|  |
| --- |
| ***Library Management System*** |

**Contents**

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Title** | **Page No.** |
|  | Introduction | 3 |
|  | Objectives | 3 |
|  | Tools and Techniques | 3-4 |
|  | Features | 4-5 |
|  | E-R diagram | 5-6 |
|  | SQL Code for Entity Set | 6-9 |
|  | Data Insertion into Entity Set | 9-11 |
|  | Query Implementation on Entity Set | 11-12 |
|  | Limitations | 13 |
|  | Conclusion | 13 |
|  | References | 13 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Title** | **Page No.** |
| 1. | An Entity-Relationship Diagram for Library Management System | 5 |
| 2. | SQL Statements | 6-11 |
| 3. | Entity Set | 11 |
| 4. | Query Implementation Statements | 11-12 |

1. **Introduction**

Library database system is designed to maintain the library assets in a suitable way. This report is about a relational database design of a school library in which members of the library can keep track on availability of books, information about membership either they are student or staff of school, they can join or leave library and some responsibilities of librarians are added.

The core goals of this report are to define the entities, attributes and their relationship to create a database that can be easily managed by admin. **Following are the facilities:**

**Librarian:** can find if the book is available or not its status and how much fine is due

**Member:** can borrow & return books and can leave and join library

In the report the findings describe the deficiency of data keeping method and, which can be increase the budget to hire manpower. Following these findings, conclusions are drawn and a number of recommendations are made as to how to increase the reliability of data, reduce redundancy and accessibility to data has been made stress-free.

1. **Objectives**

The objectives of this report are to provide a database management system for school library which will help to minimize the duplication of data, get rid from paper filling system and it makes the processing of data very easy without waste of time.

1. **Tools and Techniques**

Existing system of the library is manual system which has lack of functionalities. As due to large volume of books and daily based queries system is not comfortable to handle daily transaction properly.

It’s a right time to rebuild the library system because as day by day number of students and staff are increasing in our institute and size of library is also expanding so we need workforce to handle it that’s why there should be a seamless process which can handle all this pressure in proper way. At that point our teacher suggested the creation of relational database system which can manage all these problems

**Requirement Analysis**

Our library database system provides cost effective and efficient procedure to maintain data of library. It needs to store student details because it’s an internal library of college so we don’t need to store any customer’s data. We store staff detail, librarian details and available resources i.e. books, journals, cds, racks etc. detail in our database.

Students need to register in library to get membership which has limit of one year so at time of registration he needs to provide his name, course name, id, DOB and address. A library card can be issued to him after storing details in system.

Books has unique number for their identity and category type, author name, publisher name, copy number etc. and library must have record of which book is issued to which member of library and either its return on time or not and calculation of fine overdue.

When a student borrows a book an entry of his name is entered in database by librarian and return date is automatically comes up in system. It also shows how many books are issued to one member. They can only borrowed maximum of five books at a time. Before borrowing any book members of library can check if the book is available or not.

**Tasks of librarians are following:**

* Add members
* Add library cards and assign them to members
* Handle returns
* Check fines
* Add books/ assets to the database
* Remove assets from the database
* Receive payments from members and update the fines
* View all member’s detail

This report examines that any educational system can use it in their libraries to determine about the author detail, publisher’s detail, availability of books, can track members record etc. and it is very easy to modify by making changes in database. It does not examine online system of library

