Proposal

It is a report of a database management system of a hospital. Data are very essential to track the things going on an organization. So, it is very important to store the data in such a way where you can meet every query minimizing the time. Each organization whether it is small or has many branches, store their data in the computer system which is retrieval and can be updated with time. This software or program is known as database management system. This database management system belongs to a hospital where data related to different possible entities of the hospital are stored connected with each other with different foreign keys. The given individual task is for creating a database program using XAMPP.

Purpose

The main purpose of this database is to maintain the information of a hospital. This database helps us to know every single detail related to different entities such as doctor working or patient visiting the hospital. And another main purpose of this coursework is to clarify the concept of databases.

Problem Statement:

Data can be stored in many ways. Before, the data were saved or stored in traditional format i.e. writing the data in hard copy, where there is high chance of losing data, giving the organization many losses. So, comparatively database management system is more reliable i.e. storing the data in a computer system where the data can be retrieved and updated, having no fear of data loss. Being a student of IT, I had to create a database related to hospital. Since I did not have enough information about it, I had to visit some small hospitals in my city before starting my project. Because of this research, it was very helpful to create a small database management system since I came to know what information is outmost necessary. Secondly, I faced some problems while storing the data. The data should be stored using relevant and unique key in many cases. If the data do not have unique key, then there are higher chances of data collision. So, these human errors must be avoided.

Aims and Objectives:

The main aim of this project is to create a small database management system for an organization, overcoming all the problems of data storage. Surfing in the internet, research from different journals and books will help to accomplish the aims of this database. With lots of research and effort, the given individual task can be completed.

Proposed Approach:

In order to create the database management system for an organization, following approaches were made:

- Firstly, research will be made on the topics related to the task such as Xampp, Er-diagram, relational diagram and databases.
- Then, Er-diagram would be drawn to know the entities and link between them.
- After that, relational diagram would be drawn to know where foreign key must belong to.
- Then, data dictionary would be created describing the entities, attributes and their type.
- The program would be written to create the database with the help of XAMPP.
- Finally, the program would be tested to know if it can complete the need of the queries.

Scope of the project:

The scope of this project is, after the completion of this project, I will be able to know about creating the database for any organization. I will be able to know how to use databases and fulfil the need of any queries relevant to it. Moreover, this project will provide a efficient way to store the data. It also illustrates how a good er-diagram, data dictionary and relational diagram is constructed and be utilized to make a database management system.

Target Audience:

This project is for a small organization i.e. hospital for the queries related to different components of the hospital. It can be utilized by students, lecturers or researchers who want to acquire the knowledge about database management system.

Hardware and Software Requirements:

In order to write the program, an app called XAMPP and operating system are used as software whereas in case of hardware, laptop or PC is required.

Activity Description and Timeline:

During our lecture and workshop classes, we were pretty clear about database and mysql, but this task requires research and understanding as well. So, the task will be carried out properly from the first week.

Activities	Duration
1. Research	1st week
2. Creating required diagrams.	2nd week
3. Creating database, checking the queries and documentation.	3rd week

Table of Contents

Table of figure

Introduction:

This project is designed for a small database management system of a hospital which stores all the data relevant to different entities such as doctor, patients and staffs. In this century, storing the data in a register is not relevant. Since, we should make complex queries and changes in the database, writing it in File Management System is also not considerable. So, by making the use of the system and programs, we can store the data safely. For instance, if we link two entities, we have to iterate through both databases of the entities but by writing queries, we can easily connect and link the entities in database management system.

This is a small software for the database management system of the hospital which is very useful for storing the records. In daily cases, the hospital database management system can be really complex, but since this project is given to store at least five entities, which are linked with each other, it is the perfect example for a simpler form of the database of a hospital. So, considering this aspect, I have created a database, having six tables. The six tables or entities are doctor, patients, medical report, bill, ward and staffs. By writing queries, we can simply fulfil the requirement and since the database of the hospital requires high manipulation and update, data is fluctuating in database management system.

Using a program, which easily allows the user to make changes when necessary and can fulfil the requirement asked by the user is conversely the best way to manage the records or data of the hospitals. This database management system overcomes all the demerits of storing data in a register or by File Management System. The merits of using database management system is briefly described as below:

- Accessing data: It allows the user to find required record from hundreds or thousands of the records.
- Fluctuating data: Since, the data or information of a hospital keeps on changing, this project allows the user to change and update the record when necessary.
- Queries: Here, it displays all the data or record asked by the user in tabular form, simply by writing a query.

- Effective: It displays six different entities in tabular form which are linked with each other visualizing all the information of the entity.
- Secured data: Recording the data in a paper work can be erased or damaged but here the data is stored in a secured manner and it can be retrieved as well.

Discussion and Analysis:

This project was developed and designed using relational database management system. Here, in relational database management system, the structured databases are linked with each other by the relation between the entities of the database. It is a common type of database whose information is stored in tabular form. Most of the databases used in businesses these days are relational databases. From a small industry or organization, to a large scale, relational database management system is considered the efficient one.

Databases

Database is the structured set of data stored in a computer system which is accessible to multiple users in various manner. It makes data management easy and efficient.

Database Management System:

It is a software that handles the storage, updates and retrieval of the data in a computer system. In other words, it is for creating and managing the databases. It makes possible for users to create, read, delete and update the data in databases. It ensures the consistently organization of the data and its accessibility. It is most useful tool to provide the centralized view of data that can be accessed by multiple users, from different locations, in a secured and controlled manner. There are different types of database management system, but for the businesses from the small scale to the large one, relational database management system is used in maximum scale. (Raza, 2018)

Relational Database Management System:

It is the collection of the programs and capabilities that enables the team to create, update and administrate the relational databases. As it says itself, relational database is the collection of the entities that are related or linked with each other. Relational database management system is the database management system that has a row-based table structure that connects related data elements. It includes functions that maintain the security, integrity, accuracy and consistency of the data. (Rouse, 2005)

These are the information that was gathered in order to understand and complete this project successfully. In addition, some tools such as XAMPP and notepad were used which are briefly described as below:

XAMPP: XAMPP stands for Cross-Platform(X), Apache(A), MySQL(M), PHP(P) and Perl(P). It is a free and open source cross-platform web server which was developed by Apache Friends. The full form of XAMPP describes it properties:

- Cross-platform: It means you can install XAMPP on any OS.
- Apache: In order to run PHP script on local machine you need server.
- MySQL: It stores the data and allows you to perform the database operation.
- PHP: It is well known scripting language to create website.
- Perl: Programming language.
- Notepad: It is simply a text editor for Microsoft Windows to create the documents. It was firstly released in 1983 and was included in all versions of Microsoft Windows since 1985.

Database Model:

The task given to us was to create a database management system for any organization of our own choice with five different entities. Hereby, I created a relational database management system of a hospital where the entities are doctor, patients, medical_report, bill and ward. Since, I have used relational databases, here we have the concept of foreign key. Foreign and primary key both are the constraints used in relational databases i.e. the datatype of these fields is set. These two keys are briefly defined below:

- Primary Key: This key uniquely defines or alters the record of an entity. Here, primary key should be unique and not null by default. In this project, every entity has a primary key.
- Foreign Key: If the primary key of one table is made as a field of another table, it becomes foreign key. The main function of the foreign key is to link two tables. For example, in this database management system, doctor_id from table doctor became foreign key as doctor in patients table. (Chauhan, 2012)

Before starting this project, the skeleton work must be done. So, firstly Er-diagram was designed. This helped to know the links between the significant entities. Six entities were taken and I found out the relation between them i.e. if the relation is one to many or many to one. For example, in between the entities, doctor and patients, the relation is one to many, so the foreign key goes under patients table. Similarly, I found out relation between all these table by drawing the ER-diagram which is given as below:

Er-Diagram:

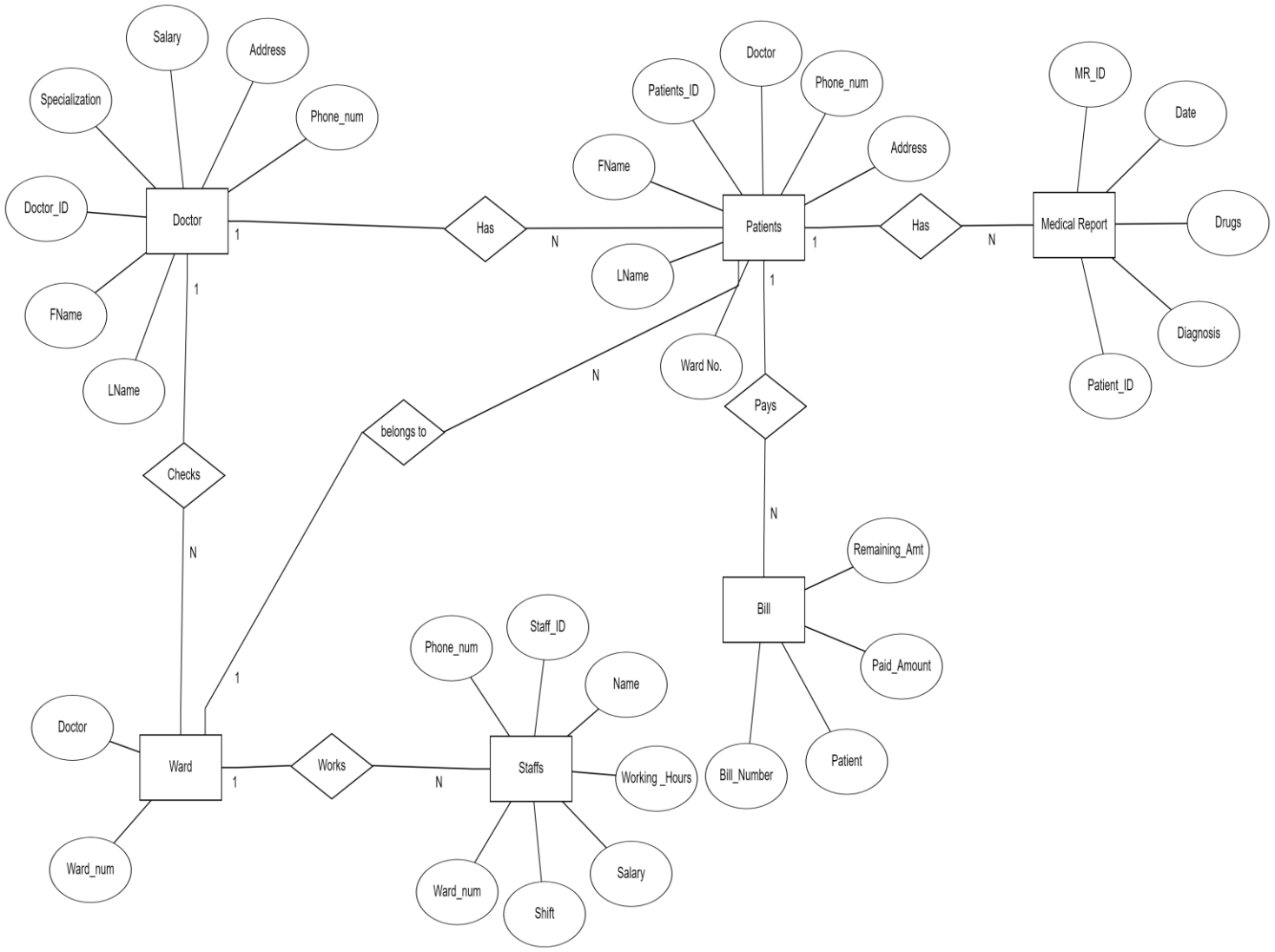


Figure 1: ER-diagram of a hospital database management system

Description:

A entity-relationship diagram(ER-Diagram) is a modeling technique of the data that graphically represents the entities and relationships between them. The above ER-Diagram is the conceptional framework of my project. With the help of this ER-diagram, the possible relations were found between the entities. These entities have different attributes such as FName, LName, Address, Phone_num,etc. Every entity have these kind of attributes respectively to the need. We have two different relation i.e. one to many and many to one. In the case where many belongs to, we add a foreign ke3y to another table. (Anon., 2019)Their relations are given as:

- A doctor can have many patients / A patient can have only one doctor.
- A patient can have many medical reports / A medical report belongs to only one patient.
- A doctor can check many ward / A ward can be checked by only one doctor.
- A ward can have many staffs / A staff works in only one ward.
- A patient belongs to only one ward / A ward can have many patients.
- A patient pays many bills / A bill belongs to only one patient.

