外键
内联和外联
InnoDB和MyISAM的区别
MVCC的实现
InnoDB的行锁算法
行级锁
MySQL索引
事务的ACID
事务并发出现的问题
事务的隔离级别
为什么要有读锁？不加可不可以？
隔离级别
死锁
索引
B树和B+树的差异
聚集索引和非聚集索引
主键索引
二级索引
覆盖索引

外键

如果表A中的一个字段是表B中的主键，则这个字段就可以是表A的外键。

内联和外联

```
内联：inner join
    a inner join b on a.id=b.id
    查找两张表都有的id记录
外联：
    左外联：left join
        a left join b on a.id=b.id
        只要a表中有即可，b表没有则用null代替
    右外联：right join
        a right join b on a.id=b.id
        只要b表中有即可，a表没有则用null代替
```

InnoDB和MyISAM的区别

```
mysql5.5之前使用的是MyISAM,之后使用的是InnoDB

是否支持行级锁：
    MyISAM：不支持，只支持表级锁，性能低
    InnoDB：支持
是否支持事务：
    MyISAM：不支持
    InnoDB：支持
是否支持外键：
    MyISAM：不支持
    InnoDB：支持
是否支持数据库崩溃后的恢复：
```

MyISAM：不支持
InnoDB：支持。 redo log(重做日志)保证事务的持久性，undo log(回滚日志)保证事务的原子性，锁和MVCC保证事务的隔离性。
是否支持MVCC：
MyISAM：不支持
InnoDB：支持

MVCC的实现

MVCC:一种多版本并发控制机制
解决的问题：
并发访问数据库时对正在事务中处理的数据做多版本控制。来避免写操作的堵塞，进而引发读操作的并发问题；
行级锁可以解决并发问题，但是开销大，因此使用MVCC来替代行级锁可以减少系统开销。
具体实现：
InnoDB通过为每一行记录添加两个额外隐藏的值来实现MVCC。一个值是记录这行数据何时被创建(事务id)，；另外一个是这行数据何时过期。
这个时间存储的是系统版本号，每开始一个新的事务，版本号都会增加。

保存着两个版本号可以使多数读操作不用加锁，提升性能。
MVCC只在读取已提交和可重复读这两个隔离级别下工作。

InnoDB的行锁算法

Record Lock：单个行记录加锁，最普通的行锁
Gap Lock：间隙锁，锁定一个间隙范围，不包括记录本身。锁的是间隙
缺点：
当锁定一个范围的时候，即使不存在的键值也会被锁定，造成在锁定的时候无法插入锁定键值范围内的任何数据。
Next-key Lock：锁定一个范围，包括记录本身

行级锁

主要使用在InnoDB存储引擎中
优点：锁的粒度小，并发度高，争用率低
缺点：开销大，加锁慢，容易出现死锁

分类：
共享锁(读锁)：多个事务对同一数据可以共享一把锁，都能访问到数据，但是无法修改；
排他锁(写锁)：一个事务如果获取了一个数据行的排他锁，其他事务就无法再获取该行的其他锁了。获得排他锁的事务可以对该行数据进行读写操作；
意向共享锁：是一种表级锁，事务给数据行加上共享锁之前必须先取得该表的意向共享锁；
意向排他锁：是一种表级锁，事务给数据行加上排他锁之前必须先取得该表的意向排他锁；
意向锁是InnoDB自动加的，无需用户干预。

行级锁的实现方式：
行级锁是给索引项加锁。只有通过索引条件检索数据才会使用行级锁，否则InnoDB使用的是表级锁；

MySQL索引

底层使用B+树。
为什么没用哈希索引：
1.hash冲突
2.hash不支持范围查询

为什么使用B+树而不用B树？
1.B+树的数据集中在叶子节点，且相邻节点有指针，能更好进行区间查找。
2.B树进行区间查找只能进行中序遍历。
3.使用B+树的最大好处是能减少IO次数。

事务的ACID

原子性：动作要么完成，要么不做
一致性：事务执行前后，数据应保持一致
隔离性：并发访问数据库，一个用户的事务不被其他事务所干扰
持久性：事务提交后，对数据的改变是永久的。

