|  |  |  |
| --- | --- | --- |
| Description: Description: Description: Description: Description: I:\ALL PICTURE\JSPM LOGO.JPG | JSPM’s | Description: BSIOTR_Logo |
| **Bhivarabai Sawant Institute of Technology and Research** |
| ***(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)***  Gat. No.720, Pune-Nagar Road, Wagholi, Pune-412207 |

**Department of Information Technology**

**Database Management System Lab**

**Subject Code: 214456**

A.Y.2021-22

SE IT (2019 Course)

Semester I

LABORATORY MANUAL

Subject Incharge:- Prof. Rekha Kotwal

**Vision of Institute:**

**”To satisfy the aspirations of youth force, who wants to lead nation towards prosperity through techno-economic developments”**

**Mission of Institute:**

**"To provide, nurture and maintain an environment of high academic excellence, research and entrepreneurship for all aspiring students which will prepare them to face global challenges, maintaining high ethical and moral standard”**

**Vision of Department:**

|  |
| --- |
| **“To be a nucleus nurturing learner to cater current & future digital needs”** |
| **Mission of Department:** |

**1. Educating aspirants to fulfill technological and social needs through effective teaching learning process.**

**2. Imparting IT skills to envelop innovative solutions catering needs of multidisciplinary**

**INDEX OF LAB EXPERIMENTS**

|  |  |  |
| --- | --- | --- |
| **LAB EXPT**  **.NO** | **Problem Definition/Statement** | **Last Date**  **Of Completion** |
|  | **Group A: Study of Databases** |  |
| 1. | Study of MySQL Open source software. Discuss the characteristics like efficiency, scalability, performance and transactional properties |  |
| 2. | Install and configure client and server of MySQL.(Show all commands and necessary steps for installation and configuration) |  |
| 3. | Study of SQLite: What is SQLite? Uses of Sqlite. Building and installing SQLite |  |
|  | | |
|  | **Group B: MySQL** |  |
| 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. |  |
|  | | |
|  | **Group C: PL/SQL** |  |
| 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. |  |

**Prof. Rekha Kotwal**

**Subject Co-ordinator Head of Department**

**Assignment: 1**

**AIM:** Study of MySQL Open source software.

PROBLEM STATEMENT / DEFINITION:

Study of MySQL Open source software & its characteristics like

* Efficiency
* Scalability
* Performance and
* Transactional properties

OBJECTIVE:

1. To understand the fundamental concepts of database management system
2. To give systematic database design approaches covering conceptual design, logical design and an overview of physical design
3. To study of advantages of various SQL Databases.
4. To compare the different database systems based on points like efficiency, scalability, characteristics and performance.

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.

* What is DBMS?

A software system that enables users to define, create, maintain, and control access to the database. The DBMS is the software that interacts with the users‟ application programs and the database.

Typically, a DBMS provides the following facilities:

* It allows users to define the database, usually through a **Data Definition Language** (DDL). The DDL allows users to specify the data types and structures and the constraints on the data to be stored in the database.
* It allows users to insert, update, delete, and retrieve data from the database, usually through a **Data Manipulation Language (DML).** The most common query language is the Structured Query Language (SQL, pronounced „S-Q-L‟, or sometimes „See-Quel‟), which is now both the formal and de facto standard language for relational DBMSs.
* It provides controlled access to the database. For example, it may provide:
  + A security system, which prevents unauthorized users accessing the database;
  + An integrity system, which maintains the consistency of stored data;
  + A concurrency control system, which allows shared access of the database;
  + A recovery control system, which restores the database to a previous consistent state following a hardware or software failure
  + A user-accessible catalog, which contains descriptions of the data in the database.
* Advantages and Disadvantages of DBMSs
  + Data redundancy and inconsistency
  + Difficulty in accessing data
  + Data isolation
  + Integrity Problems
  + Atomicity problem
  + Concurrent-access anomalies
  + Security Problem
  + Reduced application development time
  + Uniform data administration
  + Recovery from crashes.
* Disadvantages
  + Complexity
  + Size
  + Additional hardware costs
  + Cost of DBMS
  + Performance
  + Higher impact of failure
* 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 relation with collection of data.
* **Column:** Itis a field or attribute for that relation.
* **Row:** Tuple or record) is a group of related data,
* **Primary key:** A candidate key chosen as the principal means of identifying tuples within a relation
* **Foreign key:** A relation schema may have an attribute that corresponds to the primary key of another relation. The attribute is called a foreign key.
* **Instance:** Thecollection of information stored in the database at a particular moment is called an instance of the database.
* **Schema:** The overall design of the database is called the database schema.

DBMS: A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily

accessible.

**Popular types of DBMSes:**

Popular database models and their management systems include:

**Relational database** management system (RDMS) - adaptable to most use cases, but RDBMS Tier-1 products can be quite expensive.

**NoSQL DBMS** - well-suited for loosely defined data structures that may evolve over time.

**In-memory database management system (IMDBMS)** - provides faster response times and better performance.

**Columnar database management system (CDBMS)** - well-suited for da warehoatuses that have a large number of similar data items.

**Cloud-based data management system** - the cloud service provider is responsible for providing and maintaining the DBMS.

**RELATIONAL DATABASE:**

RDBMS stands for **R**elational **D**atabase **M**anagement **S**ystem. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model

RDBMSs have been a common choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data, and other applications since the 1980s. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use. However, relational databases have received unsuccessful challenge attempts by object database management systems in the 1980s and 1990s (which were introduced trying to address the so- called object-relational impedance mismatch between relational databases and object-oriented application programs) and also by XML database management systems in the 1990s.

Despite such attempts, RDBMSs keep most of the market share, which has also grown over the years.

SQL:

SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have

their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database. This tutorial will provide you with the instruction on the basics of each of these commands as well as allow you to put them to practice using the SQL Interpreter.

MySQL Database:

“MySQL is a system that helps store and manage data efficiently. Database generally stores data in a structured fashion. It is written in C and C++, and it has been tested with a variety of compilers to check for bugs and inconsistencies.”

* 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 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 datasets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, upto50millionrowsormoreinatable. The default file size limit for atableis4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8million terabytes(TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments

## Open-Source

MySQL is an open-source relational database management system (RDBMS) which means this software can be downloaded, used and modified by anyone. It is free-to-use and easy-to-understand. The source code of MySQL can be studied, and changed based on the requirements Its name is a combination of "My", the name of co-founder Michael Widenius' daughter,and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements.

## Scalable

Scalability refers to the ability of systems to work easily with small amounts of data, large amounts of data, clusters of machines, and so on. MySQL server was developed to work with large databases. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality. It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, we can increase this number to a theoretical limit of 8 TB of data.

## Secure

It provides a secure interface since it has a password system which is flexible, and ensures that it is verified based on the host before accessing the database. The password is encrypted while connecting to the server. MySQL consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.

## Data Types

It contains multiple data types such as unsigned integers, signed integers, float (FLOAT), double (DOUBLE), character (CHAR), variable character (VARCHAR), text, blob, date, time, timestamp, year, and so on.

## Client and Utility Programs

MySQL server also comes with many client and utility programs. This includes Command line programs such as ‘mysqladmin’ and graphical programs such as ‘MySQL Workbench’. MySQL client programs are written in a variety of languages. MySQL follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they can query data, save changes, etc.Client library (code encapsulated in a module) can be written in C or C++ and would be available for clients that have C bindings.

**High Performance**

MySQL is faster, more reliable, and cheaper because of its unique storage engine architecture. It provides very high-performance results in comparison to other databases without losing an essential functionality of the software. It has fast loading utilities because of the different cache memory. MySQL stores data efficiently in the memory ensuring that data is consistent, and not redundant. Hence, data access and manipulation using MySQL is quick.

.

CONCLUSION:

The study of MYSQL open source software has been done. Also its various features like scalability, security etc studied.

Assignment: 2

**AIM:** Install and configure client and server of MySQL.(Show all commands and necessary steps for installation and configuration)

PROBLEM STATEMENT / DEFINITION:

Installation and configuration of client and server for :

* + MySQL (RDBMS)

OBJECTIVE:

To study installation & configuration of MySQL database.

THEORY:

MySQL is an open-source relational database management system that works on many platforms. It provides multi-user access to support many storage engines and is backed by Oracle. So, you can buy a commercial license version from Oracle to get premium support services.

The features of MySQL are as follows:

* Ease of Management – The software very easily gets downloaded and also uses an event scheduler to schedule the tasks automatically.
* Robust Transactional Support – Holds the ACID (Atomicity, Consistency, Isolation, Durability) property, and also allows distributed multi-version support.
* Comprehensive Application Development – MySQL has plugin libraries to embed the database into any application. It also supports stored procedures, triggers, functions, views and many more for application development. You can refer to the [RDS Tutorial](https://www.edureka.co/blog/rds-aws-tutorial/), to understand Amazon’s RDBMS.
* High Performance – Provides fast load utilities with distinct memory caches and table index partitioning.
* Open Source & 24 \* 7 Support –  This RDBMS can be used on any platform and offers 24\*7 support for open source and enterprise edition.
* Secure Data Protection – MySQL supports powerful mechanisms to ensure that only authorized users have access to the databases.
* High Availability – MySQL can run high-speed master/slave replication configurations and it offers cluster servers.
* Scalability & Flexibility – With MySQL you can run deeply embedded applications and create data warehouses holding a humongous amount of data.

**Installation of MySQL:**

## Step I: Download MySQL Installer for Windows:

## The MySQL Installer for Windows helps you control the installation process by providing a user-friendly interface. It also guides you through the steps needed to configure MySQL. Access your [Windows server](https://phoenixnap.com/servers/dedicated) and download the MySQL Installer. A free Community edition MySQL Installer is available from the official page: <https://dev.mysql.com/downloads/installer/>

## Select and download your preferred version. In this example, we selected the Full MySQL Package.

## After selecting a version, you are provided with the option of signing up for a MySQL Community account. If you are not interested, select the *No thanks, just start my download* option at the bottom of the page.

