

**CSC 3170**

Database System

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CSC3170**

**Group Project Report**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Group Members:**

*Zhuoran Li 118020251*

*Yunfei Peng 117010214*

*Hesheng Huang 118010106*

*Wentian Tang 117010243*

**Group number:** *Group 37*

**Outline**

1.Introduction

2.Design and Implementation

2.1 Conceptual Design

2.1.1 Entity Set

2.1.2 Relational Schema and ER Diagram

2.2 Implementation

2.2.1 Design and Implementation of Web App

2.2.2 Sample Query

2.2.3 Security Measures

2.3 Data generating and Data analysis

3.Conclusion

4.Reference and Appendix

# 1. Introduction and motivation

Patents are the most effective carrier of technological information. They contain over 90% of latest technological intelligence in the world. The strict protection provided by patents on the intellectual property motivates people to innovate. It is a common view of people that patents are very necessary in our society, so the design of a patent system is very important.

The purpose of this project is to build a multi-functional patent system which can satisfy many patent related requirements. It provides 3 main functions: application, verification and analysis. These functions are provided to 3 different user groups to fulfill their requirements. The first function are the services for the applicators. The requirements of the applicators include applying for new patents, searching for data about their applications, patents and altering their own applications and patents. The second function is the services for the auditors. Auditors verify the application submitted by the applicators. Whether an application is pass or not is handled by them. So, they have the authority to change the application and patent status in the databases. The third function generates the statistical analysis for the patent data. For those who want to have a macroscopic viewpoint on the data of patents, function 3 is able to satisfy them. As we can see, the system is very comprehensive. Moreover, it introduces some security features like SQL injection protection. For the implementation, the system uses MySQL to manage data, and uses Django, HTML5 and JavaScript to provide an interactive interface. The following part of the report will present the detail about the design and implementation of this system.

# 2. Design and implementation

## 2.1 Conceptual design

2.1.1 Entities

The Entities involved in our systems can be divided into two groups: User and Records.

Users:

We divide the potential users into two groups. User Group 1 (ug1) is for anyone who wants to use our patent system to apply and manage their patents. They can create new applications for patents, create an application to change the owner of his/her existing patent, check the submitted application status and modify the applications that were submitted but haven't been processed yet. If the patent is approved, he/she will see the detail of the patent including the start date, expiration date, patent id, and so on.

If the application is rejected, it will see the detail of rejection.

User Group 2 (ug2) is for those who work for the patent bureau and is responsible to handle the applications from ug1. They will see the application type of an application of ug1 and corresponding details and decide if it should be rejected or not. If the patent is rejected, he/she should also include the reason for rejection and the number of patents referenced to reject the application.

Besides, we also consider that the data of patent is valuable and may help other companies in decision making, we also create User group 3. UG3 users can use the data in the database for data analysis purposes. The detailed queries of UG3 are included in section 2.1.3

Records:

There will be two main types of records included in our systems: application and patent. The application includes the application for a new patent and the application for change owner of a patent owned by him/her. For a different type of application, it has different contents.

patent stores all approved and granted patent, including its owner information, valid date, the content of patent, and id. The detail of the patent will be introduced in the next section. The content of the two records is similar, the reason that we need to maintain two tables are based on the principle of separation. As to keep the safety and integrity of data, a patent can only be inserted, updated, and deleted by ug2 users. For a ug1 user, he/she can only insert/modify the records in the application table and app\_change/app\_new table.

2.1.2 ER Diagram and Relational Schema

We have designed the schema appropriately so all schema has at most one candidate key, and there doesn't exist functional dependency or multi-value dependency on a non-prime attribute. Thus all schema is normalized to Boyce-Codd normal form. The only exception is the ref table, which has no primary key and is a trivial functional dependency.

0.ug1,ug2,ug3(id,uid,username,password){uid}->{username,password}

For each user group, it has four fields: username, password, and user\_id. user\_id is an internal identification within our system. The primary key of each is the user\_id. ID is used to uniquely identify the owner of this account. Id is a foreign key references the id in person table.

1.Applications(app\_id,app\_date,app\_type,uid){app\_id}->{app\_date,app\_type,uid}

Whenever a ug1 user creates an application, it will add one record into the application table. The app\_id is generated automatically by combining the applicant's user id and applications he/she has submitted. app\_date is generated when the submission is received. uid is the id of the applicant. app\_type is used to differentiate the application type: if app\_type=1, this means the application is for change owner of an existing patent. Otherwise, this application is for a new patent. The primary key is app\_id, and uid is the foreign key references uid in ug1.