Relational Diagram

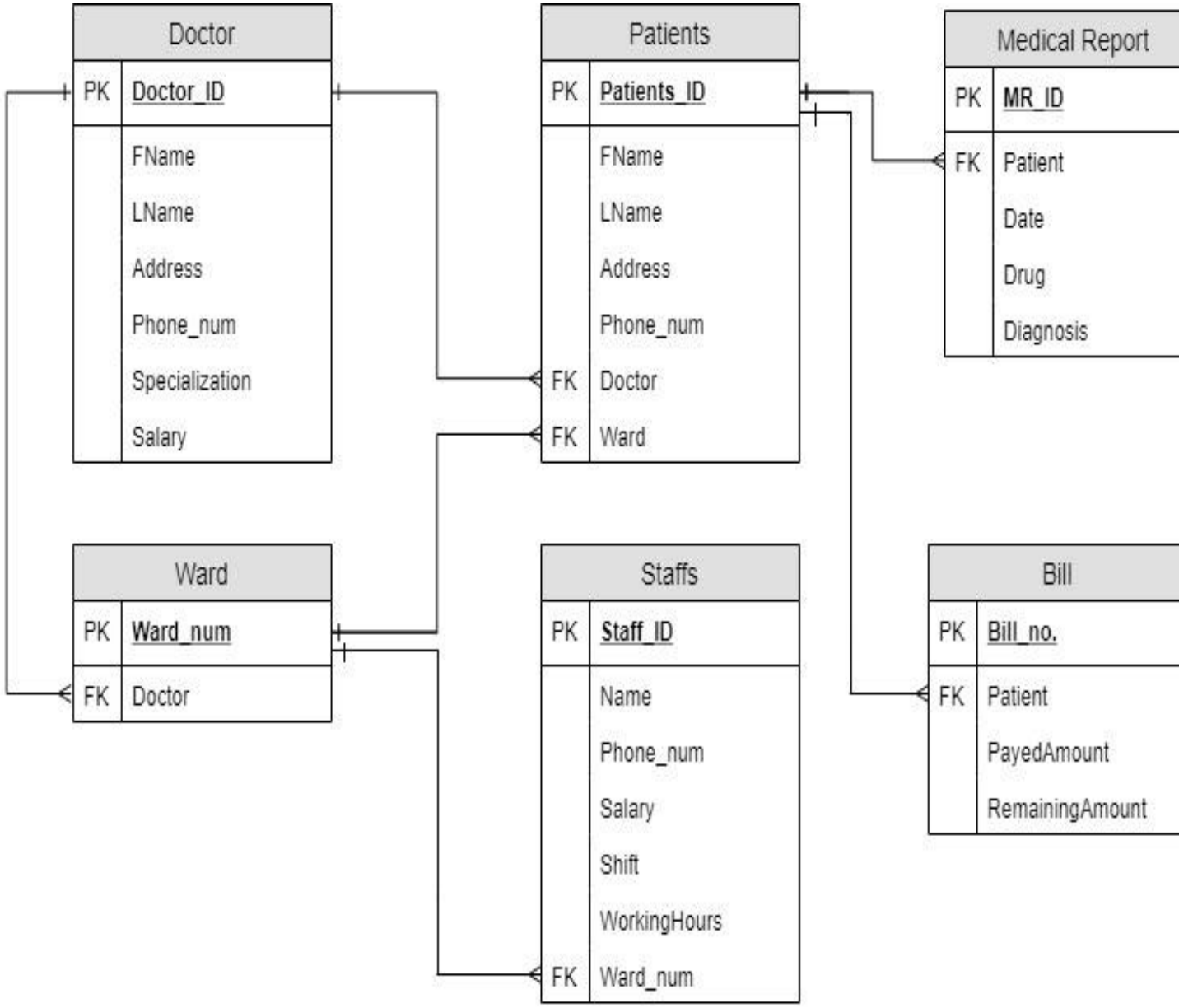


Figure 2: Relational Diagram of hospital management system

Description:

Relational diagram is also a graphical representation of the databases. It actually shows the possible relation and presents the link between the entities i.e. foreign keys. The foreign keys present in the diagram are:

- Doctor_ID in Doctor table as Doctor in Patients table.
- Ward_num in Ward table as Ward in Patients table.
- Patients_ID in Patients table as Patient in Medical_Report table.
- Patients_ID in Patients table as Patient in Bill table.
- Ward_num in Ward table as Ward_num in Staffs table.
- Doctor_ID in Doctor table as Doctor in Ward table.

Description of Entities:

This database management system belongs to a hospital, so the possible entities, I have worked out are described as below:

- **Doctor:**

  Doctor is someone who checks and consults the patients in the hospital. This entity has seven different attributes and they are Doctor_Id, FName, Lname, Address, Phone_num, Specialization and Salary.

- **Creation of Doctor Table:**

```
MariaDB [Hospital]> CREATE TABLE Doctor(
    -> Doctor_ID INT PRIMARY KEY AUTO_INCREMENT,
    -> FName VARCHAR(15) NOT NULL,
    -> LName VARCHAR(15) NOT NULL,
    -> Address VARCHAR(15) NOT NULL,
    -> Phone_num VARCHAR(15) NOT NULL UNIQUE,
    -> Specialization VARCHAR(20) NOT NULL,
    -> Salary INT NOT NULL
    -> );                          Table created
Query OK, 0 rows affected (0.02 sec)
```

Figure 3:Creation of Doctor Table

Above picture illustrates the creation of Doctor entity or table. Here, this entity has seven attributes where Doctor_ID is the primary key which represents every record uniquely. Similarly, other fields are set to NOT NULL which means that particular filed cannot be empty. Hereby, the contact number is set as UNIQUE because phone number cannot be same. Entire field is set to VARCHAR except salary because other fields do not require manipulation. Salary has datatype as INT because we need to manipulate it if necessary.

- **Describing Doctor Table:**

```
MariaDB [Hospital]> DESCRIBE Doctor;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| Doctor_ID      | int(11)     | NO   | PRI | NULL    | auto_increment |
| FName          | varchar(15) | NO   |     | NULL    |                |
| LName          | varchar(15) | NO   |     | NULL    |                |
| Address        | varchar(15) | NO   |     | NULL    |                |
| Phone_num      | varchar(15) | NO   | UNI | NULL    |                |
| Specialization | varchar(20) | NO   |     | NULL    |                |
| Salary         | int(11)     | NO   |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

Figure 4: Describing Doctor Table

Here, we see the description of the doctor table, illustrating all the required information from the datatype to nullable or not. It also shows that the phone number of the doctor is UNIQUE.

- **Insertion of Values in Doctor Table:**

```
MariaDB [Hospital]> INSERT INTO Doctor(FName,LName,Address,Phone_num,Specialization,Salar
y)VALUES
    -> ("Suraksha","Shrestha","Kalanki","+9779813762576","Cardiologist",500000),
    -> ("Amisha","Bhatta","Baneshwor","++9779808782976","Allergist",100000),
    -> ("Usha","Bohara","Kalanki","+9779841764895","Anesthesiologist",900000),
    -> ("Supriya","Ghimire","Kamalpokhari","+9779808356217","Psychiatrist",200000),
    -> ("Salon","Adhikari","Dillibazar","+9779841762903","Cardiologist",900000);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Figure 5: Insertion of Doctor Record

Above screenshot illustrates addition of the records in the doctor table. Here, I have added 5 different records.

- **Displaying the Records of Doctor Table:**

```
MariaDB [Hospital]> SELECT * FROM Doctor;
+-----------+---------+---------+-------------+----------------+-----------------+--------+
| Doctor_ID | FName   | LName   | Address     | Phone_num      | Specialization  | Salary |
+-----------+---------+---------+-------------+----------------+-----------------+--------+
|         1 | Suraksha | Shrestha | Kalanki     | +9779813762576 | Cardiologist    | 500000 |
|         2 | Amisha  | Bhatta  | Baneshwor   | ++9779808782976 | Allergist       | 100000 |
|         3 | Usha    | Bohara  | Kalanki     | +9779841764895 | Anesthesiologist | 900000 |
|         4 | Supriya | Ghimire | Kamalpokhari | +9779808356217 | Psychiatrist    | 200000 |
|         5 | Salon   | Adhikari | Dillibazar  | +9779841762903 | Cardiologist    | 900000 |
+-----------+---------+---------+-------------+----------------+-----------------+--------+
5 rows in set (0.00 sec)
```

Figure 6: Displaying Records of Doctor Table

The above screenshot displays all the records of the doctors where doctor_Id uniquely defines those records.

Patients:

Patients are those people who visit hospital for check up and diagnosis their sickness. This entity has seven attributes. They are Patients_ID, FName, LName, Address, Phone_num, doctor and ward where Patients_ID is the primary key.

- **Creating Patients Table:**

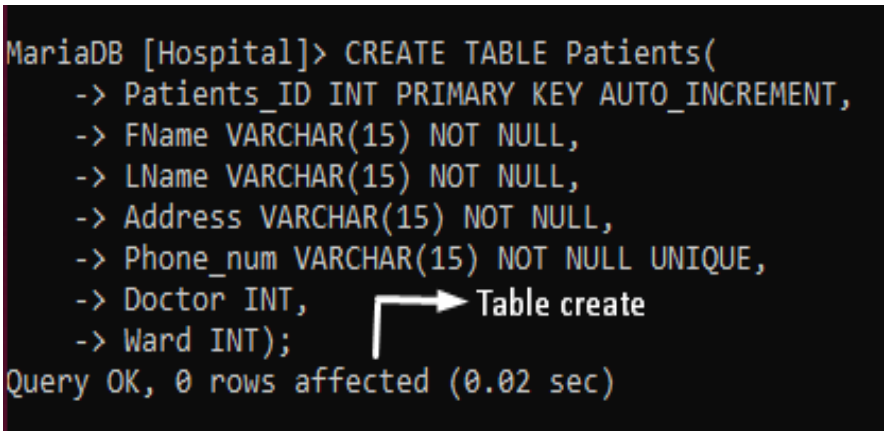


Figure 7:Creation of Table Patients

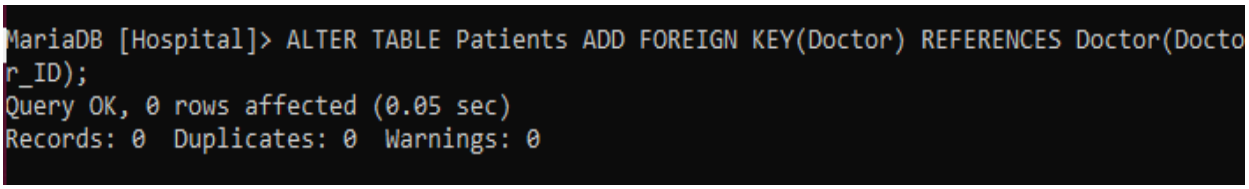


Figure 8: Declaring Doctor as Foreign Key

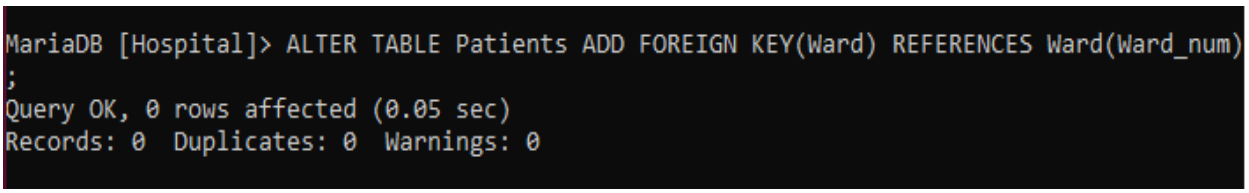


Figure 9:Declaring Ward as Foreign Key

Above screenshot shows creation of table patients. Here, we have seven attributes, most of them NOT NULL. Phone_num is set to be UNIQUE. Also,

the attributes, Doctor and Ward is set as foreign key connecting Patients table with Doctor and Ward table.

