NFA008 - Bases de Données Relationnelles

Support de cours

Christian DUCLOU
Novembre 2014

Sommaire

Introduction.

Ce document est le support de présentation de la conception d'un système d'information au travers d'une base de données relationnelle.

Nous aborderons successivement la définition des fonction d'un système de gestions de bases de données (chapitre 1), ensuite nous verrons au chapitre 2 comment donner des représentations plus ou moins abstraites d'une même base de données. Alors, nous aborderons uns méthode de modélisation, qui est la pierre fondatrice de la viabilité d'un système d'information, cela constituera le chapitre 3. Un exemple d'application apportera une assise générique à ces travaux préliminaires dans le chapitre 4.

Le chapitre 5 nous permettra de définir un formalisme d'exploitation des informations d'une base de données, avant de mettre en œuvre le langage SQL que nous étudierons au chapitre 6. L'exploitation pratique sera complétée au chapitre 7 par la présentation d'un langage "procédural" proche de celui que l'on trouve dans le SGBD Oracle, au chapitre 8 avec l'exploitation d'une base de données à partir d'une classe Java par un connecteur JDBC, et au chapitre 9 avec PHP.

Tous ces éléments seront mis en pratique dans le cadre d'un second volet du cours où les auditeurs, installerons, administreront et exploiteront un SGBD populaire, MySQL, et un SGBD Relationnel-Objet, PostgreSQL. Ce second volet sera entrelacé avec le premier, nous verrons comment définir des bases de données, des tables, des vues, des fonctions, des déclencheurs (*trigger*), des rôles d'utilisateurs et de groupes auxquels nous attribuerons des droits, et aussi nous mettrons en œuvre les sauvegardes et restaurations de bases de données. Nous verrons aussi comment définir plusieurs instances de SGBD à travers plusieurs port de réseau.

Chapitre 1. Les systèmes de Gestion de Bases de Données.

Définitions.

Une base de données (BD) est un ensemble structuré d'informations enregistrées. Plusieurs utilisateurs peuvent accéder sélectivement à ces informations. Un système de gestion de base de données (SGBD) permet à un utilisateur d'agir sur une base de données. Il organise les données sur les mémoires de masse et fournit les procédures de création, de recherche et de mise à jour de ces données. Le SGDB permet à l'utilisateur de décrire l'ensemble des données qui seront stockées dans la base de données. Il faut distinguer deux niveaux :

- description physique de l'organisation et du stockage des données.
- description logique de la base de données telles qu'elle est vue par l'utilisateur.

Fonctions d'un SGBD.

- Utilisation: la nature du dialogue dépend de l'utilisateur. Les informaticiens définissent les procédures algorithmiques des applications.
- Intégrité: les contraintes d'intégrité définissent les règles de cohérence des données qui devront être vérifiées.
- Confidentialité: les accès doivent être discriminés selon l'utilisateur.
- Concurrence d'accès: lorsque plusieurs processus utilisent la même donné, le SGDB doit garantir un résultat cohérent des opérations réalisées (accès concurrents ☞ verrouillage d'accès, file d'attente…).
- Sécurité de fonctionnement: en cas de panne, le redémarrage du SGBD doit être assujetti à des points de contrôle pour assurer la fiabilité des données. Les « transactions » sont utilisées pour des mises à jour, elles font évoluer l'état de la base de donnée. Les transactions sont assimilables aux arcs d'un automate d'états finis. Les requêtes de mises à jour sont enregistrées dans un journal avant d'être exécutées. La fin d'une transaction est marquée par le programmeur ou des action su SGBD. La fin d'une transaction sert à assurer la validité des données et donc la stabilité du système.

Chapitre 2. Représentation d'une base de données.

Un **modèle de données**[1] reflète des éléments du monde réel. Le modèle de données décrit des classes d'objets et des liens entre ces classes.

Niveau conceptuel.

La base de données est spécifiée en termes abstraits ; il est commode de regrouper les objets en catégories, autrement dit de les classer par nature.

Exemple :

- la classe *COMMANDE* groupe les informations propres d'un bon de commande,
- la classe *PRODUIT* groupe les informations propres d'un *PRODUIT*,
- la *LIGNE* (de commande) associe la classe *COMMANDE* à la classe *PRODUIT* et porte la quantité commandée de produit.

Le modèle hiérarchique et le modèle réseau.

L'élément de base de ces modèles est l'enregistrement logique.

Dans le cas du modèle réseau, les enregistrements sont reliés pour former des listes circulaires avec ou sans liens arrière entre enregistrements. Les modèles réseaux qui ont connu un développement important sont les modèles CODASYL et SOCRATE.

Dans le cas du modèle hiérarchique, les enregistrements sont reliés pour former un graphe arborescent. Les sommets correspondent aux classes d'objets et les arcs aux associations. Donc, à la différence du modèle réseau, le modèle hiérarchique n'admet pas de circuit dans le graphe.

Le modèle relationnel.

Ce modèle utilise la relation telle qu'elle est définie en mathématiques. Une relation est un ensemble de **propriétés**. Le schéma d'une relation est la liste de ses propriétés.

Par exemple l'association *LIGNE* pourra être représentée comme une relation construite sur les relations *COMMANDE* et *PRODUIT*.

Formellement on écrira :

```
LIGNE = { (x, y, z) | le produit x apparaît dans la commande y,
                      pour laquelle il est commandé en z exemplaires }.
```

Autrement dit, *LIGNE* est l'ensemble des *triplets* de la forme *(x,y,z)* qui vérifient la propriété "*le produit x apparaît dans la commande y, dans laquelle il est commandé en z exemplaires*".

Le modèle objet.

Les classes définissent les propriétés et les "méthodes" des objets. L'utilisation d'une propriété peut être restreinte par l'emploi de fonctions d'accès. Cela permet d'assurer un contrôle a priori et/ou a posteriori du traitement effectué sur la propriété (méthodes de classe).

Il est possible de définir de nouvelles classes par combinaison de classes existantes (composition), mais aussi d'agréger de nouvelles propriétés et méthodes à une classe existante en bénéficiant des définitions initiales (héritage).

[1] Les modèles de traitements ne sont pas présentés dans ce document.

Niveaux externe et interne - Mise en œuvre.

Le niveau externe est représenté par les différentes partitions du schéma conceptuel qui permettent de donner des **points de vue** contextuels.

Ces filtres aident à l'écriture des procédures de traitement (grille de saisie, états, distribution de la base sur un réseau de serveur, communication, etc…).

L'organisation des données en mémoire de masse est définie au niveau interne. Le SGBD comprend un **langage de définition de données** (LDD).

L'administrateur de base de données dispose du LDD pour définir l'organisation et le schéma physique des données, ainsi que les schémas externes requis par les applications (LDD). Il doit aussi définir les droits d'accès alloués à chaque utilisateur et mettre en œuvre les procédures de contrôle de la sécurité de fonctionnement.

Une fois que les schémas sont définis, il faut être en mesure de créer et de mettre à jour des informations; il faut bien sûr pouvoir consulter les informations ainsi enregistrées. Pour tout cela on dispose d'un **langage de manipulation de données** (LMD).

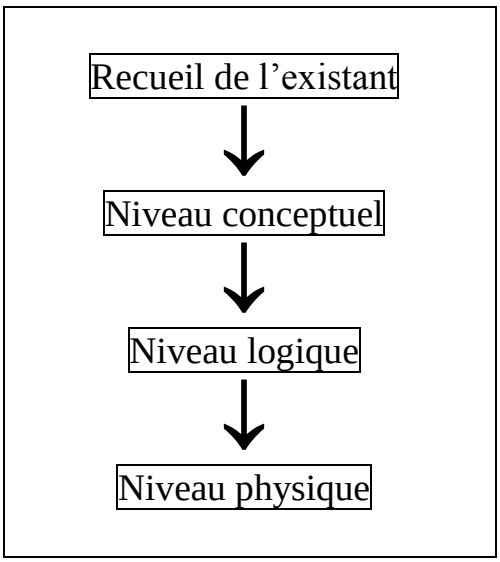
Chapitre 3. Modélisation - Exemple méthode MERISE.

La méthode MERISE est présentée dans un ouvrage de Hubert TARDIEU, Arnold ROCHFELD et René COLETTI (Voir "Bibliographie").

Cette méthode propose un outil permettant de réaliser l'étude rigoureuse des systèmes d'informations. C'est une approche systémique qui sépare la conception des données de celle des traitements.

MERISE s'appuie sur trois niveaux d'abstractions.

Figure 1 : Les niveaux d'abstraction de la méthode MERISE.



Recueil de l'existant.

Cette étape permet de fixer les données de l'analyse au travers de documents :

- Domaine de l'étude
- Objectifs,
- Schémas directeurs,
- Documents existants (bons de commande, bordereaux,…),
- Règles de gestion internes et légales,
- Graphes de flux,
- Dictionnaire de données (épuré de tout *synonyme*[2] et de tout *polysème*[3]).

[2] Plusieurs nom pour un seul sens.
[3] Plusieurs sens pour un seul nom.

