Systèmes de gestion de bases de données
Exercices sur les fonctions de synthèse et les requêtes imbriquées (série 3)

Cours de Jacques THOORENS

1 Exercices de laboratoire (solutions)

1. Compter le nombre de pays cités dans le schéma HR.

```
SELECT Count(*) FROM Countries;
```

2. Compter le nombre de pays situés dans chaque région.

```
SELECT Region_Name, count(Country_Name)
FROM Regions
INNER JOIN Countries USING(Region_id)
GROUP BY Region_Name;
```

3. Calculez la moyenne des salaires de chaque service.

```
SELECT Department_Name, AVG(Salary) AS Moyenne
FROM Departments
INNER JOIN Employees USING(Department_id)
GROUP BY Department_Name;
```

4. Déterminez la moyenne des salaires de chaque pays.

```
SELECT Country_Name, AVG(Salary) AS Moyenne
FROM Employees
INNER JOIN Departments USING(Department_id)
INNER JOIN Locations USING(Location_id)
INNER JOIN Countries USING(Country_id)
GROUP BY Country_Name;
```

5. Pour chaque service, donnez le salaire minimal, maximal et le nombre de barêmes différents.

```
SELECT Department_Name, min(Salary), max(Salary), COUNT(DISTINCT Salary)
FROM Departments
INNER JOIN Employees USING(Department_id)
GROUP BY Department_Name;
```

6. Donner la liste des services, avec chaque fois le nombre d'employés qui y travaillent.

```
SELECT Department_Name,COUNT(First_Name)
FROM Departments
INNER JOIN Employees USING(Department_id)
GROUP BY Department_Name;
```

7. Quels sont les pays où travaillent plus de 25 employés.

```
SELECT Country_name,count(last_name)
FROM countries
INNER JOIN locations using(country_id)
INNER JOIN departments using(location_id)
INNER JOIN employees using(department_id)
GROUP BY countries.country_name
HAVING count(last_name)>25;
```

8. Pour chaque manager, donnez le nombre d'employés qui dépendent directement de lui. Les managers sont en fait des employés que d'autres désignent comme leur manager (par le biais de *manager_id*).

```
SELECT M.First_Name, COUNT(E.First_Name)
FROM Employees M
INNER JOIN Employees E ON (M.Employee_id = E.Manager_id)
GROUP BY M.First_Name;
```

9. Pour chaque manager, donnez la liste des employés sur qui il a autorité. On a vu dans la question précédente que chaque employé avait un manager. Mais il faut tenir compte aussi du fait qu'un employé fait partie d'un service(*department*), dirigé par son manager. Attention, il peut arriver qu'un employé ait le chef de son service comme manager direct, il ne faut donc pas le citer deux fois (**question complexe**).

   La question n'était sans doute pas trop claire non plus. J'entendais les personnes qui dépendent du manager directement comme dans la question précédente, mais aussi par le fait que certains sont les managers d'un service (*department*). Il faut éviter de compter deux foix les mêmes employés (un employé peut dépendre d'un autre parce que c'est son manager et parce que c'est le manager de son service).

```
SELECT Manager_Name, COUNT(Employee_Name)
FROM
( -- Les subordonnés directs
  SELECT M.First_Name AS Manager_Name, E.First_Name AS Employee_Name
  FROM Employees M
  INNER JOIN Employees E ON (M.Employee_id = E.Manager_id)
  UNION -- élimine implicitement les doublons
  -- Les subordonnées appartenant au service
  Select M2.First_name, E2.First_name
  FROM employees E2
  INNER JOIN departments D USING(Department_id)
  INNER JOIN employees M2 ON d.manager_id= m2.employee_id
)
GROUP BY Manager_name
ORDER BY 1;
```

   On crée une union entre les deux types de subordonnées dans une requête imbriquée. Notons que les alias dans le premier select donne des noms utilisables dans la requête externe.

10. Quels sont les départements où on trouve un salaire moyen supérieur au salaire moyen dans l'entreprise.

```
SELECT Department_name, avg(salary)
FROM departments
```

```
INNER JOIN employees USING(department_id)
GROUP BY department_name
HAVING avg(salary)>(SELECT avg(salary) FROM employees);
```

11. Citez les villes où il n'y a pas de programmeur.

    Pour ce genre de question, il suffit de trouver les villes où il y a des programmeurs. Les villes où il n'y en pas, ce sont celles qui restent. Attention, cette première requête est fausse. Elle énumère les villes où il y a autre chose que des programmeurs. Le fait qu'il y ait des vendeurs de rollmops (qui ne sont pas programmeurs) quelque part n'interdit pas qu'il y ait aussi des programmeurs. Rappel important : une recherche d'inexistence ne peut jamais se faire avec une requête simple assortie d'une clause WHERE. Par un malheureux hasard, tous les programmeurs de la base sont à SouthLake.

```
-- Solution fausse !!!!!
SELECT DISTINCT City
FROM locations
INNER JOIN departments USING(location_id)
INNER JOIN employees USING(department_id)
INNER JOIN jobs USING(job_id)
WHERE upper(job_title) <> 'PROGRAMMER';
-- Fin de la solution fausse !!!!!
```

