Library Management System

Report Submitted By:
Name: Tariqul Islam
Id: 15.01.04.081

Group Members:

1.Mashrur Tajwar -15.01.04.062

2.Tariqul Islam -15.01.04.081

Introduction:

The Library Management System project is developed based on distributed database. In this project

- Our system takes direct user input:

```
Enter book id:5002
Enter price :300
Book Price is updated
```

- an admin is required to add other admins, members and books.
- Members can access the whole database from their home as the project is based on distributed database but if they want to borrow a book they have to come to the library in person.
- Members can also search book from their computer
- Four different tables are required to complete the database. Three of them are member, book and admin table. The forth table is the table which keeps all the record of the members who have already borrow book.
- For general purpose, we assume that one member can borrow only one book at a time.

Overview of the system:

The project starts with two options.

- One is to login as admin
- The other one is to login as member.

Here, the user has to provide used-id and password to login in his account.

Admin Access:

After login as admin, first of all the admin has to put the user-id and password. These admin ids are **generated automatically**. An **admin has all the access** around the library. Only s/he can add another admin, member and book. For adding an admin, the main admin has to put the values of name, address, phone number and password. The new admin's id will set automatically. To add another member the admin has to give name, address, phone number and password. New member's id will also be set here automatically. Finally, to add a book we have to give title, author, genre, price, quantity, available as user input. Book id will be generated automatically here. The last functionality that a admin can perform is to search. S/he can search the book list by different categories. One can search book by the book name or the genre or the author.

Member Access:

Let's take a look at the members of Library Management System. As it is a distributed database, any member can access their account via their computer and internet. By accessing their account, they can see that is he bring a book from library or not. He can also see the expired date that is given by the librarian if he brought a book from the library. In future we will add the due money in this table. Members can also search books of the library from their home. They can search the book list same as admin.

How the system works:

Borrow Book:

In this part we have a separate table named 'borrowBook'. The attributes of this table are 'bookid', 'memberid', 'receiveday', 'expiredday'. When the **member tends to borrow a book**, he has to come to the admin or librarian with the book. Then the admin has to **provide userid and bookid** in program. If the member doesn't exist in the member table then **it will show** an error **"Member doesn't exist"**. We have to keep that in mind that for the simplicity of our project we assume that one member can borrow only a book from the library at the same time. **Only after returning the book** he will be **able to take another boo**k from the library. The project shows an error message **"Member x already possesses a book y"**. here x and y represent member-id and book-id respectively. Finally, if the member exists in the table and he does not possess a book only then he will able to borrow a book from the library.

Let, a member is **able to bring a book** then the table named **borrowBook will be updated**. Two dates will be **inserted automatically** in the table **(receive-day, expire-day)**. At the same time, **book table and member table** will also be updated. In the **book table the quantity will be decreased**. In the **member table the 'book availability' attribute** will be set to **YES from NO** and also the column **expiration day will be set the expired day**.

Return Book:

The process is almost same as to take the book from the library. When the **user returns the book** the row of the **corresponding member** in the **borrowBook table will be deleted**. At the same time the **attribute named quantity** in the book table **will be incremented by 1**. The member table will update the **'book availability' attribute to NO from YES** and also update the '**expired day' attribute to NULL**.

In our project we have been used cursor, function, procedure, package, sequence,trigger etc.

Contribution:

1. Tables & Fragmentation of the tables.
2. Update Procedure,trigger,sequence,receive and return book.

Database Links:

We use two different laptops to make our database distributed. The code we are using to make the connection is giving below:

```
drop database link sitt2;

create database link site2;

connect to system identified by "123456"
```

using '(DESCRIPTION =

(ADDRESS_LIST =

(ADDRESS = (PROTOCOL = TCP)

(HOST = 192.168.0.5)

(PORT = 1521))

)

(CONNECT_DATA =(SID = XE))' ;

Sites and Tables:

We have implemented our distributed database project using two separate laptops. One of the laptop is the host and the other one is the site. Here **host** is **site1**.