Les modèles et la méthode.

A chaque étapes correspondent des modèles de données et des modèles de traitements.

Figure 2 : Les étapes de la méthode MERISE.

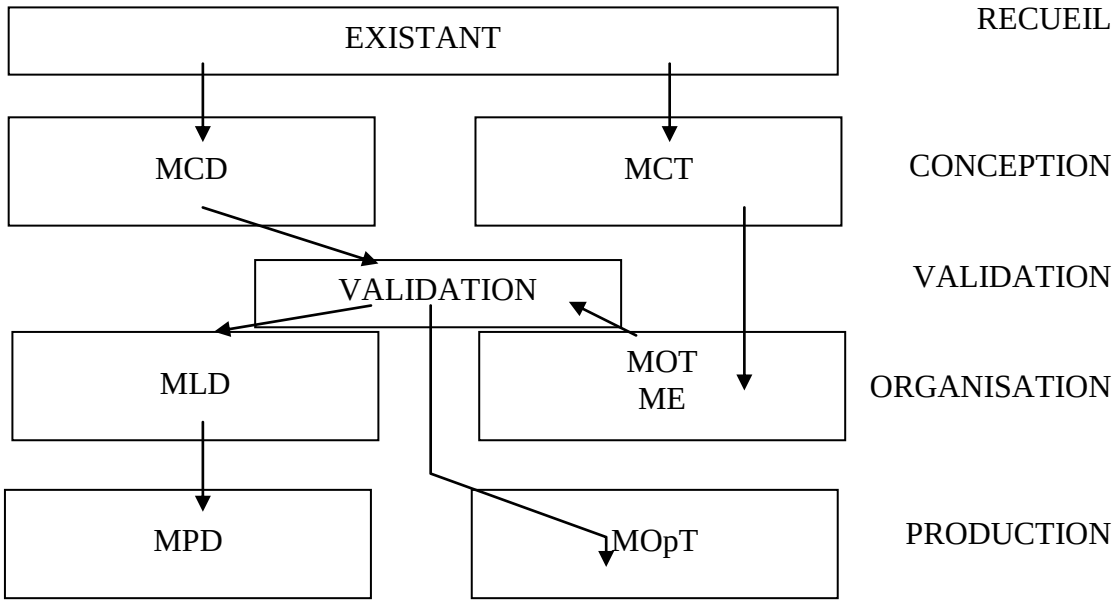


Tableau 1 : Les modèles de la méthode MERISE.

MCD: Modèle Conceptuel de Données	MCT: Modèle Conceptuel de Traitements
MLD: Modèle Logique de Données	MOT: Modèle Organisationnel de Traitements ME: Modèles Externes
MPD: Modèle Physique de Données	MopT:Modèle Opérationnel de Traitement

Dépendances Fonctionnelles (DF).

Un lien entre une propriété ou un ensemble de propriétés et une autre définit une dépendance fonctionnelle. Deux représentations sont utilisées, graphique ou matricielle. Un lien est représenté graphiquement par une flèche, et par l'intersection d'une ligne et d'une colonne dans une matrice d'éléments binaires.

La représentation graphique s'appelle le graphe des dépendances fonctionnelles, c'est un graphe orienté dont les sommets sont les propriétés et dont les arcs sont les dépendances fonctionnelles entre les propriétés. Deux points de vue sont à considérer. On cherche d'une part à connaître l'ensemble des DF entre les propriétés, et, d'autre part à minimiser le nombre de liens à définir au niveau interne. Lorsqu'un graphe contient toutes les dépendances fonctionnelles, il est appelé fermeture transitive. La couverture minimale d'un graphe est l'ensemble minimum (sans transitivité) des arcs, la forme minimale optimise les redondances de propriétés.

Deux approches peuvent être envisagées dans la construction du MCD.

- On étudie chaque couple de propriétés d'une relation universelle.
- On recherche une structure des propriétés:
  - on regroupe des propriétés en sous-arbres qui caractérisent logiquement la même entité,
  - on détermine les liens entre entités,

- les propriétés dépendant de plusieurs propriétés constituent les associations entre entités.

Exemple.

Un client est identifié par un numéro, le numéro du client fournit le nom et l'adresse du client. Une facture est identifiée par un numéro et comporte (entre autres) la date de la facture, le numéro, le nom et l'adresse du client.

Figure 3 : Fermeture transitive et couverture minimale d'un graphe.

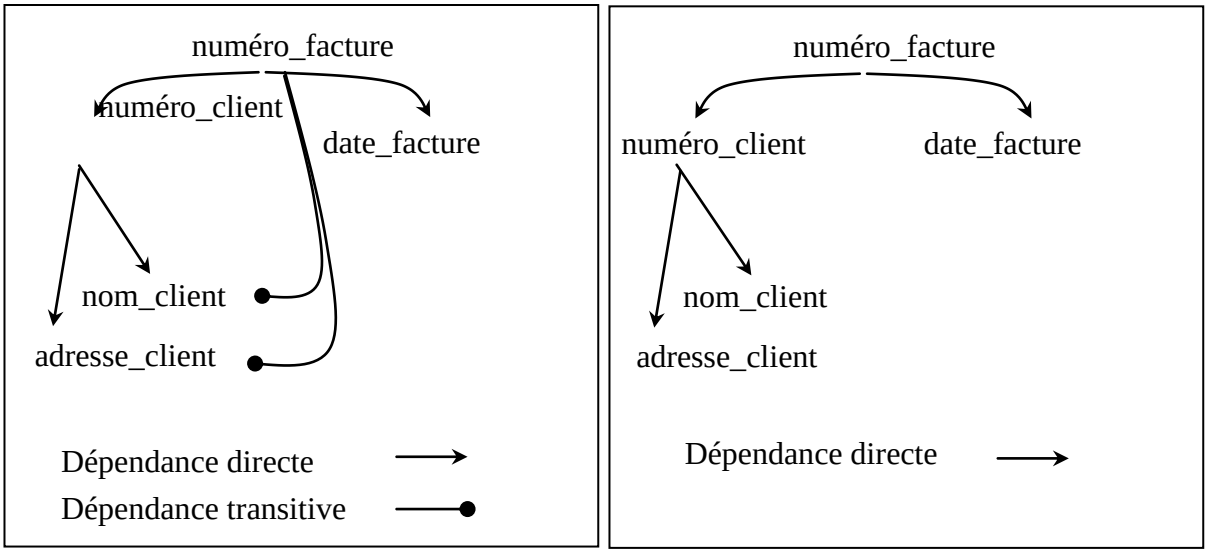


Figure 4 : Matrice des dépendances fonctionnelles (A dépend de B)

A B→ ↓	numéro_client	nom_client	adresse_client	numéro_facture	date_facture
numéro_client				OUI	
nom_client	OUI			OUI	
adresse_client	OUI			OUI	
numéro_facture					
date_facture				OUI	

Le schéma entité-association.

Le schéma entité-association représente la structure de la base.

Le MCD met en évidence la partition des propriétés et les liens entre les parties.

Les **entités** sont des objets **(type-entité)** du système d'information (CLIENT, FOURNISSEUR, PRODUIT,…). Un type-entité regroupe des **propriétés** liés directement.

La **clef** d'une entité identifie sans ambigüité une instance de l'entité. Elle est constituée d'une ou plusieurs propriétés.

Les **associations** sont des objets **(type-association)** qui relient des entités, elles portent éventuellement des propriétés.

Les **cardinalités** permettent de quantifier les liaisons entre les entités et les associations ; plus précisément, on exprime les nombres **minimum** et **maximum** de liens existants. Elles représentent le caractère **obligatoire** ou **facultatif**, **unique** ou **multiple** des **occurrences**.

