SQL Commands

By :
Dr. Rinkle Aggarwal
Associate Professor, CSED
TIET, Patiala

What is SQL?

SQL is a database language designed for the retrieval and management of data in a relational database.

Types of SQL Commands

SQL Categorizes its commands on the basis of functionalities performed by them. There are five types of SQL Commands which can be classified as:

i. Data Definition Language (DDL)
ii. Data Manipulation Language (DML)
iii. Data Control Language(DCL)
iv. Transaction Control Language(TCL)
v. Data Query Language (DQL)

Data Definition Language (DDL) helps you to define the database structure or schema. In order to perform changes on the physical structure of any table residing inside a database, DDL is used.

Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data.

DCL (Data Control Language) includes commands like GRANT and REVOKE, which are useful to give "rights & permissions.“

Transaction control language (TCL) commands deal with the transactions within the database.

Data Query Language (DQL) is used to fetch the data from the database.

SQL Commands

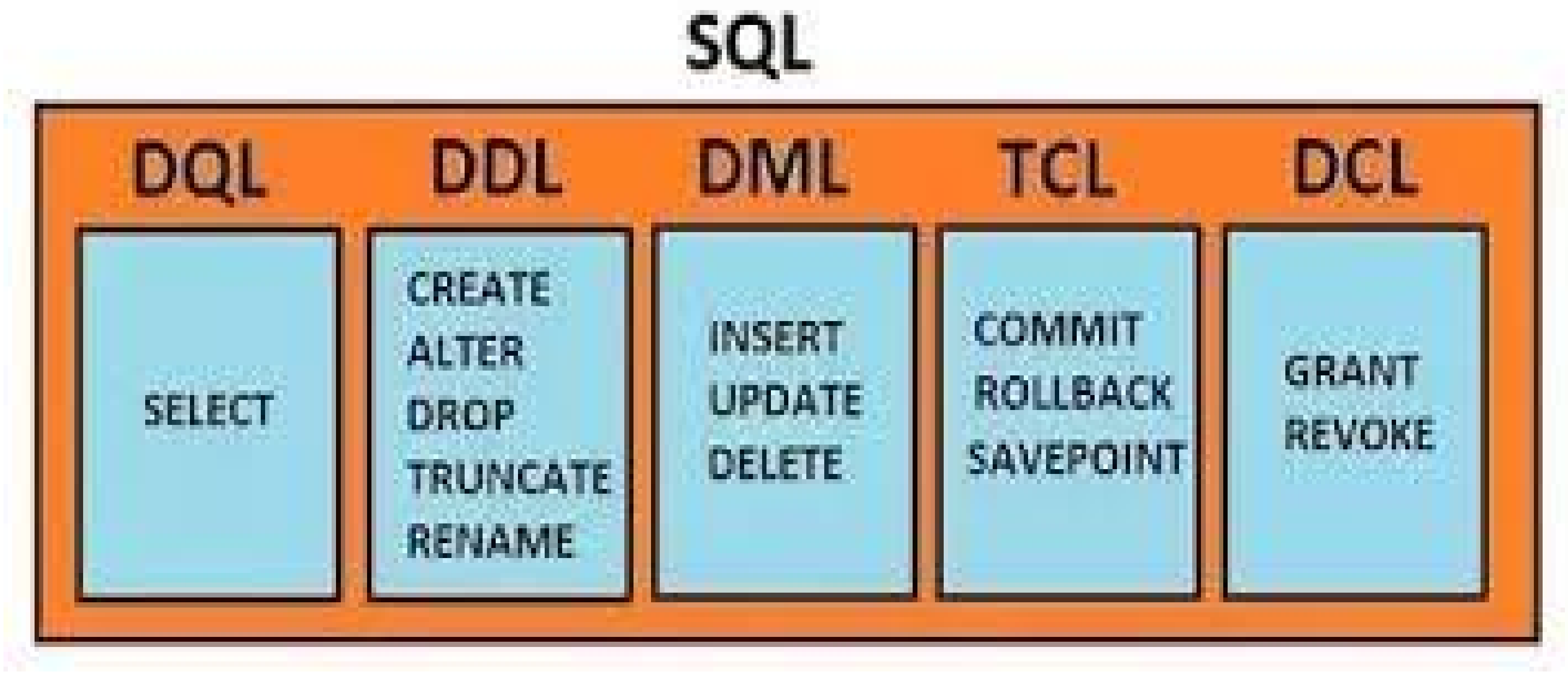

Table creation rules

- Table name and Column name can be 1 to 30 characters long. First character must be alphabetic, but name may include letters, numbers and underscores.
- Names must contain only the characters A-Z, a-z, 0-9, _(underscore),$ and # .
- Names must not be an Oracle Server reserved word.
- Names must not duplicate the names of other objects owned by the same Oracle server user.
- Table name is not case sensitive.

SYNTAX

- Create Table [schema] tablename

(

{Column datatype [DEFAULT expr] [column constraint],

column datatype[DEFAULT expr] [column constraint],…

[table_constraint]}

);

Create Table table_name(

column_name datatype(width),

column_name datatype(width),…….);

Datatype

It is the datatype of a column. Specifies the type and width of data to be stored in the column.

Default

It specifies a value to be assigned to the column if a subsequent INSERT statement omits a value for the column.

Example

- Create a table student with the following fields:

Column Name	Type	Size	Description
Name	Varchar2	20	Name of the student
Class	Varchar2	15	Class of the student
Roll_no	Number	4	Roll number of the student
Address	Varchar2	30	Address of the student

```
SQL> Create table student
(Name varchar2 (20),
Class varchar2 (15),
Roll_no number (4),
Address varchar2 (30));
```

Categories of Constraints

There are two types of constraints, these are:

- Column constraints
- Table constraints

