# Introduction aux bases de données
# Cours 4 : Contraintes sur les champs, clés primaires, tris

Jean Francis Michon [1]

[1]Université Nationale d'Economie, Kharkov

25 Novembre 2021

# Les contraintes sur les champs

On veut éviter ou imposer certaines valeurs : on peut mettre des paramètres sur les champs dans l'instruction CREATE ou ALTER :

- NOT NULL - oblige une valeur non NULL
- UNIQUE - oblige unicité dans la colonne
- PRIMARY KEY - combinaison de NOT NULL et UNIQUE pour identifier une seule ligne
- FOREIGN KEY - maintenir une liaison vers une autre table
- CHECK - Vérifie que la valeur satisfait une condition
- DEFAULT -Définit une valeur par défaut si le champ n'est pas rempli
- CREATE INDEX - Génération d'un index

# Exemple de contrainte sur la table `client` - 1

La structure de notre table `client` est donnée par :

```
DESCRIBE client;
```

ou (plus complet pour voir les contraintes) par

```
SHOW CREATE TABLE client;
```

Vous obtenez un tableau dans PhpMyAdmin : nom VARCHAR(20), prenom VARCHAR(20), codepostal VARCHAR(5).
Aucune contrainte sur les champs n'est visible.

# Exemple de contrainte sur la table `client` - 2

**Nous voulons imposer que `nom` ne soit jamais NULL.**
On passe la commande :

```
ALTER TABLE client MODIFY nom VARCHAR(20) NOT NULL;
```

Essayons de faire une nouvelle insertion avec NULL pour nom.

```
INSERT INTO client (prenom, codepostal) VALUES('Лілія', '765
```

J'obtiens une erreur. Remarquez que mon fichier contenait un nom NULL mais cela n'a pas créé d'erreur au moment de la modification par ALTER.
**Question** : Comment enlever cette contrainte ? On réécrit toute la ligne :

```
ALTER TABLE client MODIFY nom VARCHAR(20);
```

# Différencier les enregistrements avec une clé primaire

Si vous désirez que **toutes les valeurs d'une colonne (ou d'une famille de colonnes) soient différentes et non NULL**, vous pouvez utiliser une **clé primaire**. Par exemple on aurait pu créer notre table client avec

```
CREATE TABLE clientcp (
nom VARCHAR(20), prenom VARCHAR(20), codepostal VARCHAR(5),
PRIMARY KEY (nom, prenom)
);
```

Alors on ne pourra plus saisir deux couples (nom, prenom) indentiques ou NULL.
**On ne peut choisir qu'une seule clé primaire par table mais plusieurs choix sont peut-être possibles.**
**Aucun des champs qui la composent ne peut être NULL.**
**Il y a toujours une clé primaire dans une table sans doublon.**

# Clés primaires possibles ?

Considérons la table à 3 attributs :

| A:int | B:int | C:int |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 2 | 9 |
| 4 | 2 | 9 |

Donnez le(s) clés “candidates” différentes qui peuvent être choisies pour être LA clé primaire.

# Colonne spéciale comme clé primaire

On ajoute souvent une colonne “artificielle” contenant un nombre entier incrémenté à chaque saisie d'un enregistrement. Par exemple nous pouvons ajouter une colonne `clientid` à notre table **clients** et créer la nouvelle table **clientscp**:

```
CREATE TABLE clientcp (
clientid INT AUTO_INCREMENT PRIMARY KEY,
nom VARCHAR(20), prenom VARCHAR(20), codepostal VARCHAR(5)
);
```

La clé primaire crée aussi un **index** sur la ou les colonnes de la clé primaire.
On peut supprimer la clé primaire avec un `DROP PRIMARY KEY` ou `DROP CONSTRAINT`.