**Technology Used**

* HeidiSQL 9.5
* Xampp
* Lucidchart
* MS Excel 2010
* MS Word 2010

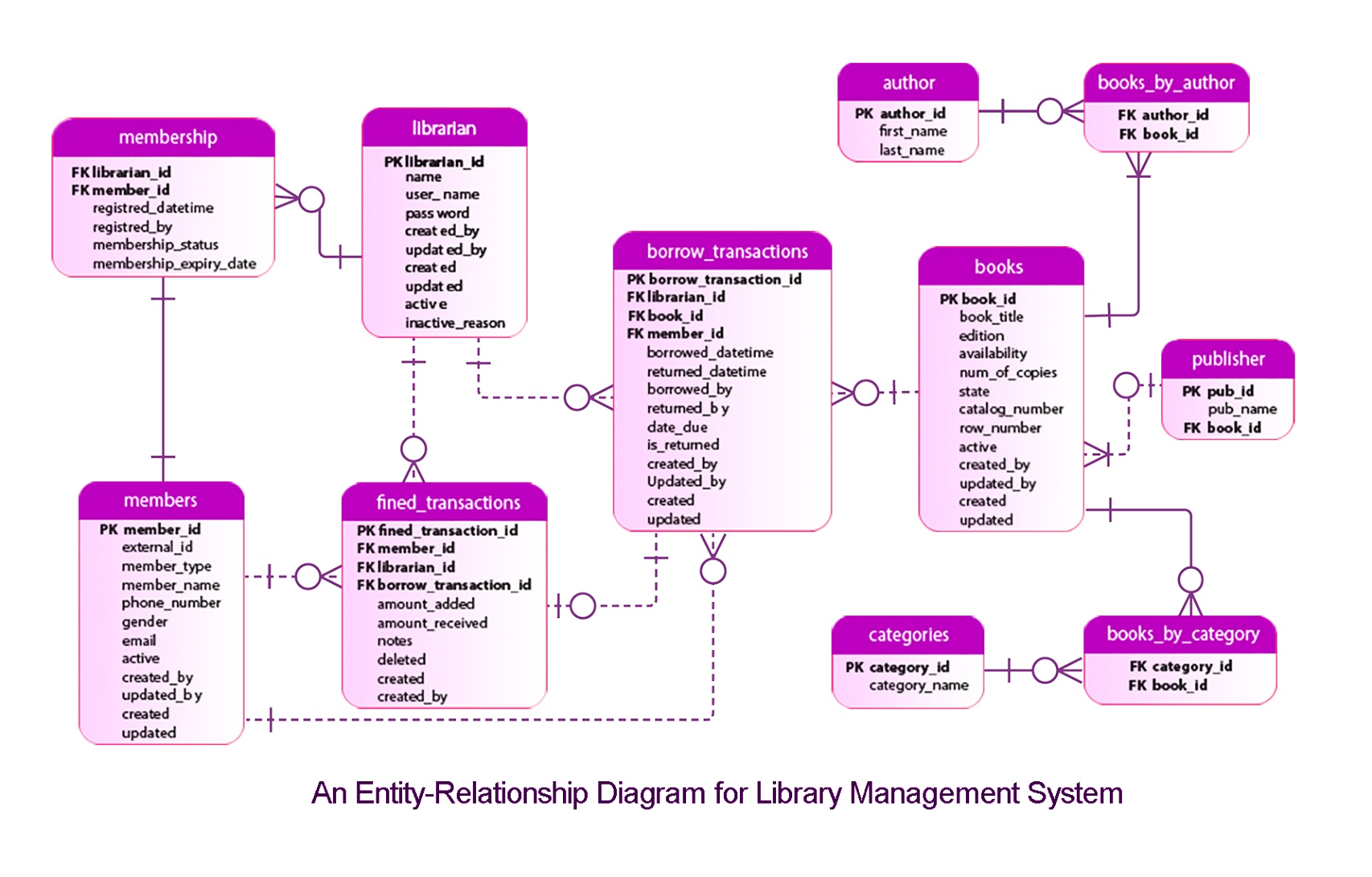
1. **Features**

* Detail of each student/member is stored in database
* Large number of volume of books are easily maintained
* List of students with same category can be viewed
* List of available books can be viewed
* Fine is easily calculated by system automatically
* Librarians can add/modify/delete students and books
* Can see the list of issued books
* List of books not returned on time by entering their id or name of book
* Borrowing of book become easy because in database librarian can find in seconds if this is book is issued to someone else or not
* Librarians has password and login detail of database so it is secure process

1. **E-R diagram**

“E-R Diagram or Entity-relationship model is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases”

It is cleared from above discussion that main entities of Library are, members (student or staff), books and librarians which store data regarding any queries e.g. how many books are borrowed, fine payable etc. In this diagram we try to show possible entities and their relationship with each other and what attributes they hold. Using ER Design redundancy of many items can be reduced and it’s a good practice to show relation among entities.



**An Entity-Relationship Diagram for Library Management System**

**Cardinality of the relationship/Multiplicity is described as:**

An Author can write many books (1:\*)

A book can have many authors (1:\*)

A book can have many categories (1:\*)

A member can be associated with zero or many books borrowed (0:\*)

A borrowed transaction must only be associated with one and only one member (1:1)

Books can have zero or many transactions…… (0: \*)

Librarian handle transactions…….. (1 :\*)

Librarian creates membership……. (1 :\*)

Librarian perform fine transaction… (1 :\*)

Borrow transactions on fines… (1:0)

Members pays fine… ……………… (1:\*)

Members can have only one membership ……….. (1:1)

**Identifying relationship** is when child object is dependent on parent object.

Shown as dotted line: - - - - - - - - -

**Non**-**identifying relationship** is when the primary key attributes of the parent must not become primary key attributes of the child defined as continuous line:

In this diagram we can see that entity borrow\_transaction is dependent on librarian, books and members entities because for every transaction these three requirements are necessary. A librarian can issue books, Librarian provides membership and done transactions on books and can also get fine from members. Because many transactions can be done on many books at same time we associate books with transactions. In this EERD we can see the clear many to many relations of books and author and similarly for books and categories entity. This diagram show dependencies of relation and reduce redundancy problems as well.

