8.3 SQL (DML & DDL)

Objective:

- Show understanding that DBMS carries out all creation/modification of database structure using its Data Definition Language.
- Show understanding that DBMS carries out all queries and maintenance of data using DML.
- Show understanding that industry standard for both DDL and DML SQL.
- Understand given SQL (DDL) commands and be able to write simple SQL (DDL) commands using a sub-set of commands.
- Create database (CREATE DATABASE). Create table (CREATE TABLE), including creation of attributes with data types: •CHARACTER •VARCHAR(n) • BOOLEAN • INTEGER • REAL • DATE •TIME change a table definition (ALTER TABLE) add a primary key to a table (PRIMARY KEY (field)) add foreign key to table (FOREIGN KEY (field) REFERENCES Table (Field)).
- Write SQL script to query or modify data (DML) that are stored in (at most two) database tables.
- **Queries including** SELECT. FROM, WHERE, ORDER BY, GROUP BY, **INNER JOIN**, SUM, COUNT, AVG
- **Data maintenance** including. INSERT INTO, DELETE FROM, UPDATE

Structured Query Language (SQL)

- SQL is programming language provided by **DBMS** to support all operations associated with a relational database.
- SQL is used when a database package offers high-level software tools for user interaction.
- SQL is used for sorting, manipulating and retrieving data stored in relational database.

Feature of SQL

- SQL Consists of a Sequence of Commands.
- Each command is terminated by ;
- A command can occupy more than one line.
- SQL keywords are NOT case sensitive: select is the same as SELECT
- We usually use upper case for commands and lower case for table names, attribute names and datatypes.
- When a command contains a list of items these are separated by a **comma**.

SQL Data Types

Data types used for attributes in SQL are as follow.

Data types for attributes	Description
CHARACTER	Fixed length text
VARCHAR(n)	Variable length text
BOOLEAN	True or False; SQL uses the integers 1 and 0
INTEGER	Whole number
REAL	Number with decimal places
DATE	A date usually formatted as YYYY-MM-DD
TIME	A time usually formatted as HH:MM:SS

Data Definition Language (DDL)

Data Definition Language (DDL) is part of SQL. DBMS use DDL to **create**, **modify** and **remove** data structures that form relational database. These commands only **create** structure. They do not put any **data** into database. DDL statements are written as **script** that uses **Syntax** similar to computer program.

Examples of DDL

To write SQL, we can use MySQL and SQLite to get experience of writing SQL scripts.

SQL (DDL) command	Description
CREATE DATABASE	Creates a database
CREATE TABLE	Creates a table definition
ALTER TABLE	Changes the definition of a table
PRIMARY KEY	Adds a primary key to a table
FOREIGN KEY … REFERENCES …	Adds a foreign key to a table

DDL commands enable you to perform following tasks.

- Create, alter, and drop schema objects
- Grant and revoke privileges and roles
- Add comments to the data dictionary

CREATE DATABASE

To create a database, standard command 'CREATE DATABASE' is used.

CREATE DATABASE <database-name> ;

Database name should be relevant; usually it shouldn't have spaces.

DROP DATABASE

Drop database command simply removes database. **Note** that it doesn't ask you for confirmation, and once you remove a database, it is gone forever.

DROP DATABASE <database-name> ;

CREATE TABLE

Most common DDL statement is 'CREATE TABLE'. **For example:**

```
CREATE TABLE Persons (
    PersonID int NOT NULL,
    LastName varchar(45) NOT NULL ,
    FirstName varchar(45),
    DateBirth Date,
    Address varchar(255),
    City varchar(30)
    PRIMARY KEY (PersonID) );
```

Explanation: Above DDL creates table named persons. **PersonID** column is of type **int** and will

hold an integer. **LastName**, **FirstName**, **Address**, and **City** columns are of type varchar. **PersonID** and **LastName** is marked as **NOT NULL**, which means it is not allowed to have NULL values.

Example:

CREATE DATABASE BandBooking;

CREATE TABLE Band (BandName varchar(25), NumberOfMembers integer);

ALTER TABLE Band **ADD PRIMARY KEY** (BandName);

ALTER TABLE Band-Booking **ADD FOREIGN KEY** (BandName **REFERENCES** Band(BandName)

BandName varchar(25) allows up to 25 characters for the band name.

Example: Database Birds has following tables: 9618/11/M/J/23

```
BIRD_TYPE(BirdID, Name, Size)

BIRD_SEEN(SeenID, BirdID, Date, Location, PersonID)

PERSON(PersonID, FirstName, LastName, EmailAddress)
```

Part of database table **BIRD_TYPE** is shown:

BirdID	Name	Size
0123	Blackbird	Medium
0035	Jay	Large
0004	Raven	Large
0085	Robin	Small

(SQL) script to define table Bird_Type

CREATE TABLE BIRD_TYPE **(**

BirdID **CHAR(4) NOT NULL,**

Name **VARCHAR(9),**

Size **VARCHAR(6),**

PRIMARY KEY (BirdID) **);**

Note: Bird ID as CHAR or VARCHAR • Name and size as VARCHAR or CHAR

Structure of database, Tables ,Fields/attributes , Indexes, Users, Primary Key , Foreign Key. Relationships can be created by using DDL . **9608/M/J/21/P11**

Example of Alter Table Command:

ALTER command is used to modify the structure of a table.

