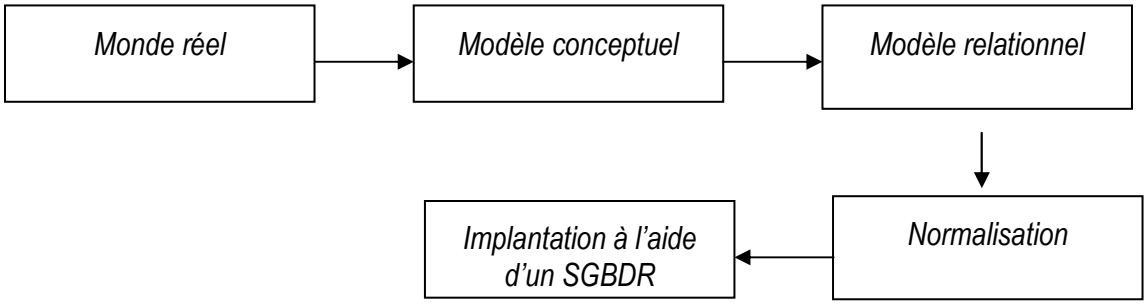
4. Normalisation

4.1. Dépendance fonctionnelle

4.1.1. Introduction

On distingue 5 formes normales, qui représentent des tests formels de validité et de cohérence de la base de données. Généralement, on s'arrête à la 3ème forme normale. La définition des formes normales se base sur les dépendances fonctionnelles entre les différents attributs de la base de données [1, 7].



4.1.2. Définition

Soit une relation $R(A, B, C, \cdots)$. A détermine B ou B dépend fonctionnellement de A si et seulement si $a = a' \Rightarrow b = b'$, c'est-à-dire que pour toute valeur de A correspond une seule valeur de B. On note $A \rightarrow B$.

R	***A***	***B***	***C***
	a	*y*	*u*
	b	*c*	*d*
	a	*c*	*d*
	f	*c*	*d*
	h	*u*	*v*

$A \rightarrow B$ est faux, mais $B \rightarrow C$ peut être est vrai.

Exemple :

R (UV, Jour, Heure, Salle, Enseignant) ; F = {Jour, Heure, Salle→UV, Enseignant}.

4.1.3. Représentation graphique des dépendances fonctionnelles

C'est une représentation graphique permettant de visualiser aisément toutes les dépendances fonctionnelles.

Exemple :

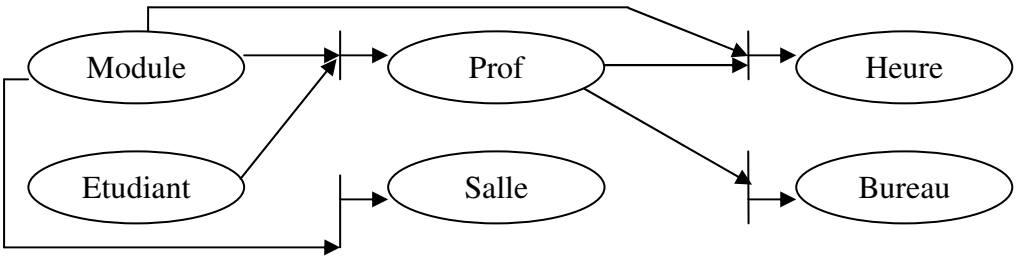
Soient les dépendances fonctionnelles suivantes :

Etudiant, module → prof

Etudiant, module → salle

Prof, module → heure

Prof → bureau



4.1.4. Propriétés (axiomes d'Armstrong)

- ***réflexivité :*** $\forall Y \subseteq X, X \rightarrow Y$
- ***augmentation :*** $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$
- ***transitivité :*** $(X \rightarrow Y \text{ et } Y \rightarrow Z) \Rightarrow X \rightarrow Z$
- ***union :*** $(X \rightarrow Y \text{ et } X \rightarrow Z) \Rightarrow X \rightarrow YZ$
- ***pseudo-transitivité :*** $(X \rightarrow Y \text{ et } WY \rightarrow Z) \Rightarrow WX \rightarrow Z$
- ***décomposition :*** $(X \rightarrow Y \text{ et } Z \subseteq Y) \Rightarrow X \rightarrow Z$

4.1.5. Dépendance fonctionnelle élémentaire

On appelle DFE une DF de la forme X→A, où A est un attribut unique et A⊄X, telle que ∀ Y ⊂ X, il n'existe pas de DF Y → A.