事务并发出现的问题

脏读：
丢失修改：
不可重复读：
幻读：

事务的隔离级别

读取未提交：
读取已提交：
可重复读：
串行化：

为什么要有读锁？不加可不可以？

读锁可以保证多个线程在读数据的时候的一致性。但是如果线程不修改数据，只读取，为什么就不可以呢？
读锁主要是为了和写锁形成互斥关系，读的时候不能写，写的时候也不能读。如果不加读锁，则在数据被修改的时候，会读取到不同的数据。

隔离级别

读取未提交READ-UNCOMMITTED：
　　一个事务正在修改某数据，但是还未提交，另外的事务这时候可以读取到这个还未提交的数据； 脏读、不可重复读、幻读。
读取已提交READ-COMMITTED：
　　只要是已提交的数据，并发的事务就可以读取； 不可重复读、幻读；
可重复读REPEATABLE-READ：
　　对同一数据的多次读取都是一样的，除非被自身事务修改； 幻读
串行化SERIALIZABLE：
　　所有事务串行执行；

脏读：读取到未提交的数据；
不可重复读：事务在读取数据过程中，其他事务对数据进行修改，导致原事务前后读取数据不一样；
幻读：事务在读取数据过程中，其他事务有插入数据的操作，导致原事务读取的数据不一样。

死锁

多个事务以不同顺序访问锁定的资源，会产生死锁。

InnoDB中处理死锁的方案是将持有最小行级排他锁的事务进行回滚。

索引

MySQL使用B+树作为索引的数据结构。

不使用Hash作为索引有两个原因：
1.Hash冲突问题；
2.Hash不支持顺序和范围查询

B树和B+树的差异

B树的所有节点都会存放键和数据，但是B+树只有叶节点才会存放键和数据，其他节点值存放键；
B树的所有叶节点都是独立的，B+树的子节点有一个指针会指向相邻叶节点，这个指针能够很好的进行范围查找；
B树的检索等价于二分查找，有可能还没有到达叶节点就搜索结束了；而B+树的检索很稳定，都是要到叶节点才结束；
MySQL中数据的存储是以页为单位的，所以每读取到一个页就发生一次磁盘IO。对于B树而言，每个节点都放键和数据，因此会产生更高的高度，导致多次磁盘IO；B+树的高度要低于B树，自然磁盘IO次数也少于B树。

聚集索引和非聚集索引

非聚集索引：
在MyISAM中，B+树的叶节点的data域存放的是数据记录的地址。在查找的时候先找到对应的数据记录地址，然后再根据这个地址读取相应的数据。索引结构和数据分开存放。
二级索引属于非聚集索引，虽然非聚集索引叶节点存放的是数据指针，但是二级索引的叶节点放的是主键，因此可能会根据主键再回表查询。
优点：
更新代价比聚集索引小
缺点：
索引非有序数据比较慢；
可能会二次查询(回表)。当索引到对应的指针或主键后，可能还需要根据指针或主键再到数据文件或表中查询。
一定会回表查询吗？
不一定，如果查询的本身就是这个索引的话：

```
SELECT name FROM table WHERE name='guang19';
```

聚集索引：
索引结构和数据一起存放的索引。主键索引属于聚集索引。在B+树中每个非叶子节点存储索引，叶节点存储索引和索引对应的数据。
优点：
查询速度快，找到索引就相当于找到了数据；
缺点：
更新聚集索引列的代价高；
如果索引的数据不是有序的，则查询的速度会比较慢。

主键索引

一张表只能有一个主键，且不能为null，不能重复。
如果没有显示的主键，则会先检查是否有唯一的字段，如果有，则作为主键。否则会自动创建一个自增主键。

二级索引

又被称为辅助索引，二级索引叶节点存储的是主键。
二级索引有：唯一索引、普通索引、前缀索引、全文索引。

唯一索引：唯一索引的属性列不能出现重复的数据，但是允许有null值，一张表允许创建多个唯一索引；
普通索引：为了加快查询数据，一张表允许创建多个普通索引，允许数据重复和null值；
前缀索引：只适用于字符串类型的数据。根据字符串前几个字符创建的索引；
全文索引：检索大文本数据中的关键字信息。

覆盖索引

一个索引包含所有需要查询字段的值。

```
SELECT name FROM table WHERE name='guang19';
```