Advance Database Management System

CSc8712 - Term Project

**A Report**

**on**

**“Library Management System”**

By

Jason Brelloch

Nabina Pradhan

Dec 10, 2013

**Library Management System**

**Introduction**

In this project called Library Management System, we have incorporated only basic features of book transactions in library. We have focus more on some advanced features such as book search engine and book recommendation engine which requires knowledge of indexing, scalability and user preference.

This project is basically designed keeping in mind that it helps with reducing librarians’ labor work. One of the important issues in a library management system is how to ensure system efficiency with respect to searching time under the fact that the number of books and users in a library increases at an incredible speed. This project resembles to a college library management system. The response time is a primary concern here.

Any person can become a member of the library by filling a prescribed form. They can get the book issued, so that they take home and return them.

**Library Management System features -**

1. Manage Books
2. Manage User Profile
3. Manage Book transaction
4. Search Engine
5. Warning for non-returned books
6. Book Recommendation

**Library Management System Users -**

In library Management System, there are two types of users

1. Administrator
   * Add, remove, modify book to the library database
   * Manage user profiles
2. Members
   * Sign up and become member to the Library management system by agreeing to the terms of Library management system.
   * Browse book collection based on various categories
   * Refer/Checkout books from the library
   * Access and manage their personal information
   * Search Books based on –
     + Type (book, journal, magazine, media)
     + Title
     + ISBN
     + Author name

The Library Management System is a web application created using *PHP, Javascript, JQuery and CSS*. The database used is *MySQL*. Communication between the database and web application is handled by MySQL’s PHP libraries. The search engine used is MySQL’s indexing capabilities to optimize book searches. The association rules mining techniques are used for recommendation engine to find commonly checked out book associations.

**Data structures**

There are number of data structures used in this Library management system in order to provide efficient recommendation to the users based on the book check out history and provide efficient book search engine.

1) Transactions table

* + A transaction table was created to record the issue of books, category of that book by each member of the Library database system. This transaction table will be provided as the input to the Recommendation engine which will provide effective future recommendation based on their book transaction history.

2) Index tables

* + Different number of tables such as *title\_index, book\_index* is created to index books and titles so that it could provide efficient search engine to search books based on category, book title, author title etc.

3) frequentItem tables

* + *frequentItem* table is created to hold information about books that are frequently checked out and the members that frequently check out books. This information will be used by the recommendation engine.

**Data Flow Diagram (DFD)**



Figure 1. Data flow diagram for Library Management System

**ER Diagram**

The member, books, library – correspond to entities in the Entity-Relationship model, and the operations to be done on those entities – holds, checkouts, and so on – correspond to relationships. However, a good design will minimize redundancy and attempt to store all the required information in a small space as possible.



Figure 2. ER diagram for Library Management System

This is the current ER diagram. We have added other entities and relationship according to the requirement or modification to the project.

**Software tools used**

* **Front End –** 
  + - 1. HTML
      2. PHP
      3. Javascript
      4. JQuery
      5. Twitter bootstrap
* **Back End -**

1. Github
2. Dot Net
3. Wamp Server 2.3
4. Microsoft Visio 2013
5. MySQL

**Assumptions**

• This application is used to convert the manual application to the online application.

• Customized data will be used in this application.

• User does not have right to enter information about books.

**Methodologies**

**GitHub**

It is the best place to share code with friends, co-workers, classmates, and complete strangers. Basically, it is an online hosting platform for sharing code and developing software. Github offers both paid and free user accounts, and has quickly become the most popular code repository site for open source projects.

Git is a distributed version control and source code management (SCM) system with an emphasis on speed. Every Git working directory is a full-fledged repository with complete history and full version tracking capabilities, not dependent on network access or a central server.

**Git Version Control**

Git is a distributed version control and file management system.

* + Everyone gets his or her own copy of the project
    1. Which means, it’s very fast
    2. Works offline
  + Track changes and merge them
    1. Developers can modified code
    2. Developers can add, moving, deleting files

