游戏数据库 TcaplusDB

快速入门

产品文档

腾讯云

文档目录

快速入门
了解基本概念
集群

最近更新时间：2022-05-13 10:34:13

集群概述

集群是 TcaplusDB 的基础管理单元，其负责为业务提供独立的 TcaplusDB 服务。
从游戏业务角度来看，一个集群可以为一个游戏的独立子模块提供服务，也可以为整体游戏提供服务。

创建集群

创建操作请参见 创建集群。

集群详细信息

用户可以通过集群详细信息查看到集群的配置和属性，用以对整个集群使用状况有整体的了解。
您可登录 TcaplusDB 控制台，单击集群ID，进入到集群详情页面，从而查看集群的详细信息。
其中涉及三个方面的主要内容：

基本信息

- 集群ID：集群的唯一 ID，此 ID 无法用于数据库连接，可使用此 ID 定位到集群，也可以使用此 ID 对集群进行访问控制。
- 接入ID：集群的连接 ID，通过 SDK 以及客户端工具访问数据库时，此 ID 用以标识集群。
- 连接协议：标识集群支持的表描述协议。当前支持 Protocol Buffers、TDR 协议。
- 集群名称：集群的名称，用户可根据集群的使用情况，对此名称进行自定义修改。
- 地域：集群所处地域。
- RESTful API：在同 VPC 子网场景下，通过 RESTful 接口访问数据库时所使用的地址。

网络信息

- 所属网络：当前集群所在的 VPC 网络信息。
- 所在子网：当前集群所在的子网。
- 内网地址：为当前集群所分配的内网 IP 地址，同 VPC 子网的云服务器可访问此 IP 地址。
- 内网端口：为当前集群所分配的访问端口号。

其他信息

- 创建时间：当前集群的创建时间。
- 连接密码：当用户忘记密码后，可通过 控制台 查看集群的访问密码。

表格组

最近更新时间：2022-05-13 11:22:13

表格组概述

表格组是 TcaplusDB 的逻辑隔离方案，其代表了一个数据分区，可将不同的表隔离开。一个集群可以包含多个表格组，一个表格组可以包含多个表格。不同表格组无法互相访问数据。

对于游戏业务角度来看，一个游戏业务支持全区全服，在一个集群之下，只对一个表格组进行读写，无需将游戏进行分区。同时也可以支持分区分服，即一个集群之下，包含多个表格组。

创建表格组

创建操作请参见 创建表格组。

表格组详细信息

用户可以通过集群列表页查看到表格组的配置和属性，用以对整个集群使用状况有整体的了解。
您可登录 TcaplusDB 控制台，进入集群列表，从而查看指定集群下的表格组信息。

表格组信息包含如下4种：

- ID：当前集群下此表格组的 ID 号，连接数据库时需要使用到此 ID。请注意不同集群下，表格组的 ID 可能会重复。
- 表格组名称：可自定义表格组名称，可根据具体表格组的实际作用为其命名。
- 表格数量：代表当前表格组之下包含多少个表格。
- 总容量：代表当前表格组下的表格共占据了多少磁盘容量。

表格

最近更新时间：2022-05-13 11:22:37

表格概述

表格是数据的集合，是业务数据根据表格的定义存放在一起的组合形式。
从游戏业务角度来看，一张表可负责一个业务模块的一段相关数据集合，例如用户表、道具表。

TcaplusDB 需要预设 Schema，同时提供两种表类型：Generic 表和 List 表。Generic 表一个 Key（1个 - 8个字段）记录对应一个 Value 记录，List 表一个 Key（1个 - 7个字段）记录对应多个 Value 记录。
List 表根据是否进行排序分为普通 List 表和 SortList 表：

- 对于普通 List 表，Tcaplus 内部是无序插入的，用户取出一个 Key 下的所有 Value 时，Value 内部是无序的。
- SortList 表按照某个指定的 Value 字段进行排序插入，用户可以按照从大到小或者从小到大的顺序进行取值。在 PB 和 TDR 表中都存在 Generic 表、 List 表、SortList 表这三种 Schema 模式，用户可以根据需求选择合适的模式。

如下示例的玩家表数据，该表可用于存储有关某一游戏中所有玩家的相关信息：游戏角色名、性别、种族、等级、战斗值、装备、坐骑、物品等信息。

```
{
player_id:11474,
player_name: "测试账号2",
gender:0,
ethnicity:"精灵",
FightingPower:10,
equipment:
{
helmet: 0,
Warframe: 0,
gloves: 0,
necklace: 0,
pants: 0,
Shoes: 0
},
horse: "0"
},
{
player_id:11475,
player_name: "测试账号1",
gender:1,
ethnicity:"兽人",
FightingPower:1477,
equipment:
{
helmet: 1478,
Warframe: 21,
gloves: 554,
necklace: 12,
pants: 64,
Shoes: 122
},
```

```
horse: "3"
}
```

记录

多个同一对象字段值的集合，称之为记录。上述示例的玩家角色表中，一个玩家角色的所有信息，由多个属性字段值组成，其所有信息表示这个玩家角色信息在此游戏的集合。单条记录支持最大1MB。

字段

字段描述了指定对象的某一特征。上述示例，玩家角色的一个独立属性我们称之为一个字段，如玩家角色表中的player_name、ethnicity。

主键字段

由如上示例表可看出，每条记录都有标志记录唯一性的字段，用于将表中的记录区分开来，我们称之为主键字段。上述示例的玩家角色表中，主键字段包含 player_id 以及 player_name。TcaplusDB 最大支持4个主键字段。

普通字段

每条记录除了标志其唯一性的字段外，还有其他的属性，这些属性所构成的字段，称之为普通字段。上述示例的玩家角色表中，普通字段则为 equipment，FightingPower 等属性。TcaplusDB 支持最多128个普通字段。

分片字段

TcaplusDB 是分布式数据库系统，可以为表格提供更高规格的并发请求能力。如果用户设置了分片字段，TcaplusDB 将自动根据当前表格访问情况，通过对分片字段 Hash 计算从而自动调整表格分片。如果用户不设置，那么系统将默认采用主键字段作为分片字段。

