CHAPTER-5 Constraints

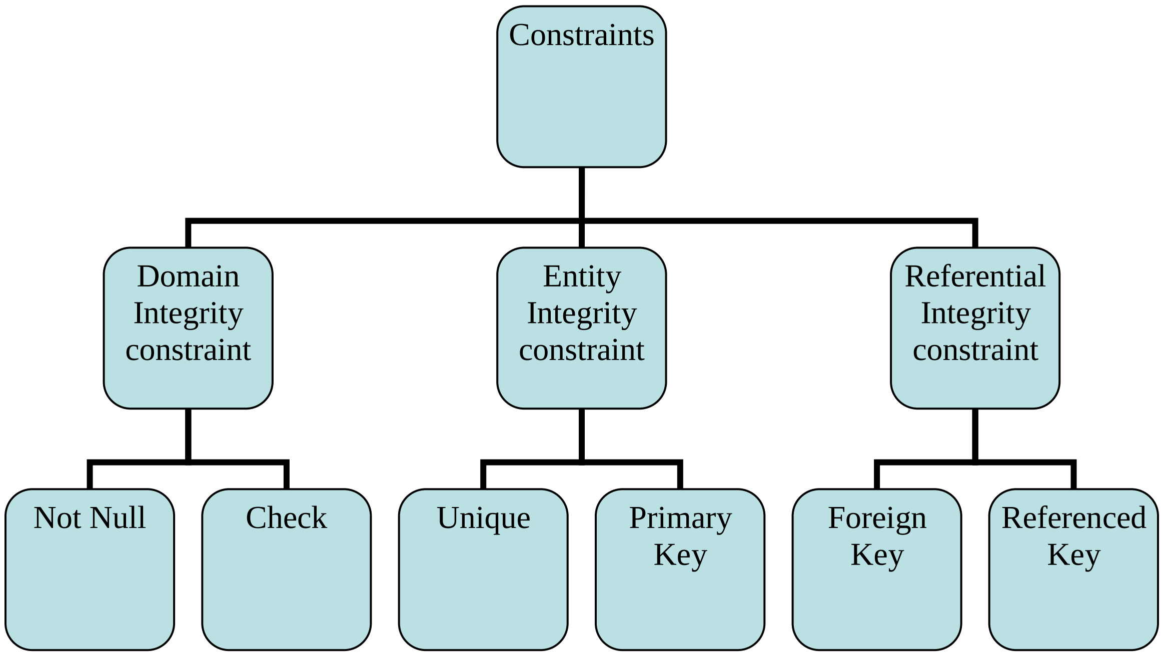
❖ **<u>DEFINATION</u>**

- **"Constraints are rules which are enforced on data that is being entered and prevents the user from entering invalid data into tables"**
- **Constraints are rules that restrict the values that may be present in database.**
- The data stored in database should be valid, correct and consistent.
- Constraints can be defined in either CREATE TABLE or ALTER TABLE commands.
- Constraints can be defined at two levels:

  i. **At column level:** when data constraints are defined along with the column definition while creating or altering table, they are known as column level constraints.

  ii. **At table level:** when data constraints are defined after defining all the columns of the table while creating or altering table, they are known as table level constraints.

- The different types of constraints are:


Constraints
Domain Integrity constraint
Entity Integrity constraint
Referential Integrity constraint
Not Null
Check
Unique
Primary Key
Foreign Key
Referenced Key

5.1 Domain Integrity constraint

Domain integrity constraints ensure that the values inserted in a particular column falls within its defined domain.

❖ NOT NULL

- When a column is defined as NOT NULL, it means that a value must be entered into the column if the record is to be accepted for storage in table
- When column is defined as not null, it becomes compulsory to enter value in that column.

- Syntax:

```
CREATE TABLE table_name
(
Columnname1 datatype(size) NOT NULL,
Columnname2 datatype(size) NOT NULL
.
.
);
```

- Example:

```
CREATE TABLE client_master
(       client_no       varchar2(6)     NOT NULL,
        Name            varchar2(20)    NOT NULL,
        Address         varchar2(30)    NOT NULL,
        City            varchar2(15),
        State           varchar2(15),
        Pincode         number(6)
);
```

- In the above query the user has to specify values for the client_no, name, and address fields, otherwise the record will not be inserted into the table.
- NOTE: **NOT NULL constraint can be defined at column level only**.

