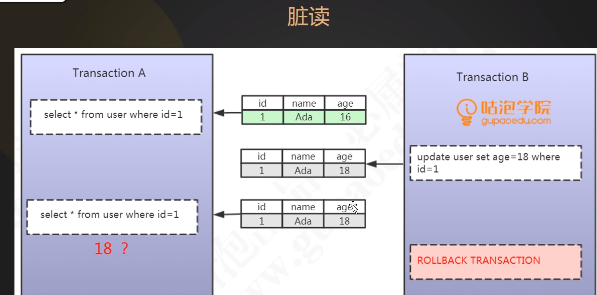
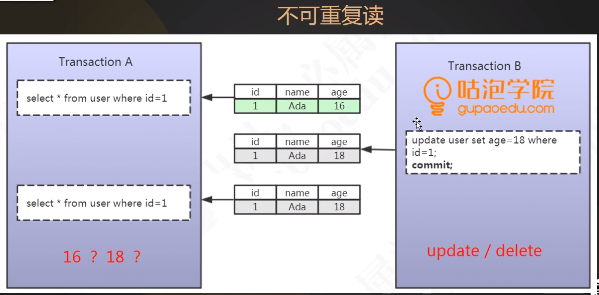
事务

* 涉及到数据修改（插入、删除、修改）都需要开启事务。
* MySQL中有两个事务日志
  + 全部成功
  + 全部失败：为了事先事务的原子性，比如一个事务有1 2 3 4 5个操作，前面三个都成功了，但是第四个失败了，这时候需要让前面三个也失败。这时候需要撤销，依靠undo log日志。

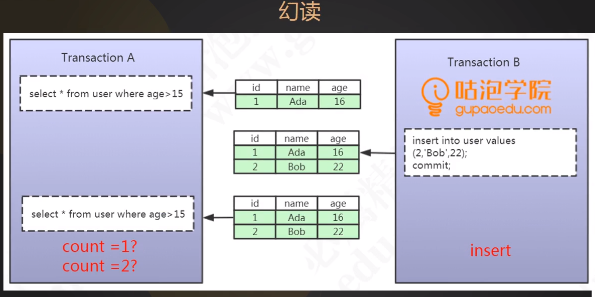
1. 事务是数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成。
2. 不同的存储引擎代表了表的不同的存储方式，对表的操作和对表的管理的方式。
3. 事务的四大特性（ACID）：
   1. 原子性（Atomicity）
   2. 一致性（Consistency）：其他三个原则都是为了保证一致性。数据库自身的完整性约束不能被破坏，比如逐渐在事务完成前后都不能为空等。用户自定的完整性约束不能被破坏，通常在代码中进行控制。比如A给B转账1000块，B只收到了500，数据库没办法发现这种不一致性，得通过用户在代码中检验。
   3. 隔离性（Isolation）：并发的事务互相不干扰，相互透明的。
   4. 持久性（Durablity）：只要事务提交成功了，修改的数据就应该被永久的保存在数据库中，不应该因为数据库重启等原因恢复之前的状态。
4. 手动开启一个事务：
   1. begin； sql语句 commit； or rollback；
   2. start transaction….



1. 事务并发会带来的问题
2. 读一致性问题：脏读，不可重复读，幻读
   1. 脏读（相对于数据库中的数据）：T1修改某一数据并将其写回磁盘，T2读取后，T1由于某种原因被撤销，则此时T2读取的数据和数据库中的数据不一致。
   2. 不可重复读（相对于之前的读取结果，update和delete语句）：T1读取数据后，T2执行更新（修改，删除）操作，使T1无法再现前一次读取结果



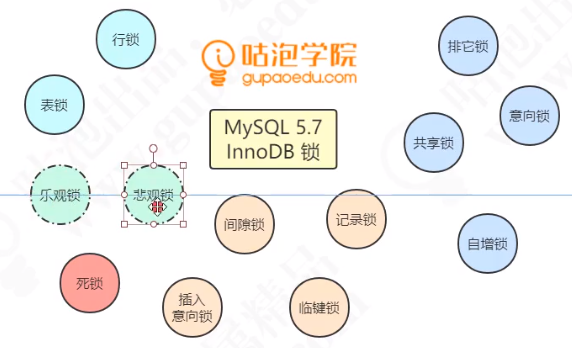
* 1. 幻读（insert语句）



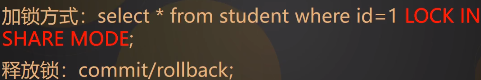
1. 解决方法
   1. 基于锁的并发控制：锁的太厉害会导致数据库并发度降低。
   2. 多版本并发控制MVCC：生成一个数据请求时间点的一致性数据快照，并用这个快照来提供一定级别（语句级或者事务级）的一致性读取
2. 锁

划分方式：

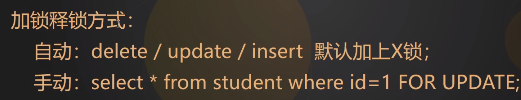
* 粒度：
  + 表锁
  + 行锁
* 用法：
  + 乐观锁
  + 悲观锁
* 类型：
  + 排它锁（行锁）
  + 共享锁（行锁）
  + 意向锁
    - 意向共享锁（表锁）
    - 意向排它锁（表锁）
* 锁的算法：
  + 间隙锁
  + 记录锁
  + 插入意向锁
  + 临键锁
* 锁带来的问题：
  + 死锁



