Designing and Implementing a Relational DBMS on

"LIBRARY Management System"

Submitted to the

SAVITRIBAI PHULE PUNE UNIVERSITY

In partial fulfillment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

By

NAME	SEAT NO / ROLL NO
AARTI RATHI	S190228501 / 4201
ISHA TYAGI	S190228516 / 4216
NISHU RAI	S190228530 / 4227
SHAMBHAVI CHOUDHARY	S190228547 / 4246

Under the guidance of

Prof. YUVRAJ GHOLAP

Department of Information Technology

Army Institute of Technology,

Dighi Hills, Pune - 411 015.

2020-2021

CERTIFICATE

This is to certify that the mini project report entitled **"LIBRARY MANAGEMENT SYSTEM"** being submitted by **Aarti Rathi (S190228501 / 4201** is a record of bonafide work carried out by her under the supervision and guidance of **Prof. YUVRAJ GHOLAP** in partial fulfillment of the requirement for S**E (Information Technology Engineering) – 2019 course** of Savitribai Phule Pune University, Pune in the academic year 2020-2021.

Date: 10 June 2021

Place: Pune

Guide Subject Coordinator Head of the Department

Principal

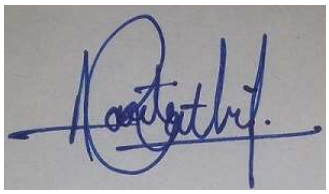
This Mini Project report has been examined by us as per the Savitribai Phule Pune University, Pune requirements at Army Institute of Technology, Pune – 411015 on . . . . . .

Internal Examiner External Examiner

ACKNOWLEDGEMENT

We would like to express our gratitude to Prof. Yuvraj Gopal for guiding and helping us, and making this project complete successfully, we are grateful to him for assigning this mini project activity that helped us understand the concepts of MySQL and build a real-time database that can be used in any of our future projects.
We would like to thank our parents and friends for helping and motivating us to complete the project in stipulated time.

Aarti Rathi, 4201
Nishu Rai,4227
Shambhavi Choudhary,4246
Isha Tyagi, 4216



(Students Name & Signature)
AARTI RATHI, 4201

CONTENTS

1. ABSTRACT

A library is a collection of organized information and resources which is made accessible to a well-defined community for borrowing or reference's sake. The collection of the resources and information are provided in digital or physical format in either a building/room or in a virtual space or even both. Library's resources and collections may include newspapers, books, films, prints maps, CD's, tapes, videotapes, microform, database etc. he Library Management System is an application for assisting a librarian in managing a book library in a university. The system would provide basic set of features to add/update members, add/update books, and manage check in specifications for the systems based on the client's statement of need. Library Management System is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all. This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, organization of an Online Library becomes much simple. The Library Management has been designed to computerize and automate the operations performed over the information about the members, book issues and returns and all other operations. This computerization of library helps in many instances of its maintenances. It reduces the workload of management as most of the manual work done is reduced The main aim of this system is to be develop a new programmed system that will conveying ever lasting solution to the manual base operations and to make available channel through which staff can maintain the record easily and students can access the information about the library at whatever place they might find themselves. Library management system allows the user to store the book details and the customer's details. The system is strong enough to withstand regressive yearly operations under conditions where the database is maintained and cleared over a certain time of span. The implementation of the system in the organization will considerably reduce data entry, time and also provide readily calculated reports. Management software for monitoring and controlling the transactions in a library. The project "Library Management System" which mainly focuses on basic operations in a library like adding new books, and updating new information, searching books and members and return books. This project gives us the complete information about the library. We can enter the record of new books and retrieve the details of books available in the library. We can issue the books to the students and maintain their records and can also check how many books are issued and stock available in the library. In this project we can maintain the late fine of students who returns the issued books after the due date. Throughout the project the focus has been on presenting information and comments in an easy and intelligible manner. The Library Management System is gaining more importance as the number of its users are increasing rapidly. As the number is rising there is a need of effective management of library, one such effective system is our Library Management System. The transactions like login, register, add , search, delete, issue are provided. The Library Management System stores the details like name, address, ID number, Date Of Birth of members working in the library and users who come to library. The details of books like book name, book number, subject to which it belongs , author, edition, year of publication , the total number of books that are present in the library etc are also stored.

2. INTRODUCTION

E. F. Codd introduced the term in his seminal paper "A Relational Model of Data for Large Shared Data Banks", published in 1970. In this paper and later papers he defined what he meant by relational. One well-known definition of what constitutes a relational database system is Codd's 12 rules. However, many of the early implementations of the relational model did not conform to all of Codd's rules, so the term gradually came to describe a broader class of database systems. At a minimum, these systems: • presented the data to the user as relations (a presentation in tabular form, i.e. as a collection of tables with each table consisting of a set of rows and columns, can satisfy this property) • provided relational operators to manipulate the data in tabular form The first systems that were relatively faithful implementations of the relational model were from the University of Michigan; Micro DBMS (1969) and from IBM UK Scientific Centre at Peterlee; IS1 (1970–72) and its follow-on PRTV (1973–79). The first system sold as an RDBMS was Multics Relational Data Store, first sold in 1978. Others have been Berkeley Ingres QUEL and IBM BS12. The most popular definition of an RDBMS is a product that presents a view of data as a collection of rows and columns, even if it is not based strictly upon relational theory. By this definition, RDBMS products typically implement some but not all of Codd's 12 rules.[1] A second, theory-based school of thought argues that if a database does not implement all of Codd's rules (or the current understanding on the relational model, as expressed by Christopher J Date, Hugh Darwen and others), it is not relational. This view, shared by many theorists and other strict adherents to Codd's principles, would disqualify most DBMSs as not relational. For clarification, they often refer to some RDBMSs as Truly-Relational Database Management Systems (TRDBMS), naming others Pseudo-Relational Database Management Systems (PRDBMS). As of now, all commercial relational DBMS employ SQL as their query language. Alternative query languages have been proposed and implemented, notably the pre-1996 implementation of Berkeley Ingres QUEL. With standardization of the SQL, both commercial and open source DBMS have adopted some degree of standards compliance.

