ARMY INSTITUTE OF TECHNOLOGY

LAB MANUAL

Name of Subject	**DBMS Lab**
Department	**IT**
Subject Code	**214456**
TW/OR/PR	**TW/PR**
Marks	**25/25**
Course (Pattern)	**2019**
Year	**SE**
Semester	**IV**

List of Experiments

(As given by SPPU)

Sr. No	Title	Page No
1	Study of MySQL Open source software.	
2	Install and configure client and server of MySQL.	
3	Study of SQLite: What is SQLite? Uses of Sqlite. Building and installing SQLite.	
4	Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system.	
5	Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them	
6	Create Table with primary key and foreign key constraints. a. Alter table with add n modify b. Drop table	
7	Perform following SQL queries on the database created in assignment 1. • Implementation of relational operators in SQL • Boolean operators and pattern matching • Arithmetic operations and built in functions • Group functions • Processing Date and Time functions • Complex queries and set operators	
8	Execute DDL/DML statements which demonstrate the use of views. Update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables.	
9	Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.	
10	Write and execute suitable database triggers .Consider row level and statement level triggers.	
11	Write a PL/SQL block to implement all types of cursor.	

Pre-requisite for Lab

Companion Courses	
Code	Subject
214452	Database Management System
Prerequisite Courses/Lab	
Code	Subject
214443	Data Structures
214441	Discrete Structure
214454	Software Engineering

Learning Objectives

Sr. No	Objective
1	Understand the fundamental concepts of database management. These concepts include aspects of database design, database languages, and database-system implementation.
2	To provide a strong formal foundation in database concepts, recent technologies and best industry practices.
3	To give systematic database design approaches covering conceptual design, logical design and an overview of physical design.
4	To learn the SQL database system.
5	To learn and understand various Database Architectures and its use for application development.
6	To program PL/SQL including stored procedures, stored functions, cursors and packages.

Precautions / Lab Safety Do's and Don'ts/ Lab Etiquettes

Sr. No	Details
1	The use of personal audio or video equipment is prohibited in the laboratory
2	Keep work area neat and free of any unnecessary objects.
3	Always perform the experiments as directed by your instructor.
4	Never work in the laboratory without the supervision of an instructor.
5	Mobile Phones are strictly prohibited.
6	Eatables are not allowed in Lab

Evaluation Guidelines (Rubrics)

(Similar guidelines pertaining to each subject (lab) shall be used to evaluate **each experiment** performed in the lab)

Grade	Poor	Average	Good	Outstanding
Marks	0-3	4-5	6-8	9-10

Criteria	**Grade**		**Marks**
Set-up and Equipment Care	Poor (0-3)	Set-up of equipment is not accurate, help is required with several major details	
	Average (4-5)	Set-up of equipment is generally workable with several details that need refinement	
	Good (6-8)	Set-up of equipment is generally accurate with 1 or 2 small details that need refinement	
	Outstanding (9-10)	All equipments accurately placed	
Following Procedure	Poor (0-3)	Lacks appropriate knowledge of the lab procedures. Often requires help from the teacher to even complete basic procedures	
	Average (4-5)	Demonstrates general knowledge of lab procedures. Requires help from the teacher with some steps in procedures.	
	Good (6-8)	Demonstrates good knowledge of the lab procedures. Will discuss with peers to solve problems in procedures.	
	Outstanding (9-10)	Demonstrates very good knowledge of the lab procedures. Gladly helps other students to follow procedures.	
Data Collection	Poor (0-3)	Measurements are incomplete, inaccurate and imprecise. Observations are incomplete or not included. Symbols, units and significant figures are not included.	
	Average (4-5)	Measurements are somewhat inaccurate and imprecise. Observations are incomplete. There are 3 or more minor errors using symbols, units and significant digits or 2 major errors	
	Good (6-8)	Measurements are mostly accurate. Observations are generally complete. Work is organized. Only 2 or 3 minor errors using symbols, units and significant digits.	
	Outstanding (9-10)	Measurements are both accurate and precise. Observations are very thorough and may recognize possible errors in data collection.	

		Work is neat and organized. Includes appropriate symbols, units and significant digits.	
Analysing and Concluding	Poor (0-3)	Provides limited analysis of the data. Demonstrates limited ability to draw conclusions based on the data.	
	Average (4-5)	Provides some analysis of the data. Demonstrates some ability to draw conclusions based on the data.	
	Good (6-8)	Provides sufficientanalysis of the data. Draws valid conclusions based on the data.	
	Outstanding (9-10)	Provides rich analysis of the data. Draws insightful conclusions based on the data.	
Safety	Poor (0-3)	Proper safety precautions are consistently missed. Needs to be reminded often during the lab.	
	Average (4-5)	Proper safety precautions are often missed. Needs to be reminded more than once during the lab.	
	Good (6-8)	Proper safety precautions are generally used. Uses general reminders of safe practices independently.	
	Outstanding (9-10)	Proper safety precautions are consistently used. Consistently thinks ahead to ensure safety. Will often help other students to conduct labs safely.	
Specifications (Computer Program)	Poor (0-3)	The program is producing incorrect results.	
	Average (4-5)	The program produces correct results but does not display them correctly.	
	Good (6-8)	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	
	Outstanding (9-10)	The program works and meets all of the specifications.	
Readability (Computer Program)	Poor (0-3)	The code is poorly organized and very difficult to understand.	
	Average (4-5)	The code is readable only by someone who knows what it is supposed to be doing	
	Good (6-8)	The code is fairly easy to understand.	
	Outstanding (9-10)	The code is exceptionally well organized and very easy to follow.	
Reusability (Computer Program)	Poor (0-3)	The code is not organized for reusability.	
	Average	Some parts of the code could be reused in	

	(4-5)	other programs.	
	Good (6-8)	Most of the code could be reused in other programs.	
	Outstanding (9-10)	The code could be reused as a whole, or each routine could be reused.	
Documentation (Computer Program)	Poor (0-3)	The documentation is simply comments embedded in the code and does not help the reader understand the code.	
	Average (4-5)	The documentation is simply comments embedded in the code with some simple header comments separating routines	
	Good (6-8)	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code.	
	Outstanding (9-10)	The documentation is well written and clearly explains what the code is accomplishing and how.	
Timely submission (Computer Program)	Poor (0-3)	The code was more than 2 weeks overdue.	
	Average (4-5)	The code was within 2 weeks of the due date.	
	Good (6-8)	The program was delivered within a week of the due date.	
	Outstanding (9-10)	The program was delivered on time	
Efficiency (Computer Program)	Poor (0-3)	The code is huge and appears to be patched together.	
	Average (4-5)	The code is brute force and unnecessarily long	
	Good (6-8)	The code is fairly efficient without sacrificing readability and understanding.	
	Outstanding (9-10)	The code is extremely efficient without sacrificing readability and understanding	

ASSIGNMENT NO. 1

Aim

Study of Open Source Database (MySQL).

Objective

To understand basic concepts of open source database.

Theory

What is Database?
A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing,managing, searching and replicating the data it holds. So nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.
A **Relational DataBase Management System (RDBMS)** is a software that:
_ Enables you to implement a database with tables, columns and indexes.
_ Guarantees the Referential Integrity between rows of various tables.
_ Updates the indexes automatically.
_ Interprets an SQL query and combines information from various tables.
RDBMS Terminology:
Before we proceed to explain MySQL database system, let's revise few definitions related to database.
_ **Database:** A database is a collection of tables, with related data.
_ **Table:** A table is a matrix with data. A table in a database looks like a simple spreadsheet.
_ **Column:** One column (data element) contains data of one and the same kind, for example the
column postcode.
_ **Row:** A row (= tuple, entry or record) is a group of related data, for example the data of one
subscription.
_ **Redundancy:** Storing data twice, redundantly to make the system faster.
_ **Primary Key:** A primary key is unique. A key value cannot occur twice in one table. With a key,
you can find at most one row.
_ **Foreign Key:** A foreign key is the linking pin between two tables.
_ **Compound Key:** A compound key (composite key) is a key that consists of multiple columns,
because one column is not sufficiently unique.
_ **Index:** An index in a database resembles an index at the back of a book.

