SQL: Structured Query Language

@MOUSSA CAMARA

SQL: c'est quoi déjà ?

- Le SQL (Structured Query Language)en français Langage de requête structurée est un langage permettant de communiquer avec une base de données.

Mais c'est quoi une base de données ?

Réponse …

- Le concept **Base de Données** (BDD) apparaît vers 1960 suite au nombre croissant d'informations que les entreprises doivent gérer et partager (évolution dans les systèmes informatiques).
- Historiquement, chaque application engendrait ses propres fichiers de données exploités par ses propres programmes. La création d'une base de données va à l'encontre de cette façon de procéder ; elle rend possible la centralisation, la coordination, l'intégration et la diffusion de l'information archivée.

- une base de données est un ensemble structuré de données enregistré sur des supports accessibles par l'ordinateur, pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en un temps opportun.

SYSTEME DE GESTION DE BASE DE DONNEES (SGBD)

- C'est l'ensemble des programmes et des langages de commande (livrés par le constructeur ou la société qui commercialise le produit) qui permettent :
  - la définition des différents "ensembles de données" de la base, et les relations existant entre eux
  - le traitement, l'exploitation de ces données (interrogations, mises à jour, calculs, extractions...)
- NB: Le SGBD reçoit des commandes aussi bien des programmes d'application que des utilisateurs; il commande les manipulations de données, généralement par l'intermédiaire du SGF de base de l'ordinateur.

OBJECTIFS DE L'APPROCHE BDD

- Pour pallier les inconvénients des méthodes classiques (à base de fichiers), les bases de données tentent d'atteindre 4 objectifs principaux :
  - intégration et corrélation
  - flexibilité (indépendance)
  - Disponibilité
  - sécurité

A quoi ça sert le SQL ?

Comme dans la vie, pour que des personnes puissent se comprendre, elles doivent parler le même langage et bien en informatique, c'est pareil.

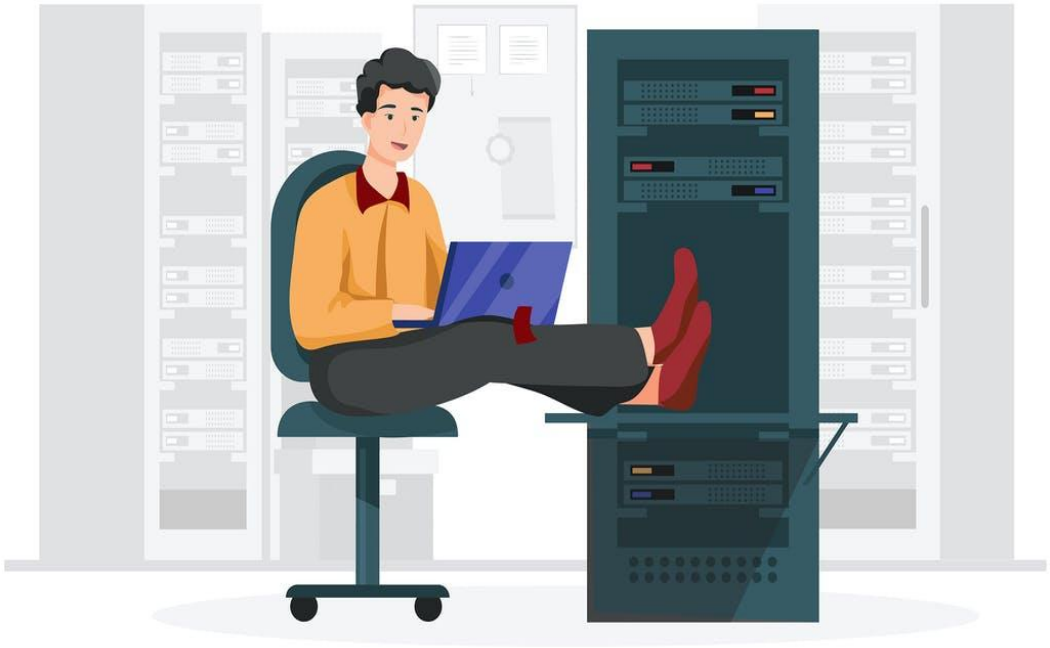
Pour que les différents logiciels et le moteur de base de données puissent se comprendre, ils utilisent un langage appelé SQL.

Pour que les différents logiciels et le moteur de base de données puissent se comprendre, ils utilisent un langage appelé SQL.

- Ce langage est complet. Il 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

A quoi ça sert le SQL ?

