Chapitre 1 : Langage d'Interrogation des Données (LID) dans SQL

Sous-requêtes

Tarek HAMROUNI
Maître de Conférences en Informatique
hamrouni.tarek@gmail.com

Sous-requêtes

- Une sous-requête n'est autre qu'**une requête SELECT qui est mise à l'intérieur d'une autre requête** : SELECT, INSERT, UPDATE, DELETE, etc.

- Dans la suite, nous nous intéressons aux sous-requêtes utilisées dans une requête SELECT, par exemple :

```
SELECT *
FROM Emp
WHERE ageE = (SELECT ageE FROM Emp WHERE numE = 100);
```

Sous-requêtes

- Nous allons considérer dans la suite le cas des sous-requêtes SELECT incluses à l'intérieur d'une clause WHERE ou HAVING d'une autre requête dite appelante ou principale.

- Dans la syntaxe et les exemples de cours qui vont suivre, les sous-requêtes seront réalisées par rapport à la clause WHERE. Le TP comporte des sous-requêtes à mettre dans le HAVING.

Types de sous-requêtes

- Une sous-requête renvoyant une seule ligne et une seule colonne
- Une sous-requête renvoyant plusieurs lignes et une seule colonne
- Une sous-requête renvoyant plusieurs colonnes
- Une sous-requête corrélée ou synchronisée

Sous-requête renvoyant une seule ligne et une seule colonne

WHERE exp1 op_comparaison (SELECT exp2 FROM…)

- **op_comparaison est un opérateur de comparaison (=, >,>=,<,<=,!=)**
- **exp1 et exp2 doivent être cohérents (de type compatible).**

Application : Emp(numEmp, prenom, nom, age, ville, salaire, numDept)

Requête 1 : Afficher la liste des employés ayant un salaire qui dépasse le salaire moyen de tous les employés.

Requête 2 : Afficher la liste des employés qui travaillent dans le même département que l'employé ayant 1 comme numEmp.

Sous-requête renvoyant une seule ligne et une seule colonne

Application : Emp(numEmp, prenom, nom, age, ville, salaire, numDept)

Requête 1 : Afficher la liste des employés ayant un salaire qui dépasse le salaire moyen de tous les employés.

```
SELECT *
FROM Emp WHERE salaire > (SELECT AVG(salaire) FROM Emp) ;
```

Requête 2 : Afficher la liste des employés qui travaillent dans le même département que l'employé ayant 1 comme numEmp.

```
SELECT *
FROM Emp
WHERE numDept = (SELECT numDept FROM Emp WHERE numEmp=1) AND numEmp != 1;
```

Sous-requête renvoyant plusieurs lignes et une seule colonne

- Les opérateurs d'inclusions IN et NOT IN

  WHERE exp1 IN (SELECT exp2 FROM …….)

  WHERE exp1 NOT IN (SELECT exp2 FROM …….)

- Les opérateurs de comparaison combinés avec ANY ou ALL

  WHERE exp1 op_comparaison ANY (SELECT exp2 FROM……..)
  WHERE exp1 op_comparaison ALL (SELECT exp2 FROM……..)

➔ op_comparaison est un opérateur de comparaison (=, >, …..)
➔ **ANY** : la condition est vraie si la comparaison est vraie pour **au moins une** des valeurs retournées.
➔ **ALL** : la condition est vraie si la comparaison est vraie **pour toutes les** valeurs retournées

Sous-requête renvoyant plusieurs lignes et une seule colonne

- Les opérateurs de comparaison combinés avec ANY ou ALL

WHERE exp1 op_comparaison ANY (SELECT exp2 FROM……..)
WHERE exp1 op_comparaison ALL (SELECT exp2 FROM……..)

➔ op_comparaison est un opérateur de comparaison (=, >, …..)
➔ **ANY** : la condition est vraie si la comparaison est vraie pour **au moins une** des valeurs retournées.
➔ **ALL** : la condition est vraie si la comparaison est vraie **pour toutes les** valeurs retournées.

