。

例如，当数据表的有效版本范围为 86400（一天），在 2016-07-21 00:00:00 UTC 时，只能写入版本号大于 1468944000000（换算成秒之后即 2016-07-20 00:00:00 UTC）并且小于 1469116800000（换算成秒之后即 2016-07-22 00:00:00 UTC）的数据。当某一行的某个属性列版本号为 1468943999000（换算成秒之后即 2016-07-19 23:59:59 UTC）时，该行数据写入失败。

建数据表时，用户若不设置有效版本偏差，将使用默认值 **86400**。

建表后，可以通过 UpdateTable 接口动态更改有效版本偏差。

有效版本偏差为非 0 值，可以大于 1970-01-01 00:00:00 UTC 时间到当前时间的秒数。

主键和属性

主键

主键是表中每一行的唯一标识。主键由 1 到 4 个主键列组成。

创建表的时候，必须明确指定主键的组成、每一个主键列的名字和数据类型以及它们的顺序。

主键列的数据类型只能是 String、Integer 和 Binary。如果为 String 或者 Binary 类型，长度不超过 1 KB。

属性

属性存放行的数据。每一行包含的属性列个数没有限制。

版本号

在一个属性列上，当写入的版本数超过数据表的最大版本数时，较早版本的数据会被删除，只保留最新的 Max Versions 的版本数。

在写入数据时可以指定属性列的版本号，如果不指定版本号，服务端会将当前时间的毫秒单位时间戳（从 1970-01-01 00:00:00 UTC 计算起的毫秒数）作为属性列生成版本号。比如属性列版本号为 1468944000000（即 2016-07-20 00:00:00 UTC），当数据表的 TTL 设置为 86400（一天）时，该版本的数据将会在 2016-07-21 00:00:00 UTC 过期，随后会被后台系统自动删除。

读取一行数据时，可以指定每列最多读取多少版本或者读取的版本号范围。

注意：

- 版本号的单位为毫秒，在进行 TTL 比较和有效版本偏差计算时，需要除以 1000 换算成秒。
- 当数据的版本号（即时间戳）完全由服务端决定时，写入的数据在写入后经过 TTL 秒后会被系统清理。
- 为了防止无效的写入，写入过期数据将会直接失败。例如在 2016-07-21 00:00:00 向 TTL 为 86400 的数据表中写入版本号小于 1468944000000（即 2016-07-20 00:00:00 UTC）的数据将会直接失败。
- 为了防止错误的写入，写入的属性列的版本号换算成秒后，需要在 **[数据写入时间-有效版本偏差，数据写入时间+有效版本偏差)** 的范围内。

列名的命名规范

主键列和属性列遵循如下命名规范：

必须由英文字母、数字或下划线（_）组成

首字符必须为英文字母或下划线（_）

大小写敏感

长度在 1~255 个字符之间

列值类型

表格存储支持 5 种类型的列值：

数据类型	定义	是否可为主键	大小限制

String	UTF-8，可为空	是	为主键列时最大为 1 KB，为属性列时请参考限制说明。
Integer	64 bit，整型	是	8 Bytes
Double	64 bit，Double 类型	否	8 Bytes
Boolean	True/False，布尔类型	否	1 Byte
Binary	二进制数据，可为空	是	为主键列时最大为 1 KB，为属性列时请参考限制说明。

分区键

组成主键的第一个主键列又称为分区键。表格存储会根据表中每一行分区键的值所属的范围自动将这一行数据分配到对应的分区和机器上，以达到负载均衡的目的。

具有相同分区键值的行属于同一个数据分区，一个分区可能包含多个分区键值。分区键的值是最小的分区单位，相同的分区键值下的数据无法再做切分。为了防止分区过大无法切分，单个分区键值下所有行的大小总和建议不超过 10 GB。