分片字段的设置与后台数据分布相关，用户需要评估作为分片字段的字段取值是否离散。取值有限的字段如性别、星期等都不建议设置为分片字段，建议使用 ID、name 类的字段作为分片字段。

字段类型

TcaplusDB 支持多种数据类型，支持的类型如下：

字段类型	字段介绍
int32	使用可变长度编码，此类型表示值介于 -2^31 到 +2^31 之间的有符号整数，但是负数表示效率较低。
int64	使用可变长度编码，此类型表示值介于 -2^63 到 +2^63 之间的有符号整数。
uint32	使用可变长度编码，该类型表示值为 0 到 2^32 的无符号整数。
uint64	使用可变长度编码，该类型表示值为 0 到 2^64 的无符号整数。
sint32	可变长度有符号的 int 值，与常规 int32 相比，可以更有效地表示负数。
sint64	可变长度有符号的 int 值，与常规 int64 相比，可以更有效地表示负数。
bool	布尔类型，表示真 Ture 或假 False。
fixed32	定长数字类型，始终占用四个字节。如果值通常大于 2^28，则比 uint32 更有效率。
fixed64	定长数字类型，始终占用八个字节。如果值通常大于 2^56，则比 uint64 更有效率。
sfixed32	定长数字类型，始终占用四个字节。
sfixed64	定长数字类型，始终占用八个字节。
float	浮点类型，浮点数占 4 个字节，用 32 位二进制描述。
double	双精度类型，浮点数占 8 个字节，用 64 位二进制描述。
string	字符串类型，只能为 UTF-8 编码或 7 位 ASCII 文本，并且长度不能超过 2^32。

字段类型	字段介绍
bytes	二进制类型，可以包含任意长度不超过 2^32 的字节序列。
message	嵌套类型，可连续嵌套，支持最高嵌套128层，嵌套层数过多会影响性能。

表格定义格式

TcaplusDB 支持 Protocol Buffers、TDR 数据存储格式，这是一种轻便高效的结构化数据存储格式，主要用于将结构化数据序列化存储。
详细关于表的格式可通过 表描述语言 查看。

Generic 表

基本特点

- 可以建索引，请参见 索引。
- 最多支持8个 Key 字段，256个 Value 字段。
- 单个 Key 最大1KB， 单个 Value 最大10MB，整条记录最大10MB。

TDR-Generic 表示例

```
<struct name="PLAYERONLINECNT" version="1" primarykey="TimeStamp,GameSvrID" splittablekey="TimeStamp">
<entry name="TimeStamp" type="uint32" desc="单位为分钟" />
<entry name="GameSvrID" type="string" size="64" />
<entry name="GameAppID" type="string" size="64" desc="gameapp id" />
<entry name="OnlineCntIOS" type="uint32" defaultvalue="0" desc="ios 在线人数" />
<entry name="OnlineCntAndroid" type="uint32" defaultvalue="0" desc="Android 在线人数" />
<entry name="BinaryLen" type="smalluint" defaultvalue="1" desc="数据来源数据长度；长度为0时，忽略来源检查"/>
<entry name="binary" type="tinyint" desc="二进制" count= "1000" refer="BinaryLen" />
<entry name="binary2" type="tinyint" desc="二进制2" count= "1000" refer="BinaryLen" />
<entry name="strstr" type="string" size="64" desc="字符串"/>
<index name="index_id" column="TimeStamp"/>
</struct>
```

PB-Generic 表示例

```
message pb_generic_index_shardingkey {
option(tcaplusservice.tcaplus_primary_key) = "openid,tconndid,timekey,svrid";
option(tcaplusservice.tcaplus_index) = "index_openid(openid)";
option(tcaplusservice.tcaplus_index) = "index_openid_tconndid(openid,tconndid)";
option(tcaplusservice.tcaplus_index) = "index_full_key(openid,tconndid,timekey,svrid)";

option(tcaplusservice.tcaplus_sharding_key) = "openid";

//4个key
required uint32 openid = 1; //QQ Uin
required string timekey = 2[(tcaplusservice.tcaplus_size) = 20, (tcaplusservice.tcaplus_desc) = "bingo"];
required int32 tconndid = 3;
required string svrid = 4;

//value
required string gamesvrid = 5[(tcaplusservice.tcaplus_size) = 11];
```

```
repeated property other_property= 6 ;//其他扩展属性
optional item_bag items = 7;
repeated int64 lockid = 8 [packed = true];
optional bytes val = 9;
optional pay_info pay = 10;
optional uint32 id_uint32 = 11;
optional int32 id_int32 = 12;
}
```

List表

基本特点

- 不能建索引。
- Key 字段最多只能有7个，Value 字段最多255个， 各有一个要预留给系统使用。
- List 支持方便的头尾操作, 适合邮件，留言板，战斗记录等场景。
- List 表需要指定单个 Key 记录下的最大元素个数（目前最大10000），超过最大元素个数时，可指定从头部或尾部删除老元素。
- 可以一次取出单个 Key 下的所有 Value，Value 内部的排列无序。
- Tcaplus 对 List 表的存储采用分级机制，包括：
  - 索引记录，Fullkey + (idx1,idx2,idx3,……,idxn)
  - 数据记录，[FullKey+idx1,value1][FullKey+idx2,value2][……][FullKey+idxn,valuen]

TDR-List表示例

```
<struct name="following_action_list" version="1" primarykey="game,myuin">
<entry name="game" type="uint64" defaultvalue="0" desc="游戏 ID"/>
<entry name="myuin" type="uint32" desc="QQ 号"/>
<entry name="actiontype" type="uint8" defaultvalue="0" desc="1-分享图片，2-赞图片，3-评论图片"/>
<entry name="uin2" type="uint32" desc="关注人QQ号"/>
<entry name="nick2" type="string" size="128" desc="关注人昵称"/>
<entry name="sex" type="uint8" defaultvalue="0" desc="关注人性别男0女1"/>
<entry name="pid" type="string" size="33" desc="关注人操作图片 ID"/>
<entry name="time" type="uint32" desc="时间"/>
<entry name="content" type="string" size="1024" desc="动态详细内容"/>
</struct>
```