Remarque :

= ANY ⇔ IN
!= ALL ⇔ NOT IN

Sous-requête renvoyant plusieurs lignes et une seule colonne

Application : Emp(num, prenom, nom, age, ville, salaire, #numDept)

Dept(numDept, nomDept)

Requête 1 : Afficher les employés gagnant plus que tous les employés du département 30.

Requête 2 : Afficher les employés ayant un salaire >1000 et qui habitent dans une ville parmi les villes des employés du département 'Ressources Humaines'.

Sous-requête renvoyant plusieurs lignes et une seule colonne

Application : Emp(num, prenom, nom, age, ville, salaire, #numDept)

Dept(numDept, nomDept)

Requête 1 : Afficher les employés gagnant plus que tous les employés du département 30.

```
SELECT *
FROM Emp
WHERE salaire > ALL (SELECT salaire FROM Emp WHERE numDept=30);
```

Requête 2 : Afficher les employés ayant un salaire >1000 et qui habitent dans une ville parmi les villes des employés du département 'Ressources Humaines'.

```
SELECT *
FROM Emp
WHERE salaire > 1000 AND ville IN (SELECT DISTINCT ville FROM Emp E, Dept D WHERE E.numDept = D.numDept AND nomDept = 'Ressources Humaines');
```

Sous-requête renvoyant plusieurs colonnes

- Avec une seule ligne sélectionnée :

  WHERE (exp1, exp2 …) op_comparaison_classique (SELECT exp1', exp2', ……)

  ➔ op_comparaison_classique ne peut être que = ou !=

- Avec plusieurs lignes sélectionnées :

  WHERE (exp1, exp2 …) IN (SELECT exp1', exp2', ……)

  WHERE (exp1, exp2 …) NOT IN (SELECT exp1', exp2', ……)

  WHERE (exp1, exp2 …) op_comparaison_classique ANY (SELECT exp1', exp2', ……)

  WHERE (exp1, exp2 …) op_comparaison_classique ALL (SELECT exp1', exp2', ……)

Sous-requête renvoyant plusieurs colonnes

Application : Emp(num, prenom, nom, age, ville, salaire, #numDept)

Dept(numDept, nomDept)

Requête 1 : Afficher le prénom et le nom des employés qui travaillent dans le même département et ayant le même salaire que l'employé ayant 3 comme numéro.

```
SELECT prenom, nom
FROM Emp
WHERE (numDept, salaire) = (SELECT numDept, salaire FROM Emp WHERE num=3) and num != 3;
```

Sous-requête renvoyant plusieurs colonnes

Application : Emp(num, prenom, nom, age, ville, salaire, #numDept)

Dept(numDept, nomDept)

Requête 2 : Afficher le prénom et le nom des employés ne travaillant pas dans le département "Informatique" et ayant le même âge et habitant la même ville qu'un des employés du département "Informatique".

```
SELECT prenom, nom
FROM Emp E, Dept D
WHERE E.numDept = D.numDept AND nomDept != "Informatique" AND
AND (age, ville) = ANY ou bien IN (SELECT DISTINCT age, ville
FROM Emp E1, Dept D1 WHERE E1.numDept = D1.numDept
                          AND nomDept = "Informatique");
```

Sous-requête corrélée ou synchronisée

- Une sous-requête corrélée est une sous-requête dont l'évaluation se fait pour chaque ligne de la requête principale.

➔ Le résultat de la sous-requête dépend des données contenues dans la ligne de la requête principale

Application : Emp(num, prenom, nom, age, ville, salaire, #numDept)

Requête : Afficher toutes les informations sur les employés dont le salaire est supérieur au salaire moyen de son département.

