Quatrième partie IV

Le langage SQL DML (2)

Plan du cours

Qu'allons nous aborder dans ce cours ?

1. L'extraction de données de plusieurs tables au moyen de l'opérateur de jointure
2. Les opérateurs ensemblistes (union, intersection et différence)
3. Le traitement des structures de données cycliques, dans lesquelles une ligne fait directement ou indirectement référence à une ligne de la même table
4. L'extraction de données groupées, qui permettent de s'intéresser non plus à des lignes mais à des groupes de données

Extraction de données de plusieurs tables

- La jointure permet de coupler les lignes de plusieurs tables afin d'en extraire des données corrélées.
- Exemple :

NCOM	NCLI	DATECOM	NOM	LOCALITE
30178	K111	21/12/2008	VANBIST	Lille
30179	C400	22/12/2008	FERARD	Poitiers
30182	S127	23/12/2008	VANDERKA	Namur
30184	C400	23/12/2008	FERARD	Poitiers
30185	F011	02/01/2009	PNCELET	Toulouse
30186	C400	02/01/2009	FERARD	Poitiers
30188	B512	03/01/2009	GILLET	Toulouse

Les colonnes NCOM, NCLI et DATECOM sont issues de la table COMMANDE tandis que NCLI, NOM et LOCALITE sont tirées de la table CLIENT. Cette table a été obtenue par une jointure entre les tables COMMANDE et CLIENT par association de la colonne NCLI.

Extraction de données de plusieurs tables

- La requête qui permet d'obtenir le résultat précédent est :

```
SELECT NCOM, NCLI, DATECOM, NOM, LOCALITE
FROM COMMANDE, CLIENT
WHERE COMMANDE.NCLI = CLIENT.NCLI
```

- Conceptuellement, le résultat pourrait être construit comme suit :
  1. On construit une table en couplant chaque ligne de la première table avec chaque ligne de la seconde :

     ```
     FROM COMMANDE, CLIENT
     ```

  2. On sélectionne parmi les lignes ainsi obtenues, celles qui vérifient la condition d'association et éventuellement les autres conditions :

     ```
     WHERE COMMANDE.NCLI = CLIENT.NCLI
     ```

  3. On retient que les colonnes demandées :

     ```
     SELECT NCOM, NCLI, DATECOM, NOM, LOCALITE
     ```

Extraction de données de plusieurs tables

- Une jointure peut s'appliquer sur plus de 2 tables. Il faut alors $n-1$ conditions d'association.
- Exemple :

```
SELECT NCLI, NOM, DATECOM, NPRO
FROM CLIENT, COMMANDE, DETAIL
WHERE CLIENT.NCLI = COMMANDE.NCLI
  AND COMMANDE.NCOM = DETAIL.NCOM
```

- Les conditions d'association sont aussi appelées condition de jointure
- En pratique, les conditions de jointure sont traitées comme des conditions de sélection ordinaires

Extraction de données de plusieurs tables

- Exemple :
  - Donner pour chaque commande antérieure au 23/12/2009 passée par des clients de catégorie C1 le numéro de commande, sa date, le numéro du client, son nom et sa ville

```
SELECT NCOM, NCLI, DATECOM, NOM, LOCALITE
FROM COMMANDE, CLIENT
WHERE COMMANDE.NCLI = CLIENT.NCLI
  AND CAT = 'C1'
  AND DATECOM < '23-12-2009'
```

Extraction de données de plusieurs tables

- Les lignes célibataires sont les lignes qui n'ont aucune correspondance dans l'une des tables de la jointure
- Exemple :
  - les clients qui n'ont passé aucune commande
  - aucune commande n'est célibataire
- Il existe une méthode pour obtenir les lignes célibataires appelée la joiture externe (*OUTER JOIN*)

Les opérateurs ensemblistes

- Un ensemble est une collection d'éléments distincts
  - Un ensemble de lignes ne peut donc contenir 2 lignes dont les attributs, considérés 2 à 2, ont la même valeur
- Une collection de lignes dont les éléments ne sont pas distincts constitue un multi-ensemble
  - Une requête dont la liste d'éléments de la clause SELECT n'inclut pas tous les éléments d'un identifiant renvoie un multi-ensemble
  - Un multi-ensemble peut se réduire si nécessaire à un ensemble par le modifieur DISTINCT

Les opérateurs ensemblistes

- L'opérateur UNION produit une collection de lignes distinctes à partir d'un ensemble de 2 collections de lignes
- Si une même ligne apparaît dans chacune des 2 collections de lignes en argument de l'opérateur UNION, cette ligne n'apparaîtra qu'une seule fois dans l'ensemble résultat
- Remarque : Les arguments de l'opérateur UNION peuvent être des muli-ensembles, auquels cas les doublons sont égalements éliminés

Les opérateurs ensemblistes

- Soit la requête suivante :

```
SELECT LOCALITE FROM CLIENT WHERE CAT = 'C1' ;
```