## Step II: Set Up MySQL Installer for Windows:

## After accepting the Oracle license agreement terms, the first screen you encounter allows you to define which MySQL products are going to be installed. You can choose between several predefined options or create your custom setup type.

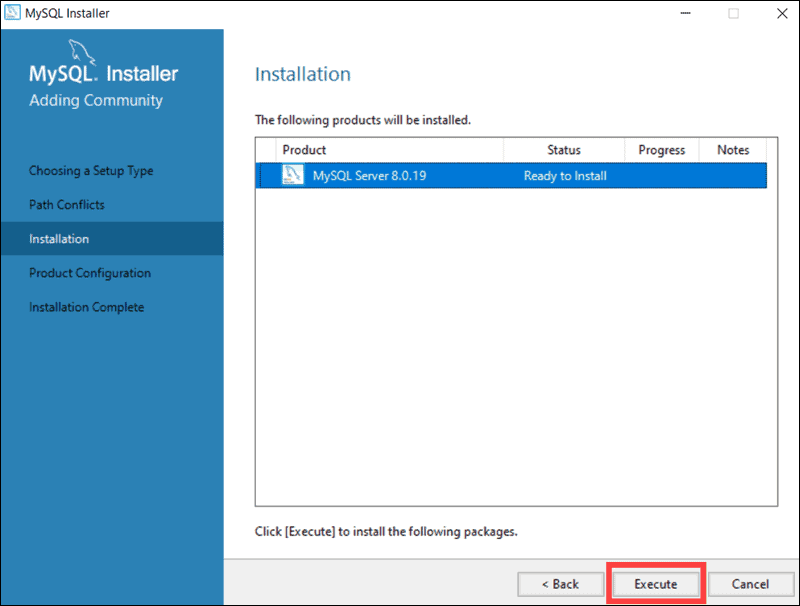
After accepting the Oracle license agreement terms, the first screen you encounter allows you to define which MySQL products are going to be installed. You can choose between several predefined options or create your custom setup type.

* Developer Default installs all the tools you need to develop and micromanage your MySQL databases effectively.
* Server Only is used to install an instance of the MySQL Server and forgo other MySQL products.
* Client Only installs all products except the MySQL Server and associated tools.
* The Full configuration installs all available MySQL products.

A Custom setup allows you to select the individual elements that are to be installed and alter predefined default settings.

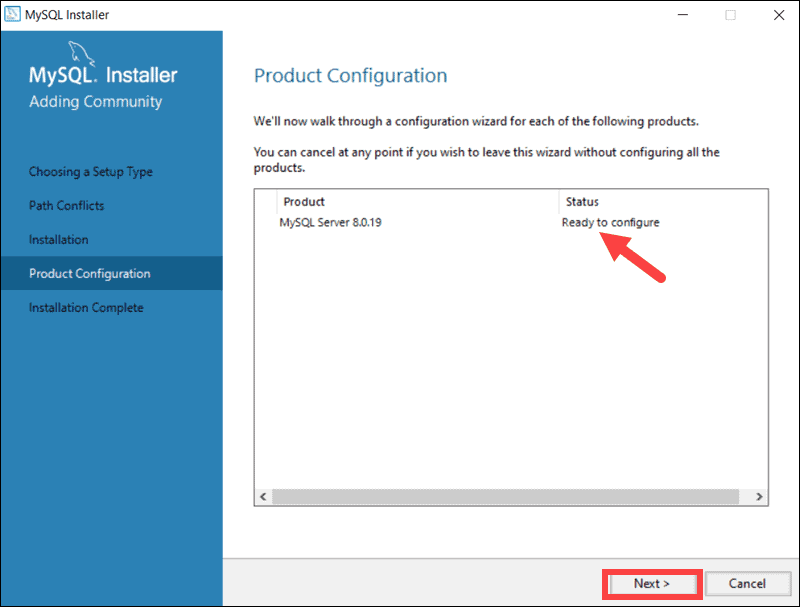
In the example below, we select the Server Only option and click Next.

## Dropdown allows you to select a MySQL installation configuration.

The **MySQL Installer auto-resolves issues** and installs the latest binary compatible version of missing software. You are now ready to start the installation process in earnest. Click **Execute** to begin the installation process.Once the status of the installation status is labeled as *Complete,* you are ready to configure the MySQL database.

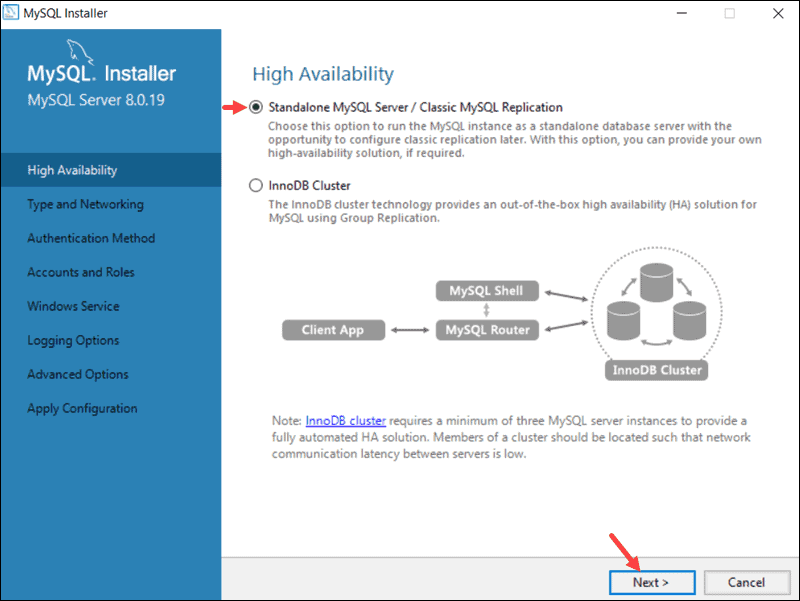
## Step-III Configure MySQL Server on Windows:

The MySQL Server 8.0.19 is now ready to be configured. Initiate the process by clicking **Next**.



### **1. High Availability**

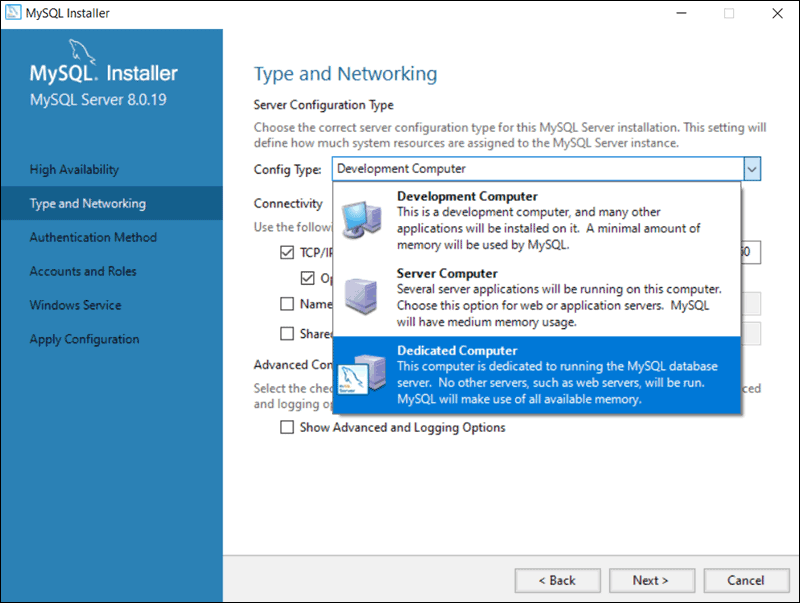
The first configuration option affects database availability. It allows you to decide if you want to set up a Standalone MySQL Server or an InnoDB server cluster to improve availability. In this instance, we selected the classic, single server option.



### **2. Type and Networking**

The **Type and Networking** section is used to define several essential features. The *Config Type* option lets you choose between three server configuration types.  *Development Computer*, *Server Computer*, and *Dedicated Computer* define whether the server is dedicated solely to running your MySQL database or is going to share the underlying system with other applications.

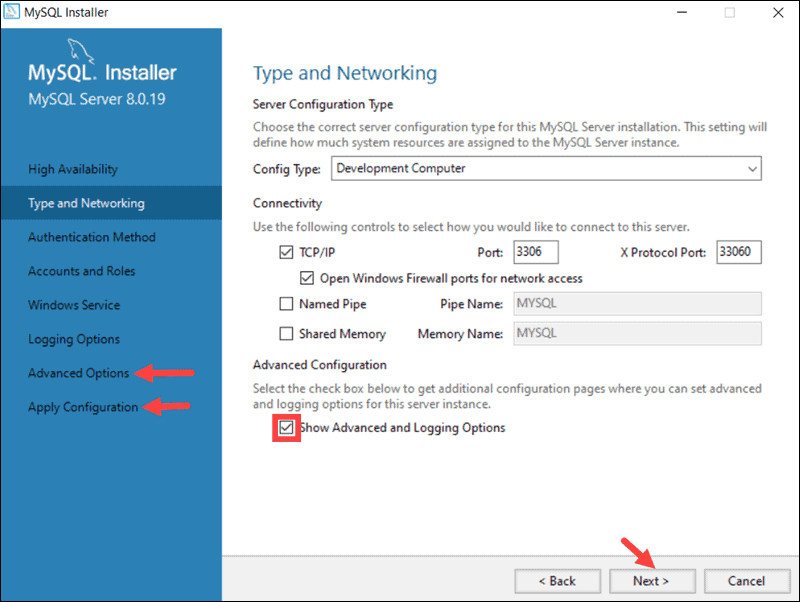
In this example, we decided to create a dedicated MySQL server.