History of SQL (A Research Project Conducted by IBM) -

The origins of the SQL language date back to a research project conducted by IBM at their research laboratories in San Jose, California in the early 1970s. The aim of the project was to develop an experimental RDBMS which would eventually lead to a marketable product. At that time, there was a lot of interest in the relational model for databases at the academic level, in conferences and seminars. IBM, which already had a large share of the commercial database market with hierarchical and network model DBMSs, realised quite quickly that the relational model would figure prominently in future database products. The project at IBM's San Jose labs was started in 1974 and was named System R. A language called Sequel (for Structured English Query Language) was chosen as the relational database language for System R. In the project, Sequel was abbreviated to SQL. This is the reason why SQL is still generally pronounced as see-quel. In the first phase of the System R project, researchers concentrated on developing a basic version of the RDBMS. The main aim at this stage was to verify that the theories of the relational model could be translated into a working, commercially viable product. This first phase was successfully completed by the end of 1975, and resulted in a rudimentary, single-user DBMS based on the relational model. The subsequent phases of System R concentrated on further developing the DBMS from the first phase. Additional features were added, multi-user capability was implemented, and by 1978, a completed RDBMS was ready for user evaluation. The

System R project was finally completed in 1979. During this time, the SQL language was modified and added to as the needs of the System R DBMS dictated. During the development of System R and SQL/DS, other companies were also at work creating their own relational database management systems. Some of them, Oracle being a prime example, even implemented SQL as the relational database language for their DBMSs concurrently with IBM. Today, the SQL language has gained ANSI (American National Standards Institute) and ISO (International Standards Organization) certification. A version of SQL is available for almost any hardware platform from CRAY supercomputers to IBM PC microcomputers. In recent years, there has been a marked trend for software manufacturers to move away from proprietary database languages and settle on the SQL standard. The microcomputer platform especially has seen a proliferation of previously proprietary packages that have implemented SQL functionality. Even spreadsheet and word processing packages have added options which allow data to be sent to and retrieved from SQL based databases via a Local Area or a Wide Area network connection.[3]

2.1 Problem Statement.

Create an online library management system of a particular institution to make the books and magazines available to the teachers and students remotely. Library Management System is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all. This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. Owing to the advancement of technology, Library Management System makes the task relatively easy.

2.2 Motivation

- **Data redundancy and inconsistency –**
  Redundancy is the concept of repetition of data, the file system cannot control redundancy of data as each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining same files data for different applications. Hence changes made by one user does not reflect in files used by second users, which leads to inconsistency of data. Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users. As there is no or less redundancy, data remains consistent.
- **Data sharing –**
  File system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to centralized system.
- **Data concurrency –**
  Concurrent access to data means more than one user is accessing the same data at the

same time. Anomalies occur when changes made by one user gets lost because of changes made by other user. File system does not provide any procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.

- **Data searching –**
  For every search operation performed on file system, a different application program has to be written. While DBMS provides inbuilt searching operations. User only have to write a small query to retrieve data from database.
- **Data integrity –**
  There may be cases when some constraints need to be applied on the data before inserting it in database. The file system does not provide any procedure to check these constraints automatically. Whereas DBMS maintains data integrity by enforcing user defined constraints on data by itself.
- **System crashing –**
  In some cases, systems might have crashes due to various reasons. It is a bane in case of file systems because once the system crashes, there will be no recovery of the data that's been lost. A DBMS will have the recovery manager which retrieves the data making it another advantage over file systems.
- **Data security –**
  A file system provides a password mechanism to protect the database but how longer can the password be protected? No one can guarantee that. This doesn't happen in the case of DBMS. DBMS has specialized features that help provide shielding to its data.

2.3 . Objectives

The library management system keeps track of all the information about the books in the library, their cost, status and total number of books available in the library. Library Management System is a system which maintains the information about the books present in the library, their authors, the members of library to whom books are issued, library staff and all. This is very difficult to organize manually. Maintenance of all this information manually is a very complex task. The user will find it easy in this automated system rather than using the manual writing system and the workload will also be reduced To eliminate the paper –work in library and to record every transaction in computerized system so that problem such as record file missing won't happen again.

3. Data Types

MySQL uses many different data types broken into three categories: numeric, date and time, and string types.

Numeric Data Types:

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you. The following list shows the common numeric data types and their descriptions:

- **INT** - A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** - A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** - A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** - A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** - A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** - A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types:

The MySQL date and time datatypes are:

- **DATE** - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** - A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- **TIME** - Stores the time in HH:MM:SS format.
- **YEAR(M)** - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

String Types:

Although numeric and date types are fun, most data you'll store will be in string format. This list describes the common string datatypes in MySQL.

- **CHAR(M)** - A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** - A variable-length string between 1 and 255 characters in length; for example VARCHAR(25). You must define a length when creating a VARCHAR field.
- **BLOB or TEXT** - A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.
- **TINYBLOB or TINYTEXT** - A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.

- **LONGBLOB or LONGTEXT** - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

4. Data Modeling using ER Model

4.1 REQUIREMENTS COLLECTION AND ANALYSIS

ENTITES:

- **BOOKS**
- **EMPLOYEE**
- **STUDENT**
- **ISSUE STATUS**
- **RETURN STATUS**

ATTRIBUTES:

- STUDENT
  - student_id
  - Dept
  - student_Name
  - student_Email
  - student_address
  - Regd_date
- BOOKS
  - Isbn
  - Title
  - Category
  - Price
  - Status
  - Author