- En résumé:
  - Stocker les données
  - Manipuler les données
  - Integrer les données dans un programme informatique



1- Enregistrer une commande (table : commande)

2- Enregistrer un compte d'utilisateur (table : membre)

3- Enregistrer un message (table : commentaire)

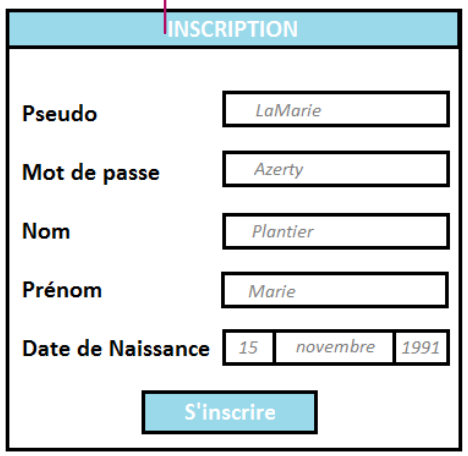
- Dans le contexte d'un site nous avons régulièrement besoin de garder des informations en mémoire.

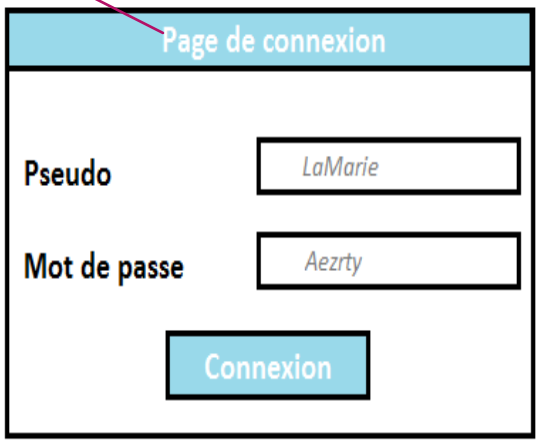
- *Acronyme*
  - MCD = **M**odèle **C**onceptuel de **D**onnées *(Plan de construction)*
  - SGBD = **S**ystème de **G**estion de **B**ase de **D**onnées *(Permet d'exploiter les données)*
  - BDD = **B**ase **D**e **D**onnées *(Emplacement des données sauvegardées)*
  - SQL = **S**tructured **Q**uery **L**angage *(Langages de requete : Structured Query Langage)*

- Lors de l'inscription, les comptes des membres sont enregistrés (INSERT) dans une table d'une base de données.

- Lors de la connexion, nous sélectionnons (SELECT) les informations en BDD pour vérifier que l'identification est bonne.

idMembre	pseudo	motDePasse	nom	prenom	dateDeNaissance
1	juju	soleil	thierry	julien	1985-05-22
2	LaMarie	Azerty	Plantier	Marie	1991-11-15
3	joker	planete123	Cottier	Jonathan	1995-01-30





Cas Concret: Etudions le cas d'une boutique ecommerce

Catalogue

- Les produits sont sélectionnés en BDD et affichés dans la page web.

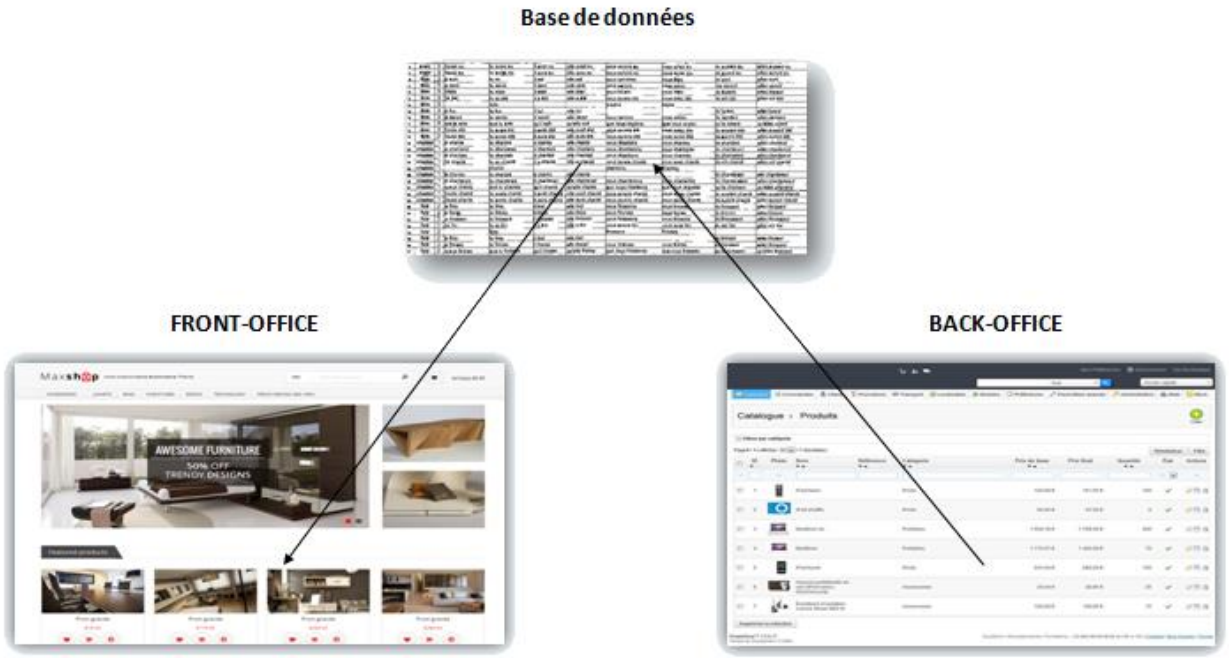
  Il y a 1 seule page fiche produit qui peut potentiellement présentés tous les produits (template).

