

Langage SQL - Initiation



Langage SQL

- Introduction à cette formation

- Votre formateur ..

Et Vous

- Le matériel

- Outils de développement

- Le cours

- L'organisation – horaires

- La forme :

- Un mélange de concepts avec application directe par un exemple simple

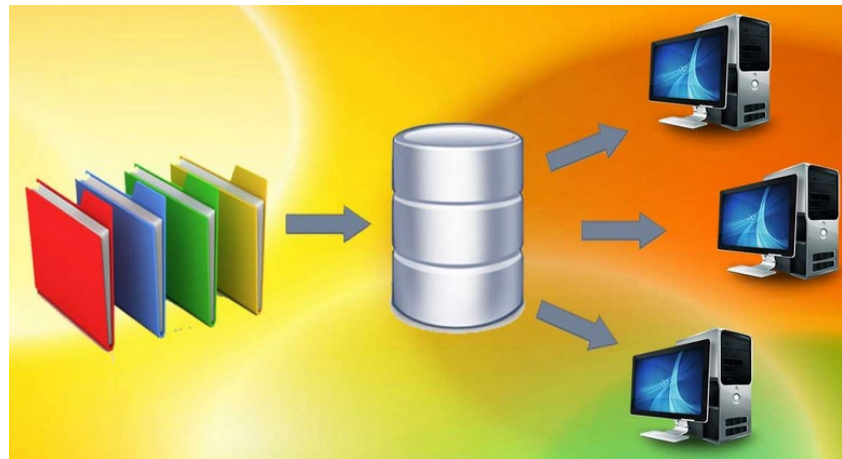
- Des exercices



- Présentation des SGBD
- Présentation du langage SQL
 - Les types de données
 - Les 4 groupes d'instructions SQL
 - Création de base
 - Gestion des tables
 - Parcourir les tables
 - Gestion des utilisateurs
 - Gestion des transactions
- Des exercices

- <https://openclassrooms.com/fr/courses/4449026-initiez-vous-a-lalgebre-relationnelle-avec-le-langage-sql/4538696-comprenez-les-bases-de-donnees-sql>
-
- <https://openclassrooms.com/fr/courses/1959476-administrez-vos-base-s-de-donnees-avec-mysql>
- https://www.w3schools.com/sql/sql_ref_keywords.asp pour SQL
- https://www.w3schools.com/mysql/mysql_ref_functions.asp pour MySQL
- <https://www.mysqltutorial.org/basic-mysql-tutorial.aspx>

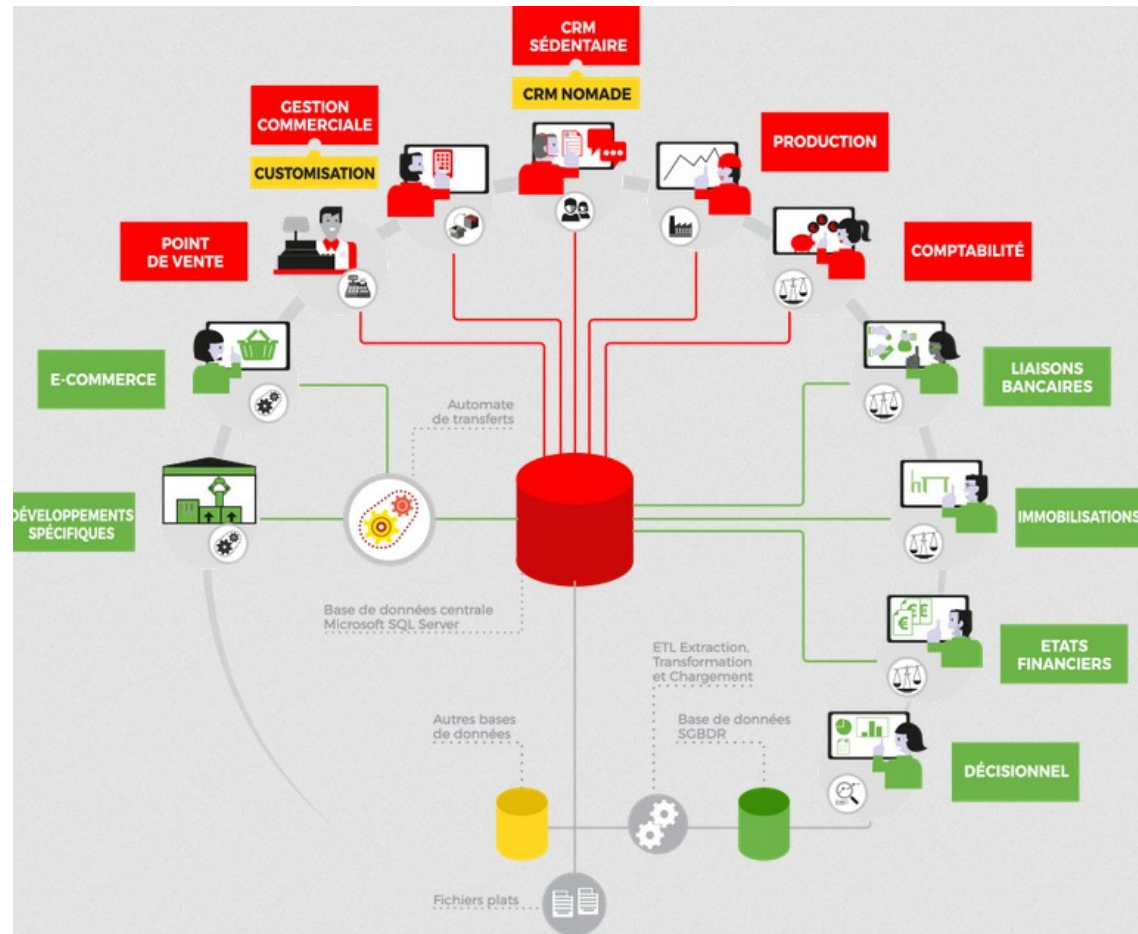
- Une base de données est :
 - Un ensemble de données stockées sur un support informatique
 - Données structurées et organisées de façon à pouvoir facilement les écrire, les modifier, les consulter



