Chapitre 9
Conception et normalisation d'une BD

- comment choisir un bon schéma de BD relationnelle?
  - il nous faudrait disposer d'une mesure en bonne et due forme permettant de décider qu'un regroupement des attributs dans un schéma relationnel est meilleur qu'un autre.
- deux niveaux d'évaluation de la qualité
  - point de vue utilisateur : sémantique des données
  - point de vue développeur : manipulation et stockage des données
- théorie de la mesure de la qualité d'un schéma : dépendances fonctionnelles

La conception de base de données peut être entreprise selon deux démarches:

- du bas vers le haut (conception ascendante, également appelée *conception par synthèse)* considère les liaisons de base entre les attributs individuels comme point de départ et s'en sert pour construire des schémas relationnels. Cette approche n'est pas souvent adoptée en pratiquel car elle impose la collecte d'un grand nombre de liaisons binaires entre des attributs pour constituer le point de départ.
- du haut vers le bas (conception descendante, également appelée *conception par analyse)* commence par considérer plusieurs regroupements d'attributs dans des relations qui existent naturellement, par exemple dans une facture, un formulaire, un rapport, etc. Les relations sont ensuite analysées de manière individuelle et collective, ce qui conduit à les décomposer jusqu'à obtenir toutes les propriétés souhaitables.
- La théorie décrite dans ce chapitre est applicable à ces deux approches mais il est plus facile de la mettre en œuvre dans le cadre d'une approche descendante.

L’approche proposée

1. faire l’approche descendante
   - définit modèle ER
   - transforme en modèle relationnel
   - applique les critères informels
2. effectuer la synthèse relationnelle
   - décompose les schémas de relation jusqu'à obtenir des schémas qui satisfont un certain niveau de forme normale (3NF, BCNF, 4NF, 5NF);
   - il existe des algorithmes pour faire cette décomposition.

9.1 Principes informels pour la conception des schéma relationnels

- Cette section aborde quatre mesures informelles de la qualité de la conception d'un schéma relationnel
  - la sémantique des attributs;
  - la réduction des valeurs redondantes dans les tuples ;
  - la réduction des valeurs nulles dans les tuples ;
  - l'élimination des tuples parasites.
- Ces mesures ne sont pas toujours indépendantes les unes des autres.

9.1.1 Sémantique des attributs des relations

Principe 1

Concevez un schéma relationnel de façon à ce qu'il soit facile d'en expliquer la signification. Ne combinez pas des attributs provenant d'entités et de liaisons de différents types en une même relation. Si un schéma relationnel correspond à un seul type d'entité ou de liaison, sa signification est évidente. Si la relation mélange différentes entités et liaisons, des ambiguïtés sémantiques surgissent et la relation devient difficile à expliquer.

.

..

