Le LANGAGE DE MANIPULATION DE DONNEES SQL

(ORACLE / MySQL)

1. Commandes de Manipulations

Il y a les commandes de modification des données (INSERT, UPDATE et DELETE) et la commande SELECT d'interrogation de données.

Voir la base de données exemple.

Les Commandes de Modification de Données.

La commande d'Insertion de Lignes

La commande INSERT permet d'insérer des lignes dans une table. Elle revêt trois formes :

- INSERT INTO *nom-de-table* VALUES (*liste-de-valeurs*);

  Exemple : `INSERT INTO employee VALUES ('E7', 'Rim', 15000, 'Rabat', 'D1');`

  Cette forme permet d'insérer une ligne unique, avec une valeur pour toutes les colonnes. S'il y a un doublon pour une colonne à valeur clé ou unique, l'insertion est refusée.

- INSERT INTO *nom-de-table* (*colonnes*) VALUES (*liste-de-valeurs*);

  Exemple : `INSERT INTO employee (enum, salary) VALUES ('E7', 15000);`

  Cette forme permet d'insérer une ligne unique, avec une valeur pour les colonnes dont on connait la valeur. Les autres valeurs seront égale à NULL, si possible (colonnes non déclarée NOT NULL).
  S'il y a un doublon pour une colonne à valeur clé ou unique, l'insertion est refusée.

- INSERT INTO *nom-de-table* SELECT ...

  Cette forme permet d'insérer dans une table les lignes résultats de SELECT. Le schéma du SELECT doit correspondre à celui de la table. On peut aussi nommer les colonnes concernées comme dans la forme précédente.

```
CREATE TABLE maTable (numero varchar(5), adresse varchar(15);
INSERT INTO maTable SELECT enum, address FROM employee;
```

La commande de Modification de Valeurs

La commande UPDATE permet de modifier les composants d'une ligne dans une table.

Update

UPDATE *nom-table* SET *colonne* = *valeur*, *colonne* = *valeur*, ... WHERE *condition*;

Delete

DELETE FROM *nom-table* WHERE *condition*;

NB. Ne pas confondre avec DROP TABLE.

Astuce : Dans la syntaxe du LMD SQL, le mot clé TABLE n'est pas utilisé.

Select

Les requêtes SELECT font l'objet de tout ce qui suit.

2. Langage d'interrogation

Commande SELECT

On a la base de données suivante sur une organisation de ventes :

La liste des employés

```
enum ename               salary address    dept

E7   Amine              7500.00 Fes        D2
E6   Aziz               8500.00 Casa       D1
E5   Amina              8000.00 Rabat      D3
E4   Said               5000.00 Agadir     D3
E3   Fatima             7000.00 Tanger     D2
E2   Ahmed              6000.00 Casa       D1
E1   Ali                8000.00 Rabat      D1
E8   Ahmed              4000.00 Casa       D4
```

La liste des départements

```
dnum dname            floor mgr

D4   HiFi                3 E8
D3   Livres              2 E5
D2   Alimentation        3 E3
D1   Jouets              1 E1
D5   Bazar               2 E1
```

La liste des produites vendus

```
pnum pname           weight      price city

P5   Linux               3       5.00 Fes
P4   Java                3       5.00 Rabat
P3   Eclair              1       1.00 Tetouan
P2   Barbie              3       4.50 Rabat
P1   ColdPlay            2       2.00 Casa
P6   Orangina            2       3.50 Agadir
```

et enfin, les ventes par département

```
dnum pnum         qty
```

```
D1   P2          100
D2   P3          200
D2   P6          300
D3   P4          100
D5   P6          100
D5   P5          200
D5   P4          400
D5   P3          300
D5   P2          100
D3   P5          300
D5   P1          200
```

1. Requêtes Simples

a) « *Liste de tous les employées* »

```
select *
from employee;
```

Résultat :

```
enum ename              salary address    dept

E7   Amine             7500.00 Fes        D2
E6   Aziz              8500.00 Casa       D1
E5   Amina             8000.00 Rabat      D3
E4   Said              5000.00 Agadir     D3
E3   Fatima            7000.00 Tanger     D2
E2   Ahmed             6000.00 Casa       D1
E1   Ali               8000.00 Rabat      D1
E8   Ahmed             4000.00 Casa       D4
```

b) Avec colonnes choisis :

