# PRIMARY KEY / FOREIGN KEY

**1. Clé Primaire (PRIMARY KEY)**
**Définition :**

- Une clé primaire est une contrainte qui identifie de manière unique chaque ligne dans une table.
- Les caractéristiques principales :
  - Les valeurs doivent être uniques.
  - Les valeurs ne peuvent pas être NULL.

**Pourquoi utiliser une clé primaire ?**

- Garantir l'intégrité des données.
- Identifier chaque ligne de manière unique.
- Utiliser comme point de référence pour établir des relations avec d'autres tables.

**Création d'une clé primaire :**

- Lors de la création de la table :

```
CREATE TABLE Employes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL,
    salaire DECIMAL(10, 2)
);
```

  Ici, `id` est la clé primaire.
- Ajout après la création :

```
ALTER TABLE Employes
ADD PRIMARY KEY (id);
```

**2. Clé Étrangère (FOREIGN KEY)**
**Définition :**

- Une clé étrangère est une contrainte utilisée pour établir un lien entre deux tables.
- Elle fait référence à une clé primaire dans une autre table.

**Pourquoi utiliser une clé étrangère ?**

- Maintenir l'intégrité référentielle (éviter les données orphelines).
- Définir les relations entre tables (un à plusieurs, plusieurs à plusieurs, etc.).

**Création d'une clé étrangère :**

- Lors de la création de la table :

```
CREATE TABLE Departements (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL
);

CREATE TABLE Employes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL,
    salaire DECIMAL(10, 2),
    dept_id INT,
    CONSTRAINT fk_departement FOREIGN KEY (dept_id) REFERENCES Departements(id)
);
```

- Ajout après la création :

```
ALTER TABLE Employes
ADD CONSTRAINT fk_departement
FOREIGN KEY (dept_id)
REFERENCES Departements(id);
```

**3. Exemple Concret**
**Étape 1 : Création des Tables**
On crée deux tables :

1. `Departements` : Contient les informations sur les départements.
2. `Employes` : Contient les informations des employés, avec une clé étrangère vers `Departements`.

```
-- Table des départements
CREATE TABLE Departements (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL );
```

```
-- Table des employés
CREATE TABLE Employes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL,
    salaire DECIMAL(10, 2),
    dept_id INT,
    CONSTRAINT fk_departement FOREIGN KEY (dept_id) REFERENCES Departements(id)
);
```

**Étape 2 : Insertion des Données**

- On insère des données dans `Departements` d'abord (car `Employes` dépend de `Departements`).
-

```
-- Insérer des départements
INSERT INTO Departements (nom) VALUES ('Informatique'),('RH'),('Marketing');

-- Insérer des employés
INSERT INTO Employes (nom, salaire, dept_id) VALUES
('Alice', 4000, 1), -- Informatique
('Bob', 3000, 2),   -- RH
('Charlie', 2000, 3); -- Marketing
```

**Étape 3 : Requête pour Afficher les Données**

- On peut utiliser une **jointure** pour afficher les employés avec leurs départements.

```
SELECT Employes.nom AS Employe, Employes.salaire, Departements.nom AS Departement
FROM Employes
JOIN Departements ON Employes.dept_id = Departements.id;
```

**4. Gestion de l'Intégrité Référentielle**

**Que se passe-t-il si on essaie :**

1. **D'ajouter un employé dans un département inexistant ?**
   `INSERT INTO Employes (nom, salaire, dept_id) VALUES ('David', 2500, 5);`
   - **Erreur** : `Cannot add or update a child row: a foreign key constraint fails.`
     Pourquoi ? Le département avec `id = 5` n'existe pas.
2. **De supprimer un département référencé ?**
   `DELETE FROM Departements WHERE id = 1;`
   - **Erreur** : `Cannot delete or update a parent row: a foreign key constraint fails.`
     Pourquoi ? Parce que le département est utilisé par des employés.

**Solutions :**

- Utiliser des actions comme `ON DELETE` ou `ON UPDATE` pour gérer ces cas.

```
CREATE TABLE Employes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL,
    salaire DECIMAL(10, 2),
    dept_id INT,
    CONSTRAINT fk_departement FOREIGN KEY (dept_id) REFERENCES Departements(id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

- **Effet de `ON DELETE CASCADE`** : Si un département est supprimé, tous les employés associés sont aussi supprimés.