表格存储服务会根据特定的规则对分区进行分裂和合并，以达到更好的负载均衡。这个过程是自动的，应用程序无需关心。

读/写吞吐量

读/写吞吐量的单位为读服务能力单元和写服务能力单元，简称 CU （ Capacity Unit ），是数据读写操作的最小计费单位。

1 单位读能力表示从数据表中读一条 4 KB 数据。

1 单位写能力表示向数据表写一条 4 KB 数据。

操作数据大小不足 4 KB 的部分向上取整，如写入 7.6 KB 数据消耗 2 单位写能力，读出 0.1 KB 数据消耗 1 单位读能力。

应用程序通过 API 进行表格存储读写操作时，会消耗对应的写服务能力单元和读服务能力单元。

预留读/写吞吐量

预留读/写吞吐量是表的一个属性。应用程序在创建表的时候，可以为该表指定预留读/写吞吐量。预留读写吞

吐量的配置不影响该数据表的访问性能和服务能力。

设置预留吞吐量之后，应用程序每秒不超过预留吞吐量的访问将会按照预留吞吐量的单价进行计费。

例如，假设应用程序从表中每秒读取 80 条记录，每条记录的大小均为 3 KB，在这种情况下，每秒将消耗80个读吞吐量。

将预留读吞吐量设置为80 ，一个小时的读费用为 80 * 预留读单价，能够处理的读操作为 80 * 3600 = 288000 次。

注意：

- 预留读/写吞吐量可以设置为 0。
- 当预留读/写吞吐量大于 0 时，表格存储根据该配置为表分配和预留相应的资源，从而获得更低的资源使用成本。
- 当预留读/写吞吐量大于 0 时，即使没有读写请求也会进行计费，所以表格存储限制用户能够自行设置的单表预留读写吞吐量最大为 5000 （读和写分别不超过 5000）。如果用户有单表预留读写吞吐量需要超出 5000 的需求，可以提交工单提高预留读写吞吐量。
- 不存在的表将被视作预留读和预留写吞吐量均为 0，访问不存在的表将根据操作类型消耗 1 个按量读 CU 或者 1 个按量写 CU。

应用程序可以通过 UpdateTable 操作动态修改表的预留读/写吞吐量配置。

按量读/写吞吐量

按量读/写吞吐量是数据表在每一秒钟实际消耗的读/写吞吐量中超出预留读/写吞吐量的部分，统计周期为 1 秒。

假如数据表设置的预留读吞吐量为 100，连续3秒的访问情况如下：

- T0：读操作实际消耗 120 读吞吐量，则这 1 秒内预留吞吐量为100，消耗的按量读吞吐量为 20。
- T1：读操作实际消耗 95 读吞吐量，则这 1 秒内预留吞吐量为100，消耗的按量读吞吐量为 0。
- T2：读操作实际消耗 110 读吞吐量，则这 1 秒内预留吞吐量为100，消耗的按量读吞吐量为 10。

T0 至 T2 时刻的消耗的读吞吐量为：100 预留读吞吐量以及 30 按量读吞吐量。

说明：每个小时内，表格存储对预留吞吐量取平均值，对按量吞吐量取累加值来作为用户实际消耗的吞吐量。

由于按量读/写吞吐量的模式无法预估需要为数据表预留的计算资源，表格存储需要提供足够的服务能力以应对突发的访问高峰，所以按量吞吐量的单价高于预留吞吐量的单价。合理设置数据表的预留吞吐量能够有效地降低使用成本。

注意：由于按量读/写吞吐量无法准确估计需要预留的资源，在某些极端访问情况下，若单个分片键每秒钟的访问需要消耗 10000 CU，表格存储可能会返回 OTSCapacityUnitExhausted 错误给应用程序。此

时，应用程序需要使用退避重试等策略来减少访问该表的频率。

更多信息请参考 Table Store 表和计费方式。

地域

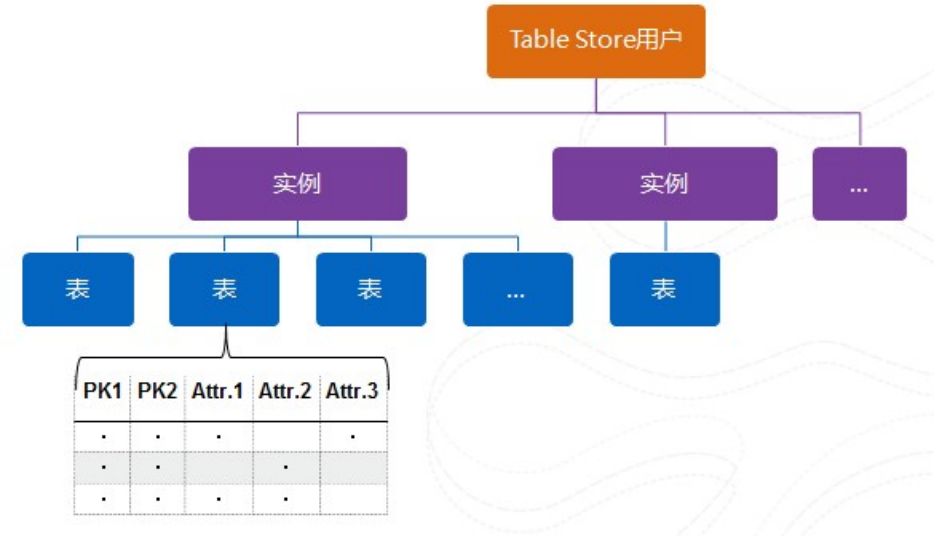
地域（Region）是指物理的数据中心，表格存储服务会部署在多个阿里云地域中，用户可以根据自身的业务需求选择不同地域的表格存储服务。

