TP 7 – CORRIGE - 1

Exercice 1 : BD employés et départements

Interrogation de la BD

0. Télécharger le script de création de la BD : employes.sql et lancer ce script de création de la BD.

1. Tous les employés travaillant dans un département qui contient au moins un 'ANALYST' (c'est un métier).

Ecrire 3 versions.

- ***Version jointure artificielle***

```
Select distinct e1.NE, e1.nom, e1.ND, e1.job
from emp e1, emp e2
where e1.ND = e2.ND
and e2.job = 'ANALYST'
;
```

- ***Version imbriquée IN***

```
Select NE, nom, ND, job
from emp
where ND in (
    select ND from emp
    where job = 'ANALYST'
);
```

- ***Version imbriquée EXISTS***

```
Select NE, nom, ND, job
from emp
where exists (
    select * from emp e1
    where job = 'ANALYST'
    and e1.ND = emp.ND
);
```

2. Tous les employés ayant le même job que les employés du département 30.

Ecrire 3 versions.

- ***Version jointure artificielle***

```
Select distinct e1.NE, e1.nom, e1.ND, e1.job
from emp e1, emp e2
where e1.job = e2.job
and e2.ND = 30
;
```

- ***Version imbriquée IN***

```
Select NE, nom, ND, job
from emp
where job in (
    select job from emp
    where ND = 30
);
```

- ***Version imbriquée EXISTS***

```
Select NE, nom, ND, job
from emp
where exists (
    select * from emp e1
    where ND = 30
    and e1.job = emp.job
);
```

3. Tous les noms et dates d'embauche des employés embauchés avant BLAKE.

Ecrire 3 versions.

- ***Version jointure artificielle***

```
Select distinct e1.NE, e1.nom, e1.datemb
from emp e1, emp e2
where e1.datemb < e2.datemb
and e2.nom = 'BLAKE'
;
```

- ***Version imbriquée IN***

```
Select NE, nom, ND, job
from emp
where datemb < any (
    select datemb from emp
    where nom = 'BLAKE'
);
```

- ***Version imbriquée EXISTS***

```
Select NE, nom, ND, job
from emp
where exists (
    select * from emp e1
    where nom = 'BLAKE'
    and emp.datemb < e1.datemb
```

```
);
```

Remarque :

La question est interprétée en considérant qu'il y a un seul BLAKE. S'il y en a plusieurs, la question est traitée en considérant qu'on prend tous les employés dont la date d'embauche est avant celle d'au moins un BLAKE. On pourrait aussi considérer qu'on veut tous les employés dont la date d'embauche est avant celles de tous les BLAKE auquel cas il faut utiliser le NOT IN ou le NOT EXISTS et auquel cas la jointure artificiel serait impossible.

4. Tous les employés ayant le même chef que ALLEN

Ecrire 3 versions.

- ***Version jointure artificielle***

```
Select distinct e1.NE, e1.nom, e1.Nechef, e2.NE, e2.nom,
e2.NEchef
from emp e1, emp e2
where e1.NEchef = e2.NEchef
and e2.nom = 'ALLEN'
and e1.nom != 'ALLEN'
;
```

- ***Version imbriquée IN***

```
Select NE, nom, NEchef
from emp
where NEchef in (
    select NEchef from emp
    where nom = 'ALLEN'
)
and nom != 'ALLEN'
;
```

- ***Version imbriquée EXISTS***

```
Select NE, nom, NEchef
from emp
where exists (
    select * from emp e1
    where nom = 'ALLEN'
    and emp.NEchef = e1.NEchef
)
and nom != 'ALLEN'
;
```

5. Tous les employés n'ayant pas le même chef que ALLEN

Ecrire 2 versions.

- ***Version imbriquée NOT IN***

```
Select NE, nom, NEchef
from emp
where NEchef not in (
    select NEchef from emp
    where nom = 'ALLEN'
);
```

La version NOT IN ne permet pas d'incorporer KING qui n'a pas de chef (NEchef à NULL).

```
Select NE, nom, NEchef
from emp
where NEchef != all (
    select NEchef from emp
    where nom = 'ALLEN'
);
```

- ***Version imbriquée NOT EXISTS***

```
Select NE, nom, NEchef
from emp
where not exists (
    select * from emp e1
    where nom = 'ALLEN'
    and emp.NEchef = e1.NEchef
)
;
```

La version NOT EXISTS permet d'incorporer KING qui n'a pas de chef (NEchef à NULL).