- To change format of table e.g. add, modify or delete a field from table

Example: **ALTER TABLE** Student **ADD** Address **varchar** (25);

- Change data type of a Field in a table. Following command changes data type of field Quantity to integer.

  Example: **ALTER TABLE** Product **Modify Column** Quantity **Integer**;
- Delete a Field from a Table.

  Example: **ALTER TABLE** Stock **DROP** Quantity;
- Add Primary key to a table

  Example: **ALTER TABLE** Student **ADD PRIMARY KEY** (StudentID);
- Add Foreign key to a table. **Example:** Command is adding foreign key to Orders table and linking it with customer table.

  ALTER TABLE Order **ADD FOREIGN KEY** (Cust_ID) **REFERENCES** Customer (Cust_ID);

Example: 9618/11/M/J/22

A teacher uses a relational database, MARKS, to store data about students and their test marks. Database has following structure:

```
STUDENT(StudentID, FirstName, LastName)

TEST(TestID, Description, TotalMarks)

STUDENT_TEST(StudentID, TestID, Mark)
```

Sample data to be stored in the table STUDENT_TEST is shown below.

StudentID	TestID	Mark
12	A1	50
12	P10	100
13	A1	75

Structured Query Language (SQL) script to create table STUDENT_TEST.

```
CREATE TABLE STUDENT_TEST (
   StudentId INTEGER,
   TestID VARCHAR,
   Mark INTEGER,
   PRIMARY KEY(StudentID,TestID),
   FOREIGN KEY(TestID) REFERENCES TEST(TestID),
   FOREIGN KEY(StudentID) REFERENCES STUDENT(StudentID)
);
```

CHAR data type stands for fixed-length character. It stores a fixed amount of characters, padding the data with spaces if actual string is shorter than specified length. It is suitable for storing strings of consistent length. **VARCHAR data type** stands for variable-length character. It stores a variable amount of characters without padding with spaces. It is suitable for storing strings of varying lengths.

8.3 SQL (DML & DDL)

Data Manipulation Language (DML)

Data Manipulation Language is used when a database is first created, to populate the tables with data. It can then be used for **ongoing maintenance.** It is used to **select**, **insert, update**, or **delete data** in objects defined with DDL.

Data Manipulation Language (DML) help us to perform following task.

- **Insertion of data** into tables when database is created.
- **Modification** or removal of data in database.
- Reading of data stored in database.

DDL is used for working on relational database **structure**, whereas **DML** is used to work with data stored in relational database.

DML Maintenance Commands

SQL (DML) maintenance commands	Description
INSERT INTO	Adds new row(s) to a table
DELETE FROM	Removes row(s) from a table
UPDATE	Edits row(s) in a table

- **INSERT INTO Command:**

SQL DML commands INSERT INTO is used to insert data into tables.

Command Syntax:

INSERT INTO <table name> (field 1, field 2, … Field n**) VALUES (**value 1, value 2, ... value n **);**

e.g: INSERT INTO Band_Booking (BandName, BookingID) **VALUES** (‘Rockz’, 65231);

Note: If orders of fields and values of all fields are known then we can skip field names.

e.g: INSERT INTO Employee **VALUES** (’mrkashif42’, ’Kashif’, ’34’, 12, ’Islamabad’)

- **DELETE FROM Command:**

DELETE FROM <table_name> **WHERE** <condition>;

Example: All records are deleted from Employee table where Employee_ID is equal to mrkashif42.

e.g: **DELETE FROM** Employee **WHERE** Employee_ID = ’mrkashif42’

- **Update Command**

SQL command UPDATE is used to make changes into table data.

E.g: UPDATE <table_name> **SET** field 1 = value 1, field 2 = value 2, **Where** <condition>;

Example: In below command, value of colour field is given new value of Red for RegNo number MH09RCM in Cars table.

UPDATE Cars **SET** Colour = 'Red' **WHERE** RegNo = 'MH09RCM'

SQL DML Query Commands

SQL DML Queries always begin with SELECT keyword. These command are used to retrieve data from table(s).

SQL (DML) query command	Description
SELECT FROM	Fetches data from a database. Queries always begin with SELECT.
WHERE	Includes only rows in a query that match a given condition
ORDER BY	Sorts the results from a query by a given column either alphabetically or numerically
GROUP BY	Arranges data into groups
INNER JOIN	Combines rows from different tables if the join condition is true
SUM	Returns the sum of all the values in the column
COUNT	Counts the number of rows where the column is not NUL
AVG	Returns the average value for a column with a numeric data type

OR, AND , NOT **Boolean operator** can be used together with standard comparisons such as =, >= , <> (not equal)etc. when specifying conditions in WHERE clause.

IS NULL is used to Checks if it's a null value contained within the variable.