PB-List 表示例

```
syntax = "proto3";

package myTcaplusTable;

import "tcaplusservice.optionv1.proto";

message tb_online_list {
option(tcaplusservice.tcaplus_primary_key) = "openid,tconndid,timekey";
option(tcaplusservice.tcaplus_customattr) = "TableType=LIST;ListNum=1900";

int32 openid = 1; //QQ Uin
int32 tconndid = 2;
```

```
string timekey = 3;
string gamesvrid = 4;
int32 logintime = 5 ;
repeated int64 lockid = 6;
pay_info pay = 7;

message pay_info {
uint64 total_money = 1;
uint64 pay_times = 2;
}
map<string, pay_info> projects = 8;
}
```

SortList 表

基本特点

最多允许4个排序字段，且在建表的xml（tdr）中指定，具体格式如下：

```
<struct name="following_action_list" version="1" primarykey="game,myuin" customattr="TableType=SORTLIST;SortRule=INSC;SortFieldNum=1">
<entry name="game" type="uint64" defaultvalue="0" desc="游戏 ID"/>
<entry name="myuin" type="uint32" desc="QQ 号"/>
<entry name="actiontype" type="uint8" defaultvalue="0" desc="1-分享图片，2-赞图片，3-评论图片"/>
<entry name="uin2" type="uint32" desc="关注人 QQ 号"/>
<entry name="nick2" type="string" size="128" desc="关注人昵称"/>
<entry name="sex" type="uint8" defaultvalue="0" desc="关注人性别男0女1"/>
<entry name="pid" type="string" size="33" desc="关注人操作图片 ID"/>
<entry name="time" type="uint32" customattr="sort1" desc="时间"/>
<entry name="content" type="string" size="1024" desc="动态详细内容"/>
</struct>
```

> 说明：
> 如上 customattr="TableType=SORTLIST;ListNum=1023;SortFieldNum=1" 表示该表格类型为 SORTLIST，SortRule=INSC 表示升序排列，SortFieldNum=1 表示排序字段有1个，customattr="sort1" 表示第一个排序字段。

使用限制

- 排序默认按照从小到大进行排序，在 ServiceApi 中可以指定是从小到大取还是从大到小取。
- 暂时不允许在表变更时从无序 List 变为 SortList，不允许从 SortList 变更为无序 List，不允许表变更更变换排序字段的顺序以及增减排序字段（均采用自己写 so 走表变更 Key-Value 方式实现）。
- 排序字段最大字节数8B，排序字段的类型：byte, uint16,uint32,uint64,int16,int32,int64,float,double【string(包含\0最多8B，暂时不支持)】。
- 排序是指 Value 字段参与排序，而不是 Key。

使用说明

- 排序
  当已有数据结构排好序后，再采用 ListAddAfter 进行数据插入时，采用插入排序效果最佳。
- 插入

- 对于 ListAddAfter，流程是：先看是否已经满了，如果满了且不允许淘汰元素，则插入失败，如果满了，允许淘汰，则删掉那个元素，并且获取一个 BiggestIndex，将新元素插在对应位置，并挪动其他元素。
- 对于 SortList，当用户进行 ListAddAfter 时，需要变更下 ServiceApi 里面的接口说明，因为我们默认是按照从小到大的顺序进行排序，当用户需要淘汰最后一个元素时，即淘汰最大的元素，当用户需要淘汰队头的元素时，即淘汰最小的元素。

TDR-Sortlist 表示例

```
<struct name="table_Sortlist_single" primarykey="key, name" customattr2="TableType=SORTLIST;ListNum=1023;SortFieldNum=1;SortRule=DESC" version="5" desc="用于 list 表遍历测试, 需要4个 shard, 建表 list 最大1023个元素" >
<entry name="key" type="uint32" desc="单个uint32作为KEY的时候, hashcode = key % 10000"/>
<entry name="name" type="int16" />
<entry name="level" type="uint32" />
<entry name="value1" type="string" size="102400" defaultvalue="" desc="最大长度:100KB"/>
<entry name="value2" type="string" size="102400" defaultvalue="" desc="最大长度:100KB"/>
<entry name="type_int8" type="int8" desc="type_int8" />
<entry name="type_uint8" type="uint8" desc="type_uint8"/>
<entry name="type_int16" type="int16" desc="type_int16" customattr2="sort1"/>
<entry name="type_uint16" type="uint16" desc="type_uint16"/>
<entry name="type_int32" type="int32" desc="type_int32"/>
<entry name="type_uint32" type="uint32" desc="type_uint32"/>
<entry name="type_int64" type="int64" desc="type_int64"/>
<entry name="type_uint64" type="uint64" desc="type_uint64"/>
<entry name="type_float" type="float" desc="type_float"/>
<entry name="type_double" type="double" desc="type_double"/>
<entry name="type_short" type="short" desc="type_short"/>
<entry name="type_string" type="string" desc="type_string" size="20"/>
<entry name="type_tinyint" type="tinyint" desc="type_tinyint"/>
<entry name="type_datetime" type="datetime" desc="type_datetime"/>
</struct>
```

PB-SortList 表示例

```
message pb_sortedlist {
option(tcaplusservice.tcaplus_primary_key) = "openid,tconndid,timekey,svrid";
option(tcaplusservice.tcaplus_customattr) = "TableType=SORTLIST;ListNum=1900;SortField=id_int32";

//4个key
required uint32 openid = 1; //QQ Uin
required string timekey = 2[(tcaplusservice.tcaplus_size) = 20, (tcaplusservice.tcaplus_desc) = "bingo"];
required int32 tconndid = 3;
required string svrid = 4;

//value
required string gamesvrid = 5[(tcaplusservice.tcaplus_size) = 11];
repeated property other_property= 6 ;//其他扩展属性
optional item_bag items = 7;
repeated int64 lockid = 8 [packed = true];
optional bytes val = 9;
optional pay_info pay = 10;
optional uint32 id_uint32 = 11;
```

```
optional int32 id_int32 = 12;
}
```

索引

最近更新时间：2022-05-13 11:22:55

索引概述

您可以在一个表上创建索引，利用索引，可以快速从数据库中获取您需要的数据，并且将为您的应用程序在数据查询方面提供更大的灵活性。

TcaplusDB 支持两种类型的索引：

- 本地索引：必须由主键字段构成，支持以主键作为条件进行快速查询。
- 全局索引：支持由除 message 类型之外的任意字段构成，可按照全局索引中配置的字段为条件进行快速查询。

