# DB2 - CONSTRAINTS

This chapter describes various constraints in the database.

## Introduction

To enforce database integrity, a set of rules is defined, called constraints. The constraints either permit or prohibit the values in the columns.

In a Real time database activities, the data should be added with certain restrictions. For example, in a sales database, sales-id or transaction-id should be unique. The constraints types are:

- NOT NULL
- Unique
- Primary key
- Foreign Key
- Check
- Informational

Constraints are only associated with tables. They are applied to only particular tables. They are defined and applied to the table at the time of table creation.

## Explanation of each constraint:

## NOT NULL

It is a rule to prohibit null values from one or more columns within the table.

**Syntax:**

```
db2 create table <table_name>(col_name col_type not null,..)
```

**Example**: [To create a sales table, with four columns *id, itemname, qty, price* in this adding “not null” constraints to all columns to avoid forming any null cell in the table.]

```
db2 create table shopper.sales(id bigint not null, itemname
varchar(40) not null, qty int not null,price double not null)
```

## Inserting NOT NULL values into table

You can insert values in the table as shown below:

**Example:** [ERRORoneous Query]

```
db2 insert into shopper.sales(id,itemname,qty)
values(1,'raagi',12)
```

**Output:** [Correct query]

```
DB21034E  The command was processed as an SQL statement because
it was not a

valid Command Line Processor command.  During SQL processing
it returned:

SQL0407N  Assignment of a NULL value to a NOT NULL column
"TBSPACEID=5,
TABLEID=4, COLNO=3" is not allowed.  SQLSTATE=23502
```

**Example:** [Correct query]

```
db2 insert into shopper.sales(id,itemname,qty,price)
values(1,'raagi',12, 120.00)

db2 insert into shopper.sales(id,itemname,qty,price)
values(1,'raagi',12, 120.00)
```

**Output:**

```
DB20000I The SQL command completed successfully.
```

## Unique constraints

Using these constraints, you can set values of columns uniquely. For this, the unique constraints are declared with “not null” constraint at the time of creating table.

**Syntax:**

```
db2 create table <tab_name>(<col> <col_type> not null unique, ...)
```

**Example:**

```
db2 create table shopper.sales1(id bigint not null unique,
itemname varchar(40) not null, qty int not null,price
double not null)
```

## Inserting the values into table

**Example:** To insert four different rows with unique ids as 1, 2, 3 and 4.

```
db2 insert into shopper.sales1(id, itemname, qty, price)
values(1, 'sweet', 100, 89)

db2 insert into shopper.sales1(id, itemname, qty, price)
values(2, 'choco', 50, 60)

db2 insert into shopper.sales1(id, itemname, qty, price)
values(3, 'butter', 30, 40)

db2 insert into shopper.sales1(id, itemname, qty, price)
values(4, 'milk', 1000, 12)
```

**Example:** To insert a new row with “id” value 3

```
db2 insert into shopper.sales1(id, itemname, qty, price)
values(3, 'cheese', 60, 80)
```

**Output**: when you try to insert a new row with existed id value it will show this result:

```
DB21034E  The command was processed as an SQL statement
because it was not a

valid Command Line Processor command.  During
SQL processing it returned:

SQL0803N  One or more values in the INSERT statement,
UPDATE statement, or foreign key update caused by a
DELETE statement are not valid because the primary key,
unique constraint or unique index identified by "1" constrains
table "SHOPPER.SALES1" from having duplicate values for the
index key. SQLSTATE=23505
```

## Primary key

Similar to the unique constraints, you can use a “primary key” and a “foreign key” constraint to declare relationships between multiple tables.

**Syntax:**

```
db2 create table <tab_name>( ,.., primary
key ())
```

**Example**: To create ‘salesboys’ table with “sid” as a primary key

```
db2 create table shopper.salesboys(sid int not null, name
varchar(40) not null, salary double not null, constraint
pk_boy_id primary key (sid))
```

## Foreign key

A foreign key is a set of columns in a table which are required to match at least one primary key of a row in another table. It is a referential constraint or referential integrity constraint. It is a logical rule about values in multiple columns in one or more tables. It enables required relationship between the tables.

Earlier, you created a table named “shopper.salesboys” . For this table, the primary key is “sid”. Now you are creating a new table that has sales boy’s personal details with different schema named “employee” and table named “salesboys”. In this case, “sid” is the foreign key.

**Syntax:**

```
db2 create table <tab_name>(<col> <col_type>,constraint
<const_name> foreign key (<col_name>)
                  reference <ref_table> (<ref_col>)
```

**Example**: [To create a table named ‘salesboys’ with foreign key column ‘sid’]

```
db2 create table employee.salesboys(
            sid int,
            name varchar(30) not null,
            phone int not null,
            constraint fk_boy_id
            foreign key (sid)
            references shopper.salesboys (sid)
    on delete restrict
                        )
```

**Example**: [Inserting values into primary key table “shopper.salesboys”]

```
db2 insert into shopper.salesboys values(100,'raju',20000.00),
(101,'kiran',15000.00),
(102,'radha',10000.00),
(103,'wali',20000.00),
(104,'rayan',15000.00)
```

**Example**: [Inserting values into foreign key table “employee.salesboys” [without error]]

```
db2 insert into employee.salesboys values(100,'raju',98998976),
(101,'kiran',98911176),
(102,'radha',943245176),
(103,'wali',89857330),
(104,'rayan',89851130)
```

If you entered an unknown number, which is not stored in “shopper.salesboys” table, it will show you SQL error.

**Example**: [error execution]

```
db2 insert into employee.salesboys values(105,'rayan',89851130)
```

**Output:**

```
DB21034E  The command was processed as an SQL statement because it
was not a valid Command Line Processor command.  During SQL
processing it returned: SQL0530N  The insert or update value of
the FOREIGN KEY "EMPLOYEE.SALESBOYS.FK_BOY_ID" is not equal to any
value of the parent key of the parent table.  SQLSTATE=23503
```

## Checking constraint

You need to use this constraint to add conditional restrictions for a specific column in a table.

**Syntax:**

```
db2 create table
 (
  primary key (),
  constraint  check (condition or condition)
 )
```

**Example**: [To create emp1 table with constraints values]

```
db2 create table empl
 (id            smallint not null,
  name          varchar(9),
  dept          smallint check (dept between 10 and 100),
  job           char(5)  check (job in ('sales', 'mgr', 'clerk')),
  hiredate     date,
  salary       decimal(7,2),
  comm          decimal(7,2),
  primary key (id),
  constraint yearsal check (year(hiredate) > 1986 or salary > 40500)
 )
```

## Inserting values

You can insert values into a table as shown below:

```
db2 insert into empl values (1,'lee', 15, 'mgr', '1985-01-01' ,
40000.00, 1000.00)
```

## Dropping the constraint

Let us see the syntaxes for dropping various constraints.

## Dropping UNIQUE constraint

**Syntax:**

```
db2 alter table <tab_name> drop unique <const_name>
```

## Dropping primary key

**Syntax:**

```
db2 alter table <tab_name> drop primary key
```

## Dropping check constraint

**Syntax:**

```
db2 alter table <tab_name> drop check <check_const_name>
```

## Dropping foreign key

**Syntax:**

```
db2 alter table <tab_name> drop foreigh key <foreign_key_name>
```