« *Le numéro et nom de tous les employés* »

```
select enum, ename
from employee;
```

résultat :

```
enum ename

E7   Amine
E6   Aziz
E5   Amina
E4   Said
E3   Fatima
E2   Ahmed
E1   Ali
E8   Ahmed
```

On peut qualifier les champs par leur table et écrire

```
select employee.enum, employee.ename
from employee;
```

c) Avec clause de restriction,

« *le numéro et le nom des employés du département* 'D1' »

```
select enum, ename
from employee
where dept = 'D1';
```

Résultat :

```
enum ename

E6   Aziz
E2   Ahmed
E1   Ali
```

d) Avec conditions mixées,

« *le numéro et le nom des employés du département* 'D1' *et ayant un salaire* > 7000 »

```
select enum, ename
from employee
where dept = 'D1' and salary > 7000;
```

Résultat :

```
enum ename

E6   Aziz
E1   Ali
```

e) On peut spécifier un ordre d'affichage :

« *le nom et le salaire des employés, classés par ordre des salaires* »

```
select ename, salary
from employee
order by salary;
```

L'ordre est croissant de façon implicite.

Résultat :

```
ename               salary

Ahmed              4000.00
Said               5000.00
Ahmed              6000.00
Fatima             7000.00
Amine              7500.00
Amina              8000.00
Ali                8000.00
Aziz               8500.00
```

f) Idem, mais si les salaires sont identiques, afficher les noms en ordre aussi.

```
select ename, salary
from employee
order by salary , 1 desc;
```

g) Noms en ordre inverse ici.

```
ename               salary

Ahmed              4000.00
```

```
Said                5000.00
Ahmed               6000.00
Fatima              7000.00
Amine               7500.00
Amina               8000.00
Ali                 8000.00
Aziz                8500.00
```

h) On peut combiner les clauses `where` et `order by`,

« *le nom et le salaire des employés du département* '`D1`', *affichés par salaires croissants* »

```
select ename, salary
from employee
where dept = 'D1'
order by salary;
```

résultat :

```
ename               salary

Ahmed               6000.00
Ali                 8000.00
Aziz                8500.00
```

i) Usage de `distinct`, qui permet d'éliminer les lignes doubles.

Avec

```
select dnum
from sell;
```

on a la table :

```
dnum

D1
D2
D2
D3
D5
D5
D5
D5
D5
D3
D5
```

et avec

```
select distinct dnum
from sell;
```

on a la table :

```
dnum

D1
D2
D3
D5
```

2. Requêtes Simples avec expressions

a) « *Donner pour chaque produit son poids en gramme* (*weight* x 1000) »

```
select pnum, weight * 1000
from product;
```

Résultat :

```
pnum     (expression)

P5               3000
P4               3000
P3               1000
P2               3000
P1               2000
P6               2000
```

Le champ calculé n'as pas de nom final, mais on peut le lui donner en indiquant un alias, e.g. `Gramme`

b)

```
select pnum, weight*1000 Gramme
from product;
```

Résultat :

```
pnum           gramme

P5               3000
P4               3000
P3               1000
P2               3000
P1               2000
P6               2000
```

Les éléments résultats dans la ligne `select` sont des expressions séparées par vigule. Un alias éventuel est signalé après une expression et séparé par un espace. L'expression la plus simple est un attribut d'une table.

Mais cela peut être une expression quelconque. La requête suivante n'est pas si curieuse que cela.

```
select sin(3.14159 / 2);
+------------------+
| sin(3.14159 / 2) |
+------------------+
| 0.99999999999912 |
+------------------+
```

Un langage de manipulation de bases de données, aussi puissant soit-il, doit permettre d'effectuer les calculs de base.

3. Interrogation de plusieurs tables (Jointure relationnelle)

a) Parfois, on a besoin de consulter deux tables.

