Requêtes complexes sous SQL

Requêtes imbriquées, agrégations, regroupement

Dr N. BAME

SQL : Requêtes imbriquées

Requête imbriquée dans la clause WHERE d'une requête externe :

```
SELECT …
FROM   …
WHERE  [Opérande] Opérateur (SELECT …
                             FROM …
                             WHERE …)
```

3 imbrications possibles :

- **(A_1 ,…A_n) IN <sous-req>** exprime une inclusion ensembliste
- **EXISTS <sous-req>** exprime une condition d'existence
- **(A_1 ,…A_n) <comp> [ALL|ANY] <sous-req>**
  exprime comparaison avec quantificateur (ANY par défaut)

Opérateur IN

SELECT …

FROM …

WHERE (A_1 ,…,A_n) [NOT] IN (SELECT B_1 ,…,B_n

FROM …

WHERE …)

Sémantique : la condition est vraie si le n-uplet désigné par *(A_1 ,…, A_n) de* la requête ***externe*** **appartient** (n'appartient pas) au résultat de la requête ***interne***.

Exemple avec IN

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans des villes où il y a des projets de budget inférieur à 50?

- Les projets de Kaolack qui ont les mêmes nom et budget d'un projet de Dakar?

Exemple avec IN

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans des villes où il y a des [n'y a pas de] projets de budget inférieur à 50?

```
SELECT Ename
FROM Emp
WHERE City IN (SELECT City
               FROM  Project
               WHERE Budget < 50)
```

Imbrication de requêtes avec IN : remarques

- La requête interne est effectuée dans un premier temps et son résultat est stocké temporairement en RAM (ou sur disque si trop gros!)

- La requête interne n'utilise pas la (les) table(s) de la requête externe

- Le IN étant ensembliste, les n-uplets retournés par la requête interne doivent être de même format que les n-uplet avant le IN (même nombre d'attributs et de même domaine, pas forcément de même nom)

- Le SELECT de la requête externe ne peut retourner que des attributs appartenant aux tables placées dans son FROM

Opérateur EXISTS

Q

```
SELECT ...
FROM ...
WHERE
```

[NOT] EXISTS

Q'

```
(SELECT *
 FROM ...
 WHERE P)
```

- ***Sémantique***

  Pour chaque n-uplet **x** de la requête externe **Q**, exécuter la requête interne Q'; **s'il existe au moins** un [*s'il n'existe aucun*] n-uplet y dans le résultat de la requête interne, alors sélectionner x

Imbrication avec EXISTS

- Les deux requêtes sont corrélées : la condition **P** dans la requête interne **Q'** exprime une jointure entre les tables de **Q'** et les tables de la requête externe **Q**.

- => requête interne exécutée pour chaque tuple de la requête externe

  - Seule l'existence d'un résultat importe, d'où le * dans le SELECT de la requête interne

- Le SELECT de la requête externe ne peut retourner que des attributs appartenant aux tables placées dans son FROM

Exemples avec EXISTS

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville où il y a un projet?

Exemples avec EXISTS

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville où il y a un projet?

```
SELECT Ename FROM Emp
WHERE EXISTS  (SELECT  *
                FROM    Project
                WHERE   Emp.City=Project.City)
```

Pour chaque employé on va vérifier si un projet est localisé dans sa ville.

Si oui, EXISTS vaut vrai et on retourne le nom de l'employé

Exemples avec EXISTS

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville sans projet ?

Exemples avec EXISTS

Emp(Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville sans projet ?

```
SELECT Ename FROM Emp
WHERE NOT EXISTS  (SELECT  *
                   FROM    Project
                   WHERE   Emp.City=Project.City)
```

- Impossible à faire avec condition dans le WHERE

- Exercice : exprimer cette requête à l'aide d'un NOT IN

Exemples avec EXISTS

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des projets ayant le plus grand budget?

