ChunkServer索引元信息数据结构

# 简述

每个ChunkServer上的tablets信息都被写入索引文件，ChunkServer在启动时，会读取这个索引文件，装载本机所属的所有tablets信息。

每日合并时，扫描每个tablet,并生成新的sstable文件，这时生成新的tablet索引文件。CS在切换过程中，读取新的索引文件和新的tablets完成切换过程。

# 索引信息数据结构

索引信息在内存中表现为一个排序数组，数组中每个元素是一个Tablet结构。

还包括一个timestamp，表示当前tablets的版本信息。

**class** ObTabletImage

{

**public**:

**int** **acquire**(**const** ObRange& range, ObTablet\* &tablet);

**int** **release**(ObTablet\* tablet);

**private**:

ObSortedVector<ObTablet\*> tablet\_list\_;

int64\_t timestamp;

};

每个Tablet元素的结构，包含Tablet自身的range,和包含的sstable file.(每个tablet包含多个sstable)

**struct** Range

{

uint64\_t table\_id\_;

ObString start\_key\_;

ObString end\_key\_;

ObBorderFlag border\_flag\_;

};

ObSSTableId表示一个sstable file对象，其中sstable\_file\_id是文件对应的id,也是文件名，sstable\_file\_offset是sstable在该文件中所在的偏移(一个sstable file中可能包含多个sstable)

sstable\_file\_id的最后一个字节，是文件所在的磁盘号(disk\_no)。

**struct** ObSSTableId

{

uint64\_t sstable\_file\_id\_;

uint64\_t sstable\_file\_offset\_;

};

**class** ObTablet

{

**private**:

ObRange range\_;

ObSSTableId sstable\_list\_[MAX\_SSTABLE\_PER\_TABLET];

};

# 序列化格式

CS上有多个磁盘，所有的sstable file是均匀分布在各个磁盘上的；ObTabletImage序列化后会在每个磁盘上生成一个索引文件，包含该磁盘的上存放的所有sstable文件信息。

注意：tablet不跨磁盘，如果一个tablet中包含多个sstable ，那么这些sstable都会在一个磁盘上。

每个磁盘上的索引文件格式相同，格式如下:

一个通用的记录头，包含索引文件本身的一些信息；

紧接着一个索引信息头，包含了tablet的个数，和timestamp

紧接着一个tablet record array,记录了每个tablet的range, sstable信息

最后一个是rowkey char stream，记录所有tablet range 的start key和end key

|  |
| --- |
| record header |
| tablet index header |
| tablet record array[tablet\_count] |
| rowkey char stream |

## 通用的记录头信息

struct ObRecordHeader

{

int64\_t magic\_; *// magic number*

int16\_t header\_length\_; *// header长度*

int16\_t version\_; *// 版本号*

int16\_t header\_checksum\_;*// header checksum*

int16\_t reserved\_; *// 保留，必须置0*

int32\_t data\_length\_; *// 压缩前的数据长度*

int32\_t data\_zlength\_; *// 磁盘上的数据长度,未压缩时等于压缩前的数据长度*

*// 压缩时为压缩后数据长度，data\_zlength <= data\_length*

*// 如果二者相等，表明数据没有压缩*

int64\_t data\_checksum\_; *// 写入磁盘中数据的checksum，不包含记录头*

};

该部分数据不经过序列化存储，每个项以网络序保存。

## 索引信息头

tablet index header记录了索引元信息的一些基本信息，包括该索引中的tablet的个数，版本信息，以及所有tablet row key的字符流偏移（相对于payload data）

**struct** ObTabletIndexHeader

{

int64\_t tablet\_count; // 索引文件中tablet的个数

int64\_t data\_version; // 当前tablets的版本

int32\_t row\_key\_char\_stream\_offset;

int32\_t reserved32;

int64\_t reserved64; // 保留字段

};

该部分数据不经过序列化存储，每个项以网络序保存。

## 索引文件tablets数组

tablet record array中的每一个元素结构如下：

**struct** ObTabletItem

{

int16\_t start\_key\_size;

int16\_t end\_key\_size;

int32\_t border\_flag;

int64\_t table\_id;

int64\_t data\_version;

int64\_t sstable\_count;

ObSSTableId sstable\_item[sstable\_count];

};

range中的start\_key, end\_key是专门存放在一个row key字符流中，start\_key\_size和end\_key\_size分别记录key的长度。

## row key char stream：

存放了每个tablet start\_key和end\_key字符串的流，所有的start\_key,end\_key按顺序存放，中间没有分隔符等。