**DBMS LAB FILE**



**NOUSHAD AHMED**

**2K19/CO/266**

**A4**

INDEX

|  |  |  |
| --- | --- | --- |
| s.no | Title | date |
| 1. | Introduction To SQL, DATABASE, and DATABASE MANAGEMENT SYSTEM. |  |
| 2. | Introduction to various software of database management system and database languages. |  |
| 3. | Introduction to ER diagram, symbol table. |  |
| 4. | Introduction to different types of constraint in SQL. |  |
| 5. | Case Study 1: Hospital Management System. |  |
| 6. | Entity Relationship Diagram, Creating Entity Tables and Relationship Tables. |  |
| 7. | Implementing DDL and DML statements |  |
| 8. | Performing Simple Queries |  |
| 9. | Implementation of joins. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**EXPERIMENT -1**

**AIM:** introduction to SQL , DATABASE, and DATABASE MANAGEMENT SYSTEM.

**THEORY:**

**SQL**

SQL is a standard language for accessing and manipulating databases.

**What is SQL?**

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

**What Can SQL do?**

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

**Note:** Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard!

**Using SQL in Your Web Site**

To build a web site that shows data from a database, you will need:

* An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
* To use a server-side scripting language, like PHP or ASP
* To use SQL to get the data you want
* To use HTML / CSS to style the page

**Some of The Most Important SQL Commands**

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

## The SQL SELECT Statement

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

### SELECT Syntax

SELECT column1, column2, ...  
FROM table\_name;

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT \* FROM table\_name;

## The SQL UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

### UPDATE Syntax

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

## The SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

### DELETE Syntax

DELETE FROM table\_name WHERE condition;

**Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!.

## The SQL CREATE DATABASE Statement

The CREATE DATABASE statement is used to create a new SQL database.

### Syntax

CREATE DATABASE databasename;

## CREATE DATABASE Example

The following SQL statement creates a database called "testDB":

### Example

CREATE DATABASE testDB;

**Tip:** Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases with the following SQL command: SHOW DATABASES;

## The SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

### Syntax

CREATE TABLE table\_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
   ....  
);

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

**Tip:** For an overview of the available data types, go to our complete [Data Types Reference](https://www.w3schools.com/sql/sql_datatypes.asp).

## SQL CREATE TABLE Example

The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

### Example

CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)   
);

## SQL ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

## ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

ALTER TABLE table\_name  
ADD column\_name datatype;

The following SQL adds an "Email" column to the "Customers" table:

### Example

ALTER TABLE Customers  
ADD Email varchar(255);

## SQL CREATE INDEX Statement

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

**Note:** Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So, only create indexes on columns that will be frequently searched against.

### CREATE INDEX Syntax

Creates an index on a table. Duplicate values are allowed:

CREATE INDEX index\_name  
ON table\_name (column1, column2, ...);

**DATABASE:** A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a [database management system (DBMS)](https://www.oracle.com/in/database/what-is-database/#WhatIsDBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

**DATABASE MANAGEMENT SYSTEM:** A database typically requires a comprehensive database software program known as a database management system (DBMS). A DBMS serves as an interface between the database and its end users or programs, allowing users to retrieve, update, and manage how the information is organized and optimized. A DBMS also facilitates oversight and control of databases, enabling a variety of administrative operations such as performance monitoring, tuning, and backup and recovery.

Some examples of popular database software or DBMSs include MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database, and dBASE.

**Data definition :** Data definition commands are used to create, modify and remove database objects such as schemas, tables, views, indexes etc.

Common Data Definition commands −

## Create

The main use of create command is to create a new table in database. It has a predefined syntax in which we specify the columns and their respective data types.

### syntax

CREATE TABLE <TABLE NAME>

( <COLUMN NAME>  <DATA TYPE>,

<COLUMN NAME>  <DATA TYPE>,

<COLUMN NAME>  <DATA TYPE>,

<COLUMN NAME>  <DATA TYPE>

);

## Example

Create a student table with columns student name and roll number.

CREATE TABLE STUDENT

(STUDENT\_NAME VARCHAR(30),

ROLL\_NUMBER INT

);

## Alter

An existing database object can be modified using the alter command. Alter command can do following changes to any table-

* Add new columns.
* Add new integrity constraints.
* Modify existing columns.
* Drop integrity constraints.

### Syntax

General Syntax of the ALTER command is mentioned below −

For adding a new column

ALTER TABLE <table\_name>  ADD  <column\_name>

 For renaming a table

ALTER TABLE <table\_name>  RENAME To <new\_table\_name >

For modifying a column

ALTER TABLE <table\_name> MODIFY <column\_name > <data type >

For deleting a column

ALTER TABLE <table\_name> DROP COLUMN <column\_name>

## Drop

This command can delete an index, table or view. Basically, any component from a relational database management system can be removed using the  Drop command. Once the object is dropped, it cannot be reused.  
  