- Exemple d'un site Web :



- Exemple d'un ERP d'entreprise



- Un Système de Gestion de Base de Données doit exister pour gérer les données : un **SGBD**

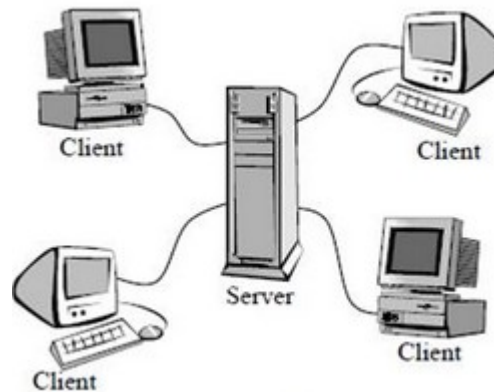
Par analogie, une feuille Excel constitue les données, l'outil Excel est le système qui gère les feuilles Excel

(**RDBMS** Relational DataBase Management System en anglais)

- Un langage standardisé existe pour interagir avec le SGBD : **le langage SQL**

Par analogie en Excel, on écrit des macros en VBA

- Un SGBD est un logiciel qui permet de manipuler les données d'une base de données
- Un SGBD est basé sur le modèle **Client – Serveur**

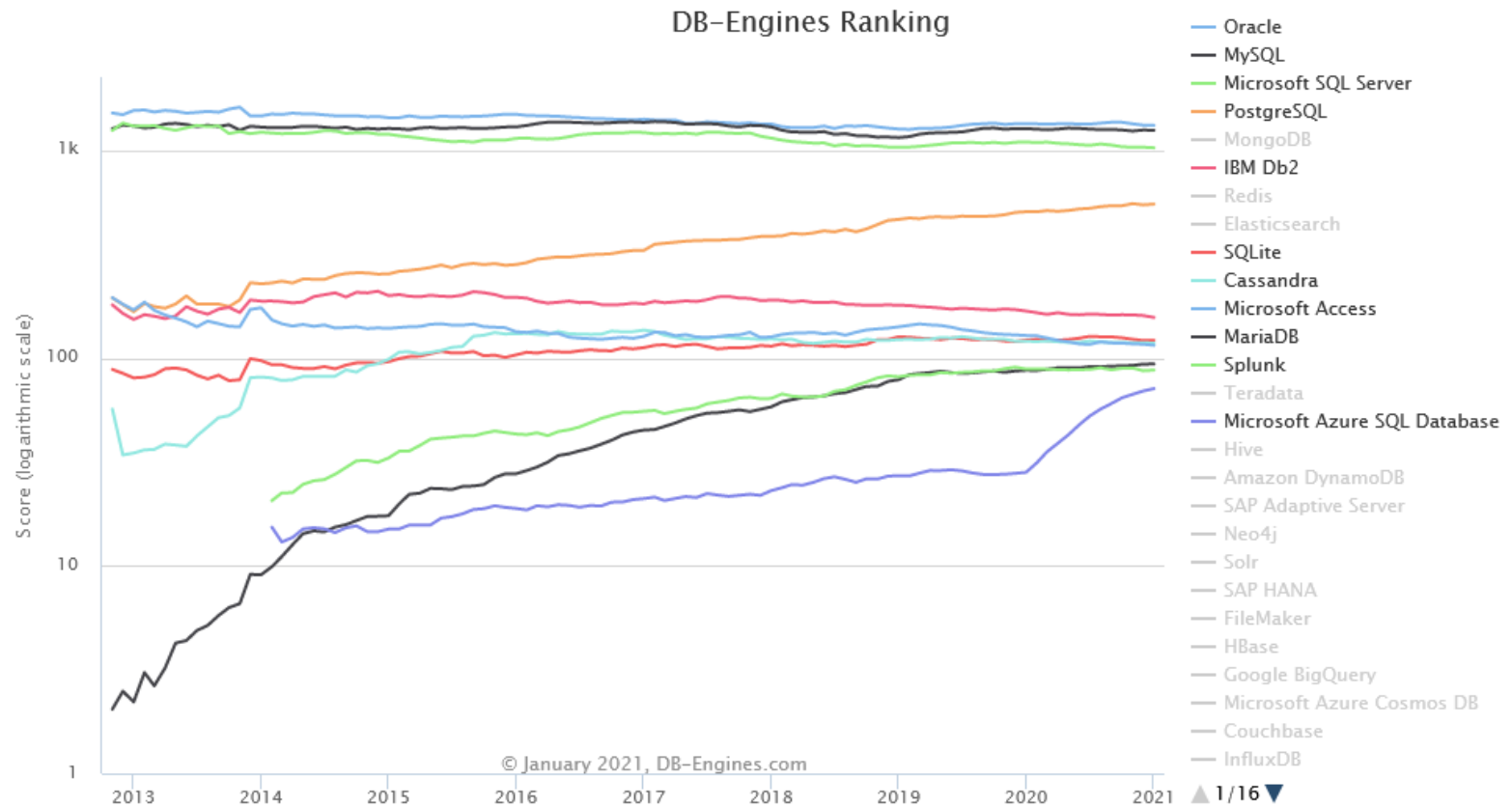


- Un serveur SGBD ne comprend que des « **requêtes** » exprimées dans son propre langage : le SQL

- Une base de données contient plusieurs tables interagissant entre elles pour décrire les données d'une application, d'un site Web, d'une entreprise ...
- Dans un SGBDR, R pour Relationnel, chaque table se nomme une Relation. Les relations sont associées entre elles en fonction des besoins de l'application

SGBD

- Les principaux SGBD du marché :
Site https://db-engines.com/en/ranking_trend



SGBD

DB2 a été développé au début des années 60 par IBM. Tout d'abord réservé aux Mainframe IBM sous VMS, puis Z/Os.

