Introduction aux bases de donnés

Les sous-requêtes

Requêtes imbriquées ou sous-requêtes

Plan de la séance

- Retour sur la dernière séance:
  - Point de vue de l'étudiant
  - Point de vue de l'enseignant.
- Rappels:
  - La commande SELECT
- Requêtes imbriquées: Définition
  - Dans la clause WHERE (SELECT, DELETE, UPDATE)
  - Dans la clause FROM
  - Dans la clause SET de la commande UPDATE,
  - La clause VALUES de la commande INSERT
  - Dans la commande CREATE.

Rappel, la commande SELECT

La commande SELECT

- La commande SELECT est la commande la plus simple à utiliser avec SQL. Cette commande n'affecte en rien la base de données et permet d'extraire des données d'une ou plusieurs tables.
  - On peut utiliser un SELECT sur une seule TABLE
  - On peur écrire des requêtes SELECT sur plusieurs tables → On utilise des jointures
  - On peut utiliser des requêtes SELECT avec des fonction de groupement
  - On peut écrire des requêtes SELECT utilisant des jointures et des fonctions de groupement

Requêtes imbriquées (sous-Requête), Définition

- Une sous requête est une requête avec la commande **SELECT** imbriquée avec les autres commandes (SELECT, UPDATE, INSERT DELETE et CREATE)
- Une sous-requête, peut être utilisée dans les clauses suivantes :
  - La clause WHERE d'une instruction UPDATE, DELETE et SELECT
  - La clause FROM de l'instruction SELECT
  - La clause VALUES de l'instruction INSERT INTO
  - La clause SET de l'instruction UPDATE
  - L'instruction CREATE TABLE ou CREATE VIEW
- On utilise les sous-requête lorsque les jointures ne sont pas possibles

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

- Ce type de sous-requête permet de comparer une valeur de la clause WHERE avec le résultat retourné par une sous-requête, dans ce cas on utilise les opérateurs de comparaison suivant :

  =, ! = , < , <= ,> ,>= , IN.

- L'écriture d'une telle requête pourrait se présenter sous la syntaxe suivante

```
SELECT colonne1, colonne2, colonnex .....
FROM  nom_tableA
WHERE colonnex  =
                  (
                    SELECT colonnex from nom_tableB....
                  .. Suite requête
                  )
Suite requête
```

Voir explications acétate suivante

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

1. La requête en vert est appelée sous-requête
2. Une sous-requête est obligatoirement entre parenthèses.
3. La colonne du WHERE de la première requête est obligatoirement la colonne du SELECT de la sous-requête. Une fonction de groupement pourrait être appliquée à la colonnex de la sous-requête : voir Exemple 2
4. La première requête et la sous-requête pourraient (et généralement c'est le cas) porter sur la même table. Dans ce cas nom_tableA est identique à nom_tableB: voir exemple 1
5. L'opérateur = pourrait être remplacé par un des opérateurs de l'acétate précédente : voir exemple 4
6. Dans le premier SELECT, colonnex n'est pas obligée d'être dans le SELECT

```
SELECT colonne1, colonne2, colonnex .....
FROM  nom_tableA
WHERE colonnex  =
                  (
                    SELECT colonnex from nom_tableB....
                  .. Suite requête
                  )
Suite requête
```

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

Exemple 1

Voici le contenu de la table syemp de votre labo1.

La question: On cherche le nom des employés qui travaillent dans le même département que l'employé FORD

Étape 1: on cherche le département (deptno) de l' employé FORD

Étape 2: on cherche le ename des employés ayant le même deptno que FORD

```
SELECT ename   FROM syemp
WHERE deptno =
                    (
                      SELECT deptno FROM syemp
                      WHERE ename ='FORD'
                    );
```

syemp

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	2450	10
7566	JONES	2975	20
7902	FORD	3000	20
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7654	MARTIN	1250	30
7844	TURNER	1500	30
7900	JAMES	950	30
7934	MILLER	1300	10

Remarque

Les deux requêtes portent sur la même table: syemp.

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

Exemple 2

Voici le contenu de la table syemp de votre labo1.

La question: On cherche le nom de l'employé qui a le plus haut salaire. :**MAX(sal)**

L'idée est d'aller chercher le plus haut salaire (MAX), puis chercher le nom de l'employé à qui ça correspond.

Étape 1: on cherche le salaire le plus élevé:

Étape 2: on cherche le ename avec le plus haut salaire.

```
SELECT ename   FROM syemp
WHERE sal =
            (
           SELECT max(sal) FROM syemp
            );
```

syemp

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	2450	10
7566	JONES	2975	20
7902	FORD	3000	20
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7654	MARTIN	1250	30
7844	TURNER	1500	30
7900	JAMES	950	30
7934	MILLER	1300	10

Attention:

Les deux requêtes portent sur la même table: syemp.

