BD01 et WGBD : Travaux pratiques 10

(Requêtes imbriquées avec EXISTS() et opérations ensemblistes)

Théorie : les sous-requêtes

- Qu'est ce qu'une sous requête ?

  Une « sous requête » est une requête *incorporée* dans une requête.

  C'est une requête qui se trouve à l'intérieur d'une autre requête.

- A quoi servent les sous requêtes ?

  L'utilisation de sous requête permet de raisonner autrement sa solution.

  Avant de parler des sous-requêtes, nous nous sommes basés sur le fonctionnement des opérateurs relationnels pour construire les requêtes. Nous avons réfléchi ces requêtes en terme de fonctionnement de opérateurs relationels. La formulation de ces requêtes se faisait en expliquant au système **comment il devait construire** le résultat de la requête.

  On lui disait en quelque sorte : **fait** une jointure, ensuite **fait** une sélection d'après ce critère, et après **fait** ceci … Le système analysait ce qu'on lui demandait de faire et optimisait la manière de faire.

  Les sous requêtes, un autre manière de réfléchir les requêtes

  Une autre façon de construire une requête consiste à décrire ce que l'on souhaite comme résultat sans pour autant expliquer comment l'obtenir : on va exprimer les caractéristiques du résultat qu'on veut obtenir. Cette expression sera ensuite traduite par le système en une suite d'opérations relationnelles.

  Les sous requêtes serviront à exprimer les caractéristiques de ce qu'on veut obtenir, c'est à dire les conditions pour que les lignes de la requête principale (le tuple) fasse partie du résultat cherché.

- Exemple : ***On cherche l'identifiant et la ville des fournisseurs qui fournissent la pièce P2.***

  Raisonnement sans sous-requête : on décrit ce que le système doit faire

  1. Faire la jointure S et SPJ sur id_S ensuite
  2. Faire une restriction en ne gardant que les tuples pour lesquels id_P='P2'
  3. Faire une projection en ne gardant que id_S et city.

```
SELECT spj.id_s, s.city
FROM spj JOIN s ON spj.id_s=s.id_s
WHERE spj.id_p='P2'
```

  Raisonnement avec sous-requête : on va décrire ce qu'on veut garder.

  *« **Ne garder** parmi les identifiants de fournisseur et les villes **de la table S** que les lignes pour lesquelles il existe au moins une livraison dans la table SPJ qui porte le numéro du fournisseur de cette table S et dont la pièce livrée est 'P2' »*

```
SELECT id_s, s.city
FROM S
WHERE EXISTS(SELECT *
             FROM SPJ
             WHERE SPJ.id_s =S.id_s
               AND SPJ.id_p='P2')
```

- Comment s'exécute cette requête comportant une sous requête ?

Une sous requête est un prédicat. Un prédicat est une expression qui est vraie ou fausse. Toutes les lignes de la requête principale, pour lesquelles la sous-requête donne une évaluation à vrai font partie de la solution.

Fonctionnement de la requête avec sous requête de l'exemple :

```
SELECT id_s, s.city
FROM S
WHERE EXISTS(SELECT *
             FROM SPJ
             WHERE SPJ.id_s =S.id_s
               AND SPJ.id_p='P2')
```

Extrait de la table S

S1	Smith	20	London
S2	Jones	10	Paris

Extrait de la table SPJ

S1	P1	J1	200	5/10/2001
S1	P1	J4	700	10/05/2001
S2	P3	J1	400	20/05/2001
S2	P3	J2	200	30/07/2000
S2	P3	J3	200	10/05/2001
S2	P3	J4	500	03/10/2001

- Le système extrait la première ligne de la requête principale « **S1**, London » et exécute la sous-requête en y remplaçant S.ID_S par **S1**.
  Le résultat de l'exécution de la sous-requête donne une table vide puisqu'il n'existe pas de livraison faite par S1 concernant la pièce P2. La condition EXISTS est donc fausse. Le système conclut que S1,London ne fait pas partie de la réponse.
- Le système extrait ensuite la deuxième ligne de la requête principale « S2, Paris » et exécute la sous-requête en remplaçant S.ID_S par **S2**.
  Le résultat de l'exécution de la sous requête donne une table vide puisqu'il n'existe pas de livraison faite par S2 concernant la pièce P2. La condition EXISTS est donc fausse. Le système conclut que S2,Paris ne fait pas partie de la réponse
- La requête principale extrait à un certain moment « S5, Athens » de la table S et exécute la sous-requête en remplaçant S.ID_S par **S5**.
  Le résultat de l'exécution de la sous-requête donne une table **non vide** puisqu'il existe des livraison faite par S5 concernant la pièce P2. La condition EXISTS est donc vraie. Le système conclut que S5,Athens fait partie de la réponse

- Le quantificateur **EXISTS** sert à tester si un ensemble est vide ou non

EXISTS(sous requête) retourne vrai s'il existe une ligne dans la table résultat de la sous-requête.
EXISTS(sous requête) retourne faux s'il n'existe aucune ligne dans la table résultat de la sous-requête

Comme EXISTS sert à tester l'existence de lignes dans une table, la sous requête sera du genre `SELECT * FROM …` : en ne spécifiant pas le nom d'un champ dans le SELECT, on laisse l'optimiseur choisir sur quelle colonne il est préférable de travailler.

Exercices

Avant d'écrire les requêtes suivantes en SQL,

- exécutez le script fourni (*fournisseurs*).
- exécutez la commande suivante :

```
SET GLOBAL sql_mode = "ONLY_FULL_GROUP_BY,STRICT_ALL_TABLES";
```

Requêtes imbriquées avec EXISTS

Même si certaines requêtes sont réalisables ns requête imbriquée, utilisez-en dans les exercices suivants.

1. On souhaite connaître les identifiants de projets pour lesquels on a fourni au total plus de 1000 pièces.
2. On souhaite obtenir les noms de fournisseurs de Londres qui ont effectué au moins une livraison dont la quantité de pièces est supérieur à 500 unités.

Union

3. On souhaite obtenir toutes les villes pour lesquelles au moins un fournisseur, une pièce ou un projet est situé.

Différences

Pour les requêtes suivantes, proposez deux versions : une avec IN et une avec EXISTS

4. Quels sont les identifiants de fournisseurs qui ne livrent aucune pièce bleue ?
5. On souhaite connaître le nombre de livraisons de moins de 350 unités pour l'ensemble des fournisseurs qui n'ont jamais fourni à Paris.
6. On souhaite connaître les fournisseurs qui n'ont jamais fourni plus de 650 pièces identiques au total de toutes leurs livraisons. (C'est à dire ceux dont aucune pièce de leur stock n'a baissé de 650 unités).

Intersections

Pour les requêtes suivantes, proposez deux versions : une avec IN et une avec EXISTS

7. On souhaite connaître les fournisseurs qui ont fait au minimum 4 livraisons représentant au moins trois pièces différentes.
8. On souhaite les fournisseurs qui fournissent au moins dans trois villes différentes et dont les pièces qu'ils fournissent proviennent d'au moins deux villes différentes.