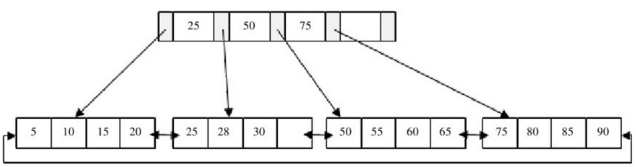
## 存储引擎

Innodb

myisam

## 索引：

1. **B+树**



非叶子节点不存放data数据，只起索引的作用，叶子节点会存放最终的数据，而且叶子节点之间通过双向链表进行链接，可以由当前节点访问前后的节点。

**2. InnoDB中的聚集索引**

InnoDB存储引擎表是索引组织表，即表中数据按照主键顺序存放，而聚集索引就是按照每张表的主键构建一颗B+树，同时叶子节点中存放的表中一行的数据，也将聚集索引的叶子节点称为数据页。通过聚集索引可以直接在叶子节点上获取数据，此外，聚集索引可以特别快的进行范围查询。

**3. 辅助索引**

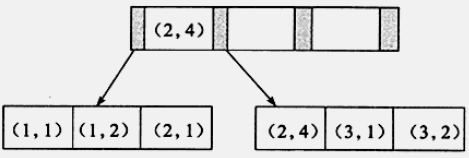
Innodb中除了聚集索引外其他索引称为辅助索引，辅助索引也是B+树的结构，与聚集索引不同的是辅助索引叶子节点存放的不是具体的某一行的数据，而是相关联的那一行的主键，也就是说，如果我们通过辅助索引查询之后得到主键，还需要去聚集索引那边根据主键来查询具体的行。

**4. Cardinality**

Cardinality中文名为基数，又称为索引选择性，索引选择性计算公式为某列中的非重复值的个数/该列所有值的个数，结果的取值范围为[0-1]，越高代表选择性越高，也就代表着索引的效果越好。例如性别属性，在一般表中男女比例为1:1，如果给性别属性加上索引，其索引选择性为0.5，相当于只能区分百分之五十的数据，这个时候才用索引完全没有必要。

**5. 联合索引**

联合索引是对多个列进行匹配，例如有列a，b，建立联合索引（a，b），其B+树为：



可以看到，索引的顺序首先按照a的值进行排序，然后在同一个a的情况下，按照b的顺序进行排序。

对于查询SELECT \* FROM TABLE WHERE a=xxx and b=xxx，是可以使用（a,b）这个联合索引的，对于单个a列查询SELECT \* FROM TABLE WHERE a=xxx，也是可以使用联合索引，因为索引的顺序就是a的顺序，而对于单个b列是不可以使用联合索引，因为联合索引中b并没有顺序。这就是联合索引的最左匹配原则。

联合索引的第二个好处就是已经对第二个键进行了排序，当第一个键确定下来后，order by第二个键的话可以直接获取，不需要进行排序操作。

## ACID

1. 原子性：

事务被视为不可分割的最小单元，事务的所有操作要么全部提交成功，要么全部失败回滚。

1. 一致性：

数据库在事务执行前后都保持一致性状态。在一致性状态下，所有事务对一个数据的读取结果都是相同的。（你给朋友转1000，你的卡上少了1000，但是他的卡上却没有多1000，这就是状态不一致的情况）

