**2. sql 语句应该考虑哪些安全性？（新浪网技术部）**

防止 Sql 注入，对特殊字符进行转义、过滤或使用预编译的 sql 语句绑定变量。

最小权限原则，特别是不要用 root 账户，为不同的类型的动作或者组建使用不同的账户。

当 sql 运行出错时，不要把数据库返回的错误信息全部显示给用户，以防止泄露服务器和数据库相关信息。

**3. 简单描述 mysql 中，索引，主键，唯一索引，联合索引的区别，对数据库的性能有什么影响（从读写两方面）（新浪网技术部）**

索引是一种特殊的文件(InnoDB 数据表上的索引是表空间的一个组成部分)，它们包含对数据表里所有记录的引用指针。

普通索引(由关键字 KEY 或 INDEX 定义的索引)的唯一任务是加快对数据的访问速度。

普通索引允许被索引的数据列包含重复的值。如果能确定某个数据列将只包含彼此各不相同的值，在为这个数据列创建索引的时候就应该用关键字 UNIQUE 把它定义为一个唯一索引。

也就是说，唯一索引可以保证数据记录的唯一性。

主键，是一种特殊的唯一索引，在一张表中只能定义一个主键索引，主键用于唯一标识一条记录，使用关键字 PRIMARY KEY 来创建。

索引可以覆盖多个数据列，如像 INDEX(columnA, columnB)索引，这就是联合索引。

索引可以极大的提高数据的查询速度，但是会降低插入、删除、更新表的速度，因为在执行这些写操作时，还要操作索引文件。

**7. MySQL 数据库中的字段类型 varchar 和 char 的主要区别是什么？哪种字段的查找效率要高，为什么?**

区别一，定长和变长

char 表示定长，长度固定，varchar表示变长，即长度可变。当所插入的字符串超出它们的长度时，视情况来处理，如果是严格模式，则会拒绝插入并提示错误信息，如果是宽松模式，则会截取然后插入。如果插入的字符串长度小于定义长度时，则会以不同的方式来处理，如char（10），表示存储的是10个字符，无论你插入的是多少，都是10个，如果少于10个，则用空格填满。而varchar（10），小于10个的话，则插入多少个字符就存多少个。

varchar怎么知道所存储字符串的长度呢？实际上，对于varchar字段来说，需要使用一个（如果字符串长度小于255）或两个字节（长度大于255）来存储字符串的长度。

区别之二，存储的容量不同

对 char 来说，最多能存放的字符个数 255，和编码无关。

而 varchar 呢，最多能存放 65532 个字符。VARCHAR 的最大有效长度由最大行大小和使用的字符集确定。整体最大长度是 65,532字节

MySQL最大行长度是 65535 字节，实际行存储从第二个字节开始，然后还需要两个字节来存放字符串的长度。如果数据表只有一个varchar类型的字段的话，有效长度就是 65535 - 1 - 2= 65532

由字符集来确定，字符集分单字节和多字节

Latin1 一个字符占一个字节，最多能存放 65532 个字符

GBK 一个字符占两个字节， 最多能存 32766 个字符

UTF8 一个字符占三个字节， 最多能存 21844 个字符

注意，char 和 varchar 后面的长度表示的是字符的个数，而不是字节数。

两相比较，char 的效率高，没有碎片，尤其更新比较频繁的时候，方便数据文件指针的操作。但不够灵活，在实际使用时，应根据实际需求来选用合适的数据类型。

举两个例说明一下实际长度的计算。

　　a) 若一个表只有一个varchar类型，如定义为

　　create table t4(c varchar(N)) charset=gbk;则此处N的最大值为(65535-1-2)/2= 32766。

　　减1的原因是实际行存储从第二个字节开始';

　　减2的原因是varchar头部的2个字节表示长度;

　　除2的原因是字符编码是gbk。

　　b) 若一个表定义为

create table t4(c int, c2 char(30), c3 varchar(N)) charset=utf8;则此处N的最大值为 (65535-1-2-4-30\*3)/3=21812

减1和减2与上例相同;

　　减4的原因是int类型的c占4个字节;

　　减30\*3的原因是char(30)占用90个字节，编码是utf8。

**[!!]14. 请简述项目中优化 sql 语句执行效率的方法,从哪些方面,sql 语句性能如何分析?**

尽量选择较小的列

将where中用的比较频繁的字段建立索引

select子句中避免使用‘\*’

避免在索引列上使用计算、not in 和<>等操作

当只需要一行数据的时候使用limit 1

保证单表数据不超过200W，适时分割表。

针对查询较慢的语句，可以使用explain 来分析该语句具体的执行情况。

**19. 数据库中的事务是什么?**

事务（transaction）是作为一个单元的一组有序的数据库操作。如果组中的所有操作都成功，则认为事务成功，即使只有一个操作失败，事务也不成功。如果所有操作完成，事务则提交，其修改将作用于所有其他数据库进程。如果一个操作失败，则事务将回滚，该事务所有操作的影响都将取消。