Global Relation:

Admin (adminId, adminName, adminAddress, adminPhone, adminRegDate, adminPassword);

Member (memberId, memberName, memberAddress, memberPhone, memberRegDate, memberPassword, receivedBook, expiredDate);

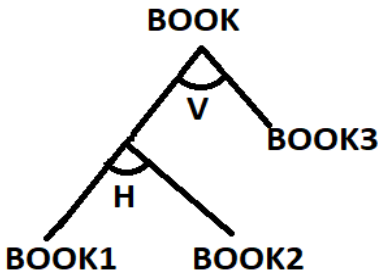
Book (bookId, bookTitle, bookAuthor, bookPrice, bookQuantity, bookAvailable);

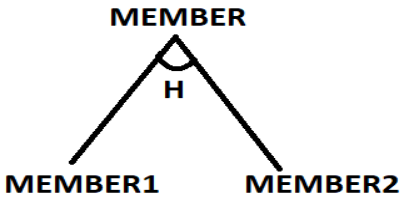
BorrowBook(memberId, bookId, receiveDay, expiredDay);

Fragmentation Schema:

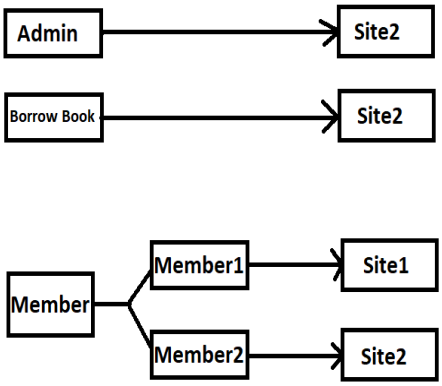
Admin= $Pj_{Id,Name,Address,Phone,RegDate,Pawssword}$(Admin)
BorrowBook= $Pj_{Id,Name,Address,Phone,RegDate,Pawssword}$(BorrowBook)
Member1=$Sl_{memberAddress='Mirpur' and memberAddress='Tejgaon'}$(Member)
Member2=$Sl_{memberAddress!='Mirpur' and memberAddress!='Tejgaon'}$(Member)
Book1=$Sl_{bookprice<100}$(Book)
Book2=$Sl_{bookprice>=100}$(Book)
Book3=$Pj_{bookid,quantity,availability}$(Book)

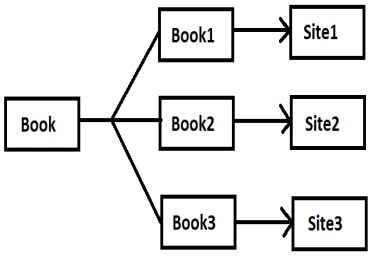
Fragmentation Tree:





Allocation Schema:





Packages, Functions and Procedures:

In our project we have one function and one package to show the demonstration. We mainly focused on procedures because working via procedure is more comfortable. We have three different procedures in the project.

Package:

addPackage():

- Description: we put a procedure in the package. In that procedure which is inside the package inserts the books depending on different genre.

- Exception: NULL_VALUE, INVALID_NUMBER, VALUE_ERROR

Function:

RecieveBook():

- Takes member id and book id
- Return: varchar2 (if the member can borrow the book or not)

Example: When member does not exist.

```
Enter Member ID: 5002
Enter Book ID: 5001
Member does not exist!

PL/SQL procedure successfully completed.
```

Example: When book does not exist.

```
Enter New Member Name: Mashrur
Enter New Member Address: Dhaka
Enter New Member Phone: 0111111
Enter New Member Password: 1234
New Member Created !
Your ID: 120 AND Your Password: 1234

PL/SQL procedure successfully completed.
```

Example: When book is successfully recieved

```
Enter Member ID: 105
Enter Book ID: 5002
Old Qauntity: 10
New Qauntity: 9
Successfully Updated!
```

ReturnBook():

- Takes member id and book id
- Return: varchar2
- Example: When book is successfully returned