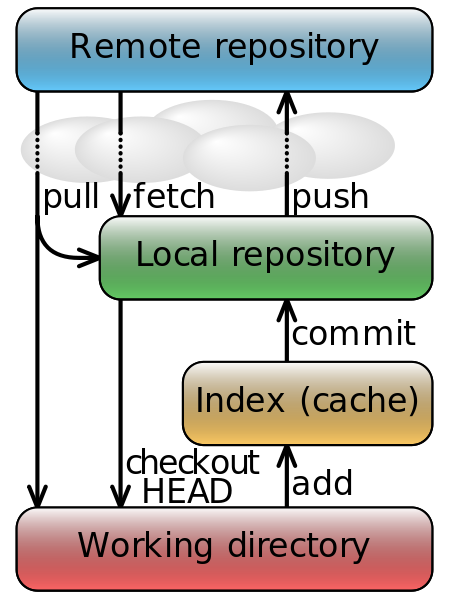


Figure . Working of Github

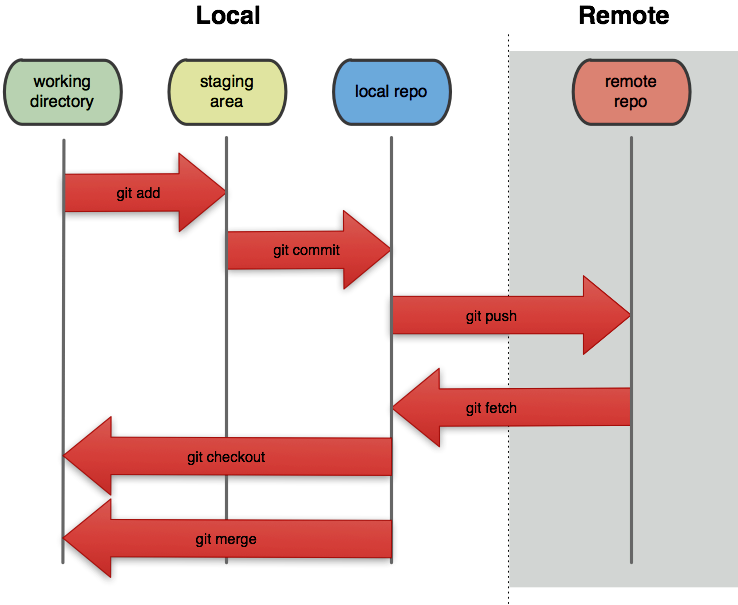


Figure . Github Flow

commit –a: Directly commit modified and deleted files into the local repository (no new files!)

add: Add a file to the staging area

checkout: Get a file from the staging area

checkout head: Get a file from the local repository

commit: Commit files from the staging area to the local repository

push: Send files to the remote repository

fetch: Get files from the remote repository

pull: Get files from the remote repository and put a copy in the working directory

***GitHub Flow***

* Anything in the master branch is deployable
* To work on something new, create a descriptively named branch off of master (ie: new-oauth2-scopes)
* Commit to that branch locally and regularly push your work to the same named branch on the server
* When you need feedback or help, or you think the branch is ready for merging, open a [pull request](http://help.github.com/send-pull-requests/)
* After someone else has reviewed and signed off on the feature, you can merge it into master
* Once it is merged and pushed to ‘master’, you can and should deploy immediately

**Front end and Back end Methodologies**

PHP is used to interact with the users. PHP is a [server-side scripting](http://en.wikipedia.org/wiki/Server-side_scripting) language designed for [web development](http://en.wikipedia.org/wiki/Web_development) but also used as a [general-purpose programming language](http://en.wikipedia.org/wiki/General-purpose_programming_language) PHP code is [interpreted](http://en.wikipedia.org/wiki/Interpreter_(computing)) by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an [HTML](http://en.wikipedia.org/wiki/HTML) source document rather than calling an external file to process data. It has also evolved to include a [command-line interface](http://en.wikipedia.org/wiki/Command-line_interface) capability and can be used in [standalone](http://en.wikipedia.org/wiki/Computer_software) [graphical applications](http://en.wikipedia.org/wiki/Graphical_user_interface). PHP is [free software](http://en.wikipedia.org/wiki/Free_software) released under the [PHP License](http://en.wikipedia.org/wiki/PHP_License), which is incompatible with the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL) due to restrictions on the usage of the term PHP.  PHP can be deployed on most web servers and also as a standalone [shell](http://en.wikipedia.org/wiki/Shell_(computing)) on almost every [operating system](http://en.wikipedia.org/wiki/Operating_system) and [platform](http://en.wikipedia.org/wiki/Computing_platform), free of charge. In order to make webpage in a more efficient manner, we have used twitter bootstrap in order to design the webpages. Bootstrap is a [free](http://en.wikipedia.org/wiki/Free_Software) collection of tools for creating [websites](http://en.wikipedia.org/wiki/Website) and [web applications](http://en.wikipedia.org/wiki/Web_application). It contains [HTML](http://en.wikipedia.org/wiki/HTML) and [CSS](http://en.wikipedia.org/wiki/CSS)-based design templates for [typography](http://en.wikipedia.org/wiki/Typography), forms, buttons, navigation and other interface components, as well as optional [JavaScript](http://en.wikipedia.org/wiki/JavaScript) extensions. We have used *JQuery* to check for different errors while handling forms.

