8.MySQL数据库&SQL语句面试（下）

**常用SQL语句**

创建表 create table 表名（列名 数据类型）

增加 insert into 表名【（列名1，列名2）】 values（值1，值2）

删除 delete from 表名 where 过滤条件

更新 update 表名 set 列名=新的值 where 过滤条件

查询 select\*from 表名

**SQL语句主要分为哪几类**

DML 数据操作语言 例如 insert delete update

DQL 数据库查询语言 例如 select

TCL 事物控制语言 例如 提交 回滚

DDL 数据定义语言 例如 创建表…

DCL 数据控制语言 例如 授权…

**SQL约束有哪几种？**

not null 非空

unique 唯一

primary key 主键约束

foreign key 外键约束

check 条件约束

**什么是子查询**

子查询就是查询语句的嵌套,即在外部查询中还包含一个内部查询。当一个查询是另一个查询的条件时,称之为子查询。

**子查询的三种情况**

where 型子查询

from 型子查询

exists型子查询

**varchar与char的区别**

区别1 char 表示定长，长度固定，varchar表示变长，即长度可变。

区别2 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。

**varchar(50)中50的涵义**

最多存放50个字符

**int(20)中20的涵义**

最多显示20个字符串

**mysql为什么这么设计**

**FLOAT和DOUBLE的区别是什么？**

**变量类型不同**

float属于单精度型浮点数据。

double属于双精度型浮点数据。

**指数范围不同**

float的指数范围为-127~128。

double而double的指数范围为-1023~1024

**占用内存空间不同**

float占4个字节（32位）内存空间，其数值范围为3.4E-38～3.4E+38。

double占8 个字节（64位）内存空间，其数值范围为1.7E-308～1.7E+308。

**有效位数不同**

float只能提供七位有效数字。

double可提供16位有效数字。

**drop、delete与truncate的区别**

相同点：  truncate和不带where自居的delete，以及drop都会删除表内的数据

不同点：

1、truncate和delete只删除数据不删除表的结构(定义)，而drop语句将删除表的结构被依赖的约束(constrain)，触发器(trigger)，索引(index)；依赖于该表的存储过程/函数将保留，但是变为invalid状态。

    2、delete命令是DML，删除的数据将存储在系统回滚段中，需要的时候，数据可以回滚恢复。

         而truncate，drop命令是DDL，删除的数据是操作立即生效，原数据不放到rollback segment中，不能回滚，数据不可以回滚恢复。

    3、delete命令，不会自动提交事务，操作会触发trigger；而truncate，drop命令，执行后会自动提交事务，操作不触发trigger。

4、速度：一般来说：drop > truncate > delete

**SQL优化**

**如何定位及优化SQL语句的性能问题？**

**SQL的生命周期？**

1. 建立服务器与数据库连接
2. 数据库拿到SQL
3. 解析执行
4. 读取数据到内存，进行业务逻辑处理
5. 发给客户端
6. 关闭连接，释放资源

**大表数据查询，怎么优化**

1．合理使用索引

2．避免或简化排序

3．消除对大型表行数据的顺序存取

4．避免相关子查询

5．避免困难的正规表达式

6．使用临时表加速查询

7．用排序来取代非顺序存取

**超大分页怎么处理？**

方法 一 ： 用INDEX和ROWCOUNT结合来做分页处理。

第二种方法：用临时表的方法来分页处理

第三种方法：用Cursor来做分页处理

**mysql分页**

limit （值1，值2）

值1：起始页

值2:分页中每页显示多少页

**关心过业务系统里面的sql耗时吗？统计过慢查询吗？对慢查询都怎么优化过？**

**为什么要尽量设定一个主键？**

主键是数据库确保数据行在整张表唯一性的保障,即使业务上本张表没有主键,也建议添加一个自增长的ID列作为主键.

设定了主键之后,在后续的删改查的时候可能更加快速以及确保操作数据范围安全.