Column Constraints

When we define constraints along the definition of the column while creating or altering the table structure, they are called as column constraints. Column constraints are applied only to the individual columns.

Table Level Constraints

Table level constraints are applied to more than one column of a table. These constraints are applied after defining all the table columns when creating or altering the structure of the table.

A table level constraint can be applied if the data constraint spans across multiple columns in a table.

Table Level Constraints

For example: In case of bank database the account number field of account holder table is not sufficient to uniquely identify the entities because two person can have joint account and hence repeating the value of account number once for each account holder, so in this case we have to apply primary key constraint on combination of account number field and name field to achieve the uniqueness. Now in this case the primary key constraint is applied on more than one column of the table so this is the table level constraint.

Types of Constraints

NULL/NOT NULL

Specifies if the column must contain value or might not contain any.

By default all columns in a table allow nulls, i.e. absence of a value in a column.

NOT NULL specifies that all rows in the table to have value for specified column.

All NOT NULL columns are mandatory fields.
NULL columns might not contain any data and can be left empty.

Syntax:
Columnname datatype (size) [constraint constraintname] NOT NULL

Example

• Create a table student with the fields name, roll_no, address and phone number.

```
SQl> Create table student
(Name varchar2 (20) CONSTRAINT NN_NAME NOT
NULL,
Roll_no number (5) NOT NULL,
Address varchar2 (40),
phone_no varchar2 (10));
```

Example

```
SQL>CREATE TABLE sale(
code char(4) NOT NULL,
description VARCHAR2(15) CONSTRAINT desc_item NOT
NULL,
quantity NUMBER(5),
sale_date DATE,
notes CHAR(15) NULL);
```

Above statement creates a table with fields code and description which is mandatory i.e. cannot be left empty and notes which is optional i.e. might not contain data.

Unique constraint

The unique key allows unique values to be entered into the column i.e. every value is a distinct value in that column. When we apply a unique constraint on a group of columns then each value of the group should be unique, they must not be repeated. A table can have more than one unique key. This constraint can be applied both at the table level and the column level.