The general syntax of drop command is as follows −

DROP TABLE <table\_name>;

DROP DATABASE <database\_name>;

DROP TABLE <index\_name>;

## Truncate

Using the truncate command, all the records in a database are deleted, but the database structure is maintained.

### syntax

TRUNCATE TABLE <table name>

## Comment

This command is used to add comments to the data dictionary.

### syntax

* Single line comments:  use ‘ --‘ before any text.
* Multiline comments:  /\* comments in between   \*/

## Rename

The rename command renames an object

### Syntax

Rename <old name> to <new name>

**EXPERIMENT -2**

**AIM:** Introduction to various software of database management system and database language.

**THEORY:**

**DBMS Software:**

* MySQL
* Microsoft Access
* Oracle
* PostgreSQL
* dBASE
* FoxPro
* SQLite
* IBM DB2
* LibreOffice Base
* MariaDB
* Microsoft SQL Server etc.

**MySql**

MySQL is a database management system that allows you to manage relational databases. It is open source software backed by Oracle. It means you can use MySQL without paying a dime. Also, if you want, you can change its source code to suit your needs.

Even though MySQL is open source software, you can buy a commercial license version from Oracle to get premium support services.

MySQL is pretty easy to master in comparison with other database software like Oracle Database, or Microsoft SQL Server.

MySQL can run on various platforms UNIX, Linux, Windows, etc. You can install it on a server or even in a desktop. Besides, MySQL is reliable, scalable, and fast.

The official way to pronounce MySQL is *My Ess Que Ell, not My Sequel.* However, you can pronounce it whatever you like, who cares?

If you develop websites or web applications, MySQL is a good choice. MySQL is an essential component of the LAMP stack, which includes Linux, Apache, MySQL, and PHP.

**Oracle**

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a [server](https://docs.oracle.com/cd/B19306_01/server.102/b14220/glossary.htm#i432724) reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

The database has logical structures and physical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

**Microsoft Access:**

**Microsoft Access** is a [database management system](https://en.wikipedia.org/wiki/Database_management_system)(DBMS) from [Microsoft](https://en.wikipedia.org/wiki/Microsoft) that combines the [relational](https://en.wikipedia.org/wiki/Relational_database) [Microsoft Jet Database Engine](https://en.wikipedia.org/wiki/Microsoft_Jet_Database_Engine) with a [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) and software-development tools. It is a member of the [Microsoft 365](https://en.wikipedia.org/wiki/Microsoft_365) suite of applications, included in the Professional and higher editions or sold separately.

Microsoft Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to [data](https://en.wikipedia.org/wiki/Data) stored in other applications and databases.[[3]](https://en.wikipedia.org/wiki/Microsoft_Access#cite_note-ms-import-3)

[Software developers](https://en.wikipedia.org/wiki/Software_developer), [data architects](https://en.wikipedia.org/wiki/Data_architect) and [power users](https://en.wikipedia.org/wiki/Power_user) can use Microsoft Access to develop [application software](https://en.wikipedia.org/wiki/Application_software). Like other Microsoft Office applications, Access is supported by [Visual Basic for Applications](https://en.wikipedia.org/wiki/Visual_Basic_for_Applications) (VBA), an [object-based](https://en.wikipedia.org/wiki/Object-based) programming language that can reference a variety of objects including the legacy DAO (Data Access Objects), [ActiveX](https://en.wikipedia.org/wiki/ActiveX) Data Objects, and many other ActiveX components. Visual objects used in forms and reports expose their methods and properties in the VBA programming environment, and VBA code modules may declare and call Windows [operating system](https://en.wikipedia.org/wiki/Operating_system) operations. It doesn't have a web version.

**PostgreSql:**

PostgreSQL (pronounced as **post-gress-Q-L**) is an open source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge.

### A Brief History of PostgreSQL

PostgreSQL, originally called Postgres, was created at UCB by a computer science professor named Michael Stonebraker. Stonebraker started Postgres in 1986 as a follow-up project to its predecessor, Ingres, now owned by Computer Associates.

* **1977-1985** − A project called INGRES was developed.
  + Proof-of-concept for relational databases
  + Established the company Ingres in 1980
  + Bought by Computer Associates in 1994
* **1986-1994** − POSTGRES
  + Development of the concepts in INGRES with a focus on object orientation and the query language - Quel
  + The code base of INGRES was not used as a basis for POSTGRES
  + Commercialized as Illustra (bought by Informix, bought by IBM)
* **1994-1995** − Postgres95
  + Support for SQL was added in 1994
  + Released as Postgres95 in 1995
  + Re-released as PostgreSQL 6.0 in 1996
  + Establishment of the PostgreSQL Global Development Team

## Key Features of PostgreSQL

PostgreSQL runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It supports text, images, sounds, and video, and includes programming interfaces for C / C++, Java, Perl, Python, Ruby, Tcl and Open Database Connectivity (ODBC).