- Publisher
- Updation_Date

❖ EMPLOYEE
- Emp_ID
- E_name
- E_Email
- Position

❖ ISSUE STATUS
- Issue_ID
- StudentID
- Issued_book
- Issue_date
- Isbn_book

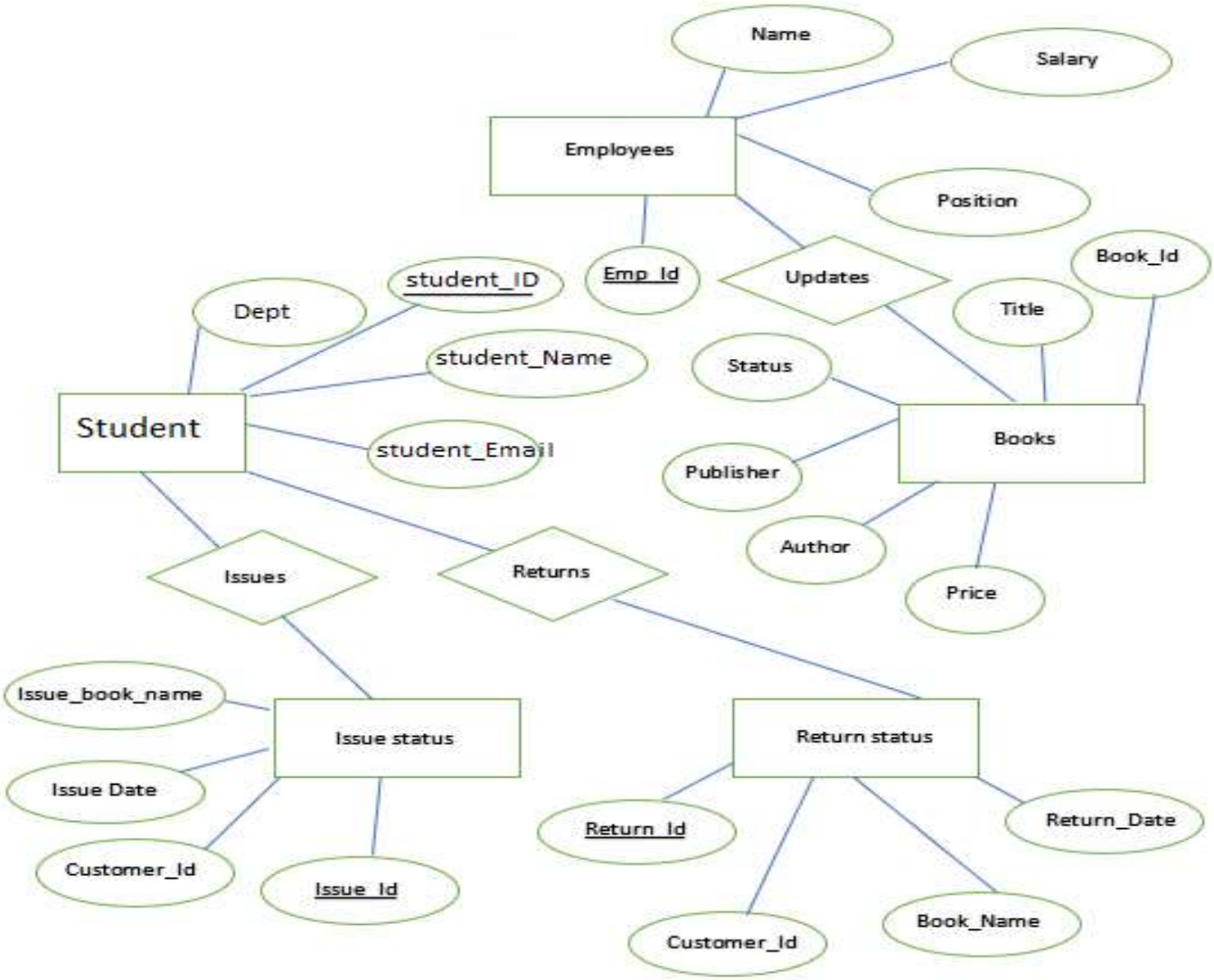
❖ RETURN STATUS
- Return_ID
- ReturnID_student
- Return_book
- Return_date
- Isbn_book2

4.1.1 Entity Types, Entity Sets, Attributes, and Keys

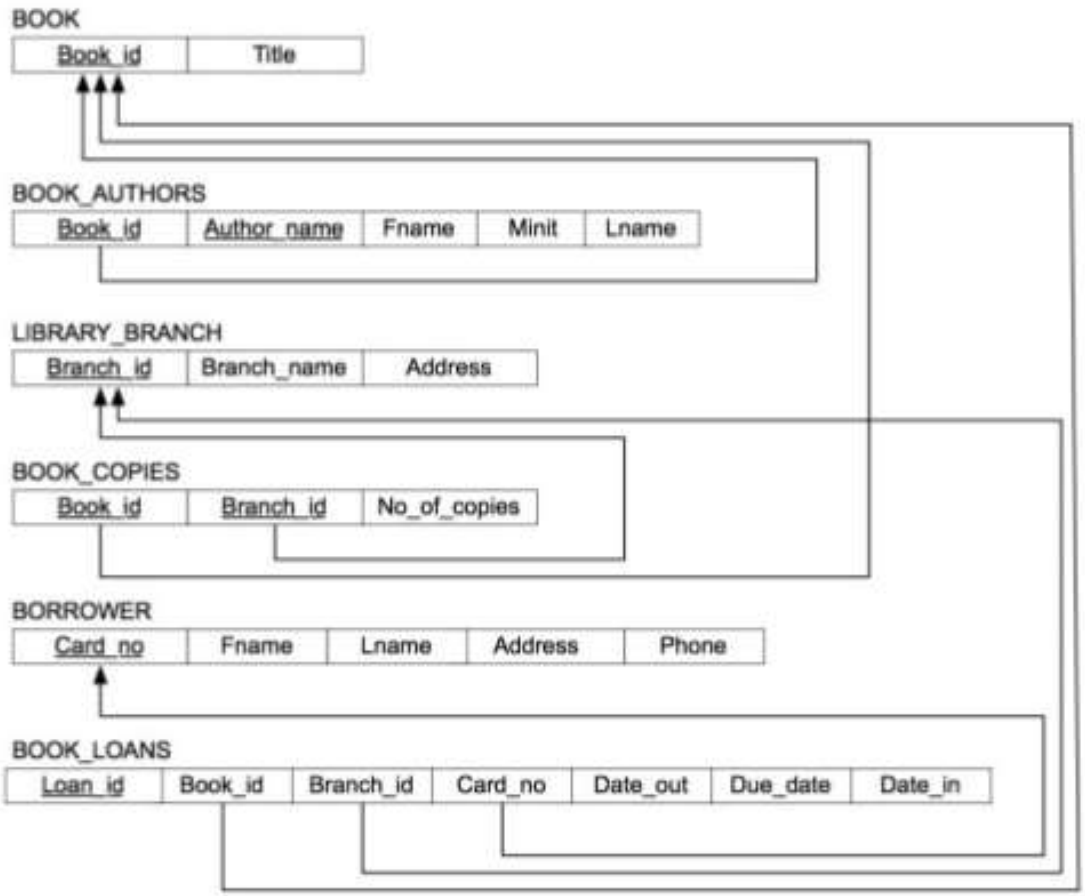
1. An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multivalued attribute. We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.
2. An entity type PROJECT with attributes Name, Number, Location, and Controlling_department. Both Name and Number are (separate) key attributes.
3. An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements. We must go back to the users to see if any of them will refer to the individual components of Name—First_name, Middle_initial, Last_name—or of Address.
4. An entity type DEPENDENT with attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).

4.2 E-R Diagram

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database. Entity–relationship modeling was developed for database design by Peter Chen and published in a 1976 paper. However, variants of the idea existed previously, some ER modelers show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific ontology. An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering.



