Introduction aux Bases de données

Cours 6 : Requêtes imbriquées en SQL

UFR 919 – Licence

2e année

ALL/ANY

```
SELECT …
FROM …
WHERE (A1,…,An) θ ALL/ANY (SELECT B1,…,Bn
                           FROM …
                           WHERE …)
```

- On peut utiliser une comparaison θ∈{=, <, <=, >, >=, <>} 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?

Imbrications avec ANY/ALL : remarques

Comme le IN :

- La requête interne est effectuée dans un premier temps
- les nuplets retournés par la requête interne doivent être du 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 obligatoire devant le comparateur car le SELECT interne retourne plusieurs nuplets auxquels on veut comparer 1 nuplet (sauf si select interne retourne 1 seule valeur, voir cours suivant)

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?

```
SELECT Pname
FROM Projet
WHERE
```

Exemple avec “ALL”

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des professions ayant le plus petit salaire?

```
SELECT  Title
FROM Pay
WHERE
```


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 où il y a un projet?
Exprimer avec un IN ou un ANY

```
SELECT Ename FROM Emp
WHERE Emp.City =
```

Noms des employés qui habitent dans une ville sans projet?
Exprimer avec NOT IN ou ALL

```
SELECT Ename FROM Emp
   WHERE Emp.City <>
```


Exercice

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des projets ayant le plus grand budget?
Exprimer avec un NOT IN (si possible)
Puis avec ALL !

```
SELECT Pname FROM Project
WHERE Budget >=
```

```
SELECT Pname FROM Project
WHERE pno not in
```


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) ;
```


Jointures vs imbrications (1)

Une jointure peut-être en général ré-écrite à l'aide d'une imbrication :

- Avec IN, ANY ou EXISTS si la (les) condition(s) de jointure sont des égalités
- Avec ANY ou EXISTS si au moins une est une inégalité

Une jointure ne peut pas être ré-écrite avec une imbrication si des attributs du SELECT appartiennent à la (aux) table(s) qu'on souhaite placer dans l'imbrication

Jointures vs imbrications (2)

A propos de la différence :

- La notion de totalité (tous, aucun, personne, jamais, plus que tous, moins que tous, etc) se traduit par une différence d'ensemble
- La différence ne peut jamais être réalisée avec seulement la jointure, éventuellement avec un MINUS et des jointures dans les 2 requêtes (donc nécessite peut-être plus de tables qu'imbrication)
- Tous aucun, personne, jamais... peuvent être traduit à l'aide d'un NOT IN, ALL ou NOT EXISTS
- Plus que tous, moins que tous (ex : plus grand, plus vieux, plus cher, etc) peuvent être traduit à l'aide d'un ALL ou d'un NOT EXISTS

Différence

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des employés ne travaillant pas sur un projet?

```
SELECT Ename FROM Emp
WHERE Eno NOT IN   (SELECT  Eno
                    FROM  work W) ;

SELECT Ename FROM  emp E
WHERE NOT EXISTS   (SELECT  *
                    FROM    work W
                    WHERE   E.Eno = W.Eno) ;

SELECT Ename FROM Emp
MINUS
SELECT Ename FROM Emp,Work WHERE Emp.Eno=Work.Eno ;
```

Division

- On souhaite connaître les employés travaillant dans tous les projets
- Quels sont les projets auxquels participent toutes les professions ?

Point commun : la notion de totalité, on veut voir si des enregistrements d'une table peuvent être associés à TOUS les enregistrements d'une autre table

Division : réécriture

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

On souhaite connaître les employés ayant travaillé sur tous les projets
=> pour être dans le résultat, un employé doit avoir une entrée dans *Work* pour chacun des projets de la table *Projet*

En calcul
{e.Ename | Emp(e) ∧ ∀ p, Project(p) ⇒ ∃ w, Works(w) ∧ w.Eno=e.Eno ∧ w.Pno=p.Pno}
Mais, pas de ∀ en SQL. Je m'intéresse donc aux

employés pour lesquels il n'existe pas de projets auxquels ils n'ont pas participé

Soit une double-négation !

Exemples de division (1)

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Quels sont les employés ayant travaillé sur tous les projets?

```
SELECT Eno,Ename
FROM   Emp
WHERE NOT EXISTS (SELECT *
                  FROM Project
                  WHERE NOT EXISTS (SELECT *
                                    FROM Work
                                    WHERE Work.Eno=Emp.Eno
                                    AND Work.Pno=Project.Pno)) ;
```

Bien observer que la (les) table(s) faisant le lien entre les 2 autres tables se trouve(nt) dans la 2ème imbrication et réalise(nt) les 2 jointures

Exemples de division (2)

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Quels sont les villes dans lesquelles tous les salaires d'employés sont représentés?

```
SELECT DISTINCT City
FROM   Emp A
WHERE
```

Exemples de division (3)

Emp (Eno, Ename, Title, City)
Pay(Title, Salary)
Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

Quels sont les projets parisiens pour lesquels toutes les professions gagnant plus de 3000 euros ont participé?

```
SELECT Pno,Pname
FROM   Projet
WHERE
```


Exemples de division (4)

(Requêtes du TD 3 : calcul relationnel)

SPONSORISE (NSP, NJO, SOMME)
JOUEUR (NJO, EQ, TAILLE, AGE)
EQUIPE (NEQ, VILLE, COULEUR, STP)
MATCH (EQ1, EQ2, DATE, ST)
DISTANCE (ST1, ST2, NBKM)

17. Quelle équipe a joué dans tous les stades ?
18. Quelle équipe a joué dans tous les stades autres que son stade préféré ?
19. Quel joueur de plus de 50 ans a été sponsorisé par tous les sponsors ?
20. Quel sponsor a sponsorisé au moins un joueur pour chaque équipe?