1. MySQL中myisam与innodb的区别?

- InnoDB支持事物，而MyISAM不支持事物
- InnoDB支持行级锁，而MyISAM支持表级锁
- InnoDB支持MVCC, 而MyISAM不支持
- InnoDB支持外键，而MyISAM不支持
- InnoDB不支持全文索引，而MyISAM支持。

2. 事务的特性

- 原子性：是指事务包含所有操作要么全部成功，要么全部失败回滚。
- 一致性：指事务必须使数据库从一个一致性状态变换成另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。
  拿转账来说，假设用户 A 和用户 B 两者的钱加起来一共是 5000，那么不管 A 和 B 之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是 5000，这就是事务的一致性。
- 隔离性：是当多个用户并发访问数据库时，比如操作同一张表时，数据表为每个用户开启的事务，不能被其他事务所干扰，多个并发事务之间要相互隔离。
- 持久性：持久性是指一个事务一旦被提交，那么对数据库中的数据的改变就是永久的，即便是在数据库系统遇到故障的性况下也不会丢失提交事务的操作。

3. 并发操作问题

- 脏读：脏读是指在一个事务处理过程中读取到了另外一个未提交事务中的数据。
- 不可重复读：不可重复读是指在对于数据库中的某个数据，一个事务范围内多次查询却返回了不同的数据值，这是由于在查询间隔，被另一个事务修改并提交了。
- 虚读(幻读)：幻读发生在当两个完全相同的查询执行时，第二次查询所返回的结果集跟第一个查询不相同。
  比如两个事务操作，A 事务查询状态为 1 的记录时，这时 B 事务插入了一条状态为 1 的记录，A 事务再次查询返回的结果不一样。

4. 事务的隔离级别

- Serializable(串行化)：可避免脏读、不可重复读、幻读。（就是串行化读数据）
- Repeatable read(可重复读)：可避免脏读、不可重复读的发生。
- Read committed(读已提交)：可避免脏读的发生。
- Read uncommitted(读未提交)：最低级别，任何情况都无法保证。

在 MySQL 数据库中，支持上面四种隔离级别，默认的为 Repeatable read (可重复读)；而在 Oracle 数据库中，只支持 Serializable (串行化)级别和 Read committed (读已提交)这两种级别，其中默认的为 Read committed 级别。##

5. 索引是什么?

索引是表的目录，在查找内容之前可以先在目录中查找索引位置，以此快速定位查询数据。对于索引，会保存在额外的文件中。

索引是帮助MySQL高效获取数据的数据结构。

6. 索引能干什么?有什么好处?

当表中的数据量越来越大时，索引对于性能的影响愈发重要。索引能够轻易将查询性能提高好几个数量级，总的来说就是可以明显的提高查询效率。

7. 索引的种类有哪些?

1、从存储结构上来划分：BTree索引（B-Tree或B+Tree索引），Hash索引，full-index全文索引，R-Tree索引。这里所描述的是索引存储时保存的形式，

2、从应用层次来分：普通索引，唯一索引，复合索引

3、根据中数据的物理顺序与键值的逻辑（索引）顺序关系：聚集索引，非聚集索引。

平时讲的索引类型一般是指在应用层次的划分。

- 普通索引：即一个索引只包含单个列，一个表可以有多个单列索引
- 复合索引：多列值组成一个索引，专门用于组合搜索，其效率大于索引合并
- 唯一索引：索引列的值必须唯一，但允许有空值

8. 为什么 MySQL 的索引要使用 B+树而不是其它树形结构? 比如 B 树?

B-tree：因为B树不管叶子节点还是非叶子节点，都会保存数据，这样导致在非叶子节点中能保存的指针数量变少（有些资料也称为扇出），指针少的情况下要保存大量数据，只能增加树的高度，导致IO操作变多，查询性能变低；

Hash：虽然可以快速定位，但是没有顺序，IO复杂度高。

二叉树：树的高度不均匀，不能自平衡，查找效率跟数据有关（树的高度），并且IO代价高。

红黑树：树的高度随着数据量增加而增加，IO代价高。

不使用平衡二叉树的原因如下：

最大原因：深度太大(因为一个节点最多只有2个子节点)，一次查询需要的I/O复杂度为O(lgN),而b+tree只需要O(log_mN),而其出度m非常大，其深度一般不会超过4
平衡二叉树逻辑上很近的父子节点，物理上可能很远，无法充分发挥磁盘顺序读和预读的高效特性。