- ***Remarque sur le NULL :***

  - Le NULL est absorbant pour toutes les opérations arithmétique. 4*NULL vaut NULL. 0 * NULL vaut NULL.
  - Le NULL est absorbant dans les opérations logiques si la réponse est fonction de la valeur du NULL. 1 et NULL vaut NULL. 1 ou NULL vaut 1 car c'est vrai pour toute valeur donnée à la place de NULL NULL. 1 in (1, NULL) vaut 1 car c'est vrai pour tout NULL. 1 in (2, NULL) car si NULL vaut 1 c'est vrai, si NULL vaut 2 c'est faux.
  - **Opérateur <=>** : permet de considérer la valeur NULL comme une valeur quelconque. Donc 1=NULL vaut NULL tandis que 1<=>NULL vaut 0 (faux). Autrement dit, <=> c'est en quelque sorte un « = »et un « is » en même temps.

- ***Version jointure artificielle : ATTENTION : ne marche que si ALLEN est unique !!!***

```
Select distinct e1.NE, e1.nom, e1.NEchef
from emp e1, emp e2
where e1.NEchef != e2.NEchef
and e2.nom = 'ALLEN'
and e1.nom != 'ALLEN'
;
```

La version jointure artificielle ne marche que si ALLEN est unique. On ne peut donc l'utiliser que si la restriction de la deuxième table produit assurément 1 tuple et un seul.

Dans les autres cas, on obtient des résultats erronés. Donc, NOT IN et NOT EXISTS ne sont pas toujours transformables en jointure artificielle.

➢ ***Remarque sur la jointure artificielle :***

En général, on ne peut faire que des comparaisons d'égalité dans une jointure artificielle.

t1.att = t2.att <=> IN ou EXISTS

et en général :

t1.att != t2.att ou t1.att < t2.att <=> NOT IN ou NOT EXISTS

6. Changer le nom de CLARK (le 7782) et passez le à ALLEN.

```
Update emp
Set nom = 'ALLEN'
where ne=7782;
```

Relancer la requête de l'exercice précédent avec jointure artificielle et avec le NOT IN. Que constatez-vous ?

On obtient 11 tuples résultats contre 5 avec le NOT IN et 6 avec le NOT EXISTS (ça prend le président, ce qui est mieux). Les 11 tuples ce sont : tous les tuples sans les ALLEN et sans le PRESIDENT.

```
En cas de NOT IN ou de NOT EXISTS : on ne peut pas faire
de JOINTURE ARTIFICIELLE mais on peut faire un MINUS
On verra qu'on peut aussi faire une jointure externe
```

➢ ***Version avec MINUS***

Le résultat c'est tous les employés moins ceux qui ont le même chef que ALLEN

```
Select NE, nom, NEchef
from emp
MINUS
Select distinct e1.NE, e1.nom, e1.Nechef
from emp e1, emp e2
where e1.NEchef = e2.NEchef
and e2.nom = 'ALLEN'
;
```

- Repassez le 7782 à CLARK

```
Update emp
Set nom = 'CLARK'
where ne=7782;
```

7. Tous les employés n'ayant pas de subordonnées.

Ecrire 3 versions.

➢ ***Version imbriquée NOT IN***

```
Select NE, nom, NEchef
from emp
where NE not in (
    select NEchef from emp
    where NEchef is not null
);
```

- ***Version imbriquée NOT EXISTS***

```
Select NE, nom, NEchef
from emp
where not exists (
    select * from emp e1
    where NEchef is not null
    and emp.NE = e1.NEchef
);
```

- ***Version MINUS (ne marche pas avec MySQL)***

```
Select NE, nom, NEchef
from emp
minus
select Echef.NE, Echef.nom, Echef.NEchef
from emp E, emp Echef
where e.Nechef = Echef.NE
;
```

A noter qu'on ne peut pas faire de jointure artificielle.

