Licences Mathématiques et Informatique
3ème année - Formation Initiale et par Apprentissage

Bases de données relationnelles

Polycopié de cours - Décomposition de schéma

Maude Manouvrier

Table des matières

Chapitre 9

Décomposition de schéma

9.1 Définition - Exemple simple

Nous avons vu précédemment que les dépendances fonctionnelles permettaient de déterminer les clés minimales d'une relation. Dans cette partie du cours, nous allons voir comment les dépendances fonctionnelles peuvent aider à décomposer un schéma de bases de données, i.e. à éliminer les anomalies d'un schéma de bases de données. Un schéma de bases de données ne doit pas contenir d'information redondante et doit permettre des mises à jour (insertion, suppression ou modification de nuplets).

Par exemple, soit R la relation de schéma $(Fournisseur, Adresse, Produit, Prix)$.
On a sur R les dépendances fonctionnelles suivantes :
$Fournisseur \longrightarrow Adresse$
$Fournisseur, Produit \longrightarrow Prix$

On remarque ici qu'il y a redondance d'information. En effet, si un fournisseur fabrique 10000 produits, son adresse sera répétée 10000 fois.
En outre, ce schéma pose des problèmes au niveau des mises à jour :

- Insertion : Si on souhaite insérer un nuplet, il n'est pas possible d'insérer un fournisseur et son adresse tant que ce dernier ne fabrique pas de produit. La clé primaire de R est le couple $(Fournisseur, Produit)$, l'attribut *Produit* ne peut donc pas être `NULL`. Il n'est donc pas possible d'insérer un nuplet de la forme :
  `('Founisseur A', '45 rue des Alouettes ...', NULL, NULL).`

- Suppression : Si on supprime le seul produit vendu par un fournisseur, on perd l'adresse du fournisseur. Comme précédemment, il n'est pas possible de mettre l'attribut *Produit* à `NULL`.

- Mise à jour : Si un fournisseur change d'adresse, il faut la modifier autant de fois qu'il vend de produits.

L'idée serait donc de décomposer la relation R en deux relations :
R_1 de schéma $(Fournisseur, Adresse)$
et R_2 de schéma $(Fournisseur, Produit, Prix)$

Il n'y a plus de redondance d'information et les modifications de la base de données ne posent plus de problème.

Il est à noter cependant, que dans ce cas si on souhaite connaître l'adresse du fournisseur qui vend un produit donné, il faut faire une jointure entre les deux relations R_1 et R_2. L'exécution de la requête est donc plus longue.

Attention : La décomposition d'une relation ne peut pas se faire au hasard. C'est ce que nous allons voir dans ce chapitre. Pour être correcte une décomposition de schéma doit vérifier 2 propriétés :

1. Elle doit être **sans perte de jointure**, ce qui signifie que l'on doit retrouver après la décomposition tous les nuplets de départ en faisant des jointures.

   Elle doit être **sans perte de dépendance**, ce qui signifie que l'on doit retrouver toutes les DF de la famille associée à la relation décomposée.

9.2 Sans perte d'information

9.2.1 Définition

Soit R une relation et F une famille de dépendances fonctionnelles sur R. La décomposition de R en R_1, R_2, ..., R_n est sans perte d'information (SPI - en anglais *Lossless jointure*) si et seulement si pour toutes les instances r de R on a : $r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \bowtie ... \bowtie \Pi_{R_n}(r)$.

La décomposition de R en R_1, R_2, ..., R_n est avec perte d'information ($\neg$SPI) si et seulement si : $\exists r$, une instance de R, telle que $r \neq \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \bowtie ... \bowtie \Pi_{R_n}(r)$.

Par exemple si on a $R(A, B, C)$ dont une instance r est présentée dans le tableau 9.1, et supposons que l'on a décomposé cette relation en deux relations $R_1(A, B)$ et $R_2(B, C)$.
$\forall r$, on peut construire r_1 une instance de R_1 en faisant $\Pi_{R_1}(r)$, i.e. $\Pi_{A,B}(r)$. De même, $r_2 = \Pi_{R_2}(r) = \Pi_{B,C}(r)$.

Pour toute instance r de R, retrouve-t-on la relation R en faisant une jointure entre R_1 et R_2 ? Est-ce que $\forall r$, $r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$?
D'après le tableau 9.3, on voit que $r \neq \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$.