- **Describing Table Patients:**

```
MariaDB [Hospital]> DESCRIBE Patients;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| Patients_ID | int(11)     | NO   | PRI | NULL    | auto_increment |
| FName       | varchar(15) | NO   |     | NULL    |                |
| LName       | varchar(15) | NO   |     | NULL    |                |
| Address     | varchar(15) | NO   |     | NULL    |                |
| Phone_num   | varchar(15) | NO   | UNI | NULL    |                |
| Doctor      | int(11)     | YES  | MUL | NULL    |                |
| Ward        | int(11)     | YES  | MUL | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

Figure 10: Description of Patients Table

Here, we see the description of the patients table, illustrating all the required information from the datatype to nullable or not. It also shows that the phone number of the patient is UNIQUE.

- **Insertion of Values in Patients Table:**

```
MariaDB [Hospital]> INSERT INTO Patients(FName,LName,Address,Phone_num,Doctor,Ward) VALU
ES
    -> ("Sandesh","Aryal","Sitapaila","+9779808261527",2,2),
    -> ("Sochan","Dangol","Gangabu","+9779808262387",1,3),
    -> ("Abhishek","Bohara","Baneshwor","+9779813725389",1,2),
    -> ("Saugat","Ghimire","Naikap","+9779808263828",2,4),
    -> ("Sunil","Yadav","Kirtipur","+9779808735283",3,4),
    -> ("Ashwin","Belbase","Ravibhawan","+9779813847493",2,5),
    -> ("Jenith","Karki","Kalimati","+9779813382633",1,4),
    -> ("Astha","Shrestha","Kalanki","+97798418276352",4,5),
    -> ("Amisha","Maharjan","Jawalakhel","+9779808375271",5,1);
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0
```

Figure 11: Inserting Values in Patients Values

Above screenshot illustrates addition of the records in the patients table. Here, I have added 9 different records.

- **Displaying Records of Patients Table:**

```
MariaDB [Hospital]> SELECT * FROM Patients;
+-------------+----------+----------+------------+-----------------+--------+------+
| Patients_ID | FName    | LName    | Address    | Phone_num       | Doctor | Ward |
+-------------+----------+----------+------------+-----------------+--------+------+
|          16 | Sandesh  | Aryal    | Sitapaila  | +9779808261527  |      2 |    2 |
|          17 | Sochan   | Dangol   | Gangabu    | +9779808262387  |      1 |    3 |
|          18 | Abhishek | Bohara   | Baneshwor  | +9779813725389  |      1 |    2 |
|          19 | Saugat   | Ghimire  | Naikap     | +9779808263828  |      2 |    4 |
|          20 | Sunil    | Yadav    | Kirtipur   | +9779808735283  |      3 |    4 |
|          21 | Ashwin   | Belbase  | Ravibhawan | +9779813847493  |      2 |    5 |
|          22 | Jenith   | Karki    | Kalimati   | +9779813382633  |      1 |    4 |
|          23 | Astha    | Shrestha | Kalanki    | +97798418276352 |      4 |    5 |
|          24 | Amisha   | Maharjan | Jawalakhel | +9779808375271  |      5 |    1 |
+-------------+----------+----------+------------+-----------------+--------+------+
9 rows in set (0.00 sec)
```

Figure 12:Displaying Patients Record

The above screenshot displays all the records of the patients where patients_Id uniquely defines those records.

Staffs:

Staff is someone who works in the hospital under the doctor. This table has eight attributes and they are Staff_ID

- **Creating Staffs Table:**

```
MariaDB [Hospital]> CREATE TABLE Staffs(
    ->      Staff_ID INT PRIMARY KEY AUTO_INCREMENT,
    ->      Name VARCHAR(30) NOT NULL,
    ->      Phone_num VARCHAR(15) UNIQUE,
    ->      Salary INT NOT NULL,
    ->      Shift VARCHAR(15) NOT NULL,
    ->      WorkingHours INT NOT NULL,
    ->      Ward_num INT NOT NULL);
Query OK, 0 rows affected (0.02 sec)
```

Table created

Figure 13:Creating Staffs Table

```
MariaDB [Hospital]> ALTER TABLE Staffs ADD FOREIGN KEY (Ward_num) REFERENCES Ward(Ward_n
um);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Figure 14: Declaring Ward_num as Foreign Key

Above picture illustrates the creation of Staffs entity or table. Here, this entity has seven attributes where Staff_ID is the primary key which represents every record uniquely. Similarly, other fields are set to NOT NULL which means that particular filed cannot be empty. Hereby, the contact number of staffs is set as UNIQUE because phone number cannot be same. Entire field is set to VARCHAR except salary and workingHours because other fields do not require manipulation. Salary and workingHours has datatype as INT because we need to manipulate it if necessary.

- **Describing Staffs Table:**

```
MariaDB [Hospital]> DESCRIBE Staffs;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| Staff_ID     | int(11)     | NO   | PRI | NULL    | auto_increment |
| Name         | varchar(30) | NO   |     | NULL    |                |
| Phone_num    | varchar(15) | YES  | UNI | NULL    |                |
| Salary       | int(11)     | NO   |     | NULL    |                |
| Shift        | varchar(15) | NO   |     | NULL    |                |
| WorkingHours | int(11)     | NO   |     | NULL    |                |
| Ward_num     | int(11)     | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

Figure 15:Description of Staffs Table

Here, we see the description of the staffs table, illustrating all the required information from the datatype to nullable or not. It also shows that the phone number of the staff is UNIQUE.

- **Insertion of Values in Staffs Table:**

```
MariaDB [Hospital]> INSERT INTO Staffs(Name,Phone_num,Salary,Shift,WorkingHours,Ward_num
)VALUES
    ->     ("Jeevan Kafley","+9779808265478",30000,"DAY",8,2),
    ->     ("Sankalpa Pokherel","+97798137564256",20000,"DAY",5,3),
    ->     ("Shruti Yadav","+9779813624372",40000,"NIGHT",5,1),
    ->     ("Shrija Khanal","+97798185341628",40000,"NIGHT",5,1),
    ->     ("Saya Shrestha","+9779818768535",50000,"NIGHT",8,4),
    ->     ("Sulav Shrestha","+9779808575435",50000,"NIGHT",8,5),
    ->     ("Sajan Tamrakar","+9779841765897",40000,"NIGHT",5,5),
    ->     ("Rabeena Joshi","+9779813724531",35000,"DAY",5,5),
    ->     ("Srijana Bhandari","+9779813534263",35000,"DAY",5,1),
    ->     ("Bikalpa Joshi","+9779808562413",40000,"DAY",8,4),
    ->     ("Bijen Ghimire","+9779808564789",40000,"NIGHT",8,2),
    ->     ("Shobha Pandey","+9779813745623",40000,"NIGHT",8,3);
Query OK, 12 rows affected (0.01 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

Figure 16:Inserting Records of Staffs

Above screenshot illustrates addition of the records in the staffs table. Here, I have added 12 different records.

- **Displaying Records of Staff Table:**

```
MariaDB [Hospital]> SELECT * FROM Staffs;
+----------+------------------+-----------------+--------+-------+--------------+----------+
| Staff_ID | Name             | Phone_num       | Salary | Shift | WorkingHours | Ward_num |
+----------+------------------+-----------------+--------+-------+--------------+----------+
|        1 | Jeevan Kafley    | +9779808265478  |  30000 | DAY   |            8 |        2 |
|        2 | Sankalpa Pokherel| +97798137564256 |  20000 | DAY   |            5 |        3 |
|        3 | Shruti Yadav     | +9779813624372  |  40000 | NIGHT |            5 |        1 |
|        4 | Shrija Khanal    | +97798185341628 |  40000 | NIGHT |            5 |        1 |
|        5 | Saya Shrestha    | +9779818768535  |  50000 | NIGHT |            8 |        4 |
|        6 | Sulav Shrestha   | +9779808575435  |  50000 | NIGHT |            8 |        5 |
|        7 | Sajan Tamrakar   | +9779841765897  |  40000 | NIGHT |            5 |        5 |
|        8 | Rabeena Joshi    | +9779813724531  |  35000 | DAY   |            5 |        5 |
|        9 | Srijana Bhandari | +9779813534263  |  35000 | DAY   |            5 |        1 |
|       10 | Bikalpa Joshi    | +9779808562413  |  40000 | DAY   |            8 |        4 |
|       11 | Bijen Ghimire    | +9779808564789  |  40000 | NIGHT |            8 |        2 |
|       12 | Shobha Pandey    | +9779813745623  |  40000 | NIGHT |            8 |        3 |
+----------+------------------+-----------------+--------+-------+--------------+----------+
12 rows in set (0.00 sec)
```

Figure 17: Displaying Staffs Records

The above screenshot displays all the records of the staffs where staff_Id uniquely defines those records.

Medical_Report:

Medical report is the hardcopy of the records which includes all the information related to the diagnosis of the patient disease. It has six attributes, they are MR_ID, patient, Date, Diagnosis and Drug. Here, MR_ID is the primary key.

- **Creating Medical_Report Table:**

```
MariaDB [Hospital]> CREATE TABLE Medical_Report(
    -> MR_ID INT PRIMARY KEY AUTO_INCREMENT,
    -> Patient INT,
    -> FOREIGN KEY(Patient) REFERENCES Patients(Patients_ID),
    -> Date DATE NOT NULL,
    -> Drug VARCHAR(225) NOT NULL,
    -> Diagnosis VARCHAR(225) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Figure 18: Creating Medical Report Table

Above picture illustrates the creation of Medical_Report entity or table. Here, this entity has six attributes where MR_ID is the primary key which represents every record uniquely. Similarly, other fields are set to NOT NULL which means that particular filed cannot be empty. Hereby, patient is set as foreign key from Patients table.

- Describing Medical_Report**:**

```
MariaDB [Hospital]> DESCRIBE Medical_Report;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| MR_ID     | int(11)      | NO   | PRI | NULL    | auto_increment |
| Patient   | int(11)      | YES  | MUL | NULL    |                |
| Date      | date         | NO   |     | NULL    |                |
| Drug      | varchar(225) | NO   |     | NULL    |                |
| Diagnosis | varchar(225) | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Figure 19: Desribing Medical_Report Table

Here, we see the description of the Medical_Report table, illustrating all the required information from the datatype to nullable or not.

- **Insertion of Values in Medical_Report Table:**

```
MariaDB [Hospital]> INSERT INTO Medical_Report(Patient,Date,Drug,Diagnosis) VALUES
    -> (18,"2018-11-08","Isoniazid","Tuberculosis"),
    -> (16,"2018-11-23","Ciprofloxacin","Typhoid"),
    -> (20,"2018-10-13","Acetaminophen","Tonsillitis"),
    -> (21,"2019-01-10","Ioperamide","Diarrhea"),
    -> (23,"2018-12-27","Macrolide","Pneumonia"),
    -> (17,"2019-02-04","Acetaminophen","Common cold"),
    -> (19,"2019-02-24","Isoniazid","Tuberculosis"),
    -> (22,"2018-09-19","Methimazole","Hyperthyroidism"),
    -> (24,"2019-03-10","Ciprofloxacin","Typhoid"),
    -> (17,"2018-11-09","Macrolide","Peumonia"),
    -> (16,"2019-02-09","Acetaminophen","Tonsillitis");
Query OK, 11 rows affected (0.01 sec)
Records: 11  Duplicates: 0  Warnings: 0
```

Figure 20: Inserting Values in Medical_Report Table

Above screenshot illustrates addition of the records in the Medical_Report table. Here, I have added 11 different records.

- Displaying the Values in Medical_Report Table:

```
MariaDB [Hospital]> SELECT * FROM Medical_Report;
+-------+---------+------------+---------------+-----------------+
| MR_ID | Patient | Date       | Drug          | Diagnosis       |
+-------+---------+------------+---------------+-----------------+
|     1 |      18 | 2018-11-08 | Isoniazid     | Tuberculosis    |
|     2 |      16 | 2018-11-23 | Ciprofloxacin | Typhoid         |
|     3 |      20 | 2018-10-13 | Acetaminophen | Tonsillitis     |
|     4 |      21 | 2019-01-10 | Ioperamide    | Diarrhea        |
|     5 |      23 | 2018-12-27 | Macrolide     | Pneumonia       |
|     6 |      17 | 2019-02-04 | Acetaminophen | Common cold     |
|     7 |      19 | 2019-02-24 | Isoniazid     | Tuberculosis    |
|     8 |      22 | 2018-09-19 | Methimazole   | Hyperthyroidism |
|     9 |      24 | 2019-03-10 | Ciprofloxacin | Typhoid         |
|    10 |      17 | 2018-11-09 | Macrolide     | Peumonia        |
|    11 |      16 | 2019-02-09 | Acetaminophen | Tonsillitis     |
+-------+---------+------------+---------------+-----------------+
11 rows in set (0.00 sec)
```

Figure 21: Displaying Records

The above screenshot displays all the records of the medical report where MR_ID uniquely defines those records.