    Voici la solution correcte

```
SELECT City
FROM Locations
MINUS
SELECT City
FROM locations
INNER JOIN departments USING(location_id)
INNER JOIN employees USING(department_id)
INNER JOIN jobs USING(job_id)
WHERE upper(job_title) = 'PROGRAMMER';
```

    On pourrait aussi avoir

```
SELECT DISTINCT City
FROM Locations
WHERE City NOT IN(
   SELECT City ...)
```

    Dernière remarque : on pourrait vouloir simplifier la vie à Oracle en ajoutant `DISTINCT` dans la requête imbriquée. Ce n'est pas une bonne idée car on rend la requête plus lente. J'ai expérimenté une dégradation significative des performances d'une telle « amélioration » sur une base Access. Il ne faut pas oublier que SQL décrit le résultat demandé et pas la manière de l'obtenir. Il est donc difficile d'optimiser sans savoir comment le système procède pour y parvenir.

12. Existe-t-il un type de travail qui se retrouve dans toutes les villes ? Citez-le ou citez-les.

```
SELECT job_title, count(distinct city)
FROM Locations
INNER JOIN departments USING(location_id)
INNER JOIN employees USING(department_id)
INNER JOIN jobs USING(job_id)
```

```
GROUP BY job_title;
HAVING count(distinct city)=(select count(distinct city)
FROM locations)
```

L'exécution de cette requête renvoie NULL et c'est normal, puisque chaque type de travail se cantonne dans une seule ville (dans les données fournies).

13. Comptez le nombre d'employés qui gagnent plus que le salaire moyen de leur pays.

```
SELECT First_Name, Salary
FROM employees e
INNER JOIN departments USING(department_id)
INNER JOIN locations l USING(location_id)
WHERE salary >(
SELECT avg(salary)
FROM employees
INNER JOIN departments USING(department_id)
INNER JOIN locations USING(location_id)
WHERE country_id = l.country_id);
```

Je n'ai pas fait de jointure sur la table *Countries* puisque le nom du pays n'est pas requis. On se contente de l'identifiant.

14. Citez le nom des employés qui travaillent dans le plus gros service.Requête finale :

```
SELECT First_name
FROM employees
WHERE department_id=(
  SELECT department_id
  FROM employees
  GROUP BY department_id
  HAVING count(*)=(SELECT max(count(*))
    FROM employees
    GROUP BY department_id));
```

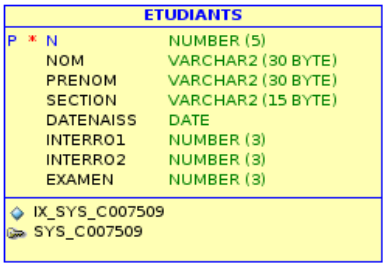
Requêtes partielles pour y parvenir :

```
-- Nombre d'employés dans le plus gros département
SELECT max(count(*))
FROM employees
GROUP BY department_id;

-- Nombre d'employés par département
SELECT department_id,count(*)
  FROM employees
  GROUP BY department_id;

-- Noms des employés du plus gros département (50)
SELECT First_name,department_id
FROM employees
WHERE department_id= 50;
```

2 Exercices à rendre sur feuille (Solutions)

2.1 Première partie

Réaliser les exercices suivant, en utilisant la table *Etudiants* présente dans le schéma *Demo*.



1. Afficher le nombre d'étudiants se trouvant dans chaque section.

```
SELECT Section, Count(Nom)
FROM Etudiants
GROUP BY Section;
```

2. Afficher les étudiants qui ont obtenu le meilleur résultat à l'examen dans chaque section ainsi que leur points.

```
SELECT Section, Nom, Examen
FROM Etudiants R
WHERE Examen = (
   SELECT Max(Examen)
   FROM Etudiants
   WHERE Section = R.Section)
ORDER BY 1,2;
```

La requête imbriquée recherche le meilleur résultat pour la classe dans laquelle figure l'étudiant de la requête principale. On vérifie, au niveau de la requête principal, que son résultat est identique au meilleur de sa classe. On notera ici l'astuce de la liaison entre la requête et la requête imbriquée au moyen d'un alias. On pourrait proposer la solution SQL2/Oracle suivante (qui n'est pas acceptée par InterBase) :

```
SELECT Section, Nom, Examen
FROM Etudiants E
WHERE (Section,Examen) IN
  ( SELECT Section, Max(Examen)
    FROM Etudiants
    GROUP BY Section)
ORDER BY 1,2;
```

3. Afficher la liste des étudiants qui obtiennent le meilleur résultat dans chaque section

```
SELECT Section, Nom, (Interro1+Interro2+Examen) AS TOTAL
FROM Etudiants R
WHERE (Interro1+Interro2+Examen) = (
   SELECT Max(Interro1+Interro2+Examen)
```

```
   FROM Etudiants
   WHERE Section = R.Section)
ORDER BY 1,2;
```

Il n'y a malheureusement pas moyen d'éviter le triple calcul du total, à moins de réaliser une vue sur la table.