« *Afficher toutes les informations sur les employés et le département où ils travaillent.* »

```
select employee.*, department.*
from employee, department
where employee.dept=department.dnum;
```

Dans la clause `from`, on annonce les deux tables. Le lien se fait par la colonne commune, à savoir `dept` de `employee` et `dnum` de `department`. On appelle ce lien, *critère de jointure*.

```
enum ename               salary address    dept   dnum dname            floor mgr

E7   Amine              7500.00 Fes        D2     D2   Alimentation         3 E3
E6   Aziz               8500.00 Casa       D1     D1   Jouets               1 E1
E5   Amina              8000.00 Rabat      D3     D3   Livres               2 E5
E4   Said               5000.00 Agadir     D3     D3   Livres               2 E5
E3   Fatima             7000.00 Tanger     D2     D2   Alimentation         3 E3
E2   Ahmed              6000.00 Casa       D1     D1   Jouets               1 E1
E1   Ali                8000.00 Rabat      D1     D1   Jouets               1 E1
E8   Ahmed              4000.00 Casa       D4     D4   HiFi                 3 E8
```

NB. La forme de la requête est détaillée ici (compréhensibilité). Le critère de jointure peut aussi s'écrire `dept=dnum`, car les champs de jointure ont des noms différents, et la première ligne peut s'écrire `select *` tout court.

Remarquer que le département 'D5' ne figure pas, car personne n'y travaille. (Voir jointure externe, plus bas).

a-bis) Même requête avec l'ordre `join` dans la clause `from`. Sans la clause `where` donc.

```
select employee.*, department.*
from employee join department on employee.dept=department.dnum;
```

Cette forme, plus algébrique, permet quand c'est possible de réserver la clause `where` pour les restrictions (cf. requête **d** ci-après). Mais cette forme est surtout utilisée avec `outer join`. cf. **6.3.d)** ci-desous.

b) Sans clause de jointure, on a le produit cartésien (toutes les combinaisons de lignes des deux tables) :

```
select employee.*, department.*
from employee, department;
```

```
enum ename               salary address    dept   dnum dname            floor mgr

E7   Amine              7500.00 Fes        D2     D2   Alimentation         3 E3
E7   Amine              7500.00 Fes        D2     D1   Jouets               1 E1
E7   Amine              7500.00 Fes        D2     D3   Livres               2 E5
E7   Amine              7500.00 Fes        D2     D4   HiFi                 3 E8 E7   Amine              7500.00 Fes        D2     D5   Bazar                2 E1
                             .....
E8   Ahmed              4000.00 Casa       D4     D4   HiFi                 3 E8
E8   Ahmed              4000.00 Casa       D4     D5   Bazar                2 E1
```

40 (8 x 5) lignes au total.

c) Mais il est plus interessant se spécifier les informations que l'on souhaite :

« *Donner pour chaque employé, son numéro, son nom, ainsi que le département et l'étage où il travaille.* »

```
select employee.enum, employee.ename, department.dname, department.floor
from employee, department
where employee.dept=department.dnum;
```

Résultat :

```
enum ename           dname            floor

E7   Amine           Alimentation         3
E6   Aziz            Jouets               1
E5   Amina           Livres               2
E4   Said            Livres               2
E3   Fatima          Alimentation         3
E2   Ahmed           Jouets               1
E1   Ali             Jouets               1
E8   Ahmed           HiFi                 3
```

Là aussi, on aurait pu écrire les nom de colonne sans qualification par le nom de table, i.e.

```
select enum, ename, dname, floor
```

d) On peut rajouter une condition supplémentaire au critère de jointure.
« *Quels sont les employés du 3e étage*? »

```
select ename, dname
from employee, department
where employee.dept=department.dnum
and floor = '3';
```

```
+--------+--------------+
| ename  | dname        |
+--------+--------------+
| Ahmed  | HiFi         |
| Amine  | Alimentation |
| Fatima | Alimentation |
+--------+--------------+
```

d) Avec la notation qui distingue la condition de restriction de celle de jointure.

```
select ename, dname
from employee join department on dept = dnum
where floor = '3';
```

```
+--------+--------------+
| ename  | dname        |
+--------+--------------+
| Ahmed  | HiFi         |
| Amine  | Alimentation |
| Fatima | Alimentation |
+--------+--------------+
```

e) Parfois, on a besoin de consulter sur trois table, et joindre donc les trois.

