

SQL



ADRAR DIGIT@L ACADEMY

PÔLE NUMERIQUE DU CENTRE DE FORMATION ADRAR
> SUPPORT, ADMINISTRATION SYSTEMES & RESEAUX
> DEVELOPPEMENT D'APPLICATIONS WEB & MOBILES
> TRANSFORMATION NUMERIQUE DES ENTREPRISES

<http://www.adrar-numerique.com>

SQL

Le **SQL** (Structured Query Language) est un langage permettant de communiquer avec une base de données.

Ressources :

- **Documentation officielle (en anglais 😊) :**
<https://dev.mysql.com/doc/refman/8.4/en/>
- **SQL.sh**
- **W3Schools**



SQL

Pré-requis :

Pour la suite de votre formation et de ce cours, vous allez avoir besoin de certains outils installés sur votre machine.

- Installation de Xampp (ensemble de logiciels pour construire un serveur local):

<https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/8.2.12/xampp-windows-x64-8.2.12-0-VS16-installer.exe>

(dans la liste, garder uniquement phpMyAdmin et MySQL)

- Installation de Workbench:

Commençons par télécharger le logiciel MySQL Community :

- Version x64 pour windows 64 bits :

<https://dev.mysql.com/downloads/file/?id=519997>

SQL

Définition d'une base données :

Une base de données permet de stocker, organiser et retrouver des données facilement.

Champ : Un champ est la zone qui permet le stockage des données, correspondant à une colonne dans une vue en liste.


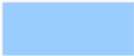
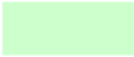

Enregistrement : Un enregistrement est un ensemble de valeurs correspondant à une ligne toujours dans une vue en liste.

Table : Une table va contenir un ensemble d'enregistrements.

Index : L'index est le conteneur des clés

SQL

id	nom	prénom	profession	code postal	ville
1	Durand	Michel	Directeur	75016	Paris
2	Dupond	Karine	Secrétaire	92000	Courbevoie
3	Mensoif	Gérard	Commercial	75001	Paris
4	Monauto	Alphonse	Commercial	75002	Paris
5	Emarre	Jean	Employé	75015	Paris
6	Abois	Nicole	Secrétaire	95000	St Denis
7	Dupond	Antoine	Assistant commercial	75014	Paris

-  intitulé du champ
-  enregistrement
-  Champ
-  Table

SQL

- Une base de données est un ensemble qui permet le stockage des données.
- Les données sont écrites de manière structurée ce qui signifie que chaque donnée est enregistrée dans un champ qui est inclus dans une table.
- Lorsque les tables ont des relations entre elles, on parle alors de bases de données relationnelles.

SQL

Le SGBD ou SGBDR

Le SGBD va gérer l'accès à la base de données. Aucun logiciel n'accédera directement à la base de données, seul le SGBD le fera.

Cela implique que le SGBD :

- disposera d'un langage pour pouvoir dialoguer avec les applications (notamment le SQL)
- gérera l'écriture et la lecture des données,
- mettra à disposition les tables, et les droits associés,
- gérera le partage des données,
- s'assurera de l'intégrité des données,
- gérera la relation entre les tables (dans le cas de bases de données relationnelles)

SQL

Les avantages à utiliser les bases de données sont les suivants :

- Une standardisation des accès :