Il a été implémenté en environnement distribué à la fin des années 80 sous Unix et Windows.

Toujours très utilisé aujourd'hui pour la gestion des Databases de gros volume.

ORACLE est un Système de Gestion de Bases de Données Relationnel.

Depuis l'introduction du modèle objet dans sa version 8, il peut être qualifié de SGBDRO (SGBD Relationnel Objet).

Il est développé par ORACLE CORPORATION.

SGBD

SQL SERVER est un SGBD en langage SQL, incorporant entre autre un SGBDR.

Il est développé et commercialisé par la société MICROSOFT.

Il fonctionne sous les OS Linux(mars 2016) et WINDOWS.

Il est possible de le lancer sous MAC OS via Docker.

ACCESS développé par Microsoft est distribué avec la suite OFFICE. Ce SGBD s'adresse aux petites entreprises et aux particuliers

MySQL a été développé par Mickael WIDENIUS. Il est distribué sous une double licence GPL et propriétaire.

C'est le SGBD le plus utilisé au monde. Très prisé des développeurs WEB, en concurrence avec ORACLE, PostgreSQL et SQL SERVER.

Son nom fait référence au Structured Query Language utilisé par tous les SGBD. My est le prénom de sa fille aînée.

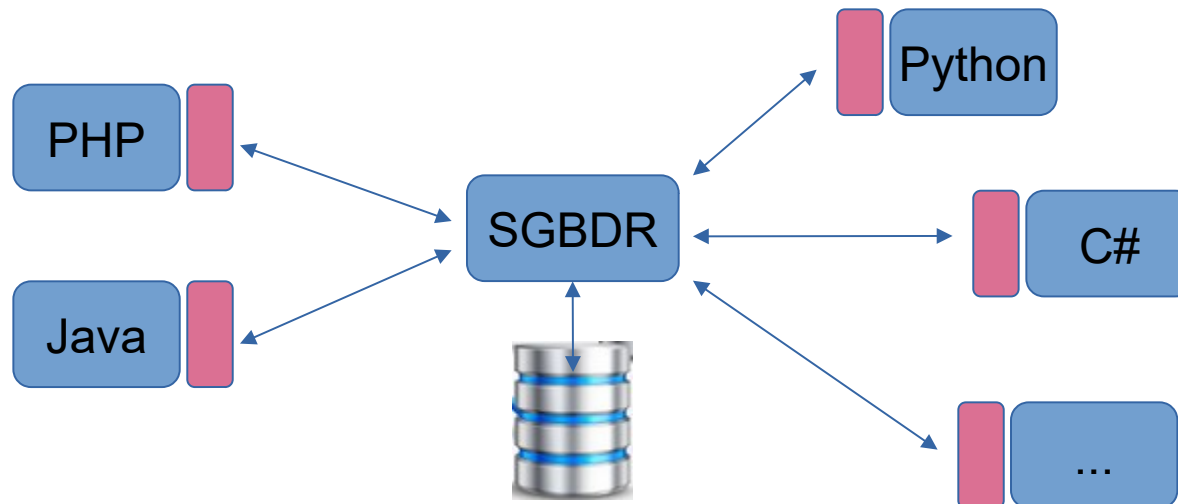
SGBD

MySQL suite...