On voit donc que la décomposition du schéma d'une relation ne peut pas être faite n'importe comment et qu'il faut utiliser pour cela les dépendances fonctionnelles.

TABLE 9.1 – Une instance de la relation R.

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

TABLE 9.2 – Les instances $\Pi_{R_1}(r)$ et $\Pi_{R_2}(r)$.

A	B
a_1	b_1
a_2	b_1

B	C
b_1	c_1
b_1	c_2

TABLE 9.3 – L'instance $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$.

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_2

9.2.2 Démonstration qu'une décomposition est Sans Perte d'Information (SPI)

Par définition des instances r_i qui sont construites à partir de $r(r_i = \Pi_{R_i}(r))$, on a toujours :
$r \subseteq \bowtie_{i=1}^{n} \Pi_{R_i}(r)$.

En effet, soit t un nuplet de r : pour tout i, $t.R_i$ est dans $\Pi_{R_i}(r)$, donc par définition de la jointure, t est dans $\bowtie_{i=1}^{n} \Pi_{R_i}(r)$.

Pour savoir si la jointure est sans perte d'information, c'est-à-dire ne crée pas de nuplets supplémentaires comme dans l'exemple du chapitre 9, il faut montrer que $\bowtie_{i=1}^{n} \Pi_{R_i}(r) \subseteq r$.

Pour cela on utilise l'algorithme suivant :
On prend un nuplet générique[1] de $t = \bowtie_{i=1}^{n} \Pi_{R_i}(r)$ et on vérifie, en raisonnant sur t et en utilisant les dépendances fonctionnelles de r, que le nuplet t est bien dans r.

Par exemple :

On a la relation $R(Fournisseur, Adresse, Produit, Prix)$ et
$F = \{Founisseur \longrightarrow Adresse;\ Fournisseur, Produit \longrightarrow Prix\}$.
Si la relation R est décomposée en deux relations :
$R_1(Fournisseur, Adresse)$ et $R_2(Fournisseur, Produit, Prix)$.

Cette décomposition est-elle SPI ?

Pour le vérifier, on choisit une nuplet générique[2] (f, a, po, pi) de $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$.

Est-ce que ce nuplet appartient à r ?
Comme $(f, a, po, pi) \in \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$, alors on a sûrement le nuplet (f, a) dans $\Pi_{R_1}(r)$ et le nuplet (f, po, pi) dans $\Pi_{R_2}(r)$.
Ces nuplets sont issus de la projection des nuplets (p, a, x_{13}, x_{14}) et (p, x_{22}, po, pi) de R, avec x_{ij} correspondant à une valeur inconnue (la valeur de l'attribut i du nuplet j). La table 9.4 montre les 2 nuplets qui permettent d'obtenir t.

TABLE 9.4 – Les 2 nuplets que r doit contenir pour que t existe.

$Fournisseur$	$Adresse$	$Produit$	$Prix$
f	a	x_{13}	x_{14}
f	x_{22}	po	pi

1. i.e. un nuplet représentant tous les nuplets possibles en donnant comme valeur d'attributs des variables pouvant prendre toutes les valeurs possibles des attributs.
2. f, a, po et pi représentent des variables.

Est-ce que le nuplet (f, a, po, pi) peut appartenir à r ?
La réponse est oui car par la dépendance fonctionnelle $Founisseur \longrightarrow Adresse$, on déduit $x_{22} = a$, comme le montre la table 9.5.

TABLE 9.5 – Les 2 nuplets que r doit contenir pour que t existe, après identification de x_{22}.

Fournisseur	*Adresse*	*Produit*	*Prix*
f	a	x_{13}	x_{14}
f	a	po	pi

Le nuplet (f, a, po, pi) appartient à r, donc $\bowtie_{i=1}^{n} \Pi_{R_i}(r) \subseteq r$. Par conséquent, comme on a toujours $r \subseteq \bowtie_{i=1}^{n} \Pi_{R_i}(r)$, on a donc $r = \bowtie_{i=1}^{n} \Pi_{R_i}(r)$. La décomposition est donc sans perte d'information.

Autre exemple :

Soit $R(A, B, C, D, E)$ une relation décomposée en $R_1(A, D)$, $R_2(A, B)$, $R_3(B, E)$, $R_4(C, D, E)$ et $R_5(A, E)$.
Soit F une famille de dépendances fonctionnelles associées à R :
$F = \{A \longrightarrow C;\ B \longrightarrow C;\ C \longrightarrow D;\ DE \longrightarrow C;\ CE \longrightarrow A;\}$.

