MySQL面试题

1. MySQL工作原理

mysql原理图各个组件说明：

MySQL逻辑架构整体分为三层，最上层为客户端层，并非MySQL所独有，诸如：连接处理、授权认证、安全等功能均在这一层处理。

MySQL大多数核心服务均在中间这一层，包括查询解析、分析、优化、缓存、内置函数(比如：时间、数学、加密等函数)。所有的跨存储引擎的功能也在这一层实现：存储过程、触发器、视图等。

最下层为存储引擎，其负责MySQL中的数据存储和提取。和Linux下的文件系统类似，每种存储引擎都有其优势和劣势。中间的服务层通过API与存储引擎通信，这些API接口屏蔽了不同存储引擎间的差异。

其八个组成部分及工作原理如下:

1. connectors

与其他编程语言中的sql 语句进行交互，如php、java等。

2. Management Serveices & Utilities

系统管理和控制工具

3. Connection Pool (连接池)

管理缓冲用户连接，线程处理等需要缓存的需求

4. SQL Interface (SQL接口)

接受用户的SQL命令，并且返回用户需要查询的结果。比如select from就是调用SQL Interface

5. Parser （解析器）

SQL命令传递到解析器的时候会被解析器验证和解析。

主要功能：

a . 将SQL语句分解成数据结构，并将这个结构传递到后续步骤，后面SQL语句的传递和处理就是基于这个结构的

b. 如果在分解构成中遇到错误，那么就说明这个sql语句是不合理的，语句将不会继续执行下去

6. Optimizer (查询优化器)

SQL语句在查询之前会使用查询优化器对查询进行优化(产生多种执行计划,最终数据库会选择最优化的方案去执行,尽快返会结果) 他使用的是“选取-投影-联接”策略进行查询。

用一个例子就可以理解： select uid,name from user where gender = 1;

这个select 查询先根据where 语句进行选取，而不是先将表全部查询出来以后再进行gender过滤

这个select查询先根据uid和name进行属性投影，而不是将属性全部取出以后再进行过滤

将这两个查询条件联接起来生成最终查询结果.

7. Cache和Buffer (查询缓存)

如果查询缓存有命中的查询结果，查询语句就可以直接去查询缓存中取数据。

这个缓存机制是由一系列小缓存组成的。比如表缓存，记录缓存，key缓存，权限缓存等

8.Engine (存储引擎)

存储引擎是MySql中具体的与文件打交道的子系统。也是Mysql最具有特色的一个地方。

Mysql的存储引擎是插件式的。它根据MySql AB公司提供的文件访问层的一个抽象接口来定制一种文件访问机制（这种访问机制就叫存储引擎）

SQL 语句执行过程

数据库通常不会被直接使用，而是由其他编程语言通过SQL语句调用mysql，由mysql处理并返回执行结果。那么Mysql接受到SQL语句后，又是如何处理

首先程序的请求会通过mysql的connectors与其进行交互，请求到处后，会暂时存放在连接池（connection pool)中并由处理器（Management Serveices & Utilities）管理。当该请求从等待队列进入到处理队列，管理器会将该请求丢给SQL接口（SQL Interface）。SQL接口接收到请求后，它会将请求进行hash处理并与缓存中的结果进行对比，如果完全匹配则通过缓存直接返回处理结果；否则，需要完整的走一趟流程：

(1)由SQL接口丢给后面的解释器（Parser），解释器会判断SQL语句正确与否，若正确则将其转化为数据结构。

(2)解释器处理完，便来到后面的优化器（Optimizer），它会产生多种执行计划,最终数据库会选择最优化的方案去执行,尽快返会结果。

(3)确定最优执行计划后，SQL语句此时便可以交由存储引擎（Engine）处理，存储引擎将会到后端的存储设备中取得相应的数据，并原路返回给程序。

注意点

(1)如何缓存查询数据

存储引擎处理完数据，并将其返回给程序的同时，它还会将一份数据保留在缓存中，以便更快速的处理下一次相同的请求。具体情况是，mysql会将查询的语句、执行结果等进行hash，并保留在cache中，等待下次查询。

