# mysql

1. 索引的基本原理
   1. 底层其实是一个B-Tree结构，目的就是缩小扫描范围，避免全表扫描
   2. 1.首先像主键或unique约束的字段会自动加一个索引，表的每一条记录都会有一个存储编号
   3. 2. 索引在不同的存储引擎中都是以B-Tree形式存在的
   4. 3. 首先它会在底层进行排序，假如第一个id索引是100，那么就从100开始，第二个是99则放在二叉树左边，第三个为120，就放在1右边，准寻左小右大原则排序
   5. 4. 进行查找的时候，从where子句后如果字段有索引，则从二叉树中去匹配，匹配的时候会将字段=值转换为每一条记录的存储编号，可以快速定位到
2. 索引适用场景
   1. 数据量大（量的衡量标准看硬件）
   2. 经常出现在where后面的字段
   3. 该字段很少DML（因为每次的增删改都会使索引重新排序）
3. 索引失效的场景
   1. 模糊查询时，百分号在前面 "%T"
   2. or条件拼接的时候，有一方的字段不是索引
   3. 复合索引，或者索引字段参与运算
4. 事务的基本特性和隔离级别和传播行为
   1. 基本特性ACID
      1. 原子性是指一个事务中的操作要么全部成功，要么全部失败
      2. 一致性是数据库从一个一致性状态转换到另一个一致性的状态
      3. 隔离性是指一个事务的修改在最终提交之前，对其他事务是不可见的
      4. 持久性是指事务一旦提交，所做的修改就会永久保存在数据库中
   2. 隔离级别
      1. 读未提交：可以读到其他未提交事务的数据
         1. 会出现脏读，不可重复读，幻读
      2. 读已提交：可读到其他事务已经提交的数据（Oracle默认的）
         1. 不可重复读，幻读
      3. 可重复读：在事务中，第一次读取到的数据是100，直到事务结束都会是100（mysql默认的）
         1. 幻读被Innodb的MVCC解决了
      4. 序列化：无并发能力，因为mysql的innoDB会给读操作加上一把锁
         1. 不会出现任何问题
      5. 并发越高，隔离级别就越小
   3. 脏读：事务小明更新了一份数据，与此同时事务小红也读取了同一数据，这个时候小明回滚了操作，则小红现在所读取的数据是不正确的
   4. 不可重复读：事务小红在多次读取相同数据的时候，发现结果不一致，可能是小明在小红读取过程中，将数据修改了（并发更新）
   5. 幻读：数据库中有一条数据张三，事务小红对其进行查询，任何将它删掉，这个时候小红还没有结束，小明就又动了怀心思，插一条数据进去，当小红再次查询的时候才发现数据不对劲
   6. 传播行为
      1. required
         1. 没有事务我就创一个，有我就加入
      2. requireNew
         1. 不管有没有，我都自立为王
      3. 一共有7种传播行为
   7. springBoot下事务的大坑!!!!!(本地事务失效问题)
      1. 就是说我现在同一个service下有3个方法ABC，其中都设置了事务，A调用B和C
      2. 但是执行起来，BC事务设置却不生效，它跟做A的事务执行了
      3. 原因就是绕过了这个代理对象，因为@Transaction是创建了一个代理对象来实现的，于是可以开启使用aspectj动态代理功能，然后使其对外暴露代理对象，然后获取一下代理对象调用即可
5. ACID靠什么保证的
   1. A原子性由 undo log日志来保证，记录了需要回滚的日志信息，事务回滚时撤销已经执行成功的sql
   2. C一致性由其他三大特性保证，程序代码保证业务上的一致性
   3. I隔离性由MVCC来保证（多版本并发控制，读取数据时通过一种类似快照的方式将数据保存下来）
   4. D持久性是由 内存+redo log来保证的
6. 简述MyISAM和InnoDB的区别
   1. MyISAM
      1. 不支持事务，每次查询都是原子的
   2. InnoDB
      1. 支持ACID事务