MySQL which is open-source [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) is used to handle the entire database for this Library Management System. Procedures such as *initial\_dataLoad.*sql are used to load data in various tables.

**Design issues and challenges:**

Following are the design issues and challenges.

1. Synchronization of the code

Since two members are working in this project, it was difficult to track the changes in files and also to merge the files at the end. Hence, we created *github* account for this project and used the *github* based on windows platform in order to manage and merge different files.

2. Database design

The recommendation engine and search engine needs indexing based on book title, author title, member id and so on. Hence, all the primary keys are kept numeric in order to make indexing more efficient.

3. Recommendation engine/ Search Engine

The recommendation engine uses the Apriori algorithm to compute frequent item sets and create association rules. This algorithm is designed for use on databases containing lots of transactions which is perfect for a library management system. The basic process takes a bottom up approach in that it starts with the smallest possible sets and works its way up from there. This allows it to only focus on sets that have a potential to be a frequent set.

The process would start by finding all sets of length 1 item contained in the list of transactions. It then calculates the support for each length 1 item set by dividing the number of transactions that contain that item set by the total number of transactions. If this is above a certain threshold it then adds that item set to a group of frequent item sets.

The next iteration it will look for frequent item sets of length two, but only the ones containing frequent item sets from before. This allows it to ignore a lot of possible item sets and to run faster. The algorithm continues to run, searching for increasingly large item sets, until it stops finding any new frequent item sets.

For the library management system only item sets of length 3 and 4 were considered important, because that is how many are displayed on the details page.

The search engine is fairly simple in that it searches for books in the database by title, author or ISBN. The only interesting part is the optimizations added to make queries on those columns run faster. MySQL typically only indexes primary keys, but can be forced to create indexes on popular columns that will be queried frequently. This was done on these 3 columns to hopefully increase the speed of this search.

4. Populate the table.

In order to test search engine and recommendation engine, we need huge data and to enter data manually in the table is very cumbersome and time consuming. Hence, to populate the data in the tables, we used database scripts and procedures..

**Lessons learned:**

Apart from getting more detailed understanding of the various topics of advanced database and get more familiar with different development tools, we learned following lessons which could be very useful for our future endeavors.

1. Correct and Efficient Database design

All the tables and primary keys and foreign keys and also indexing table should be carefully designed before starting the project. There were lots of instances where we have to modify the table and in doing so, we have to spend a lot of time fixing the code and managing other issues that arise with change in table structure.

2. Implementation of Agile Methodologies

Agile Methodologies is a solution to software development where all the members of the project develop the software through collaboration between team members. It promotes tight interaction between team members and also encourage team members develop software together. This makes interaction very easy as team members are easily accessible and any changes to the design can be made by quick interaction with the team members.

**Summary and explanations of the test cases and results**

The library management system was tested against several different size databases to simulate different use cases. The important factor in testing the search engine was the number of books, while the key factor in testing the recommendation engine is the number of orders and transactions.

The search engine showed very little need for the indexes created to optimize speed. Even when the number of books in the database exceeded 100,000 there was very little difference in response time between a database using the indexes versus one that was not (miliseconds). This could be due to several factors. One, all the databases were hosted locally which cuts out the traditional network bottleneck. Two, it is possible that MySQL was noticing the large dataset being queried and created an index by itself.

The number of orders and transactions used to test the recommendation ranged from 1000 to 20000, which corresponds to 7500 to 75000 transactions. The algorithm exhibited almost exponential growth in the amount of time it took to process the data. This was a result of several factors. First, the algorithm used was very simple without taking advantage of modern optimizations to association rule mining, such as data distribution and parallelization of processing. Second, the data sets were generated making heavy use of MySQL’s rand() function. This function is not truly random, but gives a wide distribution of possibilities. This means the support threshold had to be set low to get any frequent item sets, which resulted in lots of frequent item sets and contributed to long execution time.