La décomposition de R est-elle SPI ?

Pour vérifier que la décomposition de R est sans perte d'information, on pend un nuplet générique (a, b, c, d, e) de $\bowtie_{i=1}^{5} \Pi_{R_i}(r)$ et on vérifie qu'il s'agit bien d'un nuplet de r, instance de R.

On construit la table 9.6 à partir des nuplets de $\Pi_{R_i}(r)$ permettant de construire le nuplet $t = (a, b, c, d, e)$ appartenant au résultat de $\bowtie_{i=1}^{5} \Pi_{R_i}(r)$. t est en effet construit à partir de 5 nuplets : $t_1 = (a, d) \in r_1$, $t_2 = (a, b) \in r_2$, $t_3 = (b, e) \in r_3$, $t_4 = (c, d, e) \in r_4$ et $t_5 = (a, e) \in r_5$.

TABLE 9.6 – Contenu de l'instance r sachant qu'on a $t = (a, b, c, d, e) \in \Pi_{R_i}(r)$.

	A	**B**	**C**	**D**	**E**
nuplet ayant permis d'obtenir t_1	a	x_{12}	x_{13}	d	x_{15}
nuplet ayant permis d'obtenir t_2	a	b	x_{23}	x_{24}	x_{25}
nuplet ayant permis d'obtenir t_3	x_{31}	b	x_{33}	x_{34}	e
nuplet ayant permis d'obtenir t_4	x_{41}	x_{42}	c	d	e
nuplet ayant permis d'obtenir t_5	a	x_{52}	x_{53}	x_{54}	e

Peut-on trouver le nuplet (a, b, c, d, e) dans la table 9.6 à partir des dépendances fonctionnelles de R ?

1. On a $A \longrightarrow C$. Par conséquent on obtient que : $x_{13} = x_{23} = x_{53}$ car, dans chacun de ces nuplets, la valeur de A est a. Mais la valeur de C reste une inconnue (x_{13}).

TABLE 9.7 – Contenu de l'instance r après analyse de $A \longrightarrow C$.

	A	**B**	**C**	**D**	**E**
nuplet ayant permis d'obtenir t_1	a	x_{12}	x_{13}	d	x_{15}
nuplet ayant permis d'obtenir t_2	a	b	x_{13}	x_{24}	x_{25}
nuplet ayant permis d'obtenir t_3	x_{31}	b	x_{33}	x_{34}	e
nuplet ayant permis d'obtenir t_4	x_{41}	x_{42}	c	d	e
nuplet ayant permis d'obtenir t_5	a	x_{52}	x_{13}	x_{54}	e

2. On a $B \longrightarrow C$. Par conséquent on obtient que : $x_{13} = x_{33}$ car, dans les deux nuplets, B a pour valeur b.

TABLE 9.8 – Contenu de l'instance r après analyse de $B \longrightarrow C$.

	A	**B**	**C**	**D**	**E**
nuplet ayant permis d'obtenir t_1	a	x_{12}	x_{13}	d	x_{15}
nuplet ayant permis d'obtenir t_2	a	b	x_{13}	x_{24}	x_{25}
nuplet ayant permis d'obtenir t_3	x_{31}	b	x_{13}	x_{34}	e
nuplet ayant permis d'obtenir t_4	x_{41}	x_{42}	c	d	e
nuplet ayant permis d'obtenir t_5	a	x_{52}	x_{13}	x_{54}	e

3. On a $C \longrightarrow D$. Par conséquent on en déduit que $x_{24} = x_{34} = x_{54} = d$ car, dans quatre nuplets, C a pour valeur x_{13}.

TABLE 9.9 – Contenu de l'instance r après analyse de $C \longrightarrow D$.

	A	**B**	**C**	**D**	**E**
nuplet ayant permis d'obtenir t_1	a	x_{12}	x_{13}	d	x_{15}
nuplet ayant permis d'obtenir t_2	a	b	x_{13}	d	x_{25}
nuplet ayant permis d'obtenir t_3	x_{31}	b	x_{13}	d	e
nuplet ayant permis d'obtenir t_4	x_{41}	x_{42}	c	d	e
nuplet ayant permis d'obtenir t_5	a	x_{52}	x_{13}	x_{54}	e

4. On a $DE \longrightarrow C$. Par conséquent on obtient que : $x_{13} = c$, car, dans les deux derniers nuplets, les attributs D et E ont respectivement les valeurs d et e. Donc l'attribut C a toujours comme valeur c.