- avec la suivante qui lui ajoute des éléments :

```
SELECT LOCALITE FROM CLIENT WHERE CAT = 'C1' ;
UNION
SELECT LOCALITE FROM CLIENT WHERE COMPTE < 0 ;
```

- le résultat produit est le suivant :

LOCALITE
Namur
Poitiers
Toulouse

Les opérateurs ensemblistes

- Si l'on désire empêcher l'élimination des lignes en double, on utilisera l'opérateur UNION ALL
  - Une même ligne qui apparaît n_1 fois dans le premier membre et n_2 fois dans le second apparaîtra $n_1 + n_2$ dans le résultat, comme le montre la requête suivante :

```
SELECT LOCALITE FROM CLIENT WHERE CAT = 'C1' ;
UNION ALL
SELECT LOCALITE FROM CLIENT WHERE COMPTE < 0 ;
```

  - le résultat de la requête est le suivant :

LOCALITE
Poitiers
Namur
Poitiers
Namur
Namur
Namur
⋮

Les opérateurs ensemblistes

- Les opérateurs d'intersection (INTERSECT) et de différence (EXCEPT) fonctionne de manière similaire à l'opérateur UNION
  - L'opérateur INTERSECT construit l'ensemble des éléments simultanément présents dans deux collections
  - L'opérateur EXCEPT construit l'ensemble des éléments appartenant à la première collection mais pas à la seconde
- Remarque : Munis de la clause ALL, ces opérateurs préservent les lignes en doubles. Pour une même ligne respectivement en n_1 et n_2 exemplaires dans chaque argument :
  - INTERSECT ALL produira $min(n_1, n_2)$ exemplaires de cette ligne
  - EXCEPT ALL produira $max(n_1 - n_2, 0)$ exemplaires de cette ligne

Les opérateurs ensemblistes

- Les opérateurs d'intersection et de différence entre deux ensembles de lignes ne sont pas strictement indispensables dans la mesure ou ils peuvent être exprimés par des requête standards :
  - L'intersection ensembliste de 2 tables est obtenue par leur jointure, car celle-ci reprend les éléments qui sont simultanément présents dans ces tables
  - La différence ensembliste s'exprimera par le prédicat NOT IN dont la sous-requête définit les éléments de la seconde collection

Le produit relationnel

- Une jointure sans condition de jointure telle que la suivante n'est pas interdite

```
SELECT NCOM, CLIENT.NCLI, DATECOM, NON, ADRESSE
FROM COMMANDE, CLIENT
```

  Elle risque d'être cependant coûteuse (le résultat contiendra ici $16 \times 7 = 112$ lignes). Sauf justification, il faudra la considérer comme une erreur.
- Cette jointure particulière porte le nom de produit relationnel ou produit cartésien
- Le produit relationnel permet de comprendre le mécanisme de la jointure mais n'offre pas de grand intérêt en soi.

Le produit relationnel

- Considérons tous les couples (LOCALITE, NPRO) possibles issus de la base de données :

```
S1 = (SELECT DISTINCT LOCALITE, NPRO
      FROM CLIENT, PRODUIT)
```

- Construisons maintenant tous les couples effectifs, indiquant qu'on commande réellement le produit dans la localité :

```
S2 = (SELECT DISTINCT LOCALITE, NPRO
      FROM CLIENT C, COMMANDE M, DETAIL D
      WHERE C.NCLI = M.NCLI AND M.NCOM = D.NCOM)
```

- Le service commercial sera certainement très intéressé par les produit qu'on ne commande pas dans chaque commune, i.e., pour lesquels un efforts d'information serait utile :

```
S1 EXCEPT S2 ;
```

Requête sur des structures de données cycliques

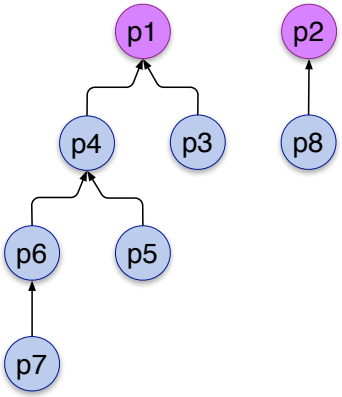
- On qualifie de cyclique une structure de données qui fait directement ou non, référence à elle-même
- Par exemple :

PERSONNE		
NPERS	NOM	RESPONSABLE

Il existe un cycle entre la colonne RESPONSABLE (clé étrangère de la table PERSONNE) et la table PERSONNE elle-même. Le rôle de cette colonne est de désigner le responsable direct de chaque personne, s'il existe.

