元数据锁介绍

MDL不需要显式使用，在访问一个表的时候会被自动加上。MDL的作用是，保证读写的正确性。你可以想象一下，如果一个查询正在遍历一个表中的数据，而执行期间另一个线程对这个表结构做变更，删了一列，那么查询线程拿到的结果跟表结构对不上，肯定是不行的。

因此，**在 MySQL 5.5 版本中引入了 MDL，当对一个表做增删改查操作的时候，加 MDL 读锁；当要对表做结构变更操作的时候，加 MDL 写锁。**

- 读锁之间不互斥，因此你可以有多个线程同时对一张表增删改查。
- 读写锁之间、写锁之间是互斥的，用来保证变更表结构操作的安全性。因此，如果有两个线程要同时给一个表加字段，其中一个要等另一个执行完才能开始执行。

MySQL行级锁

行级锁介绍

MySQL的行级锁，是由存储引擎来实现的，这里我们主要讲解InnoDB的行级锁。

- InnoDB的行级锁，按照锁定范围来说，分为三种：

```
- 记录锁（Record Locks）:锁定索引中一条记录。

- 间隙锁（Gap Locks）:要么锁住索引记录中间的值，要么锁住第一个索引记录前面的值或者最后一个索引记录后面的值。

- Next-Key Locks:是索引记录上的记录锁和在索引记录之前的间隙锁的组合。
```

- InnoDB的行级锁，按照功能来说，分为两种：

```
- 共享锁（S）：允许一个事务去读一行，阻止其他事务获得相同数据集的排他锁。
- 排他锁（X）：允许获得排他锁的事务更新数据，阻止其他事务取得相同数据集的共享读锁和排他写锁。
```

对于UPDATE、DELETE和INSERT语句，InnoDB会自动给涉及数据集加排他锁（X);

对于普通SELECT语句，InnoDB不会加任何锁，事务可以通过以下语句显示给记录集加共享锁或排他锁。

两阶段锁

传统RDBMS加锁的一个原则，就是2PL (Two-Phase Locking，二阶段锁)。相对而言，2PL比较容易理解，说的是锁操作分为两个阶段：加锁阶段与解锁阶段，并且保证加锁阶段与解锁阶段不相交。下面，仍旧以MySQL为例，来简单看看2PL在MySQL中的实现。

死锁

两个 session 互相等等待对方的资源释放之后，才能释放自己的资源,造成了死锁

事务介绍

在MySQL中的事务是由**存储引擎**实现的，而且支持事务的存储引擎不多，我们主要讲解**InnoDB**存储引擎中的事务。

事务处理可以用来维护数据库的完整性，保证成批的 SQL 语句要么全部执行，要么全部不执行。

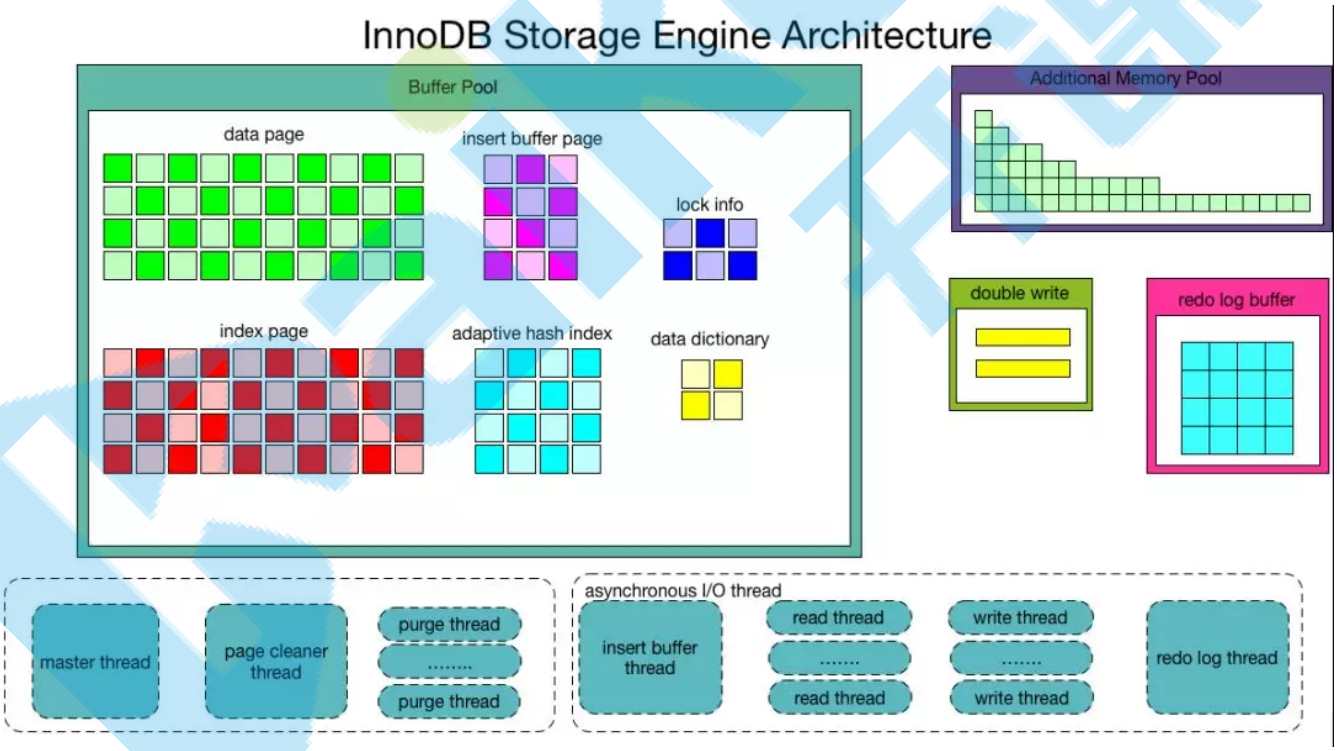
事务四大特性(ACID)