4. Afficher la moyenne des points aux différentes épreuves, par section.

```
SELECT Section, AVG(Interro1) AS I1,
AVG(Interro2) AS I2, AVG(Examen) AS E
FROM Etudiants
GROUP BY Section;
```

5. Afficher la moyenne des résultats finaux par section.

```
SELECT Section, Avg(Interro1+Interro2+Examen) AS Total
FROM Etudiants
GROUP BY Section;
```

6. Afficher le nombre d'échecs au total dans chaque section.

```
SELECT Section, Count(Nom)
FROM Etudiants
WHERE (Interro1+Interro2+Examen) < 15
GROUP BY Section;
```

7. Afficher le nombre d'étudiants dans chaque classe de points, par section.

```
SELECT Section, Examen, Count(Nom)
FROM Etudiants
GROUP BY Section, Examen
ORDER BY Section, Examen DESC;
```

8. Afficher la liste des étudiants qui sont dans une classe où certains étudiants ont obtenu plus de 23 au total.

```
SELECT Section, Nom, Prenom
FROM Etudiants WHERE Section IN (
  SELECT Section
  FROM Etudiants WHERE Interro1+Interro2+Examen > 23)
ORDER BY 1,2,3;
```

9. Afficher la liste des étudiants qui sont dans une classe où il n'y a pas d'échec à l'examen.

```
SELECT Nom
FROM Etudiants
WHERE Section NOT IN (
   SELECT Section
   FROM Etudiants
   WHERE Examen < 5)
ORDER BY Section, Nom ;
```

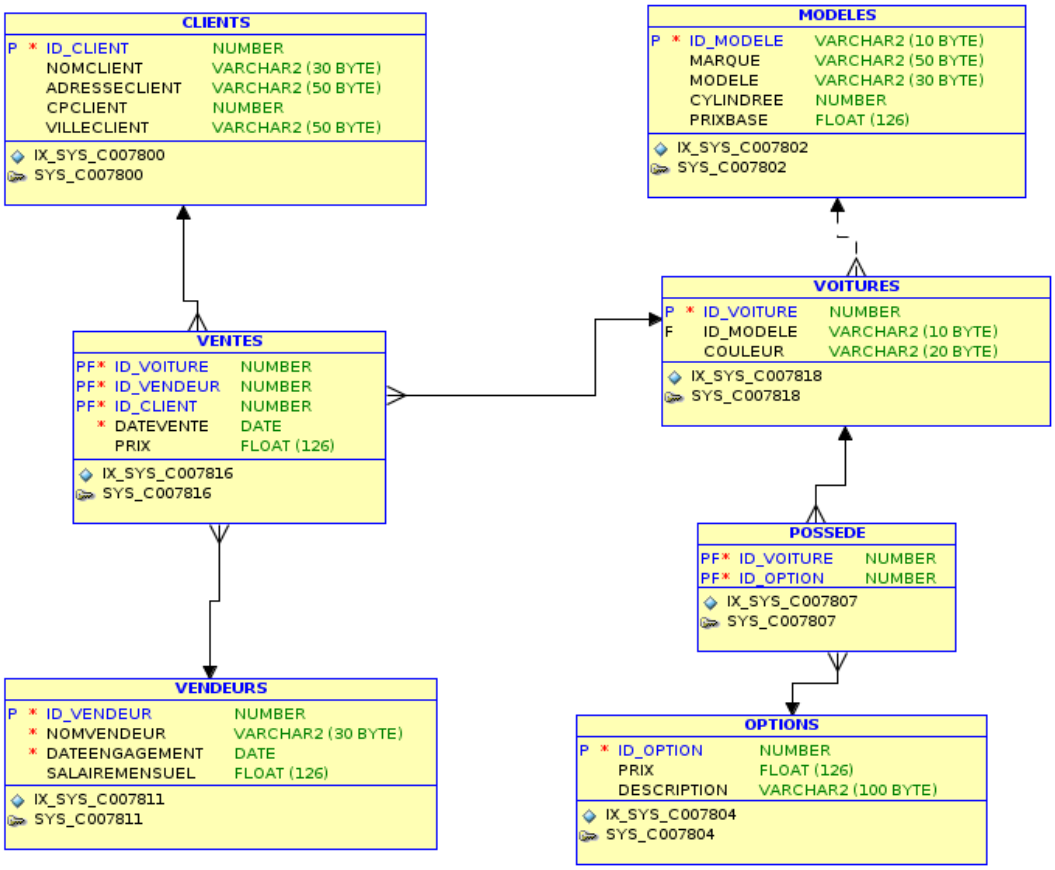
Variante :

```
SELECT Section, Nom
FROM Etudiants
WHERE Section IN(
    SELECT Section
    FROM Etudiants
    GROUP BY Section
    HAVING Min(Examen) >= 5)
ORDER BY Section, Nom ;
```

