Dhole Patil Education Society's

Dhole Patil College of Engineering, Wagholi Pune

INFORMATION TECHNOLOGY

LABORATORY MANUAL

SE IT (2019 Patt.)

DATABASE MANAGEMENT SYSTEM LABORATORY

(214456)

Prepared by: Dr. Vishvas V. Kalunge

Teaching Scheme:

Practical: 04 hrs/week/batch

Examination Scheme:

Term work: 25 Marks

Practical Exam: 25 Marks

Scanned with OKEN Scanner

Prerequisites:

1. Data structures
2. Software Engineering Principles And Practices

Course Objectives:

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 programme PL/SQL including stored procedures, stored functions, cursors and packages.

Course Outcomes:

CO1: To install and configure database systems.

CO2: To analyze database models and entity relationship models.

CO3: To design and implement a database schema for a given problem-domain.

CO4: To understand the relational database systems.

CO5: To populate and query a database using SQL DDL / DML / DCL commands.

CO6: To design a backend database of any one organization: CASE STUDY.

Scanned with OKEN Scanner

Experiment Learning Outcomes (ELO'S):

Experiment Learning outcome	Attainment	
	PO	Blooms taxonomy
Group A: Study of Databases		
ELO1: Study MySQL Open source software and its characteristics. Install and configure client and server of MySQL.	PO1, PO5, PO9	3
ELO2: Study SQLite databases. Build and install SQLite.	PO1, PO3, PO5, PO9	3
Group B: MySQL		
ELO3: Design database with E-R diagram and implement the database using DDL and DCL commands.	PO1, PO3, PO5, PO9	4
ELO4: Implement queries using DML commands with different operations and functions.	PO1, PO3, PO5, PO9	4
ELO5: Execute DDL statements which demonstrate the use of views.	PO1, PO3, PO5, PO9	4
Group C: PL/SQL		
ELO6: Write and execute PL/SQL stored procedure and function to perform a suitable task on the database.	PO1, PO3, PO5, PO9	4
ELO7: Write and execute suitable database triggers.	PO1, PO3, PO5, PO9	4
ELO8: Write a PL/SQL block to implement all types of cursor.	PO1, PO3, PO5, PO9	4
Group D: Relational Database Design		
ELO9: Design a mini project for any one organization.	PO1, PO2, PO3, PO5, PO9	4

Scanned with OKEN Scanner

INDEX

SOFTWARE LABORATORY-I

Sr. No.	Title
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: Uses of SQLite. Building and installing SQLite.
4	Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system. Design and implement a database using DDL and DCL commands.
5	Implement DML commands with following operations: a. Implement queries with relational operators, boolean operators and pattern matching, arithmetic operations and built in functions. b. Execute the aggregate functions like count, sum, avg etc. c. Retrieve the data from the database based on time and date functions like now (), date (), day (), time () etc. d. Implement nested sub queries. e. Implement queries with set operators.
6	Execute DDL statements which demonstrate the use of views.
7	Write and execute PL/SQL stored procedure and function to perform a suitable task on the database.
8	Write and execute suitable database triggers.
9	Write a PL/SQL block to implement all types of cursor.
10	Design a mini project for any one organization (back-end only).

Scanned with OKEN Scanner

Assignment No: Group A(01)

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

Objectives: To Study MySQL Open source software and compare different characteristics.

Theory:

MySQL is a relational database management system (RDBMS) based on the SQL (Structured Query Language) queries. It is one of the most popular languages for accessing and managing the records in the table. MySQL is open-source and free software under the GNU license. Oracle Company supports it.

Following are the most important features of MySQL:

Relational Database Management System (RDBMS)

MySQL is a relational database management system. This database language is based on the SQL queries to access and manage the records of the table.

Easy to use

MySQL is easy to use. We have to get only the basic knowledge of SQL. We can build and interact with MySQL by using only a few simple SQL statements.

It is secure

MySQL consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.

Client/ Server Architecture

MySQL follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they can query data, save changes, etc.

Free to download

MySQL is free to use so that we can download it from MySQL official website without any cost.

It is scalable

MySQL supports multi-threading that makes it easily scalable. It can handle almost any amount of

Scanned with OKEN Scanner

data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, we can increase this number to a theoretical limit of 8 TB of data.

Speed

MySQL is considered one of the very fast database languages, backed by a large number of the benchmark test.

High Flexibility

MySQL supports a large number of embedded applications, which makes MySQL very flexible.

Compatible on many operating systems

MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

Allows roll-back

MySQL allows transactions to be rolled back, commit, and crash recovery.

Memory efficiency

Its efficiency is high because it has a very low memory leakage problem.

High Performance

MySQL is faster, more reliable, and cheaper because of its unique storage engine architecture. It provides very high-performance results in comparison to other databases without losing an essential functionality of the software. It has fast loading utilities because of the different cache memory.

High Productivity

MySQL uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.

Platform Independent

It can download, install, and execute on most of the available operating systems.

Partitioning

This feature improves the performance and provides fast management of the large database.

GUI Support

MySQL provides a unified visual database graphical user interface tool named "**MySQL**

Scanned with OKEN Scanner

Workbench" to work with database architects, developers, and Database Administrators. MySQL Workbench provides SQL development, data modeling, data migration, and comprehensive administration tools for server configuration, user administration, backup, and many more. MySQL has a fully GUI supports from MySQL Server version 5.6 and higher.

Dual Password Support

MySQL version 8.0 provides support for dual passwords: one is the current password, and another is a secondary password, which allows us to transition to the new password.

Disadvantages/Drawback of MySQL

Following are the few disadvantages of MySQL:

- MySQL version less than 5.0 doesn't support ROLE, COMMIT, and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently, and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

Conclusion: Hence, we have studied MySQL open source software and compared with different characteristics.

Scanned with OKEN Scanner

Assignment No: Group A(2)

Title: Install and configure client and server of MySQL

Scanned with OKEN Scanner

Assignment No: Group A(2)

Aim: Install and configure client and server of MySQL.

Objectives: To install and configure client and server of MySQL.

Theory:

The simplest and recommended method is to download MySQL Installer (for Windows) and let it install and configure a specific version of MySQL Server as follows:

1. Download MySQL Installer from https://dev.mysql.com/downloads/installer/ and execute it.

Note

Unlike the standard MySQL Installer, the smaller web-community version does not bundle any MySQL applications, but downloads only the MySQL products you choose to install.

