二、数据库面试问题（全）

1、关系型数据库

> - MySQL
> - Oracle

- **==说出一些数据库优化方面的经验==**
  - 数据库优化大全

```
1、调整数据结构的设计：
    考虑是否使用ORACLE数据库的分区功能，对于经常访问的数据库表是否需要建立索引等。
2、调整应用程序结构设计：考虑应用程序使用什么样的体系结构，是使用传统的Client/Server两层体系结构，
    还是使用Browser/Web/Database的三层体系结构。不同的应用程序体系结构要求的数据库资源是不同的
3、调整数据库SQL语句。
4、调整服务器内存分配：内存分配是在信息系统运行过程中优化配置的，数据库管理员可以根据数据库运行状况调整数据库系统
    全局区（SGA区）的数据缓冲区、日志缓冲区和共享池的大小，也可调整程序全局区（PGA区）的大小。
    需要注意的是，SGA区不是越大越好，SGA区过大会占用操作系统使用的内存而引起虚拟内存的页面交换，这样反而会降低系统。
5、调整硬盘I/O：数据库管理员可以将组成同一个表空间的数据文件放在不同的硬盘上，做到硬盘之间I/O负载均衡。
6、调整操作系统参数：
   例如：运行在UNIX操作系统上的ORACLE数据库，可以调整UNIX数据缓冲池的大小，每个进程所能使用的内存大小 等参数
```

  - ==如何写出高性能的SQL语句？==

```
1.选择最优效率的表名顺序
2.Select子句中避免使用“*”
3.减少访问数据库的次数
4.整合简单，无关联的数据库访问
5.删除重复记录
6.用truncate语句代替delete
7.用where子句代替having子句
8.减少对表的查询
9.用内部函数和索引提高效率
```

- explain出来的各种item的意义

```
1、select_type：表示查询中每个select子句的类型
2、type：表示MySQL在表中找到所需行的方式，又称“访问类型”
3、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
4、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
5、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
6、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
7、Extra：包含不适合在其他列中显示但十分重要的额外信息

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
```

- **SQL语言共分为四大类**：
  - 数据查询语言DQL
  - 数据操纵语言DML
  - 数据定义语言DDL
  - 数据控制语言DCL

2、数据库存储引擎

> 存储引擎说白了就是如何存储数据、如何为存储的数据建立索引和如何更新、查询数据等技术的实现方法。

- **ISAM是什么?**
  - ISAM简称为索引顺序访问方法。它是由IBM开发的，用于在磁带等辅助存储系统上存储和检索数据。
- ==**MyISAM**==：这种引擎是mysql最早提供的。分为静态MyISAM、动态MyISAM和压缩MyISAM三种：
  - ==静态MyISAM==：
    - 如果数据表中的各数据列的长度都是预先固定好的，服务器将自动选择这种表类型。
    - 因为数据表中每一条记录所占用的空间都是一样的，所以这种表存取和更新的效率非常高
    - 当数据受损时，恢复工作也比较容易做。
  - ==动态MyISAM==：
    - 如果数据表中出现varchar、text或BLOB字段时，服务器将自动选择这种表类型。
    - ==相对于静态MyISAM，这种表存储空间比较小==，++但由于每条记录的长度不一，所以多次修改数据后，数据表中的数据就可能离散的存储在内存中，进而导致执行效率下降++。
    - 同时，内存中也可能会出现很多碎片。因此，这种类型的表要经常用optimize table 命令或优化工具来进行碎片整理。
  - ==压缩MyISAM==：
    - 以上两种的表都可以用myisamchk工具压缩。这种类型的表进一步减小了占用的存储，但是这种表压缩之后不能再被修改。
    - 另外，因为是压缩数据，所以这种表在读取的时候要先时行解压缩。

        - ==但是，不管是何种MyISAM表，目前它都不支持事务，行级锁和外键约束的功能。==
- **MyISAM Merge引擎**：
    - MyISAM类型的一种变种。
    - 合并表是将几个相同的MyISAM表合并为一个虚表。
    - 常应用于日志和数据仓库。
- **MyISAM表格将在哪里存储，并且还提供其存储格式?** (每个MyISAM表格以三种格式存储在磁盘上)：
    - •“.frm”文件存储表定义
    - •数据文件具有“.MYD”（MYData）扩展名
    - 索引文件具有“.MYI”（MYIndex）扩展名
- **MyISAM Static和MyISAM Dynamic有什么区别?**
    - 在MyISAM Static上的所有字段有固定宽度。动态MyISAM表将具有像TEXT，BLOB等字段，以适应不同长度的数据类型。
    - MyISAM Static在受损情况下更容易恢复。
- **InnoDB**：

    > - InnoDB: 表类型可以看作是对MyISAM的进一步更新产品，它提供了事务、行级锁机制和外键约束的功能。
    > - 关于后面的存储引擎

- **InnoDB是什么?**
    - lnnoDB是一个由Oracle公司开发的Innobase Oy事务安全存储引擎。
- 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的;
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
```

- 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

- 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。