1. **SQL Code for Entity Set**

Following are the queries in sql to create tables or entity set:

**Add** **Table** “books”

------------------------

**CREATE** **TABLE** books (

book\_id **INTEGER** **NOT** **NULL**,

book\_title **VARCHAR**(40),

edition **VARCHAR**(40),

availability **BOOL**,

num\_of\_copies **INTEGER**,

state **VARCHAR**(40),

catalog\_number **INTEGER**,

row\_number **INTEGER**,

active **BOOL**,

created\_by **VARCHAR**(40),

updated\_by **VARCHAR**(40),

created **DATETIME**,

updated **DATETIME**,

**PRIMARY** **KEY** (book\_id)

);

**Add** **Table** “borrow\_transaction”

------------------------

**CREATE** **TABLE** borrow\_transaction (

borrow\_transaction\_id **INTEGER** **NOT** **NULL**,

librarian\_id **INTEGER** **NOT** **NULL**,

book\_id **INTEGER** **NOT** **NULL**,

member\_id **INTEGER** **NOT** **NULL**,

borrowed\_datetime **DATETIME** **NOT** **NULL**,

returned\_datetime **DATETIME**,

borrowed\_by **VARCHAR**(40),

returned\_by **VARCHAR**(40),

date\_due **DATE**,

is\_returned **BOOL**,

created\_by **VARCHAR**(40),

updated\_by **VARCHAR**(40),

created **DATETIME**,

updated **DATETIME**,

**PRIMARY** **KEY** (borrow\_transaction\_id)

);

**Add** **Table** “librarian”

------------------------

**CREATE** **TABLE** librarian (

librarian\_id **INTEGER** **NOT** **NULL**,

name **VARCHAR**(40),

user\_name **VARCHAR**(40),

**password** **VARCHAR**(40),

created\_by **VARCHAR**(40),

updated\_by **VARCHAR**(40),

created **DATETIME**,

updated **DATETIME**,

active **BOOL**,

inactive\_reason **VARCHAR**(40),

**PRIMARY** **KEY** (librarian\_id)

);

**Add** **Table** “members”

------------------------

**CREATE** **TABLE** members (

member\_id **INTEGER** **NOT** **NULL**,

external\_id **INTEGER**,

member\_type **VARCHAR**(40),

member\_name **VARCHAR**(40),

phone\_number **INTEGER**,

gender **VARCHAR**(40),

email **VARCHAR**(40),

active **BOOL**,

created\_by **VARCHAR**(40),

updated\_by **VARCHAR**(40),

created **DATETIME**,

updated **DATETIME**,

**PRIMARY** **KEY** (member\_id)

);

**Add** **Table** “membership”

------------------------

**CREATE** **TABLE** membership (

librarian\_id **INTEGER** **NOT** **NULL**,

member\_id **INTEGER** **NOT** **NULL**,

registred\_datetime **DATETIME** **NOT** **NULL**,

registred\_by **VARCHAR**(40),

membership\_status **BOOL**,

membership\_expiry\_date **DATE**

);

**Add** **Table** “fined\_transactions”

------------------------

**CREATE** **TABLE** fined\_transactions (

fined\_transaction\_id **INTEGER** **NOT** **NULL**,

member\_id **INTEGER** **NOT** **NULL**,

librarian\_id **INTEGER** **NOT** **NULL**,

borrow\_transaction\_id **INTEGER** **NOT** **NULL**,

amount\_added **VARCHAR**(40),

amount\_received **VARCHAR**(40),

notes **VARCHAR**(40),

deleted **DATETIME**,

created **DATETIME**,

created\_by **VARCHAR**(40),

**PRIMARY** **KEY** (fined\_transaction\_id)

);

**Add** **Table** “author”

------------------------

**CREATE** **TABLE** author (

author\_id **INTEGER** **NOT** **NULL**,

first\_name **VARCHAR**(40),

last\_name **VARCHAR**(40),

**PRIMARY** **KEY** (author\_id)

);

