

214456: Database Management System Lab

SEIT (2019 Course)

Semester - II

| Teaching Scheme | | Examination Scheme | |
|-----------------|---------------|--------------------|----------|
| Practical : | 4 Hrs. / Week | Practical | 25 Marks |
| | | Term Work | 25 Marks |



LABORATORY MANUAL V 1.0

DEPARTMENT OF INFORMATION TECHNOLOGY

Sinhgad Institute of Technology and Science, Narhe, Pune

2021-2022

VISION

To get recognized for quality education and research in the field of Information Technology.

MISSION

1. To equip the faculty and students with proper skills of theoretical and practical knowledge.
2. To bridge the gap between the industry and academia by providing industry sponsored projects, and various guest lectures.
3. To inculcate technical and team work skills in students so as to enable them to take up new challenges and real life problems faced by industry and society.

PROGRAM EDUCATIONAL OBJECTIVES

The students of Information Technology course after passing out will

1. To prepare student's strong fundamental concepts in mathematics, science, engineering and technology to address technological challenges.
2. To make every effort for intellectual upbringing of students in strong core knowledge and skills in the field of Information Technology.
3. To inspire students for research, entrepreneurship and higher studies in the field of computer science and Information Technology.
4. To introduce communication, presentation, time management and team work, skills in students.
5. To inculcate technical and multi-disciplinary skills in students so as to enable them take up new challenges and real life problems faced by industry and society

PROGRAM OUTCOMES

The students in the Information Technology course will attain:

- a. an ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, and engineering and technology;
- b. an ability to define a problem and provide a systematic solution with the help of conducting experiments, as well as analyzing and interpreting the data;
- c. an ability to design, implement, and evaluate a software or a software/hardware system, component, or process to meet desired needs within realistic constraints;
- d. an ability to identify, formulate, and provide systematic solutions to complex engineering problems;
- e. an ability to use the techniques, skills, and modern engineering technologies tools, standard processes necessary for practice as a IT professional;
- f. an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems with necessary constraints and assumptions;
- g. an ability to analyze the local and global impact of computing on individuals, organizations and society;
- h. an ability to understand professional, ethical, legal, security and social issues and responsibilities;
- i. an ability to function effectively as an individual or as a team member to accomplish a desired goal(s);
- j. an ability to engage in life-long learning and continuing professional development to cope up with fast changes in the technologies/tools with the help of electives, professional organizations and extra-curricular activities;
- k. an ability to communicate effectively in engineering community at large by means of effective presentations, report writing, paper publications, demonstrations;
- l. an ability to understand engineering, management, financial aspects, performance, optimizations and time complexity necessary for professional practice;
- m. an ability to apply design and development principles in the construction of software systems of varying complexity.

Compliance Document Control

| | |
|-------------------------|---------------------------------|
| Reference Code | SITS-IT / Lab Manual Procedures |
| Version No | 1.0 |
| Compliance Status | Complete |
| Date of Compliance | 01-01-2022 |
| Security Classification | Department Specific |
| Document Status | Definitive |
| Review Period | Yearly |

Author

Mr. T D Khadtare

Professor, SITS(IT)

Document History

| Revision No. | Revision Date | Reason For Change |
|--------------|---------------|-------------------|
| | | |

Summary of Changes to Database Management System Lab

| Sr. No | Changes | Change type |
|--------|---------|-------------|
| 01 | | |
| 02 | | |

Syllabus

| | | |
|---|-----------------------|--|
| Savitribai Phule Pune University, Pune Second Year Information Technology (2019 Course) 214456: Database Management System Lab | | |
| Teaching Scheme: | Credit Scheme: | Examination Scheme: |
| Practical (PR):04hrs/week | 02 | PR: 25 Marks TW: 25 Marks |
| Prerequisites: Data structures and Software engineering principles and practices. | | |
| Objectives: <ol style="list-style-type: none"> 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. | | |
| Course Outcomes: On completion of this course student will be able to -- CO1: Install and configure database systems. CO2: Analyze database models & entity relationship models. CO3: Design and implement a database schema for a given problem-domain CO4: Implement relational database systems. CO5: Populate and query a database using SQL DDL / DML / DCL commands. CO6 : Design a backend database of any one organization: CASE STUDY | | |
| Guidelines for Instructor's Manual | | |
| The faculty member should prepare the laboratory manual for all the experiments and it should be made available to students and laboratory instructor/Assistant. | | |

Guidelines for Student's Lab Journal

1. Student should submit term work in the form of journal with write-ups based on specified list of assignments.
2. Practical and Oral Examination will be based on all the assignments in the lab manual
3. Candidate is expected to know the theory involved in the experiment.
4. The practical examination should be conducted only if the journal of the candidate is complete in all respects.

Guidelines for Oral /Practical Assessment

1. Examiners will assess the student based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of implemented assignment, attendance etc.
2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. Appropriate knowledge of usage of software and hardware related to respective laboratory should be checked by the concerned faculty member.

Suggested List of Laboratory Assignments

Group A: Study of Databases

Mapping of Course Outcomes Group A -- CO1

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

Mapping of Course Outcomes Group B -- CO2, CO3, CO4, CO5

1. Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system.
2. Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them
3. Create Table with primary key and foreign key constraints.
 - a. Alter table with add n modify b. Drop table
4. 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
5. 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

Mapping of Course Outcomes Group C -- CO6

1. Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.
2. Write and execute suitable database triggers. Consider row level and statement level triggers.
3. Write a PL/SQL block to implement all types of cursor.

Group D: Relational Database Design

Mapping of Course Outcomes Group D -- CO5, CO6

Design and case study of any organization (back end only), Project Proposal and High Level SRS To prepare for project, do the following:

1. Form teams of around 3 to 4 people
2. Create requirements document with the following information: -
 - a. Give one or two paragraph description of your goals for the topic(s).
 - b. List what all types of users will be accessing your application
 - c. List the various functionalities that your application will support. Explain each in about a paragraph worth of detail.
 - d. List the hardware and software requirements at the backend and at the front end.

- e. Give an estimate of the number of users of each type, the expected load (transactions per day), and the expected database size.

Project ER Diagram and Database Design

For ER diagram and Database design following guidelines can be used:

1. Draw an ER diagram of your project.
2. Reduce this ER diagram into the tables and complete database design.
3. Subsequently, list all the functional dependencies on each table that you expect will hold.
4. Check that the database schema is in 3NF/BCNF. If it is not, apply normalization. Use non-loss decomposition and bring the database schema in 3NF/BCNF.

Give the ER diagram and the data dictionary as part of the requirement specifications file which you created for the project proposal.

Reference Books:

1. Dr. P. S. Deshpande, "SQL and PL/SQL for Oracle 10g Black Book", DreamTech
2. Ivan Bayross, "SQL, PL/SQL: The Programming Language of Oracle", BPB Publication
3. Reese G., Yarger R., King T., Williams H, "Managing and Using MySQL", Shroff Publishers and Distributors Pvt. Ltd., ISBN: 81 - 7366 - 465 – X, 2nd Edition
4. Eric Redmond, Jim Wilson, "Seven databases in seven weeks", SPD, ISBN: 978-93-5023-91
5. Jay Kreibich, Using SQLite, SPD, ISBN: 978-93-5110-934-1, 1st edition

INDEX

| Sr.No | Title | Page No |
|------------------------------------|--|---------|
| 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 | | |
| 1 | Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system. | |
| 2 | Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them | |
| 3 | Create Table with primary key and foreign key constraints. a. Alter table with add n modify b. Drop table | |
| 4 | Perform following SQL queries on the database created in assignment 1. <ul style="list-style-type: none"> • 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 | |
| 5 | 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 | | |
| 1 | Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use. | |
| 2 | Write and execute suitable database triggers. Consider row level and statement level triggers. | |
| 3 | Write a PL/SQL block to implement all types of cursor | |
| | | |

Group D: Relational Database Design

| | | |
|---|--|--|
| 1 | <p>Design and case study of any organization (back end only), Project Proposal and High Level SRS to prepare for project, do the following:</p> <ol style="list-style-type: none">1. Form teams of around 3 to 4 people2. Create requirements document with the following information: -<ol style="list-style-type: none">a. Give one or two paragraph description of your goals for the topic(s).b. List what all types of users will be accessing your applicationc. List the various functionalities that your application will support. Explain each in about a paragraph worth of detail.d. List the hardware and software requirements at the backend and at the front end.e. Give an estimate of the number of users of each type, the expected load (transactions per day), and the expected database size. | |
|---|--|--|

SCHEDULE

| Sr.No | Title | No of Hrs. | Week |
|------------------------------------|--|------------|------|
| 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 | | | |
| 1 | Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system. | | |
| 2 | Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them. | | |
| 3 | Create Table with primary key and foreign key constraints. a. Alter table with add n modify b. Drop table | | |
| 4 | Perform following SQL queries on the database created in assignment 1. <ul style="list-style-type: none"> • 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 | | |
| 5 | 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 | | | |
| 1 | Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use. | | |
| 2 | Write and execute suitable database triggers. Consider row level and statement level triggers. | | |
| 3 | Write a PL/SQL block to implement all types of cursor | | |
| | | | |

| | | |
|--|--|--|
| Group D: Relational Database Design | | |
| 1 | <p>Design and case study of any organization (back end only), Project Proposal and High Level SRS to prepare for project, do the following:</p> <ul style="list-style-type: none">3. Form teams of around 3 to 4 people4. Create requirements document with the following information: -<ul style="list-style-type: none">a. Give one or two paragraph description of your goals for the topic(s).b. List what all types of users will be accessing your applicationc. List the various functionalities that your application will support. Explain each in about a paragraph worth of detail.d. List the hardware and software requirements at the backend and at the front end.e. Give an estimate of the number of users of each type, the expected load (transactions per day), and the expected database size. | |

Group A: Study of Databases

Assignment No.: 1

Date:

Title: -

Study of MySQL Open source software. Discuss the characteristics like efficiency, scalability, performance and transactional properties

Remarks:

Aim: Study of MySQL Open source software. Discuss the characteristics like efficiency, scalability, performance and transactional properties

Objective: To be aware MYSQL Open Source Databases.