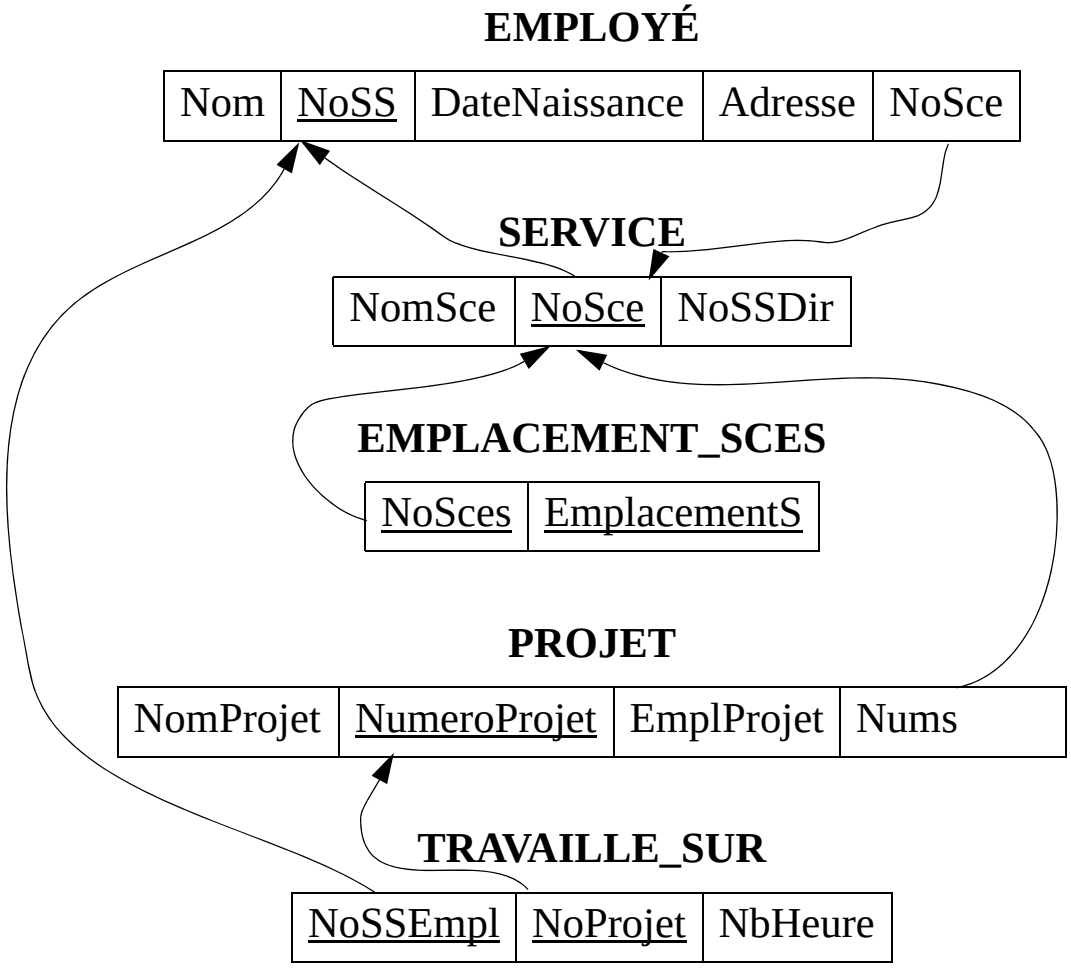

EMPLOYÉ
Nom
NoSS
DateNaissance
Adresse
NoSce
SERVICE
NomSce
NoSce
NoSSDir
EMPLACEMENT_SCES
NoSces
EmplacementS
PROJET
NomProjet
NumeroProjet
EmplProjet
Nums
TRAVAILLE_SUR
NoSSEmpl
NoProjet
NbHeure

figure 9.1

figure 9.2

EMPLOYE

NOM	NoSS	DNAISS	ADRESSE	NoSCE
Bernard, A, Schmidt	1650114258456	1965-01-09	72, rue Fardère, Les Ulis	5
Thierry, B, Wong	1551275145237	1955-12-08	12 route des Grèves, Les Ulis	5
Jeanne, C Zaoui	2680187802127	1968-01-19	33, chemin des Sources, Lanester	4
Séverine, D, Wattwiller	2410625458923	1941-06-20	12, rue Henri IV, Antibes	4
Robert, E, Nathan	1620923023265	1962-09-15	97, rue Firoque, Les Ulis	5
Albertine, F, Anglais	2720715305897	1972-07-31	56, rue des Trois-Frères, Les Ulis	5
Victor, G, Jabare	1690375230457	1969-03-29	98, rue de la Paix, Les Ulis	4
Etienne, H, Borgue	1371184245901	1937-11-10	45, avenue des Settons, Les Ulis	1

SERVICE

NOMS	NoS	NoSSDIR
Recherche	5	1551275145237
Administration	4	2410625458923
Siège	1	1371184245901

EMP_SERV

NoS	EMPL_S
1	Les Ulis
4	Meyzieu
5	Valbonne
5	Lanester
5	Les Ulis

TRAVAILLE_SUR

NoSS	No_P	HEURES
1650114258456	1	32.5
1650114258456	2	7.5
1620923023265	3	40.0
2720715305897	1	20.0
2720715305897	2	20.0
1551275145237	2	10.0
1551275145237	3	10.0
1551275145237	10	10.0

PROJET

NOM_P	No_P	EMPL_P	NUMS
ProduitX	1	Valbonne	5
ProduitY	2	Lanester	5
ProduitZ	3	Les Ulis	5
Informatisation	10	Meyzieu	4
Réorganisation	20	Les Ulis	1
Innovation	30	Meyzieu	4