The Type and Networking tab can also define the port the MySQL server is listening on. The default setting is port number 3306 and can be changed to suit your needs.

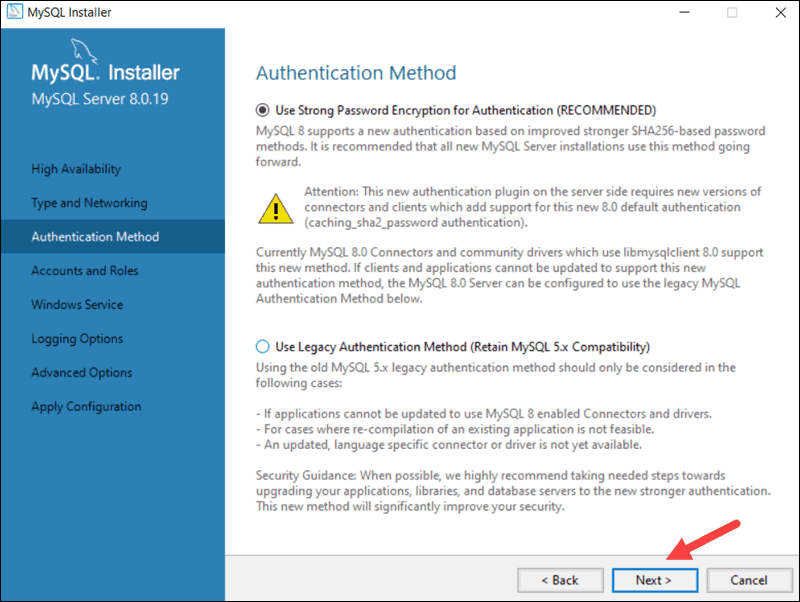
By checking the Show Advanced and Logging Option box, you can set additional logging options at a later stage.

Click **Next**once you’ve selected the options you feel meet your requirements.



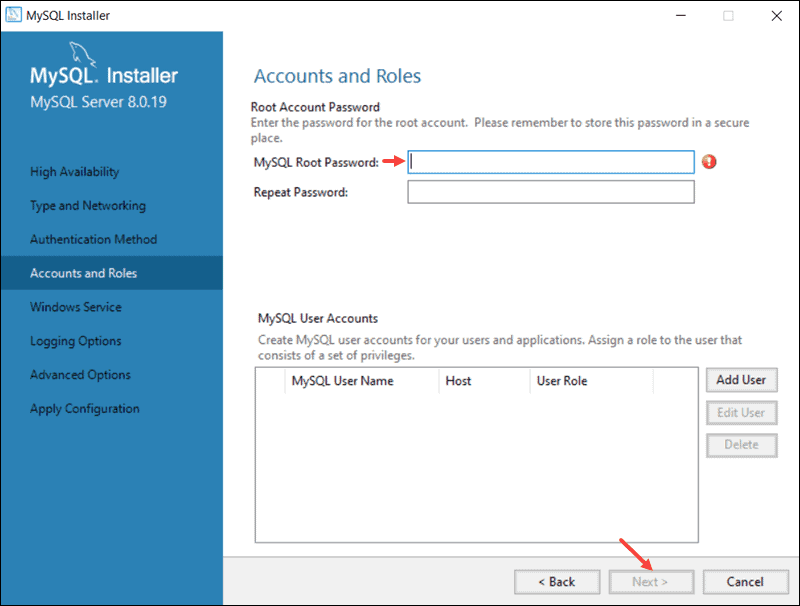
### **3. Authentication Method**

It is possible to choose between two authentication methods, the recommended *Strong Password Encryption*, and the *Legacy Authentication Method*. Select the recommended **Use Strong Password Authentication** option.



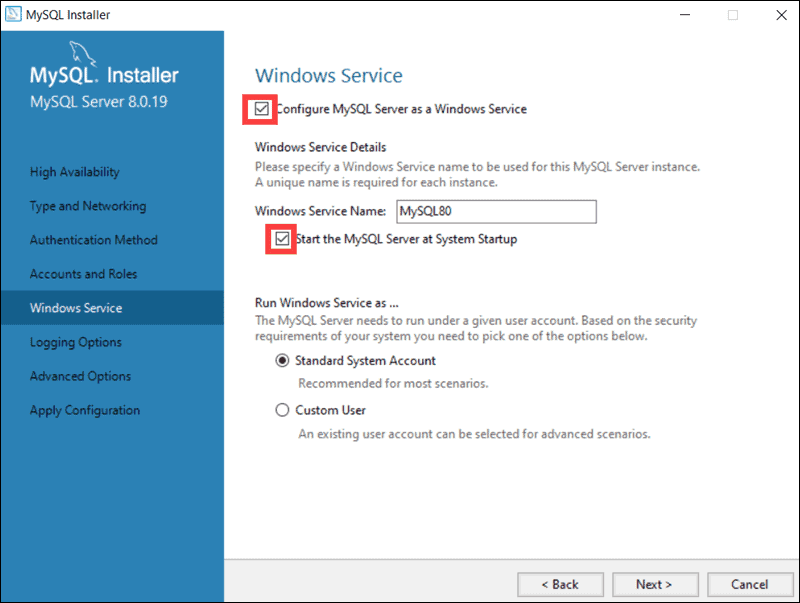
### **4. Accounts and Roles**

You are now prompted to enter a password for your MySQL root user. You can also create additional roles for various users and purposes. This is only an initial setup, and credentials can be edited once the installation is complete.



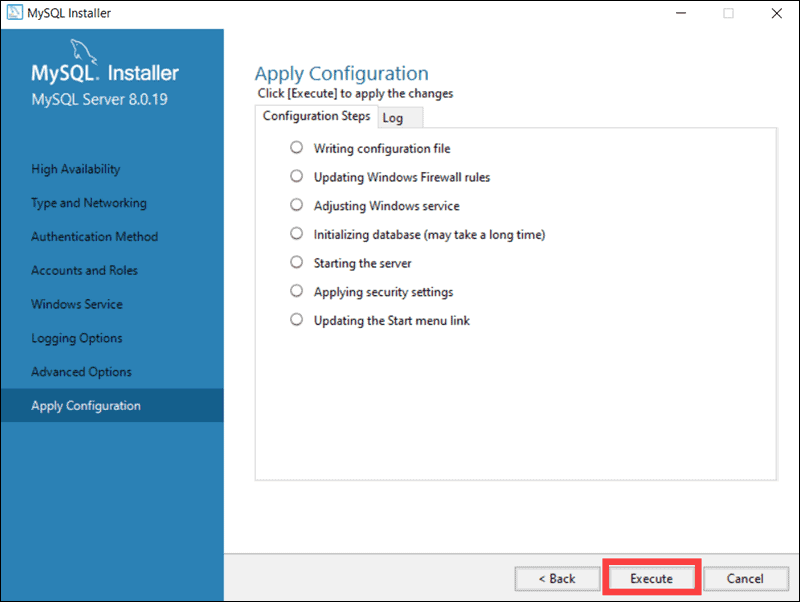
### **5. Windows Service:**

By defining MySQL as a Windows Service, it can now start automatically whenever the Windows system boots. If you decide to start MySQL as an executable application, you would need to configure it manually.

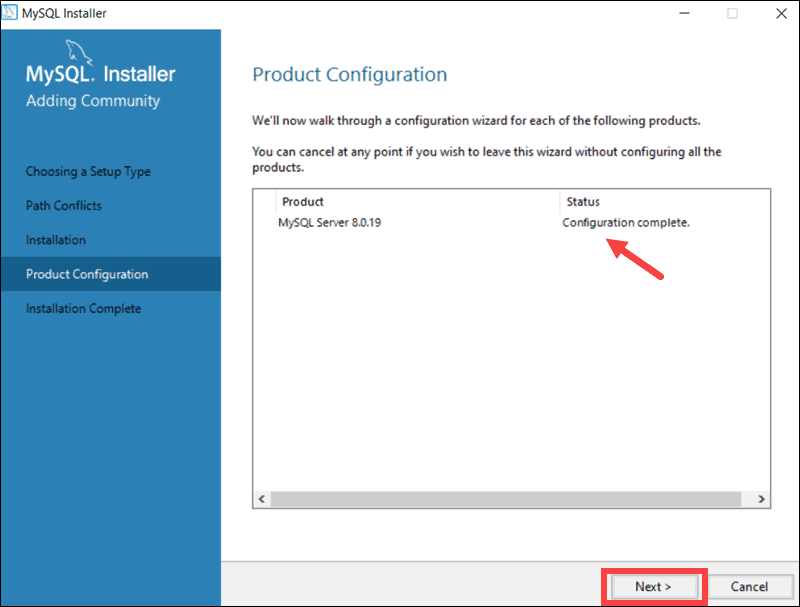


### **6. Apply Configuration**

You have successfully configured the MySQL server and need to confirm for the MySQL Installer to apply the configuration. An overview of the configurations steps appears on the screen. Click Execute to apply the configuration.



The system informs once the configuration process is completed. Select ****Next**** to continue the installation process.



## Step-IV Complete MySQL Installation on Windows Server:

## After clicking Next*,*you are given the option to copy the installation process log to the Windows Clipboard. Copy log of installation process on Clipboard and finish MySQL installation on Windows.

## Start MySQL Server on Windows:

If you need to start the MySQL Server on Windows for the first time enter the following command in the Windows Command Prompt:

"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --console

The path in this command is the default installation folder. In case you have installed MySQL in a different folder, the command needs to reflect that to launch the **mysqld** executable file successfully.

The **--console** option displays output directly on your console. Omitting this option sends the output directly to the MySQL logs.