Il y a une fonction de groupement sur la colonne sal de la sous-requête.

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

Exemple 3

Voici le contenu des tables syemp et sydept de votre labo1.

La question: On cherche le nom, ename des employés du département SALES.

Étape 1: On cherche le deptno du département SALES

Étape 2, On cherche les employés ayant le deptno

```
SELECT ename SELECT syemp
WHERE deptno =
            (
            SELECT deptno FROM sydept
            WHERE dname ='SALES'
            );
```

Remarque: la requête précédente aurait pu s'écrire avec une jointure.

Attention: Il faut toujours utiliser des jointures à la place de sous requête. C'est une obligation

syemp

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	2450	10
7566	JONES	2975	20
7902	FORD	3000	20
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7654	MARTIN	1250	30
7844	TURNER	1500	30
7900	JAMES	950	30
7934	MILLER	1300	10

sydept

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

Exemple 4

Voici le contenu de la table syemp de votre labo1.

La question: On cherche le nom des employés qui travaillent dans le même département que l'employé FORD ou l'employé MILLER

La sous-requête renvoie plus qu'une valeur, elle renvoie le deptno 20 et le deptno 10. Dans ce cas il faut utiliser IN à la place de =

```
SELECT ename   FROM syemp
WHERE deptno IN
                    (
                    SELECT deptno FROM syemp
                    WHERE ename ='FORD' OR ename ='MILLER'
                     );
```

syemp

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	2450	10
7566	JONES	2975	20
7902	FORD	3000	20
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7654	MARTIN	1250	30
7844	TURNER	1500	30
7900	JAMES	950	30
7934	MILLER	1300	10

Attention La requête interne renvoie plus qu'un résultat. Il faut utiliser IN dans ce cas.

Si = est utilisé dans la requête alors vous aurez cette erreur:

```
ORA-01427: sous-requête ramenant un enregistrement de plus d'une ligne
01427. 00000 -  "single-row subquery returns more than one row"
```

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

Les opérateurs ANY et ALL

- Ces deux opérateurs s'utilisent lorsque la sous requête retourne plus qu'un enregistrement.
- On utilise l'opérateur ANY pour que la comparaison se fasse pour toutes les valeurs retournées. Le résultat est vrai si **au moins une des valeurs** répond à la comparaison
- On utilise l'opérateur ALL pour que la comparaison se fasse pour toutes les valeurs retournées. Le résultat est vrai si **toutes les valeurs** répondent à la comparaison

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

Exemple 5

La question: Nous souhaitons chercher le nom des employés du département numéro 10 dont le salaire est plus grand que TOUS les employés du département 30

```
SELECT ename FROM syemp
WHERE deptno =10 AND sal > ALL
                    (
                     SELECT sal FROM syemp
                     WHERE deptno=30
                     );
```

Le résultat de cette requête sera KING, car la sous-requête (les salaires du département 30) est donné ci-après: ----------→

La requête principale va chercher les employés du département 10 dont le salaire est supérieur à tous les salaires de la liste précédente. **Seul KING** répond à ce critère.

syemp

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	2450	10
7566	JONES	2975	20
7902	FORD	3000	20
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7654	MARTIN	1250	30
7844	TURNER	1500	30
7900	JAMES	950	30
7934	MILLER	1300	10

Salaires du département 30

SAL
2850
1600
1250
1250
1500
950

Sous-requêtes : Dans la clause WHERE
Cas, d'une Requête SELECT

Exemple 6

La question: Nous souhaitons chercher le nom des employés du département numéro 10 dont le salaire est plus grand que n'importe lequel des salaires du département 30

```
SELECT ename FROM syemp
WHERE deptno =10 AND sal > ANY
                    (
                     SELECT sal FROM syemp
                     WHERE deptno=30
                     );
```

Le résultat de cette requête sera KING, car la sous-requête (les salaires du département 30) est donné ci-après: ----------→

La requête principale va chercher les employés du département 10 dont le salaire est supérieur à n'importe lequel des salaires de la liste précédente. Le résultat sera:

ENAME
KING
CLARK
MILLER

syemp

EMPNO	ENAME	SAL	DEPTNO
7839	KING	5000	10
7698	BLAKE	2850	30
7782	CLARK	2450	10
7566	JONES	2975	20
7902	FORD	3000	20
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7654	MARTIN	1250	30
7844	TURNER	1500	30
7900	JAMES	950	30
7934	MILLER	1300	10

Salaires du département 30

SAL
2850
1600
1250
1250
1500
950

Sous-requêtes : Dans la clause FROM de la commande SELECT

- Parfois, nous avons besoin d'extraire des informations à partir de résultats calculés, ces résultats sont ramenés par une requête SELECT. Dans ce cas nous parlons de requêtes imbriquées dans le FROM.

