1. **数据库内外网交换怎么做的，如果交换给你做，你怎么做**

方案一

           1.外网需要同步的表加上触发器，保存在临时表。

           2、task定时导出临时表记录,修改已导出记录的状态,上传到类似ftp目录上。  
           3、内网task通过连接Ftp取下文件,将记录导入到内网临时表,后台task通过调用存储过程解析数据

方案二

             1、编写应用路由程序，实现内外网通讯。  
             2、获取外网数据变革，转换为SQL脚本，由应用路由同步传输进入内网机器。  
             3、执行SQL脚本，更新内网数据。

1. **Linux修改权限的命令是什么，你对Shell编程了解多少，通常用来做什么**

 查看linux文件的权限：ls -l 文件名称  
查看linux文件夹的权限：ls -ld 文件夹名称（所在目录）  
修改文件及文件夹权限：  
sudo chmod -（代表类型）×××（所有者）×××（组用户）×××（其他用户）

r 表示可读取，w 表示可写入，x 表示可执行

hmod abc file   
其中a,b,c各为一个数字，分别表示User、Group、及Other的权限。   
r=4，w=2，x=1   
若要rwx属性则4+2+1=7；

shell应用

1．Node自动重启

2.利用crontab（定时处理命令）+shell实现每天对数据库备份：

0 6 \* \* \* backup

backup.sh

#!/bin/bash

# 数据库认证

user=""

password=""

host=""

db\_name=""

# 备份路径

backup\_path="/path/to/your/home/\_backup/mysql"

date=$(date +"%d-%b-%Y")

# 设置导出文件的缺省权限

umask 177

# Dump数据库到SQL文件，**mysqldump命令**是[mysql](http://man.linuxde.net/mysql)数据库中备份工具，用于将MySQL服务器中的数据库以标准的sql语言的方式导出，并保存到文件中。

mysqldump --user=$user --password=$password --host=$host $db\_name > $backup\_path/$db\_name-$date.sql

# 删除30天之前的就备份文件

find $backup\_path/\* -mtime +30 -exec rm {} \;

1. **设计数据库表时是否用到三范式？**

三范式：

第一范式（1NF）：符合1NF的关系中的每个属性都不可再分，实际上，1NF是所有关系型数据库的最基本要求

第二范式（2NF）满足第一范式,表中的字段必须完全依赖于全部主键而非部分主键。其他字段组成的这行记录和主键表示的是同一个东西，而主键是唯一的，它们只需要依赖于主键，也就成了唯一的

第三范式（NF）满足第二范式，非主键外的所有字段必须互不依赖，就是数据只在一个地方存储，不重复出现在多张表中，可以认为就是消除传递依赖

需要权衡代价，不一定要完全遵从三范式，使得表结构没有冗余。

第一，如果在设计的时候，已经明确地知道这个系统的数据量不会太大，比如一个中学的图书管理系统，最多有5万条书本的数据，过去一个月里借阅记录不会超过1千条。也就是说，表之间的关联代价不会太高，那么用“三范式”的原则是必需的。毕竟三范式能避免数据冗余带来的更新插入上的“需要同时多表里相同字段”的麻烦。

第二，如果表的数据量很大，我们可能就需要冗余数据。在订单流水表里，同时放入用户邮件地址和商品名的字段。

1. **假设一个a类变到b类，那么你用适配器模式怎么完成**

①、类适配器——通过继承来实现适配功能

1. **public** **class** AdapterDemo {
2. **public** **static** **void** main(String[] args) {
3. Ps2 p = **new** Adapter();
4. p.isPs2();
5. }
6. }
8. **interface** Ps2 {
9. **void** isPs2();
10. }
12. **interface** Usb {
13. **void** isUsb();
14. }
16. **class** Usber **implements** Usb {
17. @Override
18. **public** **void** isUsb() {
19. System.out.println("USB口");
20. }
21. }
23. **class** Adapter **extends** Usber **implements** Ps2 {
24. @Override
25. **public** **void** isPs2() {
26. isUsb();
27. }
28. }

②对象适配器模式——通过组合来实现适配器功能

1. **public** **class** AdapterDemo {
2. **public** **static** **void** main(String[] args) {
3. Ps2 p = **new** Adapter(**new** Usber());
4. p.isPs2();
5. }
6. }
8. **interface** Ps2 {
9. **void** isPs2();
10. }
12. **interface** Usb {
13. **void** isUsb();
14. }
16. **class** Usber **implements** Usb {
17. @Override
18. **public** **void** isUsb() {
19. System.out.println("USB口");
20. }
21. }
23. **class** Adapter **extends** Usber **implements** Ps2 {
24. **private** Usb usb;
26. **public** Adapter(Usb usb) {
27. **this**.usb = usb;
28. }
29. @Override
30. **public** **void** isPs2() {
31. usb.isUsb();
32. }
33. }
34. **数据库你通常是怎么优化的，使用过什么其他的工具优化没有，怎么用的**
    1. 选取最适用的字段属性，尽量把字段设置为NOT NULL
    2. 使用连接（JOIN）来代替子查询(Sub-Queries)

连接（JOIN）..之所以更有效率一些，是因为MySQL不需要在内存中创建临时表来完成这个逻辑上的需要两个步骤的查询工作。

* 1. 事务

以BEGIN关键字开始，COMMIT关键字结束

* 1. 使用外键

Foreign key()