TABLE 9.10 – Contenu de l'instance r après analyse de $DE \longrightarrow C$.

	A	**B**	**C**	**D**	**E**
nuplet ayant permis d'obtenir t_1	a	x_{12}	c	d	x_{15}
nuplet ayant permis d'obtenir t_2	a	b	c	d	x_{25}
nuplet ayant permis d'obtenir t_3	x_{31}	b	c	d	e
nuplet ayant permis d'obtenir t_4	x_{41}	x_{42}	c	d	e
nuplet ayant permis d'obtenir t_5	a	x_{52}	c	d	e

5. On a $CE \longrightarrow A$. Par conséquent on obtient que : $x_{41} = x_{31} = a$ car, dans les trois derniers nuplets, les attributs C et E ont respectivement les valeurs c et e. Donc l'attribut A a toujours comme valeur a.

TABLE 9.11 – Contenu de l'instance r après analyse de $CE \longrightarrow A$.

	A	**B**	**C**	**D**	**E**
nuplet ayant permis d'obtenir t_1	a	x_{12}	c	d	x_{15}
nuplet ayant permis d'obtenir t_2	a	b	c	d	x_{25}
nuplet ayant permis d'obtenir t_3	$\underline{a}$	$\underline{b}$	$\underline{c}$	$\underline{d}$	$\underline{e}$
nuplet ayant permis d'obtenir t_4	a	x_{42}	c	d	e
nuplet ayant permis d'obtenir t_5	a	x_{52}	c	d	e

Le 3ème nuplet est t. On a donc bien montré que $t = (a, b, c, d, e)$, nuplet de $\bowtie_{i=1}^{5} \pi_{R_i}(r)$, appartient à r.

9.3 Sans perte de dépendances

Les dépendances fonctionnelles contiennent de l'information. Il s'agit de liens de déduction entre attributs donc de contraintes. Certaines dépendances peuvent disparaître lors de la décomposition si tous les attributs d'une dépendance fonctionnelle ne sont pas réunis dans une même relation.

Par exemple, si on a une relation $R(Rue, Ville, CP)$ et $F = \{CP \longrightarrow V;\ V, R \longrightarrow CP\}$. Si on décompose la relation R en $R_1(CP, Ville)$ et $R_2(CP, Rue)$, on perd la dépendance fonctionnelle $V, R \longrightarrow CP$ (Rappel : une dépendance fonctionnelle est attachée à une et une relation et s'applique aux attributs de la dite relation) : on a juste $CP \longrightarrow V$ associée à la relation R_1.

9.3.1 Définition

Une décomposition de la relation R, munie d'une ensemble de dépendances fonctionnelles F, en n relations R_1, R_2, ..., R_n est sans perte de dépendance (SPD) si et seulement si [18] :
$F^+ = (F^+_{R_1} \cup F^+_{R_2} \cup ... \cup F^+_{R_n})^+$, c'est-à-dire si $\left(\cup_i F^+_{R_i}\right) +$ implique logiquement toutes les dépendances fonctionnelles de F, avec F_{R_i} l'ensemble des dépendances fonctionnelles projetées sur les attributs de R_i (n'impliquant que les attributs de R_i).

L'ensemble des dépendances fonctionnelles projetées sur R_i sont les dépendances $X \longrightarrow Y$ de F^+ telles que $XY \subseteq R_i$.

9.3.2 Démonstration qu'une décomposition est Sans Perte de Dépendance (SPD)

Exemple

Soit $R(Nom, Telephone, Bureau)$ et $F = \{N \longrightarrow T, T \longrightarrow B, B \longrightarrow N\}$.

Si on décompose R en $R_1(Nom, Telephone)$ et $R_2(Bureau, Telephone)$ conserve t'on la dépendance $B \longrightarrow N$?

Il faut faire attention qu'une dépendance fonctionnelle est définie sur un schéma et que l'on prend en compte la fermeture de F.

Par conséquent, si F^+ est projetée sur les attributs N et T de R_1, les dépendances fonctionnelles projetées de R_1 sont $N \longrightarrow T$ et $T \longrightarrow N$ (par transitivité de $T \longrightarrow B$ et $B \longrightarrow N$ dans F^+). $F_{R_1} = \{N \longrightarrow T$, $T \longrightarrow N\}$.

