Chapitre 4 Solutions aux Exercices

Représentation UML du schéma *VentesPleinDeFoin* :

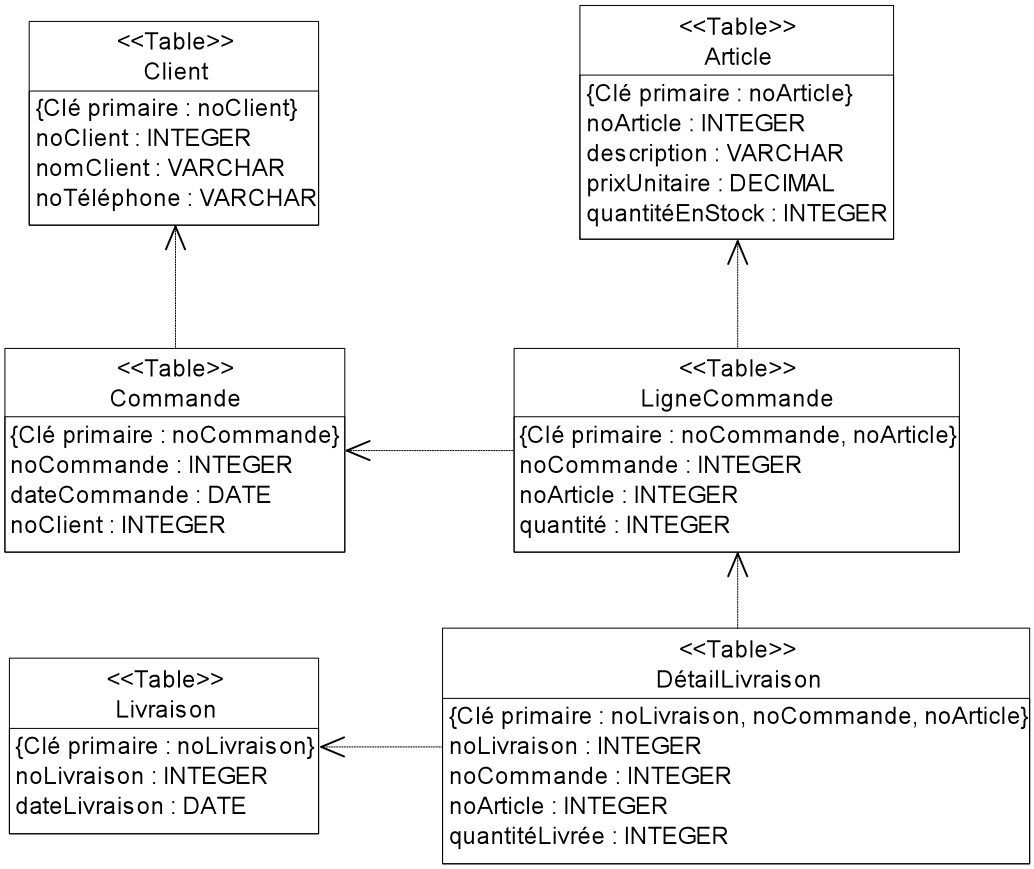

Schéma SQL *VentesPleinDeFoin* :

```
CREATE TABLE Client
(noClient         INTEGER           NOT NULL,
 nomClient        VARCHAR(20)       NOT NULL,
 noTéléphone      VARCHAR(15)       NOT NULL,
 PRIMARY KEY      (noClient)
)
CREATE TABLE Article
(noArticle        INTEGER           NOT NULL,
 description      VARCHAR(20),
 prixUnitaire     DECIMAL(10,2)     NOT NULL,
 quantitéEnStock  INTEGER           DEFAULT 0 NOT NULL
      CHECK (quantitéEnStock >= 0),
 PRIMARY KEY (noArticle))
CREATE TABLE Commande
(noCommande       INTEGER           NOT NULL,
 dateCommande     DATE              NOT NULL,
 noClient         INTEGER           NOT NULL,
 PRIMARY KEY      (noCommande),
 FOREIGN KEY      (noClient) REFERENCES Client
)
CREATE TABLE LigneCommande
(noCommande       INTEGER           NOT NULL,
 noArticle        INTEGER           NOT NULL,
 quantité         INTEGER           NOT NULL
      CHECK (quantité > 0),
 PRIMARY KEY (noCommande, noArticle),
 FOREIGN KEY (noCommande) REFERENCES Commande,
 FOREIGN KEY (noArticle) REFERENCES Article
)
CREATE TABLE Livraison
(noLivraison      INTEGER           NOT NULL,
 dateLivraison    DATE              NOT NULL,
 PRIMARY KEY (noLivraison)
)
CREATE TABLE DétailLivraison
(noLivraison      INTEGER           NOT NULL,
 noCommande       INTEGER           NOT NULL,
 noArticle        INTEGER           NOT NULL,
 quantitéLivrée   INTEGER           NOT NULL
      CHECK (quantitéLivrée > 0),
 PRIMARY KEY (noLivraison, noCommande, noArticle),
 FOREIGN KEY (noLivraison) REFERENCES Livraison,
 FOREIGN KEY (noCommande, noArticle) REFERENCES LigneCommande
)
```