Exemple :

Si on considère les deux DF : Numéro, Nom → Adresse (1) et Numéro → Adresse.

(1) est une DF non élémentaire car Numéro ∈ (Numéro, Nom) et il existe la DF Numéro → Adresse.

4.1.6. Dépendance fonctionnelle directe

On appelle DFD une DF de la forme X→A, où A est un attribut unique et A⊄X, telle que : s'il n'existe pas d'attribut Z telle que : X → Z et Z → A.

Exemple :

Soit R(A,B,C,D) ; F={A→B , B→C, A→C , D→B}.

A→C n'est pas directe car C est transitivement dépendant de A en appliquant les DFS A→B et B→C.

4.2. La fermeture transitive

Soit F un ensemble de dépendances fonctionnelles. La fermeture transitive F^+ est l'ensemble de toutes les dépendances de F augmenté de celle obtenues par les propriétés précédentes.

On note $F \Leftrightarrow G$ si et seulement si $F^+ = G^+$.

4.3. La couverture minimal

Soit F un ensemble de dépendances fonctionnelles. G est une couverture minimal de F si G est minimal, $F^+ = G^+$ et $\forall G' \subset G, G'^+ \neq F^+$.

On appelle *couverture minimale* de F un ensemble G équivalent à F qui vérifie les trois propriétés suivantes [5, 6]:

1. Toutes les parties droites sont réduites à un seul élément,

2. Aucune partie gauche ne contient d'élément redondant:

∀ X→A ∈ G et Z ⊂ X, (G - {X→ A} ∪ {Z→ A}) ≠ G^+

3. Il n'y a pas de dépendance superflue:

$\forall X \rightarrow A \in G, (G - \{X \rightarrow A\}) \neq G^+$

Un algorithme de calcul d'une (il y en a toujours au moins une) couverture minimale consiste à « imposer », dans l'ordre, ces trois conditions:

• Pas 1: On expanse les dépendances X→ A1 … An en X→ A1, …, X→An. On obtient F'.

• Pas 2: Pour chaque partie gauche de F', on choisit un ordre d'examen des attributs et on cherche à les enlever. Les suppressions successives conduisent à F".

• Pas 3: On choisit un ordre d'examen de F" et on cherche à enlever chacune des dépendances considérées. Les suppressions successives conduisent à G, le résultat final.

Exemple :

Soit la relation cours (prof, module, salle, heure) vérifiant le DF suuivantes

F = {Prof, module →salle ; module, salle →heure ;

Prof, module, heure →module ; prof, module, salle→heure, salle}

Pas 1

F' = 1- Prof, module →salle
2- module, salle →heure
3- prof, module, heure →module
4- prof, module, salle→heure
5- module, salle→salle

Pas 2

F" = 1- Prof, module →salle
2- module, salle →heure
3- prof, module, heure →module

Pas 3

G = 1- Prof, module →salle
2- module, salle →heure

4.4. Fermeture d'un ensemble d'attributs

4.4.1. Définition

La fermeture d'un ensemble d'attributs X et qu'on note X^+ est aussi un ensemble d'attributs tel que : $X^+ = \{ \cup A_i / X \rightarrow A_i$ peut être déduite de F par application des axiomes d'Armstrong }

4.4.2. Un algorithme de calcul de la fermeture

Soit F un ensemble de dépendances et $X \subseteq U$ un ensemble d'attributs [1].

L'algorithme calcule une suite d'ensembles d'attributs $X^{(0)}, X^{(1)}, \ldots$

Algorithme

Données : X, U, F.

1. $X^{(0)} = X$.

2. $X^{(i+1)} = X^{(i)} \cup \{A / \exists Z : Y \rightarrow Z \in F$ et $A \in Z$ et $Y \subseteq X^{(i)}\}$.