2. Determine the setup type to use for the initial installation of MySQL products.
   For example:
   - Developer Default: Provides a setup type that includes the selected version of MySQL Server and other MySQL tools related to MySQL development, such as MySQL Workbench.
   - Server Only: Provides a setup for the selected version of MySQL Server without other products.
   - Custom: Enables you to select any version of MySQL Server and other MySQL products.
3. Install the server instance (and products) and then begin the server configuration by following the onscreen instructions. For more information about each individual step,

MySQL is now installed. If you configured MySQL as a service, then Windows automatically starts the MySQL server every time you restart the system. Also, this process installs the MySQL Installer application on the local host, which you can use later to upgrade or reconfigure MySQL server.

Conclusion: Hence, we have studied MySQL installation.

Scanned with OKEN Scanner

Assignment No: Group A (3)

Title: Study of SQLite

Scanned with OKEN Scanner

Assignment No 3: Group A (3)

Aim: Study of SQLite

Objectives: To study SQLite.

Theory:

SQLite is an open-source, embedded, relational database management system, designed circa 2000. It is a lightweight database, with zero configuration, no requirements of a server or installation. Despite its simplicity, it is laden with popular features of database management systems.

History of SQLite:

Sr. No.	Year	Invention
1	2000	D. Richard Hipp designed SQLite to fulfill the purpose of „no administration required" for operating a program.
2	2000	In August, SQLite 1.0 was released with the GNU database manager.
3	2011	Hipp announced to add the UNQl interface to the SQLite database to develop UNQLite (Document-oriented database).

Scanned with OKEN Scanner

Why Use SQLite?

- SQLite is open-source software. The software does not require any license after installation.
- SQLite is server less as it doesn't need a different server process or system to operate.
- SQLite facilitates you to work on multiple databases on the same session simultaneously, thus making it flexible.
- SQLite is a cross-platform DBMS that can run on all platforms, including macOS, Windows, etc.
- SQLite doesn't require any configuration. It needs no setup or administration. When to Use SQLite?
- SQLite is used to develop embedded software for devices like televisions, cell phones, cameras, etc.
- It can manage low to medium-traffic HTTP requests.
- SQLite can change files into smaller size archives with lesser metadata.
- SQLite is used as a temporary dataset to get processed with some data within an application.
- Beginners use SQLite for learning and training purposes, as it requires no installation and configuration.

Installing SQLite on Windows

Now, you will see how to download and install SQLite on Windows in this SQLite tutorial. Follow these steps:

Step1: Go to the official SQLite website and download precompiled binaries from the Windows section.

Step2: Download the file (sqlite-tools-win32-x86-3270200.zip) as shown in the figure below and extract these files in a folder of your choice.

Conclusion: Hence, we have studied SQLite database.

Scanned with OKEN Scanner

Assignment 04

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

Objectives:

- To learn to design database using E-R model.
- To learn and understand database creation, relation / table creation, setting index on table, sequences using MySQL DDL statements.

Theory:

What is ER/EER?

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. At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.

EER Model

EER is a high-level data model that incorporates the extensions to the original ER model. It is a diagrammatic technique for displaying the following concepts

- Sub Class and Super Class
- Specialization and Generalization
- Union or Category

Scanned with OKEN Scanner

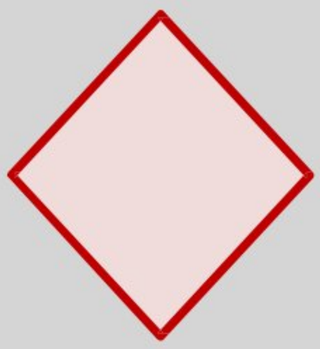
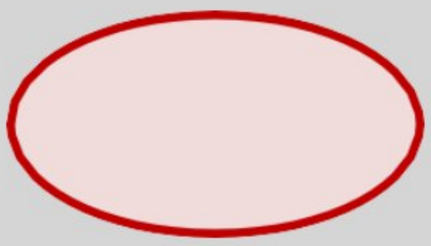
- Aggregation

These concepts are used when they come in EER schema and the resulting schema diagrams called as EER Diagrams.