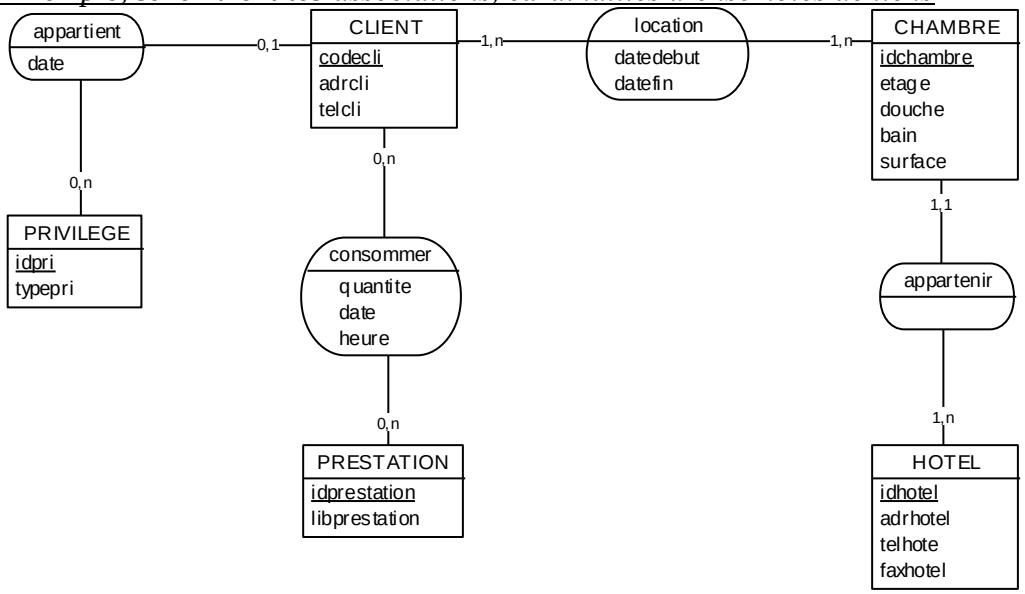
Les quatre combinaisons de cardinalités standards sont :

(0,1) : au moins zéro, au plus un
(1,1) : au moins un, au plus un
(0,n) : au moins zéro, au plus n
(1,n) : au moins un, au plus n

Les **cardinalités maximum** permettent de définir la **clef minimale** d'une relation issue d'une association (voir paragraphe suivant). Les deux combinaisons possibles dans le cas d'une association binaire[4] sont :

Association 1,N : (0,1) ou (1,1) et (0,n) ou (1,n)
Association N,M : (0,n) ou (1,n) et (0,n) ou (1,n)

Figure 5 : Exemple, schéma entités-associations, cardinalités d'ensembles de liens



"appartient" et "appartenir" sont des associations 1,N;
"consommer" et "location" sont des associations N,M

4 Arité = 2, deux entités sont associées. Il est toujours possible de traduire une relation n-aire en un ensemble équivalent de relations binaires.

Algorithme de transformation du MCD en MLD-relations.

Cette opération consiste à traduire les entités et associations en une collection relations selon l'un des algorithmes suivants.

A partir schéma entité-association:

1) 1 **type-entité** devient 1 **relation**,
2) 1 **propriété** devient 1 **attribut** de relation,
3) la **clef** d'une entité devient la **clef** d'une relation,
4) 1 **type-association** devient une relation dont la **clef** dépend des cardinalités maximum des liens de l'association :
   association **1, N** : la clef est celle de l'entité dont la cardinalité est minimale,
   association **N, M** : la clef reste la concaténation des clefs,
5) fusionner les relations de même clef,
6) supprimer éventuellement les relations dont la clef est le seul constituant.

A partir du graphe des dépendances fonctionnelles:

1) 1 **propriété** où arrivent plusieurs flèches a pour **clef** la concaténation des propriétés d'où partent les flèches,
2) 1 **propriété** où arrive une flèche a pour **clef** la propriété d'où part la flèche,
3) 1 **relation** est la liste construite à partir d'une **clef** (simple ou composée) et des **propriétés liées directement à la totalité de la clef.**

Formes normales.

Définitions.

1ère forme normale (1 FN) : Tous les attributs contiennent une valeur atomique[5].

2ème forme normale (2 FN) : La relation est en 1ère forme normale et tout attribut ne faisant pas partie de la clef ne dépend pas d'une partie de la clef.

3ème forme normale (3 FN) : La relation est en 2ème forme normale et tout attribut ne faisant pas partie de la clef ne dépend pas d'un attribut autre que la clef.

5 Atomique : « Qui ne peut être divisé ».

Exemples.

R1(no fournisseur, no article, adresse, prix)

1ère forme normale : oui, car toutes les propriétés sont atomiques

2ème forme normale : non, car toutes les propriétés ne dépendent pas de la totalité de la clef

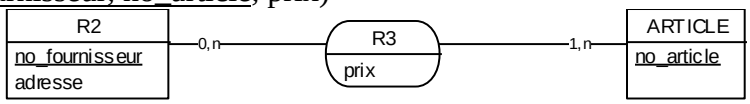
no fournisseur, no article → prix OK

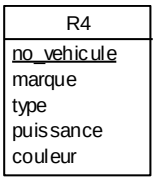
no fournisseur, no article → adresse NON : no_fournisseur suffit pour connaître l'adresse

Correction à apporter : Il faut décomposer R1 en deux relations et introduire une relation dont la clef est no_fournisseur. Cette dernière est probablement incluse dans la relation FOURNISSEUR et disparaîtra lors de l'étape de fusion des relations. On obtient ainsi le schéma suivant.

R2(no_fournisseur, adresse)

R3(no_fournisseur, no_article, prix)





R4(no vehicule, marque, type, puissance, couleur)

1ère forme normale : oui, car toutes les propriétés sont atomiques

2ème forme normale : oui, car toutes les propriétés ne dépendent pas de la totalité de la clef

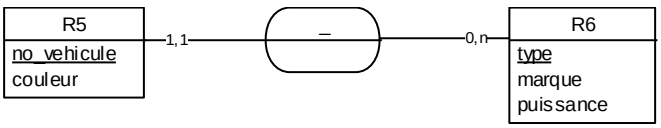
3ème forme normale : non car toutes les propriétés ne dépende pas pleinement de la clef

no_vehicule → couleur , type

type → marque, puissance

R5(no_vehicule, couleur , *type*)

R5(type, marque, puissance)



Chapitre 4. Etude de cas.

Cette étude de cas porte sur l'élaboration d'un modèle logique de données (MLD), elle sert à illustrer quelques concepts et techniques.

Existant.

Objectif.: *"Enregistrer les commandes que reçoit une société."*

Dictionnaire de données brut.

Identifiant	*Définition*
no_client	Numéro de client
nom	Nom du client
adr	Adresse du client
tel	Numéro de téléphone du client
fax	Numéro de fax du client
adr_fac	Adresse de facturation d'une commande
adr	Adresse de livraison d'une commande
no_ref	Numéro de référence interne du produit
nom	Nom interne du produit
qte_stk	Quantité en stock du produit
seuil_stk	Seuil de réapprovisionnement du produit
puht	Prix unitaire hors-taxe du produit
no_lig	Numéro de ligne de commande, identifie une ligne d'une commande.
qte_cmd	Quantité de produit commandé, cette quantité figure sur chaque ligne du bon de commande
no_cmd	Numéro de commande
date_cmd	Date d'enregistrement de la commande, les commandes sont enregistrées dès réception du bon de commande.
prix	Prix unitaire hors-taxe d'un produit

On détecte dans ce dictionnaire la présence d'homonymes[6]:

Nom	Nom du client
Nom	Nom interne du produit
Adr	Adresse du client
Adr	Adresse de livraison d'une commande

… ainsi que la présence de synonymes[7] :

Puht	Prix unitaire hors-taxe du produit
Prix	Prix unitaire hors-taxe d'un produit

[6] Cette détection est facile si l'on opère sur la liste triée selon l'identifiant.

[7] Cette détection peut être plus facile si l'on opère sur la liste triée selon les définitions.

Dictionnaire de données épuré.

Identifiant	*Définition*
adr	Adresse du client
adr_fac	Adresse de facturation d'une commande
adr_liv	Adresse de livraison d'une commande
date_cmd	Date d'enregistrement de la commande, les commandes sont enregistrées dès réception du bon de commande.
fax	Numéro de fax du client
lib	Nom interne du produit
no_client	Numéro de client
no_cmd	Numéro de commande
no-lig	Numéro de ligne de commande, identifie une ligne d'une commande.
no-ref	Numéro de référence interne du produit
nom	Nom du client
puht	Prix unitaire hors-taxe du produit
qte_cmd	Quantité de produit commandé, cette quantité figure sur chaque ligne du bon de commande
qte_stk	Quantité en stock du produit
seuil_stk	Seuil de réapprovisionnement du produit
tel	Numéro de téléphone du client

Modèle Conceptuel de Données.

On identifie clairement les quatre entités :

CLIENT,
PRODUIT,
COMMANDE et
LIGNE.

Etablissons les dépendances fonctionnelles pour ces entités :

CLIENT

```
no_client  →  nom
no_client  →  adr
no_client  →  tel
no_client  →  fax
```

PRODUIT

```
no_ref     →  lib
no_ref     →  qte_stk
no_ref     →  seuil_stk
no_ref     →  puht
```

COMMANDE

```
no_cmd     →  date_cmd
no_cmd     →  adr_fac
no_cmd     →  adr_liv
```

LIGNE

```
no_lig     →  qte_cmd
```

A ce point, il reste à déterminer les liens entre les entités.

CLIENT-COMMANDE, il est clair qu'une COMMANDE est passée par un CLIENT donc le numéro de commande détermine le numéro du client qui a passé la commande :

no_cmd → no_client

COMMANDE-LIGNE, un bon de commande groupe une liste de lignes:

no_cmd → no_lig

PRODUIT-LIGNE, un produit peut être dans les lignes de différentes commandes :

no_ref → no_lig

On traduit l'identifiant no_lig par la concaténation des clefs no_cmd, no_ref, car la quantité commandée est fonction de la COMMANDE et du PRODUIT simultanément :

no_cmd, no_ref → qte_cmd

Figure 6 : Bon de commande, graphe des dépendances fonctionnelles.