3. Si $X^{(i+1)} = X^{(i)}$, l'algorithme s'arrête.

Note : L'algorithme s'arrête toujours puisque $X(0) \subseteq X(1) \subseteq X(2) \ldots \subseteq U$ et U est un ensemble fini.

Exemple :

Soit F = {AB → C , D → EG, C → A, BE → C, BC → D, CG → BD, ACD → B, CE → AG} ;

X : BD et U : ABCDEG

L'algorithme calcule : $X^{(0)}$: BD

$X^{(1)}$: BDEG

$X^{(2)}$: BDEGC

$X^{(3)}$: BDEGCA

$X^{(4)}$: BDEGCA

4.5. Démarche de recherche des clés candidates

1. Pour toute d.f X→y vérifier par R calculer X^+
2. Si $X^+ = U$ placer X dans une table Cp des clés potentielles
3. Si $X^+ \neq U$ placer X dans une table CpA des clés potentielles par augmentation
4. Pour tout X de la table CpA, enlever X de la table CpA, pour toute partie p de U- X^+, augmenter X avec p et soit X' = X∪p cet ensemble. Réappliquer les étapes 1, 2 et 3 à X'.
5. Répéter l'étape 4 jusqu'à ce que la table CpA devienne vide
6. Supprimer de la table les éléments qui ne satisfont pas les conditions d'une clé.

Exemple :

Soit R(A,B,C,D,E) vérifiant les DFs :

AB→ C

C→D

BC→A

D→AC

E→B

Etape 1 : $\{AB\}^+ = \{ABCD\} \neq U$

$\{C\}^+ = \{CDA\} \neq U$

$\{BC\}^+ = \{BCAD\} \neq U$

$\{D\}^+ = \{DAC\} \neq U$

$\{E\}^+ = \{EB\} \neq U$

Etape 2 et 3 : construire la liste des clés potentielles et la liste des clés potentielles par augmentation

CP	CPA
	AB
	C
	BC
	D
	E

Etape 4 :

On travaille avec les éléments de la liste CpA

Cas de D : on enlève D de table CpA

Soit $Y = U - D^+ = \{ABCDE\} - \{ACD\} = \{BE\}$. On doit donc augmenter D avec les parties de Y qui sont B, E et BE et réappliquer pour chaque augmentation les étapes 1, 2 et 3

Ensemble	Augmenter par	Calcul de X^+	Décision à prendre
DB	B	$\{DB\}^+ = \{DBAC\} \neq U$	Ajouter DB à CpA
DE	E	$\{DE\}^+ = \{DEACB\} = U$	Ajouter DE à Cp
DBE	BE	$\{DBE\}^+ = \{BDEAC\} = U$	Ajouter DBE à Cp

CP		CPA
1. ABE	Eliminé car contient 7	
2. BCE	Eliminé car contient 3	
3. CE		
4. CEB	Eliminé car contient 3	
5. DE		
6. DEB	Eliminé car contient 5	
7. EA		
8. EAC	Eliminé car contient 7	
9. EACD	Eliminé car contient 7	
10. EDA	Eliminé car contient 7	
11. ED	Eliminé car identique à 5	
12. EC	Eliminé car identique à 3	
13. ECD	Eliminé car contient 3	

Les clés candidates effectives :CE, DE et EA

4.6. Mauvaise conception

Une mauvaise conception de la base de données engendre un ensemble de problèmes.

Exemple :

Soit l'extension de la relation propriétaire suivante :

Propriétaire

NS	Nom	Prénom	Date-achat	Prix	M	Marque	Type	Puiss	cour
100	Redha	Lazhar	10/02/95	10000	00019540	SEAT	Ibiza	6	R
100	Redha	Lazhar	11/06/00	50000	00010040	Opel	Corsa	9	V
400	Ali	Taha	?	?	?	?	?	?	?
?	?	?	?	?	00011040	SEAT	Leon	7	N
200	Nabil	Ahmed	20/04/12	5000	00011240	Citroën	2cv	2	B
200	Nabil	Ahmed	20/08/14	20000	00011440	Citroën	Am18	5	B

Cette table présente plusieurs problèmes (anomalies) :

Redondance des données : chaque personne apparaît autant de fois qu'elle possède de voitures (Redha, Nabil).