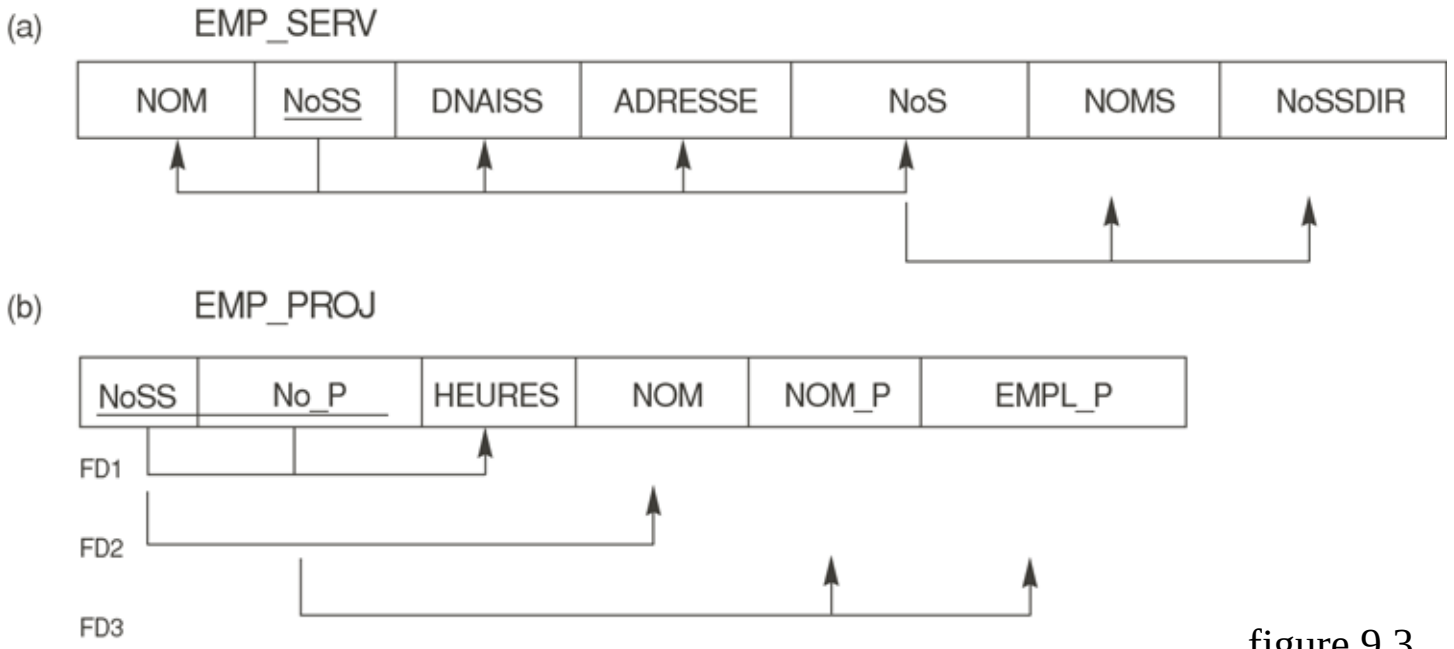


figure 9.3

- Dans le schéma EMP_SERV on ajoute aux attributs de l'entité employé, le nom de son directeur de service ainsi que son # de sécurité sociale.
- Dans EMP_PROJ, les attributs sont le # de sécurité sociale de l'employé, le # de projet, le nombre d'heures travaillé pour ce projet, le nom de l'employé, le nom du projet et l'emplacement du projet.

En résumé :

- s'assurez que chaque relation corresponde à un concept ou un fait du domaine de l'application (voir figure 9.1, 9.2 sur acétates suivantes)
- ne pas mélanger des attributs de deux "entités" différentes dans la même relation (voir figure 9.3 (contre-exemple))
- il faut tenir compte de l'usage des entités
  Exemple : gestion d'un dossier professeur au bureau de la recherche et gestion des membres du centre sportif: doit-on créer une seule relation ou deux relations?

9.1.2 Informations redondantes dans les tuples et anomalies de mise à jour

- Un objectif de la conception de schémas est de réduire l'espace de stockage utilisé par les relations de base (voir figure 9.4 sur l'acétate suivante)
- Un autre problème important que pose l'emploi des relations de la figure 9.4 comme relations de base est celui des **anomalies de mise à jour**.
  Parmi celles-ci, on peut distinguer des anomalies d'insertion, de suppression et de modification.
  - **anomalie d'insertion** : l'ajout d'un entité employé-service impose soit
    * l'ajout de l'information de deux entités à la fois, et on doit s'assurer de la cohérence entre les duplications d'information
    * ajoute l'information avec des valeur nulles si l'employé ne se voit pas attribué immédiatement un service
    * si on ajoute un nouveau service alors il n'y a pas d'employé au départ ce qui signifie que la clé primaire est nulle
  - **anomalie de modification** : si on doit modifier un service alors on doit modifier tous les tuples où l'information est dupliquée
  - **anomalie de suppression** : si on doit supprimer tous les tuples où l'information est dupliquée (par exemple tous les employés d'un service alors on perd l'information sur ce service, on doit donc préserver au moins un tuple.
- performance : on doit parfois avoir de la redondance pour améliorer la performance.

Principe 2

Concevez les schémas des relations de base de telle sorte qu'il ne puisse pas survenir d'anomalies d'insertion, de suppression et de modification dans les relations. S'il y a des anomalies, indiquez-les clairement et assurez-vous que les programmes qui mettent à jour la base de données opèreront correctement.