❖ CHECK Constraint

- A check constraint is used to apply business rule validations to a table column
- CHECK constraint must be specified as logical expression.
- It can be define at column level as well as table level.

i. CHECK constraint at column level:

- Syntax:

```
CREATE TABLE table_name
(
        Columnname1 datatype(size) CHECK(logical expression),
        Columnname2 datatype(size) CHECK(logical expression),
        …….
);
```

- Example:

```
CREATE TABLE client_master
(       client_no       varchar2(6)     CHECK ( client_no like ‘C%’),
        Name            varchar2(20)    CHECK(name = upper(name)),
        Address         varchar2(30),
        City          varchar2(15)   CHECK(cityIN(‘delhi’, ’mumbai’,’chennai’)),
        State           varchar2(15),
        Pincode         number(6)
);
```

ii. CHECK constraint at table level:

- Syntax:

```
CREATE TABLE table_name
(
        Columnname1 datatype(size) ,
        Columnname2 datatype(size),
        ……
        …….
        CHECK(logical expression)
);
```

- Example:

```
CREATE TABLE client_master
(       client_no       varchar2(6),
        Name            varchar2(20),
        Address         varchar2(30),
        City            varchar2(15),
        State           varchar2(15),
        Pincode         number(6),
        CHECK ( client_no like ‘C%’),
        CHECK (name = upper(name)),
        CHECK (city IN ( ‘delhi’, ’mumbai’, ’chennai’))
);
```

- In the above queries, the following rules are applied using the check constraint :
  - Data values inserted into the client_no column must start with the capital letter ‘C’
  - Data values inserted into name column must be in upper case only
  - Data values in city column can be either ‘delhi’, ’mumbai’ or ’chennai’
  - If the check constraint evaluates to FALSE then the processing stops and error message is displayed

- **Limitations of CHECK constraint**
  - The condition must be a Boolean expression
  - The condition cannot contain subqueries

- The condition cannot include SYSDATE, UID or USER
- Never defined on view.

5.2 Entity Integrity constraint

Entity integrity is a property which ensures that each row of a table has unique and non –null primary key value.

❖ **UNIQUE constraint**

- The purpose of UNIQUE key is to ensure that information in the column is UNIQUE.
- A value entered in column defined in the UNIQUE constraint must not be repeated across the column
- A table may have many UNIQUE keys

i. UNIQUE constraint defined at the column level:

- Syntax:

```
CREATE TABLE table_name
(
        Columnname1 datatype(size) UNIQUE,
        Columnname2 datatype(size) UNIQUE,
        ……
        ……
);
```

- Example:

```
CREATE TABLE client_master
(       client_no       varchar2(6)     UNIQUE,
        Name            varchar2(20)    UNIQUE,
        Address         varchar2(30),
        City            varchar2(15),
        State           varchar2(15),
        Pincode         number(6)
);
```

ii. UNIQUE constraint defined at the table level:

- Syntax:

```
CREATE TABLE table_name
(
        Columnname1 datatype(size),
        Columnname2 datatype(size),
        ……
        ….
        UNIQUE (columnname , columnname..)
);
```

- Example:
```
CREATE TABLE client_master
(       client_no       varchar2(6),
        Name            varchar2(20),
        Address         varchar2(30),
        City            varchar2(15),
        State           varchar2(15),
        Pincode         number(6),
        UNIQUE(client_no)
);
```

- In the above queries the client_no column can take only UNIQUE values
- If a value is repeated, an error message is displayed and the record is not inserted

- **Properties of UNIQUE constraint**
  - It does not allowed duplicate values but NULL values are allowed.
  - A table can have multiple column with UNIQUE constraint
  - Maximum 32 columns can combine in composite UNIQUE key
  - Column with LONG and LONG ROW data type can not have UNIQUE constraint.

❖ PRIMARY KEY constraint

- A primary key is one or more column in a table that is used to uniquely identify each row in the table
- The column in which primary key is set cannot be left blank
- The data stored in the column must be Unique
- There are two types of primary key :
  - **Simple Primary Key** – A single column primary key is called Simple Primary Key
  - **Composite Primary Key –** A Primary Key set on multiple columns is called Composite Primary Key

i. Primary Key defined at the column level:

- Syntax :
```
CREATE TABLE table_name
(
        Columnname1 datatype(size) PRIMARY KEY,
        Columnname2 datatype(size) PRIMARY KEY,
        …..
        …..
);
```
- Example:
```
CREATE TABLE sales_order
(       Order_no        varchar2(6)     PRIMARY KEY,
        Order_date      date,
        Client_no      varchar2(6),
        Dely_addr      varchar2(25),
        Salesman_no varchar2(6),
        Dely_type      varchar2(1),
        Dely_date      date,
        order_status varchar2(10)
);
```

- In the above query, the order_no field of the sales+order table has been assigned the Primary Key
- It means that this field cannot take Null values and all the values should be unique
- This is also an example of a simple Primary Key

ii. Primary Key defined at the table level:

- Syntax :

```
CREATE TABLE table_name
(
        Columnname1 datatype(size),
        Columnname2 datatype(size),
        ….
        ….
        PRIMARY KEY (columnname, columnname,…)
);
```

