



Cursus JAVA

M2I Formations 2022

Alan Piron-Lafleur

MODULE SQL

Langage SQL ou PL/SQL

Alan Piron-Lafleur



1.

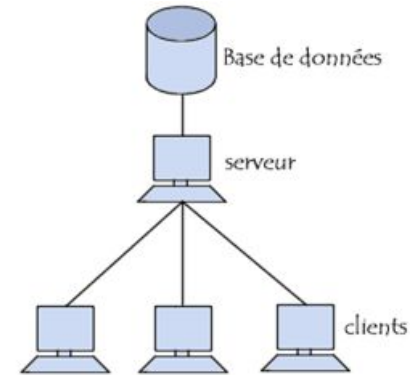
Base de données : Les concepts



Base de données

- **Ensemble structuré d'informations** (*d'une entreprise ou organisation*), **mémorisé** sur une machine (*serveur*).
- **Données stockées** et **organisées** sous forme de fichiers ou ensemble de fichiers.
- Une BDD sert à **créer, lire, modifier, et supprimer** des **données communes**.
(Create, Read, Update, Delete = CRUD)

- **Système de Gestion de Base de Données** (ou DBMS: Data Base Management System)
- **Ensemble cohérent de services** (*logiciels*) permettant aux utilisateurs **d'accéder, mettre à jour ou administrer** une DB
- Fonctionne sur le modèle **client/serveur** (requêtes/traitements)





Chronologie

Années 60:

- Apparition des premiers SGBD

Années 70:

- Ted Codd propose le modèle relationnel => 2ème generation Codd définit l'algèbre relationnelle (prémices du SQL)

Années 80:

- SGBD relationnel commercialisé (Oracle, SysBase, DB2...)

Années 90:

- SGBD relationnel dominant le marché
- Début des SGBD orientés objets

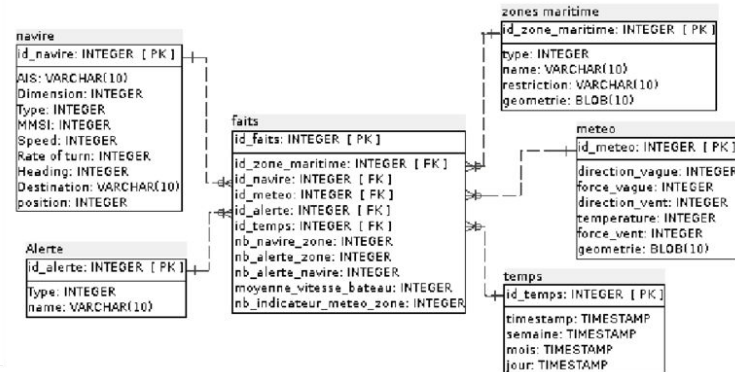


2.

Base de données : Modélisation

Modélisation

- Pourquoi modéliser ?
 - Avoir une représentation graphique de la structure
 - Connaître les propriétés attendues d'une donnée
 - Connaître les relations entre les données
- Comment modéliser ?
 - Effectuer un design conceptuel
 - Insérer des cardinalités
 - Effectuer un modèle logique
 - Grâce à l'UML (on en fera!)





Modélisation : Extraire l'information

- Lorsque vous allez recevoir un cahier des charges, il faudra tout d'abord réussir à en extraire les informations qui nécessitent un stockage persistant.
- Ces informations seront regroupées dans un document appelé le “**dictionnaire de données**”.
- **Objectif** : Avoir une structure des **tables** pour notre future base de données.