Exemples avec EXISTS

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des projets ayant le plus grand budget?

```
SELECT Pname FROM Project P1
WHERE NOT EXISTS (SELECT  *
                  FROM    Project P2
                  WHERE   P1.Budget < P2.Budget)
```

ALL/ANY

SELECT …

FROM …

WHERE (A_1 ,…,A_n) θ ALL/ANY (SELECT B_1 ,…,B_n

FROM …

WHERE …)

- On peut utiliser une comparaison θ ϵ {=, <, <=, >, <=, <>} et ALL () ou ANY () : la condition est alors vraie si la comparaison est vraie pour tous les n-uplets /au moins un n-uplet de la requête interne

- Comment peut-on exprimer IN avec ALL/ANY

Imbrication avec ALL/ANY

Comme IN

- La requête interne est effectuée dans un premier temps
- La requête interne n'utilise pas les tables de la requête externe
- Les n-uplets retournés par la requête interne doivent être au même format que le nuplet avant le ANY/ALL
- Le SELECT de la requête externe ne peut retourner que des attributs appartenant aux tables placées dans son FROM

Le ANY/ALL est obligatoire devant le comparateur car le SELECT interne retourne plusieurs nuplets auxquels on veut comparer un seul nuplet (sauf si la requête interne retourne un seul nuplet)

Alors que le IN ne couvre que l'égalité (appartenance ensembliste), le ANY/ALL permet de faire des inégalités

Exemple avec “ANY”

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des projets lyonnais de budget supérieur à (au moins) un budget de projet parisien?

Exemple avec “ANY”

Emp(Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

- Noms des projets lyonnais de budget supérieur à (au moins) un budget de projet parisien?

```
SELECT Pname
FROM Projet
WHERE City = 'Lyon'
AND Budget > ANY (SELECT Budget
                  FROM Project
                  WHERE  City='Paris')
```

Pas possible avec un IN... mais comment faire avec une jointure ?

Exemple avec “ALL”

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des professions ayant le plus petit salaire?

Exemple avec “ALL”

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des professions ayant le plus petit salaire?

```
SELECT Title
FROM Pay
WHERE Salary <= ALL (SELECT Salary
                     FROM Pay );
```

Exercice

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville où il y a un projet?

Exercice

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville où il y a un projet?

```
SELECT Ename
FROM Emp
WHERE EXISTS (SELECT *
              FROM Project
              WHERE   Emp.City=Project.City)
```

Exprimer avec un IN ou un ANY.

Exercice

Emp(Eno, Ename, Title, City)

Project(Pno, Pname, Budget, City)

Pay(Title, Salary)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville sans projet?

Exercice

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des employés qui habitent dans une ville sans projet?

```
SELECT Ename
FROM Emp
WHERE NOT EXISTS ( SELECT  *
                   FROM    Project
                   WHERE   Emp.City=Project.City)
```

Exprimer avec NOT IN ou ALL

Exercice

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Noms des projets ayant le plus grand budget?

Exercice

Emp(Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

- Noms des projets ayant le plus grand budget?

```
SELECT Pname
FROM Project P1
WHERE NOT EXISTS ( SELECT  *
                   FROM    Project P2
                   WHERE P1.Budget < P2.Budget)
```

Exprimer avec un NOT IN (si possible)

Puis avec ALL !

Equivalence IN/ANY, NOT IN/ALL

Toute requête avec un IN peut-être écrite avec un ANY:

Donner le nom des employés habitant la ville d'un ingénieur ?

```
SELECT Ename FROM Emp
WHERE City IN (SELECT City FROM Emp WHERE Title='ingénieur')
SELECT Ename FROM Emp
WHERE City = ANY(SELECT City FROM Emp WHERE Title='ingénieur')
```

Toute requête avec un NOT IN peut-être écrite avec un ALL :

Donner le nom des employés habitant une ville sans ingénieur ?

```
SELECT Ename FROM Emp
WHERE City NOT IN (SELECT City FROM Emp WHERE Title='ingénieur')
SELECT Ename FROM Emp
WHERE City <> ALL(SELECT City FROM Emp WHERE Title='ingénieur')
```