_ **Referential Integrity:** Referential Integrity makes sure that a foreign key value always points to
an existing row.

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:
• MySQL is released under an open-source license. So you have nothing to pay to use it.
• MySQL is a very powerful program in its own right. It handles a large subset of the functionality
of the most expensive and powerful database packages.
• MySQL uses a standard form of the well-known SQL data language.
• MySQL works on many operating systems and with many languages including PHP, PERL, C,
C++, JAVA, etc.
• MySQL works very quickly and works well even with large data sets.
• MySQL is very friendly to PHP, the most appreciated language for web development.
• MySQL supports large databases, up to 50 million rows or more in a table. The default file size
limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a
theoretical limit of 8 million terabytes (TB).
• MySQL is customizable. The open-source GPL license allows programmers to modify the
MySQL software to fit their own specific environments.

Creating Tables

To create table we use the **CREATE TABLE** statement. The typical form is as follows:

```
1 CREATE TABLE [IF NOT EXISTS] table_name(
2 column_list
3 ) type=table_type
```

• MySQL supports IF NOT EXISTS after CREATE TABLE statement to prevent you from error of creating table which already exists on the database server.
• table_name is the name of table you would like to create. After that, you can define a set of columns which is usually in this form: column_name data_type(size) [NOT] NULL.
• You can specify the storage engine type you prefer to use for the table. MySQL supports various storage engines such as InnoDB, MyISAM... If you don't explicit declare storage engine type,
MySQL will use MyISAM by default.
In our **classicmodels** sample database, to create **employees** table, we can use the CREATE TABLE statement as follows:

Showing and Describing Tables in a Database

```
1 SHOW TABLES
```

Here is the output of **classicmodels** database:

```
+------------------------+
```

```
| Tables_in_classicmodels |
+-------------------------+
| customers |
| employees |
+-------------------------+
2 rows in set (0.00 sec)
```

In some cases, you need to see the table's metadata, you can use DESCRIBE statement as follows:

```
1 DESCRIBE table_name;
```

For instance, we can describe employees table like below query:

```
1 DESCRIBE employees;
```

The output return from the database server:

```
+----------------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| employeeNumber | int(11) | NO | PRI | NULL | |
| lastName | varchar(50) | NO | | NULL | |
| firstName | varchar(50) | NO | | NULL | |
| extension | varchar(10) | NO | | NULL | |
| email | varchar(100) | NO | | NULL | |
| officeCode | varchar(10) | NO | | NULL | |
| reportsTo | int(11) | YES | | NULL | |
| jobTitle | varchar(50) | NO | | NULL | |
+----------------+-------------+------+-----+---------+-------+
8 rows in set (0.02 sec)
```

Altering Table Structures

The following illustrates the ALTER TABLE statement syntax:

```
01 ALTER [IGNORE] TABLE table_name options[, options...]
02 options:
03 ADD [COLUMN] create_definition [FIRST | AFTER col_name ]
04 or ADD [COLUMN] (create_definition, create_definition,...)
05 or ADD INDEX [index_name] (index_col_name,...)
06 or ADD PRIMARY KEY (index_col_name,...)
07 or ADD UNIQUE [index_name] (index_col_name,...)
08 or ADD FULLTEXT [index_name] (index_col_name,...)
09 or ADD [CONSTRAINT symbol] FOREIGN KEY [index_name]
(index_col_name,...)
10 [reference_definition]
11 or ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
12 or CHANGE [COLUMN] old_col_name create_definition
13 [FIRST | AFTER column_name]
14 or MODIFY [COLUMN] create_definition [FIRST | AFTER col_name]
15 or DROP [COLUMN] col_name
16 or DROP PRIMARY KEY
17 or DROP INDEX index_name
18 or DISABLE KEYS
```

19 or ENABLE KEYS
20 or RENAME [TO] new_table_name
21 or ORDER BY col_name
22 or table_options
Most of these options are obvious. We will explain some here:
• The CHANGE and MODIFY are the same, they allow you to change the definition of the column or its position in the table.
• The DROP COLUMN will drop the column of the table permanently, if the table contain data all the data of the column will be lost.
• The DROP PRIMARY KEY and DROP INDEX only remove the primary key or index of the column.
• The DISABLE and ENABLE KEYS turn off and on updating indexes for MyISAM table only.

The RENAME Clause allows you the change the table name to the new one.

Deleting Tables

To delete table from the database, you can use DROP TABLE statement:
1 DROP [TEMPORARY] TABLE [IF EXISTS] table_name [, table_name,...]
TEMPORARY keyword is used for deleting temporary tables. MySQL allows you to drop multiple tables at once by listing them and separated each by a comma. IF EXISTS is used to prevent you from deleting
table which does not exist in the database.

Empty Table's Data

1 TRUNCATE TABLE table_name
• TRUNCATE TABLE statement drop table and recreate it therefore it is much faster than DELETE TABLE statement. However it is not transaction-safe.
• The number of deleted rows is not returned like SQL DELETE TABLE statement.
• ON DELETE triggers are not invoked because TRUNCATE does not use DELETE statement.

Changing columns using MySQL ALTER TABLE statement

1 ALTER TABLE tasks
2 CHANGE COLUMN task_id task_id INT(11) NOT NULL AUTO_INCREMENT;

Using MySQL ALTER TABLE to add a new column into a table

1 ALTER TABLE tasks ADD COLUMN 'complete' DECIMAL(2,1) NULL
2 AFTER 'description' ;

Using MySQL ALTER TABLE to drop a column froma table

1 ALTER TABLE tasks
2 DROP COLUMN description ;

Renaming table using MySQL ALTER TABLE statement

1 ALTER TABLE 'tasks'

Managing Database Index in MySQL

Creating Indexes

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
USING [BTREE | HASH | RTREE]
ON table_name (column_name [(length)] [ASC | DESC],...)

First you specify the index based on the table types or storage engine:
• UNIQUE means MySQL will create a constraint that all values in the index must be distinct.
Duplicated NULL is allowed in all storage engine except BDB.
• FULLTEXT index is supported only by MyISAM storage engine and only accepted columns which
have data type is CHAR,VARCHAR or TEXT.
• SPATIAL index supports spatial column and available in MyISAM storage engine. In addition, the
column value must not be NULL.
Then you name the index using index types such as BTREE, HASH or RTREE also based on storage
engine. Here is the list:

Storage Engine	**Allowable Index Types**
MyISAM	BTREE, RTREE
InnoDB	BTREE
MEMORY/HEAP	HASH, BTREE
NDB	HASH

CREATE INDEX officeCode ON employees(officeCode)

Removing Indexes

DROP INDEX index_name ON table_name
DROP INDEX officeCode ON employees

MySQL SELECT Statement

In order to retrieve data from MySQ
1 SELECT column_name1,col
2 FROM tables
3 [WHERE conditions]
4 [GROUP BY group
5 [HAVING group_condition
6 [ORDER BY sort_columns]
7 [LIMIT limits];