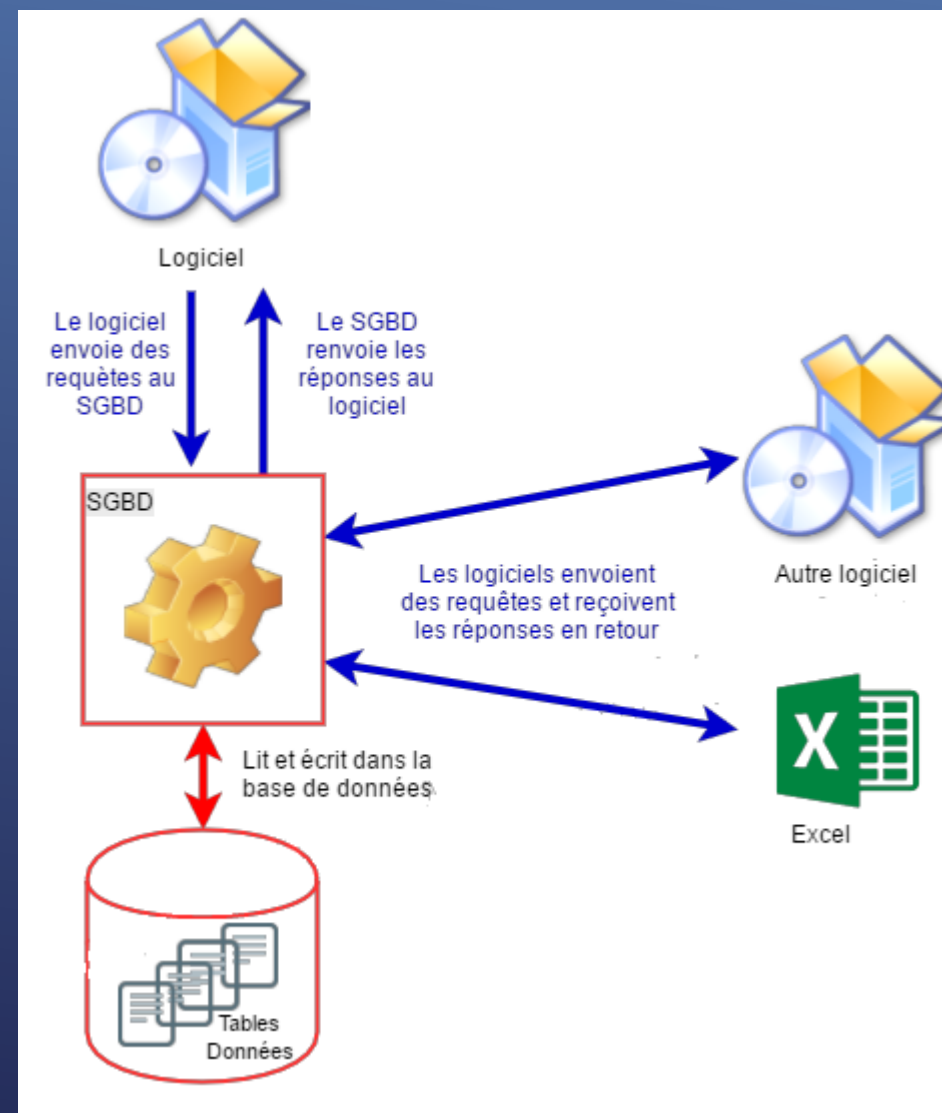
Chaque logiciel accède aux données via le SGBD en utilisant un langage standardisé. Il est possible de changer de base de données sans avoir à réécrire le logiciel.

- Un partage des données ainsi que l'accès simultané :

Le SGBD gère les accès à la base de données, donc il gère aussi le partage des données. Il est donc possible d'avoir plusieurs logiciels qui lisent et écrivent en même temps sur la base de données.

- Une grande fiabilité des données :

Chaque logiciel ne peut pas faire n'importe quoi. C'est le SGBD qui gère l'enregistrement des données.



SQL

Les principaux SGBDR utilisés en entreprise sont:

- Oracle : sans doute le plus connu,
- MySQL : Un système gratuit issu du monde libre,
- SQLServer : Le SGBD de Microsoft.

Bon à savoir : vous pourrez trouver les appellations de base de données sous les formes suivantes :

- BdD: Base de Données
- BD: Base Données
- DB: DataBase (nom en anglais)

SQL

Afin que les logiciels utilisés et le SGBD puissent se comprendre, ils utilisent un langage appelé SQL. Ce langage complet va être utilisé pour :

- Lire les données,
- Ecrire les données,
- Modifier les données,
- Supprimer les données

Il permettra aussi de modifier la structure de la base de données :

- Ajouter des tables,
- Modifier les tables,
- les supprimer
- Ajouter, ou supprimer des utilisateurs,
- Gérer les droits des utilisateurs,
- Gérer les bases de données : en créer de nouvelles, les modifier, etc.