Après le rachat de MySQL par SUN Microsystems puis le rachat de ce dernier par ORACLE Corporation, Mickaël WIDENIUS a développé un autre SGBD: Maria DB. Bien entendu, Maria comme le prénom de sa seconde fille. La gouvernance mise en place autour de ce projet assure le SGBD de rester un logiciel libre.

PostgreSQL est un SGBD relationnel et objet (SGBDRO). C'est un outil libre, disponible selon les termes d'une licence BSD. Contrairement à une licence GNU/GPL, une licence BSD initialement libre peut devenir propriétaire.

- Un SGBDR ne traite que des demandes, des requêtes, qui lui sont adressées dans le langage SQL
- Une application est écrite dans un langage de programmation autre que le SQL
- Des couches logicielles existent dans tous les langages pour adresser des commandes SQL au SGBDR : des **connecteurs SQL**



- Un exemple de relation : (source openclassroom)

Colonne ou attribut

Numéro	Prénom	Nom	Email
1	Jean	Dupont	jdupont@email.com
2	Marie	Malherbe	mama@email.com
3	Nicolas	Jacques	Jacques.nicolas@email.com
4	Hadrien	Piroux	happi@email.com

Ligne ou entrée

Une donnée appelée
Valeur

- Au niveau de la recherche de données, le langage SQL va permettre :

Numéro	Prénom	Nom	Email
1	Jean	Dupont	jdupont@email.com
2	Marie	Malherbe	mama@email.com
3	Nicolas	Jacques	Jacques.nicolas@email.com
4	Hadrien	Piroux	happi@email.com

- La **sélection** de lignes selon certains critères
- Obtenir une partie des attributs des lignes sélectionnées : la **projection**
- **L'union** : obtenir ce qui se trouve dans 2 tables différentes
- **L'intersection** : obtenir des lignes qui sont communes à 2 tables
- **La différence** : obtenir ce qui se trouve dans un table mais pas dans l'autre
- **La jointure** : lier des informations de 2 tables à partir d'une information commune

- Installons MySQL

Utiliser le document d'exercice, section 2 pour installer l'environnement



- Chaque colonne (attribut) d'une table est définie par un nom unique et un type
- Le bon type doit être choisi :
 - Pour ne pas gaspiller de la mémoire
 - Pour avoir de bonnes performances (ex entier plutôt que chaîne)
 - Pour réaliser les traitements adéquats (ex type date plutôt que chaîne)

- Type entier

Type	Nombre d'octets	Minimum	Maximum
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

L'ajout de **UNSIGNED** indique que les valeurs sont exclusivement positives.
ex : UNSIGNED TINYINT pour des valeurs entre 0 et 255

Pour définir la taille minimale d'affichage : parenthèses

ex : INT(4) pour afficher en prenant l'espace d'au moins 4 caractères

- Type décimal : 5 types possibles
DECIMAL, NUMERIC, FLOAT, REAL et DOUBLE
- DECIMAL et NUMERIC sont équivalents
DECIMAL(5,2) indique l'emploi d'un décimal sur 5 chiffres dont 2 après la virgule
La valeur fournie est enregistrée sans changement.
Conseillé pour des données comptables.
- FLOAT, REAL et DOUBLE pour des données plutôt scientifiques

- Types alphanumériques
 - CHAR et VARCHAR : il faut indiquer la taille souhaitée
CHAR(6), VARCHAR(200)

