Data Definition Language

LEARNING OUTCOMES

At the end of this session, the student should be able to:

1. Create tables in MariaDB;
2. Alter and modify existing columns in the tables;
3. Add constraints to tables and
4. Create views in MariaDB.

CONTENT

I. Data Definition Language
II. Create Statement
III. Alter Statement
    A. Modifying a Column
    B. Renaming a Table
    C. Removing a Column
    D. Removing all Rows
    E. Dropping Tables
IV. Create and Altering a Constraint
    A. NOT NULL
    B. UNIQUE
    C. Primary key
    D. Foreign key
V. Create Views and Complex Views
    A. Modify a view
    B. Removing a view

DATA DEFINITION LANGUAGE (DDL)

This is used to create and modify structure of database objects. These database objects refer to tables, views, constraints and indexes.

CREATE Statement

The CREATE statement is used to make the schema of tables in SQL. It must follow the following rules:

- It must begin with a letter.
- It can be 1-64 characters long.
- Must contain only A-Z, a-z, 0-9, _ and $.
- Must not have duplicates on tables owned by the user.
- Must not be a MariaDB reserved word.

SOME OF THE DATA TYPES IN MARIADB

Data Type	Examples
Numeric	tinyint, boolean, smallint, mediumint, int, bigint, decimal, float, double, bit
String	string literal, char, varchar, binary, char byte, var binary, tinyblob, blob, blob and text data types, mediumblob, longblob, tinytext, text, mediumtext, longtext, json data type, enum, set

Date and Time	date, time, datetime, timestamp, year data type
Other Data types	Geometry types (point, linestring, polygon, multipoint, multilinestring, multipolygon)

General Syntax:

```
CREATE TABLE <tablename> (
Column_name data_type [Default|constraint|expression]
);
```

Examples:

```
CREATE TABLE project(
     project_no INT(2),
     project_name VARCHAR(14),
     duration_start DATE,
     duration_end DATE
     );
CREATE TABLE item(
     itemNo INT(2),
     name   VARCHAR(14),
     price   decimal(4,2)
     );
```

CREATING A TABLE USING A SUBQUERY

General Syntax:

```
CREATE TABLE <tablename> AS <select query>;
```

Examples:

```
CREATE TABLE dept30 AS (SELECT empno, ename, sal*20 ANNSAL, hiredate FROM emp WHERE deptno = 30);
```

ALTER Statement

The ALTER statement is used to edit the schema. It is used to add a new column or modify existing columns in the table.

General Syntax for adding a column:

```
ALTER TABLE <tablename> ADD (column_name datatype [DEFAULT|constraint|expr]);
```

Examples:

```
ALTER TABLE dept30 ADD (job VARCHAR(9));
ALTER TABLE dept_summary ADD (highSalary decimal(4,2));
```

General Syntax for modifying an existing column:

```
ALTER TABLE <tablename> CHANGE COLUMN (old_column_name new_column_name datatype [DEFAULT|constraint|expr]);
```

Examples:

```
ALTER TABLE dept30 CHANGE COLUMN `ename` `employee_name` VARCHAR(15);
ALTER TABLE dept_summary CHANGE COLUMN `dname` `deptname` VARCHAR(15);
```

Note: Modifying an existing column with data may cause the data to be lost or truncated.

RENAMING A TABLE

General Syntax for renaming a table:

```
ALTER TABLE <old_tablename> RENAME TO <new_tablename>;
```

Examples:

```
ALTER TABLE dept30 RENAME TO d30;
ALTER TABLE dept_summary RENAME to deptSummary;
```

REMOVING A COLUMN

General Syntax for removing a column in a table:

```
ALTER TABLE <tablename> DROP <column_name>;
```

Examples:

```
ALTER TABLE d30 DROP job;
ALTER TABLE deptSummary DROP highSalary;
```

REMOVE ALL ROWS

General Syntax for removing all rows:

```
TRUNCATE TABLE <table>;
```

Examples:

```
TRUNCATE TABLE d30;
TRUNCATE TABLE deptSummary;
```

DROPPING A TABLE

Using the drop command will delete all data and structure in the table. Any pending transactions are committed. All indexes in the tables are also dropped and it cannot be roll backed.