**Conclusion**

Installing MySQL on a Windows server is simple when using the **MySQL Installer tool**.

**Assignment: 3**

**AIM:** Study the SQLite database and its uses and installation.

PROBLEM STATEMENT / DEFINITION:

1. Study the SQLite database and its uses.
2. Elaborate on building and installing of SQLite.

OBJECTIVE:

1. To study SQLite database and its uses.
2. To study installation & configuration of SQLite database.

THEORY:

SQLite:

SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. SQLite is the most used database engine in the world.

SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.[5]

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.[6] SQLite has bindings to many programming languages.

#### Installing SQLite:

1. type in the following command –

$ sudo apt-get install sqlite3 libsqlite3-dev

1. After installation check installation, sqlite terminal will give you a prompt and version info – naved@neo:~$ sqlite3

SQLite version 3.8.2 2013-12-06 14:53:30

Enter &quot;.help&quot; for instructions

Enter SQL statements terminated with a &quot;;&quot; sqlite&gt;

1. To quit – sqlite&gt; .quit
2. Go to desired folder and create database – naved@neo:~$ sqlite3 database\_name.db

It‟ll create database\_name.db in the folder you‟ve given the command.

1. To check whether the database has been created give the following command in sqlite3 terminal –

sqlite&gt; .databases

**Uses of SQLite:**

SQLite is not directly comparable to client/server SQL database engines such as MySQL, Oracle, PostgreSQL, or SQL Server since SQLite is trying to solve a different problem.

Client/server SQL database engines strive to implement a shared repository of enterprise data. They emphasis scalability, concurrency, centralization, and control. SQLite strives to provide local data storage for individual applications and devices. SQLite emphasizes economy, efficiency, reliability, independence, and simplicity.

SQLite does not compete with client/server databases.

* + Embedded devices and the internet of things
  + Application file format
  + Websites
  + Data analysis
  + Cache for enterprise data
  + Server-side database
  + File archives
  + Replacement for *ad hoc* disk files
  + Internal or temporary databases
  + Stand-in for an enterprise database during demos or testing
  + Education and Training
  + Experimental SQL language extensions

REFERENCE URL:

1. https://sysads.co.uk/2014/08/05/install-sqlite-database-browser-3-2-0-on-ubuntu-14-04/
2. <http://www.sqlitetutorial.net/>

CONCLUSION:

Studied of installation and configuration steps of SQLite database.

Assignment: 4

**AIM:**

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

PROBLEM STATEMENT / DEFINITION:

Design & Develop DB for “Order Management System” with all the constraints. (There must be At least 3 entities and relationships between them.) The statement should use SQL objects such as Table, View, Index, and Sequence. Draw suitable ER/EER diagram for the system.

Apply DCL and DDL commands to convert ER/EER diagram to tables.

OBJECTIVE:

1. To understand the concept of ER diagram.
2. To understand the details of basic ER model
3. To understand the technique for converting ER diagram into tables
4. Analyze the reflected relationship and constraints
5. To understand use of DDL , DCL

THEORY:

Basic concepts of ER Diagram:

A database can be modeled as a collection of entities and relationship among Entities. Entity: entity is an object that exists and is distinguishable from other objects. Example: specific person, company, event, plant.

**Entity set:** An entity set is a set of entities of the same type that share the same properties. Example: set of all persons, companies, trees, holidays

**Attributes:** Entities have attributes.

Example: people have names and addresses Attribute types:

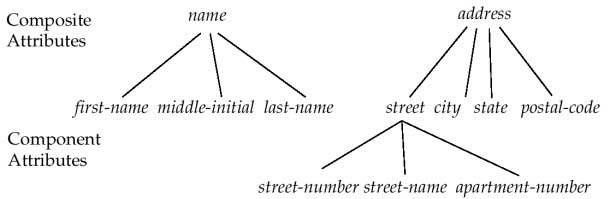
1. Simple: e.g. roll no

2. Composite attributes: e.g. address name,

3. Single-valued: roll no

1. .Multi-valued attributes: e.g. Phone-numbers
2. .Derived attributes: Can be computed from other attributes

E.g. age, given date of birth



**Relationship:** A relationship is an association among several entities

Example: Hayes depositor A-102customer entity relationship set account entity.

**Relationship set**:

A relationship set is a mathematical relation among n ≥ 2 entities, each taken from entity sets

{(e1, e2, en) | e1 ε E1, e2 ε E2, en ε En} where (e1, e2, en) is a relationship Example: (Hayes, A-102) ε depositor.

**Mapping Cardinalities:**

Express the number of entities to which another entity can be associated via relationship set. Most useful in describing binary relationship sets. For a binary relationship set the mapping cardinality must be one of the following types:

1. One to one

2. One to many

3. Many to one 4. Many to many

**Symbolic notations:**

Components to draw entity relationship diagram.

Rectangles: represent entity sets.

Diamonds: represent relationship sets.

Lines: link attributes to entity sets and entity sets to relationship sets.

Ellipses: represent attributes

Double ellipses: represent multivalued attributes.

Dashed ellipses: denote derived attributes.

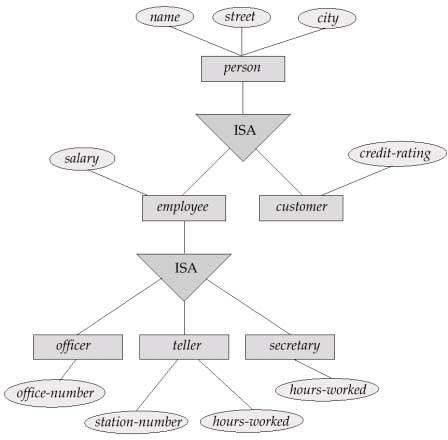
Underline: indicates primary key attributes (will study later)

**Extended ER Features:**

1. **Specialization:**

Top-down design process; we designate sub groupings within an entity set that are distinctive from other entities in the set. These sub groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher- level entity set. Depicted by a triangle component labeled ISA (E.g. customer “is a” person).

1. **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked

To convert an ER Diagram into Database tables.

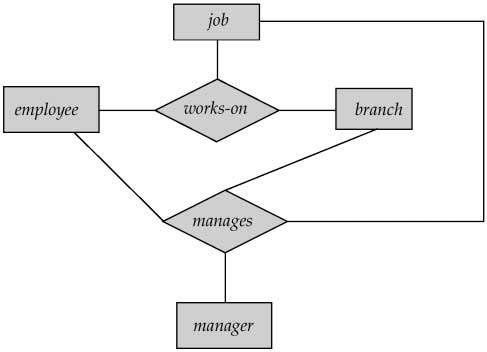
1. **Generalization:**

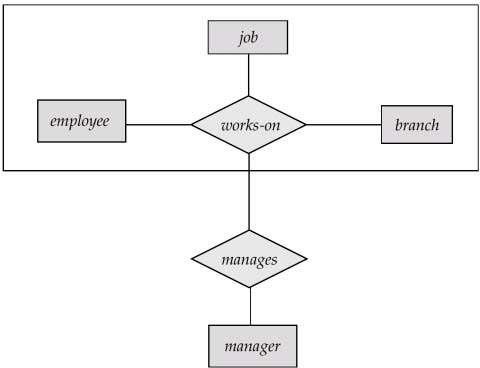
A bottom-up design process – combine a number of entities sets that share the same features into a higher-level entity set. Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way. The terms specialization and generalization are used interchangeably.

1. **Aggregation:**

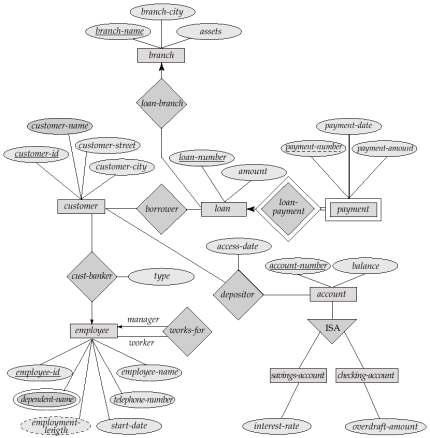
Consider the ternary relationship works-on, which we saw earlier. Suppose we want to record managers for tasks performed by an employee at a branch Relationship sets works-on and manages represent overlapping information. Every manages relationship corresponds to a works- on relationship However, some works-on relationships may not correspond to any manages relationships. So we can‟t discard the works-on relationship. Eliminate this redundancy via aggregation. Treat relationship as an abstract entity. Allows relationships between relationships. Abstraction of relationship into new entity without introducing redundancy, the following diagram represents:

An employee works on a particular job at a particular branch an employee, branch, and job combination may have an associated manager.





**Example:** E-R Diagram for Bank organization



1. Reduction of ER Schema to tables

Primary keys allow entity sets and relationship sets to be expressed uniformly as tables which represent the contents of the database. A database which conforms to an E-R diagram can be represented by a collection of tables. For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set. Each table has a number of columns (generally corresponding to attributes), which have unique names.Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.A strong entity set reduces to a table with the same attributes.

Ex. Customer (customer-id, customer-name, customer-street, customer-city)Schema can be reduce as follows.

**CONCLUSION:**

Understand to draw ER diagram and EER diagram for the given database.

Assignment: 5

**AIM:**

Design and implement a database using DDL statements and apply normalization on them

PROBLEM STATEMENT /DEFINITION

* + Design and implement a database using DDL statements and apply normalization on it.

OBJECTIVE:

1. To understand use of various DDL queries.
2. To understand how the DDL changes the structure of the table like creating a table, deleting a table, altering a table,

THEORY:

Introduction to SQL:

The Structured Query Language (SQL) comprises one of the fundamental building blocks of modern database architecture. SQL defines the methods used to create and manipulate relational databases on all major platforms. SQL comes in many flavors. Oracle databases utilize their proprietary PL/SQL. Microsoft SQL Server makes use of Transact-SQL. However, all of these variations are based upon the industry standard ANSI SQL.

