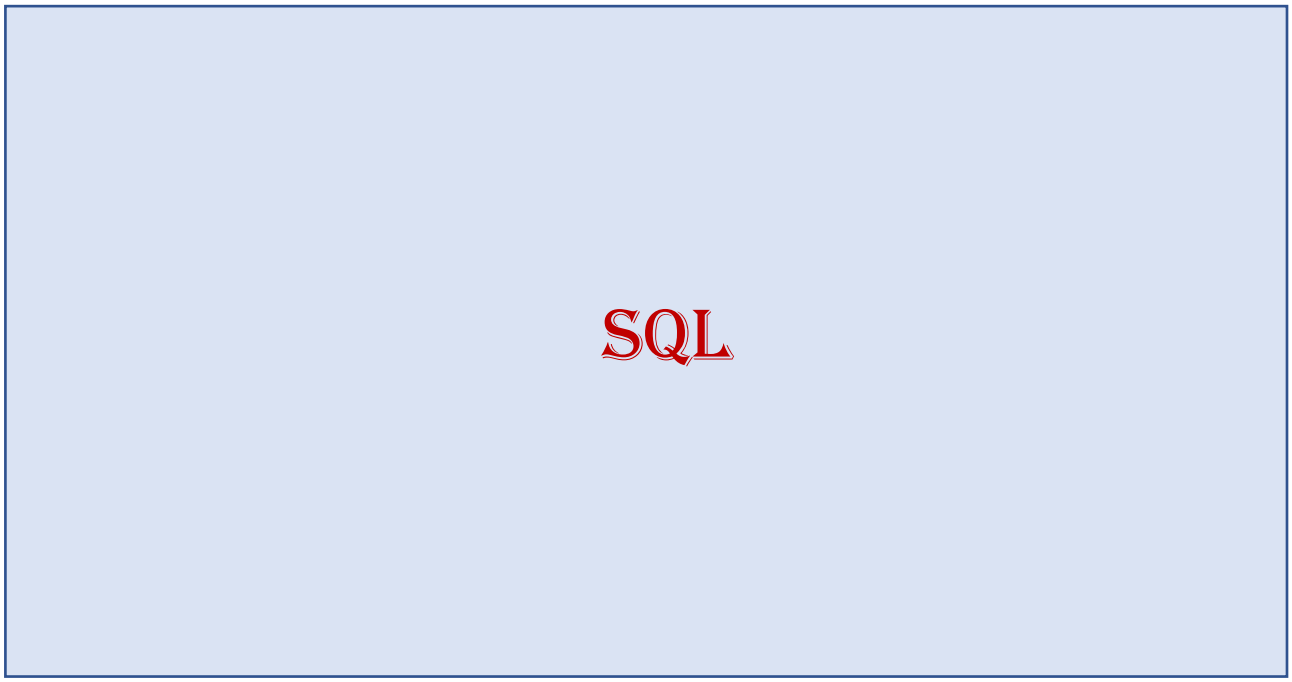
Filières : Licence 1 – IGL/RIT



PROFESSEUR : M. KONATE Ibrahima

- **Objectif de SQL :**
  - **Créer une structure de données :**
    - **Composées de tables (ou d'entités), elles même composées d'attributs ou de champs ayant un type**

Exemple : Un client est une table de nom personne qui possède un champ « prénom » de type « texte », un champ nom de même type, un champ « date de naissance » de type « Date », etc.

- **Relier les tables et leur champ entre eux**

Exemple : Un client peut faire plusieurs commandes ;

Une commande est associée à un seul client ;

Pour associer une commande à un client, la table commande possède comme un champ un numéro de client, identifiant le client en question.