On delete cascade 保证作为外键的表一条记录删除的时候，原表中所有有关记录也会被自动删除。

使用外键必须指定 engine = innodb

* 1. 使用索引

一般说来，索引应建立在那些将用于JOIN,WHERE判断和ORDERBY排序的字段上

* 1. 优化查询语句
  2. 分析慢查询

1）Windows下开启MySQL慢查询

MySQL在Windows系统中的配置文件一般是是my.ini找到[mysqld]下面加上

代码如下

log slow queries = F:/MySQL/log/mysqlslowquery.log

long\_query\_time = 2

2）Linux下启用MySQL慢查询

MySQL在Linux系统中的配置文件一般是是my.cnf找到[mysqld]下面加上

代码如下

log slow queries = /data/mysqldata/slowquery.log

long\_query\_time = 2

* 1. 使用redis缓存

pom.xml引入spring-boot-starter-data-redis库、

application.properties中配置redis的连接信息

新建一个类实现org.apache.ibatis.cache.Cache接口，使用Spring封装的redisTemplate来操作Redis

在mapper.xml中开启二级缓存，<cache type="com.wooyoo.learning.util.RedisCache"/>表示开启基于redis的二级缓存，并且在update语句中，我们设置flushCache为true，这样在更新product信息时，能够自动失效缓存（本质上调用的是clear方法）。

优化工具：

mysqldumpslow查看并分析慢查询日志

explain分析SQL，显示的信息：涉及到的表、连接类型、可能用到的索引、实际使用的索引、索引长度（不损失精确度情况下，越短越好）、Using temporary

1. **存储过程的异常怎么分类的，你所写的存储过程中，是如何使用异常的**

默认情况下，当存储过程运行出错时，过程会立即终止，并打印系统错误消息。

* 1. 定义异常处理

DECLARE ... HANDLER语句：（注意：declare……handler语句必须出现在变量或条件声明的后面）

当某个错误（condition\_value）发生时--->执行指定的语句（statement--记录错误信息），执行完之后再决定如何操作（handler\_action）。

　　1、handler\_action

　　　　continue：继续执行当前的程序(接着执行出错的SQL的下一条语句)；

　　　　exit：当前程序终止(退出当前declare所在的begin end)；

　　目前还不支持undo功能。

　　2、statement

　　　　可以是单条语句或复合语句。

3、condition\_value指明handler被何种条件触发；如果条件被触发，却没有handler被声明用于处理该条件，程序的进行将取决于条件类型。

②单个异常处理程序

在输出参数中包含出错消息的SQLSTATE码。

③多个异常处理

可以在一个过程中定义多个异常处理程序，针对不同的错误进行不同的处理。

④当不想为每个错误都定义一个处理程序时，可以使用3个处理程序

e.g：DECLARE CONTINUE HANDLER FOR SQLWARNING,NOT FOUND,SQLEXCEPTION

⑤异常命名

⑥异常传播

在嵌套块的情况下，内部块中发生异常了，首先由本块的异常处理程序来处理，如果本块没有处理，则由外部块的异常处理程序来处理。

1. **utf8和utf8mb4**

utf8mb4比utf8多了对emoji编码的支持

对应utf8mb4\_general\_ci utf8mb4\_unicode\_ci

1. **如何解决SQL注入问题**
   1. 采用正则表达式将包含有 单引号(')，分号(;) 和 注释符号(--)的语句给替换掉来防止SQL注入
   2. Mybatis实现了预编译，可防止SQL注入，因为sql注入只能对编译过程起作用，但是例外：sql语句使用${xxx}，纯粹的string替换，会引起SQL注入问题，因此一般使用#{XXX}

**9 .Linux查看CPU、内存占用**

CPU占用率查看：top

内存占用率：free –m

**10.数据库的对比**

**Oracal：**

**优点：** 高稳定性、可靠性。

 优秀的安全机制：没有读锁，采取快照SNAP方式完全消除了分布读写冲突。自动检测死锁和冲突并解决，并且数据安全级别为C2级（最高级）。

 在处理大数据方面Oracle会更稳定。

 兼容绝大多数主流平台。

 采用标准的SQL结构化查询语言。

 具有丰富的开发工具，覆盖开发周期的各阶段。

 支持大型数据库，数据类型支持数字、字符、大至2GB的二进制数据，为数据库的面向对象存储提供数据支持。

### 缺点

太贵。

管理维护麻烦一些；

对内存的高依赖导致数据库崩溃后回复很麻烦。

对数据库管理员的经验要求较高；

对硬件的要求很高；

Mysql：

优点：

 支持5000万条记录的数据仓库

 适应于所有的平台

 是开源软件，版本更新较快

 性能很出色。

 价格便宜

可自由选择引擎

缺点：

不支持热备份； 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；

SqlServer：

优点：易用性

图形化界面

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

缺点：只能在Windows上运行

价格高

不开放

# 11. JAVA访问数据库，从数据库连接池开始到数据库返回结果，都发生了哪些事情？

利用JDBC driver和mysql数据库建立TCP连接之后的连接对象放在池中，当需要操作数据库的时候从池中取出一个连接，发送sql到Mysql，Mysql经过语法解析、查询优化、生成实际物理计划及执行、链接处理与类型处理等一系列的过程之后返回要查询的数据给JDBC driver的resultset，把连接对象重新放到池子中。

**12.git使用**

git init 把目录变为仓库

git add <file>或git add .把文件添加到暂存区

git commit –m “ ” 把暂存区的所有内容提交到当前分支

git status 查看仓库的状态

git diff 查看修改的内容

git log显示提交日志，历史版本，以便确定回退到哪个版本

git reset –hard HEAD^ 回退到上个版本，其中HEAD表示当前版本，上上个版本HEAD^,上100个版本HEAD~100

Git的版本回退速度非常快，因为Git在内部有个指向当前版本的HEAD指针，当你回退版本的时候，Git仅仅是把HEAD从指向append GPL

git reflog 记录每次命令，以便确定回到未来哪个版本

git checkout –file 让文件回到最近一次git commit或git add的状态，可以恢复删除的文件

git push把当前分支推送到远程仓库

git pull拉取远程仓库到本地

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>