```
1、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; 2、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；
3、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”
```

- **memory(heap)**：
  - 这种类型的数据表只存在于内存中。
  - 它使用散列索引，所以数据的存取速度非常快。
  - 因为是存在于内存中，所以这种类型常应用于临时表中。
- **archive**：
  - 这种类型只支持select 和 insert语句，而且不支持索引。
    - Desc[ribe] tablename; //查看数据表的结构
    - show engines; 命令可以显示当前数据库支持的存储引擎情况

3、数据库存储过程

- 存储过程简介
- **数据库的左连接、有连接、内连接、外连接以及如何使用?**
  - 内连接：inner join ， 两表存在主外键关系的时候常用
  - 外连接：
    - ==左外连接：left outer join== ,左外连接的结果集包括left outer join 子句中指定的左表 的所有行，若左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有的选择列均为空值
    - ==右外连接：right outer join==,右外连接的结果集包括right outer join 子句中指定的右表的所有行，若右表的某行在左表中没有匹配行，则在相关联的结果集行中左表的所有的选择列均为空值。
    - ==完全外连接：full outer join==
  - 交叉连接：cross join： **笛卡尔积**
  - 等值连接：适用于主外键关系的表：表1 ， 表2 where 表1 . 字段 = 表2 . 字段
  - ==两表联查的sql语句==

```
   内连接：select 字段1，字段2 ... from 表1  join  表2  on  表1.字段名 = 表2.字段名;
   左外连接：显示左边表的所有信息 和右边表满足条件的数据（在left join 左边的为左表 右边的为右表）
        select  *  from 表1   left outer join   表2  on  表1.字段名 = 表2.字段名;
   右外连接：右表所有和左表满足条件的数据
        select  *  from 表1    right outer join   表2  on  表1.字段名 = 表2.字段名;
   全连接：select  *  from 表1    full outer join   表2  on  表1.字段名 = 表2.字段名;
   等值连接：select  *  from 表1 , 表2   where    表1.字段名 = 表2.字段名;
   交叉连接：select  *  from 表1    cross  join   表2  on  表1.字段名 = 表2.字段名;
```

4、数据库常见问题

- **数据库三范式是什么?**（消除歧义、消除冗余、消除依赖）
  - 1NF:字段不可分;
  - 2NF:有主键，非主键字段依赖主键;
  - 3NF:非主键字段不能相互依赖;
- **Class.forName的作用?为什么要用?**
  - 按参数中指定的字符串形式的类名去搜索并加载相应的类，如果该类字节码已经被加载过，则返回代表该字节码的Class实例对象，否则，按类加载器的委托机制去搜索和加载该类，如果所有的类加载器都无法加载到该类，则抛出ClassNotFoundException。加载完这个Class字节码后，接着就可以==使用Class字节码的newInstance==方法去创建该类的实例对象了。
  - 有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时

才能确定，这时候就需要使用Class.forName去动态加载该类，这个类名通常是在配置文件中配置的，例如，spring的ioc中每次依赖注入的具体类就是这样配置的，jdbc的驱动类名通常也是通过配置文件来配置的，以便在产品交付使用后不用修改源程序就可以更换驱动类名。

- **说出数据连接池的工作机制是什么?**
  - J2EE服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。
  - 客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。
  - 当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。
  - 实现方式，返回的Connection是原始Connection的代理，代理Connection的close方法不是真正关连接，而是把它代理的Connection对象还回到连接池中。
- **为什么要用 ORM? 和 JDBC 有何不一样?**
  - orm是一种思想，就是把object转变成数据库中的记录，或者把数据库中的记录转变成objecdt，我们可以用jdbc来实现这种思想，其实，如果我们的项目是严格按照oop方式编写的话，我们的jdbc程序不管是有意还是无意，就已经在实现orm的工作了。
  - 现在有许多orm工具，它们底层调用jdbc来实现了orm工作，我们直接使用这些工具，就省去了直接使用jdbc的繁琐细节，提高了开发效率，现在用的较多的orm工具是hibernate。也听说一些其他orm工具，如toplink,ojb等。

- **Mysql中有哪些不同的表格?** (共有5种类型的表格):
  - MyISAM
  - Heap
  - Merge
  - INNODB
  - ISAM
  - MyISAM是Mysql的默认存储引擎。
- **什么样的对象可以使用CREATE语句创建?**
  - DATABASE
  - EVENT
  - FUNCTION
  - INDEX
  - PROCEDURE
  - TABLE
  - TRIGGER
  - USER
  - VIEW
- **Mysql表中允许有多少个TRIGGERS?** （在Mysql表中允许有六个触发器） -BEFORE INSERT
  - AFTER INSERT
  - BEFORE UPDATE
  - AFTER UPDATE
  - BEFORE DELETE and
  - AFTER DELETE
- **如何显示前50行?**

    - 在Mysql中，使用以下代码查询显示前50行：
    - SELECT*FROM LIMIT 0,50;
- **可以使用多少列创建索引?**
    - ==任何标准表最多可以创建16个索引列==。
- 关于heap表

