Notes du cours de persistance de données III - Laboratoires

Nathan Furnal

2 janvier 2021

Table des matières

1 Laboratoires

Chaque laboratoire sera précédé par un court rappel de la matière qui peut servir de référence et qui sera exemplifié par des *queries* SQL.

1.1 Laboratoire Rappel

1. Listez les noms des employés qui gagnent plus de 85 000 unités

```
1 select empnom, empsal from employe
2 where empsal > 85000;
```

empnom	empsal
DURANT	104000
SMITH	118900
GOULD	100800
ALLEN	110200
DANOIS	140200
VAN HE	100200
HIEL	110300
KOALA	95300
Mc CARTNEY	85300
ALBERT	92340
STARR	104235
LISA	93456
MONA	93456
VAN BLAER	92654
DE COO	92654
DATTA	100654
PEREZ	110654
BOURVIL	98654
DELON	139054
MAES	99054
GIELS	86054

2. Listez les noms des employés dont le nom contient “ON” avec leur numéro de département

```sql
select empnom, empdpt from employe
where empnom like '%ON%';
```

empnom	empdpt
LENNON	C01
HARRISSON	D11
MONA	D11
DELON	E11

3. Comptez le nombre de femmes employées dans la société

```sql
select count(*) "Nbr femmes" from employe
where empsexe = 'F';
```

Nbr femmes
8

4. Donnez le nombre d’employés par numéro de département

```sql
select empdpt, count(*) "Nbr employés" from employe
group by empdpt;
```

empdpt	Nbr employés
A00	2
D11	6
D21	5
E21	5
C01	6
E11	6

5. Donnez les départements dont la moyenne des salaires dépasse 85 000 unités.

```sql
select empdpt, round(avg(empsal)) "Salaire moyen" from employe
group by empdpt
having avg(empsal) > 85000;
```

empdpt	Salaire moyen
A00	90300
D11	88217
D21	92563
E21	89879
C01	96104
E11	87736

6. Donnez les noms d'employés correspondant à des homographes

???

7. Listez les n° de département dirigeant plus de 2 départements

```
select d.dptadm,
min(d2.dptlib) "Nom dpt",
count(d.dptadm) "Nbr dpt dirigés"
from departement d
join departement d2 on d.dptadm = d2.dptno
group by d.dptadm;
```

dptadm	Nom dpt	Nbr dpt dirigés
D21	DIRECTION	2
A00	DEVELOPPEMENT	2
E11	VENTES	3

8. Donnez le nombre de managers différents (la solution la plus simple est fournie par une expression dialectale d'oracle)

```
select count(distinct(dptmgr)) from departement;
```

count
8

9. Listez les noms des employés avec le nom de leur département

```
select e.empnom, d.dptlib from employe e
join departement d on e.empdpt = d.dptno;
```

empnom	dptlib
Mc CARTNEY	DEVELOPPEMENT
KOALA	DEVELOPPEMENT
MARKA	MAINTENANCE
LENNON	MAINTENANCE
ALBERT	MAINTENANCE
HIEL	MAINTENANCE
ALLEN	MAINTENANCE
SMITH	MAINTENANCE
DE COO	SUPPORT
VAN BLAER	SUPPORT
LUDI	SUPPORT
MONA	SUPPORT
LISA	SUPPORT
HARRISSON	SUPPORT
PEREZ	DIRECTION
DATTA	DIRECTION
MANTE	DIRECTION
DANTE	DIRECTION
DANOIS	DIRECTION
TOOR	VENTES
DELON	VENTES
BOURVIL	VENTES
MOZART	VENTES
VAN HE	VENTES
GOULD	VENTES
GIELS	FORMATION
VAN RAES	FORMATION
MAES	FORMATION
STARR	FORMATION
DURANT	FORMATION

10. Listez les noms des employés qui sont managers

```
select empnom, dptlib "Dpt managé" from employe
join departement on empno = dptmgr;
```

empnom	Dpt managé
MAES	DEVELOPPEMENT
DURANT	PRODUCTION
SMITH	MAINTENANCE
ALLEN	SUPPORT
DANOIS	DIRECTION
GOULD	MARKETING
GIELS	VENTES
HIEL	FORMATION

11. Même exercice que 4, mais donnez le libellé du département

```
select dptlib, count(*) "Nbr employés" from employe
join departement on empdpt = dptno
group by dptlib;
```

dptlib	Nbr employés
DEVELOPPEMENT	2
DIRECTION	5
FORMATION	5
MAINTENANCE	6
SUPPORT	6
VENTES	6

12. Donnez la liste des libellés de départements avec leur masse salariale

```sql
select dptlib, sum(empsal) "Masse salariale" from employe
join departement on empdpt = dptno
group by dptno;
```

dptlib	Masse salariale
DIRECTION	462816
VENTES	526416
SUPPORT	529304
MAINTENANCE	576624
FORMATION	449397
DEVELOPPEMENT	180600

13. Donnez la liste des managers avec le nombre de personnes qu'ils dirigent

```sql
select d.dptmgr,
min(d.dptno) "Departement",
count(e.empno) "Nbr employés"
from departement d
left outer join employe e on d.dptno = e.empdpt
group by d.dptmgr;
```

dptmgr	Departement	Nbr employés
340	E11	6
020	B01	0
030	C01	6
070	D21	5
060	D11	6
050	E01	0
100	E21	5
320	A00	2

14. Donnez la liste des noms de départements employant plus de 2 femmes

```sql
select empdpt,
min(dptlib) "Departement",
count(*) "Nbr de femmes"
from employe
left join departement on empdpt = dptno
where empsexe = 'F'
group by empdpt
having count(*) >= 2;
```

empdpt	Departement	Nbr de femmes
D11	SUPPORT	2
D21	DIRECTION	2
E21	FORMATION	2

Ou bien alternativement

```sql
select empdpt, min(dptlib) "dptlib", count(*) from employe
left join departement on empdpt = dptno
group by empdpt, empsexe
having empsexe = 'F' and count(*) >= 2;
```

empdpt	dptlib	count
D11	SUPPORT	2
D21	DIRECTION	2
E21	FORMATION	2

15. Donnez les noms des managers des départements ayant plus de 5 employés

```sql
select empno, empnom "Manager" from employe
where empno in (select d.dptmgr from employe e
                              left outer join departement d on e.empdpt = d.dptno
                        group by d.dptmgr having count(*) > 5);
```

