# 需求分析

## 功能定义

用户希望定义表中的数据在指定的时间后过期，过期的数据按行自动清理。

用户可以指定表中的数据根据哪个字段的时间值作为过期依据，以及过期的时间相对值。

用户可以随时修改过期的字段，以及过期的相对值，在下一次合并时生效。

用户可能修改某条记录的相关字段来使该行数据提前过期或是延缓过期。

## 功能约束

数据过期不要求即时性。

过期字段应该为日期类型(PerciseTime)

设置的过期的相对时间为天，假如设定的过期字段为create\_time，相对时间为7天，那么

day(create\_time) + 7 < day(now) 即为过期的数据。

某些应用需要更新过期字段，来延缓或提前过期，通常他们会将过期字段设置为一个可变字段，并且在这个字段上直接写入一个过期日期值；那么只需要把相对时间设置为0即可。假定用户自己写入一个expire\_time字段为过期时间:

day(expire\_time) + 0 < day(now); 即为过期的数据。

这样一来，用户可以自行更新expire\_time来促成数据的提前或延缓过期，注意这个修改不是实时生效，而是等到下次合并时才生效。

## 基本思路

数据过期的配置与表相关，因此最好配置在schema的表相关的section中。

数据过期不要求实时性，因此最为方便的手段是在合并的时候过滤出过期的数据。考虑ChunkServer和合并Tablet时，可以在合并的结果中根据schema的配置计算过期的行，然后过滤后写入新的Tablet.

## 数据一致性问题

OB的ChunkServer三个副本独立合并，要求副本合并结果完全一致，因此，进行过期计算必须得到相同的结果，不能一个副本某行认为过期，而在另外一台ChunkServer上认为是有效的。

根据过期计算公式：

day(time\_field) + expire\_duration < day(now);

合并之后，time\_field的值肯定是一致的， expire\_duration是配置的固定值，唯一可能不一致的是day(now)，也就是ChunkServer的系统时间。如果我们假定只精确到天，那么只需要保证ChunkServer在合并的时候不跨天即可。

实际上这个保证在大多数情况下容易满足，我们只需要保证每台ChunkServer的系统时间差别不大，而且合并都在每天的12:30以后开始。

即便如此，也有可能一个ChunkServer出现合并太慢导致跨天的情况。所以安全的策略是有一个全局统一的时间，因为我们有rootserver 广播frozen version的机制，所以可以用rootserver来生成一个时间作为唯一时间来判断过期。UPS在每次做major freeze的时候记录freeze time,CS在合并时通过接口获取对应的version的freeze time.

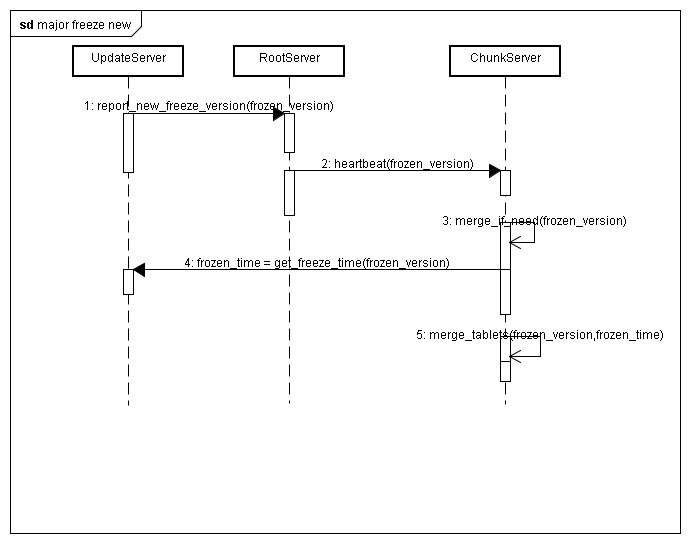
UpdateServer在生成freeze\_time的时候需要计入commit log，同步到备机。

如果可以容忍按天过期，并且不考虑合并跨天的情况，则无需生成唯一时间，只要使用ChunkServer本地时间即可。

# 功能实现

以下设计考虑用RootServer生成唯一时间做为判断依据的方式。用UPS生成freeze time的方式。

## 合并流程修改



1. UpdateServer做major freeze 后向rootserver汇报最新版本，记录当前的freeze time
2. Rootserver检查最新冻结版本是否比当前hold的last\_frozen\_version刚好大1，如果是，表示收到了一个新的version，
3. 利用heart beat广播最新的(frozen\_version)
4. ChunkServer检查本地版本，如果更新且满足合并条件就发起合并。
5. 从UpdateServer获取frozen\_version对应的frozen\_time
6. 发起合并。

## ChunkServer重启流程

## RootServer重启

## ChunkServer合并时过期数据

在合并的过程中加入一个过滤器，将符合条件的行过滤掉。

有一个问题需要注意，因为按column group合并，所以每次合并时候需要将要参与判断的过期字段读出来。对于多column group的tablet,这一做法会影响效率。

## schema配置

在schema中的table section中加入:

expire\_condition = [column\_id], [expire\_duration]

暂时仅支持一个字段设置过期

# 接口设计

## ObSchema

class ObSchema

{

/\*

\* return OB\_SUCCESS,表示有设置过期配置

\* return OB\_ENTRY\_NOT\_EXIST 表示未设置过期配置

\*/

int get\_expire\_condition(int64\_t &column\_id, int64\_t &duration);  
};

## UpdateServer

获取version对应的frozen\_time远程接口

int get\_frozen\_time(const int64\_t frozen\_version, int64\_t& frozen\_time);