Bill:

Bill is a small piece of paper where the amount paid and remained is written or stored.

- **Creating Bill Table:**

```
MariaDB [Hospital]> CREATE TABLE Bill(
    ->      BillNumber INT PRIMARY KEY AUTO_INCREMENT,
    ->      Patient INT,
    ->      PaidAmount INT NOT NULL,
    ->      RemainingAmount INT NOT NULL);
Query OK, 0 rows affected (0.02 sec)
```

Figure 22:Creating Bill Table

```
MariaDB [Hospital]> ALTER TABLE Bill ADD FOREIGN KEY (Patient) REFERENCES Patients(Patie
nts_ID);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Figure 23: Declaring Patients As Foreign Key

Above picture illustrates the creation of Bill entity or table. Here, this entity has four attributes where BillNumber is the primary key which represents every record uniquely. Similarly, other fields are set to NOT NULL which means that particular filed cannot be empty. Hereby, patient is set as foreign key from Patients table.

- **Describing Bill Table:**

```
MariaDB [Hospital]> DESCRIBE Bill;
+----------------+---------+------+-----+---------+----------------+
| Field          | Type    | Null | Key | Default | Extra          |
+----------------+---------+------+-----+---------+----------------+
| BillNumber     | int(11) | NO   | PRI | NULL    | auto_increment |
| Patient        | int(11) | YES  | MUL | NULL    |                |
| PaidAmount     | int(11) | NO   |     | NULL    |                |
| RemainingAmount | int(11) | NO   |     | NULL    |                |
+----------------+---------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Figure 24: Describing Bill Table

Here, we see the description of the Bill table, illustrating all the required information from the datatype to nullable or not.

- **Insertion of Values in Bill Table:**

```
MariaDB [Hospital]> INSERT INTO Bill(Patient,PaidAmount,RemainingAmount) VALUES
    ->     (17,5000,6000),
    ->     (16,2000,7000),
    ->     (18,10000,5000),
    ->     (19,10000,0),
    ->     (22,9000,1000),
    ->     (17,10000,0),
    ->     (16,2000,0),
    ->     (20,2000,0),
    ->     (21,3000,0),
    ->     (23,11000,0),
    ->     (24,8000,0);
Query OK, 11 rows affected (0.01 sec)
Records: 11  Duplicates: 0  Warnings: 0
```

Figure 25: Inserting Bill Records

Above screenshot illustrates addition of the records in the Bill table. Here, I have added 11 different records.

- **Displaying the Values in Bill Table:**

```
MariaDB [Hospital]> SELECT * FROM Bill;
+------------+---------+------------+------------------+
| BillNumber | Patient | PaidAmount | RemainingAmount |
+------------+---------+------------+------------------+
|         12 |      17 |       5000 |             6000 |
|         13 |      16 |       2000 |             7000 |
|         14 |      18 |      10000 |             5000 |
|         15 |      19 |      10000 |                0 |
|         16 |      22 |       9000 |             1000 |
|         17 |      17 |      10000 |                0 |
|         18 |      16 |       2000 |                0 |
|         19 |      20 |       2000 |                0 |
|         20 |      21 |       3000 |                0 |
|         21 |      23 |      11000 |                0 |
|         22 |      24 |       8000 |                0 |
+------------+---------+------------+------------------+
11 rows in set (0.00 sec)
```

Figure 26: Displaying Bill Records

The above screenshot displays all the records of the bill where BillNumber uniquely defines those records.

Ward:

Ward is the room where the patients are kept for ventilation. Ward is monitored by one doctor at one time.

- **Creating Ward Table:**

```
MariaDB [Hospital]> CREATE TABLE Ward
    ->      (Ward_num INT PRIMARY KEY AUTO_INCREMENT,
    ->      Doctor INT);
Query OK, 0 rows affected (0.02 sec)
```

Figure 27: Creating Ward Table

```
MariaDB [Hospital]> ALTER TABLE Ward ADD FOREIGN KEY(Doctor) REFERENCES Doctor(Doctor_ID
);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Figure 28: Declaring Doctor as Foreign Key

Above picture illustrates the creation of Ward entity or table. Here, this entity has two attributes where Ward_num is the primary key which represents every record uniquely and doctor is the primary key connecting table Ward with table Doctor.

- **Describing Table Ward:**

```
MariaDB [Hospital]> DESCRIBE Ward;
+----------+---------+------+-----+---------+----------------+
| Field    | Type    | Null | Key | Default | Extra          |
+----------+---------+------+-----+---------+----------------+
| Ward_num | int(11) | NO   | PRI | NULL    | auto_increment |
| Doctor   | int(11) | YES  | MUL | NULL    |                |
+----------+---------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

Figure 29: Describing Table Ward

Here, we see the description of the Ward table, illustrating all the required information from the datatype to nullable or not.

- **Insertion of Values in Ward Table:**

```
MariaDB [Hospital]> INSERT INTO Ward(Doctor) VALUES
    -> (1),
    -> (1),
    -> (3),
    -> (4),
    -> (5);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Figure 30: Inserting Record in Ward Table

Above screenshot illustrates addition of the records in the Ward table. Here, I have added 5 different records.

- **Displaying Values of Table Ward:**

```
MariaDB [Hospital]> SELECT * FROM Ward;
+----------+--------+
| Ward_num | Doctor |
+----------+--------+
|        1 |      1 |
|        2 |      1 |
|        3 |      3 |
|        4 |      4 |
|        5 |      5 |
+----------+--------+
5 rows in set (0.00 sec)
```

Figure 31: Displaying the records of Table Ward

The above screenshot displays all the records of the ward table where Ward_num uniquely defines those records.

Data Dictionary:

A data dictionary is a file or set of files that contains all the information of a entity of the database. It contains the records about the entity in the database, such as data relationships and ownership. It is the crucial component of any relational database.

The data dictionary of given entities is given below:

Data dictionary of Doctor:

Entity name	Entity Description	Column Name	Column Description	Data type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Doctor	A doctor is someone who checks the patients and diagnosis the disease.	Doctor_ID	It is for the unique identification of the doctor.	INT		True	False	False	True	Auto Incremented
		FName	First name of the doctor	VARCHAR	15	False	False	False	False	
		LName	Last name of the doctor	VARCHAR	15	False	False	False	False	
		Address	Address of the doctor	VARCHAR	15	False	False	False	False	
		Phone_num	Contact of the doctor	VARCHAR	15	False	False	False	True	
		Specialization	Specialization of the doctor	VARCHAR	20	False	False	False	False	
		Salary	Salary of the doctor	INT		False	False	False	False	

Table 1: Data Dictionary of Doctor

Data Dictionary of Patients:

Entity name	Entity Description	Column Name	Column Description	Data Type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Patients	A patient is someone who visits hospital for check up.	Patients_ID	Unique identification	INT		True	False	False	True	Auto Incremented
		FName	First name of patient	Varchar	15	False	False	False	False	
		LName	Last name of patient	Varchar	15	False	False	False	False	
		Address	Address of patient	Varchar	15	False	False	False	True	
		Phone_num	Contact number of the patient	Varchar	15	False	False	True	False	
		Doctor	Stores the doctor Id who has checked the patient	INT		False	True	False	False	References to Doctor_ID of Doctor table
		Ward	The ward number patient belongs to	INT		False	True	False	False	References to ward_num fron table ward

Table 2: Data Dictionary of Patients

Data Dictionary of Medical Report:

Entity name	Entity Description	Column Name	Column Description	Data Type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Medical_Report	A medical report is a hard copy having info of the patient.	MR_ID	Uniguely identifying the report	INT		True	False	False	True	Auto Incremented
		Patient	Name of the patient to whom this report belongs	INT		False	True	False	False	References to patient_id of patient table
		Date	Date when report is issued	DATE		False	False	False	False	
		Drug	Drug recommended by doctors	VARCHAR	225	False	False	False	False	
		Diagnosis	Disease which is diagnosed.	VARCHAR	225	False	False	False	False	

Table 3: Data Dictionary of Medical Report

Data Dictionary of Bill

Entity name	Entity Description	Column Name	Column Description	Data Type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Bill	Bill is the entity having info of the amount paid.	BillNumber	Uniquely identifies the bill	INT		True	False	False	True	Auto Increment
		Patient	Name of the patient whom this bill belongs	INT		False	True	False	False	Reference to patient_id of patient table
		PaidAmount	Amount already paid	INT		False	False	False	False	
		RemainingAmount	Amount remained to pay	INT		False	False	False	False	

Table 4: Data Dictionary for Bill

Data Dictionary of Staff:

Entity name	Entity Description	Column Name	Column Description	Data Type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Staff	A staff is a [person who works in the hospital.	Satff_ ID	Uniquely identifies the staff	INT		True	False	False	True	Auto Increment
		Name	Name of the staff	VARCHAR	30	False	False	False	False	
		Phone_num	Contact number of staff	VARCHAR	15	False	False	True	False	
		Salary	Salary of the staff	INT		False	False	False	False	
		Shift	Shift when staff works	VARCHAR	15	False	False	True	False	
		WorkingHours	Hours worked by the staff	INT		False	False	False	False	
		Ward_num	Ward where staff works	INT		False	True	False	False	Reference to ward_num from ward table

Table 5: Data Dictionary of Staffs

Data Dictionary of Ward:

Entity name	Entity Description	Column Name	Column Description	Data Type	Length	Primary Key	Foreign Key	Nullable	Unique	Notes
Ward	Ward is an entity identifying a room.	Ward_num	Uniquely identifies the ward	INT		True	False	False	True	Auto Increment
		Doctor	Dictor rounding the ward	INT		False	True	False	False	Reference to Doctor_ID from doctor table

Table 6: Data Dictionary of Ward

Queries:

Query is the request for data or information from the given database table or its combination. This data is generated with the help of SQL (Structured Query Language).

Some of the queries of this database are given as below:

1, Query to Display Doctors Having Specialization as Cardiologist:

```
MariaDB [Hospital]> SELECT * FROM Doctor WHERE Specialization="Cardiologist";
+-----------+---------+---------+-----------+---------------+---------------+--------+
| Doctor_ID | FName   | LName   | Address   | Phone_num     | Specialization | Salary |
+-----------+---------+---------+-----------+---------------+---------------+--------+
|         1 | Suraksha | Shrestha | Kalanki   | +9779813762576 | Cardiologist  | 500000 |
|         5 | Salon   | Adhikari | Dillibazar | +9779841762903 | Cardiologist  | 900000 |
+-----------+---------+---------+-----------+---------------+---------------+--------+
2 rows in set (0.00 sec)
```

This query gives the name of doctors along with their addresses, phone numbers and salary whose specialization is Cardiologist. This query uses the WHERE clause to obtain its desired output.

2. Query to Display Records of Bills having PaidAmount Greater Than 5000:

```
MariaDB [Hospital]> SELECT * FROM Bill GROUP BY BillNumber HAVING PaidAmount > 5000;
+------------+---------+------------+-----------------+
| BillNumber | Patient | PaidAmount | RemainingAmount |
+------------+---------+------------+-----------------+
|         14 |      18 |      10000 |            5000 |
|         15 |      19 |      10000 |               0 |
|         16 |      22 |       9000 |            1000 |
|         17 |      17 |      10000 |               0 |
|         21 |      23 |      11000 |               0 |
|         22 |      24 |       8000 |               0 |
+------------+---------+------------+-----------------+
6 rows in set (0.00 sec)
```

This query allows the user to view the BillNumber, PatientID, Amount Paind and Amount Remaining where the Amount Paid is greater than 5000. This query uses GROUP BY and HAVING clause to obtain the desired result.

3. Query to sum up the remainingAmount:

```
MariaDB [Hospital]> SELECT * FROM Bill;
+------------+---------+------------+-----------------+
| BillNumber | Patient | PaidAmount | RemainingAmount |
+------------+---------+------------+-----------------+
|         12 |      17 |       5000 |            6000 |
|         13 |      16 |       2000 |            7000 |
|         14 |      18 |      10000 |            5000 |
|         15 |      19 |      10000 |               0 |
|         16 |      22 |       9000 |            1000 |
|         17 |      17 |      10000 |               0 |
|         18 |      16 |       2000 |               0 |
|         19 |      20 |       2000 |               0 |
|         20 |      21 |       3000 |               0 |
|         21 |      23 |      11000 |               0 |
|         22 |      24 |       8000 |               0 |
+------------+---------+------------+-----------------+
11 rows in set (0.00 sec)