« *Donner le nom de chaque département ainsi que le noms des produits qu'il vend.* »

```
select dname, pname
from product , sell, department
where  product.pnum = sell.pnum
and department.dnum = sell.dnum;
```

Résultat :

```
dname           pname

Livres          Java
Livres          Linux
Alimentation    Eclair
Alimentation    Orangina
Jouets          Barbie
Bazar           ColdPlay
Bazar           Barbie
Bazar           Eclair
Bazar           Java
Bazar           Linux
Bazar           Orangina
```

f) On peut avoir besoin de consulter deux fois la même table. Par exemple pour

« *Afficher par paires les employés habitant la même ville.* »

```
select x.ename, y.ename
from employee x, employee y
where  x.address = y.address
and x.enum > y.enum;
```

On emploie alors des variables (*alias*) comme x et y ici, pour se referer à deux employés simultanement et comparer leur ville.

```
ename           ename

Ahmed           Aziz
Aziz            Ahmed
Ahmed           Ahmed
Amina           Ali
```

La condition supplémentaire `x.enum > y.enum`, est une astuce pour elimier une des paires symétriques ou deux fois la même personne.

g) Une requête caractéristique est *le nom des employés qui gagnent plus que leur manager*. Jointure entre trois table, dont deux fois la même.

```
select x.ename
from employee x, employee y, department d
where x.dept = d.dnum
and d.mgr = y.enum
and y.salary < x.salary;
```

Ce qui donne :

```
ename

Amine
Aziz
```

4. Fonctions incorporées

Il y a 5 fonctions incorporées : *count*(), *avg*(), *sum*(), *max*() et *min*(). le nombre d'éléments dans un ensemble, la moyenne, le total, la maximum ou le minimum d'un ensemble de valeurs.

a) « *Le nombre total des employés* »

```
select count(*)
from employee;
```

résultat :

```
(count(*))

       8
```

C'est en fait, le nombre de lignes de la table.

b) Usage de distinct pour ne pas compter deux fois la même valeur

```
select count (distinct ename)
from employee;
```

résultat :

```
 (count)

       7
```

c) La moyenne des quantités vendues

```
select avg(qty)
from sell;
```

résultat :

```
           (avg)

          209.09
```

c) La moyenne des quantités vendues pour le produit 'P4'.

```
select avg(qty)
from sell
where pnum = 'P4';
```

résultat :

```
           (avg)

          250.00
```

ou de façon équivalente

```
select sum(qty) / count(*)
from sell
where pnum = 'P4';
```

résultat :

```
    (expression)

          250.00
```

5. Usage de group by

a) La clause `group by`, permet de faire des calculs par groupe de lignes (*vs*. sur toutes les lignes résultats)

« *La moyenne des quantités vendues par produit* »

```
select pnum, avg(qty)
from sell
group by pnum;
```

résultat :

```
pnum            (avg)

P2             100.00
P3             250.00
P6             200.00
P4             250.00
P5             250.00
```

```
P1            200.00
```

Idem, mais classé par ordre croissant de moyennes

```
select pnum, avg(qty)
from sell
group by pnum
order by 2;
```

résultat :

```
pnum           (avg)

P2            100.00
P6            200.00
P1            200.00
P3            250.00
P4            250.00
P5            250.00
```

b) La clause `group by`, permet de faire des calculs par groupe de lignes. On peut chercher aussi le groupe de lignes ayant (having) une certaine propriété.

« *Quel est le total des quantités vendues par produit vendu en quantité moyenne supérieure à 200.* »

```
select pnum, sum(qty)
from sell
group by pnum
having avg(qty) >200;
```

résultat :

```
pnum           (sum)

P3               500
P4               500
P5               500
```

`Having` est à un groupe de lignes ce que `where` est à une ligne.

Requête complète

c) La requête suivante met en oeuvres toutes les clauses déjà vues.

