Cours Bases de données
3ème Année Systèmes d'Information

Chapitre 02
Le Modèle Relationnel

Fouad DAHAK
Enseignant-Chercheur
Chargé de cours Bases de données
Ecole Nationale Supérieure d'Informatique (ESI)
(f_dahak@esi.dz – http://dahak.esi.dz)

Table des matières

I. Concepts de base 4
I.1. Introduction 4
I.2. Pourquoi un tel succès 4
I.3. Concepts de base 4
I.3.1. Domaine 4
I.3.2. Produit cartésien 4
I.3.3. Relation 4
I.3.4. Extension d'une relation 5
I.3.5. Visions d'une relation 5
I.3.6. Attribut 5
I.3.7. Clé d'une relation 5
I.3.8. Schéma d'une relation 5
I.3.9. Clé étrangère 6
II. Passage de l'entité-association au modèle relationnel 6
II.1. Introduction 6
II.2. Règle 1 : Entité non faible 6
II.3. Règle 2 : Relation 1:n 6
II.4. Règle 3 : Relation n:m 7
II.5. Règle 4 : Entité Faible 7
II.6. Règle 5 : Généralisation / Spécialisation 7
II.7. Cas particulier : Association 1:1 8
II.8. Cas particulier : Entité avec un seul attribut 9
II.9. Reverse Engineering 9
III. Les Dépendances Fonctionnelles 11
III.1. Introduction 11
III.2. Définition 11
III.3. DF élémentaires 12
III.4. DF directes 12
III.5. DF triviales 12
III.6. Graphe de dépendances fonctionnelles 12
III.7. Les axiomes d'Armstrong 13
III.7.1. Couverture fonctionnelle 13
III.7.2. Définition de l'inférence 13
III.7.3. Les axiomes d'Armstrong 13
III.8. La fermeture transitive 14
III.9. La fermeture d'un attribut 14

III.10. La couverture minimale 15
III.11. Clé candidate 16
III.12. Clé et superclé 18
IV. Normalisation des relations 18
IV.1. Théorie de la normalisation 18
IV.2. Relation universelle 18
IV.3. Pourquoi normaliser ? 18
IV.4. Les formes normales 19
IV.4.1. Première forme normale (1NF) 19
IV.4.2. Deuxième forme normale (2NF) 19
IV.4.3. Troisième forme normale (3NF) 20
IV.4.4. Forme normale de Boyce Codd(BCNF) 21
IV.4.5. Résumé 21
IV.5. Formes normales de plus haut niveau 22
IV.5.1. Observation : 22
IV.5.2. Dépendances multivalués 22
IV.5.3. Quatrième forme normale (4NF) 23
V. Conception d'un schéma relationnel 24
V.1. Introduction 24
V.1.1. Théorème de Heath 24
V.1.2. Introduction 24
V.2. Approche par synthèse 24
V.2.1. Présentation 24
V.2.2. Algorithme par synthèse 24
V.2.3. Note 25
V.3. Approche par décomposition 25
V.3.1. Présentation 25
V.3.2. Décomposition sans perte 25
V.3.3. Algorithme de décomposition 25
V.3.5. Note 26

I. Concepts de base

I.1. Introduction

Le modèle relationnel a été défini par E.F Codd dans les années 70 et de nombreux chercheurs ont contribué à son développement. Les premiers SGBD bâtis sur ce modèle ont été SQL/DS et DB2 de IBM, d'où est né le langage de manipulation de bases relationnelles, SQL (Structured Query Language).

I.2. Pourquoi un tel succès

Simplicité de la structure des données

Une base relationnelle est composée de tables et est perçue par l'utilisateur comme un ensemble de tables et rien d'autre. Dans une table, une ligne correspond à un enregistrement et une colonne à un champ de cet enregistrement.

Simplicité des opérateurs

Toute opération relationnelle sur une table génère une nouvelle table, c'est-à-dire fonctionne sur un ensemble de données sans que l'on ait à se préoccuper de traiter successivement chacune des données récupérées par l'opération.

I.3. Concepts de base

I.3.1. Domaine

Définition 1 : Un domaine est un ensemble de valeurs atomiques.

I.3.2. Produit cartésien

Définition 2 : Le produit cartésien D1x D2x... x Dn est l'ensemble des tuples (n-uplets) V1,V2,....Vn> tel que quel que soit i Vi appartient à Di.

Exemple :

D1 = {Bleu,Blanc,Rouge} , D2 = {Vrai, Faux}.
D1 X D2 = (Bleu, Vrai); (Bleu, Faux); (Blanc, Vrai); (Blanc, Faux); (Rouge, Vrai); (Rouge, Faux)

I.3.3. Relation

Définition : Une relation est un sous-ensemble nommé du produit cartésien d'une liste de domaines. elle est notée ***R(A1:D1,…,An:Dn)*** où D1,…,Dn sont des domaines.
Note : On peut également noter la relation sans mentionner les domaines : ***R(A1,A2,…,An)***

Exemple

D1 = COULEUR
D2 = BOOLEEN

Couleur_Véhicule(Couleur : D1, Existe:D2)

Couleur_Véhicule	Couleur	Existe
	Bleu	FAUX
	Blanc	VRAI
	Rouge	VRAI

I.3.4. Extension d'une relation

Définition : L'extension d'une relation R(A1:D1,…,An:Dn) est un ensemble de n-uplets (V1,V2,...,Vn) tq
Note : L'extension d'une relation est variable au cours de la vie d'une base de données.

I.3.5. Visions d'une relation

Vision tabulaire du relationnel

- Une relation est une table à deux dimensions,
- Une ligne est un tuple,
- Un nom est associé à chaque colonne afin de la repérer indépendamment de son numéro d'ordre.

Vision Assertionnelle

A toute relation de schéma R(A1:D1,…,An:Dn) est associé un prédicat R tel que l'assertion R t est vraie si le n-uplet t appartient à l'extension de R et fausse sinon.

Exemple

On suppose la relation de schéma: Personne (Nom : Chaîne, Age : Entier, Marié : Booléen)
Et d'extension : {{Nom=''Hakim'' , Age=23 , Marié = Faux}; {Nom=''Lila'' , Age=36 , Marié = Vrai } }

Assertionnelle

Personne{Nom=''Hakim'', Age=23, Marié=Faux} **Vrai**
Personne{Nom=''Lila'', Age=35, Marié=Vrai} **Faux**
Personne{Nom=''Lila'', Age=23, Marié=Faux} **Faux**

Tabulaire

Nom	**Age**	**Marié**
Hakim	23	Faux
Lila	36	Vrai

I.3.6. Attribut

Un attribut est un nom donné à une colonne d'une relation, il prend ses valeurs dans un domaine.
Exemple : Nom, Age et Marié sont des attributs de Personne

I.3.7. Clé d'une relation

Définition : C'est un groupe d'attributs minimum qui détermine un tuple unique dans une relation.