We use attribute inheritance to store the information of applications in two weak entity sets: app\_new and app\_change. the table application table is considered as an identification entity.

2.app\_new(app\_id,ename,pname,lv1,lv2,lv3,text){app\_id}->{id,ename,pname,lv1,lv2,lv3,text}

This table stores the information for application for a new patent. id stands for the id number on the id\_card, ename is the name of the applicant, pname is the name of the patent. lv1 to lv3 is the index of three levels of categories. Generally, a patent can have multiple levels of categories. From lv1 to lv3 the category range narrows down. For example, A hard disk patent has lv1=Computer science,lv2=Computer hardware, lv3= Storage media. we use the number to represent different categories in each level. text is the description for a patent. The primary key is app\_id, which is also a foreign key that references app\_id in application table.

3.app\_change(app\_id,new\_owner,new\_otype,pid){app\_id}->{pid,new\_owner,new\_otype}

This table stores the information for application for change owner of an existing patent. A ug1 user can apply to change the owner of a patent that belongs to him. new\_owner is the id of a new owner, either the identity code of a natural man or the Unified Social Credit Identifier(UNISCID) of an organization. new\_type is the type of new owner. If new\_type=0, this means the new owner is another natural human. Otherwise, the new owner is a company or an organization. The primary key is app\_id, which is also a foreign key that references app\_id in the application table. pid is the id of the patent to be changed and is a foreign key that references pid in the patent table.

4.ref(app\_id,ref\_id) No Functional dependency

This stores the reference patent numbers of a new application. This is also a weak entity set. ref\_id is the id of the existing patent, it is a foreign key that references pid in the patent table.

5.result(app\_id,status\_1,status\_2,status\_3,s1\_id,s2\_id,s3\_id,finished,final\_status){app\_id}->{status\_1,status\_2,status\_3,s1\_id,s2\_id,s3\_id,finished,final\_status}

This table is for ug2 purposes. When an application is created, a record will be automatically inserted into this table. Considering that in reality a patent will be examined by different ug2 users to draw a final conclusion, we assume each patent will be examined by three different ug2 users independently, and the final conclusion will not be drawn until all status is set. Finished represents current state, -1 means not start yet, and each time a result is set, finished +1. The final status is true if all three previous status is true. When a ug2 user approves or rejects an application, the corresponding status id is set as him/her user id for responsibility check. app\_id is the primary key and also the foreign key references app\_id in application.

6.reject\_detail(app\_id,reason\_1,reason\_2,reason\_3){app\_id,reason\_1,reason\_2,reason\_3}

When an application is rejected, a ug2 user should also declare why to reject it. We choose the three most common reasons for patent rejection (Format error, patent already exist, Low originality). For each reason, the examiner should use some references to reject it, and the value of each reason is the number of quotations to reject. app\_id is the primary key and also the foreign key references app\_id in application.

7.patent(pid,start\_date,owner\_name,owner\_type,owner\_id,lv1,lv2,lv3,text){pid}->{start\_date,owner\_name,owner\_type,owner\_id,lv1,lv2,lv3,text}

When all three statuses were set and the final status is true, one record will be inserted or update depending on the type of application. Start\_date is by default the first workday of next month. Owner\_name owner\_id and owner\_type is initially the ename in the application, the id of the applicant, and personal. They can be changed after a patent is a grant. lv1,lv2,lv3, and text have the same definition as in the application table.Valid\_time is a derived attribute that varies from different categories and is fixed for the given category. It can be found in table valid\_time. Expire date is a derived attribute for a patent that equals the start\_date plus valid\_time. The primary key of the patent table is pid.

8.valid\_time(lv1,lv2,lv3,time){lv1,lv2,lv3}->{time}

This stores the valid time of patent in different categories. The primary key is lv1,lv2,lv3.

9.company(cid,cname,other\_attrs){cid}->{cname,other\_attrs}

This table is not owned by our system. It locates in an external database. We include this for data analysis purposes. cid is the UNISCID in app\_new and owner\_id in the patent which owner is company. cname is the company name. Other\_attrs can have many other attributes, such as asset scale, if a listed company, etc.

10.person(id,other\_attrs){id}->{other\_attrs}

This is another external table stores the id of all natural person. Other attributes include the gender, age, birthday, address, etc.

The correspounding ER Diagram is shown in the Appendix.

## 2.2 Implementation

2.2.1 Design and implement of web app

We will build an web application to implement the functions of our system. The system is designed based on the principle of separation. We will develop the frount-end and back-end separately and connect them with restful-api.