PostgreSQL supports a large part of the SQL standard and offers many modern features including the following −

* Complex SQL queries
* SQL Sub-selects
* Foreign keys
* Trigger
* Views
* Transactions
* Multiversion concurrency control (MVCC)
* Streaming Replication (as of 9.0)
* Hot Standby (as of 9.0)

You can check official documentation of PostgreSQL to understand the above-mentioned features. PostgreSQL can be extended by the user in many ways. For example by adding new −

* Data types
* Functions
* Operators
* Aggregate functions
* Index methods

**Dbase:**

DBase is a microcomputer database management system (DBMS) that runs on a Windows platform. DBase is unique in that it allows for the hassle-free production of a wide variety of applications, including middleware applications, Web apps hosted on Windows servers and Windows rich client applications.   
  
DBase is designed to manipulate relational databases. It is a versatile third-generation language with non-procedural capability and is a very good debugger.

**SQLite:**

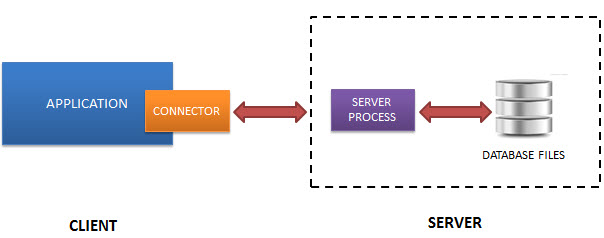
SQLite is a software library that provides a relational database management system. The lite in SQLite means lightweight in terms of setup, database administration, and required resources.

SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional.

### Serverless

Normally, an RDBMS such as MySQL, PostgreSQL, etc., requires a separate server process to operate. The applications that want to access the database server use TCP/IP protocol to send and receive requests. This is called client/server architecture.

The following diagram illustrates the RDBMS client/server architecture:



SQLite does NOT work this way.

SQLite does NOT require a server to run.

SQLite database is integrated with the application that accesses the database. The applications interact with the SQLite database read and write directly from the database files stored on disk.

**EXPERIMENT -3**

**AIM:** Introduction to ER diagram, Symbol table.

**THEORY:**

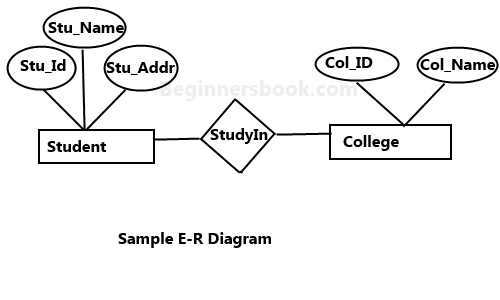
**ER DIAGRAM:**

An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

**A simple ER Diagram:**

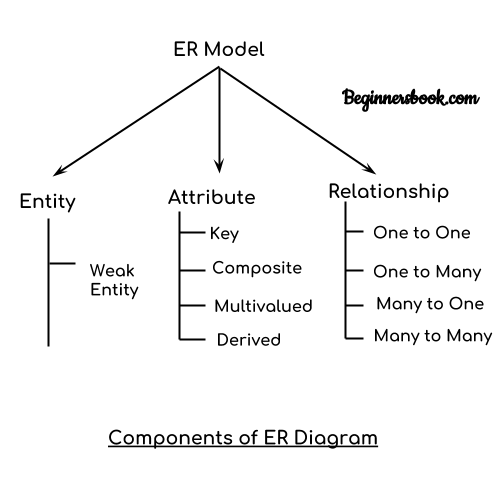
In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu\_Id, Stu\_Name & Stu\_Addr and College entity has attributes such as Col\_ID & Col\_Name.



Here are the geometric shapes and their meaning in an E-R Diagram.

**Rectangle**: Represents Entity sets.  
**Ellipses**: Attributes  
**Diamonds**: Relationship Set  
**Lines**: They link attributes to Entity Sets and Entity sets to Relationship Set  
**Double Ellipses:** Multivalued Attributes  
**Dashed Ellipses**: Derived Attributes  
**Double Rectangles**: Weak Entity Sets  
**Double Lines**: Total participation of an entity in a relationship set

## Components of a ER Diagram:



**ENTITY:**

Entity in DBMS can be a real-world object with an existence, For example, in a **College**database, the entities can be Professor, Students, Courses, etc.

Entities has attributes, which can be considered as properties describing it, for example, for Professor entity, the attributes are **Professor\_Name, Professor\_Address, Professor\_Salary,** etc. The attribute value gets stored in the database.

**ATTRIBUTE:**

Attributes are the properties which define the entity type. For example, Roll\_No, Name, DOB, Age, Address, Mobile\_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval. The attribute which uniquely identifies each entity in the entity set is called key attribute.

**REALTIONSHIP:**

A relationship in a DBMS, is primarily the way two or more data sets are linked. Relationships allow the datasets to share and store the data in separate tables. They also help link disparate data with each other.

**SYMBOL TABLE:**

Symbol table is an important data structure created and maintained by compilers in order to store information about the occurrence of various entities such as variable names, function names, objects, classes, interfaces, etc. Symbol table is used by both the analysis and the synthesis parts of a compiler.