```
- **Heap表是什么？**
  - HEAP表存在于内存中，用于临时高速存储。
  - BLOB或TEXT字段是不允许的
  - 只能使用比较运算符=，<，>，=>，= <
  - HEAP表不支持AUTO_INCREMENT
  - 索引不可为NULL
- **如何控制HEAP表的最大尺寸？**
  - Heal表的大小可通过称为max_heap_table_size的Mysql配置变量来控制。
```

- 列设置为AUTO INCREMENT时，如果在表中达到最大值，会发生什么情况?

> ++它会停止递增，任何进一步的插入都将产生错误，因为密钥已被使用++。

- 怎样才能找出最后一次插入时分配了哪个自动增量?

> LAST_INSERT_ID将返回由Auto_increment分配的最后一个值，并且不需要指定表名称。

- **列对比运算符是什么?**

```
在SELECT语句的列比较中使用=，<>，<=，<，>=，>，<<，>>，<=>，AND，OR或LIKE运算
```

- **我们如何得到受查询影响的行数?**

> SELECT COUNT(user_id) FROM users;

- **Mysql查询是否区分大小写?**

> 不区分

- **LIKE和REGEXP操作有什么区别?**

> LIKE和REGEXP运算符用于表示^和%。

- **BLOB和TEXT有什么区别?**

- BLOB是一个二进制对象，可以++容纳可变数量的数据++,它们只能在所能容纳价值的最大长度上有所不同。有四种类型的BLOB
  - TINYBLOB
  - BLOB
  - MEDIUMBLOB
  - LONGBLOB
- TEXT是一个不区分大小写的BLOB==,它们对应于四种BLOB类型，并==具有相同的最大长度和存储要求。四种TEXT类型
  - TINYTEXT
  - TEXT
  - MEDIUMTEXT和
  - LONGTEXT
- BLOB和TEXT类型之间的唯一区别在于对BLOB值进行排序和比较时区分大小写，对TEXT值不区分大小写。

- **什么是非标准字符串类型?**

  - TINYTEXT
  - TEXT
  - MEDIUMTEXT
  - LONGTEXT

- MySQL数据库cpu飙升到500%的话他怎么处理?

  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)

- **MYSQL支持事务吗?**
  - ==一般情况是支持事务的==，即MYSQL存储引擎使用InnoDB Tables 或 BDB tables时，
  - 在缺省模式下，MYSQL是autocommit模式的，所有的数据库更新操作都会即时提交，所以在缺省情况下，mysql是不支持事务的。
  - 使用SET AUTOCOMMIT=0就可以使MYSQL允许在非autocommit模式，在非autocommit模式下，你必须使用COMMIT来提交你的更改，或者用ROLLBACK来回滚你的更改。
- **MYSQL数据表在什么情况下容易损坏?**
  - 服务器突然断电导致数据文件损坏。
  - 强制关机，没有先关闭mysql 服务等。
- **mysql有关权限的表都有哪几个?**
  - Mysql服务器通过权限表来控制用户对数据库的访问，权限表存放在mysql数据库里，由mysql_install_db脚本初始化。
  - 这些权限表分别user，db，table_priv，columns_priv和host。
- **Mysql中有哪几种锁?**
  - MyISAM支持表锁，InnoDB支持表锁和行锁，默认为行锁
  - 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，并发量最

低
    - 行级锁：开销大，加锁慢，会出现死锁。==锁力度小，发生锁冲突的概率小，并发度最高==
- **mysql_fetch_array和mysql_fetch_object的区别是什么?**
    - mysql_fetch_array（） - 将结果行作为关联数组或来自数据库的常规数组返回。
    - mysql_fetch_object - 从数据库返回结果行作为对象。
- **我们如何在mysql中运行批处理模式?**
    - mysql;
    - mysql mysql.out
- **添加索引：alter table tableName add 索引（索引字段）**

```
-  主键：primary key
-  唯一：unique
-  全局：fulltext
-  普通：index
-  多列：  index index_name
-  页级:引擎 BDB。次锁定相邻的一组记录。
-  表级:引擎 MyISAM ，  理解为锁住整个表，可以同时读，写不行。
-  行级:引擎 INNODB，单独的一行记录加锁，对指定的记录进行加锁，这样其它进程还是可以对同一个表中的其它记录进行操作。
-  表级锁速度快，但冲突多，行级冲突少，但速度慢。
```

- **Mysql服务器默认端口是什么**

> Mysql服务器的默认端口是3306。

- **区分char_length和length?**

> CHAR_LENGTH是字符数，而LENGTH是字节数。Latin字符的这两个数据是相同的，但是对于Unicode和其他编码，它们是不同的。

- **在Mysql中ENUM的用法是什么**

> ENUM是一个字符串对象，用于指定一组预定义的值，并可在创建表时使用。Create table size(name ENUM('Smail,'Medium','Large');

- **如何定义regexp?**

> REGEXP是模式匹配，其中匹配模式在搜索值的任何位置。

- **CHAR和VARCHAR的区别?**