(2)buffer与cache的区别

从mysql原理图可以看到，缓存那里实际上有buffer和cache两个，那它们之间的区别：简单的说就是，buffer是写缓存，cache是读缓存。

(3)如何判断缓存中是否已缓存需要的数据

这里可能有一个误区，觉得处理SQL语句的时候，为了判断是否已缓存查询结果，会将整个流程走一遍，取得执行结果后再与需要的进行对比，看看是否命中，并以此说，既然不管缓存中有没有缓存到查询内容，都要整个流程走一遍，那缓存的优势在哪？

其实并不是这样，在第一次查询后，mysql便将查询语句以及查询结果进行hash处理并保留在缓存中，SQL查询到达之后，对其进行同样的hash处理后，将两个hash值进行对照，如果一样，则命中，从缓存中返回查询结果；否则，需要整个流程走一遍。

1. MySQL优化

数据库优化通常从三方面考虑:

1.替换有问题的硬件;

2.对MySQL进程的设置进行优化;

3.对查询进行优化;

1.替换硬件通常是第一考虑,主要原因是数据库会占用大量资源,不过解决方案也仅限于此

2.对数据库服务器进行调优,意味着适当分配内存,并让mysqld了解将会承受何种类型的负载.

3.查询优化,意味着对表应用了适当的索引.还可以记录慢查询.

1 优化你的MySQL查询缓存

2 用EXPLAIN使你的SELECT查询更加清晰：了解MySQL正在进行什么样的查询操作，这可以帮助你发现瓶颈的所在，并显示出查询或表结构在哪里出了问题。

EXPLAIN查询的结果，可以告诉你那些索引正在被引用，表是如何被扫描和排序的等等。

3 尽量避免SELECT \*命令

1. MySQL读写分离

让master（主数据库）来响应事务性操作，让slave（从数据库）来响应select非事务性操作，然后再采用主从复制来把master上的事务性操作同步到slave数据库中。

多台MySQL服务器,分别提供读写服务,均衡流量,通过主从复制保持数据一致性.

由MySQL代理面向客户端,收到SQL写请求时,交给服务器A处理.收到SQL读请求时,交给服务器B处理.具体区分策略由服务设置.

1. MySQL主从同步

MySQL主从复制是数据备份的一种手段,其用途主要有:实时灾备，用于故障切换;读写分离，提供查询服务;备份，避免影响业务.

MySQL的主从复制方案是其自带的同步复制功能,并且它不是从硬盘上给文件直接同步,而是逻辑的binlog日志同步到本地的应用执行的过程。

MySQL的主从复制是一个异步的复制过程（虽然一般情况下感觉是实时的），数据将从一个Mysql数据库（我们称之为Master）复制到另一个Mysql数据库（我们称之为Slave），在Master与Slave之间实现整个主从复制的过程是由三个线程参与完成的。其中有两个线程（SQL线程和IO线程）在Slave端，另一个线程（binlog dump线程）在Master端。

master:（1）binlog dump线程：当主库中有数据更新时，那么主库就会根据按照设置的binlog格式，将此次更新的事件类型写入到主库的binlog文件中，此时主库会创建log dump线程通知slave有数据更新，当I/O线程请求日志内容时，会将此时的binlog名称和当前更新的位置同时传给slave的I/O线程。

slave:（2）I/O线程：该线程会连接到master，向log dump线程请求一份指定binlog文件位置的副本，并将请求回来的binlog存到本地的relay log中，relay log和binlog日志一样也是记录了数据更新的事件，它也是按照递增后缀名的方式，产生多个relay log（ host\_name-relay-bin.000001）文件，slave会使用一个index文件（ host\_name-relay-bin.index）来追踪当前正在使用的relay log文件;（3）SQL线程：该线程检测到relay log有更新后，会读取并在本地做redo操作，将发生在主库的事件在本地重新执行一遍，来保证主从数据同步。此外，如果一个relay log文件中的全部事件都执行完毕，那么SQL线程会自动将该relay log 文件删除掉。

附加:

主动同步延迟问题

mysql的主从复制都是单线程的操作，主库对所有DDL和DML产生binlog，binlog是顺序写，所以效率很高，slave的I/O线程到主库取日志，效率也比较高，但是，slave的SQL线程将主库的DDL和DML操作在slave实施。DML和DDL的IO操作是随即的，不是顺序的，成本高很多，还可能存在slave上的其他查询产生lock争用的情况，由于SQL也是单线程的，所以一个DDL卡住了，需要执行很长一段事件，后续的DDL线程会等待这个DDL执行完毕之后才执行，这就导致了延时。当主库的TPS并发较高时，产生的DDL数量超过slave一个sql线程所能承受的范围，延时就产生了，除此之外，还有可能与slave的大型query语句产生了锁等待导致。