Exemples:

– {modèle,Année} DANS Voiture --> Couleur, Type
– NSS Dans Personne
Contrainte : Toute relation doit posséder au moins une clé.
Note : Le concept de clé sera repris plus en détail dans la section traitant les dépendances fonctionnelles.

I.3.8. Schéma d'une relation

Définition : Nom de la relation, liste des attributs avec domaines, et liste des clés d'une relation.

Exemples: Voiture(Modèle: texte, Année:Int, Couleur:texte, Type: texte)

Intention et extension d'une relation : Un schéma de relation définit l'intention de la relation alors qu'une instance de table représente une extension de la relation.

Schéma d'une base de données : C'est l'ensemble des schémas des relations composant la base de données.

I.3.9. Clé étrangère

Définition : Groupe d'attributs devant apparaître comme clé dans une autre relation.
Les clés étrangères définissent les contraintes d'intégrité référentielles suivantes:

- Lors d'une insertion, la valeur des attributs doit exister dans la relation référencée;
- Lors d'une suppression dans la relation référencée les tuples référençant doivent disparaître;
- Elles correspondent aux liens entité-association obligatoires.

Exemple
PERSONNE (**NSS**, NOM, PRENOM)
VOITURE (**MODELE**, ANNEE, COULEUR, TYPE, ***NSS***)

CLES ETRANGERES : VOITURE.NSS **REFERENCE** PERSONNE.NSS

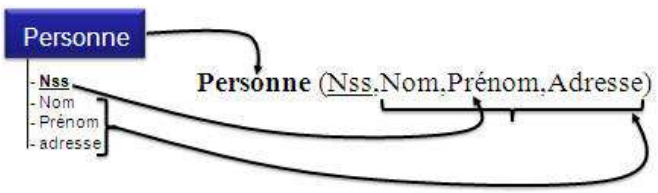
II. Passage de l'entité-association au modèle relationnel

II.1. Introduction

Le modèle Entité/Association est un modèle conceptuel permettant de modéliser une réalité à l'aide de trois concepts essentiels : l'entité, l'association et les attributs. au niveau du relationnel on ne traite que des relations. pour passer du premier modèle au second on a besoin de règles pour faire la correspondance entre les concepts de l'un et de l'autre.
Pour effectuer ce passage, on ne prend en considération que les cardinalités maximales, les cardinalités minimales donnent naissance à une contrainte de valeur NULL sur l'attribut qui deviendra une clé étrangère.

II.2. Règle 1 : Entité non faible

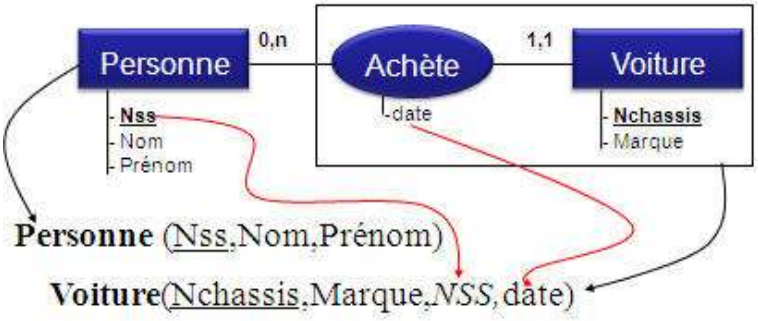
Une entité E non faible est représentée par une relation T dont les attributs simples sont les attributs de l'entité E et la clé de T est l'identifiant de E.



II.3. Règle 2 : Relation 1:n

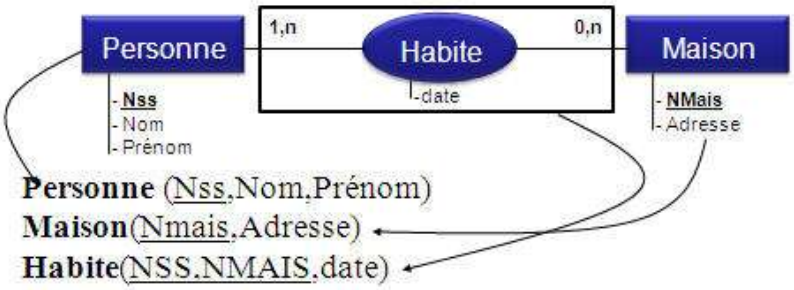
Dans le cas d'une relation 1:n (Père/fils), l'association n'est pas représentée par une relation, cependant ses attributs migrent vers la

relation représentant le fils et la clé du père migre vers le fils comme clé étrangère.



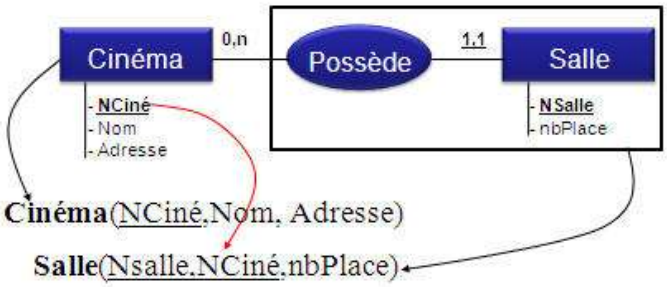
II.4. Règle 3 : Relation n:m

Dans le cas d'une relation n:m, l'association A est représentée par une relation T dont les attributs sont les attributs de A et la clé est la concaténation des clés des entités participant à l'association A.



II.5. Règle 4 : Entité Faible

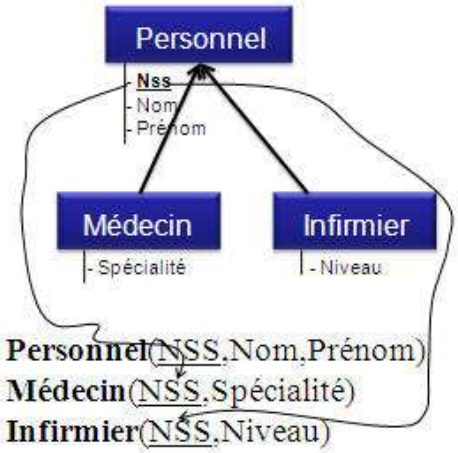
Le passage d'une entité faible à un schéma relationnel est identique à celui d'une association 1-n classique. La seule différence est que la clé du père migre vers le fils est entre dans la constitution de la clé primaire de ce dernier.