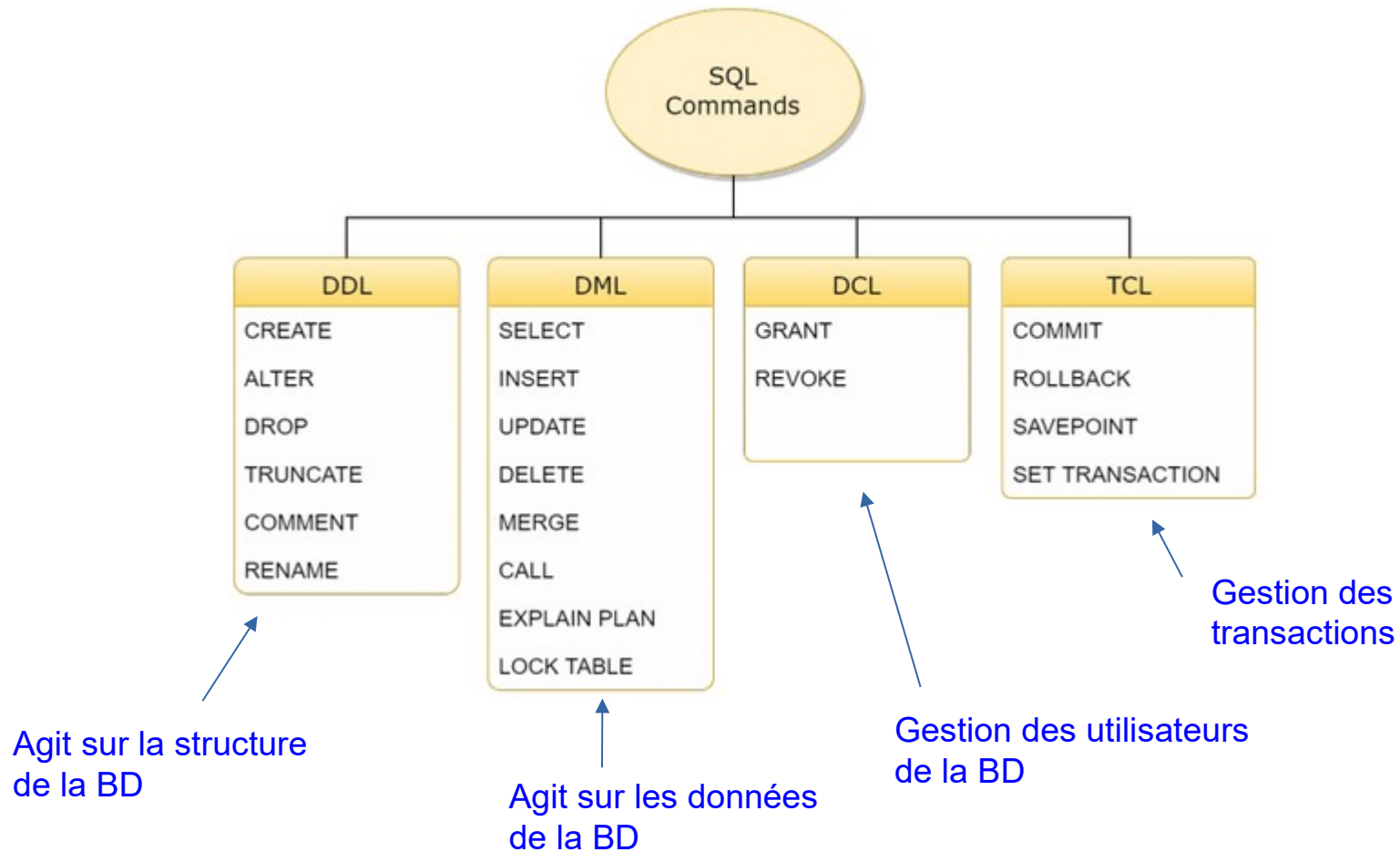
Texte	CHAR(5)	Mémoire requise	VARCHAR(5)	Mémoire requise
"	' '	5 octets	' '	1 octet
'tex'	'tex '	5 octets	'tex'	4 octets
'texte'	'texte'	5 octets	'texte'	6 octets
'texte trop long'	'texte'	5 octets	'texte'	6 octets

- Types alphanumériques (suite)
 - TEXT

Type	Longueur maximale	Mémoire occupée
TINYTEXT	2^8 octets	Longueur de la chaîne + 1 octet
TEXT	2^{16} octets	Longueur de la chaîne + 2 octets
MEDIUMTEXT	2^{24} octets	Longueur de la chaîne + 3 octets
LONGTEXT	2^{32} octets	Longueur de la chaîne + 4 octets

- Types indiquant une date
DATE, DATETIME, TIME, TIMESTAMP et YEAR
- DATE stocke une date sous la forme AAAA-MM-JJ
- DATETIME stocke une date et une heure sous la forme AAAA-MM-JJ HH:MM:SS
- TIME stocke un temps ou un écart de temps (l'heure peut être négative et dépasser 24)
HH:MM:SS
- YEAR stocke uniquement une année (entre 1901 et 2155)
- TIMESTAMP stocke le nombre de secondes depuis le 1^{er} janvier 1970

- Les commandes SQL sont réunies en 4 groupes




- Créer une base de nom nomBD :
 - `CREATE DATABASE nomBD ;`
 - `CREATE DATABASE nomBD CHARACTER SET 'utf8';`
- Supprimer une base :
 - `DROP DATABASE nomDB ;`
 - `DROP DATABASE IF EXISTS nomDB ;`
- Utiliser une base :
 - `USE nomDB ;`
- Exercice jusqu'au chapitre 3.3



- Il faut déterminer les colonnes, leur nom et type
- Il faut pouvoir identifier de manière sûre toute ligne d'une table : il faut une clé unique de recherche : une clé primaire
- Une clé primaire est définie par une colonne ou plusieurs colonnes
- La valeur (ou les valeurs) doit être alors unique pour retrouver la bonne ligne.
- L'usage et la simplicité est d'ajouter aux tables une colonne spécifique pour cette clé primaire souvent nommée id (identifiant). Cette colonne est identifiée en tant que **PRIMARY KEY** et de type **AUTO_INCREMENT**

- **CREATE TABLE [IF NOT EXISTS]** Nom_table (
 colonne1 description_colonne1,
 [colonne2 description_colonne2,
 colonne3 description_colonne3,
 ...,]
 [**PRIMARY KEY** (colonne_clé_primaire)]
)
[**ENGINE**=moteur];
- **SHOW TABLES** ; montre les tables de cette BD
- **DESCRIBE** Nom_table ;montre la structure de la table
- **SHOW COLUMNS FROM** Nom_table ;
- **SHOW CREATE TABLE** Nom_table ;