Comme aucune classe ne peut se targuer d'avoir tous les élèves qui réussissent, on peut vérifier le fonctionnement des deux requêtes en abaissant les exigences à 4 (`< 4` ou `>=4` respectivement).

2.2 Deuxième partie

En utilisant les tables du garage



Les associations suivantes sont définies :

- une vente concerne un vendeur, un client et une voiture
- une voiture appartient à un modèle
- une voiture possède zéro, une ou plusieurs options et chaque option peut se retrouver sur aucune voiture, ou une ou plusieurs.

10. Afficher le nombre de voitures vendues par vendeur.

```
SELECT NomVendeur, Count(DateVente)
FROM Vendeurs Ve INNER JOIN Ventes Vt
ON Ve.id_Vendeur = Vt.Id_Vendeur
GROUP BY NomVendeur;
```

11. Afficher le nombre d'accessoires en options vendus, dans chaque catégorie.

```
SELECT Description, count(O.id_Option)
FROM Options O
INNER JOIN Possede P ON O.id_Option=P.id_Option
GROUP BY Description
ORDER BY Description;
```

12. Afficher le nom du vendeur qui a vendu le plus de voitures.

```
SELECT NomVendeur, Count(Id_Vendeur)
FROM Vendeurs Vd
INNER JOIN Ventes Vt
ON Vd.id_Vendeur= Vt.id_Vendeur
GROUP BY NomVendeur
HAVING Count(id_Vendeur) >= ALL(
  SELECT Count(id_Vendeur)
  FROM Ventes
  GROUP BY id_Vendeur);
```

La requête imbriquée établit la liste du nombre de ventes pour chaque vendeur. Il ne reste plus à vérifier pour chaque vendeur qu'il a vendu autant ou plus que tous les autres. Il faut >= parce que le score du meilleur vendeur se trouve aussi dans la liste.

13. Afficher le nom du vendeur qui a vendu le moins de voitures (mais qui en a quand même vendu).

Il suffit d'inverser l'opérateur de comparaison dans la clause `HAVING` de la requête précédente.

14. Calculez le prix des options pour chaque voiture.

```
SELECT P.id_Voiture, Sum(Prix)
FROM Options O
INNER JOIN Possede P ON O.id_Option = P.id_Option
GROUP BY id_Voiture;
```

15. Affichez les vendeurs qui n'ont pas vendu, sans utiliser de jointures extérieures.

```
SELECT NomVendeur
FROM Vendeurs
WHERE id_Vendeur NOT IN(
SELECT id_Vendeur
FROM Ventes);
```

On affiche le nom des vendeurs qui ne figurent pas dans la liste des vendeurs.

16. Quel vendeur n'a jamais vendu d'auto radio ?

```
SELECT NomVendeur
FROM Vendeurs
```

```
WHERE NOT Id_Vendeur IN(
  SELECT Id_Vendeur
  FROM Ventes Ve
  INNER JOIN Possede P ON Ve.Id_Voiture = P.Id_Voiture
  INNER JOIN Options O ON O.Id_Option = P.Id_Option
  WHERE Description = 'Autoradio' );
```

La requête imbriquée est ici indispensable. Une requête multitable éliminera forcément celui qui n'a pas vendu d'autoradio. La solution consiste ici à faire l'inverse : vérifier les vendeurs qui ne figurent pas dans la liste des vendeurs d'autoradios.

17. Quelles sont les voitures qui n'ont pas de jantes en aluminium ?

```
SELECT id_Voiture
FROM Voitures
WHERE id_Voiture NOT IN(
  SELECT id_Voiture
  FROM Options O
  INNER JOIN Possede P ON O.id_Option = P.id_Option
  WHERE Description Like '%alu%');
```

18. Comment exprimer une semi-jointure sans utiliser `JOIN` ? Cherchez plusieurs solutions.

Soit une table T1 et une table T2 et une clé primaire et étrangère idK :

on crée une requête qui reprend toutes les valeurs de idK dans T2 et on ajoute une clause `WHERE idK IN(RequêteImbriquée)` pour lister les éléments de T1.