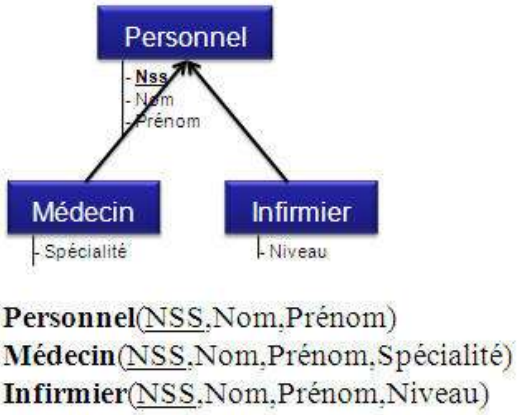
II.6. Règle 5 : Généralisation / Spécialisation

Dans le cas d'une généralisation / spécialisation, il existe trois manières de passer au schéma relationnel :

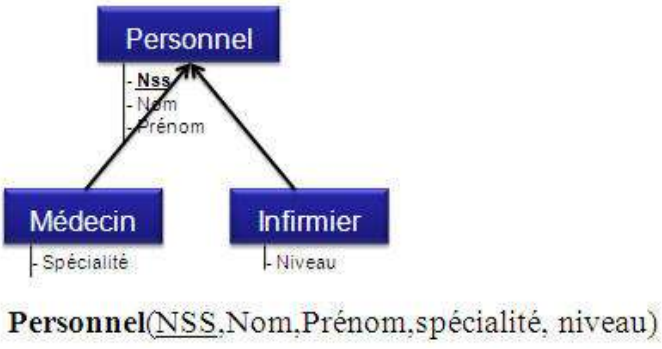
Premier Cas :



Deuxième Cas : Push down



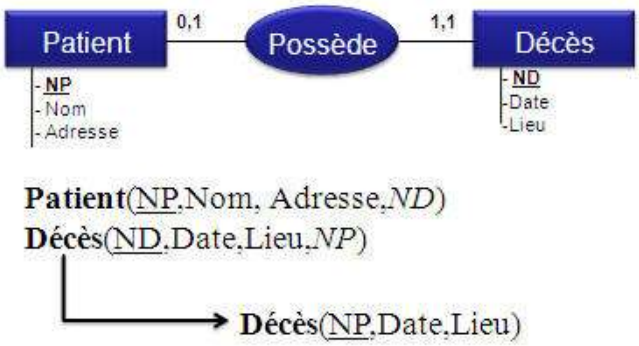
Troisième Cas : Push up



II.7. Cas particulier : Association 1:1

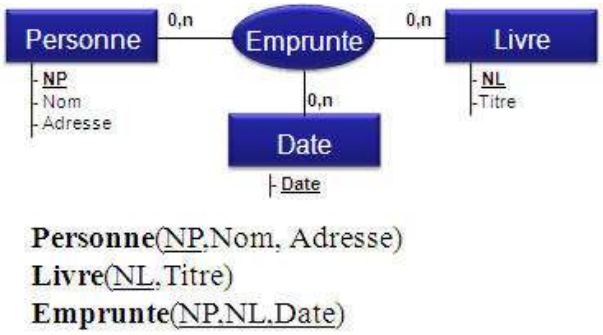
Dans le cas d'une association 1:1, la migration de la clé peut se faire dans un seul sens comme on peut le faire dans les deux sens. il est à noter que quand un cas pareil se présente il est presque

impossible que les cardinalités minimales des deux côtés soit toutes les deux égales à **1**, l'une d'elles est toujours un **0.** Ce qui fait que le sens le plus recommandé pour la migration de la clé est celui allant du côté de la cardinalité minimale **0** vers celui de la cardinalité minimale **1**.



II.8. Cas particulier : Entité avec un seul attribut

Quand on est en présence d'une entité ne possédant qu'un seul attribut qui est sont identifiant dans le modèle Entité Association et que les valeurs de cet attribut sont connus à l'avance ou ne sont pas assez importantes pour être sauvegardées. cette entité n'est pas représentée dans le schéma relationnel correspondant comme est le cas pour l'entité date.



II.9. Reverse Engineering

Le reverse engineering dans le case de la conception des bases de données consiste à déduire le modèle entité association à partir d'un schéma relationnel. Ce qui signifie de passer d'un modèle n'utilisant qu'un seul concept (relation) vers un autre modèle utilisant trois concepts (entité, association, cardinalités).
Le passage d'un schéma relationnel vers le modèle entité association peut être formalisé est écrit sous forme d'un algorithme. On remarque dans ce cas que ce qui permet le passage c'est la clé de la relation.
Avant tout, il faut ordonner les relations selon le nombre d'attributs composants la clé primaire allant du plus petit au plus grand. Ensuite, appliquer l'algorithme ci-dessous pour chaque relation.

Soit R(A1,A2,...,An) une relation dont la clé primaire est **X** tel que X=(A1,A2,...,Ap) avec 1<=p<=n.

if (X apparait ailleurs dans le schéma)
{
    if (X apparait en tant que clé primaire)
    {
        R est une entité faisant partie d'une spécialisation / Généralisation avec toutes les autres relations dont la clé primaire est X. Dans ce cas, l'entité génératrice est celle dont les attributs soit figurent tous dans les autres entités soit ne figurent pas du tout dans les autres entités. si l'entité génératrice n'est pas détectée il faut la créer avec comme identifiant X et les attributs sont ceux qui sont communs aux autres relations dont la clé est X.
    }
    else
    {
        if (X apparait en tant que clé étrangère)
        {
            R est une entité simple avec X comme identifiant.

            Toutes les relations où X apparait en tant que clé étrangère sont liées à R avec une association de type 1:n.
        }
        else
        {
            R est une entité simple avec X comme identifiant.

