Partie 2

Qu’est ce que CRUD ?

En termes simples, le terme CRUD résume les fonctions dont les utilisateurs ont besoin pour créer et gérer des données.

- **C**reate - Créer (ajouter, insérer)
- **R**ead - Lire (extraire, lister, consulter, interroger, rechercher)
- **U**pdate - Mettre à jour (modifier, éditer)
- **D**elete - Supprimer (effacer)

INSERT INTO

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO.

Insérer une ligne en spécifiant toutes les colonnes

```
INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)
```

Cette syntaxe possède les avantages et inconvénients suivants :

- On est obligé de remplir toutes les données, tout en respectant l'ordre des colonnes
- Il n'y a pas le nom de colonne, donc les fautes de frappe sont limitées. Par ailleurs, les colonnes peuvent être renommées sans avoir à changer la requête
- L'ordre des colonnes doit rester identique sinon certaines valeurs prennent le risque d'être complétée dans la mauvaise colonne

INSERT INTO

Insérer une ligne en spécifiant seulement les colonnes souhaitées

```
INSERT INTO table (nom_colonne_1, nom_colonne_2, …)

 VALUES ('valeur 1', 'valeur 2', ...);
```

Ici il faut indiquer le nom des colonnes avant “VALUES”.

A noter : il est possible de ne pas renseigner toutes les colonnes. De plus, l’ordre des colonnes n’est pas important.

INSERT INTO

Insertion de plusieurs lignes à la fois

```
INSERT INTO client (prenom, nom, ville, age)
 VALUES
 ('Rébecca', 'Armand', 'Saint-Didier-des-Bois', 24),
 ('Aimée', 'Hebert', 'Marigny-le-Châtel', 36),
 ('Marielle', 'Ribeiro', 'Maillères', 27),
 ('Hilaire', 'Savary', 'Conie-Molitard', 58);
```

Lorsque le champ à remplir est de type VARCHAR ou TEXT il faut indiquer le texte entre guillemet simple. En revanche, lorsque la colonne est un numérique tel que INT ou BIGINT il n'y a pas besoin d'utiliser de guillemet, il suffit juste d'indiquer le nombre.

INSERT INTO

Exemples

```
INSERT INTO `products` VALUES (21,'coussin', 58, 18.99);
```

```
INSERT INTO `products` (product_id, name, quantity_in_stock, unit_price)
    VALUES (22,'couette', 43, 35.99);
```

INSERT INTO

Exercices

Utilisez le lien suivant et ajoutez 5 clients dans la table customers

https://fr.fakenamegenerator.com/gen-random-us-fr.php

UPDATE

La commande UPDATE permet d'effectuer des modifications sur des lignes existantes. Très souvent cette commande est utilisée avec WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

Syntaxe

```
UPDATE table
SET nom_colonne_1 = 'nouvelle valeur'
WHERE condition
```

```
UPDATE table
SET colonne_1 = 'valeur 1', colonne_2 = 'valeur 2', colonne_3 = 'valeur 3'
WHERE condition
```

UPDATE

Exemples

```
UPDATE products

SET name = 'coussin douillet'

WHERE product_id = 21;
```

```
UPDATE customers

SET last_name = 'Bonjour';
```

```
SET SQL_SAFE_UPDATES = 0;
```

UPDATE

Exercices

Grâce au lien suivant assignez un nom de famille différent à chaque client https://fr.fakenamegenerator.com/gen-random-us-fr.php

Notre premier client à déménagé, veuillez changer son adresse à la suivante:

38 rue de Champs 69001 Lyon

DELETE

La commande DELETE en SQL permet de supprimer des lignes dans une table. En utilisant cette commande associé à WHERE il est possible de sélectionner les lignes concernées qui seront supprimées.

Syntaxe

```
DELETE FROM `table`

WHERE condition
```

Attention : s'il n'y a pas de condition WHERE alors **toutes** les lignes seront supprimées et la table sera alors vide.

DELETE VS TRUNCATE

Pour supprimer toutes les lignes d'une table il convient d'utiliser la commande DELETE ou TRUNCATE sans utiliser de clause conditionnelle.

Syntaxe

```
DELETE FROM `utilisateur`
```

```
TRUNCATE TABLE `utilisateur`
```

La différence majeure étant que la commande TRUNCATE va ré-initialiser l'auto-incrémente s'il y en a un. Tandis que la commande DELETE ne ré-initialise pas l'auto-incrément.

DELETE

Exemples

```
DELETE FROM products

WHERE product_id = 21;
```

DELETE

Exercices

Supprimez les 5 derniers clients.

