Cours 2 : Le langage SQL (partie 1) requêtes et opérateurs

Rabii EL GHORFI

Module : Technique de programmation avancées

Département : Mathématiques, informatique et géomatique (MIG)

EHTP 2017-2018

Principaux axes du cours

- Les types de données dans SQL
- Les fonctions prédéfinies
- Les requêtes simples
- Les clauses Where, Order by, Group by
- Les requêtes multi-tables (jointures)
- Les opérateurs Union, Intersect, Except

SQL (1)

SQL (**S**tructured **Q**uery **L**anguage)

- Langage déclaratif introduit par IBM milieu des années 70
- Implémenté dans plusieurs SGBD tel que MySQL
- Standardisé par l'ANSI (American National Standard Institute)
  - SQL 1 (v1) initial ANSI 1986
  - SQL 1 (v2) avec intégrité référentielle ANSI 1989
  - SQL 2 ANSI 1992
  - SQL 3 incorpore la notion d'objet ANSI 1999

SQL (2)

Fonctionnalités :

- Créer la structure de la base de données et de ses tables
- Exécuter les taches de base de la gestion des données, telle que :
  - l'insertion,
  - la modification,
  - la suppression de données dans les tables
- Effectuer des requêtes simples ou complexes

Langage orienté transformation

SQL (3)

Autres fonctionnalités :

- Contrôle des accès aux données
  - Gestion des utilisateurs
  - Gestion des habilitations
- Gestion des transactions

Principaux types d'utilisateurs :

- DBA = DataBase Administrator
- Développeurs
- Clients (utilisateurs finaux)

SQL (4)

Important : **MySQL**

- Avant d'introduire la syntaxe du langage SQL, il est important de noter que la syntaxe peut varier d'un SGBD à un autre
  - Toutefois, cette variation est minime
  - La migration d'un SGBD à un autre requiert une adaptation de la syntaxe

- Dans la suite de ce cours, on considèrera le SGBD MySQL
  - MySQL dérive directement de SQL
  - L'interface phpMyAdmin développé en PHP offre un outil graphique de manipulation du SGBD MySQL

Types de données

- Les entiers : Nombres entiers signés ou non (int)
- Les flottants : Nombres à virgule (float)
- Les chaines : Chaines de caractères (varchar)
- Les dates : Date et heure (date)
- Enumération
- Ensemble

Les entiers

nom	borne inférieure	borne supérieure
TINYINT	-128	127
TINYINT UNSIGNED	0	255
SMALLINT	-32768	32767
SMALLINT UNSIGNED	0	65535
MEDIUMINT	-8388608	8388607
MEDIUMINT UNSIGNED	0	16777215
INT*	-2147483648	2147483647
INT* UNSIGNED	0	4294967295
BIGINT	-9223372036854775808	9223372036854775807
BIGINT UNSIGNED	0	18446744073709551615

(*) : **INTEGER** est un synonyme de **INT**.

UNSIGNED permet d'avoir un type non signé.

ZEROFILL : remplissage des zéros non significatifs.

Les flottants

nom	domaine négatif : borne inférieure borne supérieure	Domaine positif : borne inférieure borne supérieure
FLOAT	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
DOUBLE*	-1.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

(*) : **REAL** est un synonyme de **DOUBLE**.

Les chaines

nom	longueur
CHAR(M)	Chaîne de taille fixée à M, où 1<M<255, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où 1<M<255, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65535 caractères.
MEDIUMTEXT	Longueur maximale de 16777215 caractères.
LONGTEXT	Longueur maximale de 4294967295 caractères.
DECIMAL(M,D)*	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupe un caractère.

(*) : **NUMERIC** est un synonyme de **DECIMAL**.

Les dates

nom	description
DATE	Date au format anglophone AAAA-MM-JJ.
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(M)	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de **TIMESTAMP**.
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

nom	description
TIMESTAMP(2)	AA
TIMESTAMP(4)	AAMM
TIMESTAMP(6)	AAMMJJ
TIMESTAMP(8)	AAAAMMJJ
...	...