ACID 四大特性,原子性、隔离性、一致性、持久性。

**[!]25. 请简述数据库设计的范式及应用。**

一般第 3 范式就足以，用于表结构的优化，这样做既可以避免应用程序过于复杂同时也避免了 SQL 语句过于庞大所造成系统效率低下。

第一范式：关系中的每个属性都不可再分，是所有关系型数据库的最基本要求

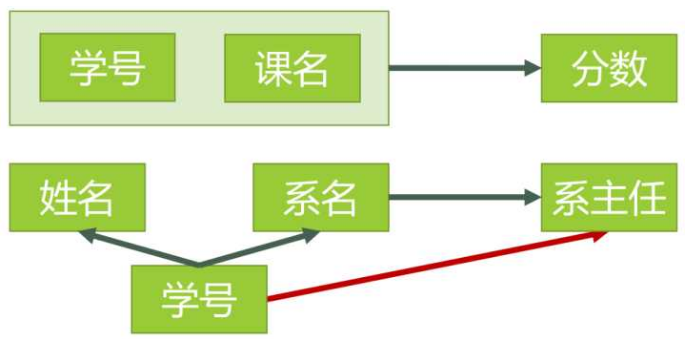
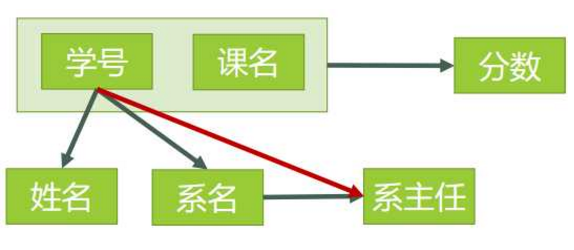
第二范式：在1NF的基础之上，消除了非主属性对于码的部分函数依赖

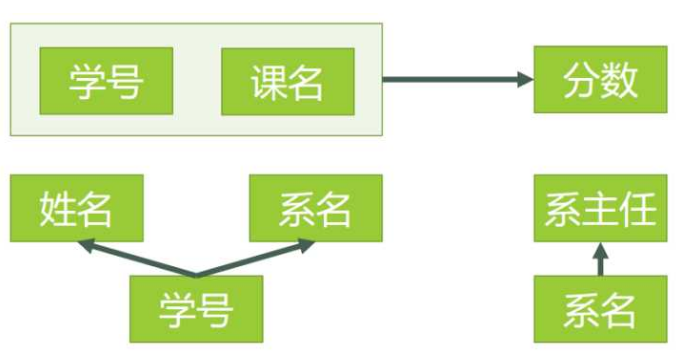
第三范式：在2NF的基础之上，消除了非主属性对于码的传递函数依赖

BCNF 范式：3NF 的基础上消除主属性对于码的部分与传递函数依赖

在实际使用中，可以根据需求适当的逆范式。

<https://www.zhihu.com/question/24696366>





**[!!]32. 解释 MySQL 外连接、内连接与自连接的区别 （小米）**

先说什么是交叉连接，交叉连接又叫笛卡尔积，它是指不使用任何条件，直接将一个表的所有记录和另一个表中的所有记录一一匹配。

内连接则是只有条件的交叉连接，根据某个条件筛选出符合条件的记录，不符合条件的记录不会出现在结果集中，即内连接只连接匹配的行。

而外连接其结果集中不仅包含符合连接条件的行，而且还会包括左表、右表或两个表中的所有数据行，这三种情况依次称之为左外连接，右外连接，和全外连接。

内连接：在每个表中找出符合条件的共有记录

左连接：根据左表的记录，在被连接的右表中找出符合条件的记录与之匹配

找不到与左表匹配的，用null表示。

右连接：根据右表的记录，在被连接的左表中找出符合条件的记录与之匹配，

找不到匹配的，用null填充

左连接和右连接可以互换，MySQL 目前还不支持全外连接。

全连接：返回符合条件的所有表的记录，没有与之匹配的，用null表示

自连接：连接的表都是同一个表，同样可以由内连接，外连接各种组合

**[!!!]37. 简述在 MySQL 数据库中 MyISAM 和 InnoDB 的区别 （亿邮）**

区别主要有以下几个：

1) Innodb 支持事务， myisam 不支持；

2) Innodb 支持有条件的行级锁， myisam 不支持；

3) innodb 支持外键，myisam不支持

4) Innodb 是索引组织表， myisam 是堆表；

5) Innodb 表不容易损坏， myisam 容易

6) Innodb不能通过直接拷贝表文件的方法拷贝表到另外一台机器,myisam可以

7) Innodb 支持崩溃后的安全恢复，myisam不支持;