empno	Manager
030	SMITH
060	ALLEN
340	GIELS

16. Donnez les noms des managers dirigeants plus de 5 employés.

```sql
select e.empnom from employe e
join departement d on e.empno = d.dptmgr
join employe e2 on e2.empdpt = d.dptno
where e2.empno != e.empno
group by e.empnom
having count(*) > 5;
```

empnom
ALLEN
GIELS

17. Donnez par département le libellé de département et le nombre de départements dirigés

```sql
select d.dptlib, count(*) from departement d
join departement d2 on d.dptno = d2.dptadm
group by d.dptlib;
```

dptlib	count
DIRECTION	2
DEVELOPPEMENT	2
VENTES	3

18. Donnez les libellés des départements administrés par un autre département

```sql
select dptlib from departement
where dptno != dptadm;
```

dptlib
DEVELOPPEMENT
PRODUCTION
MAINTENANCE
SUPPORT
MARKETING
VENTES
FORMATION

19. Listez les managers dont le département auquel ils appartiennent est dirigé par un homographe.

1.2 Laboratoire `select` imbriqués

1. Donnez le nom de(s) l'employé(s) ayant le plus haut salaire.

```sql
select empnom, empsal from employe
where empsal >= all(select empsal from employe);
```

empnom	empsal
DANOIS	140200

2. Donnez la liste des employés gagnant plus que la moyenne des employés

```
select empnom from employe
where empsal > (select avg(empsal) from employe);
```

empnom
DURANT
SMITH
GOULD
ALLEN
DANOIS
VAN HE
HIEL
KOALA
ALBERT
STARR
LISA
MONA
VAN BLAER
DE COO
DATTA
PEREZ
BOURVIL
DELON
MAES

3. Donnez la liste des femmes gagnant plus que la moyenne des hommes

```
select empnom, empsexe, empsal from employe
where empsexe = 'F' and empsal > (select avg(empsal) from employe
                                                    where empsexe = 'M');
```

empnom	empsexe	empsal
ALLEN	F	110200
VAN HE	F	100200
LISA	F	93456
MONA	F	93456
PEREZ	F	110654
MAES	F	99054

4. Donnez le libellé du(es) département(s) employant le plus de personnel.

```
select d.dptlib, count(*) from departement d
join employe e on e.empdpt = d.dptno
group by d.dptno
having count(*)  >= all(select count(*) from departement
                                  join employe on dptno = empdpt
                        group by dptno);
```

dptlib	count
VENTES	6
SUPPORT	6
MAINTENANCE	6

5. Donnez le libellé du(es) département(s) ayant la masse salariale la plus élevée

```
select d.dptlib from departement d
join employe e on e.empdpt = d.dptno
group by d.dptno
having sum(e.empsal) >= all(select sum(empsal) from employe
                                    join departement on dptno = empdpt
                            group by dptno);
```

dptlib
MAINTENANCE

6. Donnez la liste des employés qui ne sont pas des managers

```
select empno, empnom from employe
where empno in (select empno from employe
                                except /*MINUS pour Oracle*/
                select dptmgr from departement);
```

empno	empnom
090	VAN HE
280	MOZART
250	DATTA
150	STARR
260	MARKA
170	LISA
270	PEREZ
200	VAN BLAER
160	HARRISSON
220	DE COO
140	LENNON
240	MANTE
300	DELON
230	DANTE
290	BOURVIL
120	Mc CARTNEY
180	MONA
190	LUDI
330	VAN RAES
110	KOALA
130	ALBERT
310	TOOR

7. Donnez le nombre de managers différents

```
select count(distinct(dptmgr)) "Nbr managers différents" from departement;
```

Nbr managers différents
8

8. Donnez la liste des employés gagnant plus de la moyenne des salaires de leur département

```
select e.empno, e.empnom, e.empsal, e.empdpt from employe e
join departement d on e.empdpt = d.dptno
group by e.empno
having e.empsal >= all(select avg(empsal) from employe
                                    join departement on empdpt = dptno
                               group by dptno);
```

empno	empnom	empsal	empdpt
090	VAN HE	100200	E11
020	DURANT	104000	E21
250	DATTA	100654	D21
150	STARR	104235	E21
060	ALLEN	110200	C01
270	PEREZ	110654	D21
050	GOULD	100800	E11
100	HIEL	110300	C01
030	SMITH	118900	C01
320	MAES	99054	E21
070	DANOIS	140200	D21
300	DELON	139054	E11
290	BOURVIL	98654	E11

Pour référence, voici les salaires moyens par département.

```sql
select dptno, dptlib, round(avg(empsal)) "Salaire moyen" from departement
join employe on dptno = empdpt
group by dptno;
```

dptno	dptlib	Salaire moyen
D21	DIRECTION	92563
E11	VENTES	87736
D11	SUPPORT	88217
C01	MAINTENANCE	96104
E21	FORMATION	89879
A00	DEVELOPPEMENT	90300

9. Liste des départements dont la moyenne des salaires est supérieure d'au moins 10% à la moyenne des salaires des employés des autres départements.

N.B : Au delà de 5% il n'y a pas de département sélectionné, j'ai pris 1%. Ici je fais un `cross join` pour toutes les clés primaires de la table département avec elle même, sauf dans le cas où le département est déjà sélectionné. Ensuite, pour chaque groupe et sous-groupe, je compare le salaire moyen du département avec celui des autres. Cette requête donne chaque couple pour laquelle une moyenne est supérieure à l'autre, avec `distinct`, on ne retrouve qu'une fois chaque département.

```sql
select distinct(d1.dptno), d1.dptlib from departement d1
cross join departement d2
join employe e1 on e1.empdpt = d1.dptno
join employe e2 on e2.empdpt = d2.dptno
where d1.dptno != d2.dptno
group by d1.dptno, d2.dptno
having avg(e1.empsal) >= 1.01*avg(e2.empsal);
```

dptno	dptlib
E21	FORMATION
D21	DIRECTION
C01	MAINTENANCE
A00	DEVELOPPEMENT

10. Donnez la liste des libellés de département dont la masse salariale est supérieure à celle de leur administrateur

```sql
select d.dptlib from departement d
join (
     select empdpt, sum(empsal) "Masse salariale" from employe group by empdpt
     ) gb on gb.empdpt = d.dptno
left join (
     select empdpt, sum(empsal) "Masse salariale adm" from employe group by empdpt
)     gba on gba.empdpt = d.dptadm

where "Masse salariale" > "Masse salariale adm";
```

dptlib
MAINTENANCE
SUPPORT
VENTES

1.3 Laboratoire `outer join`

1. Donnez la liste de tous les libellés de département avec en regard – si il existe – le libellé du département qui l'administre

```sql
select dptlib, dptadm from departement;
```

dptlib	dptadm
DEVELOPPEMENT	D21
PRODUCTION	A00
MAINTENANCE	A00
SUPPORT	E11
DIRECTION	
MARKETING	E11
VENTES	D21
FORMATION	E11