SQL

Démarrer le serveur local

- démarrer Xampp (Apache + MySQL)
- ouvrir Workbench
- cliquer sur l'instance locale OU créer une nouvelle connexion (un nom, un user et un mdp)
- (OU cliquer sur une connexion existante)

SQL : les requêtes

Il existe 3 types de requêtes :

- Les requêtes de STRUCTURE
- Les requêtes de MISE à JOUR
- Les requêtes de CONSULTATION

SQL : les requêtes de structure

CREATION DE LA BASE DE DONNEES :

CREATE DATABASE nom_base_de_donnée;

- ▶ cliquer sur l'éclair pour appliquer
- ▶ ne pas oublier la commande **USE nom_base** ; pour spécifier la base à utiliser pour la suite



SQL : les requêtes de structure

CREATION DE TABLES :

CREATE TABLE nom_table (...) : permet de créer une table, en définissant les champs et le type des données qui seront contenues dans ces champs.

Ex :

```
CREATE TABLE users (
    id_user INT auto_increment PRIMARY KEY NOT NULL,
    name_user VARCHAR(50) NOT NULL,
    password_user VARCHAR(255) NOT NULL,
    mail_user VARCHAR(100) NOT NULL,
    age_user INT,
    id_town INT
);
```

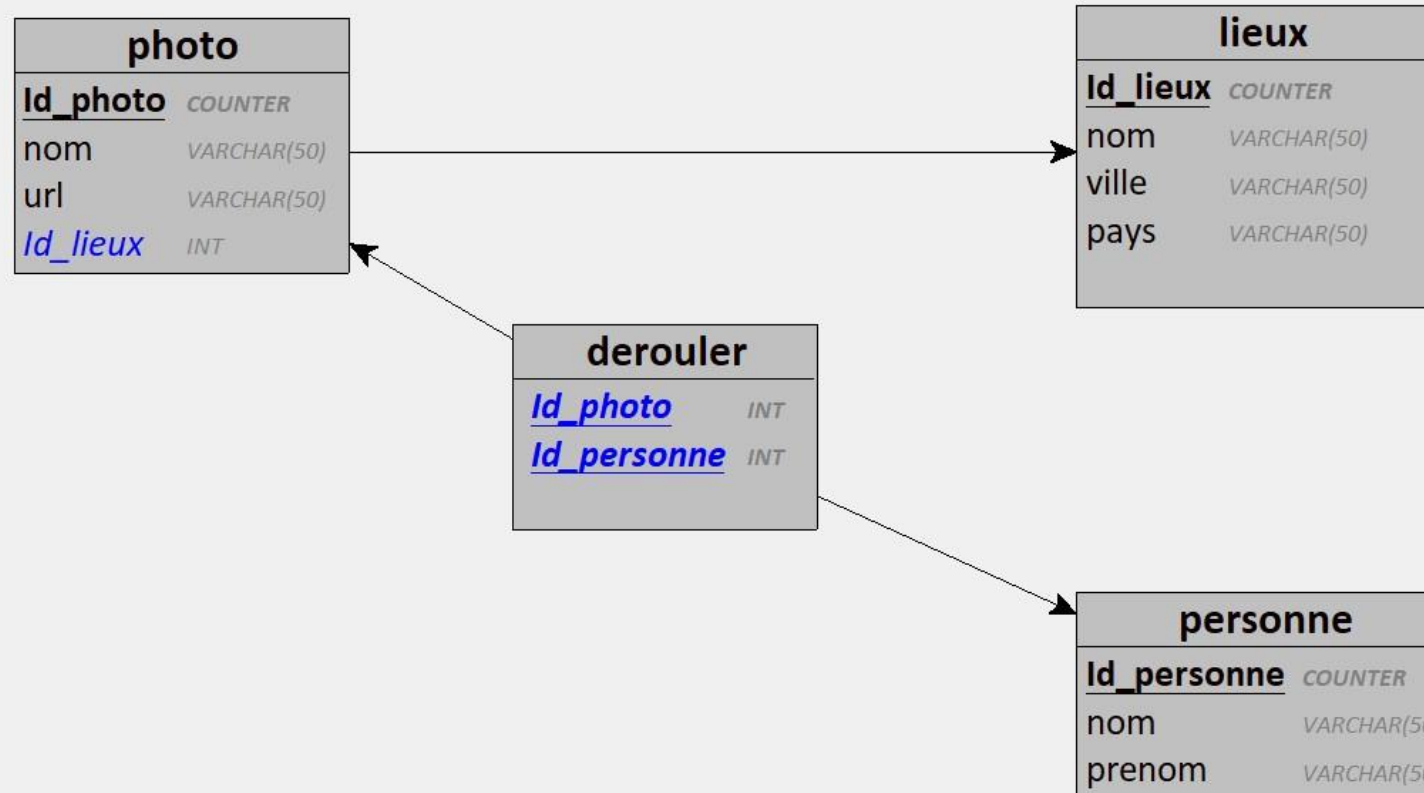
SQL : les requêtes de structure

Les types principaux :

- Int : integer : un entier
- Float : nombre réel
- Varchar(nbrcaractère) : chaîne de caractères qui peut contenir des chiffres et des lettres
- Char(nbrcaractère) : chaîne composée que de caractères
- Date

SQL : les requêtes de structure

Avant d'aller plus loin, pour construire une base de données, on se réfère au MLD



SQL : les requêtes de structure

La requête ALTER TABLE :

Elle va nous permettre de modifier la structure des tables déjà créées dans notre BDD.

- Spécifier une clé primaire, après création de la table :

ALTER TABLE nom_table **ADD PRIMARY KEY** (nom_du_champ) : **spécifier la clé primaire** d'une table déjà créée (dans le cas où cela n'aurait pas été fait à la création de la table)

