***Date de dernière modification** : 30/6/2015*

Annexe **8**

Le langage SQL-DML (2)

Ce chapitre étudie le problème de la génération de tableaux de données et propose un jeu d'exercices le plus souvent accompagnés de leur solution.

A8.1 GÉNÉRATION DE TABLEAUX

Nous abordons ici un problème classique dont la solution en SQL requiert un peu de subtilité.

A8.1.1 Le problème

Considérons les tableaux de la figure A8.1, qui contiennent les mêmes données, mais sous des présentations différentes.

Pro	I1	I2	I3	I4	I5
P1	2	0	0	0	3
P2	0	4	0	5	0
P3	5	8	1	0	4
P4	0	0	8	0	2

Ing	P1	P2	P3	P4
I1	2	0	5	0
I2	0	4	8	0
I3	0	0	1	8
I4	0	5	0	0
I5	3	0	4	2

Figure A8.1 - Deux tableaux inverses

Le tableau de gauche indique, pour chacun des produits **P1** à **P4** la quantité d'ingrédients **I1** à **I5** nécessaire à sa fabrication. Le tableau de droite est son inverse (on dit sa *transposée*) : il indique pour chaque ingrédient les produits qui l'utilisent et en quelle quantité.

Nous nous posons la question de la production de ces tableaux à partir d'une même collection de données.

A8.1.2 Les données sources

La représentation directe d'un de ces tableaux sous la forme d'une table SQL serait une très mauvaise idée pour plusieurs raisons. D'abord, elle privilégierait un de ces tableaux et ne serait donc pas neutre vis à vis des autres représentations qu'on pourrait en donner (l'autre tableau par exemple). Ensuite, elle limiterait le nombre d'ingrédients, puisque ceux-ci apparaissent explicitement sous la forme de nom de colonnes.

La solution la plus générale est celle qui consiste à représenter chaque cellule du tableau par une ligne qui précise le *produit*, l'*ingrédient* et la *quantité*. On peut ainsi traiter le problème quel que soit le nombre de produits et le nombre de colonnes. La construction de la table est proposée à la figure A8.2, suivie de son contenu. L'identifiant de la table est (Pro,Ing) mais n'est pas déclaré. Hypothèse importante : les données sont complètes. Si un ingrédient n'est pas utilisé dans un produit, il y intervient néanmoins mais avec une quantité = 0.

```
create table T(Pro char(3) not null,
               Ing char(3) not null,
               Q integer);

insert into T values('P1','I1',2),('P1','I2',0),('P1','I3',0),
                    ('P1','I4',0),('P1','I5',3),
                    ('P2','I1',0),('P2','I2',4),('P2','I3',0),
                    ('P2','I4',5),('P2','I5',0),
                    ('P3','I1',5),('P3','I2',8),('P3','I3',1),
                    ('P3','I4',0),('P3','I5',4),
                    ('P4','I1',0),('P4','I2',0),('P4','I3',8),
                    ('P4','I4',0),('P4','I5',2);
```

Figure A8.2 - Représentation générale des données des tableaux

Pro	**Ing**	**Q**
P1	I1	2
P1	I2	0
P1	I3	0
P1	I4	0
P1	I5	3
P2	I1	0
P2	I2	4
P2	I3	0
P2	I4	5

```
| P2  | I5  | 0 |
| P3  | I1  | 5 |
| P3  | I2  | 8 |
| P3  | I3  | 1 |
| P3  | I4  | 0 |
| P3  | I5  | 4 |
| P4  | I1  | 0 |
| P4  | I2  | 0 |
| P4  | I3  | 8 |
| P4  | I4  | 0 |
| P4  | I5  | 2 |
+-----+-----+---+
```

A8.1.3 Génération des tableaux

Le problème posé s'exprime désormais comme ceci : comment extraire les données pour les présenter sous la forme du tableau de gauche ? La dérivation du tableau de droite est alors toute simple, puisqu'elle est symétrique par rapport au premier problème.

Le script A8.3 produit le tableau de gauche via une **auto-jointure multiple** articulée autour d'une valeur commune de **Pro**. Chacune des cinq instances[1] de **T** apporte la quantité d'un des cinq ingrédients.

```
select T1.Pro,
       T1.Q as "I1",
       T2.Q as "I2",
       T3.Q as "I3",
       T4.Q as "I4",
       T5.Q as "I5"
from   T T1,T T2,T T3,T T4,T T5
where                      T1.Ing = 'I1'
and    T2.Pro = T1.Pro and T2.Ing = 'I2'
and    T3.Pro = T1.Pro and T3.Ing = 'I3'
and    T4.Pro = T1.Pro and T4.Ing = 'I4'
and    T5.Pro = T1.Pro and T5.Ing = 'I5';
```

Figure A8.3 - Génération du tableau de gauche - Première technique

Le script A8.4 produit le tableau de gauche selon une seconde technique, qui mérite quelques mots d'explication.

