## 2020/08/03==================================

1. 自我介绍
2. 哈希表怎么实现，冲突怎么解决
   1. 哈希表的底层数据结构是数组，很多地方也叫Bucket。首先通过将key的值传给hash函数，求出对应的索引，找到相应的下标进行存储，时间复杂度是O（1）。
   2. 解决方法：
      1. 开放定址法
      2. 再hash法
      3. 链地址法（HashMap）
3. 维护一个堆（logn）
4. Nlogn
5. B树和B+树
   1. B树是多路平衡搜索树，它类似于普通平衡二叉树，区别是允许每个节点有多个子节点。B树为外部存储器（读写磁盘）设计，用于读写大块数据。
   2. 空间局部性原理：存储器的某个位置被访问，它附近的位置也被访问。
   3. B+树
      1. 叶子节点存储数据，非叶子节点并不存储数据
      2. 叶子节点增加了链指针
   4. 区别
      1. B树的非叶子节点保存key和value，而B+树的非叶子节点只保存key的副本，叶子节点保存value（data值）。B+树查询时间复杂的logn，B树则与位置有关。
      2. B+树叶子节点数据是用链表连起来的，可以做到区间访问性，访问磁盘某个位置，附件位置也被访问。
      3. B+树适合外部存储，key小，磁盘单次IO信息量大，IO次数少。
   5. Mysql的数据结构是B+树
6. 聚集索引和非聚集索引
   1. 聚集索引包含索引和数据，索引的叶子节点就是对应的数据。索引顺序和表中记录的物理顺序是一致的。
   2. 非聚集索引将索引和数据分开，索引的叶子节点指向数据的对应行，等于做了一个映射。索引顺序和物理顺序不一致。
   3. 一个表只能又一个聚集索引，通常设为主键。
7. MVCC多版本并发控制（Multi-Version Concurrency Control），实现对数据库的并发访问。MVCC是通过保存数据在某个时间点的快照来实现的，也就是同一时刻不同事物看到的相同表里的数据可能不同。从而实现并发控制。写写用锁，写读用mvcc。
8. DB日志
   1. 重做日志：持久性，记录事务执行后的状态。
   2. 回滚日志：原子性，保证事务发生前的版本
   3. 二进制日志：实现备份，是增量备份，只记录改变的数据。（备份）
   4. 错误日志：启动停止以及运行过程中的错误信息
   5. 慢查询日志：查询时间长或无索引的查询语句
   6. 通用查询日志：记录所有查询
   7. 中继日志：主从复制，读取主服务器的二进制日志，本地回放，实现同步。（复制）
9. 事务用在并发操作多张表时，保证数据的完整性。一方发生错误，回滚数据，保证两次操作的安全。特征：
   1. 原子性：同一个事务是一个不可分割的操作单元，要么全部成功，要么全部失败。重做日志。
   2. 一致性：事务操作的前后状态是一致的，符合逻辑运算。
   3. 隔离性：并发执行多个不同的事务之间互不干扰。
   4. 持久性：事务一旦提交，对数据库的改变是永久性的。回滚日志。
10. 事务的隔离级别
    1. 并发问题
       1. 脏读：一个事务读取了其他事务未提交的数据，读的是未提交。
       2. 不可重复读：事务两次读取数据不一样，读的是已提交。
       3. 幻读：幻读是插入或删除操作，是已提交的。
    2. 隔离级别
       1. 未提交读：事务A写数据，事务B不可写但可读修改未提交的数据。
       2. 提交读：事务A写数据，禁止事务B读未提交的数据。
       3. 可重复读：事务A写数据禁止事务B任何操作，事务A读数据禁止事务B写事务，。
       4. 序列化：事务被定义为串行执行。
    3. 隔离级别对比
       1. 
11. TCP和UDP