- Créer une contrainte de clé étrangère (pour les relations entre nos tables) :

ALTER TABLE nom_table **ADD CONSTRAINT** nom_contrainte **FOREIGN KEY** (nom_champ) REFERENCES nom_table (nom_champ) : on va pouvoir **créer une contrainte de clé étrangère**. On signale à notre SGBD que ce champ fait référence à un champ d'une autre table.

- Modifier un champ :

ALTER TABLE nom_table **MODIFY** nom_champ : modifier un champ existant (pour lui ajouter une propriété « NOT NULL » par exemple)

SQL : les requêtes de structure

La requête ALTER TABLE :

Elle va nous permettre de modifier la structure des tables déjà créées dans notre BDD.

- Ajouter un champ (colonne), après création de la table :

ALTER TABLE nom_table **ADD COLUMN** nom_du_champ type : **ajouter un nouveau champ à une table existante** (il faut préciser son type)

- Modifier le nom et/ou le type d'un champ :

ALTER TABLE nom_table **CHANGE** ancien_nom_champ nouveau_nom_champ type

- Supprimer un champ (une colonne) :

ALTER TABLE nom_table **DROP** nom_champ;

ou alors

ALTER TABLE nom_table **DROP COLUMN** nom_champ;



SQL : les requêtes de structure

La requête DROP TABLE :

Elle va nous permettre de supprimer définitivement une table entière de notre BDD.

```
DROP TABLE nom_table;
```

Si cette table est liée à une autre il faut commencer par supprimer le lien ou la table où elle est clé étrangère :

```
ALTER TABLE nom_table DROP FOREIGN KEY nom_lien ;
```



SQL : les requêtes de mise à jour

La requête INSERT INTO:

Cette méthode nous permettra d'insérer des données dans une base en créant un nouvel enregistrement.

2 syntaxes possibles :

- Insérer une ligne en indiquant les informations pour chaque colonne existante (**il faut toutes les colonnes, en respectant l'ordre**)

INSERT INTO nom_table VALUES ('valeur 1', 'valeur 2', ...)