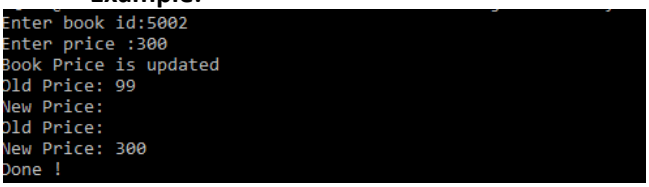
```
Enter Member ID: 105
Enter Book ID: 5002
Old Qauntity: 9
New Qauntity: 10
```

Search ()

- Parameter: num number, name varchar2
- Return: number
- Description: takes a number. ‘1’ denotes book title wise search, ‘2’ denotes author wise search and ‘3’ denotes genre wise search. Then takes a varchar. It denotes the name that the user wants to search.
- Exception: NONE

Update Book Price:

Parameter: bookId, bookTitle, bookAuthor, bookPrice, bookQuantity, bookAvailable
Return: number
Description: takes bookid and updates book table according to price.
Exception: NONE
Example:



Example Update Process:
Update Price to 200 where bookid=1;
- Store corresponding data of given bookid.
-Insert in Book2 With Updated Value
-Delete data from book1.
Book1:

bookId	bookTitle	Author	price	Quantity	Available
1	Himu	Humayan	100	5	Dhaka

Book2:

bookId	bookTitle	Author	price	Quantity	Available
1	Himu	Humayan	200	5	Dhaka

Book1:

bookId	bookTitle	Author	price	Quantity	Available
~~1~~	~~Himu~~	~~Humayan~~	~~100~~	~~5~~	~~Dhaka~~

Procedure:

addAdmin()

- Parameter: name in varchar2, address in varchar2, phone in number, password in varchar2
- Return: NONE
- Description: to add a new admin in the database
- Exception: INVALID_NUMBER, VALUE_ERROR

addMember()

- Parameter: name in varchar2, address in varchar2, phone in number, password in varchar2
- Return: NONE
- Description: to add a new member in the database
- Exception: INVALID_NUMBER, VALUE_ERROR

addBook()

- Parameter: title in varchar2, author in varchar2, genre in varchar2, price in number, quantity in number, available in varchar2
- Return: NONE
- Description: to add a new book in the database
- Exception: NULL_VALUE, INVALID_NUMBER, VALUE_ERROR

Sequence:

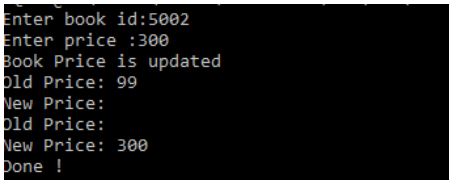
- Sequence are used for incrementing member Id, admin id and different genre id.

Triggers:

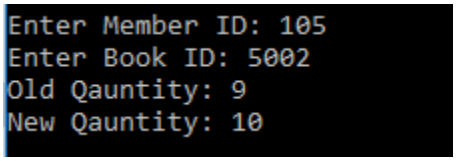
Book Price Update Trigger for Table: Book

- Shows the old price, updated price and difference between the price.

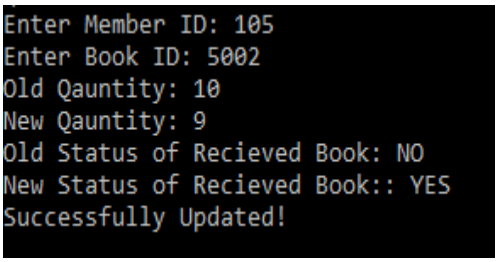
Example of Trigger:



Book Quantity Update Trigger for Table: Book



Member Receive Book Trigger: Member



Conclusion & Future Work:

This is a simple library management system which keeps track the member activity of borrowing book and returning book and also helps to search all the information of the book. The system can update book information and the tables are fragmented based on conditions and allocated in different sites.
The limitation of the system is the member can only borrow one book at a time, we will try to overcome this limitation and make this and make this application usable in real life with proper and user friendly interface.