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；

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；

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；

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：中继日志也是二进制日志，用来给slave 库恢复

事务日志：重做日志redo和回滚日志undo

(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、MySQL binlog的几种日志录入格式以及区别

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
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。
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、MySQL数据库cpu飙升到500%的话他怎么处理?

- 1、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)
- 2、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优化各种方法

(1)、explain出来的各种item的意义；

`select_type`

表示查询中每个select子句的类型

`type`

表示MySQL在表中找到所需行的方式，又称“访问类型”

`possible_keys`

指出MySQL能使用哪个索引在表中找到行，查询涉及到的字段上若存在索引，则该索引将被列出，但不一定被查询使用

`key`

显示MySQL在查询中实际使用的索引，若没有使用索引，显示为NULL

`key_len`

表示索引中使用的字节数，可通过该列计算查询中使用的索引的长度

`ref`

表示上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值

`Extra`

包含不适合在其他列中显示但十分重要的额外信息

(2)、profile的意义以及使用场景；

查询到 SQL 会执行多少时间, 并看出 CPU/Memory 使用量, 执行过程中 Systemlock, Table lock 花多少时间等等

8、备份计划，mysqldump以及xtranbackup的实现原理

(1)、备份计划；

这里每个公司都不一样，您别说那种1小时1全备什么的就行

(2)、备份恢复时间；

这里跟机器，尤其是硬盘的速率有关系，以下列举几个仅供参考

20G的2分钟（mysqldump）

80G的30分钟(mysqldump)

111G的30分钟（mysqldump）

288G的3小时（xtra)

3T的4小时（xtra)

逻辑导入时间一般是备份时间的5倍以上

(3)、xtrabackup实现原理

在InnoDB内部会维护一个redo日志文件，我们也可以叫做事务日志文件。事务日志会存储每一个InnoDB表数据的记录修改。当InnoDB启动时，InnoDB会检查数据文件和事务日志，并执行两个步骤：它应用（前滚）已经提交的事务日志到数据文件，并将修改过但没有提交的数据进行回滚操作。

9、mysqldump中备份出来的sql，如果我想sql文件中，一行只有一个insert....value()的话，怎么办？如果备份需要带上master的复制点信息怎么办?

```
--skip-extended-insert

[root@helei-zhuanshu ~]# mysqldump -uroot -p helei --skip-extended-insert

Enter password:

KEY `idx_c1` (`c1`),

KEY `idx_c2` (`c2`)

) ENGINE=InnoDB AUTO_INCREMENT=51 DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;

--

-- Dumping data for table `helei`

--

LOCK TABLES `helei` WRITE;

/*!40000 ALTER TABLE `helei` DISABLE KEYS */;

INSERT INTO `helei` VALUES (1,32,37,38,'2016-10-18 06:19:24','susususususususususususu');

INSERT INTO `helei` VALUES (2,37,46,21,'2016-10-18 06:19:24','susususususu');
```

```
INSERT INTO `helei` VALUES (3,21,5,14,'2016-10-18 06:19:24','susu');
```

10、500台db，在最快时间之内重启

可以使用批量 ssh 工具 pssh 来对需要重启的机器执行重启命令。 也可以使用 salt（前提是客户端有安装 salt）或者 ansible（ ansible 只需要 ssh 免登通了就行）等多线程工具同时操作多台服务器

11、innodb的读写参数优化

(1)、读取参数

`global buffer pool以及 local buffer;`

(2)、写入参数;

```
innodb_flush_log_at_trx_commit

innodb_buffer_pool_size
```

(3)、与IO相关的参数;

```
innodb_write_io_threads = 8

innodb_read_io_threads = 8

innodb_thread_concurrency = 0
```

(4)、缓存参数以及缓存的适用场景。

```
query cache/query_cache_type
```

并不是所有表都适合使用query cache。造成query cache失效的原因主要是相应的table发生了变更

第一个：读操作多的话看看比例，简单来说，如果是用户清单表，或者说是数据比例比较固定，比如说商品列表，是可以打开的，前提是这些库比较集中，数据库中的实务比较小。

第二个：我们“行骗”的时候，比如说我们竞标的时候压测，把query cache打开，还是能收到qps激增的效果，当然前提示前端的连接池什么的都配置一样。大部分情况下如果写入的居多，访问量并不多，那么就不要打开，例如社交网站的，10%的人产生内容，其余的90%都在消费，打开还是效果很好的，但是你如果是qq消息，或者聊天，那就很要命。

第三个：小网站或者没有高并发的无所谓，高并发下，会看到 很多 qcache 锁等待，所以一般高并发下，不建议打开query cache

12、你是如何监控你们的数据库的？你们的慢日志都是怎么查询的？

监控的工具有很多，例如zabbix，lepus，我这里用的是lepus

13、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

14、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

15、你是如何维护数据库的数据字典的?

这个大家维护的方法都不同，我一般是直接在生产库进行注释，利用工具导出成excel方便流通。

16、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

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；

- 1、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
- 2、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

17、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?