        }
    }
    else
    {
        if (Une partie de X apparait ailleurs en tant que clé primaire et l'autre partie n'apparait nul part ailleurs)
        {
            R est une entité faible avec comme identifiant la partie de X qui n'apparait pas ailleurs, R sera reliée à la relation dont la clé est la partie de X avec une relation Père fils.
        }
        else
        {
            if (X se composent de plusieurs parties dont chacune apparait ailleurs en tant que clé primaire)
            {

```
                                R est une association entre les
                                relations dont les clés sont ces
                                parties de X
                        }
                        else
                        {
                                if (une de ces parties de X n'apparait
                                nul part ailleurs alors que les autres
                                apparaissent ailleurs en tant que des
                                clés primaires)
                                {
                                        R est une association entre
                                        ces relations dont les clés
                                        sont ces parties de X et une
                                        entité ne comportant qu'un
                                        seul attribut qu'est son
                                        identifiant et qui est cette
                                        partie de X qui n'apparait nul
                                        part ailleurs.
                                }
                        }
                }
        }
}
```

III. Les Dépendances Fonctionnelles

III.1. Introduction

La notion de dépendance fonctionnelle permet d'établir des liens sémantiques entre attributs ou groupe d'attributs. On dit qu'il existe une dépendance fonctionnelle de A vers B ou A détermine B ou que B dépend fonctionnellement de A et on note A → B où A est dit source de la dépendance fonctionnelle et B sa destination.

III.2. Définition

Étant donné une relation R, nous disons qu'il y a dépendance fonctionnelle (DF) X de R vers Y de R si à une valeur de X est associée au plus une valeur de Y.
Question à se poser : connaissant la valeur de X, puis-je connaître la valeur de Y ?
Soit R(A1, A2,…,An) une relation, et X et Y des sous-ensembles de {A1,A2,…,An}. On dit que X→Y si pour toute extension r de R, pour tout tuples t1, t2 de r, on a :
$\prod_x(t1)=\prod_x(t2) \Rightarrow \prod_y(t1)=\prod_y(t2)$

Exemple

Voiture(Châssis, Couleur,Type,Marque,Puissance)
Châssis → Couleur
Type → Marque
Type → Puissance

III.3. DF élémentaires

Une dépendance fonctionnelle A → B est dite élémentaire, s'il n'existe pas C, inclus dans A, qui assure lui-même une dépendance fonctionnelle C → B.

Exemple

1. Ref_Article → Nom_Article
2. Num_Facture, Ref_Article → Qte_Article
3. Num_Facture, Ref_Article → Nom_Article

Les dépendances fonctionnelles 1 et 2 sont élémentaires alors que la 3 ne l'est pas. parce qu'il existe une partie de la source de la df qui assure elle-même la dépendance fonctionnelle : c'est Ref_Article au niveau de la df 1.

III.4. DF directes

Une dépendance fonctionnelle A → B est dite directe, s'il n'existe pas un attribut C qui engendrerait une dépendance fonctionnelle transitive A → C → B.

Exemple

1. Num_Facture → Num_Représentant VRAI
2. Num_Représentant → Nom_Représentant VRAI
3. Num_Facture → Nom_Représentant FAUX

Les dépendances fonctionnelles 1 et 2 sont directes alors que la troisième ne l'est pas parce qu'elle peut être déduite par transitivité à partir des deux premières.

III.5. DF triviales

Une dépendance fonctionnelle est dite triviale s'il est impossible qu'elle ne soit pas satisfaite. Une dépendance fonctionnelle est triviale si et seulement si le membre droit (destination) est un sous-ensemble du membre gauche (source).

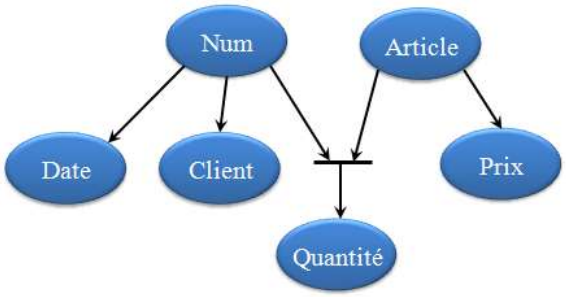
A1, A2, …, An → B1, B2, …, Bm
Triviale: Si B1, B2, …, Bm est un sous-ensemble de A1,A2,..,An
Non triviale : Si au moins un Bi n'appartient pas à A1,A2,..,An
Complètement non triviale : Si aucun des Bi n'appartient à A1,A2,…,An

III.6. Graphe de dépendances fonctionnelles

Un ensemble F de dépendances fonctionnelles peut être représenté avec un graphe orienté où chaque nœud représente un attribut et les arcs les dépendances fonctionnelles entre les attributs.

Exemple

Commande(Num,Date,Client,Article,Prix,Quantité)
Num → Date
Num → Client
Article → Prix
Article,Num→Quantité



III.7. Les axiomes d'Armstrong

III.7.1. Couverture fonctionnelle

On appelle couverture fonctionnelle d'une relation l'ensemble des dépendances fonctionnelles qui lui sont associées.

III.7.2. Définition de l'inférence

Si F est une couverture fonctionnelle de la relation R, On dit qu'une dépendance fonctionnelle : X → Y est inférée (déduite) de F si X → Y est valide pour tout tuple d'une extension de R.

III.7.3. Les axiomes d'Armstrong

Les Axiomes d'Armstrong sont un ensemble de règles qui permettent d'effectuer des inférences de dépendances fonctionnelles à partir d'autres dépendances fonctionnelles.

(i) Réflexivité $Y \subseteq X \Rightarrow X \rightarrow Y$

(ii) Augmentation $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

(iii) Transitivité $X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Z$

Note : Il est prouvé que les règles d'Armstrong sont fondées et complètes. Ce qui signifie que toute dépendance fonctionnelle déduite en utilisant la réflexivité, l'augmentation ou la transitivité est une dépendance fonctionnelle inférée et que toute dépendance fonctionnelle qui peut être inférée de F, peut l'être en utilisant seulement les axiomes mentionnés ci-dessus.
Il existe cependant d'autres règles d'inférences qu'on peut utiliser :

Projection $X \rightarrow YZ \Rightarrow X \rightarrow Y$

Union (addition) $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ$

Pseudo-transitivité $X \rightarrow Y, WY \rightarrow Z \Rightarrow WX \rightarrow Z$

Décomposition $X \rightarrow YZ \Rightarrow X \rightarrow Y, X \rightarrow Z$

Auto-Détermiation $X \rightarrow X$

Composition $X \rightarrow Y, V \rightarrow Z \Rightarrow XV \rightarrow YZ$

Augmentation à gauche $X \rightarrow Y \Rightarrow XW \rightarrow Y$

III.8. La fermeture transitive

Si F est une couverture fonctionnelle de la relation R, la fermeture transitive de F notée F+ est l'ensemble des dépendances fonctionnelles qui peuvent être inférées de F par transitivité.

Exemple

F={Châssis → Type; Type → Marque; Type → Puissance}
F+=F UNION {Châssis → Marque; Châssis → Puissance}

Note : Deux ensembles de dépendances fonctionnelles sont équivalents s'ils ont la même fermeture transitive.