1. SELECT * FROM employees;

2. SELECT lastname,firstna
   FROM employees;

3. SELECT firstname,lastna
FROM employees
WHERE jobtitle="president";

4. SELECT DISTINCT jobTitle
FROM employees;

5. SELECT firstname,lastna
FROM employees
ORDER BY firstname ASC, jobtitle DESC;

6. SELECT firstname,lastname
FROM employees
LIMIT 5

7. SELECT firstname,lastname
FROM employees
LIMIT 10,5

8. SELECT officeCode, city, phone
FROM offices
WHERE country IN ('USA','France')

9. SELECT officeCode, city, phone
FROM offices
WHERE country NOT IN ('USA','France')

10. SELECT orderNumber
FROM orderDetails
GROUP BY orderNumber
HAVING SUM(quantityOrdered * priceEach) > 60000
11. SELECT employeeNumber, lastName, firstName
FROM employees
WHERE firstName LIKE 'a%'

12. SELECT productCode,ProductName,buyPrice
FROM products
WHERE buyPrice BETWEEN 90 AND 100
ORDER BY buyPrice DESC

13. SELECT employeeNumber, lastName, firstName
FROM employees
WHERE firstname LIKE "T_m"

14. SELECT customerNumber id, contactLastname name
FROM customers
UNION
SELECT employeeNumber id,firstname name
FROM employees

MySQL INNER JOIN

```
 SELECT column_list
 FROM t1
 INNER JOIN t2 ON join_con
 INNER JOIN t3 ON join_con
5 ...
WHERE where_conditions;
```

```
SELECT A.productCode, A
FROM products A
INNER JOIN orderDetails B on A.productCode = B.productCode;
```

MySQL LEFT JOIN

```
SELECT t1.c1, t1.c2,…t2
FROM t1
LEFT JOIN t2 ON t1.c1 = t2.c1 …(join_condition)
WHERE where_condition
```

```
SELECT c.customerNumber
FROM customers c
LEFT JOIN orders o ON c.customerNumber = o.customerNumber;
```

MySQL GROUP BY with aggregate function

```
SELECT status, count(*)
FROM orders
GROUP BY status
```

References:

1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition “Database Management Systems”
2. Avi Silberschatz , Henry F. Korth , S. Sudarshan, “Database System Concepts, Sixth Edition”, ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	What is DBMS?	1	1
2	What are advantages of DBMS?	2	1
3	Explain data model with suitable example.	2	1
4	What are the features of NoSQL database?	2	1
5	Compare RDBMS and NoSQL	2	1

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.

d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

e. **Output**: describe the results in few lines

f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

ASSIGNMENT NO. 2

Aim

Install and configure client and server for MySQL.

Objective

To study, install and configure MySQL.

Theory

MySQL server installation in MySQL

Step I Update System
Sudo apt-get update
sudo apt-get upgrade
Step II Install
sudo apt-get install mysql-server

by default MySQL will bind to 127.0.0.1 port number 3306
We can change by modifying the bind-address parameter in /etc/my.cnf
(in case of Ubuntu 16.04 /etc/mysql/mysql.conf.d)
0.0.0.0 allows access from all machines
Step III
sudo mysql_secure_installation

Root login
mysql –u root –p
Help
\h for help
Create Database
create database testdb;
create user 'testuser'@'localhost' identified by 'password'
Access MySQL remotely
create user 'testuser1'@'192.168.100.137' identified by 'password'
grant all on testdb.*@192.168.100.137 to testuser1
sudo iptables –I –p udp --dport 3306 –j ACCEPT
sudo iptables –I –p tcp --dport 3306 –j ACCEPT

mysql –u testuser1 –p –h 192.168.100.138
Start/stop MySQL
sudo service start mysql
sudo service stop mysql
sudo service restart mysql

restart –nlt|grep 3306
telnet IP_Address 3306
nmap 192.168.100.138

References:

1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition "Database Management Systems"
2. Avi Silberschatz , Henry F. Korth , S. Sudarshan, "Database System Concepts, Sixth Edition", ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Explain difference between MongoDB and MySQL?	2	2
2	Explain data model for MongoDB?	2	2
3	Explain step by step installation of MySQL.	2	2
4	Explain step by step installation of MongoDB?	3	2,3

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.

d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

e. **Output**: describe the results in few lines

f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)

Lab file (neatness and regularity)
Viva (from time to time)
Mock Practical Exam
Exam (end term): Practical + Viva

- **Assessment Methodology**
  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

ASSIGNMENT NO. 3

Aim

Study of SQLite: What is SQLite? Uses of Sqlite. Building and installing SQLite.

Objective

Install and configure database System.

Theory

What is SQLite?

- **SQLite** is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available. It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine. Ordinary disk files can be easily read and write by SQLite because it does not have any separate server like SQL. The SQLite database file format is cross-platform so that anyone can easily copy a database between 32-bit and 64-bit systems. Due to all these features, it is a popular choice as an Application File Format.

History

- It was designed by D. Richard Hipp for the purpose of no administration required for operating a program. in August 2000. As it is very lightweight compared to others like MySql and Oracle, it is called SQLite. Different versions of SQLite are released since 2000.

Installation on Windows

- Installation on Windows:
- Visit the official website of SQLite for downloading the zip file.
- Download that zip file.
- Create a folder in C or D (wherever you want) for storing SQLite by expanding zip file.

- Open the command prompt and set the path for the location of SQLite folder given in the previous step. After that write “sqlite3” and press enter.

Installation on Linux

- Sudo apt-get install sqlite3 libsqlite3-dev
- It will automatically install and once it asks Do you want to continue (Y/N) type Y and press enter. After successful installation, we can check it by command sqlite3.

Features

- The transactions follow ACID properties i.e. atomicity, consistency, isolation, and durability even after system crashes and power failures.
- The configuration process is very easy, no setup or administration needed.
- All the features of SQL are implemented in it with some additional features like partial indexes, indexes on expressions, JSON, and common table expressions.
- Sometimes it is faster than the direct file system I/O.
- It supports terabyte-sized databases and gigabyte-sized strings and blobs.
- Almost all OS supports SQLite like Android, BSD, iOS, Linux, Mac, Solaris, VxWorks, and Windows (Win32, WinCE, etc. It is very much easy to port to other systems.
- Complete database can be stored in a single cross-platform disk file.

Applications

- Due to its small code print and efficient usage of memory, it is the popular choice for the database engine in cellphones, PDAs, MP3 players, set-top boxes, and other electronic gadgets.
- It is used as an alternative for open to writing XML, JSON, CSV or some proprietary format into disk files used by the application.

- As it has no complication for configuration and easily stores file in an ordinary disk file, so it can be used as a database for small to medium sized websites.
- It is faster and accessible through a wide variety of third-party tools, so it has great application in different software platforms.

Disadvantages

- It is only used where there is low to medium traffic requests are there.
- The database size is restricted i.e. it is 2GB in most cases.

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Explain what is SQLite?	1	1
2	Explain features of SQLite Database?	2	1
3	What is difference between MySQL and SQLite?	2	1
4			
5			

ASSIGNMENT NO. 4

Aim

Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system.

Objective

Understand and design ER Diagram.

Theory

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Attributes

Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes

- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved

directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.