TcaplusDB 最大支持1个全局索引，6个本地索引。

本地索引

特点

- 本地索引是实时索引，当插入或者删除数据时，会同时更新索引数据。
- 本地索引的字段必须包含在主键字段中，并且字段中还必须包含分片字段，因此，查询时最终只会落到一个数据分片上进行查询。
- 本地索引只支持等值查询。
- 一个表可以建立多个本地索引，查询时必须包含某一个本地索引的全部字段。
- 目前只有 generic 表支持本地索引。

本地索引只能在创建表时进行定义，并且只能由主键字段构成。所有索引字段集合的交集不能为空。

示例

例如一张表存在4个主键字段，如下示例的 HeroInfo 表，其中 heroId、heroName、heroFightingType、heroQuality 为此表的主键字段。

```
{
heroId:1,
heroName:"Arthur",
heroFightingType:1,
heroQuality:3
heroSkill:{
BasicSkill1:1,
BasicSkill2:2,
SpecialSkill:3
},
heroLevel:12,
heroskin:2,
heroAttackpower：141,
heroPhysicalDefense: 283,
heroMagicdefense：124
}
{
heroId:4,
heroName:"Shooter",
heroFightingType:3,
heroQuality:4
heroSkill:{
BasicSkill1:1,
```

```
BasicSkill2:2,
SpecialSkill:3
},
heroLevel:11,
heroskin:1,
heroAttackpower：225,
heroPhysicalDefense: 57,
heroMagicdefense：41
}
```

针对上表来说，可以根据4个主键字段自由组合而进行创建索引，但要求所有创建的索引交集不能为空。例如以 heroId 作为交集字段，进行索引的设计，就有如下几种组合方案：

- heroId,heroName
- heroId,heroFightingType
- heroId,heroQuality
- heroId,heroName,heroFightingType
- heroId,heroFightingType,heroQuality
- heroId,heroName,heroFightingType,heroQuality

同时您可以根据以上创建了索引的字段进行数据查询。在进行索引创建时，请根据业务的实际需求进行创建，否则会因为索引冗余而降低性能。

全局索引

在前面显示的 HeroInfo 表中，您可以根据已经创建的本地索引中所包含的字段为条件查询数据项。如果您还想要按 heroLevel 和 heroskin 等信息查询数据，可将 heroLevel 和 heroskin 字段加入到全局索引中，然后通过全局索引中所包含的字段进行数据查询。

请注意，全局索引同样也不支持嵌套类型，如上示例表中的 heroSkill 字段。

使用限制

本地索引使用限制

- 本地索引一旦创建，无法在使用期间修改、删除、新增、随表删除而删除。
- 本地索引个数最多不超过4个。
- 本地索引只支持精确匹配，即在用本地索引字段作为查询条件时，只能精确匹配到具体值，不支持模糊、范围匹配。

全局二级索引使用限制

- 全局二级索引只支持表一级字段，对于嵌套字段、数组列表类型字段不支持创建二级索引。
- 全局二级索引只支持 generic 类型表，对于 list 类型表不支持。
- 全局二级索引只支持在 tcaplus_client 工具、C++ SDK（TDR 协议表 & PB 协议表）中使用，暂不支持通过非 C++ 语言进行 API 层面的使用。

数据类型

最近更新时间：2022-05-13 11:23:15

TcaplusDB 支持的 Proto Buffers 数据类型。

proto3 数据类型与编程语言类型映射

.proto 类型	C++ 类型	Java 类型	Python 类型	Go 类型	Ruby 类型	C# 类型	PHP 类型	Dart 类型	描述
double	double	double	float	float64	float	double	float	double	–
float	float	float	float	float32	float	float	float	double	–
int32	int32	int	int	int32	fixnum/ bignum (根据需要)	int	integer	int	使用可变长度编码。负数编码效率低，如果您的字段可能具有负值，请改用 sint32。
int64	int64	long	int/long	int64	bignum	long	integer/ string	Int64	使用可变长度编码。负数编码效率低，如果您的字段可能具有负值，请改用 sint64。
uint32	uint32	int	int/long	uint32	fixnum/ bignum (根据需要)	uint	integer	int	使用可变长度编码。
uint64	uint64	long	int/long	uint64	bignum	ulong	integer/ string	Int64	使用可变长度编码。
sint32	int32	int	int	int32	fixnum/ bignum (根据需要)	int	integer	int	使用可变长度编码。

.proto 类型	C++ 类型	Java 类型	Python 类型	Go 类型	Ruby 类型	C# 类型	PHP 类型	Dart 类型	描述
sint64	int64	long	int/long	int64	bignum	long	integer/ string	Int64	使用可变长度编码。有符号的 int 值。与常规 int32 相比，它们能更有效地编码负数。
fixed32	uint32	int	int/long	uint32	fixnum/ bignum (根据需要)	uint	integer	int	始终为四个字节。如果值通常大于 2^28，则比 uint32 更有效。
fixed64	uint64	long	int/long	uint64	bignum	ulong	integer/ string	Int64	始终为八个字节。如果值通常大于 2^56，则比 uint64 更有效。
sfixed32	int32	int	int	int32	fixnum/ bignum (根据需要)	int	integer	int	始终为四个字节。
sfixed64	int64	long	int/long	int64	bignum	long	integer/ string	Int64	始终为八个字节。
bool	bool	boolean	bool	bool	trueClass/ falseClass	bool	boolean	bool	-
string	string	String	str/unicode	string	string (UTF -8)	string	string	string	字符串必须始终包含 UTF-8 编码或 7位 ASCII 文本，并且不能超过 2^32。
bytes	string	ByteString	str	byte	string (ASCII-8BIT)	bytestring	string	-	可以包含不超过 2^32个任意字节序列。