**主键使用自增ID还是UUID？**

使用自增长做主键的优点：

1、很小的数据存储空间

2、性能最好

3、容易记忆

使用自增长做主键的缺点：

1、如果存在大量的数据，可能会超出自增长的取值范围

2、很难（并不是不能）处理分布式存储的数据表，尤其是需要合并表的情况下

3、安全性低，因为是有规律的，容易被非法获取数据

使用UUID做主键的优点：

1、它是独一无二的，出现重复的机会少

2、适合大量数据中的插入和更新操作，尤其是在高并发和分布式环境下

3、跨服务器数据合并非常方便

4、安全性较高

使用UUID做主键的缺点：

1、存储空间大（16 byte），因此它将会占用更多的磁盘空间

2、会降低性能

3、很难记忆

**字段为什么要求定义为notnull？**

字段设置为NOT NULL，可以插入空值。设置了NOT NULL之后便不能插入NULL值了，但仍然可以插入像 ' ' 这样的空值。另外空值是不占用空间的。

**优化查询过程中的数据访问**

访问数据太多导致查询性能下降

确定应用程序是否在检索大量超过需要的数据，可能是太多行或列

确认MySQL服务器是否在分析大量不必要的数据行

避免犯如下SQL语句错误

查询不需要的数据。解决办法：使用limit解决

多表关联返回全部列。解决办法：指定列名

总是返回全部列。解决办法：避免使用SELECT \*

重复查询相同的数据。解决办法：可以缓存数据，下次直接读取缓存

是否在扫描额外的记录。解决办法：

使用explain进行分析，如果发现查询需要扫描大量的数据，但只返回少数的行，可以通过如下技巧去优化：

使用索引覆盖扫描，把所有的列都放到索引中，这样存储引擎不需要回表获取对应行就可以返回结果。

改变数据库和表的结构，修改数据表范式

重写SQL语句，让优化器可以以更优的方式执行查询

**优化LIMIT分页**

尽可能的使用索引覆盖扫描，而不是查询所有的列。然后根据需要做一次关联操作再返回所需的列，对于偏移量很大的时候，这样做的效率会提升很大。以下面的查询作为示例，以主键info\_id（主键）作为覆盖的索引

**数据库优化**

**为什么要优化**

避免网站页面出现访问错误

增加数据库的稳定性

优化用户的体验

**数据库结构优化**

* 减少数据冗余
* 尽量避免数据维护中出现更新,插入和删除异常

**MySQL数据库cpu飙升到500%的话他怎么处理？**

当 cpu 飙升到 500%时，先用操作系统命令 top 命令观察是不是 mysqld 占用导致的，如果不是，找出占用高的进程，并进行相关处理。

如果是 mysqld 造成的， show processlist，看看里面跑的 session 情况，是不是有消耗资源的 sql 在运行。找出消耗高的 sql，看看执行计划是否准确， index 是否缺失，或者实在是数据量太大造成。

一般来说，肯定要 kill 掉这些线程(同时观察 cpu 使用率是否下降)，等进行相应的调整(比如说加索引、改 sql、改内存参数)之后，再重新跑这些 SQL。

也有可能是每个 sql 消耗资源并不多，但是突然之间，有大量的 session 连进来导致 cpu 飙升，这种情况就需要跟应用一起来分析为何连接数会激增，再做出相应的调整，比如说限制连接数等

**大表怎么优化某个表有近千万数据，CRUD比较慢，如何优化？分库分表了是怎么做的？分表分库了有什么问题？有用到中间件么？他们的原理知道么？**

数据千万级别之多，占用的存储空间也比较大，可想而知它不会存储在一块连续的物理空间上，而是链式存储在多个碎片的物理空间上。可能对于长字符串的比较，就用更多的时间查找与比较，这就导致用更多的时间。

可以做表拆分，减少单表字段数量，优化表结构。

在保证主键有效的情况下，检查主键索引的字段顺序，使得查询语句中条件的字段顺序和主键索引的字段顺序保持一致。

**垂直分表**

**适用场景**

用户表中既有用户的登录信息又有用户的基本信息，可以将用户表拆分成两个单独的表，甚至放到单独的库做分库。简单来说垂直拆分是指数据表列的拆分，把一张列比较多的表拆分为多张表**。**

**缺点**

主键会出现冗余，需要管理冗余列，并会引起Join操作，可以通过在应用层进行Join来解决。此外，垂直分区会让事务变得更加复杂；

**水平分表**

**适用场景**

水平拆分可以支持非常大的数据量。需要注意的一点是:分表仅仅是解决了单一表数据过大的问题，但由于表的数据还是在同一台机器上，其实对于提升MySQL并发能力没有什么意义，所以水平拆分最好分库 。

**水平切分的缺点**

水平拆分能够 支持非常大的数据量存储，应用端改造也少，但 分片事务难以解决 ，跨界点Join性能较差，逻辑复杂。

**MySQL的复制原理以及流程**

1）在Slave 服务器上执行sart slave命令开启主从复制开关，开始进行主从复制。

2）此时，Slave服务器的IO线程会通过在master上已经授权的复制用户权限请求连接master服务器，并请求从执行binlog日志文件的指定位置（日志文件名和位置就是在配置主从复制服务时执行change master命令指定的）之后开始发送binlog日志内容

3）Master服务器接收到来自Slave服务器的IO线程的请求后，二进制转储IO线程会根据Slave服务器的IO线程请求的信息分批读取指定binlog日志文件指定位置之后的binlog日志信息，然后返回给Slave端的IO线程。返回的信息中除了binlog日志内容外，还有在master服务器端记录的新的binlog文件名称，以及在新的binlog中的下一个指定更新位置。

4）当Slave服务器的IO线程获取到Master服务器上IO线程发送的日志内容、日志文件及位置点后，会将binlog日志内容依次写到Slave端自身的Relay Log（即中继日志）文件（[MySQL](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)-relay-bin.xxx）的最末端，并将新的binlog文件名和位置记录到master-info文件中，以便下一次读取master端新binlog日志时能告诉Master服务器从新binlog日志的指定文件及位置开始读取新的binlog日志内容

5）Slave服务器端的SQL线程会实时检测本地Relay Log 中IO线程新增的日志内容，然后及时把Relay LOG 文件中的内容解析成sql语句，并在自身Slave服务器上按解析SQL语句的位置顺序执行应用这样sql语句，并在relay-log.info中记录当前应用中继日志的文件名和位置点