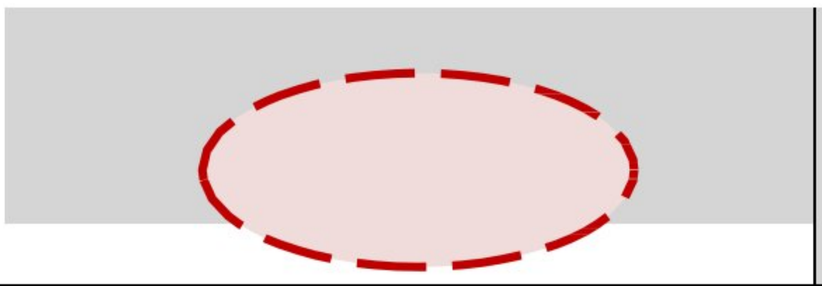
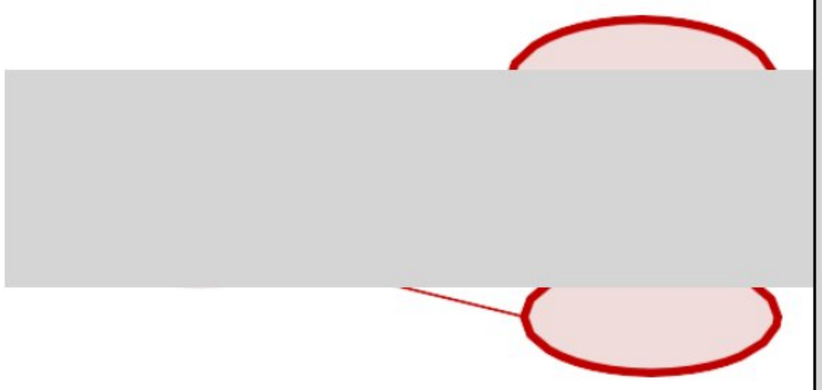
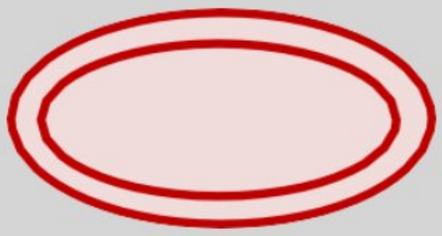
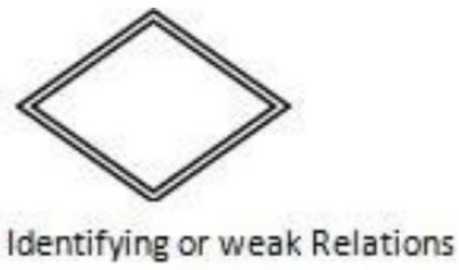
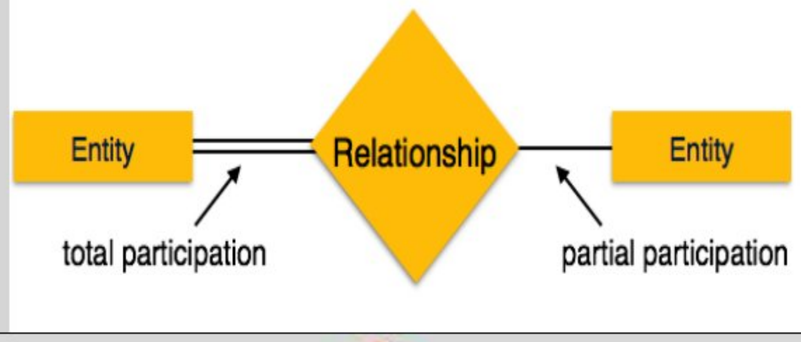
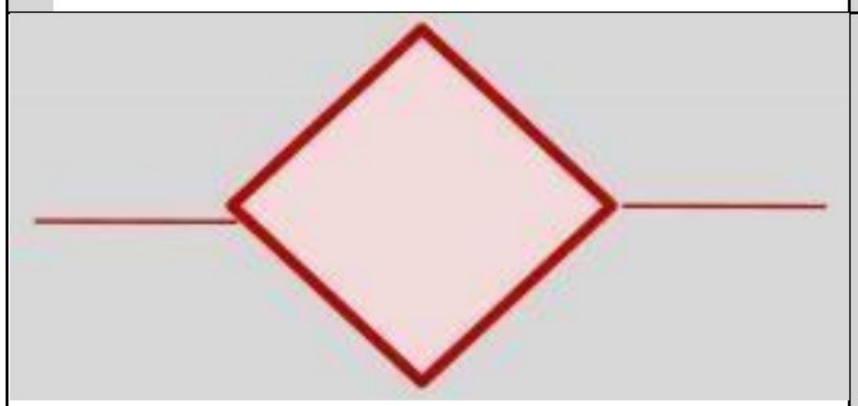
Features of EER Model

- EER creates a design more accurate to database schemas.
- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model.
- Diagrammatic technique helps for displaying the EER schema.
- It includes the concept of specialization and generalization.
- It is used to represent a collection of objects that is union of objects of different of different entity types.

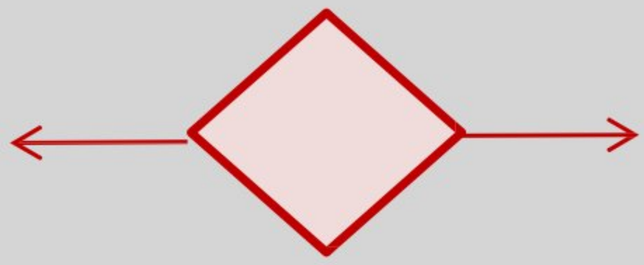
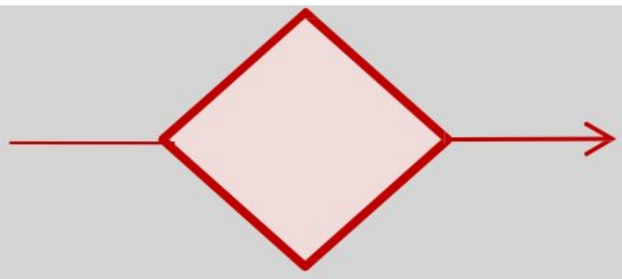
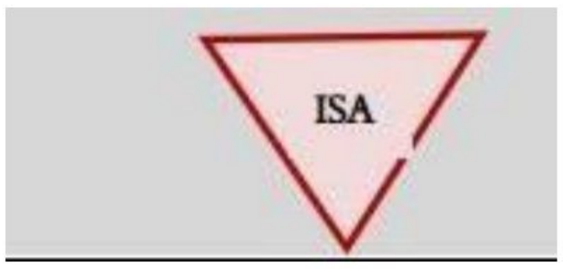
Notations used in ER/EER Diagram:

	Entity Example: Student
	**Relationship Example: Plays**
	**Attribute Examples: Roll_no, Name, Email**

Scanned with OKEN Scanner

	**Derived Attribute** **Example: Age derived from Birth_date**
	**Composite Attribute** **Example: Name - First_name, Middle_name, Last_name**
	**Multivalued Attribute** **Example: Contact_no**
	Key Attribute **Example:Roll_no**
	Weak Entity Set **Example: Animal**
	**Identifying Relationship set for Weak Entity**
	**Total Participation & Partial Participation**
	**Many to Many Relationship**

Scanned with OKEN Scanner

	**One to One Relationship**
	**Many to One Relationship**
	**ISA Relationship:** **Specialization/ Generalization**

Structured Query Language (SQL):