Sous-requête corrélée ou synchronisée

SELECT *
FROM Emp E
WHERE salaire > (La moyenne des salaires des employés du même département que l'employé) ;

SELECT *
FROM Emp E
WHERE salaire > (SELECT AVG (salaire) FROM Emp E1 WHERE E1.numDept = E.numDept) ;

L'exécution de cette requête se produit ainsi :

- *Etape 1 : La requête principale fixe une ligne de la table Emp ayant pour alias E.*
- *Etape 2 : Ayant E.numDept la sous-requête est évaluée.*
- *Etape 3 : Ensuite, la condition de la clause WHERE de la requête principale est évaluée. La ligne est alors retournée ou non suivant la valeur de la condition.*
- *Etape 4 : Itération des étapes 1, 2 et 3 pour les lignes restantes de la table Emp dont l'alias est E*

Opérateurs EXISTS et NOT EXISTS

Si la sous-requête renvoie au moins une valeur :

- la clause **EXISTS (SELECT …)** vaut **True**
- la clause **NOT EXISTS (SELECT …)** vaut **False**

et inversement dans le cas contraire (renvoi d'aucune valeur).

La clause EXISTS est suivie d'une sous-requête et prend la valeur vraie s'il existe au moins une ligne satisfaisant les conditions de la sous-requête. Inversement pour NOT EXISTS. Toutefois, il est important de noter que le résultat en tant que tel de la sous-requête n'a aucune importance.

Opérateurs EXISTS et NOT EXISTS

Application : Soient les relations :

Client(numC, NomC, VilleC)

Produit(numP, libP, prixP, categP)

Commande(numCo, qteCo, dateCo, #numC, #numP)

Requête 1 : Afficher toutes les informations sur les clients qui ont passé au moins une commande.

```
SELECT *
FROM Client C
WHERE EXISTS(SELECT 1 FROM Commande Co WHERE Co.numC = C.numC);
```

Opérateurs EXISTS et NOT EXISTS

Application : Soient les relations :

Client(numC, NomC, VilleC)

Produit(numP, libP, prixP, categP)

Commande(numCo, qteCo, dateCo, #numC, #numP)

Requête 2 : Afficher le nom et ville des clients qui n'ont passé aucune commande.

```
SELECT nomC, villeC
FROM Client C
WHERE NOT EXISTS(SELECT 1 FROM Commande Co WHERE Co.numC = C.numC);
```

Langage d'Interrogation des Données (LID) dans SQL

Opérateurs ensemblistes

Tarek HAMROUNI
Maître de Conférences en Informatique
hamrouni.tarek@gmail.com

Opérateurs ensemblistes

Une requête composée est l'association de plusieurs requêtes par le biais d'opérateurs de combinaison : **les opérateurs ensemblistes**.

SELECT …

UNION / UNION ALL / INTERSECT / MINUS

SELECT …;

→ **Les deux requêtes doivent retourner exactement le même nombre de colonnes, avec les mêmes types de données (ou des types compatibles) et dans le même ordre.**

Opérateurs UNION et UNION ALL

- L'opérateur **UNION** permet de fusionner deux sélections de tables pour obtenir un ensemble de lignes égal à la réunion des lignes des deux sélections.
- Les lignes communes n'apparaîtront qu'une fois. Si on veut conserver les doublons, on peut utiliser la variante **UNION ALL**.

Opérateurs UNION et UNION ALL

Application : Les deux tables EMP1 et EMP2 contiennent les informations sur deux filiales de l'entreprise.

EMP1 (numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

EMP2(numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

Requête : Afficher sans redondance la liste des ingénieurs des deux filiales.

```
SELECT * FROM EMP1 WHERE POSTE='INGENIEUR'
UNION
SELECT * FROM EMP2 WHERE POSTE='INGENIEUR' ;
```

Opérateurs UNION et UNION ALL

Application : Les deux tables EMP1 et EMP2 contiennent les informations sur deux filiales de l'entreprise.