Ces redondances peuvent engendrer une incohérence de données lors une modification :

Par exemple, si on change l'adresse de Redha, il faut la modifier dans tous les enregistrements de Redha.

Il est indispensable d'utiliser la valeur *NULL* dans les cas des personnes qui ne possèdent pas de véhicule.

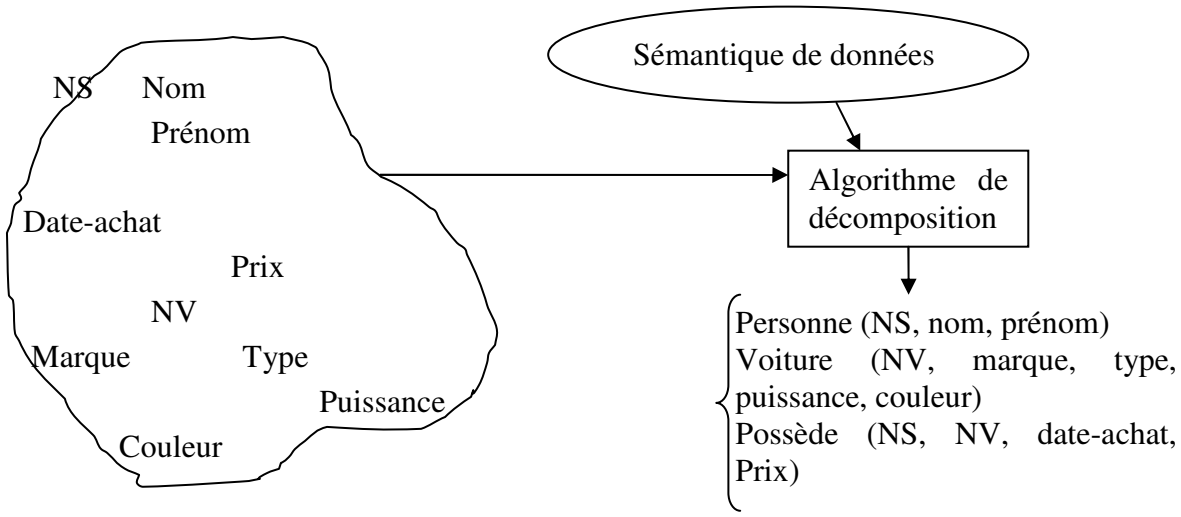
En conclusion ; ceci est la conséquence d'une mauvaise conception : il fallait trois relation (personne, possède, voiture) au lieu d'une seule.

L'approche par décomposition :

L'approche par décomposition est une approche méthodologique pour la conception de schémas relationnels.

Elle consiste à partir d'une relation universelle (contient tous les attributs) et à la décomposer en plusieurs relations qui ne poseraient plus de problèmes.

Cette décomposition doit être réalisée par un algorithme de décomposition et à partir d'une bonne compréhension des propriétés sémantiques des données.



4.7. La décomposition

4.7.1. Définition

C'est l'opération de remplacement d'une relation R (A_1, A_2,…, A_n) par une ensemble de relations R_1, R_2,…, R_M où chaque R_I est obtenue par une opération de projection de R et telles que la jointure naturelle des relations R_1, R_2,…, R_M ($R_1 \infty R_2 \ldots \infty R_M$) donne une relation R' ayant le même schéma que la relation R. une relation peut donc décomposée de différents manières [6].

Exemple :

Voiture (M, Marque, Type, Puissance, couleur)

M	Marque	Type	Puissance	Couleur
0666601240	SEAT	IBIZA	6	Bleue
0999601440	SEAT	IBIZA	6	Rouge

Décomposition 1 :

r1 (M, Type, couleur)

M	Type	Couleur
0666601240	IBIZA	Bleue
0999601440	IBIZA	Rouge

r2 (Type, Marque, Puissance)

Type	Marque	Puissance
IBIZA	SEAT	6

r1 ∞ r2 (M, Marque, Type, Puissance, couleur) même schéma que voiture.