2. Donnez la liste des départements qui n'en administrent pas d'autres

```
select dptno, dptlib from departement
where dptno in (select dptno from departement
                    except /*MINUS pour Oracle DB*/
               select dptadm from departement);
```

dptno	dptlib
C01	MAINTENANCE
E01	MARKETING
E21	FORMATION
D11	SUPPORT
B01	PRODUCTION

3. Donnez la liste des noms de managers dirigeant un département autre que celui auquel ils sont affectés.

```
select empnom, empdpt,dptno, dptmgr from employe
join departement on empno = dptmgr
where empdpt != dptno;
```

empnom	empdpt	dptno	dptmgr
MAES	E21	A00	320
DURANT	E21	B01	020
ALLEN	C01	D11	060
GOULD	E11	E01	050
GIELS	E21	E11	340
HIEL	C01	E21	100

4. Donnez le salaire moyen d'un manager.

```
select round(avg(empsal)) "Salaire moyen manager" from employe
join departement on empno = dptmgr;
```

Salaire moyen manager
108689

5. Donnez le salaire moyen d'un employé qui n'est pas un manager.

```
select round(avg(empsal)) "Salaire moyen non-manager" from employe
join departement on empno != dptmgr;
```

Salaire moyen non-manager
90223

6. Donnez la liste des employés dirigeant plus d'un département.

```
select distinct(e.empno), e.empnom, d.dptlib from employe e
join departement d on e.empno = d.dptmgr
join departement d2 on d.dptno = d2.dptadm;
```

empno	empnom	dptlib
070	DANOIS	DIRECTION
320	MAES	DEVELOPPEMENT
340	GIELS	VENTES

7. Listez les managers [n°, nom] qui n'appartiennent pas à un département qu'ils dirigent.

```sql
select e.empno, e.empnom, e.empdpt, d.dptno from employe e
join departement d on e.empno = d.dptmgr
where e.empdpt != d.dptno;
```

empno	empnom	empdpt	dptno
320	MAES	E21	A00
020	DURANT	E21	B01
060	ALLEN	C01	D11
050	GOULD	E11	E01
340	GIELS	E21	E11
100	HIEL	C01	E21

8. Donnez la liste des libellés des départements auxquels appartient au moins un employé dont le nom commence par D ou M.

```sql
select d.dptlib from departement d
join employe e on e.empdpt = d.dptno
where e.empnom like 'D%' or e.empnom like 'M%'
group by d.dptno;
```

dptlib
DEVELOPPEMENT
MAINTENANCE
SUPPORT
DIRECTION
VENTES
FORMATION

9. Donnez la liste des libellés des départements auxquels appartient au moins un employé dont le nom commence par D et au moins un employé dont le nom commence par M.

```sql
select d.dptlib
from departement d
left join employe e on d.dptno = e.empdpt
group by d.dptlib
having sum(case when e.empnom like 'M%' then 1 else 0 end) >= 1
       and
       sum(case when e.empnom like 'D%' then 1 else 0 end) >= 1;
```

dptlib
SUPPORT
DIRECTION
FORMATION
VENTES

Sinon, il existe l'alternative avec `intersect`.

```sql
select d.dptlib from departement d
join employe e on d.dptno = e.empdpt
where e.empnom like 'M%'
intersect
select d.dptlib from departement d
join employe e on d.dptno = e.empdpt
where e.empnom like 'D%';
```

dptlib
DIRECTION
FORMATION
VENTES
SUPPORT

10. Donnez la liste des libellés des départements auxquels appartient au moins un employé dont le nom commence par D et pas d'employé dont le nom commence par M

```sql
select d.dptlib from departement d
left join employe e on d.dptno = e.empdpt
group by d.dptlib
having sum(case when e.empnom like 'D%' then 1 else 0 end) >= 1
       and
       sum(case when e.empnom like 'M%' then 1 else 0 end) = 0;

--- Aucun département ne correspond à cette requête
```

dptlib

Une solution alternative avec `except`.

```sql
select d.dptlib from departement d
join employe e on d.dptno = e.empdpt
where e.empnom like 'D%'
except /* Utiliser =except= pour enlever un match*/
select distinct d.dptlib from departement d
join employe e on d.dptno = e.empdpt
where e.empnom like 'M%';
```

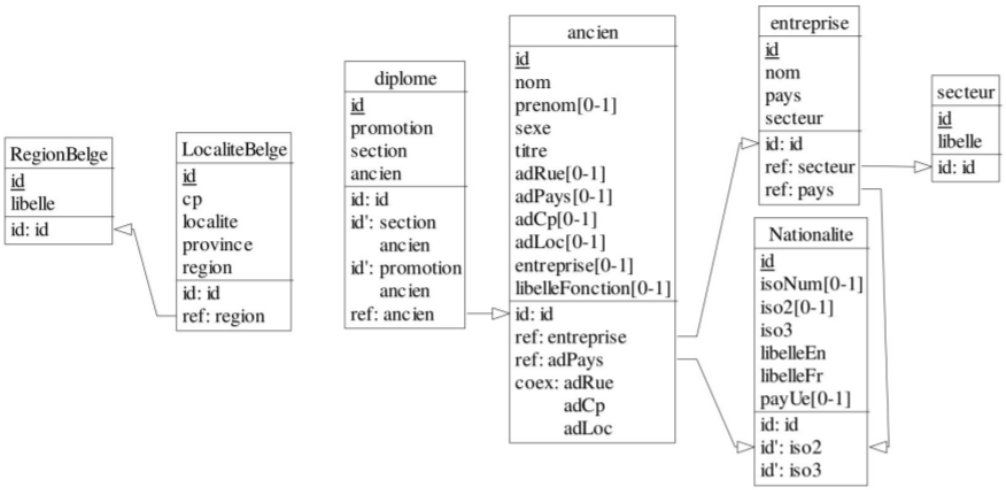
dptlib

11. Donnez par département le nombre de femmes (n'oubliez pas les 0).

```sql
select d.dptno, count(e.empsexe) "Nb femmes" from departement d
left join employe e on d.dptno = e.empdpt
where e.empsexe is null or e.empsexe = 'F'
group by d.dptno, e.empsexe;
```

dptno	Nb femmes
E01	0
C01	1
D21	2
E21	2
E11	1
B01	0
D11	2

1.4 Laboratoire schéma "Ancien"



Créer des requêtes permettant de donner :