EMP1 (numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

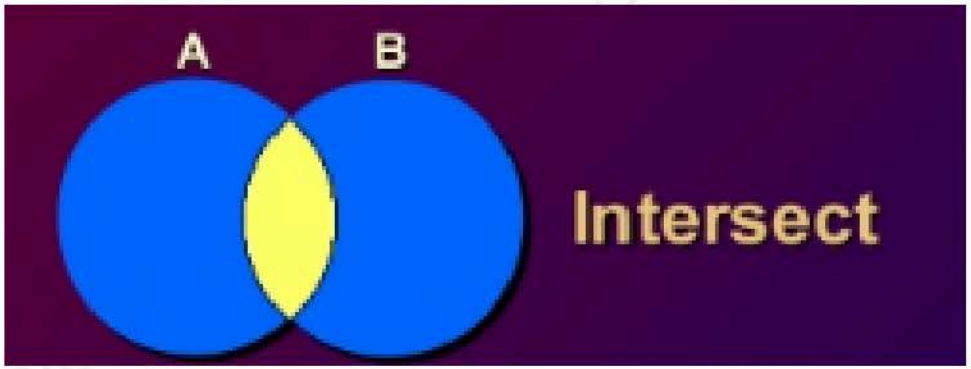
EMP2(numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

Requête : Afficher **avec** redondance la liste des ingénieurs des deux filiales.

```
SELECT * FROM EMP1 WHERE POSTE='INGENIEUR'
UNION ALL
SELECT * FROM EMP2 WHERE POSTE='INGENIEUR' ;
```

Opérateur INTERSECT

- L'opérateur INTERSECT permet d'obtenir l'ensemble des lignes communes à deux interrogations.



Opérateur INTERSECT

EMP1(numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

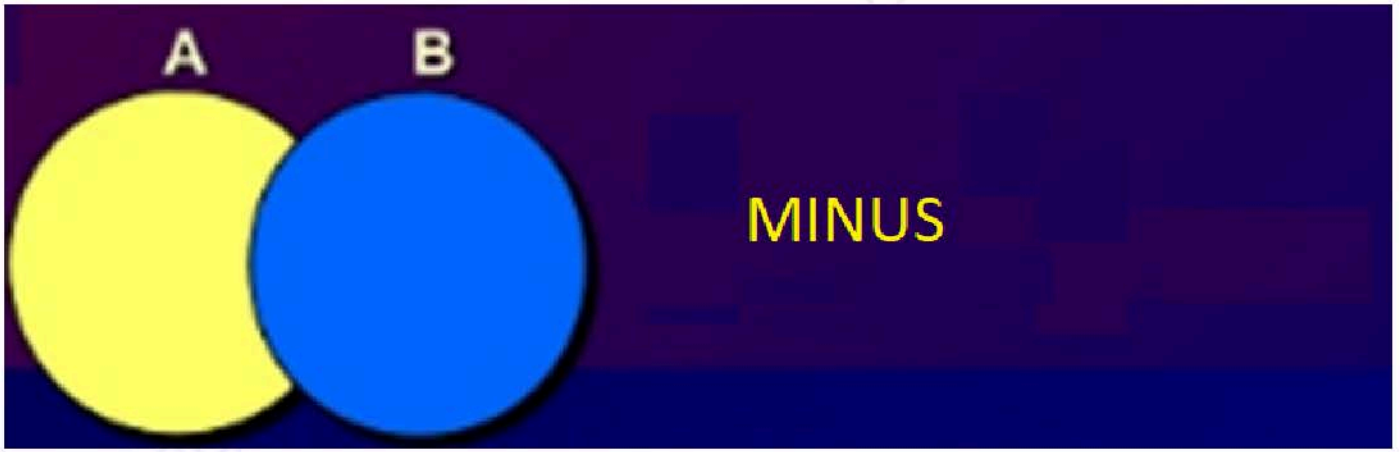
EMP2(numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

Requête : Afficher la liste des départements qui ont des employés dans les deux filiales.

```
SELECT DISTINCT numDept FROM EMP1
INTERSECT
SELECT DISTINCT numDept FROM EMP2 ;
```

Opérateur MINUS

- L'opérateur MINUS permet d'extraire des données présentes dans une première table sans être présentes dans la deuxième table



Opérateur MINUS

EMP1 (numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

EMP2(numEmp, nomEmp, prenomEmp, salaire, poste, numDept)

Requête : Afficher la liste des départements qui ont des employés dans la première filiale mais pas dans la deuxième.

```
SELECT DISTINCT numDept FROM EMP1
MINUS
SELECT DISTINCT numDept FROM EMP2 ;
```