8) Innodb不保存表的具体行数，myisam保存，当count()语句带where条件时则扫描全表

9)对带有AUTO\_INCREMENT属性的字段，InnoDB中必须包含只有该字段的索引，但是在MyISAM表中，可以和其他字段一起建立联合索引。

10) DELETE FROM table的时候，Innodb是一行行删除，myisam是先drop表再重建表

**1、MySQL的复制原理以及流程**

基本原理：

(1) master将数据库更改记录到二进制日志(binary log)中（这些记录叫做二进制日志事件，binary log events）；

(2) slave将master的binary log events拷贝到它的中继日志(relay log)；

(3) slave重做中继日志中的事件，将改变反映它自己的数据。

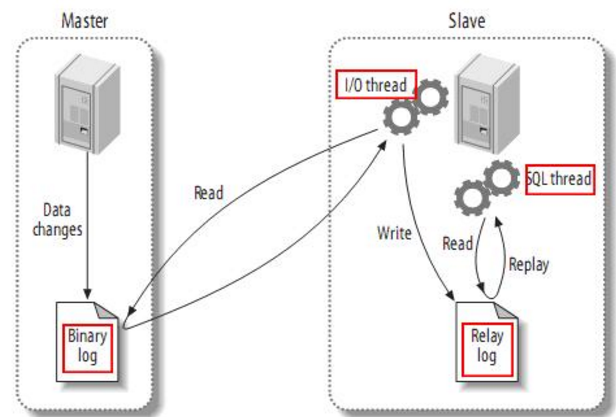
MySQL 的复制是基于如下 3 个线程的交互（ 多线程复制里面应该是 4 类线程）：

a. Master的 binlog dump 线程从二进制日志(binlog file)中读取事件(binlog event)传到slave，如果已经跟上master，它会睡眠并等待master产生新的binlog event

b. Slave的 I/O 线程接收 Master 传过来的 binlog event，并写入中继日志(relay log)

c. Slave的 SQL线程从中继日志读取事件，并重放其中的事件而更新slave的数据，使其与master中的数据一致。

d. 如果是多线程复制，无论是 5.6 库级别的假多线程还是 MariaDB 或者 5.7 的真正的多线程复制， SQL 线程只做 coordinator(协调者)，只负责把 relay log 中的 binlog读出来然后交给 worker 线程， woker 线程负责具体 binlog event 的执行；



**2、问了innodb的事务与日志的实现方式**

(1)、有多少种日志；

错误日志：记录出错信息，也记录一些警告信息或者正确的信息。

查询日志：记录所有对数据库请求的信息，不论这些请求是否得到了正确的执行。

慢查询日志：设置一个阈值，将运行时间超过该值的所有SQL语句都记录到慢查询的日志文件中。

二进制日志：记录对数据库执行更改的所有操作。

中继日志：

事务日志：

(2)、事务的4种隔离级别

读未提交(read uncommitted)

读已提交(read committed)

可重复读(repeatable read)

串行(serializable)

(3)、事务是如何通过日志来实现的，说得越深入越好。

事务日志是通过redo和innodb的存储引擎日志缓冲（Innodb log buffer）来实现的，当开始一个事务的时候，会记录该事务的lsn(log sequence number)号; 当事务执行时，会往InnoDB存储引擎的日志

的日志缓存里面插入事务日志；当事务提交时，必须将存储引擎的日志缓冲写入磁盘（通过innodb\_flush\_log\_at\_trx\_commit来控制），也就是写数据前，需要先写日志。这种方式称为“预写日志方式”

**3、你们数据库是否支持emoji表情，如果不支持，如何操作？**

如果是utf8字符集的话，需要升级至utf8\_mb4方可支持

**4、表中有大字段X(例如：text类型)，且字段X不会经常更新，以读为为主，请问**

(1)、您是选择拆成子表，还是继续放一起；

(2)、写出您这样选择的理由。

答：拆带来的问题：连接消耗 + 存储拆分空间；不拆可能带来的问题：查询性能；

如果能容忍拆分带来的空间问题,拆的话最好和经常要查询的表的主键在物理结构上放置在一起(分区) 顺序IO,减少连接消耗,最后这是一个文本列再加上一个全文索引来尽量抵消连接消耗

如果能容忍不拆分带来的查询性能损失的话:上面的方案在某个极致条件下肯定会出现问题,那么不拆就是最好的选择

**5**、MySQL中InnoDB引擎的行锁是通过加在什么上完成(或称实现)的？为什么是这样子的？

答：InnoDB是基于索引来完成行锁

例: select \* from tab\_with\_index where id = 1 for update;

for update 可以根据条件来完成行锁锁定,并且 id 是有索引键的列,

如果 id 不是索引键那么InnoDB将完成表锁,,并发将无从谈起

**6**、**你是否做过主从一致性校验，如果有，怎么做的，如果没有，你打算怎么做？**

主从一致性校验有多种工具 例如checksum、mysqldiff、pt-table-checksum等

**7**、**innodb的四个关键特性是什么 ？**

(1) 插入缓冲（insert buffer）

(2) 两次写（Double write）

(3) 自适应哈希索引（adaptive hash index）

(4) 预读(read ahead)

开放性问题：据说是腾讯的

**一个6亿的表a，一个3亿的表b，通过外间tid关联，你如何最快的查询出满足条件的第50000到第50200中的这200条数据记录**。

1、如果A表TID是自增长,并且是连续的,B表的ID为索引

select \* from a,b where a.tid = b.id and a.tid>500000 limit 200;

2、如果A表的TID不是连续的,那么就需要使用覆盖索引.TID要么是主键,要么是辅助索引,B表ID也需要有索引。

select \* from b , (select tid from a limit 50000,200) a where b.id = a .tid;