8. Tous les départements vides avec leurs noms et villes.

Ecrire 3 versions.

- ***Version imbriquée IN***

```
Select *
from dept
where ND not in (
    select ND from emp
);
```

- ***Version imbriquée EXISTS***

```
Select *
from dept
where not exists (
    select * from emp
    where emp.ND = dept.ND
);
```

- ***Version MINUS (ne marche pas avec MySQL)***

```
Select *
from dept
```

```
minus
select dept.*
from emp e, dept d
where e.ND = d.ND
;
```

A noter qu'on ne peut pas faire de jointure artificielle.

9. Tous les employés ayant le même job et le même chef que MARTIN.

Ecrire 3 versions.

A noter qu'il pourrait y avoir plusieurs MARTIN différents.

- ***Version jointure artificielle***

```
Select distinct e1.NE, e1.nom, e1.job, e1.NEchef, e2.NE,
e2.nom, e2.job, e2.NEchef
from emp e1, emp e2
where e1.NEchef = e2.NEchef
and e1.job = e2.job
and e2.nom = 'MARTIN'
and e1.nom != 'MARTIN'
;
```

- ***Version imbriquée IN***

```
Select NE, nom, job, NEchef
from emp
where (job, Nechef) in (
    select job, NEchef from emp
    where nom = 'MARTIN'
)
and nom != 'MARTIN'
;
```

- ***Version imbriquée EXISTS***

```
Select NE, nom, job, NEchef
from emp
where exists (
    select * from emp e1
    where nom = 'MARTIN'
    and emp.NEchef = e1.NEchef
    and emp.job = e1.job
)
and nom != 'MARTIN'
;
```

10. Tous les employés travaillant à CHICAGO et ayant le même job qu'ALLEN.

Ecrire 3 versions.

- ***Version jointure artificielle***

```
Select distinct e1.NE, e1.nom, e1.job, d.ville, e2.NE,
e2.nom, e2.job
from emp e1, dept d, emp e2
where e1.ND = d.ND
and d.ville = 'CHICAGO'
and e1.job = e2.job
and e2.nom = 'ALLEN'
and e1.nom != 'ALLEN'
;
```

- ***Version imbriquée IN***

```
Select e.NE, e.nom, e.job, d.ville
from emp e, dept d
where e.ND = d.ND
and d.ville = 'CHICAGO'
and job in (
    select job from emp
    where nom = 'ALLEN'
)
and e.nom != 'ALLEN'
;
```

- ***Version imbriquée EXISTS***

```
Select e.NE, e.nom, e.job, d.ville
from emp e, dept d
where e.ND = d.ND
and d.ville = 'CHICAGO'
and exists (
    select * from emp e1
    where nom = 'ALLEN'
    and e.job = e1.job
)
and e.nom != 'ALLEN'
;
```

11. Tous les employés du département RESEARCH embauchés la même année que quelqu'un du département SALES

Ecrire 3 versions.

- ***Version jointure artificielle***

```
Select distinct e1.NE, e1.nom, e1.datemb, d1.nom
from emp e1, dept d1, emp e2, dept d2
where e1.ND = d1.ND
and e2.ND = d2.ND
and d1.nom = 'RESEARCH'
```

```
and d2.nom = 'SALES'
and year(e1.datemb) = year(e2.datemb)
;
```

On ne doit rien projeter de E2 pour éviter les doublons. Toutefois, pour vérifier, on peut projeter des informations de E2 :

```
Select distinct e1.NE, e1.nom, e1.datemb, d1.nom, e2.NE,
e2.nom, e2.datemb, d2.nom
from emp e1, dept d1, emp e2, dept d2
where e1.ND = d1.ND
and e2.ND = d2.ND
and d1.nom = 'RESEARCH'
and d2.nom = 'SALES'
and year(e1.datemb) = year(e2.datemb)
order by e1.ne, e2.ne
;
```

- ***Version imbriquée IN***

```
Select e.NE, e.nom, e.datemb, d.nom
from emp e, dept d
where e.ND = d.ND
and d.nom = 'RESEARCH'
and year(e.datemb) in (
    select year(datemb)
    from emp e, dept d
    where e.ND = d.ND
    and d.nom = 'SALES'
)
;
```