**Apriori Algorithm**


\begin{align}
& \mathrm{Apriori}(T,\epsilon)\\
&\qquad L_1 \gets \{ \mathrm{large~1-item sets} \} \\
&\qquad k \gets 2\\
&\qquad\qquad \mathrm{\textbf{while}}~ L_{k-1} \neq \ \mathit{empty set} \\
&\qquad\qquad\qquad C_k \gets \{ a \cup \{b\} \mid a \in L_{k-1} \land b \in \bigcup L_{k-1} \land b \not \in a \}\\
&\qquad\qquad\qquad \mathrm{\textbf{for}~transactions}~t \in T\\
&\qquad\qquad\qquad\qquad C_t \gets \{ c \mid c \in C_k \land c \subseteq t \} \\
&\qquad\qquad\qquad\qquad \mathrm{\textbf{for}~candidates}~c \in C_t\\
&\qquad\qquad\qquad\qquad\qquad \mathit{count}[c] \gets \mathit{count}[c]+1\\
&\qquad\qquad\qquad L_k \gets \{ c \mid c \in C_k \land ~ \mathit{count}[c] \geq \epsilon \}\\
&\qquad\qquad\qquad k \gets k+1\\
&\qquad\qquad \mathrm{\textbf{return}}~\bigcup_k L_k
\end{align}


**Database Scripts**

CREATE TABLE members (

`m\_id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`email` varchar(50) NOT NULL UNIQUE,

`password` varchar(50) DEFAULT NULL,

`first\_name` varchar(50) DEFAULT NULL,

`last\_name` varchar(50) DEFAULT NULL,

`date\_of\_birth` datetime(6),

`book\_limit` int(10) unsigned,

`status` bit(1),

`admin` bit(1),

PRIMARY KEY (`m\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE addresses (

`m\_id` int(10) unsigned NOT NULL,

`line\_1` varchar(50) DEFAULT NULL,

`line\_2` varchar(50) DEFAULT NULL,

`line\_3` varchar(50) DEFAULT NULL,

`city` varchar(50) DEFAULT NULL,

`zip` varchar(50) DEFAULT NULL,

`state` varchar(50) DEFAULT NULL,

`country` varchar(50) DEFAULT NULL,

`other` varchar(50) DEFAULT NULL,

FOREIGN KEY (`m\_id`) REFERENCES members(m\_id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE books (

`b\_id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`title` varchar(50) DEFAULT NULL,

`author` varchar(50) DEFAULT NULL,

`isbn` varchar(50) DEFAULT NULL,

`quantity` int(10) unsigned,

`subject` varchar(50) DEFAULT NULL,

`details` varchar(1000) DEFAULT NULL,

`imageurl` varchar(250) DEFAULT NULL,

PRIMARY KEY (`b\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE INDEX title\_index ON books(`title`) USING BTREE;

CREATE TABLE orders (

`o\_id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`m\_id` int(10) unsigned NOT NULL,

PRIMARY KEY (`o\_id`, `m\_id`),

FOREIGN KEY (`m\_id`) REFERENCES members(`m\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE transactions (

`o\_id` int(10) unsigned NOT NULL,

`b\_id` int(10) unsigned NOT NULL,

`quantity` int(10) unsigned NOT NULL,

PRIMARY KEY (`o\_id`, `b\_id`),

FOREIGN KEY (`o\_id`) REFERENCES orders(`o\_id`),

FOREIGN KEY (`b\_id`) REFERENCES books(`b\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE frequentItems1 (

`b\_id` int(10) unsigned NOT NULL,

`frequency` double,

PRIMARY KEY (`b\_id`),

FOREIGN KEY (`b\_id`) REFERENCES books(`b\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE frequentItems2 (

`b\_id1` int(10) unsigned NOT NULL,

`b\_id2` int(10) unsigned NOT NULL,

`frequency` double,

PRIMARY KEY (`b\_id1`,`b\_id2`),

FOREIGN KEY (`b\_id1`) REFERENCES books(`b\_id`),

FOREIGN KEY (`b\_id2`) REFERENCES books(`b\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE frequentItems3 (

`b\_id1` int(10) unsigned NOT NULL,

`b\_id2` int(10) unsigned NOT NULL,

`b\_id3` int(10) unsigned NOT NULL,

`frequency` double,

PRIMARY KEY (`b\_id1`,`b\_id2`,`b\_id3`),

FOREIGN KEY (`b\_id1`) REFERENCES books(`b\_id`),

FOREIGN KEY (`b\_id2`) REFERENCES books(`b\_id`),

FOREIGN KEY (`b\_id3`) REFERENCES books(`b\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

CREATE TABLE frequentItems4 (

`b\_id1` int(10) unsigned NOT NULL,

`b\_id2` int(10) unsigned NOT NULL,

`b\_id3` int(10) unsigned NOT NULL,

`b\_id4` int(10) unsigned NOT NULL,

`frequency` double,

PRIMARY KEY (`b\_id1`,`b\_id2`,`b\_id3`,`b\_id4`),

FOREIGN KEY (`b\_id1`) REFERENCES books(`b\_id`),

FOREIGN KEY (`b\_id2`) REFERENCES books(`b\_id`),

FOREIGN KEY (`b\_id3`) REFERENCES books(`b\_id`),

FOREIGN KEY (`b\_id4`) REFERENCES books(`b\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

delimiter $$

CREATE PROCEDURE dataLoad()

