truncate 功能设计文档

修改历史

版本	修订日期	修订描述	作者	备注
Cedar 0.3	2017-07-01	truncate功能设计文档	贺小龙	无

1.需求分析

> Cedar是由华东师范大学数据科学与工程研究院基于OceanBase 0.4.2 研发的可扩展的关系数据库 ，实现了巨大数据量上的跨行跨表事务。Truncate Table是标准化CBASE数据库DDL语言。在Cedar中增加该语法的目的是能够实现在大量删除数据时保留表的结构，从而满足某些业务需求。

2.适用场景

> truncate table 功能实现之后，在如下场景可能会得到应用:
>
> - 表中数据量非常大，且不需要回滚，删除数据后保留表的结构

3.功能简述

在Cedar中执行TRUNCATE TABLE语法如下：

```
truncate table [if exists] table_name1, table_name2, ..., table_name
N
    允许truncate多张表
    成功返回"0 rows affected"
    失败返回错误代码
```

因为设计方案的原因，在使用方面有如下几个限制：

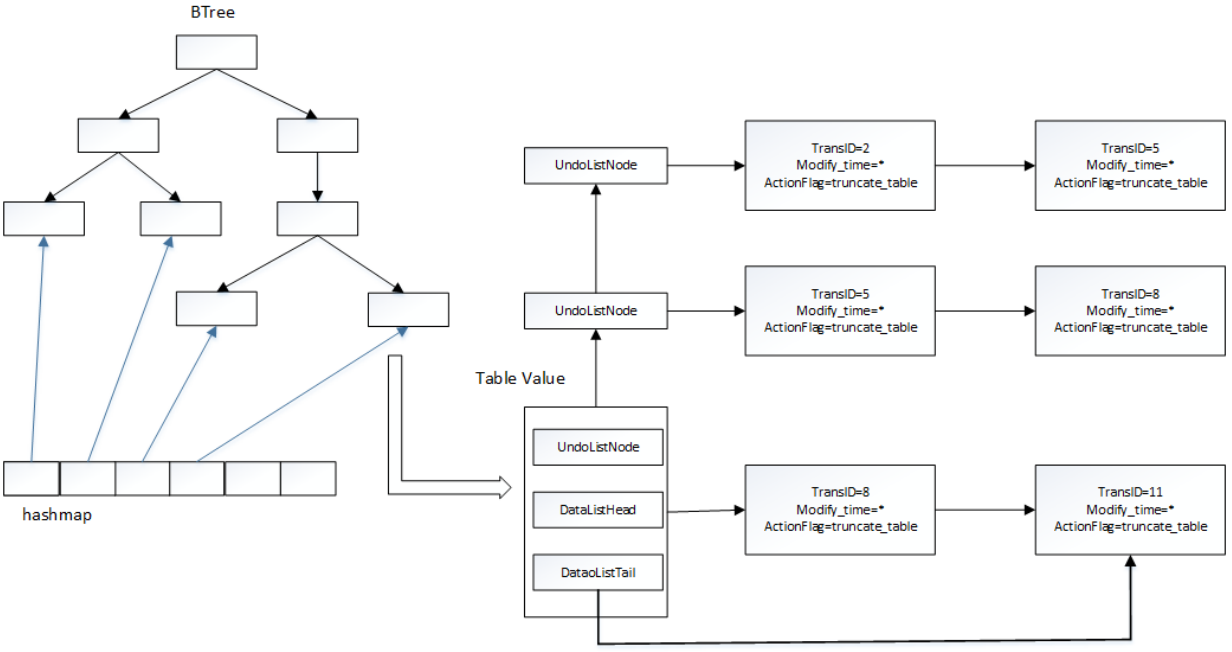
- truncate之后表不允许更新直至下一次memtable冻结完成（不等合并完成即可恢复写）
- memtable 冻结手动触发

即在TRUNCATE TABLE操作执行之后,还需要手动执行小版本合并或者大版本合并操作才算真正完成TRUNCATE操作。具体操作命令如下：

```
1. 主动发起大版本合并(工具在oceanbase/bin目录下)
    ./ups_admin -a ip -p port -t major_freeze
2. 主动发起小版本合并
    ./ups_admin -a ip -p port -t minor_freeze
```