1) Formulez en SQL les requêtes suivantes sur le schéma de la BD de la pépinière *PleinDeFoin* (N.B. *a*) à *m*) sont identiques à l'exercice 2 sur l'algèbre relationnelle). Cherchez les formulations équivalentes :

   a) Les *Clients* dont le *noTéléphone* = (999)999-9999

```
SELECT    *
FROM      Client
WHERE     noTéléphone = '(999)999-9999'
```

   b) Le *noCommande* et la *dateCommande* des *Commandes* du *Client* #10 dont le *noCommande* est supérieur à 5.

```
SELECT    noCommande, dateCommande
FROM      Commande
WHERE     noClient = 10 AND noCommande > 5
```

   c) Les *noArticle* et *description* des *Articles* dont le *prixUnitaire* est entre $10 et $20.

```
SELECT    noArticle, description
FROM      Article
WHERE     prixUnitaire BETWEEN 10 AND 20
```

```
SELECT    noArticle, description
FROM      Article
WHERE     prixUnitaire >= 10 AND prixUnitaire <= 20
```

   d) Le *noClient*, *noTéléphone* du *Client* et *noCommande* pour les *Commandes* faites le 4/06/2000.

```
SELECT    Client.noClient, noTéléphone, noCommande
FROM      Client, Commande
WHERE     Client.noCLient = Commande.noClient AND
          DateCommande = '4/06/2000'
```

   e) Les *noArticles* commandés au moins une fois par le *Client* #10 après le 01/06/2000.

```
SELECT    DISTINCT noArticle
FROM      Commande, LigneCommande
WHERE     Commande.noCommande = LigneCommande.noCommande AND
          noClient = 10 AND
          DateCommande > '1/06/2000'
```

Solution avec SELECT imbriqué :

```
SELECT    DISTINCT noArticle
FROM      LigneCommande
WHERE     noCommande IN
     (SELECT   noCommande
      FROM     Commande
      WHERE    noClient = 10 AND
          DateCommande > '1/06/2000')
```

f) Les *noLivraisons* correspondant aux *Commandes* faites par le *Client* #10.

```
SELECT    DISTINCT noLivraison
FROM      Commande C, DétailLivraison D
WHERE     C.noCommande = D.noCommande AND
          noClient = 10
```

Solution avec SELECT imbriqué :

```
SELECT    DISTINCT noLivraison
FROM      DétailLivraison
WHERE     noCommande IN
          (SELECT   noCommande
           FROM     Commande
           WHERE    noClient = 10)
```

g) Les *noCommandes* des *Commandes* qui ont été placées à la même date que la *Commande* #2.

```
SELECT    Commande.noCommande
FROM      Commande, Commande C2
WHERE     Commande.dateCommande = C2.dateCommande AND
          C2.noCommande = 2
```

h) Les *noLivraison* faites à la même date qu'une des *Commandes* correspondant à la *Livraison*.

```
SELECT    DISTINCT V.noLivraison
FROM      Commande C, DétailLivraison D, Livraison V
WHERE     C.noCommande = D.noCommande AND
          D.noLivraison = V.noLivraison AND
          C.dateCommande = V.dateLivraison
```

i) La liste des *noCommande* avec les *noLivraisons* associées incluant les *noCommandes* sans livraison.

```
SELECT    DISTINCT noCommande, noLivraison
FROM      Commande NATURAL LEFT OUTER JOIN DétailLivraison
```