- ***Version imbriquée EXISTS***

```
Select e1.NE, e1.nom, e1.job, d1.nom
from emp e1, dept d1
where e1.ND = d1.ND
and d1.nom = 'RESEARCH'
and exists (
    select * from emp e2, dept d2
    where e2.ND=d2.ND
    and year(e1.datemb) = year(e2.datemb)
    and d2.nom = 'SALES'
)
;
```

12. Tous les employés embauchés avant tous les employés du département 10.

Ecrire trois versions.

- ***Version imbriquée ALL***

```
Select NE, nom, datemb, ND
from emp
```

```
    where datemb < ALL (
        select datemb
        from emp e
        where ND = 10
    )
    ;
```

A noter que la version « <ALL » ne marche pas avec un sql_mode à ONLY_FULL-GROUP_BY. On peut repasser à SQL_MODE= « » pour vérifier la requête.

- ***Version imbriquée NOT EXISTS***

```
    Select NE, nom, datemb, ND
    from emp
    where not exists(
        select *
        from emp e
        where ND = 10
        and emp.datemb >= e.datem
    )
    ;
```

- ***Version imbriquée MIN***

```
Select NE, nom, datemb, ND
    from emp
    where datemb < (
        select min(datemb)
        from emp e
        where ND = 10
    )
    ;
```

Pour vérifier on affichera tous les employés du 10 ordonnés par date d'embauche.

```
Select *
from emp
where nd=10
order by datemb;
```

13. Tous les employés qui gagnent plus que la moyenne classés par salaire croissant

Faire deux versions.

- ***Version imbriquée dans le where***

```
    Select NE, nom, sal
    from emp
    where sal > (
        Select avg(sal)
        from emp
    )
    order by sal;
```

➢ ***Version imbriquée dans le from***

```
Select NE, nom, sal, moySal
from emp, (Select avg(sal) moySal from emp) tmoy
where sal > moySal
order by sal ;
```

14. Tous les départements (avec leurs noms et villes) qui contiennent plus d'employés que le nombre moyen d'employés par ville.

Faire deux versions imbriqués (avec et sans imbrications dans le where)

Faire une version avec vue.

➢ ***Version imbriquée dans le where***

```
Select d.nd, d.nom, d.ville, count(*) as nbEmpParDept
from emp e, dept d
where e.nd = d.nd
group by d.nd, d.nom, d.ville
having count(*) > (
    Select avg(nbEmpParDept)
    from ( Select count(*)nbEmpParDept
           from emp Group by nd) t1
);
```

➢ ***Version imbriquée dans le from***

```
Select d.nd, d.nom, d.ville, count(*)as nbEmpParDept
,nbMoyEmpParVille
from emp e, dept d,
    (Select avg(nbEmpParVille) nbMoyEmpParVille from
           (Select count(*) nbEmpParVille
           from emp e, dept d
           where e.nd = d.nd
           Group by ville
           ) t1
    )t2
where e.nd = d.nd
group by d.nd, d.nom, d.ville
having nbEmpParDept > nbMoyEmpParVille
;
```

➢ ***Create View***

```
Create or replace view v_nbEmpParVille as
Select ville, count(*) nbEmpParVille
from emp e, dept d
where e.nd = d.nd
Group by ville ;

Select * from v_nbEmpParVille;

-- ------------------------------------------------------
Create or replace view v_nbEmpParVilleMoy as
```

```
    Select avg(nbEmpParVille) nbMoyEmpParVille
    From v_nbEmpParVille;

    Select * from v_nbEmpParVilleMoy;

-- ------------------------------------------------------------
    Create or replace view v_nbEmpParDept as
    Select d.nd, d.nom, d.ville, count(*) nbEmpParDept
    from emp e, dept d
    where e.nd = d.nd
    Group by nd;

    Select * from v_nbEmpParDept;

-- ------------------------------------------------------------
    Select *
    from v_nbEmpParDept, v_nbEmpParVilleMoy
    where nbEmpParDept  > nbMoyEmpParVille ;
```