Nom donnée	Type donnée	Référence
nom_client	TEXT	Client
prenom_client	TEXT	Client
etc		



Modélisation : Les types de données

Dans une base de données il existe plusieurs **types** de données utilisables en fonction de la nature de l'information que l'on souhaite stocker :

INTEGER

FLOAT

BOOLEAN

CHAR

TIME

DATE

TIMESTAMP

VARCHAR



Modélisation : la clef primaire

- Le stockage d'informations en base doit être **unique** (une donnée = une entrée)
- On parle alors de “**clef primaire**” (ou Primary Key)
- Généralement au format numérique et nommée “**id**”.

id	prenom	nom
1	Chandler	Bing
2	Phoebe	Buffay
3	Monica	Geller
4	Ross	Geller
5	Chandler	Bing



Modélisation : les relations

- Une relation veut dire que des données sont liées.
Par exemple une table qui stock les clients et une table qui stock les commandes peuvent être liées.
- Il existe 3 types de relations :
 - un à un (one-to-one)
 - un à plusieurs (one-to-many) ou plusieurs à un (many-to-one)
 - plusieurs à plusieurs (many-to-many)

Chacun de ces types engendre une conséquence différente sur le modèle de données.
Nous allons les voir une par une.



Modélisation : les relations

Afin d'éviter les doublons de données il est possible de mettre en place des **relations** entre nos différentes **tables/entités**.

Par exemple :

id	nom	modele	annee_sortie
1	Jean Manchzeck	Kawasaki 750	2015
2	Edouard Bracame	Kawasaki 750	2015

Pourrait devenir :

id	nom	0,1	0,1	id	modele	annee_sortie
1	Jean Manchzeck	_____		1	Kawasaki 750	2015
2	Edouard Bracame					



Modélisation : les relations

One-to-One : Une relation one-to-one implique la création d'une **clé étrangère** dans l'une des deux tables. Cette clé représente la référence de la seconde table.

Dans cet exemple, un conducteur peut conduire ??? moto et une moto peut être conduite par ??? conducteur.

id	nom	id	modele	annee_sortie	conducteur
1	Jean Manchzeck				
2	Edouard Bracame				
		1	Kawasaki 750	2015	1
		2	Kawasaki 750	2015	2

Modélisation : les relations

One-to-Many : Une relation one-to-many implique également la création d'une **clé étrangère** dans l'une des deux tables. Par contre dans ce cas nous n'avons pas le choix de la table qui portera la référence. Si on reprend l'exemple précédent :

id	nom
1	Jean Manchzeck
2	Edouard Bracame



Dans cet exemple, un conducteur peut conduire ???? moto et une moto peut être conduite par ???? conducteur.

id	modele	annee_sortie	conducteur
1	Kawasaki 750	2015	1
2	Suzuki 800	2016	1



Modélisation : les relations

Many-to-Many : Une relation many-to-many engendrera la création d'une table de correspondance. Si on reprend l'exemple précédent :

id	nom	conducteur	moto
1	Jean Manchzeck	1	2
2	Edouard Bracame	2	2

Dans cet exemple, un conducteur peut conduire ??? moto et une moto peut être conduite par ??? conducteur.

id	modele	annee_sortie
1	Kawasaki 750	2015
2	Suzuki 800	2015



Les règles à respecter - diagramme de classe

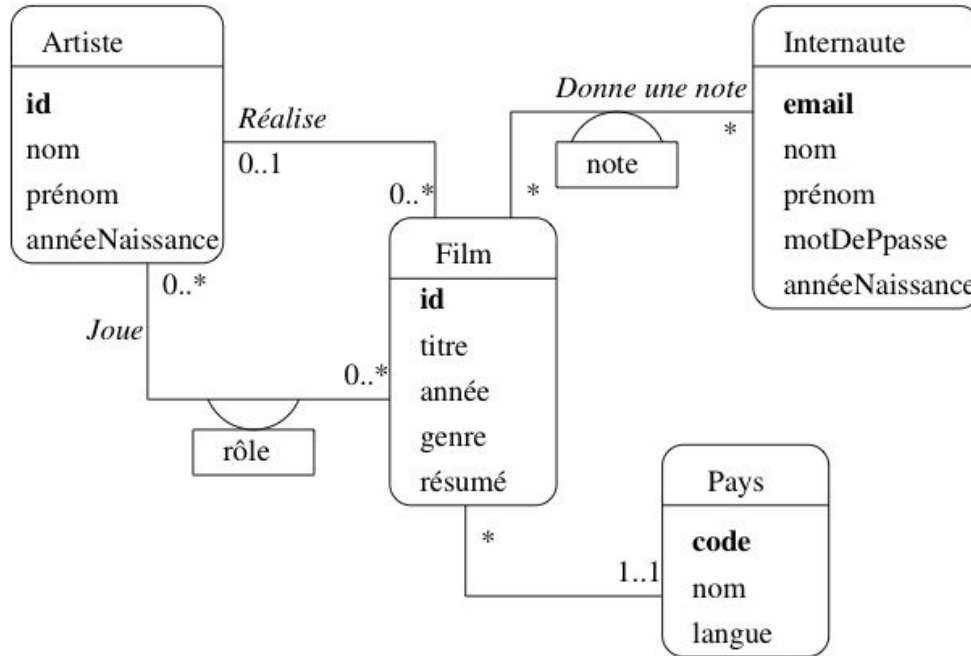
- **Normalisation des tables**
 - Ne contient pas d'espace, d'accents ni de caractères spéciaux
 - Ecriture en camelCase
 - La clé primaire porte un # (exemple : id#)
 - La portée des attributs est visible (+ ou - ou #)
 - Les clés étrangères ne sont pas visibles
 - Si une table associative n'a qu'un id, on ne le met pas (la table est vide)
- **Les relations**
 - Une relation permet de définir le type de lien entre 2 entités
 - Les relations "0,n"
 - Les relations "1,n"
 - Les relations "n, n"
- **Un identifiant**
 - Un champ d'identification unique est obligatoire, la clef primaire (souvent nommée "id").