4.2.2.Relational Database Design Using ER-to-Relational Mapping



ER-to-Relational Mapping Algorithm

The relational model constraints, which include primary keys, unique keys (if any), and referential integrity constraints on the relations, will also be specified in the mapping results.

Step 1: Mapping of Regular Entity Types. For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R. If multiple keys were identified for E during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation R. Knowledge about keys is also kept for indexing purposes and other types of analyses. In our example, we create the relations EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.2 to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.1. The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps. These include the attributes Super_ssn and Dno of EMPLOYEE, Mgr_ssn and Mgr_start_date of DEPARTMENT, and Dnum of PROJECT. In our example, we choose Ssn, Dnumber, and Pnumber as

primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively. Knowledge that Dname of DEPARTMENT and Pname of PROJECT are secondary keys is kept for possible use later in the design. The relations that are created from the mapping of entity types are sometimes called entity relations because each tuple represents an entity instance. The result after this mapping step is shown in Figure 9.3(a).

Step 2: Mapping of Weak Entity Types. For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any. If there is a weak entity type E2 whose owner is also a weak entity type E1, then E1 should be mapped before E2 to determine its primary key first. In our example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT (see Figure 9.3(b)). We include the primary key Ssn of the EMPLOYEE relation—which corresponds to the owner entity type—as a foreign key attribute of DEPENDENT; we rename it Essn, although this is not necessary.
The primary key of the DEPENDENT relation is the combination {Essn, Dependent_name}, because Dependent_name (also renamed from Name in Figure 9.1) is the partial key of DEPENDENT.
Step 3: Mapping of Binary 1:1 Relationship Types. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. There are three possible approaches: (1) the foreign key approach, (2) the merged relationship approach, and (3) the cross reference or relationship relation approach. The first approach is the most useful and should be followed unless special conditions exist, as we discuss below.

1. Foreign key approach: Choose one of the relations—S, say—and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.

In our project, we map the 1:1 relationship type MANAGES from Figure 9.1 by choosing the participating entity type DEPARTMENT to serve in the role of S because its participation in the MANAGES relationship type is total (every department has a manager). We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it Mgr_ssn. We also include the simple attribute Start_date of the MANAGES relationship type in the DEPARTMENT relation and rename it Mgr_start_date (see Figure 9.2). Note that it is possible to include the primary key of S

as a foreign key in T instead. In our example, this amounts to having a foreign key attribute, say Department_managed in the EMPLOYEE relation, but it will have a NULL value for employee tuples who do not manage a department. If only 2 percent of employees manage a department, then 98 percent of the foreign keys would be NULL in this case. Another possibility is to have foreign keys in both relations S and T redundantly, but this creates redundancy and incurs a penalty for consistency maintenance

Step 4: Mapping of Binary 1:N Relationship Types.

For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; we do this because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S. In our example, we now map the 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION from Figure 9.1. For WORKS_FOR we include the primary key Dnumber of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it Dno. For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself—because the relationship is recursive—and call it Super_ssn. The CONTROLS relationship is mapped to the foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT relation. These foreign keys are shown in Figure 9.2. An alternative approach is to use the relationship relation (cross-reference) option as in the third option for binary 1:1 relationships. We create a separate relation R whose attributes are the primary keys of S and T, which will also be foreign keys to S and T. The primary key of R is the same as the primary key of S. This option can be used if few tuples in S participate in the relationship to avoid excessive NULL values in the foreign key

Step 5: Mapping of Binary M:N Relationship Types. For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S. Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1:N relationship types) because of the M:N cardinality ratio; we must create a separate relationship relation S. In our project, we map the M:N relationship type WORKS_ON from Figure 9.1 by creating the relation WORKS_ON in Figure 9.2. We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS_ON and rename them Pno and Essn, respectively. We also include an attribute Hours in WORKS_ON to represent the Hours

attribute of the relationship type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {Essn, Pno}. This relationship relation is shown in Figure 9.3(c).

Step 6: Mapping of Multivalued Attributes. For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as a multivalued attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components. In our example, we create a relation DEPT_LOCATIONS (see Figure 9.3(d)). The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation. The primary key of DEPT_LOCATIONS is the combination of {Dnumber, Dlocation}. A separate tuple will exist in DEPT_LOCATIONS for each location that a department has.

RELATIONSHIPS-CARDINALITY

- **Student issues BOOKS**
- **Student returns BOOKS**
- **Employee updates BOOKS**
- **Student registers the updates**