- **Single-value attribute** − Single-value attributes contain single value. For example − Social_Security_Number.
- **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

These attribute types can come together in a way like −

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

Entity-Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Relationship

The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

Relationship Set

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

Degree of Relationship

The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3

- n-ary = degree

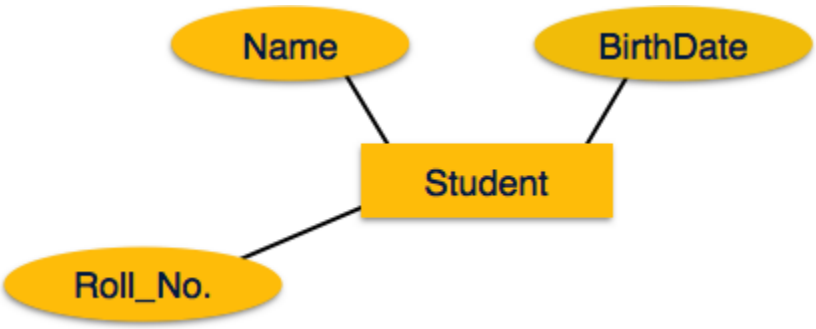
how the ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

Entity

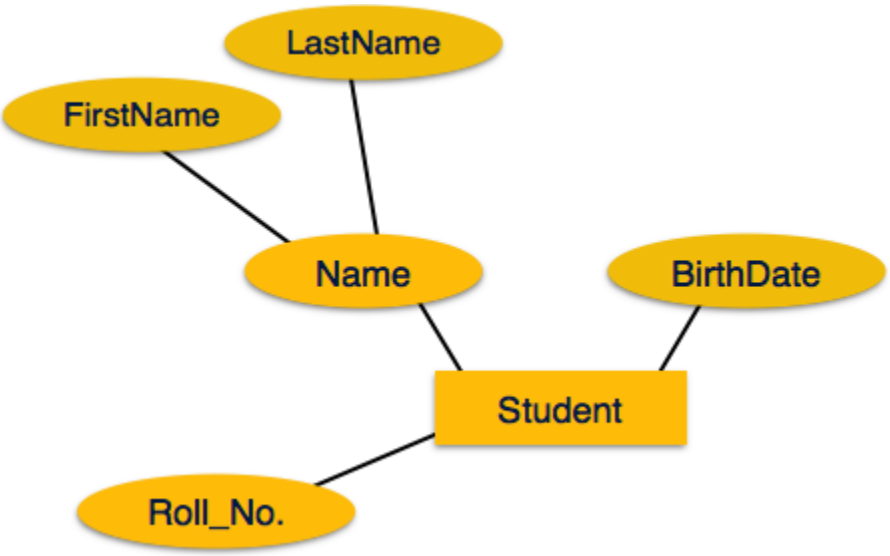
Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Attributes

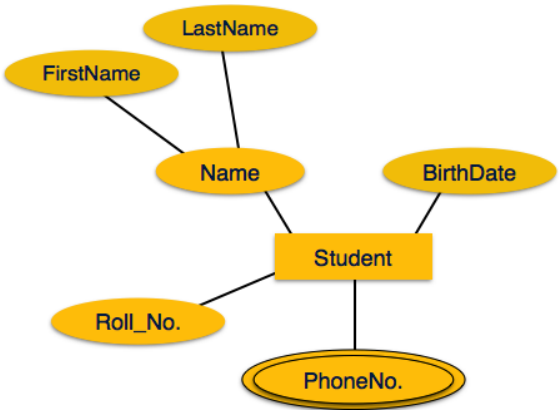
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



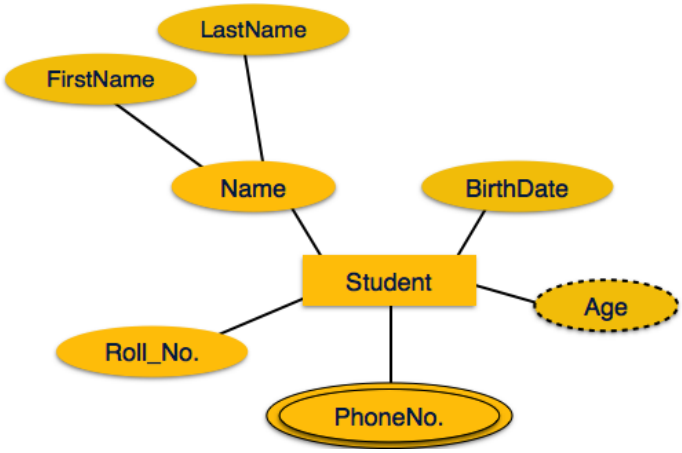
If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multivalued attributes are depicted by double ellipse.



Derived attributes are depicted by dashed ellipse.



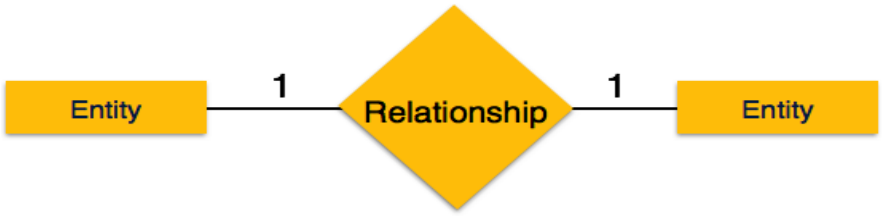
Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

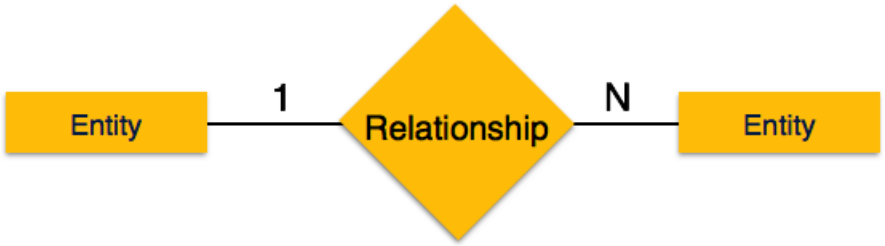
Binary Relationship and Cardinality

A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

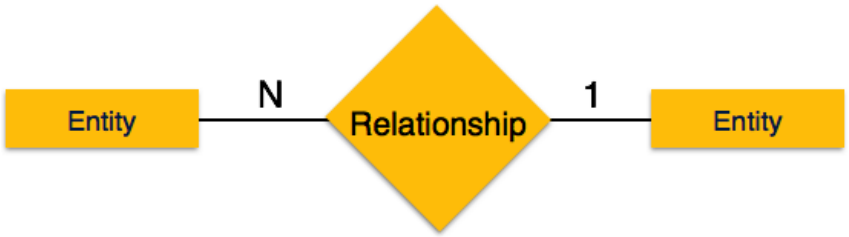
- **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



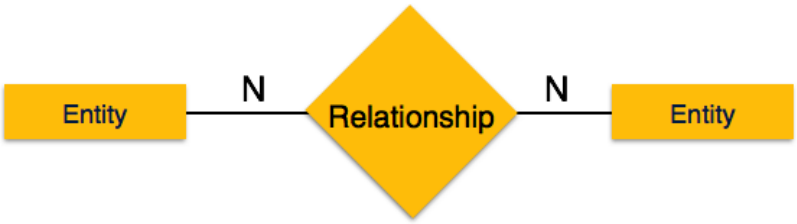
- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.



- **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.

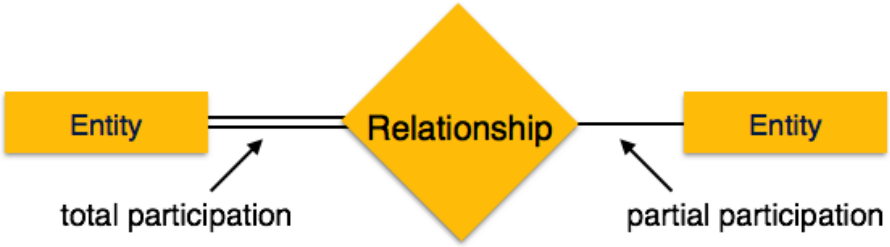


- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



Participation Constraints

- **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.