SQL commands can be divided into two main sublanguages.

1. Data Definition Language
2. Data Manipulation Language

1.1 DATA DEFINITION LANGUAGE (DDL)

It contains the commands used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project.

DDL Commands:

a) Create table command:

**Syntax :**

CREATE TABLE table name (column\_name1 data type (size), column\_name2 data\_type(size),

....... )

Example 1

This example demonstrates how you can create a table named “Person", with four columns. The column names will be "LastName", "FirstName", "Address", and "Age":

CREATE TABLE Person (LastName varchar, FirstName varchar, Address varchar, Age int ) This example demonstrates how you can specify a maximum length for some columns:

Example 2

CREATE TABLE Person (LastName varchar(30), FirstName varchar, Address varchar, Age int(3))

Creating table from another (existing table) table: Syntax

CREATE TABLE tablename [(columnname,column name)]] AS SELECT columnname,columnname FROM tablename;

b. Alter table command:

Once table is created within a database, we may wish to modify the definition of that table.The ALTER command allows making changes to the structure of a table without deleting and recreating it.

Syntax

ALTER TABLE table\_name ADD (newcolumn\_name1 data\_type(size), newcolumn\_name2 data\_type(size), )

Example

ALTER TABLE personal\_info ADD salary money null

This example adds a new attribute to the personal\_info table -- an employee's salary. The "money" argument specifies that an employee's salary will be stored using a dollars and cents

format. Finally, the "null” keyword tells the database that it's OK for this field to contain no value for any given employee.

c. Drop table command:

DROP command allows us to remove entire database objects from our DBMS. For example, if we want to permanently remove the personal\_info table that we created, we'd use the following command:

Syntax

DROP TABLE table\_name;

Example

DROP TABLE personal\_info;

**d. TRUNCATE:**

 It is used to delete all the rows from the table and free the space containing the table.

**Syntax:**

TRUNCATE TABLE table\_name;

**Example:**

TRUNCATE TABLE EMPLOYEE;

**e.** [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/)**:**

This is used to rename an object existing in the database.

Syntax:

Rename Table old\_table\_name to new\_table\_name;

Example:

Rename table student to student1;

**Example**

All Basic commands of MySql .Like :

mysql> create database ManageCust;

Query OK, 1 row affected (0.00 sec) // to user ur own database, other than default.

mysql> use ManageCust;

Database changed

mysql> QUIT To exit the MySQL Shell, just type QUIT or EXIT: mysql> exit

mysql> SHOW TABLES;

mysql> DESCRIBE <Table Name>;

CONCLUSION:

The structure of table is designed by applying all the DDL commands.

Assignment: 6

**AIM:** Create Table with primary key and foreign key constraints.

PROBLEM STATEMENT /DEFINITION

Design and implement a database by creating table with primary key and foreign key constraints like –

a. Alter table with add and modify

b. Drop table

OBJECTIVE:

1. To understand how to Alter table using DDL Alter command.
2. To understand how to Drop table using DDL drop command

THEORY:

DATA INTEGRITY:

Enforcing data integrity ensures the quality of the data in the database. For example, if an employee is entered with an employee\_id value of “123”, the database should not allow another employee to have an ID with the same value. Two important steps in planning tables are to identify valid values for a column and to decide how to enforce the integrity of the data in the column. Data integrity falls into four categories:

* Entity integrity
* Domain integrity
* Referential integrity
* User-defined integrity

There are several ways of enforcing each type of integrity.

|  |  |
| --- | --- |
| Integrity type | Recommended options |
| Entity | PRIMARY KEY constraintUNIQUE constraint |
| Domain | FOREIGN KEY constraint CHECK constraint NOT  NULL |
| Referential | FOREIGN KEY constraint  CHECK constraint |
| User-defined | All column- and table-level constraints in CREATE TABLE StoredProcedures Triggers |

ENTITY INTEGRITY:

Entity integrity defines a row as a unique entity for a particular table. Entity integrity enforces the integrity of the identifier column(s) or the primary key of a table (through indexes, UNIQUE constraints, PRIMARY KEY constraints, or IDENTITY properties).

DOMAIN INTEGRITY:

Domain integrity is the validity of entries for a given column. You can enforce domain integrity by restricting the type (through data types), the format (through CHECK constraints and rules), or the range of possible values (through FOREIGN KEY constraints, CHECK constraints, DEFAULT definitions, NOT NULL definitions, and rules).

REFERENTIAL INTEGRITY:

Referential integrity preserves the defined relationships between tables when records are entered or deleted. In Microsoft® SQL Server™, referential integrity is based on relationships between foreign keys and primary keys or between foreign keys and unique keys. Referential integrity ensures that key values are consistent across tables. Such consistency requires that there be no references to nonexistent values and that if a key value changes, all references to it change consistently throughout the database.

PRIMARY KEY CONSTRAINT:

**Definition:** - The primary key of a relational table uniquely identifies each record in the table. A primary key constraint ensures no duplicate values are entered in particular columns and that NULL values are not entered in those columns.

Properties :

1. No duplicate values are allowed, i.e. Column assigned as primary key should have UNIQUE values only.
2. NO NULL values are present in column with Primary key. Hence there is Mandatory value in column having Primary key.
3. Only one primary key per table exist although Primary key may have multiple columns.
4. No new row can be inserted with the already existing primary key.
5. Classified as : a) Simple primary key that has a Single column 2) Composite primary key has Multiple column.
6. Defined in Create table / Alter table statement.

The primary key can be created in a table using PRIMARY KEY constraint. It can be created at two levels.

* + Column
  + Table.

**SQL PRIMARY KEY at Column Level :**  
If Primary key contains just one column, it should be defined at column level. The following code creates the Primary key “ID” on the person table.

**Syntax :**

**Create Table Person**

**(**

**Id int NOT NULL PRIMARY KEY,**

**Name varchar2(20),**

**Address varchar2(50)**

**);**

Here NOT NULL is added to make sure ID should have unique values. SQL will automatically set null values to the primary key if it is not specified.

To verify the working of Primary key :

**Insert into Person values(1, 'Ajay', 'Mumbai');**

Output :

1 row created

Let’s see if you will insert the same values again.

**Insert into Person values(1, 'Ajay', 'Mumbai');**

Output :

Error at line 1: unique constraint violated

Since we are inserting duplicate values, an error will be thrown since the Primary key “Id” can contain only unique values.

Example-3 :

**Insert into Person values('' ” , 'Ajay', 'Mumbai');**

**Output :**

Error at line 1: cannot insert Null into<"user"."Person"."ID">

Primary Key cannot contain Null Values so that too will throw an error.

1. NOT NULL CONSTRAINT:

This constraint ensures that NULL values are not entered in those columns.

1. UNIQUE CONSTRAINT:

This constraint ensures that no duplicate values are entered in those columns.

1. CHECK CONSTRAINT:

The CHECK constraint enforces column value restrictions. Such constraints can restrict a column, for example, to a set of values, only positive numbers, or reasonable dates.not working in mysql.

**SQL PRIMARY KEY at Table Level :**

Whenever the primary key contains multiple columns it has to be specified at Table level.

**Syntax:**

**Create Table Person**

**(Id int NOT NULL,**

**Name varchar2(20),**

**Address varchar2(50),**

**PRIMARY KEY(Id, Name)**

**);**

Here, you have only one Primary Key in a table but it consists of Multiple Columns(Id, Name). However, the Following are permissible.

Insert into Person values(1, 'Ajay', 'Mumbai');

Insert into Person values(2, 'Ajay', 'Mumbai');

Since multiple columns make up Primary Key so both the rows are considered different. SQL permits either of the two values can be duplicated but the combination must be unique.

**SQL PRIMARY KEY with ALTER TABLE :**  
Most of the time, Primary Key is defined during the creation of the table but sometimes the Primary key may not be created in the already existing table. We can however add Primary Key using Alter Statement.

**Syntax :**

**Alter Table Person add Primary Key(Id);**

To add Primary key in multiple columns using the following query.

**Alter Table Person add Primary Key(Id, Name);**

It is necessary that the column added as primary key MUST contain unique values or else it will be violated. An id cannot be made Primary key if it contains duplicate values. It violates the basic rule of Primary Key. Altering the table to add Id as a primary key that may contain duplicate values generates an error.

**Output :**

Error at line 1: cannot validate- primary key violated

Also, A column added as primary key using alter statement should not have null values. Altering table to add Id as primary key that may contain null values generates an error.

**Output :**

Error at line 1: column contains NULL values; cannot alter to NOT NULL

**DELETE PRIMARY KEY CONSTRAINT :**  
To remove Primary Key constraint on table use given SQL as follows.

**ALTER table Person DROP PRIMARY KEY;**

## SQL FOREIGN KEY Constraint:

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Look at the following two tables:

### Persons Table

|  |  |  |  |
| --- | --- | --- | --- |
| **PersonID** | **LastName** | **FirstName** | **Age** |
| 1 | Hansen | Ola | 30 |
| 2 | Svendson | Tove | 23 |
| 3 | Pettersen | Kari | 20 |

### Orders Table

|  |  |  |
| --- | --- | --- |
| **OrderID** | **OrderNumber** | **PersonID** |
| 1 | 77895 | 3 |
| 2 | 44678 | 3 |
| 3 | 22456 | 2 |
| 4 | 24562 | 1 |

The above table "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

## SQL FOREIGN KEY on CREATE TABLE

The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

**CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);**

## SQL FOREIGN KEY on ALTER TABLE:

To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

**ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);**

## DROP a FOREIGN KEY Constraint:

To drop a FOREIGN KEY constraint, use the following SQL:

**ALTER TABLE Orders  
DROP FOREIGN KEY FK\_PersonOrder;**

CONCLUSION:

The primary key and foreign key is created on table. Also altered table and removed table using drop table.