SQL as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

DDL – Data Definition Language

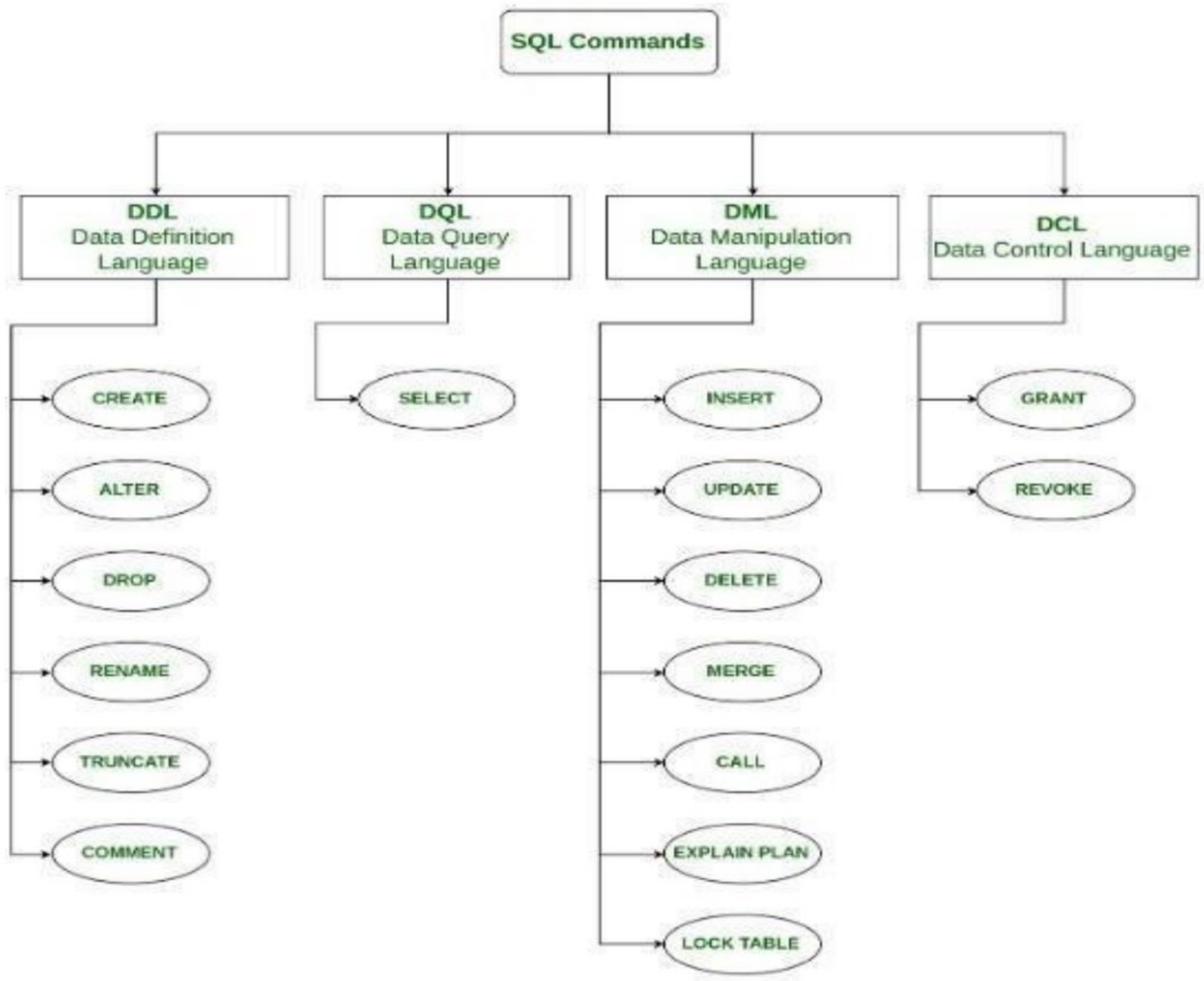
DQl – Data Query Language

DML – Data Manipulation Language DCL

– Data Control Language

Scanned with OKEN Scanner

Though many resources claim there to be another category of SQL clauses TCL – Transaction Control Language. So we will see in detail about TCL as well.



Data Definition Language (DDL) :

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

CREATE – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).

CREATE DATABASE:

Syntax: CREATE DATABASE DatabaseName;

Example: SQL> CREATE DATABASE testDB;

Scanned with OKEN Scanner

CREATE TABLE:

Syntax: CREATE TABLE table_name(

```
    column1 datatype,
    column2 datatype,
    column3 datatype,
    .....
    columnN datatype,

    PRIMARY KEY( one or more columns )

);
```

Example:

```
SQL> CREATE TABLE CUSTOMERS(

    ID      INT                     NOT NULL,
    NAME VARCHAR (20)       NOT NULL,
    AGE    INT                      NOT NULL,
    ADDRESS    CHAR (25) ,
    SALARY     DECIMAL (18, 2),
    PRIMARY KEY (ID)
);
```

DROP – is used to delete objects from the database. **DROP TABLE:**

Syntax: DROP TABLE table_name;

Example: SQL> DROP TABLE CUSTOMERS;

DROP DATABASE:

Syntax: DROP DATABASE **DatabaseName;**

Example: SQL> DROP DATABASE testDB;

ALTER-is used to alter the structure of the database.

Scanned with OKEN Scanner

ALTER TABLE:

Syntax: i) ALTER TABLE table_name ADD column_name datatype;

ii) ALTER TABLE table_name DROP COLUMN column_name;

iii) ALTER TABLE table_name MODIFY COLUMN column_name datatype;

iv) ALTER TABLE table_name MODIFY column_name datatype NOT NULL;

v) ALTER TABLE table_name ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);

vi) ALTER TABLE table_name ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

vii) ALTER TABLE table_name ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

viii) ALTER TABLE table_name DROP CONSTRAINT MyUniqueConstraint;

ix) ALTER TABLE table_name DROP CONSTRAINT MyPrimaryKey;

TRUNCATE–is used to remove all records from a table, including all spaces allocated for the records are removed.

TRUNCATE TABLE:

Syntax: TRUNCATE TABLE table_name;

Example: TRUNCATE TABLE CUSTOMERS;

COMMENT –is used to add comments to the data dictionary.

RENAME –is used to rename an object existing in the database.

Scanned with OKEN Scanner

What are Keys?

Keys play an important role in the relational database. It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables. For example: In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

Types of key:

1. Primary key

It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key. In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique. For each entity, selection of the primary key is based on requirement and developers.

2. Candidate key

A candidate key is an attribute or set of an attribute which can uniquely identify a tuple. The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key. For example: In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.

3. Super Key

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key. For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key. The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

Scanned with OKEN Scanner

4. Foreign key

Foreign keys are the column of the table which is used to point to the primary key of another table. In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table. We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table. Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

Database Constraint:

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be column level or table level. Column level constraints are applied only to one column, whereas table level constraints are applied to the whole table.

Following are commonly used constraints available in SQL.

