1. **mysql有哪些存储引擎？什么区别？**

主要存储引擎有：InnoDB、MyISAM、Memory

InnoDB和MyISAM区别：

InnoDB支持事物，MyISAM不支持事物，Innodb默认事物自动提交，一条sql就是一个事物，在大表插入的时候很影响性能，建议改为手动提交事物。

Innodb支持行级锁，MyISAM不支持，只支持表级锁，默认写操作优先级高于读操作，myisam不适合大量查询与修改并存的操作，那样会阻塞查询操作，造成锁表。

对于大量select操作，几乎很少的update和insert操作，myisam更合适。对于频繁更新的表，Innodb比较合适。

Innodb支持外键，MyISAM不支持外键。

Innodb不支持全文索引，MyISAM支持全文索引。

MyISAM在没有where条件的count(\*)查询是比较快的，因为MyISAM维护一个计数器，查询时直接从计数器中读取，而Innodb必须进行全表扫描。

Innodb的primary index(主键索引)和raw data放在一起的，其他索引都是单独存放的，然后有个指向主键的指针。

Innodb如果更新不确定的一些数据，同样会进行锁表，例如：

update table set num=1 where name like “%aaa%”

所以，不建议以上操作。

1. **使用mysql索引有哪些原则？索引数据结构**

选择唯一性索引，可以快速定位到该记录。

经常需要排序、分组和join的字段建立索引。例如：外键

经常作为查询条件的字段建立索引。

限制索引的数量，索引并不是越多越好，过多的索引增加索引的维护成本，影响数据库性能。

索引字段的可选择性要高，例如在性别字段建立索引没有任何意义。

对于某些文本类型的字段尽量使用前缀索引。如果使用全文索引，在查询的时候非常浪费时间。使用前缀索引可以提高查询速度。

及时更新索引，对于很少使用到的索引及时删除。

Innodb使用的数据结构是B+Tree

B+Tree是BTree的一种变型。

区别：

B+Tree：1.n棵子树的节点含有n个关键字2.所有叶子节点全部包含关键字信息，并且含有指向该关键字的指针。而且叶子节点本身按照自小到大的顺序排列。3.所有非中端节点看成索引节点，索引节点中含所有子树中最大（最小）的关键字

使用索引可以大大提高查询的速度，不使用索引查询的时候会进行全表扫描，例如表中有N条记录，查询的时间复杂度是N，使用索引查询的时间复杂度是logN。

1. **数据库锁问题**

**数据库有哪些锁？如何实现？**

锁就是为了控制数据库的并发操作，防止出现脏数据的问题：

**悲观锁**：悲观锁认为用户在读取数据的时候，别人刚好在写自己读的数据，为了不让别人修改自己读的数据，在读的数据进行加锁，不允许修改，只有当自己读取完了（事物提交以后）才会释放锁，别人才能操作。

**实现：在查询的时候 select xxx from table for update进行锁定该数据**

**乐观锁**：认为自己读取数据的时候别人不会去操作该数据，在查询的时候不做任何处理，在更新的时候加version来控制。

乐观锁一般实现：在表中加一个字段version

1. select name,version from t1 where id=1

假如得到数据：[‘张三’,12332]

1. update set name = ‘aaa’,version=versoin+1 where id=1 and version=12332

**共享锁**（读锁）：读取时候创建的锁，其他用户可以对数据进行读取，但是不能修改和删除，直到释放所有共享锁

select xxxx from table lock in share mode;

**排他锁**（写锁）：用户对数据加排他锁后，其他任何用户对该数据不能加任何锁，直到释放锁，

select xx from tb for update;

**表锁**：表锁是mysql中粒度最大的一种锁，mysql存储引擎大部分都支持，加锁简单，缺点：锁住整张表，其他查询需要进行等待，效率低下。

**行锁**：行级锁是粒度最小的一种锁，加锁比较慢，开销大，容易出现死锁

避免死锁：

读取的时候尽量指定顺序，减少大事务，在同一事物中尽量锁定所有资源，

1. **事物问题：四大特性？四大隔离级别？七大传播特性？**

四大特性：ACID 原子性、一致性、隔离性、永久性

四大隔离级别：脏读、未提交读、已提交读、可序列化。

七大传播特性：

   1.支持当前事务，如果当前没有事务，就新建一个事务

     2.支持当前事务，如果当前没有事务，就以非事务方式执行

     3.支持当前事务，如果当前没有事务，就抛出异常

     4.新建事务，如果当前存在事务，把当前事务挂起

     5.以非事务方式执行操作，如果当前存在事务，就把当前事务挂起

     6.以非事务方式执行，如果当前存在事务，则抛出异常

     7.加入当前正要执行的事务不在另外一个事务里，那么就起一个新的事务

     8.如果当前在事务中，即以事务的形式运行，如果当前不再一个事务中，那么就以非事务的形式运行

     9.必须在一个事务中运行。也就是说，他只能被一个父事务调用。否则，他就要抛出异常

     10.当执行到ServiceB.methodB的时候，ServiceA.methodA所在的事务就会挂起，

**缓存问题：**

**Redis和Memcache哪些区别：**

1. 支持的数据类型不同，redis支持string,list,set,sortedSet,map（Hash）等数据类型，memcache只支持stirng类型
2. 线程模型不同：reids采用单线程模型，memache是多线程模型。
3. Memache不支持数据的持久化，redis支持数据持久化
4. 集群管理方式不同：Memache本身不支持分布式，可以在客户端通过一致性Hash这样的分布式算法来实现。客户端向节点发送数据之前，首先通过内置的分布式算法来计算目标节点，然后直接将数据发送到该节点。缺陷：节点宕机，容易出现数据丢失的情况。

Redis对于集群管理的功能是比较强大的。可以通过RedisCluster来实现集群。引入Master和Slave节点，每个Master节点会有两个冗余的slave节点，当Master宕机，集群会自动选取一个Slave作为Master节点。

**Redis有哪些数据结构？应用场景？**

1. String类型，应用场景：太多了，缓存，计数器，等等等等。。。。

常用命令：

get,set,getset(设置新值，返回旧值) mget(得到所有给定的key的value)

setex(设置值并指定过期时间，) setnx（设置值，如果存在，返回-1，一般用作分布式锁）（mset同时设定多个key） incr decr

1. List 链表，redis中的链表是双向的，应用场景：消息队列；例如评论列表，关注列表，收藏列表等等。常用命令：lpush,rpush，lpop，rpop
2. Hash(Map) 类似于Java中的HashMap 引用场景：存储用户信息，id为key,具体信息为value.
3. Set集合 （不重复元素的集合列表） 提供了交集、并集合差集的操作，例如：将每个用户的好友存储在Set中，很容易获取用户的共同好友，（QQ和微信的共同好友）
4. SortedSet （有序集合）对集合的元素添加一个分数，实现排序 应用场景：可以利用sortedSet实现优先级的队列。

**Redis中的key如何寻址？**

**Redis持久化？**

两种方式:

第一种：RDB，发起快照，在生成快照时，fork出一个子进程，在子进程中循环所有数据，将数据写入到RDB文件。缺陷：一旦系统挂机，在上次发起快照到挂机这段时间的数据会全部丢掉。第二种方式：AOF（Append Only File） aof的开销比RDB要低很多，建议使用。

具体实现：

**Redis过期策略：**

**Redis分布式锁实现方案：**

**Redis主从复制：**

**Redis集群：**