Les règles à respecter - schéma de base de données

- **Normalisation des tables**
 - Ne contient pas d'espace, d'accents ni de caractères spéciaux
 - Tout doit être écrit en lowercase (minuscule)
 - Les espaces sont remplacés par des underscores : “_”
 - Les foreign key apparaissent
 - La clé primaire est notée “PK” dans une colonne à part
- **Les relations**
 - Une relation permet de définir le type de lien entre 2 entités
 - Les relations “0,n”
 - Les relations “1,n”
 - Les relations “n, n”
- **Un identifiant**
 - Un champ d'identification unique est obligatoire, la clef primaire (souvent nommée “id”).

Cardinalité





La modélisation UML

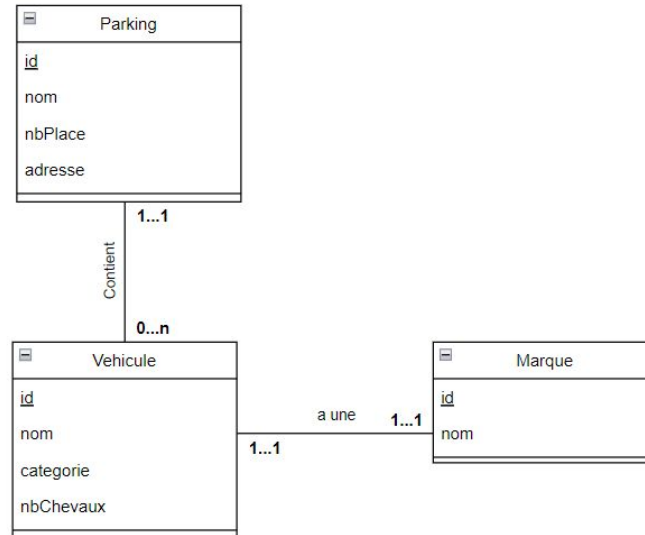
- **Comment utiliser UML**
 - Le langage UML permet de schématiser les différentes entités
 - UML à ses propres règles et codes pour représenter ces entités
 - On parle de diagramme de Classe
- **Les étapes ?**
 - L'objectif est de passer d'un texte (une spec, un cahier des charges...) à une représentation graphique

Exemple: du besoin à la modélisation

Besoin client :

Je souhaite avoir une modélisation représentant la gestion d'un parking de voiture.

Un parking à un nom, un nombre de place et une adresse. Le parking contient plusieurs véhicules, de différentes tailles.





Exercice 1

3.