The syntax is:
Column level syntax:
Columnname datatype (size) [constraint constraintname] UNIQUE

Unique constraint Example

• Create a table student with roll_no not null, unique name, unique address and unique phone number.

```
SQL> Create table student
(Roll_no number (5) NOT NULL,
name varchar2(20) contraint un_name unique,
Address varchar2 (40) unique,
phone_no varchar2 (10) unique);
```

Table level syntax of unique constraint

```
[Constraint constraintname] Unique (columnname [,
columnname, …….])
```

Unique constraint

- Create a table student with roll_no not null and the combination of name, address and phone number must be unique.

```
SQL> Create table student
(Roll_no number (5) NOT NULL,
Name varchar2 (20),
Address varchar2 (40),
phone_no varchar2 (10),
Constraint tb_un UNIQUE (name, address,
phone_no));
```

Example

```
SQL>CREATE TABLE item(
code CHAR(4), description CHAR(10), rate NUMBER,
UNIQUE (CODE, DESC));
```

The above statement creates a table item in which fields code and description cannot be duplicate or same for more than one record. In this example unique has been specified as table constraint.

Primary Key Constraint
A primary key is used to identify each row of the table uniquely. A primary key may be either consists of a single field or group of fields. It is a combination of both the unique key constraint as well as the not null constraint i.e. no column with the primary key constraint can be left blank and it must contain unique value in each row.

We candeclare the Primary key of a table with NOT NULL and UNIQUE constraint .SQL supports primary keys directly with the Primary Key constraint. When primary key is applied on a single field it is called as Simple Key and when applied on multiple columns it is called as Composite Primary Key.

In a table there is only one primary key.

Primary Key Constraint
Column level syntax:
Columnname datatype(size) [constraint_name]
PRIMARY KEY

- Create a table student with roll_no as the Primary Key, unique name, address cannot be NULL and phone number.

```
SQL> Create table student
(Roll_no number (5) Primary key,
Name varchar2 (20) Unique,
Address varchar2 (40) Not Null,
phone_no varchar2 (10));
```

Table level syntax of Primary Key constraint
PRIMARY KEY (columnname [, columnname , ……])

• Create a table student with name and class as the composite primary key, address and phone_no are the other fields.

```
SQL> Create table student
(Name varchar2 (20),
Class varchar2 (15),
Address varchar2 (40) Not Null,
phone_no varchar2 (10)
PRIMARY KEY (name, class));
```

Examples

```
SQL>CREATE TABLE student (
name CHAR(10) PRIMARY KEY, class NUMBER, marks
NUMBER);

SQL>CREATE TABLE emp(
empno NUMBER(4) PRIMARY KEY,
ename VARCHAR2(10),
job VARCHAR2(9),
mgr NUMBER(4),
hiredate DATE,
sal NUMBER(7,2),
comm NUMBER(7,2),
deptno NUMBER(2));
```

Check Constraint

Check constraints allow Oracle to verify the validity of data being entered on a table against a set of constants. These constants act as valid values. The Check constraint consists of the keyword CHECK followed by parenthesized conditions.
Check constraints must be specified as a logical expression that evaluates either to TRUE or FALSE. If an SQL statement causes the condition to be false an error message will be displayed.

Column level syntax:
Columnname datatype (size) [constraint constraintname] CHECK (logical expression)

Check constraint

- Create a table EMP1 with empid as primary key, phone number as unique attribute and the salary of all the employees must be greater than 5000 with name and class as the composite primary key ,address and phone_no are the other fields.

```
SQL> CREATE TABLE emp1
(empid number (10) PRIMARY KEY,
salary number (10, 3) CHECK (salary > 5000),
home_phone number (12),
CONSTRAINT uk_phone UNIQUE
(home_phone));
```

Check constraint

- Create a table emp1 with empid as primary key, phone number as unique attribute and the department of the employees must be either general or accounts, address and phone_no are the other fields.

```
SQL> CREATE TABLE emp1
(empid number (10) PRIMARY KEY,
deptname varchar2 (20) CHECK
(deptname in (‘general’,’accounts’)),
home_phone number (12) UNIQUE);
```