La question: nous souhaitons extraire le nom des employés ayant les 3 meilleurs salaires, donc les trois premières lignes (KING, FORD, JONES)

Dans ORACLE, la colonne qui indique le numéro de ligne est appelée **ROWNUM**

```
SELECT ename
FROM
        (
           SELECT ename , sal, deptno FROM syemp
           ORDER BY sal DESC
        )
WHERE ROWNUM <=3;
```

Syemp, ordonnée par SAL DESC

	ENAME	SAL	DEPTNO
1	KING	5000	10
2	FORD	3000	20
3	JONES	2975	20
4	BLAKE	2850	30
5	CLARK	2450	10
6	ALLEN	1600	30
7	TURNER	1500	30
8	MILLER	1300	10
9	MARTIN	1250	30
10	WARD	1250	30
11	JAMES	950	30
12	SMITH	800	20

Sous-requêtes : Dans la clause SET d'une requête UPDATE

- On peut chercher des valeurs existantes dans la base de données afin de mettre à jour des données d'une table.
- La sous-requête est une requête SELECT qui ne doit ramener qu'un seul résultat.

La question: Nous souhaitons mettre à jour les commissions null, par la moyenne de toutes les commissions. La requête serait:

```
UPDATE emp SET comm =
                  (
                    SELECT AVG(comm) FROM emp
                  )
WHERE comm IS NULL ;
```

Les commissions (comm) a NULL seront mises à jour par la valeur: 550 qui est la moyennes de toutes les comm

emp

	ENAME	SAL	COMM
1	KING	5000	(null)
2	BLAKE	2850	(null)
3	CLARK	2450	(null)
4	JONES	2975	(null)
5	FORD	3000	(null)
6	SMITH	800	(null)
7	ALLEN	1600	300
8	WARD	1250	500
9	MARTIN	1250	1400
0	TURNER	1500	0
1	JAMES	950	(null)
2	MILLER	1300	(null)

Sous-requêtes : Dans la clause VALUES d'une requête INSERT INTO

- Ce type de sous-requête permet d'insérer des données dans une table à partir d'une autre table. La sous requête est utilisée à la place de la clause **VALUES** de la requête principale et peut retourner plusieurs résultats.

```
INSERT INTO ETUDIANTS (numad,nom)
                                        (
                                         SELECT empno,ename FROM syemp
                                        WHERE deptno =10
                                        );
```

Attention !

Lors de l'insertion, les contraintes d'intégrité doivent être respectées

Sous-requêtes : Dans la commande CREATE TABLE

- Ce type de requête est utilisé pour créer une table à partir d'une ou plusieurs table existantes. La table créée contient les données issues d'une ou de plusieurs table.
- Pour créer une table avec une sous requête, on utilise la syntaxe:

CREATE TABLE …. AS (SELECT …)

Exemple 1, La table emp contient exactement les mêmes colonnes et les mêmes données que la table syemp.

```
CREATE TABLE emp AS
                    (
                     SELECT * FROM syemp
                    );
```

Quel est l'intérêt de dupliquer ainsi une table ?

Dans ce cas précis, nous avons plus de privilèges sur la table emp. (nous pouvons faire des UPDATE, INSERT, DELETE sur emp mais pas sur syemp)

Sous-requêtes : Dans la commande CREATE TABLE

Exemple 2, regrouper les données éparpillée.

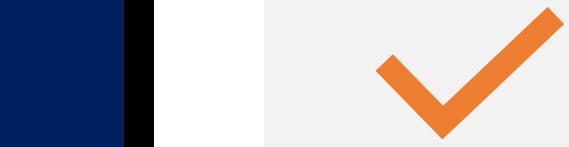
```
CREATE TABLE notesKB6 (nomEtudiant, PrenomEtudiant,titreCours,noteKB6) AS
      SELECT nom, prenom,titrecours,note
      FROM ((etudiants E
         INNER JOIN resultats R  ON E.numad=R.numad)
         INNER JOIN cours C ON C.codecours=R.codecours)
      WHERE titrecours ='Introduction aux bases de données'
      ORDER BY note DESC;
```

Nous avons utilisé les tables: Etudiants, Cours et Resultats pour créer les la tables notesKB6

Les colonnes de la table notesKB6 sont différentes des colonnes des tables sources .

Attention ! La modification des données des tables Etudiants, Cours et Resultats n'implique pas la modifications de la table noteKB6.

Les sous-requêtes



Conclusion

Questions