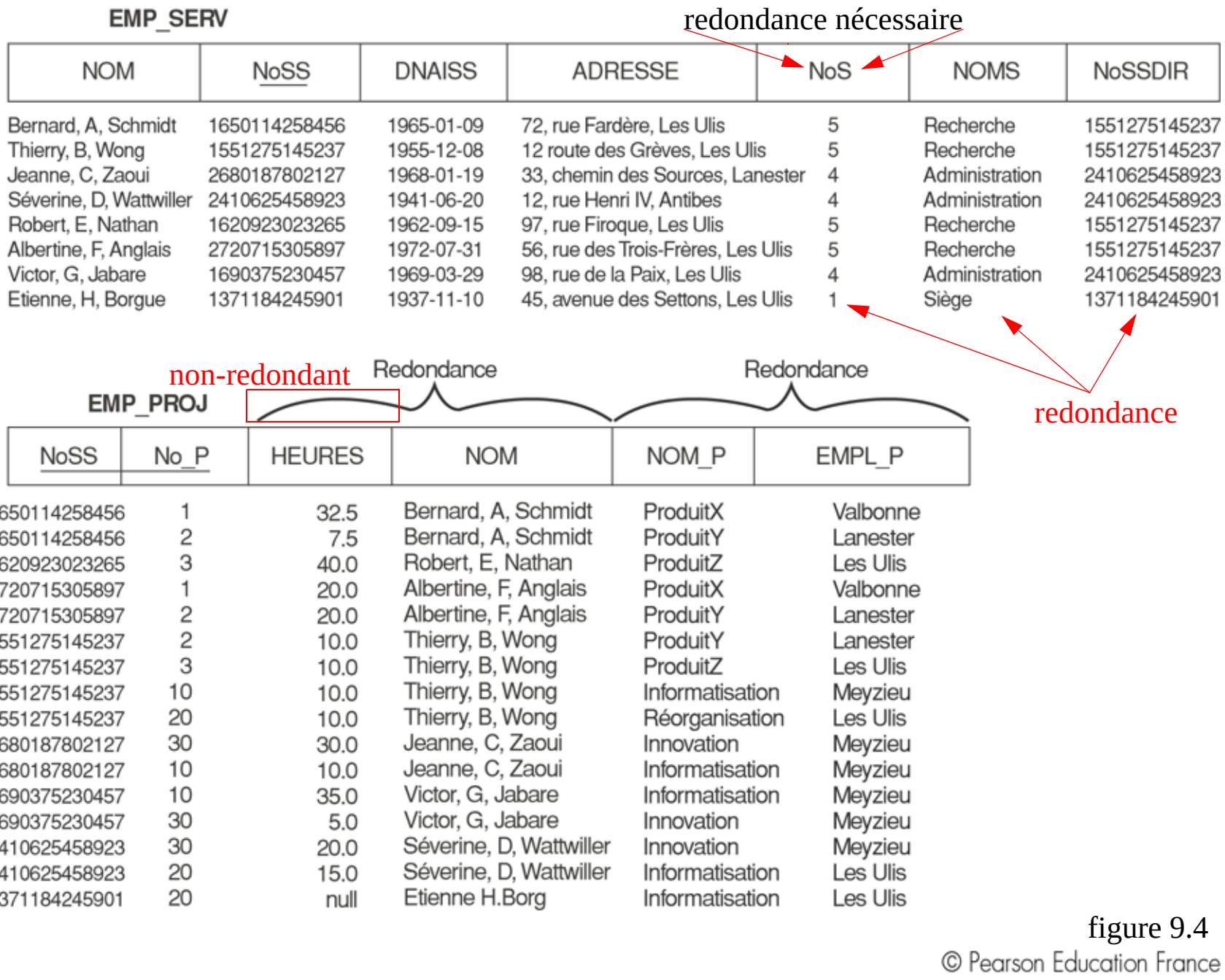


figure 9.4

9.1.3 Valeurs nulles dans les tuples

- éviter les valeurs nulles
- comment les traiter dans une fonction (ex: count, sum, etc)
- elles ont 3 interprétations possibles
  - l'attribut ne s'applique pas à ce tuple (ex: NoSSDir dans relation EMPLOYEE);
  - la valeur de l'attribut est inconnue pour le tuple (elle le sera plus tard);
  - la valeur de l'attribut est connue pour le tuple, mais elle n'est pas encore enregistrée (elle le sera plus tard);
- créer une nouvelle entité regroupant les valeur peu fréquente.
  Par exemple, si 10 % des employés disposent de bureaux individuels, il y a peu de raisons d'inclure un attribut NUMERO_BUREAU dans la relation EMPLOYE. Mieux vaut dans ce cas créer une relation BUREAU_EMPLOYE (NoSS_EMPL, NUMERO_BUREAU) qui inclut des tuples pour les seuls employés disposant d'un bureau individuel.

Principe 3

Dans la mesure du possible, évitez de placer dans une relation de base des attributs dont les valeurs sont susceptibles d'être souvent nulles. Si cela est inévitable, faites en sorte qu'elles n'apparaissent que pour des cas exceptionnels et qu'elles ne concernent pas une majorité de tuples dans la relation.

9.1.4 Génération de tuples parasites

- s'assurer que la jointure de deux relations sur des clés étrangères et des clés primaires ne donnent pas de tuples erronés.
  Considérer les trois tables suivantes .

EMP_SITE

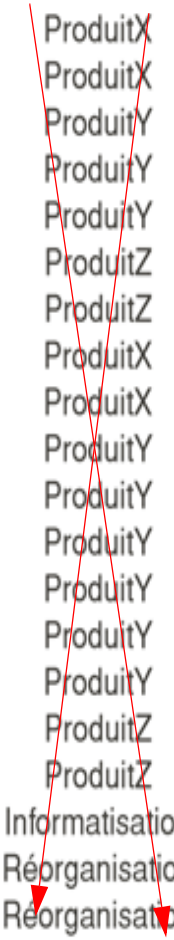
Nom	EmplProjet
Bernard, A, Schmidt	Valbonne
Bernard, A Schmidt	Lanester
Robert, E, Nathan	Les Ulis
Albertine, F, Anglais	Valbonne
Albertine, F, Anglais	Lanester
Thierry B, Wong	Lanester
Thierry, B, Wong	Les Ulis
Thierry, B, Wong	Meyzieu
Jeanne C Zaoui	Meyzieu
Victor, Jabare	Meyzieu
Séverin, D, Waftwiller	Meyzieu
Séverin,Wattwiller	Les Ulis
Etienne H.Borg	Les Ulis

EMP_PROJ1

NoSS	NoProjet	Heures	NomProjet	EmplProjet
1650114258456	1	32.5	ProduitX	Valbonne
1650114258456	2	7.5	ProduitY	Lanester
1620923023285	3	40.0	ProduitZ	Les Ulis
2720715305697	1	20.0	ProduitX	Valbonne
2720715305897	2	20.0	ProduitX	Lanester
1551275145237	2	10.0	ProduitY	Lanester
1551275145237	3	10.0	ProduitZ	Les Ulis
1551275145237	10	10.0	Informatisation	Meyzieu
1551275145237	20	10.0	Réorganisation	Les Ulis
2680187802127	30	30.0	Innovation	Meyzieu
2680187802127	10	10.0	Informatisation	Meyzieu
1690375230457	10	35.0	Informatisation	Meyzieu
1690375230457	30	50	Innovation	Meyzieu
24106254513923	30	20.0	Innovation	Meyzieu
2410625458923	20	15.0	Informatisation	Les Ulis
1371184245901	20	null	Informatisation	Les Ulis