- **NOT NULL Constraint**: Ensures that a column cannot have NULL value.
- **DEFAULT Constraint**: Provides a default value for a column when none is specified.
- **UNIQUE Constraint**: Ensures that all values in a column are different.
- **PRIMARY Key**: Uniquely identified each rows/records in a database table.
- **FOREIGN Key**: Uniquely identified a rows/records in any another database table.
- **CHECK Constraint**: The CHECK constraint ensures that all values in a column satisfy certain conditions.
- **INDEX**: Use to create and retrieve data from the database very quickly.

NOT NULL Constraint

By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column. A NULL is not the same as no data, rather, it represents unknown data.

DEFAULT Constraint:

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.

Scanned with OKEN Scanner

UNIQUE Constraint:

The UNIQUE Constraint prevents two records from having identical values in a particular column. In the CUSTOMERS table, for example, you might want to prevent two or more people from having identical age.

CHECK Constraint

The CHECK Constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and isn't entered into the table.

Conclusion:

ER Diagram is a design part, which is expected to complete prior to database creation. It gives idea about overall application and how many tables you have to create in database. In this assignment we have also studied relationship between entities, DCL and DDL commands.

Scanned with OKEN Scanner

Assignment 05

Aim: Implement DML commands with following operations:

a. Implement queries with relational operators, boolean operators and pattern matching, arithmetic operations and built in functions.
b. Execute the aggregate functions like count, sum, avg etc.
c. Retrieve the data from the database based on time and date functions like now (), date (), day (), time () etc.
d. Implement nested sub queries.
e. Implement queries with set operators.

Objectives:

- To learn and understand database operations such as Insert, Select, Update, and Delete using MySQL DML statements.
- To embed various operators and functions in queries.
- To implement nested sub-queries.

Theory:

Data-manipulation language (DML): The SQL DML provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.

Inserting Data into Table:

To insert data into MySQL table, you would need to use SQL **INSERT INTO** command. You can insert data into MySQL table by using mysql> prompt or by using any script like PHP.

Syntax:

Here is generic SQL syntax of INSERT INTO command to insert data into MySQL table:

```
INSERT INTO table_name ( field1, field2,...fieldN )
                VALUES
                ( value1, value2,...valueN );
```

To insert string data types, it is required to keep all the values into double or single

Scanned with OKEN Scanner

quote, for example:-**"value"**.

Inserting Data from Command Prompt:

Displaying Data from Table:

The SQL **SELECT** command is used to fetch data from MySQL database. You can use this command at mysql> prompt as well as in any script like PHP.

Syntax:

Here is generic SQL syntax of SELECT command to fetch data from MySQL table:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE Clause]
[OFFSET M ][LIMIT N]
```

- You can use one or more tables separated by comma to include various conditions using a WHERE clause, but WHERE clause is an optional part of SELECT command.
- You can fetch one or more fields in a single SELECT command.
- You can specify star (*) in place of fields. In this case, SELECT will return all the fields.
- You can specify any condition using WHERE clause.
- You can specify an offset using **OFFSET** from where SELECT will start returning records. By default offset is zero.

You can limit the number of returns using **LIMIT** attribute.

Scanned with OKEN Scanner

Aggregate Functions in SQL: Total five aggregate functions are used in MySQL.

Sr. No.	Aggregate Function	User	Example
1.	COUNT ()	It is used to count number of records available.	Select count(*) from student; Select count(roll_no) from student;
2.	MAX ()	It returns Maximum value	Select max(salary) from employee;
3.	MIN ()	It returns Minimum value	Select min(salary) from employee;
4.	AVG ()	It returns Average value	Select avg(salary) from employee;
5.	SUM ()	It returns sum of all value	Select sum(salary) from employee;

Date and Time Functions:

Function	Description
CURDATE	Returns the current date
CURRENT_DATE	Returns the current date
CURRENT_TIME	Returns the current time
CURRENT_TIMESTAMP	Returns the current date and time
CURTIME	Returns the current time
DATE	Extracts the date value from a date or datetime expression
DATEDIFF	Returns the difference in days between two date values
DATE_FORMAT	Formats a date as specified by a format mask
DAY	Returns the day portion of a date value

Scanned with OKEN Scanner

DAYNAME	Returns the weekday name for a date
DAYOFMONTH	Returns the day portion of a date value
DAYOFWEEK	Returns the weekday index for a date value
DAYOFYEAR	Returns the day of the year for a date value
EXTRACT	Extracts parts from a date
NOW	Returns the current date and time
SYSDATE	Returns the current date and time
TIME	Extracts the time value from a time/datetime expression
TIMEDIFF	Returns the difference (expressed as a time value) between two time/datetime values
TIMESTAMP	Converts an expression to a datetime value and if specified adds an optional time interval to the value
TO_DAYS	Converts a date into numeric days
YEARWEEK	Returns the year and week for a date value

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.

Scanned with OKEN Scanner

Sub queries 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])

Numeric Functions:

Function	**Description**
ABS	Returns the absolute value of a number
CEIL	Returns the smallest integer value that is greater than or equal to a number
DIV	Used for integer division
EXP	Returns e raised to the power of number
FLOOR	Returns the largest integer value that is less than or equal to a number
GREATEST	Returns the greatest value in a list of expressions
LEAST	Returns the smallest value in a list of expressions
LN	Returns the natural logarithm of a number
MOD	Returns the remainder of n divided by m
POWER	Returns m raised to the nth power
ROUND	Returns a number rounded to a certain number of decimal places
SQRT	Returns the square root of a number
TRUNCATE	Returns a number truncated to a certain number of decimal places

Scanned with OKEN Scanner

String Functions:

S. No.	Name & Description
1	**ASCII()** Returns numeric value of left-most character
2	**CHAR_LENGTH()** Returns number of characters in argument
3	**CHARACTER_LENGTH()** A synonym for CHAR_LENGTH()
4	**CONCAT()** Returns concatenated string

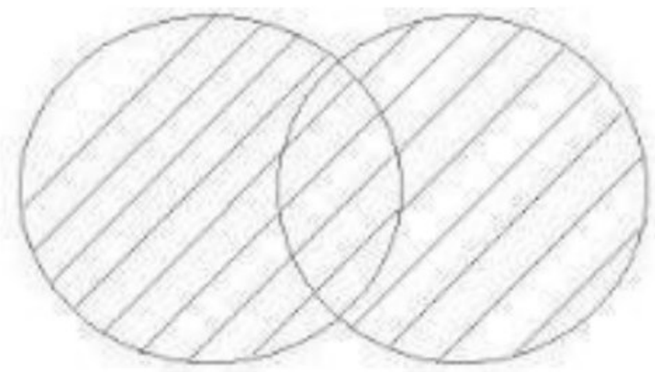
Scanned with OKEN Scanner

5	**INSERT()** Inserts a substring at the specified position up to the specified number of characters
6	**LCASE()** Synonym for LOWER()
7	**LENGTH()** Returns the length of a string in bytes
8	**LOWER()** Returns the argument in lowercase
9	**REPLACE()** Replaces occurrences of a specified string
10	**REVERSE()** Reverses the characters in a string
11	**STRCMP()** Compares two strings
12	**UCASE()** Synonym for UPPER()
13	**UPPER()** Converts to uppercase

----------Set operators----------

Union

UNION is used to combine the results of two or more Select statements. However it will eliminate duplicate rows from its result set. In case of union, number of columns and datatype must be same in both the tables.



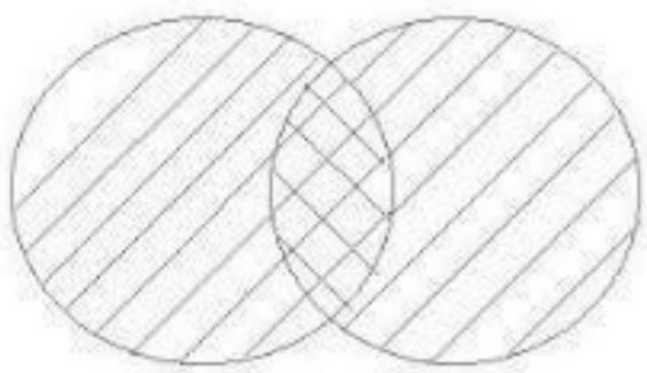
Scanned with OKEN Scanner

Union SQL query will be,

*Select * from first UNION Select * from second;*

Union All

This operation is similar to Union. But it also shows the duplicate rows.

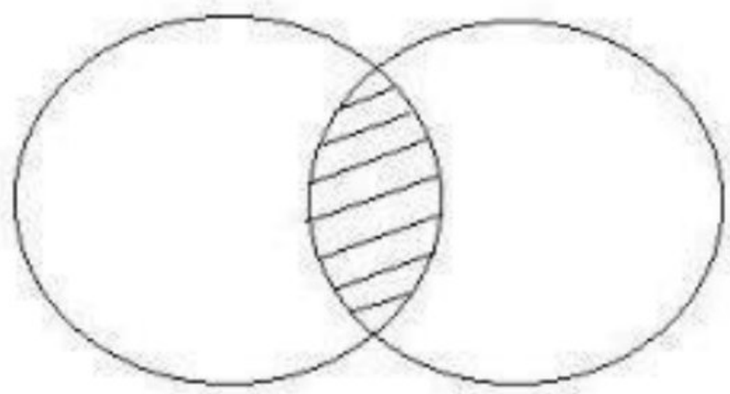


Union All query will be like,

*Select * from first UNION ALL Select * from second;*

Intersect

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of Intersect the number of columns and datatype must be same. MySQL does not support INTERSECT operator.



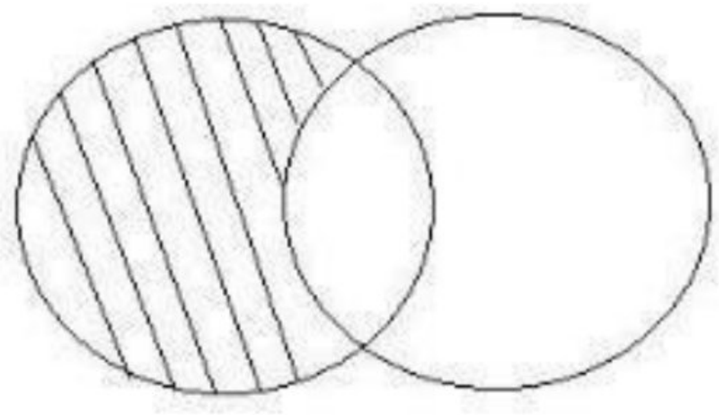
Intersect query will be,

*Select * from first intersect Select * from second;*

Scanned with OKEN Scanner

Minus

Minus operation combines result of two Select statements and return only those result



which belongs to first set of result.

Minus query will be,

*Select * from first minus Select * from second;*

<u>MySQL However does not support Minus and Intersect Operator..</u>

Conclusion:

In this assignment, we have learnt to implement DML queries to include relational operators, boolean operators and pattern matching, arithmetic operations and built in functions, aggregate functions, time and date functions and set operators.

Scanned with OKEN Scanner

Assignment 06

Aim: Execute DDL statements which demonstrate the use of views.

Objectives:

- To create view and retrieve data from tables using view.
- To understand restriction on implementation of views.
- To compare between view and table.

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;

Scanned with OKEN Scanner

Then, we can query the view as follows:

SELECT * FROM Current_Product_List;

MySQL supports version system for views. Each time when the view is altered or replaced, a copy of the existing view is back up in arc (archive) folder which resides in a specific database folder.

MySQL also allows you to create a view of views. In the SELECT statement of view definition, you can refer to another view.

If the cache is enabled, the query against a view is stored in the cache. As the result, it increases the performance of the query by pulling data from the cache instead of querying data from the underlying tables.

Creating Updateable Views

In MySQL, views are not only read-only but also updateable. However in order to create an updateable view, the SELECT statement that defines the view has to follow several following rules:

- The SELECT statement must only refer to one database table.
- The SELECT statement must not use GROUP BY or HAVING clause.
- The SELECT statement must not use DISTINCT in the column list of the SELECT clause.
- The SELECT statement must not refer to read-only views.
- The SELECT statement must not contain any expression (aggregates, functions, computed columns…)

MySQL Create View with JOIN

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

Example:

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

Scanned with OKEN Scanner

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:

*CREATE VIEW view_author AS SELECT * FROM author WHERE aut_name 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:

CREATE VIEW view_purchase AS SELECT invoice_no,book_name,cate_id FROM purchase WHERE cate_id= (SELECT cate_id FROM book_mast WHERE no_pag e=201)

MySQL drop views

Once a view created, you can remove it by using the DROP VIEW statement. The following illustrates the syntax of the DROP VIEW statement:

DROP VIEW [IF EXISTS] [database_name].[view_name]

The IF EXISTS is the optional element of the statement, which allows you to check whether the view exists or not. It helps you avoid an error of removing a non-existent view.

Scanned with OKEN Scanner

For example, if you want to remove the organization view, you can use the DROP VIEW statement as follows:

DROP VIEW IF EXISTS organization

Each time you modify or remove a view, MySQL makes a backup of the view definition file to the/database_name/arc/ folder. In case you modify or remove a view by accident, you can get a backup from there.

Conclusion:

- We have learnt how to create an updateable view and how to update data in the underlying table through the view.
- We have learnt how to manage views in MySQL including displaying, modifying and removing views.

Scanned with OKEN Scanner

Assignment 07

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

Objectives:

- To learn about MySQL stored procedures, their advantages and disadvantages.
- To learn about variables in stored procedure, its scope, how to declare and use variables.
- To learn how to write MySQL stored procedures with parameters.
- To learn how to create stored functions using CREATE FUNCTION statement.

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 Subprogram:

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:

Scanned with OKEN Scanner

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

Definition of stored procedures

A stored procedure is a segment of declarative SQL statements stored inside the database catalog. A stored procedure can be invoked by triggers, other stored procedures or applications such as Java, C#, PHP, etc.

A stored procedure that calls itself is known as a recursive stored procedure. Most database management system supports recursive stored procedures.

However MySQL does not support it very well. You should check your version of MySQL database before implementing recursive stored procedures in MySQL.

MySQL stored procedures advantages

- Typically stored procedures help increase the performance of the applications. Once created, stored procedures are compiled and stored in the database. However MySQL implements the stored procedures slightly different. MySQL stored procedures are compiled on demand. After compiling a stored procedure, MySQL puts it to a cache. And MySQL maintains its own stored procedure cache for every single connection. If an application uses a stored procedure multiple times in a single connection, the compiled version is used, otherwise the stored procedure works like a query.
- A stored procedure helps reduce the traffic between application and database server because instead of sending multiple lengthy SQL statements, the application has to send only name and parameters of the stored procedure.
- Stored procedures are reusable and transparent to any applications. Stored procedures expose the database interface to all applications so that developers don't have to develop functions that are already supported in stored procedures.
- Stored procedures are secure. Database administrator can grant appropriate permissions to applications that access stored procedures in the database without giving any permission on the underlying database tables.

Besides those advantages, stored procedures have their own disadvantages, which

Scanned with OKEN Scanner

you should be aware of before using the store procedures.

MySQL stored procedures disadvantages

- If you use a lot of stored procedures, the memory usage of every connection that is using those stored procedures will increase substantially. In addition, if you overuse a large number of logical operations inside store procedures, the CPU usage will also increase because database server is not well-designed for logical operations.
- A constructs of stored procedures make it more difficult to develop stored procedures that have complicated business logic.
- It is difficult to debug stored procedures. Only few database management systems allow you to debug stored procedures. Unfortunately, MySQL does not provide facilities for debugging stored procedures.
- It is not easy to develop and maintain stored procedures. Developing and maintaining stored procedures are often required specialized skill set that not all application developers possess. This may lead to problems in both application development and maintenance phases.

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.

Scanned with OKEN Scanner

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.

How to execute procedure?

Executing a Standalone Procedure

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

Call greeting( )

Deleting a Standalone Procedure

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

DROP PROCEDURE procedure-name;

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

Scanned with OKEN Scanner

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.

Calling Procedure:

```
mysql> delimiter ;
mysql> call addp();
```

```
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;
```

```
mysql> call difference(5,9,@x);
-> select @x;
-> //
```

Scanned with OKEN Scanner

PROCEDURES ON TABLES

To run the procedures on table, lets create a sample table and insert some values in that.

```
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)
```

Insert few rows in the table created above.

FUNCTIONS

Functions: these subprograms return a single value, mainly used to compute and return a value. A stored function is a special kind stored program that returns a single value. You use stored functions to encapsulate common formulas or business rules that may be reusable among SQL statements or stored programs.

Different from a stored procedure, you can use a stored function in SQL statements wherever an expression is used. This helps improve the readability and maintainability of the procedural code.

Creating a Function:

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

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

Scanned with OKEN Scanner

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

Scanned with OKEN Scanner

CUSTOMERS table.

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

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');
```

Conclusion:

In this assignment, we have learned how to write a simple stored procedure by using the CREATE PROCEDURE statement and call it by using the CALL statement. Also we have learnt, creation of function using CREATE FUNCTION and calling it by its name.

Scanned with OKEN Scanner

Assignment 08

Aim: Write and execute suitable database triggers.

Objectives:

- To write trigger to implement set of actions on various events.
- To understand types of triggers.

Theory:

What is Database Trigger?

A trigger is a named database object that is associated with a table, and that activates when a particular event occurs for the table. Some uses for triggers are to perform checks of values to be inserted into a table or to perform calculations on values involved in an update.

A trigger is defined to activate when a statement inserts, updates, or deletes rows in the associated table. These row operations are trigger events. For example, rows can be inserted by INSERT or LOAD DATA statements, and an insert trigger activates for each inserted row. A trigger can be set to activate either before or after the trigger event. For example, you can have a trigger activate before each row that is inserted into a table or after each row that is updated.

Trigger is applied for following commands:

1. Before Insert
2. After Insert
3. Before Delete
4. After Delete
5. Before Update
6. After Update

Scanned with OKEN Scanner

Scanned with OKEN Scanner

	Before	After
Insert	**new**	**new**
Update	**old**	**new**
Delete	**old**	**old**

Here is a simple example that associates a trigger with a table, to activate for INSERT operations. The trigger acts as an accumulator, summing the values inserted into one of the columns of the table.

mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2)); Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER
ins_sum BEFORE
INSERT ON account FOR
EACH ROW
SET @sum = @sum + NEW.amount; Query OK, 0 rows affected (0.01 sec)

The CREATE TRIGGER statement creates a trigger named ins_sum that is associated with the account table. It also includes clauses that specify the trigger action time, the triggering event, and what to do when the trigger activates:

- The keyword BEFORE indicates the trigger action time. In this case, the trigger activates before each row inserted into the table. The other permitted keyword here is AFTER.
- The keyword INSERT indicates the trigger event; that is, the type of operation that activates the trigger. In the example, INSERT operations cause trigger activation. You can also create triggers for DELETE and UPDATE operations.

