Key Constraints in DBMS:

In Database Management Systems (DBMS), **key constraints** are rules or conditions that ensure the integrity and uniqueness of the data in the database. They define how the keys (such as primary keys, foreign keys, and others) should behave to maintain data accuracy and prevent anomalies. Below is a list of the key constraints in DBMS along with their definitions:

1. Primary Key Constraint

- **Definition**: The primary key constraint ensures that a column (or a set of columns) in a table uniquely identifies each record. It enforces two main rules:
    - **Uniqueness**: Every value in the primary key column(s) must be unique.
    - **Not Null**: A primary key column cannot have null values, as nulls would violate the uniqueness constraint.
- **Example**: In a table `Employee`, the `Employee_ID` might be the primary key.

```
CREATE TABLE Employee (
   Employee_ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Age INT
);
```

2. Foreign Key Constraint

- **Definition**: A foreign key constraint is used to link two tables together. It ensures that the values in a column (or a set of columns) match the primary key or unique key in another table. This helps maintain referential integrity between the tables.
    - The foreign key can accept null values unless it is specified to be **NOT NULL**.
- **Example**: In a table `Orders`, the `Customer_ID` could be a foreign key referencing `Customer_ID` in the `Customer` table.

```
CREATE TABLE Orders (
   Order_ID INT PRIMARY KEY,
   Order_Date DATE,
   Customer_ID INT,
   FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);
```

3. Unique Key Constraint

- **Definition**: The unique key constraint ensures that all values in a column (or a set of columns) are unique across the table. It is similar to the primary key, but a table can

have multiple unique keys, while only one primary key is allowed. Unlike primary keys, unique keys can accept null values (but only once for each column).

- **Example**: In a table `Employee`, the `Email` could be a unique key because no two employees should have the same email.

```
CREATE TABLE Employee (
   Employee_ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Email VARCHAR(100) UNIQUE
);
```

4. Candidate Key

- **Definition**: A candidate key is a set of one or more columns that can uniquely identify each record in a table. Each table can have multiple candidate keys, but only one of them will be chosen as the **primary key**. The remaining candidate keys are known as **alternate keys**.
- **Example**: In a table `Employee`, both `Employee_ID` and `Email` could be candidate keys if both are unique for every employee.

```
CREATE TABLE Employee (
   Employee_ID INT,
   Email VARCHAR(100),
   PRIMARY KEY (Employee_ID),
   UNIQUE (Email)
);
```

5. Composite Key

- **Definition**: A composite key is a primary key or unique key that consists of two or more columns. It is used when a single column is not sufficient to uniquely identify a record.
- **Example**: In a table `Enrollment`, the combination of `Student_ID` and `Course_ID` could form a composite primary key.

```
CREATE TABLE Enrollment (
   Student_ID INT,
   Course_ID INT,
   PRIMARY KEY (Student_ID, Course_ID)
);
```

6. Superkey

- **Definition**: A superkey is any set of columns that can uniquely identify each record in a table. A superkey may contain additional attributes beyond what is strictly necessary to maintain uniqueness. Every candidate key is a superkey, but not every superkey is a candidate key.
- **Example**: In a table `Employee`, `Employee_ID` and `(Employee_ID, Email)` are both superkeys. While `Employee_ID` is minimal and can be a candidate key, `(Employee_ID, Email)` is a superkey but not a candidate key because it includes unnecessary attributes.

7. Null Key

- **Definition**: A null key is a situation where a column that is part of a key (either primary or unique) contains a null value. In general, keys are not allowed to have null values, except in the case of unique constraints where a column can accept one null value.
- **Example**: A foreign key in a `Orders` table can be null if no relationship is established (like when an order is not associated with a customer yet).

8. Referential Integrity Constraint

- **Definition**: Referential integrity is a key concept that ensures that a foreign key in one table corresponds to a primary key or unique key in another table. The referential integrity constraint ensures that there are no orphaned records (records in the child table without a matching record in the parent table).
- **Example**: In the `Orders` table, the `Customer_ID` foreign key must match an existing `Customer_ID` in the `Customer` table.

```
CREATE TABLE Orders (
   Order_ID INT PRIMARY KEY,
   Order_Date DATE,
   Customer_ID INT,
   FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);
```

9. Check Constraint

- **Definition**: The check constraint allows you to specify a condition that must be satisfied for data to be inserted or updated in a table. It is not specifically related to keys, but it is often used alongside keys to enforce data validity.
- **Example**: In a table `Employee`, the `Age` column can have a check constraint to ensure the age is greater than 18.

```
CREATE TABLE Employee (
   Employee_ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Age INT,
   CHECK (Age > 18)
);
```

Summary of Key Constraints:

- **Primary Key**: Uniquely identifies each record and cannot be null.
- **Foreign Key**: Ensures referential integrity by linking tables.
- **Unique Key**: Ensures all values are unique but can accept a single null.
- **Candidate Key**: A set of columns that could be used as a primary key.
- **Composite Key**: A key made up of multiple columns.
- **Superkey**: A set of columns that can uniquely identify records; may include unnecessary columns.
- **Null Key**: A situation where a key column contains null values (usually avoided in primary or unique keys).
- **Referential Integrity**: Ensures the correctness of relationships between tables.
- **Check Constraint**: Ensures that data entered into a column meets specific criteria or conditions.

These constraints are critical for maintaining the accuracy, integrity, and consistency of the data in a database.