- Insérer une ligne en spécifiant les colonnes que vous souhaitez compléter. Il est possible d'insérer une ligne en renseignant seulement une partie des colonnes (**on est pas obligé de mettre toutes les colonnes, et on peut modifier l'ordre**)

INSERT INTO table (nom_colonne_1, nom_colonne_2, ...) VALUES ('valeur 1', 'valeur 2', ...)

SQL : les requêtes de mise à jour

La requête INSERT INTO:

Cette méthode nous permettra d'insérer des données dans une base en créant un nouvel enregistrement.

Pour résumer:

- Si on ne spécifie pas le nom des colonnes, on doit respecter l'ordre et les remplir obligatoirement toutes
- Si on spécifie les colonnes, on peut choisir les quelles on remplit
- Avec l'une ou l'autre méthode, on peut effectuer des insertions multiples.

SQL : les requêtes de mise à jour

La requête UPDATE:

Cette méthode nous permet d'effectuer des modifications sur des enregistrements déjà existants.

Souvent, cette commande est utilisée avec WHERE, qui est une contrainte permettant de spécifier les enregistrements à modifier, grâce à une condition.

La syntaxe de base est écrite ainsi :

UPDATE table

SET nom_colonne_1 = 'nouvelle valeur'

WHERE condition

Si la condition WHERE n'est pas précisée, toutes les données du champ nom_colonne_1 seraient alors modifiées (pour tous les enregistrements de la table !).

SQL : les requêtes de mise à jour

La commande WHERE

Voici les opérateurs de comparaison qui peuvent être utilisés avec le WHERE :

Opérateur	Description
=	Egale
<>	Pas égale
!=	Pas égale
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal à
IN	Liste de plusieurs valeurs possibles
BETWEEN AND	Valeur comprise dans un intervalle de données
LIKE	Recherche en spécifiant le début, le milieu ou la fin d'un mot
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle

SQL : les requêtes de mise à jour

```
UPDATE users
SET idTown = "11"
WHERE idUser = 3;
```

idUser	idsession	name	password	mail	phone	idtown
1	NULL	test1	pass1	NULL	NULL	33330
2	NULL	test2	pass2	NULL	NULL	NULL
3	NULL	test3	pass3	NULL	NULL	11
4	NULL	test4	pass4	NULL	NULL	NULL

```
UPDATE users
SET idTown = "74";
```

idUser	idsession	name	password	mail	phone	idtown
1	NULL	test1	pass1	NULL	NULL	74
2	NULL	test2	pass2	NULL	NULL	74
3	NULL	test3	pass3	NULL	NULL	74
4	NULL	test4	pass4	NULL	NULL	74

SQL : les requêtes de mise à jour

La requête DELETE:

Cette commande permet de supprimer des enregistrements dans une table.

En associant cette commande avec WHERE, il est donc possible de sélectionner seulement les lignes soumises à la condition.

La syntaxe est la suivante :

```
DELETE FROM `table`
```

```
WHERE condition
```

SQL : les requêtes de mise à jour

idUser	idsession	name	password	mail	phone	idtown
1	NULL	test1	pass1	NULL	NULL	74
2	NULL	test2	pass2	NULL	NULL	74
3	NULL	test3	pass3	NULL	NULL	74
4	NULL	test4	pass4	NULL	NULL	74

```
DELETE FROM users
WHERE idUser = 2;
```

idUser	idsession	name	password	mail	phone	idtown
1	NULL	test1	pass1	NULL	NULL	74
3	NULL	test3	pass3	NULL	NULL	74
4	NULL	test4	pass4	NULL	NULL	74

```
DELETE FROM users
WHERE idUser < 3;
```

idUser	idsession	name	password	mail	phone	idtown
3	NULL	test3	pass3	NULL	NULL	74
4	NULL	test4	pass4	NULL	NULL	74

SQL : les requêtes de mise à jour

La requête DELETE:

Vider une table ?

```
DELETE FROM nom_table
```

(sans le WHERE)

Voir aussi :

```
TRUNCATE TABLE nom_table
```

SQL : les requêtes de consultation

La requête **SELECT**:

La commande **SELECT** permet de faire des opérations de recherche et de calcul à partir des enregistrements d'un ou plusieurs attributs d'une table.