```
- CHAR和VARCHAR类型在存储和检索方面有所不同
- CHAR列长度固定为创建表时声明的长度，长度值范围是1到255,当CHAR值被存储时，它们被用空格填充到特定长度，检索CHAR值时需删除尾随空格。
```

- **列的字符串类型可以是什么?**
    - SET
    - BLOB
    - ENUM
    - CHAR

    - TEXT
    - VARCHAR
- **如何获取当前的Mysql版本?**
    - SELECT VERSION();用于获取当前Mysql的版本。
- **Mysql中使用什么存储引擎?**
    - 存储引擎称为表类型，数据使用各种技术存储在文件中。
    - Storage mechanism
    - Locking levels
    - Indexing
    - Capabilities and functions.
- **主键和候选键有什么区别?**
    - 表格的每一行都由主键唯一标识,一个表只有一个主键。
    - 主键也是候选键。按照惯例，候选键可以被指定为主键，并且可以用于任何外键引用。
- **myisamchk是用来做什么的?**
    - 它用来压缩MyISAM表，这减少了磁盘或内存使用。
- **请说明数据库主键、外键的作用，以及建立索引的好处与坏处?**
    - 主键的作用：
        - 唯一的标识一行
        - 作为一个可以被外键有效引用的对象
    - 外键的作用：
        - 数据库通过外键保证数据的一致性和完整性
        - 增加E-R图的可读性
    - 建立索引的好处：
        - 提高查询速度，加速表与表之间的连接，减低查询中分组与排序的时间
    - 建立索引的坏处：
        - 存储索引占用磁盘时间，执行数据修改操作（insert ， update ， delete）产生索引维护
- **一张表里面有ID自增主键，当insert了17条记录之后，删除了第15,16,17条记录，再把mysql重启，再insert一条记录，这条记录的ID是18还是15 ？ 如果是删除的是10,11,12三条记录呢?**
    - ==一般情况下，我们创建的表的类型是InnoDB==，如果新增一条记录（不重启mysql的情况下），这条记录的id是18；但是如果重启（文中提到的）MySQL的话，这条记录的ID是15。因为InnoDB表只把自增主键的最大ID记录到内存中，所以重启数据库或者对表OPTIMIZE操作，都会使最大ID丢失。
    - 但是，如果我们使用表的类型是MyISAM，那么这条记录的ID就是18。因为MyISAM表会把自增主键的最大ID记录到数据文件里面，重启MYSQL后，自增主键的最大ID也不会丢失。
    - 如果在这17条记录里面删除的是中间10,11,12三条记录，无论是否重启MySQL以及何种存储引擎，insert一条记录后，ID都是18。因为内存或者数据库文件存储都是自增主键最大ID
- **在数据库中如何保证字段唯一确定一个值?**
    - 有两种方法：
        - 将字段设置为主码（一个表只能定义一个主码）：
        - 设置字段为唯一值约束(一个表可以定义多个唯一值约束)：
- **主键生成策略有哪些？在数据库转移时会产生哪些常见的问题?**
    - 主键生成方式主要有以下几种：

```
  – 自增长：当我们想向表中插入一行数据时，不必考虑主键字段的取值，记录插入后，数据库系统会自动为其分配一个值，这个值是成自然数字增长的，确保不出现主键重复。
    – 在MySql中，把主键设为auto_increment类型，数据库会自动为其赋值。
     ep:create table user{id int auto_increment primary key not null,name varchar(15)}。
    – 使用Navicat工具，要设置主键为int型时，设计表结构里面才有Auto increment。
    – 手动增长：手动设置字段作为主键，即主键的值需要自己维护。通常情况下需要单独的表存储当前主键值。
    – GUID值：GUID是全局唯一标示符的简称，也叫UUID。GUID是一种算法生成的二进制长度为128位的数字标识符。
 – COMB类型：COMB类型可以说是GUID的改进类型，它组合GUID和系统时间，使其在索引和检索上有更有的性能。
```

- **请简述数据库事务？以及事务的特性?**
  - 数据库事务(MySQL&Oracle等)
- **注册Jdbc驱动程序的三种方式**
  - 第一种：通过Class.forName("com.mysql.jdbc.Driver");//加载数据库驱动
  - 第二种：通过系统属性 System.setProperty("jdbc.driver","com.mysql.jdbc.Driver");//系统属性指定数据库驱动。
  - 第三种：看起来比较直观的一种方式，注册相应的db的jdbc驱动，在编译时需要导入对应的lib。
- **如何将数据库（实例）备份和还原**
  - 数据备份：
    - BACKUP DATABASE 要备份的数据库名 TO DISK ='路径' with name = “备份的数据库起的名称”
  - 数据还原：
    - RESTORE DATABASE 要还原的数据库名 FROM DISK='路径' with file = 次数（第几次备份就是几）
  - 对于MySql数据库：
    - 数据备份