III.9. La fermeture d'un attribut

La fermeture d'un attribut X par rapport à F notée X+(F) est l'ensemble des attributs fonctionnellement dépendant de X sous la couverture fonctionnelle F.

Algorithme de calcul de X+

```
Result = X
Tant Que  (Result évolue)
        Pour chaque (DF : Y→ Z dans F)
        {
            SI (Y Inclus dans result) alors result=result UNION  Z
        }
}
```

Algorithme détaillé

```
Résultat : Ensemble d'attribut; // variable qui contiendra le résultat
Nouveau : Booléen // pour savoir si on a évolué ou non

Début
        Résultat = X
        Nouveau = Vrai
        Tant Que Nouveau Faire
        Début
                Nouveau = Faux
                Pour Chaque f Î F Faire
                        Si gauche(f) Í Résultat Alors
                                Résultat=Résultat È droite(f)
                                Nouveau = Vrai
                        FSI
                FPour
        FTQ
        Retourner Résultat
Fin
```

Exemple

Employé(NE, ENom, NP, PNom, Loc, Temps)
F = { NE → ENom ; NP → {PNom, Loc} ; {NE,NP} → Temps}
NE+ = {NE , ENom }

NP+ = {NP, PNom, Loc }
{NE,NP}+ = {NE,NP,ENom,PNom,Loc,Temps}

III.10. La couverture minimale

Définition : Une couverture minimale appelée aussi couverture irrédondante et notée IRR(F) est l'ensemble F de dépendances fonctionnelles élémentaires associé à un ensemble d'attributs vérifiant les propriétés suivantes:

- Aucune dépendance dans F n'est redondante;
- Toute dépendance fonctionnelle élémentaire des attributs est dans F+;

C'est le sous-ensemble minimal de dépendances fonctionnelles permettant de générer toutes les autres.

Algorithme de calcul de IRR(F)

Cet algorithme consiste à détecter les dépendances fonctionnelles redondantes et les écarter. Une dépendance est dite redondante si on peut la déduire à partir des dépendances restantes en utilisant les axiomes d'Armstrong.

Couverture(F);
Begin
IRR := F
Remplacer Chaque DF X→(A1,…,An) par n DF X→A1,…, X→An
Pour chaque DF : f =X → A dans IRR
SI (IRR – {f}) Implique {f} **Alors** IRR = IRR - {f}
FPOUR
Return IRR
Fin

Algorithme détaillé de IRR(F)

Cet algorithme est le même que le précédent quoiqu'on détaille ici la notion de redondance de dépendances fonctionnelles. On dit donc qu'une dépendance fonctionnelle *f* est redondante si la fermeture de sa partie gauche sur l'ensemble des dépendances restantes (F- *f*) comporte la partie droite de *f*. ce qui signifie que pour une dépendance fonctionnelle X→Y le lien sémantique entre X et Y est gardé même en enlevant *f*.

Procédure CouvertureMinimale(F:Ensemble de DF)
Var CM : Ensemble de DF
Début
CM = F
Décomposer les parties droites de DF
Pour chaque DF f : X→Y MC snad **Faire**
Début
Calculer X+(F-{f})
Si Y est inclus dans X+ **Alors** CM=CM-{f}
FPour
Retourner CM
Fin

Exemple

M → A

HDA → S
S → V
HDS → AMV
HDJ → S
HDM → VS
J → E

Calcul de la couverture minimale
Décomposition des parties droites
M →A
HDS → A
HDM → S
HDS →M
HDS → V
HDA → S
HDJ →S
J → E
S → V
HDM → V

Traitement des DF

1. M → A : M+={M}. A n'est pas inclus dans M+ donc la df n'est pas redondante
2. HDS → A : HDS+={H,D,S,M,A,V,E,S}. A est inclus dans HDS+ donc la df est redondante
3. HDM → S : HDM+={H,D,M,A,V,S}. S est inclus dans HDM+ donc la df est redondante
4. HDS → M : HDS+={H,D,S,A,V}. M n'est pas inclus dans HDS+ donc cette df n'est pas redondante
5. HDS → V : HDS+={H,D,S,A,M,V}. V est inclus dans HDS+ donc la df est redondante
6. HDA → S: HDA+={H,D,A}. S n'est pas inclus dans HDA+ donc cette df n'est pas redondante
7. HDJ → S: HDJ+={H,D,J,E}. S n'est pas inclus dans HDJ+ donc cette df n'est pas redondante
8. J →E : J+={J}. E n'est pas inclus dans J+ donc cette df n'est pas redondante
9. S →V : S+={S}. V n'est pas inclus dans S+ donc cette df n'est pas redondante
10. HDM→V : HDM+={H,D,M,A,S,V}. V est inclus dans HDM+ donc la df est redondante.

La couverture minimale est CM={M → A; HDS → M; HDA → S; HDJ → S; J → E; S → V}

III.11. Clé candidate

Il existe plusieurs algorithmes de recherche des clés d'une relation comme ceux de

1. FADOUS R; FORSYTH J : Finding Candidate Keys of relational Databases, ACM SIGMOD, 1975
2. LUCCHESI CL; ORSBORN SL: candidate keys of relations, Technical report, U. or Waterloo, Ontario, Canada, 1976.
3. KUNDU S, An Impoved Algorithm for finding a Key of a relation., Conf. PODS, 1975.

Nous donnons ici le principe de recherche d'une clé et un algorithme détaillé.

Définition : Un ensemble X d'attributs d'une relation R(Δ) est une clé si et seulement si : X+=Δ et pour tout Y inclus dans X, Y+ n'est pas égal à Δ.

Algorithme de recherche d'une clé

Soit une relation R.
SI R ne contient qu'un seul attribut **ALORS**
cet attribut forme l'unique clé candidate de R
SINON
Soient X et Y deux ensembles non vides disjoints d'attributs de R tel
que X **Union** Y = R.
SI on a X → Y Avec X minimal **ALORS**
X est une clé candidate de R //*il peut y en avoir plusieurs*
SINON //*Aucun X ne peut être trouvé ainsi*
l'unique clé candidate de R est formée de tous ses attributs.
FSI
FSI

Algorithme détaillé de recherche de clé

Procedure FindKey(R)
Variable
C : ensemble d'ensembles d'attribut; //*les clés candidates*
X : ensemble d'attributs
Fonction
Partie(X) //*donne tous les ensembles obtenu en combinant les attributs de X*
Min(C) //*Renvoi l'ensemble minimal des ensembles de C*
Début
C={}
Pour chaque X de Partie(Δ) **Faire**
Si X+=Δ Alors C=C+X
FPour
Retourner Min(C);

Fin

Amélioration

Source : On dit qu'un attribut est une source si et seulement si cet attribut n'apparaît dans aucune partie droite des dépendances fonctionnelles de F.
Puits : On dit qu'un attribut est un puits si et seulement si cet attribut n'est pas une source et n'apparaît dans aucune partie gauche des dépendances fonctionnelles de F ou s'il apparaît alors la partie droite de cette dépendance fonctionnelle doit être un puits.
Note : Toute clé de R ne contient que des sources de R et ne contient aucun puits. Donc pour améliorer l'algorithme précédent, on ne considère que les attributs qui ne sont pas des puits.
Observations : Dans le cas où une relation possède plusieurs clés candidates, on ne peut trouver de sources. Dans ce cas, on détecte les puits de façon récursive puis le reste des attributs font partie des

clés candidates. On Exécute l'algorithme précédent pour cet ensemble d'attributs.