1. Qui se prononcent, rappelons-le, "sink instence" et non ""sink-z-instence" comme on l'entend trop souvent.

```
select Pro,
       max(case when Ing = 'I1' then Q end) as "I1",
       max(case when Ing = 'I2' then Q end) as "I2",
       max(case when Ing = 'I3' then Q end) as "I3",
       max(case when Ing = 'I4' then Q end) as "I4",
       max(case when Ing = 'I5' then Q end) as "I5"
from   T
group by Pro;
```

Figure A8.4 - Génération du tableau de gauche - Deuxième technique

On observe d'abord qu'elle n'exige pas de jointure mais qu'elle exploite le regroupement des lignes selon la valeur de **Pro**. Chaque groupe est constitué des lignes des ingrédients intervenant dans un produit. Chacune de ces lignes va donner la contribution d'un ingrédient et reste évidemment muette pour les autres. On pourrait visualiser les lignes d'un groupe comme un sous-tableau carré dont seule la diagonale NO-SE est garnie. Le tableau ci-dessous montre le contenu du groupe Pro = p3.

```
+-----+----+----+----+----+----+
| Pro | I1 | I2 | I3 | I4 | I5 |
+-----+----+----+----+----+----+
| ..  | .  | .  | .  | .  | .  |
| P3  | 5  |    |    |    |    |
| P3  |    | 8  |    |    |    |
| P3  |    |    | 1  |    |    |
| P3  |    |    |    | 0  |    |
| P3  |    |    |    |    | 4  |
| ..  | .  | .  | .  | .  | .  |
+-----+----+----+----+----+----+
```

La valeur de **Q** pour chaque ingrédient de **p3** peut se calculer comme la valeur maximum ou la somme de l'ensemble (qui n'est qu'un singleton) des valeurs de cet ingrédient. Pourquoi une fonction agrégative alors qu'il n'y a qu'une seule valeur par groupe ? Tout simplement par respect de la syntaxe de la clause **`select`** des requêtes de groupement. Le script A8.5 produit le tableau de droite par la seconde technique[2].

```
select Ing,
       max(case when Pro = 'P1' then Q end) as "P1",
       max(case when Pro = 'P2' then Q end) as "P2",
       max(case when Pro = 'P3' then Q end) as "P3",
       max(case when Pro = 'P4' then Q end) as "P4"
from   T
group by Ing;
```

Figure A8.5 - Génération du tableau de droite

2. On pourrait doter les structures `case-end` d'une branche *faux* : `else 0`.

A8.1.4 Limite de ces techniques

Le processus de génération des tableaux présente une particularité peu courante : des **valeurs** de la table source sont transformées en **colonnes**. Dès lors, il suffirait d'insérer une nouvelle ligne ou d'en modifier ou supprimer une autre pour que la forme du tableau, et non pas seulement son contenu, soit différent.

A8.1.5 Hypothèses des données manquantes

Au lieu d'indiquer une quantité 0 pour les ingrédients absents, il serait certainement plus raisonnable d'ignorer ces ingrédients dans les données sources. Le contenu de **T** serait dès lors le suivant :

Pro	**Ing**	**Q**
P1	I1	2
P1	I5	3
P2	I2	4
P2	I4	5
P3	I1	5
P3	I2	8
P3	I3	1
P3	I5	4
P4	I3	8
P4	I5	2

En rejouant les requêtes élaborées ci-dessus, on tire deux observations intéressantes :

– La technique des auto-jointures multiples ne fonctionne pas. En effet, on sait que si un des membres d'une jointure est absent, le résultat l'est aussi. Les produits qui n'utilisent pas tous les ingrédients disparaissent donc du résultat. Une auto-**jointure externe multiple** pourrait résoudre la question mais au risque d'affecter la santé mentale du programmeur, en particulier lorsque le nombre total des ingrédients est important (100 ingrédients, utilisés ou non, impliquerait une auto-jointure externe composée de 100 fois la table T).

– Selon la forme de la structure **`case-end`**, les ingrédients manquant apparaissent avec une quantité *null* (tableau ci-dessous) ou une quantité 0 (tableau de gauche).