La syntaxe de base est écrite ainsi :

```
SELECT nom_attribut
```

```
FROM table1 ;
```

Pour sélectionner plusieurs attributs on les sépare par des virgules :

```
SELECT nom_attribut1, nom_attribut2
```

```
FROM table1 ;
```

Et pour sélectionner tous les attributs :

```
SELECT *
```

```
FROM table1 ;
```


SQL : les requêtes de consultation

La commande DISTINCT

La commande SELECT peut potentiellement afficher des enregistrements en double. La commande DISTINCT permet d'éviter des redondances dans les résultats.

```
SELECT DISTINCT nom_attribut FROM table1 ;
```

SQL : les requêtes de consultation

La commande WHERE

Voici les opérateurs de comparaison qui peuvent être utilisés avec le WHERE :

Opérateur	Description
=	Egale
<>	Pas égale
!=	Pas égale
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal à
IN	Liste de plusieurs valeurs possibles
BETWEEN AND	Valeur comprise dans un intervalle de données
LIKE	Recherche en spécifiant le début, le milieu ou la fin d'un mot
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle

SQL : les requêtes de consultation

Les opérateurs logiques AND et OR :

Les opérateurs logiques AND et OR peuvent être utilisés dans la commande WHERE pour combiner des conditions.

- L'opérateur AND permet d'afficher un résultat seulement si les conditions 1 et 2 sont vérifiées.

```
SELECT nom_attribut1, nom_attribut2 FROM table1
```

```
WHERE condition1 AND condition2;
```

- L'opérateur OR permet d'afficher un résultat dès lors qu'une des 2 conditions est vérifiée.

```
SELECT nom_attribut1, nom_attribut2 FROM table1
```

```
WHERE condition1 OR condition2;
```

Il est également possible de combiner ces 2 opérateurs logiques ce qui pourrait donner une syntaxe comme celle-ci par exemple :

```
SELECT nom_attribut1, nom_attribut2
```

```
FROM table1
```

```
WHERE condition1 AND (condition2 OR condition3);
```

SQL : les requêtes de consultation

La commande AS

La commande AS permet d'utiliser des alias pour renommer temporairement un attribut ou une table dans une requête (juste le temps d'une requête)

```
SELECT nom_attribut1 AS na1, nom_attribut2 AS na2 FROM table1 ;
```

```
SELECT nom_attribut1 na1, nom_attribut2 na2 FROM table1 ;
```

```
SELECT nom_attribut1 FROM table1 AS t1 ;
```

SQL : les requêtes de consultation

La commande ORDER BY

La commande ORDER BY permet de trier des lignes dans un résultat d'une requête. Il est possible de trier les données sur un ou plusieurs attributs par ordre ascendant ou descendant.

La syntaxe de base est écrite ainsi :

```
SELECT nom_attribut1, nom_attribut2
```

```
FROM table1
```

```
ORDER BY nom_attribut1
```

Par défaut, les résultats sont classés par ordre ascendant.

Pour trier le résultat par ordre décroissant nous utilisons le suffixe DESC :

```
SELECT nom_attribut1, nom_attribut2
```

```
FROM table1
```

```
ORDER BY nom_attribut1 DESC, nom_attribut2 ASC ;
```

SQL : les requêtes de consultation

La commande GROUP BY

La commande GROUP BY est utilisée avec les fonctions de calcul. Elle permet de regrouper le résultat.

Elle est obligatoire sur tous les attributs qui entourent la fonction de calcul dans le SELECT

```
SELECT nom_attribut1, fonction (nom_attribut2) FROM table1  
GROUP BY nom_attribut1;
```

Cette commande doit toujours s'utiliser après la commande WHERE et avant la commande HAVING.

