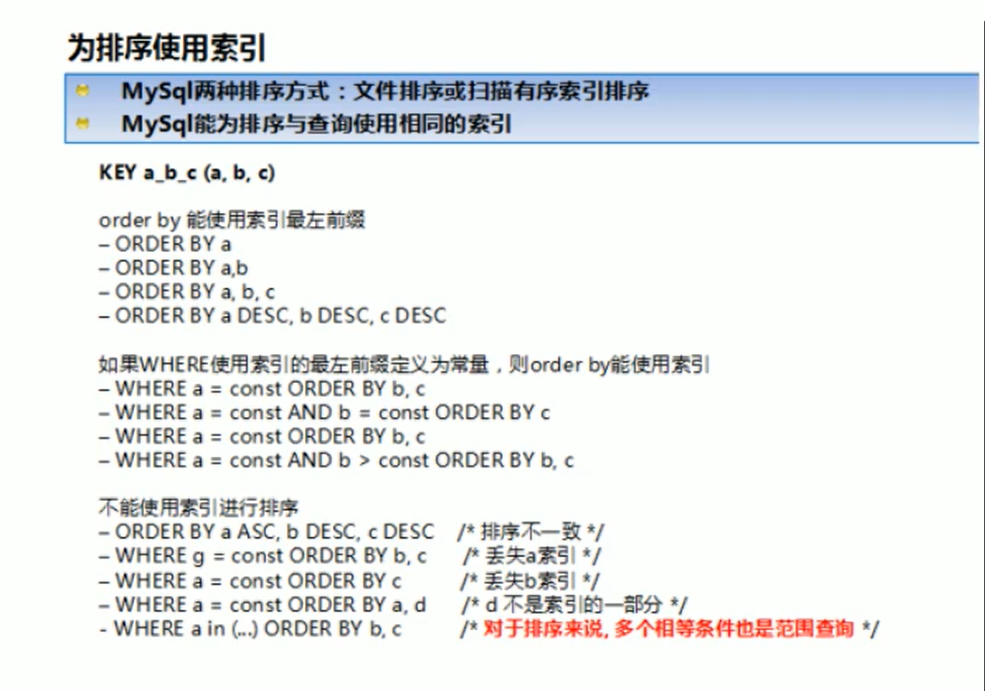
Mysql学习笔记



Mvcc(多版本并发控制)

1. 什么是当前读
2. 什么是快照读

**说白了MVCC就是为了实现读-写冲突不加锁，而这个读指的就是快照读, 而非当前读，当前读实际上是一种加锁的操作，是悲观锁的实现**

**MVCC带来的好处是？**

**多版本并发控制（MVCC）是一种用来解决读-写冲突的无锁并发控制，也就是为事务分配单向增长的时间戳，为每个修改保存一个版本，版本与事务时间戳关联，读操作只读该事务开始前的数据库的快照。 所以MVCC可以为数据库解决以下问题**

1. **在并发读写数据库时，可以做到在读操作时不用阻塞写操作，写操作也不用阻塞读操作，提高了数据库并发读写的性能**
2. **同时还可以解决脏读，幻读，不可重复读等事务隔离问题，但不能解决更新丢失问题**

**Mvcc依赖 三个隐式字段（隐式主键，事务ID，rollback指针），Undo\_log,readview达到效果。**

**RC,RR级别下的InnoDB快照读有什么不同？**

在RR级别下的某个事务的对某条记录的第一次快照读会创建一个快照及Read View, 将当前系统活跃的其他事务记录起来，此后在调用快照读的时候，还是使用的是同一个Read View，所以只要当前事务在其他事务提交更新之前使用过快照读，那么之后的快照读使用的都是同一个Read View，所以对之后的修改不可见；

即RR级别下，快照读生成Read View时，Read View会记录此时所有其他活动事务的快照，这些事务的修改对于当前事务都是不可见的。而早于Read View创建的事务所做的修改均是可见

而在RC级别下的，事务中，每次快照读都会新生成一个快照和Read View, 这就是我们在RC级别下的事务中可以看到别的事务提交的更新的原因

**总之在RC隔离级别下，是每个快照读都会生成并获取最新的Read View；而在RR隔离级别下，则是同一个事务中的第一个快照读才会创建Read View, 之后的快照读获取的都是同一个Read View**

**Innodb 聚簇索引 辅助索引**

**聚簇索引的树叶子节点直接存放了行记录**

**辅助索引对应的树节点存放的则是相应的主键值**

**Myisam使用非聚簇索引，索引表和数据表分离，**

数据库为什么使用B+树构建索引，而不是红黑树，B树？

熟悉的二分查找，二叉树可以把速度提升到O(log(n,2))，查询的瓶颈在于树的深度，最坏的情况要查找到二叉树的最深层，由于，每查找深一层，就要访问更深一层的索引文件。在多达数G的索引文件中，这将是很大的开销。所以，尽量把数据结构设计的更为‘矮胖’一点就可以减少访问的层数。

而对于B树，

1. **B+树的内部节点并没有指向关键字具体信息的指针，因此其内部节点相对B树更小，如果把所有同一内部节点的关键字存放在同一盘块中，那么盘块所能容纳的关键字数量也越多，一次性读入内存的需要查找的关键字也就越多，相对IO读写次数就降低了。**
2. **由于非终结点并不是最终指向文件内容的结点，而只是叶子结点中关键字的索引。所以任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字查询的路径长度相同，导致每一个数据的查询效率相当。（查找效率稳定）**
3. 由于B+树的数据都存储在叶子结点中，分支结点均为索引，方便扫库，只需要扫一遍叶子结点即可，但是B树因为其分支结点同样存储着数据，我们要找到具体的数据，需要进行一次中序遍历按序来扫，所以B+树更加适合在区间查询的情况，所以通常B+树用于数据库索引