Les contraintes

Cours

Les contraintes sont utilisées pour spécifier les règles concernant les données dans la table. Elles peuvent être appliquées à un ou plusieurs champs d'une table SQL pendant la création de la table ou après sa création à l'aide de la commande ALTER TABLE. Les contraintes sont les suivantes :

- NOT NULL - Empêche l'insertion d'une valeur NULL dans une colonne.
- CHECK - Vérifie que toutes les valeurs d'un champ satisfont à une condition.
- DEFAULT - Attribue automatiquement une valeur par défaut si aucune valeur n'a été spécifiée pour le champ.
- UNIQUE - Assure l'insertion de valeurs uniques dans le champ.
- INDEX/KEY - Permet d'indexer un champ pour accélérer la recherche d'enregistrements.
- PRIMARY KEY - Identifie de manière unique chaque enregistrement d'une table.
- FOREIGN KEY - Assure l'intégrité référentielle d'un enregistrement dans une autre table.

Les contraintes

Exemples

```
CREATE TABLE `order_statuses` (
  `order_status_id` tinyint(4) NOT NULL,
  `status` varchar(50) NOT NULL,
  PRIMARY KEY (`order_status_id`)
);
```

Les contraintes

Exercices

Y a t'il des colonnes n'ayant pas la contrainte not null ?

Quelles sont les différentes valeurs par défaut indiqués (toutes tables et colonnes confondus) ?

A quoi correspond le mot clef KEY ?

A quoi correspond le mot clef CONSTRAINT ?

CREATE TABLE

La commande CREATE TABLE permet de créer une table en SQL. La création d'une table sert à définir les colonnes et le type de données qui seront contenus dans chacun des colonne (entier, chaîne de caractères, date, valeur binaire ...).

```
CREATE TABLE nom_de_la_table
(
    colonne1 type_donnees contraintes,
    colonne2 type_donnees contraintes,
    colonne3 type_donnees contraintes,
    colonne4 type_donnees contraintes
)
```

CREATE TABLE

Exemples

```
CREATE TABLE `products` (
  `product_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `quantity_in_stock` int(11) NOT NULL,
  `unit_price` decimal(4,2) NOT NULL,
  PRIMARY KEY (`product_id`)
);
```

CREATE TABLE

Exercices

Recréez la table suivante que vous appellerez clients

customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553
5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	NY	14450
7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	NY	14215
8	Jacquline	Duncan	NULL	jacquline.duncan@yahoo.com	15 Brown St.	Jackson Heights	NY	11372
9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	NY	11050
10	Pamelia	Newman	NULL	pamelia.newman@gmail.com	476 Chestnut Ave.	Monroe	NY	10950

ALTER TABLE

La commande ALTER TABLE en SQL permet de modifier une table existante par exemple pour ajouter une colonne, supprimer une colonne ou modifier le type d'une colonne existante.

Ajouter un colonne

```
ALTER TABLE nom_table

ADD nom_colonne type_donnees
```

Supprimer une colonne

```
ALTER TABLE nom_table

DROP nom_colonne
```

ALTER TABLE

Modifier une colonne

```
ALTER TABLE nom_table

MODIFY nom_colonne type_donnees
```

Renommer une colonne

```
ALTER TABLE nom_table

CHANGE colonne_ancien_nom colonne_nouveau_nom type_donnees
```

ALTER TABLE

Exemples

```
ALTER TABLE customers
ADD country VARCHAR(255)
```

```
SET SQL_SAFE_UPDATES = 0;
UPDATE customers
SET country = 'France';
```

```
ALTER TABLE customers
CHANGE country pays VARCHAR(255);
```

```
ALTER TABLE customers
DROP pays;
```

ALTER TABLE

Exercices

Ajoutez une colonne country à votre table clients et remplissez votre colonne avec les valeurs appropriés

DROP TABLE

La commande DROP TABLE en SQL permet de supprimer définitivement une table d'une base de données. Cela supprime en même temps les éventuels index, trigger, contraintes et permissions associées à cette table.

Syntaxe

```
DROP TABLE nom_table
```

Attention : il faut utiliser cette commande avec attention car une fois supprimée, les données sont perdues. Avant de l'utiliser sur une base importante il peut être judicieux d'effectuer un backup (une sauvegarde) pour éviter les mauvaises surprises.

DROP TABLE

Exercices

Supprimez la table clients

CREATE - DROP - USE DATABASES

Créer une base de données

```
CREATE DATABASE ma_base
```

Supprimer une base de données

```
DROP DATABASE IF EXISTS ma_base
```

Utiliser une base de données

```
USE ma_base
```

CREATE - DROP - USE DATABASES

Exemples

```
DROP DATABASE IF EXISTS `sql_store`;

CREATE DATABASE `sql_store`;

USE `sql_store`;
```

CREATE - DROP - USE DATABASES

Exercices

Nous avons ouvert un nouveau magasin.

Créez une nouvelle base de données appelée new_store

Dans cette base de donnés créez une table “clients” avec les mêmes champs que la table customers de “sql_store”.

Nous avons 5 nouveaux clients dans notre nouveau magasin. Ajoutez les à la table “clients”.

Nos 5 clients les plus fidèles de notre 1 er magasin sont également allés visiter notre nouveau magasin. Ajoutez les à la table “clients”.