Pro	I1	I2	I3	I4	I5
P1	2	--	--	--	3
P2	--	4	--	5	--
P3	5	8	1	--	4
P4	--	--	8	--	2

On adoptera donc la forme

```
case when Ing = 'I1' then Q end) as "I1"
```

ou

```
case when Ing = 'I1' then Q else 0 end) as "I1"
```

selon le résultat désiré.

A8.1.6 Génération de générateurs de tableaux

Puisque le contenu de la table source peut évoluer rapidement, il est nécessaire de rédiger chaque requête juste avant de l'exécuter, ce qui exige évidemment de l'assembler de manière automatique et non plus manuelle. Ciblons la production du tableau de gauche à l'aide de la seconde technique. On observe que la requête A8.4 est constituée de trois parties :

– une chaîne constante : `'select Pro,'`
– une suite de sous-chaînes séparées par des virgules, du type :
```
max(case when Ing = 'I1' then Q end) as "I1"
```
– une chaîne constante : `' from T group by Pro'`.

La suite de sous-chaînes sera idéalement produite par la fonction `group_concat`. Le script de génération A8.6 crée la requête de génération dans la variable **Query**, puis en exécute le contenu. Son adaptation au tableau de droite est immédiate.

```
extract Query = select
                    'select Pro,'
                   ||group_concat('max(case
                                       when Ing = '''||Ing||'''
                                       then Q end) as "'
                   ||Ing||'"',',')
                   ||' from T group by Pro'
                from (select distinct Ing from T order by Ing);
$Query$;
```

Figure A8.6 - Création et exécution dynamique de requêtes de génération de tableaux

Les scripts de cette section sont disponibles dans le fichier **Generation de tableaux.sql**.

A8.2 LES SCRIPTS SQLfast DU CHAPITRE 8

Le script **DML2-requetes.sql** reprend en un unique fichier les requêtes du chapitre 8, adaptées à SQLite 3. La rédaction des scripts relatifs aux autres exercices de cette annexe est laissée au bons soins du lecteur. La bonne exécution de ces scripts réclame l'existence préalable de la base de données CLICOM.db.

A8.3 EXERCICES DU CHAPITRE 8

a) Énoncés de type 2

A8.1 Calculer le montant de chaque détail de commande du client 'C400'.

```
select M.NCOM, P.NPRO, QCOM*PRIX as MONTANT
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM and D.NPRO = P.NPRO and NCLI = 'C400';
```

A8.2 Calculer le montant commandé des produits en sapin.

```
select sum(QCOM*PRIX) as MONTANT
from   DETAIL D,PRODUIT P
where  D.NPRO = P.NPRO
and    P.LIBELLE like '%SAPIN%';
```

b) Énoncés de type 3

A8.3 Afficher le total et la moyenne des comptes des clients, ainsi que le nombre de clients, selon chacune des classifications suivantes :

- par catégorie,

```
select CAT, sum(COMPTE), avg(COMPTE), count(*)
from   CLIENT group  by CAT;
```

- par localité,

- par catégorie dans chaque localité.

```
select CAT, sum(COMPTE), avg(COMPTE), count(*)
from   CLIENT group  by LOCALITE, CAT;
```

A8.4 Combien y a-t-il de commandes spécifiant un (ou plusieurs) produit(s) en acier (on proposera une solution différente de celle de la section 7.6.3) ?

```
select count(distinct M.NCOM)
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
and    LIBELLE like '%ACIER%';
     ou
select count(*)
from   COMMANDE M
where  NCOM in (select NCOM
                from DETAIL D, PRODUIT P
                where  D.NPRO = P.NPRO
                and    LIBELLE like '%ACIER%');
```

A8.5 Créer une table et y ranger les données suivantes relatives aux détails de commande : numéro et date de la commande, quantité commandée, numéro et prix du produit, montant du détail.

```
create table DETAIL_COM((NCOM ..., DATECOM ..., ..., MONTANT ...);

insert into DETAIL_COM(NCOM, DATECOM, QCOM, NPRO, PRIX, MONTANT)
select M.NCOM, DATECOM, QCOM, P.NPRO, PRIX, QCOM*PRIX
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO;
```

A8.6 Annuler les comptes négatifs des clients de catégorie C1.

```
update CLIENT
set    COMPTE = 0
where  COMPTE < 0
and    CAT = 'C1';
```

A8.7 Compléter le fragment suivant de manière à former une requête valide
`select CAT, NPRO, sum(QCOM*PRIX) from ...`

```
select CAT, NPRO, sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by CAT, P.NPRO;
```