« *Quels sont par département, le nom et la somme des quantités vendues, pour les départements situés en dessus du 1er étage, sachant que la quantité moyenne vendue par ce département est supérieure à 200. Présenter le résultat en ordre alphabétique inverse.* »

```
select d.dname, sum(qty)
from department d, sell s
where d.floor >1 and
      d.dnum = s.dnum
group by d.dname
having avg(qty)>200
order by 1 desc;
```

résultat :

```
dname                    (sum)

Bazar                     1300
Alimentation               500
```

5-bis. Relations temporaires

On peut créer des relations temporaires, t_1 et t_2 ici, pour stocker un résultat intermédiaire.

« produit dont le poids s'ecarte le plus de la moyenne »

```
create table t1 (moy decimal(8,2));

insert into t1 select avg(weight)
               from product;
select * from t1;
```

résultat :

```
       moy

      2.33


create table t2 (pnum char(4), ecart decimal(8,2));

insert into t2 select pnum, weight-moy
               from product, t1;
update t2 set ecart = ecart*(-1)
          where ecart <0;
select * from t2;
```

résultat :

```
pnum      ecart

P5         0.67
P4         0.67
P3         1.33
P2         0.67
P1         0.33
P6         0.33

select pnum
from t2
where ecart = (select max(ecart)
               from t2);
```

résultat :

```
pnum

P3

drop table t1; drop table t2;
```

6. Caractéristiques Avancées

6.1 Clause LIKE

Permet de comparer par rapport à un motif. Dans le motif, '`%`' signifie zéro, un ou plusieurs caractères (une chaîne quelconque), et '_' signifie un caractère et un seul. '`%a%`' est n'importe que mot contenant la lettre `a`, et '`Raba_`' un mot de 5 lettres commençant par `Raba`.

```
select ename, address
from employee
```

```
where address like 'Raba_';
```

résultat :

```
ename           address

Ali             Rabat
Amina           Rabat
```

```
select ename, address
from employee
where address like '%a%';
```

résultat :

```
ename           address

Aziz            Casa
Amina           Rabat
Said            Agadir
Fatima          Tanger
Ahmed           Casa
Ali             Rabat
Ahmed           Casa
```

```
select enum, ename, address
from employee
where ename like '%e_';
```

résultat :

```
enum    ename           address

E2      Ahmed           Casa
E8      Ahmed           Casa
```

6.2 Clause BETWEEN

Permet de désigner un intervalle de valeurs. « *Salaire et nom des employées ayant un salaire compris en 7000 et 8000* »

```
select salary, ename
from employee
where salary between 7000 and 8000;
```

résultat :

```
    salary ename

   7500.00 Amine
   8000.00 Amina
   7000.00 Fatima
   8000.00 Ali
```

Clause `IN`

Dans la clause `where`, `in` permet de tester l'appartenance à un ensemble de valeurs.

```
select enum, ename
from employee
where enum in ('E5', 'E7', 'E2');
```

résultat :

```
enum ename

E2   Ahmed
E7   Amine
E5   Amina
```

L'ensemble de `in` peut être défini (en intension donc) comme résultat de `select`.

6.3 Bloc SELECT imbriqué

a) « *Quels sont le numéro et le nom des employés travaillant dans un département situé au 3e étage*? »

Deux blocs.

```
select enum, ename
from employee
where dept in (select dnum
          from department
              where floor = 3);
```

résultat :

```
enum ename

E7   Amine
E3   Fatima
E8   Ahmed
```

b) « *Quels sont le numéro et le nom des employés travaillant dans un département dirigé par un manager habitant* '`Tanger`'? »

3 blocs

```
select enum, ename
from employee
where dept in (select dnum
          from department
          where mgr in ( select enum
                    from employee
                    where address='Tanger'));
```

résultat :

```
enum ename

E7   Amine
E3   Fatima
```

C'est cette forme qui a donné son nom "*Structured*" au langage SQL. En effet, cette forme reflète la lecture de la requête associée. Mais cette forme n'est possible que si les attributs du résultat final proviennent d'une seule table (celle de `from` du premier `select`). Autrement, il faut utiliser la forme algébrique de la jointure. cf. **3.d)** ci-dessus.

