1、MySQL的复制原理以及流程

基本原理流程，3个线程以及之间的关联；

1. 主：binlog线程——记录下所有改变了数据库数据的语句，放进master上的binlog中；

2. 从：io线程——在使用start slave 之后，负责从master上拉取 binlog 内容，放进 自己的relay log中；

3. 从：sql执行线程——执行relay log中的语句；

2、MySQL中myisam与innodb的区别，至少5点

(1)、问5点不同；

1>.InnoDB支持事物，而MyISAM不支持事物

2>.InnoDB支持行级锁，而MyISAM支持表级锁

3>.InnoDB支持MVCC, 而MyISAM不支持

4>.InnoDB支持外键，而MyISAM不支持

5>.InnoDB不支持全文索引，而MyISAM支持。

(2)、innodb引擎的4大特性

插入缓冲（insert buffer),二次写(double write),自适应哈希索引(ahi),预读(read ahead)

(3)、2者selectcount(\*)哪个更快，为什么

myisam更快，因为myisam内部维护了一个计数器，可以直接调取。

3、MySQL中varchar与char的区别以及varchar(50)中的50代表的涵义

(1)、varchar与char的区别

char是一种固定长度的类型，varchar则是一种可变长度的类型

(2)、varchar(50)中50的涵义

最多存放50个字符，varchar(50)和(200)存储hello所占空间一样，但后者在排序时会消耗更多内存，因为order by col采用fixed\_length计算col长度(memory引擎也一样)

(3)、int（20）中20的涵义

是指显示字符的长度

但要加参数的，最大为255，比如它是记录行数的id,插入10笔资料，它就显示00000000001 ~~~00000000010，当字符的位数超过11,它也只显示11位，如果你没有加那个让它未满11位就前面加0的参数，它不会在前面加0

20表示最大显示宽度为20，但仍占4字节存储，存储范围不变；

(4)、mysql为什么这么设计

对大多数应用没有意义，只是规定一些工具用来显示字符的个数；int(1)和int(20)存储和计算均一样；

4、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)、事物的4种隔离级别

隔离级别

读未提交(RU)

读已提交(RC)

可重复读(RR)

串行

(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”

5、问了MySQL binlog的几种日志录入格式以及区别

(1)、binlog的日志格式的种类和分别

(2)、适用场景；

(3)、结合第一个问题，每一种日志格式在复制中的优劣。

Statement：每一条会修改数据的sql都会记录在binlog中。

优点：不需要记录每一行的变化，减少了binlog日志量，节约了IO，提高性能。(相比row能节约多少性能 与日志量，这个取决于应用的SQL情况，正常同一条记录修改或者插入row格式所产生的日志量还小于Statement产生的日志量，但是考虑到如果带条 件的update操作，以及整表删除，alter表等操作，ROW格式会产生大量日志，因此在考虑是否使用ROW格式日志时应该跟据应用的实际情况，其所 产生的日志量会增加多少，以及带来的IO性能问题。)

缺点：由于记录的只是执行语句，为了这些语句能在slave上正确运行，因此还必须记录每条语句在执行的时候的 一些相关信息，以保证所有语句能在slave得到和在master端执行时候相同 的结果。另外mysql 的复制,像一些特定函数功能，slave可与master上要保持一致会有很多相关问题(如sleep()函数， last\_insert\_id()，以及user-defined functions(udf)会出现问题).

使用以下函数的语句也无法被复制：

\* LOAD\_FILE()

\* UUID()

\* USER()

\* FOUND\_ROWS()

\* SYSDATE() (除非启动时启用了 --sysdate-is-now 选项)

同时在INSERT ...SELECT 会产生比 RBR 更多的行级锁

2.Row:不记录sql语句上下文相关信息，仅保存哪条记录被修改。

优点： binlog中可以不记录执行的sql语句的上下文相关的信息，仅需要记录那一条记录被修改成什么了。所以rowlevel的日志内容会非常清楚的记录下 每一行数据修改的细节。而且不会出现某些特定情况下的存储过程，或function，以及trigger的调用和触发无法被正确复制的问题

缺点:所有的执行的语句当记录到日志中的时候，都将以每行记录的修改来记录，这样可能会产生大量的日志内容,比 如一条update语句，修改多条记录，则binlog中每一条修改都会有记录，这样造成binlog日志量会很大，特别是当执行alter table之类的语句的时候，由于表结构修改，每条记录都发生改变，那么该表每一条记录都会记录到日志中。

3.Mixedlevel: 是以上两种level的混合使用，一般的语句修改使用statment格式保存binlog，如一些函数，statement无法完成主从复制的操作，则 采用row格式保存binlog,MySQL会根据执行的每一条具体的sql语句来区分对待记录的日志形式，也就是在Statement和Row之间选择 一种.新版本的MySQL中队row level模式也被做了优化，并不是所有的修改都会以row level来记录，像遇到表结构变更的时候就会以statement模式来记录。至于update或者delete等修改数据的语句，还是会记录所有行的 变更。

6、问了下MySQL数据库cpu飙升到500%的话他怎么处理？

(1)、没有经验的，可以不问；

(2)、有经验的，问他们的处理思路。

列出所有进程 show processlist 观察所有进程 多秒没有状态变化的(干掉)

查看超时日志或者错误日志 (做了几年开发,一般会是查询以及大批量的插入会导致cpu与i/o上涨,,,,当然不排除网络状态突然断了,,导致一个请求服务器只接受到一半，比如where子句或分页子句没有发送,,当然的一次被坑经历)

7、sql优化