表格存储

产品简介

产品简介

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。

并且，您在使用表格存储服务时，只需要按照预留和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。

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。

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。

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

表格存储对应用程序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，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。

按量付费

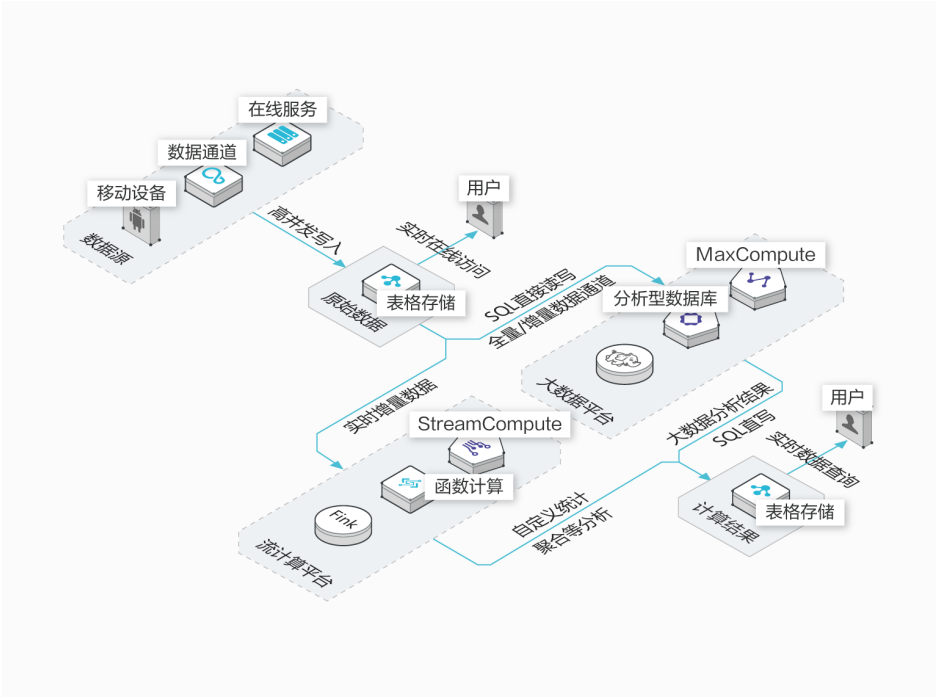
表格存储根据用户预留和实际使用的资源进行收费，起步门槛低。

监控集成

用户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。

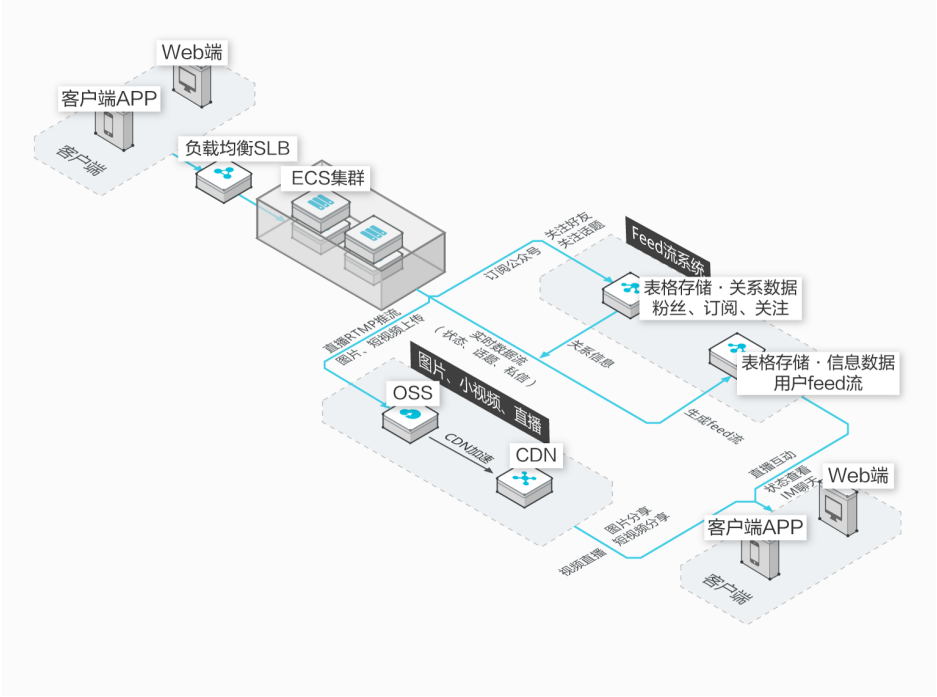
大数据存储与分析

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。



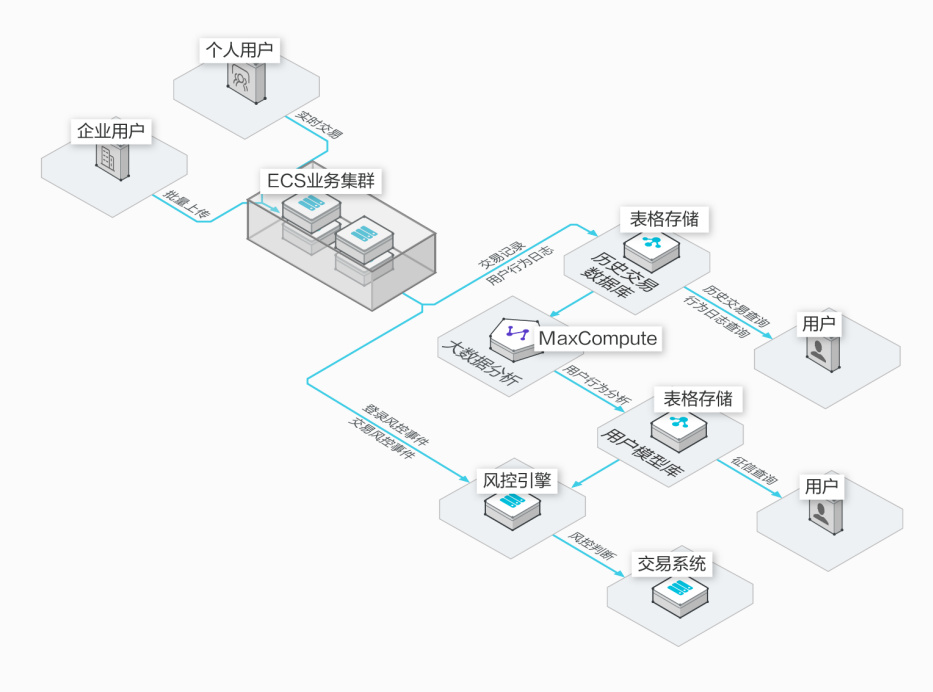
社交 Feed 流

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。



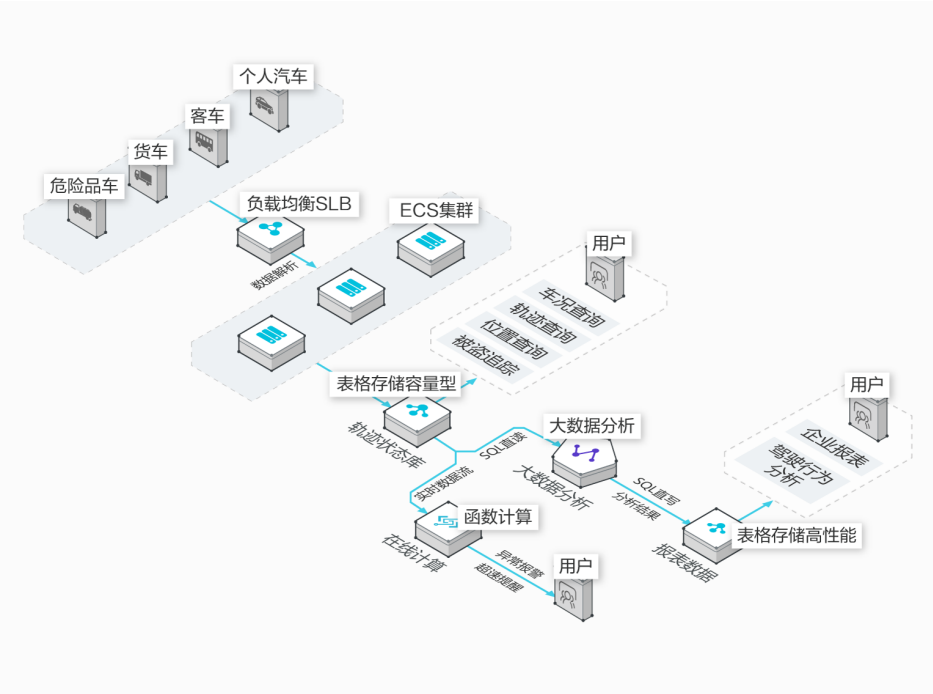
金融风控

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。