•

NoSS	No_P	HEURES	NOM_P	EMPL_P	NOM	
1650114258456	1	32.5	ProduitX	Valbonne	ProduitX	Bernard, A, Schmidt
1650114258456 *	1	32.5	ProduitX	Valbonne	ProduitX	Albertine, F, Anglais
1650114258456	2	7.5	ProduitY	Lanester	ProduitY	Bernard, A, Schmidt
1650114258456 *	2	7.5	ProduitY	Lanester	ProduitY	Albertine, F, Anglais
1650114258456 *	2	7.5	ProduitY	Lanester	ProduitY	Thierry, B, Wong
1620923023265	3	40.0	ProduitZ	Les Ulis	ProduitZ	Robert, E, Nathan
1620923023265 *	3	40.0	ProduitZ	Les Ulis	ProduitZ	Thierry, B, Wong
2720715305897 *	1	20.0	ProduitX	Valbonne	ProduitX	Bernard, A, Schmidt
2720715305897	1	20.0	ProduitX	Valbonne	ProduitX	Albertine, F, Anglais
2720715305897 *	2	20.0	ProduitY	Lanester	ProduitY	Bernard, A, Schmidt
2720715305897	2	20.0	ProduitY	Lanester	ProduitY	Albertine, F, Anglais
2720715305897 *	2	20.0	ProduitY	Lanester	ProduitY	Thierry, B, Wong
1551275145237 *	2	10.0	ProduitY	Lanester	ProduitY	Bernard, A, Schmidt
1551275145237 *	2	10.0	ProduitY	Lanester	ProduitY	Albertine, F, Anglais
1551275145237	2	10.0	ProduitY	Lanester	ProduitY	Thierry, B, Wong
1551275145237 *	3	10.0	ProduitZ	Les Ulis	ProduitZ	Robert, E, Nathan
1551275145237	3	10.0	ProduitZ	Les Ulis	ProduitZ	Thierry, B, Wong
1551275145237	10	10.0	Informatisation	Meyzieu	Informatisation	Thierry, B, Wong
1551275145237 *	20	10.0	Réorganisation	Les Ulis	Réorganisation	Robert, E, Nathan
1551275145237	20	10.0	Réorganisation	Les Ulis	Réorganisation	Thierry, B, Wong



Principe 4

Veillez, au cours de la définition des schémas relationnels, à ce qu'ils puissent être réunis à l'aide de conditions d'égalité spécifiées sur des attributs jouant le rôle de clés primaires ou de clés étrangères d'une manière qui garantisse l'absence de tuples parasites dans le résultat de la jointure.

Évitez de produire des relations qui ne résultent pas de l'association d'attributs uniques (qui ne sont pas des clés primaires ou étrangères) car les jointures réalisées à partir d'attributs de ce type sont susceptibles de contenir des tuples parasites

9.2 Dépendance fonctionnelle

9.2.1 Définition de la dépendance fonctionnelle

Supposons que la totalité de la base de données peut être décrite par un seul schéma relationnel universel $R = \{A_1, A_2 \ldots, A_n\}$. Nous noterons par Z l'ensemble des attributs i.e. $A_1, A_2 \ldots, A_n$

Soit R(Z) une relation. Il existe une **dépendance fonctionnelle** dans R entre deux ensembles d'attributs $X \subseteq Z$ et $Y \subseteq Z$, notée $X \rightarrow Y$, ssi pour tous tuples t_1, t_2 de tout état r de R, on a

$$t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

- On dit qu'il y a dépendance fonctionnelle de X vers Y ou que *Y* est fonctionnellement dépendant de X.
- L'abréviation de la dépendance fonctionnelle est DF ou d.f.
- L'ensemble des attributs de X est appelé la partie gauche de DF tandis que Y est appelé la partie droite.

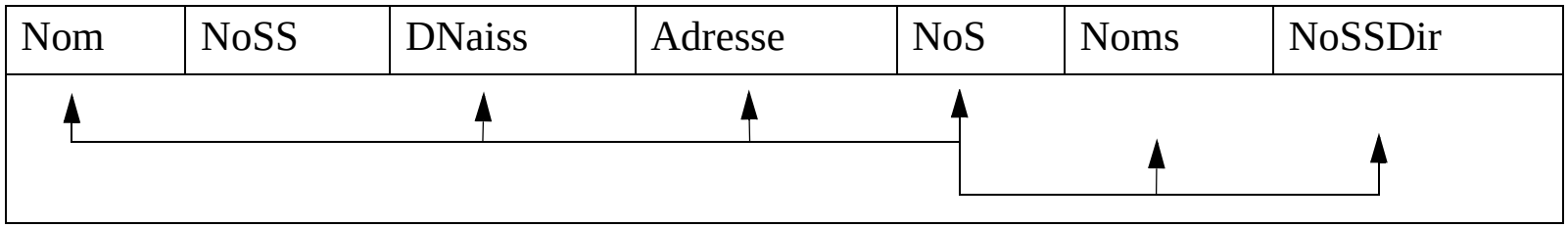
- Si une contrainte sur R établit qu'il ne peut pas y avoir plus d'un tuple avec une valeur X donnée dans n'importe quelle instance r(R)
  (c'est-à-dire si X est une clé candidate de R),
  cela implique que $X \rightarrow Y$ pour n'importe quel sous-ensemble d'attributs Y de R.

- Si $X \rightarrow Y$ dans R, cela ne permet pas de savoir si $Y \rightarrow X$ est vrai ou faux.