MariaDB [Hospital]> SELECT sum(RemainingAmount) AS ToReceive FROM Bill;
+-----------+
| ToReceive |
+-----------+
|     19000 |
+-----------+
1 row in set (0.01 sec)
```

This query sums up all of the money that is yet to be received from the patients for their treatment. This query uses the aggregate function SUM to ontain the desired output. It also uses the alias clause AS.

4. Listing details of the patients who visited this year:

```
MariaDB [Hospital]> SELECT * FROM Medical_Report WHERE YEAR(Date)="2019";
+-------+---------+------------+---------------+--------------+
| MR_ID | Patient | Date       | Drug          | Diagnosis    |
+-------+---------+------------+---------------+--------------+
|     4 |      21 | 2019-01-10 | Ioperamide    | Diarrhea     |
|     6 |      17 | 2019-02-04 | Acetaminophen | Common cold  |
|     7 |      19 | 2019-02-24 | Isoniazid     | Tuberculosis |
|     9 |      24 | 2019-03-10 | Ciprofloxacin | Typhoid      |
|    11 |      16 | 2019-02-09 | Acetaminophen | Tonsillitis  |
+-------+---------+------------+---------------+--------------+
5 rows in set (0.01 sec)
```

This query gives the Patient ID, Medical Report ID, Date , Name of Drug and the diagnosis of the patients who were admitted in a certain year. This uses the Where clause along with YEAR() Function to obtain the desired Output.

5. Query to List Record of the Doctors Alphabetically:

```
MariaDB [Hospital]> SELECT * FROM Doctor ORDER BY FName;
+-----------+---------+---------+-------------+-----------------+------------------+--------+
| Doctor_ID | FName   | LName   | Address     | Phone_num       | Specialization   | Salary |
+-----------+---------+---------+-------------+-----------------+------------------+--------+
|         2 | Amisha  | Bhatta  | Baneshwor   | ++9779808782976 | Allergist        | 100000 |
|         5 | Salon   | Adhikari| Dillibazar  | +9779841762903  | Cardiologist     | 900000 |
|         4 | Supriya | Ghimire | Kamalpokhari| +9779808356217  | Psychiatrist     | 200000 |
|         1 | Suraksha| Shrestha| Kalanki     | +9779813762576  | Cardiologist     | 500000 |
|         3 | Usha    | Bohara  | Kalanki     | +9779841764895  | Anesthesiologist | 900000 |
+-----------+---------+---------+-------------+-----------------+------------------+--------+
5 rows in set (0.00 sec)
```

This query gives every data contained in the Doctor table in an alphabetical order according to Fname. This query uses ORDER BY clause to obtain the desired output.

6. Query to Limit the Record of Patients:

```
MariaDB [Hospital]> SELECT * FROM Patients ORDER BY FName LIMIT 7;
+-------------+---------+---------+-----------+-----------------+--------+------+
| Patients_ID | FName   | LName   | Address   | Phone_num       | Doctor | Ward |
+-------------+---------+---------+-----------+-----------------+--------+------+
|          18 | Abhishek | Bohara  | Baneshwor | +9779813725389  |      1 |    2 |
|          24 | Amisha  | Maharjan | Jawalakhel | +9779808375271 |      5 |    1 |
|          21 | Ashwin  | Belbase | Ravibhawan | +9779813847493 |      2 |    5 |
|          23 | Astha   | Shrestha | Kalanki  | +97798418276352 |      4 |    5 |
|          22 | Jenith  | Karki   | Kalimati  | +9779813382633  |      1 |    4 |
|          16 | Sandesh | Aryal   | Sitapaila | +9779808261527  |      2 |    2 |
|          19 | Saugat  | Ghimire | Naikap    | +9779808263828  |      2 |    4 |
+-------------+---------+---------+-----------+-----------------+--------+------+
7 rows in set (0.00 sec)
```

This query shows the top 7 from the patients table while the table is sorted in ascending order by the FName column. This uses the LIMIT clause along with ORDER BY clause to obtain the desired output.

7. Query to Shifts:

```
MariaDB [Hospital]> SELECT DISTINCT (Shift) FROM Staffs;
+-------+
| Shift |
+-------+
| DAY   |
| NIGHT |
+-------+
2 rows in set (0.01 sec)

MariaDB [Hospital]>
```

This query displays the unique values int eh shift column from the Staffs table. This query uses DISTINCT clause to obtain the output desired by the user.

8. Query to Count Total Patients:

```
MariaDB [Hospital]> SELECT COUNT(*) AS PatientsNum FROM Patients;
+-------------+
| PatientsNum |
+-------------+
|           9 |
+-------------+
1 row in set (0.00 sec)
```

This query counts the number of records or the number of patients from the Patients table. This is done by using the COUNT() function that counts the number of records in the specified column.

9. Query to see Which Doctor is Checking which Patient:

```
MariaDB [Hospital]> SELECT doctor.FName AS DoctorFname,doctor.LName AS DoctorLname,patients.FName AS PatientFname,patients.LName AS PatientLname
    -> FROM doctor
    -> JOIN patients
    -> ON doctor.doctor_id=patients.doctor;
+------------+------------+-------------+-------------+
| DoctorFname | DoctorLname | PatientFname | PatientLname |
+------------+------------+-------------+-------------+
| Amisha     | Bhatta     | Sandesh     | Aryal       |
| Suraksha   | Shrestha   | Sochan      | Dangol      |
| Suraksha   | Shrestha   | Abhishek    | Bohara      |
| Amisha     | Bhatta     | Saugat      | Ghimire     |
| Usha       | Bohara     | Sunil       | Yadav       |
| Amisha     | Bhatta     | Ashwin      | Belbase     |
| Suraksha   | Shrestha   | Jenith      | Karki       |
| Supriya    | Ghimire    | Astha       | Shrestha    |
| Salon      | Adhikari   | Amisha      | Maharjan    |
+------------+------------+-------------+-------------+
9 rows in set (0.00 sec)
```

This query displays the Names of Doctors and the respective Names of patients that they are incharge of. This is done using the JOIN clause or the INNER JOIN clause in MySQL.

1. Query to Join Multiple Tables

```
MariaDB [hospital]> SELECT patients.FName AS PatientFname,patients.LName AS PatientLname,doctor.Fname AS DoctorFname,doctor.Lname AS DoctorLname,
    -> medical_report.MR_ID,medical_report.Drug, medical_report.Diagnosis
    -> FROM patients
    -> JOIN doctor ON patients.Doctor=doctor.Doctor_ID
    -> JOIN medical_report ON
    -> patients.Patients_ID=medical_report.Patient;
+-------------+-------------+------------+------------+-------+--------------+----------------+
| PatientFname | PatientLname | DoctorFname | DoctorLname | MR_ID | Drug         | Diagnosis      |
+-------------+-------------+------------+------------+-------+--------------+----------------+
| Abhishek    | Bohara      | Suraksha   | Shrestha   |     1 | Isoniazid    | Tuberculosis   |
| Sochan      | Dangol      | Suraksha   | Shrestha   |     6 | Acetaminophen | Common cold    |
| Jenith      | Karki       | Suraksha   | Shrestha   |     8 | Methimazole  | Hyperthyroidism |
| Sochan      | Dangol      | Suraksha   | Shrestha   |    10 | Macrolide    | Peumonia       |
| Sandesh     | Aryal       | Amisha     | Bhatta     |     2 | Ciprofloxacin | Typhoid        |
| Ashwin      | Belbase     | Amisha     | Bhatta     |     4 | Ioperamide   | Diarrhea       |
| Saugat      | Ghimire     | Amisha     | Bhatta     |     7 | Isoniazid    | Tuberculosis   |
| Sandesh     | Aryal       | Amisha     | Bhatta     |    11 | Acetaminophen | Tonsillitis    |
| Sunil       | Yadav       | Usha       | Bohara     |     3 | Acetaminophen | Tonsillitis    |
| Astha       | Shrestha    | Supriya    | Ghimire    |     5 | Macrolide    | Pneumonia      |
| Amisha      | Maharjan    | Salon      | Adhikari   |     9 | Ciprofloxacin | Typhoid        |
+-------------+-------------+------------+------------+-------+--------------+----------------+
11 rows in set (0.00 sec)
```

This query gives the name of patients, their doctors, their medical report and their diagnosis with the drug they need to take. The data is stored in three different tables, so the three different tables need to be joined by using the JOIN clause.

10. Query using left join.