5.Creating database using MySQL

An SQL schema is identified by a schema name, and includes an authorization identifier to indicate the user or account who owns the schema, as well as descriptors for each element in the schema. Schema elements include tables, constraints, views, domains, and other constructs (such as authorization grants) that describe the schema. A schema is created via the CREATE SCHEMA statement, which can include all the schema elements' definitions. Alternatively, the schema can be assigned a name and authorization identifier, and the elements can be defined later. The CREATE TABLE command is used to specify a new relation by giving it a name and specifying its attributes and initial constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain of values, and any attribute constraints, such as NOT NULL. The key, entity integrity, and referential integrity constraints can be specified within the CREATE TABLE statement after the attributes are declared, or they can be added later using the ALTER TABLE command:

```
mysql> create database mydb;

mysql> use mydb;
Database changed

mysql> create table BOOKS(isbn int(20) not null, Title varchar(25) not
null, Category varchar(50) not null, Price int (8) not null, Status
varchar(20), Author varchar(20) not null, Publisher varchar(50) not null,
Updatation_Date timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE
CURRENT_TIMESTAMP, primary key(isbn));
Query OK, 0 rows affected, 2 warnings (1.52 sec)

mysql> create table Employee(Emp_ID int (5) not null, E_name varchar(50)
not null, E_Email varchar(25) not null, Position varchar(10) not null,
primary key(Emp_ID));
Query OK, 0 rows affected, 1 warning (0.61 sec)

mysql> create table Student(student_ID int (5) not null, Dept varchar(10)
not null, student_Name varchar(20), student_Email varchar(25) not null,
student_Address varchar(50) not null, Regd_date timestamp not null DEFAULT
CURRENT_TIMESTAMP, primary key(student_ID));
Query OK, 0 rows affected, 1 warning (2.19 sec)

mysql> create table Issue_Status(Issue_ID int(5) not null, StudentID int(5)
not null, Issued_book varchar(25) not null, Issue_date timestamp not null
DEFAULT CURRENT_TIMESTAMP,isbn_book int(20) not null, primary
```

```
key(Issue_ID), constraint foreign key(isbn_book) references BOOKS(isbn),
constraint foreign key(StudentID) references Student(student_ID));
Query OK, 0 rows affected, 3 warnings (1.31 sec)

mysql> create table Return_Status(Return_ID int(5) not null,
Return_StudentID int(5) not null, Return_book varchar(25) not null,
Return_date timestamp not null DEFAULT CURRENT_TIMESTAMP, isbn_book2
int(20) not null, primary key(Return_ID), constraint foreign
key(isbn_book2) references BOOKS(isbn), constraint foreign
key(Return_StudentID) references Issue_Status(StudentID));
Query OK, 0 rows affected, 3 warnings (0.94 sec)
```

```
mysql> show tables;
+----------------+
| Tables_in_mydb |
+----------------+
| books          |
| employee       |
| issue_status   |
| return_status  |
| student        |
+----------------+
5 rows in set (0.17 sec)
```

```
mysql> describe books;
+-----------------+-------------+------+-----+---------------------+-----------------------------+
| Field           | Type        | Null | Key | Default             | Extra                       |
+-----------------+-------------+------+-----+---------------------+-----------------------------+
| isbn            | int         | NO   | PRI | NULL                |                             |
| Title           | varchar(25) | NO   |     | NULL                |                             |
| Category        | varchar(50) | NO   |     | NULL                |                             |
| Price           | int         | NO   |     | NULL                |                             |
| Status          | varchar(20) | YES  |     | NULL                |                             |
| Author          | varchar(20) | NO   |     | NULL                |                             |
| Publisher       | varchar(50) | NO   |     | NULL                |                             |
| Updatation_Date | timestamp   | NO   |     | 0000-00-00 00:00:00 | on update CURRENT_TIMESTAMP |
+-----------------+-------------+------+-----+---------------------+-----------------------------+
8 rows in set (0.20 sec)
```

```
mysql> describe employee;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Emp_ID   | int         | NO   | PRI | NULL    |       |
| E_name   | varchar(50) | NO   |     | NULL    |       |
| E_Email  | varchar(25) | NO   |     | NULL    |       |
| Position | varchar(10) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.11 sec)
```

```
mysql> describe issue_status;
+-------------+-------------+------+-----+-------------------+-------------------+
| Field       | Type        | Null | Key | Default           | Extra             |
+-------------+-------------+------+-----+-------------------+-------------------+
| Issue_ID    | int         | NO   | PRI | NULL              |                   |
| StudentID   | int         | NO   | MUL | NULL              |                   |
| Issued_book | varchar(25) | NO   |     | NULL              |                   |
| Issue_date  | timestamp   | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| isbn_book   | int         | NO   | MUL | NULL              |                   |
+-------------+-------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)

mysql> describe return_status;
+------------------+-------------+------+-----+-------------------+-------------------+
| Field            | Type        | Null | Key | Default           | Extra             |
+------------------+-------------+------+-----+-------------------+-------------------+
| Return_ID        | int         | NO   | PRI | NULL              |                   |
| Return_StudentID | int         | NO   | MUL | NULL              |                   |
| Return_book      | varchar(25) | NO   |     | NULL              |                   |
| Return_date      | timestamp   | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| isbn_book2       | int         | NO   | MUL | NULL              |                   |
+------------------+-------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)
```

```
mysql> describe student;
+-----------------+-------------+------+-----+-------------------+-------------------+
| Field           | Type        | Null | Key | Default           | Extra             |
+-----------------+-------------+------+-----+-------------------+-------------------+
| student_ID      | int         | NO   | PRI | NULL              |                   |
| Dept            | varchar(10) | NO   |     | NULL              |                   |
| student_Name    | varchar(20) | YES  |     | NULL              |                   |
| student_Email   | varchar(25) | NO   |     | NULL              |                   |
| student_Address | varchar(50) | NO   |     | NULL              |                   |
| Regd_date       | timestamp   | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------------+-------------+------+-----+-------------------+-------------------+
6 rows in set (0.00 sec)
```

```
mysql> INSERT INTO books
    -> VALUES (1001, "Half Girlfriend","Romantic",500,"Available","Chetan
Bhagatt","Rupa Publications",'2017-07-08 12:49:09'), (1002, "PHP And MySql
programming","Management",2000,"Available","Mike McGrath","Mike Murach &
Associates Inc.",'2018-08-18 15:40:31'), (1003, "HC
Verma","Science",700,"Not-Available","HC Verma","BHARATI BHAWAN Publishers
& Distributors",'2016-07-08 11:30:29'), (1004, "RD
SHARMA","Mathematics",1500,"Available","RD SHARMA","Dhanpat Rai
Publications",'2019-10-17 10:09:55');
Query OK, 4 rows affected (0.49 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> select * from books;
+------+--------------------------+------------+-------+-------------+---------------+-----------------------------------------+---------------------+
| isbn | Title                    | Category   | Price | Status      | Author        | Publisher                               | Updatation_Date     |
+------+--------------------------+------------+-------+-------------+---------------+-----------------------------------------+---------------------+
| 1001 | Half Girlfriend          | Romantic   |   500 | Available   | Chetan Bhagatt | Rupa Publications                       | 2017-07-08 12:49:09 |
| 1002 | PHP And MySql programming | Management |  2000 | Available   | Mike McGrath  | Mike Murach & Associates Inc.           | 2018-08-18 15:40:31 |
| 1003 | HC Verma                 | Science    |   700 | Not-Available | HC Verma      | BHARATI BHAWAN Publishers & Distributors | 2016-07-08 11:30:29 |
| 1004 | RD SHARMA                | Mathematics |  1500 | Available   | RD SHARMA     | Dhanpat Rai Publications                | 2019-10-17 10:09:55 |
+------+--------------------------+------------+-------+-------------+---------------+-----------------------------------------+---------------------+
4 rows in set (0.00 sec)
```

```
mysql> INSERT INTO employee
    -> VALUES (1202, "Rakesh Kumar","rk141@gmail.com","Admin"), (1205,
"Rupali Singh","rupali@gmail.com","Assistant"), (1211, "Ranbir
Rathore","Rathore_ranbir@gmail.com","Manager");
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> select * from employee;
+--------+----------------+---------------------------+-----------+
| Emp_ID | E_name         | E_Email                   | Position  |
+--------+----------------+---------------------------+-----------+
|   1202 | Rakesh Kumar   | rk141@gmail.com           | Admin     |
|   1205 | Rupali Singh   | rupali@gmail.com          | Assistant |
|   1211 | Ranbir Rathore | Rathore_ranbir@gmail.com  | Manager   |
+--------+----------------+---------------------------+-----------+
3 rows in set (0.00 sec)
```

```
mysql> INSERT INTO student VALUES (2101, "ENTC","Aarti
Rathi","aartirathi@gmail.com","Pune",'2019-07-08 11:30:29'), (2106,
"COMP","Riya Singh","riya@gmail.com","Jaipur",'2019-10-18 16:02:22'),
(2108, "ENTC","Aditya Kumar","adiKumar@gmail.com","Bhopal",'2018-04-10
12:03:51'),(2111, "IT","Isha tyagi","isha_here@gmail.com","Ambala",'2018-
03-16 13:40:21'), (2104, "MECH","Nishu
Rai","RaiNishu@gmail.com","Bangalore",'2019-02-21 17:23:21'),(2118,
"IT","Shambhavi","shambhavi@gmail.com","New Delhi",'2019-07-31 13:58:29');
Query OK, 6 rows affected (0.38 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
mysql> select * from student;
+------------+------+-------------+---------------------+----------------+---------------------+
| student_ID | Dept | student_Name | student_Email       | student_Address | Regd_date           |
+------------+------+-------------+---------------------+----------------+---------------------+
|       2101 | ENTC | Aarti Rathi  | aartirathi@gmail.com | Pune           | 2019-07-08 11:30:29 |
|       2104 | MECH | Nishu Rai    | RaiNishu@gmail.com   | Bangalore      | 2019-02-21 17:23:21 |
|       2106 | COMP | Riya Singh   | riya@gmail.com       | Jaipur         | 2019-10-18 16:02:22 |
|       2108 | ENTC | Aditya Kumar | adiKumar@gmail.com   | Bhopal         | 2018-04-10 12:03:51 |
|       2111 | IT   | Isha tyagi   | isha_here@gmail.com  | Ambala         | 2018-03-16 13:40:21 |
|       2118 | IT   | Shambhavi    | shambhavi@gmail.com  | New Delhi      | 2019-07-31 13:58:29 |
+------------+------+-------------+---------------------+----------------+---------------------+
6 rows in set (0.00 sec)
```

```
mysql> INSERT INTO issue_status VALUES (1314, 2108,"HC Verma",'2020-03-28
10:30:29',1003), (1374, 2101,"RD SHARMA",'2021-11-18 15:12:25',1004),(1302,
2111,"Half Girlfriend",'2020-05-14 11:44:05',1001);
Query OK, 3 rows affected (0.44 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> select * from issue_status;
+----------+-----------+-----------------+---------------------+-----------+
| Issue_ID | StudentID | Issued_book     | Issue_date          | isbn_book |
+----------+-----------+-----------------+---------------------+-----------+
|     1302 |      2111 | Half Girlfriend | 2020-05-14 11:44:05 |      1001 |
|     1314 |      2108 | HC Verma        | 2020-03-28 10:30:29 |      1003 |
|     1374 |      2101 | RD SHARMA       | 2021-11-18 15:12:25 |      1004 |
+----------+-----------+-----------------+---------------------+-----------+
3 rows in set (0.00 sec)
```

```
mysql> INSERT INTO return_status VALUES (1412, 2108,"HC Verma",'2020-07-15
12:05:20',1003),(1421, 2111,"Half Girlfriend",'2020-10-16 15:31:15',1001);
Query OK, 2 rows affected (0.28 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