- Example:

```
CREATE TABLE sales_order
(       Order_no       varchar2(6),
        Order_date     date,
        Client_no      varchar2(6),
        Dely_addr      varchar2(25),
        Salesman_no varchar2(6),
        Dely_type      varchar2(1),
        Dely_date      date,
        order_status   varchar2(10)
        PRIMARY KEY (Order_no)
);
```

- In the above query a composite Primary Key has been set on detlorder_no and product_no columns of the sales_order_details table
- This means that both these columns together uniquely identify each data value

- **Properties of PRIMARY KEY constraint**
  - It does not allow null value.
  - It allows only unique value only.
  - Primary key is not compulsory to define in table but it is recommended.
  - It is used to join multiple tables.
  - Column with LONG and LONG ROW data type cannot have **PRIMARY KEY** constraint.
  - We can combine maximum 16 columns in composite primary key.

- A table cannot have more than one primary key.

5.3 Referential Integrity constraint

- Referential Integrity constraint ensures that connected tables does not contain the contradictory data.
- Primary key and foreign key are used to join two tables.

❖ FOREIGN KEY constraint

- A Foreign key represents a relationship between tables
- A foreign key is column or a group of columns whose values are derived from the primary key of other table
- Foreign table - The table in which the foreign key is defined is called the foreign table or detail table
- Master table – The table that defines the primary key and is referenced by the foreign key is called the primary table or master table
- The master table can be referenced in the foreign key definition by using REFERENCES keyword
- If an insert or update operation is performed in the foreign table, the corresponding data value must exist in the primary table, otherwise the operation is not allowed
- If a delete operation is performed on the primary table, then the data value must first be deleted from the foreign table. Otherwise the operation is not allowed
- If data from the primary table and the foreign table is to be deleted then the ON DELETE CASCADE option should be specified

i. Foreign key constraint defined at the column level:

- Syntax:

```
CREATE TABLE table_name
(
Columnname1 datatype(size),
Columnname2 datatype(size),
Columnname3 datatype(size) REFERENCES tablename (columnname)
      [ON DELETE {NO ACTION | CASCADE |SET NULL | SET DEFAULT}]
      [ON UPDATE{NO ACTION | CASCADE |SET NULL | SET DEFAULT}],

Columnname4 datatype(size),
                         …..
                         …..
);
```

- Example:

```
CREATE TABLE Department
(       dep_no          varchar2(6)     PRIMARY KEY,
        Dep_name        varchar2(6),
        Location        varchar2(6)
);

CREATE TABLE employee
(       emplyee_no      varchar2(6) PRIMARY KEY,
        D_no    varchar2(6) REFERENCES  Department
                ON DELETE CASCADE
                ON UPDATE CASCADE ,
                        ,
        salary          number(8),
);
```

- The REFERENCES keyword points to the table sales_order
- The table sales_order has the column order_no as Primary Key

ii. **Foreign key constraint defined at the table level**

- Syntax:

```
CREATE TABLE table_name
(
Columnname1 datatype(size),
Columnname2 datatype(size),
Columnname3 datatype(size)
FOREIGN     KEY(columnname1,columnname2….)REFERENCES     tablename
[(columnname1, columnname2)]
  [ON DELETE {NO ACTION | CASCADE |SET NULL | SET DEFAULT}]
  [ON UPDATE {NO ACTION | CASCADE |SET NULL | SET DEFAULT}],

);
```

- Example:

```
CREATE TABLE employee
(       emplyee_no      varchar2(6) PRIMARY KEY,
        D_no            varchar2(6),
        salary          number(8),
        FOREIGN KEY(D_no) REFERENCES Department (dep_no)
                ON DELETE  CASCADE
```

```
                    ON UPDATE CASCADE
        );
```

- **Properties of FOREIGN KEY constraint**
  - FOREIGN KEY column and primary key column must have same data type and size.
  - FOREIGN KEY column and primary key column name may be different or same.

- **Assigning user defined names to constraints:**
  - A constraint can be given a user-defined name by preceding the constraint definition with keyword CONSTRAINT and a user-defined name
  - Syntax: CONSTRAINT constraintname constraintdefinition
  - Example

```
1. Create table client_master
(client_no varchar2(6) CONSTRAINT p_clientkey("constraintname")
        PRIMARY KEY, Name varchar2(20),
                Address varchar2(30),
                City varchar2(15),
                State varchar2(15),
                Pincode number(6));

2. create table sales_order_details
        ( detloredr_no varchar2(6) REFERENCES
                Sales_order(order_no),
                Product_no varchar2(6), qty_ordered number(8),
                Qty_disp number(8),product_rate number(8,2),
                CONSTRAINT f_orderkey ("constraintname")
                FOREIGN KEY (detlorder_no)
                REFERENCES sales_order);
```