Theory:

MySQL

MySQL "My S-Q-L", officially, but also called "My Sequel" is the world's most widely used open-source relational database management system (RDBMS). It is named after co-founder Michael Widenius's daughter, My. The SQL phrase stands for Structured Query Language. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

MySQL is also used in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr, and YouTube. MySQL is the most popular and widely used relational database management system that provides multi-user access to number of databases. MySQL is now owned by Oracle and uses Sequential Query Language to manage database. Its source is available under GNU license and propriety agreements. MySQL is most popular among PHP developers and used for websites, web applications and online services.

Features of MySQL:

- Because of its unique storage engine architecture MySQL performance is very high.
- Supports large number of embedded applications which makes MySQL very flexible.
- Use of Triggers, Stored procedures and views which allows the developer to give a higher productivity.
- Allows transactions to be rolled back, commit and crash recovery

- Embedded database library
- Full-text indexing and searching
- Updatable views
- Cursors
- Triggers
- Cross-platform support

Limitation of MySQL:

- Like other SQL databases, MySQL does not currently comply with the full SQL standard for some of the implemented functionality, including foreign key references when using some storage engines other than the default of InnoDB
- No triggers can be defined on views.
- MySQL, like most other transactional relational databases, is strongly limited by hard disk performance. This is especially true in terms of write latency.

System Properties Comparison CouchDB vs. MongoDB vs. MySQL

| Name | CouchDB | MongoDB | MySQL |
|-------------------------|--|---|-------------------------------|
| Description | A document store inspired by Lotus Notes | One of the most popular document stores | Widely used open source RDBMS |
| Developer | Apache Software Foundation | MongoDB, Inc | Oracle |
| Initial release | 2005 | 2009 | 1995 |
| License | Open Source | Open Source | Open Source |
| Implementation language | Erlang | C++ | C and C++ |
| Database model | Document store | Document store | Relational DBMS |

| | | | |
|-----------------------|-------------|-------------|--------|
| Data scheme | schema-free | schema-free | yes |
| SQL | No | No | yes |
| Supported | C | C | C |
| programming languages | C# | C# | C# |
| | Java | Java | Java |
| | JavaScript | JavaScript | PHP |
| | Lisp | Lisp | Python |
| Triggers | Yes | No | Yes |
| Foreign keys | No | no | yes |
| Map Reduce | Yes | yes | no |

Conclusion: Awareness of MySQL Open Source Databases.

FAQs: -

1. What is Open Source Software?
2. What is Open Source Database?
3. What are different open source databases?
4. What is difference between MySQL and MongoDB?
5. Foreign key is possible in MongoDB or Not?
6. List more open-source database systems other than MySQL and MongoDB?

Assignment No. 2

Date:

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

Remarks:

Aim: Install and configure client and server for MySQL and MongoDB

Objective: Study all commands and necessary steps for installation and configuration.

Theory:

Setting Up the MySQL Database Server in the Windows Operating System

- Starting the Download
- Starting the Installation

Starting the Download

1. Go to <http://dev.mysql.com/downloads/installer/>.
2. Click the Download button.
3. Save the installer file to your system.

Starting the Installation

After the download completes, run the installer as follows:

1. Right-click the downloaded installation file (for example, `mysql-installer-community-5.6.14.0.msi`) and click Run.
The MySQL Installer starts.
2. On the Welcome panel, select Install MySQL Products.
3. On the License Information panel, review the license agreement, click the acceptance checkbox, and click Next.
4. On the Find latest products panel, click Execute.
When the operation is complete, click Next.
5. On the Setup Type panel, choose the Custom option and click Next.
6. On the Feature Selection panel, ensure MySQL Server 5.6.x is selected, and click Next.
7. On the Check Requirements panel, click Next.
8. On the Installation panel, click Execute.
When the server installation is completed successfully, the information message appears on the Installation panel. Click Next.
9. On the Configuration panel, click Next.

10. At the first MySQL Server Configuration page (1/3), set the following options:

- **Server Configuration Type.** Select the Development Machine option.
- **Enable TCP/IP Networking.** Ensure the checkbox is selected and specify the options below:
- **Port Number.** Specify the connection port. The default setting is 3306 - leave it unchanged if there is not special reason to change it.
- **Open Firewall port for network access.** Select to add firewall exception for the specified port.
- **Advanced Configuration.** Select the Show Advanced Options checkbox to display an additional configuration page for setting advanced options for the server instance if required.

Note: Choosing this option is necessary to get to the panel for setting the network options where you will turn off the firewall for the port used by the MySQL server.

11. Click Next.

12. At the second MySQL Server Configuration page (2/3), set the following options:

- **Root Account Password.**
- **MySQL Root Password.** Enter the root user's password.
- **Repeat Password.** Retype the root user's password.

Note: The root user is a user who has full access to the MySQL database server - creating, updating, and removing users, and so on. Remember the root password - you will need it later when creating a sample database.

- **MySQL User Accounts.** Click Add User to create a user account. In the MySQL User Details dialog box, enter a user name, a database role, and a password (for example, !phpuser). Click OK.

Click Next.

13. At the third MySQL Server Configuration page (3/3), set the following options:

- **Windows Service Name.** Specify a Windows Service Name to be used for the MySQL server instance.
- **Start the MySQL Server at System Startup.** Leave the checkbox selected if the MySQL server is required to automatically start at system startup time.
- **Run Windows Service as.** Choose either:
- **Standard System Account.** Recommended for most scenarios.
- **Custom User.** An existing user account recommended for advanced scenarios.

Click Next.

14. At the Configuration Overview page, click Next.

15. When the configuration is completed successfully, the information message appears on the Complete panel. Click Finish.

Note: To check that the installation has completed successfully, run the Task Manager. If the MySQLd-nt.exe is on the Processes list - the database server is running.

Conclusion: Studied the installation process of MYSQL.

Assignment No. 3

Date:

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

Remarks:

Aim: Study the SQLite database and its uses. Also elaborate on building and installing of SQLite

Objective: To study SQLite and perform the installation steps

Theory:

What is SQLite

- A very powerful, embedded relational database management system.
- SQLite is an amazing library that gets embedded inside the application that makes use of.
- SQLite Data Base is used in mobile devices (Android, iPhone) and it is light, takes only Kb space
- Sqlite is very light version of SQL supporting many features of SQL. Basically is been developed for small devices like mobile phones, tablets etc
- As a self-contained, file-based database.
- SQL is query language. Sqlite is embeddable relational database management system.
- Sqlite also doesn't require a special database server or anything. It's just a direct filesystem engine that uses SQL syntax
- SQLite offers an amazing set of tools to handle all sorts of data with much less constraint and ease compared to hosted, process based (server) relational databases.
- When an application uses SQLite, the integration works with functional and direct calls made to a file holding the data (i.e. SQLite database) instead of communicating through an interface of sorts (i.e. ports, sockets).
- This makes SQLite extremely fast and efficient, and also powerful .

SQLite's Supported Data Types

- **NULL:**
- NULL value.
- **INTEGER:**
- Signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
- **REAL:**
- Floating point value, stored as an 8-byte IEEE floating point number.
- **TEXT:**
- Text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).

Advantages of SQLite

- **File based:**
- The entire database consists of a single file on the disk, which makes it extremely portable.

- **Great for developing and even testing:**
- During the development phase of most applications, for a majority of people it is extremely likely to need a solution that can scale for concurrency.
- SQLite, with its rich feature base, can offer more than what is needed for development with the simplicity of working with a single file and a linked C based library.

Disadvantages of SQLite

- **No user management:**

Advanced databases come with the support for users, i.e. managed connections with set access privileges to the database and tables.

Given the purpose and nature of SQLite (no higher-levels of multi-client concurrency), this feature does not exist.

- **Lack of possibility to tinker with for additional performance:**
- Again by design, SQLite is not possible to tinker with to obtain a great deal of additional performance.
- The library is simple to tune and simple to use. Since it is not complicated, it is technically not possible to make it more performant than it already, amazingly is.

When To Use SQLite

- **Embedded applications:**
- All applications that need portability, that do not require expansion, e.g. single-user local applications, mobile applications or games.
- **Disk access replacement:**
- In many cases, applications that need to read/write files to disk directly can benefit from switching to SQLite for additional functionality and simplicity that comes from using the *Structured Query Language*(SQL).
- **Testing:**
- It is an overkill for a large portion of applications to use an additional process for testing the business-logic (i.e. the application's main purpose: functionality).

When Not To Use SQLite

- **Multi-user applications:**
- If you are working on an application whereby multiple clients need to access and use the same database, a fully-featured RDBM (e.g. MySQL) is probably better to choose over SQLite.
- **Applications requiring high write volumes:**

One of the limitations of SQLite is the *write* operations. This DBMS allows only one single write*operating to take place at any given time, hence allowing a limited throughput.


Installation Steps of SQLite

- SQLite is famous for its great feature zero-configuration, which means no complex setup or administration is needed.
- Install SQLite on Windows
- First, go to the <https://www.sqlite.org> website.
- Second, open the download page <https://www.sqlite.org/download.html>
- **Step 1** – Go to [SQLite download page](#), and download precompiled binaries from Windows section.
- **Step 2** – Download `sqlite-shell-win32-*.zip` and `sqlite-dll-win32-*.zip` zipped files.(Tool kit)
- **Step 3** – Create a folder `C:\>sqlite` and unzip above two zipped files in this folder, which will give you `sqlite3.def`, `sqlite3.dll` and `sqlite3.exe` files.

SQLite provides various versions for various platforms e.g., Windows, Linux, Mac, etc. You should choose an appropriate version to download.