Equivalence ANY/EXISTS, ALL/NOT EXISTS

Toute requête avec un ANY peut-être écrite avec un EXISTS :
Donner les travaux durant plus qu'un travail dirigé par 'Prof' ?

```
SELECT * FROM Work
WHERE Dur > ANY (SELECT Dur FROM Work
                 WHERE Resp='Prof')
SELECT * FROM Work W1
WHERE EXISTS (SELECT * FROM Work W2
              WHERE Resp='Prof'
              AND W1.Dur > W2.Dur)
```

Toute requête avec un ALL peut-être écrite avec un NOT EXISTS :
Donner les travaux durant plus que tous les travaux dirigés par 'Prof' ?

```
SELECT * FROM Work WHERE Dur > ALL (SELECT Dur
                                   FROM Work
                                   WHERE  Resp='Prof')
SELECT * FROM Work W1
WHERE NOT EXISTS (SELECT * FROM Work W2
                  WHERE Resp='Prof'
                  AND W1.Dur < W2.Dur)
```

Fonctions d'agrégation en SQL

- Jusqu'à présent on a vu des requêtes comme suite d'opérations tuple à tuple

- Les ***fonctions d'agrégation*** permettent d'exprimer des conditions et de faire des opérations sur des **groupes de tuples**

- Avec SQL nous pouvons:
  - Partitionner une relation en groupes
  - Exprimer des relations sur des groupes
  - Agréger des valeurs sur les groupes

SQL : Fonctions d'agrégation

Elles peuvent servir à calculer une valeur numérique à partir d'une relation : **Agg(A)** où A est un d'attribut (ou *) et Agg est une fonction parmi :

- **COUNT(A)** ou **COUNT(*)** : nombre de valeurs ou n-uplets,
- **SUM(A) :** somme des valeurs,
- **MAX(A) :** valeur maximale,
- **MIN(A) :** une valeur minimale,
- **AVG(A) :** moyenne des valeurs

dans l'*ensemble de valeurs* désignées par A

Agrégation sur l'agrégat par défaut

On applique des fonctions de calcul sur le résultat d'une requête

SELECT **Agg1(A_i), …, AggN(A_j)**
FROM R_1, …, R_m
WHERE *Conditions*

- On exécute la requête (clause from et where).
- On applique les différentes fonctions Aggk sur l'ensemble du résultat.

Agrégation sur l'agrégat par défaut quelques remarques

La fonction d'agrégation **Agg** peut s'appliquer

- Sur une ligne, uniquement pour le **COUNT** : **COUNT(*),** pour compter le nombre de ligne du résultat d'une requête

- Sur un attribut : **Agg(att)** on applique la fonction aux **valeurs NON NULLES** de l'attribut att

- Sur un attribut avec *DISTINCT* : **Agg*(DISTINCT att)*** pour **appliquer la fonction aux valeurs distinctes** de l'attribut *att*

Exemples d'agrégation

Emp(Eno, Ename, #Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(#Eno, #Pno, Resp, Dur)

- Nombre d'employés parisiens ?

- Plus grand salaire et plus petit salaire de toutes les professions?

Exemples d'agrégation

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Nombre d'employés parisiens ?

```
SELECT COUNT(*)
FROM Emp
WHERE City='Paris'
```

- Plus grand salaire et plus petit salaire de toutes les professions?

```
SELECT MAX(Salary), MIN(Salary)
FROM  Pay
```

Exemples d'agrégation

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Budgets totaux des projets de Paris?

- Nombre de villes où il y a un projet avec l'employé E4?

Exemples d'agrégation

Emp(Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

- Budgets totaux des projets de Paris?

```
SELECT SUM (Budget)
FROM  Project
WHERE City = 'Paris'
```

- Nombre de villes où il y a un projet avec l'employé E4?

```
SELECT COUNT(DISTINCT  City)
FROM  Project, Works
WHERE Project.Pno = Works.Pno
AND   Works.Eno = 'E4'
```

Agrégation sur l'agrégat par défaut quelques remarques

Attention aux agrégations dans le WHERE :

```
SELECT TITLE
FROM  Pay
WHERE Salary = MAX(Salary)
```

=> cette formulation est INTERDITE !

- Pas d'attribut dans le SELECT si agrégation (sauf si GROUP BY voir après)

```
SELECT TITLE, MAX(Salary)
FROM  Pay
```

=> cette formulation est INTERDITE !