```
MariaDB [hospital]> SELECT patients.FName,patients.LName,patients.Address,medical_report.MR_ID FROM patients
    -> LEFT JOIN medical_report
    -> ON patients.Patients_ID=medical_report.Patient;
+----------+----------+------------+-------+
| FName    | LName    | Address    | MR_ID |
+----------+----------+------------+-------+
| Sandesh  | Aryal    | Sitapaila  |     2 |
| Sandesh  | Aryal    | Sitapaila  |    11 |
| Sochan   | Dangol   | Gangabu    |     6 |
| Sochan   | Dangol   | Gangabu    |    10 |
| Abhishek | Bohara   | Baneshwor  |     1 |
| Saugat   | Ghimire  | Naikap     |     7 |
| Sunil    | Yadav    | Kirtipur   |     3 |
| Ashwin   | Belbase  | Ravibhawan |     4 |
| Jenith   | Karki    | Kalimati   |     8 |
| Astha    | Shrestha | Kalanki    |     5 |
| Amisha   | Maharjan | Jawalakhel |     9 |
| John     | Smith    | Kenya      |  NULL |
+----------+----------+------------+-------+
12 rows in set (0.00 sec)
```

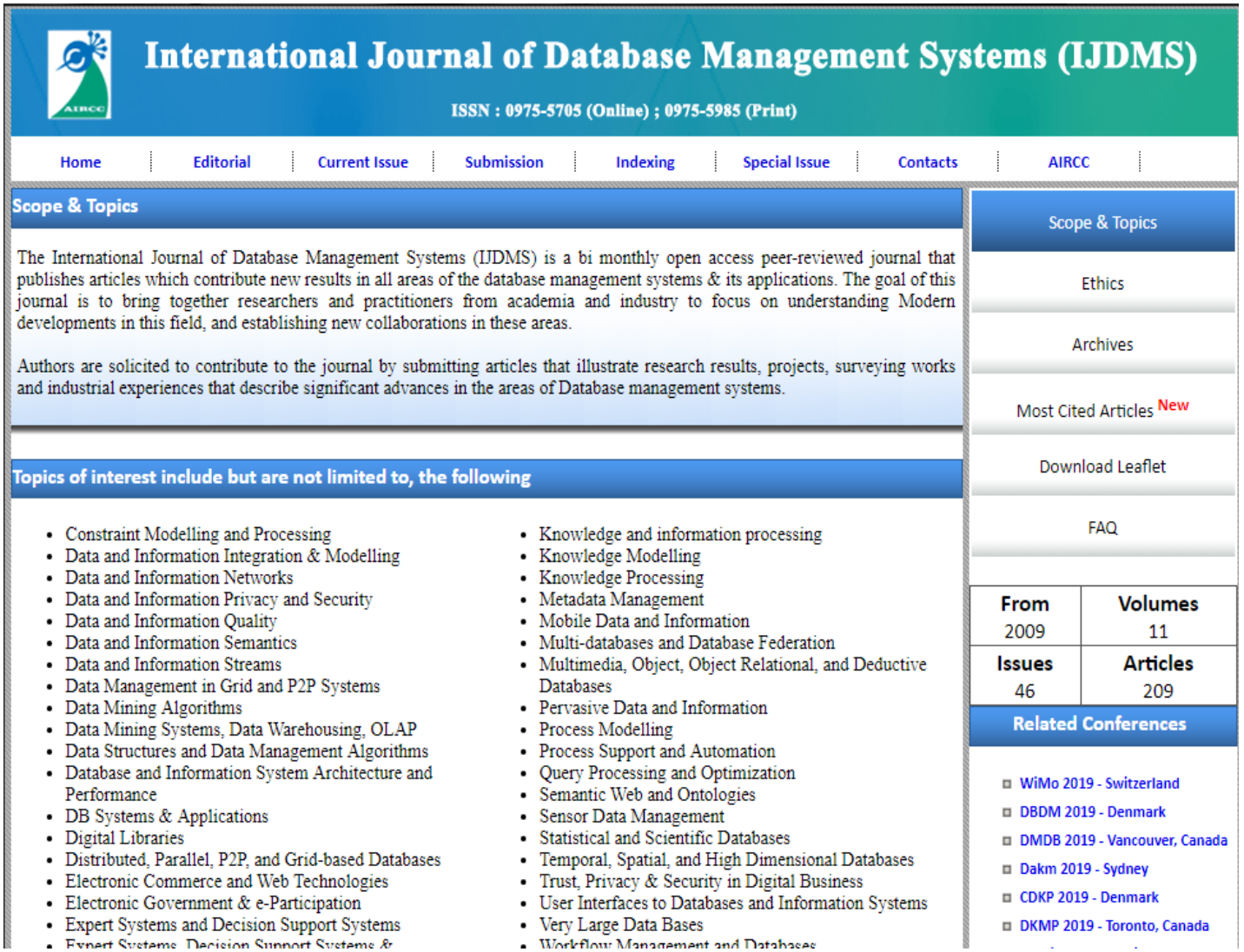
This query is to join two tables in such a manner that all the data from the left table need to be shown with corresponding data from the right table. Here we are showing all the data from patients table with their corresponding Medical Report ID. If the patient has no medical report, a NULL is displayed. This is obtained using the LEFT JOIN clause.

Research:

For the completion of the task, a lot of research is needed on the relevant topics such as databases, management system, RDMS, etc. With constant effort and research, this coursework was completed in time. I researched many websites, journals and accessed many books, to bring up the final project. This research clarified my concept related to the relevant topic

Some of the journals, books and websites researched are as follows:

- Journals:
  - http://airccse.org/journal/ijdms/index.html



International Journal of Database Management Systems (IJDMS)

ISSN : 0975-5705 (Online) ; 0975-5985 (Print)

Home | Editorial | Current Issue | Submission | Indexing | Special Issue | Contacts | AIRCC

Scope & Topics

The International Journal of Database Management Systems (IJDMS) is a bi monthly open access peer-reviewed journal that publishes articles which contribute new results in all areas of the database management systems & its applications. The goal of this journal is to bring together researchers and practitioners from academia and industry to focus on understanding Modern developments in this field, and establishing new collaborations in these areas.

Authors are solicited to contribute to the journal by submitting articles that illustrate research results, projects, surveying works and industrial experiences that describe significant advances in the areas of Database management systems.

Topics of interest include but are not limited to, the following

- Constraint Modelling and Processing
- Data and Information Integration & Modelling