9. MyISAM和InnoDB实现BTree索引方式的区别

MyISAM

B+Tree叶节点的data域存放的是数据记录的地址。在索引检索的时候，首先按照B+Tree搜索算法搜索索引，如果指定的Key存在，则取出其 data 域的值，然后以 data 域的值为地址读取相应的数据记录。这被称为“非聚簇索引”。
索引文件和数据文件是分离的

InnoDB

- InnoDB 的 B+Tree 索引分为主索引（聚集索引）和辅助索引(非聚集索引)。一张表一定包含一个聚集索引构成的 B+ 树以及若干辅助索引的构成的 B+ 树。
- 辅助索引的存在并不会影响聚集索引，因为聚集索引构成的 B+ 树是数据实际存储的形式，而辅助索引只用于加速数据的查找，所以一张表上往往有多个辅助索引以此来提升数据库的性能。
- 就很容易明白为什么不建议使用过长的字段作为主键，因为所有辅助索引都引用主索引，过长的主索引会令辅助索引变得过大。再例如，用非单调的字段作为主键在InnoDB中不是个好主意，因为InnoDB数据文件本身是一颗B+Tree，非单调的主键会造成在插入新记录时数据文件为了维持B+Tree的特性而频繁的分裂调整，十分低效，而使用自增字段作为主键则是一个很好的选择。

10. 什么是最左匹配原则?

最左优先，以最左边的为起点任何连续的索引都能匹配上。同时遇到范围查询(>、<、between、like)就会停止匹配。
例如：b = 2 如果建立(a,b)顺序的索引，是匹配不到(a,b)索引的；但是如果查询条件是a = 1 and b = 2,就可以，因为**优化器会自动调整a,b的顺序。**再比如a = 1 and b = 2 and c > 3 and d = 4 如果建立(a,b,c,d)顺序的索引，d是用不到索引的，因为c字段是一个范围查询，它之后的字段会停止匹配。

最左匹配原则的原理

MySQL中的索引可以以一定顺序引用多列，这种索引叫作联合索引.最左匹配原则都是针对联合索引来说的

- 我们都知道索引的底层是一颗B+树，那么联合索引当然还是一颗B+树，只不过联合索引的健值数量不是一个，而是多个。构建一颗B+树只能根据一个值来构建，因此数据库依据联合索引最左的字段来构建B+树。
  例子：假如创建一个（a,b)的联合索引，那么它的索引树是这样的可以看到a的值是有顺序的，1，1，2，2，3，3，而b的值是没有顺序的1，2，1，4，1，2。所以b = 2这种查询条件没有办法利用索引，因为联合索引首先是按a排序的，b是无序的。

同时我们还可以发现在a值相等的情况下，b值又是按顺序排列的，但是这种顺序是相对的。所以最左匹配原则遇上范围查询就会停止，剩下的字段都无法使用索引。例如a = 1 and b = 2 a,b字段都可以使用索引，因为在a值确定的情况下b是相对有序的，而a>1and b=2，a字段可以匹配上索引，但b值不可以，因为a的值是一个范围，在这个范围中b是无序的。

优点：最左前缀原则的利用也可以显著提高查询效率，是常见的MySQL性能优化手段。

11. 哪些列上适合创建索引？创建索引有哪些开销?

经常需要作为条件查询的列上适合创建索引，并且该列上也必须有一定的区分度。创建索引需要维护，在插入数据的时候会重新维护各个索引树（数据页的分裂与合并），对性能造成影响

12. 索引这么多优点，为什么不对表中的每一个列创建一个索引呢?

1. 当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。
2. 索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。
3. 创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