SQL : les requêtes de consultation

La commande GROUP BY

Les fonctions possibles :

Fonctions	Description
AVG()	Calcul de la moyenne d'un ensemble de valeurs
COUNT()	Pour compter le nombre de lignes concernées
MAX()	Permet de récupérer la plus haute valeur
MIN()	Permet de récupérer la plus petite valeur
SUM()	Somme de plusieurs lignes

SQL : les requêtes de consultation

La commande HAVING

Cette commande est presque similaire au WHERE à la seule différence que le HAVING permet de filtrer en utilisant les fonctions de calcul.

```
SELECT nom_attribut1, fonction (nom_attribut2) FROM table1
```

```
GROUP BY nom_attribut1
```

```
HAVING fonction (nom_attribut2) = condition1;
```

SQL : les requêtes de consultation

Les jointures

Les jointures permettent d'afficher le résultat de l'association plusieurs tables dans une requête en utilisant la clé étrangère.

Il existe plusieurs méthodes pour associer 2 tables ensemble :

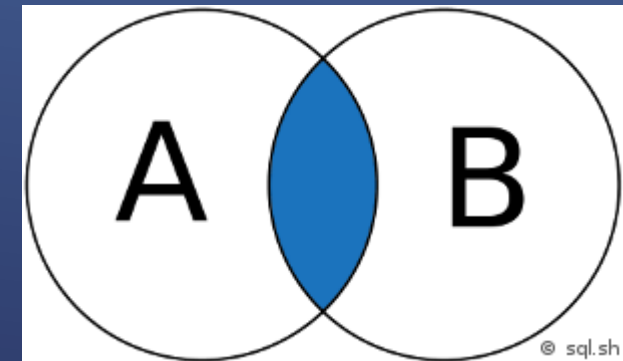
1. INNER JOIN
2. CROSS JOIN
3. LEFT JOIN (ou LEFT OUTER JOIN)
4. RIGHT JOIN (ou RIGHT OUTER JOIN)
5. FULL JOIN (ou FULL OUTER JOIN)
6. SELF JOIN
7. NATURAL JOIN

SQL : les requêtes de consultation

Les jointures

- ▶ INNER JOIN : jointure interne pour retourner les enregistrements quand la condition est vraie dans les 2 tables. C'est l'une des jointures les plus communes.

```
SELECT *  
FROM A  
INNER JOIN B  
ON A.key = B.key
```



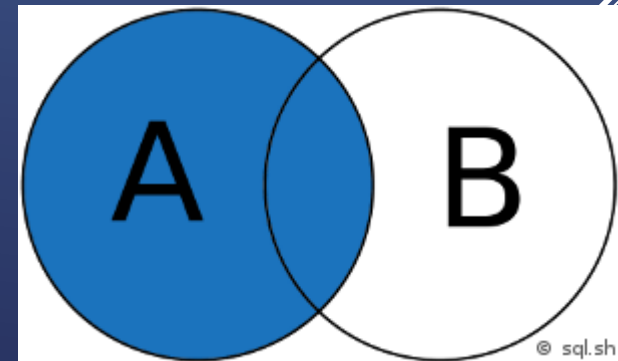
SQL : les requêtes de consultation

Les jointures

- ▶ LEFT JOIN : C'est une jointure externe gauche.

Cette commande retourne tous les enregistrements de la table A même si la condition n'est pas vérifiée dans l'autre table.

```
SELECT *  
FROM A  
LEFT JOIN B  
ON A.key = B.key ;
```



SQL : les requêtes de consultation

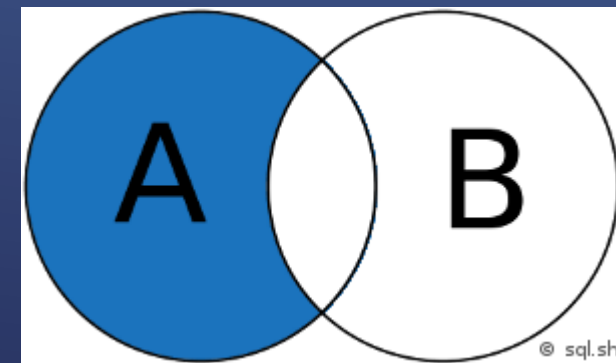
Les jointures

- ▶ LEFT JOIN : C'est une jointure externe gauche.