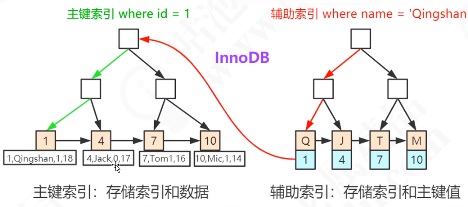
* + 1. 表锁与行锁的区别：MyISAM支持表锁，InnoDB既支持表锁又支持行锁
  1. 锁的粒度：表锁 > 行锁
  2. 加锁的效率：表锁 > 行锁
  3. 冲突概率：表锁 > 行锁
  4. 并发性能：表锁 < 行锁
     1. 共享锁（S锁）：又称读锁，共享锁就是多个事务对于同一个数据可以共享一把锁，都能够访问数据，但是只能读，不可修改。



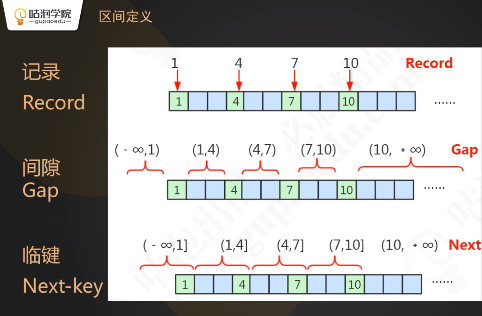
* + 1. 排他锁（X锁）：又称写锁，排他锁不能与其他锁并存，如一个事务获取了一个数据行的排他锁，其他事务就不能在获取该行的锁（共享锁和排他锁），只有获得排他锁的事务可以对数据进行读取和修改。



* + 1. 意向锁：由数据引擎自己维护，用户无法手动操作意向锁。
* 一个事务成功地给一张表加上表锁的前提：没有其他任何事务已经锁定了这张表的任意一行数据。这可能需要全表扫描。有了下面两个锁之后，就不用全表扫描了，因为只需要判断表上面是不是有意向锁就好了。即——提高加表锁的效率（把下面两个锁当成一种“标记”）。
  + - 1. 意向共享锁（IS锁）表示事务准备给数据行加入共享锁，也就是说一个数据行加共享锁前必须先取得该表的IS锁。
      2. 意向排他锁（IX锁）表示事务准备给数据行加入排他锁，说明事务在数据行加排他锁前必须先取得该表的IX锁。
    1. 到底锁住了什么东西？**锁住了索引项**。
       1. 表中没有显式的索引（之所以说“显式”，是因为不使用primary key语句定义索引的话，引擎会给表创建主键索引，一张表必须有聚集索引）
       2. 表中有unique key（必须not null时）可以作为聚集索引。
       3. 如果没有主键，而且没有非空的unique key，那么就是用rowid（一个隐藏的字段，自增）
       4. 数据库中有主键索引（聚集索引）和二级索引（其他的索引）之分，其结构如下，如果既有主键索引又有二级索引，则查询二级索引的字段时，先查二级索引，然后找到主键之后，在使用主键索引进行查找。

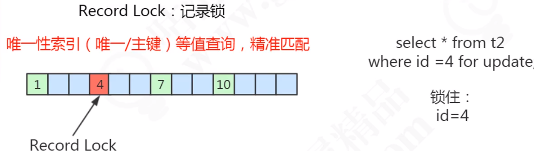


1. 区间定义

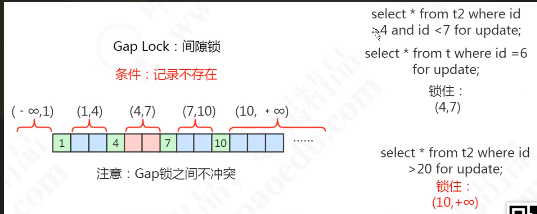


记录（索引记录，index record）：使用主键索引划分数据的区间

记录锁：唯一性索引（唯一/主键）等值查询，精准匹配（锁住离散的值）



间隙锁：锁定范围。记录不存在。间隙锁之间不冲突（也就是可以重复在间隙之间获取排他锁）最主要的作用就是为了阻塞插入。**间隙锁只会出现在辅助索引上，唯一索引和主键索引是没有间隙锁。间隙锁（无论是S还是X）只会阻塞insert操作**



临键锁：锁定范围加记录





InnoDB解决了幻读，并且可以保持很高的并发性能。使用间隙锁和临键锁来阻塞插入，解决幻读。



1. 乐观锁&悲观锁

悲观锁：当要对数据库中的一条数据进行修改的时候，借助数据库提供的锁机制（排他锁，共享锁等）提前锁定数据。

乐观锁：乐观锁假设数据一般情况下不会造成冲突，所以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检查，如果发现冲突了，则可以回滚。实现乐观锁的方式就是记录数据版本（多版本并发控制，MVCC， CAS(自旋锁)）