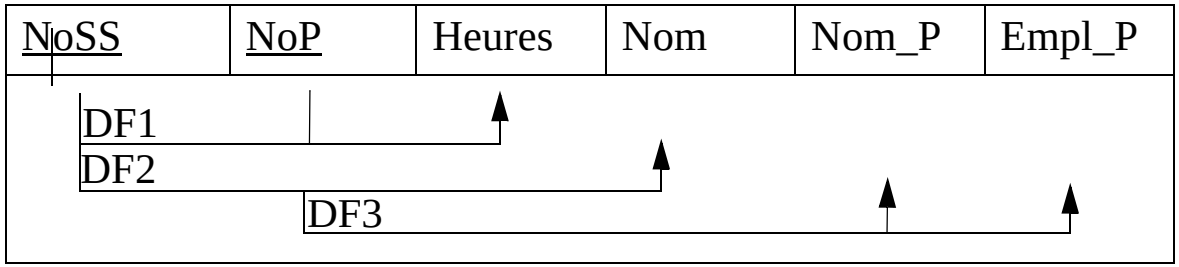
- Une dépendance fonctionnelle est une propriété de la sémantique ou de la signification des attributs.
  - Les concepteurs de la base de données se serviront de la compréhension de la sémantique des attributs de *R* pour spécifier les dépendances fonctionnelles.
  - Chaque fois que la sémantique de deux ensembles d'attributs de *R* indique qu'il faut placer une dépendance fonctionnelle, on spécifie la dépendance sous forme de contrainte.
- Des extensions relationnelles *r(R)* qui satisfont aux contraintes des dépendances fonctionnelles sont appelées états relationnels légaux (ou extensions légales).
  - Un extension relationnelle est un ensemble de tuples

Exemples de dépendances fonctionnelles

EMP_SERV



EMP_PROJ



- Dans le schéma EMP_SERV on ajoute aux attributs de l'entité employé, le nom de son directeur de service ainsi que son # de sécurité sociale.
- Dans EMP_PROJ, les attributs sont le # de sécurité sociale de l'employé, le # de projet, le nombre d'heures travaillé pour ce projet, le nom de l'employé, le nom du projet et l'emplacement du projet.
- On semble constater que les anomalies de mises à jour sont reliés aux dépendances fonctionnelles. Cette intuition sera développée au cours du présent chapitre.

9.2.2 Règles d'inférence pour les dépendances fonctionnelles

on spécifie habituellement les dépendances fonctionnelles évidentes à partir de la description du problème.

on dénote par F l'ensemble de ces dépendances fonctionnelles

on dénote par F^+ la fermeture de F

- F^+ contient toutes les dépendances fonctionnelles que l'on peut déduire à partir de F

on peut calculer F^+ à l'aide des règles suivantes.
Soit *W, X, Y, Z* des ensembles d'attributs.

1. réflexivité : si $X \supseteq Y$, alors $X \rightarrow Y$
2. augmentation : si $X \rightarrow Y$, alors XZ[1] $\rightarrow YZ$
3. transitivité : si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$
4. décomposition : si $X \rightarrow YZ$, alors $X \rightarrow Y$ et $X \rightarrow Z$
5. union : si $X \rightarrow Y$ et $X \rightarrow Z$, alors $X \rightarrow YZ$
6. pseudo-transitivité : si $X \rightarrow Y$ et $WY \rightarrow Z$, alors $WX \rightarrow Z$

Les 3 premières règles sont suffisantes pour calculer F^+ (théorème de Armstrong, 1974).

1.Notation : XZ est une abbréviation de X U Z.

Preuve de RI-1 (réflexivité) si $X \supseteq Y$ alors $X \rightarrow Y$

Soit R une relation
Soit t_1 et t_2 deux tuples appartenant à r un état de R

1. $X \supseteq Y$
2. $t_1[X] = t_2[X] \quad \Rightarrow \quad t_1[Y] = t_2[Y]$ à prouver
    1. $t_1[X] = t_2[X]$
       Ceci signifie que tout attribut appartenant X a la même valeur dans chacun des tuples
    2. Soit A un attribut quelconque de Y,
       alors A est aussi un attribut de X puisque par 1 nous avons que $X \supseteq Y$
       $t_1[A] = t_2[A]$
       comme c'est vrai pour un attribut quelconque, nous généraliser à tous les autres attributs de Y
       $t_1[Y] = t_2[Y]$
3. $X \rightarrow Y$ puisque 2 est la définition de dépendance fonctionnelle.

Preuve de RI-2 (augmentation) : si $X \rightarrow Y$ alors $XZ \rightarrow YZ$
Soit R une relation, et t_1 et t_2 deux tuples appartenant à r un état de R

1. Supposons R tel que $X \rightarrow Y$
2. $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$
3. $t_1[XZ] = t_2[XZ] \Rightarrow t_1[YZ] = t_2[YZ]$ à prouver
    1. $t_1[XZ] = t_2[XZ]$
    2. $XZ \supseteq Z$
    3. $XZ \rightarrow Z$
    4. $t_1[XZ] = t_2[XZ] \Rightarrow t_1[Z] = t_2[Z]$
    5. $t_1[Z] = t_2[Z]$
    6. $XZ \supseteq X$
    7. $XZ \rightarrow X$
    8. $t_1[XZ] = t_2[XZ] \Rightarrow t_1[X] = t_2[X]$
    9. $t_1[X] = t_2[X]$
    10. $t_1[Y] = t_2[Y]$ voir le 2) de la preuve principal
    11. $t_1[YZ] = t_2[YZ]$; Montrons que pour un attribut quelconque C de YZ nous avons $t_1[C] = t_2[C]$
        Comme C est un attribut de YZ alors “C est un attribut de Y” ou “C est un attribut de Z”
        cas A) c’est un attribut de Y, alors $t_1[C] = t_2[C]$,
        cas B) c’est un attribut de Z, alors $t_1[C] = t_2[C]$;
4. $XZ \rightarrow YZ$