Requête sur des structures de données cycliques

PERSONNE		
NPERS	NOM	RESPONSABLE
p1	Mercier	–
p2	Durant	–
p3	Noirons	p1
p4	Dupont	p1
p5	Verger	p4
p6	Dupont	p4
p7	Dermiez	p6
p8	Anciers	p2



Requête sur des structures de données cycliques

- La table PERSONNE permet de répondre à la question suivante : donner, pour chaque personne (S, pour subordonné) ayant un responsable (R), le numéro et le nom de celui-ci

```
SELECT S.NPERS, R.NPERS, R.NOM
FROM PERSONNE S, PERSONNE R
WHERE S.RESPONSABLE = R.NPERS ;
```

- Cette requête construit des couples de personnes, la première étant la personnes subordonnée (S) et la seconde son responsable (R)
- Elle réalise donc une jointure de la table PERSONNE avec elle-même, ce qu'on appelle une auto-jointure

Requête sur des structures de données cycliques

- Donner pour chaque personne dont le nom est Dupont, son numéro ainsi que le numéro et le nom de son responsable s'il existe

```
SELECT S.NPERS, R.NPERS, R.NOM
FROM PERSONNE S, PERSONNE R
WHERE S.RESPONSABLE = R.NPERS AND S.NOM = 'Dupont'
UNION
SELECT NPERS, '--', '--'
FROM PERSONNE
WHERE RESPONSABLE IS NULL AND Nom = 'Dupont' ;
```

Requête sur des structures de données cycliques

- Donner, pour chaque personne subordonnée à la personne de numéro p4, son numéro et son nom. On ignorera les personnes qui n'ont pas de responsable.

```
SELECT SS.NPERS, SS.NOM
FROM PERSONNE R, PERSONNE.S, PERSONNE SS
WHERE R.NPERS = 'p4'
  AND R.NPERS = S.RESPONSABLE
  AND S.NPERS = SS.RESPONSABLE ;
```

- Remarques :
  - Nous obtenons ici les subordonnées de niveau 2 en effectuant une double jointure
  - Il est impossible d'effectuer des jointures récursivement
  - Cette requête montre que SQL ne permet pas d'obtenir facilement les responsables directs et indirects d'une personne déterminée sans recourir à la programmation procédurale

Requête sur des structures de données cycliques

- On modélise la nomenclature de produit. Chaque produit est composé de sous-produits qui sont eux-même composés de sous-produits etc.

PRODUIT			
NPRO	LIBELLE	PRIX_U	POIDS_U

COMPOSITION		
COMPOSE	COMPOSANT	QTE

- La table COMPOSITION représente les relations de composition entre produits
- Une ligne (h, b, q) indique que le produit b est un composant du produit h, et qu'il faut q unités de b pour fabriquer 1 unité de h
- Les matières premières ont un prix et un poids unitaires fixés
- Le prix et le poids des autres produits peuvent être déterminés à partir des caractéristiques de leurs composants

Requête sur des structures de données cycliques

PRODUIT			
NPRO	LIBELLE	PRIX_U	POIDS_U
p1	A-200	–	–
p2	A-056	–	–
p3	B-661	–	–
p4	B-122	–	–
p5	B-326	–	–
p6	D-822	3.50	0.70
p7	D-507	8.00	0.25
p8	G-993	5.00	1.15
p9	F-016	–	–
p10	J-500	–	–
p11	J-544	0.50	0.90
p12	L-009	1.70	2.30

COMPOSITION		
COMPOSE	COMPOSANT	QTE
p1	p2	2
p1	p3	1
p1	p4	2
p2	p7	8
p2	p8	2
p3	p8	5
p4	p8	4
p4	p9	5
p4	p10	5
p5	p4	2
p5	p6	7
p9	p11	2
p10	p11	4
p10	p12	3

Requête sur des structures de données cycliques

- Donner les informations relatives aux produits p4 ainsi que sa composition

```
SELECT H.NPRO, H.LIBELLE, C.QTE, B.LIBELLE
FROM PRODUIT H, COMPOSITION C, PRODUIT B
WHERE C.COMPOSE = H.NPRO
  AND C.COMPOSANT = B.NPRO
  AND H.NPRO = 'p4' ;
```

- H et B désigne respectivement le produit composé (haut) et le produit composant (bas)

Compléments sur les jointures

- La jointure est un opérateur fondamental en ce qu'il permet de naviguer parmi les données.
- Dans la suite nous allons nous intéresser en particulier :
  1. à la pertinence d'utiliser une sous-requête ou une jointure
  2. aux valeurs dérivées dans une jointure
  3. aux jointures généralisées
  4. à l'interprétation du résultat d'une jointure

Compléments sur les jointures

- Certaines conditions utilisant une sous-requête (SELECT emboité) peuvent s'exprimer à l'aide d'une jointure
- Examinons 2 cas :
  1. Le cas des conditions d'association et de non association
  2. Le cas de sous-requête et de clé étrangère multicomposants

Compléments sur les jointures

```
SELECT NCOM, DATECOM
FROM COMMANDE
WHERE NCLI IN (SELECT NCLI
               FROM CLIENT
               WHERE LOCALITE = 'Poitiers') ;
```

peut s'écrire également sous la forme d'une jointure :

```
SELECT NCOM, DATECOM
FROM COMMANDE, CLIENT
WHERE COMMANDE.NCLI = CLIENT.NCLI
  AND LOCALITE = 'Poitiers' ;
```

