08 SQL Keys

In SQL, keys are used to identify and establish relationships between different tables in a relational database.

1. Primary Key:

- A primary key uniquely identifies each record in a table.
- Example:

```SQL
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  name VARCHAR(50),
  department_id INT
);
```

2. Foreign Key:

- A foreign key is a field in a table that is a primary key in another table, establishing a link between the two tables.
- Example:

```SQL
CREATE TABLE departments (
  department_id INT PRIMARY KEY,
  name VARCHAR(50)
);

CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  name VARCHAR(50),
  department_id INT,
  FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

3. Unique Key:
   - A unique key constraint ensures that all values in the key column are unique.
   - Example:

```SQL
CREATE TABLE students (
  student_id INT PRIMARY KEY,
  email VARCHAR(50) UNIQUE,
  name VARCHAR(50)
);
```

4. Candidate Key:

- A candidate key is a column or set of columns that can uniquely identify a row in a table.
- **Multiple Candidate Keys**: A table can have more than one candidate key. For instance, in a table with employee records, both `EmployeeID` and `SocialSecurityNumber` could serve as candidate keys if each uniquely identifies an employee.
- **Primary Key**: One of the candidate keys is chosen to be the primary key of the table.
- **Examples**:
  - In a table of students, possible candidate keys might include `StudentID`, `EmailAddress`, and `PassportNumber` as each of these can uniquely identify a student.
  - In a table of books, `ISBN` (International Standard Book Number) would be a candidate key because it uniquely identifies each book.
  - Example:

```SQL
CREATE TABLE products (
product_id INT,
barcode VARCHAR(20),
name VARCHAR(50),
PRIMARY KEY (product_id),
UNIQUE (barcode)
);
```

UserID	Email	Username
1	user1@domain.com	user1
2	user2@domain.com	user2
3	user3@domain.com	user3

- `UserID` is a candidate key because it uniquely identifies each user and is minimal.
- `Email` is also a candidate key if it is guaranteed to be unique for each user.
- `Username` could be a candidate key if usernames are unique.

5. Composite Key:

- A composite key is a key that consists of multiple columns, used to uniquely identify rows in a table.
- Example:

```sql
CREATE TABLE orders (
  order_id INT,
  product_id INT,
  customer_id INT,
  PRIMARY KEY (order_id, product_id, customer_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id),
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

OrderID	ProductID	Quantity	Price
1001	2001	2	10.00
1001	2002	1	15.00
1002	2001	1	10.00

In this `OrderDetails` table:

- `OrderID` and `ProductID` together form a composite key.
- `OrderID` alone is not unique because an order can contain multiple products.
- `ProductID` alone is not unique because a product can be part of multiple orders.
- The combination of `OrderID` and `ProductID` ensures that each record is unique, as the same product cannot be listed more than once in the same order.

6. Alternate Keys

Candidate keys which could not be primary keys.

Keys in MySQL and RDBMS

In relational databases, keys are essential for identifying and establishing relationships between records in tables. Here's a detailed overview of different types of keys, including examples of how to create them in MySQL.

---

1. Primary Key

Definition: A primary key is a unique identifier for a record in a table. Each table can have only one primary key, which can consist of one or more columns.

Characteristics:

- Uniqueness: No two records can have the same primary key value.
- Not Null: Primary key columns cannot contain NULL values.

Creating a Primary Key:

```sql
CREATE TABLE Employees (
    EmployeeID INT NOT NULL,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    PRIMARY KEY (EmployeeID)
);
```

Choosing a Primary Key: The primary key should be unique for each record and should be stable (i.e., it shouldn't change). Typically, an integer column with an auto-increment property is used, but other types like UUIDs can also be used.

---

2. Unique Key

Definition: A unique key ensures that all values in a column (or a set of columns) are unique across the table. Unlike the primary key, a table can have multiple unique keys.

Characteristics:

- Uniqueness: No duplicate values are allowed.
- NULL Values: Unique keys allow NULL values unless specified otherwise.

Creating a Unique Key:

```sql
CREATE TABLE Users (
    UserID INT NOT NULL AUTO_INCREMENT,
    Username VARCHAR(50) UNIQUE,
    Email VARCHAR(100) UNIQUE,
    PRIMARY KEY (UserID)
);
```

Note: `Username` and `Email` columns must have unique values.

---

3. Composite Key

Definition: A composite key is a primary key that consists of two or more columns used together to uniquely identify a record.

Characteristics:

- Uniqueness: The combination of the columns must be unique.
- All Columns: All columns in the composite key must be part of the primary key or unique constraint.

Creating a Composite Key:

```sql
CREATE TABLE OrderDetails (
    OrderID INT,
    ProductID INT,
    Quantity INT,
    PRIMARY KEY (OrderID, ProductID)
);
```

Note: The combination of `OrderID` and `ProductID` uniquely identifies each record in the `OrderDetails` table.

---

4. Foreign Key

Definition: A foreign key is a column (or a set of columns) in one table that refers to the primary key in another table. It is used to enforce referential integrity between tables.

Characteristics:

- Referential Integrity: Ensures that the value in the foreign key column must exist in the referenced primary key column of another table.

Creating a Foreign Key:

```sql
CREATE TABLE Orders (
    OrderID INT NOT NULL AUTO_INCREMENT,
    CustomerID INT,
    OrderDate DATE,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Note: `CustomerID` in the `Orders` table must exist in the `Customers` table.

---

5. Secondary Key (Alternate Key)

Definition: A secondary key (also known as an alternate key) is a unique key that is not the primary key. It can be used to uniquely identify records but is not designated as the primary key.

Characteristics:

- Uniqueness: Ensures unique values in the column(s).
- Non-primary: It is not the primary key but still provides unique identification.

Creating a Secondary Key:

```sql
CREATE TABLE Products (
    ProductID INT NOT NULL AUTO_INCREMENT,
    ProductName VARCHAR(100),
    SKU VARCHAR(50) UNIQUE,
    PRIMARY KEY (ProductID)
);
```

Note: `SKU` acts as a secondary key with unique constraints.

---

6. Other Key Types

a. Natural Key:

A natural key is a key that has a logical relationship to the data. It is derived from the data itself.

Example:

```sql
CREATE TABLE Employees (
    EmployeeNumber VARCHAR(10) PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50)
);
```

b. Surrogate Key:

A surrogate key is a synthetic key used as the primary key. It has no business meaning and is often an auto-incremented integer.

Example:

```sql
CREATE TABLE Students (
    StudentID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    EnrollmentDate DATE
);
```

c. Alternate Key:

Alternate keys are candidate keys that are not selected as the primary key. They are still unique and can be used for indexing.

Example:

```sql
CREATE TABLE Accounts (
    AccountID INT NOT NULL AUTO_INCREMENT,
    AccountNumber VARCHAR(20) UNIQUE,
    BranchCode VARCHAR(10) UNIQUE,
    PRIMARY KEY (AccountID)
);
```

Summary

- **Primary Key**: Unique identifier for records, ensures uniqueness and not null.
- **Unique Key**: Ensures unique values in a column or set of columns, can allow NULLs.
- **Composite Key**: Primary key made up of multiple columns.
- **Foreign Key**: Links records in one table to another, enforcing referential integrity.
- **Secondary Key (Alternate Key)**: Unique key not chosen as the primary key.
- **Natural Key**: Derived from the data and has business meaning.
- **Surrogate Key**: Synthetic key with no business meaning, often auto-incremented.

Each key type plays a crucial role in maintaining the integrity and efficiency of the database schema.