A8.8 En considérant le schéma 8.5, écrire les requêtes SQL qui donnent :

- les matières premières (produits qui n'ont pas de composants);

```
select NPRO
from   PRODUIT
where  NPRO not in (select COMPOSE from COMPOSITION);
```

- les produits finis (qui n'entrent dans la composition d'aucun autre);

```
select NPRO
from   PRODUIT
where  NPRO not in (select COMPOSANT from COMPOSITION);
```

- les produits semi-finis (tous les autres);

```
select NPRO
from   PRODUIT
where  NPRO in (select COMPOSE   from COMPOSITION)
```

```
and    NPRO in (select COMPOSANT from COMPOSITION);
```

- le prix et poids unitaires d'un produit fini ou semi-fini dont tous les composants ont un poids et un prix unitaires.

```
select PH.NPRO, sum(QTE*PB.PRIX_U), sum(QTE*PB.POIDS_U)
from   PRODUIT PH, COMPOSITION C, PRODUIT PB
where  PH.NPRO = C.COMPOSE
and    C.COMPOSANT = PB.NPRO
and    not exists (select *
                   from  COMPOSITION CC, PRODUIT BB
                   where CC.COMPOSANT = BB.NPRO
                   and   CC.COMPOSE = PH.NPRO
                   and   (BB.PRIX_U is null
                          or BB.POIDS_U is null))
group by PH.NPRO;
```

A8.9 Quels sont les personnes qui ont le même responsable que p4 ?

```
select NPERS, NOM
from   PERSONNE
where  RESPONSABLE in (select RESPONSABLE from PERSONNE
                       where NPERS = 'p4';
```

c) Énoncés de type 4

A8.10 Calculer le montant de chaque commande.

A8.11 Calculer le montant dû par chaque client. Dans ce calcul, on ne prend en compte que le montant des commandes. Attention aux clients qui n'ont pas passé de commandes.

```
select NCLI, sum(QCOM*PRIX)
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by NCLI
union
select NCLI, 0
from   CLIENT C
where  not exists (select * from COMMANDE where NCLI = C.NCLI);
```

A8.12 Calculer le montant dû par les clients de chaque localité. Dans ce calcul, on ne prend en compte que le montant des commandes. Attention aux localités dans lesquelles aucun client n'a passé de commandes.

```
select LOCALITE, sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
```

```
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE
union
select distinct LOCALITE, 0
from   CLIENT
where  LOCALITE not in (select LOCALITE
                        from   CLIENT C
                        where  exists (select *
                                       from   COMMANDE
                                       where  NCLI = C.NCLI));
```

A8.13 Calculer, par jour, le total des montants des commandes.

```
select DATECOM, sum(QCOM*PRIX)
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by DATECOM;
```

A8.14 On suppose qu'on n'a pas trouvé utile d'imposer un identifiant primaire sur la table PRODUIT. Il se peut donc que plusieurs lignes aient même valeur de la colonne NPRO, ce qui viole le principe d'unicité des valeurs de cette colonne.

- Ecrire une requête qui recherche les valeurs de NPRO présentes en plus d'un exemplaire.

```
select NPRO, count(*)
from   PRODUIT
group by NPRO
having count(*) > 1
```

- Écrire une requête qui indique combien de valeurs de NPRO sont présentes en plus d'un exemplaire.

```
create view OCCURRENCES(NPRO,NOMBRE) as
   select NPRO, count(*)
   from   PRODUIT
   group by NPRO;

select count(*)
from   OCCURRENCES
where  NOMBRE > 1;
```

- Écrire une requête qui indique combien de lignes comportent une erreur d'unicité de NPRO.

- Écrire une requête qui, pour chaque valeur de NPRO présente dans la table, indique dans combien de lignes cette valeur est présente.

- Écrire une requête qui, pour chaque valeur de NPRO qui n'est pas unique, indique dans combien de lignes cette valeur est présente.

- Écrire une suite de requêtes qui crée une table contenant les numéros de NPRO qui ne sont pas uniques.