Exemple

M → A
HDA → S
S → V
HDS →AMV
HDJ → S
HDM → VS
J →E
Trouver la clé de R(A,M,H,D,S,V,J,E)

Solution

Les sources de R sont : H, D, J
Les Puits de R sont : E, V
La clé de R est **HDJ** car HDJ+=Δ et pour tout Y de Partie(HDJ), Y+ n'est pas égal à Δ
Preuve :
1. HDJ+={A,M,H,D,S,V,J,E}
2. H+={H}
3. D+={D}
4. J+={J,E}
5. HD+={H,D}
6. HJ+={H,J,E}
7. DJ+={D,J,E}

III.12. Clé et superclé

La clé est un sous-ensemble X des attributs d'une relation R(A1,A2,…,An) tel que:
1. X→A1,A2,…,An
2. Il n'existe pas de sous-ensemble Y inclus dans X tel que Y→A1,A2,...,An

Une clé est un ensemble minimal d'attributs qui détermine tous les autres.
Un ensemble d'attributs qui inclut une clé est appelé superclé.
Indication : Pour vérifier que X est minimal, il faut contrôler qu'il n'est pas possible de prendre un attribut de X pour le mettre dans Y tout en maintenant X →Y.

IV. Normalisation des relations

IV.1. Théorie de la normalisation

Cette théorie est basée sur les dépendances fonctionnelles qui permettent de décomposer l'ensemble des informations en diverses relations. Chaque nouvelle forme normale marque une étape supplémentaire de la progression vers des relations présentant de moins en moins de redondance.

IV.2. Relation universelle

La relation universelle est la relation qui regroupe toutes les informations à stocker.

IV.3. Pourquoi normaliser ?

Pour limiter les redondances de données,
Pour limiter les pertes de données,
Pour limiter les incohérences au sein des données,
Pour améliorer les performances des traitements.

IV.4. Les formes normales

IV.4.1. Première forme normale (1NF)

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

Conséquences

Un attribut représente une donnée élémentaire du monde réel.
Un attribut ne peut désigner, ni une donnée composée d'entités de nature quelconque, ni une liste de données de même nature.

Exemple

Pers1(nom, prénom, rueEtVille, prénomEnfants)
Pers2(nom, prénom, nombreEnfants)
Les valeurs des attributs rueEtVille et prénomEnfants ne sont pas simples.

Normaliser en première forme normale

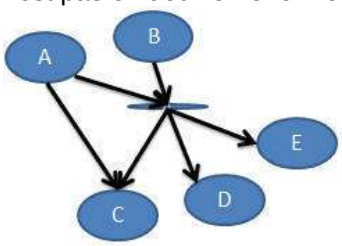
Il existe deux solutions pour normaliser une relation en première forme normale:

1ère solution : Créer autant d'attributs que le nombre maximum de valeurs de l'attribut multi-valué (stockage horizontal). On utilise cette solution quand le nombre maximum de valeurs est connu à l'avance et qu'il ne risque pas de changer plus tard. par exemple pour représenter les parents d'une personne, on sait qu'une personne ne peut avoir plus de deux parents donc on décompose en deux attributs un pour le père et un pour la mère.

2ème solution : Créer une nouvelle relation comportant la clef de la relation initiale et l'attribut multi-valué puis éliminer l'attribut multi-valué de la relation initiale (stockage vertical). Cette solution est utilisée quand on ne connait pas le nombre maximum de valeur ou si ce dernier et trop important.

IV.4.2. Deuxième forme normale (2NF)

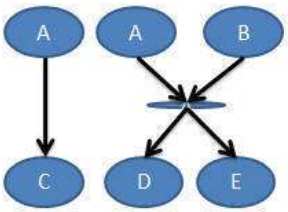
Une relation est en deuxième forme normale si et seulement si : elle est en première forme normale et tout attribut n'appartenant pas à une clé ne dépend pas que d'une partie de cette clé.
la relation suivante n'est pas en deuxième forme normale.



Exemple

Enseignement(**NUM,CODE_MATIERE**,NOM,VOLUME_HORAIRE)
Avec: NUM→Nom
Normaliser en deuxième forme normale
Pour normaliser une relation en deuxième forme normale, il faut :

- isoler la DF partielle dans une nouvelle relation;
- enlever la cible de cette DF de la relation initiale;



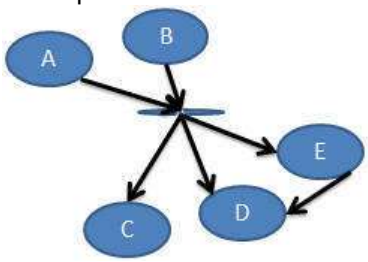
Normalisation de l'exemple ci-dessus

Enseignement(**NUM**,**CODE_MATIERE**,VOLUME_HORAIRE)
Enseignant(**NUM**,NOM)
Note

- Une relation en 1NF dont la clé est mono-attribut (se compose d'un seul attribut) est forcément en 2NF.
- La deuxième forme normale traduit le fait que les attribut non clé dépendent complètement de la clé.

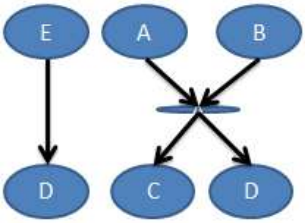
IV.4.3. Troisième forme normale (3NF)

Une relation est en troisième forme normale si et seulement si : elle est en deuxième forme normale et tout attribut n'appartenant pas à une clé ne dépend pas d'un autre attribut non clé.
La relation suivante n'est pas en troisième forme normale.