**Assignment: 7**

**AIM:** Write SQL queries on the database created already to implement various operations and use built in functions in the queries.

**PROBLEM STATEMENT /DEFINITION:**

**Write queries using all of the following operators-**

* Implementation of relational operators in SQL
* Boolean operators and pattern matching
* Arithmetic operations and built in functions
* Group functions /Aggregate Functions
* Processing Date and Time functions/Built in Functions
* Complex queries and set operators

**OBJECTIVE:**

1. To understand use of various DML, and DQL statement with clause and nested queries, DCL commands.
2. To understand use of various functions: Aggregate, Arithmetic and nested queries.
3. To understand the SQL Built in functions (now (), date (), day (), time () etc)

**THEORY:**

**SELECT COMMAND:**

The SELECT command is the most commonly used command in SQL. It allows database users to retrieve the specific information they desire from an operational database. Syntax

SELECT A1, A2…….. FROM table name WHERE predicate

A1, A2 is the list of attributes and predicate is the condition which must be satisfied by the resulting rows.

**Example 1**

It displays list of all last names in personal\_info table

SELECT last\_name FROM personal\_info

The command shown below retrieves all of the information contained within the personal\_info table. The asterisk is used as a wildcard in SQL. This means "Select everything from the personal\_info table."

**Example 2**

SELECT \* FROM account

Finally, the WHERE clause can be used to limit the records that are retrieved to those that meet specified criteria. The CEO might be interested in reviewing the personnel records of all highly paid employees. The following command retrieves all of the data contained within personal\_info for records that have a salary value greater than $50,000:

**Example 3**

SELECT \* FROM account WHERE balance > $50000

INPUT: A specific query set.

OUTPUT: Result of the given query set.

**AGGREGATE FUNCTIONS:**

If we want to summarize our data to produce top-level reports (For example, the purchasing manager may not be interested in a listing of all widget sales, but may simply want to know the number of widgets sold this month),

SQL provides aggregate functions to assist with the summarization of large volumes of data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OrderID | FirstName | LastName | Quantity | UnitPrice | Continent |
| 122 | John | Jacob | 21 | 4.52 | North America |
| 923 | Ralph | Wiggum | 192 | 3.99 | North America |
| 238 | Ryan | Johnson | 87 | 4.49 | Africa |
| 829 | Mary | Smith | 842 | 2.99 | North America |
| 824 | Elizabeth | Marks | 48 | 3.48 | Africa |
| 753 | James | Linea | 9 | 7.85 | North America |
| 942 | Alan | Jonas | 638 | 3.29 | Europe |

**1. SUM:**

It is used within a SELECT statement and, predictably, returns the summation of a series of values. If the widget project manager wanted to know the total number of widgets sold to date, we could use the Following query:

Example:

SELECT SUM (salary) AS TOTAL FROM account

Total

-----------

1837

**2. AVG:**

The AVG (average) function works in a similar manner to provide the mathematical average of a series of values. To find out the average amount of all orders placed on the North American continent.

Example:

SELECT AVG (balance) as AveragePrice FROM account WHERE branch-name=’delhi’;

**3. COUNT:**

SQL provides the COUNT function to retrieve the number of records in a table that meet given criteria. We can use the COUNT (\*) syntax alone to retrieve the number of rows in a table. Alternatively, a WHERE clause can be included to restrict the counting to specific records.

Example:

SELECT COUNT(\*) AS 'Number of Large Orders' FROM account WHERE BALANCE > 100000 ;

Above query displays how many orders are processed by company that requested over 100 widgets.

**4. MAX AND MIN:**

SQL provides Min and Max functions to locate the records containing the smallest and largest values for a given expression

The MAX () function returns the largest value in a given data series. We can provide the function with a field name to return the largest value for a given field in a table. To find the order in our database that produced the most revenue for the company, following query will be used:

Example:

SELECT MAX (BALANCE) as ‘largest balance’ FROM account;

The MIN () function functions in the same manner, but returns the minimum value for the expression.

## SQL DATE DATA TYPES:

**MySQL** comes with the following data types for storing a date or a date/time value in the database:

* DATE - format YYYY-MM-DD
* DATETIME - format: YYYY-MM-DD HH:MI:SS
* TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
* YEAR - format YYYY or YY

**SQL Server** comes with the following data types for storing a date or a date/time value in the database:

* DATE - format YYYY-MM-DD
* DATETIME - format: YYYY-MM-DD HH:MI:SS
* SMALLDATETIME - format: YYYY-MM-DD HH:MI:SS
* TIMESTAMP - format: a unique number

**Note:** The date types are chosen for a column when you create a new table in your database!

## SQL Working with Dates

You can compare two dates easily if there is no time component involved!

Assume we have the following "Orders" table:

|  |  |  |
| --- | --- | --- |
| **OrderId** | **ProductName** | **OrderDate** |
| 1 | Geitost | 2008-11-11 |
| 2 | Camembert Pierrot | 2008-11-09 |
| 3 | Mozzarella di Giovanni | 2008-11-11 |
| 4 | Mascarpone Fabioli | 2008-10-29 |

Now we want to select the records with an OrderDate of "2008-11-11" from the table above.

We use the following SELECT statement:

SELECT \* FROM Orders WHERE OrderDate='2008-11-11'

The result-set will look like this:

|  |  |
| --- | --- |
| **OrderId** | **ProductName** |
| 1 | Geitost |
| 3 | Mozzarella di Giovanni |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Now, assume that the "Orders" table looks like this (notice the time component in the "OrderDate" column):

|  |  |  |
| --- | --- | --- |
| **OrderId** | **ProductName** | **OrderDate** |
| 1 | Geitost | 2008-11-11 13:23:44 |
| 2 | Camembert Pierrot | 2008-11-09 15:45:21 |
| 3 | Mozzarella di Giovanni | 2008-11-11 11:12:01 |
| 4 | Mascarpone Fabioli | 2008-10-29 14:56:59 |

If we use the same SELECT statement as above:

SELECT \* FROM Orders WHERE OrderDate='2008-11-11'

we will get no result! This is because the query is looking only for dates with no time portion.

**Arithmetic function:**

A mathematical function executes a mathematical operation usually based on input values that are provided as arguments, and return a numeric value as the result of the operation. Mathematical functions operates on numeric data such as decimal, integer, float, real, smallint, and tinyint.

By default, the precision of built-in operations on float data type data is six decimal places.

## SQL Arithmetic Operators

The **Arithmetic Operators** perform the mathematical operation on the numerical data of the SQL tables. These operators perform addition, subtraction, multiplication, and division operations on the numerical operands.

**Following are the various arithmetic operators performed on the SQL data:**

1. SQL Addition Operator (+)
2. SQL Subtraction Operator (-)
3. SQL Multiplication Operator (+)
4. SQL Division Operator (-)
5. SQL Modulus Operator (+)

### **SQL Addition Operator (+)**

The **Addition Operator** in SQL performs the addition on the numerical data of the database table. In SQL, we can easily add the numerical values of two columns of the same table by specifying both the column names as the first and second operand. We can also add the numbers to the existing numbers of the specific column.

**Syntax of SQL Addition Operator:**

SELECT operand1 + operand2;

This example consists of an **Employee\_details** table, which has four columns **Emp\_Id, Emp\_Name, Emp\_Salary, and Emp\_Monthlybonus.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp Id** | **Emp Name** | **Emp Salary** | **Emp Monthlybonus** |
| 101 | Tushar | 25000 | 4000 |
| 102 | Anuj | 30000 | 200 |

* Suppose, we want to add **20,000** to the salary of each employee specified in the table. Then, we have to write the following query in the SQL:

SELECT Emp\_Salary + 20000 as Emp\_New\_Salary FROM Employee\_details;

In this query, we have performed the SQL addition operation on the single column of the given table.

* Suppose, we want to add the Salary and monthly bonus columns of the above table, then we have to write the following query in SQL:

SELECT Emp\_Salary + Emp\_Monthlybonus as Emp\_Total\_Salary FROM Employee\_details;

In this query, we have added two columns with each other of the above table.

## **SQL Subtraction Operator (-):**

The Subtraction Operator in SQL performs the subtraction on the numerical data of the database table. In SQL, we can easily subtract the numerical values of two columns of the same table by specifying both the column names as the first and second operand. We can also subtract the number from the existing number of the specific table column.

**Syntax of SQL Subtraction Operator:**

SELECT operand1 - operand2;

## **SQL Multiplication Operator (\*)**

The Multiplication Operator in SQL performs the Multiplication on the numerical data of the database table. In SQL, we can easily multiply the numerical values of two columns of the same table by specifying both the column names as the first and second operand.

**Syntax of SQL Multiplication Operator:**

SELECT operand1 \* operand2;

## **SQL Division Operator (/)**

The Division Operator in SQL divides the operand on the left side by the operand on the right side.

**Syntax of SQL Division Operator:**

SELECT operand1 / operand2;

## **SQL Modulus Operator (%)**

The Modulus Operator in SQL provides the remainder when the operand on the left side is divided by the operand on the right side.

