

**南京航空航天大学**

数据库系统概论实验报告

**实验：** **CVE公共漏洞库查询系统**

**姓 名： 张帅**

**学 号： 161410113**

**指导老师： 郑吉平**

**日 期： 2017年6月12日**

**CVE公共漏洞库查询系统**

161410113 张帅 18251829058

计算机科学与技术

Email：358029637@qq.com

**摘 要:** 一个简单的cve公共漏洞查询修改系统，用户可通过相关关键字查询相关漏洞的具体信息，包括cve编号，漏洞发现作者，漏洞软件，平台及公司等信息。并且查询后可对具体信息进行修改和删除。

**关键词:** JSP; DAO; SQL; CVE

# **概述**

cve漏洞是安全人员在研究挖掘漏洞过程中必须要参考借鉴的重要资源，本系统旨在实现一个小型的cve公共漏洞查询系统，可方便用户对cve漏洞进行查询修改。

# **需求分析**

## **2.1 需求陈述**

用户登录此系统即可对进行相关操作，主要功能有：插入漏洞信息。如果用户获得已cve漏洞的 具体信息，即可通过此功能插入数据库进行保存以方便日后查看。删除功能：如果不需要此漏洞信息， 用户可根据cve编号对cve漏洞进行删除。更新功能：用户可直接根据cve编号对漏洞信息进行更新， 当然也可以在漏洞信息查询完后对其进行更新。查询功能：用户可提供相关关键字对漏洞进行查询， 在查询时用户可根据单一信息进行查询，也可以尽量提供多的信息进行查询，当然也可以模糊查询。

## 2**.2 操作用例**

**一**旦用户登录系统，点击相应的按钮即可进行各种操作。当进行插入时，用户根据相应提示进 行相关信息的填写，当全部填写选择完毕时，即可点击插入按钮进行插入，如果插入成功或失败系统 会有相应的提示。其他操作都类似，只要根据系统提示进行相应操作即可。

**2.3 功能分析**

2.2.1 删除漏洞信息

用户进行删除时，系统会自动检测删除的信息是否为有效数据，如果不是，会提示用户没 有此数据。

2.2.2 更新漏洞信息

如果cve漏洞信息发行变更，用户可通过此功能进行修改。用户更新时，不能把cve编号 更新为已存在的编号，系统会自动提示更新不成功，因为已存在相应的记录。

2.2.3 查询漏洞信息

用户可根据自己的查询需求进行各种查询，在没有建立索引的情况下查询可能会比较慢， 所以在适当情况下可建立索引对其进行查询。

2.2.4 插入漏洞信息

如果用户得到cve的具体信息想要保存，即可通过此功能把信息插入数据库。同样，用户 只能插入不存在的cve信息，如果已存在，则会插入不成功。

2.2.4 用户注册功能

用户需要注册才能登录此系统进行相应的操作，否则，无法进行本系统的使用，本系统的 所有用户相当于管理员，所以注册功能也有一定限制，要提供相应的邀请码才行。

2.2.5 用户登录功能

同理，当用户注册完成后，随即可登陆此系统进行相应的操作。

## **2.4 对性能的规定**

表1 性能规定图

|  |  |
| --- | --- |
| 编号 | 性能要求 |
| 1 | 插入信息时间小于3秒 |
| 2 | 删除响应时间小于2秒 |
| 3 | 更新响应时间小于3秒 |
| 4 | 查询响应时间小于3秒 |
| 5 | 登录响应时间小于1.5秒 |

## **2.5 数据管理能力要求**

需要管理的记录个数：大于100万。记录的总个数每年将增长25%-30%。

## **2.6 故障处理要求**

发生错误时，保证数据完整，对于数据库发生故障时要能够进行故障恢复，以保证数据的一致 性。

## **2.7 运行环境**

运行系统：windows10

开发软件：myeclipse2016 + MySQL + JDK1.7

# **3 总体设计**

## 

密码

邀请码

## **3.1 E-R图**

用户名

用户

**n**

查询

平台

**m**

公司

漏洞编号号

漏洞

漏洞名称

软件名称

作者

描述

发布时间

类型

图1 E-R图

## 3**.2 初始对象表**

查询漏洞

更新漏洞

删除漏洞

插入漏洞

注册

登录

用户

图2 用例图

初始对象类/表：

|  |  |  |
| --- | --- | --- |
| 可能的类 | 英文类名 | 简单描述 |
| 用户 | User | 属性：用户名，密码，邀请码等基本属性。  方法：与用户对象相关的一系列操作。 |
| 漏洞 | Cve | 属性：漏洞id，漏洞作者，漏洞软件，漏洞平台，公司等属性。  方法：一系列与漏洞相关的操作。 |
| 漏洞类型 | Vuln\_type | 属性：id，漏洞类型名称等基本属性。  方法：一系列与类型相关的操作。 |