Compléments sur les jointures

```
SELECT *
FROM COMMANDE
WHERE NCOM IN (SELECT NCOM
                 FROM DETAIL
                 WHERE NPRO = 'PA60'
                   AND QCOM < (SELECT QCOM
                               FROM DETAIL
                               WHERE NPRO = 'PA60'
                                 AND NCOM = '30182';
```

peut s'écrire également sous la forme d'une jointure :

```
SELECT M.NCOM, DATENCOM, NCLI
FROM COMMANDE M, DETAIL D1, DETAIL D2
WHERE D1.NPRO = D1.NCOM AND D2.NCOM = '30182'
  AND D2.NPRO = 'PA60' AND D1.QCOM < D2.QCOM ;
```

Compléments sur les jointures

```
SELECT NCOM, DATECOM, NCLI
FROM COMMANDE
WHERE NCOM NOT IN (SELECT NCOM
                   FROM DETAIL
                   WHERE NPRO = 'PA60') ;
```

n'est absolument pas équivalente à

```
SELECT DISTINCT COMMANDE.NCOM, DATECOM, NCLI
FROM COMMANDE, DETAIL
WHERE COMMANDE.NCOM = DETAIL.NCOM AND NPRO <> 'PA60';
```

- Remarque : il faut se souvenir qu'une jointure fondée sur le couple identifiant primaire/clé étrangère permet de matérialiser des associations entre lignes et non l'inexistence d'association.

Compléments sur les jointures

- Conclusion
  1. La jointure et la sous-requête permettent d'exprimer des conditions d'association entre lignes
  2. En revanche, des conditions de non-association ne sont généralement exprimables que pas des sous-requêtes, ainsi que par la forme NOT EXISTS

Compléments sur les jointures

- Une jointure permet également d'effectuer des calculs sur des quantités extraites de plusieurs tables
- Le raisonnement est simple : la jointure constitue des lignes fictives dont la clause SELECT extrait des valeurs comme elle le ferait d'une ligne réelle issue d'une table
- Par exemple, la requête suivant associe à chaque ligne de DETAIL le montant à payer :

```
SELECT NCOM, D.NPRO, QCOM*PRIX
FROM DETAIL D, PRODUIT P
WHERE D.NPRO = P.NPRO ;
```

- La requête suivante, elle établit le montant de la commande 30184 :

```
SELECT 'Montant commande 30184 = ', SUM(QCOM*PRIX)
FROM DETAIL D, PRODUIT P
WHERE D.NCOM = '30184' AND D.NPRO = P.NPRO ;
```

Compléments sur les jointures

- Les jointures étudiées jusqu'ici étaient fondées sur l'égalité des valeurs d'une clé étrangère avec celles d'un identifiant
- Toutefois, la forme même de la condition de jointure suggère que toute compararison peut servir à indiquer comment associer les lignes des tables concernées

Compléments sur les jointures

- Considérons le schéma suivant :
  - **VENTE** (CHAINE, PRODUIT, PRIX) : chaque ligne (c, p, x) indique que le produit p a été vendu dans le magasin c au prix x
  - **IMPLANTATION** (CHAINE, VILLE) : chaque ligne (c, v) indique que le magasin c est implanté dans la ville v
- La colonne CHAINE n'est pas une clé étrangère, ni un identifiant de la relation IMPLANTATION
- On peut seulement espérer que les colonnes aient des valeurs communes
- Nous ne sommes pas dans le schéma classique représentant des associations explicites
- ```
  SELECT DISTINCT PRODUIT, VILLE, PRIX
  FROM VENTE V, IMPLANTATION I
  WHERE V.CHAINE = I.CHAINE ;
  ```

  indique, pour chaque ligne (p, v, x), que le produit p est disponible dans la ville v au prix x quelles que soient les magasins qui proposent p

Compléments sur les jointures

- L'exemple ci-dessous illustre une opération fréquente qui consiste à condenser de l'information de manière à la rendre plus lisible
- La table ci-dessous établit des intervalles successifs de valeurs de compte (MIN_CPT et MAX_CPT) leur attribue un code (CODE_CPT)

CLASSE_CPT		
MIN_CPT	MAX_CPT	CODE_CPT
10000	32000	A
5000	10000	B
2000	5000	C
1000	2000	D
500	1000	F
0	500	G
-500	0	U
-1000	-500	V
-2000	-1000	W
-5000	-2000	X
-10000	-5000	Y
-32000	-10000	Z

Compléments sur les jointures

- Comment associer à chaque client le code de son compte ?

```
SELECT NCLI, NOM, CODE_CPT
FROM CLIENT, CLASSE_CPT
WHERE CAT = 'C1'
  AND COMPTE >= MIN_CPT AND COMPTE < MAX_CPT ;
```