Examples:

```
DROP TABLE d30;
DROP TABLE deptSummary;
```

DATA INTEGRITY CONSTRAINTS

There are several data integrity constraints. These are the following:

1. **NOT NULL**
   Specifies that this column may not contain a NULL value.
   Example:

   ```
   CREATE TABLE manager(
   ```

```
      empno INT(4),
      ename VARCHAR(10) NOT NULL,
      hiredate DATE,
      deptno INT(2) NOT NULL
);
```

2. **UNIQUE KEY**
   Specifies a column or combination of columns whose values are unique for all rows in the table.
   Example:

```
CREATE TABLE branch(
      branchno INT(2),
      branchname  VARCHAR(14),
      loc VARCHAR(30),
      CONSTRAINT branch_branchname_uk UNIQUE(branchname)
);
```

3. **PRIMARY KEY**
   Uniquely identifies each row of the table.
   Example:

```
CREATE TABLE branchLaguna(
      branchno INT(2),
      branchname VARCHAR(14),
      loc VARCHAR(30),
      CONSTRAINT branch_branchname_uk UNIQUE(branchname),
      CONSTRAINT branch_branchno_pk PRIMARY KEY(branchno)
);
```

4. **FOREIGN KEY**
   Establishes and enforces a foreign key relationship between the column and a column of the referenced table.
   Example:

```
CREATE TABLE onLeave(
      leaveno INT(5) NOT NULL,
      branchno INT(2),
      ename VARCHAR(10) NOT NULL,
      dateapproved DATE,
      numdays INT(3),
      CONSTRAINT onleave_deptno_pk PRIMARY KEY(leaveno),
      CONSTRAINT  onleave_branchno_fk  FOREIGN  KEY(branchno)  REFERENCES
branchLaguna(branchno)
);
```

ALTERING TABLE CONSTRAINTS

General Syntax for adding constraints:

```
ALTER TABLE <table> ADD [CONSTRAINT constraint] type(COLUMN);
```

Examples:

```
ALTER TABLE branch
ADD CONSTRAINT branch_branchno_pk PRIMARY KEY(branchno);
```

DROPPING A CONSTRAINT

General Syntax for dropping constraints:

ALTER TABLE <table> DROP PRIMARY KEY | UNIQUE(column) | constraint constraint_name[CASCADE];

Examples:

```
ALTER TABLE branch
DROP PRIMARY KEY;

ALTER TABLE onleave
DROP FOREIGN KEY onleave_branchno_fk;
```

CREATING VIEWS

Views are used to present a different representation of data besides the original table. It is used to restrict database access. It can also be used to make complex queries easy and allows data independence.

General Syntax for creating views:

CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED|MERGE|TEMPTABLE}][DEFINER = {user|CURRENT_USER}] [SQL SECURITY {DEFINER | INVOKER}] VIEW <viewname> [{column_list}] AS select_statement [WITH [CASCADED | LOCAL] CHECK OPTION];

Examples:

```
CREATE VIEW empvu10 AS SELECT empno, ename, job FROM emp WHERE deptno=10;
CREATE VIEW salvu10 AS SELECT empno EMPLOYEE_NUMBER, ename NAME, sal SALARY
FROM emp WHERE deptno=30;
```

SHOWING VIEWS

Views are considered just like tables.

```
SHOW tables;
```

MODIFYING VIEWS

```
CREATE OR REPLACE VIEW empvu10(employee_number, employee_name, job_title)
AS SELECT empno, ename, job FROM emp WHERE deptno = 10;
```

CREATING A VIEWS

```
CREATE VIEW dept_sum_vu(name, min_sal, max_sal, avg_sal) AS SELECT d.dname,
min(e.sal), max(e.sal), avg(e.sal) FROM emp e, dept d WHERE e.deptno =
d.deptno group by d.dname;
```

REMOVING VIEWS

General Syntax for removing views:

DROP VIEW view;

Example:

```
DROP VIEW empvu10;
```

Try this!

1. Create a table named disease with columns disease number (integer with 4 digits), disease name (must have a character length from 1 to 25), common name (character length 1 to 15), origin (character 1 to 25)and datefound with date.
2. Alter the table disease and make disease number as the primary key.
3. Create a table named emp1030 whose contents came from the emp table with columns empno, ename, sal and comm. Get only those who have a commission and whose department is 10 and 30.
4. Alter the table disease and make the disease name unique.
5. Create a view name disease_vu with only the disease name, common name and datefound from the disease table. Get only diseases that came from Asia.
6. Remove the rows of table disease.
7. Remove the table disease and diease_vu.

REFERENCES

R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* 6th ed. Addison-Wesley, 2011.
RNC. Recario. *CMSC 127 Laboratory Manual.* 2016.
https://mariadb.com/kb/en/numeric-data-type-overview/
https://www.techopedia.com/definition/1175/data-definition-language-ddl