Solution avec le dialecte Oracle

```
SELECT    DISTINCT C.noCommande, D.noLivraison
```

```
FROM      Commande C,DétailLivraison D
WHERE     C.noCommande = D.noCommande (+)
```

j) Les *noClient*, *nomClient* des *Clients* qui n'ont pas placé de *Commande* au mois de mars de l'année 2000.

```
SELECT    noClient, nomClient
FROM      Client
WHERE     NOT EXISTS
     (SELECT   *
      FROM     Commande
      WHERE    noClient = Client.noClient AND
               dateCommande BETWEEN '01/03/2000' AND '31/03/2000')
```

Solution avec MINUS (N.B. Oracle utilise MINUS plutôt que EXCEPT)

```
(SELECT   noClient, nomClient
 FROM     Client)
MINUS
(SELECT   noClient, nomClient
 FROM     Client, Commande
 WHERE    Commande. noClient = Client.noClient AND
          dateCommande BETWEEN '01/03/2000' AND '31/03/2000')
```

k) Les *noCommandes* qui ne contiennent pas l'*Article* # 10.

```
SELECT    noCommande
FROM      Commande
WHERE     NOT EXISTS
     (SELECT   *
      FROM     LigneCommande
      WHERE    Commande.noCommande = noCommande AND
               noArticle =10)
```

l) Les *noArticle* qui apparaissent dans toutes les *Commandes*.

```
     SELECT    noArticle
     FROM      Article
     WHERE     NOT EXISTS
          (SELECT   noCommande
           FROM     Commande
           WHERE NOT EXISTS
               (SELECT   *
                FROM     LigneCommande
                WHERE    noArticle = Article.noArticle AND
                         noCommande = Commande.noCommande))
```

ou :

```
SELECT    noArticle
FROM      Article
WHERE     NOT EXISTS
    ((SELECT   noCommande
      FROM     Commande
     )
       EXCEPT
     (SELECT   noCommande
      FROM     LigneCommande
      WHERE    noArticle = Article.noArticle
     )
    )
```

(N.B. Oracle utilise MINUS plutôt que EXCEPT)

ou en utilisant l'équivalence T_1 (X) ÷ T_2 (Y) = $\pi_{X-Y}(T_1)$ - $\pi_{X-Y}((\pi_{X-Y}(T_1) \times T_2)$ - $T_1)$:

```
(SELECT DISTINCT noArticle
FROM LigneCommande)
     EXCEPT
(SELECT DISTINCT noArticle FROM
 ((SELECT DISTICNT LigneCommande.noArticle,Commande.noCommande
  FROM LigneCommande, Commande)
       EXCEPT
 (SELECT DISTINCT noArticle,noCommande
  FROM LigneCommande)))
```

m) Les *noArticles* qui apparaissent dans toutes les *Commandes* du *Client* #10.

```
SELECT    noArticle
FROM      Article
WHERE     NOT EXISTS
     (SELECT   noCommande
      FROM     Commande
      WHERE    noClient = 10 AND NOT EXISTS
           (SELECT   *
            FROM     LigneCommande
            WHERE    noArticle = Article.noArticle AND
                     noCommande = Commande.noCommande))
```

ou :

```
SELECT    noArticle
FROM      Article
WHERE     NOT EXISTS
    ((SELECT   noCommande
      FROM     Commande
```

```
            WHERE    noClient = 10
           )
             EXCEPT
           (SELECT   C.noCommande
            FROM     Commande C, LigneCommande L
            WHERE    noArticle = Article.noArticle AND
                     C.noCommande = L.noCommande AND
                     noClient = 10))
```

(N.B. Oracle utilise MINUS plutôt que EXCEPT)

n) Les *Articles* dont la *description* débute par la lettre « C ».

```
SELECT    *
FROM      Article
WHERE     description LIKE 'C%'
```

o) Le *Clients* dont le *noTéléphone* n'est pas NULL.

```
SELECT    *
FROM      Client
WHERE     noTéléphone IS NOT NULL
```

p) Les *Articles* dont le prix est supérieur à la moyenne.

```
SELECT    *
FROM      Article
WHERE     prixUnitaire >
     (SELECT AVG(prixUnitaire)
      FROM Article)
```

q) Le montant total de la *Commande* #1 avant et après la taxe de 15%.

```
     SELECT    SUM(quantité*prixUnitaire)AS totalCommande,
               SUM(quantité*prixUnitaire*1.15)AS totalPlusTaxe
     FROM      LigneCommande L, Article A
     WHERE     L.noArticle = A.noArticle AND
               noCommande = 1
```