— Nom et prénom des anciennes.

```sql
select nom, prenom from ancien
where sexe = 2;
```

— Nombre d'anciens résidant à Uccle.

```sql
select count(*) from ancien
where adLoc = 'Uccle';
```

— Nombre d'anciens par sexe.

```sql
select sexe, count(*) from ancien
group by sexe;
```

— Nom et prénom des anciens résidant en région flamande.

```sql
select nom, prenom from ancien
where adLoc in (select localite from LocaliteBelge
                                    where region='Région Flamande');
```

— Nom et prénom des anciens résidant hors de l'UE, ordonnés sur le nom de l'ancien.

```sql
select a.nom, a.prenom
from ancien a
join nationalite n on a.adPays = n.iso2
where n.paysUe = 0
order by a.nom;
```

— Nombre d'anciens diplômés par année.

```sql
select d.promotion, count(*) from diplome d
join ancien a on d.ancien = a.id
group by d.promotion;
```

— Nombre d'anciens par Entreprise, ordonnés décroissant sur le nombre d'anciens.

```sql
select e.id, count(*) "Nbr anciens" from entreprise e
join ancien a on e.id = a.entreprise
group by e.id
order by "Nbr anciens" desc;
```

— Liste des années de promotion pour lesquelles au moins 3 filles ont été diplômées.

```sql
select d.promotion, count(*) from diplome d
join ancien a on d.ancien = a.id
where a.sexe = 2
group by d.promotion
having count(*) >= 3;
```

— Liste des noms d'entreprise avec le secteur d'activité.

```sql
select e.nom, s.libelle from entreprise e
join secteur s on e.secteur = s.id;
```

— Liste des anciens ayant obtenu deux diplômes à l'Institut.

```sql
select ancien from diplome
group by ancien
having count(*) = 2;
```

— Liste des promus après 1987 travaillant pour une firme étrangère.

```sql
select a.id, a.nom from ancien a
join diplome d on a.id = d.ancien
join entreprise e on a.entreprise = e.id
where d.promotion > 1987 and e.pays != 'BE';
```

— Liste des anciens promus dans la même section et la même année que DEE, Jacques.

```sql
create or replace view ancienDip
as
select * from ancien
join diplome on ancien.id = diplome.ancien;


select ad.id, ad.nom from ancienDip
where ad.section = (select "section" from ancienDip
                                    where nom = 'DEE' and prenom = 'Jacques';)
and ad.promotion = (select promotion from ancienDip
                                    where nom = 'DEE' and prenom = 'Jacques';)
;
```

— Nombre d'anciens employés par secteur d'activité.

```sql
select s.libelle, count(*) from ancien a
join entreprise e on a.entreprise = e.id
join secteur s on e.secteur = s.id
group by s.libelle;
```

— Nombre d'anciens diplômés par année et section.

```sql
select d.promotion, d.section, count(*) from diplome d
join ancien a on a.id = d.ancien
group by (d.promotion, d.section);
```

— En vue de la réalisation d'un mailing, nom, prénom, titre, adresse, localité, code postal, pays de résidence des anciens diplômés entre 1985 et 1995.

```sql
select * from ancien
join diplome on diplome.ancien = ancien.id
where diplome.promotion between 1985 and 1995;
```

— Année(s) de promotion ayant vu le plus de filles diplômées en informatique de gestion.

```sql
select d.promotion from diplome d
join ancien a on d.ancien = a.id
group by d.promotion
having count(*) >= all (select count(*) from ancien a
                                    join diplome d on d.ancien = a.id
                                where a.sexe = 2 and d.section = 'G'
                                 group by d.promotion;)
;
```

1.5 Laboratoire Sémantique

La requête soustrait le nombre de départements administrateurs au nombre de départements. En clair, c'est le nombre de départements qui n'ont pas d'administrateurs.

```sql
SELECT (COUNT(*) - COUNT(dptadm)) as "Column"
FROM Departement
```

Column
1

La requête donne le numéro et le nom des employés dont le nom du manager est «MAES».

```sql
SELECT e.empno, e.empnom
FROM Employe e
JOIN Departement ON dptno = e.empdpt
JOIN Employe m ON dptmgr = m.empno
WHERE UPPER(m.empnom) = 'MAES';
```

empno	empnom
110	KOALA
120	Mc CARTNEY

La requête donne les départements pour lesquels il y a le plus de femmes.

```sql
SELECT dptlib, dptno
FROM Departement
JOIN Employe ON dptno = empdpt
WHERE empsexe = 'F'
GROUP BY dptlib, dptno
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM Employe
                        WHERE empsexe = 'F'
                        GROUP BY empdpt);
```

dptlib	dptno
SUPPORT	D11
DIRECTION	D21
FORMATION	E21

Cette requête sélectionne les employés qui ne sont pas des managers.

```sql
SELECT empno, empnom FROM Employe
LEFT JOIN Departement ON dptmgr = empno
where dptmgr is null;
```

empno	empnom
090	VAN HE
110	KOALA
120	Mc CARTNEY
130	ALBERT
140	LENNON
150	STARR
160	HARRISSON
170	LISA
180	MONA
190	LUDI
200	VAN BLAER
220	DE COO
230	DANTE
240	MANTE
250	DATTA
260	MARKA
270	PEREZ
280	MOZART
290	BOURVIL
300	DELON
310	TOOR
330	VAN RAES