Exemple
ENSEIGNANT(**NUM**, NOM, CATÉGORIE, CLASSE, SALAIRE) avec CATÉGORIE,CLASSE → SALAIRE
Normalisation en troisième forme normale
pour normaliser une relation en troisième forme normale il faut:

- Isoler la DF transitive dans une nouvelle relation,
- Éliminer la cible de la DF de la relation initiale.

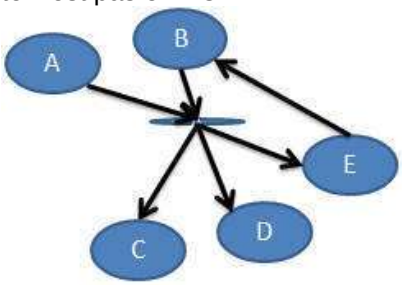


Exemple précédent

ENSEIGNEMENT(**NUM**, NOM, CATÉGORIE, CLASSE)
SALAIRE(**CATÉGORIE, CLASSE**, SALAIRE)
Note
La 3NF exprime le fait que tous les attribut non clé dépendent complètement et uniquement de la clé de la relation.

IV.4.4. Forme normale de Boyce Codd(BCNF)

Une relation est en Boyce Codd forme normale si et seulement si : elle est en deuxième forme normale et toute source de dépendance fonctionnelle est une clé primaire minimale.
La relation suivante n'est pas en BCNF.

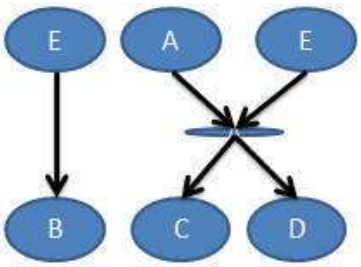


Exemple

ADRESSE(**Rue, Commune, Bureau-Postal**, CodePostal) avec
CodePostal → Commune
Remarque
SI R est en BCNF alors elle est en 3 NF
Preuve : Si R est en BCNF alors quelque soit X TQ X → U, X est une clé minimale. Supposons qu'il existe Y, Z appartenant à U tel que Y → Z, mais dans ce cas Y est une clé minimale. On ne peut donc jamais avoir de transitivité avec deux attributs non clé.
SI R est en 3 NF alors elle n'est pas forcément en BCNF.

Normalisation en Boyce Codd forme normale

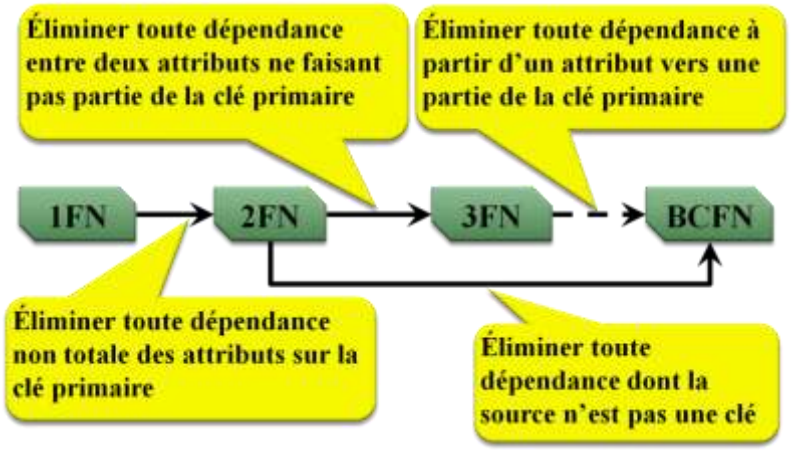
pour normaliser une relation en Boyce Codd forme normale il faut:
- Isoler la DF problématique dans une nouvelle relation;
- Éliminer la cible de cette DF et la remplacer par sa source dans la relation initiale.



IV.4.5. Résumé

- La première forme normale exprime le fait que toutes les valeurs des attributs sont atomiques (simples, non composées).
- La seconde forme normale exprime le fait que tous les attributs non clé d'une relation dépendent **complètement** de la clé de cette dernière et non pas d'une seule partie de cette clé.

- La troisième forme normale ajoute une autre restriction à la seconde forme normale en exprimant le fait que tous les attributs non clé dépendent complètement et **uniquement** de la clé de la relation.
- La Boyce Codd forme normale pousse plus loin la restriction de la troisième forme normale en exprimant le fait que dans toute relation en deuxième forme normale, s'il existe une dépendance fonctionnelle alors sa partie gauche est forcément une clé de cette relation.
- Le passage d'une forme normale à une autre est exprimé par le schéma ci-dessous.



Attention

- On ne peut passer vers la 2NF que si la relation est en 1NF, on ne peut passer vers la 3NF que si la relation est en 2FN, mais le passage vers la BCNF n'exige pas la 3NF mais uniquement la 2NF.
- Même si la BCNF n'exige pas une 3FN, mais le passage vers la BCNF implique forcément la 3NF.

IV.5. Formes normales de plus haut niveau

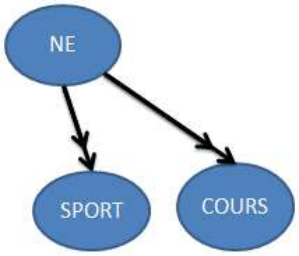
IV.5.1. Observation :

La BCNF n'est pas suffisante pour éliminer les redondances.

NE	Cours	Sport
100	BDD	Tennis
100	RESEAU	Tennis
100	BDD	Foot
100	BDD	Vélo
100	RESEAU	Vélo
200	RESEAU	Vélo

IV.5.2. Dépendances multivalués

Soit R(A1,A2,…,An) un schéma de relation, et X et Y des sous-ensembles de A1,A2,…,An. On dit que X -->> Y (X multidétérmine Y, ou il y a une dépendance multivaluée de Y sur X) si, étant donné des valeurs de X, il y a un ensemble de Y associé et cet ensemble est indépendant des autres attributs Z=R-X-Y de la relation R.



Exemple

Remarque : X-->> Y Alors X -->>Ai − {X U Y}.
un employé est qualifié pour effectuer un certain type d'intervention sur certaines marques de voiture.