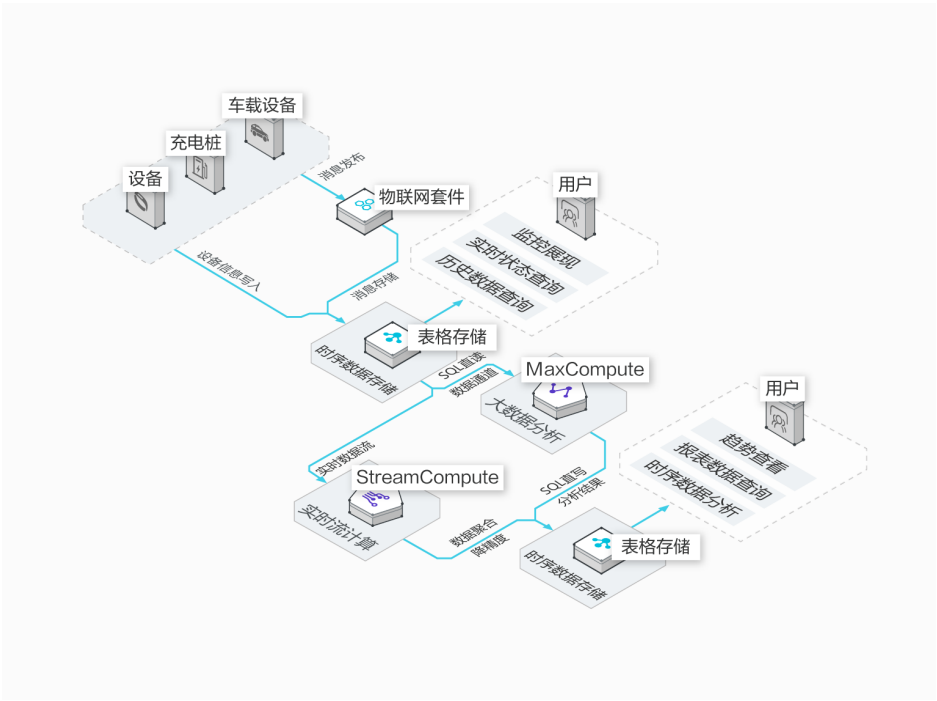
车联网数据

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。



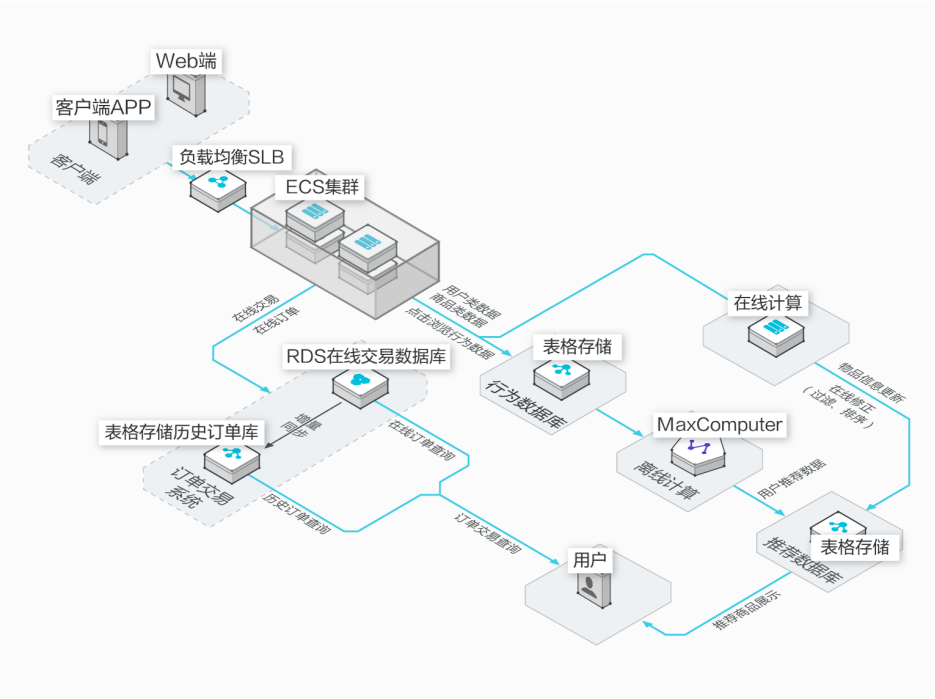
物联网时序数据

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

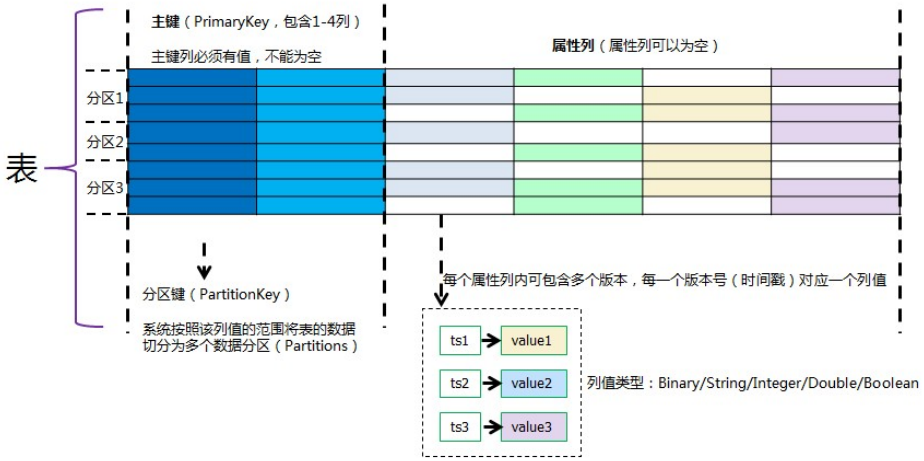
使用表格存储让您无需担心大量历史交易订单的数据规模与访问性能，配合大数据计算服务，轻松实现精准营销，弹性资源、按量付费，让您从容应对所有用户在线高峰时刻。



名词解释

概念介绍

表格存储的数据模型概念包括表、行、主键和属性，如下图所示。



表是行的集合，行由主键和属性组成。

主键列和属性列均由名称和值组成。

所有行都必须包含主键列，其主键列的数目和名称必须相同。

每行包含的属性列的数目可以不固定，名字和数据类型也可以不同。

与主键列不同，每个属性列可以包含多个版本，每个版本号（时间戳）对应一个列值。

说明：时间戳是从 1970-01-01 00:00:00 UTC 时间到当前写入时间的毫秒数。

示例

如下例子展示了同一张表中的两行，其中 ID 列为主键列：

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

ID 是表的主键，ID 为'4776'和'6555'的行拥有不同的属性，它们可以被存在一张表中。

ID 为'4776'行的Type属性列只有一个版本数据，版本号为1466676354000的数据为 'Book'。

ID 为'6555'行的Length属性列有两个版本数据，版本号为1466676354000的数据为'400'，版本号为1466762754000的数据为'500'。

