

Lab Exercise From Fundamentals of DB

**(Structured Query Language)
“SQL”**

PREPARED BY: -INST. D.M

2 Introduction

- **Query:** An instance of questioning written in SQL.
- **Database language:**
 - **DDL** - Database Definition Language
 - **DML** - Database Manipulation Language
 - **DCL** - Database Control Language

3 –Introduction...

- SQL stands for Structured Query Language and it is a standard language for accessing and manipulating Relational databases.
- SQL is originally based on Relational algebra.
- SQL is not case sensitive.
- RDBMS is the basis for SQL:
 - E.g. RDBMS: MS SQL Server, Oracle, MySQL, and Microsoft Access

4 SQL: (DDL)

- DDL are used to define the structure of our database. (Design)
- Create (new database structure definition)
- Alter (Modifying an existing database structure)
 - ❖ *Drop (Deleting a structure)*
 - ❖ *Add (adding structural components in to our defined database structure)*

5 SQL: (DML)

- **DML** are used to manipulate the records we have in our database. (Data manipulation)
- **Insert** (adding a Record in to a database)
- **Select** (Retrieving Records from a database)
- **Update** (updating a record in a database)
- **Delete** (Deleting a record from a database)

6 SQL: CREATE TABLE STATEMENT (DDL)

– **Oracle 10 g Express Edition command**

CREATE TABLE Employee(

EID number,

Ename varchar2(30),

Address varchar2(30),

Primary key(EID)

);

7 SQL: ALTER TABLE STATEMENT (DDL)

- **To add a new column**

- **Syntax:**

ALTER TABLE table_name ADD column_name datatype;

- ***E.g. To add a new column sex***

- **Oracle**

- ***ALTER TABLE Employee ADD sex varchar2(6);***

8

SQL: ALTER TABLE STATEMENT (DDL)...

- **To delete a column**

- **Syntax:**

ALTER TABLE table_name Drop column Colname;

- **E.g. delete the sex column**

- ***ALTER TABLE Employee DROP COLUMN sex;***

9

SQL: DROP TABLE STATEMENT (DDL)...

- **Used to delete entire table structure**

- **Syntax:**

DROP TABLE table_name;

- **E.g. To delete employee table**

- ***DROP TABLE Employee;***

10 SQL: Creating relationship between tables

- Relation ship between tables is created through the *foreign key* referential integrity rule.
- E.g. relationship b/n department & Employee.
- **CREATE TABLE Department (deptId int, DeptName char(20), Office char(15), Primary key(deptId));**

11

SQL: Creating relationship between tables...

```
CREATE TABLE Employee(Eid char(10) PRIMARY KEY,  
    Ename char(20),  
    Birth_date varchar(15),  
    Sex char(6),  
    Salary number,  
    deptId int,  
    FOREIGN KEY (deptId) REFERENCES Department(deptId));
```

12 SQL: INSERT INTO STATEMENT (DML)

- To insert new records in to tables.
- Syntax: two forms
 - **INSERT INTO table_name
VALUES (value1, value2, value3,...)** -> All columns
 - **INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2, value3,...)** -> Some columns
- E.g. inserting a new employee info on employee table
 - INSERT INTO Employee
VALUES (1, 'Michael', '11/10/1988', 'A.A', 3800, 5)
 - INSERT INTO Employee (Eid, Salary, Ename)
VALUES (1, 3800, 'Michael')
- *While inserting records, text and character fields are enclosed in a **single quote** (' ') and numeric fields are not !*

13 SQL: UPDATE STATEMENT (DML)

- To update a record.
- Syntax
 - `UPDATE table_name`
`SET column1=value, column2=value2,...`
`WHERE some_column=some_value`
- ***WHERE => Search criteria***
- ***Notice:-** WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!*

14

SQL: UPDATE STATEMENT (DML)...

- **E.g. Update Michaels' salary to 5000.**

- **UPDATE Employee
SET Salary= 5000
WHERE Ename = 'Michael';** -> **Salary will be 5000
For all employees
named Michael.**
- **UPDATE Employee
SET Salary= 5000
WHERE Eid =1;** -> **Salary will be 5000
For one employee
Michael (EID=1).**
- **UPDATE Employee
SET Salary= 5000;** -> **Salary will be 5000
For all employees
in our table.**

15 SQL: DELETE STATEMENT (DML)

- It is used to Delete a record
- Syntax
 - DELETE FROM table_name
WHERE some_column=some_value
 - *WHERE => Search criteria*
- Notice:- WHERE clause specifies w/c record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

16

SQL: DELETE STATEMENT (DML)...