UNION

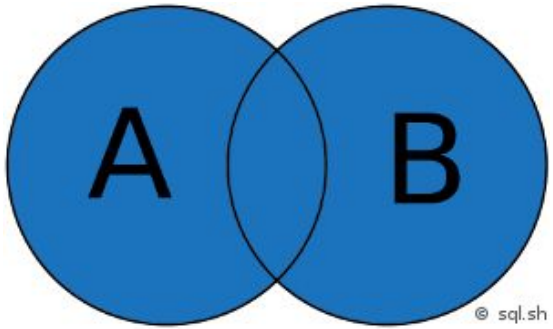
La commande UNION permet de mettre bout-à-bout les résultats de plusieurs requêtes utilisant elles-même la commande SELECT. C'est donc une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l'utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre

Syntaxe

```
SELECT * FROM table1

UNION

SELECT * FROM table2
```

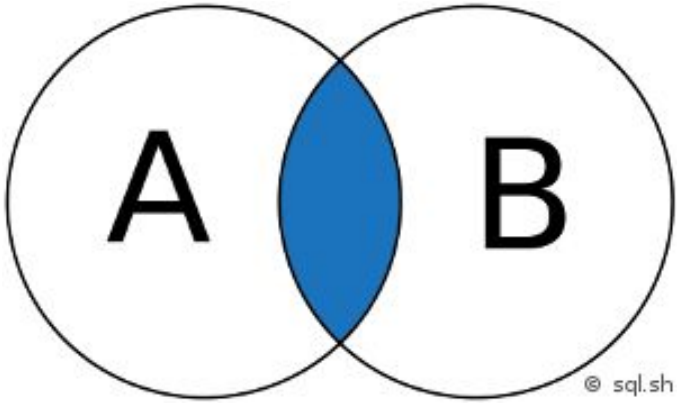


INTERSECT

MySQL ne propose malheureusement pas cette commande SQL, heureusement le fonctionnement de cette requête peut-être simulé grâce à une petite astuce. La requête SQL ci-dessous est l'alternative à INTERSECT.

Syntaxe

```
SELECT DISTINCT value FROM `table1`
WHERE value IN (
  SELECT value
  FROM `table2`
);
```



MINUS

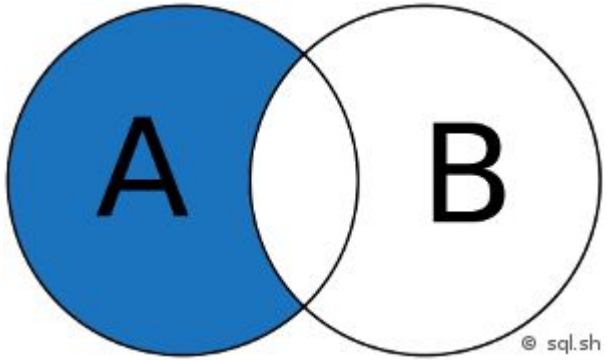
Dans le langage SQL la commande EXCEPT s'utilise entre 2 instructions pour récupérer les enregistrements de la première instruction sans inclure les résultats de la seconde requête. Si un même enregistrement devait être présent dans les résultats des 2 syntaxes, ils ne seront pas présent dans le résultat final.

Syntaxe

```
SELECT * FROM table1

MINUS

SELECT * FROM table2
```



UNION - MINUS - INTERSECT

Exercices

Affichez la table de tous nos clients (tous magasins confondus)

Affichez la table des clients seulement clients dans le 1er magasin

Affichez la table des clients seulement clients dans le 2e magasin

Affichez la table des clients, clients dans le 1er magasin ET le 2e magasin

ENTITY DIAGRAMS

Cours

Je vous invite à regarder les deux vidéos suivantes pour comprendre tout le système des entity diagrams

https://www.youtube.com/watch?v=QpdhBUYk7Kk

https://www.youtube.com/watch?v=-CuY5ADwn24

ENTITY DIAGRAMS

Exercices

Construisez le entity diagram de la base de donnés sql_store

A vous de jouer !

Exercices globaux

A partir du brief client suivant construisez un entity diagram de la base de données qu'il faudrait construire.
Construisez cette base de données et ajoutez au moins 3 enregistrements dans chaque table

Supposons que l'on vous donne les exigences suivantes pour une base de données simple pour la Ligue nationale de hockey (LNH) :
- la NHL a de nombreuses équipes,
- chaque équipe a un nom, une ville, un entraîneur, un capitaine et un ensemble de joueurs,
- chaque joueur n'appartient qu'à une seule équipe,
- chaque joueur a un nom, une position (comme ailier gauche ou gardien de but), un niveau de compétence, et un ensemble de blessures,
- le capitaine d'une équipe est également un joueur,
- un match est joué entre deux équipes (appelées équipe_hôte et équipe_invitée) et a une date (comme le 11 mai 2017) et un score (comme 4 à 2).