表格存储当前支持的地域与RegionID的对应关系如下表所示：

地域名称	RegionID
华东 1	cn-hangzhou
华东 1 金融云	cn-hangzhou-finance
华东 2	cn-shanghai
华东 2 金融云	cn-shanghai-finance-1
华北 2	cn-beijing
华北 3	cn-zhangjiakou
华北 5	cn-huhehaote
华南 1	cn-shenzhen
香港	cn-hongkong
亚太东南 1（新加坡）	ap-southeast-1
美国东部 1（弗吉尼亚）	us-east-1
美国西部 1（硅谷）	us-west-1
亚太东北 1（东京）	ap-northeast-1
欧洲中部 1（法兰克福）	eu-central-1
中东东部 1 (迪拜)	me-east-1
亚太东南 2 （悉尼）	ap-southeast-2
亚太东南 3 （吉隆坡）	ap-southeast-3
亚太东南 5 (雅加达)	ap-southeast-5
亚太南部 1 (孟买)	ap-south-1

实例

实例（Instance）是您使用和管理表格存储服务的实体，对应于关系型数据库的 Database。您在开通表格存储服务之后，需要通过管理控制台来创建实例，然后在实例内进行表的创建和管理。实例是表格存储资源管理的基础单元，表格存储对应用程序的访问控制和资源计量都在实例级别完成。



您可以为不同的业务创建不同的实例来管理相关的表，也可以为同一个业务的开发测试和生产环境创建不同的实例。

表格存储允许一个云账号最多创建 10 个实例，每个实例内最多创建 64 张表。如果您需要增加限额，请提交工单。

命名规范

实例的名称在一个地域内必须唯一，不同的地域内实例名称可以相同。具体命名规范如下：

必须由英文字母、数字或连字符（-）组成

首字符必须为英文字母

末尾字符不能为连字符（-）

大小写不敏感

长度在 3 – 16 Byte 之间

实例名称不能包含 [‘ali’ ， ‘ay’ ， ‘ots’ ， ‘taobao’ ， ‘admin’] 这几个单词

规格

表格存储支持两种实例规格：高性能实例和容量型实例。

注意：创建实例时请谨慎选择规格类型，创建后无法修改。

两种实例规格在功能上保持一致，都能够支持单表 PB 级别的数据量，主要区别在于使用成本及适用场景。

高性能实例

高性能实例能够提供百万级读写 TPS，单行的读写操作的平均延时为单个毫秒，适用于对读写性能和并发都要求非常高的场景，例如游戏、金融风控、社交应用、推荐系统、舆情监控等。

容量型实例

容量型实例能够提供极高的写吞吐量，能够提供接近于高性能实例的写性能以及更低的使用成本，但读性能和并发能力均低于高性能实例，适用于写多读少、对读性能不敏感但对成本较为敏感的业务，例如日志监控数据、车联网数据、设备数据、时序数据以及物流数据。

注意：容量型实例不支持预留读/写吞吐量，所有的读写访问均依照按量读/写吞吐量进行计费。

各区域实例规格支持情况

地域名称	高性能实例	容量型实例
华东 1	支持	支持
华东 1 金融云	支持	暂不支持
华东 2	支持	支持
华东 2 金融云	暂不支持	支持
华北 2	支持	支持
华北 3	暂不支持	支持
华北 5	暂不支持	支持
华南 1	支持	支持
香港	暂不支持	支持
亚太东南 1（新加坡）	支持	暂不支持
美国东部 1（弗吉尼亚）	支持	暂不支持

美国西部 1（硅谷）	支持	暂不支持
亚太东北 1（东京）	暂不支持	支持
欧洲中部 1（法兰克福）	暂不支持	支持
中东东部 1（迪拜）	暂不支持	支持
亚太东南 2（悉尼）	暂不支持	支持
亚太东南 3（吉隆坡）	暂不支持	支持
亚太东南 5（雅加达）	暂不支持	支持
亚太南部 1（孟买）	暂不支持	支持