由于主从同步延迟是客观存在的，我们只能从我们自己的架构上进行设计， 尽量让主库的DDL快速执行。下面列出几种常见的解决方案：

1. 业务的持久化层的实现采用分库架构，mysql服务可平行扩展，分散压力;

2.服务的基础架构在业务和mysql之间加入memcache或者Redis的cache层。降低mysql的读压力；

3.使用比主库更好的硬件设备作为slave；

4.sync\_binlog在slave端设置为0；

5.–logs-slave-updates 从服务器从主服务器接收到的更新不记入它的二进制日志;

6.禁用slave的binlog。

1. MHA高可用

MHA是一套优秀的实现MySQL高可用的解决方案.数据库的自动故障切换操作能做到在0~30秒之内,MHA能确保在故障切换过程中保证数据的一致性,以达到真正意义上的高可用.

主库挂了，但是主库的binlog都被全部从库接收，此时会选中应用binlog最全的一台从库作为新的主库，其他从主只需要重新指定一下主库即可(因为此时,所有从库都是一致的，所以只需要重新指定一下从库即可)。

主库挂了，所有的binlog都已经被从库接收了，但是，主库上有几条记录还没有sync到binlog中，所以从库也没有接收到这个event，如果此时做切换，会丢失这个event。此时，如果主库还可以通过ssh访问到，binlog文件可以查看，那么先copy该event到所有的从库上，最后再切换主库。如果使用半同步复制，可以极大的减少此类风险。

主库挂了，从库上有部分从库没有接收到所有的events，选择出最新的slave，从中拷贝其他从所缺少的events。

1. mycat工作原理

Mycat的原理中最重要的一个动词是“拦截”，它拦截了用户发送过来的SQL语句，首先对SQL语句做了一些特定的分析：如分片分析、路由分析、读写分离分析、缓存分析等，然后将此SQL发往后端的真实数据库，并将返回的结果做适当的处理，最终再返回给用户

1. 数据库事物

事务是最小的逻辑工作单元。

事务特性(ACID)

1).原子性:一个事务里面的操作要么不做,要么都做;

2).一致性:事务启动之前和启动之后要保持平衡状态

例如完整性约束a+b=10，一个事务改变了a，那么b也应随之改变。

3).隔离性:在一个会话里面读取不到另一个会话里未提交的数据.

4).永久性:事务一经提交永不回退。

1. MySQL的常用引擎与数据结构

在MySQL数据库中，常用的引擎主要就是2个：Innodb和MyIASM

1).MyISAM，表锁，不支持事务，表损坏率较高，分为MYD数据文件和MYI索引二个文件，读写并发不如InnoDB，适用于insert较多的场景，且支持直接复制文件，用以备份数据。

2).InnoDB，行锁，支持事务，crash后具有recove机制，只有ibd文件，分为数据区和索引区，有较好的读写并发能力，但做count()运算时相当消耗CPU。

所使用的索引的数据结构都是B树。

MyIASM引擎，B树的数据结构中存储的内容实际上是实际数据的地址值。也就是说它的索引和实际数据是分开的，只不过使用索引指向了实际数据。这种索引的模式被称为非聚集索引。

Innodb引擎的索引的数据结构也是B+树，只不过数据结构中存储的都是实际的数据，这种索引有被称为聚集索引。

1. 创建索引规则与注意事项

1).避免索引过多，会影响写性能.

2).给筛选效果低的字段加索引，几乎无效，如性别、状态标志等.

3).每条查询执行时，只会使用一个索引，有需要时应该创建复合索引.

4).复合索引使用时遵守“从左到右”原则，严禁左百分号.