Donc c'est bien une décomposition.

r1 ∞ r2

M	Marque	Type	Puissance	Couleur
0666601240	SEAT	IBIZA	6	Bleue
0999601440	SEAT	IBIZA	6	Rouge

Décomposition 2

R1 (M, Type)

M	Type
0666601240	IBIZA
0999601440	IBIZA

R2 (type, puissance, couleur)

Type	Puissance	Couleur
IBIZA	6	Bleue
IBIZA	6	Rouge

R3 (type, marque)

Type	Marque
IBIZA	SEAT
IBIZA	SEAT

R1 ∞ R2 ∞ R3 a le même schéma que voiture donc c'est bien une décomposition.

R1 ∞ R2

M	Type	Puissance	Couleur
0666601240	IBIZA	6	Bleue
0666601240	IBIZA	6	Rouge
0999601440	IBIZA	6	Bleue
0999601440	IBIZA	6	Rouge

Soit t = R1 ∞ R2

t ∞ R3

M	Marque	Type	Puissance	Couleur
0666601240	SEAT	IBIZA	6	Bleue
0666601240	SEAT	IBIZA	6	Rouge
0999601440	SEAT	IBIZA	6	Bleue
0999601440	SEAT	IBIZA	6	Rouge

Remarque

r1 ∞ r2 = voiture

R1 ∞ R2 ∞ R3 ≠ voiture

La décomposition (r1, r2) permet de retrouver toute l'information par jointure alors que la décomposition (R1, R2, R3) ne permet pas de retrouver la couleur d'une voiture.

De ce fait, on est amené à introduire la notion de décomposition sans perte et avec perte d'information.

4.7.2. Décomposition Sans Perte d'Information

Une décomposition de R en R1 et R2 est SPI si au moins l'un des deux DF suivantes appartient à F^+

1. $R1 \cap R2 \rightarrow R1-R2$

2. $R1 \cap R2 \rightarrow R2-R1$

Exemple :

Soit R(A,B,C) vérifiant F={C→A ; B→C} et soit la décomposition suivante : R1(A, C) et R2(B,C), on a $R1 \cap R2=\{C\}$; $R1-R2=\{A\}$; on remarque que la DF $C \rightarrow A \in F^+$ donc cette décomposition est SPI.

4.7.3. Décomposition sans perte de DF

Une décomposition (R1,… Rm) de R préserve les DFs (SPD) si la fermeture transitive des DFs de R est la même que la fermeture transitive de l'union des Dfs des R1, … Rm.

Exemple :

Soit VOITURE (N°VEH, TYPE, COULEUR, MARQUE, PUISSANCE) et

F = {N°VEH→ TYPE, COULEUR

TYPE → MARQUE, PUISSANCE}

Exemple :

1) R1 (NVH, TYPE, COULEUR) ⇒ F1 = {NVH → TYPE, NVH → COULEUR}

R2 (TYPE, MARQUE, PUIS) ⇒ F2 = {TYPE → MARQUE, TYPE → PUIS}

⇒ OK

2) R'1 (NVH, TYPE) ⇒ F'1 = {NVH → TYPE}

R'2(TYPE, PUIS, COULEUR) ⇒ F'2 = {TYPE → PUIS}

R'3(TYPE, MARQUE) ⇒ F'3 = {TYPE → MARQUE}

⇒ ON A PERDU LA DF : NVH →COULEUR

4.8. Pourquoi normaliser ?

La normalisation est utile :

- Pour limiter les redondances de données,
- Pour limiter les pertes de données,
- Pour limiter les incohérences au sein des données et
- Pour améliorer les performances des traitements [3].

4.8.1 Première forme normale

Définition : Une relation est en première forme normale si et seulement si tous ses attributs ont des valeurs atomique (non multiples, non composées).

La première forme normale est notée 1NF (1FN en français).

Si besoin est, on décompose les attributs ou la relation pour respecter la 1NF.