Si attribut de type TIMESTAMP

= *NULL* :

- prend la date et heure de l'insertion.
- suit le format indiqué

Enumérations

- Le type énumération ENUM restreint le choix de la valeur à une liste de chaines de caractères
- Exemple : **ENUM**

```
CREATE TABLE shirts (
    nom VARCHAR(40),
    taille ENUM('x-small', 'small', 'medium', 'large', 'x-large')
);
```

- Dans cette exemple la valeur de taille ne peut contenir qu'une des cinq valeurs prédéfinies

Ensembles

- Le type ensemble SET restreint le choix de la valeur à l'ensemble des combinaisons possibles d'une liste de chaines de caractères
- Exemple : **SET**

```
CREATE TABLE equipe (
   joueur SET('saad', 'jed')
);
```

- Dans cette exemple la valeur de joueur peut être l'une des quatre valeurs suivantes '', 'saad', 'jed', 'saad, jed'

Les fonctions prédéfinies (1)

- **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

Les fonctions prédéfinies (2)

- **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)

Les fonctions prédéfinies (3)

- La moyenne : **AVG**

```
SELECT AVG(prix)
```

- Le minimum : **MIN**

```
SELECT MIN(prix)
```

- Le maximum : **MAX**

```
SELECT MAX(prix)
```

- La somme : **SUM**

```
SELECT SUM(prix)
```

- Le décompte : **COUNT**

```
SELECT COUNT(*)                    -> compte toutes les lignes de la table
SELECT COUNT(DISTINCT prix)        -> compte toutes les lignes avec prix distincts
```

Les requêtes simples (1)

- Syntaxe d'une requête simple :

SELECT … FROM Table WHERE … GROUP BY … HAVING … ORDER BY …

- SELECT spécifie les colonnes qui doivent apparaitre dans les résultats
- FROM spécifie la table ou les tables à utiliser (requête simple = 1 table)
- 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 à une certaine condition
- ORDER BY spécifie l'ordre d'apparition des données dans le résultat

Les requêtes simples (2)

Quelques exemples :

- Affichage de tous les noms et adresses des clients

```
SELECT nom, adresse FROM Client;
```

- Affichage de toutes les informations des clients

```
SELECT * FROM Client;
```

- Affichage de toutes les adresses sans doublons

```
SELECT DISTINCT adresse FROM Client;
```

La clause WHERE

- Objectif : Ajouter des contraintes à la sélection

Quelques exemples :

- Une comparaison — WHERE salaire>10000, ville='Casablanca'
- Un intervalle — WHERE salaire BETWEEN 20000 and 30000
- Un ensemble — WHERE couleur IN ('red', 'vert')
- Une correspondance — WHERE adresse LIKE '%Rabat%'
- Une valeur NULL — WHERE adresse IS NULL

Les requêtes simples (3)

Quelques exemples :

- Affichage des produits dont le nom est XBOX

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

- Affichage des ventes réalisées il y'a plus de 30 jours

```
SELECT * FROM Vente WHERE date<CURRENT_DATE()-30;
```

- Affichage des ventes réalisées après le 01 Janvier 2017

```
SELECT * FROM Vente WHERE date>DATE('2017-01-01');
```

Les requêtes simples (4)

- Affichage des ventes dont le prix est entre 1000 et 3000 et dont le client n'est pas le numéro 23

```
SELECT * FROM Ventre WHERE (prix between 1000 and 3000) and (numero != 23);
```

- Affichage des clients dont le nom est soit saad ou jed ou karam

```
SELECT * FROM Client WHERE nom in ('saad', 'jed', 'karam');
```

Les requêtes simples (5)

- Affichage des clients dont le nom commence par P

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

- Affichage des clients dont le nom commence par P et à la 4eme lettre un S

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

Les requêtes simples (6)

- Affichage des ventes dont la date est inconnue

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

Attention : On n'utilise pas la notation date = NULL

Toute opération comparé à NULL donne comme résultat NULL

Pour une valeur non-nulle, on utilisera date is not NULL

La clause ORDER BY (1)

- Objectif : Trier selon les attributs

ORDER BY Attribut1 DESC, Attribut2 ASC

- Lorsqu'il s'agit d'un entier ORDER BY effectue un tri : ASC du plus petit au plus grand et DESC du plus grand au plus petit
- Lorsqu'il s'agit d'une chaine de caractère ORDER BY effectue un tri alphabétique : ASC de A à Z et DESC de Z à A

Attention : Dans cette exemple, le tri s'effectue d'abord pour Attribut1, puis une fois terminé, il s'effectue pour Attribut2

La clause ORDER BY (2)

SELECT Marque, Prix, CodeP FROM Produit ORDER BY Marque DESC, Prix ASC;

Table : Produit

Marque	Prix	CodeP
BMW	400000	BM8908
Renault	250000	RE3672
BMW	800000	BM1208
Volvo	600000	VO3412
Volvo	450000	VO3779
BMW	300000	BM3459
Renault	180000	RE1196

Table : Produit

Marque ↘	Prix ↗	CodeP
Volvo	450000	VO3779
Volvo	600000	VO3412
Renault	180000	RE1196
Renault	250000	RE3672
BMW	300000	BM3459
BMW	400000	BM8908
BMW	800000	BM1208

La clause GROUP BY (1)