Fiche Produit

Front & Back: Fonctionnement d'un site web

- Chaque site est scindé en deux parties :
- Front -> affichage.
- Back -> paramétrage.

- Ce système peut exister grâce à la présence d'une base de données.



- SQL fournit des instructions permettant de créer, supprimer, modifier, renommer des fichiers. Ces instructions sont les suivantes :
  - CREATE
  - DROP
  - ALTER
  - RENAME

Créer une BDD : CREATE database

- Pour créer une base donnée en ligne de commande:
- CREATE DATABASE ma_base

- CREATE DATABASE IF NOT EXISTS ma_base
- On peut aussi rajouter des options:
  - On peut rajouter l'encodage avec un UTF-8
  - CREATE DATABASE ma_base CHARACTER SET 'utf8';

Créer une table : CREATE TABLE

- La commande de création de tables la plus simple ne comportera que le nom et le type de chaque colonne de la table.
- À la création, la table sera vide, mais un certain espace lui sera alloué. La syntaxe est la suivante :
  - Ex: CREATE TABLE nom_table ()
- Supprimer une table : DROP TABLE
  - DROP TABLE nom_table
- Modifier une table : ALTER TABLE
  - Ex: ALTER TABLE nom_table {MODIFY} ()

- La syntaxe de base est:
- Create table: permet de créer une table en SQL
  - Ex: CREATE TABLE nom_table (nom VARCHAR(100), prenom VARCHAR(100))

- Alter table permet de modifier une table existante.
  - Ex: ALTER TABLE utilisateur *ADD/DROP/MODIFY* adresse_rue VARCHAR(255)
  - Pour renommer une colonne:
    - Ex: ALTER TABLE nom_table MODIFY nom_colonne type_donnees
- Drop table permet de supprimer une table
  - Ex: DROP TABLE nom_table

Utilisation

- Select: permet d'afficher les champs de la table
  - Ex: select id, nom, prenom ou select * (* Affiche tous les champs)
- From: extraire les données de la table
  - Ex: from Table
- Les options
  - Where condition
    - Ex: where id=2
  - And
    - Ex: and nom='moussa'
- Exemple complet:
  - Select * from user where id=2 and nom='moussa'; // Affiche tous les champs du user ayant id =2 et s'appelle moussa dans la table user

- Insert into: permet d'ajouter en base
  - Ex: insert into user (nom, prenom), ("Camara", "Moussa") // ajoute moussa camara dans user
- Update: permet de modifier les infos
  - Ex: update user SET prenom = "mouskito" WHERE nom="Camara" // Change leprenom moussa par mouskito
- Delete: permet de supprimer de la base
  - Ex: delete from user //Supprime toute la table user
- Delete from
  - Ex: DELETE FROM user WHERE nom="Camara" AND prenom="Moussa"

Exemple

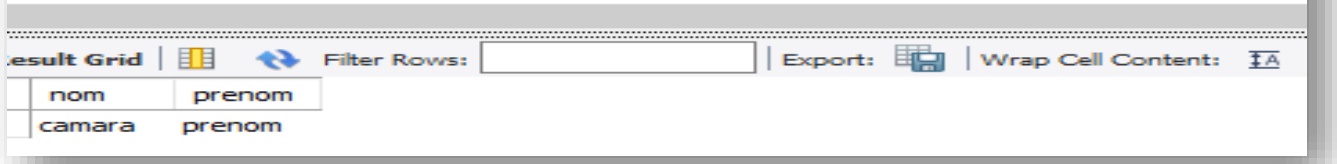
```sql
CREATE database test;

show databases;
use test;

create table user (nom varchar(50), prenom varchar(30));
use user;

insert into user(nom,prenom) value ('camara', 'prenom');

select * from user;
```



Exercice: insert into

- Créer les tables stagiaire et centre dans votre base de données.
- Ajouter les stagiaires:
  - Moussa camara
  - Jean Dupont
  - Zack Henri
  - Machin Bidule