GROUP BY clause in SQL is used to organize identical data into groups. It is often used with aggregate functions (such as SUM, COUNT, AVG, etc.).This clause ensures that values in specified field(s) are not repeated in result set, and result is grouped based on **unique values** in those fields.

Example: If you have a table of sales data and want to find total sales for each product category, you would use GROUP BY on the "Product Category" field. Result set will show each unique product category with corresponding total sales, and GROUP BY ensures that each product category is represented only once.

8.3 SQL (DML & DDL)

ORDER BY Command is used to **sort result** set of a query in either ascending or descending order based on one or more columns.
By default, **ORDER BY** sorts in ascending order. To sort in descending order, **DESC** keyword is used.
Example: If you have a table of employees and want to retrieve a list of employees sorted by their salaries in descending order, you would use **ORDER BY "Salary" DESC**.

Example: DML Command used to query and update school database. Query will show, in alphabetical order of second name, first and second names of all students in class 7A:

```
SELECT FirstName, SecondName
FROM Student
WHERE ClassID = '7A'
ORDER BY SecondName
```

Example: Use of Count () Function
Below Table shows data from table SCHEDULE. Write SQL script to count number of people working in morning of 26/05/2020.

ScheduleID	StaffID	WorkDate	Morning	Afternoon
210520-1	BC	21/05/2020	TRUE	TRUE
210520-2	JB	21/05/2020	TRUE	FALSE
220520-1	BC	22/05/2020	FALSE	TRUE
220520-2	LK	22/05/2020	TRUE	FALSE

Ans: SELECT Count (StaffID)
FROM Schedule
WHERE WorkDate = '26/05/2020' **AND** Morning = TRUE ;

Example: Use of Sum() Function
SELECT SUM (column_name) **FROM** table_name **WHERE** condition;
e.g: SELECT SUM (CS_Test)
From Test
Where Age > 16 **AND** Std_Name = "Haider";
e.g:

```
SELECT SUM (ExamMark)
FROM STUDENTSUBJECT
```

Example: Use of AVG () Function
SELECT AVG(column_name) **FROM** table_name **WHERE** condition;

Example: Following command Display all records form given table

Select * **From** table_name;

ESQ: Part of the EMPLOYEE table is shown. 9608/11/M/J/21

EmployeeID	FirstName	LastName	Role	Language
001	Jasmine	Chen	Leader	French
002	Kenton	Archer	Leader	English
003	Michael	Roux	Cook	French
004	Conrad	Slavorski	Leader	Russian

DML statement to return first name and last name of all employees, who are leaders, and speak either French or English.

```
SELECT FirstName, LastName
FROM EMPLOYEE
WHERE Role = "Leader"
AND (Language = "French" OR Language = "English");
```

INNER JOIN command is used in SQL to combine rows from two or more tables based on a related column between them. It returns only rows that have matching values in both tables.

Example: Consider two tables: CUSTOMER and ORDERS.

Customer Table	
CustomerID	CustomerName
1	John Doe
2	Jane Smith
3	Bob Johnson
--------	--------

Order Table			
OrderId	CustomerID	Status	OrderDate
101	1	Pending	2023-01-15
102	2	Shipped	2023-02-05
103	1	Pending	2023-02-10
--------	----------	----------	----------

Method # 1

SELECT OrderID, OrderDate, CustomerName
FROM CUSTOMER, ORDERS
WHERE CUSTOMER.CustomerID = ORDERS.CustomerID
AND ORDERS.Status = 'Pending';

Method # 2

SELECT OrderID, OrderDate, CustomerName
FROM CUSTOMER **INNER JOIN** ORDERS
ON CUSTOMER.CustomerID = ORDERS.CustomerID
WHERE ORDERS.Status = 'Pending';

Expected Output		
OrderID	OrderDate	CustomerName
101	2023-01-15	John Doe
103	2023-02-10	John Doe

Example: Holiday company has several members of staff. Database has two additional tables to store data about the staff. 9618/01/SP/21

```
STAFF(StaffID, FirstName, SecondName, DateOfBirth, Role, Salary)

SCHEDULE(ScheduleID, StaffID, WorkDate, Morning, Afternoon)
```

Following table shows some sample data from the table SCHEDULE.

ScheduleID	StaffID	WorkDate	Morning	Afternoon
210520-1	BC	21/05/2020	TRUE	TRUE
210520-2	JB	21/05/2020	TRUE	FALSE
220520-1	BC	22/05/2020	FALSE	TRUE
220520-2	LK	22/05/2020	TRUE	FALSE

SQL script to display first name and second name of all staff member working on 22/05/2020.

```
SELECT STAFF.FirstName, STAFF.SecondName
 FROM STAFF, SCHEDULE
WHERE SCHEDULE.WorkDate = '22/05/2020' AND SCHEDULE.StaffID = STAFF.StaffID;
```

Delete Example: These statements will delete the specified row(s) from the Student table (take care: DELETE FROM Student will delete the whole table!):

```
DELETE FROM Student
WHERE StudentID = 'S1301'
```

****