- **Exécuter les tâches de base de la gestion des données telle que l'insertion, la modification, la suppression de données.**
- **Effectuer des requêtes simples ou complexes.**

- **SQL : Structured Query Language**
  - **Origine : IBM, System R, milieu années 70**
  - **Implémenté dans de nombreux SGBD (Système de Gestion de Base de Données)**

- **Fonctionnalités :**
  - **Définition des objets de la base de données (LDD : Langage de Définition de Données)**
  - **Manipulation des données (LMD : Langage de Manipulation de Données)**

- **Contrôles des accès aux données**
- **Gestion de transactions**

- **Principales instructions**
  - **Définitions (LDD) :**
    CREATE, DROP, ALTER
  - **Mise à jour (LMD) :**
    INSERT, UPDATE, DELETE
  - **Interrogations (LMD) :**
    SELECT
  - **Contrôle d'accès aux données :**
    GRANT, REVOKE
  - **Gestion de transactions :**
    COMMIT, ROLLBACK

- **Manipulation base de données :**
  - **Créer une base de données :**
    CREATE DATABASE [IF NOT EXISTS] nameDB;
  - **Supprimer une base de données :**
    DROP DATABASE nameDB ;

- **Gestion des privilèges :**
  GRANT ALL PRIVILEGS ON nameDB TO 'user'@localhost IDENTIFIED BY 'user'
- **Manipulation table :**
  - **Créer une table :**

    CREATE TABLE produit (reference INT, nom VARCHAR(30),marque VARCHAR(20),constructeur CHAR(20)) ;

  - **Déclarer un champ comme devant toujours avoir une valeur**
    reference INT NOT NULL
  - **Déclarer un champ qui s'auto-incremente :**
    reference INT NOT NULL AUTO_INCREMENT

- **Définir une valeur par défaut pour un champ :** ***Adresse VARCHAR(30) DEFAULT 'inconnue'***
- **Créer une table :**
  - **Identification des tuples :** ***PRIMARY KEY (att1,att2,…)***
  - **Attributs figurants dans une clé : forcément** ***NOT NULL***

```
CREATE TABLE produit(reference INT NOT NULL, nom
VARCHAR(30) NOT NULL, marque VARCHAR(20), constructeur
CHAR(20) DEFAULT 'Renault', PRIMARY KEY(reference));
```

  - **Unicité :**
    - ***Unique(att1,att2,…)***
    - **Principalement pour les clés secondaires**
    - **Unique ne s'applique pas aux valeurs nulles**

```
CREATE TABLE produit(reference INT NOT NULL, nom
VARCHAR(30) NOT NULL, prenom VARCHAR(30) NOT NULL,
marque VARCHAR(20), constructeur CHAR(20) DEFAULT
'Renault', PRIMARY KEY(reference),UNIQUE(nom,prenom));
```

  - **Clé étrangères :**
    FOREIGN KEY(att1,att2,…) REFERENCES …

```
CREATE TABLE vente(numero INT NOT NULL,ref_produit
CHAR(4)NOT NULL, ref_client INT NOT NULL, date DATE
,PRIMARY KEY(numero), FOREIGN
KEY(ref_produit)REFERENCES produit(reference), FOREIGN
KEY(ref_client) REFERENCES client(num);
```

  - **Clé étrangères, gestion de l'intégrité référentielle :**
    -Objectif vérifier que les valeurs de la clé étrangère correspondant bien à une clé primaire de la table référencée
    -Décider quoi si l'intégrité référentielle n'est pas vérifiée : par défaut rejet de l'opération

- **Moment de vérification :**

  - ***ON UPDATE(en cas de mise à jour de la clé primaire référencée)***

  - ***ON DELETE(en cas de suppression d'un tuple référencé)***

Que faire en cas de non-respect de la contrainte ?

- ***SET NULL : clé étrangère mise à NULL***

- ***CASCADE : application de la même opération***

- ***SET DEFAULT : valeur par défaut pour la clé étrangère***

- ***Clé étrangères, gestion de l'intégrité référentielle :***
  - ***Si suppression Produit, ref_produit=NULL***
  - ***Si mise à jour Produit, ref_produit=mise à jour***

```
CREATE TABLE vente( numero INT NOT NULL, ref_produit
CHAR(4) NOT NULL, ref_client INT, date DATE, PRIMARY
KEY(numero),FOREIGN KEY(ref_produit) REFERENCES
produit(reference) ON DELETE CASCADE ON UPDATE,FOREIGN
KEY(ref_client)REFERENCES client(num)ON DELETE SET NULL
ON UPDATE CASCADE);
```