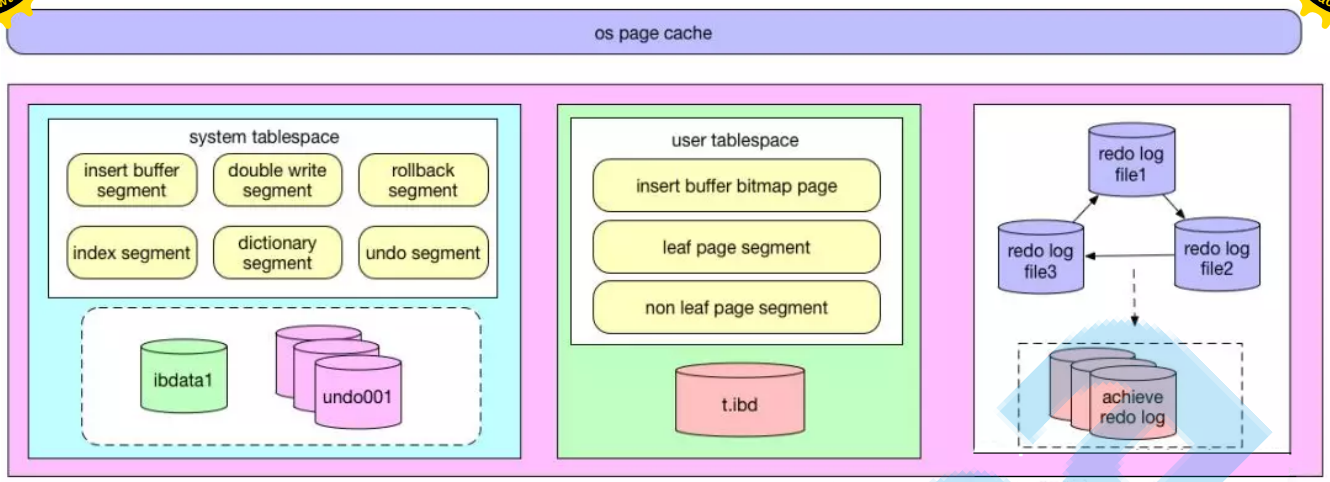
- Atomicity（原子性）：构成事务的的所有操作必须是一个逻辑单元，要么全部执行，要么全部不执行。
- Consistency（一致性）：数据库在事务执行前后状态都必须是稳定的或者是一致的。
- **Isolation（隔离性）：事务之间不会相互影响。**

  由锁机制和MVCC机制来实现的

  MVCC(多版本并发控制)：优化读写性能（读不加锁、读写不冲突）
- Durability（持久性）：事务执行成功后必须全部写入磁盘。

InnoDB架构分析





InnoDB内存结构

Buffer Pool缓冲池、数据页和索引页、自适应哈希索引(Adaptive Hash Index)等

InnoDB磁盘文件

系统表空间和用户表空间、重做日志文件和归档文件、重做日志的落盘机制