```
    1、备份一个数据库
        mysqldump基本语法：mysqldump -u  那个用户  -p 数据库名称 table1 table2 ... >  BackupName.sql
        其中：table1和table2参数表示需要备份的表的名称，为空则整个数据库备份；
         BackupName.sql参数表设计备份文件的名称，文件名前面可以加上一个绝对路径，通常将数据库被分成一个后缀名为sql的文件；
         例如：mysqldump -u root -p test person > D:\backup.sql
    2、备份多个数据库
        mysqldump基本语法：mysqldump -u 那个用户 -p  --databases  数据库名 ,  数据库名 > Backup.sql
         例如：mysqldump -u root -p --databases test mysql > D:\backup.sql
    3、备份所有数据库
         mysqldump命令备份所有数据库的语法如下： mysqldump -u username -p -all-databases > BackupName.sql
         例如:mysqldump -u -root -p -all-databases > D:\all.sql
```

- 数据还原

```
    使用mysqldump命令备份的数据库的语法如下：
    mysql -u root -p [dbname] < backup.sq
    例如：mysql -u root -p < C:\backup.sql
```

- 备份计划

```
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）
3、逻辑导入时间一般是备份时间的5倍以上
4、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。
```

- 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');
```

- MySQL的复制原理以及流程
- MySQL中myisam与innodb的区别，至少5点
- MySQL中varchar与char的区别以及varchar(50)中的50代表的涵义
- 500台db，在最快时间之内重启

> puppet，dsh

- 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

- 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

- 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。

- 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，请问你是选择拆成子表，还是继续放一起？这样选择的理由

```
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
```

- 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;
```

- **常用三个关系型数据库的区别**
  - 这三个数据库不是同一个公司的产品
  - 其所对应的使用对象也不一样，oracle是主流的大型数据库，大多数电信项目都是使用的oracle， 而sqlserver与mysql主要是个人以及小型公司使用的的数据库，但是sqlserver需要收费，mysql不用；
  - 如果按功能上来说，oracle最为强大，oracle支持递归查询，二后两者不支持；
  - 三个数据库中，只有sqlserver有完整的图形化操作界面，而oracle与mysql都要借助于其他的第三方数据库图形操作界面，比如oracle用的大多都是plsql；
- **三个数据库的优缺点**
  - SQLserver
    - 优点

```
  1. 易用性、适合分布式组织的可伸缩性、用于决策支持的数据仓库功能、与许多其他服务器软件紧密关联的集成性、良好的性价比等；
2. 为数据管理与分析带来了灵活性，允许单位在快速变化的环境中从容响应，从而获得竞争优势。
3. 从数据管理和分析角度看，将原始数据转化为商业智能和充分利用Web带来的机会非常重要。
4. 作为一个完备的数据库和数据分析包，SQLServer为快速开发新一代企业级商业应用程序、为企业赢得核心竞争优势打开了胜利之门。
5. 作为重要的基准测试可伸缩性和速度奖的记录保持者，SQLServer是一个具备完全Web支持的数据库产品，
提供了对可扩展标记语言(XML)的核心支持以及在Internet上和防火墙外进行查询的能力；
```

    - 缺点