Base de données : BDD relationnelles



Introduction

- **Qu'est-ce qu'une BDD relationnelle ?**
 - Concept basé sur le modèle de relation des données (Tables)
 - Organisation en colonnes et lignes
 - Attributs, données, tuples ...
- **SGBD couramment utilisés ?**
 - MySQL, PostgreSQL, Oracle, SQLite, Microsoft SQL Server

Propriété 1	Propriété 2	Propriété N...
Objet1, donnée1	Objet1, donnée2	Objet1, donnée N
Objet2, donnée1	Objet2, donnée2	...

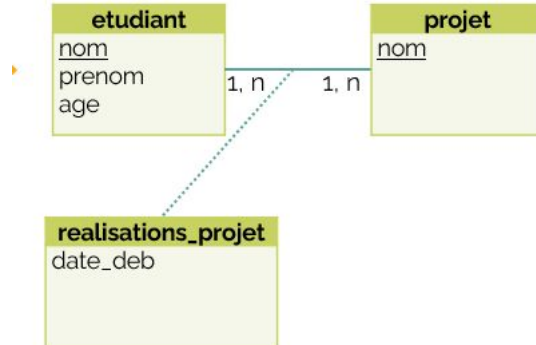


Type de données et contraintes

- **Types de données divisés en 3 catégories :**
 - Alphanumériques : VARCHAR, TEXT, BLOB ...
 - Numériques : INTEGER, FLOAT, DECIMAL, BOOLEAN ...
 - Temporels : DATE, DATETIME, TIME ...
- **Contraintes de données :**
 - Clés primaires □ Unicité (SERIAL)
 - Tailles de champs □ Min, max
 - Contrôles de champs □ Obligatoire, Défaut, facultatif
 - Clé étrangères □ Références aux autres tables
 - Et bien d'autres ...

Table d'association

- **Qu'est-ce qu'une table d'association ?**
 - On utilise une table d'association pour ajouter des informations entre 2 tables
 - La clé primaire d'une table d'association est le couple des clés primaires des 2 tables
 - Règle: La relation n-n crée une table d'association
- **Exemple :**





Draw.io – démo



TP 1 - part 1

On considère un site internet défini par une url unique et une langue autorisée. Ce site internet contient au moins une page internet.

Une page est présente uniquement sur un site et contient toujours un titre, un contenu, un nombre de ligne de code, une date de création et une date de mise à jour.

Réaliser le diagramme représentant les tables et leurs relations



4.

Base de données : Le SQL



Introduction

- SQL (Structured Query Language): Langage permettant d'interagir avec une base de données relationnelle.
- Langage introduit par IBM et commercialisé uniquement par Oracle dans un premier temps.
- Héritage du langage SEQUEL en 1977 reposant sur la théorie relationnelle proposée par Ted Codd.
- Terminologie définit par SQL:
 - Table*
 - Colonne*
 - Ligne*
 - Primary Key*
 - Foreign Key*
 - ...



Type de SGBD – Les principaux



ORACLE



PostgreSQL





Installation de MySQL

- Installation de MySQL via XAMPP

<https://www.apachefriends.org/fr/index.html>





☐ **Vous pouvez laisser les configurations de base**

Connexion via PHPMYADMIN

- Lancer MYSQL et APACHE

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

XAMPP Control Panel v3.3.0

Service	Module	PID(s)	Port(s)	Actions
	Apache	29464 29264	80, 443	<input type="button" value="Stop"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
	MySQL	26336	3306	<input type="button" value="Stop"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
	FileZilla			<input type="button" value="Start"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
<input type="checkbox"/>	Mercury			<input type="button" value="Start"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>
	Tomcat			<input type="button" value="Start"/> <input type="button" value="Admin"/> <input type="button" value="Config"/> <input type="button" value="Logs"/>

23:50:49 [Apache] Attempting to stop Apache (PID: 25708)
23:50:50 [mysql] Attempting to stop MySQL app...
23:50:50 [Apache] Status change detected: stopped
23:50:50 [mysql] Status change detected: stopped
23:50:52 [mysql] Attempting to start MySQL app...
23:50:52 [mysql] Status change detected: running
00:01:44 [Apache] Attempting to start Apache app...
00:01:44 [Apache] Status change detected: running



Connexion via PHPMYADMIN

- Lancer PHPMYADMIN en cliquant sur ADMIN (ligne MYSQL)



SQL: Les requêtes (CRUD)

Il existe plusieurs instructions possibles sur les **données** d'une base :

LES CRUD'S : vous vous en rappelez ?