NCLI	NOM	CODE_CPT
B112	HANSENNE	D
C123	MERCIER	X
F010	TOUSSAINT	G
L422	FRANCK	G
S127	VANDERKA	X

Compléments sur les jointures

- La construction d'une requête qui utilise une ou plusieurs jointures peut s'avérer délicate
- Il importe donc de bien comprendre ce que représente le résultat d'une jointure
```

Compléments sur les jointures

- La question est la suivante : sachant que toute ligne d'une table représente une entité du domaine d'application (un client, un achat, un détail, etc.) quelles entités les lignes d'une jointure représentent-elles ?
- Par exemple, chaque ligne produite par l'évaluation de la requête :

```
SELECT C.NCLI, NOM, LOCALITE
FROM CLIENT C, COMMANDE M
WHERE M.NCLI = C.NCLI ;
```

- représente-t-elle ?
  1. un client
  2. un client qui a passé une ou plusieurs commandes
  3. une commande
- Réponse : des commandes

Compléments sur les jointures

- La règle relative à une jointure élémentaire fondée sur l'égalité indentifiant/clé étrangère est simple
- Soit une table TA, d'indentifiant IA, et une table TB de clé éntrangère RA obligatoire référençant TA :
  - TA (IA, DA)
  - TB (IB, RA, DB)
- Le résultat de la requête :

```
SELECT *
FROM TA, TB
WHERE TA.IA = TB.RA ;
```

  contient autant de lignes qu'il y en a dans la table TB. Autrement dit, chaque ligne du résultat d'une jointure représente une ligne de TB
- En bref : le résultat d'une jointure représente des entités de la table contenant la clé étrangère

Compléments sur les jointures

- Qu'en est il pour cette requête ?

```
SELECT COMMANDE.NCOM, DATECOM, NCLI
FROM COMMANDE, DETAIL
WHERE COMMANDE.NCOM = DETAIL.NCOM ;
```

- Réponse : les lignes sont celles de la table DETAIL

Compléments sur les jointures

- Ce que nous venons de discuter conduit à une autre règle : l'identifiant du résultat de la jointure

```
SELECT *
FROM TA, TB
WHERE TA.IA = TB.RA ;
```

  est constitué des colonnes de l'identifiant primaire de TB (soit IB)
- Si l'identifiant primaire de TB n'est pas répris dans la clause SELECT, le résultat n'a pas d'identifiant :

```
SELECT LOCALITE, LIBELLE
FROM CLIENT CLI, COMMANDE COM, DETAIL D, PRODUIT P
WHERE CLI.NCLI = COM.NCLI
  AND COM.NCOM  D.NCOM
  AND D.NPRO = P.NPRO ;
```

Extraction de données groupées

- Les requêtes examinées jusqu'à maintenant produisent des lignes qui sont généralement une correspondance une pour une avec les lignes d'une table de la clause FROM
- Nous allons maintenant examiner comment il est possible d'extraire d'une table, ou d'une jointure, des informations sur les concepts latents dans ces tables

Extraction de données groupées

- Considérons la table CLIENT. Il est permis d'y percevoir, virtuellement du moins des groupes de clients selon leur localité, ou selon leur catégorie, ou encore selon leur nom.

CLIENT			
NCLI	NOM	LOCALITE	COMPTE
F400	JACOB	Bruxelle	0
B332	MONTI	Genève	0
K111	VANBIST	Lille	720
S127	VANDERKA	Namur	-4580
L422	FRANCK	Namur	0
C123	MERCIER	Namur	-2300
B062	GOFFIN	Namur	-3200
S712	GUILLAUME	Paris	0
F010	TOUSSAINT	Poitiers	0
B112	HANSENNE	Poitiers	1250
C400	FERARD	Poitiers	350

Extraction de données groupées

- Par exemple, la requête suivante donne, pour chaque groupe de clients classés ou regroupés par localité, le nom de celle-ci, le nombre de clients dans le groupe et la valeur moyenne des comptes des clients du groupe

```
SELECT LOCALITE,
       COUNT(*) AS NOMBRE_CLIENT,
       AVG(COMPTE) AS MOYENNE_COMPTE
FROM CLIENT
GROUP BY LOCALITE ;
```

- Le résultat est le suivant :

LOCALITE	NOMBRE_CLIENTS	MOYENNE_COMPTE
Bruxelles	1	0.00
Genève	1	0.00
Lille	1	720.00
Namur	4	-2520.00
Paris	1	0.00
Poitiers	3	533.33
Toulouse	5	-2530.00

Extraction de données groupées

- Des conditions de sélection peuvent être imposées aux groupes à sélectionner
- Elles sont exprimées dans une clause HAVING pour éviter toute confusion avec la clause WHERE
- Par exemple, la requête suivante donne le montant moyen des comptes des clients des villes en comptant au moins 3 :