- **DROP TABLE** Nom_table ; supprime la table
- Exercice chapitre 3.4 : 
- **ALTER TABLE** Nom_table **DROP** Nom_col ; supprimer une colonne
- **ALTER TABLE** Nom_table **CHANGE** Nom_col ... ; modifier la structure d'une colonne

Le Data Manipulation Language (DML)

4 ordres SQL qui servent à manipuler les données :

- INSERT
- SELECT
- UPDATE
- DELETE

- **INSERT INTO** Nom_table(col1, col2, ...)
VALUE(valeur1, valeur2, ...) ;

```
1 INSERT INTO Animal (espece, sexe, date_naissance)
2   VALUES ('tortue', 'F', '2009-08-03 05:12:00');
3 INSERT INTO Animal (nom, commentaires, date_naissance, espece)
4   VALUES ('Choupi', 'Né sans oreille gauche', '2010-10-03 16:44:00', 'chat');
5 INSERT INTO Animal (espece, date_naissance, commentaires, nom, sexe)
6   VALUES ('tortue', '2009-06-13 08:17:00', 'Carapace bizarre', 'Bobosse', 'F');
```

- Insertion multiple

```
1 INSERT INTO Animal (espece, sexe, date_naissance, nom)
2 VALUES ('chien', 'F', '2008-12-06 05:18:00', 'Caroline'),
3         ('chat', 'M', '2008-09-11 15:38:00', 'Bagherra'),
4         ('tortue', NULL, '2010-08-23 05:18:00', NULL);
```

- Exercice chapitre 3.5

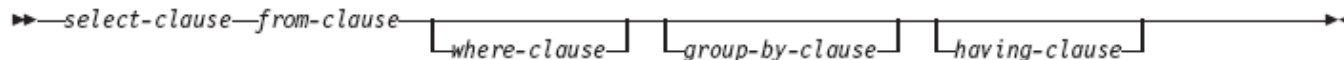


L'ordre SELECT

- La SELECTION est le processus pour obtenir un sous-ensemble de lignes depuis une table donnée.

Se nomme aussi une projection

- L'effet d'une sélection est appelé une table résultante.



Exemples :

```
SELECT * FROM maTable ;
```

affiche toute la table

```
SELECT col1, col2 from maTable ;
```

affiche uniquement les col1 et col2

```
SELECT col1, col2 FROM maTable
```

```
    WHERE col3 = 'xyz'
```

```
    ORDER BY col2 ;
```

affiche col1 et col2 pour les lignes
dont la valeur de col3 vaut 'xyz'

Le résultat est ordonné par valeur
croissante de col2

```
SELECT col1, (SALARY+BONUS) as gain FROM maTable
```

```
    ORDER BY gain ;
```

Affiche col1 et une colonne nommée gain qui est la somme pour chaque ligne de
SALARY et BONUS, ordonné par gain croissant

Résumé des conditions de Recherche avec la clause **WHERE**

- Prédicats de base:

= ^= ou <> (différent de) > >= < <=

- Autres prédicats :

- *Expressions*
- BETWEEN x AND y
- EXISTS
- NOT
- LIKE
- AND/OR
- IN (x,y,z)
- NULL

Exercices jusque 3.9



Les fonctions colonnes, ou fonction scalaires : calculent une valeur unique en utilisant les valeurs d'une colonne

- SUM() : Total des valeurs dans une colonne
- AVG() : Moyenne des valeurs de la colonne
- MIN() : La plus petite valeur d'une colonne
- MAX() : La plus grande valeur d'une colonne
- COUNT() : Nombre d'occurrences

- Critères de groupage :
 - GROUP BY
 - HAVING
 - ORDER BY

https://www.w3schools.com/sql/sql_ref_group_by.asp

Retour sur le INSERT en utilisant un SELECT :

- On peut insérer une ou plusieurs lignes dans une table en utilisant une sous requête de type SELECT.
- Exemple

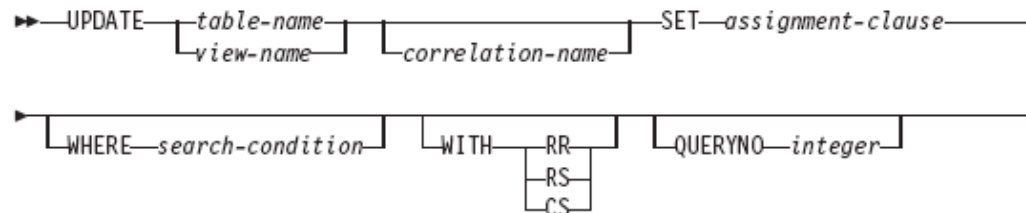
```
INSERT INTO tableDestination (Nom,Prenom, Pays)
      SELECT
            nomActeur, prenomActeur,paysActeur
      FROM tableOrigine ;
```

Dans ce cas les colonnes retournées par l'ordre SELECT doivent avoir les contraintes suivantes :

- être en nombre identique aux colonnes précisées dans la liste ou en l'absence de précision de cette liste le même nombre de colonnes que la table
- avoir le même ordre que l'ordre des noms de colonnes de la liste ou bien le même ordre que les colonnes de la table si l'on omet cette liste
- avoir des types correspondants
- répondre à toutes les contraintes et dans le cas ou au moins une seule valeur viole une contrainte aucune ligne n'est insérée

L'ordre UPDATE

- L'UPDATE est le processus qui permet de mettre à jour des valeurs de colonnes dans des lignes d'une table ou d'une vue .