```
mysql> select * from return_status;
+-----------+------------------+-----------------+---------------------+------------+
| Return_ID | Return_StudentID | Return_book     | Return_date         | isbn_book2 |
+-----------+------------------+-----------------+---------------------+------------+
|      1412 |             2108 | HC Verma        | 2020-07-15 12:05:20 |       1003 |
|      1421 |             2111 | Half Girlfriend | 2020-10-16 15:31:15 |       1001 |
+-----------+------------------+-----------------+---------------------+------------+
2 rows in set (0.00 sec)
```

```
mysql> INSERT INTO return_status VALUES (1416, 2104,"HC Verma",'2020-09-15 12:05:20',1003);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`mydb`.`return_status`, CONSTRAINT `retu
rn_status_ibfk_2` FOREIGN KEY (`Return_StudentID`) REFERENCES `issue_status` (`StudentID`))
```

Since student_ID 2104 isn't taken any book, we cannot blindly give the input!!

6. Test Queries(Minimum 25 Queries)

1. Display the student name who issued the book “mathematics” type.

```
mysql> SELECT B.Title, B.Author, B.Category, A.student_ID, A.student_Name,C.Issue_ID
    -> FROM books B, student A, issue_status C
    -> where B.isbn=C.isbn_book
    -> And A.student_ID=C.studentID
    -> AND B.Category="Mathematics";
+-----------+-----------+-------------+------------+--------------+----------+
| Title     | Author    | Category    | student_ID | student_Name | Issue_ID |
+-----------+-----------+-------------+------------+--------------+----------+
| RD SHARMA | RD SHARMA | Mathematics |       2101 | Aarti Rathi  |     1374 |
+-----------+-----------+-------------+------------+--------------+----------+
1 row in set (0.00 sec)
```

2. Display the Issue ID, Student Name whose issued the book with isbn book number as 1003.

```
mysql> SELECT C.Issue_ID,A.student_Name
    -> FROM student A, issue_status C
    -> where A.student_ID=C.StudentID
    -> and C.isbn_book=1003;
+----------+--------------+
| Issue_ID | student_Name |
+----------+--------------+
|     1314 | Aditya Kumar |
+----------+--------------+
1 row in set (0.02 sec)
```

3. Display the name, student id and date of return of student who have returned the issued book.

```
mysql> SELECT A.student_Name, A.student_ID, C.Return_date
    -> FROM student A, return_status C
    -> where A.student_ID=C.Return_StudentID;
+--------------+------------+---------------------+
| student_Name | student_ID | Return_date         |
+--------------+------------+---------------------+
| Aditya Kumar |       2108 | 2020-07-15 12:05:20 |
| Isha tyagi   |       2111 | 2020-10-16 15:31:15 |
+--------------+------------+---------------------+
2 rows in set (0.00 sec)
```

4. Add a column called book rating in customer table.