- **INSERT** pour ajouter des lignes à une table
- **UPDATE** pour modifier des lignes d'une table
- **DELETE** pour supprimer des lignes d'une table
- **SELECT** pour extraire des données à partir de tables existantes



Types et contraintes

- Déclaration d'un type de données :
VARCHAR(*size*) , INTEGER, DECIMAL(*size*, *decimalSize*), TEXT, DATE, BOOLEAN
- Contraintes de données :
Clé primaire/Auto incrément : **PRIMARY KEY AUTO_INCREMENT**
Valeur par défaut: **DEFAULT**
Non null: **NOT NULL**
- Exemple :
`CREATE TABLE table1 (id INT, date DATE DEFAULT NOW());`



Clé étrangère

- Lorsque vous créez deux tables, pour les lier entre elle (one to many / one to one), il faut mettre en place une clé étrangère.
- Par exemple, si on a une table : conducteur et que l'on veut créer une table moto, on ferait :

```
CREATE TABLE moto(  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    id_conducteur INT NOT NULL,  
    nom VARCHAR(30) UNIQUE NOT NULL,  
    modele VARCHAR(50) NOT NULL,  
    CONSTRAINT fk_conducteur FOREIGN KEY(id_conducteur) REFERENCES conducteur(id)  
);
```



Dans l'onglet SQL de PHPMYADMIN

- Créer une base de données :
`CREATE DATABASE coursSQL;`
- Créer un utilisateur : il faut créer un utilisateur qui aura le droit d'accès à la BDD
`CREATE USER sqlUser;`
- Accorder des droits :
`GRANT ALL PRIVILEGES ON coursSql.* TO 'sqlUser'`
Exemple1 : `GRANT` all privileges `ON DATABASE` mabase `TO` user1;
Exemple2 : `GRANT` SELECT, UPDATE `ON` table1 `TO` user2;
- Cliquer sur la base de données courssql sur la gauche, aller dans l'onglet SQL puis :
Créer une table :
`CREATE TABLE user (id INT PRIMARY KEY AUTO_INCREMENT);`



Insertions et modifications

- Modifier une table : ajoutez une colonne nom et une colonne age
`ALTER TABLE nom_table ADD nom type;`
Exemple : `ALTER TABLE test ADD nom VARCHAR(255);`
- Insérer des données dans une table : insérez 3 lignes
/!\ l'ID se renseigne automatiquement
`INSERT INTO nom_table (col1, col3) VALUES (value1, value2);`
- Supprimer des données dans une table :
`DELETE FROM nom_table WHERE conditions;`
Exemple : `DELETE FROM table1 WHERE prenom = 'Jean';`
- Modifier des données dans une table :
`UPDATE nom_table SET col1=new_val, col2=new_val2, ... WHERE conditions;`
Exemple : `UPDATE table1 SET prenom='Jean-Michel' WHERE id = 1;`



TP 1 – part 2

On considère un site internet défini par une url unique et une langue autorisée. Ce site internet contient au moins une page internet.

Une page est présente uniquement sur un site et contient toujours un titre, un contenu, un nombre de ligne de code, une date de creation et une date de mise à jour.

Créer un utilisateur, une base de données et les tables correspondantes

Insérer 1 site avec 4 pages correspondantes



Requêtes (1/2)

- Requêter une table :
`SELECT nom_champ1, nom_ch2... FROM nom_table`
`SELECT * FROM nom_table`
- Ajouter une condition à une requête :
`SELECT * FROM nom_table WHERE nom_champ = x`
- Ajouter des restrictions (SELECT... FROM ...):

<code>LIMIT 3</code>	<input type="checkbox"/> Limiter résultat
<code>ORDER BY Nom</code>	<input type="checkbox"/> Ordonner pour un champs donné
<code>ORDER BY N DESC</code>	<input type="checkbox"/> Ordonner pour un champs donné inversé



Requêtes (2/2)

- Opérateurs classiques : `SELECT *FROM Table...`
`WHERE champ = "Représentant"`
`WHERE champ <> "Représentant"`
`WHERE champ = "Représentant" AND champ2 = "M." AND champ3 < 8;`

`WHERE champ BETWEEN 3 AND 8;`
`WHERE champ IN ('Mlle', 'Mme');`

`WHERE champ LIKE 'a%';`
`WHERE champ LIKE '%a%';`
- Sélection pour un champ renommé :
`SELECT ville AS liste_des_villes FROM Ville;`



TP 2

Contexte :

Un organisme de formation souhaite avoir une base de données.