4.8.2. Deuxième forme normale

Définition : Soient C une clé candidate de R et A un attribut de R. R est en deuxième forme normale si et seulement si elle est en 1NF et pour tout A tel que A n'appartient pas à C, on a C→A élémentaire.

La deuxième forme normale est notée 2NF (2FN en français).

Une relation peut être en 2NF par rapport à une de ses clés candidates et ne pas l'être par rapport à une autre.

Pour rechercher une 2NF, il est au préalable nécessaire de déterminer *toutes les DF* et de choisir une clé candidate. Il est recommandé de trouver *toutes les clés candidates* afin de ne pas en laisser passer une plus intéressante qu'une autre.

Si besoin est, on décompose les attributs ou la relation pour respecter la 2NF.

Une relation avec une clé candidate choisie réduite à un seul attribut est, par définition, forcément en 2NF.

Exemples :

1. Commande (codeClient, codeArticle, client, article) avec les DF

codeClient → client et codeArticle → article n'est pas en 2NF

2. Pers(nom, prénom, âge, nombreEnfants) avec la DF

nom, prénom →âge, nombreEnfants est en 2NF.

3. Fournisseur1 (NF, NomProduit, Adr, Tel, Prix) avec les DF

NF, NomProduit → prix et NF → Adr, Tel n'est pas en 2ème forme normale.

Une telle relation pose des problèmes :

Redondances : s'il existe 100 produits pour un fournisseur on va répéter 100 fois le nom, l'adresse, le téléphone du fournisseur.

Problème de mise à jour pour les insertions : quand on veut rajouter un produit, il faut rentrer à nouveau l'adresse et le téléphone du fournisseur.

Problème pour les suppressions : si on supprime (momentanément) la liste des produits d'un fournisseur, alors on supprime aussi le fournisseur.

Problème de mise à jour des tuples : si un fournisseur change d'adresse ou de téléphone, il faut faire cette mise à jour sur tous les 100 tuples.

Solution :

On décompose Fournisseur1 en deux relations :

Fournisseur (NF, Adr, Tel)

Catalogue (NF, NomProduit, Prix)

Qui sont en 2ème forme normale

Cette décomposition est :

Sans perte d'information (NF est l'identifiant de la relation Fournisseur)

Sans perte de dépendance fonctionnelle (les DF sont soit dans l'une, soit dans l'autre des deux relations décomposées).

4.8.3. Troisième forme normale

Définition : Soient C une clé candidate de R et A et B deux ensembles non vides disjoints d'attributs de R. R est en troisième forme normale si et seulement si elle est en 2NF et pour tous A et B tels que A et B disjoints de C, on n'a pas A→ B.

La troisième forme normale est notée 3NF (3FN en français).

Une relation peut être en 3NF par rapport à une de ses clés candidates et ne pas l'être par rapport à une autre.

Si besoin est, on décompose les attributs ou la relation pour respecter la 3NF.

Une relation en 2NF avec au plus un attribut qui n'appartient pas à la clé candidate choisie est, par définition, forcément en 3NF.

Exemple :

Commande(numéroCommande, codeClient, client, article) avec les DF

numéroCommande → codeClient, client, article et

codeClient → client

N'est pas en 3NF alors que Pers(nom, prénom, âge, nombreEnfants) avec la DF

nom, prénom → âge, nombreEnfants

Est en 3NF.

La 3NF ne résout pas tout. Des DF plus nombreuses que dans la 2NF sont désormais prises en compte, ce qui est une chose positive. Mais toutes les DF ne le sont pas.

Exemple :

Université(étudiant, matière, enseignant, note) avec les DF

étudiant, matière → enseignant, note et

enseignant → matière

Est en 3NF. Il est cependant possible d'y trouver les occurrences ⟨Joseph, SGBD, Taha, 5⟩ et ⟨Patricia, IA, Taha, 17⟩, ce qui est incohérent.

4.8.4. Forme normale de Boyce-Codd

Définition : Une relation est en forme normale de Boyce-Codd si et seulement si ses clés candidates sont les uniques sources de DF [1, 4].

La forme normale de Boyce-Codd est notée BCNF (FNBC en français).