## **3.3 初始类图**

User

uid

username

password

invite\_code

Cve

cid

cve\_name

author

published\_time

description

type

app\_name

company

platform

Vuln\_type

tid

type\_name

proportion

0..n

1..n

图3 初始化类图

## **3.4 细化类图**

3.4.1 属性的识别

类名: User（用户类）

属性表如表2所示：

表2 User类属性表

|  |  |  |
| --- | --- | --- |
| 中文名 | 英文名 | 数据类型 |
| 用户ID | uid | int |
| 用户名 | username | varchar |
| 密码 | password | varchar |
| 邀请码 | invite\_code | varchar |

类名: Vuln\_type（漏洞类型类）

属性表如表3所示：

表3 Cve类属性表

|  |  |  |
| --- | --- | --- |
| 中文名 | 英文名 | 数据类型 |
| 类型ID | tid | int |
| 类型名 | type\_name | varchar |
| 类型比重 | proportion | float |

类名: Cve（漏洞类）

属性表如表4所示：

表4 Cve类属性表

|  |  |  |
| --- | --- | --- |
| 中文名 | 英文名 | 数据类型 |
| 漏洞ID | cid | int |
| 漏洞编号 | cve\_name | varchar |
| 作者 | author | varchar |
| 发布时间 | published\_time | date |
| 描述 | description | varchar |
| 类型 | type | int |
| 软件名称 | app\_name | varhcar |
| 公司 | company | varchar |
| 平台 | platform | varchar |

### 3.4.2 操作的识别

1. 操作的分类

基本操作：包括数据库检索和更新，如增加、删除、修改、分类、选择、查询。

关键操作：必须由对象提供的、在算法上复杂的业务操作。

1. 操作的识别

类名：User

操作表如表5所示：

表5： User操作表

|  |  |  |
| --- | --- | --- |
| 中文名 | 英文名 | 返回值类型 |
| 添加用户 | add\_user() | bool |
| 返回用户密码 | get\_user\_password() | string |

类名：Cve

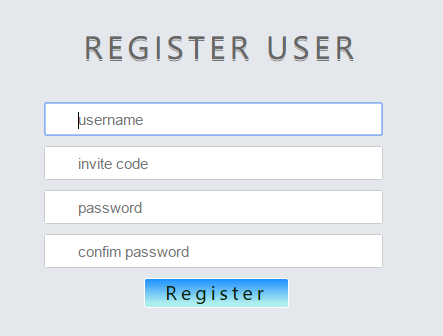
操作表如表6所示：

表6： Cve操作表

|  |  |  |
| --- | --- | --- |
| 中文名 | 英文名 | 返回值类型 |
| 插入数据 | insert\_data() | bool |
| 删除数据 | delete\_data() | bool |
| 更新数据 | update\_data() | bool |
| 查询数据 | search\_data() | cve |