C'est la forme la plus simple de l'ordre UPDATE :

```
UPDATE T_TARIF SET BREAKFAST = 15 ;
```

Nous voulons par exemple fixer à 15 € les tarifs de **tous** nos petits déjeuners dans la table T_TARIF .

```
UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;
```

```
UPDATE Customers  
SET ContactName = UPPER(ContactName) ; // passage en majuscules
```


- Autres exemples
 - **UPDATE** Tarif
 SET TRF = TRF * 1.15
 WHERE YEAR(DATE_DEBUT) > 2019 ;

Comme dans les ordres INSERT et DELETE, il est possible d'utiliser une sous requête dans la clause WHERE de l'ordre UPDATE afin de filtrer de manière plus complète :

```
UPDATE EMP.T1  
  SET SALARY=(SELECT AVG(T2.SALARY) FROM EMP.T2)  
  WHERE WORKDEPT='E11' AND  
    SALARY < (SELECT AVG(T3.SALARY) FROM EMP.T3);
```

Pour les employés de la table T1 dont le département vaut 'E11' et pour lesquels le salaire est inférieur à la moyenne des salaires de la table T3, modifier leur salaire avec la moyenne des salaires de la table T2.

Voici les principaux cas pour lesquels un ordre de modification ne peut aboutir :

- violation de clef (index primaire)
- violation de contrainte d'index secondaire unique
- violation de contrainte de données (colonne not NULL)
- violation d'intégrité référentielle (la valeur est une clé étrangère dans une autre table)

L'ordre **DELETE**

Le DELETE est le processus par lequel on détruit une ou plusieurs lignes dans une table ou une vue
(un delete dans une vue entraîne le Delete dans la table attachée) .

```
DELETE FROM EMP WHERE  
    WORKDEPT='E11' OR WORKDEPT='D21';
```

La syntaxe de base de l'ordre SQL de suppression de données dans une table est la suivante :

Le seul cas pour lequel cet ordre peut ne pas aboutir est lorsque la suppression viole la **contrainte d'intégrité référentielle**.

Il est en effet absurde de vouloir supprimer un client si les factures relatives à ce client n'ont pas été préalablement supprimées.

Dans certains cas, il se peut que la suppression d'une ligne entraîne la suppression d'autres lignes dans d'autres tables lorsqu'il existe des intégrités référentielles de suppression en cascade.

‘Le bon Coin du vélo’ :

Un site permet à des propriétaires d'exposer des articles d'occasion de vélo

id	nom	tel
1	Albert95	06 11 22 33 44
2	Nemo75	01 25 32 23 45
3	Nanou	
4	Yoan22	06 11 22 11 22

Table
proprietaires

id	nom	marque	prix	id_client
1	EX bike 700S	Asics	700	1
2	Dérailleur asics 700	Asics	80	1
3	Vélocity 400	Bobike	450	2
4	Brunch 27	Curana	350	4
5	Delta 500	Delta	650	2
6	Delta 600	Delta	700	

Table articles

Problématique :

Récupérer la liste des articles avec leur propriétaire

Solution : utiliser une Jointure

2 types de Jointures :

Jointure interne : ne sélectionne que les données qui ont une correspondance dans les 2 tables

Dans l'ex précédent le propriétaire Nanou et l'article Delta 600 se seront pas extraits

Emploi de WHERE (ancienne méthode) et INNER JOIN

Jointure externe : sélectionne aussi les données qui n'ont pas de correspondance

- Dans la 1^{ère} table (celle de 'gauche') : LEFT JOIN
- Dans la 2^{ème} table (celle de 'droite') : RIGHT JOIN

2 types de Jointures :

Jointure interne : ne sélectionne que les données qui ont une correspondance dans les 2 tables

Dans l'ex précédent le propriétaire Nanou et l'article Delta 600 se seront pas extraits

Emploi de WHERE (ancienne méthode) et INNER JOIN

Jointure externe : sélectionne aussi les données qui n'ont pas de correspondance

- Dans la 1^{ère} table (celle de 'gauche') : LEFT JOIN
- Dans la 2^{ème} table (celle de 'droite') : RIGHT JOIN

id	nom	tel
1	Albert95	06 11 22 33 44
2	Nemo75	01 25 32 23 45
3	Nanou	
4	Yoan22	06 11 22 11 22

Table proprietaires

id	nom	marque	prix	id_client
1	EX bike 700S	Asics	700	1
2	Dérailleur asics 700	Asics	80	1
3	Vélocity 400	Bobike	450	2
4	Brunch 27	Curana	350	4
5	Delta 500	Delta	650	2
6	Delta 600	Delta	700	

Table articles

Exemples

Jointure interne

```
SELECT articles.nom, proprietaires.nom
FROM proprietaires, articles
WHERE articles.id_client = proprietaires.id
```