5).不要在索引字段上有运算操作和使用函数，将无法使用索引

1. SQL语句分类

SQL语句主要可以划分为以下几类：

1).DDL(Data Definition Language):数据定义语言，定义对数据库对象(库、表、列、索引)的操作。

包括：CREATE、DROP、ALTER、RENAME、 TRUNCATE等

2).DML(Data Manipulation Language): 数据操作语言，定义对数据库记录的操作。

包括：INSERT、DELETE、UPDATE、SELECT等

3).DCL(Data Control Language): 数据控制语言，定义对数据库、表、字段、用户的访问权限和安全级别。

包括：GRANT、REVOKE等

4).Transaction Control:事务控制

包括：COMMIT、ROLLBACK、SAVEPOINT等

1. Delete、truncate、drop都是删除语句，它们有什么分别？

1).delete 属于DML语句，删除数据，保留表结构，需要commit，可以回滚，如果数据量大，很慢。

2).truncate 属于DDL语句，删除所有数据，保留表结构，自动commit，不可以回滚，一次全部删除所有数据，速度相对较快。

3).Drop属于 DDL语句，删除数据和表结构,不需要commit，删除速度最快。

1. MySQL原生支持的备份方式有哪些，并说出其优缺点？

1).直接拷贝数据文件，必须是MyISAM表，且使用flush tables with read lock;语句，优点是简单方便，缺点是须要锁写，且只能在同版本的MySQL上恢复使用。

2). mysqldump，导出的是SQL语句，所以可以跨版本恢复，但是需要导入数据和重建索引，恢复用时会较长，如果是MyISAM表，同样需要锁表，如果是InnoDB表，可以使用--single-transaction参数避免此。

1. 什么情况下应不建或少建索引

表记录太少

如果一个表只有5条记录，采用索引去访问记录的话，那首先需访问索引表，再通过索引表访问数据表，一般索引表与数据表不在同一个数据块，这种情况下DB至少要往返读取数据块两次。而不用索引的情况下DB会将所有的数据一次读出，处理速度显然会比用索引快。

经常插入、删除、修改的表 对一些经常处理的业务表应在查询允许的情况下尽量减少索引。

数据重复且分布平均的表字段

假如一个表有10万行记录，有一个字段A只有T和F两种值，且每个值的分布概率大约为50%，那么对这种表A字段建索引一般不会提高数据库的查询速度。

1. mysql和oracle的区别？

1). Oracle是大型数据库而Mysql是中小型数据库，市场占有率达 同时Mysql是开源的而Oracle价格非常高。

2). Oracle支持大并发，大访问量

3). 安装所用的空间差别也是很大的，Mysql安装完后才152M而Oracle有3G左右，且使用的时候Oracle占用特别大的内存空间和其他机器性能。

4).Oracle也Mysql操作上的一些区别

①自动增长类型的处理

MYSQL有自动增长的数据类型，插入记录时不用操作此字段，会自动获得数据值。ORACLE没有自动增长的数据类型，需要建立一个自动增长的序列号，插入记录时要把序列号的下一个值赋于此字段。

②单引号的处理

MYSQL里可以用双引号包起字符串，ORACLE里只可以用单引号包起字符串。

③分页的SQL语句的处理

MYSQL处理分页的SQL语句比较简单，用LIMIT 开始位置, 记录个数；ORACLE处理分页的SQL语句就比较繁琐了。每个结果集只有一个ROWNUM字段标明它的位置, 并且只能用ROWNUM<100, 不能用ROWNUM>80

④ 长字符串的处理

ORACLE也有它特殊的地方。INSERT和UPDATE时最大可操作的字符串长度小于等于4000个单字节, 如果要插入更长的字符串, 请考虑字段用CLOB类型，方法借用ORACLE里自带的DBMS\_LOB程序包。插入修改记录前一定要做进行非空和长度判断，不能为空的字段值和超出长度字段值都应该提出警告,返回上次操作。

⑤空字符的处理

MYSQL的非空字段允许为空字符串，ORACLE里定义了非空字段就不容许为空字符串。

⑥字符串的模糊比较

MYSQL里用 字段名 like '%字符串%',ORACLE里也可以用 字段名 like '%字符串%' 但这种方法不能使用索引, 速度不快。

⑦日期字段的处理

MYSQL日期字段分DATE和TIME两种，ORACLE日期字段只有DATE，包含年月日时分秒信息，用当前数据库的系统时间为SYSDATE, 精确到秒

1. 数据库的锁：行锁，表锁；乐观锁，悲观锁

表级锁：开销小，加锁快；不会出现死锁；锁定粒度大，发生锁冲突的概率最高，并发度最低。

行级锁：开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低，并发度也最高

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。不能解决脏读问题

1. MySQL服务器CPU跑满100%的情况分析

1.第一步,万能的重启.当然可能重启了一会儿还是继续高上去

2.检查连接数和慢查询语句.开始分析了.