A8.15 Afficher pour chaque localité, les libellés des produits qui y sont commandés.

```
select LOCALITE, LIBELLE
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE, LIBELLE
order by LOCALITE, LIBELLE
```

A8.16 Afficher par localité, et pour chaque catégorie dans celle-ci, le total des montants des commandes.

```
select LOCALITE, CAT, sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE, CAT
   union
select LOCALITE, CAT, 0
from   CLIENT C
group by LOCALITE, CAT
having not exists (select *
                   from COMMANDE
                   where  NCLI in (select NCLI
                                   from   CLIENT
                                   where  LOCALITE = C.LOCALITE
                                   and    CAT = C.CAT));
```

A8.17 Indiquer, pour chaque localité, les catégories de clients qui n'y sont pas représentées

Suggestion. Construire l'ensemble de tous les couples (LOCALITE, CAT) possibles et en retirer ceux qui existent dans la base. Attention aux valeurs `null`, qui ne doivent pas être prises en compte.

```
select distinct L.LOCALITE, C.CAT
from   CLIENT L, CLIENT C
where  CAT is not null
and not exists (select *
                from CLIENT
                where LOCALITE = L.LOCALITE and CAT = C.CAT);
```

A8.18 Produire (à l'écran) une table de couples <X,Y> de clients tels que X et Y habitent dans la même localité. On évitera de renseigner <X,X>, mais aussi <Y,X> si <X,Y> est déjà repris.

Suggestion. Auto-jointure de CLIENT. On évitera les couples inverse en imposant un ordre sur les valeurs de NPRO (p.ex. X < Y).

```
select C1.LOCALITE, C1.NCLI, C2.NCLI
from   CLIENT C1, CLIENT C2
where  C1.NCLI < C2.NCLI
and    C1.LOCALITE = C2.LOCALITE
order by C1.LOCALITE, C1.NCLI, C2.NCLI
```

A8.19 En considérant le schéma 8.5, écrire une requête de mise à jour qui complète les prix et poids unitaires des produits finis ou semi-finis. Pour simplifier la procédure, on admet que cette requête soit exécutée autant de fois que nécessaire pour que tous les produits soient complétés.

```
update PRODUIT PH
set PRIX_U = (select  sum(QTE*PB.PRIX_U)
              from    COMPOSITION C, PRODUIT PB
              where   PH.NPRO = C.COMPOSE
              and     C.COMPOSANT = PB.NPRO)
where  PRIX_U is null
and    not exists (select * from  COMPOSITION CC, PRODUIT BB
                   where CC.COMPOSANT = BB.NPRO
                   and   CC.COMPOSE = PH.NPRO
                   and   BB.PRIX_U is null);
```

A8.20 En considérant le schéma de la figure 8.8, calculer pour chaque ville, le prix moyen de chaque produit.

```
select VILLE, PRODUIT, avg(PRIX)
from   VENTE V, LOCALISATION L
where  V.CHAINE = L.CHAINE
group by VILLE, PRODUIT;
```

A8.21 Afficher pour chaque client, le nombre de commandes, le nombre de produits commandés et le nombre de détails. On se limite aux clients qui ont passé au moins une commande.

Suggestion : il s'agit d'une requête basée sur des groupements multi-niveaux.

```
select  C.NCLI, count(distinct NCOM) as Commandes,
        count(distinct NPRO) as Produits,
        count(NPRO) as Details
from    CLIENT C, COMMANDE M, DETAIL D
where   C.NCLI = M.NCLI and M.NCOM = D.NCOM
group by C.NCLI
```

```
select  C.NCLI, count(distinct NCOM) as Commandes,
        count(distinct NPRO) as Produits,
        count(NPRO) as Details
from    CLIENT C, COMMANDE M, DETAIL D
where   C.NCLI = M.NCLI and M.NCOM = D.NCOM
group by C.NCLI;
```

A8.22 Afficher, pour chaque localité et pour chaque catégorie, (1) le nombre de commandes passées par les clients de cette localité et de cette catégorie, (2) le montant total de ces commandes.

```
select LOCALITE, CAT, count(distinct NCOM), sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE, CAT
```

A8.23 On considères trois tables `T1`, `T2` et `T3`, chacune possédant un unique attribut `A` identifiant. Calculer l'ensemble des valeurs de `A` qui

- sont présentes dans les trois tables,

```
select A from T1
intersect
select A from T2
intersect
select A from T3;
```

ou

```
select A from T1
where A in (select A from T2)
and A in select A from T3);
```

- sont présentes dans `T1` seulement,

```
select A from T1
except
select A from T2
except
select A from T3;
```

ou

```
select A from T1
where A not in (select A from T2)
and A not in select A from T3);
```

- sont présentes dans `T1` et `T2` mais pas dans `T3`,

```
select A from T1
intersect
select A from T2
except
select A from T3;
```

ou

```
select A from T1
where A in (select A from T2)
and A not in select A from T3);
```

- sont présentes dans une seule table,

```
create view T123(A ...)
as select A from T1
union all select A from T2
union all select A from T2;

select A from T123 group by A having count(*) = 1;
```

- sont présentes dans deux tables seulement.

```
select A from T123 group by A having count(*) = 3;
```

d) Énoncés de type 5

A8.24 Calculer le nombre moyen de produits par commande. De même : le nombre moyen de commandes par client, par localité ou par catégorie.