```
SELECT LOCALITE, COUNT(*), AVG(COMPTE)
FROM CLIENT
GROUP BY LOCALITE
HAVING count(*) >= 3 ;
```

- Le résultat est le suivant :

LOCALITE	COUNT(*)	AVG(COMTE)
Poitiers	3	533.33
Namur	4	-2520.00
Toulouse	5	-2530.00

Extraction de données groupées

- Remarque : la condition exprimée par la clause HAVING peut porter sur les éléments cités dans la clause SELECT, mais aussi sur toute autre fonction d'agrégation calculable sur chaque groupe

Extraction de données groupées

- Exemple : on souhaite obtenir la liste des clients ayant commandé au moins 2 fois le produit PA45
  1. On regroupe les lignes de la table COMMANDE en les regroupant par client :

```
SELECT NCLI, COUNT(*)
FROM COMMANDE
GROUP BY NCLI ;
```

  2. On ne retient ensuite que les groupes d'un moins 2 commandes :

```
SELECT NCLI, COUNT(*)
FROM COMMANDE
GROUP BY NCLI
HAVING COUNT(*) >= 2 ;
```

Extraction de données groupées

3. Finalement, on ne considère, avant groupement, que les commandes spécifiant le produit PA45

```
SELECT NCLI, COUNT(*)
FROM COMMAND
WHERE NCOM IN (SELECT NCOM
               FROM DETAIL
               WHERE NPRO = 'PA45')
GROUP BY NCLI
HAVING COUNT(*) >= 2 ;
```

Extraction de données groupées

- Il est également possible d'obtenir la quantité totale du produit PA45
- Les données appartenant à plusieurs tables (NCLI et QCOM), il faut effectuer un groupement sur le résulat de la jointure de COMMANDE et DETAIL

```
SELECT NCLI, COUNT(*), SUM(QCOM)
FROM COMMANDE M, DETAIL D
WHERE M.NCOM = D.NCOM
  AND NPRO = 'PA45'
GROUP BY NCLI
HAVING COUNT(*) >= 2 ;
```

Extraction de données groupées

- Considérons un autre exemple : Pour chaque client de Poitiers donner le montant total de ses commandes

```
SELECT 'Montant du par ',
       C.NCLI,
       ' = ', SUM(QCOM*PRIX)
FROM CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
WHERE LOCALITE = 'Poitiers'
  AND M.NCLI = C.NCLI
  AND M.NCOM = D.NCOM
  AND D.NPRO = P. NPRO
GROUP BY M.NCLI ;
```

Extraction de données groupées

- Considérons un dernier exemple : Donner la quantité qui reste en stock si on déduit les quantités commandées

```
SELECT P.NRPO, QSTOCK - SUM(D.QCOM) AS SOLDE
FROM DETAIL D, PRODUIT P
WHERE D.NPRO = P.NPRO
GROUP BY P.NPRO, QSTOCK ;
```

- Le résultat est le suivant :

NRPO	SOLDE
CS262	-15
CS464	-135
PA45	535
PA60	-1
PH222	690
PS222	620

Extraction de données groupées

- Remarques :
  1. Le critère de groupement peut inclure plusieurs noms de colonne
  2. L'ordre des colonnes est indifférent
- Exemple : Donner pour chaque localité et produit le montant total commandé

```
SELECT LOCALITE, P.NPRO, SUMM(QCOM*PRIX) AS Montant
FROM CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
WHERE M.NCLI = C.NCLI
  AND M.NCOM = D.NCOM
  AND D.NPRO = P. NPRO
GROUP BY LOCALITE, P.NPRO ;
```

Extraction de données groupées

- Remarque : Le critère de groupement peut aussi inclure une expression de calcul quelconque
- Exemple : Donner la liste des clients en fonction de la première lettre de leur catégorie

```
SELECT SUBSTRING(CAT FROM 1 for 1) AS CAT,
       COUNT(*) AS N
FROM CLIENT
GROUP BY SUBSTRINB(CAT FROM 1 FOR 1) ;
```

- Le résultat est le suivant :

CAT	N
<null>	2
B	8
C	6

Extraction de données groupées

- Un regroupement très intéressant serait de regrouper les clients selon leurs valeurs de COMPTE par intervalle de 1.000

```
SELECT 'de', int(COMPTE/1000)*1000 AS MIN,
       ' a ', int(COMPTE/1000)*1000 + 999 AS MAX,
       COUNT(*) AS N
FROM CLIENT C
GROUP BY int(COMPTE/1000) ;
```

- Le résultat est le suivant :

de	Min	a	Max	N
de	-9000	a	-8001	1
de	-5000	a	-4001	1
de	-4000	a	-3001	1
de	-3000	a	-2001	2
de	-2000	a	-1001	1
de	0	a	999	9
de	1000	a	1999	1

Extraction de données groupées

- L'extraction de données groupées est à définir avec précaution avec des jointures
- Exemple : Donner pour chaque localité, la somme des comptes des clients et le nombre de commandes. On serait tenté d'écrire :