- Example 1

```
CREATE TABLE emp1
(empid number (10),
salary number (10, 3),
Comm. number (10, 3),
home_phone number (12),
CONSTRAINT pk_empid
PRIMARY KEY (empid),
CONSTRAINT uk_phone UNIQUE
(home_phone),
CHECK (salary > comm));
```

- Example 2

```
CREATE TABLE emp1
(empid number (10),
salary number (10, 3),
Home_phone number (12),
CONSTRAINT pk_empid
PRIMARY KEY (empid),
CONSTRAINT uk_phone UNIQUE
(home_phone),
CHECK (salary < 500000));
```

Example

CREATE TABLE emp(
code CHAR(4) PRIMARY KEY, name VARCHAR2(15) NOT NULL,
department CHAR(10) CHECK (department IN (‘mkt’,’sales’,’Acct’))
age NUMBER CHECK (age between 18 and 55),
basic NUMBER);

Using in clause with a column the valid values to be entered in a column can be specified. Using between clauses the valid range can be specified.

Default constraint

The default value constraint allows the user to insert the values in the columns where the user do not want to insert the value .The most common example of this constraint is NULL value, which is inserted in columns not defined with the NOT NULL constraint. This is not actually a constraint, it only specifies that a value should be inserted in a column if the user does not enter a value .The default values assignments are defined at the time of create table command. The datatype of the default value should match the datatype of the column.

The syntax is:
Columnname datatype (size) DEFAULT value;

```
SQL> Create table student
(roll_no number(5) primary key,
Name varchar2 (20) not null,
Marks number (3) default 100,
address varchar2 (30) not null));
```

Foreign Key constraint

A foreign key is a kind of constraint, which establishes a relationship among tables. A foreign key may be a single column or the combinations of columns, which derive their values, based on the primary key or unique key values from another table. A foreign key constraint is also known as the referential integrity constraint, as its values correspond to the actual values of the primary key of other table.

A table in which there is a foreign key, it is called as the detail table or the child table and the table from which it refers the values of the primary key, it is called as the master table or the parent table .

A foreign key must have the corresponding primary key or unique key value in the master table. Whenever you insert a value in the foreign key column, it checks the value from the primary key of the parent table.

If the value is not present, it gives an error. The parent table can be referenced in the foreign key by using the references option.

Whenever a user tries to delete a record in the master table, which is being referenced in the child table, or the record exists in the child table, then the Oracle displays an error
message.
To avoid this error message, the ON DELETE CASCADE option is used which will delete the record in the detail table when the user deletes the record in the master table.
The syntax of the foreign key at the column
level is:
Columnname datatype (size) references
tablename [(columnname)] [ON DELETE CASCADE]

Examples

Referenced table

```
SQL>CREATE TABLE item( item_code CHAR(4) PRIMARY
KEY,
description VARCHAR(20,
rate NUMBER(6,2));
```

Dependent table

```
SQL>CREATE TABLE sales(
invoice_no NUMBER(5), item_code CHAR(4) REFERENCES
item(item_code),
qty_sold NUMBER(5));
```

Item_code can be inserted in sales table only if it has
references in
item table i.e. it exists in the item table.

Referenced table

```
SQL>CREATE TABLE dept (DEPTNO NUMBER(2)
PRIMARY KEY,
DNAME VARCHAR2(14), LOC VARCHAR2(13))
```

Dependent table

```
SQL>CREATE TABLE emp(
empno NUMBER(4) PRIMARY KEY,
ename VARCHAR2(10), job VARCHAR2(9), mgr
NUMBER(4), hiredate DATE,
sal NUMBER(7,2), comm NUMBER(7,2), deptno
NUMBER(2) REFERENCES dept(deptno) on delete
set null);
```

```
SQL> Create table emp_detail
(Empno number (4) primary key,
Ename varchar2 (20) not null,
Hiredate date not null,
Deptno number (2),
dname varchar2 (10),
Salary number (6, 2)
Foreign key (deptno,dname) references
dept(deptno,dname)) ;
```

So in this table deptno and dname is the foreign key, which references the corresponding fields in the dept table.