For the back-end, we use Django and Django REST Framework in python and uses MySQL as our database. For the frount-end, we use Django template . The frount-end uses HTML and jQuery.ajax to perform function and present data.

Since Some of our data query have complex structure, we use Django.connections.cursor to execute the original SQL command. The following are some Sample queries in our system

2.2.2 Sample query for ug1 and ug2

The following are sample query for user group 1 (applicants) and user group 2 (examiners), and their corresponding API name in our web app.

|  |  |  |
| --- | --- | --- |
| **API interface** | **SQL code** | **Note** |
| **General** | | |
| Create\_account | **insert** into{ug} set ( uid, username, password, id ) **values** (**select** count(uid) **from** {ug}); |  |
| login | **select** pwd, uid **from** ug1 **where** username={#user name}; |  |
| **Applicant User (UG1)** | | |
| UG1\_Search\_Patent | 1. **select** pid **from** patent **where** uid={#user id}; |  |
| UG1\_Search\_Application | 1. **select** app\_id,final\_status **from** result **where** app\_id in (**select** app\_id **from** application **where** uid={#user id}&&app\_type=0); 2. **select** status\_1,status\_2,status\_3,app\_id **from** result **where** app\_id in (**select** app\_id **from** application **where** uid={#user id}&&app\_type=1); |  |
| UG1\_insert | 1. **insert** **into** application (app\_id, app\_date, uid, app\_type) **values** (……); 2. **insert** **into** app\_new (app\_id,id,ename, pname, lv1, lv2, lv3, text) **values** (……); 3. **insert into** result (finished, status\_1, status\_2, status\_3, final\_status,s1\_id,s2\_id,s3\_id,app\_id) **values** (-1,……); | **Create an application for new patent** |
| UG1\_modify\_owner | 1. **insert** **into** application (app\_id, uid, app\_type, app\_date) **values** (……)； 2. **insert** **into** app\_change (app\_id, new\_owner, new\_type, new\_owner\_id) **values** (……)； 3. **insert into** result (finished, status\_1, status\_2, status\_3, final\_status,s1\_id,s2\_id,s3\_id,app\_id) **values** (-1,……); | **Create an application for change patent owner** |
| UG1\_modify\_application | 1. **update** application **set** pname={#patent name}, text={#description} **where** app\_id={#application id}; |  |
| **Examiner (UG2)** | | |
| ug2\_fetch | 1. **select** app\_id,new\_owner,owner\_type,new\_owner\_id,pid **from** app\_change **where** app\_id in (**select** app\_id **from** application **where** app\_id={#application id})&&app\_type=1; | **Fetch applications for change owner** |
| 1. **select** app\_id,ename,pname,lv1,lv2,lv3,text **from** app\_new **where** app\_id in (**select** app\_id **from** application **where** app\_id={#application id})&&app\_type=0; | **Fetch applications for new patent** |
| ug2\_change | 1. **update** pattent **set** owner=(**select** new\_owner **from** app\_change **where** app\_id={application\_id}), owner\_id=( **select** new\_owner\_id **from** app\_change **where** app\_id={application\_id}), owner\_type=( **select** new\_type **from** app\_change **where** app\_id={application\_id}) **where** pattent\_id={pid}**;** 2. **update** result **set** status\_1={0/1},finished=2, final\_status={0/1},s1\_id={uid} **where** app\_id={app\_id}; | **1.Change record in patent table**  **2. Change record in result table** |
| ug2\_new | 1. **update** result **set** {status}={0/1},{status\_id}={uid},finished=( **select** finished **from (select \* from** result**) as** temp **where** app\_id={app\_id}) **where** app\_id={app\_id}; 2. **update** result **set** final\_status = ( status\_1\* status\_2 \*status\_3) **where** app\_id={app\_id} **and** finished=2 | **1. Change record in result table**  **2. If three stage complete, generate final result automatically** |
| Reject | 1. **insert into**  reject\_detail (app\_id, status, reason\_1, reason\_2, reason\_3 ) **values** (……); |  |

2.2.3 Security measure

①User authority and identity check

Protect user’s private information such as password is an important part of the system. So we use one of the most popular: Data Encryption Standard algorithm (DES), it can effectively prevent any private data from being stolen.

The encryption process of DES consists of three main parts:

1、Initial permutation: Disarrange the order of the input 64-bit data blocks, and recombine each data according to the transposition table. Then, divide it equally into two 32-bit left and right parts.