Si une relation est en BCNF, elle l'est par définition pour toutes ses clés candidates.

Si besoin est, on décompose les attributs ou la relation pour respecter la BCNF.

Exemple :

Université(étudiant, matière, enseignant, note) avec les DF

étudiant, matière → enseignant, note et enseignant → matière n'est pas en BCNF alors que Pers(nom, prénom, âge, nombreEnfants) avec la DF nom, prénom → âge, nombreEnfants Est en BCNF.

4.9 Décomposition en 3NF

Principe

À partir d'un schéma <U, F> dont K est une clé:

1. On cherche une couverture minimale G pour F,

2. Pour chaque partie gauche X des dépendances de G telle que $X \rightarrow A_1, \ldots, X \rightarrow A_n$, on produit un sous schéma $<XA_1 \ldots A_n, G[XA_1 \ldots A_n]>$,

3. Si aucun des sous schémas produits ne contient de clé, on ajoute le sous schéma <K, ∅>.

Algorithme de décomposition en 3NF

Entrée

<U, F>, schéma initial

Structure de données

S, ensemble de couples <U, F>

Sortie

Valeur finale de S: la collection de sous schémas

Algorithme

1) G := couverture_minimale(F) ;

2) S := ∅ ;

pourtout X telque X → A1, …, X → An

(i.e. soit les DF de G de partie gauche X)

faire

S := S ∪{<XA1…An, G[XA1…An]>};

fait ;

3) si aucun élément de S ne contient de clé

alors

K := une clé ;

S := S ∪{<K, {∅}>};

fsi ;

Remarques

1. C'est le fait de partir d'une couverture minimale qui assure que les sous schémas obtenus sont en 3NF.

2. La préservation des dépendances est, à partir de G, triviale.

3. C'est l'ajout d'une clé qui assure la validité de la décomposition.

4. Simplification : sans modifier les propriétés générales de l'algorithme, on peut simplifier le résultat S en retirant tout sous schéma contenu dans un autre sous schéma de S. Cette étape n'est pas nécessaire, mais elle simplifie le résultat en éliminant les redondances.

Exemples

• La base transport:<R (f, ville, frais), F : {f → ville, ville → frais, f → frais}>

1) G = {f → ville, ville → frais}

2) S = {<f ville, {f → ville}>, <ville frais, {ville → frais}>}

3) La clé f est utilisée dans S ; on n'ajoute rien

• U = {Cours, Prof, Heure, Salle, Etudiant, Formation}

F = {C → P, HS → C, HP → S, CE → F, HE → S}

1) F est minimal.

2) On obtient Δ= {CP, HSC, HPS, CEF, HES}.

3) Il n'y a rien à ajouter.

4.10. Décomposition valide en BCNF

Principe

Pour obtenir une décomposition valide avec des sous schémas en BCNF, on part d'un schéma initial <U0, F0>. S'il n'est pas en BCNF on va l'éclater en deux sous schémas puis on va itérer le processus sur chacun des sous schémas obtenus. On établit donc un arbre binaire de décompositions successives. La diminution stricte du nombre d'attributs d'un pas sur l'autre assure la terminaison sur chacune des branches.

Algorithme de décomposition en BCNF

Entrée

<U0, F0>, schéma initial

Structure de données

X, ensemble de couples <U, F>

Sortie

Valeur finale de X: la collection de sous schémas

Algorithme

Procédure BCNF (U: attributs, F: dépendances) ;
Début
si <U, F> en BCNF
alors
X := X ∪{<U, F>}
sinon
soit X→ A ∈ F+, X ne contenant pas de clé ;
BCNF (XA, F[XA]) ;
BCNF (U - A, F[U - A])
fsi
fin ;
X := ∅ ;
BCNF(U0, F0) ;

Exemple :

Soit la relation R(numéro, nom, spécialité, ville)vérifiant les DF :

Numéro→ nom

Ville →spécialité

La clé est (numéro, ville)

R est en 1FN

Première décomposition

Relation	**Clé**	**DF**	**FN**
R1(ville, spécialité)	ville	Ville →spécialité	BCNF
R2(numéro, ville, nom)	(numéro, ville)	Numéro→ nom	1FN

Deuxième décomposition

Relation	Clé	DF	FN
R21(numéro, nom)	ville	Numéro→ nom	BCNF
R22(numéro, ville)	(numéro, ville)	∅	BCNF

4.11. Décomposition sans perte d'information : algorithme d'Ullman

Cet algorithme permet de vérifier si un décomposition (R1,…,Rn) avec n >= 2 d'une relation R(A1,…, An) vérifiant un ensemble de DF F est une décomposition sans perte d'information ou non [5, 13, 14].