A symbol table may serve the following purposes depending upon the language in hand:

* To store the names of all entities in a structured form at one place.
* To verify if a variable has been declared.
* To implement type checking, by verifying assignments and expressions in the source code are semantically correct.
* To determine the scope of a name (scope resolution).

A symbol table is simply a table which can be either linear or a hash table. It maintains an entry for each name in the following format:

<symbol name, type, attribute>

For example, if a symbol table has to store information about the following variable declaration:

static int interest;

then it should store the entry such as:

<interest, int, static>

The attribute clause contains the entries related to the name.

## Implementation:

If a compiler is to handle a small amount of data, then the symbol table can be implemented as an unordered list, which is easy to code, but it is only suitable for small tables only. A symbol table can be implemented in one of the following ways:

* Linear (sorted or unsorted) list
* Binary Search Tree
* Hash table

Among all, symbol tables are mostly implemented as hash tables, where the source code symbol itself is treated as a key for the hash function and the return value is the information about the symbol.

## Operations:

A symbol table, either linear or hash, should provide the following operations.

### **insert()**

This operation is more frequently used by analysis phase, i.e., the first half of the compiler where tokens are identified and names are stored in the table. This operation is used to add information in the symbol table about unique names occurring in the source code. The format or structure in which the names are stored depends upon the compiler in hand.

An attribute for a symbol in the source code is the information associated with that symbol. This information contains the value, state, scope, and type about the symbol. The insert() function takes the symbol and its attributes as arguments and stores the information in the symbol table.

For example:

int a;

should be processed by the compiler as:

insert(a, int);

### **lookup()**

lookup() operation is used to search a name in the symbol table to determine:

* if the symbol exists in the table.
* if it is declared before it is being used.
* if the name is used in the scope.
* if the symbol is initialized.
* if the symbol declared multiple times.

The format of lookup() function varies according to the programming language. The basic format should match the following:

lookup(symbol)

This method returns 0 (zero) if the symbol does not exist in the symbol table. If the symbol exists in the symbol table, it returns its attributes stored in the

Table.

**EXPERIMENT-4**

**AIM:** Introduction to different types of constraints in SQL.

**THEORY:**

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraints can be divided into the following two types,

1. **Column level constraints:** Limits only column data.
2. **Table level constraints:** Limits whole table data.

Constraints are used to make sure that the integrity of data is maintained in the database. Following are the most used constraints that can be applied to a table.

* NOT NULL
* UNIQUE
* PRIMARY KEY
* FOREIGN KEY
* CHECK
* DEFAULT

**NOT NULL Constraint**

By default, a [column](https://www.studytonight.com/dbms/rdbms-concept.php) can hold NULL values. If you do not want a column to have a NULL value, use the NOT NULL constraint.

* It restricts a column from having a NULL value.
* We use [ALTER](https://www.studytonight.com/dbms/alter-query.php) statement and [MODIFY](https://www.studytonight.com/dbms/alter-query.php) statement to specify this constraint.

One important point to note about this constraint is that it cannot be defined at table level.

Example using NOT NULL constraint:

CREATE TABLE Student

( s\_id int NOT NULL,

name varchar(60),

age int

);

The above query will declare that the **s\_id** field of **Student** table will not take NULL value.

If you wish to alter the table after it has been created, then we can use the ALTER command for it:

ALTER TABLE Student

MODIFY s\_id int NOT NULL;

**UNIQUE Constraint**

It ensures that a column will only have unique values. A UNIQUE constraint field cannot have any duplicate data.

* It prevents two records from having identical values in a column
* We use [ALTER](https://www.studytonight.com/dbms/alter-query.php) statement and [MODIFY](https://www.studytonight.com/dbms/alter-query.php) statement to specify this constraint.

Example of UNIQUE Constraint:

Here we have a simple CREATE query to create a table, which will have a column **s\_id** with unique values.

CREATE TABLE Student

( s\_id int NOT NULL,

name varchar(60),

age int NOT NULL UNIQUE

);

The above query will declare that the **s\_id** field of **Student** table will only have unique values and wont take NULL value.

If you wish to alter the table after it has been created, then we can use the ALTER command for it:

ALTER TABLE Student

MODIFY age INT NOT NULL UNIQUE;

The above query specifies that **s\_id** field of **Student** table will only have unique value.

**Primary Key Constraint**

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

PRIMARY KEY constraint at Table Level

CREATE table Student

( s\_id int PRIMARY KEY,

Name varchar(60) NOT NULL,

Age int);

The above command will creates a PRIMARY KEY on the s\_id.

PRIMARY KEY constraint at Column Level

ALTER table Student

ADD PRIMARY KEY (s\_id);

The above command will creates a PRIMARY KEY on the s\_id.