- Objectif : Grouper selon les attributs

GROUP BY Attribut HAVING (Condition)

- La condition qui suit HAVING agit comme la clause WHERE, elle peut donc éliminer des tuples du résultat

Attention : Dans cette exemple, si l'on utilise une fonction statistique sur Attribut, cette fonction sera appliquée uniquement par groupe

La clause GROUP BY (2)

SELECT Marque, AVG(Prix) FROM Produit GROUP BY Marque HAVING AVG(prix)>300000;

Table : Produit

Marque	Prix	CodeP
BMW	400000	BM8908
Renault	250000	RE3672
BMW	800000	BM1208
Volvo	600000	VO3412
Volvo	450000	VO3779
BMW	300000	BM3459
Renault	180000	RE1196

Table : Produit

Marque	AVG(Prix)
BMW	500000
Volvo	525000

La clause GROUP BY calcule AVG(prix) par marque

Pour BMW : AVG(Prix) = 500000 ; Renault : AVG(Prix) = 215000 ; VOLVO : AVG(Prix) = 525000

Les requêtes multi-tables (1)

- Syntaxe d'une requête multi-tables :

SELECT … FROM Table1, Table2 … WHERE … GROUP BY … HAVING … ORDER BY …

- Le résultat regroupe toutes les occurrences qui satisfont la clause WHERE

Exemple :

```
SELECT Produit.nom, Vente.prix FROM Produit, Vente WHERE Vente.numProduit =
Produit.numero ;
```

- La clause WHERE désigne le critère de jointure
- Le résultat implique plusieurs tables, il faut alors joindre les tables (Jointure)

Les requêtes multi-tables (2)

- Jointure sans critère de jointure

```
SELECT Table1.attribut5, Table2.attribut4 FROM Table1, Table2;
```

- Dans cette exemple, il n'y pas de clause WHERE donc pas de critère de jointure
- Le résultat est alors le produit cartésien des deux tables = l'ensemble des tuples composés de (Table1.attribut5, Table2.attribut4)
- Il n'y a aucun intérêt à effectuer ce genre de requêtes multi-tables

Les requêtes multi-tables (3)

- Les Jointures par défaut sont du type **INNER JOIN**

SELECT … FROM Table1, Table2 WHERE …

SELECT … FROM Table1 INNER JOIN Table2 WHERE …

- Ces deux requêtes sont équivalentes, c'est pour cette raison qu'en pratique, on ne spécifie pas INNER JOIN
- INNER JOIN = Les tuples qui ne satisfont pas la clause WHERE sont éliminés

Les requêtes multi-tables (4)

- Les Jointures du type **OUTER JOIN**

SELECT … FROM Table1 FULL OUTER JOIN Table2 WHERE …

SELECT … FROM Table1 LEFT OUTER JOIN Table2 WHERE …

SELECT … FROM Table1 RIGHT OUTER JOIN Table2 WHERE …

- LEFT OUTER JOIN : Les tuples de la table de gauche (LEFT) sont conservés même si la clause WHERE n'est pas vérifiée
- RIGHT OUTER JOIN : Les tuples de la table de droite (RIGHT) sont conservés même si la clause WHERE n'est pas vérifiée
- FULL OUTER JOIN : Les tuples de la table de droite et gauche sont conservés même si la clause WHERE n'est pas vérifiée

INNER JOIN

```
SELECT Personne.Nom, Adresse.Ville FROM Personne, Adresse
WHERE Personne.NumA = Adresse.NumA;
```

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat

LEFT OUTER JOIN

```
SELECT Personne.Nom, Adresse.Ville FROM Personne LEFT OUTER JOIN Adresse
WHERE Personne.NumA = Adresse.NumA;
```

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat
Elyoussefi	NULL
Dahbi	NULL

RIGHT OUTER JOIN

SELECT Personne.Nom, Adresse.Ville FROM Personne RIGHT OUTER JOIN Adresse
WHERE Personne.NumA = Adresse.NumA;

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat
NULL	Casablanca
NULL	Marrakech

FULL OUTER JOIN

SELECT Personne.Nom, Adresse.Ville FROM Personne FULL OUTER JOIN Adresse
WHERE Personne.NumA = Adresse.NumA;

Table : Personne

NumP	Nom	Age	Sexe	NumA
150	Elyoussefi	40	M	508
19	Dahbi	25	F	403
11	Marzouk	20	M	**301**

Table : Adresse

NumA	Ville	Quartier	Avenue	Numéro
301	Rabat	HayRiad	Nakhil	120
302	Casablanca	Maarif	Atlas	7
304	Marrakech	Gueliz	Mlyrachid	5