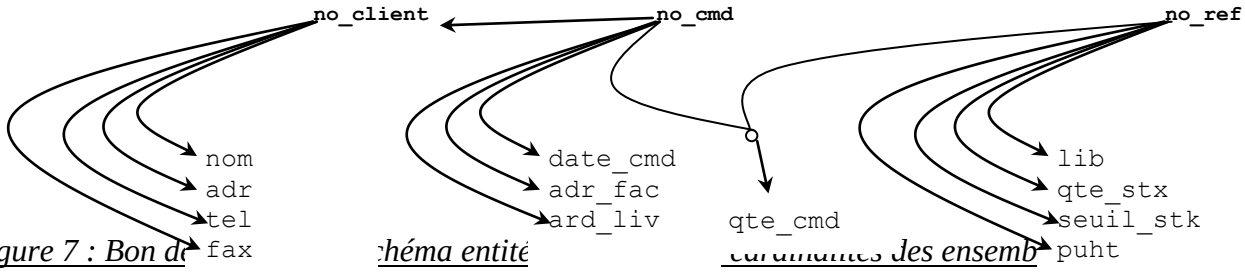
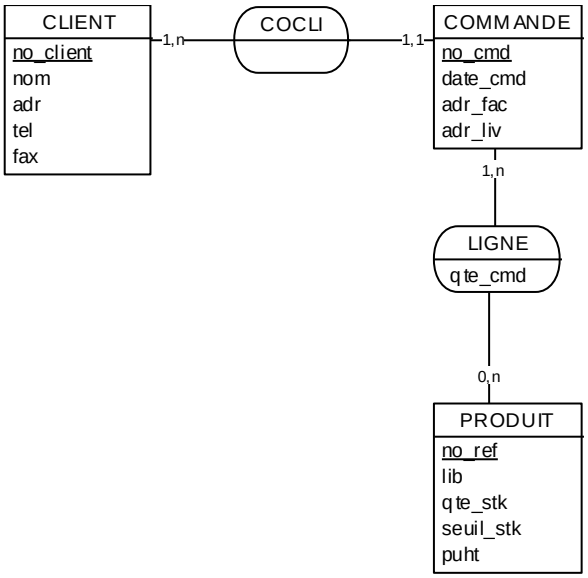


Figure 7 : Bon de chéma entité cardinalités des ensemb



Modèle Logique de Données.

Après transformation on obtient la collection de relations :

CLIENT (**no_client**, nom, adr, tel, fax)
PRODUIT (**no_ref**, lib, qte_stk, seuil_stk, puht)
COMMANDE (**no_cmd**, date_cmd, adr_fac, adr_liv, *no_client*)
LIGNE (**no_cmd, no_ref**, qte_cmd)

Les clefs des relations figurent en **gras** et les clefs étrangères en *italique*.

Modèle Physique de Données.

Table : CLIENT

nom	*type*	*longueur*	*Valeur nulle autorisée*
no_client	alpha numérique	5	NON
nom	alpha numérique	30	NON
adr	alpha numérique	50	OUI
tel	alpha numérique	15	NON
fax	alpha numérique	15	NON

Table : COMMANDE

nom	*type*	*longueur*	*Valeur nulle autorisée*
no_cmd	Alpha numérique	5	NON
date_cmd	date		NON
adr_fac	alpha numérique	50	OUI
adr_liv	alpha numérique	50	OUI
no_client	alpha numérique	5	NON ***clef étrangère***

Table : LIGNE

nom	*type*	*longueur*	*Valeur nulle autorisée*
no_cmd	alpha numérique	5	NON
prod	alpha numérique	5	NON
qte_cmd	entier		NON

Table : PRODUIT

nom	*type*	*longueur*	*Valeur nulle autorisée*
no_ref	alpha numérique	5	NON
lib	alpha numérique	50	NON
qte_stk	entier		NON
seuil_stk	entier		NON
puht	réel	6 dont 4 décimales	NON

Jeu d'essai.

Table CLIENT

no_client	nom	Adr	tel	fax	ville	cp
1	SARL DUPONT	8, rue des allouettes	83.00.00.00	83.00.00.01	NANCY	54000
2	M.MONTJOIE	55, rue des ibellules	83.00.01.00	83.00.01.01	FROUARD	54390

Table PRODUIT

no_ref	lib	qte_stk	seuil_stk	puht
I0001	IMPRIMANTE LASER CANON LPB-81V	10	3	7500
I0002	IMPRIMANTE LASER CANON LPB-4U	20	5	3900
I0003	IMPRIMANTE CANON BJC-600 COULEUR	20	5	3500
I0004	IMPRIMANTE HP DESKJET 500C	20	5	2000
I0005	IMPRIMANTE HP DESKJET 550C	20	5	2700
I0006	IMPRIMANTE HP DESKJET 310	20	5	1500
I0007	IMPRIMANTE HP LASERJET 4L	20	5	3500
I0008	IMPRIMANTE HP DESKJET 510	20	5	1200
I0009	IMPRIMANTE HP DESKJET 1200C	10	3	9000
I0010	TONER EPL LBP LPB-4	50	10	350
I0011	TONER EPL LBP LPB-8	50	10	550
I0012	MEMOIRE 4 Mo LPB 8 IV	20	5	1200
I0013	MEMOIRE 2 Mo LPB 4	20	5	1300
I0014	MEMOIRE 1 Mo HP 4L	20	5	550
I0015	TONER HP 4L	50	10	550
I0016	BAC ENVELOPPES HP 4L	5	3	1800

Table COMMANDE

no_cmd	date	adr_fac	adr_liv	*no_client*
1	10/02/2000			*1*
2	10/02/2000			*2*
3	15/03/2000			*1*
4	16/04/2000			*1*
5	18/04/2000			*2*

Table LIGNE

no_cmd	**no_ref**	qte_cmd
1	**I0001**	2
1	**I0011**	2
2	**I0005**	6
1	**I0012**	1
2	**I0007**	4
2	**I0015**	2
2	**I0016**	2
2	**I0014**	6
3	**I0009**	1
4	**I0015**	2
5	**I0011**	2

Chapitre 5. Algèbre relationnelle.

Les données de cette algèbre sont les relations telles qu'elles ont été décrites précédemment. Les opérations de bases[8] sont au nombre de trois :

- la projection,
- la restriction,
- le produit cartésien.

La projection.

Cette opération permet d'extraire une ou plusieurs propriétés d'une relation, le résultat d'une projection est une relation. Elle permet aussi de calculer sur des propriétés à travers une expression arithmétique

P (<relation>) → <relation>
<liste de propriétés>

Tableau 2: Expressions : opérateurs arithmétiques.

opérateur	***opération***
+	Addition
-	Soustraction ou moins unaire
*	Produit
/	Rapport
MOD	Modulo

- *Liste des libellés de produits.*

$R_1 \leftarrow P_{lib}$ (PRODUIT)

La restriction.

Cette opération permet d'extraire un ou plusieurs n-uples d'une relation, le résultat d'une restriction est une relation. Les n-uples extraits vérifient un ou plusieurs critères de restriction.

R (<relation>) → <relation>
<liste de critères>

Un critère est défini à l'aide d'une comparaison à résultat booléen. Chaque comparaison porte sur une propriété de la relation par rapport à une référence. La référence est une valeur constante, ou une valeur extraite d'une autre relation.

<opérande_1> <opérateur> <opérande_2> → <booléen>

[8] Noyau générateur – opérations génératrices

Tableau 3: Opérateurs de comparaison.

opérateur	*critère déterminé*
<A> = <B>	<A> est égal <B>
<A> ≠ <B>	<A> est différent de <B>
<A> < <B>	<A> est inférieur à <B>
<A> ≤ <B>	<A> est inférieur ou égal à <B>
<A> > <B>	<A> est supérieur à <B>
<A> ≥ <B>	<A> est supérieur ou égal à <B>
<A> ∈ <B>	<A> appartient à l' ensemble <B>
<A> ∉ <B>	<A> n'appartient pas à l'ensemble <B>

Comme il est d'usage, les résultats booléens de critères peuvent être combinés au moyens des opérateurs booléens habituels : NON, OU, ET, pour définir un critère composé.

- *Liste des produits dont le prix unitaire hors taxe est inférieur à 1000.*

$R_2 \leftarrow R\ (PRODUIT)$
puht < 1000

- *Liste des produits dont le prix unitaire hors taxe est supérieur à 3000 dont le seuil de d'approvisionnement est supérieur ou égal à 5.*

$R_2 \leftarrow R\ (PRODUIT)$
puht > 3000 ET seuil_stk ≥ 5

Le produit cartésien.

Cette opération permet de composer une relation à partir de deux relations. Elle consiste à composer chaque nuple d'une relation avec chaque nuple de l'autre relation.

<relation1> ⊗ <relation2> → <relation>

- *Produit cartésien des relations COMMANDE et CLIENT :*

R3 ← COMMANDE ⊗ CLIENT

☞ Le produit cartésien ne s'utilise généralement pas en tant que tel. Cet opérateur est utilisé dans la composition d'une autre opération, la jointure (voir paragraphe suivant).

