Systèmes de gestion de bases de données
Exercices sur plusieurs tables (série 2)

Cours de Jacques THOORENS

1 Exercices de laboratoire (solutions)

1. Donnez la liste des pays et des régions.

```
SELECT country_name
FROM countries
UNION
SELECT region_name
FROM regions
ORDER BY 1;
```

2. Donnez une liste alphabétique complète des nom géographiques employés dans la base de données.

```
SELECT country_name
FROM countries
UNION
SELECT region_name
FROM regions
UNION
SELECT city
FROM locations
UNION
SELECT state_province
FROM locations
ORDER BY 1;
```

3. Donnez la liste des employés de chaque service (*department*)

```
SELECT  department_name, First_Name, Last_Name
FROM Employees
INNER JOIN departments USING(department_id)
order by 1,2,3;
```

4. Donnez la liste des pays, classés par régions.

```
SELECT Region_name, country_name
FROM regions
Inner JOIN Countries USING(region_id)
ORDER BY 1,2;
```

5. Donnez la liste des employés avec le salaire minimum de leur fonction, leur salaire effectif et le salaire maximum auquel il peuvent prétendre.

```
SELECT e.last_name, e.first_name, j.min_salary, e.salary, j.max_salary
FROM employees e
INNER JOIN jobs j USING(job_id)
order by 1,2;
```

6. Donnez la liste des employés qui travaillent dans l'état d'Ontario.

```
SELECT e.last_name, e.first_name, city, state_province
FROM employees e
INNER JOIN departments d USING(department_id)
INNER JOIN locations l USING(location_id)
where state_province = 'Ontario';
```

7. Pour chaque implantation (*location*), donnez une adresse complète (rue, ville, province, pays). Faites un classement par numéro d'implantation.

```
SELECT location_id, Street_address, postal_code,
city, state_province, country_name
FROM locations
INNER JOIN Countries USING(country_id)
order by 1;
```

8. Donnez la liste des services, avec le nom complet de leur chef.

```
SELECT department_name, Last_Name as NomChef
FROM departments
INNER JOIN employees ON departments.manager_id=employee_id
order by 1;
```

9. Donnez la liste des services, avec le nom du pays où ils sont situés.

```
SELECT Department_name, country_name
FROM departments
INNER JOIN locations using(location_id)
INNER JOIN countries USING(country_id);
```

10. Même question, mais en classant les services, par régions, puis par pays.

```
SELECT region_name, country_name, department_name
FROM departments
INNER JOIN locations using(location_id)
INNER JOIN countries USING(country_id)
INNER JOIN regions USING(region_id)
ORDER BY 1,2,3;
```

11. Pour chaque employé, donnez son nom complet, le nom de son service, le nom complet de son chef de service et le nom complet de son supérieur direct.

```
SELECT e.last_name as NomEmp, e.first_name as PreEmp
, m.last_name as NomChefService, m.first_name as PrenomChefSe
, d.department_name as Service
,dm.last_name as NomCDirect, dm.first_name as PrenomCDirect
FROM Employees E
INNER JOIN employees M ON e.manager_id=m.employee_id
INNER JOIN departments D on d.department_id= e.department_id
```

```
INNER JOIN employees DM on D.manager_id=DM.employee_id
order by 1,2;
```

12. Donnez la liste des employés qui ont leur chef de service comme supérieur direct.

```
SELECT e.last_name as NomEmp, e.first_name as PreEmp
, m.last_name as NomChefService, m.first_name as PrenomChefSe
, d.department_name as Service
,dm.last_name as NomCDirect, dm.first_name as PrenomCDirect
FROM Employees E
INNER JOIN employees M ON e.manager_id=m.employee_id
INNER JOIN departments D on d.department_id= e.department_id
INNER JOIN employees DM on D.manager_id=DM.employee_id
WHERE dm.employee_id= m.employee_id
order by 1,2;
```

13. Donnez la liste des employés qui travaillent dans un service basé en Europe.

```
SELECT Last_Name, First_Name
FROM employees
INNER JOIN departments USING(department_id)
INNER JOIN locations USING(location_id)
INNER JOIN countries USING(country_id)
INNER JOIN regions USING(region_id)
WHERE regions.region_name='Europe';
```

14. Existe-t-il une région à laquelle ne serait rattaché aucun pays ?

```
SELECT Region_name, Country_name
FROM regions
LEFT JOIN countries using(region_id)
WHERE Country_name is null;
```