proto2 数据类型与编程语言类型映射

.proto 类型	C++ 类型	Java 类型	Python 类型	Go 类型	描述
double	double	double	float	*float64	-
float	float	float	float	*float32	-
int32	int32	int	int	*int32	使用可变长度编码。负数编码效率低，如果您的字段可能具有负值，请改用 sint32。
int64	int64	long	int/long	*int64	使用可变长度编码。负数编码效率低，如果您的字段可能具有负值，请改用 sint64。
uint32	uint32	int	int/long	*uint32	使用可变长度编码。
uint64	uint64	long	int/long	*uint64	使用可变长度编码。
sint32	int32	int	int	*int32	使用可变长度编码。有符号的 int 值。与常规 int32 相比，它们能更有效地编码负数。
sint64	int64	long	int/long	*int64	使用可变长度编码。有符号的 int 值。与常规 int64 相比，它们能更有效地编码负数。
fixed32	uint32	int	int/long	*uint32	始终为四个字节。如果值通常大于2^28，则比 uint32 更有效。
fixed64	uint64	long	int/long	*uint64	始终为八个字节。如果值通常大于2^56，则比 uint64 更有效。
sfixed32	int32	int	int	*int32	始终为四个字节。
sfixed64	int64	long	int/long	*int64	始终为八个字节。
bool	bool	boolean	bool	*bool	-
string	string	String	unicode(Python 2) 或 str(Python 3)	*string	字符串必须始终包含 UTF-8 编码或7位 ASCII 文本。
bytes	string	ByteString	bytes	byte	可以包含任意字节序列。

读写容量模式

最近更新时间：2022-05-13 11:24:09

读写容量模式概述

TcaplusDB 使用读写容量模式来计算表的读写请求量。

容量单元（CU，Capacity Unit）

指访问资源的最小请求单位，是请求数据读和请求数据写的最小统计单元。

读容量单元（RCU，Read Capacity Unit）

读容量单元是读取请求的最小单元，一次请求所产生的数据响应可能包含多个读容量单位，一个读容量单位最大值为4KB，即一个读取请求单位表示对大小最多为4KB的记录执行一次读取请求。如果需要对大于4KB的数据进行读取请求，则需要额外的请求单位。

读容量单元的基数为4KB。未满4KB的单次请求以4KB计算，超过4KB的请求以4KB的倍数算，余数均向4取整为4KB。

示例如下：
读请求容量
用户发送了一个请求，需要查询的某条记录中的某个字段，此记录共10KB，此时数据库将返回此记录的指定字段值给用户。此时响应的数据为10KB，计算为3RCU。

写容量单元（WCU，Write Capacity Unit）

写容量单元是写入请求的最小单元，一次写请求所产生的数据写入容量可能包含多个写容量单元，单个写容量单元最大值为4KB，即每写入4KB的数据则记作一次写请求单位。

写容量单元的基数是4KB，单次请求写入容量不超过4KB的以4KB计算。超过4KB的写入请求，小数均向上取整。
当用户执行写操作时，可选择是否开启写操作返回功能（Result Flag），当用户开启此功能时，系统将自动计算返回的数据容量作为WCU。
消耗 WCU 数 = 数据写所产生的 CU + 返回 CU

示例如下：

- **写请求**
  一个用户发送了一个请求，需要更新5KB的内容，后台将需要更新的数据所在的记录从磁盘读取到内存（共50KB），
  然后更新相关记录，完成更新后，将50KB内容刷新到磁盘当中。
  如果用户开启了写操作返回记录，将返回5KB的内容给用户。
- **开启写操作返回**
  产生了 13CU（50KB写盘）+ 2CU（返回的5KB）= 15WCU
- **未开启写操作返回**
  产生了13CU（50KB写盘）= 13WCU

预置模式

如果您选择预置模式，则需要根据访问变化调整表的预置读写容量。此操作可帮助您控制对 TcaplusDB 的使用，使之保持或低于定义的请求速率，以便获得成本可预测性。
为了更好地配置预置容量，您需要对应用程序流量进行提前预估。

预留读

指对表格进行读取容量单元的预配置，您需要预估此表格当日的最高的每秒RCU消耗，然后设置此值为预留RCU。

预留写

指对表格进行写入容量单元的预配置，您需要预估此表格当日的最高的每秒WCU消耗，然后设置此值为预留WCU。

预留容量

指对表格进行占用磁盘容量的预配置，您需要预估此表格当日的容量大小。

请求限制与超出配额

针对用户因 RCU 或者 WCU 配置得过低，又因突然增长的访问请求场景，TcaplusDB默认将对超过预配置 RCU 和预配置 WCU 40%的请求进行限制，此限制并非限制请求无法访问，而是对请求进行排队，降低访问频率，程序可能会出现超时现象。用户可通过监控告警功能得知当前请求数超过预配置能力。

限制会避免您的应用程序消耗太多容量单位。当请求受到限制时，请求将失败并出现大量系统错误。

您可以使用 TcaplusDB 表格监控监视实际读写容量单元，并在必要时修改表格配额。

Protobuf 表定义

最近更新时间：2022-05-13 11:24:35

TcaplusDB 支持2种表定义格式：Protobuf（Protocol Buffers，PB）表、TDR（Tencent Data Representation）表。
两者在使用上没有根本性的变化，请根据具体业务使用习惯选择表定义语言。

- Protobuf 是 Google 开发的一种描述性语言，针对结构化数据进行序列化，同时强调数据结构的简单化和性能。
- TDR 是由腾讯开发的跨平台数据表示语言，结合了 XML、二进制、ORM（对象关系映射）的优势，在腾讯游戏数据的序列化场景中广泛使用。

本文主要描述 Protobuf 表定义语言的定义形式。

Protobuf 表

TcaplusDB 创建表，首先需要使用表描述语言定义表的格式，将表格定义内容写入文件中，此文件被称之为表的 IDL 描述文件。
Protobuf 表的 IDL 描述文件根据 Protocol Buffers 的规则来进行表格定义。

主要分为以下3类定义：

- 文件定义信息
- 表定义信息
- 嵌套类型信息

文件定义信息

文件定义信息区域主要定义当前表描述文件的公共信息，主要涉及如下3种类型的内容：

选项名称	功能说明	值示例	必填
syntax	指明当前文件书写的语法规范版本。	支持 proto2、proto3	是
package	指明当前文件自定义包名，包名可以避免对 message 类型之间的名字冲突。	包名信息	是
import	引入 Tcaplus 表的一些公共信息，必须在您的表定义中被引用。	tcaplusservice.optionv1.proto	是