BEGIN

DECLARE a INT;

DECLARE b INT;

DECLARE c INT;

DECLARE d INT;

DECLARE i1 INT;DECLARE i2 INT;DECLARE i3 INT;DECLARE i4 INT;DECLARE i5 INT;DECLARE i6 INT;DECLARE i7 INT;DECLARE i8 INT;DECLARE i9 INT;

DECLARE cat INT;

DECLARE category VARCHAR(20);

DECLARE imageurl VARCHAR(250);

DECLARE details VARCHAR(1000);

DECLARE stopper INT;

DECLARE member\_holder INT;

DECLARE order\_holder INT;

/\* load books \*/

SET details = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.";

SET a=1;

WHILE (a<=100) DO

SET i1 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i2 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i3 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i4 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i5 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i6 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i7 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i8 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET i9 = FLOOR(0 + RAND() \* (10));/\* 0-9 \*/

SET cat = FLOOR(1 + RAND() \* (6));/\* 1-6 \*/

IF cat = 1 THEN SET category = 'Textbook';

SET imageurl = "http://www.webweaver.nu/clipart/img/education/stack-of-books.png";

ELSEIF cat = 2 THEN SET category = 'Historical';

SET imageurl = "http://www-tc.pbs.org/wgbh/aia/part2/images/2cris2378b.jpg";

ELSEIF cat = 3 THEN SET category = 'Biography';

SET imageurl = "http://www.biography.com/imported/images/Biography/Images/Profiles/E/Albert-Einstein-9285408-1-402.jpg";

ELSEIF cat = 4 THEN SET category = 'Fantasy';

SET imageurl = "http://s3.amazonaws.com/rapgenius/1362582359\_unicorn.jpg";

ELSEIF cat = 5 THEN SET category = 'ScienceFiction';

SET imageurl = "http://static4.wikia.nocookie.net/\_\_cb20130310133315/starwars/images/5/58/Soldier\_stub.png";

ELSEIF cat = 6 THEN SET category = 'Romance';

SET imageurl = "http://nyoobserver.files.wordpress.com/2013/01/50-shades-of-grey-cover-thumbnail.jpeg";

ELSEIF cat = 7 THEN SET category = 'ERROR';

END IF;

INSERT INTO books (`b\_id`,`title`,`author`,`isbn`,`quantity`,`subject`,`details`,`imageurl`) VALUES(NULL,CONCAT('title\_',a),CONCAT('author\_',a),CONCAT(i1,i2,i3,i4,i5,i6,i7,i8,i9),10000,category,details,imageurl);

SET a=a+1;

END WHILE;

/\* load members and addresses \*/

SET b=1;

WHILE (b<=200) DO

INSERT INTO members (`m\_id`,`email`,`password`,`first\_name`,`last\_name`,`date\_of\_birth`,`book\_limit`,`status`,`admin`) VALUES(NULL,CONCAT('user\_',b,'@users.com'),CONCAT('user\_',b),CONCAT('first\_',b),CONCAT('last\_',b),'1111-11-11 11:11:11',100,1,0);

SET member\_holder=LAST\_INSERT\_ID();

INSERT INTO addresses (`m\_id`,`line\_1`,`line\_2`,`line\_3`,`city`,`zip`,`state`,`country`,`other`) VALUES(member\_holder,CONCAT('line1\_',b),CONCAT('line2\_',b),CONCAT('line3\_',b),CONCAT('city\_',b),CONCAT('zip\_',b),CONCAT('state\_',b),CONCAT('country\_',b),CONCAT('other\_',b));

SET c=1;

/\* load transactions \*/

WHILE (c<=100) DO

INSERT INTO orders (`o\_id`,`m\_id`) VALUES(NULL,member\_holder);

SET order\_holder=LAST\_INSERT\_ID();

SET stopper = FLOOR(1 + RAND() \* (7));/\* each order will have 1-7 transactions \*/

SET d=1;

WHILE (d<=stopper) DO

INSERT INTO transactions (`o\_id`,`b\_id`,`quantity`) SELECT order\_holder,books.b\_id,1 FROM books WHERE b\_id=FLOOR(1 + RAND() \* (100)) ON DUPLICATE KEY UPDATE transactions.quantity=transactions.quantity+1;

SET d=d+1;

END WHILE;

SET c=c+1;

END WHILE;

SET b=b+1;

END WHILE;

END$$

delimiter ;

call dataLoad();