- **Supprimer une table:**
  DROP TABLE nameTable;
- **Modifier une table**
  - ***ALTER TABLE nameTable ACTION description***
    ADD, MODIFY, ALTER, DROP , RENAME

```
ALTER TABLE produit MODIFY nom
VARCHAR(40) ;
ALTER TABLE produit ADD quantite_stock
INT ;
```

```
ALTER TABLE client ALTER adresse SET
DEFAULT 'NICE';
ALTER TABLE client DROP adresse ;
ALTER TABLE client ADD CONSTRAINT ct_name
UNIQUE (nom) ;
ALTER TABLE client ADD CONSTRAINT ct_pr
PRIMARY KEY (numero) ;
ALTER TABLE vente ADD CONSTRAINT
ct_fk_client FOREIGN KEY (ref_client)
REFERENCES client(num) ;
```

- **Mise à jour des données :**
  INSERT INTO : **pour insérer des tuples**
  DELETE FROM : **pour supprimer des tuples**
  UPDATE : **pour mettre à jour des tuples**

Insertion avec désignation des colonnes :

```
INSERT INTO
produit(nom,marque,constructeur)
VALUES('Auto model
T','AutoSpeed','AutoSpeed') ;
```

Insertion sans désignation des colonnes :

```
INSERT INTO produit VALUES(DEFAULT,'Auto
Model T','AutoSpeed','AutoSpeed');
```

Suppression de tous les tuples :

```
DELETE FROM produit
```

Suppression conditionnelle :

```
DELETE FROM produit WHERE marque='AutoSpeed'
```

Mise à jour des tuples :

```
UPDATE produit SET nom='Auto model X' WHERE
reference=153
```

▪ Consultation des données

Client			
numéro	**nom**	**adresse**	**téléphone**
101	Durand	Nice	034553324
106	Fabre	Paris	NULL
110	Nassim	Paris	NULL
125	Antonin	Marseille	032234555

Produit		
référence	**marque**	**prix**
153	PWM	8000 €
589	TransElec	3465 €
158	Yadus	12045€
553	Citron	7000 €

Vente			
numéro	**ref_produit#**	**no_client#**	**date**
102	153	101	12/10/2014
845	589	106	15/03/2015
1003	158	106	03/11/2016
1058	589	125	23/07/2017

▪ Format des requêtes

- **FROM** spécifie la table ou les tables à utiliser
- **WHERE** filtre les lignes selon une condition donnée
- **GROUP BY** forme des groupes de lignes de même valeur de colonne
- **HAVING** filtre les groupes sujets a` une certaine condition
- **SELECT** spécifie les colonnes qui doivent apparaître dans les résultats
- **ORDER BY** spécifie l'ordre d'apparition des données dans le résultat

▪ Requêtes simples (SELECT-FROM)

TRAVAUX DIRIGES :

Reproduire les tables client, produit et vente

1-Afficher le nom et l'adresse des clients

2-Afficher toutes les informations des clients
3-Afficher l'adresse de tous les clients
4-Afficher toutes les adresses existantes (sans doublons)

- **Sélection de colonne (clause WHERE)**
  - Les conditions fondamentales de recherche
    - Comparaison : (salaire>10000, ville='Paris')
    - Etendue ou intervalle : salaire BETWEEN 20000 and 30000
    - Appartenance à un ensemble : couleur IN ('red','green')
    - Correspondance à un masque : adresse LIKE '%Montréal%
    - Nul : adresse IS NULL

- **Opérateur de sélection**

5-Quels sont les clients dont l'adresse est paris ?
6-Quels sont les produits dont le prix TTC est supérieur à 1000
TTC=prix_HT + prix_HT*TVA avec TVA=0.05

- **Opérations possibles(MySQL)**
  - **Booléennes**
    And, or, xor, =, !=, <, >, <=, >=
  - **Arithmétiques**
    +, -, *, /
    +, - (opérateur unaires)
  - **Fonctions numériques**
    Abs, log, cos, sin, mod, power…
  - **Fonctions sur les chaînes**
    Length, concat, …

- Dans SELECT : l'attribut est calculé et affiché dans le résultat
- Dans WHERE : vous participer aux critères de de la sélection