Sélectionne les employés dont le salaire est inférieur à celui d'au moins une femme du même département.

```sql
SELECT empno, empnom
FROM Employe e
WHERE empsal < ANY(SELECT empsal FROM Employe ee
                   WHERE empsexe = 'F' AND ee.empdpt = e.empdpt);
```

empno	empnom
130	ALBERT
140	LENNON
160	HARRISSON
190	LUDI
200	VAN BLAER
220	DE COO
230	DANTE
240	MANTE
250	DATTA
260	MARKA
280	MOZART
290	BOURVIL
310	TOOR
330	VAN RAES
340	GIELS

La requête sélectionne le département pour lequel le manager gagne le moins, c'est-à-dire le manager avec le plus faible salaire parmi tous les managers.

```
SELECT dptlib, dptno
FROM Departement JOIN Employe ON dptmgr = empno
WHERE empsal <= ALL (SELECT empsal FROM Employe
                                        JOIN Departement ON dptmgr = empno);
```

dptlib	dptno
VENTES	E11

La requête sélectionne les départements où plus de trois personnes du même sexe gagnent au moins 110 000.

```
SELECT DISTINCT empdpt FROM Employe
WHERE empsal > 110000
GROUP BY empdpt, empsexe
HAVING COUNT(*) > 3;
```

empdpt

Cette requête sélectionne les départements pour lesquels il y a des managers de sexes différents.

```
SELECT distinct d.dptno, d.dptlib
FROM Departement d
JOIN Employe e1 ON d.dptno = e1.empdpt
JOIN Employe e2 ON d.dptno = e2.empdpt
JOIN Departement d1 ON d1.dptmgr = e1.empno
JOIN Departement d2 ON d2.dptmgr = e2.empno
WHERE e1.empsexe != e2.empsexe;
```

dptno	dptlib
C01	MAINTENANCE
E21	FORMATION

1.6 Laboratoire DDL-DML-DCL

1.6.1 Création de table

```
drop table if exists test;

create table test (
tid int not null,
tlib varchar(50) not null,
tnb1 decimal(5,2) default 12 not null,
tnb2 decimal(8,2) not null);
```

1. Créez quelques tuples en vérifiant que les différents attributs sont effectivement obligatoires.

```sql
insert into test(tid, tlib, tnb1, tnb2) values (0, 'John', 11.1, 23);
insert into test(tid, tlib, tnb1, tnb2) values (1, 'Mary', 14.4, 76);
```

Mais par exemple, l'insertion suivante provoque une erreur :

```sql
insert into test(tid, tlib, tnb1, tnb2) values (3, 'a');
```

2. Veillez à insérer 2 tuples ayant la même valeur pour tId.

```sql
insert into test(tid, tlib, tnb1, tnb2) values (1, 'Derek', 2.3, 4.5);
insert into test(tid, tlib, tnb1, tnb2) values (1, 'Amon', 3, 4);
```

3. Réalisez une insertion d'un tuple pour lequel vous ne spécifiez pas la valeur de tNb1. Vérifiez que l'attribut reçoit bien la valeur 12.

```sql
insert into test(tid, tlib, tnb2) values (2, 'Lucy', 27);
```

1.6.2 Ajout de contrainte

— Ajoutez chacune des contraintes suivantes :
  — tId est clé primaire ;
  — tNb1 doit être supérieur à 0 ;
  — tNb2 doit être supérieur à tNb1 ;
  — Deux tuples de Test ne peuvent pas avoir les mêmes valeurs pour la paire tNb1 et tNb2.

Tout d'abord il faut modifier la table `test`, car elle ne respecte pas les contraintes :

```sql
update test
set tId = 2 where tlib = 'Derek';

update test
set tId = 3 where tlib = 'Amon';

update test
set tId = 4 where tlib = 'Lucy';
```

```sql
alter table test
add constraint testPK primary key(tId),
add constraint tnb1Pos check (tnb1 > 0),
add constraint tnb2Suptnb1 check (tnb2 > tnb1),
add constraint uniqueTnb1Tnb2 unique(tnb1, tnb2);
```

Pour chaque contrainte, introduisez des tuples la satisfaisant et d'autres ne la satisfaisant pas. Remarquez que l'ajout d'une contrainte n'est accepté que si les données déjà présentes la respectent.

Pour le moment, la table `test` ressemble à cela :

```sql
select * from test;
```

tid	tlib	tnb1	tnb2
0	John	11.10	23.00
1	Mary	14.40	76.00
2	Derek	2.30	4.50
3	Amon	3.00	4.00
4	Lucy	12.00	27.00
7	Freddy	0.50	2.00

— Modifiez un ensemble de tuples de Test en veillant à ce que cette modification entraîne, pour un tuple, la violation d'une contrainte. Vérifiez qu'aucun tuple n'aura été modifié.

Dans l'exemple suivant, la table ne sera pas mise à jour puisque la modification enfreint la contrainte de positivité.

```
update test
set tnb1 = -1
where tid = 1;
```

1. Clé étrangère

   (a) Exécutez la requête :

```
drop table if exists test2;

create table Test2 (
ttId int,
t2Ref int);
```

   (a) Ajoutez la contrainte définissant l'attribut t2Ref comme clé étrangère vers la table Test.

```
alter table test2
add constraint FK_t2ref foreign key(t2Ref)
references test(tid);
```

   (a) Ajoutez des tuples à Test2 pour vérifier la mise en œuvre de cette dernière contrainte.

```
insert into test2(ttid, t2ref) values (11, 0);
insert into test2(ttid, t2ref) values (12, 1);
insert into test2(ttid, t2ref) values (13, 3);
```

   Les entrées précédentes fonctionnent parce qu'elles font bien référence à la clé primaire de `test` mais l'insertion suivante ne fonctionnera pas.

```
insert into test2(ttid, t2ref) values (100, 10);
```

   (a) Supprimez la contrainte de clé étrangère. Recréez-la avec l'option `ON DELETE CASCADE`. Testez.

```
alter table test2
drop constraint if exists FK_t2Ref;

/*Avec on delete cascade
Si des éléments de test(tid) sont supprimés
ils seront supprimés dans test2 aussi
alors qu'auparavant on aurait eu une erreur.*/

alter table test2
add constraint FK_t2Ref foreign key(t2Ref)
references test(tid)
on delete cascade;
```

2. Contrainte `deferrable`

   (a) Supprimez la contrainte foreign key définie à l'étape précédente.

```
alter table test2
drop constraint FK_t2Ref;
```