For example, to work with SQLite on Windows, you download the command-line shell program as shown in the screenshot below

Precompiled Binaries for Windows



| | |
|---|---|
| sqlite-shell-win32-x86-3090200.zip (364.10 KiB) | The command-line shell program (version 3.9.2). (sha1: 25d78bbba37d2a0d9b9f86ed897e454ccc94d7b2) |
| sqlite-dll-win32-x86-3090200.zip (398.55 KiB) | 32-bit DLL (x86) for SQLite version 3.9.2. (sha1: f295747951897ecdea59a687e7722ec513b8a05) |
| sqlite-dll-win64-x64-3090200.zip (678.80 KiB) | 64-bit DLL (x64) for SQLite version 3.9.2. (sha1: f1f0dc69f14bea302dee93a7d49e67ff3bb379e1) |
| sqlite-analyzer-win32-x86-3090200.zip (695.47 KiB) | A program to analyze how space is allocated inside an SQLite (sha1: 8f86b63c9e23a70994fe2f9432979c8cb066cf34) |

- To verify the installation, you perform the following steps:
- First, open the command line window and navigate to the `C:\sqlite` folder.

```
Command Prompt

C:\>cd c:\sqlite
c:\sqlite>
```

- Second, enter sqlite3, you should see the following window:

```
Command Prompt - sqlite3

C:\>cd c:\sqlite
c:\sqlite>sqlite3
SQLite version 3.9.2 2015-11-02 18:31:45
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

- Third, you can type the .help command from the sqlite> prompt to see all available commands in sqlite3

```
Command Prompt - sqlite3

c:\sqlite>sqlite3
SQLite version 3.9.2 2015-11-02 18:31:45
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .help
.backup /DB? FILE      Backup DB (default "main") to FILE
.bail on|off           Stop after hitting an error. Default OFF
.binary on|off         Turn binary output on or off. Default OFF
.clone NEWDB            Clone data into NEWDB from the existing database
.databases              List names and files of attached databases
.dbinfo ?DB?           Show status information about the database
.dump ?TABLE? ...      Dump the database in an SQL text format
                        If TABLE specified, only dump tables matching
                        LIKE pattern TABLE.
.echo on|off           Turn command echo on or off
.epp on|off            Enable or disable automatic EXPLAIN QUERY PLAN
.exit                  Exit this program
.explain ?on|off?       Turn output mode suitable for EXPLAIN on or off.
                        With no args, it turns EXPLAIN on.
.fullschema            Show schema and the content of sqlite_stat tables
.headers on|off        Turn display of headers on or off
.help                  Show this message
.import FILE TABLE     Import data from FILE into TABLE
```

- Fourth, to quit the sqlite>, you use .quit command as follows:

```
.tables ?TABLE?      List names of tables
                      If TABLE specified, only list tables matching
                      LIKE pattern TABLE.
.timeout MS          Try opening locked tables for MS milliseconds
.timer on|off        Turn SQL timer on or off
.trace FILE|off      Output each SQL statement as it is run
.ofsname ?AUX?       Print the name of the UPS stack
.width NUM1 NUM2 ... Set column widths for "column" mode
                      Negative values right-justify
sqlite> .quit
c:\sqlite>
```

Conclusion: Studied SQLite database and the installation process

FAQs: -

1. What is SQLite?
2. What is Difference between SQL and SQLite?
3. What is Lite in SQLite?
4. What are four difference features of SQLite?
5. What are distinct features of SQLite?

Group B: MySQL

Assignment No. 1

Date:

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

Remarks:

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

Objective: To study the component of ER and EER Diagrams.

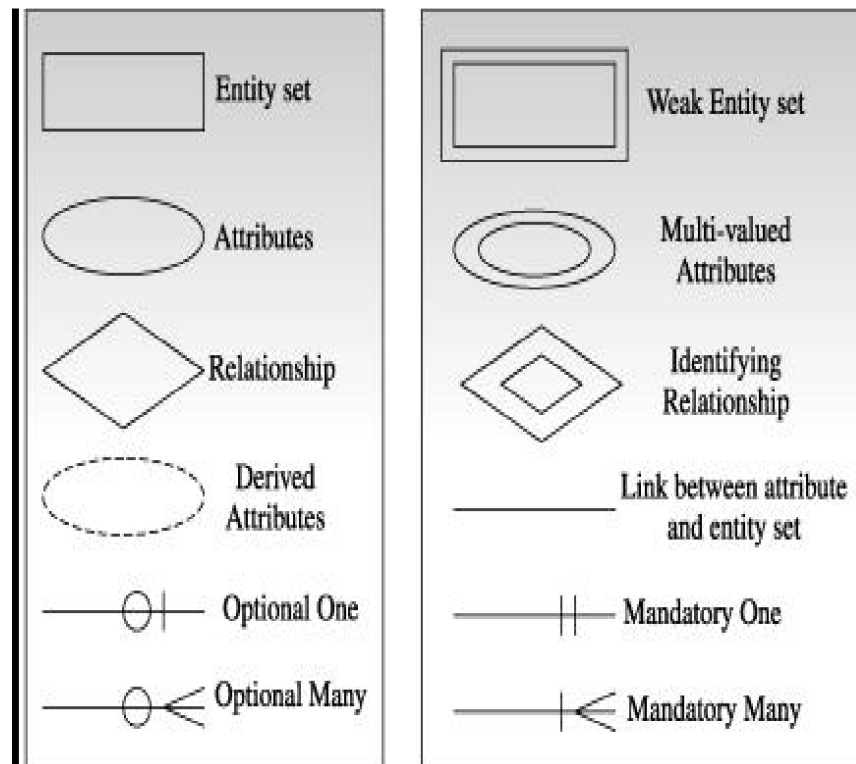
Theory:

Entity Relationship (ER) and Extended Entity Relationship (EER) Diagram

Entity Relationship (ER) Diagram

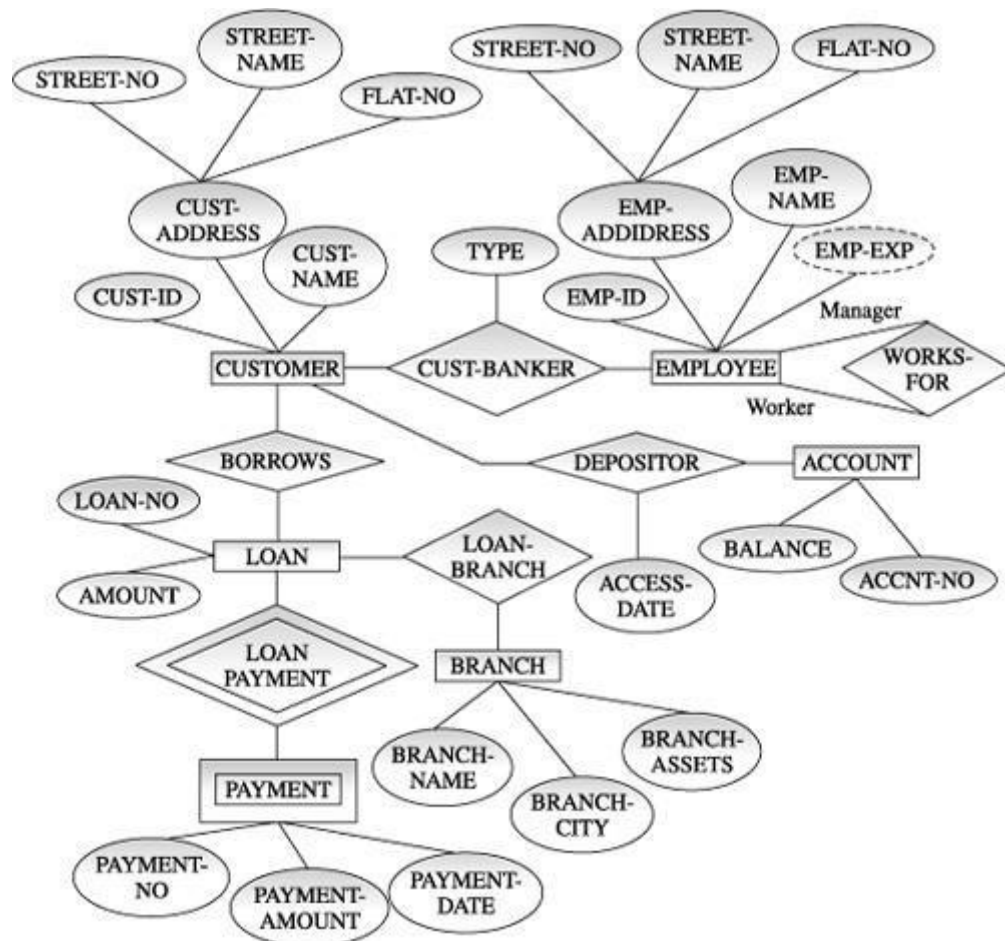
An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases. ER-Diagram is a visual representation of data that describes how data is related to each other.

Basic Building Blocks of ER Diagram



Sample ER Diagram

Complete E-R diagram of banking organization database

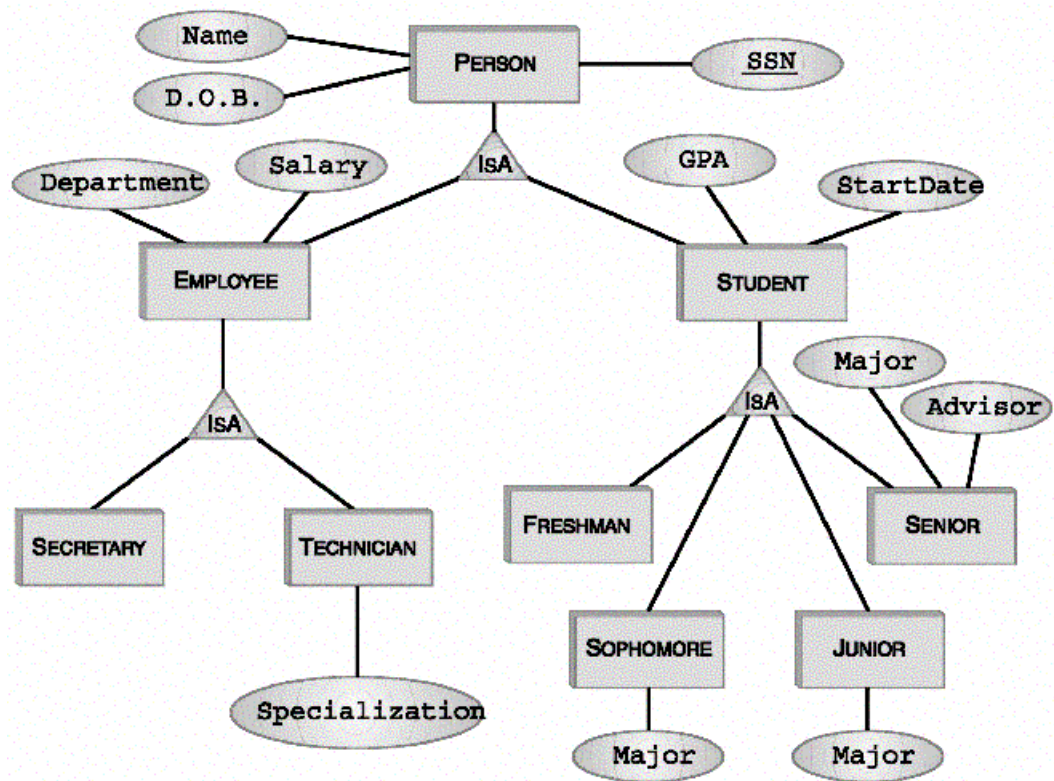


Extended Entity Relationship

The **enhanced entity–relationship (EER)** model (or **extended entity–relationship** model) in computer science is a high-level or conceptual data model incorporating extensions to the original entity–relationship (ER) model, used in the design of databases.

The Extended Entity-Relationship Model is a more complex and high-level model that extends an E-R diagram to include more types of abstraction, and to more clearly express constraints. All of the concepts contained within an E-R diagram are included in the EE-R model, along with additional concepts that cover more semantic information. These additional concepts include generalization/specialization, union, inheritance, and subclass/super class.

Sample EER Diagram



Conclusion: Understand the component of ER and EER Diagram. Draw the ER/EER diagram for the system.

FAQs:-

1. What ER and EER Diagram?
2. What are the components of ER/EER Diagram?
3. Difference between ER and EER Diagram?
4. How to represent generalization in EER Diagram?
5. What is ISA relationship?
6. What is Entity?

Assignment No. 2

Date:

Title: Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them.

Remarks:

Aim: Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them.

Objective: To understand the DDL statements and concept of Normalization.

Theory:

1 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

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.

- **Examples of DDL commands:**
- [CREATE](#) – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- [DROP](#) – is used to delete objects from the database.
- [ALTER](#) – is used to alter the structure of the database.
- [TRUNCATE](#) – is used to remove all records from a table, including all spaces allocated for the records are removed.
- [RENAME](#) – is used to rename an object existing in the database.

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:

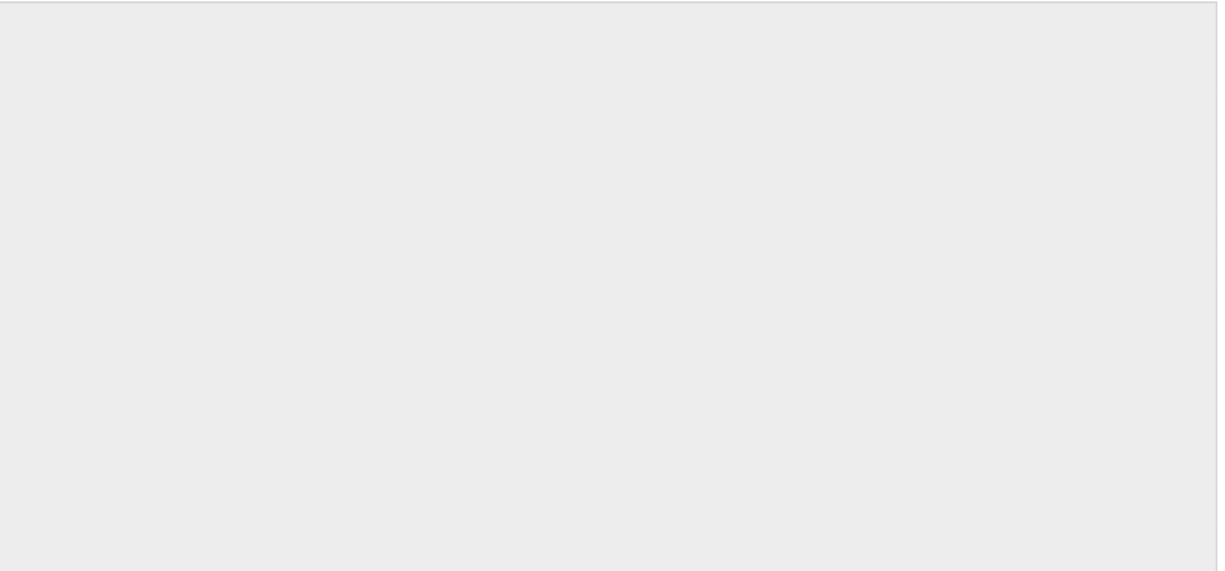
Syntax

