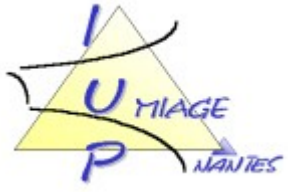
Module M4
Base de données

Chapitre 3
SQL
Requêtes simples

Auteurs : Laura Monceaux / Véronique Laime



Le LMD ou DML (Data Manipulation Language) est la partie du SQL qui traite les données. Il permet de sélectionner des données des tables de la base de données. Ce langage est utilisé par les utilisateurs de la base de données.

3.1 Sélectionner les données : commande SELECT

La commande **SELECT** permet de sélectionner par projection un ou des champs issus d'une ou de plusieurs tables de la base de données relationnelle.

La syntaxe de la commande SELECT est la suivante:

```
SELECT [ALL|DISTINCT|Ɛ] attributs_projection
FROM tables_necessaires
[WHERE conditions_sélection]
[ORDER BY attributs_tri];
```

Remarques:

- Entre crochets sont indiquées les parties optionnelles.
- Une requête SQL se termine par un point virgule.
- * Indique la totalité des champs de la table

Il doit obligatoirement y avoir au moins les mots clés SELECT et FROM.

- La clause SELECT permet de préciser les attributs sur lequel on va projeter le résultat de notre requête
  - attributs_projection est
    - soit un attribut (att),
    - soit une liste d'attributs (att [,att]*)
    - soit une étoile * (retourne tous les attributs)
  - L'option ALL est une option par défaut de la commande SELECT et permet de sélectionner tous les enregistrements de la ou des tables remplissant la condition éventuellement précisée.
  - L'option DISTINCT, par opposition à l'option ALL, permet de ne conserver que les enregistrements distincts en éliminant les doublons. Un enregistrement n'apparaitra qu'une seule fois dans la table résultant de la commande SELECT.

- La clause FROM permet de préciser les tables sur lesquelles la requête va s'appuyer. Attention, si plusieurs tables sont nécessaires il est indispensable que les tables soient reliées. Peu importe l'ordre des tables dans la liste.
  - tables_nécessaires est
    - soit une seule table : Table
    - soit plusieurs tables liées entre elle :

      Table T1 [opJoin Table T2 [on cond]]*
  - la **condition associée à une opération de jointure** est de la forme : **T1.att op T2.att**

- La clause WHERE permet d'exprimer des conditions de sélection sur les valeurs d'un ou de

plusieurs champs applicable à **UN tuple**. On utilise les opérateurs logiques et les comparateurs arithmétiques classiques.
  - conditions_sélection est
    - soit une seule condition : att op exp
    - soit plusieurs conditions liées par AND ou OR cond [(AND | OR) cond]*
- La clause ORDER BY permet de trier les données du résultat sur les attributs indiqués.

Voici quelques requêtes simples :

Soit la table Etudiant suivante:

NumEtu	NomEtu	PrenEtu	DateNaissEtu	EmailEtu
123456	DUPOND	Jean	20/10/1988	jean.dupond-1@etu.univ.fr
234567	DURAND	Jeanne	05/02/1989	jeanne.durand@etu.univ.fr
345667	MARTIN	Marie	21/03/1988	marie.martin@etu.univ.fr
987655	DUPOND	Jean	08/10/1990	jean.dupond-2@etu.univ.fr
457832	DUPOND	Pierre	14/12/1989	pierre.dupond@etu.univ.fr
135790	ROUX	Alain	08/05/1989	alain.roux@etu.univ.fr

Voici ce que pourrait être une sélection permettant d'obtenir toutes les informations concernant tous les étudiants de la base de données donc de retourner la table entière à l'utilisateur :

```
SELECT *
FROM Etudiant;
```

Exemple : Si on désire obtenir les noms et prénoms de tous les étudiants de la base de données:

```
SELECT NomEtu, PrenEtu
FROM Etudiant;
```

La table résultante est la suivante:

NomEtu	PrenEtu
DUPOND	Jean
DURAND	Jeanne
MARTIN	Marie
DUPOND	Jean
DUPOND	Pierre
ROUX	Alain

Exemple : Si on désire obtenir les noms et prénoms de tous les étudiants de la base de données sans doublons (les mêmes noms et prénoms n'apparaissent plus qu'une seule fois):

```
SELECT DISNTINCT NomEtu, PrenEtu
FROM Etudiants;
```

La table résultante est la suivante:

NomEtu	PrenEtu
DUPOND	Jean
DURAND	Jeanne
MARTIN	Marie
DUPOND	Pierre
ROUX	Alain

Exemple : Si on désire obtenir les noms et prénoms des étudiants dont le nom est égal à DUPOND:

```
SELECT NomEtu, PrenEtu
FROM Etudiants
WHERE NomEtu = "DUPOND";
```

La table résultante est la suivante:

NomEtu	PrenEtu
DUPOND	Jean
DURAND	Jeanne
DUPOND	Pierre

3.2 Conditions de sélection (clause WHERE)

Dans la clause WHERE, on peut exprimer toutes les opérations de sélection que l'on souhaite réaliser sur les tuples. La condition de sélection peut être de plusieurs formes :

- ***att op expression***
  - op est un opérateur mathématique (=, <>, <=, >=, <, >)

Exemple : si l'on veut les numéros des étudiants dont le nom est DUPOND

```
SELECT NumEtu
FROM Etudiants
WHERE NomEtu = "DUPOND";
```

- ***att like chaine***
  - chaine = chaine de caractères dont on ne connait qu'une partie
  - % dans cette chaîne représentera 0 ou n caractères
  - + 1 ou n caractères

Exemple : si l'on veut les numéros des étudiants dont le nom commence par un D

```
SELECT NumEtu
FROM Etudiants
WHERE NomEtu like  "D%";
```

- ***att in (val1, val2 ...)*** permettant de vérifier qu'un attribut fait partie d'un ensemble de valeurs

Exemple : si l'on veut les numéros des étudiants dont le prénom fait partie d'une liste

```
SELECT NumEtu
FROM Etudiants
WHERE PrenEtu in ('Jean','Marie','Alain');
```

- ***att between (val1,val2)*** permettant de gérer les intervalles de valeurs
- ***att is null*** permettant de sélectionner l'attribut n'ayant pas de valeur

On peut également avoir des conditions sur des dates, il est alors utile d'utiliser la fonction ***extract DAY | MONTH | YEAR from att_date*** pour récupérer des parties de la la date (resp. le jour, le mois

et la date) sous forme d'entier.

Exemple : si l'on veut les numéros des étudiants nés en 1988 en mars.

```
SELECT NumEtu
FROM Etudiants
WHERE extract MONTH from dateNaissEtu = 3
        and extract YEAR from dateNaissEtu = 1988;
```

on pourrait aussi utiliser l'opérateur like

```
SELECT NumEtu
FROM Etudiants
WHERE dateNaissEtu like '%/03/1988';
```

3.3 Jointures

Les opérations de jointures en SQL, sont celles définies dans l'algèbre relationnelle. Elles utilisent les opérateurs suivants, exprimées dans la clause FROM :

- jointure interne : T1 JOIN T2 ON cond_jointure
- jointure naturelle : T1 NATURAL JOIN T2
- jointure externe : T1 (LEFT | RIGHT | FULL) OUTER JOIN T2 ON cond_jointure
  - attention toute jointure externe est accompagnée d'une condition de jointure
  - la relation directrice est indiquée par le choix de la clause
    - LEFT OUTER JOIN : T1 est la relation directrice
    - RIGHT OUTER JOIN : T2 est la relation directrice
    - FULL OUTER JOIN : T1 et T2 sont les relations directrices

Attention, nous rappelons que toute condition de jointure est de la forme T1.att op T2.att.

On rajoute dans notre base de données EXEMPLE, une nouvelle table Notes qui stocke les notes des étudiants dans certains modules.

Soit la table NOTES suivante:

NumEtu	codemat	noteCC	noteEX
123456	M1	12	14
234567	M2	10	8
234567	M1	17	14
345667	M3	5	7
457832	M1	14	7
123456	M4	4	10
345667	M4	12	12
987655	M1	13	8
457832	M2	11	12
135790	M3	17	14

Quelques exemples de requête avec une jointure :

Exemple : si l'on désire les noms et prénoms des étudiants inscrits au module M1 et nés en 1988, triés sur le nom puis le prénom

```
SELECT nomEtu,prenEtu
FROM Etudiant NATURAL JOIN Notes
WHERE codemat = 'M1' AND dateNaissEtu LIKE '%1988'
ORDER BY nom, prenom ASC;
```

Exemple : si l'on désire les noms et prénoms des étudiants qui ne sont pas inscrits dans le module M1

Attention l'erreur ici serait de faire une condition de sélection : codemat <> 'M1' car un étudiant peut être inscrit en M1 et dans une autre matière.

```
SELECT nomEtu,prenEtu
FROM Etudiant E
        LEFT OUTER JOIN
        (select * from Notes where codemat='M1') N
        ON E.numEtu=N.numEtu
WHERE codemat is null;
```