Un organisme de formation contient 2 **centres de formations** (définis au minimum par un **nom**, une **adresse**, une **année** de construction et un ensemble de **classes**).

Une **classe** est définie par un **code** unique (ex: JAVA_001, PYTHON_015, ...), un nombre **d'apprenants** (4 par défaut) et un **formateur** (champs obligatoire).

Un **formateur** est défini par un **nom**, **prénom**, et un **taux journalier** (en euros). Le **taux journalier** par défaut est de 350€.

Besoin :

1. Réaliser le diagramme représentant les tables et les relations associées
2. Créer une base de données *organismeformation* et un utilisateur *admininformation* avec tous les droits sur la base.
3. Insérer les tables correspondantes au diagramme
4. Ajouter les données (2 centres de formations, 3 classes par centre et des formateurs qui interviennent sur plusieurs classes.



TP 2 - suite

Ecrire les requêtes SQL permettant d'afficher :

1. La liste des classes existantes
2. Les centres de formation, triée par année (plus ancien au plus récent)
3. Les noms des centres de formation pour lesquelles l'année est avant 2000
4. Le code et le nombre d'apprenant de la classe si ce code contient le mot 'java' ou 'Java' ou 'JAVA'
5. La liste des formateur pour qui le taux journalier est compris entre 300 et 500 €
6. Le code de la classe, triée par nombre d'apprenants décroissant, en limitant l'affichage à 3 lignes.



5.

Base de données : Le SQL avancé



Mise à jour des tables

- Modifier / supprimer une table :
`ALTER TABLE nom_table ADD nom type;`
`DROP TABLE nom_table;`
- Modification de colonne :
`ALTER TABLE nom_table CHANGE COLUMN ancien_nom nouveau_nom type;`



Imports / Exports

- Il est possible d'importer et exporter la structure d'une base de données, avec ses données : démo



TP 1 – part 3

Sur la base de données des sites internet, effectuez les requêtes pour :

- Modifier la table “*Page*” en ajoutant une nouvelle colonne “*path_photos*” de type varchar
- Modifier les données de la table “*Page*” en ajoutant une valeur par défaut “*no path*” pour la colonne “*path_photos*”
- Mettre à jour toutes les données de la table “*Page*” en mettant “*no path*” sur la colonne colonne “*path_photos*”
- Mettre à jour les données de la table “*Page*” pour la colonne “*nbLigne*” en mettant ‘250’ SI la Valeur enregistrée est supérieur à 250.

Effectuer les requêtes

Faire un export de la base

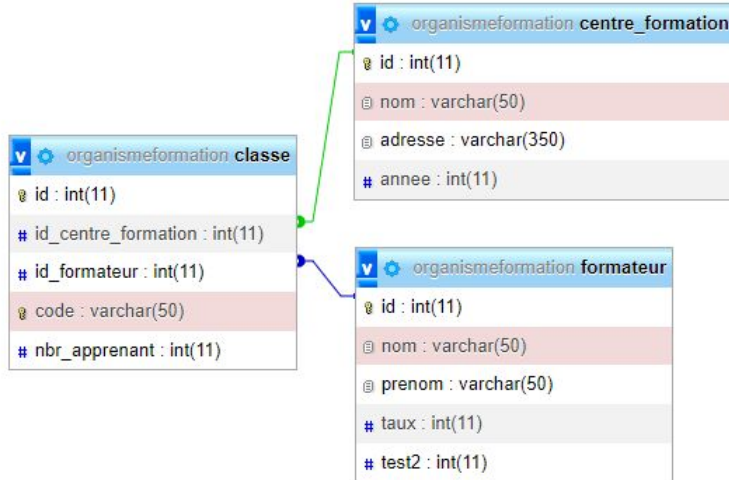
Faire une nouvelle base (site_internet_v2) et importer le fichier exporté.