Les jointures.

La jointure est une restriction d'un produit cartésien, le critère de restriction porte sur la clef commune aux deux relations du produit cartésien. Le résultat d'une jointure est une relation sur les propriétés de différentes tables.

Les n-uplets vérifient un critère de restriction.

J (<relation1> ⊗ <relation2>) → <relation>
<critère>

Le type de jointure dépend de l'opérateur intervenant dans le critère ainsi que les relations mises en œuvre :

Tableau 4: Types de jointures.

type de jointure	*nombre de relations différentes*	*opérateur(s)*
équi-jointure	plusieurs	= ∈
théta-jointure	plusieurs	≠ < ≤ > ≥ ∉
auto-jointure	une	

Jointure COMMANDE et CLIENT sur numéro de client (équi-jointure) :

R_4 ← J (COMMANDE ⊗ CLIENT)
commande.no_client = client.no_client

écriture simplifiée

R_5 ← J (COMMANDE ⊗ CLIENT)
no_client

Composition d'opérations relationnelles.

Chaque opération relationnelle produit un résultat de type relation. Il est donc possible de transmettre le résultat d'une opération en paramètre d'une autre opération et de combiner ainsi des projections, des restrictions et des jointures.

- *Noms et prix des produits dont le prix unitaire hors taxe est inférieur à 1000.*

$R_{6'}$ ← R (PRODUIT)
puht <1000

$R_{6''}$ ← P ($R_{6'}$)
lib,
puht

autres écritures :

$R_{7'} \leftarrow P_{\text{lib, puht}}(\text{PRODUIT})$

$R_{7''} \leftarrow R_{\text{puht}<1000}(R_{7'})$

$R_8 \leftarrow P_{\text{lib, puht}}(R_{\text{puht}<1000}(\text{PRODUIT}))$

$R_9 \leftarrow R_{\text{puht}<1000}(P_{\text{lib, puht}}(\text{PRODUIT}))$

- *Liste des produits (libellé) de la commande numéro 1.*

$R_{10'} \leftarrow J_{\text{no\_ref}}(\text{PRODUIT} \otimes \text{LIGNE})$

$R_{10''} \leftarrow R_{\text{no\_cmd} = 1}(R_{10'})$

$R_{10'''} \leftarrow P_{\text{lib}}(R_{10''})$

autres écritures possibles :

$R_{11'} \leftarrow R_{\text{no\_cmd} = 1}(\text{LIGNE})$

$R_{11''} \leftarrow J_{\text{no\_ref}}(\text{PRODUIT} \otimes R_{11'})$

$R_{11'''} \leftarrow P_{\text{lib}}(R_{11''})$

$R_{12} \leftarrow P_{\text{lib no\_ref}}(J_{\text{no\_cmd} = 1}(\text{PRODUIT} \otimes R(\text{LIGNE})))$

Les tris.

Pour des raisons de commodité il est parfois utile de trier les données à l'édition (édition de catalogue, analyse, etc…).

Un tri détermine un ordre entre les lignes d'une table, il utilise un critère de comparaison des lignes sur une colonne.

Lorsque le tri porte sur plusieurs colonnes on le nomme tri multi-critères. Dans ce cas, l'ordre des critères n'est pas indifférent.

- *Liste des produits triée par prix unitaire hors taxe croissant et quantité en stock décroissant.*

$R_{13} \leftarrow$ PRODUIT [triée par puht croissant et qte_stk décroissant]

- *Liste des produits triée par quantité en stock décroissant et prix unitaire hors taxe croissant.*

R_{14} ← PRODUIT [triée par qte_stk décroissant et puht croissant]

Les fonctions de groupe.

Ces fonctions permettent d'effectuer des opérations sur les valeurs d'une colonne.

R_{15} ← P (<TABLE>)
<fonction> (<paramètre>)

Dans le tableau suivant, un paramètre de type "expression" dénote la possibilité de transmettre soit un nom de colonne, soit une expression arithmétique.

Il est à noter qu'une expression peut être aussi utilisée dans une projection.

Tableau 5: Fonctions de groupe

Nom	*Objet*	*Paramètre*
COMPTAGE	Compte les lignes sélectionnées	* ou nom de colonne
SOMME	Cumule de valeur	expression
MINIMUM	Détermine la valeur minimum	expression
MAXIMUM	Détermine la valeur maximum	expression
MOYENNE	Calcule la moyenne	expression
VARIANCE	Calcule la variance	expression
ECART- TYPE	Calcule l'écart-type (déviation standard)	expression

- *Nombre de produits au catalogue.*

R_{16} ← P (PRODUIT)
COMPTAGE (no_ref)

ou encore

R_{17} ← P (PRODUIT)
COMPTAGE (*)

- *Prix unitaires : minimum, maximum et moyen des produits.*

R_{18} ← P (PRODUIT)
MINIMUM (puht),
MAXIMUM (puht),
MOYENNE (puht)

- *Ratios seuil de réapprovisionnement sur quantité en stock par référence.*

R_{19} ← P (PRODUIT)
no_ref,
seuil_stk / qte_stk

- *Ratios seuil de réapprovisionnement sur quantité en stock minimum, maximum et moyen.*

R_{20} ← P (PRODUIT)
MINIMUM (seuil_stk / qte_stk),
MAXIMUM (seuil_stk / qte_stk),
MOYENNE (seuil_stk / qte_stk)

Les regroupements.

Il est utile de pouvoir grouper des lignes afin d'opérer des calculs sur des sous-groupes. La liste de sous-groupes peut être restreinte par un critère de validité de sous-groupe.

- *Nombre de produits commandés par commande :*

R_{22} ← P (LIGNE)
no_cmd
count(no_ref)

- *Nombre de produits commandés par commande de plus de 2 produits :*

R_{22} ← P (LIGNE) [**par commande de plus de 2 produits]**
no_cmd
count(no_ref)

Union.

C'est l'opération ensembliste de réunion appliquée aux relations.

<relation1> ∪ <relation2> → <relation>

- *Liste sans doublon des références produits des commande 1 et 2.*

R_{23} ← R (LIGNE)
no_cmd = 1

R_{24} ← R (LIGNE)
no_cmd = 2

R_{25} ← P (R_{23})
no_ref

R_{26} ← P (R_{24})
no_ref

R_{25} ← $R_{25} \cup R_{26}$

Intersection.

C'est l'opération ensembliste d'intersection appliquée aux relations.

<relation1> ∩ <relation2> → <relation>

- *Liste sans doublon des références produits des commande 1 et 2.*

$R_{23} \leftarrow R\ (\text{LIGNE})$ no_cmd = 1

$R_{24} \leftarrow R\ (\text{LIGNE})$ no_cmd = 2

$R_{25} \leftarrow R_{23} \cap R_{24}$

$R_{26} \leftarrow P\ (R_{25})$ no_ref

Soustraction.

C'est l'opération ensembliste de soustraction appliquée aux relations. Cette opération n'est pas commutative.

<relation1> - <relation2> → <relation>

- *Liste des références des produits de la commande 1 sauf ceux apparaissant aussi dans la commande 2.*

$R_{27} \leftarrow R\ (\text{LIGNE})$ no_cmd = 1

$R_{28} \leftarrow R\ (\text{LIGNE})$ no_cmd = 2

$R_{29} \leftarrow P\ (R_{27})$ no_ref

$R_{30} \leftarrow P\ (R_{28})$ no_ref

$R_{31} \leftarrow R_{29} - R_{30}$

Division.

C'est l'opération ensembliste de division appliquée aux relations.

<relation1> ÷ <relation2> → <relation>

Application: "*La liste de clients qui ont commandé tous les produits du catalogue*"

Les aliases.

Un alias permet de renommer une propriété lors d'une projection.

- *Montant hors taxe des lignes de la commande numéro 1.*

R_{32} ← J (PRODUIT ⊗ LIGNE)
no_ref

R_{33} ← R (R_{32})
no_cmd = 1

R_{34} ← P (R_{33})
`(puht * qte_cmd) : montant_ht`

Un alias permet de renommer une relation, par exemple si l'on explicite le critère de jointure de R_{32} , l'écriture devient:

R_{35} ← J (PRODUIT ⊗ LIGNE)
PRODUIT.no_ref = LIGNE.no_ref

…

Un alias permet d'alléger l'écriture :

R_{36} ← J (PRODUIT : P ⊗ LIGNE : L)
P.no_ref = L.no_ref

…

Pour une auto-jointure on utilisera 2 synonyme :

R_{37} ← J (LIGNE : L1 ⊗ LIGNE : L2)
L1.no_ref = L2.no_ref

…

Chapitre 6. De l'algèbre au langage SQL.

L'implantation de la collection de relation est un ensemble de tables.
Une relation devient une table.
Chaque propriété d'une relation devient une colonne d'une table.
Une instance de relation est enregistrée dans une ligne d'une table.

Le langage SQL[9] est le plus répandu dans l'univers des SGBD[10], il présente l'avantage d'être normalisé[11] ce qui amène une bonne portabilité des programmes.