```
mysql> Alter table books add B_ratings varchar(10) not null;
Query OK, 0 rows affected (3.25 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc books;
+----------------+------------+------+-----+---------------------+-----------------------------+
| Field          | Type       | Null | Key | Default             | Extra                       |
+----------------+------------+------+-----+---------------------+-----------------------------+
| isbn           | int        | NO   | PRI | NULL                |                             |
| Title          | varchar(25)| NO   |     | NULL                |                             |
| Category       | varchar(50)| NO   |     | NULL                |                             |
| Price          | int        | NO   |     | NULL                |                             |
| Status         | varchar(20)| YES  |     | NULL                |                             |
| Author         | varchar(20)| NO   |     | NULL                |                             |
| Publisher      | varchar(50)| NO   |     | NULL                |                             |
| Updatation_Date| timestamp  | NO   |     | 0000-00-00 00:00:00 | on update CURRENT_TIMESTAMP |
| B_ratings      | varchar(10)| NO   |     | NULL                |                             |
+----------------+------------+------+-----+---------------------+-----------------------------+
9 rows in set (0.25 sec)
```

5. Display the Name of books with book id which were issued after April 2020.

```
mysql> SELECT Issued_book, isbn_book
    -> FROM issue_status
    -> where Issue_date > "2020-04-30 00:00:00";
+-----------------+-----------+
| Issued_book     | isbn_book |
+-----------------+-----------+
| Half Girlfriend |      1001 |
| RD SHARMA       |      1004 |
+-----------------+-----------+
2 rows in set (0.01 sec)
```

6. Insert new issue entry at present day and time.

```
mysql> INSERT INTO issue_status (Issue_ID, StudentID,Issued_book,isbn_book) VALUES (1321, 2106,"HC Verma",1003);
Query OK, 1 row affected (0.55 sec)

mysql> select * from issue_status;
+----------+-----------+----------------+---------------------+-----------+
| Issue_ID | StudentID | Issued_book    | Issue_date          | isbn_book |
+----------+-----------+----------------+---------------------+-----------+
|     1302 |      2111 | Half Girlfriend | 2020-05-14 11:44:05 |      1001 |
|     1314 |      2108 | HC Verma       | 2020-03-28 10:30:29 |      1003 |
|     1321 |      2106 | HC Verma       | 2021-06-11 17:08:54 |      1003 |
|     1374 |      2101 | RD SHARMA      | 2021-11-18 15:12:25 |      1004 |
+----------+-----------+----------------+---------------------+-----------+
4 rows in set (0.00 sec)
```

7. Try to insert value of a student in return table who haven't issued the book yet.

```
mysql> INSERT INTO return_status VALUES (1416, 2104,"HC Verma",'2020-09-15 12:05:20',1003);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
(`mydb`.`return_status`, CONSTRAINT `return_status_ibfk_2` FOREIGN KEY (`Return_StudentID`)
REFERENCES `issue_status` (`StudentID`))
```

Since student_ID 2104 isn't taken any book, we cannot blindly give the input!!

8. Display the Issue ID, Student Name who's issued the book with isbn book number as 1004.

```
mysql> SELECT C.Issue_ID , A.student_Name
    -> FROM student A , issue_status C
    -> where A.student_ID = C.StudentID
    -> and C.isbn_book = 1004;.
+----------+--------------+
| Issue_ID | student_Name |
+----------+--------------+
|     1374 | Aarti Rathi  |
+----------+--------------+
1 row in set (0.18 sec)
```

9. Add a column called edition in customer table.

```
mysql> Alter table books add Edition varchar(10) not null;
Query OK, 0 rows affected (0.70 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc books ;
+-----------------+-------------+------+-----+---------------------+-----------------------------+
| Field           | Type        | Null | Key | Default             | Extra                       |
+-----------------+-------------+------+-----+---------------------+-----------------------------+
| isbn            | int(20)     | NO   | PRI | NULL                |                             |
| Title           | varchar(25) | NO   |     | NULL                |                             |
| Category        | varchar(50) | NO   |     | NULL                |                             |
| Price           | int(8)      | NO   |     | NULL                |                             |
| Status          | varchar(20) | YES  |     | NULL                |                             |
| Author          | varchar(20) | NO   |     | NULL                |                             |
| Publisher       | varchar(50) | NO   |     | NULL                |                             |
| Updatation_Date | timestamp   | NO   |     | 0000-00-00 00:00:00 | on update CURRENT_TIMESTAMP |
| Edition         | varchar(10) | NO   |     | NULL                |                             |
+-----------------+-------------+------+-----+---------------------+-----------------------------+
9 rows in set (0.19 sec)
```

10. Display name and department in ascending order of registration date

```
mysql> SELECT student_Name , Dept
    -> FROM student ORDER BY Regd_date ASC ;
+--------------+------+
| student_Name | Dept |
+--------------+------+
| Isha tyagi   | IT   |
| Aditya Kumar | ENTC |
| Nishu Rai    | MECH |
| Aarti Rathi  | ENTC |
| Shambhavi    | IT   |
| Riya Singh   | COMP |
+--------------+------+
6 rows in set (0.00 sec)
```

11. Insert new issue entry at present day and time.

```
mysql> INSERT INTO issue_status(Issue_ID , StudentID , Issued_Book , isbn_book)
    -> VALUES (1321 , 2106 , "HC Verma" , 1003);
Query OK, 1 row affected (0.07 sec)

mysql> select * from issue_status ;
+----------+-----------+----------------+---------------------+-----------+
| Issue_ID | StudentID | Issued_book    | Issue_date          | isbn_book |
+----------+-----------+----------------+---------------------+-----------+
|     1302 |      2111 | Half Girlfriend | 2020-05-14 11:44:05 |      1001 |
|     1314 |      2108 | HC Verma       | 2020-03-28 10:30:29 |      1003 |
|     1321 |      2106 | HC Verma       | 2021-06-11 18:27:06 |      1003 |
|     1374 |      2101 | RD SHARMA      | 2021-11-18 15:12:25 |      1004 |
+----------+-----------+----------------+---------------------+-----------+
4 rows in set (0.00 sec)
```

12. Display books with price greater than Rs. 1000

```
mysql> select Title , isbn
    -> from books
    -> where price > 1000;
+--------------------------+------+
| Title                    | isbn |
+--------------------------+------+
| PHP And MySql programming | 1002 |
| RD SHARMA                | 1004 |
+--------------------------+------+
2 rows in set (0.02 sec)
```

13. Display the student name who issued the book “science” type.