**Foreign Key Constraint**

[Foreign Key](https://www.studytonight.com/dbms/database-key.php) is used to relate two tables. The relationship between the two tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

* This is also called a referencing key.
* We use [ALTER](https://www.studytonight.com/dbms/alter-query.php) statement and [ADD](https://www.studytonight.com/dbms/alter-query.php) statement to specify this constraint.

To understand FOREIGN KEY, let's see its use, with help of the below tables:

**Customer\_Detail** Table

|  |  |  |
| --- | --- | --- |
| **c\_id** | **Customer\_Name** | **address** |
| 101 | Adam | Noida |
| 102 | Alex | Delhi |
| 103 | Stuart | Rohtak |

**Order\_Detail** Table

|  |  |  |
| --- | --- | --- |
| **Order\_id** | **Order\_Name** | **c\_id** |
| 10 | Order1 | 101 |
| 11 | Order2 | 103 |
| 12 | Order3 | 102 |

In **Customer\_Detail** table, **c\_id** is the primary key which is set as foreign key in **Order\_Detail** table. The value that is entered in **c\_id** which is set as foreign key in **Order\_Detail** table must be present in **Customer\_Detail** table where it is set as primary key. This prevents invalid data to be inserted into **c\_id** column of **Order\_Detail** table.

If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data.

FOREIGN KEY constraint at Table Level

CREATE table Order\_Detail(

order\_id int PRIMARY KEY,

order\_name varchar(60) NOT NULL,

c\_id int FOREIGN KEY REFERENCES Customer\_Detail(c\_id)

);

In this query, **c\_id** in table Order\_Detail is made as foriegn key, which is a reference of **c\_id** column in Customer\_Detail table.

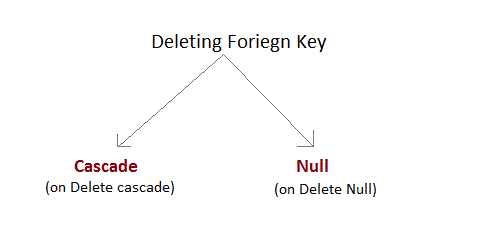
FOREIGN KEY constraint at Column Level

ALTER table Order\_Detail

ADD FOREIGN KEY (c\_id) REFERENCES Customer\_Detail(c\_id);

Behaviour of Foriegn Key Column on Delete

There are two ways to maintin the integrity of data in Child table, when a particular record is deleted in the main table. When two tables are connected with Foriegn key, and certain data in the main table is deleted, for which a record exits in the child table, then we must have some mechanism to save the integrity of data in the child table.



1. **On Delete Cascade :** This will remove the record from child table, if that value of foriegn key is deleted from the main table.
2. **On Delete Null :** This will set all the values in that record of child table as NULL, for which the value of foriegn key is deleted from the main table.
3. If we don't use any of the above, then we cannot delete data from the main table for which data in child table exists. We will get an error if we try to do so.

ERROR : Record in child table exist

**CHECK Constraint**

**CHECK** constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

Using CHECK constraint at Table Level

CREATE table Student(

s\_id int NOT NULL CHECK(s\_id > 0),

Name varchar(60) NOT NULL,

Age int

);

The above query will restrict the **s\_id** value to be greater than zero.

Using CHECK constraint at Column Level

ALTER table Student ADD CHECK(s\_id > 0);

**Experiment – 5**

### CASE STUDY:

**HOSPITAL MANAGEMENT SYSTEM**

**Entity Tables:**

* DEPARTMENT TABLE: It will maintain information about a particular department.
  + DEP\_ID: The id of the particular department.
  + NAME: The name of the department
  + HOD: The name of the hod of that particular department.
  + HOD\_ID: The id of the particular hod of that department.
* DOCTOR TABLE: It will maintain information about each doctor.
  + DOC\_ID: The id of the particular doctor.
  + NAME: The name of the doctor.
  + DEPARTMENT: The department for which the doctor works for.
  + CONTACT\_NO: The contact number of the doctor.
  + ROOM\_NO: The room no where the doctor sits.
* PATIENT TABLE: It will maintain information about the patients.
  + PATIENT\_ID: The id of the particular patient.
  + PATIENT\_NAME: The name of the patient.
  + ADDRESS: The address of the patient.
  + CONTACT NO: The contact number of the patient.
  + STATUS: It stores whether the patient is discharged or admitted.
* NURSE TABLE: It will maintain information about the patients.
  + NURSE\_ID: The id of the particular nurse.
  + NAME: The name of the nurse.
  + CONTACT NO: The contact number of the nurse.
  + SALARY: The salary of the nurse.
* BILL TABLE: It will maintain information about the patients.
  + BILL\_ID: The id of the bill.
  + BILL\_DATE: The date when the bill was generated.
  + AMOUNT: The payable bill amount.
* MEDICINE TABLE: It will maintain information about the patients.
  + MEDICINE\_CODE: The code of the medicine.
  + NAME: The name of the medicine.
  + QUANTITY: The quantity of medicine stocked.
  + PRICE: The price of the medicine