r) Le montant total de la *Livraison* #1 avant et après la taxe de 15%

```
     SELECT    SUM(quantitéLivrée*prixUnitaire)AS totalLivraison,
               SUM(quantitéLivrée*prixUnitaire*1.15)AS
totalPlusTaxe
```

```
     FROM      DétailLivraison D, Article A
     WHERE     D.noArticle = A.noArticle AND
               noLivraison = 1
```

s) La quantité commandée et quantité en attente pour chaque *LigneCommande*.

```
     SELECT    noCommande, noArticle, quantité,
          quantité-CASE WHEN SUM(quantitéLivrée) IS NULL THEN 0
                         ELSE SUM(quantitéLivrée) END
          AS quantitéEnAttente
     FROM      LigneCommande     NATURAL     LEFT     OUTER     JOIN
DétailLivraison
     GROUP BY  noCommande, noArticle, quantité
```

Solutions avec le dialecte Oracle :

```
     SELECT    L.noCommande, L.noArticle, quantité,
               quantité-NVL(SUM(quantitéLivrée),0)
               AS quantitéEnAttente
     FROM      LigneCommande L, DétailLivraison D
     WHERE     L.noArticle = D.noArticle (+) AND
               L.noCommande = D.noCommande (+)
     GROUP BY  L.noCommande, L.noArticle, quantité
```

Ou encore avec DECODE

```
SELECT      L.noCommande, L.noArticle, quantité,
            quantité-DECODE(SUM(quantitéLivrée),NULL,0,SUM(quantitéLivrée))
            AS quantitéEnAttente
FROM        LigneCommande L, DétailLivraison D
WHERE       L.noArticle = D.noArticle (+) AND
            L.noCommande = D.noCommande (+)
GROUP BY    L.noCommande, L.noArticle, quantité
```

noCommande	noArticle	quantité	quantitéEnAttente
1	10	10	0
1	70	5	0
1	90	1	0
2	40	2	0
2	95	3	2
3	20	1	0
4	40	1	0
4	50	1	1
5	10	5	5
5	20	5	5
5	70	3	1
6	10	15	15

6	40	1	1
7	50	1	1
7	95	2	2
8	20	3	3

t) La quantité commandée et quantité en attente pour chaque *LigneCommande* dont la quantité en attente est supérieur à 0.

```
     SELECT    noCommande, noArticle, quantité,
          quantité-CASE WHEN SUM(quantitéLivrée) IS NULL THEN 0
                         ELSE SUM(quantitéLivrée) END
          AS quantitéEnAttente
     FROM      LigneCommande     NATURAL     LEFT     OUTER     JOIN
DétailLivraison
     GROUP BY  noCommande, noArticle, quantité
     HAVING    (quantité-
                CASE WHEN SUM(quantitéLivrée) IS NULL THEN 0
                         ELSE SUM(quantitéLivrée) END )  > 0
```

Solutions avec le dialecte Oracle :

```
     SELECT    L.noCommande, L.noArticle, quantité,
               quantité-NVL(SUM(quantitéLivrée),0)
               AS quantitéEnAttente
     FROM      LigneCommande L, DétailLivraison D
     WHERE     L.noArticle = D.noArticle (+) AND
               L.noCommande = D.noCommande (+)
     GROUP BY  L.noCommande, L.noArticle, quantité
     HAVING    (quantité-NVL(SUM(quantitéLivrée),0)) > 0
```

u) L'article de prix minimum

```
SELECT    *
FROM      Article
WHERE     prixUnitaire =
     (SELECT MIN(prixUnitaire)
      FROM Article)
```

ou encore :

```
SELECT    *
FROM      Article
WHERE     prixUnitaire <= ALL
     (SELECT prixUnitaire
      FROM Article)
```

v) Les *noLivraison* des *Livraisons* effectuées le 4/06/2000 qui contiennent au moins deux *DétailLivraison*. Le résultat doit être trié par le *noLivraison*.

```
SELECT    L.noLivraison
FROM      Livraison L, DétailLivraison D
WHERE     L.noLivraison = D.noLivraison AND
          dateLivraison = '4/06/2000'
GROUP BY  L.noLivraison
HAVING    count(*) >=2
ORDER BY  L.noLivraison
```