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

18、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;
```

19、什么是存储过程？有哪些优缺点?

存储过程是一些预编译的SQL语句。

- 1、更加直白的理解：存储过程可以说是一个记录集，它是由一些T-SQL语句组成的代码块，这些T-SQL语句代码像一个方法一样实现一些功能（对单表或多表的增删改查），然后再给这个代码块取一个名字，在用到这个功能的时候调用他就行了。
- 2、存储过程是一个预编译的代码块，执行效率比较高,一个存储过程替代大量T_SQL语句 ，可以降低网络通信量，提高通信速率,可以一定程度上确保数据安全

20、索引是什么？有什么作用以及优缺点?

- 1、索引是对数据库表中一或多个列的值进行排序的结构，是帮助MySQL高效获取数据的数据结构
- 2、索引就是加快检索表中数据的方法。数据库的索引类似于书籍的索引。在书籍中，索引允许用户不必翻阅完整个书就能迅速地找到所需要的信息。在数据库中，索引也允许数据库程序迅速地找到表中的数据，而不必扫描整个数据库。

MySQL数据库几个基本的索引类型：普通索引、唯一索引、主键索引、全文索引

- 1、索引加快数据库的检索速度
- 2、索引降低了插入、删除、修改等维护任务的速度
- 3、唯一索引可以确保每一行数据的唯一性
- 4、通过使用索引，可以在查询的过程中使用优化隐藏器，提高系统的性能
- 5、索引需要占物理和数据空间

21、什么是事务?

事务（Transaction）是并发控制的基本单位。所谓的事务，它是一个操作序列，这些操作要么都执行，要么都不执行，它是一个不可分割的工作单位。事务是数据库维护数据一致性的单位，在每个事务结束时，都能保持数据一致性。

22、使用索引查询一定能提高查询的性能吗？为什么

通常,通过索引查询数据比全表扫描要快.但是我们也必须注意到它的代价.

- 1、索引需要空间来存储,也需要定期维护, 每当有记录在表中增减或索引列被修改时,索引本身也会被修改. 这意味着每条记录的INSERT,DELETE,UPDATE将为此多付出4,5 次的磁盘I/O. 因为索引需要额外的存储空间和处理,那些不必要的索引反而会使查询反应时间变慢.使用索

引查询不一定能提高查询性能,索引范围查询(INDEX RANGE SCAN)适用于两种情况:

- 2、基于一个范围的检索,一般查询返回结果集小于表中记录数的30%
- 3、基于非唯一性索引的检索

23、简单说一说drop、delete与truncate的区

SQL中的drop、delete、truncate都表示删除，但是三者有一些差别

- 1、delete和truncate只删除表的数据不删除表的结构
- 2、速度,一般来说: drop> truncate >delete
- 3、delete语句是dml,这个操作会放到rollback segement中,事务提交之后才生效;
- 4、如果有相应的trigger,执行的时候将被触发. truncate,drop是ddl, 操作立即生效,原数据不放到rollback segment中,不能回滚. 操作不触发trigger.

24、drop、delete与truncate分别在什么场景之下使用?

- 1、不再需要一张表的时候，用drop
- 2、想删除部分数据行时候，用delete，并且带上where子句
- 3、保留表而删除所有数据的时候用truncate

25、超键、候选键、主键、外键分别是什么?

- 1、超键：在关系中能唯一标识元组的属性集称为关系模式的超键。一个属性可以为作为一个超键，多个属性组合在一起也可以作为一个超键。超键包含候选键和主键。
- 2、候选键：是最小超键，即没有冗余元素的超键。
- 3、主键：数据库表中对储存数据对象予以唯一和完整标识的数据列或属性的组合。一个数据列只能有一个主键，且主键的取值不能缺失，即不能为空值（Null）。
- 4、外键：在一个表中存在的另一个表的主键称此表的外键。

26、什么是视图？以及视图的使用场景有哪些?

- 1、视图是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，试图通常是有一个表或者多个表的行或列的子集。对视图的修改不影响基本表。它使得我们获取数据更容易，相比多表查询。
- 2、只暴露部分字段给访问者，所以就建一个虚表，就是视图。
- 3、查询的数据来源于不同的表，而查询者希望以统一的方式查询，这样也可以建立一个视图，把多个表查询结果联合起来，查询者只需要直

接从视图中获取数据，不必考虑数据来源于不同表所带来的差异

27、说一说三个范式。

- 第一范式（1NF）：数据库表中的字段都是单一属性的，不可再分。这个单一属性由基本类型构成，包括整型、实数、字符型、逻辑型、日期型等。
- 第二范式（2NF）：数据库表中不存在非关键字段对任一候选关键字段的部分函数依赖（部分函数依赖指的是存在组合关键字中的某些字段决定非关键字段的情况），也即所有非关键字段都完全依赖于任意一组候选关键字。
- 第三范式（3NF）：在第二范式的基础上，数据表中如果不存在非关键字段对任一候选关键字段的传递函数依赖则符合第三范式。所谓传递函数依赖，指的是如 果存在"A → B → C"的决定关系，则C传递函数依赖于A。因此，满足第三范式的数据库表应该不存在如下依赖关系： 关键字段 → 非关键字段 x → 非关键字段y

28、数据库的乐观锁和悲观锁是什么?

数据库管理系统（DBMS）中的并发控制的任务是确保在多个事务同时存取数据库中同一数据时不破坏事务的隔离性和统一性以及数据库的统一性。乐观并发控制(乐观锁)和悲观并发控制（悲观锁）是并发控制主要采用的技术手段。

悲观锁：假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作

乐观锁：假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性。