Table : Résultat

Nom	Ville
Marzouk	Rabat
NULL	Casablanca
NULL	Marrakech
Elyoussefi	NULL
Dahbi	NULL

L'opérateur UNION (1)

- L'union de deux tables rassemble les tuples de la première table et les tuples de la seconde table
- Les tuples en communs ne sont conservées qu'en un seul exemplaire, c'est à dire que l'opération d'union élimine les doublons

Exemple :

```
SELECT Nom, Prenom FROM Professeur UNION SELECT Nom, Prenom FROM Etudiant;
```

- L'opération retourne une table avec nom et prénom de tous les professeurs et étudiants

Remarque : L'opérateur UNION ne peut pas être reproduit avec des jointures

L'opérateur UNION (2)

SELECT Nom, Prenom FROM Professeur UNION SELECT Nom, Prenom FROM Etudiant;

Table : Professeur

Nom	Prenom	Ville
Boukili	Driss	Casablanca
Rziza	Sara	Marrakech
El ghorfi	Rabii	Rabat

Table : Etudiant

Nom	Prenom	Age
Elyoussefi	Abdelali	40
Rziza	Sara	22
Mohtaram	Saad	21

Table : Résultat

Nom	Prenom
Boukili	Driss
Rziza	Sara
El ghorfi	Rabii
Elyoussefi	Abdelali
Mohtaram	Saad

L'opérateur INTERSECT (1)

- L'intersection de deux tables revoie les tuples communs aux deux tables

Exemple :

```
SELECT Nom, Prenom FROM Etudiant INTERSECT SELECT Nom, Prenom FROM Entrepreneur;
```

- L'opération retourne les noms et prénoms qui se trouvent à la fois dans la table Etudiant et la table Entrepreneur

Remarque : L'opérateur INTERSECT peut être reproduit avec des jointures

L'opérateur INTERSECT (2)

SELECT Nom, Prenom FROM Etudiant INTERSECT SELECT Nom, Prenom FROM Entrepreneur;

Table : Etudiant

Nom	Prenom	Age
Elyoussefi	Abdelali	40
Rziza	Sara	22
Mohtaram	Saad	21

Table : Entrepreneur

Nom	Prenom	Age
Marzouk	Reda	35
Dhiman	Alae	30
Mohtaram	Saad	21

Table : Résultat

Nom	Prenom
Mohtaram	Saad

L'opérateur EXCEPT (1)

- La différence de deux tables renvoie les tuples de la première table qu'on ne retrouve pas dans la seconde

Exemple :

```
SELECT Nom, Prenom FROM Etudiant EXCEPT SELECT Nom, Prenom FROM Entrepreneur;
```

- L'opération retourne les noms et prénoms qui se trouvent dans la table Etudiant et qui ne se trouvent pas dans la table Entrepreneur

Remarque : L'opérateur EXCEPT peut être reproduit avec des jointures

L'opérateur EXCEPT (2)

SELECT Nom, Prenom FROM Etudiant EXCEPT SELECT Nom, Prenom FROM Entrepreneur;

Table : Etudiant

Nom	Prenom	Age
Elyoussefi	Abdelali	40
Rziza	Sara	22
Mohtaram	Saad	21

Table : Entrepreneur

Nom	Prenom	Age
Marzouk	Reda	35
Dhiman	Alae	30
Mohtaram	Saad	21

Table : Résultat

Nom	Prenom
Elyoussefi	Abdelali
Rziza	Sara

L'opérateur EXCEPT (3)

SELECT Nom, Prenom FROM Etudiant EXCEPT SELECT Nom, Prenom FROM Entrepreneur;

- LE SGBD mySQL ne supporte pas l'utilisation de l'opérateur EXCEPT
- L'opérateur EXCEPT est très couteux en mémoire

Remarque : Toutefois, on peut trouver un équivalent de l'opérateur EXCEPT avec les requêtes imbriquées (Voir Cour 3)

SELECT Nom, Prenom FROM Etudiant WHERE NOT EXISTS (SELECT * FROM Entrepreneur WHERE Etudiant.nom = Entrepreneur.nom AND Etudiant.nom = Entrepreneur.nom)

L'opérateur EXCEPT (4)

De manière générale, on a les équivalences suivantes :

- Pour un seul attribut

SELECT A FROM T1 EXCEPT SELECT A FROM T2

⇔

SELECT A FROM T1 WHERE A NOT IN (SELECT * FROM T2 WHERE T1.A = T2.A)

- Pour plusieurs attribut

SELECT A, B FROM T1 EXCEPT SELECT A, B FROM T2

⇔

SELECT A, B FROM T1 WHERE NOT EXISTS (SELECT * FROM T2 WHERE T1.A = T2.A AND T1.B = T2.B)