15. Donnez la liste des employés classés par fonction (*job*).

```
SELECT job_title, e.last_name, e.first_name
FROM Jobs j
INNER JOIN employees e USING(job_id)
order by 1,2;
```

16. Donnez la liste des fonctions éventuellement non occupées.

```
SELECT job_title, e.last_name, e.first_name
FROM Jobs j
LEFT JOIN employees e USING(job_id)
where j.job_title is null
order by 1,2;
```

17. Donnez la liste des employés dont on possède l'historique des fonctions

```
SELECT e.last_name, e.first_name
FROM Employees e
INNER JOIN job_history USING(employee_id);
```

18. Donnez la liste des employés dont on ne possède pas l'historique des fonctions (deux manières)

```
SELECT last_name, first_name
FROM employees
MINUS
SELECT e.last_name, e.first_name
FROM Employees e
INNER JOIN job_history USING(employee_id);
---
SELECT e.last_name, e.first_name
FROM Employees e
LEFT JOIN job_history USING(employee_id)
WHERE start_date is null;
```

19. Donnez la liste des employés qui touchent éventuellement un salaire plus important que celui qui est normalement prévu pour leur fonction.

```
SELECT e.last_name, e.first_name
FROM Employees e
INNER JOIN Jobs j USING(job_id)
where e.salary not between j.min_salary and j.max_salary;
```

20. Donnez la liste des emplois successifs des employés, y compris l'emploi actuel, avec le nom de la fonction et les dates de début et de fin.

```
SELECT e.last_name, e.first_name, job_title, start_date, end_date
FROM Employees e
INNER JOIN job_history jh using(employee_id)
INNER JOIN jobs on jh.job_id= jobs.job_id
UNION
SELECT e.last_name, e.first_name, job_title, hire_date, null
FROM Employees e
INNER JOIN jobs on e.job_id= jobs.job_id
order by 1,4,5;
```

21. Donnez la liste de tous les employés qui ont travaillé ou travaillent dans le service informatique (« IT »).

```
SELECT e.last_name, e.first_name, department_name, jh.end_date
FROM Employees e
INNER JOIN job_history jh using(employee_id)
INNER JOIN departments on jh.department_id= departments.department_id
where departments.department_name='IT'UNION
SELECT e.last_name, e.first_name, department_name,null
FROM Employees e
INNER JOIN departments on e.department_id = departments.department_id
where departments.department_name='IT'
order by 1,2;
```

2 Exercices à rendre sur feuille (Solutions)

1. Afficher le nom de toutes les personnes impliquées par les émissions de la chaîne.
2. Afficher les producteurs, avec le nom des séries qu'ils produisent.

```
SELECT NomProd, TitreSerie
FROM Producteurs P
INNER JOIN Series S ON P.idProducteur = S.idProducteur;
```

On peut également utiliser USING sur les systèmes qui l'autorisent

```
SELECT NomProd, TitreSerie
FROM Producteurs P
INNER JOIN Series S USING (idProducteur);
```

3. Afficher tous les présentateurs, avec le thème des émissions qu'ils animent et la date.

```
SELECT PrenomPres, Theme, DateDiffusion
FROM Presentateurs P1
INNER JOIN Presenter P2 ON P1.idPresentateur = P2.idPresentateur
INNER JOIN Emissions E ON E.IdEmission = P2.idEmission
ORDER BY 1,3;
```

4. Afficher la liste chronologique des émissions, en spécifiant le nom de la série et leur producteur.

```
SELECT DateDiffusion, Theme, TitreSerie, NomProd
FROM Producteurs P
INNER JOIN Series S ON P.idProducteur = S.idProducteur
INNER JOIN Emissions E ON S.idSerie = E.idSerie
ORDER BY 1;
```

5. Afficher la liste des invités, le thème et la date des émissions auxquelles ils participent.

```
SELECT NomInv, Theme, DateDiffusion
FROM Invites I
INNER JOIN Participer P ON I.idInvite = P.idInvite
INNER JOIN Emissions E ON E.IdEmission = P.idEmission
ORDER BY 1,3;
```

6. Afficher la liste des producteurs qui ne produisent pas de série.

```
SELECT NomProd
FROM Producteurs P
LEFT JOIN Series S ON S.idProducteur = P.idProducteur
WHERE idSerie IS NULL;
```

7. Afficher la liste des présentateurs qui travaillent pour des émissions produites par Georges Gorini.

```
SELECT Distinct PrenomPres
FROM Presentateurs P1
INNER JOIN Presenter P2 ON P1.idPresentateur = P2.idPresentateur
INNER JOIN Emissions E ON E.idEmission = P2.idEmission
INNER JOIN Series S ON S.idSerie = E.idSerie
INNER JOIN Producteurs P3 ON S.idProducteur = P3.idProducteur
WHERE NomProd = 'Gorini' AND PrenomProd='Georges';
```

8. Afficher les séries qui ont des émissions entre le 26 et le 30 septembre.

```
SELECT Distinct TitreSerie, DateDiffusion
FROM Emissions E
INNER JOIN Series S ON S.idSerie=E.idSerie
WHERE DateDiffusion Between date '2008-09-26' and date '2008-09-30';
```

On notera l'usage du « japonnais ».