- Ajouter les centres: avec le nom, le lieu et specialité
  - Afpa ,Paris, CDA
  - Afpa, Marseille, Dev Mobile
  - Afpa,Lyon, marketing

Les clauses

- GROUP BY : Cette clause permet de définir des groupes
  - Ex: SELECT * FROM stagiaire GROUP BY nom
- HAVING : Cette clause permet de spécifier un filtre
  - Ex: SELECT * FROM stagiaire GROUP BY nom HAVING nom LIKE 'camara'
  - // 1 Camara moussa
- UNION, INTERSECT et EXCEPT : Cette clause permet d'effectuer des opérations ensemblistes entre plusieurs résultats de requête
  - Ex: SELECT nom,penom FROM stagiaire UNION SELECT nom_centre,lieu FROM centre
- ORDER BY : Cette clause permet de trier les n-uplets du résultat
  - Ex: SELECT * FROM stagiaire ORDER BY penom

- LIMIT: permet de spécifier le nombre maximum de résultats que l'on souhaite obtenir
  - Ex: SELECT une_colonne FROM une_table LIMIT une_limite;
    - SELECT * FROM clients LIMIT 2;
      - SELECT * FROM stagiaire ORDER BY penom LIMIT 2

- OFFSET permet de ne pas retourner une partie des résultats de la requête.
  - Ex: SELECT * FROM clients LIMIT 2 OFFSET 1;
    - //sélectionner uniquement les 2 premiers résultats sans le tout premier.
      - Ex: SELECT * FROM stagiaire ORDER BY penom LIMIT 3 OFFSET 1

Expression de contraintes d'intégrité

- Une contrainte d'intégrité est une clause permettant de contraindre la modification de tables, faite par l'intermédiaire de requêtes d'utilisateurs, afin que les données saisies dans la base soient conformes aux données attendues.

- Ces contraintes doivent être exprimées dès la création de la table grâce aux mots clés suivants :
  - **NULL / NOT NULL** : précise si une valeur doit obligatoirement être saisie dans la colonne ou non
    - Exemple:

```
CREATE TABLE STAGIAIRE
( id            INTEGER     NOT NULL
 nom            VARCHAR(32) NOT NULL,
 prenom         VARCHAR(32) NULL,
 date_naissance DATE)
```

Crée une table dont les colonnes id et nom doivent obligatoirement être renseignés.

- DEFAULT : valeur par défaut qui est placée dans la colonne lors des insertions et de certaines opération particulières, lorsque l'on a pas donné de valeur explicite à la colonne

```
CREATE TABLE STAGIAIRE
(
        id              INTEGER     NOT NULL
        nom             VARCHAR(32) NOT NULL,
        prenom          VARCHAR(32) NULL,
        date_naissance DATE,
        genre           CHAR(1)     DEFAULT 'M'
)
```

Clé primaire

- Dans une base de données relationnelle, une clé primaire est la donnée qui permet d'identifier de manière unique un enregistrement dans une table.

- Une clé primaire PRIMARY KEY identifie de façon unique chaque enregistrement dans une table de base de données.
- La clé primaire doit contenir des valeurs uniques.
- Une colonne de clé primaire ne peut pas contenir des valeurs NULL.
- Chaque table doit avoir une clé primaire, et chaque table peut avoir qu'une seule clé primaire.

SQL contrainte PRIMARY KEY CREATE TABLE

Le SQL suivante lorsque le "Personnes" table est créée pour créer une contrainte PRIMARY KEY sur la colonne "p_id":

MySQL:

```
CREATE TABLE User
(
    user_Id int NOT NULL,
    nom varchar(255) NOT NULL,
    prenom varchar(255),
    addresse varchar(255),
    ville varchar(255),
    PRIMARY KEY (user_Id)
)
```

SQL PRIMAIRE de contrainte KEY ALTER TABLE

- Lorsque la table a été créée, créer la contrainte PRIMARY KEY pour une colonne dans "user_id", s'il vous plaît utiliser l'instruction SQL suivante:

- **MySQL / SQL Server / Oracle / MS Accès:**

  - ALTER TABLE Persons
    ADD PRIMARY KEY (user_Id)

- Pour nommer les contraintes de clé primaire, et de définir une pluralité de colonnes PRIMARY KEY, utilisez la syntaxe SQL suivante:

- **MySQL / SQL Server / Oracle / MS Accès:**
  - ALTER TABLE User
    ADD CONSTRAINT pk_UserID PRIMARY KEY (user_Id,nom)

- NOTE: Si vous utilisez l'instruction ALTER TABLE pour ajouter une clé primaire, vous devez déclarer la colonne de clé primaire ne contient pas une valeur NULL