我们的原则是,重启能解决的,绝对不开client

cpu100%通常情况下就是有慢sql造成的，这里的慢sql包括全表扫描，扫描数据量过大，内存排序，磁盘排序，锁争用等待等...

一般表现现象sql执行状态为：Sending data，Copying to tmp table，Copying to tmp table on disk，Sorting result，locked

通过show processlist查看当前正在执行的sql，当执行完show processlist后出现大量的语句，通常其状态出现Sending data，Copying to tmp table，Copying to tmp table on disk，Sorting result, Using filesort都是sql有性能问题

可以用explain查看sql执行效率,分析索引

解决方案整理

1.Sending data表示：sql正在从表中查询数据，如果查询条件没有适当的索引，则会导致sql执行时间过长；

2.Copying to tmp table on disk：出现这种状态，通常情况下是由于临时结果集太大，超过了数据库规定的临时内存大小，需要拷贝临时结果集到磁盘上，这个时候需要用户对sql进行优化；

3.Sorting result, Using filesort：出现这种状态，表示sql正在执行排序操作，排序操作都会引起较多的cpu消耗，通常的优化方法会添加适当的索引来消除排序，或者缩小排序的结果集；

4.出现sending data的情况：这种一般就是SQL不规范,优化SQL吧.

5.检查网站是不是有被攻击之类的

1. innobackupex 备份过程

PXB 备份过程

innobackupex 在启动后，会先 fork 一个进程，启动 xtrabackup进程，然后就等待 xtrabackup 备份完 ibd 数据文件；

xtrabackup 在备份 InnoDB 相关数据时，是有2种线程的，1种是 redo 拷贝线程，负责拷贝 redo 文件，1种是 ibd 拷贝线程，负责拷贝 ibd 文件；redo 拷贝线程只有一个，在 ibd 拷贝线程之前启动，在 ibd 线程结束后结束。xtrabackup 进程开始执行后，先启动 redo 拷贝线程，从最新的 checkpoint 点开始顺序拷贝 redo 日志；然后再启动 ibd 数据拷贝线程，在xtrabackup 拷贝 ibd 过程中，innobackupex 进程一直处于等待状态（等待文件被创建）。

xtrabackup 拷贝完成idb后，通知innobackupex（通过创建文件），同时自己进入等待（redo 线程仍然继续拷贝）;

innobackupex 收到 xtrabackup 通知后，执行FLUSH TABLES WITH READ LOCK(FTWRL)，取得一致性位点，然后开始备份非 InnoDB 文件（包括 frm、MYD、MYI、CSV、opt、par等）。拷贝非 InnoDB 文件过程中，因为数据库处于全局只读状态，如果在业务的主库备份的话，要特别小心，非 InnoDB 表（主要是MyISAM）比较多的话整库只读时间就会比较长，这个影响一定要评估到。

当 innobackupex 拷贝完所有非 InnoDB 表文件后，通知 xtrabackup（通过删文件） ，同时自己进入等待（等待另一个文件被创建）；

xtrabackup 收到 innobackupex 备份完非 InnoDB 通知后，就停止 redo 拷贝线程，然后通知 innobackupex redo log 拷贝完成（通过创建文件）；

innobackupex 收到 redo 备份完成通知后，就开始解锁，执行 UNLOCK TABLES；

最后 innobackupex 和 xtrabackup 进程各自完成收尾工作，如资源的释放、写备份元数据信息等，innobackupex 等待xtrabackup 子进程结束后退出。

在上面描述的文件拷贝，都是备份进程直接通过操作系统读取数据文件的，只在执行 SQL 命令时和数据库有交互，基本不影响数据库的运行，在备份非 InnoDB 时会有一段时间只读（如果没有MyISAM表的话，只读时间在几秒左右），在备份 InnoDB 数据文件时，对数据库完全没有影响，是真正的热备。

InnoDB 和非 InnoDB 文件的备份都是通过拷贝文件来做的，但是实现的方式不同，前者是以page为粒度做的(xtrabackup)，后者是 cp 或者 tar 命令(innobackupex)，xtrabackup 在读取每个page时会校验 checksum 值，保证数据块是一致的，而innobackupex 在 cp MyISAM 文件时已经做了flush（FTWRL），磁盘上的文件也是完整的，所以最终备份集里的数据文件都是写入完整的。

