

**目 录**

[**一、 系统简介 3**](#_Toc17838658)

[**二、 系统需求 3**](#_Toc17838659)

[**1. 基本信息显示 3**](#_Toc17838660)

[**2. 数据操作 3**](#_Toc17838661)

[**3. 数据批量导入 3**](#_Toc17838662)

[**4. 项目展开 3**](#_Toc17838663)

[**5. 供应商查询 3**](#_Toc17838664)

[**三、 功能实现 4**](#_Toc17838665)

[**1. 数据库设计与实现 4**](#_Toc17838666)

[**2. 需求分析与实现 4**](#_Toc17838667)

[**（1） 登录与用户过滤 4**](#_Toc17838668)

[**（2） 数据库信息操作 4**](#_Toc17838669)

[**（3） 文件上传与批量导入 5**](#_Toc17838670)

[**（4） 产品展开结构树 5**](#_Toc17838671)

[**（5） 产品供应商过滤 6**](#_Toc17838672)

[**四、 系统测试 7**](#_Toc17838673)

[**1. 数据库测试 7**](#_Toc17838674)

[**2. 功能测试 7**](#_Toc17838675)

## 系统简介

针对东浦暑期实训中，最后阶段的实训考核内容进行简单的系统开发，业务需求根据东浦公司提供的需求文档进行设计与实现。

## 系统需求

#### 基本信息显示

所有数据库信息都可以显示，包括用户表，这里可以操作用户表主要是为了数据库测试。

#### 数据操作

首先实现用户登录系统，登录后的用户可以对数据库信息进行操作，具体的操作内容:

单条记录的定位，增加，修改，删除，维护单条记录需要考虑父子关系（难度2），叶子节点才能删除（难度1），需要考虑循环（难度3）。

#### 数据批量导入

从页面上传指定的txt文件，批量导入到数据库中 可以认为里面的数据都是正确，不需要校验 同时显示处理时间（难度3）。

#### 项目展开

输入项目号，进行树状结构的展开，并展现到web页面，同时需要展现处理时间。

#### 供应商查询

对指定产品展开树进行供应上查询过滤，叶子节点只保留该供应商零件，同时需要展现处理时间（难度5）。

## 功能实现

#### 数据库设计与实现

用户表: TB\_USER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用户ID | userId | varchar2（32） |  | 主键 |
| 用户密码 (加密) | userPwd | varchar2（33） |  |  |
| 用户姓名 | userName | varchar2（33） |  |  |

供应商表 TB\_SUP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供应商编号 | supId | varchar2（2） | 2位数，从10到99 | 主键 |
| 供应商名称 | supName | varchar2（128） |  |  |

产品项目表 TB\_ITMSTR

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目编号 | itmId | varchar2（32） |  | 主键 |
| 父项目编号 | itmPid | varchar2（32） | 根项目为NULL | 主键 |
| 项目名称 | itmName | varchar2（128） |  |  |
| 顺序号 | seq | number |  |  |
| 使用数量 | itmCount | number |  |  |
| 供应商编号 | supId | varchar2（2） |  |  |

所有数据库建立之后，小组成员进行相应的数据测试编写，得到测试用例数据。所有的数据用例是围绕一颗产品树构建的。

#### 需求分析与实现

###### 登录与用户过滤

在基本的用户登录之外，需要考虑用户直接利用URL进行后台登录，这时候可以利用保存登录会话进行信息存储，当没有登录信息的时候，会被检测到是未登录状态，让用户弹出。

###### 数据库信息操作

所有的数据库操作都是利用工厂模式进行创建和管理，使用之后自动进行资源回收和连接断开，保证系统效率。其次方法写完之后分给小组成员，对每一个方法进行数据测试，在确保所有的操作都不出现问题，才进行业务逻辑的实现。

###### 文件上传与批量导入

这个功能首先考虑到大量数据对服务器的内存承载，所以如果内存不够用，需要将上传的文件首先存在本地，然后在本地进行读取和数据库插入操作。

具体实现思路：首先创建页面，对用户上传操作进行验证，这里既可以验证上传文件格式，也可以确定上传文件的最大容量，之后服务器获取到这文件，并写入文件流，存储到项目中，同时利用该路径进行文件读取。