L'ordre **SELECT** permet de réaliser les opérations de base sur les tables :

- Projection,
- Restriction,
- Jointure,

il permet aussi d'opérer des tris, des regroupements et des calculs sur les colonnes projetée.

Les paragraphes suivants décrivent la syntaxe de SELECT. Les règles de syntaxe sont représentées de la manière suivante :

Figure 8 : Syntaxe de présentation du langage SQL.

NOM	désigne le mot "NOM" qui doit figurer en toutes lettres
<nom>	désigne une partie variable <nom> à laquelle il faudra substituer un mot
a | b	désigne une alternative entre "a" et "b" qui est marquée par le symbole |
[OPTION]	désigne une partie optionnelle : "OPTION" dans une commande

Le terme liste désigne une suite d'éléments séparés par des virgules.

Figure 9 : Syntaxe de la commande SELECT.

```
SELECT ALL | DISTINCT * | table. * | expr alias,…
FROM table alias,…
WHERE condition
GROUP BY expr, expr…
HAVING condition
ORDER BY expr [ ASC | DESC ],…
UNION | INTERSECT | MINUS
```

Projection.

```
SELECT  * |<liste de colonnes>
FROM    <table> ;
```

9 "Structured Query Langage"
10 SYBASE, ORACLE, POSTGRES, MySQL, SQL Server, Interbase,ACCESS, …
11 ANSI X3 135.

- Liste des propriétés des produits.

```
SELECT *
FROM PRODUIT ;
```

no_ref	lib	qte_stk	seuil_stk	puht
I0001	IMPRIMANTE LASER CANON LPB-81V	10	3	7500
I0002	IMPRIMANTE LASER CANON LPB-4U	20	5	3900
I0003	IMPRIMANTE CANON BJC-600 COULEUR	20	5	3500
I0004	IMPRIMANTE HP DESKJET 500C	20	5	2000
I0005	IMPRIMANTE HP DESKJET 550C	20	5	2700
I0006	IMPRIMANTE HP DESKJET 310	20	5	1500
I0007	IMPRIMANTE HP LASERJET 4L	20	5	3500
I0008	IMPRIMANTE HP DESKJET 510	20	5	1200
I0009	IMPRIMANTE HP DESKJET 1200C	10	3	9000
I0010	TONER EPL LBP LPB-4	50	10	350
I0011	TONER EPL LBP LPB-8	50	10	550
I0012	MEMOIRE 4 Mo LPB 8 IV	20	5	1200
I0013	MEMOIRE 2 Mo LPB 4	20	5	1300
I0014	MEMOIRE 1 Mo HP 4L	20	5	550
I0015	TONER HP 4L	50	10	550
I0016	BAC ENVELOPPES HP 4L	5	3	1800

- Liste des libellés de produits.

```
SELECT lib
FROM PRODUIT ;
```

lib
IMPRIMANTE LASER CANON LPB-81V
IMPRIMANTE LASER CANON LPB-4U
IMPRIMANTE CANON BJC-600 COULEUR
IMPRIMANTE HP DESKJET 500C
IMPRIMANTE HP DESKJET 550C
IMPRIMANTE HP DESKJET 310
IMPRIMANTE HP LASERJET 4L
IMPRIMANTE HP DESKJET 510
IMPRIMANTE HP DESKJET 1200C
TONER EPL LBP-4
TONER EPL LBP LPB-8
MEMOIRE 2 Mo LPB 4
MEMOIRE 1 Mo HP 4L
TONER HP 4L
BAC ENVELOPPES HP 4L

Restriction.

```
SELECT   * |<liste de colonnes>
FROM     <table>
WHERE    <critère>   ;
```

Le critère est défini par une expression booléenne faisant intervenir des fonctions et opérations de comparaison. L'expression peut-être composée de comparaisons reliées par des opérations booléennes.

Tableau 6: Opérateurs de comparaison.

critère déterminé	*Opérateur*
<A> est égal <B>	<A> = <B>
<A> est différent de <B>	<A> != <B>
<A> est inférieur à <B>	<A> < <B>
<A> est inférieur ou égal à <B>	<A> <= <B>
<A> est supérieur à <B>	<A> > <B>
<A> est supérieur ou égal à <B>	<A> >= <B>
<A> appartient à l'ensemble <B>	<A> IN <B>
<A> n'appartient pas à l'ensemble <B>	<A> NOT IN <B>

Les opérateurs OR, AND et NOT et les parenthèses permettent de construire des critères composés.

L'opérateurs **LIKE** et les méta-caractères[12] % (signe de pourcentage) et _ (soulignement) permettent de définir des critères génériques. Dans certains interprètes SQL, tel celui de MS Access, le caractère * remplace le caractère %. Les méta-caractères permettent de définir des expressions régulières.

- *Liste des produits dont le prix unitaire hors taxe est inférieur à 1000.*

```
SELECT *
FROM      PRODUIT
WHERE  puht < 1000 ;
```

no_ref	lib	qte_stk	seuil_stk	puht
I0010	TONER EPL LBP LPB-4	50	10	350
I0011	TONER EPL LBP LPB-8	50	10	550
I0014	MEMOIRE 1 Mo HP 4L	20	5	550
I0015	TONER HP 4L	50	10	550

- *Liste des produits dont le prix unitaire hors taxe est supérieur à 3000 dont le seuil de d'approvisionnement est supérieur ou égal à 5.*

```
SELECT *
FROM  PRODUIT
WHERE       puht > 3000 AND seuil_stk >= 5 ;
```

no_ref	lib	qte_stk	seuil_stk	puht
I0002	IMPRIMANTE LASER CANON LPB-4U	20	5	3900
I0003	IMPRIMANTE CANON BJC-600 COULEUR	20	5	3500
I0007	IMPRIMANTE HP LASERJET 4L	20	5	3500

- *Liste des produits dont le libellé contient "LASER".*

```
SELECT *
FROM  PRODUIT
WHERE       lib LIKE "%LASER%" ;
```

no_ref	lib	qte_stk	seuil_stk	puht
I0001	IMPRIMANTE LASER CANON LPB-81V	10	3	7500
I0002	IMPRIMANTE LASER CANON LPB-4U	20	5	3900
I0007	IMPRIMANTE HP LASERJET 4L	20	5	3500

[12] *méta-caractère* : caractère générateur de chaîne de caractères.

Produit cartésien.

```
SELECT   * |<liste de colonnes>
FROM     <liste de table>
```

- *Produit cartésien COMMANDE CLIENT*

```
SELECT *
FROM   COMMANDE, CLIENT ;
```

no_cmd	date	adr-fac	adr_liv	T_COMMANDE.no_client	**T_CLIENT.no_client**	nom	adr	tel	fax	ville	cp
1	10/02/2000			1	**1**	SARL DUPONT	8, rue des alouettes	83.00.00.00	83.00.00.01	NANCY	54 000
2	10/02/2000			2	**1**	SARL DUPONT	8, rue des alouettes	83.00.00.00	83.00.00.01	NANCY	54 000
3	15/03/2000			1	**1**	SARL DUPONT	8, rue des alouettes	83.00.00.00	83.00.00.01	NANCY	54 000
4	16/04/2000			1	**1**	SARL DUPONT	8, rue des alouettes	83.00.00.00	83.00.00.01	NANCY	54 000
5	18/04/2000			2	**1**	SARL DUPONT	8, rue des alouettes	83.00.00.00	83.00.00.01	NANCY	54 000
1	10/02/2000			1	**2**	M.MONTJOIE	55, rue des libellules	83.00.01.00	83.00.01.01	FROUARD	54 390
2	10/02/2000			2	**2**	M.MONTJOIE	55, rue des libellules	83.00.01.00	83.00.01.01	FROUARD	54 390
3	15/03/2000			1	**2**	M.MONTJOIE	55, rue des libellules	83.00.01.00	83.00.01.01	FROUARD	54 390
4	16/04/2000			1	**2**	M.MONTJOIE	55, rue des libellules	83.00.01.00	83.00.01.01	FROUARD	54 390
5	18/04/2000			2	**2**	M.MONTJOIE	55, rue des libellules	83.00.01.00	83.00.01.01	FROUARD	54 390

Jointure.

```
SELECT          * |<liste de colonnes>
FROM            <liste de table>
WHERE           <critères>              ;
```

- *Jointure COMMANDE et CLIENT sur numéro de client (équi-jointure) :*

```
SELECT   *
FROM   COMMANDE, CLIENT
WHERE COMMANDE.no_client = CLIENT.no_client ;
```

no_cmd	date	adr-fac	adr_liv	T_COMMANDE. no_client	**T_CLIENT. no_client**	nom	adr	tel	Fax	ville	cp
1	10/02/2000			*1*	**1**	SARL DUPONT	8, rue des alouettes	83.00.00.00	83.00.00.01	NANCY	54 000
3	15/03/2000			*1*	**1**	SARL DUPONT	8, rue des	83.00.00.00	83.00.00.01	NANCY	54 000
4	16/04/2000			*1*	**1**	SARL DUPONT	8, rue des alouettes	83.00.00.00	83.00.00.01	NANCY	54 000
2	10/02/2000			*2*	**2**	M.MONTJOIE	55, rue des libellules	83.00.01.00	83.00.01.01	FROUARD	54 390
5	18/04/2000			*2*	**2**	M.MONTJOIE	55, rue des libellules	83.00.00.00	83.00.00.01	FROUARD	54 390

Composition d'opérations.