```
1. 开放性 ： SQL Server只能windows上运行,没有丝毫开放性操作系统系统稳定对数据库十分重要,Windows9X系列产品偏重于桌面应用NTserver只适合小型企业而且windows平台可靠性安全性和伸缩性非常有限，不象unix样久经考验尤其处理大数据库；
2. 伸缩性并行性 ： SQL server 并行实施和共存模型并成熟难处理日益增多用户数和数据卷伸缩性有限；
3. 安全性：没有获得任何安全证书。
4. 性能 ： SQL Server 多用户时性能佳 ；
5. 客户端支持及应用模式： 客户端支持及应用模式。只支持C/S模式，SQL Server C/S结构只支持windows客户用ADO、DAO、OLEDB、ODBC连接；
6. 使用风险：SQL server 完全重写代码经历了长期测试断延迟许多功能需要时间来证明并十分兼容；
```

  - Oracle

- 关于Oracle的面试题
- 优点

1．开放性：Oracle能所有主流平台上运行（包括windows）完全支持所有工业标准采用完全开放策略使客户选择适合解决方案对开发商全力支持；
2．可伸缩性,并行性：Oracle并行服务器通过使组结点共享同簇工作来扩展windownt能力提供高用性和高伸缩性簇解决方案windowsNT能满足需要用户把数据库移UNIXOracle并行服务器对各种UNIX平台集群机制都有着相当高集成度；
3．安全性：获得最高认证级别的ISO标准认证。
4．性能：Oracle 性能高保持开放平台下TPC-D和TPC-C世界记录；
5．客户端支持及应用模式：Oracle多层次网络计算支持多种工业标准用ODBC、JDBC、OCI等网络客户连接
6．使用风险：Oracle 长时间开发经验完全向下兼容得广泛应用地风险低

- 缺点

1．对硬件的要求很高；
2．价格比较昂贵；
3．管理维护麻烦一些；
4．操作比较复杂，需要技术含量较高；

- MySQL
  - 优点

1．体积小、速度快、总体拥有成本低，开源；
2．支持多种操作系统；
3．是开源数据库，提供的接口支持多种语言连接操作
4．MySql的核心程序采用完全的多线程编程。线程是轻量级的进程，它可以灵活地为用户提供服务，而不过多的系统资源。用多线程和C语言实现的MySql能很容易充分利用CPU；
5．MySql有一个非常灵活而且安全的权限和口令系统。当客户与MySql服务器连接时，他们之间所有的口令传送被加密，而且MySql支持主机认证；
6．支持ODBC for Windows， 支持所有的ODBC 2.5函数和其他许多函数，可以用Access连接MySql服务器， 使得应用被扩展；
7．支持大型的数据库，可以方便地支持上千万条记录的数据库。
8．作为一个开放源代码的数据库，可以针对不同的应用进行相应的修改。
9．拥有一个非常快速而且稳定的基于线程的内存分配系统，可以持续使用面不必担心其稳定性
10．MySQL同时提供高度多样性，能够提供很多不同的使用者介面，包括命令行客户端操作，网页浏览器，以及各式各样的程序语言介面，
例如C+，Perl，Java，PHP，以及Python，你可以使用事先包装好的客户端，或者干脆自己写一个合适的应用程序
MySQL可用于Unix，Windows，以及OS/2等平台，因此它可以用在个人电脑或者是服务器上；

  - 缺点