**Add** **Table** “books\_by\_author”

------------------------

**CREATE** **TABLE** books\_by\_author (

author\_id **INTEGER** **NOT** **NULL**,

book\_id **INTEGER** **not** **null**

);

**Add** **Table** “books\_by\_category”

------------------------

**CREATE** **TABLE** books\_by\_category (

category\_id **INTEGER** **NOT** **NULL**,

book\_id **INTEGER** **not** **null**

);

**Add** **Table** “categories”

------------------------

**CREATE** **TABLE** categories (

category\_id **INTEGER** **NOT** **NULL**,

category\_name **VARCHAr**(40),

**PRIMARY** **KEY** (category\_id )

);

**Add** **Table** “publisher”

------------------------

**CREATE** **TABLE** publisher (

pub\_id **INTEGER** **NOT** **NULL**,

pub\_name **VARCHAR**(40),

book\_id **INTEGER** **NOT** **NULL**,

**PRIMARY** **KEY** (pub\_id )

);

**Foreign** **key** constraints

------------------------------

**ALTER** **TABLE** publisher

**ADD** **FOREIGN** **KEY** (book\_id) **REFERENCES** books(book\_id);

**ALTER** **TABLE** borrow\_transaction

**ADD** **FOREIGN** **KEY** (book\_id) **REFERENCES** books (book\_id);

**ALTER** **TABLE** borrow\_transaction

**ADD** **FOREIGN** **KEY** (librarian\_id) **REFERENCES** librarian (librarian\_id);

**ALTER** **TABLE** borrow\_transaction

**ADD** **FOREIGN** **KEY** (member\_id) **REFERENCES** members (member\_id);

**ALTER** **TABLE** membership

**ADD** **FOREIGN** **KEY** (librarian\_id) **REFERENCES** librarian (librarian\_id);

**ALTER** **TABLE** membership

**ADD** **FOREIGN** **KEY** (member\_id) **REFERENCES** members (member\_id);

**ALTER** **TABLE** fined\_transactions

**ADD** **FOREIGN** **KEY** (librarian\_id) **REFERENCES** librarian (librarian\_id);

**ALTER** **TABLE** fined\_transactions

**ADD** **FOREIGN** **KEY** (borrow\_transaction\_id) **REFERENCES** borrow\_transaction ( borrow\_transaction\_id);

**ALTER** **TABLE** fined\_transactions

**ADD** **FOREIGN** **KEY** (member\_id) **REFERENCES** members (member\_id);

**ALTER** **TABLE** books\_by\_author

**ADD** **FOREIGN** **KEY** (author\_id) **REFERENCES** author(author\_id );

**ALTER** **TABLE** books\_by\_author

**ADD** **FOREIGN** **KEY** (book\_id) **REFERENCES** books(book\_id );

**ALTER** **TABLE** books\_by\_category

**ADD** **FOREIGN** **KEY** (category\_id) **REFERENCES** Categories(category\_id );

**ALTER** **TABLE** books\_by\_category

**ADD** **FOREIGN** **KEY** (book\_id) **REFERENCES** books(book\_id );

1. **Data Insertion into Entity Set**

Following queries are used to insert data in an entity set (shown few queries only):

**INSERT** **INTO** books (book\_id, book\_title, edition, availability, num\_of\_copies,state, catalog\_number, row\_number, active, created\_by, updated\_by, created, updated)

**VALUES** (101,'Numerical Methods For Engineer','4th',1,10,'EXCELLENT',3,4,1,'KAUSHIK','APU','2016-10-12 12:10:03','2017-10-12 12:10:03'),

(102,'Discrete Mathematics And Its Application','7th',1,15,'EXCELLENT',5,6,1,'KAUSHIK','APU','2016-12-10 12:10:03','2018-10-12 12:10:03');

**INSERT** **INTO** librarian (librarian\_id,name,user\_name,**password**,created\_by,updated\_by,created,updated,active,inactive\_reason)

**VALUES** ( 1001,'Kaushik','kaushik','kaushik123','admin','admin','2015-10-12 12:10:03','2017-10-12 12:10:03',1,**NULL**),

( 1002,'nafisa','nafisa','nafisa123','admin','admin','2015-10-12 12:10:03','2017-10-12 12:10:03',1,**NULL**);

**INSERT** **INTO** members (member\_id,external\_id,member\_type,member\_name,phone\_number,gender,email,active,created\_by,updated\_by,created,updated)