## **3.5 界面设计**

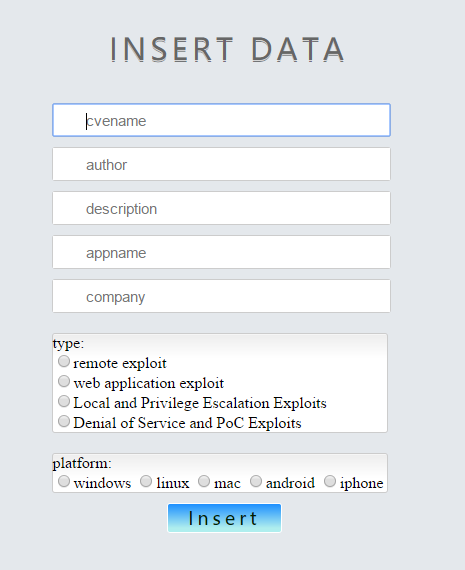
3.5.1 登录界面



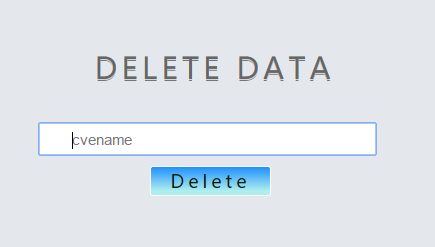
3.5.2 注册界面



3.5.3 插入数据



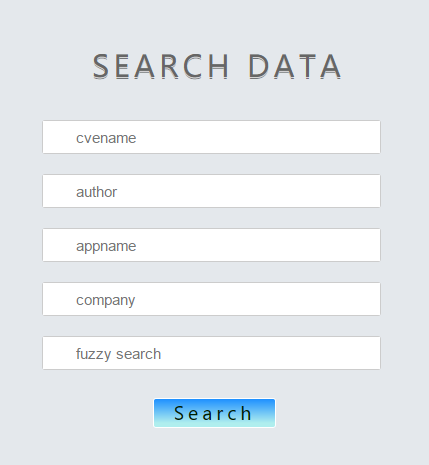
3.5.4 删除数据



3.5.5 更新数据

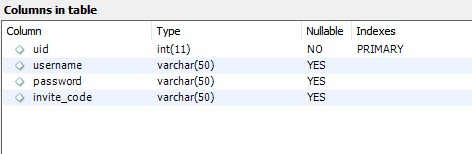


3.5.6 查询数据

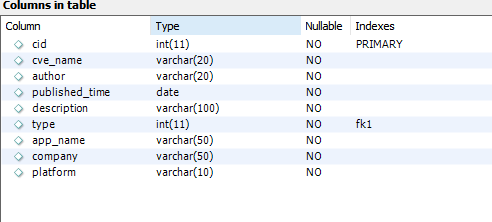


## 3.6 数据库结构设计

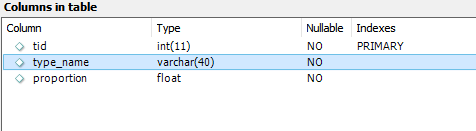
3.6.1 User 用户表



3.6.2 Cve 漏洞表



3.6.3 Vuln\_type 类型表



# **4 详细设计**

此系统是用DAO模式进行设计，开发人员可以将底层数据访问逻辑与业务逻辑分离开来，具体详细设计如下：

1.VO(Value Object)：一个用于存放网页的一行数据即一条记录的类，比如网页要显示一个用户的信息，则这个类就是用户的类。

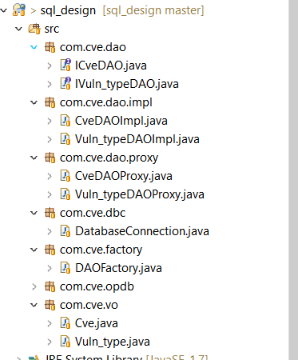
2.DatabaseConnection：用于打开和关闭数据库。

3.DAO接口：用于声明对于数据库的操作。

4.DAOImpl：必须实现DAO接口，真实实现DAO接口的函数，但是不包括数据库的打开和关闭。

5.DAOProxy：也是实现DAO接口，但是只需要借助DAOImpl即可，但是包括数据库的打开和关闭。

6.DAOFactory：工厂类，含有getInstance()创建一个Proxy类



**4.1 登录功能**

输入用户和密码

存在该记录？

**N**

**Y**

登录失败

登录成功

**4.2 注册功能**

输入用户,密码，邀请码

存在该记录？

**Y**

**N**

注册失败

注册成功

**4.3 插入数据**

输入漏洞相关数据

存在该记录？

**Y**

**N**

插入失败

插入成功

**4.4 删除数据**

确定漏洞编号

存在该记录？

**Y**

**N**

删除成功

删除失败

**4.5 更新和查询数据**

逻辑跟上面都类似。

# **实现**

**5.1 主要代码**

主要代码是DAO模式中对数据进行的相关操作，由于篇幅所限，只截取少许代码。

package com.cve.dao.impl;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import com.cve.dao.ICveDAO;

import com.cve.vo.Cve;

public class CveDAOImpl implements ICveDAO {

private Connection conn = null;

private PreparedStatement ps = null;

public CveDAOImpl(Connection conn){

this.conn = conn;

}

@Override

public boolean insertCve(Cve cve) throws Exception {

// TODO Auto-generated method stub

boolean flag = false;

String sql = "insert into cve (cve\_name, author, published\_time, description, type, app\_name, "

+ "company, platform) values (?, ?, ?, ?, ?, ?, ?, ?)";

this.ps = this.conn.prepareStatement(sql);

this.ps.setString(1, cve.getCve\_name());

this.ps.setString(2, cve.getAuthour());

this.ps.setDate(3, new java.sql.Date(cve.getPublished\_time().getTime()));

this.ps.setString(4, cve.getDescription());

this.ps.setInt(5, cve.getType());

this.ps.setString(6, cve.getApp\_name());

this.ps.setString(7, cve.getCompany());

this.ps.setString(8, cve.getPlatform());

if( this.ps.executeUpdate() > 0 ){

flag = true;

}

this.ps.close();

return flag;

}

@Override

public boolean deleteCve(String cvename) throws Exception {

// TODO Auto-generated method stub

boolean flag = false;

String sql = "delete from cve where cve\_name=?";

this.ps = this.conn.prepareStatement(sql);

this.ps.setString(1, cvename);

if( this.ps.executeUpdate() > 0 ){

flag = true;

}

this.ps.close();

return flag;

}

@Override

//just update cve\_name

public boolean updateCve(String new\_cvename, String old\_cvename, String description) throws Exception {

// TODO Auto-generated method stub

boolean flag = false;

String sql = "update cve set cve\_name=?, description=? where cve\_name=?";

this.ps = this.conn.prepareCall(sql);

this.ps.setString(1, new\_cvename);

this.ps.setString(2, description);

this.ps.setString(3, old\_cvename);

if( this.ps.executeUpdate() > 0 ){

flag = true;

}

this.ps.close();

return flag;

}