Avec ALIAS

```
SELECT articles.nom AS nom_art, proprietaires.nom AS nom_prop
FROM proprietaires, articles
WHERE articles.id_client = proprietaires.id
```

Avec INNER JOIN

```
SELECT articles.nom AS nom_art, proprietaires.nom AS nom_prop
FROM proprietaires
INNER JOIN articles
ON articles.id_client = proprietaires.id
```

id	nom	tel
1	Albert95	06 11 22 33 44
2	Nemo75	01 25 32 23 45
3	Nanou	
4	Yoan22	06 11 22 11 22

Table proprietaires

id	nom	marque	prix	id_client
1	EX bike 700S	Asics	700	1
2	Dérailleur asics 700	Asics	80	1
3	Vélocity 400	Bobike	450	2
4	Brunch 27	Curana	350	4
5	Delta 500	Delta	650	2
6	Delta 600	Delta	700	

Table articles

Exemples

Jointure externe

JOIN imposé

LEFT JOIN : récupérer toute la table de gauche

```
SELECT articles.nom AS nom_art, proprietaires.nom AS nom_prop
FROM proprietaires
LEFT JOIN articles
ON articles.id_client = proprietaires.id
   fournit NULL Nanou
```

RIGHT JOIN : récupérer toute la table de droite

```
SELECT articles.nom AS nom_art, proprietaires.nom AS nom_prop
FROM proprietaires
RIGHT JOIN articles
ON articles.id_client = proprietaires.id
   fournit Delta 600 NULL
```

Les Index

Ils sont utilisés pour permettre l'**accès direct** à une ligne ou à un groupe de lignes basé sur une valeur de clé.

C'est l'objet le plus significatif en termes d'amélioration des performances d'application car en l'absence d'index, la seule alternative offerte au SGBD pour accéder aux données est le balayage de la table, soit un **accès séquentiel**.

```
CREATE INDEX idx_lastname  
ON Persons (LastName);
```

```
CREATE INDEX idx_pname  
ON Persons (LastName, FirstName);
```

Les requêtes SELECT qui suivent utilisent normalement LastName ou LastName et FirstName, elles seront alors plus rapides

La *VUE(VIEW)* est une autre solution pour accéder aux données .

C'est une sélection prédéfinie de données en provenance d'une ou plusieurs tables.

Elle est manipulable comme une table en lecture.

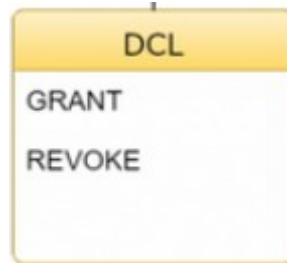
Les données d'une *VUE* ne sont pas stockées. Elles existent uniquement en mémoire.

Les *VIEW* sont utilisées pour :

- Réduire la complexité d'accès(Ordre SQL plus simple
- Augmenter la sécurité et confidentialité –accès limité aux données de la vue pour un utilisateur

- Exemple

```
CREATE VIEW ProductsAboveAveragePrice AS  
  SELECT ProductName, Price  
  FROM Products  
  WHERE Price > (SELECT AVG(Price) FROM Products);
```



Des utilisateurs ou des types d'utilisateur doivent être créés pour leur associer les bons droits – les privilèges

- Créer un utilisateur :
`CREATE USER 'login'@'hote' [IDENTIFIED BY 'mot_de_passe'];`
- Supprimer un utilisateur :
`DROP USER 'login'@'hote';`
- Exemple
`CREATE USER 'max'@'localhost' IDENTIFIED BY 'maxisthebest';`

- Associer des privilèges : GRANT
GRANT <droits> ON <tables> TO <utilisateur>
- Exemples sur <http://www.bts-sio.com/cours/SI3/sql/8%20-%20GRANT.php>

```
GRANT SELECT ON bddgestion.produit TO visiteur ;
```

```
GRANT SELECT, UPDATE, DELETE, INSERT  
ON bddgestion.produit, bddgestion.gamme  
TO admin, visiteur;
```

- Liste des droits :

Droit	Signification
ALL ou ALL PRIVILEGES	Tous les droits sauf WITH GRANT OPTION.
ALTER	Autorise l'utilisation de ALTER TABLE.
CREATE	Autorise l'utilisation de CREATE TABLE.
DELETE	Autorise l'utilisation de DELETE.
DROP	Autorise l'utilisation de DROP TABLE.
INSERT	Autorise l'utilisation de INSERT.
INDEX	Autorise l'utilisation de CREATE INDEX et DROP INDEX.
SELECT	Autorise l'utilisation de SELECT.
UPDATE	Autorise l'utilisation de UPDATE.
GRANT OPTION	Synonyme pour WITH GRANT OPTION

- Par défaut MySQL est en mode autocommit : pas de mécanisme de transaction
- Pour enclencher le mécanisme :
`SET autocommit=0 ;`
- Pour valider un bloc : `COMMIT`
- Pour un retour en arrière : `ROLLBACK`
- Si on laisse `SET autocommit=1 ;` (mode autocommit) on peut toutefois commencer une transaction par `START TRANSACTION`
- Exemples sur
<https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql/1970063-transactions>