表定义信息

表定义信息主要通过 message 定义表的格式，在 message 中可对表的扩展信息进行定义，也可以对表的字段信息进行定义。

扩展信息定义

TcaplusDB 表定义可以在 protobuf 语法基础上通过 option 进行扩展，可以实现更丰富的语义功能，可定义的内容如下表所示。
详细的定义格式为： option(tcaplusservice.选项) = "值";

选项名称	功能说明	设置示例	必填
tcaplus_primary_key	设置 TcaplusDB 表主键字段	option(tcaplusservice.tcaplus_primary_key) = "uin,name,region";	是
tcaplus_index	设置 TcaplusDB 表索引键字段	option(tcaplusservice.tcaplus_index) = "index_1(uin,region)";	否
tcaplus_sharding_key	用户可以自定义表分片键	option(tcaplusservice.tcaplus_sharding_key) = "uin";	否

选项名称	功能说明	设置示例	必填
tcaplus_field_cipher_suite	如需使用字段加密功能，请参考设置示例进行设置；如果用户需要指定自己的加密算法，请参考 API 中的例子	option(tcaplusservice.tcaplus_field_cipher_suite) = "DefaultAesCipherSuite";	否
tcaplus_cipher_md5	如需使用字段加密功能，需要设置用户侧保存加密密码字符串的 MD5	option(tcaplusservice.tcaplus_cipher_md5)= "62fee3b53619b7f303c939964c6f2c4b";	否

字段信息定义

TcaplusDB 定义字段的格式为： 字段修饰符 字段类型 字段名称 = 标识号[特殊定义];

字段修饰符

proto2 支持3种类型的限定修饰符，proto3 不再支持 required 修饰，默认为 optional 类型。

- required：表示字段为必填字段，proto2 中主键字段必须被 required 修饰。
- optional：表示此字段为可选字段，支持设定字段的默认值。
- repeated：表示该字段可以包含0 – N个元素，其特性和 optional 一样，但是每一次可以包含多个值，可看作是在传递一个数组的值，必须指定 [packed = true] 的特殊定义。

字段类型

TcaplusDB 支持普通字段以及嵌套型字段，详细的字段介绍可参考 字段类型。

字段名称

针对当前属性而进行对字段进行命名，支持大小写字母，数字与下划线。建议字段的命名采用驼峰式命名方式，不能以数字开头。

标识号

标识号使用范围：[1,2^29 – 1]，不可使用 [19000 – 19999] 标识号，因为 Protobuf 协议实现中对这些标识号进行了预留，若使用，则会报错。
每个字段在进行编码时都会占用内存，而占用内存大小取决于标识号：

- 范围 [1,15] 标识号的字段在编码时占用1个字节。
- 范围 [16,2047] 标识号的字段在编码时占用2个字节。

特殊定义

- 当 repeated 修饰的字段需要指定 packed=true 选项。用法如下：

```
repeated int64 lockid = 6 [packed = true];
```

- 使用 optional 修饰的字段可以通过 default = 1 指定其默认值。用法如下：

```
optional int32 logintime = 5 [default = 1];
```

- 字段类型为 string 和 bytes 类型的字段可以指定为加密字段。用法如下：

```
required string name = 2[(tcaplusservice.tcaplus_crypto) = true];
```

嵌套类型信息

TcaplusDB 支持嵌套类型，嵌套类型可以包含另一个嵌套类型作为其字段，也可以在嵌套类型内定义一个新的嵌套类型。
嵌套类型的定义与表的定义相似，均是通过 message 进行声明，但是结构体中不能包含扩展信息定义，如“主键”，“索引”等定义。
TcaplusDB 支持最多128层连续嵌套，但是不建议大量使用嵌套，嵌套层数过多会导致数据访问性能降低。

proto2 表描述文件示例

以下是一个符合 proto2 语法规范的表描述文件：

```
syntax = "proto2"; // 指明符合 proto2 语法规范
package myTcaplusTable; // 自定义包名

import "tcaplusservice.optionv1.proto"; // 这个文件定义了 Tcaplus 表的一些公共信息，需要在您的表定义中被引用(import)

message player { //使用 message 定义表，message 名即表名

// 必须是指定了主键选项（tcaplusservice.tcaplus_primary_key）的 message 才能被识别为一张 TcaplusDB 业务数据表，否则此 message 只是一个结构体
// 主键选项指定了主键字段名列表，字段名用逗号分割，最多四个字段能够被指定为主键
option(tcaplusservice.tcaplus_primary_key) = "uin,name,region";

// tcaplusservice.tcaplus_index 选项指定 Tcaplus 索引
// 每个索引所包含的索引键必须是主键，并且所有索引字段集合的交集不能为空
option(tcaplusservice.tcaplus_index) = "index_1(uin,region)";
option(tcaplusservice.tcaplus_index) = "index_2(uin,name)";

//option(tcaplusservice.tcaplus_sharding_key) = "uin"; 用户可以自己显式设置索引分片字段。如果用户不显式设置，那系统将默认采用主键字段作为分片字段

option(tcaplusservice.tcaplus_field_cipher_suite) = "DefaultAesCipherSuite"; // 使用默认的 DefaultAesCipherSuite 加密函数，非必须设置选项
option(tcaplusservice.tcaplus_cipher_md5)= "62fee3b53619b7f303c939964c6f2c4b"; //设置加密密码字符串的 md5 值，非必须设置选项

// 接下来是表的字段定义
// TcaplusDB 表支持以下的数据类型：
// 非嵌套类型：int32, int64, uint32, uint64, sint32, sint64, bool, fixed64, sfixed64, double, fixed32, sfixed32, float, string, bytes
// 嵌套类型：message
// Tcaplus 支持 REQUIRED, OPTIONAL 和 REPEATED 作为字段修饰符

// 主键字段 (最多4个)
required int64 uin = 1; // 主键字段必须被 required 类型修饰，并且不能是嵌套类型
required string name = 2[(tcaplusservice.tcaplus_crypto) = true]; //message 中 string 和 bytes 类型的字段可以指定为加密字段，非必须设置选项
required int32 region = 3;
// 一张表最多只能包含4个主键字段

// 普通值字段
required int32 gamesvrid = 4; // 普通字段可以被 required,optional 或 repeated 类型修饰
optional int32 logintime = 5 [default = 1];
repeated int64 lockid = 6 [packed = true]; // repeated 修饰的字段需要指定 packed=true 选项
optional bool is_available = 7 [default = false]; //optional 类型字段可以指定默认值
optional pay_info pay = 8; // 值字段的类型可以是自定义的结构体类型
}

message pay_info { //使用 message 定义结构体
```

```
required int64 pay_id = 1;
optional uint64 total_money = 2;
optional uint64 pay_times = 3;
optional pay_auth_info auth = 4;

message pay_auth_info { // 结构体类型支持嵌套定义
required string pay_keys = 1;
optional int64 update_time = 2;
}
}
```