9. Vérifier si la table *Presenter* est cohérente (ne présente pas de liens orphelins vers les présentateurs ou les émissions).

```
SELECT Distinct idEmission, idPresentateur
FROM Presenter P1
LEFT JOIN Presentateurs P2 ON P1.idPresentateur=P2.idPresentateur
LEFT JOIN Emissions E ON E.idEmission = P1.idEmission
WHERE E.idEmission IS NULL Or P2.idPresentateur IS NULL;
```

10. Afficher une liste chronologique des émissions, reprenant tous les renseignements sur leur correspondant : titre de la série, thème, producteur, présentateurs et invités.

```
SELECT DateDiffusion, TitreSerie, Theme, PrenomPres, NomProd, NomInv
FROM Emissions E
INNER JOIN Series S ON S.idSerie=E.idSerie
INNER JOIN Producteurs P0 ON P0.idProducteur=S.idProducteur
INNER JOIN Presenter P1  ON P1.idEmission = E.idEmission
INNER JOIN Presentateurs P2 ON P2.idPresentateur = P1.idPresentateur
INNER JOIN Participer P3 ON E.idEmission = P3.idEmission
INNER JOIN Invites I ON I.idInvite = P3.idInvite
ORDER BY 1;
```

11. Afficher les producteurs des émissions où Pierre Tchernia figure comme invité.

```
SELECT Distinct NomProd
FROM Emissions E
INNER JOIN Series S ON S.idSerie=E.idSerie
INNER JOIN Producteurs P0 ON P0.idProducteur=S.idProducteur
INNER JOIN Participer P3 ON E.idEmission = P3.idEmission
INNER JOIN Invites I ON I.idInvite = P3.idInvite
WHERE NomInv = 'Tchernia' AND PrenomInv = 'Pierre'
ORDER BY 1;
```

12. Afficher les producteurs et les invités présents à l'une des émissions qu'ils produisent.

```
SELECT Distinct NomProd, NomInv
FROM Emissions E
INNER JOIN Series S ON S.idSerie=E.idSerie
INNER JOIN Producteurs P0 ON P0.idProducteur=S.idProducteur
INNER JOIN Participer P3 ON E.idEmission = P3.idEmission
INNER JOIN Invites I ON I.idInvite = P3.idInvite
ORDER BY 1;
```

13. Afficher la liste des émissions (série+thème) qui n'ont pas d'invités.

```
SELECT TitreSerie, Theme
FROM Emissions E
```

```
INNER JOIN Series S ON S.idSerie=E.idSerie
LEFT JOIN Participer P3 ON E.idEmission = P3.idEmission
WHERE P3.idInvite IS NULL
ORDER BY 1;
```

14. Afficher les invités qui ne participent à aucune émission prévue.

```
SELECT PrenomInv, NomInv
FROM Invites I
LEFT JOIN Participer P3 ON I.idInvite = P3.idInvite
WHERE P3.idEmission IS NULL
ORDER BY 1;
```

15. Afficher les émissions présentées **conjointement** par Simon et Jacquot. Il ne sert à rien d'écrire une requête se terminant par `WHERE prenompres = 'Simon' AND prenompres='Jacquot'`, parce qu'aucun présentateur ne porte deux prénoms.

```
SELECT Theme
FROM emissions E
INNER JOIN presenter P1 ON P1.idemission = E.idemission
INNER JOIN presentateurs P2 ON p1.idpresentateur = p2.idpresentateur
WHERE prenompres = 'Simon'
AND idEmission in(
  SELECT idEmission
  FROM Presenter P3 ON P3.idemission = E2.idemission
INNER JOIN presentateurs P4 ON P3.idPresentateur = P4.idPresentateur
WHERE prenompres = 'Jacquot');
```