   (a) Recréez-la avec l'option `DEFERRABLE INITIALLY DEFERRED`.

```
alter table test2
add constraint FK_t2Ref foreign key(t2Ref)
references test(tid)
deferrable initially deferred;
```

(a) Donnez une suite d'instructions mettant en évidence la différence de comportement du SGBD avec ou sans l'option du point précédent. Terminez cette suite d'instructions par un `COMMIT`. Exécutez-la au moyen d'un script SQL.

```
\set AUTOCOMMIT off
```

L'expression suivante fonctionne malgré le fait que le commit introduit une référence à une clé de `test` qui n'existe pas encore. La contrainte n'est vérifiée qu'au moment du commit et pas au moment de l'insertion.

```
begin;
insert into test2(ttid, t2ref) values (111, 7);
insert into test(tid, tlib, tnb1, tnb2) values (7, 'Freddy', 0.5, 2);
commit;
```

Je remet l'auto-commit par facilité.

```
\set AUTOCOMMIT on
```

(a) Une fois de plus, supprimez la contrainte de clé étrangère. Recréez-la maintenant avec l'option `DEFERRABLE INITIALLY IMMEDIATE`. Remarquez que la mention `IMMEDIATE` n'est pas nécessaire car il s'agit de l'option par défaut.

```
alter table test2
drop constraint FK_t2Ref;

alter table test2
add constraint FK_t2Ref foreign key(t2Ref)
references test(tid)
deferrable initially immediate;
```

(a) Lors des tests, vous voyez que nous semblons nous retrouver dans la situation sans `DEFERRABLE`. Pour pouvoir profiter ou non du fait de différer le contrôle de la contrainte, une instruction particulière doit être émise en début de transaction. Accédez à la page 19-48 (p. 1306) du References Guide SQL Oracle qui vous est fourni sur poESI dans Ressources. Testez.

```
--- Syntaxe Oracle DB
alter session set constraints = deferred;
```

1.6.3 Création du schéma conceptuel

Cet exercice vous donne l'occasion de disposer d'un script pour la création des tables de la base de donnée Département / Employé. Ce script est utilisé ensuite à plusieurs reprises. Le schéma conceptuel des tables Département / Employé est fourni sur poESI.

1. Créez les tables Employe et Departement dans votre schéma sans définir l'attribut EmpSal. Importez les données à partir du schéma ADT. N'oubliez pas de commenter chacune des tables et chacun des attributs. Veillez à garder toutes les commandes DDL dans un fichier texte qui constitue un script SQL.

```
create table salary
as select empno, empsal from employe; /*copie à insérer plus tard*/

drop table employe;
drop table departement;
```

```
create table employe (
empno char(3) not null,
empnom varchar not null,
empsexe char(1) not null,
empdpt char(3) not null,
primary key(empno)
```

```
);

create table departement (
dptno char(3) not null,
dptlib varchar not null,
dptmgr char(3) not null,
dptadm char(3),
primary key(dptno)
);

alter table departement
add constraint fk_dptmgr foreign key(dptmgr) references employe(empno) deferred
add constraint fk_dptadm foreign key(dptadm) references departement(dptno) deferred;
```

1. Supprimez les tables créées et faites exécuter votre script SQL pour les recréer.

```
insert into employe (empno, empnom, empsexe, empdpt) values ('020','DURANT','M','E21');
insert into employe (empno, empnom, empsexe, empdpt) values ('030','SMITH','M', 'C01');
insert into employe (empno, empnom, empsexe, empdpt) values ('050','GOULD','M', 'E11');
insert into employe (empno, empnom, empsexe, empdpt) values ('060','ALLEN','F', 'C01');
insert into employe (empno, empnom, empsexe, empdpt) values ('070','DANOIS','M','D21');
insert into employe (empno, empnom, empsexe, empdpt) values ('090','VAN HE','F','E11');
insert into employe (empno, empnom, empsexe, empdpt) values ('100','HIEL','M','C01');
insert into employe (empno, empnom, empsexe, empdpt) values ('110','KOALA','M','A00');
insert into employe (empno, empnom, empsexe, empdpt) values ('120','Mc CARTNEY','M','A00');
insert into employe (empno, empnom, empsexe, empdpt) values ('130','ALBERT','M','C01');
insert into employe (empno, empnom, empsexe, empdpt) values ('140','LENNON','M','C01');
insert into employe (empno, empnom, empsexe, empdpt) values ('150','STARR','M','E21');
insert into employe (empno, empnom, empsexe, empdpt) values ('160','HARRISSON','M','D11');
insert into employe (empno, empnom, empsexe, empdpt) values ('170','LISA','F','D11');
insert into employe (empno, empnom, empsexe, empdpt) values ('180','MONA','F','D11');
insert into employe (empno, empnom, empsexe, empdpt) values ('190','LUDI','M','D11');
insert into employe (empno, empnom, empsexe, empdpt) values ('200','VAN BLAER','M','D11');
insert into employe (empno, empnom, empsexe, empdpt) values ('220','DE COO','M','D11');
insert into employe (empno, empnom, empsexe, empdpt) values ('230','DANTE','M','D21');
insert into employe (empno, empnom, empsexe, empdpt) values ('240','MANTE','F','D21');
insert into employe (empno, empnom, empsexe, empdpt) values ('250','DATTA','M','D21');
insert into employe (empno, empnom, empsexe, empdpt) values ('260','MARKA','M','C01');
insert into employe (empno, empnom, empsexe, empdpt) values ('270','PEREZ','F','D21');
insert into employe (empno, empnom, empsexe, empdpt) values ('280','MOZART','M','E11');
insert into employe (empno, empnom, empsexe, empdpt) values ('290','BOURVIL','M','E11');
insert into employe (empno, empnom, empsexe, empdpt) values ('300','DELON','M','E11');
insert into employe (empno, empnom, empsexe, empdpt) values ('310','TOOR','M','E11');
insert into employe (empno, empnom, empsexe, empdpt) values ('320','MAES','F','E21');
insert into employe (empno, empnom, empsexe, empdpt) values ('330','VAN RAES','F','E21');
insert into employe (empno, empnom, empsexe, empdpt) values ('340','GIELS','M','E21');

insert into departement (dptno, dptlib, dptmgr, dptadm) values ('A00','DEVELOPPEMENT','320','D21');
insert into departement (dptno, dptlib, dptmgr, dptadm) values ('B01','PRODUCTION','020','A00');
insert into departement (dptno, dptlib, dptmgr, dptadm) values ('C01','MAINTENANCE','030','A00');
insert into departement (dptno, dptlib, dptmgr, dptadm) values ('D11','SUPPORT','060','E11');
insert into departement (dptno, dptlib, dptmgr, dptadm) values ('D21','DIRECTION','070',null);
insert into departement (dptno, dptlib, dptmgr, dptadm) values ('E01','MARKETING','050','E11');
insert into departement (dptno, dptlib, dptmgr, dptadm) values ('E11','VENTES','340','D21');
insert into departement (dptno, dptlib, dptmgr, dptadm) values ('E21','FORMATION','100','E11');
```