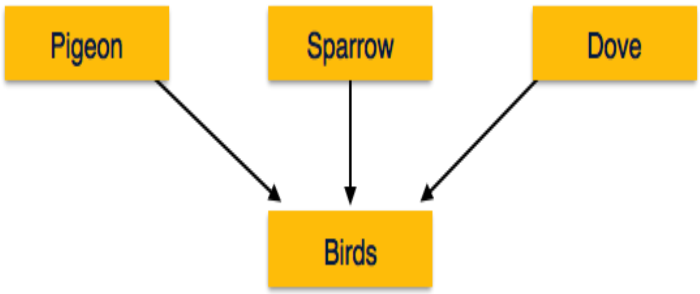


The ER Model has the power of expressing database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities, and as we go deep in the hierarchy, it gives us the detail of every entity included.

Going up in this structure is called **generalization**, where entities are clubbed together to represent a more generalized view. For example, a particular student named Mira can be generalized along with all the students. The entity shall be a student, and further, the student is a person. The reverse is called **specialization** where a person is a student, and that student is Mira.

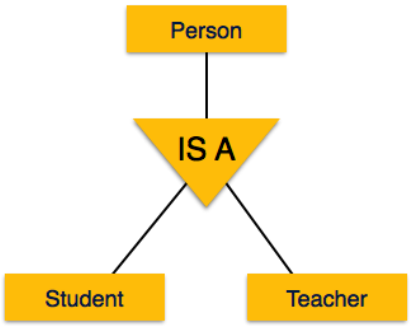
Generalization

As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



Specialization

Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group ‘Person’ for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.

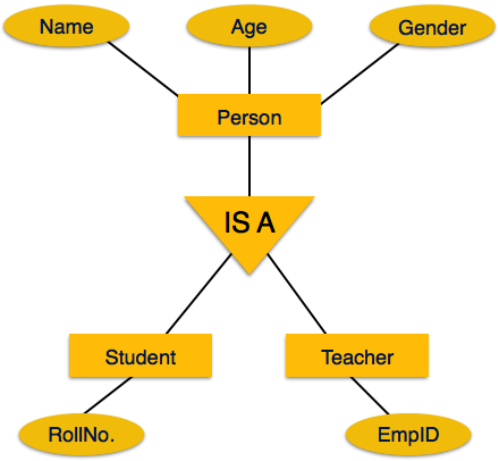


Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.

Inheritance

We use all the above features of ER-Model in order to create classes of objects in object-oriented programming. The details of entities are generally hidden from the user; this process known as **abstraction**.

Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.



Multinational Company Manages information of Employees, Department and Project. Each Employee has unique ID. Each Employee works on project in some department. Every employee is assigned at least one project.
Design ER Diagram for Employee Management System.

References:

1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition "Database Management Systems"
2. AviSilberschatz , Henry F. Korth , S. Sudarshan, "Database System Concepts, Sixth Edition", ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	What is ER Diagram?	1	1
2	What are different graphical notations used?	2	1
3	Explain data model with suitable example.	2	1
4	Explain Specialization and Generalization?	2	1
5	What is importance of ER diagram?	2	1

ASSIGNMENT NO. 5

Aim

Design and implement a database (for assignment no 4) using DDL statements and apply normalization on the.

Objective

Understand and design Database and apply normalization.

Theory

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

Numeric Data Types:

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

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

- **DECIMAL(M,D)** - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types:

The MySQL date and time datatypes are:

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

String Types:

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

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

- **MEDIUMBLOB or MEDIUMTEXT** - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

Table

CREATE TABLE table_name (column_name column_type);

drop table table_name;

alter table table_name [add|modify|drop]

Index

A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

While creating index, it should be considered that what are the columns which will be used to make SQL queries and create one or more indexes on those columns.

Practically, indexes are also type of tables, which keep primary key or index field and a pointer to each record into the actual table.

The users cannot see the indexes, they are just used to speed up queries and will be used by Database Search Engine to locate records very fast.

INSERT and UPDATE statements take more time on tables having indexes where as SELECT statements become fast on those tables. The reason is that while doing insert or update, database need to insert or update index values as well.

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX *index_name*
[*index_type*]

ON *tbl_name* (*index_col_name*,...)
[*index_type*]

index_col_name:
col_name [(*length*)] [ASC | DESC]

index_type:
USING {BTREE | HASH}

Sequence

In MySQL, a sequence is a list of integers generated in the ascending order i.e., 1,2,3… Many applications need sequences to generate unique numbers mainly for identification e.g., customer ID in CRM, employee number in HR, equipment number in services management system, etc.

To create a sequence in MySQL automatically, you set the AUTO_INCREMENT attribute to a column, which typically is a primary key column. The following rules you must follow when you use AUTO_INCREMENT attribute:

- Each table has only one AUTO_INCREMENT column whose data type is typically integer
- The AUTO_INCREMENT column must be indexed, which means it can be either PRIMARY KEY or UNIQUE index.
- The AUTO_INCREMENT column must have NOT NULL constraint. When you set AUTO_INCREMENT attribute to a column, MySQL will make it NOT NULL for you in case you don't define it explicitly.

Start MySQL
mysql -u root -p

DCL

create database mydb;
use mydb;
create user user1@localhost identified by 'test123';
grant all on mydb to user1@localhost;
revoke all on mydb from user1@localhost;

DDL

Multinational Company Manages information of Employees. Each Employee has unique ID.
Each Employee works on project in some department.
Design ER Diagram for Employee Management System

Create tables for above.

Employee(EmpID, EName, Address, Contactno, DOB, DOJ, Salary)
Department(DID, DName)
Project(PId, PName, Location)

e.g
create table Employee(empid int, ename varchar(20), doj date);

Normalization
What is Normalization?
Normalization is the process of organizing the data in the database.
Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
Normalization divides the larger table into smaller and links them using relationships.
The normal form is used to reduce redundancy from the database table.
First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

Boyce Codd Normal Form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Use of ER diagram and various graphical notations used?	2	2
2	Explain EER features?	2	2
3	Explain constraints with suitable Example	2	2
4	Explain DDL and DCL with example?	2	2
5	Design ER diagram for Employee Management System	3	3, 4

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.

d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

e. **Output**: describe the results in few lines

f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

ASSIGNMENT NO. 6

Aim

Create table with primary key and foreign key constraint.
a Alter table with add and modify b. Drop Table

Objective

Understand and implement DDL statements.

Theory

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

Numeric Data Types:

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

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

- **DECIMAL(M,D)** - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types:

The MySQL date and time datatypes are:

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

String Types:

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

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

- **MEDIUMBLOB or MEDIUMTEXT** - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

Table

CREATE TABLE table_name (column_name column_type);

drop table table_name;

alter table table_name [add|modify|drop]

Index

A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

While creating index, it should be considered that what are the columns which will be used to make SQL queries and create one or more indexes on those columns.

Practically, indexes are also type of tables, which keep primary key or index field and a pointer to each record into the actual table.

The users cannot see the indexes, they are just used to speed up queries and will be used by Database Search Engine to locate records very fast.

INSERT and UPDATE statements take more time on tables having indexes where as SELECT statements become fast on those tables. The reason is that while doing insert or update, database need to insert or update index values as well.