```
CREATE TABLE tablename
[(columnname,columnname)]
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 to make changes to the structure of a table without deleting and recreating it.

Let's begin with creation of a table called **testalter_tbl**.



Dropping, Adding or Repositioning a Column:

Suppose you want to drop an existing column **i** from above MySQL table then you will use **DROP** clause along with **ALTER** command as follows:

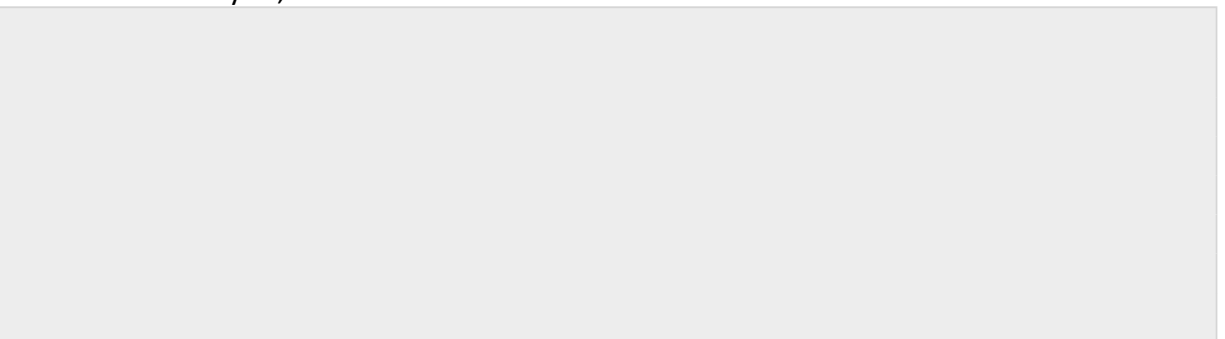
```
mysql> ALTER TABLE testalter_tbl DROP i;
```

A **DROP** will not work if the column is the only one left in the table.

To add a column, use ADD and specify the column definition. The following statement restores the **i** column to testalter_tbl:

```
mysql> ALTER TABLE testalter_tbl ADD i INT;
```

After issuing this statement, testalter will contain the same two columns that it had when you first created the table, but will not have quite the same structure. That's because new columns are added to the end of the table by default. So even though **i** originally was the first column in mytbl, now it is the last one.



To indicate that you want a column at a specific position within the table, either use **FIRST** to make it the first column or **AFTER col_name** to indicate that the new column should be placed after col_name. Try the following **ALTER TABLE** statements, using **SHOW COLUMNS** after each one to see what effect each one has:

```
ALTER TABLE testalter_tbl DROP i;  
ALTER TABLE testalter_tbl ADD i INT FIRST;  
ALTER TABLE testalter_tbl DROP i;  
ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

The **FIRST** and **AFTER** specifiers work only with the **ADD** clause. This means that if you want to reposition an existing column within a table, you first must **DROP** it and then **ADD** it at the new position.

Changing a Column Definition or Name:

To change a column's definition, use **MODIFY** or **CHANGE** clause along with **ALTER** command. For example, to change column **c** from **CHAR(1)** to **CHAR(10)**, do this:

```
mysql> ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```

With **CHANGE**, the syntax is a bit different. After the **CHANGE** keyword, you name the column you want to change, then specify the new definition, which includes the new name. Try out the following example:

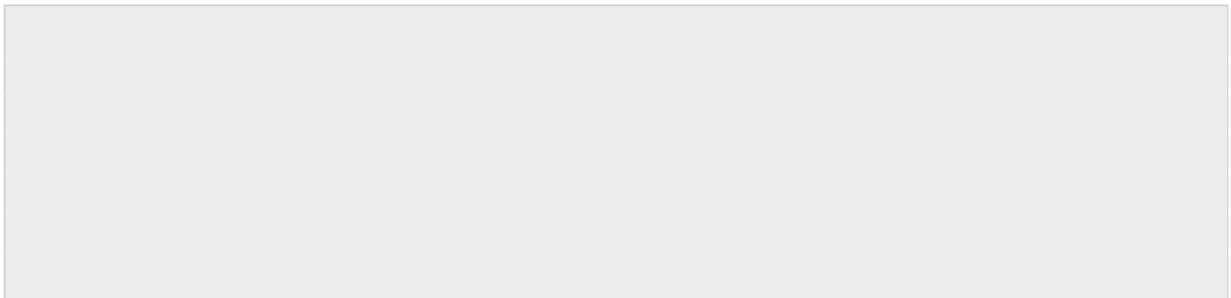
```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

If you now use **CHANGE** to convert **j** from **BIGINT** back to **INT** without changing the column name, the statement will be as expected:

```
mysql> ALTER TABLE testalter_tbl CHANGE j j INT;
```

Changing a Column's Default Value:

You can change a default value for any column using **ALTER** command. Try out the following example.



```
+ .....+- .....-+ .....+ .....+ .....+ .....-+  
2 rows in set (0.00 sec)
```

Renaming a Table:

To rename a table, use the **RENAME** option of the ALTER TABLE statement. Try out the following example to rename testalter_tbl to alter_tbl.

```
mysql> ALTER TABLE testalter_tbl RENAME TO alter_tbl;
```

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;
```

TRUNCATE Statement

- TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse).
- The result of this operation quickly removes all data from a table, typically bypassing a number of integrity enforcing mechanisms.
- The TRUNCATE TABLE mytable statement is logically (though not physically) equivalent to the DELETE FROM mytable statement (without a WHERE clause).
- Syntax:
 - TRUNCATE TABLE table_name;
 - table_name: Name of the table to be truncated.

Difference between DELETE, DROP and TRUNCATE

- DELETE :
Basically, it is a Data Manipulation Language Command (DML).
- It is use to delete the one or more tuples of a table.
- With the help of “DELETE” command we can either delete all the rows in one go or can delete row one by one. i.e., we can use it as per the requirement or the condition using Where clause.
- It is comparatively slower than TRUNCATE cmd.
- SYNTAX
If we want to delete all the rows of the table:
 - DELETE from table name;
- SYNTAX –
If we want to delete the row of the table as per the condition then we use WHERE clause,
 - DELETE from WHERE;

Note –

Here we can use the “ROLLBACK” command to restore the tuple.
Once COMMIT is executed, then ROLLBACK will not cause anything

SQL | ALTER (RENAME)

Sometimes we may want to rename our table to give it a more relevant name.
For this purpose, we can use **ALTER TABLE** to rename the name of table.