Remarque. Il n'est pas possible de demander directement la moyenne d'une somme. *Suggestion* : construction et interrogation d'une vue ou calcul de la moyenne dans le `from`.

```
create view NOMBRE(NCOM,NBRE) as
select NCOM, count(*)
from   DETAIL
group by NCOM;

select avg(NBRE)
from NOMBRE;
```

A8.25 Quel est, pour chaque localité, le nombre moyen de commandes par client.

```
create view NOMBRE(NCLI,NBRE) as
select C.NCLI, count(*)
from   CLIENT C, COMMANDE M
where  C.NCLI = M.NCLI
group by C.NCLI;

select LOCALITE, avg(NBRE)
from   CLIENT C, NOMBRE N
where  C.NCLI = N.NCLI
group by LOCALITE;
```

A8.26 Ecrire une requête SQL qui donne, pour chaque catégorie de produit, le nombre de produits qui ont été commandés le 23-12-2008.

```
select CAT, count(distinct NPRO)
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
AND    DATECOM = '23-DEC-2008'
group by CAT
```

A8.27 Donner pour chaque localité dans laquelle se trouve au moins un client de catégorie 'C1' la liste des produits en sapin qu'on y a commandés.

```
select LOCALITE, D.NPRO
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
and    LOCALITE in (select LOCALITE from CLIENT where CAT = 'C1')
and LIBELLE like '%SAPIN%'
group by LOCALITE, D.NPRO
```

A8.28 Donner pour chaque produit la liste des localités dans lesquelles ce produit est commandé en plus de 500 unités (= au total pour la localité).

```
select D.NPRO, LOCALITE, sum(QCOM)
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
group by D.NPRO, LOCALITE
having sum(QCOM) > 500
```

A8.29 Afficher, pour chaque localité, les produits qu'on y commande et qui sont aussi commandés dans au moins une autre localité.
Suggestion. Un produit est intéressant si le nombre de localités dans lesquelles il est commandé est supérieur ou égal à 2.

```
select LOCALITE, NPRO
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO in (select NPRO
                  from   CLIENT C, COMMANDE M, DETAIL D
                  where  C.NCLI = M.NCLI
                  and    M.NCOM = D.NCOM
                  group by NPRO
                  having count(distinct LOCALITE) > 1)
group by LOCALITE, D.NPRO
```

A8.30 Dans quelles localités peut-on trouver au moins un client qui a commandé tous les produits en sapin, et ce, pour un montant total, pour ces produits, dépassant 10.000 ?

A8.31 Calculer, pour chaque localité, le nombre de catégories distinctes.

```
select LOCALITE, count(distinct CAT)
from   CLIENT
where  CAT is not null
group by LOCALITE
```

A8.32 La section 8.13.1 suggère une comparaison entre un instantané et une vue de même définition. En effet, ces deux techniques pourraient être considérées comme équivalentes pour la formulation de requêtes. Établir une liste de critères de comparaison et l'appliquer aux deux techniques.

A8.33 Mettre à jour les comptes des clients en en déduisant le montant des commandes en cours.

Suggestion. *cfr* mise à jour des quantités en stock des produits; attention aux clients qui n'ont pas commandé.

```
update PRODUIT P
set QSTOCK = QSTOCK - (select sum(QCOM)
                       from   DETAIL
                       where  NPRO = P.NPRO)
where exists (select *
              from   DETAIL
              where  NPRO = P.NPRO)
```

A8.34 Mettre à jour les quantités en stock des produits en en déduisant les quantités commandées par les clients de catégorie B1 et C1.

```
update PRODUIT P
set QSTOCK = QSTOCK - (select sum(QCOM)
                       from   DETAIL D, COMMANDE M, CLIENT C
                       where  D.NPRO = P.NPRO
                       and    D.NCOM = M.NCOM
                       and    M.NCLI = C.NCLI
                       and    CAT in ('B1','C1'))
where exists (select *
              from   DETAIL D, COMMANDE M, CLIENT C
              where  D.NPRO = P.NPRO
              and    D.NCOM = M.NCOM
              and    M.NCLI = C.NCLI
              and    CAT in ('B1','C1'))
```