De même pour R_2, les dépendances projetées sont : $T \longrightarrow B$ et $B \longrightarrow T$ (par transitivité de $B \longrightarrow N$ et $N \longrightarrow T$ dans F^+). $F_{R_2} = \{T \longrightarrow B,\, B \longrightarrow T\}$.

Par conséquent, la fermeture de $F_{R_1} \cup F_{R_2}$ contient $B \longrightarrow N$. La dépendance n'est pas perdue, elle peut être retrouvée à partir de F_{R_1} et F_{R_2}. La décomposition est donc sans perte de dépendance (SPD).

Algorithme

En pratique, on ne calcule pas les fermetures d'ensemble de dépendances fonctionnelles, mais les fermeture d'attributs en tenant compte des relation de la décomposition et des dépendances projetées.

Si on croit qu'une dépendance fonctionnelle $X \longrightarrow Y$ est perdue alors on vérifie si la dépendance fonctionnelle appartient à $[X]^+_{\cup F_{R_i}}$.

La fermeture d'attribut $[X]^+_{\cup F_{R_i}}$ se calcule itérativement en utilisant la formule $X \cup ([X \cap R_i]^+ \cap R_i)$.

$(X \cap R_i)^+$ correspond à la fermeture de l'attribut X projetée sur les attributs de R_i. Comme on ne doit tenir compte que des attributs de R_i, on enlève de la fermeture les attributs n'appartenant pas à R_i.

Par exemple soient $R(A, B, C, D)$ et $F = \{A \longrightarrow B, B \longrightarrow C, C \longrightarrow D, D \longrightarrow A\}$.

Si la relation R est décomposée en $R_1(A, B)$, $R_2(B, C)$ et $R_3(C, D)$, alors on a :
$F_{R_1} = \{A \longrightarrow B,\, B \longrightarrow A\}$.
$F_{R_2} = \{B \longrightarrow C,\, C \longrightarrow B\}$.
$F_{R_3} = \{C \longrightarrow D,\, D \longrightarrow C\}$.
La dépendance fonctionnelle $D \longrightarrow A$ est-elle perdue ?

Pour le vérifier, on applique itérativement la formule :

1. $D_0 = \{D\}$
2. En premier lieu, on regarde la relation R_3 qui contient l'attribut D et on projete la fermeture de D sur les attributs de R_3.
   On regarde la fermeture de l'attribut D en se restreignant aux attributs de R_3 qui contient D :
   $D_1 = \{D\} \cup ([\{D\} \cap \{C, D\}]^+ \cap \{C, D\})$
   $\Longrightarrow D_1 = \{D\} \cup ([D]^+ \cap \{C, D\})$
   or $[D]^+ = \{A, B, C, D\}$
   $\Longrightarrow D_1 = \{D\} \cup (\{A, B, C, D\} \cap \{C, D\})$
   $\Longrightarrow D_1 = [D]^+_{F_{R_3}} = \{C, D\}$

   La fermeture de D contient tous les attributs (D est clé de R). En se restreignant aux attributs de R_3, on conserve les attributs C et D.

3. On regarde la fermeture de l'ensemble d'attributs $\{C, D\}$ en se restreignant aux attributs de R_2 qui contient C :
   $D_2 = \{C, D\} \cup ([\{C, D\} \cap \{B, C\}]^+ \cap \{B, C\})$
   $\Longrightarrow D_2 = \{C, D\} \cup ([C]^+ \cap \{C, D\})$
   or $[C]^+ = \{A, B, C, D\}$
   $\Longrightarrow D_2 = \{C, D\} \cup (\{A, B, C, D\} \cap \{B, C\})$
   $\Longrightarrow D_2 = [D]^+_{F_{R_3} \cup F_{R_2}} = \{B, C, D\}$

4. On regarde la fermeture de l'ensemble d'attributs $\{B, C, D\}$ en se restreignant aux attributs de R_1 qui contient B :
   $D_3 = \{B, C, D\} \cup ([\{B, C, D\} \cap \{A, B\}]^+ \cap \{A, B\})$
   $\Longrightarrow D_3 = \{B, C, D\} \cup ([B]^+ \cap \{A, B\})$
   or $[B]^+ = \{A, B, C, D\}$
   $\Longrightarrow D_3 = \{B, C, D\} \cup (\{A, B, C, D\} \cap \{A, B\})$
   $\Longrightarrow D_3 = [D]^+_{F_{R_3} \cup F_{R_2} \cup F_{R_1}} = \{A, B, C, D\}$

   D'où $[D]^+_{\cup F_{R_i}} = \{A, B, C, D\}$, donc la dépendance $D \longrightarrow A$ n'est pas perdue.

