CONSTRAINTS IN SQL

SPECIFYING CONSTRAINTS IN SQL

- SQL constraints are used to specify rules for data in a table.
- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.
- This section describes the basic constraints that can be **specified in SQL as part of table creation**.
- These include :
  - **Key constraint,**
  - **Referential integrity constraints,**
  - **Restrictions on attribute domains and NULLs, and**
  - **Constraints on individual tuples within a relation.**

SPECIFYING CONSTRAINTS IN SQL

- This section describes the basic constraints that can be **specified in SQL as part of table creation**.
- These include :
  - **Key constraint,**
  - **referential integrity constraints,**
  - **Restrictions on attribute domains and NULLs, and**
  - **Constraints on individual tuples within a relation.**

Specifying Attribute Constraints & Attribute Defaults

1) NOT NULL CONSTRAINT

- SQL allows NULLs as attribute values.
- By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such a constraint on this column (NOT NULL) specifying that NULL is now not allowed for that column.
- **This is always implicitly specified for the attributes that are part of the primary key of each relation**, but it can be specified for any other attributes whose values are required not to be NULL
- **A NULL is not the same as no data, rather, it represents unknown data**.

1) NOT NULL CONSTRAINT

- For example, the following SQL query creates a new table called CUSTOMERS and adds five columns.
- In this we specify not to accept NULLs - for NAME and AGE.

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

- If CUSTOMERS table has already been created, then to add a NOT NULL constraint to the SALARY column in Oracle and MySQL, you would write a query like the one that is shown in the following code block.

```
ALTER TABLE CUSTOMERS
   MODIFY SALARY  DECIMAL (18, 2) NOT NULL;
```

2) THE UNIQUE CONSTRAINT

- The UNIQUE Constraint prevents two records from having identical values in a column - The UNIQUE constraint ensures that all values in a column are different.
- The UNIQUE clause specifies alternate (secondary) keys.
- The UNIQUE clause can also be specified directly for a secondary key if the secondary key is a single attribute.
- **Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.**
- **A PRIMARY KEY constraint automatically has a UNIQUE constraint**.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

2) THE UNIQUE CONSTRAINT

- The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL UNIQUE,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

- If the CUSTOMERS table has already been created, then to add a UNIQUE constraint to the AGE column. You would write a statement like the query that is given in the code block below.

```
ALTER TABLE CUSTOMERS
   MODIFY AGE INT NOT NULL UNIQUE;
```

3) THE CHECK CONSTRAINT

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a column it will allow only certain values for this column.
- The CHECK Constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and isn't entered the table.
- For example, for the CUSTOMERS table, we add a CHECK with AGE column, so that you cannot have any CUSTOMER who is below 18 years.

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL CHECK (AGE >= 18),
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

3) THE CHECK CONSTRAINT

- For defining a CHECK constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

- If the CUSTOMERS table has already been created, then to add a CHECK constraint to AGE column, you would write a statement like the one given below.

```
ALTER TABLE CUSTOMERS
   MODIFY AGE INT NOT NULL CHECK (AGE >= 18 );
```

- You can also use the following syntax, which supports naming the constraint in multiple columns as well:

- You can also use the following syntax, which supports naming the constraint in multiple columns as well:

```sql
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

4) THE DEFAULT CONSTRAINT

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.
- The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.
- This constraint is used to provide a default value for the fields. That is, if at the time of entering new records in the table if the user does not specify any value for these fields then the default value will be assigned to them.

4) THE DEFAULT CONSTRAINT

- For example, the following SQL creates a new table called CUSTOMERS and adds five columns. Here, the SALARY column is set to 5000.00 by default, so in case the INSERT INTO statement does not provide a value for this column, then by default this column would be set to 5000.00

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2) DEFAULT 5000.00,
   PRIMARY KEY (ID)
);
```

- If the CUSTOMERS table has already been created, then to add a DEFAULT constraint to the SALARY column, you would write a query like the one which is shown in the code block below.

```
ALTER TABLE CUSTOMERS
MODIFY SALARY  DECIMAL (18, 2) DEFAULT 5000.00;
```

5) THE PRIMARY KEY CONSTRAINT

- A primary key is a field/attribute in a table which uniquely identifies each row/record in a database table.
  - The PRIMARY KEY constraint uniquely identifies each record in a table.
- **Primary keys must contain UNIQUE values, and cannot contain NULL values.**
- A table can have only ONE primary key.
- **The primary key, may consist of single or multiple attributes**.
  - When multiple fields are used as a primary key, they are called **a composite key.**

5) THE PRIMARY KEY CONSTRAINT

- The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

- To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

5) THE PRIMARY KEY CONSTRAINT

- To create a PRIMARY KEY constraint on the "ID" column when the CUSTOMERS table already exists, use the following SQL syntax:

```
ALTER TABLE CUSTOMER ADD PRIMARY KEY (ID);
```

```
ALTER TABLE CUSTOMERS
   ADD CONSTRAINT PK_CUSTID PRIMARY KEY (ID, NAME);
```

- **NOTE :** If you use the ALTER TABLE statement to add a primary key, the primary key column(s) should have already been declared to not contain NULL values (when the table was first created).

Delete Primary Key

- You can clear the primary key constraints from the table with the syntax given below.

```
ALTER TABLE CUSTOMERS DROP PRIMARY KEY ;
```

6) THE FOREIGN KEY CONSTRAINT

- A FOREIGN KEY is a key **used to link two tables together**. This is sometimes also called as a **referencing key**.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the **child table**, and the table with the primary key is called the referenced or **parent table**.
- A Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table.
- The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

6) THE FOREIGN KEY CONSTRAINT:

CUSTOMERS table

```
CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

ORDERS table

```
CREATE TABLE ORDERS (
   ID          INT         NOT NULL,
   DATE        DATETIME,
   CUSTOMER_ID INT references CUSTOMERS(ID),
   AMOUNT     double,
   PRIMARY KEY (ID)
);
```

6) THE FOREIGN KEY CONSTRAINT:

- If the ORDERS table has already been created and the foreign key has not yet been set, the use the syntax for specifying a foreign key by altering a table.

```
ALTER TABLE ORDERS
   ADD FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (ID);
```

DROP a FOREIGN KEY Constraint

- To drop a FOREIGN KEY constraint, use the following SQL syntax.

```
ALTER TABLE ORDERS
   DROP FOREIGN KEY;
```

SPECIFYING CONSTRAINTS IN SQL

- This section described the basic constraints that can be **specified in SQL as part of table creation**.
- These include :
  - **Key constraint,**
  - **referential integrity constraints,**
  - **Restrictions on attribute domains and NULLs, and**
  - **Constraints on individual tuples within a relation.**