```
mysql> SELECT B.Title , B.Author , B.Category , A.student_ID , A.student_Name , C.Issue_ID
    -> FROM books B , student A , issue_status C
    -> where B.isbn = C.isbn_book
    -> And A.student_ID = C.studentID
    -> AND B.Category = "Science";
+----------+----------+----------+------------+--------------+----------+
| Title    | Author   | Category | student_ID | student_Name | Issue_ID |
+----------+----------+----------+------------+--------------+----------+
| HC Verma | HC Verma | Science  |       2108 | Aditya Kumar |     1314 |
| HC Verma | HC Verma | Science  |       2106 | Riya Singh   |     1321 |
+----------+----------+----------+------------+--------------+----------+
2 rows in set (0.19 sec)
```

14. Display the names in alphabetical order

```
mysql> SELECT student_name FROM student ORDER BY student_name ASC;
+--------------+
| student_name |
+--------------+
| Aarti Rathi  |
| Aditya Kumar |
| Isha tyagi   |
| Nishu Rai    |
| Riya Singh   |
| Shambhavi    |
+--------------+
6 rows in set (0.02 sec)
```

15. Display all student name in upper case

```
mysql> SELECT UPPER (student_Name) AS student_Name from student;
+--------------+
| student_Name |
+--------------+
| AARTI RATHI  |
| NISHU RAI    |
| RIYA SINGH   |
| ADITYA KUMAR |
| ISHA TYAGI   |
| SHAMBHAVI    |
+--------------+
6 rows in set (0.04 sec)
```

16. Create a view consisting of employee name and position.

```
mysql> CREATE VIEW information AS SELECT E_name , Position from employee ;
Query OK, 0 rows affected (0.15 sec)

mysql> select * from information;
+----------------+-----------+
| E_name         | Position  |
+----------------+-----------+
| Rakesh Kumar   | Admin     |
| Rupali Singh   | Assistant |
| Ranbir Rathore | Manager   |
+----------------+-----------+
3 rows in set (0.03 sec)
```

17. Display name of students starting with ‘A’

```
mysql> SELECT * from student
    -> WHERE student_Name LIKE 'A%';
+------------+------+-------------+---------------------+-----------------+---------------------+
| student_ID | Dept | student_Name | student_Email       | student_Address | Regd_date           |
+------------+------+-------------+---------------------+-----------------+---------------------+
|       2101 | ENTC | Aarti Rathi  | aartirathi@gmail.com | Pune            | 2019-07-08 11:30:29 |
|       2108 | ENTC | Aditya Kumar | adiKumar@gmail.com   | Bhopal          | 2018-04-10 12:03:51 |
+------------+------+-------------+---------------------+-----------------+---------------------+
2 rows in set (0.14 sec)
```

18. Display the total number of issued books till date

```
mysql> SELECT COUNT(isbn_book) from issue_status;
+------------------+
| COUNT(isbn_book) |
+------------------+
|                4 |
+------------------+
1 row in set (0.03 sec)
```

19) Display name and department of students having ‘S’ in their name

```
mysql> select Dept,student_Name from student where student_Name like '%S%';
+------+--------------+
| Dept | student_Name |
+------+--------------+
| MECH | Nishu Rai    |
| COMP | Riya Singh   |
| IT   | Isha tyagi   |
| IT   | Shambhavi    |
+------+--------------+
4 rows in set (0.00 sec)
```

20) Display sum of price of all the books from ‘Books’ table.

```
mysql> select sum(price) from books;
+-------------+
| sum(price) |
+-------------+
|        4700 |
+-------------+
1 row in set (0.00 sec)
```

21) Display the Title of book having minimum price

```
mysql> select min(price),Title from books;
+------------+-----------------+
| min(price) | Title           |
+------------+-----------------+
|        500 | Half Girlfriend |
+------------+-----------------+
1 row in set (0.00 sec)
```

22) Group the students according to their branch

```
mysql> select Dept,student_name from student group by dept;
+------+--------------+
| Dept | student_name |
+------+--------------+
| ENTC | Aarti Rathi  |
| MECH | Nishu Rai    |
| COMP | Riya Singh   |
| IT   | Isha tyagi   |
+------+--------------+
4 rows in set (0.00 sec)
```

23) Create a view having attributes name, email, and student address.

```
mysql> create view student_detail as select student_Name,student_Address,student_Email from student;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from student_detail;
+--------------+----------------+---------------------+
| student_Name | student_Address | student_Email       |
+--------------+----------------+---------------------+
| Aarti Rathi  | Pune           | aartirathi@gmail.com |
| Nishu Rai    | Bangalore      | RaiNishu@gmail.com  |
| Riya Singh   | Jaipur         | riya@gmail.com      |
| Aditya Kumar | Bhopal         | adiKumar@gmail.com  |
| Isha tyagi   | Ambala         | isha_here@gmail.com |
| Shambhavi    | New Delhi      | shambhavi@gmail.com |
+--------------+----------------+---------------------+
6 rows in set (0.01 sec)
```

24) Select details of the books whose category name is starting with ‘M’

```
mysql> select Title,category from books where category like 'M%';
+---------------------------+-------------+
| Title                     | category    |
+---------------------------+-------------+
| PHP And MySql programming | Management  |
| RD SHARMA                 | Mathematics |
+---------------------------+-------------+
2 rows in set (0.00 sec)
```

25) Select name of student having 10 digits in their name

```
mysql> select * from student where student_name like '_________';
+------------+------+--------------+--------------------+----------------+---------------------+
| student_ID | Dept | student_Name | student_Email      | student_Address | Regd_date           |
+------------+------+--------------+--------------------+----------------+---------------------+
|       2106 | COMP | Riya Singh   | riya@gmail.com     | Jaipur          | 2019-10-18 16:02:22 |
|       2111 | IT   | Isha tyagi   | isha_here@gmail.com | Ambala          | 2018-03-16 13:40:21 |
+------------+------+--------------+--------------------+----------------+---------------------+
2 rows in set (0.00 sec)
```

7. Conclusion

SQL database management application which is very well used in the modern world in organizing and manipulating a database.

Through SQL doesn't have the GUI interface like Microsoft access is having and they all manage the database comfortable.

Depending on the user or users, if an organization has multiple users then they should go for SQL server based application.

This project shows how to create tables in SQL and how to create simple data manipulation language and data definition language with how to execute them.

It also shows how relationships are established with the concepts of primary and foreign key within a table.

Lastly, a project shows how queries are created in SQL server, queries like the create command, view, update, alter etc.

8. References.

- http://people.cs.pitt.edu/~chang/156/03ERmodel.html
- https://lbsitbytes2010.wordpress.com/2013/09/21/er-diagram-of- library-management
- https://www.slideshare.net/fiu025/library-management-32343393?next_slideshow=1
- http://www.c-sharpcorner.com/UploadFile/ea3ed6/database-design-for-library-management-system/
- http://stackoverflow.com/questions/17641134/what-is-different- between-er-diagram- and-database-schema