增量备份

PXB 是支持增量备份的，但是只能对 InnoDB 做增量，InnoDB 每个 page 有个 LSN 号，LSN 是全局递增的，page 被更改时会记录当前的 LSN 号，page中的 LSN 越大，说明当前page越新（最近被更新）。每次备份会记录当前备份到的LSN（xtrabackup\_checkpoints 文件中），增量备份就是只拷贝LSN大于上次备份的page，比上次备份小的跳过，每个 ibd 文件最终备份出来的是增量 delta 文件。

1. BLOB和TEXT之间的区别是什么?

-在BLOB排序和比较中，对BLOB值区分大小写。

-在TEXT文本类型中，不区分大小写进行排序和比较

1、BLOB是什么?

- BLOB表示二进制大对象。

-可以保存可变数量的数据。

根据所能容纳的值的最大长度，有四种BLOB类型:

- TINYBLOB

- BLOB

- MEDIUMBLOB

- LONGBLOB

2、TEXT数据类型是什么?

TEXT是不区分大小写的BLOB。四种文本类型是：

- TINYTEXT

- TEXT

- MEDIUMTEXT

- LONGTEXT

1. MySQL常见监控项。

操作系统层面

CPU

内存

IO

网卡

MySQL层面

MySQL是否运行

连接数

qps

tps

慢查询数

打开表数

当前脏页数

锁情况

主从同步状态（包括Slave\_IO\_Running、Slave\_SQL\_Running、Seconds\_Behind\_Master等

1. 简单说一下 MySQL5.7有哪些新特性。

从MySQL 5.7开始，root用户的密码不再是空，而是随机产生一个密码

支持JSON

MySQL5.7.6支持使用 generated columns（p1875），这一列由其他列计算而得

sys schema

可以知道哪些语句使用了临时表，哪个用户请求了最多的io，哪个线程占用了最多的内存

索引利用率和冗余索引统计冗余索引

支持将临时表设置为innodb存储引擎

支持虚拟列

1. 半同步复制与异步复制的区别，在5.7做了哪些增强？

异步复制：主库执行完commit提交操作后，在主库写入binlog日志后即可成功返回客户端，无需等待binlog日志传送给从库。

MySQL5.7之前的半同步复制：主库在每次事物成功提交时，并不及时反馈给前端应用用户，而是等待其中一个从库也接收到binlog事务并成功写入中继日志后（不等待从库应用这部分日志），主库才返回commit操作成功给客户端。

MySQL5.7半同步复制增强：主库将事务写入binlog，传递到从库并写入从库的relay log，主库等待从库反馈接受到relay log的ack之后，再提交事务并且返回commit ok结果给客户端。

1. 数据库高负载的排查和解决办法。

检查操作系统

查看整体负载，使用命令w或者sar -q 1

判断瓶颈在哪个子系统，使用命令w

判断磁盘IO是否较大，使用命令：sar -d 1

判断具体哪个进程消耗的磁盘IO最多，使用命令：iotop

检查MySQL层

查看当前的MySQL查询语句，使用命令：mysqladmin pr|grep -v Sleep

分析slow log

分析慢查询语句

1. Explain执行计划中要关注哪些要素？

（1）、type:本次查询表联接类型，从这里可以看到本次查询大概的效率

（2）、key:最终选择的索引，如果没有索引的话，本次查询效率通常很差

（3）、key\_len:本次查询用于结果过滤的索引实际长度

（4）、rows:预计需要扫描的记录数，预计需要扫描的记录数越小越好

（5）、extra:额外附加信息，主要确认是否出现 Using filesort、Using temporary 类似情况

1. 重做日志和二进制日志的区别（至少三点）

（1）涉及存储引擎不一样：

binlog记录的是所有存储引擎的操作记录

redo log只记录innodb存储引擎的日志

（2）记录内容不一样：

binlog记录的是关于一个事务的具体操作内容。为逻辑日志

而redo log记录的是每个页更改的物理情况

（3）写的时间不一样：

binlog文件仅在事务提交前进行提交，即只写磁盘一次

而在事务进行过程中，却不断有重做日志条目被写入到重做日志文件中。