Requêtes avec fonctions

- Syntaxe générale :
`SELECT function_name (column_name) FROM table_name`
- Fonctions principales :
 - COUNT:** Compte le nombre d'éléments.
 - AVG:** Calcul une moyenne.
 - MIN / MAX:** Identifie le plus petit/grand.
 - SUM:** Calcul la somme.
 - DISTINCT:** Ne pas compter les doublons.
- Exemple :
`SELECT COUNT(field3) FROM table1 GROUP BY field1;`
- Affichage / Regroupement :
 - GROUP BY:** Regrouper les valeurs identiques.
 - HAVING « *condition* »:** Limiter l'affichage

Requêtes avec jointures



```
SELECT * FROM `classe`  
INNER JOIN formateur  
ON classe.id_formateur = formateur.id
```



Création de vues

- Les vues:
Une vue est une « **table virtuelle** » et permet de **pré-enregistré** des requêtes
- Syntaxe :
`CREATE VIEW view_name AS « my_request »`

```
CREATE VIEW formateurs_chers AS  
SELECT * FROM formateur WHERE taux > 400
```

```
SELECT * FROM formateurs_chers
```



TP 2 – part 3

Sur la base données de l'organisme de formation, effectuez les requêtes suivantes :

1. Afficher le nombre total d'apprenant dans le centre de formation
2. Afficher le nombre de formateur, renommé en 'Nb_formateur'
3. Afficher le taux journalier moyen des formateurs
4. Afficher la liste des classes (code, nb apprenant) et le nom et prénom du formateur associé
5. Afficher la liste des classes (CODE) et le NOM du centre de formation correspondant, seulement si le nombre d'apprenant est supérieur à 10
6. Afficher le nom du centre de formation, le code de la classe et le nombre d'apprenant.
7. Créer une vue qui affiche liste les code formation et nom de formateur associé



SQL: Les jointures avec JOIN

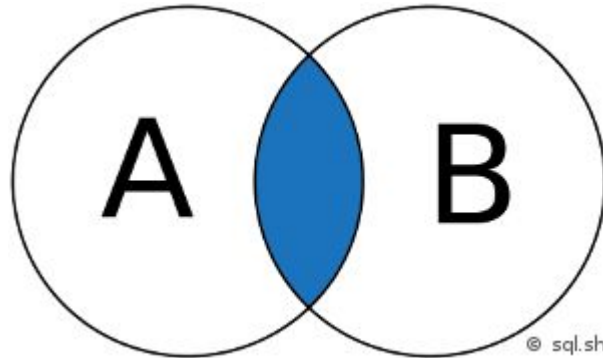
- Les jointures en SQL permettent d'associer plusieurs tables dans une même requête.
- Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.
- Il existe plusieurs types de jointure :

□ **INNER JOIN: Jointure quand la condition est vrai dans les 2 tables (PLUS COMMUN)**

SQL: Les jointures INNER JOIN

- Voici la syntaxe de l'INNER JOIN :

```
SELECT * FROM table1 INNER JOIN table2 ON table1.id = table2.fk_id;
```



SQL: Les jointures INNER JOIN

	id	nom	prenom	taux
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	Leblanc	José	400
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	May	Brian	350
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	Ribero	David	500

	id	id_centre_formation	id_formateur	code	nbr_apprenant
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	1	1	JAVA_001	15
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	4	2	SF-4550	10

```
SELECT * FROM formateur INNER JOIN classe ON formateur.id = classe.id_formateur
```

Résultat

id	nom	prenom	taux	id	id_centre_formation	id_formateur	code	nbr_apprenant
1	Leblanc	José	400	2	1	1	JAVA_001	15
2	May	Brian	350	3	4	2	SF-4550	10



