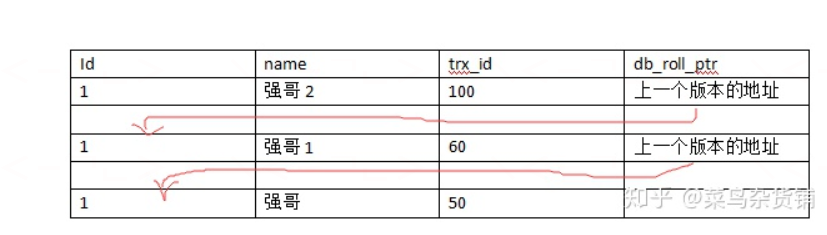
# mysql

1. 为什么mysql索引使用B+树而不是B树

B+树会把我们所有的数据存贮到叶子节点，而且叶子结点直接按可以做顺序存放；进行范围查找跨页数据的时候，B由于B树的不确定性（数据有可能在叶子节点也可能在根节点，需要多次随机访问磁盘才能找出对应数据）。但是B+树由于数据全部在叶子几点，而且叶子之间跟链表一样顺序存放，我们可以像链表一样遍历我们的范围数据

1. Mysql的mvcc

Mysql的innerDb实现的mvcc，我们的mysql事务中的读提交和可重复读都是通过mvcc进行实现的。InnerDb的undolog行记录有两个列，trx\_id和db\_roll\_ptr，其中trx\_id就是我们常说的事务id，每次新增一个事务的时候，事务id就会+1，所以通过事务id能够判断我们的一个事务执行顺序，innerDb每次执行更新的时候会在undolog里面生成一个事务更新记录，这个记录会关联我们的一个事务id。这个就是我们mvcc的多版本的含义，就是每次更新都记录了一个更新记录，而且是一个更新版本链，他们有逻辑关系。当没有事务关联这个记录的时候，这个记录才被允许删除。db\_roll\_ptr作用是指向上一个版本地址，相当于版本链的指针。



1. Mysql怎么利用mvcc实现读提交和可重复读的隔离级别

已提交读和可重复读的大实现区别在于他们生成视图的策略不同。视图中有个列表保存了系统中当前活跃的读写事务，通过这个列表来判断记录的某个版本是否对当前事务可见。其中最重要的几个属性为：

1. up\_limit\_id:当前已提交的事务号+1，当前事务号<up\_limit\_id的数据都是可见的。也就是说创建事务的时候，之前已经提交的事务对于当前视图来说是可见的
2. low\_limit\_id:当前最大事务号+1，当前事务号>=low\_limit\_id的数据都是不可见的，也就是说，在创建事务之后创建的事务对于当前视图来说肯定是不可见的
3. trx\_ids：当前活跃的事务号列表，当前事务开启创建视图时候，都有哪些事务是开启但未提交的。

对于读提交和可重复读来说，他们的版本比较可见性策略其实都是一样的，他们的区别在于创建视图的时机：读提交是在事务执行到sql语句的时候才开始创建的视图，这时候获取up\_limit\_id很可能比事务刚开启时候的高，这时候看到的数据也就不一样，能看到更高版本的数据。可重复读创建视图的时机是开启事务的时候就创建视图，开启事务的那一刹那就决定了你能看到哪个版本的数据。但是如果事务里面有更新的话，会导致当前数据获取一次当前读。因为自己更新，再次读肯定要获取最新更新的数据。

# Jvm

1. Cms垃圾收集器

Cms垃圾收集过程：

1. 初始标记：仅仅是标记一下gc roots能直接关联到的对象，很快
2. 并发标记：会进行可达性分析计算，标识回收对象
3. 重新标记：修正并发标记期间系统产生的新的垃圾
4. 并发清除：清除数据

Cms停顿时间较少，但是对cpu比较敏感，而且进行并发标记和并发清除的时候会占用cpu资源，默认启动回收线程数量是（cpu数量+3）/4，所以，cpu数量较少的时候影响很大

2.G1。