Créer ou modifier une table

Table des matières

1. Créer une table

On va utiliser ici une syntaxe nommée **CREATE TABLE**, puis lui passer la liste des champs qu'elle doit contenir, ainsi que leurs **types**.

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

Ici on voit trois colonnes, id, nom, et prix.

Chaque colonne a en plus de son nom, son **TYPE** (INT, VARCHAR(255), BOOLEAN, DATE…)

Et enfin, après avoir précisé le type, on peut préciser des paramètres supplémentaires comme l'auto-incrément, l'indexation de la clé primaire, les valeurs par défaut… L'ordre de ces paramètres n'a pas d'importance. Exemple :

```
CREATE TABLE user (
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  nom VARCHAR(50) NOT NULL DEFAULT "Michel",
  prenom VARCHAR(50) NOT NULL DEFAULT "Michel",
  actif BOOLEAN DEFAULT 1
);
```

On utilise les mots clés `PRIMARY KEY` pour définir quel est le champ qui sera clé primaire.

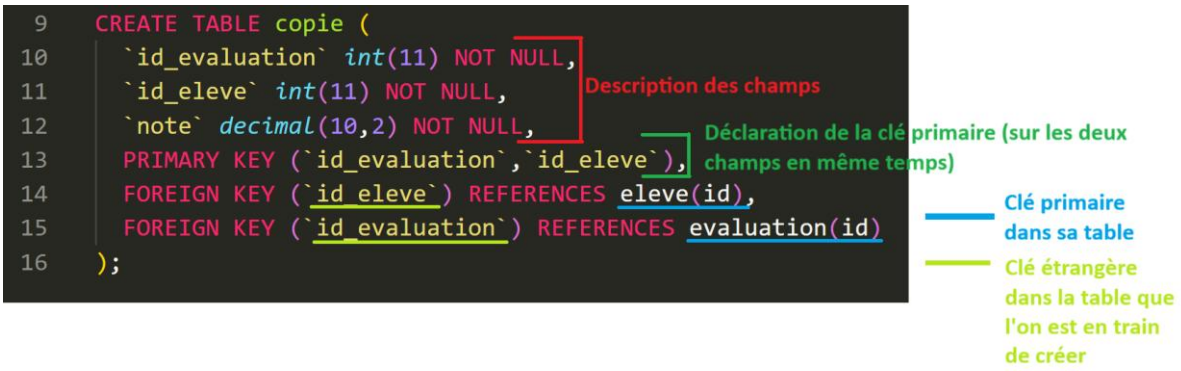
Et quand on a une clé primaire sur plusieurs champs ? Et quand on a des clés étrangères ?

Quand on a des clés étrangères dans une table, on peut se retrouver avec une clé primaire composée de plusieurs champs, et il faut que l'on déclare quelles tables vont être liées à la nôtre par le biais de ces clés étrangères.

Pour faire cela, la syntaxe champ par champ **ne suffit pas**. Il faudra **après avoir déclaré nos champs et leurs types** déclarer ces **contraintes** (On parle de contraintes, CONSTRAINT en anglais. Il s'agit de contraintes d'intégrité référentielle) séparément, comme ceci :

```
CREATE TABLE copie (
  `id_evaluation` int(11) NOT NULL,
  `id_eleve` int(11) NOT NULL,
  `note` decimal(10,2) NOT NULL,
  PRIMARY KEY (`id_evaluation`,`id_eleve`),
  FOREIGN KEY (`id_eleve`) REFERENCES eleve(id),
  FOREIGN KEY (`id_evaluation`) REFERENCES evaluation(id)
);
```

Explication de cette syntaxe en images ici :



2. Modifier une table

Pour modifier une table, on va utiliser presque la même syntaxe, mais cette fois avec le mot clé **ALTER TABLE**.

ALTER TABLE s'utilise en tandem avec trois mots-clés : **ADD, MODIFY, ou DROP**. Respectivement pour ajouter des champs, les modifier ou les supprimer.

Exemple avec un ajout de champ :

```
ALTER TABLE eleve ADD examen_reussi BOOLEAN DEFAULT 0;
```

Ici on a ajouté un seul champ, en lui précisant comme avec **CREATE TABLE** son type et ses paramètres comme la valeur par défaut.

Si on veut rajouter plusieurs champs, alors il faudra les séparer par des virgules en répétant l'instruction ADD, comme ici :

```
ALTER TABLE eleve ADD examen_reussi BOOLEAN DEFAULT 0,
ADD parfum_prefere VARCHAR(50),
ADD voiture_de_fonction BOOLEAN DEFAULT 1;
```

Et si on veut rajouter une clé étrangère ?

Facile, il suffit de lui préciser que ce que l'on ajoute est une clé étrangère, et la suite se passe comme avec la syntaxe **CREATE TABLE** :

```
ALTER TABLE eleve ADD FOREIGN KEY (id_formation) REFERENCES formation(id);
```

Et si on veut modifier un champ ?

On utilise ALTER TABLE **MODIFY**, et on lui passe son nouveau type, ses nouveaux paramètres…

```
ALTER TABLE eleve
MODIFY parfum_prefere VARCHAR(255);
```

Et si on veut supprimer un champ ?

Ici on utilise ALTER TABLE **DROP**, et on lui passe simplement le nom de la colonne à supprimer. On peut en supprimer plusieurs en répétant l'instruction plusieurs fois, séparées par des virgules.

```
ALTER TABLE eleve
DROP parfum_prefere;
```