DDL AND DML

Engr. Laraib Siddiqui

Database Creation

The CREATE DATABASE statement is used to create a database.

Syntax

CREATE DATABASE *dbname*;

After creating database the next step is to create tables …

Create Table Command

Create table command is used to:

- Create a table
- Define attributes of the table with data types
- Define different constraints on attributes, like primary and foreign keys, check constraint, not null, default value etc.

Create Table Command

CREATE TABLE *table_name* (

column_name1 data_type (*size*),

column_name2 data_type (*size*),

column_name3 data_type (*size*),....

);

- The column_name parameters specify the names of the columns of the table.
- The data_type parameter specifies what type of data the column can hold (e.g. varchar, integer, decimal, date, etc.).
- The size parameter specifies the maximum length of the column of the table.

Example

```
CREATE TABLE Customer (
    customerID INTEGER
    name VARCHAR(30),
    street VARCHAR(30),
    postcode INTEGER
    city VARCHAR(20)
);
```

Create Table with Constraint

- Constraints are used to specify rules for the data in a table.
- If there is any violation between the constraint and the data action, the action is aborted by the constraint.
- Constraints can be specified when the table is created (inside the CREATE TABLE statement) or after the table is created (inside the ALTER TABLE statement).

Create Table with Constraint

Syntax

CREATE TABLE *table_name*(

column_name1 data_type (*size*) *constraint_name*,

column_name2 data_type (*size*) *constraint_name*,

column_name3 data_type (*size*) *constraint_name*,....

);

Constraints

In SQL, we have the following constraints:

- NOT NULL- Indicates that a column cannot store NULL value
- UNIQUE- Ensures that each row for a column must have a unique value
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
- FOREIGN KEY-Ensure the referential integrity of the data in one table to match values in another table
- CHECK-Ensures that the value in a column meets a specific condition
- DEFAULT-Specifies a default value for a column

Example

```
CREATE TABLE Customer (
    customerID INTEGER PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    street VARCHAR(30) NOT NULL,
    postcode SMALLINT CHECK (postcode > 0),
    city VARCHAR(20)
);
```

Example

```
CREATE TABLE Orders (
        OrderID int NOT NULL,
        OrderNumber int NOT NULL,
        customerID INTEGER ,
        PRIMARY KEY (OrderID),
        FOREIGN KEY (customerID) REFERENCES   Customers(customerID)
);
```

Alter

- Purpose is to make changes in the definition of a table already created through Create statement.

- Can add, drop attributes or constraints, activate or deactivate constraints.

Alter

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

```
ALTER TABLE table_name
DROP COLUMN column_name
```

To change the data type of a column in a table, use the following syntax:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

DROP

DROP TABLE command to delete complete table but it would remove complete table structure form the database and you would need to re-create this table once again if you wish you store some data.

Syntax

DROP TABLE *table_name*;

Example

DROP TABLE *Customer*;

INSERT

After a table has been created, we can use the INSERT command to add tuples

It is possible to write the INSERT INTO statement in two forms.

The first form does not specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Example

The following SQL statement will insert a new row, but only insert data in the "CustomerName", "City", and "Country" columns (and the CustomerID field will of course also be updated automatically):

Example

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

Updated Table

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	*null*	*null*	Stavanger	*null*	Norway

Insert data in multiple rows

```
INSERT INTO table_name (column_list)
VALUES (value_list_1), (value_list_2), ... (value_list_n);
```

UPDATE

- We can modify the column values in an existing row using the UPDATE command.
- You can use WHERE clause with UPDATE query to update selected rows otherwise all the rows would be affected.

Syntax

```
UPDATE table_name
SETcolumn1=value1,column2=value2,...
WHERE some_column=some_value;
```

Example

Before Update

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico

```
UPDATE Customers
SET ContactName='Alfred Schmidt', City='Hamburg'
WHERE CustomerName='Alfreds Futterkiste';
```

After Update

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Hamburg	12209	Germany

Update Warning!

Be careful when updating records. If we had omitted the WHERE clause, in the example like this:

```
UPDATE Customers
SET ContactName='Alfred Schmidt', City='Hamburg';
```

The "Customers" table would have looked like this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Hamburg	12209	Germany
2	Ana Trujillo Emparedados y helados	Alfred Schmidt	Avda. de la Constitución 2222	Hamburg	05021	Mexico
3	Antonio Moreno Taquería	Alfred Schmidt	Mataderos 2312	Hamburg	05023	Mexico
4	Around the Horn	Alfred Schmidt	120 Hanover Sq.	Hamburg	WA1 1DP	UK

DELETE

We can delete tuples using the DELETE command.

We can use WHERE clause with DELETE query to delete selected rows, otherwise all the records would be deleted.

SQL DELETE Syntax

```
DELETE FROM table_name
WHERE some_column=some_value;
```

Example

Before

Delete

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste' AND ContactName='Maria Anders';
```

After

Delete

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico

Delete all data

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM *table_name;*

or

DELETE * FROM *table_name;*