Preuve de RI-3 (transitivité) : si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$
Soit R une relation, et t_1 et t_2 deux tuples appartenant à r un état de R

1. Supposons R tel que $X \rightarrow Y$ et $Y \rightarrow Z$
2. $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$
3. $t_1[Y] = t_2[Y] \Rightarrow t_1[Z] = t_2[Z]$
4. $t_1[X] = t_2[X] \Rightarrow t_1[Z] = t_2[Z]$ à prouver
    1. $t_1[X] = t_2[X]$
    2. $t_1[Y] = t_2[Y]$
    3. $t_1[Z] = t_2[Z]$
5. $X \rightarrow Z$

Preuve de RI4 (décomposition) : si $X \rightarrow YZ$, alors $X \rightarrow Y$ et $X \rightarrow Z$
Soit R une relation, et t_1 et t_2 deux tuples appartenant à r un état de R

1. Supposons R tel que $X \rightarrow YZ$
2. $YZ \supseteq Y$
3. $YZ \rightarrow Y$ RI-1
4. $X \rightarrow Y$ RI-3
5. $YZ \supseteq Z$
6. $YZ \rightarrow Z$ RI-1
7. $X \rightarrow Z$ RI-3
8. $X \rightarrow Y$ et $X \rightarrow Z$

Preuve de RI5 (union) : si $X \rightarrow Y$ et $X \rightarrow Z$, alors $X \rightarrow YZ$
Soit R une relation, et t_1 et t_2 deux tuples appartenant à r un état de R

1. Supposons R tel que $X \rightarrow Y$ et $X \rightarrow Z$
2. $XX \rightarrow XY$ RI2
3. $X \rightarrow XY$
4. $XY \rightarrow ZY$ RI2
5. $X \rightarrow ZY$ RI3
6. $X \rightarrow YZ$

Preuve de RI6 (pseudo-transitivité) : si $X \rightarrow Y$ et $WY \rightarrow Z$, alors $WX \rightarrow Z$
Soit R une relation, et t_1 et t_2 deux tuples appartenant à r un état de R

1. Supposons R tel que $X \rightarrow Y$ et $WY \rightarrow Z$
2. $WX \rightarrow WY$ RI2
3. $WX \rightarrow Z$ RI3

- on dénote par $\mathbf{X}^+$ la **fermeture de X sous F**, c'est-à-dire le plus grand ensemble d'attributs *Z* tel que $X \rightarrow Z$, à partir d'inférences sur *F*.

$$X^+ = \{A \mid F \models (X \rightarrow A)\}$$

- on peut calculer X^+ comme suit :

```
X+ := X
répéter
  oldX+ := X+
  pour chaque Y → Z de F faire
    si Y ⊆ X+ alors X+ := X+Z
jusqu'à oldX+ = X+
```

Exemple : Soit

```
F={ A1 → A2,
    A3 → {A4, A5},
    {A1, A3} → A6
  }
```

Alors

$A_1^+ = \{A_1, A_2\}$

$A_3^+ = \{A_3, A_4, A_5\},$

$\{A_1, A_3\}^+ = \{A_1, A_2, A_3, A_4, A_5, A_6\}$

- à l'aide de cet algorithme de fermeture, on peut calculer si deux ensembles E et F de dépendances fonctionnelles sont équivalents (prochaine section).

9.2.3 Équivalence d'ensembles de dépendances fonctionnelles

Soit *E*, *F* des ensembles de dépendances fonctionnelles.

On dit que *F* comprend *E* si chaque DF de *E* est aussi dans F^+ ; c'est-à-dire
si chaque dépendance de E peut être inférée de F ;
ou encore (d'une façon pratique)

si pour toute dépendance $X \rightarrow Y$ de *E*, on a $Y \subseteq X^+$ sous *F*.

Exemple : 9.3 Soit

E={ A1 → A3, A2 → A4, A3 → A4, A4 → A3 }	F={ A1 → A4, A2 → A3, A3 → A4, A4 → {A3,A5} }

Si on calcule $A_1^+, A_2^+, A_3^+, A_4^+$ sous F, on obtient

$A_1^+ = \{A_1, A_3, A_4, A_5\}$

$A_2^+ = \{A_2, A_3, A_4, A_5\}$

$A_3^+ = \{A_3, A_4, A_5\}$

$A_4^+ = \{A_3, A_4, A_5\}$

On observe que *F* comprend *E* puisque $\{A_3\} \subseteq \{A_1, A_3, A_4, A_5\}$; $\{A_4\} \subseteq \{A_2, A_3, A_4, A_5\}$; $\{A_4\} \subseteq \{A_3, A_4, A_5\}$ et $\{A_4\} \subseteq \{A_3, A_4, A_5\}$

Cependant,

E={ A1 → A3, A2 → A4, A3 → A4, A4 → A3 }	F={ A1 → A4, A2 → A3, A3 → A4, A4 → {A3,A5} }

si on calcule A_1^+,A_2^+,A_3^+,A_4^+ sous *E*, on obtient

$A_1^+ = \{A_1, A_3, A_4\}$

$A_2^+ = \{A_2, A_3, A_4\}$

$A_3^+ = \{A_3, A_4\}$

$A_4^+ = \{A_3, A_4\}\}$

On observe que *E* ne comprend pas *F*, car pour $A_4 \rightarrow \{A_3, A_5\}$, on a

$$A_4^+ = \{A_3, A_5\} \not\subset \{A_3, A_4\}$$

Soit *E*, *F* des ensembles de dépendances fonctionnelles. On dit que *E* et *F* **sont équivalents** ssi *E* comprend *F* et *F* comprend *E*.