On peut aussi envisager une intersection :

```
SELECT Theme
FROM emissions E
INNER JOIN presenter P1 ON P1.idemission = E.idemission
INNER JOIN presentateurs P2 ON p1.idpresentateur = p2.idpresentateur
WHERE prenompres = 'Jacquot'
INTERSECT
SELECT Theme
FROM emissions E
INNER JOIN presenter P1 ON P1.idemission = E.idemission
INNER JOIN presentateurs P2 ON p1.idpresentateur = p2.idpresentateur
WHERE prenompres = 'Simon';
```

16. Réaliser les opérations ensemblistes suivantes sans utiliser les opérateurs `EXCEPT` (`MINUS`) et `INTERSECT` :
SELECT * FROM Ventes EXCEPT SELECT * FROM Ventes2 ;

```
SELECT * FROM Ventes MINUS SELECT * From Ventes2;
SELECT * FROM Ventes
WHERE NV NOT IN (Select NV FROM Ventes2);
SELECT V.* FROM Ventes V
LEFT JOIN Ventes2 V2 ON V.NV=V2.NV WHERE V2.N IS NULL;
```

SELECT * FROM Ventes2 **EXCEPT** SELECT * FROM Ventes ;
Il suffit d'inverser les tables dans les trois requêtes précédentes.
SELECT * FROM Ventes **INTERSECT** SELECT * FROM Ventes2 ;

```sql
SELECT * FROM Ventes INTERSECT SELECT * FROM Ventes2;

SELECT * FROM Ventes
WHERE NV IN (Select NV FROM Ventes2);

SELECT V.* FROM Ventes V
LEFT JOIN Ventes2 V2 ON V.NV=V2.NV
WHERE V2.N IS NOT NULL;
```

17. Écrire trois requêtes simulant les trois cas de jointures externes vu dans le cours, en utilisant successivement la syntaxe ancienne d'Oracle, puis les moyens que vous jugerez bons.

– jointure externe gauche

```sql
SELECT Prenom, Nom, Marque, Modele
FROM Employes2 LEFT JOIN Voitures
ON Employes2.NVoiture = Voitures.NVoiture ;

SELECT Prenom, Nom, Marque, Modele
FROM Employes2, Voitures
WHERE Employes2.NVoiture = Voitures.NVoiture(+);

SELECT Prenom, Nom, Marque, Modele
FROM Employes2 E,Voitures V
WHERE E.NVoiture=V.NVoiture
UNION
SELECT Prenom, Nom, Null, Null
FROM Employes2 E
WHERE NVoiture IS NULL;
```

Dans cette dernière solution, le champ *NVoiture* de la table *Employes2* est une clé étrangère et on peut donc déterminer les lignes qui ne sont pas reliées à la table *Voitures*.

– jointure externe droite

```sql
SELECT Prenom, Nom, Marque, Modele
FROM Employes2 RIGHT JOIN Voitures
ON Employes2.NVoiture = Voitures.NVoiture;

SELECT Prenom, Nom, Marque, Modele
FROM Employes2, Voitures
WHERE Employes2.NVoiture(+) = Voitures.NVoiture;

SELECT Prenom, Nom, Marque, Modele
FROM Employes2 E,Voitures V
WHERE E.NVoiture=V.NVoiture
UNION
SELECT Null, Null, Marque, Modele
FROM Voitures V
WHERE NVoiture NOT IN
  (SELECT E.NVoiture
   FROM Employes2 E WHERE NVoiture IS NOT NULL);
```

La deuxième partie de l'union n'est pas symétrique de la jointure gauche, parce qu'il n'y a pas de clé étrangère à tester. Il faut donc tester la présence de la clé *NVoiture* comme clé étrangère de l'autre table. La clause `WHERE` de la requête imbriquée est nécessaire : sans elle, la requête renvoie parmi ses valeurs une valeur nulle, donc inconnue, qui rend non vrai le test `NOT IN`.

– jointure complète

```
SELECT Prenom, Nom, Marque, Modele
FROM Employes2 FULL OUTER JOIN Voitures USING (NVoiture);
```

La syntaxe Oracle n'accepte qu'un `(+)`. Par contre, on peut combiner les jointures gauche et droite réalisées avec les unions pour obtenir une jointure complète.

```
SELECT Prenom, Nom, Marque, Modele
FROM Employes2 E,Voitures V
WHERE E.NVoiture=V.NVoiture
UNION
SELECT Prenom, Nom, Null, Null
FROM Employes2 E
WHERE NVoiture IS NULL
UNION
SELECT Null, Null, Marque, Modele
FROM Voitures V
WHERE NVoiture NOT IN
  (SELECT E.NVoiture
   FROM Employes2 E WHERE NVoiture IS NOT NULL);
```