- *Noms et prix des produits dont le prix unitaire hors taxe est inférieur à 1000.*

```
SELECT  lib, puht
FROM     PRODUIT
WHERE      puht<1000 ;
```

Lib	puht
TONER EPL LBP LPB-4	350
TONER EPL LBP LPB-8	550
MEMOIRE 1 Mo HP 4L	550
TONER HP 4L	550

- *Liste des produits (libellé) de la commande numéro 1.*

```
SELECT PRODUIT.lib
FROM    PRODUIT, LIGNE
WHERE PRODUIT.no_ref = LIGNE.no_ref AND LIGNE.no_cmd = 1;
```

Lib
IMPRIMANTE LASER CANON LPB-8IV
TONER EPL LBP LPB-8
MEMOIRE 4 Mo LPB 8 IV

Projection avec tri.

```
SELECT  * | <liste de colonnes>
FROM     <liste de table>
ORDER BY <liste de critères>          ;
```

Avec pour 1 critère

```
        <critère>   <nom de colonne> [ASC | DESC ]
```

Dans les exemples suivants, les sous-groupes pour lesquels le second critère est employé sont encadrés par une bordure à trait double; le chiffre en tête de colonne indique le numéro d'ordre du critère.

- *Liste de produits triés par prix unitaire hors taxe croissant et quantité en stock décroissant.*

```
SELECT  *
FROM      PRODUIT
ORDER BY puht ASC, qte_stk DESC ;
```

		2 ↓		1 ↓
no_ref	lib	qte_stk	seuil_stk	puht
I0010	TONER EPL LBP LPB-4	50	10	350
I0015	TONER HP 4L	50	10	550
I0011	TONER EPL LBP LPB-8	50	10	550
I0014	MEMOIRE 1Mo HP 4L	20	5	550
I0008	IMPRIMANTE HP DESKJET 510	20	5	1200
I0012	MEMOIRE 4 Mo LPB 8 IV	20	5	1200
I0013	MEMOIRE 2 Mo LPB 4L	20	5	1300
I0006	IMPRIMANTE HP DESKJET 310	20	5	1500
I0016	BAC ENVELOPPES HP 4L	5	3	1800
I0004	IMPRIMANTE HP DESKJET 500C	20	5	2000
I0005	IMPRIMANTE HP DESKJET 550C	20	5	2700
I0003	IMPRIMANTE CANON BJC-600 COULEUR	20	5	3500
I0007	IMPRIMANTE HP LASERJET 4L	20	5	3500
I0002	IMPRIMANTE LASER CANON LPB-4U	20	5	3900
I0001	IMPRIMANTE LASER CANON LPB-8IV	10	3	7500
I0009	IMPRIMANTE HP DESKJET 1200C	10	3	9000

- *Liste de produits triés par quantité en stock décroissant et prix unitaire hors taxe croissant.*

```
SELECT *
FROM     PRODUIT
ORDER BY qte_stk DESC, puht ASC ;
```

		1 ↓		2 ↓
no_ref	lib	qte_stk	seuil_stk	puht
I0010	TONER EPL LBP LPB-4	50	10	350
I0015	TONER HP 4L	50	10	550
I0011	TONER EPL LBP LPB-8	50	10	550
I0014	MEMOIRE 1Mo HP 4L	20	5	550
I0008	IMPRIMANTE HP DESKJET 510	20	5	1200
I0012	MEMOIRE 4 Mo LPB 8 IV	20	5	1200
I0013	MEMOIRE 2 Mo LPB 4L	20	5	1300
I0006	IMPRIMANTE HP DESKJET 310	20	5	1500
I0004	IMPRIMANTE HP DESKJET 500C	20	5	2000
I0005	IMPRIMANTE HP DESKJET 550C	20	5	2700
I0003	IMPRIMANTE CANON BJC-600 COULEUR	20	5	3500
I0007	IMPRIMANTE HP LASERJET 4L	20	5	3500
I0002	IMPRIMANTE LASER CANON LPB-4U	20	5	3900
I0001	IMPRIMANTE LASER CANON LPB-8IV	10	3	7500
I0009	IMPRIMANTE HP DESKJET 1200C	10	3	9000
I0016	BAC ENVELOPPES HP 4L	5	3	1800

Les fonctions de groupe et les regroupements.

```
SELECT <fonction> ( * | <expression> )
FROM       <TABLE> ;
```

Tableau 7: Fonctions de groupe.

Nom	*Objet*	*Paramètre*
COUNT	COMPTAGE	* ou nom de colonne
SUM	SOMME	expression
MIN	MINIMUM	expression
MAX	MAXIMUM	expression
AVG	MOYENNE	expression
VAR	VARIANCE	expression
STDDEV	ECART TYPE	expression

- *Nombre de produits au catalogue.*

```
SELECT     COUNT (no_ref)     ☞
FROM        PRODUIT ;
```

COUNT (no_ref)
16

ou encore

```
SELECT     COUNT (*)          ☞
FROM       PRODUIT ;
```

COUNT (*)
16

- *Prix unitaires : minimum, maximum et moyen des produits.*

```
SELECT MIN ( puht ), MAX ( puht ), AVG ( puht )
FROM     PRODUIT ;
```

MIN (puht)	MAX (puht)	AVG (puht)
350	9000	2568,75

- *Ratios seuil de réapprovisionnement sur quantité en stock par référence.*

```
SELECT no_ref, seuil_stk / qte_stk
FROM     PRODUIT ;
```

no_ref	seuil_stk / qte_stk
I0001	0,3
I0002	0,25
I0003	0,25
I0004	0,25
I0005	0,25
I0006	0,25
I0007	0,25
I0008	0,25
I0009	0,3
I0010	0,2
I0011	0,2
I0012	0,25
I0013	0,25
I0014	0,25
I0015	0,2
I0016	0,6

- *Ratios minimum, maximum et moyen de seuil de réapprovisionnement sur quantité en stock.*

```
SELECT MIN (seuil_stk / qte_stk ), MAX (seuil_stk / qte_stk ),
            AVG (seuil_stk / qte_stk )
FROM     PRODUIT ;
```

MIN (seuil_stk / qte_stk)	MAX (seuil_stk / qte_stk)	AVG (seuil_stk / qte_stk)
0,2	0,6	0,26875

Les regroupements.

```
SELECT <liste de colonnes ou expressions de groupe>
FROM    <TABLE>
GROUP BY  <liste de colonnes>
HAVING    <liste de critères de sous groupe> ;
```

- *Nombre de produits commandés par commande.*

```
SELECT      no_cmd, COUNT (no_ref)
FROM        LIGNE
GROUP BY    no_cmd;
```

no_cmd	COUNT (no_ref)
1	3
2	5
3	1
4	1
5	1

- *Nombre de produits commandés par commande de plus de 2 produits.*

```
SELECT      no_cmd, COUNT (no_ref)
FROM        LIGNE
GROUP BY    no_cmd
HAVING      COUNT (no_ref) > 2;
```

no_cmd	COUNT (no_ref)
1	3
2	5

Union.

C'est l'opération ensembliste de réunion appliquée aux relations.

```
SELECT <liste de colonnes ou expressions de groupe>
FROM    <TABLE>
UNION
SELECT <liste de colonnes ou expressions de groupe>
FROM    <TABLE>
```

- *Liste sans doublon des références produits des commandes 1 ou 2.*

```
SELECT ligne.no_ref
FROM ligne
WHERE ligne.no_cmd = 1
union
SELECT ligne.no_ref
FROM ligne
WHERE ligne.no_cmd = 2 ;
```

no_ref
I0001
I0005
I0007
I0011
I0012
I0014
I0015
I0016

Il faut noter cette opération n'est pas identique à la restriction basée sur une expression utilisant un OU dans la clause where d'une seule requête ; dans ce dernier cas, des doublons apparaissent et le tri n'a pas lieu:

no_ref
I0001
I0011
I0005
I0012
I0007
I0015
I0016
I0014
I0005

Intersection.

C'est l'opération ensembliste d'intersection appliquée aux relations.

```
SELECT <liste de colonnes ou expressions de groupe>
FROM    <TABLE>
INTERSECT
SELECT <liste de colonnes ou expressions de groupe>
FROM    <TABLE>
```

- *Liste des références produits apparaissant dans la commandes 1 et aussi dans la commande 2.*

```
SELECT ligne.no_ref
FROM ligne
WHERE ligne.no_cmd = 1
INTERSECT
SELECT ligne.no_ref
FROM ligne
WHERE ligne.no_cmd = 2 ;
```

Soustraction.