```
SELECT LOCALITE, SUM(COMPTE), COUNT(*)
FROM CLIENT C, COMMANDE M
WHERE C.NCLI = M.NCLI
GROUP BY LOCALITE ;
```

- Ce qui donnerait :

LOCALITE	SUM(COMPTE)	COUNT(*)
Lille	720.00	1
Namur	-4580.00	1
Poitiers	1050.00	3
Toulouse	-8700.00	2

- Ce résultat, en apparence correct, est pourtant erroné (indépendemment du fait que les clients sans commande ne sont pas repris)

Extraction de données groupées

LOCALITE	SUM(COMPTE)	COUNT(*)
Lille	720.00	1
Namur	-4580.00	1
Poitiers	1050.00	3
Toulouse	-8700.00	2

- Le résultat de la jointure n'est pas des clients mais des commandes
  - Rappel : le résultat d'une jointure représente des entités de la table contenant la clé étrangère
- En particulier, le compte du client CS400 est comptabilisé 3 fois, pour un total de 1050 au lieu de 350
- Le calcul de la somme des comptes s'effectue donc sur des ensembles de commandes et non des clients
- Pour répondre correctement à la question il faut procéder en deux étapes

Extraction de données groupées

- Il est possible d'éviter la clause GROUP BY lorsque le concept latent dans une table est explicitement représenté par une autre table, et que le regroupement ne sert qu'à la sélection
- Exemple : Donner les produits dont plus de 500 unités ont été commandées en 2009

```
SELECT D.NPRO
FROM DETAIL D, COMMANDE M
WHERE C.NCOM = M.NCOM AND DATECOM LIKE '%2009'
GROUP BY D.NPRO HAVING SUM(QCOM) > 500 ;
```

est équivalente à

```
SELECT NPRO FROM PRODUIT P
WHERE (SELECT SUM(QCOM) FROM DETAIL
  WHERE NPRO = P.NPRO AND NCOM IN
    (SELECT NCOM FORM COMMANDE
      WHERE DATECOM LIKE '%2009')) > 500 ;
```

Ordre des lignes d'un résultat

- Par construction, l'ordre des lignes d'une table est arbitraire
- On ne peut pas supposer que les lignes sont stockées dans un ordre déterminé
- Par principe, l'ordre des lignes du résultat d'une requête est aussi arbitraire
- Il est possible d'imposer un ordre de présentation en utilisant la clause ORDER BY

Ordre des lignes d'un résultat

- Donner la liste ordonnée par localité des clients (numéro, nom et localité) de catégorie C1 et C2

```
SELECT NCLI, NOM, LOCALITE
FROM CLIENT
WHERE CAT IN ('C1', 'C2')
ORDER BY LOCALITE ;
```

- On peut également indiquer plusieurs critères de tri :

```
SELECT *
FROM CLIENT
ORDER BY LOCALITE, CAT ;
```

Les clients vont apparaître classés par localité, puis dans chaque localité classés par catégorie. Attention l'ordre à une importance.

Ordre des lignes d'un résultat

- Il est possible de modifier l'ordre utilisé pour le tri
  - La clause ASC pour le tri ascendant
  - La clause DESC pour le trie descendant
- Par défaut, le tri est ascendant
- Par exemple :

```
SELECT *
FROM PRODUIT
WHERE LIBELLE LIKE '%SAPIN%'
ORDER BY QSTOCK DESC ;
```

Ordre des lignes d'un résultat

- Si une expression apparaît dans le SELECT, elle sera spécifiée par son nom
- S'il s'agit d'une colonne avec un alias, elle sera spécifié par celui-ci
- Exemple : Donner les localités par valeurs décroissantes de leur population de clients

```
SELECT LOCALITE,
  COUNT(*) AS POPULATION,
  SUM(COMPTE)
FROM CLIENT
GROUP BY LOCALITE
ORDER BY POPULATION DESC ;
```

Ordre des lignes d'un résultat

- Il est possible d'utiliser des critères qui ne sont pas dans la clause SELECT
- Exemple :

```
SELECT NCOM, NPRO, QCOM
FROM DETAIL D, PRODUIT P
WHERE D.NPRO = P.NPRO
ORDER BY NCOM, QCOM*PRIX DESC ;
```

Interprétation d'une requête

- Pour une requête monotable :
  1. On considère la table spécifiée dans la clause FROM
  2. On sélectionne les lignes sur la base de la clause WHERE
  3. On classe ces lignes en groupes comme spécifié dans la clause GROUP BY
  4. On ne retient que les lignes qui vérifient la clause HAVING
  5. Les lignes des groupes sont ordonnées selon la clause ORDER BY éventuellement
  6. De chacune des lignes, on extrait les valeurs demandées dans la clause SELECT

Interprétation d'une requête

- Pour une requête multitable :
  1. On considère les tables spécifiées dans la clause FROM
  2. On effectue la jointure de ces tables selon le critère de jointure de la clause WHERE
  3. On sélectionne les lignes de la jointure sur la base des autres conditions de la clause WHERE
  4. On classe ces lignes en groupes comme spécifié dans la clause GROUP BY
  5. On ne retient que les groupes qui vérifient la clause HAVING
  6. Les lignes des groupes sont ordonnées selon la clause ORDER BY évetuellement
  7. De chacune des lignes, on extrait les valeurs demandées dans la clause SELECT

Interprétation d'une requête

- À titre d'exemple, on indique, pour la requête suivante, l'ordre d'interprétation qui conduisent à l'élaboration du résultat :