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX *index_name*
[*index_type*]

ON *tbl_name* (*index_col_name*,...)
[*index_type*]

index_col_name:
col_name [(*length*)] [ASC | DESC]

index_type:
USING {BTREE | HASH}

Sequence

In MySQL, a sequence is a list of integers generated in the ascending order i.e., 1,2,3… Many applications need sequences to generate unique numbers mainly for identification e.g., customer ID in CRM, employee number in HR, equipment number in services management system, etc.

To create a sequence in MySQL automatically, you set the AUTO_INCREMENT attribute to a column, which typically is a primary key column. The following rules you must follow when you use AUTO_INCREMENT attribute:

- Each table has only one AUTO_INCREMENT column whose data type is typically integer
- The AUTO_INCREMENT column must be indexed, which means it can be either PRIMARY KEY or UNIQUE index.
- The AUTO_INCREMENT column must have NOT NULL constraint. When you set AUTO_INCREMENT attribute to a column, MySQL will make it NOT NULL for you in case you don't define it explicitly.

Start MySQL
mysql -u root -p

DCL

create database mydb;
use mydb;
create user user1@localhost identified by 'test123';
grant all on mydb to user1@localhost;
revoke all on mydb from user1@localhost;

DDL

Multinational Company Manages information of Employees.Each Employee has unique ID.
Each Employee works on project in some department.
Design ER Diagram for Employee Management System

Create tables for above.

Employee(EmpID, EName, Address, Contactno, DOB, DOJ, Salary, DID, PId)
Department(DID, DName)
Project(PId, PName, LocationId)
Location(LocationId, LocationCity)

1 use mydb;
2 create table Department(did int primary key, dname varchar(20));
3 create table Employee(empid int primary key auto_increment, ename varchar(20) not null, contactno int unique, salary int, did int, check (salary>15000), foreign key(did) references Department(did));
4. drop table employee;
5. create table Project(Pid int primary key, Pname varchar(20), LocationId int);
6. alter table Employee add Pid int;
7. desc Employee;
8. alter table Employee add foreign key(Pid) references Project(Pid);
9. alter table Project drop LocationId;
10 alter table Project modify Pname varchar(50);
Example of composite key
11. create table Emp(empid int, name varchar(20), primary key(empid, name));

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Use of ER diagram and various graphical notations used?	2	2
2	Explain EER features?	2	2
3	Explain constraints with suitable Example	2	2
4	Explain DDL and DCL with example?	2	2
5	Design ER diagram for Employee Management System	3	3, 4

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including

problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.

d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

e. **Output**: describe the results in few lines

f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)

  Lab file (neatness and regularity)

  Viva (from time to time)

  Mock Practical Exam

  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks

  Program demonstration- 4 marks

  Viva-voce -2 marks

  Timely submission of journal- 2 marks

ASSIGNMENT NO. 7

Aim

Perform following SQL queries on the database created in assignment 1.
• Implementation of relational operators in SQL
• Boolean operators and pattern matching
• Arithmetic operations and built in functions
• Group functions
• Processing Date and Time functions
• Complex queries and set operators

Objective

Understand and implement DML statements.

Theory

Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data. It is responsible for performing all types of data modification in a database.

There are three basic constructs which allow database program and user to enter data and information are:

Here are some important DML commands in SQL:

INSERT
UPDATE
DELETE

General syntax of **select** query:

```
select A1, A2, ..., An
from r1, r2, ..., rm
where P
```

Insert

```
INSERT INTO table_name
(col1, col2,…..)
values
(col1_value, col2_value, ……)
```

Update

```
update table_name
set
column_name::expression
….
where condition
```

Delete

```
delete from table_name
where condition;
```

MySQL Inbuilt Functions

The MID() function extracts a substring from a string (starting at any position).

MID(*string*, *start*, *length*)

The LENGTH() function returns the length of a string (in bytes).
LENGTH(*string*)
The STRCMP() function compares two strings.
STRCMP(*string1*, *string2*)
The SUBSTR() function extracts a substring from a string (starting at any position).
SUBSTR(*string*, *start*, *length*)
The LCASE() function converts a string to lower-case.
LCASE(*text*)
The ABS() function returns the absolute (positive) value of a number.
ABS(*number*)
The DATE() function extracts the date part from a datetime expression.
DATE(*expression*)
The NOW() function returns the current date and time.
NOW()
The TIME() function extracts the time part from a given time/datetime.
TIME(*expression*)
The BIN() function returns a binary representation of a number, as a string value.
BIN(*number*)

Aggregate Functions

The data that you need is not always stored in the tables. However, you can get it by performing the calculations of the stored data when you select it.

For example, you cannot get the total amount of each order by simply querying from the order details table because the order details table stores only quantity and price of each item. You have to select the quantity and price of an item for each order and calculate the order's total.

To perform such calculations in a query, you use aggregate functions.

By definition, an aggregate function performs a calculation on a set of values and returns a single value.

MySQL provides many aggregate functions that include AVG, COUNT, SUM, MIN, MAX, etc. An aggregate function ignores NULL values when it performs calculation except for the COUNT function.

AVG function

The AVG function calculates the average value of a set of values. It ignores NULL values in the calculation.

select avg(Salary) from Employee;

SUM function

The SUM function returns the sum of a set of values. The SUM function ignores NULL values. If no matching row found, the SUM function returns a NULL value.

MAX function

The MAX function returns the maximum value in a set of values.

MIN function

The MIN function returns the minimum value in a set of values.

Nested Queries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow −

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Multinational Company Manages information of Employees.Each Employee has unique ID.
Each Employee works on project in some department.
Design ER Diagram for Employee Management System
Create tables for above.

Employee(EmpID, EName, Address, Contactno, DOB, DOJ, Salary, DID, PId)
Department(DID, DName)
Project(PId, PName, LocationId)
Location(LocationId, LocationCity)

Example Queries:

1 select * from employee;
2 select * from employee where pid is null;
3 select * from employee where pid is not null;
4 select * from employee where salary>=20000;
5 select * from employee where salary>20000 and pid=102;
6 select * from employee where did<>11;
7 select * from employee where doj>='2012-01-01' and doj<='2012-12-31';
8 select * from employee where doj>='2012-01-01' and doj<='2012-12-31' and did=12;
9 select * from employee where doj between '2012-01-01' and '2012-12-31';

Boolean Operators and Pattern Matching

1. select * from Employee where Address like 'PUNE';
2. select * from Employee where EmpName like 'A%';
3. select * from Employee where Address like '%bad';
4. select * from Employee where EmpName like '_a%';
5. select * from Employee where EmpName like '____'; //four underscores
6. select * from Employee where Address like '%/_%';
7. select * from Employee where Address like '%bad' and EmpName like '____';
8. select * from Employee where Address not like 'PUNE';

Inbuilt Functions
select length(EName) from Employee;
select ucase(EName) from Employee;
select lcase(EName) from Employee;
select abs(salary) from Employee;

Aggregate function : sum, min, max, avg, count

1. select sum(Salary) from Employee;
2. select count(EmpID) from Employee;
3. select count(EmpID) from Employee where DId='D101';
4. select count(EmpID) from Employee where DOJ between '2016-01-01' and '2016-12-31';
5. select sum(Salary) from Employee where DId='D101';
6. select sum(Salary) from Employee where DId in ('D101', 'D102', 'D103');
7. select DId, sum(salary) from Employee group by DId;
8. select DId, sum(salary) as totalsal from Employee group by DId having totalsal>150000;
9. select DId, sum(salary) as totalsal from Employee where DOJ <= now() group by DId;
10. select DId, sum(salary) as totalsal from Employee where DOJ between '1992-01-01' and date() group by DId;
11. select DName, sum(Salary) as totalsal from Employee, Department where Emplyoee.DID=Department.DID group by DName;