4.11.1. Formalisme de l'algorithme :

Function ullman (F,(R1, …,Rk),(A1,…,An))

Entrée :

- F : ensemble de DF
- (R1, …, Rk) une décomposition
- (A1,…,An) liste des attributs de R

Sortie :

- Ullman = vrai si la décomposition se fait Sans perte
- Ullman = faux si la décomposition se fait avec perte

Début

- Construire une matrice Mat(k,n)
- La colonne j dans Mat(k,n) correspond à l'attribut Aj de R
- La ligne i dans Mat(k,n) correspond à la relation Ri de la décomposition

Remplir de la matrice Mat(k,n) comme suit
A l'intersection d'une ligne i et d'une colonne j faire
Si l'attribut Aj de R appartient au schéma de Ri
    Alors placer le symbole aj
Sinon placer le symbole bij ;
encore ← vrai ;
Tant que encore = vrai faire
    Ya_Eu_Changement ← faux ;
    Fixer un ordre de parcours de DF de F
    Tant que il reste une DF à examiner ET encore = vrai faire
        Prendre une DF X→Y dans F ;
        Choisir 2 lignes dans Mat
        Si dans la colonne X (ou les colonnes X si X est composé)
        Les 2 ligne ont des symboles identiques
        Alors /* transformer les symboles de la colonne Y de ces 2 lignes comme suit */
        Ya_Eu_Changement ← vrai ;
        Si un des deux symboles est un symbole aj
            Alors remplacer l'autre par aj
            /* les 2 symboles sont de type bij */
            les deux par bij ou blj
        Fin si
        Fin si
    Fin tant que
    Si ∃ dans Mat une ligne remplie de symboles aj
        Alors Ullman ← vrai ; encore ← faux ;
    Sinon si Ya_Eu_Changement = vrai
        Alors Ullman ← faux ; encore ← faux ;

Fin si
Fin si
Fin tant que

Fin Ullman

4.11.2. Exemple d'application

Soit la relation R(numéro, nom, ville, spécialité) vérifiant les DF suivantes :

F = {ville→spécialité ; numéro→nom} et soit la décomposition suivante de R :

{R1(ville, spécialité) ; R2(numéro, nom) ; R3 (numéro, ville)}

Etape1 : construction de la matrice

	Numéro	Nom	Ville	Spécialité
R1	b11	b12	a3	a4
R2	a1	a2	b23	b24
R3	a1	b32	a3	b34

Etape 2 : On choisit une DF par exemple ville→spécialité

On remarque que seules R1 et R3 ont le même symbole sur la colonne ville (a3). On doit donc effectuer un changement dans la colonne spécialité. Donc on change sur la ligne R3 le symbole b34 par a4. on aura

	Numéro	Nom	Ville	Spécialité
R1	b11	b12	a3	a4
R2	a1	a2	b23	b24
R3	a1	b32	a3	a4

Il n'y a pas de lignes composée de symbole ai, on passe à l'autre DF.

On choisit la DF numéro→nom

On remarque que seules R2 et R3 ont le même symbole sur la colonne Numéro (a1).On doit donc effectuer un changement dans la colonne Nom. Donc on change sur la ligne R3 le symbole b32 par a2.On aura :

	Numéro	Nom	Ville	Spécialité
R1	b11	b12	a3	a4
R2	a1	a2	b23	b24
R3	a1	a2	a3	a4

On remarque qu'on a une ligne R3 composée uniquement de symboles ai. On conclut que la décomposition se fait sans perte d'information.