**Relationships Between Records in SQL**

In relational databases, **relationships** define how data in one table is related to data in another table. They help maintain data integrity and eliminate redundancy.

**1. 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 ensures referential integrity.
* The foreign key table (child) must reference values that exist in the parent table.

**Syntax Example:**

CREATE TABLE departments (

id INT PRIMARY KEY,

dept\_name VARCHAR(50)

);

CREATE TABLE employees (

id INT PRIMARY KEY,

name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES departments(id)

);

In the above example:

* departments.id is the **primary key**.
* employees.dept\_id is the **foreign key**, referencing departments(id).

**2. Types of Relationships**

**2.1 One-to-One (1:1)**

**Definition:**  
One record in a table is related to one and only one record in another table.

**Example:**

Let's assume each employee has **exactly one** company laptop.

CREATE TABLE employees (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE laptops (

id INT PRIMARY KEY,

employee\_id INT UNIQUE,

brand VARCHAR(50),

FOREIGN KEY (employee\_id) REFERENCES employees(id)

);

Here:

* Each employee can have only one laptop.
* employee\_id is UNIQUE, ensuring a **1:1** relationship.

**2.2 One-to-Many (1:N)**

**Definition:**  
One record in a table is related to **many** records in another table.

**Example:**

A department can have **multiple employees**.

CREATE TABLE departments (

id INT PRIMARY KEY,

dept\_name VARCHAR(50)

);

CREATE TABLE employees (

id INT PRIMARY KEY,

name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES departments(id)

);

Here:

* One department (departments.id) can be referenced by many employees (employees.dept\_id).
* This is the most common relationship type.

**2.3 Many-to-Many (M:N)**

**Definition:**  
Multiple records in one table relate to **multiple records** in another table.

**Example:**

Students can enroll in multiple courses, and each course can have multiple students.

To represent this, we use a **junction (bridge) table**.

CREATE TABLE students (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE courses (

id INT PRIMARY KEY,

course\_name VARCHAR(50)

);

CREATE TABLE student\_courses (

student\_id INT,

course\_id INT,

PRIMARY KEY (student\_id, course\_id),

FOREIGN KEY (student\_id) REFERENCES students(id),

FOREIGN KEY (course\_id) REFERENCES courses(id)

);

Here:

* The student\_courses table connects students and courses using **foreign keys**.
* Each student can be linked to multiple courses and vice versa.

**Summary of Relationship Types**

| **Relationship Type** | **Description** | **Real-World Example** |
| --- | --- | --- |
| One-to-One (1:1) | One row in Table A = one row in Table B | Employee & Laptop |
| One-to-Many (1:N) | One row in Table A = many rows in Table B | Department & Employees |
| Many-to-Many (M:N) | Many rows in Table A = many in Table B | Students & Courses |