Pour supprimer une contrainte PRIMARY KEY

- **MySQL:**
  - ```
    ALTER TABLE User
    DROP PRIMARY KEY
    ```

- **SQL Server / Oracle / MS Access:**
  - ```
    ALTER TABLE User
    DROP CONSTRAINT pk_UserID
    ```

Unicité (UNIQUE)

- La contrainte d'unicité exige que toutes les valeurs explicites contenues dans la colonne soient uniques au sein de la table.
- Exemple:

```
CREATE TABLE STAGIAIRE
(
    nom         VARCHAR(32) NOT NULL,
    prenom      VARCHAR(32) NULL,
    Email       varchar(50) UNIQUE
)
```

- INSERT INTO STAGIAIRE (nom, prenom, email) VALUES ('Camara', 'Moussa', 'test@test.fr')
- INSERT INTO STAGIAIRE (nom, prenom, email) VALUES ('Sultan', 'Vanessa', 'manager@afpa.fr')
- INSERT INTO STAGIAIRE (nom, prenom, email) VALUES ('Kasi', 'Zack',   'test@test.fr')

Error Code: 1062. Duplicate entry 'test@test.fr' for key 'stagiaire.email'

La contrainte CHECK

- La contrainte CHECK de validation est celle qui offre le plus de possibilité. En contre partie son exécution est très coûteuse.

- Elle permet de définir un prédicat complexe, basé sur une comparaison pouvant contenir une requête de type SELECT. Pour valider la contrainte, le prédicat doit être évalué à TRUE ou UNKNOWN (présence de NULL).

```
CREATE TABLE STAGIAIRE (
        nom VARCHAR(32) NOT NULL,
        prenom VARCHAR(32) NULL,
        genre CHAR(1),
        CHECK (genre IN ('M', 'F'))
);
```

Error Code: 3819. Check constraint 'stagiaire_chk_1' is violated.

La contrainte de FOREIGN KEY

- La contrainte de type FOREIGN KEY permet de mettre en place une intégrité référentielle entre une (ou plusieurs) colonnes d'une table et la (ou les) colonne composant la clef d'une autre table afin d'assurer les relations existantes et joindre les tables dans le requête selon le modèle relationnel que l'on a défini.

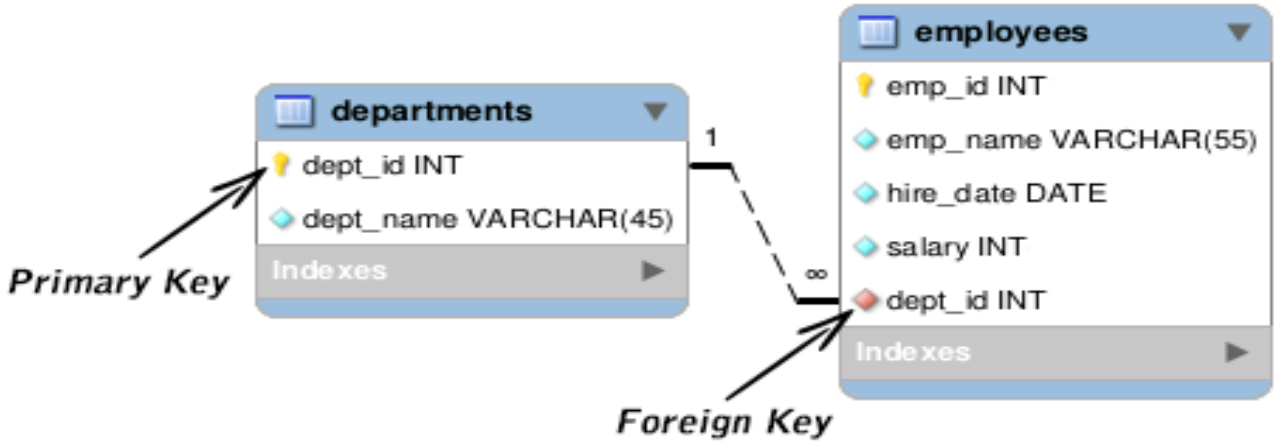
- Une clé étrangère dans un des points de table à une clé primaire dans une autre table.

- La contrainte FOREIGN KEY est utilisé pour prévenir les actions qui détruiraient les liens entre les tables.

- La contrainte FOREIGN KEY empêche également des données non valides d'être insérée dans la colonne de clé étrangère, car elle doit être l'une des valeurs contenues dans le tableau il pointe.