```
7 : SELECT NCLI, COUNT(*), SUM(QCOM)
1 : FROM COMMANDE M, DETAIL D
2 : WHERE M.NCOM = D.NCOM
3 :   AND NPRO = 'PA60'
4 : GROUP BY NCLI
5 : HAVING COUNT(*) > = 2
6 : ORDER BY NCLI
```

- Remarque : Il s'agit d'une évaluation fictive, le SBGD utilisant généralement d'autres procédés plus efficaces pour construire le résultat

Exercice 1

- Exprimer les requêtes suivantes en SQL :
  1. Calculer le montant de chaque détail de commande du client 'C400'
  2. Calculer le montant commandé des produits en sapin
  3. Afficher le total et la moyenne des comptes clients, ainsi que le nombre de clients, selon chacune des classifications suivantes : (1) par catégorie, (2) par localité, (3) par catégorie dans chaque localité

Exercice 1

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

```
SELECT D.NCOM, P.NPRO, QCOM*PRIX AS MONTANT
FROM COMMANDE M, DETAIL D, PRODUIT P
WHERE D.NCOM = M.NCOM AND D.NPRO = P.NPRO AND NCLI = 'CS400' ;
```

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%' ;
```

3. Afficher le total et la moyenne des comptes clients, ainsi que le nombre de clients, selon chacune des classifications suivantes : (1) par catégorie, (2) par localité, (3) par catégorie dans chaque localité

```
SELECT LOCALITE, CAT, SUM(COMPTE), AVG(COMPTE), COUNT(*)
FROM CLIENT
GROUP BY LOCALITE, CAT ;
```

Exercice 2

- Exprimer les requêtes suivantes en SQL :
  1. Combien y a-t-il de commandes spécifiant un (ou plusieurs) produit(s) en acier ?
  2. Calculer le montant dû par chaque client. Dans le calcul, on ne prendra en compte que le montant des commandes
  3. Afficher pour chaque localité, les libellés des produits qui y sont commandés

Exercice 2

1. Combien y a-t-il de commandes spécifiant un (ou plusieurs) produit(s) en acier ?

```
SELECT COUNT(DISCTINCT M.NCOM)
FROM COMMANDE M, DETAIL D, PRODUIT P
WHERE M.NCOM = D.NCOM AND D.NPRO = P.NPRO
  AND LIBELLE LIKE '%ACIER%' ;
```

2. Calculer le montant dû par chaque client. Dans le calcul, on ne prendra en compte que le montant des 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) ;
```

3. 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.NCOL = D.NCOM AND D.NPRO = P.NPRO
GROUP BY LOCALITE, LIBELLE ORDER BY LOCALITE, LIBELLE
```

Exercice 3

Soit le schéma relationnel suivant :

- **PRODUIT**(NPRO, LIBELLE, PRIX_U, POIDS_U)
- **COMPOSITION**(COMPOSE, COMPOSANT, QTE)

Exprimer en SQL les requêtes suivantes :

1. Les matières premières (produit qui n'ont pas de composants)
2. Les produit finis (qui n'entre dans la composition d'aucun autre)
3. Les produit semi-finis (tous les autres)
4. Le prix et le poids unitaires d'un produit fini ou semi-fini dont tous les composants ont un poids et un prix unitaires

Exercice 3

1. Les matières premières (produit qui n'ont pas de composants)

```
SELECT NPRO FROM PRODUIT
WHERE NPRO NOT IN (SELECT COMPOSE FROM COMPOSITION) ;
```

2. Les produit finis (qui n'entre dans la composition d'aucun autre)

```
SELECT NPRO FROM PRODUIT
WHERE NPRO NOT IN (SELECT COMPOSANT FROM COMPOSITION) ;
```

3. Les produit semi-finis (tous les autres)

```
SELECT NPRO FROM PRODUIT
WHERE NPRO IN (SELECT COMPOSE FROM COMPOSITION)
  AND NPRO IN (SELECT COMPOSANT FROM COMPOSITION) ;
```

4. Le prix et le 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*PN.PRIX_U), SUM(QTE*PB.POIDS_U)
FROM PRODUIT PH, COMPOSITION C, PRODUIT PB
WHERE PH.BPRO = 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 (NN.PRIX_U is null or BB.POIDS_U is null) ;
```