读取内容进行数据操作操作这里利用了java的JDBC中的addBatch操作，这里没有利用事物机制，主要考虑数据太多，如果中间回滚需要浪费更多的时间，就算数据在插入过程中失败了，错误操作也不会影响其他数据的插入操作。

同时为了考虑读取和插入速度，以及计算机和数据库性能，批量操作的数据容量也需要测试，在多次测试中我们发现，小于5000条的单次操作很快，但是需要大量的数据库连接与释放操作，这个过程耗费资源也浪费时间。将数据调整到3万条的时候，单次操作时间要大于5000条的时间的六倍，也就是说同时处理5000条，在每5000次提交和每30000次提交，前者的速度更快。即 T(5000) < T(30000) / 6 。

所以我们降低了单次批处理的容量，改为1.5W，结果和上面类似，所以再次选择降低容量，改为10000次提交模式，发现比提交两次5000所需要的时间要短，所以10000 – 15000次次之间是一个比较好的方案，最后我们选择了在文件中读取12000条之后，进行一次批处理。

###### 产品展开结构树

**要解决的问题**

* 从数据库查找对对应的数据
* 将从数据库中得到的数据转换为list树状
* 将list树状数转换为json数据格式
* 在前端页面上将json数据转换为树状展开形式

**实现思路与方法**

* 使用Oracle数据库自带的树状查询语句 （select…start with…connect by…prior）
* 查找所有可以构成一棵树的数据
* 根据父结点编号查询子树的数据
* 使用双重循环，把从数据库中查找到的list线性数据转换为带有子结点的list结点型数据。
* 对每一个数据要进行父亲节点判断（是否为null与node2结点的父项目，编号是否==node1的项目编号）
* 不符合条件：为父项目结点，添加到node list中。
* 符合条件：为子项目结点。在符合条件里面需要对父项目结点进行children结点是否为null，若==null，新建list到children。反之将子项目结点添加到父项目结点中的children。
* 以此类推直到所有数据都转换结束。
* 调用json数据转换类直接将node list数据转换成json数据。需导包（jackson-annotations-2.9.9.jar、jackson-core-2.9.9.jar、jackson-databind-2.9.9.jar）
* 接受到的json数据按照jquery-treeview.js的构件树状的格式进行展开。
* 展开过程中需要判断数据是否存在子节点。若存在，节点格式为父项目结点格式，反之为子项目结点格式。
* 最后在页面展示用到了tree-view 和回溯构建的方式完成的。

###### 产品供应商过滤

起初的思路是每次过滤当前产品结构树的叶子节点，对所有叶子节点的供应商进行过滤，然后得到一棵新的树结构，然后再次对这棵树进行叶子节点的供应商过滤。每次过滤的时候需要判断当前节点是否在当前这棵树中作为叶子节点还是父亲节点，控制好不同的处理流程。

这种方法因为是从叶子节点开始过滤，所以本质上还是从底层向上操作，所以基本的算法时间复杂度为log2n \* n2, 最坏为2 \*n2，所以算法不是很好，老师的方法在于直接遍历一次，从低向上操作因为得到的数据刚好是一个排列好的树结构，所以每次操作都是基于之前的操作的结论，根据节点结构进行操作的，所以只需要遍历一遍。

当然其中，我们在实现的时候遇到的问题是，如果当前树结构层次很深，过滤的节点跨度很大，会导致上层根节点忽略底层节点的操作情况，会丢失操作信息，所以添加一个栈来临时存储，，使用完毕后就会删除当前内容，不会造成不必要的循环。这样基本的算法时间复杂度为n 。

实际测试和总结两种方法，我们发现第一种方法虽然理论复杂度要远高于第二种，但是由于业务处理逻辑的关系，每次从最后循环，每次都只会循环最后一部分，所以不会全部循环，后续的循环中，不会再次循环已经被删除的节点，最终的数据操作数量级应该在同一个水平，所以基本的算法耗时虽然有差异，但仍然很短。一万条的数据过滤基本在0.2秒之内完成。

## 系统测试

#### 数据库测试

数据库在写好接口与实现后，交给小组成员进行测试的。

#### 功能测试

所有的实现功能，我们在50条记录，8千条，七万条，和180万条记录下都测试过了，同时实现了复杂操作的时间显示。