c) « *Quels sont le numéro et le nom des employés ayant un salaire supérieur à celui de leur chef* »

```
select enum, ename
from employee x
where dept in (select dnum
          from department
```

```
         where mgr in ( select enum
                  from employee y
                  where x.salary > y.salary));
```

résultat :

```
enum ename

E7   Amine
E6   Aziz
```

d) Usage mixte de la forme algébrique (JOIN) et de bloc SELECT

Même requête que précédemment avec le nom de département aussi. Les deux tables sont nécessaires dans la première clause `from`.

```
select enum, ename, dname
from employee x, department d
where d.dnum = x.dept
and mgr in ( select enum
        from employee y
        where x.salary > y.salary);
```

résultat :

```
enum ename        dname

E7   Amine        Alimentation
E6   Aziz         Jouets
```

d-bis) La même requête en notation avec la clause `join`.

```
select enum, ename, dname
from employee x join department d on d.dnum = x.dept
where d.mgr in ( select enum
               from employee y
               where x.salary > y.salary)
```

Exercice: Réécrire la requête c) précédente (sans `dname`) de plusieurs manières différentes en utilisant la forme algébrique avec JOIN.

Réponse:

```
SELECT enum, ename
FROM employee x JOIN department ON dept = dnum
WHERE mgr IN ( SELECT enum
               FROM employee y
               WHERE x.salary > y.salary)
```

ou

```
SELECT enum, ename
FROM employee x
WHERE dept IN (SELECT dnum
               FROM department JOIN employee y ON mgr = enum
               WHERE x.salary > y.salary)
```

ou

```
SELECT x.enum, x.ename
FROM employee x JOIN (department JOIN employee y ON  x.dept = dnum)
     ON mgr = y.enum
WHERE x.salary > y.salary
```

sinon la forme classique,

```
SELECT x.enum, x.ename
FROM employee x, department, employee y
WHERE x.dept = dnum  AND mgr = y.enum
         AND  x.salary > y.salary
```

e) EXISTS (cf IN ci-dessus)

Même requête que **a)** ci-desssus, paraphrasée : « *Quels sont le numéro et le nom des employés tel que il existe un département ayant le même numéro que celui où l'employé travaille et situé au 3e étage*? »

```
select enum, ename
from employee e
where exists (select *
              from department d
              where d.dnum = e.dept
                    and d.floor = 3);
```

résultat :

```
enum ename

E7   Amine
E3   Fatima
E8   Ahmed
```

Noter l'ajout de la condition de jointure (`d.dnum=e.dept`) dans la clause `where`.

Les alias *e* et *d* ne sont pas nécessaires ici, mais sont là pour la lisibilité.

En fait, EXISTS existe pour être utilisée négativement, avec NOT EXISTS.

Usage de NOT EXISTS

d) « Quels sont les départements où ne travaille aucun employé? »

```
select dnum, dname
from department d
where not exists (select * from employee e
            where e.dept = d.dnum);
```

résultat :

```
dnum dname

D5   Bazar
```

On exprime donc ainsi, l'opérateur algébrique *différence* entre deux ensembles.

e) Autre exemple : « *numéro de dept qui ne vend pas P2* ».

```
select dnum
from department d
where not exists (select pnum
```

```
                from sell s
                where d.dnum=s.dnum and pnum ='P2')
```

résultat :

```
 dnum
 D2
 D3
 D4
```

f) `not exists` permet aussi d'exprimer l'opérateur algèbrique *division*.

« Le dept qui vend tous les produits »

```
select dnum
from department d
where not exists (select *
                  from product p
                  where not exists (select *
                                    from sell s
                                    where d.dnum = s.dnum
                                    and p.pnum=s.pnum));
```

résultat :

```
dnum

D5
```

En paraphrasant légèrement : « *Quels sont les départements tel que il n'existe pas un produit qu'ils ne vendent pas* »

g) autre méthode

On peut utiliser la fonction *count*() pour tester l'existance. En effet, dans le cas précédent, le nombre de produits vendus par D5 et égale au nombre total de produits.

```
 select dnum from department
 where (select count(sell.pnum)
        from sell
        where sell.dnum = department.dnum)
     = (select count(product.pnum)
        from product)
```

résultat :

```
dnum

D5
```

h) Jointure externe *outer join*

On voudrait le numéro et nom de ***tous*** les départements avec le numéro et le nom des employés qui y travaillent

```
select  dnum, dname, enum, ename
from  department, employee
where dept = dnum
order by dnum
```