**Syntax may vary in different databases.*

Syntax(Oracle,MySQL,MariaDB):

ALTER TABLE table_name RENAME TO new_table_name;

- Columns can be also be given new name with the use of **ALTER TABLE**.
-

Syntax(MySQL,MariaDB):

ALTER TABLE table_name CHANGE COLUMN old_name TO new_name;

Conclusion: Understand the DDL commands.

FAQs: -

1. What is DDL
2. What is use of Alter command?
3. Alter command is DDL or DML?
4. What is Truncate Command?
5. What is difference between Drop and Delete?

Assignment No. 3

Date:

Title: Create Table with primary key and foreign key constraints.

a. Alter table with add n modify b. Drop table

Remarks:

Aim: Create Table with primary key and foreign key constraints. a. Alter table with add n modify b. Drop table

Objective: To understand the constraint and use of Alter, 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 UNIQUE | KEY | constraint constraint |
| Domain | FOREIGN CHECK NOT NULL | KEY | constraint constraint |
| Referential | FOREIGN CHECK constraint | KEY | 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.

a. 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.

b. NOT NULL CONSTRAINT:

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

c. UNIQUE CONSTRAINT:

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

d. 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.

e. FOREIGN KEY CONSTRAINT:

Foreign keys constrain data based on columns in other tables. They are called *foreign keys* because the constraints are foreign--that is, outside the table. For example, suppose a table contains customer addresses, and part of each address is a United States two-character

state code. If a table held all valid state codes, a foreign key constraint could be created to prevent a user from entering invalid state codes.

To create a table with different types of constraints:

Syntax

```
CREATE TABLE table_name  
(  
  column_name1 data_type [constraint],  
  column_name2 data_type [constraint],  
  .....  
)
```

Example

```
Create table customer  
( customer-name char(20) not null,  
  customer-street char(30),  
  customer-city char(30),  
  primary key ( customer-name));
```

```
create table branch  
( branch-name char(15) not null,  
  branch-city char(30),  
  assets number,  
  primary key ( branch-name));
```

```
create table account  
( branch-name char(15),  
  account-number char(10) not null,  
  balance number,  
  primary key ( account-number),  
  foreign key ( branch-name) references branch,  
  check (balance>500));
```

```
create table depositor  
( customer-name char(20) not null,  
  account-number char(10) not null,  
  primary key ( customer-name,account-number),  
  foreign key ( account-number) references account,
```

foreign key (customer-name) references customer);

Conclusion: Understand the primary key and foreign key constraints, use of Alter table with add n modify and Drop table command.

FAQs: -

1. What is constraints?
2. What are the types of constraints?
3. What is primary key constraints?
4. What is foreign key constraints?
5. What is difference between primary key constraints and foreign key constraints.
6. What is use of Alter command?
7. Alter command is DDL or DML?
8. What is difference between Drop and Delete?

Assignment No. 4

Date:

Title: 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

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:** To understand the relational operators, Boolean operators and pattern matching, Arithmetic operations and built in functions, Group functions, Date and Time functions, Complex queries and set operators

Theory:

DATA MANIPULATION LANGUAGE (DML):

After the database structure is defined with DDL, database administrators and users can utilize the Data Manipulation Language to insert, retrieve and modify the data contained within it.

INSERT COMMAND:

The INSERT command in MYSQL is used to add records to an existing table.

Format 1:-Inserting a single row of data into a table

Syntax

```
INSERT INTO table_name  
[(columnname,columnname)]  
VALUES (expression,expression);
```

To add a new employee to the personal_info table

Example

```
INSERT INTO personal_info  
values('bart','simpson',12345,$45000)
```


Format 2: Inserting data into a table from another table

Syntax

```
INSERT INTO tablename  
SELECT columnname,columnname  
FROM tablename
```

SELECT COMMAND:

Syntax

```
SELECT * FROM tablename.
```

OR

```
SELECT columnname,columnname,.....  
FROM tablename ;
```

UPDATE COMMAND:

The UPDATE command can be used to modify information contained within a table.

Syntax UPDATE tablename

```
SET columnname=expression,columnname=expression,.....  
WHERE columnname=expression;
```

Each year,company gives all employees a 3% cost-of-living increase in their salary. The following SQL command could be used to quickly apply this to all of the employees stored in the database:

Example

```
UPDATE personal_info  
SET salary=salary*1.03
```

DELETE COMMAND:

The DELETE command can be used to delete information contained within a table.

Syntax

```
DELETE FROM tablename WHERE search condition
```

The DELETE command with a WHERE clause can be used to remove his record from the personal_info table:

Example

```
DELETE FROM personal_info  
WHERE employee_id=12345
```

The following command deletes all the rows from the table

Example `DELETE FROM personal_info;`

LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

The percent sign and the underscore can also be used in combinations.

LIKE Syntax

```
SELECT column1, column2,  
FROM table_name  
WHERE columnN LIKE pattern;
```

The basic syntax of % and _ is as follows:

```
SELECT FROM table_name  
WHERE column LIKE 'XXXX%'  
  
or  
  
SELECT FROM table_name  
WHERE column LIKE '%XXXX%'  
  
or  
  
SELECT FROM table_name  
WHERE column LIKE 'XXXX_'  
  
or  
  
SELECT FROM table_name  
WHERE column LIKE '_XXXX'
```

or

```
SELECT FROM table_name  
WHERE column LIKE '_XXXX_'
```

Here are some examples showing different LIKE operators with '%' and '_' wildcards:

| LIKE Operator | Description |
|--------------------------------|---|
| WHERE CustomerName LIKE 'a%' | Finds any values that starts with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that ends with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a_%_' | Finds any values that starts with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that starts with "a" and ends with "o" |

WHERE LIKE Examples

Problem: List all products with names that start with 'Ca'

```
SELECT Id, ProductName, UnitPrice, Package
```

```
FROM Product
```

```
WHERE ProductName LIKE 'Ca%'
```

Results:

| Id | ProductName | UnitPrice | Package |
|----|-------------------|-----------|-----------------|
| 18 | Carnarvon Tigers | 62.50 | 16 kg pkg. |
| 60 | Camembert Pierrot | 34.00 | 15-300 g rounds |

Aggregate Functions

Aggregate functions return a single result row based on groups of rows, rather than on single rows. Aggregate functions can appear in select lists and in ORDER BY and HAVING clauses. They are commonly used with the GROUP BY clause in a SELECT statement. In a query containing a GROUP BY clause, the elements of the select list can be aggregate functions, GROUP BY expressions, constants, or expressions involving one of these.

Aggregate functions are used to compute against a "returned column of numeric data" from your SELECT statement. They basically summarize the results of a particular column of selected data.

SQL has many built-in functions for performing calculations on data.

| | |
|-------|---|
| MIN | returns the smallest value in a given column |
| MAX | returns the largest value in a given column |
| SUM | returns the sum of the numeric values in a given column |
| AVG | returns the average value of a given column |
| COUNT | returns the total number of values in a given column |

| | |
|----------|--|
| COUNT(*) | returns the number of rows in a table |
| ROUND() | Rounds a numeric field to the number of decimals specified |

The AVG () Function

The AVG () function returns the average value of a numeric column.

SQL AVG () Syntax

```
SELECT AVG (column_name) FROM table_name
```

SQL AVG() Example

The following SQL statement gets the average value of the "Price" column from the "Products" table:

```
SELECT AVG(Price) AS PriceAverage FROM Products;
```

The COUNT () Function

The COUNT() function returns the number of rows that matches a specified criteria.

SQL COUNT(column_name) Syntax

The COUNT(column_name) function returns the number of values (NULL values will not be counted) of the specified column:

```
SELECT COUNT(column_name) FROM table_name;
```

SQL COUNT(*) Syntax

The COUNT(*) function returns the number of records in a table:

```
SELECT COUNT(*) FROM table_name;
```

SQL COUNT(*) Example

The following SQL statement counts the total number of orders in the "Orders" table:

```
SELECT COUNT(*) AS NumberOfOrders FROM Orders;
```

The MAX () Function

The MAX() function returns the largest value of the selected column.

SQL MAX() Syntax

```
SELECT MAX(column_name) FROM table_name;
```

SQL MAX() Example

The following SQL statement gets the largest value of the "Price" column from the "Products" table:

```
SELECT MAX(Price) AS HighestPrice FROM Products;
```

The MIN () Function

The MIN() function returns the smallest value of the selected column.

SQL MIN() Syntax

```
SELECT MIN(column_name) FROM table_name;
```

SQL MIN() Example

The following SQL statement gets the smallest value of the "Price" column from the "Products" table:

```
SELECT MIN(Price) AS SmallestOrderPrice FROM Products;
```

The ROUND () Function

The ROUND () function is used to round a numeric field to the number of decimals specified.

SQL ROUND () Syntax

```
SELECT ROUND(column_name,decimals) FROM table_name;
```

| Parameter | Description |
|-------------|---|
| column_name | Required. The field to round. |
| decimals | Required. Specifies the number of decimals to be returned |

SQL ROUND () Example

The following SQL statement selects the product name and rounds the price in the "Products" table:

```
SELECT ProductName, ROUND(Price,0) AS RoundedPrice  
FROM Products;
```

The SUM () Function

The SUM() function returns the total sum of a numeric column.

SQL SUM() Syntax

```
SELECT SUM(column_name) FROM table_name;
```

SQL SUM() Example

The following SQL statement finds the sum of all the "Quantity" fields for the "OrderDetails" table:

```
SELECT SUM(Quantity) AS TotalItemsOrdered FROM OrderDetails;
```

Date Functions

The following table lists the most important built-in date functions.

| Function | Description |
|---------------|--|
| NOW() | Returns the current date and time |
| CURDATE() | Returns the current date |
| CURTIME() | Returns the current time |
| DATE() | Extracts the date part of a date or date/time expression |
| EXTRACT() | Returns a single part of a date/time |
| DATE_ADD() | Adds a specified time interval to a date |
| DATE_SUB() | Subtracts a specified time interval from a date |
| DATEDIFF() | Returns the number of days between two dates |
| DATE_FORMAT() | Displays date/time data in different formats |

Date Data Types

- 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

NOW () Function

NOW () returns the current date and time.

- **Syntax**
- NOW()
- **Example**
- The following SELECT statement:
- `SELECT NOW(),CURDATE(),CURTIME()`
- will result in something like this:

| NOW() | CURDATE() | CURTIME() |
|---------------------|------------|-----------|
| 2014-11-22 12:45:34 | 2014-11-22 | 12:45:34 |

DATE () Function

The DATE () function extracts the date part of a date or date/time expression.