Comment faire pour connaître alors la profession payant le plus?

Fonctions d'agrégation et imbrication

Emp(Eno, Ename, Title, City)

Project(Pno, Pname, Budget, City)

Pay(Title, Salary)

Works(Eno, Pno, Resp, Dur)

Pour utiliser des attributs de tables du FROM et une agrégation, il faut souvent passer par une sous-requête :

- Noms des professions qui payent le plus (et les salaires correspondant) ?

```
SELECT Title, Salary
FROM Pay
WHERE Salary = (SELECT MAX(Salary)
                FROM Pay)
```

Fonctions d'agrégation et imbrication

Emp(Eno, Ename, Title, City)

Project(Pno, Pname, Budget, City)

Pay(Title, Salary)

Works(Eno, Pno, Resp, Dur)

- Noms des projets dont le budget est supérieur au budget moyen?

```
SELECT Pname
FROM Project
WHERE Budget > (SELECT AVG(Budget)
                FROM Project)
```

Remarque : dans le cas où la requête imbriquée retourne le résultat d'un agrégat (donc un seul résultat), pas besoin d'utiliser un IN, ANY ou ALL (mais pas faux)

Requêtes de groupement

- Pour partitionner les n-uplets résultats en fonction des valeurs de certains attributs :

SELECT A_i, …, A_n agg1, agg2, …
FROM R_1, …, R_m
WHERE P
GROUP BY A_j, …, A_k

Principes :

- On exécute la requête FROM WHERE
- On regroupe le résultat en paquets d'enregistrements en plaçant dans un paquet tous les enregistrements ayant la même valeur pour A_j, …, A_k
- On fait le SELECT en appliquant les agrégations à chaque paquet

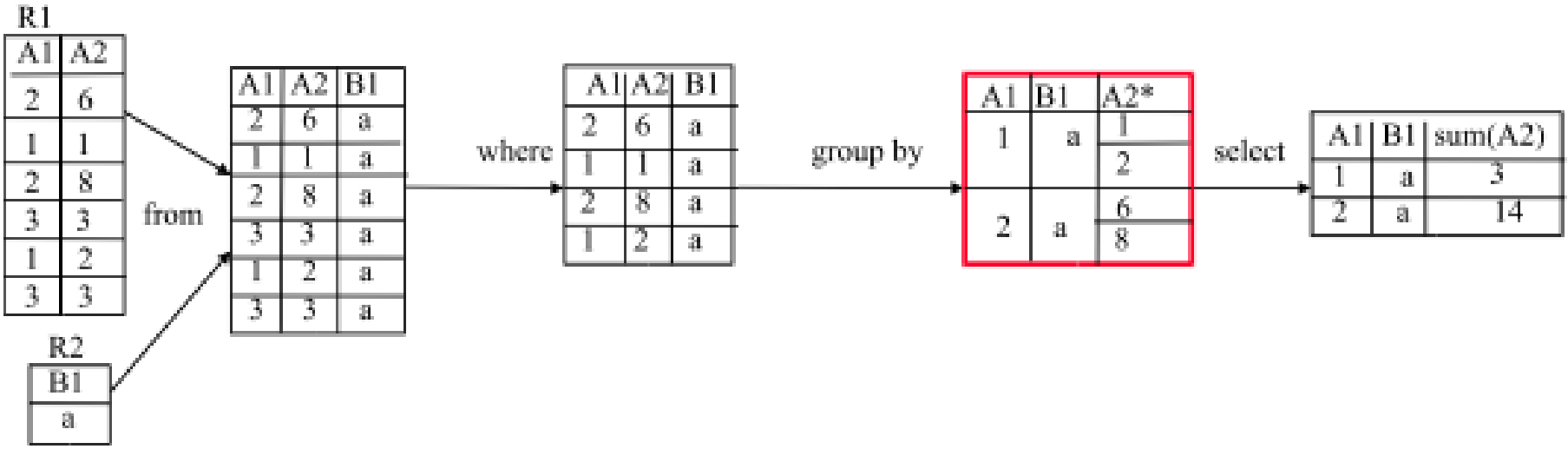
GROUP BY

SELECT A1, B1, sum(A2)