- **Priorité des opérateurs**
  - :=
  - || , OR,XOR
  - && , AND
  - BETWEEN, CASE, WHEN, THEN, ELSE
  - =, <=> ,>=, >,<=,<,<>,!=,IS,LIKE,REGEXP,IN
  - |
  - &
  - <<,>>
  - -,+
  - *,/,DIV,%,MOD
  - ^
  - -(unary minus),
  - !,NOT
  - BINARY, COLLATE

```
SELECT 1+2*3;
7
```

- **SELECT** ABS (-32); 32
- **SELECT** FLOOR (1.23); 1
- **SELECT** MOD (234, 10); 4
- **SELECT** 253 % 7; 1
- **SELECT** ROUND (1.298, 1); 1,3
- **SELECT** ROUND (1.298, 0); 1
- **SELECT** SIGN (234) SIGN (-32) SIGN (0); 1 / -1 / 0
- **SELECT** 3 / 5; 0,60

Bien sur on peut utiliser des attributs

Peut se trouver dans Where

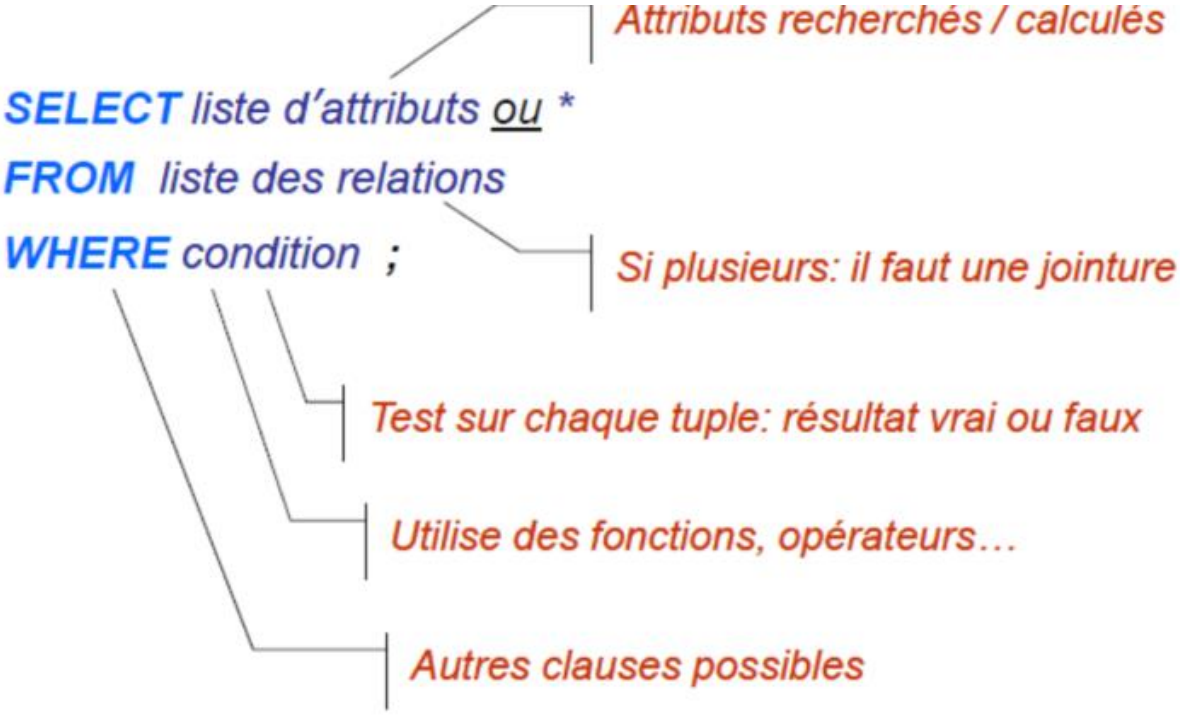
WHERE 3 / 5 < 1;

- **SELECT** CONCAT ('*My*', '*S*', '*QL*'); 'MySQL'
- **SELECT** CHAR_LENGTH (*'MySQL'*); 5
- **SELECT** LOCATE ('bar', 'foobarbar'); 4
- **SELECT** LOCATE ('bar', 'foobarbar', 5); 7
- **SELECT** INSERT ('Quadratic', 3, 4, 'What'); 'QuWhattic'
- **SELECT** LOWER (*'MySQL'*); 'mysql'
- **SELECT** SUBSTRING ('Quadratically',5,6); 'ratica'
- **SELECT** 'David!' LIKE 'David_'; 1
- **SELECT** 'David!' LIKE '%D%v%'; 1
- **SELECT** STRCMP (S1, S2); -1,0,1