A8.35 On suppose qu'on dispose de deux tables PRODUIT1 et PRODUIT2 décrivant des produits de deux filiales distinctes, et de même structure que PRODUIT. Il est possible qu'un produit soit présent simultanément dans les deux filiales. Le même numéro de produit est repris alors dans les deux tables. On désire intégrer les deux dépôts. En conséquence, on fusionne les deux tables initiales en une troisième selon les règles suivantes :

- si un produit n'est présent que dans une seule filiale, sa ligne est reprise telle quelle,
- si un produit est présent dans les deux filiales, on ne le décrit qu'une seule fois. Son libellé est celui de PRODUIT1, son prix est le minimum des deux valeurs et sa quantité en stock est la somme des deux valeurs.

Ecrire les requêtes d'intégration.

```
create table PRODUIT1
(NPRO char(10) not null primary key,
 LIBELLE char(30) not null,
 PRIX decimal(5,2) not null,
 QSTOCK decimal(6) not null);

insert into PRODUIT1 values ('A001','Farine', 0.8, 65);
insert into PRODUIT1 values ('B001','Sel', 0.5, 120);
insert into PRODUIT1 values ('A003','Huile olive', 5.2, 9);

create table PRODUIT2
(NPRO char(10) not null primary key,
 LIBELLE char(30) not null,
 PRIX decimal(5,2) not null,
 QSTOCK decimal(6) not null);

insert into PRODUIT2 values ('A002','Sucre', 1.2, 45);
insert into PRODUIT2 values ('B001','Sel iodé', 0.45, 35);
insert into PRODUIT2 values ('A003','Huile', 5.6, 22);

create table PRODUIT3
(NPRO char(10) not null primary key,
 LIBELLE char(30) not null,
 PRIX decimal(5,2) not null,
 QSTOCK decimal(6) not null);

insert into PRODUIT3
select *
from PRODUIT1 P1
where not exists (select * from PRODUIT2 where NPRO = P1.NPRO);

insert into PRODUIT3
select *
from PRODUIT2 P2
where not exists (select * from PRODUIT1 where NPRO = P2.NPRO);

insert into PRODUIT3
select P1.NPRO, P1.LIBELLE, P1.PRIX, P1.QSTOCK + P2.QSTOCK
from   PRODUIT1 P1, PRODUIT2 P2
```

```
where  P1.NPRO = P2.NPRO
and    P1.PRIX <= P2.PRIX;

insert into PRODUIT3
select P1.NPRO, P1.LIBELLE, P2.PRIX, P1.QSTOCK + P2.QSTOCK
from   PRODUIT1 P1, PRODUIT2 P2
where  P1.NPRO = P2.NPRO
and    P1.PRIX > P2.PRIX;
```

A8.36 Soit une jointure de deux ou plusieurs tables, dont la clause `select` spécifie certaines colonnes. On désire que le résultat ne comporte pas de lignes en double. Indiquer dans quelles conditions le modifieur `distinct` est inutile.

Suggestion. `distinct` est inutile lorsque la clause `select` comprend tous les composants de l'identifiant de la table mentionnée dans la clause `from` ou (cas d'une jointure par exemple) construite dans les clauses `from-where-group-by`.

A8.37 On considère une base de données constituée des deux tables suivantes : R(A,B) et S(B,C). Evaluer l'ordre de grandeur du nombre de requêtes différentes qu'il est possible de formuler sur cette base de données.

Réponse : une infinité.

e) Énoncés de type 6

A8.38 La figure 12.27 (Voyages en train) représente un schéma qui est accompagné de suggestions de questions auxquelles une base de données traduisant ce schéma permettrait de répondre. On suppose que ce schéma est traduit en structure de tables. Exprimer chacune de ces questions sous la forme de requêtes SQL.

A8.39 On désire réaliser le couplage deux à deux de certaines localités où résident des clients. À cette fin, on construit des listes de couples candidats. On se limite cependant aux couples de localités dans lesquelles on commande au moins un même produit.