# La contrainte UNIQUE sur un champ

Cette contrainte vous assure que toutes la valeurs de la colonne sont différentes. C'est comme pour une clé primaire , mais il n'y a qu'une clé primaire, alors que la contrainte UNIQUE peut être imposée sur autant de colonnes que l'on veut. La commande

```
ALTER TABLE client ADD UNIQUE (prenom );
```

vous permet de créer la contrainte sur un champ. Pour la supprimer

```
ALTER TABLE client DROP INDEX prenom;
```

# Commandes SELECT - page 1

Il y a de nombreuses variantes :

```
SELECT champ1, champ2,...FROM table [WHERE condition]
                [ ORDER BY ASC|DESC] [LIMIT n OFFSET m];
```

le crochet signifie “optionnel” , la barre verticale donne des choix possibles .
La condition est booléenne et s'exprime avec des opérateurs booléens

```
= , <, <=, >, >=,  !=, AND, OR, NOT
```

le terme LIMIT permet de n'afficher que n enregistrements, et OFFSET un décalage d'affichage de m.

# Commandes SELECT - page 2

`SELECT champ1, champ2,...FROM table [WHERE champ1 LIKE chaîn`

permet avec “chaîne”, en utilisant % (n’importe quelle chaîne) ou _ (n’importe quel caractère), de proposer un modèle pour la recherche.
Par exemple :

```
SELECT nom FROM client  where nom LIKE a%b ;
```

affiche tous les noms de la table client qui commencent par a et finissent par b.

# Commandes SELECT -3

La commande SELECT donne aussi des informations plus globales.
On utilise COUNT(champ) : nombre d'enregistrements,
AVG(champ): moyenne de champs numériques SUM(champ),
MAX(champ), MIN(champ) : somme, max et min dans une colonne particulière.
Les valeurs NULL sont toujours écartées.
Par exemple :

```
SELECT COUNT(*) FROM client  ;
```

donne le nombre d'enregistrements de la table client. Mais

```
SELECT COUNT(nom) FROM client  ;
```

donnera le nombre d'enregistrements avec `nom` $\neq$ NULL.

# Les tris simples

Supposons que nous voulions voir toute notre table `client` **triée sur les noms** : il suffit de faire

```
SELECT  * FROM client ORDER  BY nom ASC;
```

(remplacer ASC par DESC pour avoir l'ordre inverse). La commande suivante permet d'éviter des doublons de noms mais elle oubliera les lignes où `nom= NULL`:

```
SELECT DISTINCT nom  FROM client ORDER  BY nom  ASC;
```

Enfin vous pouvez mettre votre résultat de votre tri dans une nouvelle table :

```
CREATE TABLE client_tri_nom
        AS SELECT  *  FROM client ORDER  BY nom ASC;
```

# Calcul d'une valeur et tri sur celle-ci

Une table `sport` a pour schéma :
`equipe VARCHAR(20), nbv INT(3), nbd INT(3)`
(nombre de victoires et défaites).
On veut afficher la liste des équipes avec nbv, nbd et le ratio : $\frac{nbv}{nbv+nbd}$, ordonnée par ratio décroissant :

```
SELECT equipe, nbv, nbd ,
IF(nbd +nbv > 0, CEIL(100*nbv/(nbv + nbd)),0) AS ratio
FROM sport ORDER BY ratio DESC;
```

Il faut faire attention à la division par 0 donc utiliser un IF. Le mot ratio est introduit comme titre de colonne et clé de tri.

# Les index

Si vous voulez **accélérer vos tris** sur certains attributs vous pouvez créer un index sur ces attributs (sauf les TEXT ou BLOB). Un fichier d'index sera créé et mis à jour à chaque fois que vous modifiez la table. Cela peut être coûteux ! Vous pouvez créer autant d'index que vous voulez.
Sur la table `client`, créons un index sur l'attribut codepostal et appelons le `idxcodepostal` :

```
CREATE INDEX  idxcodepostal
ON client( codepostal );
```

On supprime l'index avec un

```
ALTER TABLE client DROP INDEX idxcodepostal;
```

**Rappel** : inutile de mettre un index sur le(s) champs de clé primaire, il y est déjà ! Vous pouvez aussi améliorer les performances de vos recherches avec la commande OPTIMIZE TABLE (défragmentation et recalcul des index).