2、Sixteen rounds of round changes: For R0, the transformation F is carried out under the control of the round key K1 (the round key is generated by the key expansion algorithm), and the result is noted as F (R0, K1). Then, the bit-by-bit XOR operation is conducted with L0, and the result is noted as R1, and R0 is directly used as L1 of the next round. This cycle is repeated for 16 rounds to obtain the pre-output results R16 and L16

3、Inverse initial permutation. Obtain the 64-bit ciphertext by using the inverse initial permutation IP-1 for the 64-bit results after the combination of R16 and L16.

In practical, we use des package in python to utilize the DES encryption and decryption.

②SQL injection and Defense of it

The security problem of a system with a SQL database is mainly from the structure loophole.

SQL injection is the most common and simplest loophole existed in the website. The main reason led to it is that the user input data is constructed into malicious SQL code, and the Web application does not examine the parameters used in dynamically constructed SQL statements.

Common SQL injection operations are mainly in following fields:

1、Guess the background database, this is the most common way to steal the sensitive information from the website.

2、Bypass the authentication, such as bypass verification and login in the site background.

3、Use database stored procedures for elevating authority and other operations

The core concept of defending SQL injection is to strictly examine the data inputted by users, and using the database with the minimum permission allocation principle.

We use following methods to defend the SQL injection attacks:

1、Matching filtering based on attack features: We construct a black list to record the injection feature. Once the input data matched the feature in the black list, then it will be identified as injection and rejected.

2、Transform the specific character inputted: We transform specific character used in injection statement to others. For example, we replace the character “ inputted into “/”, then the injection statement will not work.

## 2.3 Data generation and Analysis

Initially we want to use real world data to populate our database. However, the patent data of US patent bureau is confidential and not publically avalible. Thus we use our simulated data to fill our database.

Besides the two user group above, we also considered the chance that our data will serve as a public database and provide data services for some special clients such as consulting agency, investment bank and government, we added another user group: Third party users (UG3). UG3 have no authority to insert/update/delete any record, they can only query what is given by us.

Currently we have developed an function for them: Given an company id and the index of category, we can find the number of patents in all sub categories owned by this company.

|  |  |
| --- | --- |
| **API interface** | **SQL code** |
| Create view | create view ug3 as(select \* from ((select cid from company) as A left join (select owner\_id,pid,lv1,lv2,lv3,lv4 from patent where owner\_type=1) as B on A.cid=B.owner\_id); |
| UG3\_Behavior() | **Select** count({lv\_i}) **from** ug3 **where** ug3.cid={cid} **and** {lv\_j}={#lv\_j} for j<i |

# 3. Conclusion

This system is a powerful tool. Functions of this system can largely cover the procedure of processing patents. Moreover, the framework of this system can not only be applied to the patent system, but can also be applied to other auditing systems. For example, register system in a hospital works like this: patients submit the details of their sickness and the auditors in the hospital front desk will assign them to different doctors. As we can see, a hospital register system requires auditors to audit the applications submitted by patients and then provide the result to them. The framework of our system is applicable to it.

Of course, our system is not perfect. There are many ways to optimize it. For example, we can add records about auditors’ behaviors into the database in order to help us find out the auditor for a certain application. And there are more kinds of statistical data that can be provided by function 3. Overall, this system is comprehensive and its framework is widely applicable, but it can be optimized

Contributions

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Coding and other | Presentation | Report |
| Zhuoran Li | 1.Conception Design,create and optimize database  2.UG2 & UG3 Backend and frout end | Relational Schema and ER Diagram | 2.1,2.3, Format |
| Yunfei Peng | 1.UG1 backend  2.Data generation | User activities | 1,3 |
| Hesheng Huang | 1.UG1 frountend  2.Data generation | Demo operation | 2.2.1,2.2.2,Format |
| Wentian Tang | 1.slides | Introduction | 2.2.3 |

# 4. References

Demba, Moussa. (2013). Algorithm for Relational Database Normalization Up to 3NF. International Journal of Database Management Systems. 5. 39-51. 10.5121/ijdms.2013.5303.

Garrett, J. J. (2005). Ajax: A new approach to web applications.

Papenbrock, T., & Naumann, F. (2017, March). Data-driven Schema Normalization. In EDBT (Vol. 17, pp. 342-353).

Forcier, J., Bissex, P., & Chun, W. J. (2008). Python web development with Django. Addison-Wesley Professional.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (1997). Database system concepts (Vol. 4). New York: Mcgraw-hill.

Stallings, W., Brown, L., Bauer, M. D., & Bhattacharjee, A. K. (2012). Computer security: principles and practice (pp. 978-0). Upper Saddle River, NJ, USA: Pearson Education.

# 5. Appendix

ER Diagram

图示

描述已自动生成