La contrainte de FOREIGN KEY

- Exemple:



Messages

```
Msg 547, Level 16, State 0, Line 2
The INSERT statement conflicted with the FOREIGN KEY constraint
"FK__CustomerO__Custo__6FE99F9F". The conflict occurred
in database "TestDB", table "dbo.Customers", column 'ID'.
The statement has been terminated.
```

Exercices

- Créer les tables suivantes:
- Etudiant
  - Matricule, nom, prenom, date de naissance, niveau
- Cours
  - Code, nom du cours, enseignant
- Examen
  - matricule, code, notes

Correction

- Correction en live

BETWEEN

- BETWEEN permet de sélectionner des enregistrements en fonction d'une intervalle
- SELECT une_colonne FROM une_table WHERE une_colonne BETWEEN "valeur_1" AND "valeur_2";

  - Ex: SELECT * FROM commandes WHERE date_ajout BETWEEN "2021-02-10" AND "2021-02-20";

IN

- IN permet de sélectionner des enregistrements si la valeur d'une colonne est comprise dans une liste de valeurs.
- Elle permet d'éviter d'avoir à recourir à de multiples OR.

- SELECT une_colonne FROM une_table WHERE une_colonne IN ("valeur_1", "valeur_2", "valeur_3");
  - Ex:
  - SELECT * FROM commandes WHERE id_commande = 3 OR id_commande = 4 OR id_commande = 5;
  - SELECT * FROM commandes WHERE id_commande IN(3, 4, 5);

NOT IN

- NOT IN permet de sélectionner des enregistrements si la valeur d'une colonne n'est pas comprise dans une liste de valeurs.

- Ex: sélectionner les commandes qui n'ont pas les identifiants 1 ou 5.

  - SELECT * FROM commandes WHERE id_commande NOT IN(1, 5);

Les opérateurs

- L'opérateur étoile (*) permet de récupérer automatiquement tous les attributs de la table générée par la clause FROM de la requête.
  - SELECT * FROM user

- Lorsque le SGBD construit la réponse d'une requête, il rapatrie toutes les lignes qui satisfont la requête, généralement dans l'ordre ou il les trouve, même si ces dernières sont en double (comportement ALL par défaut).
- C'est pourquoi il est souvent nécessaire d'utiliser le mot-clé DISTINCT qui permet d'éliminer les doublons dans la réponse.

  - SELECT DISTINCT nom FROM user

Les opérations mathématiques de base

- Les maths …(encore, ils sont partout) 

- Il est possible d'utiliser les opérateurs mathématiques de base (+, -, * et /) pour générer de nouvelles colonnes à partir, en général, d'une ou plusieurs colonnes existantes.
  - Ex: SELECT nom, prenom, salaire*12 FROM employee

- **L'opérateur AS**
  - Le mot-clé AS permet de renommer une colonne, ou de nommer une colonne créée dans la requête.
    - Ex1: SELECT surname AS nom, name AS prénom FROM user
    - Ex2: FROM nom_de_table AS nouveau_nom

Operation de comparaison

=	égal
!=	différent
<	strictement inférieur
<=	inférieur ou égal
>	strictement supérieur
>=	supérieur ou égal

Operation de comparaison

- Un prédicat simple peut également correspondre à un test de description d'une chaîne de caractères par une expression régulière :

~	décrit par l'expression régulière
~*	comme `LIKE`, mais sans tenir compte de la casse
!~	non décrit par l'expression régulière
!~*	comme `NOT LIKE`, mais sans tenir compte de la casse

L'opérateur LIKE 'chaîne' permet d'insérer des jokers dans l'opération de comparaison (alors que l'opérateur = teste une égalité stricte)

Exemple de LIKE

- LIKE est inséparable des caractères jokers (wildcards) : % (pourcentage) et _ (underscore).
- LIKE est un opérateur SQL qui permet d'effectuer une comparaison partielle, ce qui peut être une utile pour effectuer une recherche dans la base de données.

- % : il représente aucun, un seul ou plusieurs caractères. Il peut remplacer n'importe qu'elle chaîne de caractères
- Joker _ : il représente un seul et unique caractère. Il remplace un seul et unique caractère.

  - Ex: SELECT * FROM clients WHERE email LIKE "%@yeah.com";
  - SELECT * FROM clients WHERE nom LIKE "_e%";

Exemple de LIKE

LIKE "a%"	Recherche toutes les chaînes de caractères qui **commencent par le caractère *a***.
LIKE "%a"	Recherche toutes les chaînes de caractères qui **terminent par le caractère *a***.
LIKE "%a%"	Recherche toutes les chaînes de caractères qui **contiennent au moins un caractère *a***.
LIKE "a%b"	Recherche toutes les chaînes de caractères qui **commencent par le caractère *a* et terminent par le caractère *b***.
LIKE "a__"	Recherche **toutes les chaînes de caractères de trois caractères qui commencent par le caractère *a***.
LIKE "_a%"	Recherche toutes les chaînes de caractères qui **possèdent le caractère *a* en deuxième position**.