Nested Queries

1. select * from employee where Salary **in** (select max(Salary) from employee);
2. select * from employee where Salary **not in** (select max(Salary) from employee);
3. select Salary from employee where Salary **<> any** (select e.Salary from employee as e where e.did=11);
4. select Salary from employee where Salary **> any** (select e.Salary from employee as e where e.did=11);
5. select Salary from employee where Salary **> some** (select e.Salary from employee as e where e.did=11);
6. select Salary from employee where Salary = **some** (select e.Salary from employee as e where e.did=11);
7. select * from employee where Salary >= **all (**select Salary from employee);
8. select Salary from employee where Salary > **all** (select e.Salary from employee as e where e.did=11);
9. select Salary from employee where **exists** (select e.Salary from employee as e where e.did=employee.did and did=11);
10. select Salary from employee where **not exists** (select e.Salary from employee as e where e.did=11 and e.did=employee.did);

Join

Natural Join
select * from Employee natural join Department;
Inner Join
select * from Employee inner join Department on Employee.DID=Department.DID;

Outer Join
select * from Employee left join Department on Employee.DID=Department.DID;
select * from Employee right join Department on Employee.DID=Department.DID;

Output

- Syntax of Various SQL DML statements.

References:
1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition "Database Management Systems"
2. Avi Silberschatz , Henry F. Korth , S. Sudarshan, "Database System Concepts, Sixth Edition", ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Explain DML	2	2
2	Explain syntax of select query with suitable exxample?	3	2,3
3	Explain syntax to check null values	3	2,3
4	Explain significance of % and _	2	2,3

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.
d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.
e. **Output**: describe the results in few lines
f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.
g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

ASSIGNMENT NO. 8

Aim

Execute DDL statements which demonstrate use of Views.

Objective

Understand and implement DDL statements to demonstrate views.

Theory

MySQL View:
A database view is a virtual table or logical table which is defined as a SQL SELECT query with joins. Because a database view is similar to a database table, which consists of rows and columns, so you can query data against it.

create view view_name as select statement

MySQL view's restrictions

You cannot create an index on a view. MySQL uses indexes of the underlying tables when you query data against the views that use the merge algorithm. For the views that use the temptable algorithm, indexes are not utilized when you query data against the views.

You cannot use subqueries in the FROM clause of the SELECT statement defined the view before MySQL 5.7.7

If you drop or rename tables that a view is based on, MySQL does not issue any errors. However, MySQL does invalidate the view. You can use the CHECK TABLE statement to check whether the view is valid.

A simple view can be updatable. A view created based on a complex SELECT statement with join, subquery, etc., cannot be updatable

Syntax
create view view_name as select query;

Example ;

Simple View
create view v1 as select empname, address, salary from employee;

View using joins
create view v2 as select empname, dname from employee, department where employee.did=department.did;

View using aggregate function
create view v3 as select did, sum(salary) from employee group by did;

View using nested Queries

create view v4 as select * from Employee where salary in (select max(salary) from Employee);

Output

- Create view.

References:
1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition "Database Management Systems"
2. Avi Silberschatz , Henry F. Korth , S. Sudarshan, "Database System Concepts, Sixth Edition", ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Explain concept of views?	2	2
2	Explain view updating rules?	2	2
3	Explain constraints on views.	2	2

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.

d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

e. **Output**: describe the results in few lines
f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.
g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks

ASSIGNMENT NO. 9

Aim

Write and execute PL/SQL stored procedure and function to perform a suitable task on the database.

Objective

To study and implement function and procedure for suitable database application.

Theory

A database stored program (stored module or stored routine) is a computer program that is stored within and executes within the database server.
When program is executed, it is executed within the memory address of a database server process or thread.

Syntax

```
Create procedure example()
Begin
//declare program variables
//statements
End;

Call example();
```

Use of Stored Procedure

1 Allow modular programming.
2 Allow faster execution.
3 Reduce network traffic.
4 Used as a security mechanism.

Hello World Procedure

```
mysql> use mydb;
Database changed

mysql> delimiter $$
mysql> drop procedure if exists helloworld$$
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> create procedure helloworld()
    -> begin
    -> select 'Hello world';
    -> end$$
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> call helloworld()$$
+-------------+
| Hello world |
+-------------+
| Hello world |
+-------------+
1 row in set (0.03 sec)

Query OK, 0 rows affected (0.03 sec)
```

Procedure to implement in, out parameters

```
mysql> create procedure calroot(in num1 int, out result int)
    -> begin
    -> set result = sqrt(num1);
    -> select result;
    -> end$$
Query OK, 0 rows affected (0.00 sec)

mysql> call calroot(16, @out_value)$$
+--------+
| result |
+--------+
|      4 |
+--------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Procedure to calculate Square Root of Number

```
Drop procedure if exists calsqrt $$
Create procedure calsqrt(num1 int)
Begin
Declare result int;
Set result = sqrt(num1);
```

```
Select result;
End $$
```

Procedure to find even or odd number

```
delimiter $$
drop procedure if exists evenodd $$
create procedure evenodd(num1 int)
begin
if mod(num1, 2) = 0 then
select ‘Even number';
end if;
end$$
```

Procedure to get the all employee details

```
create procedure proc_emp()
begin
select * from emp;
end$$
```

Procedure to get the details of Employee

```
create procedure proc_emp1(empid int)
begin
select * from emp where id=empid;
end$$
```

PL/SQL Function Example

```
mysql> create function func1(price int)
   -> returns int
```

```
   -> deterministic
   -> begin
   -> declare discount int;
   -> if price<1000 then
   -> set discount=100;
   -> else
   -> set discount=200;
   -> end if;
   -> return discount;
   -> end$
Query OK, 0 rows affected (0.26 sec)

mysql> select func1(800)$
+------------+
| func1(800) |
+------------+
|        100 |
+------------+
1 row in set (0.06 sec)

mysql> select func1(1200)$
+-------------+
| func1(1200) |
+-------------+
|         200 |
+-------------+
1 row in set (0.00 sec)

mysql> create table product(id int, price int)$
Query OK, 0 rows affected (0.46 sec)

mysql> insert into product values(1, 900)$
Query OK, 1 row affected (0.02 sec)

mysql> insert into product values(2, 1200)$
Query OK, 1 row affected (0.05 sec)

mysql> select * from product$
+------+-------+
| id   | price |
+------+-------+
|    1 |   900 |
|    2 |  1200 |
+------+-------+
2 rows in set (0.02 sec)
```

```
mysql> select id, func1(price) from product$
+------+--------------+
| id   | func1(price) |
+------+--------------+
|    1 |          100 |
|    2 |          200 |
+------+--------------+
2 rows in set (0.00 sec)

mysql> select id, func1(price) as discount from product$
+------+----------+
| id   | discount |
+------+----------+
|    1 |      100 |
|    2 |      200 |
+------+----------+
2 rows in set (0.00 sec)

mysql>
```

Procedure to Find Class of Employee

```
create procedure proc_emp2(empid int)
 begin
 declare sal int;
 select salary into sal from emp where id=empid;
 if sal<10000 then
 select 'Class III'as Status;
 elseif sal between 10000 and 50000 then
 select 'Class II';
 else
 select 'Class I';
 end if;
end$$
```

PL/SQL Function

```
CREATE FUNCTION func_emp(esal int) RETURNS VARCHAR(10)
   DETERMINISTIC
```

```
BEGIN
   DECLARE status varchar(10);

   IF esal < 10000 THEN
 SET status = 'Class III';
   ELSEIF (esal <= 50000 AND esal >= 10000) THEN
      SET status = 'Class II';
   ELSE
      SET status = 'Class I';
   END IF;