**VALUES** (18101,1,'Student','Ahmed Apu',1700000,'Male','ahmed@gmail.com',1,'kaushik',**NULL**,'2018-2-10 12:10:03',**NULL**),

(18102,2,'STAFF','nafisa',012563663,'female','nafisa@gmail.com',1,'nafisa',**NULL**,'2018-2-10 12:10:03',**NULL**);

**INSERT** **INTO** publisher(pub\_id,pub\_name,book\_id)

**VALUES** (10,'McGraw-Hill Education',101),(20,'William C Brown',102);

**INSERT** **INTO** author(author\_id,first\_name,last\_name)

**VALUES** (1001,'Steven C.','Chapra'),(1002,'Kenneth H.','Rosen');

**INSERT** **INTO** books\_by\_author(author\_id,book\_id)

**VALUES** (1001,101),(1002,102);

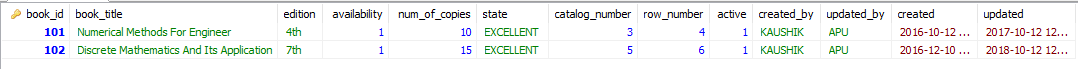
**INSERT** **INTO** categories(category\_id,category\_name)

**VALUES** (4,'Numerical Analysis'),(6,'Computer Mathematics');

**INSERT** **INTO** books\_by\_category(category\_id,book\_id)

**VALUES** (4,101),(6,102);

**Table: books**



**Table: librarian**



**Table: members**



**Table: publisher**



|  |  |
| --- | --- |
| **Table: author** | **Table: books\_by\_author** |
| **Table: categories** | **Table:** **books\_by\_category** |

1. **Query Implementation on Entity Set**

Following is some query implementation for these database entities:

**To find books with overdue date**

**SELECT** members.member\_name, books.book\_title, borrow\_transaction.borrowed\_datetime, borrow\_transaction.date\_due

**FROM** borrow\_transaction

**INNER** **JOIN** members **ON** borrow\_transaction.member\_id=members.member\_id

**INNER** **JOIN** books **ON** borrow\_transaction.book\_id=books.book\_id

**WHERE** date\_due <= '2011-02-07' **AND** is\_returned=0;

**To find publisher name of a book**

**SELECT** books.book\_title, publisher.pub\_name

**FROM** books

**INNER** **JOIN** publisher **on** books.book\_id=publisher.book\_id;

**Some other functions**

**SELECT** member\_type, member\_name **from** members **WHERE** member\_type!='student';

**SELECT** member\_name,member\_type,gender

**FROM** members **ORDER** **BY** member\_name **DESC**;

**SELECT** book\_title

**FROM** books;

**Using triggers**

**CREATE** **TABLE** members\_audit (

serial\_no **INT** **AUTO\_INCREMENT** **PRIMARY** **KEY**,

member\_id **INT** **NOT** **NULL**,

new\_name **VARCHAR**(50) **NOT** **NULL**,

old\_name **VARCHAR**(50) **NOT** **NULL**,

updated\_by **VARCHAR**(50) **NOT** **NULL**,

updated **DATETIME** **DEFAULT** **NULL**,

**action** **VARCHAR**(50) **DEFAULT** **NULL**

);

DELIMITER $$

**CREATE** **TRIGGER** after\_members\_update

**after** **UPDATE** **ON** members

**FOR** **EACH** **ROW**

**BEGIN**

**INSERT** **INTO** members\_audit

**SET** **action** = 'update',

member\_id = **OLD**.member\_id,

new\_name = **new**.member\_name,

old\_name = **old**.member\_name,

updated\_by = **USER**(),

updated = **NOW**();

**END**

$$ DELIMITER ;

**SHOW** **TRIGGERS**;

**UPDATE** members

**SET** member\_name = 'Nafisa Anjum'

**WHERE** member\_id = 18102;

**SELECT** \***FROM** members\_audit;



1. **Limitations**

Though computerized library systems are widely used because this is the world of technology and development so people preferred to use computer based systems rather than manual it has some limitations too. These limitations can be happen not only in these library management systems but also in any kind of software based system too.

* Risk of hardware or software failure
* Data can be crashed
* Training required to understand the system
* It required high conversion cost to transform file-based system into computer based system

1. **Conclusion**

The main conclusion that can be drawn is therefore that using MySQL database system has been designed for school library which can be used to store data regarding books, members of library and can also update, delete and add new records.

This database is designed with that flexibility that it can be implemented in any library in future; it can be modified easily into new technology. We can add as many data as required.