proto3 表描述文件示例

以下为一个符合 proto3 语法规范的表描述文件：

```
syntax = "proto3"; // 指明符合 proto3 语法规范
package myTcaplusTable; // 自定义包名

import "tcaplusservice.optionv1.proto"; // 这个文件定义了 Tcaplus 表的一些公共信息，需要在您的表定义中被引用(import)

message player { //使用 message 定义表，message 名即表名

// 必须是指定了主键选项（tcaplusservice.tcaplus_primary_key）的 message 才能被识别为一张 TcaplusDB 业务数据表，否则此 message 只是一个结构体
// 主键选项指定了主键字段名列表，字段名用逗号分割，最多四个字段能够被指定为主键
option(tcaplusservice.tcaplus_primary_key) = "uin,name,region";

// tcaplusservice.tcaplus_index 选项指定 Tcaplus 索引
// 每个索引所包含的索引键必须是主键，并且所有索引字段集合的交集不能为空
option(tcaplusservice.tcaplus_index) = "index_1(uin,region)";
option(tcaplusservice.tcaplus_index) = "index_2(uin,name)";

//option(tcaplusservice.tcaplus_sharding_key) = "uin"; 用户可以自己显式设置分片字段，如果用户不显式设置，那系统将默认采用主键字段为分片字段

option(tcaplusservice.tcaplus_field_cipher_suite) = "DefaultAesCipherSuite"; // 使用默认的 DefaultAesCipherSuite 加密函数，非必须设置选项
option(tcaplusservice.tcaplus_cipher_md5)= "62fee3b53619b7f303c939964c6f2c4b"; //设置加密密码字符串的 md5 值，非必须设置选项

// 接下来是表的字段定义
// Tcaplus 表支持以下的数据类型：
// 非嵌套类型：int32, int64, uint32, uint64, sint32, sint64, bool, fixed64, sfixed64, double, fixed32, sfixed32, float, string, bytes
// 嵌套类型：message

// 主键字段（最多4个）
int64 uin = 1; // 主键字段必须是非嵌套类型
string name = 2[(tcaplusservice.tcaplus_crypto) = true]; //message 中 string 和 bytes 类型的字段可以指定为加密字段，非必须设置选项
```

```
int32 region = 3;
// 一张表最多只能包含4个主键字段

// 普通值字段
int32 gamesvrid = 4;
int32 logintime = 5;
int64 lockid = 6;
bool is_available = 7;
pay_info pay = 8; // 值字段的类型可以是自定义的结构体类型
}

message pay_info { //使用 message 定义结构体

int64 pay_id = 1;
uint64 total_money = 2;
uint64 pay_times = 3;
pay_auth_info auth = 4;

message pay_auth_info { // 结构体类型支持嵌套定义
string pay_keys = 1;
int64 update_time = 2;
}
}
```

TDR 表定义

最近更新时间：2022-05-13 11:25:10

TcaplusDB 支持2种表定义格式：Protobuf（Protocol Buffers，PB）表、TDR（Tencent Data Representation）表。

- Protobuf 是 Google 开发的一种描述性语言，针对结构化数据进行序列化，同时强调数据结构的简单化和性能。
- TDR 是由腾讯开发的跨平台数据表示语言，结合了 XML、二进制、ORM（对象关系映射）的优势，在腾讯游戏数据的序列化场景中广泛使用。两者在使用上没有根本性的变化，请根据具体业务使用习惯选择表定义语言。

本文主要描述 TDR 表定义语言的定义形式。

TDR 表

TDR 支持通用（generic）表和列表（list）表。可以在一个 xml 文件中创建不同类型的表。

- generic 表是以表的形式表示元素属性的表，例如，学生，雇主，游戏玩家。
- list 表是一系列记录，例如游戏排行榜，游戏中的邮件（通常是最近的100封邮件）。

表定义信息

- 元素 metalib 是 xml 文件的根元素。
- 属性 tagsetversion 应该始终为1。
- 包含 primarykey 的 struct 元素是一个表，不包含 primarykey 的 struct 元素为一个普通结构体。
- 每次修改表结构时，版本属性值需要相应地加1，初始版本始终为1。
- primarykey 属性指定主键字段；对于 generic 表，您最多可以指定4个主键字段，对于 list 表，则可以指定3个。
- splittablekey 属性等效于分片键（shard key），TcaplusDB 表被拆分存储到多个存储节点。splittablekey 必须是主键字段之一，一个好的 splittablekey 应该具有高度分散性，这意味着值的范围很广，建议选用字符串类型。
- desc 属性包含当前元素的描述。
- entry 元素定义一个字段，支持的值类型包括 int32，string，char，int64，double，short 等。
- index 元素定义一个索引，该索引必须包含 splittablekey。由于可以使用主键查询表，因此索引不应与主键属性相同。

TDR 表描述文件示例

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>

<metalib name="tcaplus_tb" tagsetversion="1" version="1">

<!-- generic_table `users`, store the user' information -->
<!-- an user may has many roles -->
<struct name="users" version="1" primarykey="user_id,username,role_id" splittablekey="user_id" desc="user table">
<entry name="user_id" type="uint64" desc="user id"/>
<entry name="username" type="string" size="64" desc="login username"/>
<entry name="role_id" type="int32" desc="a user can have multiple roles"/>

<entry name="level" type="int32" defaultvalue="1" desc="role's level"/>
<entry name="role_name" type="string" size="1024" desc="role's name"/>
<entry name="last_login_time" type="string" size="64" defaultvalue="" desc="user login timestamp"/>
<entry name="last_logout_time" type="string" size="64" defaultvalue="" desc="user logout timestamp"/>

