SMI-S5

SGBD ORACLE : Requêtes imbriquées

Pr. Issam QAFFOU

Laboratoire Ingénierie des Systèmes d'Information
Département d'Informatique
FSSM-UCA

Sous-Requêtes

- Il est possible de faire appel dans une requête principale à une **autre requête SELECT** appelée **sous-requête**.
- La sous-requête doit obligatoirement être délimitée par des parenthèses.
- Seules les **colonnes** de la table qui se trouve au niveau du **premier SELECT peuvent être extraites**.
- On les appelle sous-requête, requêtes imbriquées ou jointures procédurales.
- Une requête imbriquée peut renvoyer trois types de résultats :
  - une valeur scalaire
  - une colonne
  - une table

Sous requêtes renvoyant une valeur scalaire

- Le résultat d'une requête est dit scalaire s'il comporte une seule ligne et une seule colonne. Par exemple :

```
SQL> SELECT COUNT(*) FROM Fournisseur;
```

- On peut placer dans une requête une sous-requête calculant un résultat scalaire. Un tel type de sous-requête se place:
  - soit comme une colonne supplémentaire,
  - soit comme une valeur servant à évaluer des conditions (WHERE ou HAVING).
- On peut ajouter une colonne (colonne fictive) dans une requête, et choisir comme valeurs pour cette colonne le résultat d'une requête.
- Ce type de requête est souvent une alternative à GROUP BY.

Sous requêtes renvoyant une valeur scalaire

- Exemple: la requête suivante renvoie, pour tout produit, le nombre de fournisseurs proposant ce produit :

```
SQL> SELECT nomprod, (SELECT COUNT(*)
  2                   FROM PROPOSER PR
  3                   WHERE PR.numprod = P.numprod)
  4                    AS NB_FOURNISSEURS
  5  FROM PRODUIT P;
```

NOMPROD	NB_FOURNISSEURS
Roue de secours	1
Poupee Batman	2
Cotons tiges	1
Cornichons	0

Sous requêtes renvoyant une valeur scalaire

Conditions complexes

- On peut construire une condition en utilisant le résultat d'une requête. Par exemple, recherchons les noms des fournisseurs proposant le plus de produits.
- Déclarons d'abord une vue contenant le nombre d'articles proposés par chaque fournisseur,

```
SQL> CREATE VIEW NB_PROD_PAR_FOU AS
  2  SELECT numfou, (SELECT COUNT(*)
  3          FROM PROPOSER P
  4          WHERE P.numfou = F.numfou)
  5          AS NB_PROD
  6  FROM FOURNISSEUR F;
```

```
SQL> SELECT nomfou
  2  FROM FOURNISSEUR F, NB_PROD_PAR_FOU N
  3  WHERE F.numfou = N.numfou
  4  AND NB_PROD = (SELECT MAX(NB_PROD)
  5          FROM NB_PROD_PAR_FOU);
```

Sous requêtes renvoyant une valeur scalaire

INSERT et UPDATE

- On peut placer dans des instructions de mises à jour ou d'insertions des requêtes imbriquées.
- Par exemple,

```
SQL> INSERT INTO PERSONNE (numpers, nom, prenom)
  2  VALUES ((SELECT MAX(numpers) + 1 FROM PERSONNE),
  3    'Darth', 'Vador');
```

```
SQL> UPDATE PERSONNE SET
  2    pere = (SELECT numpers
  3       FROM PERSONNE
  4       WHERE nom = 'Socrate'
  5       AND prenom IS NULL),
  6    mere = (SELECT numpers
  7       FROM PERSONNE
  8       WHERE nom = 'Fabian'
  9       AND prenom = 'Lara')
 10   WHERE numpers = (SELECT numpers
 11          FROM PERSONNE
 12          WHERE nom = 'Darth'
 13          AND prenom = 'Vador');
```

Sous-requête renvoyant une colonne

- Lorsque la sous-requête renvoie une colonne de valeurs, la requête principale peut utiliser un opérateur de comparaison classique accompagné d'un opérateur multi-lignes.

0pérateurs multi-lignes

- **IN** compare un élément à une donnée quelconque d'une **liste** ramenée par la sous-interrogation.

- **ANY** compare l'élément à, au moins, une des données ramenées par la sous-requête.
  =ANY équivaut à **IN**.
  <ANY équivaut à « inférieur au maximum ».
  >ANY équivaut à « supérieur au minimum ».