Cette commande retourne tous les enregistrements de la table A même si la condition n'est pas vérifiée dans l'autre table.

(sans l'intersection de B)

```
SELECT *  
FROM A  
LEFT JOIN B  
ON A.key = B.key  
WHERE B.key IS NULL
```



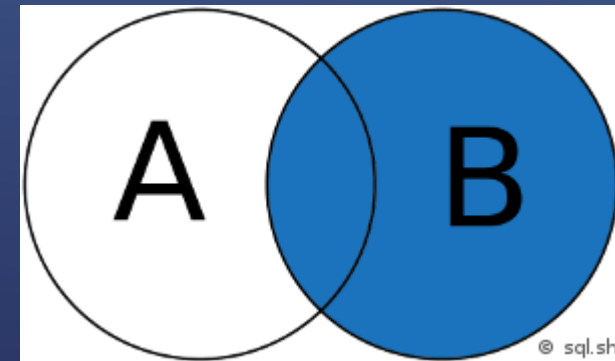
SQL : les requêtes de consultation

Les jointures

- ▶ RIGHT JOIN : C'est une jointure externe droite.

Cette commande retourne tous les enregistrements de la table B même si la condition n'est pas vérifiée dans l'autre table.

```
SELECT *  
FROM A  
RIGHT JOIN B  
ON A.key = B.key
```



SQL : les requêtes de consultation

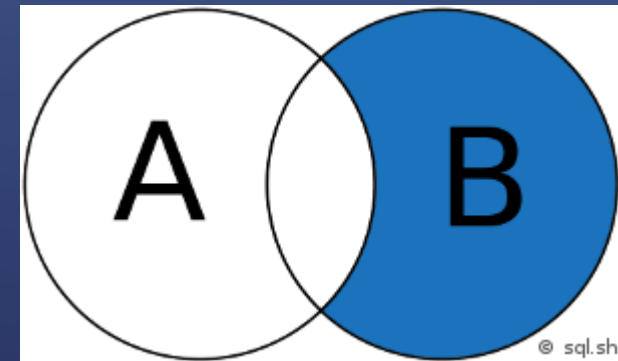
Les jointures

- ▶ RIGHT JOIN : C'est une jointure externe droite.

Cette commande retourne tous les enregistrements de la table B même si la condition n'est pas vérifiée dans l'autre table.

(sans l'intersection de A)

```
SELECT *  
FROM A  
RIGHT JOIN B  
ON A.key = B.key  
WHERE B.key IS NULL
```

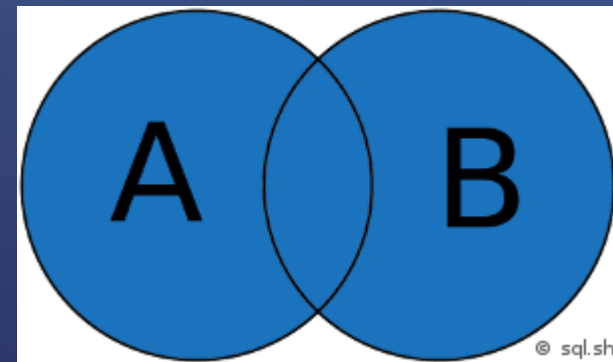


SQL : les requêtes de consultation

Les jointures

- FULL JOIN : permet de combiner les résultats de deux tables.

Attention : le FULL JOIN ne fonctionne pas dans MySQL



SQL : les requêtes de consultation

Les sous requêtes

Une sous requête consiste à exécuter une requête à l'intérieur d'une autre requête.

- Une requête imbriquée est souvent utilisée au sein d'une clause WHERE ou HAVING pour remplacer une ou plusieurs constantes.

```
SELECT * FROM table1
WHERE nom_attribut1 IN (
    SELECT nom_attribut2 FROM table2
    WHERE nom_attribut2= " valeur"
);
```