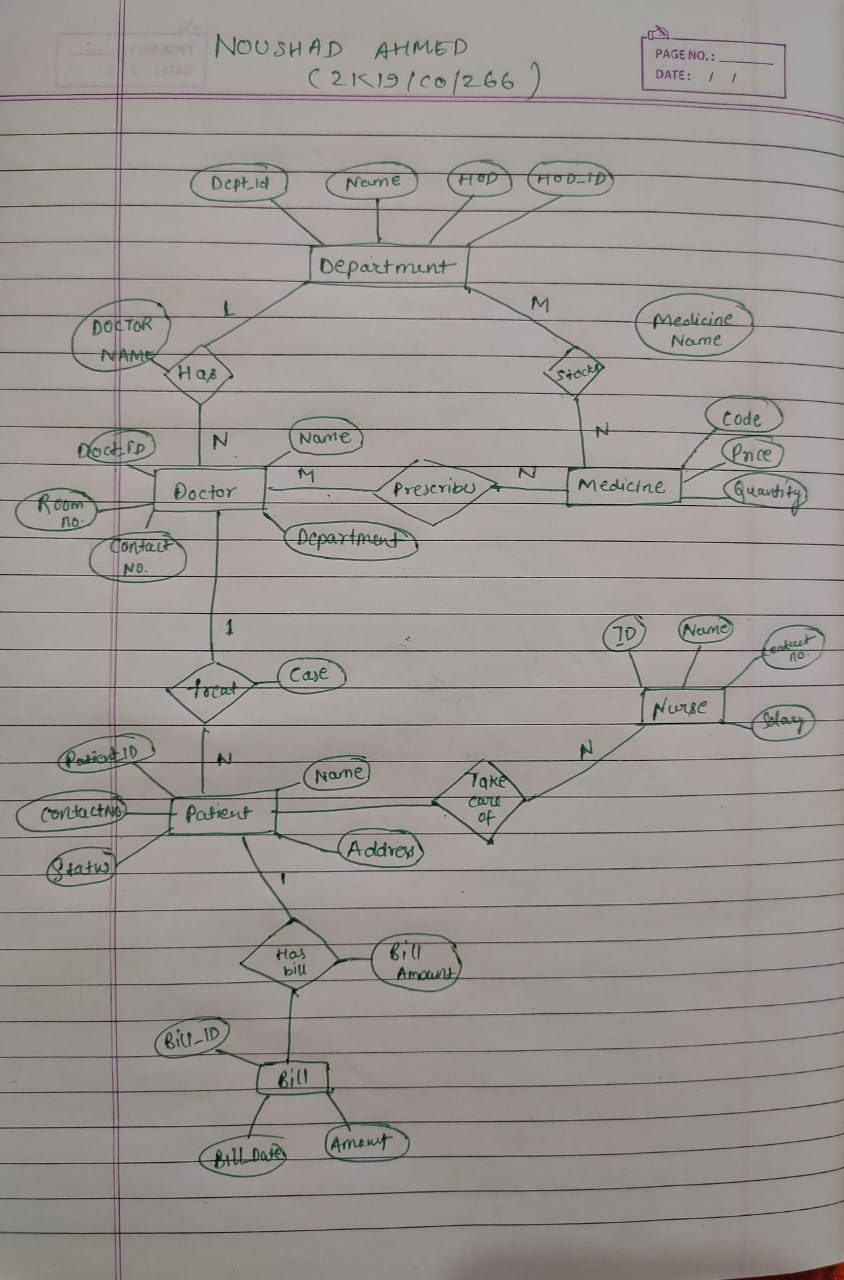
### ASSUMPTIONS WHILE CREATING THE RELATIONSHIP TABLE:

* A department can have many doctors but a doctor can work only for one department.
* A nurse can take care of many patients but a patient can have only one nurse.
* A Medicine can be used by multiple departments and a department can use multiple medicines.
* A doctor can treat a lot of patients but a patient can only be treated by a single doctor.
* A patient can have multiple bills but a bill belongs to a single patient.

**EXPERIMENT-6**

**AIM:** Constructing ER Diagram, entity tables and relationship tables.

**ER DIAGRAM:**



**ENTITY TABLES:**

**DEPARTMENT TABLE**:DEPT\_ID, NAME, HOD and HOD\_ID

**DOCTOR TABLE**: DOC\_ID, NAME, DEPARTMENT, CONTACT\_NO and ROOM\_NO.

**PATIENT TABLE**: PATIENT\_ID, PATIENT\_NAME, ADDRESS, CONTACT\_NO

and STATUS

**NURSE TABLE**: NURSE\_ID, NAME, CONTACT\_NO and SALARY. **MEDICINE TABLE**: MEDICINE\_CODE, PRICE, QUANTITY and NAME. **BILL TABLE**: BILL\_ID, BILL\_DATE and AMOUNT.