De manière à y voir plus clair, on construit d'abord une vue (`view` ou table temporaire) LP(LOCALITE,NPRO) qui indique que dans telle localité on commande tel produit.

```
create view LP(LOCALITE,NPRO) as
select distinct LOCALITE, NPRO
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI and M.NCOM = D.NCOM
```

On effectue ensuite une autojointure de LP sur la colonne NPRO. On obtient ainsi des couples (loc1, loc2) de valeurs de LOCALITE tels que loc1 et loc2 sont associées à (au moins) une même valeur de NPRO.

```
select LP1.LOCALITE, LP2.LOCALITE
from   LP LP1, LP LP2
where  LP1.NPRO = LP2.NPRO
```

Ces deux valeurs de LOCALITE doivent être différentes : on ajoute la condition LP1.LOCALITE <> LP2.LOCALITE. Pour éviter les associations symétriques (inutile de retenir (Poitiers,Lille) et (Lille,Poitiers)), on affine la condition précédente en : LP1.LOCALITE < LP2.LOCALITE.

```
select distinct LP1.LOCALITE, LP2.LOCALITE
from   LP LP1, LP LP2
where  LP1.NPRO = LP2.NPRO
and    LP1.LOCALITE < LP2.LOCALITE
```

Pour éviter la définition de la vue, on peut fusionner les deux requêtes :

```
select distinct C1.LOCALITE, C2.LOCALITE
from   CLIENT C1, COMMANDE M1, DETAIL D1,
       DETAIL D2, COMMANDE M2, CLIENT C2
where  C1.NCLI = M1.NCLI and M1.NCOM = D1.NCOM
and    D1.NPRO = D2.NPRO
and    D2.NCOM = M2.NCOM and M2.NCLI = C2.NCLI
and    C1.LOCALITE < C2.LOCALITE
```

A8.40 Même question que A8.39 ci-dessus, mais on se limite aux couples de localités dans lesquelles on achète exactement les mêmes produits.

Cette question est un peu plus complexe que la précédente. Nous allons d'abord répondre à une question plus large : produire les couples (loc1, loc2) de localités tels que tous les produits commandés dans la localité loc1 ont été aussi commandés dans la localité loc2. En réutilisant la vue LP, on peut répondre par la requête suivante, qui traduit la propriété "*il n'existe pas de valeur de NPRO dépendant de LP1.LOCALITE qui soit absente de celles qui dépendent de LP2.LOCALITE*" (on notera la condition `LP1.LOCALITE <> LP2.LOCALITE`) :

```
select distinct LP1.LOCALITE, LP2.LOCALITE
from   LP AS LP1, LP AS LP2
where  LP1.LOCALITE <> LP2.LOCALITE
and    not exists(select * from LP
                  where  LOCALITE = LP1.LOCALITE
                  and    NPRO not in (select NPRO from LP
                                      where  LOCALITE = LP2.LOCALITE));
```

Parenthèse. Observons que si deux localités loc1 et loc2 ont exactement le même ensemble de produits, elles apparaîtront naturellement sous la forme des deux couples (loc1, loc2) et (loc2, loc1). On pourrait donc à ce stade rechercher dans les couples du résultat ceux qui possède un inverse. Cet exercice, très facile, est laissé aux soins du lecteur. *Fin de parenthèse*.

La demande initiale exige en plus la condition symétrique, qui veut que tous les produits dépendant de LP2.LOCALITE dépendent aussi de LP1.LOCALITE (on notera la condition `LP1.LOCALITE < LP2.LOCALITE`) :

```
select distinct LP1.LOCALITE, LP2.LOCALITE
from   LP AS LP1, LP AS LP2
where  LP1.LOCALITE < LP2.LOCALITE
and    not exists(select * from LP
                  where  LP.LOCALITE = LP1.LOCALITE
                  and    NPRO not in (select NPRO from LP
                                      where  LOCALITE = LP2.LOCALITE))
and    not exists(select * from LP
                  where  LP.LOCALITE = LP2.LOCALITE
                  and    NPRO not in (select NPRO from LP
                                     where  LOCALITE = LP1.LOCALITE));
```

A8.41 Pour chaque produit, indiquer la ville dans laquelle la quantité totale commandée est minimale, et celle dans laquelle cette quantité est maximale.

f) Énoncé de type 7

A8.42 Rédiger un script (suite de requêtes) SQL qui réalise en fin de mois les opérations de gestion régies par les règles suivantes :

- pour chaque client, on calcule le total des montants des commandes du mois écoulé (= les commandes d'un mois et d'une année déterminés);
- si ce montant est supérieur à celui du mois précédent, on applique une réduction de 5%;
- ensuite, si le client est le seul de sa localité, on applique une réduction supplémentaire de 5%;
- on range dans une table les informations nécessaires à l'édition des factures du mois;
- on met à jour le compte des clients;
- on met à jour le niveau de stock (QSTOCK) des produits commandés;
- pour chaque produit dont le niveau de stock a été mis à jour, et dont le niveau est inférieur ou égal à 0, on prépare dans une table adéquate les informations de rapprovisionnement.