- E.g. Delete Michaels' record from employee table.
- **DELETE FROM Employee
WHERE Ename= 'Michael';** -> all employees named Michael will be deleted.
- **DELETE FROM Employee
WHERE EID= 1;** -> one employee named Michael(1) will be deleted.
- **DELETE FROM Employee;
are
TRUNCATE.** -> all employee records deleted! Same as

17 SQL: SELECT STATEMENT (DML)

- SELECT: Retrieves rows from the database and allows the selection of one or many rows or columns from one or many tables.
- **Summarized syntax:**
 - **SELECT *Col1,col2,..***
FROM *table_Name ,...*
[WHERE *search_condition*]
[GROUP BY *Colmun_name*]
[HAVING *search_condition*]
[ORDER BY *Colmun_name* [ASC | DESC]]
 - ***search_condition* => some_column=some_value**

18

SQL: SELECT STATEMENT (DML)...

- ***SELECT Clause*** Specifies the columns to be returned by the query.

- ***Syntax:***

SELECT [ALL | DISTINCT]

< Col1,col2,.... > | * FROM table_name

19 SQL: SELECT STATEMENT (DML)...

- E.g. Retrieve all employees information.
 - **SELECT * FROM Employee;**
- E.g. Retrieve id & name of all employees.
 - **SELECT EID, Ename FROM Employee;**
- E.g. Retrieve all employee name (without duplicate values)
 - **SELECT DISTINCT ENAME FROM Employee;**

20 SQL: SELECT STATEMENT (DML)...

- E.g. retrieve all employee information of EID 1.
 - `SELECT * FROM Employee WHERE EID=1;`
- E.g. retrieve the employee name & salary of EID 1.
 - `SELECT Ename, Salary FROM Employee WHERE EID=1;`
- E.g. Retrieve the name of all employees who are paid salary 3800 in department 5.
 - `SELECT Ename FROM Employee WHERE Salary=3800 AND DeptId=5;`

21 LIKE operator

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

– Syntax

– **SELECT** column_name(s)
FROM table_name
WHERE column_name **LIKE** pattern

22 LIKE operator...

Example

PROJECT

Project_id	City
001	Jimma
002	Adama
003	Jinka

- to select the Project found in a city that starts with “J”
 - `SELECT * FROM Project`
`WHERE City LIKE 'J%'` (Result-Row1&3)
- to select the Project found in a city that ends with “a”
 - `SELECT * FROM Project`
`WHERE City LIKE '%a'` (Result-all Rows)

23 LIKE operator...

- to select the Project found in a city that contains the pattern “mm”
 - `SELECT * FROM Project`
`WHERE City LIKE '%mm%';` Result(Row 1)
- to select the Project found in a city that does NOT contain the pattern “mm”, by using the NOT keyword.
 - `SELECT * FROM Project`
`WHERE City NOT LIKE '%mm%';` Result(Row 2&3)

24 LIKE operator...

- To select Project found in a city that starts with any character, followed by "a"
 - `SELECT * FROM Project`
`WHERE City LIKE '_a'` `Result(None)`
- to select Project found in a city that starts with "J", followed by any character, followed by "M", followed by any character, followed by "a"
 - `SELECT * FROM project`
`WHERE City LIKE 'J_m_a'` `Result(Row1)`

25 LIKE operator...

- To select all record found in employee table where the name of employee start with letter 'G';
- `select * from employee where ename like '%G';`

Result (all Rows!)

26 IN Operator

- The IN operator allows you to specify **multiple values** in a WHERE clause.
- Syntax
 - **SELECT column_name(s)**
FROM table_name
WHERE column_name IN (value1,value2,...)

27 IN Operator...

–to select Project city equal to “Jimma” or “Adama”

–**SELECT * FROM project**
WHERE City IN ('Jimma', 'Adama');

BETWEEN Operator

- The BETWEEN operator selects a **range of data** between two values.
- The values can be numbers, text, or dates
- **Syntax**
 - **SELECT** column_name(s) **FROM** table_name
WHERE column_name **BETWEEN** value1 **AND** value2;
- **Select Age From Employee Where Age Between 23 And 30;**

29 BETWEEN Operator...

- The **BETWEEN** operator is treated differently in different databases.
- In some databases
 - selects fields that are between and excluding the test values. **OR**
 - selects fields that are between and including the test values. **OR**
 - selects fields between the test values, including the first test value and excluding the last test value. **OR**
- **Q: Check how Ms Access & SQL server treats the BETWEEN operator?**

ORDER BY clause

- is used to **sort** the result-set by a specified column. (ascending order by default).
- If you want to sort the records in a descending order, you can use the **DESC** keyword.
- **Syntax**
 - **SELECT** column_name(s)
FROM table_name
ORDER BY column_name(s) **ASC** | **DESC**

31 ORDER BY clause ..

- To order the project table in descending order using Project id.
- `SELECT * FROM employee
ORDER BY eid DESC ;`
- `SELECT * FROM employee
ORDER BY eid ASC ;`

Advanced DB Lab Portion.

❖ By the end of this practical session the student will be able to:

- ✓ Understand the basic concept of access control especially : discretionary access control based on granting and revoking privilege.
- ✓ Know how to implement discretionary access control.
- ✓ Know how to create user
- ✓ How to grant certain privilege to the user and how revoke the privilege as well.

END OF COURSE



THANK YOU !!!