表锁功能设计文档

修订历史

版本	修订日期	修订描述	作者	备注
Cedar 0.2	2016-07-01	表锁功能设计文档	王嘉豪	无

1. 需求分析

在Cedar 0.1中只有记录级别的锁，没有更大级别的锁（如页锁、表锁等），表锁开发的主要目的是为Cedar增加表级粒度的锁，从而满足某些业务的需求。

在MySQL中可以使用`Lock table`语句对表进行加锁。Cedar中的表锁原理将与MySQL中的设计大致相同。

2. 适用场景

表锁实现之后，在如下场景中可能会得到利用：

- 在冲突率较高的负载下，一个长的更新事务之前添加表锁可以增加其执行成功率。
- 通过使用表锁可以在某张表上实现串行化，满足一些特殊应用的需求。

3. 功能简述

在事务执行时，支持以下的SQL语句用于给t1表添加排它锁：

- `LOCK TABLE t1`

当事务拿到排它锁后，只有该事务可以修改该表。

4. 设计思路

4.1 子功能模块划分

主要设计了三个模块：

- 表锁的数据结构
  - 用于锁的实现，主要采用的是自旋锁，使用了原子交换保证多线程安全
  - Cedar中仅支持2048个用户表，每个表都对应一个表锁数据结构，每个数据结构占12B的UPS内存空间，这些空间需要在UPS启动时就进行初始化

- 全局的表锁管理器
  - 用于记录每张表的加锁状态，排它锁会记录被那个事务持有，而意向锁和共享锁不会记录被哪些事务持有，只会记录个数。
  - 全局的表锁管理器也是在UPS启动时被初始化的。

- 每个session的表锁管理器
  - 用于记录对应的session添加的表锁信息。
  - 在session结束后，会根据添加的表锁信息释放锁。
  - 这个管理器的空间是在session创建时分配的，session结束后会释放。

4.2 总体设计思路

当事务想对表添加排它锁时，需要检查是否有其他事务正在对该表进行修改，如果没有事务修改，那么加锁成功，否则会重试加锁，直到加锁语句超时。为了方便这个检查，事务在修改某张表时，向该表的表锁添加意向排它锁，表示有事务正在修改该表。如果加锁发生了冲突，则该语句回滚。

4.2.1 INSERT/UPDATE/REPLACE流程修改

> 主要修改UPS更新操作的执行流程：
>
> - 在全局表锁管理器对表添加意向排它锁（新加）
> - 在每个session表锁管理器中添加这次的加锁信息（新加）
> - 为更新的记录添加行锁（原有）
> - 执行更新操作（原有）
> - 事务提交或撤销后，释放行锁（原有），根据session记录的加锁信息，释放所有表锁（新加）

4.2.2 LOCK TABLE流程

> 在事务执行过程中新增对表加锁的过程:
>
> - 在全局表锁管理器对表添加排它锁（新加）
> - 在每个session的表锁管理器中添加加锁信息（新加）
> - 事务提交或撤销后，根据session记录的加锁信息，释放所有表锁（新加）

5. 参考文献

[1] Kim K, Wang T, Johnson R, et al. ERMIA: Fast Memory-Optimized Database System for Heterogeneous Workloads[C]// International Conference. 2016.

[2] Levandoski J, Lomet D, Sengupta S, et al. Multi-version range concurrency control in Deuteronomy[J]. Proceedings of the Vldb Endowment, 2015, 8(13):2146-2157.