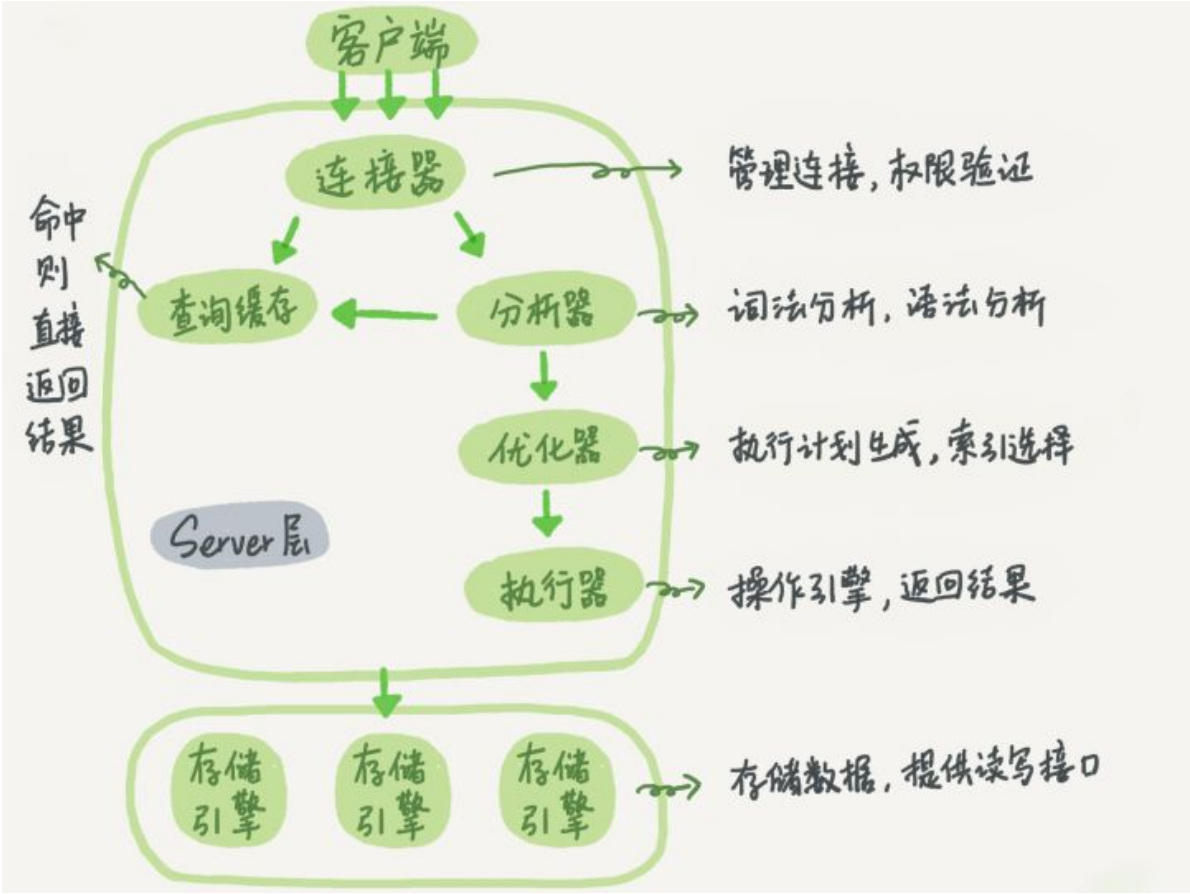
13. MySQL建表的约束条件有哪些?

- 主键约束（Primay Key Coustraint） 唯一性，非空性
- 唯一约束 （Unique Counstraint）唯一性，可以空，但只能有一个
- 检查约束 (Check Counstraint) 对该列数据的范围、格式的限制
- 默认约束 (Default Counstraint) 该数据的默认值
- 外键约束 (Foreign Key Counstraint) 需要建立两表间的关系并引用主表的列

14. MySQL执行查询的过程?

1. 客户端通过TCP连接发送连接请求到mysql连接器，连接器会对该请求进行权限验证及连接资源分配
2. 查缓存。（当判断缓存是否命中时，MySQL不会进行解析查询语句，而是直接使用SQL语句和客户端发送过来的其他原始信息。所以，任何字符上的不同，例如空格、注解等都会导致缓存的不命中。）

3. 语法分析（SQL语法是否写错了）。 如何把语句给到预处理器，检查数据表和数据列是否存在，解析别名看是否存在歧义。
4. 优化。是否使用索引，生成执行计划。
5. 交给执行器，将数据保存到结果集中，同时会逐步将数据缓存到查询缓存中，最终将结果集返回给客户端。



15. MySQL的binlog有有几种录入格式?分别有什么区别?

有三种格式,statement,row和mixed.

- statement模式下,记录单元为语句.即每一个sql造成的影响会记录.由于sql的执行是有上下文的,因此在保存的时候需要保存相关的信息,同时还有一些使用了函数之类的语句无法被记录复制.
- row级别下,记录单元为每一行的改动,基本是可以全部记下来但是由于很多操作,会导致大量行的改动(比如alter table),因此这种模式的文件保存的信息太多,日志量太大。
- mixed. 一种折中的方案,普通操作使用statement记录,当无法使用statement的时候使用row. 此外,新版的MySQL中对row级别也做了一些优化,当表结构发生变化的时候,会记录语句而不是逐行记录.