Syntax

DATE (date)

Example

Assume we have the following "Orders" table:

| OrderId | ProductName | OrderDate |
|---------|------------------|-------------------------|
| 1 | Jarlsberg Cheese | 2014-11-22 13:23:44.657 |

The following SELECT statement:

```
SELECT ProductName, DATE(OrderDate) AS OrderDate
FROM Orders
WHERE OrderId=1
```

Will result in this:

| ProductName | OrderDate |
|------------------|------------|
| Jarlsberg Cheese | 2014-11-22 |

EXTRACT () Function

The EXTRACT () function is used to return a single part of a date/time, such as year, month, day, hour, minute, etc.

Syntax

EXTRACT(unit FROM date)

Example

Assume we have the following "Orders" table:

| OrderId | ProductName | OrderDate |
|---------|------------------|-------------------------|
| 1 | Jarlsberg Cheese | 2014-11-22 13:23:44.657 |

The following SELECT statement:

```
SELECT EXTRACT(YEAR FROM OrderDate) AS OrderYear,
EXTRACT(MONTH FROM OrderDate) AS OrderMonth,
EXTRACT(DAY FROM OrderDate) AS OrderDay
FROM Orders
WHERE OrderId=1
```

Will result in this:

| OrderYear | OrderMonth | OrderDate |
|-----------|------------|-----------|
| 2014 | 11 | 22 |

DATE_ADD () Function

The DATE_ADD () function adds a specified time interval to a date.

Syntax

DATE_ADD(date,INTERVAL expr type)

Where date is a valid date expression and expr is the number of interval you want to add.

Example

Assume we have the following "Orders" table:

| OrderId | ProductName | OrderDate |
|---------|------------------|-------------------------|
| 1 | Jarlsberg Cheese | 2014-11-22 13:23:44.657 |

Now we want to add 30 days to the "OrderDate", to find the payment date.

We use the following SELECT statement:

```
SELECT OrderId,DATE_ADD(OrderDate,INTERVAL 30 DAY) AS OrderPayDate
FROM Orders
```

Result:

| OrderId | OrderPayDate |
|---------|-------------------------|
| 1 | 2014-12-22 13:23:44.657 |

DATE_SUB () Function

The DATE_SUB () function subtracts a specified time interval from a date.

Syntax

DATE_SUB(date,INTERVAL expr type)

Where date is a valid date expression and expr is the number of interval you want to subtract.

Example

Assume we have the following "Orders" table:

| OrderId | ProductName | OrderDate |
|---------|------------------|-------------------------|
| 1 | Jarlsberg Cheese | 2014-11-22 13:23:44.657 |

Now we want to subtract 5 days from the "OrderDate" date.

We use the following SELECT statement:

```
SELECT OrderId, DATE_SUB (OrderDate, INTERVAL 5 DAY) AS SubtractDate  
FROM Orders
```

Result:

| OrderId | SubtractDate |
|---------|-------------------------|
| 1 | 2014-11-17 13:23:44.657 |

DATEDIFF () Function

The DATEDIFF () function returns the time between two dates.

Syntax

DATEDIFF (date1,date2)

Where date1 and date2 are valid date or date/time expressions.

Example

The following SELECT statement:

```
SELECT DATEDIFF ('2014-11-30','2014-11-29') AS DiffDate
```

Will result in this:

| DiffDate |
|----------|
| 1 |

Example

The following SELECT statement:

```
SELECT DATEDIFF('2014-11-29','2014-11-30') AS DiffDate
```

Will result in this:

| DiffDate |
|----------|
| 1 |

DATE_FORMAT () Function

The DATE_FORMAT () function is used to display date/time data in different formats.

Syntax

```
DATE_FORMAT (date,format)
```

Where date is a valid date and format specifies the output format for the date/time.

Example

The following script uses the DATE_FORMAT () function to display different formats. We will use the NOW () function to get the current date/time:

```
DATE_FORMAT(NOW(),'%b %d %Y %h:%i %p')  
DATE_FORMAT(NOW(),'%m-%d-%Y')  
DATE_FORMAT(NOW(),'%d %b %y')  
DATE_FORMAT(NOW(),'%d %b %Y %T:%f')
```

The result would look something like this:

```
Nov 04 2014 11:45 PM  
11-04-2014  
04 Nov 14  
04 Nov 2014 11:45:34:243
```

Nested Sub Query

Definition of Nested Sub Query: -

A Sub query or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause. A sub query is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

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

A sub query can be nested inside other sub queries. SQL has an ability to nest queries within one another. A sub query is a SELECT statement that is nested within another SELECT statement and which return intermediate results. SQL executes innermost sub query first, then next level.

Subqueries with the SELECT Statement:

Sub queries are most frequently used with the SELECT statement.

The basic syntax is as follows:

```
SELECT column_name [, column_name ]
FROM table1 [, table2]
WHERE column_name OPERATOR
      (SELECT column_name [, column_name ]
      FROM table1 [, table2]
      [WHERE])
```

Example of IN Operator

Consider the CUSTOMERS table having the following records:

| ID | NAME | AGE | ADDRESS | SALARY |
|----|----------|-----|-----------|----------|
| 1 | Ramesh | 35 | 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 |
| 7 | Muffy | 24 | Indore | 10000.00 |

Now, let us check following sub query with SELECT statement:

```
SELECT *  
  FROM CUSTOMERS  
 WHERE ID IN (SELECT ID  
              FROM CUSTOMERS  
              WHERE SALARY > 4500);
```

This would produce the following result

| ID | NAME | AGE | ADDRESS | SALARY |
|----|----------|-----|---------|----------|
| 4 | Chaitali | 25 | Mumbai | 6500.00 |
| 5 | Hardik | 27 | Bhopal | 8500.00 |
| 7 | Muffy | 24 | Indore | 10000.00 |

Example of NOT IN Operator

Now, let us check following sub query with SELECT statement:

```
SELECT *  
  FROM CUSTOMERS  
 WHERE ID NOT IN (SELECT ID  
                  FROM CUSTOMERS  
                  WHERE SALARY > 4500);
```

This would produce the following result

| ID | NAME | AGE | ADDRESS | SALARY |
|----|---------|-----|-----------|---------|
| 1 | Ramesh | 35 | Ahmedabad | 2000.00 |
| 2 | Khilan | 25 | Delhi | 1500.00 |
| 3 | kaushik | 23 | Kota | 2000.00 |
| 6 | Komal | 22 | MP | 4500.00 |

Subqueries with the INSERT Statement:

Sub queries also can be used with INSERT statements. The INSERT statement uses the data returned from the sub query to insert into another table. The selected data in the sub query can be modified with any of the character, date or number functions.

Example:

Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy complete CUSTOMERS table into CUSTOMERS_BKP, following is the syntax:

```
INSERT INTO CUSTOMERS_BKP SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID FROM CUSTOMERS);
```

Subqueries with the UPDATE Statement:

The sub query can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a sub query with the UPDATE statement.

Example:

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table.

Following example updates SALARY by 0.25 times in CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
UPDATE CUSTOMERS
SET SALARY = SALARY * 0.25
WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
WHERE AGE >= 27);
```

This would impact two rows and finally CUSTOMERS table would have the following records:

| ID | NAME | AGE | ADDRESS | SALARY |
|----|----------|-----|-----------|----------|
| 1 | Ramesh | 35 | Ahmedabad | 125.00 |
| 2 | Khilan | 25 | Delhi | 1500.00 |
| 3 | kaushik | 23 | Kota | 2000.00 |
| 4 | Chaitali | 25 | Mumbai | 6500.00 |
| 5 | Hardik | 27 | Bhopal | 2125.00 |
| 6 | Komal | 22 | MP | 4500.00 |
| 7 | Muffy | 24 | Indore | 10000.00 |

Subqueries with the DELETE Statement:

The sub query can be used in conjunction with the DELETE statement like with any other statements mentioned above.

Example:

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table.

Following example deletes records from CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
DELETE FROM CUSTOMERS
WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
              WHERE AGE >= 27);
```

This would impact two rows and finally CUSTOMERS table would have the following records:

| + | + | + | + | + | + |
|----|----------|-----|---------|----------|---|
| ID | NAME | AGE | ADDRESS | SALARY | |
| 2 | Khilan | 25 | Delhi | 1500.00 | |
| 3 | kaushik | 23 | Kota | 2000.00 | |
| 4 | Chaitali | 25 | Mumbai | 6500.00 | |
| 6 | Komal | 22 | MP | 4500.00 | |
| 7 | Muffy | 24 | Indore | 10000.00 | |

Conclusion: Understand the relational operators, Boolean operators and pattern matching, Arithmetic operations and built in functions, Group functions, Date and Time functions, Complex queries and set operators in MYSQL.

FAQs: -

1. What are different built in Function in MYSQL?
2. What is aggregate Function?
3. What is pattern matching and which wild card character used for pattern matching?
4. What are different date and time functions?
5. What is nested query?
6. What are different Arithmetic operators used in MYSQL?

Assignment No. 5**Date:**

Title: 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.

Remarks:

Aim: 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.

Objective: Understand the concept of views and his use.

Theory:

What is View?