**Syntax of SQL Modulus Operator:**

SELECT operand1 % operand2;

## **SQL Comparison Operators**

The **Comparison Operators** in SQL compare two different data of SQL table and check whether they are the same, greater, and lesser. The SQL comparison operators are used with the WHERE clause in the SQL queries

**Following are the various comparison operators which are performed on the data stored in the SQL database tables:**

1. SQL Equal Operator (=)
2. SQL Not Equal Operator (!=)
3. SQL Greater Than Operator (>)
4. SQL Greater Than Equals to Operator (>=)
5. SQL Less Than Operator (<)\
6. SQL Less Than Equals to Operator (<=)

## SQL arithmetic functions are:

|  |  |
| --- | --- |
| **Functions** | **Description** |
| [ABS()](https://www.w3resource.com/sql/arithmetic-functions/abs.php) | This SQL ABS() returns the absolute value of a number passed as an argument. |
| [CEIL()](https://www.w3resource.com/sql/arithmetic-functions/ceil-with-positive-value.php) | This SQL CEIL() will rounded up any positive or negative decimal value within the function upwards. |
| [FLOOR()](https://www.w3resource.com/sql/arithmetic-functions/floor-wth-positive-value.php) | The SQL FLOOR() rounded up any positive or negative decimal value down to the next least integer value. |
| [EXP()](https://www.w3resource.com/sql/arithmetic-functions/exp.php) | The SQL EXP() returns e raised to the n-th power(n is the numeric expression), where e is the base of natural algorithm and the value of e is approximately 2.71828183. |
| [LN()](https://www.w3resource.com/sql/arithmetic-functions/ln.php) | The SQL LN() function returns the natural logarithm of n, where n is greater than 0 and its base is a number equal to approximately 2.71828183. |
| [MOD()](https://www.w3resource.com/sql/arithmetic-functions/mod.php) | This SQL MOD() function returns the remainder from a division. |
| [POWER()](https://www.w3resource.com/sql/arithmetic-functions/power.php) | This SQL POWER() function returns the value of a number raised to another, where both of the numbers are passed as arguments. |

**CONCLUSION:**

Understand to retrieve data from database with the help of sql statement using various operators like arithmetic operators, group functions comparison operators etc.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Assignment: 08

**AIM:**

Write DDL/DML statements to create VIEWS on single and multiple tables from above DB.

PROBLEM STATEMENT /DEFINITION

Write DDL statements to create VIEWS on single and multiple tables from above DB. Do the following operation to demonstrate the use of view:

* + Update the base table
  + Insert new record in the base table.
  + Delete record in the base table.
  + DML on VIEW.

What are the restrictions applicable while creating or modifying views? Demonstrate using suitable queries.

OBJECTIVE

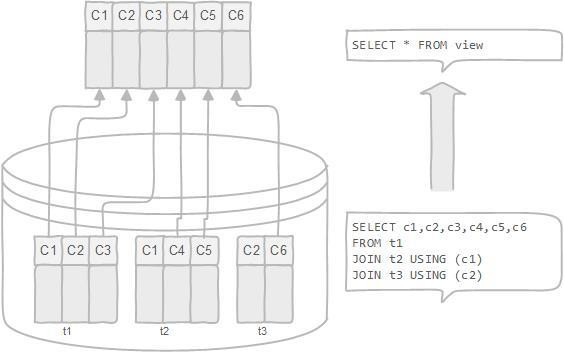
1. To understand use of VIEWS.
2. To understand implementation of different types of VIEWS.
3. To implement and analyze various operations on VIEWS.

THEORY:

**VIEW:**

Database view is known as a “virtual table” that allows you to query the data in it. Understanding the database views and using them correctly are very important. In this section, we will discuss the database views, how they are implemented in MySQL, and how to use them more effectively.

A database view is a virtual table or logical table which is defined as a [SQL SELECT query](http://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) with [joins](http://www.mysqltutorial.org/mysql-inner-join.aspx). Because a database view is similar to a database table, which consists of rows and columns, so you can query data against it. Most database management systems, including MySQL, allow you to [update data](http://www.mysqltutorial.org/mysql-update-data.aspx) in the underlying tables through the database view with some prerequisites.



A database view is dynamic because it is not related to the physical schema. The database system stores database views as a [SQL SELECT](http://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement with joins. When the data of the tables changes, the view reflects that changes as well.

Advantages of database view

The following are advantages of using database views.

* A database view allows you to simplify complex queries: a database view is defined by an SQL statement that associates with many underlying tables. You can use database view to hide the complexity of underlying tables to the end-users and external applications. Through a database view, you only have to use simple SQL statements instead of complex ones with many joins.
* A database view helps limit data access to specific users. You may not want a subset of sensitive data can be queryable by all users. You can use a database view to expose only non-sensitive data to a specific group of users.
* A database view provides extra security layer. Security is a vital part of any relational database management system. The database view provides extra security for a database management system. The database view allows you to create the read-only

view to expose read-only data to specific users. Users can only retrieve data in read- only view but cannot update it.

We can create View using CREATE VIEW statement. A View can be created from a single table or multiple tables.

**Syntax:**

**CREATE VIEW view\_name AS**

**SELECT column1, column2.....**

**FROM table\_name**

**WHERE condition;**

**view\_name: Name for the View**

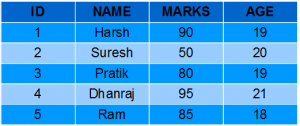
**table\_name: Name of the table**

**condition: Condition to select rows**

**Sample Tables**:



**Table- student\_details**



**Table- Student\_marks**

**Examples**:

* **Creating View from a single table:**

In this example we will create a View named DetailsView from the table StudentDetails.  
Query:

CREATE VIEW DetailsView AS

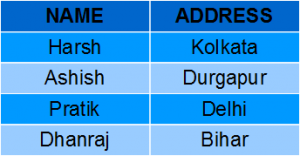
SELECT NAME, ADDRESS

FROM StudentDetails

WHERE S\_ID < 5;

To see the data in the View, we can query the view in the same manner as we query a table.

SELECT \* FROM DetailsView;

Output:  


Creating View from multiple tables: In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement. Query:

CREATE VIEW MarksView AS

SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS

FROM StudentDetails, StudentMarks

WHERE StudentDetails.NAME = StudentMarks.NAME;

To display data of View MarksView:  
SELECT \* FROM MarksView;

**Output:**



**DELETING VIEWS:**

We have learned about creating a View, but what if a created View is not needed any more? Obviously we will want to delete it. SQL allows us to delete an existing View. We can delete or drop a View using the DROP statement.

**Syntax**:

DROP VIEW view\_name;

view\_name: Name of the View which we want to delete.

For example, if we want to delete the View MarksView, we can do this as:

DROP VIEW MarksView;

**UPDATING VIEWS**

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is not met, then we will not be allowed to update the view.

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.
5. The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

* We can use the CREATE OR REPLACE VIEW statement to add or remove fields from a view.  
  **Syntax:**
* CREATE OR REPLACE VIEW view\_name AS
* SELECT column1,coulmn2,..
* FROM table\_name
* WHERE condition;

For example, if we want to update the view MarksView and add the field AGE to this View from StudentMarks Table, we can do this as:

CREATE OR REPLACE VIEW MarksView AS

SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS, StudentMarks.AGE

FROM StudentDetails, StudentMarks

WHERE StudentDetails.NAME = StudentMarks.NAME;

If we fetch all the data from MarksView now as:

SELECT \* FROM MarksView;

Output:



CONCLUSION:

The DML commands are implemented using views. Views are created from single table and multiple tables as well. Alongwith it updated the views.

Assignment: 9

**AIM:**

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

PROBLEM STATEMENT /DEFINITION

1. PL/SQL Assignmentsbased on tables created.
2. Write Simple PL/SQL programs to perform different operations on tables
3. Write trigger and execute it on a table.

OBJECTIVE

* To understand PL/SQL
* To understand the concept of **procedures /Functions**

THEORY:

1. **Creating a Stored Procedure:**

**Syntax to create Stored Procedure in mySql:**

DELIMITER //

CREATE PROCEDURE p2() LANGUAGE SQL DETERMINISTIC

SQL SECURITY DEFINER

COMMENT 'A procedure' BEGIN

SELECT 'Hello World !'; END//

The first part of the statement creates the procedure. The next clauses defines the optional characteristics of the procedure. Then you have the name and finally the body or routine code.

Stored procedure names are case insensitive, and you cannot create procedures with the same name. Inside a procedure body, you can't put database-manipulation statements.Modify a Stored Procedure

MySQL provides an ALTER PROCEDURE statement to modify a routine, but only allows for the ability to change certain characteristics. If you need to alter the body or the parameters, you must drop and recreate the procedure.

Delete a Stored Procedure

DROP PROCEDURE IF EXISTS p2;

CREATE PROCEDURE proc1 (IN varname DATA-TYPE) : One input parameter. The word IN is optional because parameters are IN (input) by default.

CREATE PROCEDURE proc1 (OUT varname DATA-TYPE) : One output parameter.

CREATE PROCEDURE proc1 (INOUT varname DATA-TYPE) : One parameter which is both input and output.

Of course, you can define multiple parameters defined with different types.

IN example

DELIMITER //

CREATE PROCEDURE proc\_IN (IN var1 INT) BEGIN

SELECT var1 + 2 AS result; END//

OUT example

DELIMITER //

CREATE PROCEDURE proc\_OUT (OUT var1 VARCHAR(100)) BEGIN

SET var1 = 'This is a test'; END //

INOUT example

DELIMITER //

CREATE PROCEDURE proc\_INOUT(INOUT var1 INT) BEGIN

SET var1 = var1 \* 2;

END //

Step 4 - Variables

The following step will teach you how to define variables, and store values inside a procedure. You must declare them explicitly at the start of the BEGIN/END block, along with their data types. Once you've declared a variable, you can use it anywhere that you could use a session variable, or literal, or column name.