- **ALL** compare l'élément à tous ceux ramenés par la sous-requête.
  <ALL signifie « inférieur au minimum »
  >ALL signifie « supérieur au maximum ».
  <>ALL équivaut à **NOT IN**

Sous-requête renvoyant une colonne

- Afficher les commandes des clients dont la date de naissance est le 21/03/1970 ? (utiliser l'opérateur IN)

```
SELECT *
FROM commandes
WHERE client IN ( SELECT numero
                  FROM clients
                  WHERE dateNaiss = '21/03/1970' );
```

Sous-requête renvoyant une colonne

- Afficher les articles dont le prix est supérieur à tous ceux des articles bleus ? (l'opérateur ALL)

```
SELECT  *
FROM articles
WHERE prix >=ALL ( SELECT prix
                   FROM articles
                   WHERE couleur= 'bleu' );
```

Sous-requête renvoyant une colonne

- Afficher les articles dont le prix est supérieur à celui de l'un des articles bleus ?

```
SELECT  *
FROM articles
WHERE prix >=ANY ( SELECT prix
                   FROM articles
                   WHERE couleur= 'bleu' );
```

Sous-requête renvoyant plusieurs colonnes

- Une sous-requête qui renvoie plusieurs colonnes peut être utilisée avec un opérateur de comparaison entre couple de valeur, triplet …

Exemple

- Afficher pour chaque article, le ou les numéro de client qui ont commandé la plus grande quantité et cette quantité ?

```
SELECT  a.article, a.quantite, b.client
FROM lignes_commande a, commandes b
WHERE a.article=b.numero
AND (a.article, a.quantite) IN
                    ( SELECT article, MAX(quantite)
                    FROM lignes_commande
                    GROUP BY article);
```

Opérateur EXISTS

- **EXISTS** retourne :
  - ✓ TRUE dès le premier enregistrement trouvé.
  - ✓ FALSE si aucun enregistrement n'est extrait par la sous-interrogation.

- **NOT EXISTS** retourne la valeur TRUE si aucun enregistrement n'est extrait par la sous-interrogation.

Opérateur EXISTS

- Afficher les clients qui ont commandé le 13 Mai 2010 ?

```
SELECT  a.*
FROM client a
WHERE   EXISTS  ( SELECT *
                  FROM  commandes b
                  WHERE  a.numero= b.client
                  AND b.date = '13/5/2010'  );
```

Opérateur EXISTS

Afficher les commandes sur lesquelles figurent tous les articles?

Equivalent à afficher les commandes pour lesquelles il n'existe pas d'articles qui n'y figurent pas

```
SELECT  *  FROM commandes a
WHERE  NOT EXISTS ( SELECT *
                    FROM  articles b
                    WHERE  NOT EXISTS  ( SELECT *
                                         FROM ligne_commandes c
                                         WHERE  c.article=b.numero
                                         AND c.commande = a.numero ));
```

Sous requêtes corrélées

- Une sous-requête peut être de deux types :
  - simple : elle évaluée avant la requête principale.
  - corrélée : elle est évaluée pour chaque ligne de la requête principale.
- Par exemple, la requête suivante renvoie le nombre de produits livrés pour chaque fournisseur. Elle contient une sous-requête corrélée.

```
SQL> SELECT numfou,
  2       (SELECT SUM(qte)
  3        FROM DETAILLIVRAISON D
  4        WHERE D.numfou = F.numfou
  5          ) NB_PROD_L
  6  FROM FOURNISSEUR F;
```

NUMFOU	NB_PROD_L
1	45
2	
3	10
4	

Sous requêtes corrélées

- Cette même requête, une fois évaluée, peut servir de requête non corrélée si on souhaite connaître les noms de ces fournisseurs :

```
SQL> SELECT nomfou, NB_PROD_L
  2  FROM FOURNISSEUR F,
  3         (SELECT numfou,
  4            (SELECT SUM(qte)
  5             FROM DETAILLIVRAISON D
  6             WHERE D.numfou = F.numfou
  7            ) NB_PROD_L
  8          FROM FOURNISSEUR F
  9          ) L
 10  WHERE F.numfou = L.numfou;
```

NOMFOU	NB_PROD_L
f1	45
f2	
f3	10
f4	