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（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（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

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、数字和/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

如下例子为有效列名：

- _id
- Message

如下例子为无效列名：

- sn序列号_21 *（不能包含中文）*
- 5store *（首字母不能是数字）*
- shoping(new) *（不能包含除下划线以外的其他符号）*

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项。
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项。

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。

具有相同分区键的行属于同一个数据分区，一个分区可能包含多个分区键。分区键是最小的分区单位，一个分区键下的数据无法再做切分。为了防止分区过大无法切分，单个分区键下所有行的大小总和建议不超过 1 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。

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。

假如某数据表设置的预留读吞吐量为 100，某 1 秒内读操作实际消耗 120 读吞吐量，则这 1 秒内消耗的按量读吞吐量为 20。如果数据表设置的预留读吞吐量为 0，则该数据表上所有的读访问消耗的读吞吐量均为按量读吞吐量。

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。

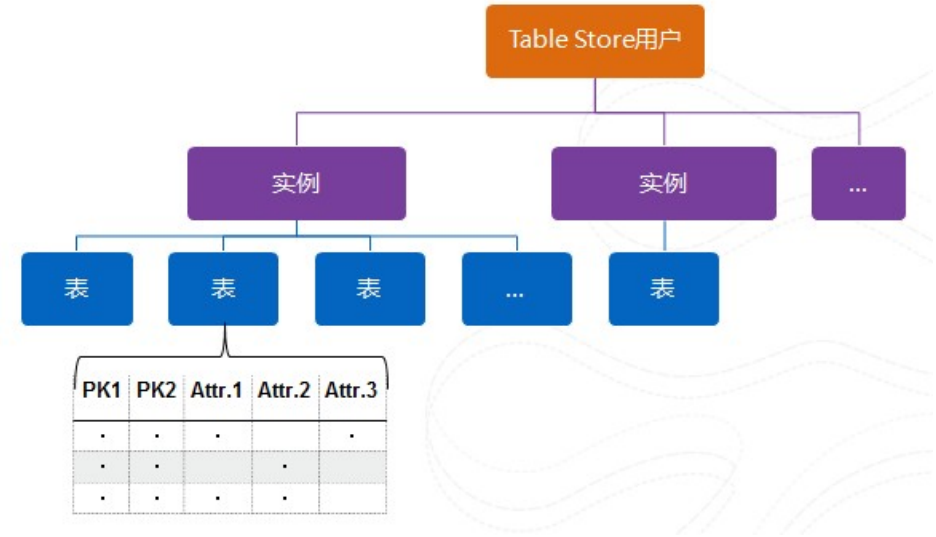
节点（Region）是指阿里云服务节点，表格存储服务会部署在多个阿里云服务节点内，用户可以根据自身的业务需求选择不同节点内的表格存储服务。

表格存储当前支持的服务节点与地域名称的对应关系如下表所示：

节点名称	地域名称（Region Name）
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
华南 1	cn-shenzhen
香港	cn-hongkong
亚太东南 1（新加坡）	ap-southeast-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
亚太东南 2（悉尼）	ap-southeast-2

实例（Instance）是用户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。用户在开通表

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。



用户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

实例的名称在单个节点内必须唯一，不同的节点内实例名称可以相同。具体命名规范如下：

必须由英文字母、数字和/或“-”组成

首字符必须为英文字母

末尾字符不能为“-”

大小写不敏感

长度在 3 Byte ~ 16 Byte 之间

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

节点名称	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
亚太东南 2 （悉尼）	暂不支持	支持

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：

```
http：//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：

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

```
http：//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：http：//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。