Declare a variable using the following syntax:

DECLARE varname DATA-TYPE DEFAULT defaultvalue; Let's declare a few variables:

DECLARE a, b INT DEFAULT 5; DECLARE strVARCHAR(50);

DECLARE today TIMESTAMP DEFAULT CURRENT\_DATE; DECLARE v1, v2, v3 TINYINT;

Working with variables

Once the variables have been declared, you can assign them values using the SET or SELECT command:

DELIMITER //

CREATE PROCEDURE var\_proc(IN paramstr VARCHAR(20)) BEGIN

DECLARE a, b INT DEFAULT 5; DECLARE strVARCHAR(50);

DECLARE today TIMESTAMP DEFAULT CURRENT\_DATE; DECLARE v1, v2, v3 TINYINT;

INSERT INTO table1 VALUES (a);

SET str = 'I am a string';

SELECT CONCAT(str,paramstr), today FROM table2 WHERE b >=5; END //

Step 5 - Flow Control Structures

MySQL supports the IF, CASE, ITERATE, LEAVE LOOP, WHILE and REPEAT constructs for flow control within stored programs. We're going to review how to use IF, CASE and WHILE specifically, since they happen to be the most commonly used statements in routines.

* + IF statement

With the IF statement, we can handle tasks which involves conditions: DELIMITER //

CREATE PROCEDURE proc\_IF(IN param1 INT) BEGIN

DECLARE variable1 INT; SET variable1 = param1 + 1; IF variable1 = 0 THEN

SELECT variable1;

END IF;

IF param1 = 0 THEN

SELECT 'Parameter value = 0'; ELSE

SELECT 'Parameter value <> 0'; END IF;

END //

* + CASE statement

The CASE statement is another way to check conditions and take the appropriate path. It's an excellent way to replace multiple IF statements. The statement can be written in two different ways, providing great flexibility to handle multiple conditions.

DELIMITER //

CREATE PROCEDURE proc\_CASE(IN param1 INT) BEGIN

DECLARE variable1 INT;

SET variable1 = param1 + 1; CASE variable1

WHEN 0 THEN

INSERT INTO table1 VALUES (param1); WHEN 1 THEN

INSERT INTO table1 VALUES (variable1); ELSE

INSERT INTO table1 VALUES (99); END CASE;

END // or:

DELIMITER //

CREATE PROCEDURE proc\_CASE (IN param1 INT) BEGIN

DECLARE variable1 INT; SET variable1 = param1 + 1;

CASE

WHEN variable1 = 0 THEN

INSERT INTO table1 VALUES (param1); WHEN variable1 = 1 THEN

INSERT INTO table1 VALUES (variable1); ELSE

INSERT INTO table1 VALUES (99); END CASE;

END //

* + WHILE statement

There are technically three standard loops: WHILE loops, LOOP loops, and REPEAT loops. You also have the option of creating a loop using the “Darth Vader” of programming techniques: the GOTO statement. Check out this example of a loop in action:

DELIMITER //

CREATE PROCEDURE proc\_WHILE (IN param1 INT) BEGIN

DECLARE variable1, variable2 INT; SET variable1 = 0;

WHILE variable1 < param1 DO

INSERT INTO table1 VALUES (param1);

SELECT COUNT(\*) INTO variable2 FROM table1; SET variable1 = variable1 + 1;

END WHILE;

END //

**Conclusion:**

PL/SQL stored procedure is created and function is also created.

**Assignment: 10**

**Aim:**

Write and execute PL/SQL block to implement all types of triggers on above DB. (Consider row level and statement level triggers)

PROBLEM STATEMENT /DEFINITION

Write trigger and execute it on a table.

Write Simple PL/SQL programs to perform different operations on tables

OBJECTIVE:

To understand PL/SQL

To understand the concept of **triggers, pl /sql block**. **THEORY:**

1. TRIGGER ON UPDATE DISPLAY ERROR MESSAGE:

CREATE TRIGGER account\_bal BEFORE UPDATE

ON account

FOR EACH ROW BEGIN

IF (NEW.bal< 0) THEN SIGNAL SQLSTATE '80000'

SET MESSAGE\_TEXT='Account balance cannot be less than 0'; END IF;

END;

* 1. Create trigger on update which will create log file of update

CREATE TRIGGER account\_bal1 AFTER UPDATE ON account FOR EACH ROW

BEGIN

INSERT into transaction\_log(user\_id, description)

VALUES (user( ),CONCAT('Adjusted account ',NEW.accno,' from ',OLD.bal,' to ', NEW.bal)); end;

Sample PL/SQL Statement:

1. Write a PL/SQL block to find the sum of first 100 odd nos. and even nos)
2. Write a PL/SQL block to display the Information of given student on following table Stud (sno, sname, address, city).
3. Write a PL/SQL block for preparing a Net Salary, given employee on following table Emp (eno, ename, address, city)

Salary (eno, basic, da, hra, it)

Net\_Salary (eno, total\_allowance, total\_deduction, netpay) Notes : D.A. = 59% of basic , H.R.A. = 500, I.T. = 2% of basic

Total\_Allowance = Basic + D.A. + H.R.A., Total\_Deduction = I.T. Netpay = Total\_Allowance – Total\_Deduction.

1. Write a PL/SQL block to raise the salary by 20% of given employee on following table.

Emp\_Salary (eno, ename, city, salary)

1. Write an Implicit Cursor to accept the employee number from the user. You have to delete this record and display the appropriate message on the following table.

Emp (eno, ename, address, city)

1. Write a Cursor to display the employee number, name, department and salary of first employee getting the highest salary.

Emp (eno, ename, department, address, city) Salary (eno, salary)

1. Write a Cursor to display the first five records on the following table.

Student(sno, sname, address, city)

1. Write Cursor for preparing a Net Salary for employee‟s of finance department and Net Pay is more than 10,000 on following table.
2. Writes a Function to check whether the given number is prime or not

CONCLUSION:

Understand the concept of PL/SQL block by implementing all types of triggers on Database.

Assignment: 11

**AIM:**

Write and execute PL/SQL block to implement all types of cursor on above DB.

PROBLEM STATEMENT /DEFINITION

1. PL/SQL Assignments based on tables created.
2. Write Simple PL/SQL programs to perform different operations on tables
3. Write cursor and execute it on a table.

OBJECTIVE

* + To understand PL/SQL
  + To understand the concept of **cursor**

THEORY:

**Cursor:**

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc.

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

* + Implicit cursors
  + Explicit cursors Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed,

when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the**SQL cursor**, which always has attributes such as **%FOUND, %ISOPEN, %NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes, **%BULK\_ROWCOUNT** and**%BULK\_EXCEPTIONS**, designed for use with the **FORALL**statement. The following table provides the description of the most used attributes −

|  |  |
| --- | --- |
| **S.No** | **Attribute & Description** |
| 1 | **%FOUND**  Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE. |
| 2 | **%NOTFOUND**  The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE. |
| 3 | **%ISOPEN**  Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement. |
| 4 | **%ROWCOUNT**  Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement. |

Any SQL cursor attribute will be accessed as**sql%attribute\_name** as shown below in the example.

Example

We will be using the CUSTOMERS table we had created and used in the previous chapters.

Select \* from customers;

+ + + + + +

| ID | NAME | AGE | ADDRESS | SALARY |

+ + + + + +

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

+ + + + + +

The following program will update the table and increase the salary of each customer by 500 and use the **SQL%ROWCOUNT**attribute to determine the number of rows affected −

DECLARE

total\_rows number(2); BEGIN

UPDATE customers

SET salary = salary + 500;

IF sql%notfound THEN dbms\_output.put\_line('no customers selected');

ELSIF sql%found THEN total\_rows := sql%rowcount;

dbms\_output.put\_line( total\_rows || ' customers selected '); END IF;

END;

/

When the above code is executed at the SQL prompt, it produces the following result −

PL/SQL procedure successfully completed.

6 customers selected

If you check the records in customers table, you will find that the rows have been updated −

Select \* from customers;

+ + + + + +

| ID | NAME | AGE | ADDRESS | SALARY |

+ + + + + +

| 1 | Ramesh | 32 | Ahmedabad | 2500.00 |

| 2 | Khilan | 25 | Delhi | 2000.00 |

| 3 | kaushik | 23 | Kota | 2500.00 |

| 4 | Chaitali | 25 | Mumbai | 7000.00 |

| 5 | Hardik | 27 | Bhopal | 9000.00 |

| 6 | Komal | 22 | MP | 5000.00 |

+ + + + + +

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is −

CURSOR cursor\_name IS select\_statement;

Working with an explicit cursor includes the following steps −

* + Declaring the cursor for initializing the memory
  + Opening the cursor for allocating the memory
  + Fetching the cursor for retrieving the data
  + Closing the cursor to release the allocated memory Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For

example −

CURSOR c\_customers IS

SELECT id, name, address FROM customers;

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows −

OPEN c\_customers;

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows −

FETCH c\_customers INTO c\_id, c\_name, c\_addr;

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows −

CLOSE c\_customers;

Example

Following is a complete example to illustrate the concepts of explicit cursors &minua;

END;

/

DECLARE

c\_id customers.id%type;

c\_name customerS.No.ame%type; c\_addr customers.address%type; CURSOR c\_customers is

SELECT id, name, address FROM customers; BEGIN

OPEN c\_customers; LOOP

FETCH c\_customers into c\_id, c\_name, c\_addr; EXIT WHEN c\_customers%notfound;

dbms\_output.put\_line(c\_id || ' ' || c\_name || ' ' || c\_addr); END LOOP;

CLOSE c\_customers;

When the above code is executed at the SQL prompt, it produces the following result −

1. Ramesh Ahmedabad
2. Khilan Delhi
3. kaushik Kota
4. Chaitali Mumbai
5. Hardik Bhopal
6. Komal MP

PL/SQL procedure successfully completed.

CONCLUSION:

The concept of PL/SQL block by implementing all types cursor on DB.

**References**

**Annexure:**

1. GUIs / Screen Snapshot of the System Developed
2. Part –A: SQL and PL/SQL, Triggers, Stored Procedures, Functions