```
1．不支持热备份
2．MySQL最大的缺点是其安全系统，主要是复杂而非标准，另外只有到调用mysqladmin来重读用户权限时才发生改变；
3．没有一种存储过程(Stored Procedure)语言，这是对习惯于企业级数据库的程序员的最大限制；
4．MySQL的价格随平台和安装方式变化。Linux的MySQL如果由用户自己或系统管理员而不是第三方安装则是免费的，第三方案则必须付许可费。Unix或Linux 自行安装 免费、Unix或Linux 第三方安装 收费；
```

5、非关系型数据库

- Redis
- Redis导图
- MongoDB

Redis

- **Redis是什么？两句话可以做下概括**
  - 是一个完全开源免费的key-value内存数据库
  - 通常被认为是一个数据结构服务器，主要是因为其有着丰富的数据结构 strings、map、list、sets、 sorted sets
  - Redis使用最佳方式是全部数据in-memory。
  - Redis更多场景是作为Memcached的替代者来使用。
  - 当需要除key/value之外的更多数据类型支持时，使用Redis更合适。
  - 当存储的数据不能被剔除时，使用Redis更合适。
- **Redis的数据类型有哪些?**
  - Redis支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及zset(sorted set：有序集合)。
- **redis（管道，哈希）**
  - Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
  - Redis支持数据的备份，即master-slave模式的数据备份。
  - Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。
- **实现原理或机制**

> Redis是一个key-value存储系统。和Memcached类似，==但是解决了断电后数据完全丢失的情况==，而且她支持更多无化的value类型，除了和string外，还支持lists（链表）、sets（集合）和zsets（有序集合）几种数据类型。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。

- **Redis是一种基于客户端-服务端模型以及请求/响应协议的TCP服务。** 这意味着通常情况下一个请求会遵循以下步骤：
  - 客户端向服务端发送一个查询请求，并监听Socket返回，通常是以阻塞模式，等待服务端响应。
  - 服务端处理命令，并将结果返回给客户端。

- **Redis 管道技术**
    - 在服务端未响应时，客户端可以继续向服务端发送请求，并最终一次性读取所有服务端的响应。
    - 管道技术最显著的优势是提高了 redis 服务的性能。
    - 分区是分割数据到多个Redis实例的处理过程，因此每个实例只保存key的一个子集。
    - 通过利用多台计算机内存的和值，允许我们构造更大的数据库。
    - 通过多核和多台计算机，允许我们扩展计算能力；通过多台计算机和网络适配器，允许我们扩展网络带宽。
- **Redis有什么用?**（只有了解了它有哪些特性，我们在用的时候才能扬长避短）
    - 速度快：使用标准C写，所有数据都在内存中完成，读写速度分别达到10万/20万
    - 持久化：对数据的更新采用Copy-on-write技术，可以异步地保存到磁盘上，主要有两种策略，一是根据时间，更新次数的快照（save 300 10 ）二是基于语句追加方式(Append-only file，aof)
    - 自动操作：对不同数据类型的操作都是自动的，很安全
    - 快速的主--从复制：官方提供了一个数据，Slave在21秒即完成了对Amazon网站10G key set的复制。
    - Sharding技术： 很容易将数据分布到多个Redis实例中，数据库的扩展是个永恒的话题，==在关系型数据库中，主要是以添加硬件、以分区为主要技术形式的纵向扩展解决了很多的应用场景==，但随着web2.0、移动互联网、云计算等应用的兴起，这种扩展模式已经不太适合了，所以近年来，像采用主从配置、数据库复制形式的，Sharding这种技术把负载分布到多个特理节点上去的横向扩展方式用处越来越多。
- **redis的一些特性在分区方面表现的不是很好**：

> 涉及多个key的操作通常是不被支持的。
>
> - 举例来说，当两个set映射到不同的redis实例上时，你就不能对这两个set执行交集操作。
> - 涉及多个key的redis事务不能使用。
> - 当使用分区时，数据处理较为复杂，比如你需要处理多个rdb/aof文件，并且从多个实例和主机备份持久化文件。
> - 增加或删除容量也比较复杂。++redis集群大多数支持在运行时增加、删除节点的透明数据平衡的能力，但是类似于客户端分区、代理等其他系统则不支持这项特性++。==然而，一种叫做presharding的技术对此是有帮助的。==

- Redis 有两种类型分区
    - ==最简单的分区方式是按范围分区==，就是映射一定范围的对象到特定的Redis实例。
        - 比如，ID从0到10000的用户会保存到实例R0，ID从10001到20000的用户会保存到R1，以此类推。
        - 这种方式是可行的，并且在实际中使用，不足就是要有一个区间范围到实例的映射表。这个表要被管理，同时还需要各 种对象的映射表，通常对Redis来说并非是好的方法。
    - ==哈希分区==：另外一种分区方法是hash分区。这对任何key都适用，也无需是object_name:这种形式，像下面描述的一样简单：
        - 用一个hash函数将key转换为一个数字，比如使用crc32 hash函数。对key foobar执行crc32(foobar)会输出类似93024922的整数。
        - 对这个整数取模，将其转化为0-3之间的数字，就可以将这个整数映射到4个Redis实例中的一个了。93024922 % 4 = 2，就是说key foobar应该被存到R2实例中。

        - 注意：取模操作是取除的余数，通常在多种编程语言中用%操作符实现。
    - 实际上，上面的集群模式还存在两个问题：
        - 扩容问题：

> 因为使用了一致性哈稀进行分片，那么不同的key分布到不同的Redis-Server上，
> 当我们需要扩容时，需要增加机器到分片列表中，这时候会使得同样的key算出来落到跟原来不同的机器上，这样如果要取某一个值，会出现取不到的情况，
> 对于这种情况，Redis的作者提出了一种名为Pre-Sharding的方式：
> ----------
> Pre-Sharding方法是将每一个台物理机上，运行多个不同断口的Redis实例，
> 假如有三个物理机，每个物理机运行三个Redis实际，那么我们的分片列表中实际有9个Redis实例，当我们需要扩容时，增加一台物理机，步骤如下：
> ----------
> A.在新的物理机上运行Redis-Server；
> B.该Redis-Server从属于(slaveof)分片列表中的某一Redis-Server（假设叫RedisA）；
> C.等主从复制(Replication)完成后，将客户端分片列表中RedisA的IP和端口改为新物理机上Redis-Server的IP和端口；