_ et %

0 (false), **1** (true)

- **Combinaison Sélection + projection**



- **Requêtes simples**
  - Y a-t-il des produits dont le nom est « XBOX »

```
SELECT *
FROM Produit
WHERE nom = 'XBOX' ;
```

$$\sigma_{nom\ =\ 'XBOX'}(Produit)$$

  - Quels sont les ventes réalisées il y a plus de 30 jours

```
SELECT *
FROM Vente
WHERE CURRENT_DATE () > 30 + date ;
```

$$\sigma_{date\ +\ 30j\ <\ AUJOURD'HUI}(Vente)$$

  - Quels sont les ventes faites après le 1er janvier 2007 ?

```
SELECT *
FROM Vente
WHERE date > DATE ('2007-01-01') ;
```

$$\sigma_{date > '01\_01\_2007'}(Vente)$$

- Quels sont les ventes dont le montant HT est entre 1000 et 3000 euros et dont le client n'est pas le numéro 101 ?

```
SELECT *
FROM Vente
WHERE (prix_ht between 1000 and 3000) and
      (numero != 101);
```

$$\sigma_{prixht > 1000 \; and \; prixht < 3000 \; and \; numero_{!} = 101}(Vente)$$

- Quels sont les clients dont le nom est soit Prosper, soit Durand, soit Anthonin ?

```
SELECT *
FROM Client
WHERE nom in ('Prosper', 'Durand', 'Anthonin') ;
```

$$\sigma_{nom = 'Prosper' \; or \; nom = 'Durand' \; or \; nom = 'Anthonin'}(Client)$$

- Quels sont les clients dont le nom commence par 'P' ?

```
SELECT *
FROM Employe
WHERE nom LIKE 'P%' ;
```

- Quels sont les clients dont le nom commence par 'P' et à un 'S' comme 4ème lettre ?

```
SELECT *
FROM Client
WHERE nom LIKE 'P__S%' ;
```

- **Requêtes et valeurs nulles**

  - Quels sont les ventes dont la date de réalisation est inconnue ?

```
SELECT *
FROM Vente
WHERE date is null ;
```

Ou : IS NOT NULL

Attention :

WHERE date = NULL ⇒ *Whatever* comparé avec NULL → Ni vrai ni faux

COUNT, MIN, SUM ⇒ *Ignore les valeurs* NULL → sauf COUNT (*)

Toute opération appliquée à NULL donne pour résultat NULL

- **Les autres clauses (tri)**

```
SELECT attribut1,..
FROM
WHERE expression
ORDER BY attribut1 [ASC] [DESC]...
```

Affichage!

  - Donner le numero, le prix HT et la marque des produits selon l'ordre décroissant des marques et l'ordre croissant des prix HT

```
SELECT marque, prix_ht, numero
FROM  Produit
ORDER BY marque DESC, prix_ht ASC;
```

Mais aussi:

```
ORDER BY 1 DESC, 2 ASC;
```

Ordre d'affichage!

- **Fonctions d'agrégat**

  - Somme : **SUM**(nom d'attribut)
  - Nombre d'enregistrements :
    - **Count**(*)
    - **Count**(**Distinct** nom d'attribut)
  - Valeur maximum : **Max**(nom d'attribut)
  - Valeur minimum :**Min**(nom d'attribut)
  - Moyenne :
    - **AVG**(nom d'attribut)
    - **AVG(**Distinct nom d'attribut)
  - **Dans les clauses :**
    - SELECT
    - HAVING

- **Clause Group By**

```
SELECT attributs recherchés
FROM  liste des relations
WHERE condition
GROUP BY attributs de regroupement
[HAVING   condition sur le groupe ];
```

```
SELECT COUNT (*)
FROM  Produit
GROUP BY marque
```

```
SELECT AVG (prix_ht), nom
FROM  Produit
GROUP BY marque
```

```
SELECT COUNT (*)
FROM  Produit
WHERE marque <> 'BMW'
GROUP BY marque
HAVING AVG (prix_ht) > 10.5
```