Exemple

- EX: SELECT nom FROM stagiaire WHERE nom LIKE 'A%'
  - recherche tous les noms commençant par le caractère "A", il faut donc écrire "A%" pour que % soit égale à n'importe quelle suite de caractères.
- SELECT nom FROM stagiaire WHERE nom LIKE '%ma%'
  - cherche tous les noms contenant la chaîne de caractères 'ma',
- SELECT nom FROM stagiaire WHERE nom LIKE 'ca____%'
  - Recherche des stagiaires dont le prénom comporte exactement 6 caractéres et qui commence par ca
- SELECT * FROM stagiaire WHERE prenom LIKE "____";
  - Recherche des stagiaires dont le prénom comporte exactement 4 caractéres

Operation de comparaison

- Un prédicat simple peut enfin correspondre à l'un des tests suivants :

expr IS NULL	test sur l'indétermination de `expr`
expr IN (expr_1 [...])	comparaison de `expr` à une liste de valeurs
expr NOT IN (expr_1 […])	test d'absence d'une liste de valeurs
expr IN (requête) expr NOT IN (requête)	même chose, mais la liste de valeurs est le résultat d'une sous-requête qui doit impérativement retourner une table ne contenant qu'une colonne
EXIST (requête)	*vraie* si la sous-requête retourne au moins un n-uplet
expr operateur ANY (requête)	*vraie* si au moins un n-uplet de la sous-requête vérifie la comparaison « `expr opérateur` n-uplet » ; la sous-requête doit impérativement retourner une table ne contenant qu'une colonne ; `IN` est équivalent à `= ANY`
expr operateur ALL (requête)	*vraie* si tous les n-uplets de la sous-requête vérifient la comparaison « `expr opérateur` n-uplet » ; la sous-requête doit impérativement retourner une table ne contenant qu'une colonne

Opération de logique

- OR
  - Ex: SELECT une_colonne FROM une_table WHERE une_condition OR encore_condition;
- AND
  - Ex: SELECT une_colonne FROM une_table WHERE une_condition AND une_autre_condition;
- NOT
  - Ex: SELECT une_colonne FROM une_table WHERE une_condition AND NOT une_autre_condition;

EXERCICE

- Créer les tables:
  - Avion
    - NumeroAv, NomAv, Capacité, Localisation
  - Vol
    - NumVol, NumPil, NumAv, Ville_depart, Ville_arrive, Heure_dep, heure_arrivee
  - Pilote
    - NomPil, NumPil, Adresse, Salaire

- Insert into
  - Avions
    - Avion 1: 00001, Alpha, 300, Paris
    - Avion 2: 00002, Delta,400, Milan
    - Avion 3: 00002, Delta,350, Rome
  - Vols :
    - Vol 1 : AF351, 152010, 00001, Paris, new York, 9h00, 17h00
    - Vol 2 : RU154, 458950, 00002, Canada, Londres, 13h00, 22h00
    - Vol 3 : LH1792, 538710, 00003, Paris, Tunis 11h00, 21h00
  - Pilotes:
    - Pilote1: Moussa, 00001, Paris, 650 000
    - Pilote2: Afpa, 00003, Marseille, 900 000
    - Pilote3: Zack, 00002, Cergy, 750 901
    - Pilote4: Zack, 00002, Cergy, 350 000

Les types de données

- Les types de données peuvent être :
  - INTEGER :
    - Ce type permet de stocker des entiers signés codés sur 4 octets.
  - BIGINT :
    - Ce type permet de stocker des entiers signés codés sur 8 octets.
  - REAL :
    - Ce type permet de stocker des réels comportant 6 chiffres significatifs codés sur 4 octets.
  - DOUBLE PRÉCISION :
    - Ce type permet de stocker des réels comportant 15 chiffres significatifs codés sur 8 octets.
  - NUMERIC[(précision, [longueur])] :
    - Ce type de données permet de stocker des données numériques à la fois entières et réelles avec une précision de 1000 chiffres significatifs. longueur précise le nombre maximum de chiffres significatifs stockés et précision donne le nombre maximum de chiffres après la virgule.