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax

```
CREATE VIEW view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

SQL CREATE VIEW Examples

If you have the Northwind database you can see that it has several views installed by default.

The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

```
CREATE VIEW [Current Product List] AS  
SELECT ProductID, ProductName  
FROM Products  
WHERE Discontinued = No;
```

Then, we can query the view as follows:

```
SELECT * FROM [Current Product List];
```

MySQL Create View with JOIN

CREATE VIEW command can be used along with a JOIN statement.

Example :

Sample table : category

Sample table : purchase

- CREATE VIEW view_purchase
- AS SELECT a.cate_id,a.cate_descrip, b.invoice_no,
- b.invoice_dt,b.book_name
- FROM category a,purchase b
- WHERE a.cate_id=b.cate_id;

The above MySQL statement will create a view 'view_purchase' along with a JOIN statement.

The JOIN statement here retrieves cate_id, cate_descrip from category table and invoice_no, invoice_dt and book_name from purchase table if cate_id of category table and that of purchase are same.

MySQL Create View with LIKE

CREATE VIEW command can be used with LIKE operator.

Example :

Sample table : author

Code :

1. CREATE VIEW view_author
2. AS SELECT *
3. FROM author
4. WHERE aut_name
5. NOT LIKE 'T%' AND aut_name NOT LIKE 'W%';

The above MySQL statement will create a view 'view_author' taking all the records of author table, if (A)name of the author (aut_name) does not start with 'T' and (B) name of the author (aut_name) does not start with 'W'.

MySQL Create View using Subquery

CREATE VIEW command can be used with subqueries.

Example :

Sample table : purchase

Sample table : book_mast

Code :

1. CREATE VIEW view_purchase
2. AS SELECT invoice_no,book_name,cate_id
3. FROM purchase
4. WHERE cate_id= (SELECT cate_id FROM book_mast WHERE no_page=201)
- 5.

Create table Employee (ID, First_Name,Last_Name,Start_Date,End_Date,Salary,City).

1. Create a simple view to display First_Name,Last_Name from employee.
2. Create a view to display First_Name,Last_Name of those employee whose salary is greater than 2000 from employee table.
3. Create a view to display first_name starting with "S" and last_name end with "t".

Conclusion: Implemented Views and performed operation on view

FAQs: -

1. What is View?
2. What is use of View?
- 3.How to create a view in MYSQL?
4. Can we create view using multiple tables?
5. Can we create view on single tables?

Group C: PL/SQL

Assignment No. 1

Date:

Title: Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.

Remarks:

Aim: Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.

Objective: Understand the concept of PL/SQL stored procedure and function and its use.

Theory:

A subprogram is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the calling program.

A subprogram can be created:

- At schema level
- Inside a package
- Inside a MYSQL block

Parts of a MYSQL Subprog ram

Each MYSQL subprogram has a name, and may have a parameter list. Like anonymous PL/SQL blocks and, the named blocks a subprograms will also have following three parts:

1. Declarative Part
2. Executable part
3. Exception-handling

What is procedure? How to create it?

Procedures: these subprogram rams do not return a value directly, mainly used to perform an action.

Creating a Procedure

A procedure is created with the CREATE OR REPLACE PROCEDURE statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[[parameter_name [IN | OUT | IN OUT] type [, ...]]]

BEGIN
< procedure_body >
END ;
```


Where,
procedure-name specifies the name of the procedure.
[OR REPLACE] option allows modifying an existing procedure.

The optional parameter list contains name, mode and types of the parameters. IN represents, that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.

Procedure-body contains the executable part.

The IS keyword is used for creating a standalone procedure.

The following example creates a simple procedure that displays the string 'Hello World!' on the screen when executed.

```
Delimiter //  
CREATE OR REPLACE PROCEDURE greeting  
Select concat('Hello World!');/
```

When above code is executed using SQL prompt, it will produce the following result:

```
Query ok
```

(2) How to execute procedure?

Executing a Standalone Procedure

- Calling the name of the procedure from a PL/SQL block

```
Call greeting( )
```

```
Output:  
Hello world
```

Deleting a Standalone Procedure

A standalone procedure is deleted with the DROP PROCEDURE statement. Syntax for deleting a procedure is:

```
DROP PROCEDURE procedure-name;
```

So you can drop *greetings* procedure by using the following statement:

```
DROP PROCEDURE greetings;
```

Parameter modes in PL/SQL subprograms:

1. IN:

An IN parameter lets you pass a value to the subprogram.

It is a read-only parameter.

It is the default mode of parameter passing.

Parameters are passed by reference.

2. OUT:

An OUT parameter returns a value to the calling program.

The actual parameter must be variable and it is passed by value.

3. IN-OUT:

An IN OUT parameter passes an initial value to a subprogram and returns an updated value to the caller.

Actual parameter is passed by value.

IN & OUT Mode Example 1

This program finds the minimum of two values, here procedure takes two numbers using IN mode and returns their minimum using OUT parameters.

```
delimiter $
create procedure addp()
begin
declare a,b,c int;
set a=2;
set b=3;
set c=a+b;
select concat('value',c);
end;
$
```

```
delimiter ;
call addp();
Result: value 5
```

```
mysql> delimiter //
mysql> create procedure difference (in a int,in b int, out c int)
-> begin
-> if a>b then
-> set c=1;
-> else if a=b then
-> set c=2;
-> else
-> set c=3;
-> end if;
-> end if;
-> end; //
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> call difference(5,9,@x);
-> select @x;
-> //
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> create table student
-> ( sid int(5) not null,
-> student_name varchar(9),
-> DOB date,
-> primary key(sid));
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> insert into student values(5,'Harry',20130412);
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into student values(6,'Jhon',20100215);
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into student values(7,'Mary',20140516);
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into student values(8,'Kay',20131116);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from student;
+----+-----+-----+
| sid | student_name | DOB      |
+----+-----+-----+
| 5 | Harry      | 2013-04-12 |
| 6 | Jhon       | 2010-02-15 |
| 7 | Mary       | 2014-05-16 |
| 8 | Kay        | 2013-11-16 |
+----+-----+-----+
```

Q] Write a Procedure to display SID & Student.

```
mysql> delimiter //
mysql> create procedure myprocedure()
-> select sid,student_name from student
-> //
Query OK, 0 rows affected (0.55 sec)
```

```
mysql> call myprocedure();//
```

| + | + | + |
|---|-----|--------------|
| + | + | + |
| | sid | student_name |
| + | + | + |
| | 5 | Harry |
| | 6 | Jhon |
| | 7 | Mary |
| | 8 | Kay |

Q] Write a procedure which gets the name of the student when the student id is passed ?

```
mysql> create procedure stud(IN id INT(5),OUT name varchar(9))
```

```
-> begin
```

```
-> select student_name into name
```

```
-> from student
```

```
-> where sid=id;
```

```
-> end//
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> call stud(5,@x)//
```

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

```
mysql> select @x//
```

| + | + |
|---|-------|
| + | + |
| | @x |
| + | + |
| | Harry |
| + | + |

```
1 row in set (0.00 sec)
```

```
mysql> call stud(7,@x)//
```

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

```
mysql> select @x//
```

| + | + |
|---|------|
| + | + |
| | @x |
| + | + |
| | Mary |
| + | + |

```
1 row in set (0.00 sec)
```

```
mysql> call stud(5,@x);
```

```
-> select @x;
```

```
-> //
Query OK, 0 rows affected (0.00 sec)
```

```
+-----+
| @x   |
+-----+
| Harry |
+-----+
```

Q] Write a procedure cleanup() to delete all the students records from student table.

```
mysql> create procedure cleanup()
-> delete from student;
-> //
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call cleanup();//
Query OK, 4 rows affected (0.03 sec)
```

```
mysql> select * from student;//
Empty set (0.00 sec)
```

2.*FUNCTIONS*

Functions: these subprograms return a single value, mainly used to compute and return a value.

Creating a Function:

A standalone function is created using the CREATE FUNCTION statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows:

```
CREATE FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
BEGIN
< function_body >
RETURN variable
END [function_name];
```

Where,

function-name specifies the name of the function.

[OR REPLACE] option allows modifying an existing function.

The optional parameter list contains name, mode and types of the parameters. IN represents that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.

The function must contain a **return** statement.

RETURN clause specifies that data type you are going to return from the function.

function-body contains the executable part.

Example:

Following example illustrates creating and calling a standalone function. The function returns the total number of CUSTOMERS in the customers table. We will use the CUSTOMERS table.

```
delimiter &
mysql> create function hello(s char(20))
-> returns char(50)
-> return concat('hello,s,!');
-> &
```

When above code is executed using MYSQL prompt, it will produce the following result:

```
Query OK, 0 rows affected (0.01 sec)
```

Calling a Function

While creating a function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task. When a program calls a function, program control is transferred to the called function.

A called function performs defined task and when its return statement is executed or when it last end statement is reached; it returns program control back to the main program.

To call a function you simply need to pass the required parameters along with function name and if function returns a value then you can store returned value. Following program calls the function from an anonymous block:

```
->select hello('world');
```

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

```
-> Hell world
```

```
mysql> delimiter *
mysql> create function add1(a int, b int) returns int
-> return (a+b);
-> select add1(10,20);
-> *
```

Query OK, 0 rows affected (0.00 sec)

```
+ .....+
| add1(10,20) |
+ .....+
|      30 |
+ .....+
1 row in set (0.02 sec)
```

Example:

The following is one more example which demonstrates Declaring , Defining , and Invoking a Simple MYSQL Function that computes and returns the maximum of two values.

```
mysql> delimiter //
mysql> CREATE FUNCTION grt(a INT,b INT,c INT) RETURNS INT
-> BEGIN
-> if a>b AND a>c then
-> RETURN a;
-> end if;
-> if b>c AND b>a then
-> RETURN b;
-> end if;
-> RETURN c;
-> end;
-> //
```

Query OK, 0 rows affected (0.12 sec)

```
mysql> select grt(23,78,98);
-> //
```

```
+ .....+
| grt(23,78,98) |
+ .....+
|      98 |
+ .....+
1 row in set (0.05 sec)
```

```
mysql> select grt(23,98,72);
-> //
```

```

+-----+
| grt(23,98,72) |
+-----+
|      98      |
+-----+
1 row in set (0.01 sec)

```

```
mysql> select grt(45,2,3); //
```

```

+-----+
| grt(45,2,3) |
+-----+
|      45      |
+-----+
1 row in set (0.00 sec)

```

```
mysql> delimiter //
```

```

mysql> CREATE FUNCTION odd_even(a INT) RETURNS varchar(20)
-> BEGIN
-> if a%2=0 then
-> RETURN 'even';
-> end if;
-> RETURN 'odd';
-> end;
-> //

```

Query OK, 0 rows affected (0.06 sec)

```
mysql> select odd_even(54);
```

```

-> //
+-----+
| odd_even(54) |
+-----+
| even         |
+-----+
1 row in set (0.03 sec)

```

```
mysql> select odd_even(51); //
```

```

+-----+
| odd_even(51) |
+-----+
| odd          |
+-----+
1 row in set (0.00 sec)

```


Conclusion: performed implementation of procedures and functions in MYSQL successfully.

FAQs: -

1. What is Stored Procedure?
2. What is Stored Function?
3. What are different parameters used Stored Procedure?
4. What is difference between Stored Procedure and Stored Function?
5. What is use of Delimiter?

Assignment No. 2

Date:

Title: Write and execute suitable database triggers. Consider row level and statement level triggers.