FROM R1 , R2

WHERE A1 < 3

GROUP BY A1, B1



GROUP BY : remarques

Tous les attributs (A_i, …, A_n) dans la clause SELECT :

- **Ne doivent pas apparaître** dans une **opération d'agrégation** et
- **doivent être inclus** dans l'ensemble des attributs (A_j, …, A_k) **de la clause GROUP BY** (qui peut avoir d'autres attributs en plus)

Le SELECT ne retourne qu'une seule ligne par agrégat (**paquet**)

=> Impossibilité de mettre un attribut non présent dans le GROUP BY car prend plusieurs valeurs dans un même paquet

Exemples de groupement

Emp(Eno, Ename, Title, City)

Pay(Title, Salary)

Project(Pno, Pname, Budget, City)

Works(Eno, Pno, Resp, Dur)

- Numéros des projets avec leurs effectifs ?

```
SELECT Pno, Count(Eno)
FROM  Works
GROUP BY  Pno
```

- Pour chaque ville, nombre d'employés par profession?

```
SELECT  City, Title, Count(*)
FROM  Emp
GROUP BY  City, Title
```

Exemples de groupement

Emp(Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

- Numéros des projets avec leurs effectifs Numéros et noms des projets avec la durée moyenne et la durée maximale de participation des emplyés

```
SELECT  Pno, Pname, AVG(Dur), MAX(Dur)
FROM  Works, Project
WHERE Works.Pno=Project.Pno
GROUP BY  Pno, Pname
```

=> OK pour ajouter Pname, la clé de Projet est dans GROUP BY

On a ajouté Pname dans le GROUP BY (alors que Pno suffisait pour faire les agrégats) afin de le mettre dans le SELECT

- Numéro, nom, et responsable des projets et la durée maximale de participation d'un employé à ce projet?

```
SELECT  Pno, Pname, resp, MAX(Dur)
FROM  Works, Project
WHERE Works.Pno=Project.Pno
GROUP BY  Pno, Pname
```

=>INTERDIT, pas de clé de Works dans GROUP BY

Prédicats sur des groupes

Pour garder (éliminer) les groupes (partitions) qui satisfont (ne satisfont) pas une certaine condition :

SELECT A_i , …, A_n, agg1,agg2,...
FROM R_1 , …, R_m
WHERE P
GROUP BY A_j …, A_k
HAVING Q

Principe : une fois la requête FROM-WHERE exécutée, on regroupe en agrégat avec le GROUP BY et on ne garde ensuite que les agrégats satisfaisant le HAVING

Remarques

Ordre d'exécution des clauses :

5 SELECT
1 FROM
2 WHERE
3 GROUP BY
4 HAVING
6 ORDER BY

Conséquence : à partir de 3 on manipule des agrégats donc pour 4, 5 et 6 on ne manipule que des *fonctions d'agrégations ou des attributs apparaissant dans le GROUP BY*

Exemples de groupement (1)

Emp(Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

- Villes dans lesquelles habitent plus de 2 employés?

  SELECT City
  FROM Emp
  GROUP BY City
  HAVING COUNT(ENO) > 2

- Projets demandant plus de 1000 jours/homme?

  SELECT Pno, Pname
  FROM Project, Works
  WHERE Projet.Pno=Work.Pno
  GROUP BY Pno, Pname
  HAVING SUM(Dur) > 1000

Exemples de groupement (2)

Emp(Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

- Numéros et noms des projets ayant des employés venant de plus de 5 villes différentes ?

```
SELECT Pno, Pname
FROM  Project, Works, Emp
WHERE Projet.Pno=Work.Pno AND Work.Eno=Emp.Eno
GROUP BY  Pno,Pname
HAVING COUNT(DISTINCT Emp.City) > 5
```

- Liste des employés triés par temps total de travail décroissants?

```
SELECT Eno, Ename, SUM(Dur)
FROM   Works, Emp
WHERE  Work.Eno=Emp.Eno
GROUP BY  Eno,Ename
ORDER BY SUM(Dur) DESC
```