Correction

- Correction en live

Les types de données

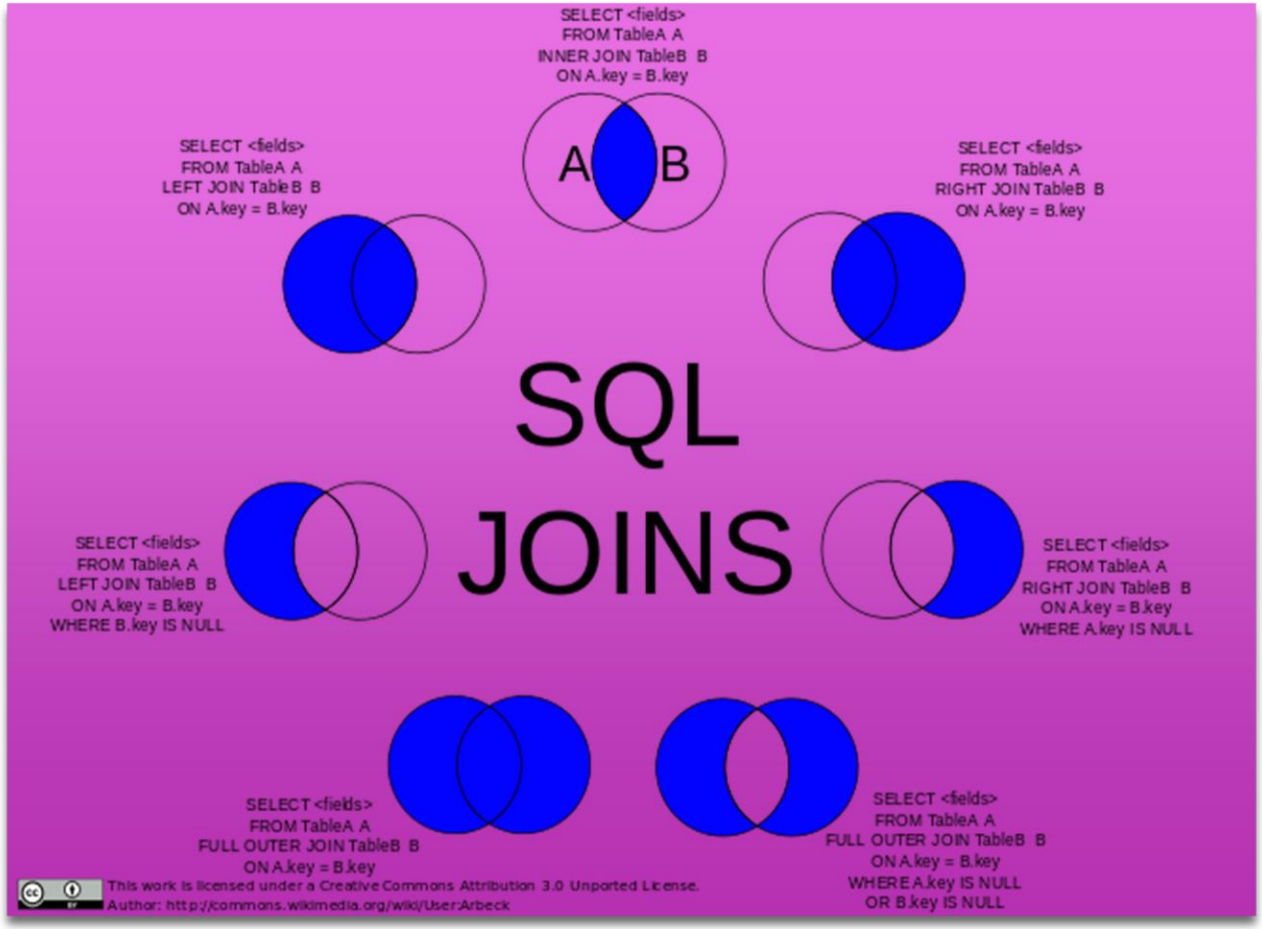
- CHAR(longueur) :
  - Ce type de données permet de stocker des chaînes de caractères de longueur fixe. longueur doit être inférieur à 255, sa valeur par défaut est 1.
- VARCHAR(longueur) :
  - Ce type de données permet de stocker des chaînes de caractères de longueur variable. longueur doit être inférieur à 2000, il n'y a pas de valeur par défaut.
- DATE :
  - Ce type de données permet de stocker des données constituées d'une date.
- TIMESTAMP :
  - Ce type de données permet de stocker des données constituées d'une date et d'une heure.
- BOOLEAN :
  - Ce type de données permet de stocker des valeurs Booléenne.
- MONEY :
  - Ce type de données permet de stocker des valeurs monétaires.
- TEXT :
  - Ce type de données permet de stocker des chaînes de caractères de longueur variable.

- Il existe une notion simple en apparence, mais assez puissante pour manipuler des bases de données. Ce sont les jointures

C’est quoi une jointure SQL déjà ?

- Une jointure permet de créer une liaison entre deux ou plusieurs tables pour pouvoir récupérer un résultat bien précis.

- Ce résultat est en quelque sorte une fusion virtuelle de plusieurs tables. Chaque entrée de chaque table est retournée dans le résultat en fonction de la jointure utilisée, ça peut vite faire beaucoup de lignes !



Les différents types de jointures SQL

- Il existe plusieurs types de jointures et il est facile de s'emmêler les pinceaux lorsqu'on commence à faire des requêtes SQL complexes avec des jointures en tout genre.

SQL – les jointures

- A venir …