## 1. Introduction to Keys in DBMS

### What is a Key in DBMS?

A key in a database management system (DBMS) is an attribute or a set of attributes that helps to uniquely identify a tuple (row) in a table. Keys ensure data integrity and help establish relationships between tables.

### Importance of Keys

- Uniquely identify records.
- Prevent duplication of data.
- Establish relationships between tables.
- Enforce referential integrity.

### Real-life Analogy:

A key can be compared to a student ID in a university, which uniquely identifies each student even if they have the same name.

---

## 2. Types of Keys in DBMS

### 2.1. Primary Key

**Definition:** A primary key is a column or a set of columns that uniquely identifies each record in a table. It cannot contain NULL values and must contain unique values.

**Example Table:**

| Student_ID | Name | Age |
|---|---|---|
| 101 | John | 20 |
| 102 | Alice | 22 |

In the above table, `Student_ID` is the primary key.

**SQL Example:**

```
CREATE TABLE Students (
    Student_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT
);
```

---

### 2.2. Candidate Key

**Definition:** A candidate key is a set of attributes that can uniquely identify a record. A table can have multiple candidate keys, but only one can be chosen as the primary key.

**Example Table:**

| Employee_ID | Email | Phone |
|---|---|---|
| 1 | john@example.com | 1234567890 |
| 2 | alice@example.com | 9876543210 |

Both `Employee_ID` and `Email` can act as candidate keys.

**SQL Example:**

```
CREATE TABLE Employees (
    Employee_ID INT,
    Email VARCHAR(50) UNIQUE,
    Phone VARCHAR(20) UNIQUE,
    PRIMARY KEY (Employee_ID)
);
```

---

### 2.3. Super Key

**Definition:** A super key is a set of one or more attributes that can uniquely identify a record in a table. A candidate key is a minimal super key.

**Example Table:**

| Order_ID | Customer_ID | Order_Date |
|---|---|---|
| 5001 | 1001 | 2023-01-10 |
| 5002 | 1002 | 2023-01-11 |

A super key could be `Order_ID` alone or `Order_ID` + `Customer_ID`.

---

### 2.4. Alternate Key

**Definition:** An alternate key is a candidate key that was not selected as the primary key.

**Example:** In the `Employees` table, if `Email` was chosen as the primary key, `Employee_ID` becomes the alternate key.

---

### 2.5. Composite Key

**Definition:** A composite key is a combination of two or more columns that uniquely identify a row.

**Example Table:**

| Student_ID | Course_ID | Enrollment_Date |
|---|---|---|
| 101 | CSE101 | 2023-09-01 |
| 102 | CSE102 | 2023-09-02 |

**SQL Example:**

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

---

### 2.6. Foreign Key

**Definition:** A foreign key is an attribute in one table that refers to the primary key in another table to establish a relationship.

**Example Tables:**

**Customers Table:**

| Customer_ID | Name |
|---|---|
| 1001 | John |
| 1002 | Alice |

**Orders Table:**

| Order_ID | Customer_ID |
|---|---|
| 5001 | 1001 |
| 5002 | 1002 |

**SQL Example:**

```
CREATE TABLE Orders (
    Order_ID INT PRIMARY KEY,
    Customer_ID INT,
    FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID)
);
```

---

### 2.7. Surrogate Key

**Definition:** A surrogate key is an artificial key, usually auto-generated, that uniquely identifies a row.

**Example Table:**

| Product_ID | Product_Name |
|---|---|
| 1 | Laptop |
| 2 | Phone |

**SQL Example:**

```
CREATE TABLE Products (
    Product_ID INT AUTO_INCREMENT PRIMARY KEY,
    Product_Name VARCHAR(100)
);
```

---

### 2.8. Unique Key

**Definition:** A unique key ensures that all values in the column are distinct from each other.

**Example Table:**

| User_ID | Username |
|---|---|
| 1 | johndoe |
| 2 | alice123 |

**SQL Example:**

```
CREATE TABLE Users (
    User_ID INT PRIMARY KEY,
    Username VARCHAR(50) UNIQUE
);
```

---

## 3. Common Mistakes and Best Practices

- Avoid NULL values in primary keys.
- Choose the smallest possible key.
- Index foreign keys for better performance.

---

## 4. Conclusion

Understanding the types of keys in DBMS is crucial for designing efficient and well-structured databases. Proper selection and use of keys ensure data integrity and establish meaningful relationships between tables.

---

## 5. Assignment

Design a database for a Library Management System using different types of keys and write SQL queries to implement them.