Remarks:

Aim: Write and execute suitable database triggers. Consider row level and statement level triggers.

Objectives: To understand the concept of database MYSQL Trigger

Theory:

- 1) Introduction to MYSQL Trigger

What is a Trigger?

A trigger is a MYSQL block structure which is fired when a DML statements like Insert, Delete, Update is executed on a database table. A trigger is triggered automatically when an associated DML statement is executed.

2) Types of triggers

1) Types of Triggers

There are two types of triggers based on the which level it is triggered.

- 1) **Row level trigger** - An event is triggered for each row updated, inserted or deleted.
- 2) **Statement level trigger** - An event is triggered for each sql statement executed.

Trigger Execution Hierarchy

The following hierarchy is followed when a trigger is fired.

- 1) BEFORE statement trigger fires first.
- 2) Next BEFORE row level trigger fires, once for each row affected.
- 3) Then AFTER row level trigger fires once for each affected row. This events will alternates between BEFORE and AFTER row level triggers.
- 4) Finally the AFTER statement level trigger fires.

Syntax of Triggers

The Syntax for creating a trigger is:

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER | }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
```

```

ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
BEGIN
  --- sql statements
END;

```

- *CREATE TRIGGER trigger_name* - This clause creates a trigger with the given name or overwrites an existing trigger with the same name.
- *{BEFORE | AFTER | INSTEAD OF}* - This clause indicates at what time should the trigger get fired. i.e for example: before or after updating a table. INSTEAD OF is used to create a trigger on a view. before and after cannot be used to create a trigger on a view.
- *{INSERT [OR] | UPDATE [OR] | DELETE}* - This clause determines the triggering event. More than one triggering events can be used together separated by OR keyword. The trigger gets fired at all the specified triggering event.
- *[OF col_name]* - This clause is used with update triggers. This clause is used when you want to trigger an event only when a specific column is updated.
- *CREATE [OR REPLACE] TRIGGER trigger_name* - This clause creates a trigger with the given name or overwrites an existing trigger with the same name.
- *[ON table_name]* - This clause identifies the name of the table or view to which the trigger is associated.
- *[REFERENCING OLD AS o NEW AS n]* - This clause is used to reference the old and new values of the data being changed. By default, you reference the values as :old.column_name or :new.column_name. The reference names can also be changed from old (or new) to any other user-defined name. You cannot reference old values when inserting a record, or new values when deleting a record, because they do not exist.
- *[FOR EACH ROW]* - This clause is used to determine whether a trigger must fire when each row gets affected (i.e. a Row Level Trigger) or just once when the entire sql statement is executed(i.e.statement level Trigger).

- *WHEN (condition)* - This clause is valid only for row level triggers. The trigger is fired only for rows that satisfy the condition specified

Trigger Examples

Example 1)

This example is based on the following two tables:

```
CREATE TABLE T4 ( a INTEGER , b CHAR(10));
```

```
CREATE TABLE T5 ( c CHAR(10) , d INTEGER);
```

-- create a trigger that may insert a tuple into T5 when a tuple is inserted into T4.
inserts the reverse tuple into T5:

1) Create trigger as follows:

```
CREATE TRIGGER trig1 AFTER INSERT ON T4
```

```
FOR EACH ROW BEGIN
```

```
INSERT INTO t5 SET c = NEW.b,d = NEW.a;
```

```
END;
```

2) Insert values in T4.

3) Check the values in T5.

Example2)

1)The price of a product changes constantly. It is important to maintain the history of the prices of the products. Create a trigger to update the 'product_price_history' table when the price of the product is updated in the 'product' table.

Create the 'product' table and 'product_price_history' table

```
CREATE TABLE product_price_history
(product_id number(5),
product_name varchar2(32),
supplier_name varchar2(32),
unit_price number(7,2) );
```

```
CREATE TABLE product
(product_id number(5),
product_name varchar2(32),
supplier_name varchar2(32),
unit_price number(7,2) );
drop trigger if exists price_history_trigger;
```

```
CREATE TRIGGER price_history_trigger
```

```
BEFORE UPDATE on product1
```

```
FOR EACH ROW BEGIN
```

```
INSERT INTO product_price_history
```

```
set product_id=old.product_id,
```

```
product_name=old.product_name,
```

```
supplier_name=old.supplier_name,
```

```
unit_price=old.unit_price;
```

```
END
```

3) Lets update the price of a product.

```
UPDATE PRODUCT SET unit_price = 800 WHERE product_id = 100
```

Once the above update query is executed, the trigger fires and updates the 'product_price_history' table.

Example 3

```
create table account(accno int,amount int)
```

Create a trigger on account table before update in new inserted amount is less than "0" then set amount "0" else if amount is greater than 100 then set amount 100

```
CREATE TRIGGER upd_check BEFORE UPDATE ON account
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF NEW.amount < 0 THEN  
  
SET NEW.amount = 0;  
  
ELSEIF NEW.amount > 100 THEN  
  
SET NEW.amount = 100;  
  
END IF;  
  
END  
  
update account set amount= -12 where accno=101
```

Deleting a trigger
DROP TRIGGER

Name

DROP TRIGGER -- Removes a trigger definition from a database.

Synopsis

DROP TRIGGER *name* ON *table*

Parameters

name

The name of the trigger you wish to remove.

table

The name of the table the trigger is on.

Conclusion: Studied and implemented MYSQL Trigger

FAQs:-

- 1) What is trigger and cursor in PL/SQL?
- 2) What are the types of trigger and cursor?
- 3) How to delete a trigger?
- 4) Why we write a create or replace in PL/SQL Block?
- 5) What is row level and statement level trigger?

Assignment No. 3

Date:

Title: Write a PL/SQL block to implement all types of cursor.

Remarks:

Aim: Write a PL/SQL block to implement cursors.

Objective: 1] to understand the basic concept of cursors used in PL/SQL

Theory:

1] Cursor its use:

- 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.

2] Types of 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 , UPDAT E and DELET E) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDAT E and

DELET E operations, the cursor identifies the rows that would be affected.

| Attribute | Desc ription |
|------------|---|
| %FOUND | Returns T RUE if an INSERT , UPDAT E, or DELET E statement affected one or more rows or a SELECT INT O statement returned one or more rows. Otherwise, it returns FALSE. |
| %NOT FOUND | T he logical opposite of %FOUND. It returns T RUE if an INSERT , UPDAT E, or DELET E statement affected no rows, or a SELECT INT O statement returned no rows. Otherwise, it returns FALSE. |
| %ISOPEN | Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement. |
| %ROWCOUNT | Returns the number of rows affected by an INSERT , UPDAT E, or DELET E statement, or returned by a SELECT INT O statement. |

- **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.

The syntax for creating an explicit cursor is :

```
CURSOR cursor_name IS select_statement;
```

Working with an explicit cursor involves four steps:

Declaring the cursor for initializing in the memory

Opening the cursor for allocating memory

Fetching the cursor for retrieving data

Closing the cursor to release 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 memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open 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 aboveopened 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 above-opened cursor as follows:

```
CLOSE c_customers;
```

Cursor Example

Example 1

Create table emp_tbl as follows

```
Emp_tbl(first_name,last_name,salary);
```

Write a procedure with cursor to display employees first name and last name whose salary greater than 1000;

```
drop procedure if exists pcursor;
```

```
create procedure pcursor()
```

```
begin
```

```
DECLARE fn varchar(30);
```

```
declare ln varchar(30);
```

```
DECLARE cur1 CURSOR FOR SELECT first_name,last_name from emp_tbl where  
salary>1000;
```

```
OPEN cur1;
```

```
read_loop: LOOP
```

```
    FETCH cur1 INTO fn,ln;
```

```
select concat(fn,' ',ln) as name;
```

```
end loop;
```

```
CLOSE cur1;
```

```
END
```

Example 2

```
create table t1(id int,data int);( id char(16),data int)
```

```
create table t2(i int);
```

```
create table t3(i1 int,12 int);  //t3 table blank (i1 char(16),i2 int)
```

```
CREATE PROCEDURE curdemo()
```

```
BEGIN
```

```
DECLARE done INT DEFAULT FALSE;
DECLARE a CHAR(16);
DECLARE b, c INT;
DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
OPEN cur1;
OPEN cur2;
read_loop: LOOP
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF b < c THEN
        INSERT INTO test.t3 VALUES (a,b);
    ELSE
        INSERT INTO test.t3 VALUES (a,c);
    END IF;
END LOOP;
CLOSE cur1;
CLOSE cur2;
END;
```

FAQs:

- 1) What is cursor? State its type.
- 2) Explain difference between implicit cursor and explicit cursors.

Conclusion: Thoroughly understood the basic concept of cursors used in PL/SQL.

Group D: Relational Database Design

Design and case study of any organization (**back end only**), Project Proposal and High Level SRS to prepare for project, do the following:

1. Form teams of around 3 to 4 people
2. Create requirements document with the following information: -
 - a. Give one or two paragraph description of your goals for the topic(s).
 - b. List what all types of users will be accessing your application
 - c. List the various functionalities that your application will support. Explain each in about a paragraph worth of detail.
 - d. List the hardware and software requirements at the backend and at the front end.
 - e. Give an estimate of the number of users of each type, the expected load (transactions per day), and the expected database size.

Project ER Diagram and Database Design

For ER diagram and Database design following guidelines can be used:

1. Draw an ER diagram of your project.
2. Reduce this ER diagram into the tables and complete database design.
3. Subsequently, list all the functional dependencies on each table that you expect will hold.
4. Check that the database schema is in 3NF/BCNF. If it is not, apply normalization. Use non-loss decomposition and bring the database schema in 3NF/BCNF.

Give the ER diagram and the data dictionary as part of the requirement specifications file which you created for the project proposal.

Reference Books:

1. Dr. P. S. Deshpande, "SQL and PL/SQL for Oracle 10g Black Book", DreamTech
2. Ivan Bayross, "SQL, PL/SQL: The Programming Language of Oracle", BPB Publication
3. Reese G., Yarger R., King T., Williams H, "Managing and Using MySQL", Shroff Publishers and Distributors Pvt. Ltd., ISBN: 81 - 7366 - 465 – X, 2nd Edition
4. Eric Redmond, Jim Wilson, "Seven databases in seven weeks", SPD, ISBN: 978-93-5023-91
5. Jay Kreibich, Using SQLite, SPD, ISBN: 978-93-5110-934-1, 1st editio