##### RELATIONSHIP TABLES:

* Department : Doctor (1:N)
* Doctor : Patient(1:N)
* Doctor :Medicine (M:N)
* Patient : Bill (1:N)
* Patient : Nurse (N:1)
* Department : Medicine (M:N)

**EXPERIMENT-7**

**AIM:** Implementing DDL and DML statements

##### THEORY:

**DDL STATEMENTS**

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

##### CREATE TABLE:

1. **CREATE TABLE:** CREATE TABLE statement is used to create a new table in database



1. **ALTER TABLE:** The ALTER TABLE statement is used to add, delete or modify columns in an existing table. It is also used to add and drop various constraints on existing table.



1. **DROP TABLE:** The DROP TABLE command is used to remove a definition and all data, indexes, triggers, constraints and permissions specifications for that table.

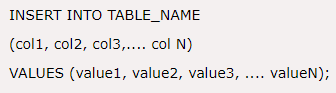


1. **TRUNCATE:** This command is used to delete complete data from and existing table



##### DML STATEMENTS

* 1. **INSERT:** The Insert statement is used to insert data into table.



* 1. **DELETE:** The Delete statement is used to delete records from database table.



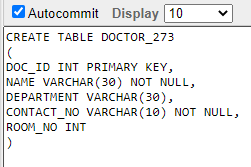
* 1. **UPDATE:** The Update statement is used to update data in an existing table.



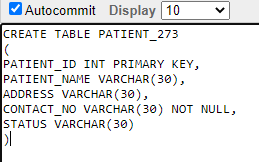
##### IMPLEMENTING DDL STATEMENTS:

1. **CREATE TABLE:**

Doctor Table:

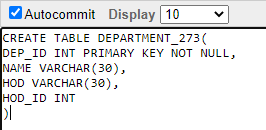


**Patient Table**

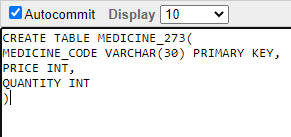




Department Table:

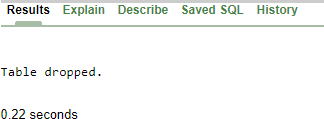


Medicine Table:



1. **DROP TABLE:**



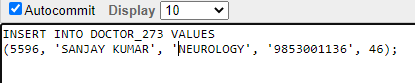


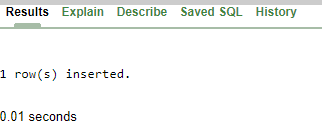
1. **ALTER TABLE:**



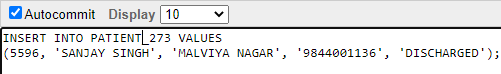
##### Implementing DML Statements:

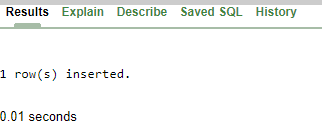
* 1. **INSERT INTO**: DOCTOR TABLE:



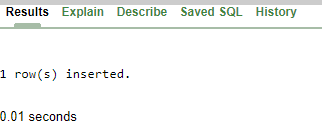
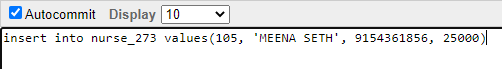


PATIENT TABLE:

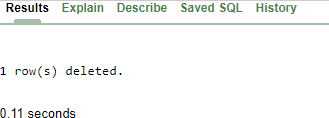
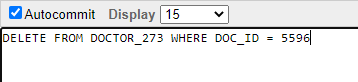




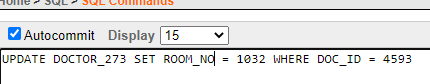
NURSE TABLE:



* 1. **DELETE**: DOCTOR



* 1. **UPDATE:**



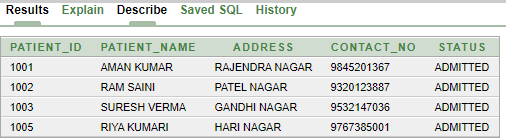
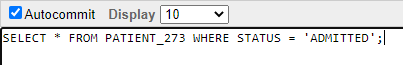
**EXPERIMENT-8**

**Aim:** To implement simple queries **Theory:**

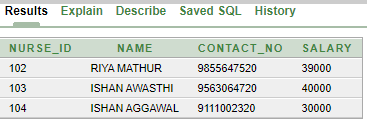
The SELECT statement is used to fetch the data from the database table which returns the result in the form of the table. The result table is also known as result set.

**Performing Simple Queries**

* Select those patients from patient table who are currently admitted.