Scanned with OKEN Scanner

- The statement following FOR EACH ROW defines the trigger body; that is, the statement to execute each time the trigger activates, which occurs once for each row affected by the triggering event. In the example, the trigger body is a simple SET that accumulates into a user variable the values inserted into the amount column. The statement refers to the column as NEW.amount which means "the value of the amount column to be inserted into the new row."
- To use the trigger, set the accumulator variable to zero, execute an INSERT statement, and then see what value the variable has afterward:

mysql> SET @sum = 0;

mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,- 100.00);

mysql> SELECT @sum AS 'Total amount inserted';

+---------+ | Total amount inserted | +-----+ | 1852.48 | + +

Types of Triggers

There are two types of triggers based on 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.

Conclusion:

In this assignment, we have learnt how trigger helps to implement action on some event. Also we understood the types of trigger.

Scanned with OKEN Scanner

Assignment 09

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

Objectives:

- To learn how to use **MySQL cursor** in stored procedures to iterate through a result set returned by a SELECT statement.
- To differentiate between the types of cursor.

Theory:

MySQL cursor

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

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

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

Scanned with OKEN Scanner

You can use MySQL cursors in stored procedures, stored functions and triggers.

Working with MySQL cursor

First, you have to declare a cursor by using the DECLARE statement:

```
DECLARE cursor_name CURSOR FOR SELECT_statement;
```

The cursor declaration must be after any variable declaration. If you declare a cursor before variables declaration, MySQL will issue an error. A cursor must always be associated with a SELECT statement.

Next, you open the cursor by using the OPEN statement. The OPEN statement initializes the result set for the cursor therefore you must call the OPEN statement before fetching rows from the result set.

```
OPEN cursor_name;
```

Then, you use the FETCH statement to retrieve the next row pointed by the cursor and move the cursor to the next row in the result set.

```
FETCH cursor_name INTO variables list;
```

After that, you can check to see if there is any row available before fetching it.

Finally, you call the CLOSE statement to deactivate the cursor and release the memory associated with it as follows:

```
CLOSE cursor_name;
```

When the cursor is no longer used, you should close it.

When working with MySQL cursor, you must also declare a NOT FOUND handler to handle the situation when the cursor could not find any row.

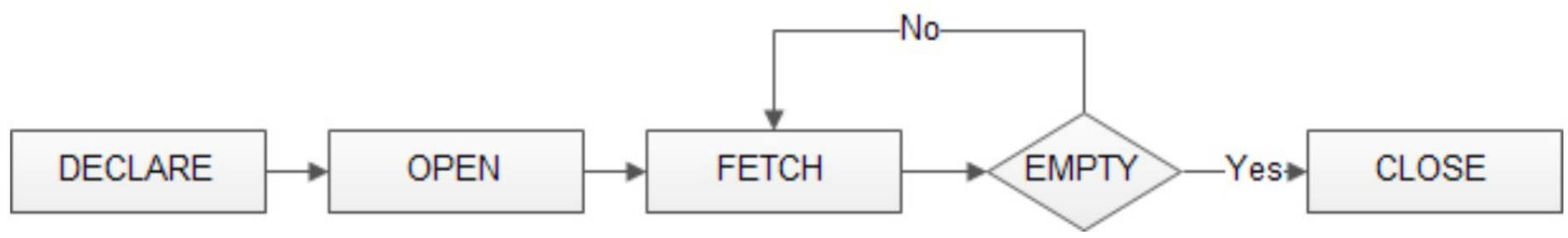
Scanned with OKEN Scanner

Because each time you call the FETCH statement, the cursor attempts to read the next row in the result set. When the cursor reaches the end of the result set, it will not be able to get the data, and a condition is raised. The handler is used to handle this condition.

To declare a NOT FOUND handler, you use the following syntax:

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
```

The following diagram illustrates how MySQL cursor works.



MySQL Cursor Example

We are going to develop a stored procedure that builds an email list of all employees in the employees table in the company sample database provided with this assignment

First, we declare some variables, a cursor for looping over the emails of employees, and a NOT FOUND handler:

```
DECLARE finished INTEGER DEFAULT 0;
DECLARE email varchar(255) DEFAULT "";

-- declare cursor for employee email
DEClARE email_cursor CURSOR FOR
    SELECT email FROM employees;

-- declare NOT FOUND handler
DECLARE CONTINUE HANDLER
```

Scanned with OKEN Scanner

```
FOR NOT FOUND SET finished = 1;
```

Next, we open the email_cursor by using the OPEN statement:

```
OPEN email_cursor;
```

Then, we iterate the email list, and concatenate all emails where each email is separated by a semicolon(;):

```
get_email: LOOP
    FETCH email_cursor INTO v_email;
    IF v_finished = 1 THEN
        LEAVE get_email;
    END IF;
    -- build email list
    SET email_list = CONCAT(v_email,";",email_list);
END LOOP get_email;
```

After that, inside the loop we used the v_finished variable to check if there is any email in the list to terminate the loop.

Finally, we close the cursor using the CLOSE statement:

```
CLOSE email_cursor;
```

Conclusion:

In this assignment, we have learnt the significance of cursor, its components and types of cursor.

Scanned with OKEN Scanner

Assignment 10

Aim: Build mini project based on the requirement.

Objectives:

- Create database using MySQL for any one system.
- Design and implement back-end for the application.
- Prepare report for the mini project.

Guidelines:

Design database of any organization (back end only) and prepare Project Proposal and High Level SRS.

To prepare for your project, do the following:

1. Form teams of around 3 to 4 people.
2. Create a requirements document with the following information;

    a. Give a one or two paragraph description of your goals for the topic(s).

    b. List all what all types of users will be accessing your application (e.g., for moodle, the types are teachers, students, teaching assistants, and a few more types).

    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.

Scanned with OKEN Scanner

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 lossless 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.

Scanned with OKEN Scanner