9.4 Conclusion

Une bonne décomposition de schéma doit être sans perte d'information et sans perte de dépendances fonctionnelles, c'est-à-dire sans créer de nuplets ni perdre l'information de type structurel contenue dans les dépendances fonctionnelles.

Bibliographie

[1] D. Austin, *Using Oracle8^{TM}*, Simple Solutions - Essential Skills, QUE, 1998, ISBN : 0-7897-1653-4

[2] R. Chapuis, *Oracle 8*, Editions Dunes et Laser, 1998, ISBN : 2-913010-07-5

[3] P. Chen, *The Entity-Relationship Model–Toward a Unified View of Data*, ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, Pages 9-36, http ://www.csc.lsu.edu/~chen/pdf/erd.pdf

[4] T. Connolly, C. Begg et A. Strachan, *Database Systems - A pratical Approach to Design, Implementation and Management*, Addison-Wesley, 1998, ISBN : 0-201-34287-1, disponible à la BU 055.7 CON

[5] C.J. Date, *Introduction aux bases de données*, 6ème édition, Thomson Publishing, 1998, ISBN : 2-84180-964-1, disponible à la BU 005.7 DAT

[6] R. Elamsri et S.B. Navathe, *Fundamentals of Database Systems*, 3ème édition, Addison Wesley-disponible à la BU 005.7 ELM

[7] P. Delmal, *SQL2 - Application à Oracle, Access et RDB*, 2ème édition, Bibliothèque des Universités - Informatique, De Boeck Université, 1988, ISBN : 2-8041-2995-0,disponible à la BU 005.74 SQL

[8] S. Feuerstein, B. Pribyl et C. Dawes, *Oracle PL/SQL - précis et concis*, O'Reilly, 2000, ISBN : 2-84177-108-3

[9] G. Gardarin, *Bases de Données - objet & relationnel*, Eyrolles, 1999, ISBN : 2-212-09060-9, disponible à la BU 005.74 GAR

[10] R. Grin, *Introduction aux bases de données, modèle relationnel*, Université Sophia-Antipolis, jan. 2000

[11] R. Grin, *Langage SQL*, Université Sophia-Antipolis, jan. 2000

[12] G. Gardarin et O. Gardarin *Le Client-Serveur*, Eyrolles, 1999, disponible à la BU

[13] H. Garcia-Molina, J.D. Ulmann et J. Widow, *Database SYstem Implementation*, Prentice Hall, 2000, ISBN :0-13-040264-8, disponible à la BU 005.7 GAR

[14] H. Garcia-Molina, J.D. Ulmann et J. Widow, *Database Systems - The Complete Book*, Prentice Hall, 2002, ISBN :0-13-031995-3

[15] S. Krakowiak, *Gestion Physique des données*, Ecole Thématique "Jeunes Chercheurs" en Base de Données, Volume 2, Port Camargue, mars 1997

[16] D. Lockman, *Oracle8^{TM} Dévéloppement de bases de données*, Le programmeur - Formation en 21 jours, Editions Simon et Schuster Macmillan (S&SM), 1997, ISBN : 2-7440-0350-6, disponible à la BU 005.74 ORA

[17] P.J. Pratt, *Initiation à SQL - Cours et exercices corrigés*, Eyrolles, 2001, ISBN : 2-212-09285-7

[18] R. Ramakrishnan et J. Gehrke, *Database Management Systems*, Second Edition ; McGraw-Hill, 2000, ISBN : 0-07-232206-3, disponible à la BU 055.7 RAM

[19] A. Silberschatz, H.F. Korth et S. Sudarshan, *Database System Concepts*, Third Edition, McGraw-Hill, 1996, ISBN : 0-07-114810-8, disponible à la BU 005.7 DAT

[20] C. Soutou, *De UML à SQL - Conception de bases de données*, Eyrolles, 2002, ISBN : 2-212-11098-7

[21] J.D. Ullman et J. Widom, *A first Course in Database Systems*, Prentice Hall, 1997, ISBN : 0-13-887647-9, disponible à la BU 005.7 ULL