Résultat :

```
+------+--------------+------+--------+
| dnum | dname        | enum | ename  |
+------+--------------+------+--------+
| D1   | Jouets       | E2   | Ahmed  |
| D1   | Jouets       | E6   | Aziz   |
| D1   | Jouets       | E1   | Ali    |
| D2   | Alimentation | E7   | Amine  |
| D2   | Alimentation | E3   | Fatima |
| D3   | Livres       | E5   | Amina  |
| D3   | Livres       | E4   | Said   |
| D4   | HiFi         | E8   | Ahmed  |
+------+--------------+------+--------+
```

Ici, on n'a pas le dértement 'D5' où personne ne travaille. Or il devrait apparaître dans la réponse (tous les départements) avec des valeurs indéfinies pour les autres champs. C'est ici qu'on utilise la jointure externe (ou outer join). Elle s'exprime comme ceci :

```
select  dnum, dname, enum, ename
from  department LEFT OUTER JOIN employee ON dept = dnum
order by dnum
```

Résultat :

```
+------+--------------+------+--------+
| dnum | dname        | enum | ename  |
+------+--------------+------+--------+
| D1   | Jouets       | E6   | Aziz   |
| D1   | Jouets       | E2   | Ahmed  |
| D1   | Jouets       | E1   | Ali    |
| D2   | Alimentation | E7   | Amine  |
| D2   | Alimentation | E3   | Fatima |
| D3   | Livres       | E5   | Amina  |
| D3   | Livres       | E4   | Said   |
| D4   | HiFi         | E8   | Ahmed  |
| D5   | Bazar        | NULL | NULL   |
+------+--------------+------+--------
```

Noter que cette fois-ci, le département D5 apparaît, avec des valeurs NULL pour les champs concernant un employé.

La forme `LEFT OUTER JOIN` exprime une jointure externe où l'on prend toute les lignes de la table de gauche (`LEFT`) de la clause from, même celles qui ne se comparent pas.

La même forme de requête, avec `RIGHT OUTER JOIN`, considère une jointure externe mais avec la table de droite (*employee* ici)

```
select  dnum, dname, enum, ename
from  department RIGHT OUTER JOIN employee ON dept = dnum
order by dnum
```

```
+------+--------------+------+--------+
| dnum | dname        | enum | ename  |
+------+--------------+------+--------+
| D1   | Jouets       | E2   | Ahmed  |
| D1   | Jouets       | E6   | Aziz   |
| D1   | Jouets       | E1   | Ali    |
| D2   | Alimentation | E7   | Amine  |
| D2   | Alimentation | E3   | Fatima |
| D3   | Livres       | E5   | Amina  |
| D3   | Livres       | E4   | Said   |
| D4   | HiFi         | E8   | Ahmed  |
+------+--------------+------+--------+
```

Comme tous les employés travaillent dans un département, toutes les lignes de la table *employee* se comparent, et la jointure externe donne le même résultat que la jointure normale dans ce cas.

NB. Le jointure externe n'est pas commutative. *R* `RIGHT OUTER JOIN` *S*, n'est pas équivalent à *S* `RIGHT OUTER JOIN` *R*.

Par contre *R* RIGHT OUTER JOIN *S*, est équivalent à *S* LEFT OUTER JOIN *R*. Constatez-le.

```
mysql> select enum, ename, dname
from employee LEFT JOIN department ON dept = dnum;
+------+--------+--------------+
| enum | ename  | dname        |
+------+--------+--------------+
| E7   | Amine  | Alimentation |
| E6   | Aziz   | Jouets       |
| E5   | Amina  | Livres       |
| E4   | Said   | Livres       |
| E3   | Fatima | Alimentation |
| E2   | Ahmed  | Jouets       |
| E1   | Ali    | Jouets       |
| E8   | Ahmed  | HiFi         |
+------+--------+--------------+
8 rows in set (0.00 sec)
```

Exercices :

Exprimer cette jointure externe sans utilser LEFT JOIN. Utiliser des relations temporaires.

Exprimer la requête "*numéro de produit dont le poids s'écarte le plus de la moyenne*".

Exprimer la requête 6.3 b), en utilisant la forme algébrique.