1. Ajoutez l'attribut EmpSal et affectez lui les valeurs en provenance de ADT.Employe.EmpSal. (Ici j'utilise la copie des salaires, `salary`.)

```
alter table employe
add empsal integer;
```

```
update employe
set empsal = (select s.empsal from salary s
                  where s.empno = employe.empno);
```

1. Vérifiez au travers de quelques tests que les différentes CI (contraintes d'intégrité) explicitées sont correctement implémentées.

```
--- Produit des erreurs

--- Non unique
insert into departement(dptno, dptlib, dptmgr, dptadm) values
('D21', 'DIRECTION', '320', null);

--- empno n'a pas trois charactères
insert into employe (empno, empnom, empsexe, empsal, empdpt) values
('3212', 'BLA', 'M', 10000, 'A00');

--- Manager pas dans la table employé
insert into departement(dptno, dptlib, dptmgr, dptadm) values
('A31', 'Nouveau', '355', 'D21');
```

1.6.4 Schémas externes

1. Créez la vue `Manager(mgrNo, mgrNom, dptDirigéLib, nbEmpDirigés)`. Elle permet d'accéder aux informations des managers de la société.

```
create or replace view Manager(mgrNo, mgrNom, dptDirigéLib, nbEmpDirigés)
as
select e.empno, e.empnom, d.dptlib, gb."nbrEmpDirigés" from employe e
join departement d on e.empno = d.dptmgr
join (select dptno, dptmgr, count(empno) "nbrEmpDirigés" from departement
       left join employe on dptno = empdpt
       group by dptno) gb
 on d.dptmgr = gb.dptmgr;
```

```
select * from Manager;
```

mgrno	mgrnom	dptdirigélib	nbempdirigés
320	MAES	DEVELOPPEMENT	2
020	DURANT	PRODUCTION	0
030	SMITH	MAINTENANCE	6
060	ALLEN	SUPPORT	6
070	DANOIS	DIRECTION	5
050	GOULD	MARKETING	0
340	GIELS	VENTES	6
100	HIEL	FORMATION	5

1. Créez la vue `EmployeDirection(empno, empnom, empsal, empdpt)`. Elle reprend les employés du département de libellé « DIRECTION ».

```
create or replace view EmployeDirection(empno, empnom, empsal, empdpt)
as
select empno, empnom, empsal, empdpt from employe
join departement on empdpt = dptno
where dptlib = 'DIRECTION';
```

```
select * from EmployeDirection;
```

empno	empnom	empsal	empdpt
070	DANOIS	140200	D21
230	DANTE	50654	D21
240	MANTE	60654	D21
250	DATTA	100654	D21
270	PEREZ	110654	D21

1. Ces deux vues sont-elles modifiables ? En cas de réponse positive, vérifiez l'effet de la clause `WITH CHECK OPTION`.

Les vues ne sont pas modifiables en PostgreSQL quand elles sélectionnent des données de plusieurs tables, autrement je pouvais rajouter `with check option` lors de la création de la vue.

```
insert into EmployeDirection(empno, empnom, empsal, empdpt)
values ('080', 'Jacky', 100000, 'D21');
```

1.6.5 Gestion de privilèges

Pour cette partie, nous vous demandons de travailler d'abord avec un·e de vos camarades de classe, puis avec plusieurs d'entre eux.

1. Donnez le privilège à votre camarade de classe de consulter la vue Manager. `GRANT SELECT ON` MaVue `TO SchemaCamarade1` ;

```
grant select on Manager to camarade;
```

1. Donnez le privilège à votre camarade de consulter la vue `EmployeDirection` en lui permettant de propager ce droit. Que se passe-t-il pour les utilisateurs ayant reçu de votre camarade ce privilège lorsque vous révoquez le privilège de votre camarade ?

Une personne qui a obtenu son droit via `with grant option` le perd aussi quand il est révoqué.

```
grant select on EmployeDirection to camarade with grant option;
```

1. Donnez à votre camarade le privilège de « mise à jour » sur les employés masculins de votre table `Employe`.

```
create or replace view EmployeDirection(empno, empnom, empsal, empdpt)
as
select empno, empnom, empsal, empdpt from employe
join departement on empdpt = dptno
where dptlib = 'DIRECTION' and empsexe = 'M';
grant update on EmployeDirectionM to camarade;
```

1.6.6 Synonyme

Vous devez ici encore travailler avec un·e camarades de classe.

1. Créez un synonyme sur une des vues auxquelles votre camarade vous donne accès. Si votre camarade supprime sa vue ou vous retire le droit d'accès, le synonyme existe-t-il toujours ?

Le synonyme existe toujours mais ne peux pas être utilisé.

```
create or replace synonym TblCamarade for camarade.EmployeDirection;
```

2. Supprimez le synonyme.

```
drop synonym TblCamarade;
```

1.6.7 Schéma interne

1. Créez un index sur l'attribut EmpNom de votre table Employe.

```sql
create index id_empnom on employe(empnom);
```

1.6.8 Consultation de catalogue

1. Consultez les vues utilisateur $\text{USER}_{\text{TABLES}}$, $\text{USER}_{\text{TABCOLUMNS}}$, $\text{USER}_{\text{COLCOMMENTS}}$, $\text{USER}_{\text{CONSTRAINTS}}$ et $\text{USER}_{\text{CONSCOLUMNS}}$. Une description sommaire de ces vues sur le catalogue est consultable sur poESI dans le document Catalogue Oracle. Vérifiez la présence des contraintes, commentaires, etc. que vous avez créés.

```sql
--- Syntaxe Oracle
select * from USER_TABLES;
select * from USER_TAB_COLUMNS;
select * from USER_COL_COMMENTS;
select * from USER_CONSTRAINTS;
select * from USER_CONS_COLUMNS;
```

1.6.9 Mise en œuvre des transactions

Ouvrez deux sessions distinctes de votre console SQL et :

1. Ajoutez des tuples à la table Test à partir d'une session en contrôlant le remplissage de la table à partir de l'autre session.
2. Testez l'annulation de transaction.
3. Testez la gestion de conflit lorsque deux sessions modifient les mêmes données « en même temps ». Expliquez ce que cette locution imprécise « en même temps » veut dire.