4 设计思路

此方案的设计思路仿照UPS中行Btree数据结构+行锁，实现表级Btree+表锁，用于保存表级truncate信息。即是在memtable中增加一个btree树结构，命名为table_btree。具体结构如下:



> 根据此方案的设计相当于把Truncate Table 这一功能分为三个阶段，第一个阶段是执行truncate table语句，在该阶段只是将表中的数据锁住，不让外部进行操作，但对表的结构的操作依旧可以执行，如drop table等。第二阶段是进行Memtable冻结，此阶段是重新分配内存表，将之前的内存表dump到磁盘中。该阶段能够对表进行正常读写。最后一个阶段是数据合并，是在每日合并时完成。

4.1 dump memtable 设计

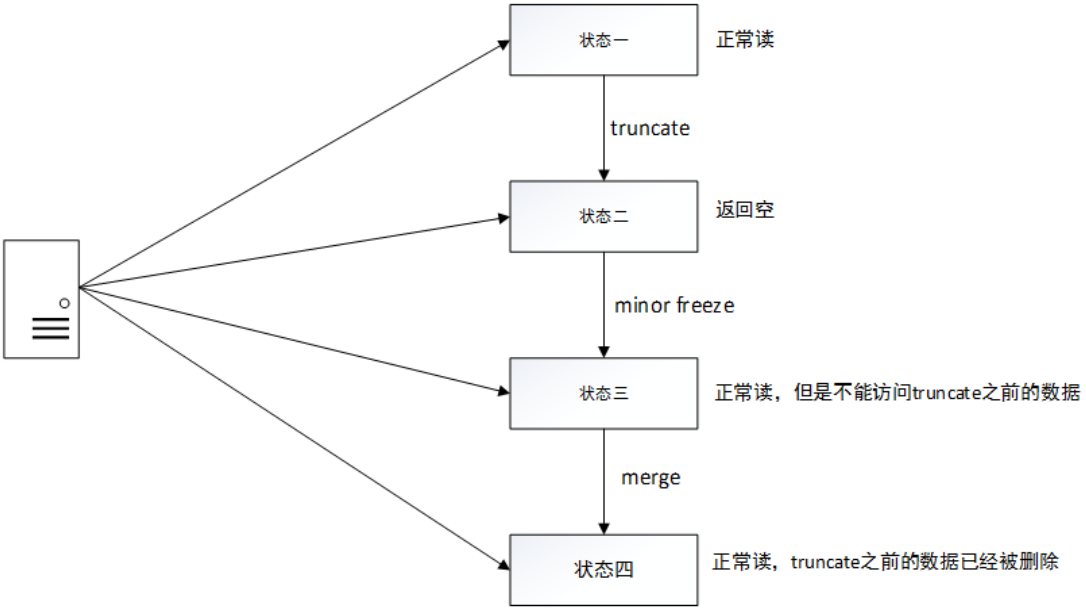
CS向UPS拉数据时，尽量读取冻结后的memtable数据，如果内存不够，系统会自动删除比较旧的memtable，这时候需要读取磁盘中的memtable ，由于dump memtable采用稀疏sstable格式存储，因此将ObSStableSchema 类中增加truncate_flag属性，用于标识该表在本memtable活跃期间成功执行过truncate。对该表的读操作，一旦发现dump memtable中truncate_flag字段有效，则一律返回ROW_DELETED。

4.3并发控制设计

由于将实现truncate之后对该表读写操作进行禁止，因为并发控制模块可不实现。

4.4 truncate的读控制

针对Truncate功能的读控制比较复杂，因为针对一张表被Truncate之后，表内的数据并没有被立即删除，而是每次在Memtable上增加一个Btree数据结构，等到每日合并合并时再进行删除。这样每次Select时，CS都需要向UPS查询该表是否被Truncate,如果已经被Truncate,则需要改变查询的数据范围，在该表被Truncate之前的数据都不应该被访问到。



4.5 truncate的写控制

在实现truncate之后对该表读操作进行禁止，返回空，因此并发控制模块可不实现。因为Cedar是单点更新，所以写控制比较容易。每次更新时都需要访问Memtable内存表。查看该表是否被truncate,如被truncate，则拒绝更新。