w) Les *noArticle* avec la quantité totale commandée depuis le 05/07/2000 dans le cas où cette quantité dépasse 10.

```
SELECT    noArticle, SUM(quantité)
FROM      Commande C, LigneCommande L
WHERE     C.noCommande = L.noCommande AND
          dateCommande > '5/07/2000'
GROUP BY  noArticle
HAVING    SUM(quantité) >=10
```

x) Supprimer les *Articles* qui n'ont jamais été commandés.

```
DELETE    FROM Article
WHERE     NOT EXISTS
     (SELECT *
      FROM LigneCommande
      WHERE noArticle = Article.noArticle)
```

y) Augmenter la quantité commandée de 2 unités pour la *Commande* #1 et *l'Article* #10.

```
UPDATE    LigneCommande
SET       quantité = quantité + 2
WHERE     noCommande = 1 AND noArticle = 10
```

z) Supprimer le *Client* # 1 avec toutes les données qui lui sont associées (*Commandes*, *Livraisons*, etc.)

La solution suivante n'est pas la plus efficace car elle crée une table intermédiaire pour stocker temporairement les numéros de livraison des livraisons du client #1. Cette liste pourrait être conservée en mémoire centrale en utilisant un langage procédural. Dans la solution, on ne considère pas cette possibilité et tout doit être fait directement en SQL.

Il est nécessaire de stocker temporairement les numéros de livraison des livraisons du client #1 car il faut supprimer les *DétailLivraisons* avant les *Livraisons*. Mais après avoir supprimer les *DétailLivraisons*, il devient impossible de déterminer le numéro du client correspondant à une *Livraison* !

```
CREATE TABLE noLivraisonDuClient1(noLivraison INTEGER PRIMARY KEY)

INSERT INTO noLivraisonDuClient1
SELECT    DISTINCT noLivraison
FROM      DétailLivraison D, Commande C
WHERE     C.noCommande = D.noCommande AND
          noClient = 1

DELETE    FROM DétailLivraison
WHERE     noLivraison IN
     (SELECT * FROM noLivraisonDuClient1)

DELETE FROM Livraison
WHERE     noLivraison IN
     (SELECT * FROM noLivraisonDuClient1)

DELETE    FROM LigneCommande
WHERE     noCommande IN
     (SELECT   noCommande
      FROM     Commande
      WHERE    noCLient = 1)

DELETE    FROM Commande
WHERE     noCLient = 1

DELETE    FROM Client
WHERE     noCLient = 1

DROP TABLE noLivraisonDuClient1
```

2) Exercices supplémentaires :

a) Le nombre d'*Articles* à prix modique dont le *prixUnitaire* est inférieur à $15.00 et le nombre d'*Articles* dispendieux dont le *prixUnitaire* est supérieur à $25.00 (dans le même SELECT)

```
SELECT Cheap.nombre, Dispendieux.nombre
FROM
     (SELECT   COUNT(*) AS nombre
      FROM     Article
      WHERE    prixUnitaire < 15.00) Cheap,
     (SELECT   COUNT(*) AS nombre
      FROM     Article
      WHERE    prixUnitaire > 25.00) Dispendieux
```

b) Le *noClient*, son nom, le numéro de téléphone du client, le montant total commandé pour les articles dispendieux dont le prix unitaire est supérieur à $20, et le montant total commandé pour les articles à prix modiques dont le prix unitaire est inférieur à $15 pour les clients qui ont commandé un montant moins élevé d'articles dispendieux que d'articles à prix modique.

```
SELECT      Client.noClient,      nomClient,      noTéléphone,
Dispendieux.total, Modique.total
FROM Client,
     (SELECT   noClient, SUM(quantité*prixUnitaire) AS total
      FROM     Commande C, LigneCommande L, Article A
      WHERE    L.noArticle = A.noArticle AND
               C.noCommande = L.noCommande AND
               prixUnitaire >20
      GROUP BY noClient) Dispendieux,
     (SELECT   noClient, SUM(quantité*prixUnitaire) AS total
      FROM     Commande C, LigneCommande L, Article A
      WHERE    L.noArticle = A.noArticle AND
               C.noCommande = L.noCommande AND
               prixUnitaire < 15
      GROUP BY noClient) Modique
WHERE     Client.noClient = Dispendieux.noClient AND
          Client.noClient = Modique.noClient AND
          Dispendieux.total < Modique.total
```