- Data and Information Networks
- Data and Information Privacy and Security
- Data and Information Quality
- Data and Information Semantics
- Data and Information Streams
- Data Management in Grid and P2P Systems
- Data Mining Algorithms
- Data Mining Systems, Data Warehousing, OLAP
- Data Structures and Data Management Algorithms
- Database and Information System Architecture and Performance
- DB Systems & Applications
- Digital Libraries
- Distributed, Parallel, P2P, and Grid-based Databases
- Electronic Commerce and Web Technologies
- Electronic Government & e-Participation
- Expert Systems and Decision Support Systems
- Knowledge and information processing
- Knowledge Modelling
- Knowledge Processing
- Metadata Management
- Mobile Data and Information
- Multi-databases and Database Federation
- Multimedia, Object, Object Relational, and Deductive Databases
- Pervasive Data and Information
- Process Modelling
- Process Support and Automation
- Query Processing and Optimization
- Semantic Web and Ontologies
- Sensor Data Management
- Statistical and Scientific Databases
- Temporal, Spatial, and High Dimensional Databases
- Trust, Privacy & Security in Digital Business
- User Interfaces to Databases and Information Systems
- Very Large Data Bases

Scope & Topics

Ethics

Archives

Most Cited Articles New

Download Leaflet

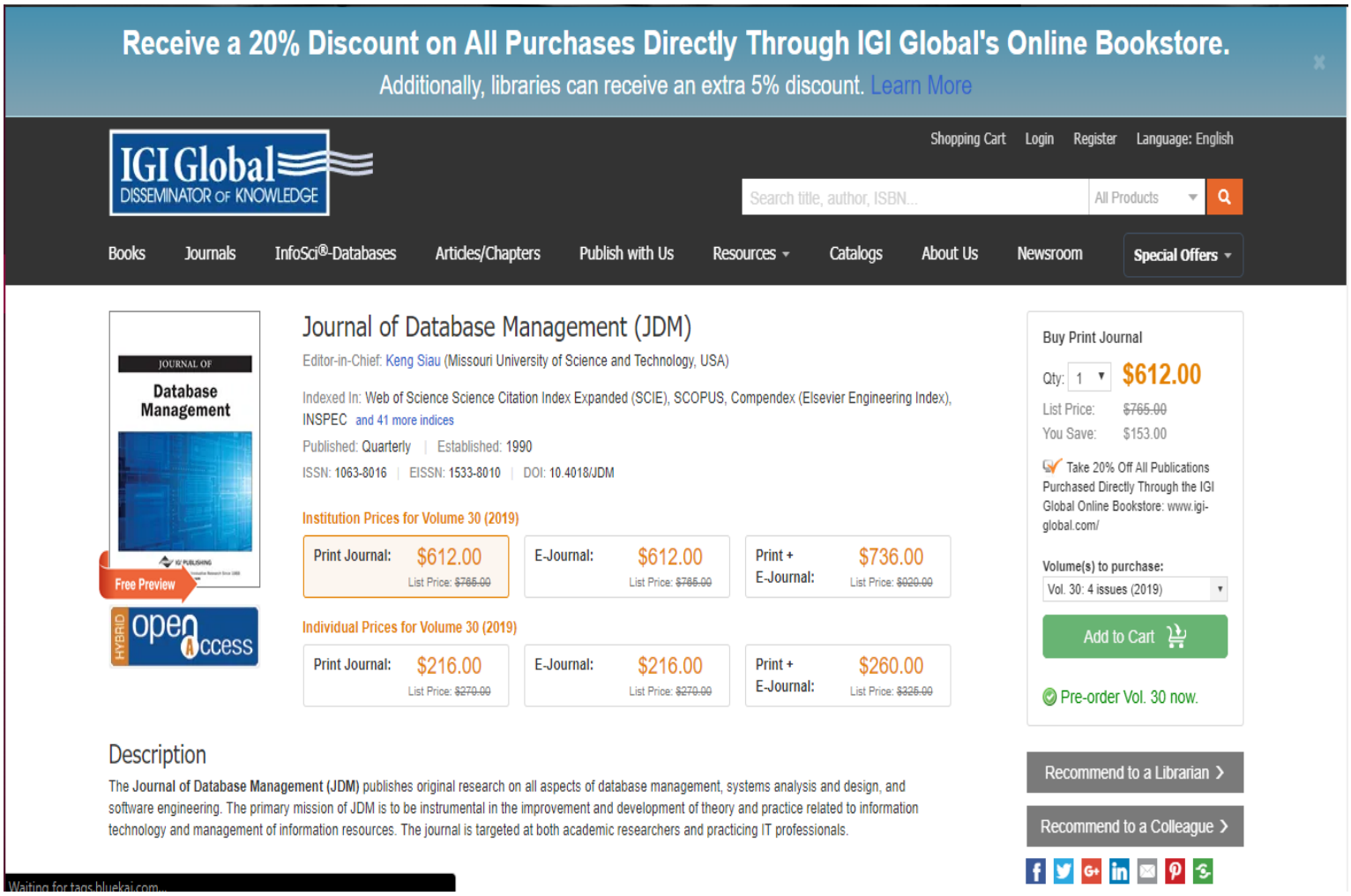
FAQ

From	Volumes
2009	11
Issues	**Articles**
46	209

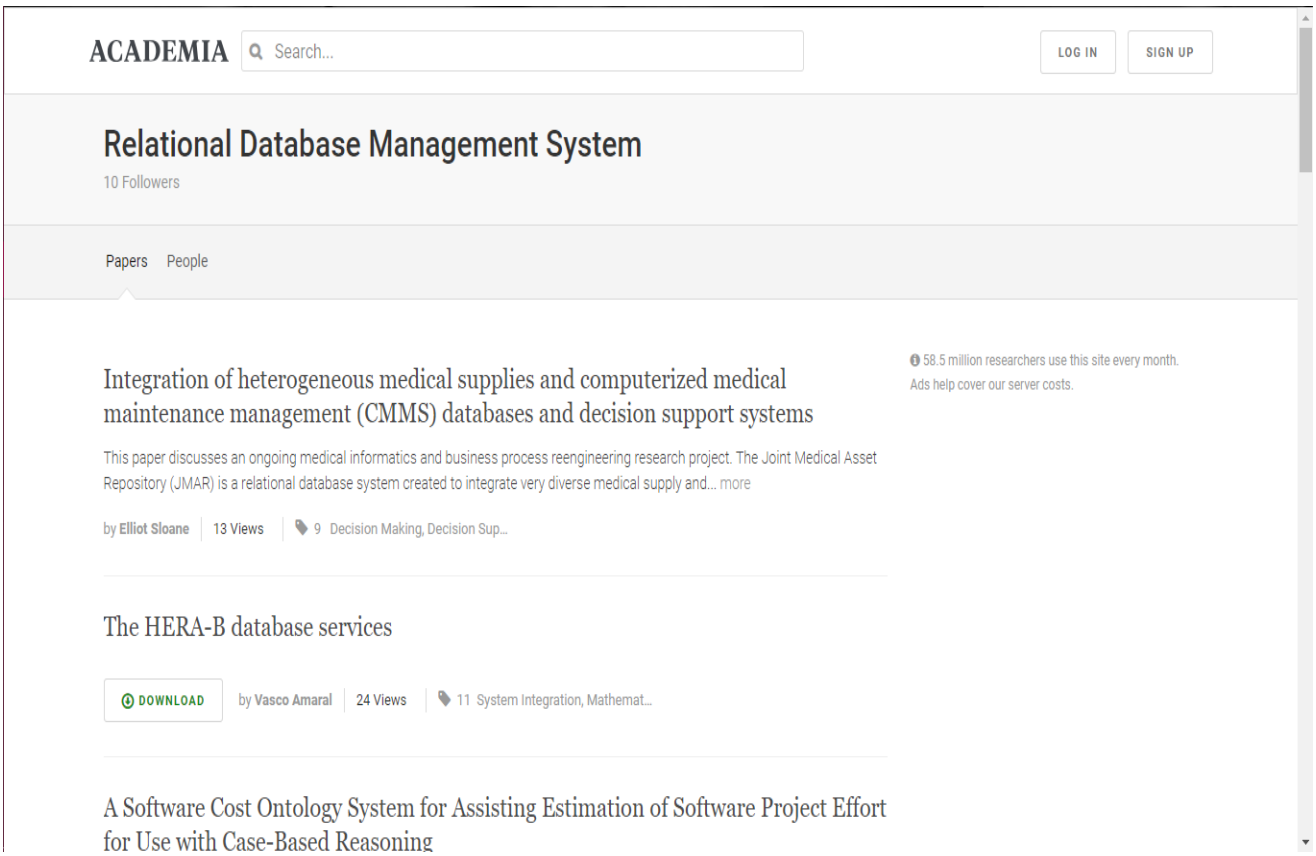
Related Conferences

- WiMo 2019 - Switzerland
- DBDM 2019 - Denmark
- DMDB 2019 - Vancouver, Canada
- Dakm 2019 - Sydney
- CDKP 2019 - Denmark
- DKMP 2019 - Toronto, Canada

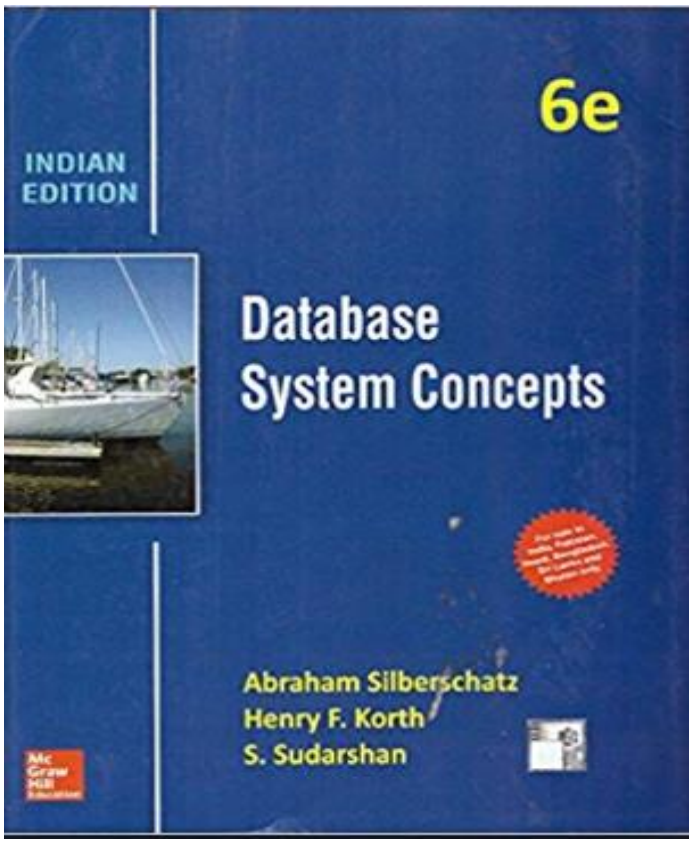
- https://www.igi-global.com/journal/journal-database-management/1072



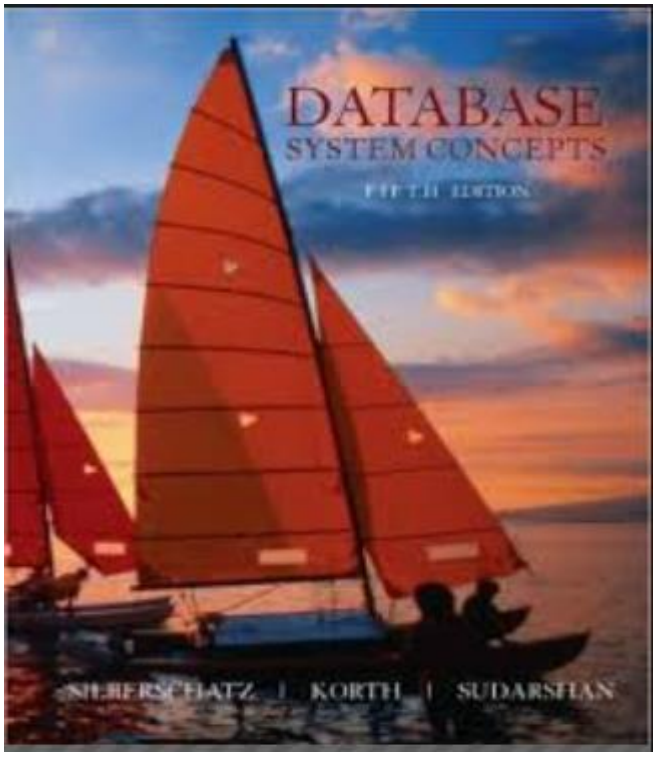
- http://www.academia.edu/Documents/in/Relational_Database_Management_System



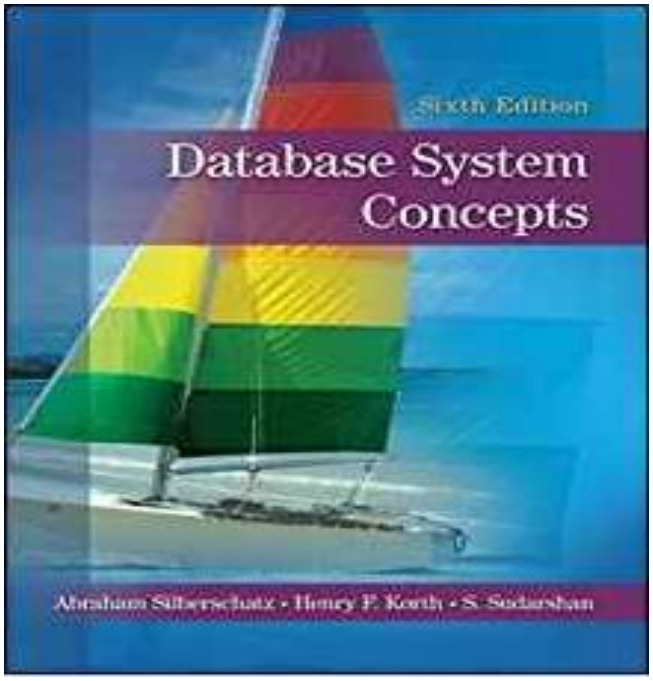
- Books:
  - Database System Concepts 1:



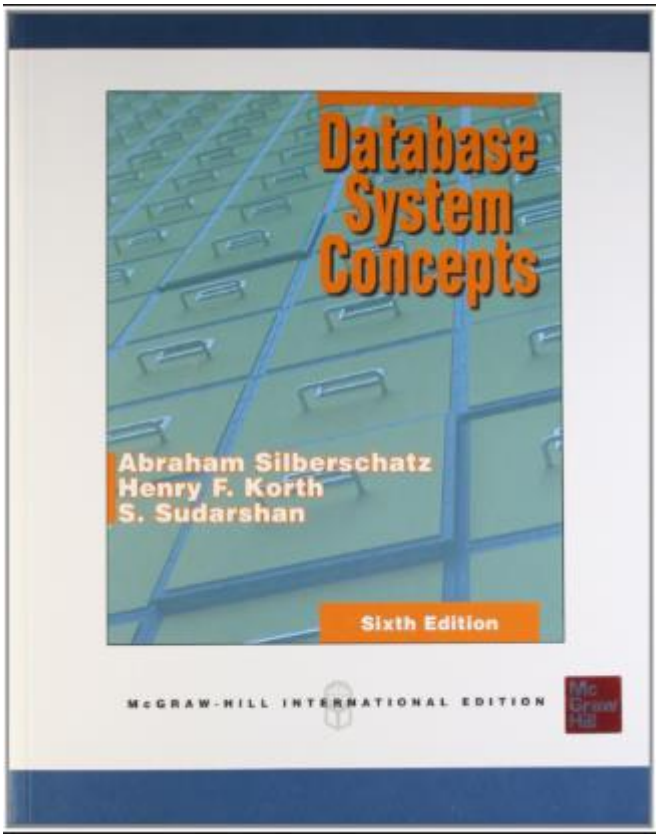
- Database Systems Concept 2:



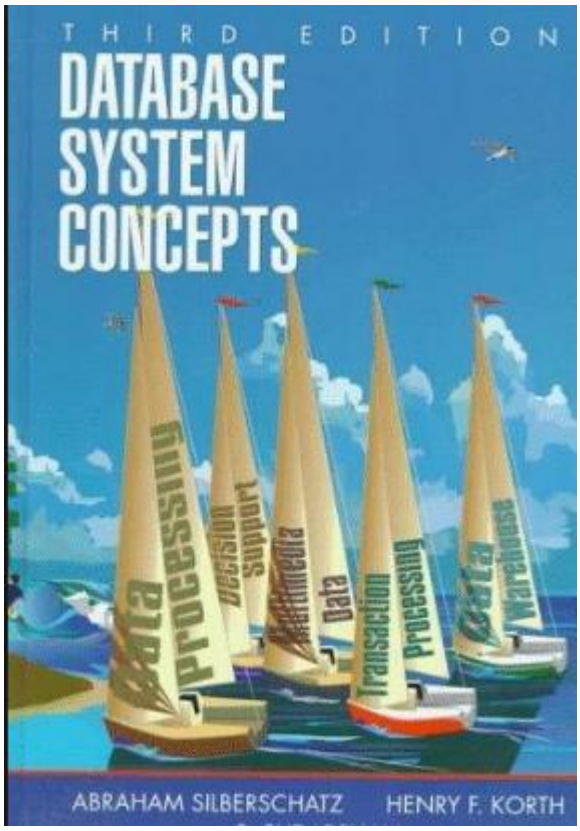
- Database System Concepts 3:



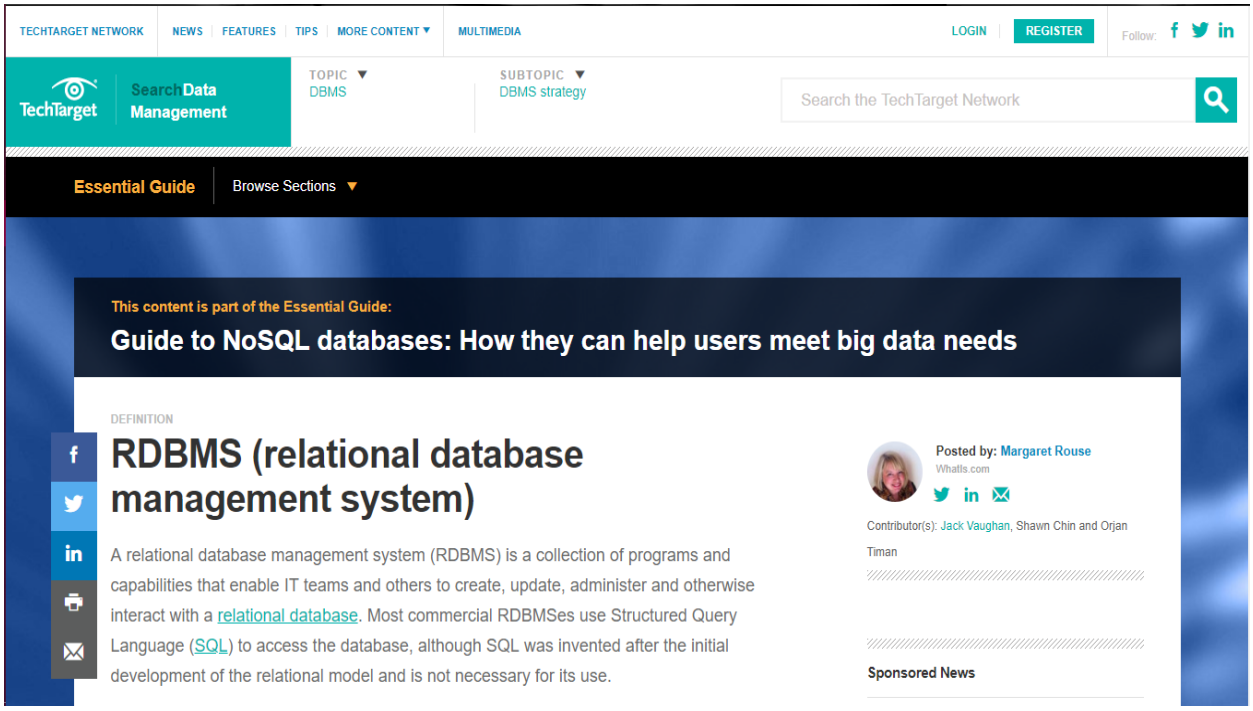
- Database System Concepts 4:



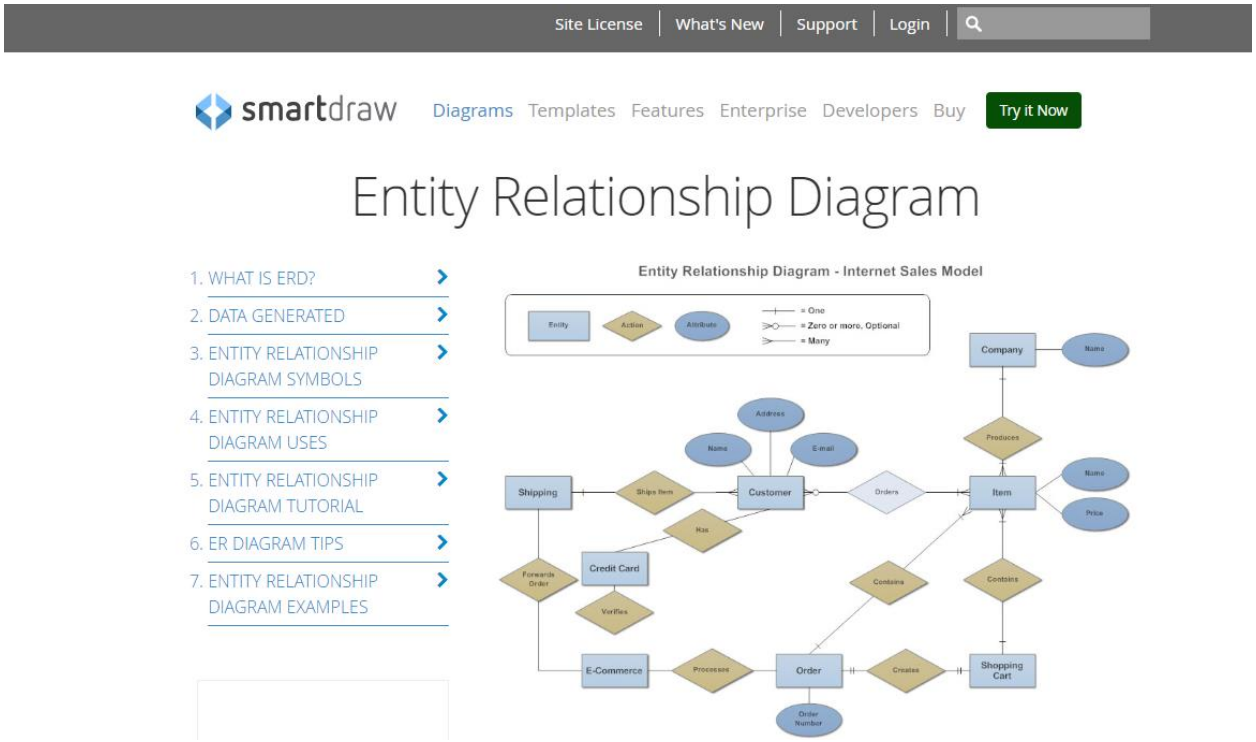
- Database System Concepts 5:



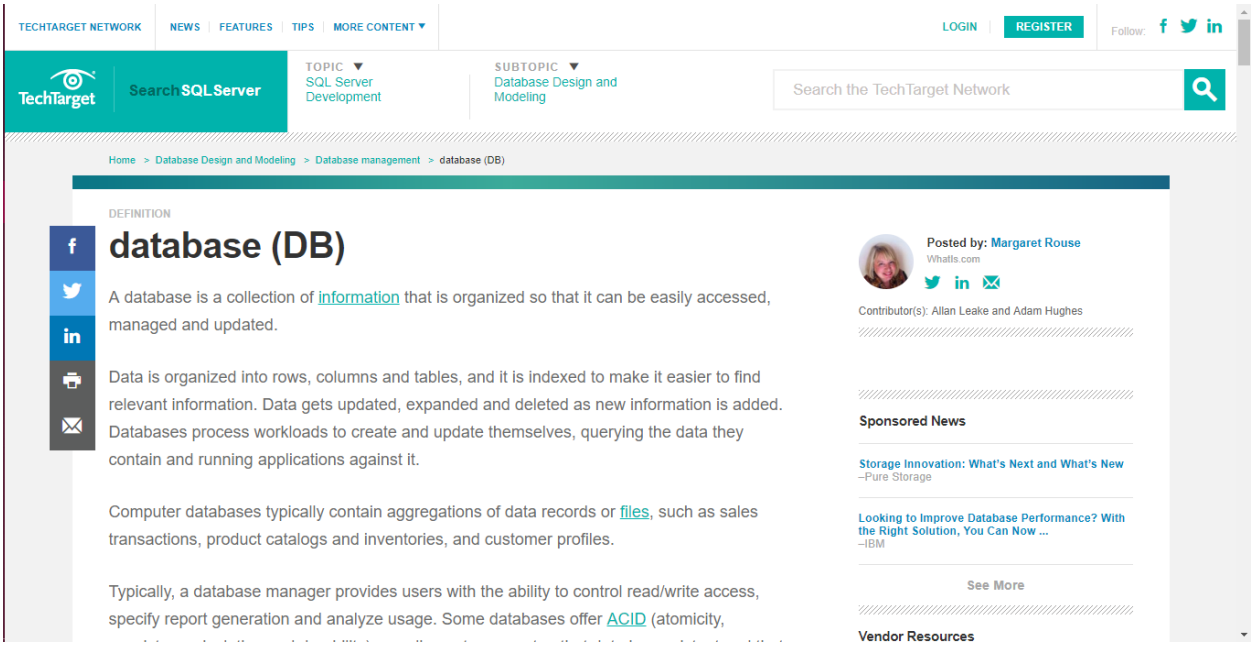
- Websites:
    - https://searchdatamanagement.techtarget.com/definition/RDBMS-relational-database-management-system



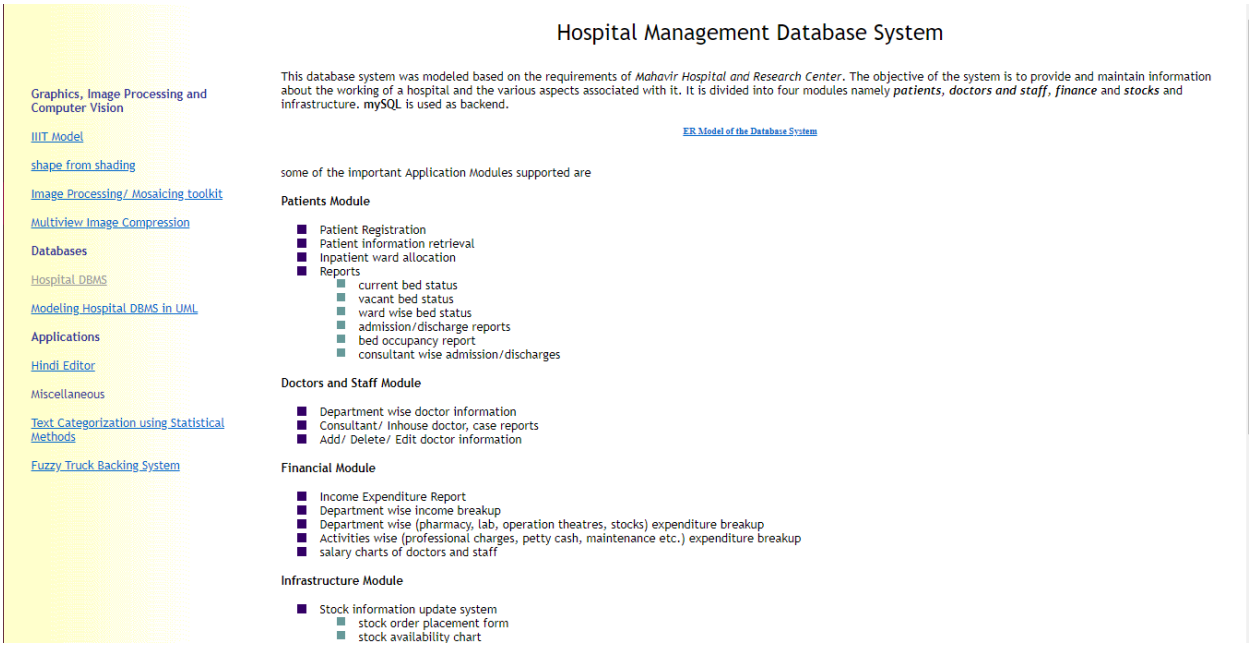
- https://www.smartdraw.com/entity-relationship-diagram/



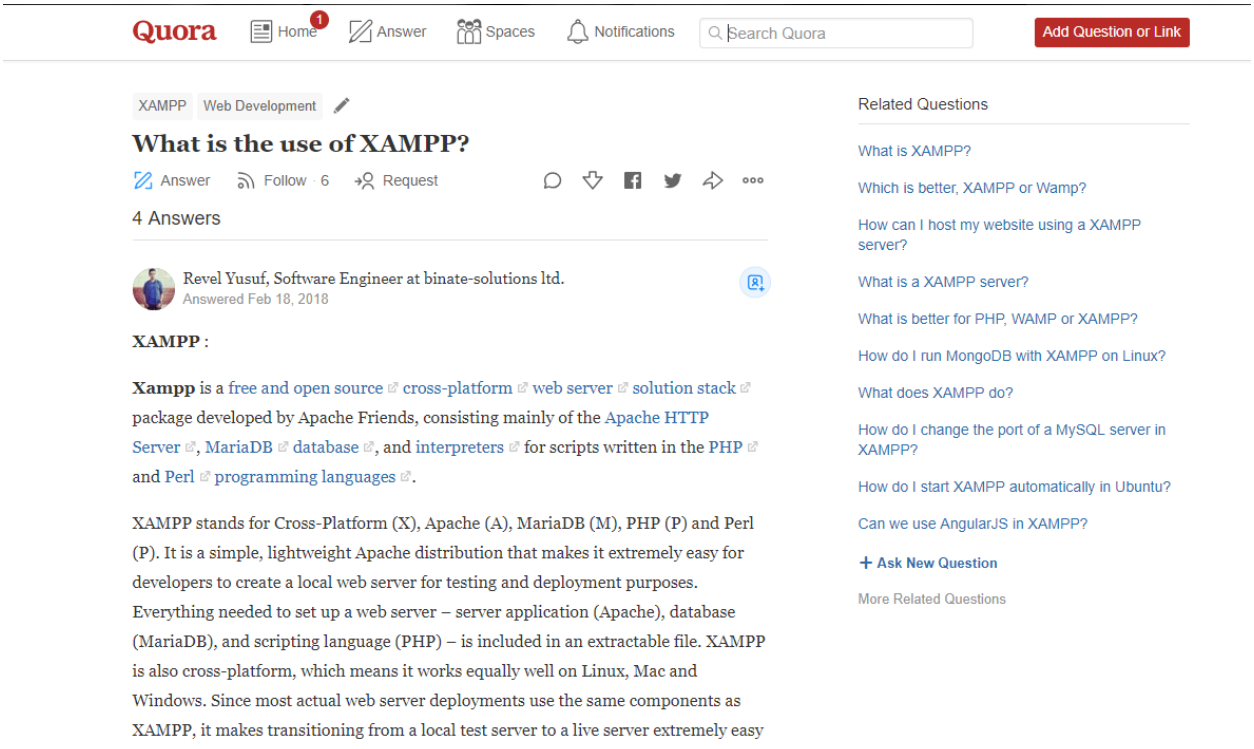
- https://searchsqlserver.techtarget.com/definition/database



- https://web.iiit.ac.in/~kamisetty/projects/hospital.html



- https://www.quora.com/What-is-the-use-of-XAMPP



Conclusion:

After the completion of this task, the concept to create the database for any organization became clear. Many researches were made, and websites were iterated so that a frictionless database management system could be created. Time had to be given while making the ER-Diagram and relational diagram because here we had to find the link between two tables. With the help of medias such as journals, websites and books, first the relevant topic was covered up. The main aim of the project was to cover up possible entities and link them in a proper and efficient manner so that the possible queries could be fulfilled.

While starting the research, basic concept was only implemented but as soon as the research was started many more concepts became clear. I learned many topics from the research in MySQL commands and queries. The source of all this information were the journals and books listed above. By skimming through these sources, ample of knowledge was gained about how to approach a database system. Concepts of primary key, foreign key was essential requirement of this coursework. Primary keys were essential to uniquely identify a detail whereas in the aspect of showing links between attributes, foreign key was essential. Foreign keys are primary key of a table which are referenced in another table such that link was established between two entities. Initially, these concepts were very vague but as the books and journals were read, the concept was clear and made life easy to create this database.

Finally, this project was created with lots of research and hard work. The coursework was not that tough but documentation was time consuming. In documentation, all the possible queries are listed and by using the SQL code they are fulfilled as well. All the entities are described form their creation to the addition of the data in the table. Search engines are provided from which I achieved great help. Books and journals are also mentioned.

Thus, a small hospital management system was created with help of some tools such as XAMPP and notepad. This project can be helpful to the lecturers and the students who want to learn database management system. It is the perfect example for small database management system of the hospital.

References

Anon., 2019. *What is an ER diagram (ERD)?.* [Online]
Available at: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/
[Accessed 17 3 2019].

Chauhan, S., 2012. Difference between Primary Key and Foreign Key. *Difference between Primary Key and Foreign Key*, 26 Sep, pp. https://www.dotnettricks.com/learn/sqlserver/difference-between-primary-key-and-foreign-key.

Raza, M., 2018. *Introduction to Database Management Systems (DBMS).* [Online]
Available at: https://www.bmc.com/blogs/dbms-database-management-systems/
[Accessed 20 3 2019].

Rouse, M., 2005. *RDBMS (relational database management system).* [Online]
Available at: https://searchdatamanagement.techtarget.com/definition/RDBMS-relational-database-management-system
[Accessed 15 03 2019].