In a Database Management System (DBMS), both **Primary Key** and **Unique Key** are used to uniquely identify records in a table, but they have some key differences in functionality and constraints. Let’s break down each:

1. Primary Key

A **Primary Key** is a column (or set of columns) in a table that uniquely identifies each row in that table. It ensures that no duplicate or null values exist for the primary key field.

Key Characteristics:

- **Uniqueness**: Each value in the primary key must be unique across all rows of the table. No two rows can have the same primary key value.
- **Not Null**: A primary key column cannot contain `NULL` values. Every row must have a value for the primary key.
- **Single Key**: A table can only have **one** primary key.
- **Composite Primary Key**: A primary key can consist of more than one column, known as a composite key.

Example:

In a table of employees, an "Employee ID" can be the primary key because every employee has a unique ID:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,  -- Primary Key
    FirstName  VARCHAR(50),
    LastName   VARCHAR(50)
);
```

2. Unique Key

A **Unique Key** is also used to ensure that all the values in a column (or set of columns) are unique, but with more flexibility than a primary key.

Key Characteristics:

- **Uniqueness**: Like the primary key, each value in the unique key column must be unique across all rows.
- **Nullable**: A unique key can contain `NULL` values, but only **one** `NULL` value is allowed (since `NULL` is considered a unique, non-comparable value).
- **Multiple Unique Keys**: A table can have multiple unique key constraints, meaning you can enforce uniqueness on several columns.
- **Enforces Uniqueness but Isn't a Primary Identifier**: It can help identify rows uniquely, but unlike the primary key, it isn’t the main identifier for the table.

Example:

In an employee table, an "Email" column could be a unique key, because no two employees should have the same email address, but one might not have an email (thus allowing a `NULL` value):

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,   -- Primary Key
    FirstName  VARCHAR(50),
    LastName   VARCHAR(50),
    Email      VARCHAR(100) UNIQUE  -- Unique Key
);
```

Key Differences Between Primary Key and Unique Key:

Feature	Primary Key	Unique Key
Uniqueness	Must be unique across the table	Must be unique across the table
NULL Values	Cannot contain NULL values	Can contain one NULL value
Number per Table	Only one primary key per table	Can have multiple unique keys
Purpose	Used to uniquely identify each row	Ensures column uniqueness, not necessarily the main identifier
Indexing	Automatically creates a clustered index	Usually creates a non-clustered index

When to Use:

- **Primary Key**: Use when you need to uniquely identify every record in a table and you don’t want any `NULL` values (e.g., ID numbers).
- **Unique Key**: Use when you want to enforce uniqueness for a column, but the column doesn’t need to be the primary identifier (e.g., email addresses, usernames).

Both keys are essential for maintaining **data integrity** and ensuring **uniqueness** in relational databases.