1. 隔离性：

一个事务所做的修改在最终提交以前，对其它事务是不可见的。

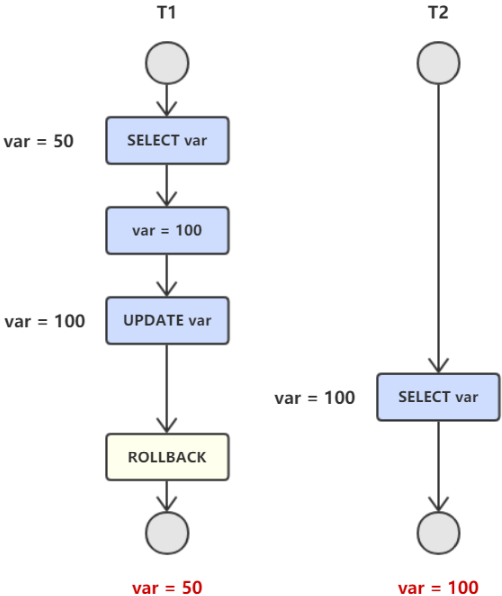
1. 持久性：

一旦事务提交，则其所做的修改将会永远保存到数据库中。即使系统发生崩溃，事务执行的结果也不能丢失。

## 事务隔离级别

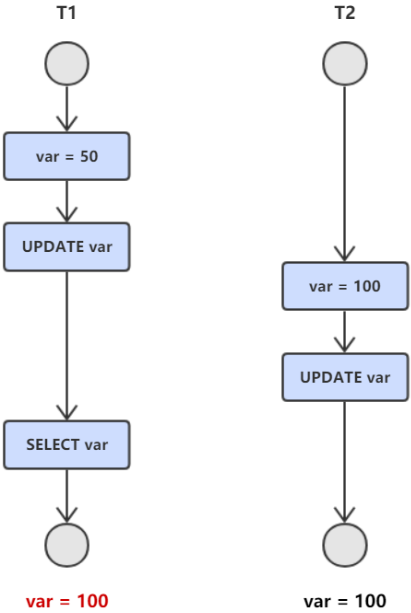
读未提交：一个事务可以读取另一个事务未提交的更新结果。

问题：存在数据库脏读的情况，会读取了还未提交的数据。



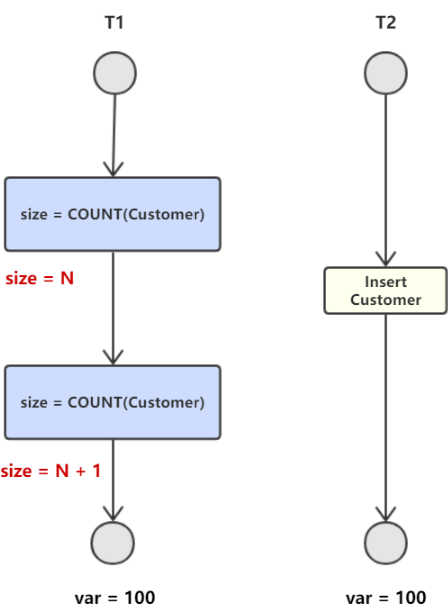
读已提交：一个事务的更新结果只有在该事务提交之后，另一个事务才可以读取到数据的更新结果。

问题：存在不可重复读的现象，事务A在update前后看到的数据不一致。



重复读：保证同一事务的多个实例在并发读取数据时，会看到同样的数据行。事务A在读取的时候会将数据加锁，其他事务无法修改这些数据，但是可以进行插入操作。

问题：存在幻读的情况，数据的数量前后不一致



可串行化：进行事务的强制排序，使之不可能相互冲突，在每个读的数据行上加上共享锁。

## 范式

1NF：属性不可分

2NF：首先是1NF，另外包含两部分内容，一是表必须有一个主键；二是没有包含主键的列必须完全依赖于主键，而不能只依赖于主键的一部分。 （强调非主键完全依赖主键）

3NF：首先是2NF，另外非主键必须直接依赖于主键，不能存在传递依赖。即不能存在：非主键列A依赖于非主键列B，非主键列B依赖于主键的情况。（强调非主键不能依赖于非主键）

## 优化：

1. 只返回必要的列：最好不要select \* 语句
2. 只返回必要的行：使用limit语句来限制返回的数据。
3. 缓存重复查询的数据：使用缓存可以避免在数据库中进行查询。
4. 使用索引，在where或者order by之类的。
5. 分解大连接查询（重用缓存，减少锁的竞争）