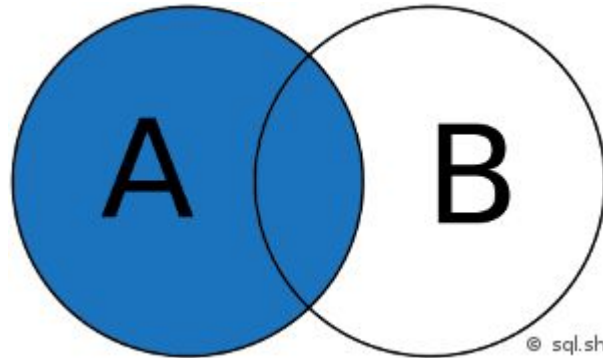
SQL: Les jointures avec JOIN

- Les jointures en SQL permettent d'associer plusieurs tables dans une même requête.
- Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.
- Il existe plusieurs types de jointure :
 - **INNER JOIN:** Jointure quand la condition est vrai dans les 2 tables (PLUS COMMUN)
 - **LEFT JOIN:** Jointure avec tous les enregistrements de la table gauche, même si la condition n'est pas vérifiée dans l'autre table.

SQL: Les jointures LEFT JOIN

- Voici la syntaxe du LEFT JOIN :

```
SELECT * FROM table1 LEFT JOIN table2 ON table1.id = table2.fk_id;
```



SQL: Les jointures LEFT JOIN

	id	nom	prenom	taux
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	Leblanc	José	400
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	May	Brian	350
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	Ribero	David	500

	id	id_centre_formation	id_formateur	code	nbr_apprenant
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2		1	1 JAVA_001	15
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3		4	2 SF-4550	10

```
SELECT * FROM formateur LEFT JOIN classe ON formateur.id = classe.id_formateur
```

Résultat

id	nom	prenom	taux	id	id_centre_formation	id_formateur	code	nbr_apprenant
1	Leblanc	José	400	2		1	1 JAVA_001	15
2	May	Brian	350	3		4	2 SF-4550	10
3	Ribero	David	500	NULL	NULL	NULL	NULL	NULL



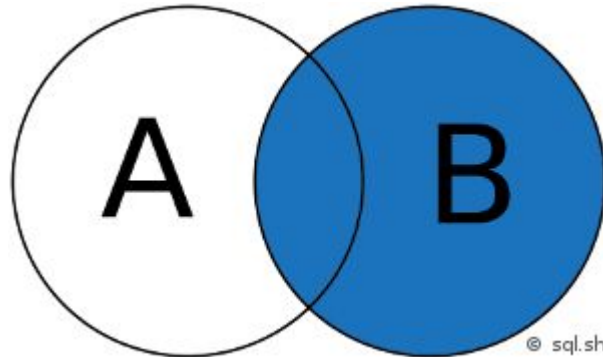
SQL: Les jointures avec JOIN

- Les jointures en SQL permettent d'associer plusieurs tables dans une même requête.
- Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.
- Il existe plusieurs types de jointure :
 - ❑ **INNER JOIN:** Jointure quand la condition est vrai dans les 2 tables (PLUS COMMUN)
 - ❑ **LEFT JOIN:** Jointure avec tous les enregistrements de la table gauche, même si la condition n'est pas vérifiée dans l'autre table.
 - ❑ **RIGHT JOIN:** Jointure avec tous les enregistrements de la table droite, même si la condition n'est pas vérifiée dans l'autre table.

SQL: Les jointures RIGHT JOIN

- Voici la syntaxe du RIGHT JOIN :

```
SELECT * FROM table1 RIGHT JOIN table2 ON table1.id = table2.fk_id;
```



SQL: Les jointures RIGHT JOIN

	id	nom	prenom	taux
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	Leblanc	José	400
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	May	Brian	350
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3	Ribero	David	500

	id	id_centre_formation	id_formateur	code	nbr_apprenant
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2		1	1 JAVA_001	15
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	3		4	2 SF-4550	10

```
SELECT * FROM formateur RIGHT JOIN classe ON formateur.id = classe.id_formateur
```

Résultat

id	nom	prenom	taux	id	id_centre_formation	id_formateur	code	nbr_apprenant
1	Leblanc	José	400	2		1	1 JAVA_001	15
2	May	Brian	350	3		4	2 SF-4550	10

FIN DU COURS DE SQL