1.7 Laboratoire PL/SQL

1.7.1 Rédaction et tests de fonctions

— Écrire une fonction stockée FLibSexe(S) recevant en paramètre le code du sexe d'une personne et renvoyant 'Masculin' pour 'M' et 'Féminin' pour 'F' et génère une erreur dans les autres cas.

```sql
create or replace function flibsexe(sexe char)
return varchar as
begin
if sexe = 'M' then
return 'Masculin';
else if sexe = 'F' then
return 'Féminin';
else raise_application_error(-20100, 'Column has entries different from M or F');
end if;
end if;
end;
```

— Écrire une fonction stockée MasseSal(dpt) renvoyant la masse salariale du département dont le n° est fourni en paramètre.

```sql
create or replace function MasseSal(dpt gcuv.employe.empdpt%TYPE)
return number as nb number;
begin
select sum(empsal) into nb from gcuv.employe
where empdpt = dpt;
return nb;
end;
```

— Écrire une fonction stockée Fcondense(Chaine) recevant une chaîne de caractères et restituant la chaîne en majuscule en ayant extrait les caractères spéciaux, les blancs et en transformant les caractères accentués en caractères de base. Ajoutez à la table EMPLOYE la colonne EMPNOMCD et affectez lui les noms condensés. [Consultez le SQL User's guide d'Oracle pour l'utilisation de la fonction Translate].

```sql
create or replace function fcondense(chaine varchar)
return varchar as str varchar(1000);
begin
str := translate(chaine, 'ùèéàîïêûâ$*&@#?!'' ', 'ueeaiieua');
return upper(str);
end;
```

— Écrire une fonction stockée FnumNiv(Dpt) recevant un n° de département et renvoyant le niveau du département dans la hiérarchie de ceux-ci (ex : 0 pour direction, . . .).

```sql
create or replace function FnumNiv(dpt gcuv.departement.dptno%TYPE)
return integer as lvl integer;
tmp_dpt varchar(3);
begin
lvl := 0;
select dptadm into tmp_dpt from gcuv.departement
where dptno = dpt;
while tmp_dpt is not null loop
lvl := lvl + 1;
select dptadm into tmp_dpt from gcuv.departement
where dptno = tmp_dpt;
end loop;
return lvl;
end;
```

1.7.2 Création de vues dont la définition utilise des fonctions

— Créer une vue VDptDetail(DptNo, DptLib, DptNomsEmp) reprenant par Département son numéro, son libellé et la liste des noms des employés (triée par ordre alphabétique) du département [nous supposons que cette chaîne ne dépasse pas 255 caractères].

```sql
create or replace view VDptDetail(Dptno, DptLib, DptNomsEmp)
as
select dptno, dptlib, empnom
from departement
join employe on dptno = empdpt
order by Fcondense(empnom);
```

— Créer une vue VdptResp(DptNo, DptLib, DptNbDir) reprenant par département le n°, le libellé et le nombre de départements dépendant directement ou indirectement de celui-ci.

La fonction est adaptée de le fonction récursive du slide 50 de la partie PL/SQL.

```sql
create or replace function cntDptDependent(dpt gcuv.departement.dptno%TYPE)
return integer as nb integer;
curr_dep gcuv.departement.dptno%TYPE;
cursor childDep is
select dptno from gcuv.departement where dptadm = dpt;
begin
nb := 0;
select count(*) into nb from gcuv.departement
where dptadm = dpt;
open childDep;
fetch childDep into curr_dep;
while childDep%FOUND loop
nb := nb + cntDptDependent(curr_dep);
fetch childDep into curr_dep;
end loop;
close childDep;
return nb;
end;
```

```sql
create or replace view VdptResp(Dptno, Dptlib, "Nbr départments dirigés")
as
select dptno, dptlib, cntDptDependent(dptno)
from gcuv.departement;
```

1.7.3 Rédaction de procédures stockées

— Écrire une procédure stockée PtsfGroupe(Dpt1,Dpt2) permettant de transférer tous les employés du département DPT1 dans le département DPT2.

```sql
create or replace procedure PtsfGroupe(Dpt1 departement.dptno%TYPE,
Dpt2 departement.dptno%TYPE)
as
begin
update employe e
set e.empdpt = dpt2
where e.empdpt = dpt1;
end;
```

— Écrire une procédure stockée PmodAdm(Dpt,DptAdm) permettant de modifier en DptAdm le département administrateur de Dpt. Cette modification ne peut être acceptée que dans la mesure où elle n'introduit pas de cycle dans l'arbre de hiérarchie des départements.

```sql
create or replace procedure PmodAdm(Dpt dptcopy.dptno%TYPE, dadmin dptcopy.dptno%TYPE)
as
niveau integer;
nv_niveau integer;
begin
select distinct(FnumNiv(dpt)) into niveau from dptcopy
where dptno = dpt;
select distinct(FnumNiv(dadmin)) into nv_niveau from dptcopy
where dptno = dadmin;
if (nv_niveau > niveau) then
RAISE_APPLICATION_ERROR(-20001,'un cycle!');
else
update dptcopy
set dptadm = dadmin
where dptno = dpt;
end if;
end;
```

— Modifier la procédure précédente pour mémoriser dans une table créée à cet effet le moment, l'utilisateur et le terminal qui a réalisé une modification au travers de votre procédure. [Consultez le SQL User's guide d'Oracle pour l'utilisation de la fonction $\text{SYS}_{\text{CONTEXT}}$].

???

1.7.4 Mise en œuvre

— Donnez le droit d'utilisation de ces fonctions et procédures à votre voisin sans lui donner accès aux tables manipulées. Vérifiez la bonne exécution.

```sql
grant execute on procedure PmodAdm to camarade1;
grant execute on function FnumNiv to camarade2;
```