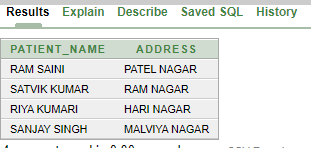


* Select Only Those Values From Nurse Table Where Nurse\_Id Is More Than Or Equal To 102 And Salary Is More Than 30000

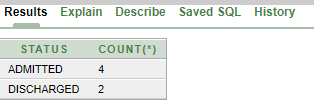
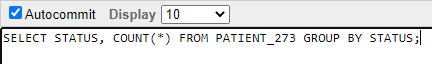


* Select Only Those Patient Name And Address From Patient Table Where Patient Name Has A Pattern Like “I”

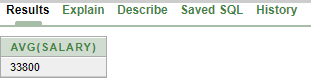




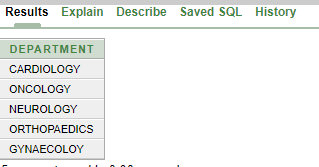
* Find the Number Of Patients Who Are Discharged And Admitted From Patient Table



* Display Average Salary From Nurse Table



* Retrieve All Distinct Departments From Doctor Table



**LEARNINGS AND FINDINGS**

* We learnt how to use various Simple Queries using SELECT command
* We learnt how to query a database to extract the relevant data from the database.

**RESULT:**

* Various Simple Queries and syntax are successfully implemented and understood on database in the experiment.

**Experiment-9**

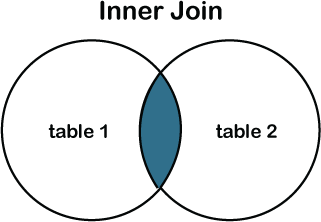
**Aim:** To implement the joins

**Theory:**

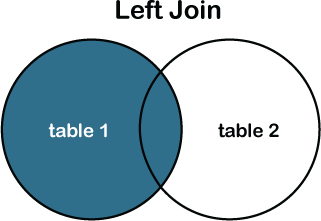
## A join clause is used to combine rows from two or more tables, based on a related column between them.

Different types of joins are:

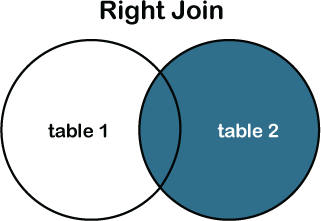
* + Inner Join: return records that have matching values in both the tables



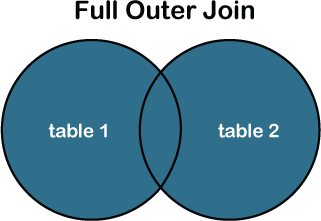
* + Left Outer Join: Return all the records from the left table and the matching records from the right table.



* + Right Outer Join: Return all the record from the right table and the matching records from the left table.



* + Full Outer Join: Return all the records when there is a match between the left table or the right table.

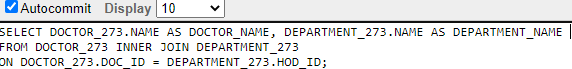


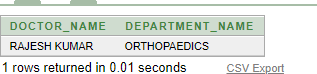
**INNER JOIN**

The syntax for the inner join in SQL is:

**Syntax:**

SELECT columns FROM table1 INNER JOIN table2 ON table1.column = table2.column;



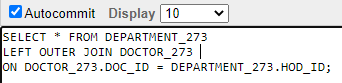


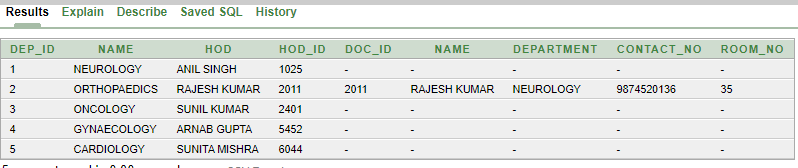
**LEFT OUTER JOIN**

The syntax for the left outer join in SQL is:

**Syntax:**

SELECT columns FROM table1 LEFT OUTER JOIN table2 ON table1.column = table2.column;



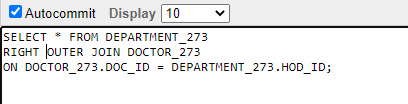


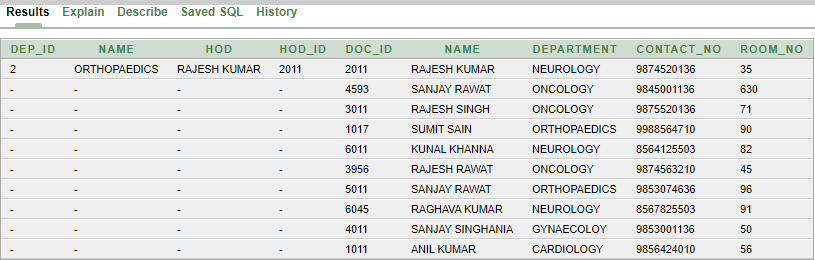
**RIGHT OUTER JOIN**

The syntax for the right outer join in SQL is:

**Syntax:**

SELECT columns FROM table1 RIGHT OUTER JOIN table2 ON table1.column = table2.column;

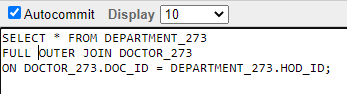




**FULL OUTER JOIN**

The syntax for the right outer join in SQL is:

**Syntax:**

SELECT columns FROM table1 FULL OUTER JOIN table2 ON table1.column = table2.column;