– Employé(Nom-Employé)
– Intervention(Type-Intervetion)
– Constructeur(Marque)
– Intervenir(Nom-Employé, Type-Intervetion, Marque)

Supposons qu'un employé est capable d'effectuer chacun des types d'interventions sur chacune des marques de voitures. Dans ce cas, il existe des dépendances multivaluées dans la relation Intervenir :

Nom-Employé -->>Type-Intervention et Nom-Employé -->>Marque.
La relation suivante est en BCNF mais il existe encore des redondances

N°Article	Taille	Couleur
Pull	38	Bleu
Pull	40	Bleu
Pull	42	Bleu
Pull	44	Bleu
Pull	38	Vert
Pull	40	Vert
Pull	42	Vert
Pull	44	Vert

IV.5.3. Quatrième forme normale (4NF)

Une relation est en quatrième forme normale si et seulement si les seules dépendances multivaluées élémentaires sont celles dans lesquelles une superclé détermine un attribut.
R n'est pas en 4NF si l'on peut trouver une Dépendance multivaluée (DM) de la forme X-->>Y où X n'inclut pas une clé de R.
R est en 4NF si et seulement si chaque fois qu'on a X-->>Y alors les autres attributs de R dépendent de X.

Normaliser en Quatrième forme normale

Soit R une relation et F un ensemble de dépendances (fonctionnelles et multivaluées).
Si R n'est pas en 4NF, soit une dépendance X -->>Y de F+ telle que

- Y n'est pas vide;

- Y n'est pas inclut dans X
- XY sont inclus dans R et X n'est pas une superclé.

On décompose R en R1=XY et R2=X(R-XY)

V. Conception d'un schéma relationnel

V.1. Introduction

V.1.1. Théorème de Heath

Toute relation R(X,Y,Z) est décomposable sans perte d'information en R1=π[X,Y]R et R2=π[X,Z]R s'il y a dans R une dépendance fonctionnelle de X vers Y (X→Y).

V.1.2. Introduction

Le processus de conception d'un schéma relationnel dont la qualité déterminera la pertinence et l'efficacité de la base de données, doit prendre en compte les critères suivants :

- Normalité;
- Préservation des contenus;
- Préservation des dépendances fonctionnelles.

A partir de l'ensemble des dépendances fonctionnelles, il est possible de construire un bon schéma relationnel en exploitant des manipulations des dépendances fonctionnelles (recherche de clé) et des algorithmes de conception de schéma relationnel.
Il existe deux approches possibles pour concevoir un schéma relationnel, l'approche par synthèse et l'approche par décomposition.

V.2. Approche par synthèse

V.2.1. Présentation

L'approche par synthèse consiste à recomposer des relations à partir d'un ensemble d'attributs indépendants et de la couverture minimale d'un ensemble de dépendances fonctionnelles.

V.2.2. Algorithme par synthèse

Procédure Synthèse(R:Relation universelle; F:Ensemble des DF)
Début
1. Calculer la couverture minimale CM(F);
2. Partitionner **CM(F**) en F_1, F_2,…, F_n Tel que toutes les DF d'un même groupe aient la même partie gauche.
3. Construire les $R_i(\Delta_i)$ avec Δ_i constitué de l'union des attributs de F_i
4. **S'il** reste des attributs isolés **Alors** on les regroupe dans une relation ayant comme clé l'ensemble de ces attributs.

Fin;

Exemple
Soit R(A,B,C,D,E) et l'ensemble des dépendances fonctionnelles
A→B
A→C
C,D → E
B →D

1. La seule clé est A

2. L'ensemble des dépendances fonctionnelles fournies est une couverture minimale;
3. F1={A → B; A → C} F2={B → D} F3={C,D → E}
4. R1(**A**,B,C), R2(**B**,D), R3(**C,D**,E).

V.2.3. Note

Les relations obtenues sont en troisième forme normale grâce au fait d'avoir travaillé sur la couverture minimale ce qui signifie que toutes les dépendances fonctionnelles partielles et transitives ont été écartées.
Le résultat obtenu n'est pas toujours optimal.
L'union des Fi donne CM(F) donc il n'y a pas de perte de dépendances fonctionnelles.

V.3. Approche par décomposition

V.3.1. Présentation

L'approche par décomposition consiste à remplacer la relation R(A1,A2,…,An) par une collection de relations R1,R2,…,Rp obtenues par projection de R sur des sous-ensembles d'attributs dont l'union contient tous les attributs de R.

V.3.2. Décomposition sans perte

Une décomposition est dite sans perte si pour toute extension de R on ait

V.3.3. Algorithme de décomposition

Procédure Décomposition(
R:Relation universelle;
F:Esemble des DF)
Début
Result={R}
SI (Il existe Ri de Result TQ Ri n'est pas en BCNF) **Alors**
Début
Chercher une DF non triviale X --> Y dans F+ sur Ri tq X n'est pas une clé;
Décomposition((Result-Ri)È(Ri-Y)È(XY));
Fin;
Pour tout Ri(Di) et Rj(Dj) TQ Di Ì Dj dans result **Faire** Supprimer(Ri)
Return Result;

Fin;

Exemple

Soit R(A,B,C,D,E) et l'ensemble des DF
A→B
A→C
C,D → E
B →D
La seule clé est A
Result=R;
R n'est pas en BCNF à cause de B → D on décompose
Result={R1(**B**,D), R2(**A**,B,C,E)}
R2 n'est pas en BCNF à cause de B,C → E (obtenu par pseudo transitivité entre B -> D et C,D → E), on décompose
Result={R1(**B**,D), R21(**A**,B,C),R22(**B,C**,E)}

toutes les Ri sont en BCNF, l'algorithme s'arrête.
Aucune relation imbriquée;
Résultat est {R1(**B**,D), R21(**A**,B,C),R22(**B,C**,E)}

On refait le même exemple mais en commençant par la DF C, D → E

1. La seule clé est A
2. Result=R;
3. R n'est pas en BCNF à cause de C,D → E on décompose
    1. Result={R1(**C,D**,E), R2(**A**,B,C,D)}
    2. R2 n'est pas en BCNF à cause de B → D, on décompose
        1. Result={R1(**C,D**,E), R21(**B**,D),R22(**A**,B,C)}
        2. toutes les Ri sont en BCNF, l'algorithme s'arrête.
        4. Aucune relation imbriquée;
        5. Résultat est {R1(**C,D**,E), R21(**B**,D),R22(**A**,B,C)}

V.3.5. Note

Avant d'utiliser l'algorithme de décomposition, il faut d'abord regrouper les dépendances fonctionnelles qui ont la même partie gauche pour éviter d'avoir des relations avec la même clé et des attributs différents que, en réalité, ne forment qu'une seule relation.
Le résultat fourni par l'algorithme de décomposition dépend de l'ordre des dépendances fonctionnelles traitées.
Il est certain qu'une décomposition en BCNF est préférable à une décomposition en 3NF sauf si celle-ci ne conserve pas les dépendances fonctionnelles. Dans ce cas, il serait préférable de rester avec une 3NF.