POSTGRESQL - CONSTRAINTS

Constraints are the rules enforced on data columns on table. These are used to prevent invalid data from being entered into the database. This ensures the accuracy and reliability of the data in the database.

Constraints could be column level or table level. Column level constraints are applied only to one column where as table level constraints are applied to the whole table. Defining a data type for a column is a constraint in itself. For example, a column of type DATE constrains the column to valid dates.

Following are commonly used constraints available in PostgreSQL.

- **NOT NULL Constraint**: Ensures that a column cannot have NULL value.
- **UNIQUE Constraint**: Ensures that all values in a column are different.
- **PRIMARY Key**: Uniquely identifies each row/record in a database table.
- **FOREIGN Key**: Constrains data based on columns in other tables.
- **CHECK Constraint**: The CHECK constraint ensures that all values in a column satisfy certain conditions.
- **EXCLUSION Constraint**: The EXCLUDE constraint ensures that if any two rows are compared on the specified column*s* or expression*s* using the specified operator*s*, not all of these comparisons will return TRUE.

NOT NULL Constraint

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 constraint on this column specifying that NULL is now not allowed for that column. A NOT NULL constraint is always written as a column constraint.

A NULL is not the same as no data, rather, it represents unknown data.

Example:

For example, the following PostgreSQL statement creates a new table called COMPANY1 and adds five columns, three of which, ID and NAME and AGE, specify not to accept NULL values:

```
CREATE TABLE COMPANY1(
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT    NOT NULL,
   AGE            INT     NOT NULL,
   ADDRESS        CHAR(50),
   SALARY         REAL
);
```

UNIQUE Constraint

The UNIQUE Constraint prevents two records from having identical values in a particular column. In the COMPANY table, for example, you might want to prevent two or more people from having identical age.

Example:

For example, the following PostgreSQL statement creates a new table called COMPANY3 and adds five columns. Here, AGE column is set to UNIQUE, so that you can not have two records with same age:

```
CREATE TABLE COMPANY3(
```

```
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT    NOT NULL,
   AGE            INT     NOT NULL UNIQUE,
   ADDRESS        CHAR(50),
   SALARY         REAL    DEFAULT 50000.00
);
```

PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table. There can be more UNIQUE columns, but only one primary key in a table. Primary keys are important when designing the database tables. Primary keys are unique ids.

We use them to refer to table rows. Primary keys become foreign keys in other tables, when creating relations among tables. Due to a 'longstanding coding oversight', primary keys can be NULL in SQLite. This is not the case with other databases

A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.

A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a **composite key**.

If a table has a primary key defined on any field*s*, then you can not have two records having the same value of that field*s*.

Example:

You already have seen various examples above where we have created COMAPNY4 table with ID as primary key:

```
CREATE TABLE COMPANY4(
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT    NOT NULL,
   AGE            INT     NOT NULL,
   ADDRESS        CHAR(50),
   SALARY         REAL
);
```

FOREIGN KEY Constraint

A foreign key constraint specifies that the values in a column *oragroupofcolumns* must match the values appearing in some row of another table. We say this maintains the referential integrity between two related tables. They are called foreign keys because the constraints are foreign; that is, outside the table. Foreign keys are sometimes called a referencing key.

Example

For example, the following PostgreSQL statement creates a new table called COMPANY5 and adds five columns.

```
CREATE TABLE COMPANY6(
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT    NOT NULL,
   AGE            INT     NOT NULL,
   ADDRESS        CHAR(50),
   SALARY         REAL
);
```

For example, the following PostgreSQL statement creates a new table called DEPARTMENT1, which adds three columns. The column EMP_ID is the foreign key and references the ID field of the table COMPANY6.

```
CREATE TABLE DEPARTMENT1(
   ID INT PRIMARY KEY      NOT NULL,
```

```
   DEPT           CHAR(50) NOT NULL,
   EMP_ID         INT      references COMPANY6(ID)
);
```

CHECK Constraint

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 into the table.

Example:

For example, the following PostgreSQL statement creates a new table called COMPANY5 and adds five columns. Here, we add a CHECK with SALARY column, so that you can not have any SALARY Zero:

```
CREATE TABLE COMPANY5(
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT    NOT NULL,
   AGE            INT     NOT NULL,
   ADDRESS        CHAR(50),
   SALARY         REAL    CHECK(SALARY > 0)
);
```

EXCLUSION Constraint

Exclusion constraints ensure that if any two rows are compared on the specified columns or expressions using the specified operators, at least one of these operator comparisons will return false or null.

Example

For example, the following PostgreSQL statement creates a new table called COMPANY7 and adds five columns. Here, we add an EXCLUDE constraint:

```
CREATE TABLE COMPANY7(
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT ,
   AGE            INT   ,
   ADDRESS        CHAR(50),
   SALARY         REAL,
   EXCLUDE USING gist
   (NAME WITH =,
   AGE WITH <>)
);
```

Here, *USING gist* is the type of index to build and use for enforcement.

> *You need to execute the command CREATE EXTENSION btree_gist;, once per database. This will install the btree_gist extension, which defines the exclusion constraints on plain scalar data types.*

As we have enforced the age has to be same, let's see this by inserting records to the table:

```
 INSERT INTO COMPANY7 VALUES(1, 'Paul', 32, 'California', 20000.00 );
 INSERT INTO COMPANY7 VALUES(2, 'Paul', 32, 'Texas', 20000.00 );
 INSERT INTO COMPANY7 VALUES(3, 'Allen', 42, 'California', 20000.00 );
```

For the first two INSERT statements the records are added to the COMPANY7 table. For the third INSERT statement the following error is displayed:

```
ERROR:  duplicate key value violates unique constraint "company7_pkey"
DETAIL:  Key (id)=(3) already exists.
```

Dropping Constraints:

To remove a constraint you need to know its name. If the name is known, it's easy to drop. Else you need to find out the system generated name. The psql command \d tablename can be helpful here. The general syntax is:

```
ALTER TABLE table_name DROP CONSTRAINT some_name;
```