**Constraints in DBMS**

**Conduct of presentation**

1. Broadly what is a constraint?

2. Types of constraints

3. Few examples of syntax for writing constraints

**Conclusion**

* In DBMS (Database Management Systems), constraints are imitations imposed on database tables
  + To maintain the integrity, correctness, and consistency of the data.
  + Ensures only a particular type of data be entered into the database
  + Ensures only a particular sort of operation can be performed on the data inside.
  + Verify that data is unique, and prevent insertion of erroneous data.
  + Constraints used to enforce data linkages across tables,

Constraints ensures -

* **Data Accuracy** – Constraint will stop a user from entering a negative value into a field that only accepts positive numbers.
* **Data Consistency** - Constraints are able to ensure that the primary key value in one table is followed by the foreign key value in another table.
* **Data integrity** – Constraint will stop a user from putting a null value into a field that requires a valid value.

### **Types of Constraints in DBMS**

* Domain Constraints
* Key Constraints
* Tuple Uniqueness Constraints
* Entity Integrity Constraints( at column or field level)
* Referential Integrity Constraints(data between tables - foreign key)

## Domain Constraints

* Guidelines that specify the acceptable values for a certain property or field.
* These restrictions guarantee data consistency; prevent entry of inaccurate or inconsistent data into the database.
* The following are some instances of domain restrictions in a DBMS −
* **Data type constraints** –
  + - Define the kinds of data that can be kept in a column.
    - A column created as VARCHAR can take string values, but a column specified as INTEGER can only accept integer values.
* **Length Constraints** –
  + - Specify the largest amount of data that may be put in a column.
    - Column with the definition VARCHAR(10) may only take strings that are up to 10 characters long.
* **Range constraints** –
  + - Allowed range of values for a column is specified by range restrictions.
    - Column designated as DECIMAL(5,2).
      * Will only take decimal values up to 5 digits long, including 2 decimal places.
* **Nullability constraints** –
  + - Can a column accept NULL values.
    - Column that has the NOT NULL definition cannot take NULL values.
* **Unique constraints** –
  + - Constraints that require presence of **unique values** in **a column** **or group of columns** are known as unique constraints.
    - For instance, duplicate values are not allowed in a column with the UNIQUE definition.
* **Check constraints** –
  + - Constraints for checking any column data is in accepted value range.
    - Column with the definition CHECK (age > 0) can only accept ages that are greater than zero.
* **Default constraints** –
  + - Default constraints automatically assign a value to a column in case no value is provided.
    - Column with a DEFAULT value of 0 will have 0 as its value if no other value is specified.

## Key Constraints

* Define how values in columns or a group of columns of one table related to values in other tables.
* Types of Key constraints −
  + - **Primary Key Constraint** –
      * Is an individual identifier for each record in a database.
      * Ensures - each record entry contains a single, distinct value - or a pair of values that cannot be null as its method of identification.
    - **Foreign Key Constraint** –
      * Reference to the primary key in another table is a foreign key constraint.
      * Ensures values of a column or set of columns in one table correspond to the primary key column(s) in another table.
    - **Unique Constraint** –
      * Ensures that no two values inside a column or collection of columns are the same.

## Tuple Uniqeness Contraints(TUCs)

* DBMS uses constraints called Tuple Uniqueness Constraints (TUCs)
  + Ensures every entry or tuple in a table is distinct.
  + This is helpful in cases when it is vital to avoid the duplicate entries.
  + TUCs impose uniqueness on the whole row or tuple,
    - Contrast to Entity Integrity Constraints (EICs), which enforce uniqueness on certain columns or groups of columns.
  + TUC on table would make sure that no two rows have same values in all columns.

## Purpose of each constraint summarized

|  |  |
| --- | --- |
| Constraint | Description |
| NOT NULL | values cannot be null |
| UNIQUE | values cannot match any older value |
| PRIMARY KEY | used to uniquely identify a row |
| FOREIGN KEY | references a row in another table |
| CHECK | validates condition for new value |
| DEFAULT | set default value if not passed |
|  |  |

**Examples of Syntax for writing constraints**

## NOT NULL Constraint

The NOT NULL constraint in a column means that the column cannot store NULL values.

CREATE TABLE Colleges (

college\_id INT NOT NULL,

college\_code VARCHAR(20) NOT NULL,

college\_name VARCHAR(50)

);

Here, the college\_id and the college\_code columns of the Colleges table won't allow NULL values.

## UNIQUE Constraint

The UNIQUE constraint in a column means that the column must have unique value.

CREATE TABLE Colleges (

college\_id INT NOT NULL UNIQUE,

college\_code VARCHAR(20) UNIQUE,

college\_name VARCHAR(50)

);

Here, the value of the college\_code column must be unique. Similarly, the value of college\_id must be unique and it also cannot store NULL values.

## PRIMARY KEY Constraint

The PRIMARY constraint is simply a combination of NOT NULL and UNIQUE constraints. It means that the column value is used to uniquely identify the row.

CREATE TABLE Colleges (

college\_id INT PRIMARY KEY,

college\_code VARCHAR(20) NOT NULL,

college\_name VARCHAR(50)

);

Here, the value of the college\_id column is a unique identifier for a row. Similarly, it cannot store NULL value and must be UNIQUE

## FOREIGN KEY Constraint

The FOREIGN KEY (REFERENCES in some databases) constraint in a column is used to reference a record that exists in another table.

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id int REFERENCES Customers(id)

);

Here, the value of the customer\_id column references the column ‘id’ in another table named Customers.

It means that the value of customer\_id in the Orders table must be a value from the id column of the Customers table

## CHECK Constraint

The CHECK constraint checks the condition before allowing values in a table.

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

amount int CHECK (amount >= 100)

);

Here, the value of the amount column must be **greater than or equal to 100**.

## DEFAULT Constraint

The DEFAULT constraint is used to set the default value if we try to store NULL in a column.

CREATE TABLE College (

college\_id INT PRIMARY KEY,

college\_code VARCHAR(20),

college\_country VARCHAR(20) DEFAULT 'INDIA'

);

Here, the default value of the college\_country column is INDIA.

If we try to store the NULL value in the college\_country column, its value will be INDIA.

**Conclusion**

Questions?