> D.停止RedisA。
> 这样相当于将某一Redis-Server转移到了一台新机器上。Prd-Sharding实际上是一种在线扩容的办法，但还是很依赖Redis本身的复制功能的，
> 如果主库快照数据文件过大，这个复制的过程也会很久，同时会给主库带来压力。所以做这个拆分的过程最好选择为业务访问低峰时段进行。

        - 单点故障问题：

> 还是用到Redis主从复制的功能，两台物理主机上分别都运行有Redis-Server，其中一个Redis-Server是另一个的从库，
> 采用双机热备技术，客户端通过虚拟IP访问主库的物理IP，当主库宕机时，切换到从库的物理IP。
> 只是事后修复主库时，应该将之前的从库改为主库（使用命令slaveof no one），主库变为其从库（使命令slaveof IP PORT），这样才能保证修复期间新增数据的一致性。

- **这里对Redis数据库做下小结**
    - 提高了DB的可扩展性，只需要将新加的数据放到新加的服务器上就可以了
    - 提高了DB的可用性，只影响到需要访问的shard服务器上的数据的用户
    - 提高了DB的可维护性，对系统的升级和配置可以按shard一个个来搞，对服务产生的影响较小
    - 小的数据库存的查询压力小，查询更快，性能更好

mongodb

- **Mongo DB：非关系型数据库(NoSql)**
    - Mongo DB很好的实现了面向对象的思想(OO思想),在MongoDB中每一条记录都是一个Document对象。

    - Mongo DB最大的优势在于所有的数据持久操作都无需开发人员手动编写SQL语句,直接调用方法就可以轻松的实现CRUD操作。
- **特点:**
    - 高性能、易部署、易使用，存储数据非常方便。
    - 主要功能特性有：
        - 面向集合存储，易存储对象类型的数据。
        - 模式自由。
        - 支持动态查询。
        - 支持完全索引，包含内部对象。
        - 支持查询。
        - 支持复制和故障恢复。
        - 使用高效的二进制数据存储，包括大型对象（如视频等）。
        - 自动处理碎片，以支持云计算层次的扩展性
        - 支持Python，PHP，Ruby，Java，C，C#，Javascript，Perl及C++语言的驱动程序，社区中也提供了对Erlang及.NET等平台的驱动程序。
        - 文件存储格式为BSON（一种JSON的扩展）。
        - 可通过网络访问。
    - 功能:
        - 面向集合的存储：适合存储对象及JSON形式的数据。
        - 动态查询：Mongo支持丰富的查询表达式。查询指令使用JSON形式的标记，可轻易查询文档中内嵌的对象及数组。
        - 完整的索引支持：包括文档内嵌对象及数组。Mongo的查询优化器会分析查询表达式，并生成一个高效的查询计划。
        - 查询监视：Mongo包含一个监视工具用于分析数据库操作的性能。
        - 复制及自动故障转移：Mongo数据库支持服务器之间的数据复制，支持主-从模式及服务器之间的相互复制。复制的主要目标是提供冗余及自动故障转移。
        - 高效的传统存储方式：支持二进制数据及大型对象（如照片或图片）
        - 自动分片以支持云级别的伸缩性：自动分片功能支持水平的数据库集群，可动态添加额外的机器。
- **适用场景**
    - 网站数据：Mongo非常适合实时的插入，更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性。
    - 缓存：由于性能很高，Mongo也适合作为信息基础设施的缓存层。在系统重启之后，由Mongo搭建的持久化缓存层可以避免下层的数据源 过载。
    - 大尺寸，低价值的数据：使用传统的关系型数据库存储一些数据时可能会比较昂贵，在此之前，很多时候程序员往往会选择传统的文件进行存储。
    - 高伸缩性的场景：Mongo非常适合由数十或数百台服务器组成的数据库。Mongo的路线图中已经包含对MapReduce引擎的内置支持。
    - 用于对象及JSON数据的存储：Mongo的BSON数据格式非常适合文档化格式的存储及查询。
- **redis、memcache、mongoDB 对比**
    - mongodb和memcached不是一个范畴内的东西。mongodb是文档型的非关系型数据库，其优势在于查询功能比较强大，能存储海量数据。

- memcached更为接近的是redis。它们都是内存型数据库，数据保存在内存中，通过tcp直接存取，==优势是速度快，并发高==，++缺点是数据类型有限，查询功能不强，一般用作缓存++。
- 性能：redis和memcache差不多，要大于mongodb
- 操作的便利性
    - memcache数据结构单一
    - redis丰富一些，数据操作方面，redis更好一些，较少的网络IO次数
    - mongodb支持丰富的数据表达，索引，最类似关系型数据库，支持的查询语言非常丰富
- 内存空间的大小和数据量的大小
    - redis在2.0版本后增加了自己的VM特性，突破物理内存的限制；可以对key value设置过期时间（类似memcache）
    - memcache可以修改最大可用内存,采用LRU算法
    - mongoDB适合大数据量的存储，依赖操作系统VM做内存管理，吃内存也比较厉害，服务不要和别的服务在一起
- 可用性（单点问题）
    - redis，依赖客户端来实现分布式读写；主从复制时，每次从节点重新连接主节点都要依赖整个快照,无增量复制==，因性能和效率问题，所以单点问题比较复杂==；不支持自动sharding,需要依赖程序设定一致hash机制。一种替代方案是，不用redis本身的复制机制，采用自己做主动复制（多份存储），或者改成增量复制的方式（需要自己实现），一致性问题和性能的权衡
    - Memcache本身没有数据冗余机制，也没必要；对于故障预防，采用依赖成熟的hash或者环状的算法，解决单点故障引起的抖动问题。
    - mongoDB支持master-slave,replicaset（内部采用paxos选举算法，自动故障恢复）,auto sharding机制，对客户端屏蔽了故障转移和切分机制。
- 可靠性（持久化：数据持久化和恢复）
    - redis支持（快照、AOF）：依赖快照进行持久化，aof增强了可靠性的同时，对性能有所影响
    - memcache不支持，通常用在做缓存,提升性能；
    - MongoDB从1.8版本开始采用binlog方式支持持久化的可靠性
- 数据一致性（事务支持）
    - Memcache 在并发场景下，用cas保证一致性
    - redis事务支持比较弱，只能保证事务中的每个操作连续执行
    - mongoDB不支持事务
- 数据分析
    - mongoDB内置了数据分析的功能(mapreduce),其他不支持
- 应用场景
    - redis：数据量较小的更性能操作和运算上
    - memcache：用于在动态系统中减少数据库负载，提升性能;做缓存，提高性能（适合读多写少，对于数据量比较大，可以采用sharding）
    - MongoDB:主要解决海量数据的访问效率问题。