A **foreign key** is a constraint that:

* Establishes a **relationship between two tables**.
* Ensures that the value in the foreign key column must exist in the **referenced table's primary key (or unique key)**.
* Maintains **referential integrity** between parent and child tables.

**✅ Example**

**🔹 1. Parent Table: Department**

CREATE TABLE Department (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50)

);

| **dept\_id** | **dept\_name** |
| --- | --- |
| 1 | HR |
| 2 | IT |
| 3 | Finance |

**🔹 2. Child Table: Employee**

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id)

);

| **emp\_id** | **emp\_name** | **dept\_id** |
| --- | --- | --- |
| 101 | John | 1 |
| 102 | Alice | 2 |

* dept\_id in Employee is a **foreign key** that references dept\_id in Department.

**✅ Key Behavior**

* If you try to **insert an Employee with a non-existent dept\_id**, the database will **throw an error**.

INSERT INTO Employee (emp\_id, emp\_name, dept\_id)

VALUES (103, 'Mark', 5); -- ERROR because dept\_id 5 doesn't exist in Department

**✅ Bonus: ON DELETE / ON UPDATE Options**

When creating a foreign key, you can define behavior on parent table changes:

FOREIGN KEY (dept\_id)

REFERENCES Department(dept\_id)

ON DELETE CASCADE

ON UPDATE CASCADE

* **ON DELETE CASCADE**: If a department is deleted, all its employees are also deleted.
* **ON UPDATE CASCADE**: If a department's dept\_id is updated, the change reflects in the Employee table too.

ON delete set null-

When a referenced record in the **parent table** is deleted, the **foreign key column in the child table is automatically set to NULL**.

### ****Example****

#### 🔹 ****1. Parent Table: Department****

CREATE TABLE Department (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50)

);

#### 🔹 ****2. Child Table: Employee****

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id)

REFERENCES Department(dept\_id)

ON DELETE SET NULL

);

### ✅ ****Scenario****

| **Department Table** |

| **dept\_id** | **dept\_name** |
| --- | --- |
| 1 | HR |
| 2 | IT |

| **Employee Table** |

| **emp\_id** | **emp\_name** | **dept\_id** |
| --- | --- | --- |
| 101 | John | 1 |
| 102 | Alice | 2 |

### 🔍 ****Now, Delete the Department with**** dept\_id = 1****:****

DELETE FROM Department WHERE dept\_id = 1;

### ✅ ****Effect on Employee Table****

| **emp\_id** | **emp\_name** | **dept\_id** |
| --- | --- | --- |
| 101 | John | **NULL** |
| 102 | Alice | 2 |

Since the **HR department (dept\_id = 1)** was deleted, the corresponding dept\_id in **Employee** is set to NULL for John.

### ****Summary of**** ON DELETE SET NULL

* Prevents **deletion failures** when related records exist.
* Helps in keeping **child records** (like employees) but marks the relationship as **NULL** (unassigned or orphaned).
* Only works if the **foreign key column allows NULL values**.

| **Action** | **Behavior When Parent Record is Deleted** |
| --- | --- |
| **ON DELETE CASCADE** | Automatically deletes related child records. |
| **ON DELETE RESTRICT** | Prevents deletion of the parent record if child records exist. |
| **ON DELETE NO ACTION** | Similar to RESTRICT: prevents deletion if related child records exist. |

**1. ON DELETE CASCADE**

Deleting a parent row **automatically deletes** all matching child rows.

CREATE TABLE Department (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50)

);

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id)

REFERENCES Department(dept\_id)

ON DELETE CASCADE

);

**Effect:** If you delete a department:

DELETE FROM Department WHERE dept\_id = 1;

👉 All employees with dept\_id = 1 will also be deleted.

**🔹 2. ON DELETE RESTRICT**

Prevents the parent from being deleted if any child records are **still referencing it**.

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id)

REFERENCES Department(dept\_id)

ON DELETE RESTRICT

);

**Effect:** If you try:

DELETE FROM Department WHERE dept\_id = 1;

👉 **Error will occur** if there are any employees with dept\_id = 1.

**🔹 3. ON DELETE NO ACTION**

Works **like RESTRICT** — prevents deletion of the parent if child records exist. The only difference is:

* NO ACTION checks for restriction **after the statement execution**, whereas
* RESTRICT checks **immediately**.

However, in most databases like **MySQL**, **NO ACTION behaves the same as RESTRICT**.

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id)

REFERENCES Department(dept\_id)

ON DELETE NO ACTION

);

Trying to delete a department that's still referenced will throw an error.