 RETURN (status);
END$$
```

Output

- Procedure and function.

References:

1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition "Database Management Systems"
2. Avi Silberschatz , Henry F. Korth , S. Sudarshan, "Database System Concepts, Sixth Edition", ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Compare procedure and function.	2	2,3
2	Explain different types of parameters	2	2,3

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.
d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.
e. **Output**: describe the results in few lines
f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.
g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks

ASSIGNMENT NO. 10

Aim

Write a database trigger (Row level and statement level)

Objective

To study and implement database trigger

Theory

- Stored programs that are executed in respnse to some kind of event that occures in database.
- Triggers fires in response to a DML statement (insert, update delete) on specified table.
- Powerful mechanism for ensuring the integrity of data

```
Create trigger trigger_name
{before|after}
{update|insert|delete}
On table_name
For each row
Trigger statements
```

- Before|after specifies weather triger fires before of after the DML statement itself has been executed.
- Update|insert|delete specifies DML statement to which trigger is associated
- On table_name associates the trigger with a specific table
- For each row indicates that the trigger will be executed once for every row affected by the DML statement
- With after we are not able to modify the values about to modify the values about to be inserted into or updated with the table in question

Example of Row Level Trigger

```
Create trigger acctbalance
before update on account
For each row
Begin
Declare dummy int;
If new.balance < 0 then
Set new.balance=null;
End if;
```

```
End $$
```

Example of Statement Level Trigger

```
Create trigger acctbalance
before update on account
when new.balance < 0
Begin
Update account set new.balance = null where accno=new.accno;
End if;
End $$
```

```
CREATE TABLE employees_audit (
   id int(11) NOT NULL AUTO_INCREMENT,
   employeeNumber int(11) NOT NULL,
   lastname varchar(50) NOT NULL,
   changedon datetime DEFAULT NULL,
   action varchar(50) DEFAULT NULL,
   PRIMARY KEY (id)
);
```

```
DELIMITER $$
CREATE TRIGGER before_employee_update
   BEFORE UPDATE ON employees
   FOR EACH ROW BEGIN

   INSERT INTO employees_audit
   SET action = 'update',
       employeeNumber = OLD.emp_no,
      lastname = OLD.lastname,
      changedon = NOW();
END$$
DELIMITER ;
```

Output

- Row level and statement level trigger.

References:

1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition "Database Management Systems"
2. Avi Silberschatz , Henry F. Korth , S. Sudarshan, "Database System Concepts, Sixth Edition", ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Explain trigger concept?	2	2
2	Explain EER features?	2	2

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.

d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

e. **Output**: describe the results in few lines

f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**
  - Experiment completion (Timely)
  - Lab file (neatness and regularity)
  - Viva (from time to time)
  - Mock Practical Exam
  - Exam (end term): Practical + Viva
- **Assessment Methodology**
  - Timely completion of assignment- 2marks
  - Program demonstration- 4 marks
  - Viva-voce -2 marks
  - Timely submission of journal- 2 marks

ASSIGNMENT NO. 11

Aim

Write a PL/SQL block to implement all types of Cursors

Objective

To study and implement cursor.

Theory

A database stored program (stored module or stored routine) is a computer program that is stored within and executes within the database server.
When program is executed, it is executed within the memory address of a database server process or thread.

Syntax

Create procedure example()
Begin
//declare program variables
//statements
End;

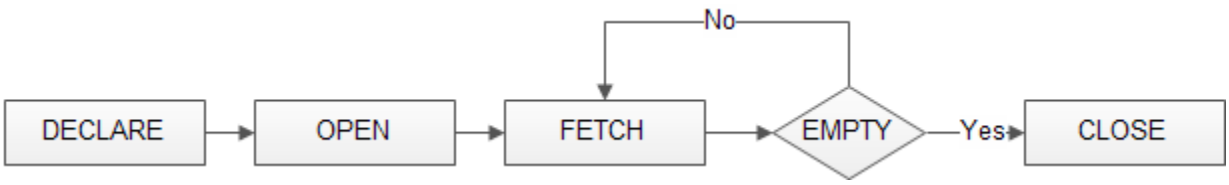
Call example1();

To handle a result set inside a stored procedure, you use a cursor. A cursor allows you to iterate a set of rows returned by a query and process each row accordingly.

MySQL cursor is read-only, non-scrollable and asensitive.

- **Read only**: you cannot update data in the underlying table through the cursor.
- **Non-scrollable**: you can only fetch rows in the order determined by the SELECT statement. You cannot fetch rows in the reversed order. In addition, you cannot skip rows or jump to a specific row in the result set.
- **Asensitive**: there are two kinds of cursors: asensitive cursor and insensitive cursor. An asensitive cursor points to the actual data, whereas an insensitive cursor uses a temporary copy of the data. An asensitive cursor performs faster than an insensitive cursor because it does not have to make a temporary copy of data. However, any change that made to the data from other connections will affect the data that is being used by an asensitive cursor, therefore, it is safer if you don't update the data that is being used by an asensitive cursor. MySQL cursor is asensitive.

Working of Cursor



DECLARE cursor_name CURSOR FOR SELECT_statement;

OPEN cursor_name;

FETCH cursor_name INTO variables list;

CLOSE cursor_name;

Procedure to display status of Employee (explicit cursor)

```
Delimiter $$
Drop procedure if exists empcursor $$
Create procedure empcursor()
Begin
Declare eno int;
Declare ename varchar(20);
Declare esal int;
Declare flag int default 0;
Declare c1 cursor for select eid, empname, Salary from employee;
Open c1;
emp_loop :  loop
                    Fetch c1 into eno, ename, esal;
                    If flag = 1 then
                    Leave emp_loop;
                    End if;

                        if esal < 10000 then
                        select 'C Class';
                        elseif esal > 10000 and esal < 20000 then
                        select 'B Class';
                        else
                        select 'A Class';
                        end if;
```

```
End loop emp_loop;
Close c1;
End $$
delimiter ;
```

Output

- Procedure and function.

References:

1. Raghu Ramkrishanan, Johannes Gehrke 4 th Edition "Database Management Systems"
2. Avi Silberschatz , Henry F. Korth , S. Sudarshan, "Database System Concepts, Sixth Edition", ISBN-13: 978-93-3290-138-4, MCGraw Hill

Frequently Asked Questions

Q. No	**Questions**	BT	CO
1	Explain cursors with example?	2	2
2	Explain implicit and explicit cursor?	2	2

Guidelines for Students

The experiments should be completed and get checked by the concerned teacher in the lab on or before the date of submission. After which the experiment will not be signed.

Every experiment must be included in the file in following format.

a. **Aim**: In this section write complete objective of the program you are going to make in the lab. This section specifies the complete description of the including problem analysis, input description, method used, fundamental concept and desired output format.

b. **Theory**: Write brief theory related to practical.

c. **Algorithm**: Write Algorithm for given task.

d. **Input**: Write input test data/ or program that are used to test program objective to see whether program is achieving the given objective or not.

e. **Output**: describe the results in few lines

f. **Conclusion**: Write complete conclusion whether what the student has learned from this experiment.

g. **Source Code**: Submit in the form of soft copies.

- **Marking criteria.**

  Experiment completion (Timely)
  Lab file (neatness and regularity)
  Viva (from time to time)
  Mock Practical Exam
  Exam (end term): Practical + Viva

- **Assessment Methodology**

  Timely completion of assignment- 2marks
  Program demonstration- 4 marks
  Viva-voce -2 marks
  Timely submission of journal- 2 marks