C'est l'opération ensembliste de soustraction appliquée aux relations. Cette opération n'est pas commutative.

```
SELECT <liste de colonnes ou expressions de groupe>
FROM    <TABLE>
MINUS
SELECT <liste de colonnes ou expressions de groupe>
FROM    <TABLE>
```

- *Liste des références produits de la commande 1 sauf ceux apparaissant aussi dans la commande numéro 2.*

```
SELECT ligne.no_ref
FROM ligne
WHERE ligne.no_cmd = 1
minus
SELECT ligne.no_ref
FROM ligne
WHERE ligne.no_cmd = 2 ;
```

Si l'opérateur « MINUS » n'est pas implanté dans l'interprète SQL, on obtient le même résultat en imbriquant deux requêtes :

```
SELECT ligne.no_ref
FROM ligne
WHERE ligne.no_cmd=1 AND ligne.no_ref NOT IN
      (
            SELECT ligne.no_ref
            FROM ligne
            WHERE ligne.no_cmd=2
      );
```

Les aliases.

- *Montant hors taxe des lignes de la commande numéro 1.*

```
SELECT (PRODUIT.puht * LIGNE.qte_cmd) AS montant_ht
FROM    PRODUIT, LIGNE
WHERE PRODUIT.no_ref = LIGNE.no_ref AND LIGNE.no_cmd = 1;
```

Chapitre 7. Triggers et procédures stockées

Les triggers et les procédures stockées permettent de placer des instructions SQL dans la base de données pour qu'elles soient utilisables dans tous les gabarits (templates) d'applications. Elles permettent de renforcer la sécurité, l'efficacité et la standardisation des bases de données.

Qu'est-ce qu'un trigger?

Un trigger est un segment de code SQL associé à une table et stocké dans une base de données. Ce code est appelé automatiquement chaque fois qu'un utilisateur tente de modifier des données dans la table associée au trigger à l'aide d'une commande d'insertion, de suppression ou de mise à jour. Un trigger peut être considéré comme une série d'actions et de vérifications requises par une commande afin qu'une opération soit correctement effectuée.

Exemples

Sur une gestion des bons de commande : on enregistre

- une adresse de client, obligatoire ;
- une adresse de facturation, obligatoire ;

on peut décider que si la deuxième n'est pas valorisée lors d'une insertion il faut l'affecter avec la première.

Intégrité référentielle

Vous pouvez utiliser des triggers pour mettre en oeuvre l'intégrité référentielle, lorsque les contraintes déclaratives ne suffisent pas. Vous pouvez également utiliser les triggers pour mettre en œuvre les séquences pour les colonnes.

Procédures stockées

Définition de procédures stockées et de fonctions

Une procédure stockée est une collection compilée d'instructions SQL stockées sous un nom et traitées comme une unité. Les procédures stockées sont conservées dans une base de données ; elles peuvent être exécutées via un appel émis par une application et permettent l'utilisation de variables déclarées par l'utilisateur, l'exécution conditionnelle et d'autres fonctionnalités de programmation.

L'utilisation de procédures stockées peut s'avérer utile pour contrôle l'accès aux données (les utilisateurs peuvent saisir ou modifier des données mais pas écrire de procédures), pour préserver l'intégrité des données (les informations sont entrées de façon cohérente) et pour améliorer la productivité (les instructions incluses dans une procédure stockée sont écrites une seule fois et réutilisées).

Une fonction utilisateur est une forme de procédure qui renvoie une valeur à l'environnement appelant afin que ce dernier l'utilise dans des requêtes et autres instructions SQL.

Chapitre 8. Exploitation avec JAVA - Connecteur JDBC

Installation

Pilotes

Les pilotes JDBC pour postgreSQL sont disponibles à l'adresse:
http://jdbc.postgresql.org/download.html
Il est aussi possible de les obtenir en compilant postgreSQL avec l'obtion
`--with-java`
dans ce cas le pilote s'appellera postgresql.jar

Mise à jour du CLASSPATH

Il faut ajouter le répertoire où est le pilote (en .jar) dans la variable CLASSPATH pour qu'il soit pris en compte par la suite.

Préparation de la base

Il faut activer les socket ip ; dans le fichier postgresql.conf mettre :
`tcpip_socket = true`
Il faut ouvrir le droit d'accès à la base; dans le fichier pg_hba.conf mettre :

Utilisation des pilotes dans une classe Java.

Importation de JDBC

Pour utiliser les classes JDBC (Java Data Base Classes) il est nécessaire d'importer le package sql en mettant en tête de classe :
import java.sql.*;

Chargement des pilotes

Le chargement des pilotes se fait de la façon suivante :

```
Class.forName("org.postgresql.Driver");
```

Ouvrir une connexion à la base.

La connexion à la base se fait par une URL qui peut prendre l'une des formes suivantes:

```
jdbc:postgresql:nom_de_la_base
jdbc:postgresql://adresse_serveur/nom_de_la_base
jdbc:postgresql://adresse_serveur:port/nom_de_la_base
```

La connexion est établie par la méthode :
`DriverManager.getConnection`
On obtient :

URL url = new URL("jdbc:postgresql://194.199.145.22:5432/cinema");
Connection db = DriverManager.getConnection(url, utilisateur, mdp);

Fermer une connexion à la base.

La fermeture se fait par la méthode :

```
db.close();
```

Exécuter une requête (Query)

Une requête s'effectue par la méthode executeQuery() de l'objet Statement. Le résultat est récupéré dans un objet de type ResultSet.

```
import java.sql.*;
public class bddLecture{
	String chemin = "jdbc:postgresql://199.198.55.21:5432/cinema";
	public static void main(argv[] String){
		try {
			Class.forName("org.postgresql.Driver");
		} // try
		catch (ClassNotFoundException e)
		{
			System.out.println("Echec chargement des pilotes");
		} //catch
		try{
			Connection con = DriverManager.getConnection(chemin,
		"utilisateur", "mdp");
			String cmd = "SELECT * FROM acteurs_vivants "
			Statement st = con.createStatement();
			ResultSet rs = st.executeQuery(cmd);
			while (rs.next()) {
			//à chaque boucle, on lit une nouvelle ligne du résultat.
			System.out.println(rs.getString(1)+" | "+rs.getString(2)+" |
			"+rs.getString(3)+" | "+rs.getString(4)+" |
			"+rs.getString(5)+" | "+rs.getString(6));
			// rs.getString(col), où col est le numéro de la colonne dans
			// la table acteurs_vivants,
			// permet d'obtenir une colone de la ligne actuelle sous forme
		de String.
			}//while
			rs.close();
			st.close();
			con.close();
		} // try
		catch (java.sql.SQLException e)
		{
			System.out.println("Echec connection : "+e);
		} // catch
	} // main
} // class
```

Effectuer une mise à jour (Update)

La mise à jour s'effectue par la méthode executeUpdate() de l'objet preparedStatement.

```
import java.sql.*;
public class bddEcriture{
	String chemin = "jdbc:postgresql://199.198.55.21:5432/cinema";
	public static void main(argv[] String){
		//chargement de pilotes
		try {
			Class.forName("org.postgresql.Driver");
		}
		catch (ClassNotFoundException e)
		{
			System.out.println("Echec chargement des pilotes");
		}
		try{
			Connection con = DriverManager.getConnection(chemin,
		"utilisateur", "mdp");
			String cmd = "INSERT INTO acteurs_vivants (
			'','FR','Duris','Romain','1974-05-28','');"
			PreparedStatement st = con.prepareStatement(cmd);
			int rows = st.executeUpdate();
			st.close();
			con.close();
		} // try
		catch (java.sql.SQLException e)
		{
			System.out.println("Echec connection : "+e);
		} // catch
	} // main
} // class
```

Chapitre 9. Exploitation avec PHP : mysqli()

```
<?php

echo "debut<br>" ;

$link = new mysqli("localhost", "login", "mot de passe", "ecomm");

echo "Connexion<br>" ;

// Exécution de la requête
$result = $link->query("SELECT nom_cli, prenom_cli FROM client");

while ($row = $result->fetch_assoc()) {
    echo $row['nom_cli'];
    echo $row['prenom_cli'];
    echo "<br>";
}


?>

// La fin du script _____________________________________________
```

Bibliographie.

1. "La méthode MERISE, Tome 1, Principes et outils",
   H.TARDIEU, A. ROCHFELD, R. COLLETTI,
   Editions d'Organisation, 1986.

2. "La méthode MERISE, Tome 2, Démarches et Pratiques",
   H.TARDIEU, A. ROCHFELD, R. COLLETTI, G. PANET, G. VAHEE,
   Editions d'Organisation, 1986

3. "Se former à MERISE",
   G. LOUVET ,
   Editions d'Organisation, 1990

4. "MERISE par l'exemple",
   R. MOUNYOL,
   Editions Ellipses, 1991.

5. "Base de données et Systèmes relationnels",
   C. DELOBEL, M. ADIBA,
   Editions Dunod, 1982.

Liste des illustrations

Tables des Matières.