- **Utilisation des fonctions statistiques**
  - **Donner la moyenne des prix HT et les prix min et max**

```
SELECT AVG (prix_ht), MIN (prix_ht), MAX (prix_ht)
FROM  Produit
```

  - **Même chose mais par marque**

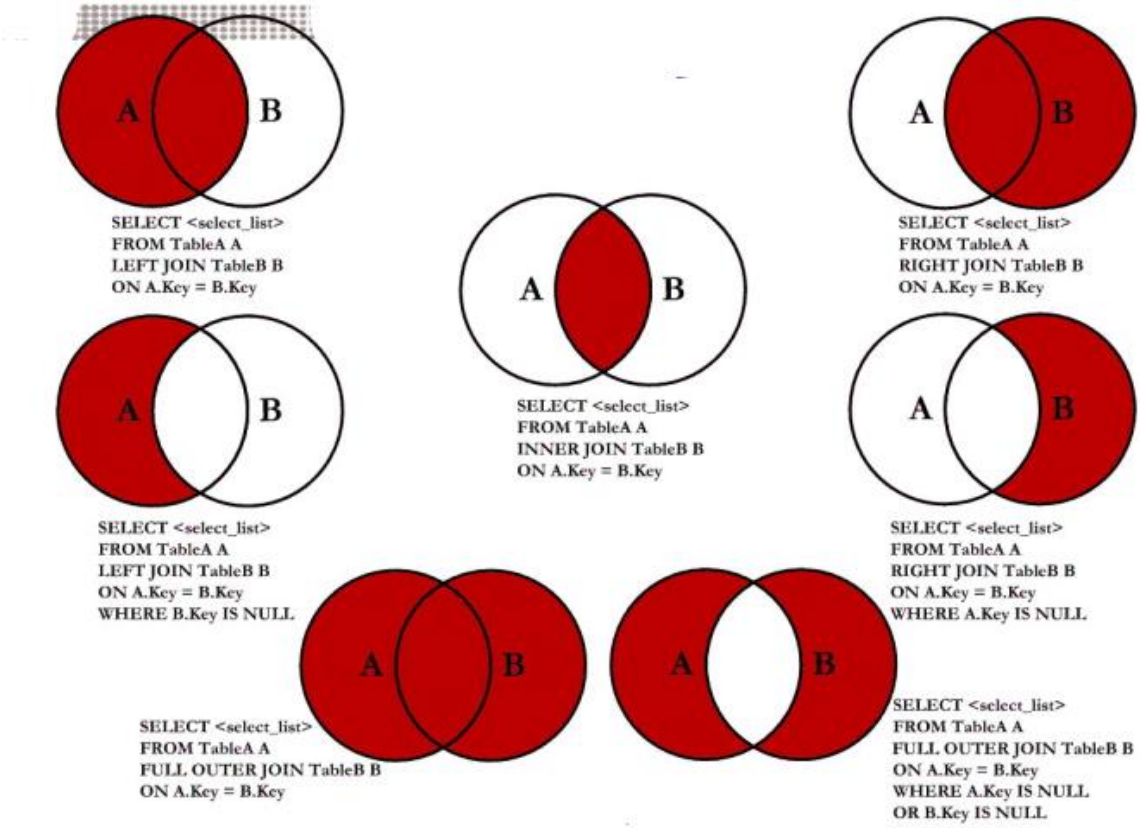
```
SELECT AVG (prix_ht), MIN (prix_ht), MAX (prix_ht)
FROM  Produit
GROUP BY marque
```

  - **Même chose mais par marque si la moyenne > 10,5**

```
SELECT AVG (prix_ht), MIN (prix_ht), MAX (prix_ht)
FROM Produit
GROUP BY marque HAVING AVG (prix_ht) > 10.5
```

- **Requêtes multi-tables (Opérateur Jointure)**

SQL JOINS :



- **Requêtes imbriquées**

  - **Quelles sont les références produit dont le prix HT est supérieur au prix HT moyen des produits ?**

```
SELECT reference

FROM produit

WHERE produit.prix_ht >(SELECT AVG(p.prix_ht) from
produit P)
```

FIN !