3.4 Opérations ensemblistes

Les opérations ensemblistes en SQL, sont celles définies dans l'algèbre relationnelle. Elles utilisent les opérateurs suivants, placés entre deux clauses SELECT :

- UNION
- INTERSECT (ne fait pas partie de la norme SQL et n'est donc pas implémenté dans tous les SGBDR notamment pour MySQL)
- EXCEPT (ne fait pas partie de la norme SQL et n'est donc pas implémenté dans tous les SGBDR notamment pour MySQL)

2.2.1. L'opérateur UNION

Cet opérateur permet d'effectuer une union des enregistrements sélectionnés par deux clauses SELECT. Par défaut les doublons sont automatiquement éliminés.

```
(SELECT ..... FROM ..... WHERE.....)
UNION
(SELECT ..... FROM ..... WHERE .....)
```

ATTENTION, comme en algèbre relationnel, l'union ne peut se faire que sur des tables ayant même attribut ainsi les attributs retournés dans les SELECT doivent être les mêmes.

2.2.2. L'opérateur INTERSECT

Cet opérateur permet d'effectuer une intersection des enregistrements sélectionnés par deux clauses SELECT.

```
(SELECT liste_att FROM  TablesRequête1   WHERE conditionsRequête1)
    INTERSECT
(SELECT liste_att FROM  TablesRequête2   WHERE conditionsRequête2) ;
```

ATTENTION, comme en algèbre relationnel, l'intersection ne peut se faire que sur des tables ayant même attribut ainsi les attributs retournés dans les SELECT doivent être les mêmes.

L'opérateur INTERSECT n'étant pas implémenté dans tous les SGBDR, il est possible de le remplacer par une autojointure :

```
SELECT liste_att
FROM (TablesRequête1)  JOIN (TablesRequête2) ON cond_jointure_liste_att
WHERE  conditionsRequête1 and conditionsRequête2 ;
```

Exemple : si l'on veut les noms et prénoms des étudiants inscrits dans les deux modules M1 et M2. On pourra écrire les requêtes suivantes :

```
(SELECT nomEtu,prenEtu FROM Etudiant Natural Join Notes WHERE codemat='M1')
INTERSECT
(SELECT nomEtu,prenEtu FROM Etudiant Natural Join Notes WHERE codemat='M2');

ou

SELECT nom Etu,prenEtu
FROM (Etudiant Natural Join Notes) T1 join (Etudiant Natural Join Notes) T2
                    on T1.nomEtu=T2.nomEtu and T1.prenomEtu=T2.prenomEtu
WHERE T1.codemat='M1' and T2.codemat='M2';
```

2.2.2. L'opérateur EXCEPT

Cet opérateur permet d'effectuer une différence entre les enregistrements sélectionnés par deux clauses SELECT, c'est-à-dire sélectionner les enregistrements de la première table n'appartenant pas à la seconde.

```
(SELECT liste_att FROM TablesRequête1 [WHERE  conditionsRequête1] )
EXCEPT
(SELECT liste_att FROM TablesRequête2 [WHERE  conditionsRequête2] );
```

L'opérateur EXCEPT n'étant pas implémenté dans tous les SGBDR, il est possible de le remplacer par une jointure externe :

```
SELECT  liste_att
FROM TablesRequête1  LEFT OUTER JOIN
        (select liste_att from TablesRequête2 [WHERE  conditionsRequête2])
        ON cond_join_liste_att
WHERE  attTablesRequête2 is null  [AND conditionsRequête1];
```

Exemple : si l'on veut les noms et prénoms des étudiants inscrits dans le module M1 mais pas dans le module M2. On pourra écrire les requêtes suivantes :

```
(SELECT nomEtu,prenEtu FROM Etudiant Natural Join Notes WHERE codemat='M1')
EXCEPT
(SELECT nomEtu,prenEtu FROM Etudiant Natural Join Notes WHERE codemat='M2');

ou
```

```
SELECT nom Etu,prenEtu
FROM (Etudiant Natural Join Notes) T1
        LEFT OUTER JOIN
       (SELECT nomEtu,prenEtu
        FROM Etudiant Natural Join Notes
        WHERE codemat='M2) T2
       ON T1.nomEtu=T2.nomEtu andT1.prenomEtu=T2.prenomEtu
WHERE T1.codemat='M1' and T2.nomEtu is null;
```