<index name="index1" column="user_id"/>
```

```
</struct>

<!-- list_table `mails`, store the role's mails -->
<struct name="mails" version="1" primarykey="user_id,role_id" desc="mail table">
<entry name="user_id" type="uint64" desc="user id"/>
<entry name="role_id" type="int32" desc="a user may has many roles"/>

<entry name="text" type="string" size="2048" desc="mail text"/>
<entry name="send_time" type="string" size="64" defaultvalue="" desc="timestamp of the mail sent"/>
<entry name="read_time" type="string" size="64" defaultvalue="" desc="timestamp of the mall read"/>
</struct>

</metalib>
```

另外，您可以使用 union 创建嵌套类型。

union 元素包含原始类型的集合，例如整数和字符串，可以将 union 作为自定义类型来引用。
macro 标签用于定义常量。

```
<macro name="DB_MAX_USER_MSG_LEN" value="301" desc="Max length of the message that user can define"/>
<union name="DBPlayerMsg" version="1" desc="DB Player message">
<entry name="SysMsgID" type="uint8" desc="Message ID" />
<entry name="UsrMsg" type="string" size="DB_MAX_USER_MSG_LEN" desc="player created message" />
</union>
```

创建集群

最近更新时间：2022-05-13 14:15:48

操作场景

本文为您介绍如何通过 TcaplusDB 控制台创建集群。

前提条件

已 注册腾讯云账号，并 实名认证成功。

操作步骤

1. 登录 TcaplusDB 控制台，在左侧导航选择**集群列表**页，单击**创建集群**。
2. 在弹出的新建集群用对话框，配置集群信息后，单击**立即购买**。
   - **网络**：选择私有网络 VPC 及子网 Subnet，每个集群只能在创建时选定私有网络和子网，创建完成后不能修改。
   - **连接协议**：选择 TcaplusDB 集群连接协议（数据描述协议）。
   - **集群名称**：同账号下集群名称不能重复，长度要求[1-32]个字符，可随时修改集群名称。
   - **访问密码**：设置集群访问密码，其中要求如下：
     - 长度[8-64]位，推荐使用12位以上的密码
     - 不能以"/"开头
     - 至少包含以下类型的3种：
       - 小写字母a - z
       - 大写字母A - Z
       - 数字0 - 9
       - 特殊符号\~!@#$%^&*_-+=\|(){}[]:;'<>,.?/
   - **表格组名称、表格组ID**：为此集群默认创建一个表格组，并制定其名称，表格组ID可以自动生成或自定义设置。
3. 购买完成后，返回集群列表可查看创建完的集群和表格组，系统会为每个集群分配一个唯一的集群 ID。

创建表格组

最近更新时间：2022-05-13 14:16:08

操作场景

本文为您介绍如何通过 TcaplusDB 控制台创建表格组。

前提条件

已创建 TcaplusDB 集群。

操作步骤

1. 登录 TcaplusDB 控制台，在左侧导航选择**集群列表**页，选择需要创建表格组的集群，单击**新建表格组**。

2. 在弹出的对话框，配置表格组信息。单击**新建表格组**，系统将会提示创建成功。
   - **选择集群**：可切换集群。
   - **表格组名称**：长度要求4个 - 64个字符。
3. 返回表格组列表，可查看表格组的名称、表格组 ID、表格数量、总容量等信息。

创建表格

最近更新时间：2022-05-13 14:16:23

操作场景

本文为您介绍如何通过 TcaplusDB 控制台创建数据表格。

前提条件

- 已创建 TcaplusDB 集群 和 表格组。
- 已根据 表描述文件示例 准备好表格文件。

操作步骤

1. 登录 TcaplusDB 控制台，在左侧导航选择**表格列表**页，单击**新建表格**。
2. 在新建表格页面，配置表格信息。
   - **集群、表格组**：选择目标集群和表格组。
   - **表格文件**：从本地上传或从已经上传过的历史文件中选择表定义文件，用户可以同时选择新上传和已上传的历史文件来创建表格，不能上传或选择同名文件。可参考 ProtoBuf 协议的示例表格文件 game_players.proto。
   - **备注**：输入表格备注信息。

3. 单击**下一步**，系统将校验用户选定的表格定义文件，
   - 校验不通过，将返回错误，请根据错误提示修改文件后重新上传。
   - 校验通过，将显示文件中定义的表格元信息，请继续执行下一步。
4. 在配置表格页，勾选要创建的表格，输入容量、预留读和预留写参数，系统将自动计算表格每日的花费价格。
5. 确认表格信息无误后，单击**创建**，系统返回创建成功提示。

获取连接信息

最近更新时间：2021-10-21 12:07:01

本文为您介绍如何通过 TcaplusDB 控制台获取连接信息。

在 集群列表 页，可直接看到集群部分信息。集群简介页中的接入ID、连接密码、内网地址和内网端口信息，是使用客户端连接 TcaplusDB 时的重要参数。

- 接入ID：集群所属的接入 ID。
- 连接密码：应用密码信息。
- 内网地址：TcaplusDB 连接地址。
- 内网端口：TcaplusDB 连接端口。

访问 TcaplusDB

最近更新时间：2022-02-11 10:17:31

TcaplusDB 可以通过多种方式进行访问读取，如客户端工具、各语言 SDK 工具包、RESTful 接口。

通过 client 客户端工具访问 TcaplusDB 数据

支持通过 Linux 客户端工具 tcaplus_client 访问 TcaplusDB 数据。

通过 C++ SDK 接口访问 TcaplusDB 数据

支持 C++ 语言 通过 SDK 工具访问 TcaplusDB 数据。

通过 RESTful API 接口访问 TcaplusDB 数据

可以通过 RestFul 接口访问 TcaplusDB 数据，详细可参考 RESTFul API 接口说明。

- 可参考 Python 接口说明，使用 Python 语言通过 RESTful 接口访问 TcaplusDB 数据。
- 可参考 PHP 接口说明，使用 PHP 语言通过 RESTful 接口访问 TcaplusDB 数据。
- 可参考 Go 接口说明，使用 Go 语言通过 RESTful 接口访问 TcaplusDB 数据。
- 可参考 Java 接口说明，使用 Java 语言通过 RESTful 接口访问 TcaplusDB 数据。