

# SQLITE - CONSTRAINTS

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. 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.

Following are commonly used constraints available in SQLite.

- **NOT NULL Constraint:** Ensures that a column cannot have NULL value.
- **DEFAULT Constraint :** Provides a default value for a column when none is specified.
- **UNIQUE Constraint:** Ensures that all values in a column are different.
- **PRIMARY Key:** Uniquely identified each rows/records in a database table.
- **CHECK Constraint:** The CHECK constraint ensures that all values in a column satisfy certain conditions.

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

### Example:

For example, the following SQLite statement creates a new table called COMPANY and adds five columns, three of which, ID and NAME and AGE, specify not to accept NULLs:

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

## DEFAULT Constraint

The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.

### Example:

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

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

## 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 SQLite statement creates a new table called COMPANY and adds five columns. Here, AGE column is set to UNIQUE, so that you can not have two records with same age:

```
CREATE TABLE COMPANY(  
  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 the each rows/records 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 fields, then you can not have two records having the same value of that fields.

### Example:

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

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

## 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 SQLite creates a new table called COMPANY and adds five columns. Here, we add a CHECK with SALARY column, so that you can not have any SALARY Zero:

```
CREATE TABLE COMPANY3(  
  ID INT PRIMARY KEY      NOT NULL,  
  NAME                    TEXT   NOT NULL,  
  AGE                     INT    NOT NULL,  
  ADDRESS                  CHAR(50),  
  SALARY                   REAL  
);
```

```
SALARY REAL CHECK(SALARY > 0)
);
```

## Dropping Constraints:

SQLite supports a limited subset of ALTER TABLE. The ALTER TABLE command in SQLite allows the user to rename a table or to add a new column to an existing table. It is not possible to rename a column, remove a column, or add or remove constraints from a table.

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js