服务地址

每个实例对应一个服务地址（EndPoint），应用程序在进行表和数据操作时需要指定服务地址。

如果应用程序从公网访问表格存储，使用如下格式的服务地址：

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

如果应用程序从同区域的 ECS 服务器访问表格存储，使用如下格式的服务地址：

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

例如，华东 1 节点，实例名称为 myInstance 的服务地址为：

```
公网服务地址：https://myInstance.cn-hangzhou.ots.aliyuncs.com
内网服务地址：https://myInstance.cn-hangzhou.ots-internal.aliyuncs.com
```

应用程序从同区域的 ECS 服务器上通过内网访问表格存储，可以获得更低的响应延迟，也不产生外网流量。

如果应用程序从 VPC 网络的 ECS 服务器访问表格存储，使用如下格式的服务地址：

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

例如，华东 1 节点，应用程序从名称为 testVPC 的网络访问实例名称为 myInstance 的服务地址为：

```
VPC 访问地址：https://testVPC-myInstance.cn-hangzhou.vpc.ots.aliyuncs.com
```

该 VPC 访问地址只能支持来自于 testVPC 下 VPC 网络内的服务器的访问。

Stream

Stream是数据表的一个属性，可用于增量数据流的实时增量分析和数据增量同步。

在创建数据表时，您可以选择开启Stream功能。详细内容请参见Stream增量数据流。

Stream Expiration Time

Stream记录过期时长（Expiration Time）是增量数据的有效时间，在启用Stream功能时设置。

Expiration Time的单位为小时，最大可以设置为24小时。例如，期望过期时间为24小时，则Expiration Time应设置为24。

如果超过设置的有效时间，则表格存储会对这些增量数据进行清理，您不会查到过期数据，从而减少因为存储增量数据而占用的存储空间。

您可以关闭Stream后重新打开来修改Expiration Time。但重新打开后，之前的数据都无法被查询。

使用限制

以下为表格存储的限制项说明，部分限制范围表明用户能够使用的最大值，并不为建议值。为保证更好的性能，请合理设计表结构和单行数据大小。

限制项	限制范围	说明
一个阿里云用户账号下可以保有实例数	10	如有需求提高上限，请提交工单。
一个实例中表的个数	64	如有需求提高上限，请请提交工单。
实例名字长度	3-16 Bytes	字符集为 [a-z, A-Z, 0-9] 和连词符（-）,首字符必须是字母且末尾字符不能为连词符（-）。实例名称不能包含[‘ali’，‘ay’，‘ots’，‘taobao’，‘admin’] 这几个单词

表名长度	1-255 Bytes	字符集为 [a-z, A-Z, 0-9]和下划线（_），首字符必须是字母或下划线（_）。
列名长度限制	1-255 Bytes	字符集为 [a-z, A-Z, 0-9]和下划线（_），首字符必须是字母或下划线（_）。
主键包含的列数	1-4	至少 1 列，至多 4 列。
String 类型主键列列值大小	1 KB	单一主键列 String 类型的列列值大小上限 1 KB。
String 类型属性列列值大小	2 MB	单一属性列 String 类型的列列值大小上限 2 MB。
Binary 类型主键列列值大小	1 KB	单一主键列 Binary 类型的列列值大小上限 1 KB。
Binary 类型属性列列值大小	2 MB	单一属性列 Binary 类型的列列值大小上限 2 MB。
一行中属性列的个数	不限制	不限制单一行拥有的属性列个数。
一次请求写入的属性列的个数	1024 列	PutRow、UpdateRow 或 BatchWriteRow 操作时，单行写入的属性列的个数不能超过 1024 列。
单行数据大小	不限制	不限制单一行中所有列名与列值总和大小。
单表的预留读写吞吐量	0-5000	如有需求提高上限，请请提交工单。
读请求中 columns_to_get 参数的列的个数	0-128	读请求一行数据中获取的列的最大个数。
表级别操作 QPS	10	一个实例的表级别操作每秒不超过 10 次，表级别操作见表操作。
单表 UpdateTable 的次数	上调：无限制，下调：无限制	需要遵循单表的调整频率限制。
单表 UpdateTable 的频率	每 2 分钟 1 次	单表在 2 分钟之内，最多允许调整 1 次预留读/写能力值。
BatchGetRow 一次操作请求读取的行数	100	N/A
BatchWriteRow 一次操作请求写入行数	200	N/A
BatchWriteRow 一次操作的数据大小	4 MB	N/A
GetRange 一次返回的数据	5000 行或者 4 MB	一次返回数据的行数超过 5000 行，或者返回数据的数据大小大于 4 MB。满足以上任一条件时，超出上限的数据将会按行级别被截掉并返回下一行数据主键信

		息。
一次 HTTP 请求 Request Body 的数据大小	5 MB	N/A