Exemple

On observe que *E* et *F* de l'exemple précédents ne sont pas équivalents, car *F* couvre *E*, mais *E* ne couvre pas *F*. Soit

E={ A1 → A3, A2 → A4, A3 → A4, A4 → A3 }	F'= { A_1 → A_4, A_2 → A_3, A_3 → A_4, A_4 → A_3 }

Si on calcule A_1^+,A_2^+,A_3^+,A_4^+

sous *E*, on obtient	sous F', on obtient
$A_1^+ = \{A_1, A_3, A_4\}$	$A_1^+ = \{A_1, A_3, A_4\}$
$A_2^+ = \{A_2, A_3, A_4\}$	$A_2^+ = \{A_2, A_3, A_4\}$
$A_3^+ = \{A_3, A_4\}$	$A_3^+ = \{A_3, A_4\}$
$A_4^+ = \{A_3, A_4\}\}$	$A_4^+ = \{A_3, A_4\}$

Donc, *E* et *F*' sont équivalents.

9.2.4 Ensembles minimals de dépendances fonctionnelles.

Une **dépendance fonctionnelle X $\rightarrow$ Z est minimale** ssi il n'existe pas de dépendance fonctionnelle Y $\rightarrow$ Z telle que Y $\subset$ X.

Un **ensemble de dépendances fonctionnelles *F* est minimal** ssi

1. pour chaque X $\rightarrow$ A $\in$ *F*, A est un singleton ;
2. on ne peut enlever une dépendance de *F* et obtenir un ensemble de dépendances fonctionnelles équivalent ;
3. on ne peut remplacer une X $\rightarrow$ A $\in$ F par Y $\rightarrow$ A $\in$ *F*, avec Y $\subset$ X.

• Chaque ensemble de DF possède un ensemble minimal équivalent.
• Il peut y avoir plusieurs ensembles minimals équivalents

• Algorithme pour calculer *E* un ensemble minimal équivalent à *F*, un ensemble de DF

Posons F := E
Remplaçons chaque X $\rightarrow$ {A_1, A_2… A_n} par X $\rightarrow$ A_1 ; X $\rightarrow$ A_2 … X $\rightarrow$ A_n

Pour chaque X $\rightarrow$ A dans F, pour chaque attribut B dans X
si {{F - {X $\rightarrow$ A}} U {(X-{B})$\rightarrow$ A}} est équivalent à F
alors remplaçons X $\rightarrow$ A par (X-{B})$\rightarrow$ A

Pour chaque X $\rightarrow$ A dans F,
si {F - {X $\rightarrow$ A}} est équivalent à F alors éliminons X $\rightarrow$ A de F.

9.3 Formes normales basés sur des clés primaires

9.3.1 Normalisation des relations

normalisation des données := processus d'analyse des schémas relationnels sur la base de leurs DF et de leurs clés primaires dans le but de parvenir aux propriétés suivantes :
-la réduction de la redondance ;
- la réduction des anomalies d'insertion, de suppression et de mise à jour .

Les schémas relationnels qui ne satisfont pas certaines conditions (c'est-à-dire les tests des formes normales) sont décomposés en des schémas relationnels plus petits qui satisfont aux tests et, par suite, possèdent les propriétés recherchées.
C'est ainsi que la procédure de normalisation fournit aux concepteurs de bases de données:

- un cadre formel d'analyse des schémas relationnels en fonction de leurs clés et des dépendances fonctionnelles au sein de leurs attributs ;
- une série de tests basés sur des formes normales qui peuvent être réalisés sur des schémas relationnels individuels de manière à ce que la base de données puisse être normalisée au degré souhaité.

La forme normale d'une relation est liée à la condition de forme normale la plus élevée qu'elle satisfait et indique le degré auquel celle-ci a été normalisée.

Lorsqu'elles sont considérées *isolément* d'autres facteurs, les formes normales ne garantissent pas la qualité de la conception d'une base de données. Généralement, il ne suffit pas de contrôler séparément que chaque schéma relationnel de la base de données est, par exemple, en FNBC ou 3FN. Il faut aussi que le processus de normalisation, par le biais de la décomposition, confirme l'existence de propriétés supplémentaires dans les schémas relationnels considérés dans leur ensemble.
Parmi ces propriétés figurent les deux suivantes:

- la propriété de non perte de jointure (non-addittion de tuple), qui garantit que des tuples parasites ne seront pas générés (voir la section 9.1.4) à partir des schémas relationnels créés après la décomposition ;
- la propriété de préservation des dépendances, qui assure que chaque dépendance fonctionnelle est représentée dans les relations individuelles produites par la décomposition.
  - La propriété de non perte de jointure est extrêmement importante et doit être obtenue à tout prix tandis que la propriété de préservation des dépendances, bien que souhaitable, peut parfois être sacrifiée.

Les formes normales représentent des contraintes sur des schémas de relation. Elles sont ordonnées comme suit. Soit rel(FN) l'ensemble des relations satisfaisant la forme normale FN. On a

$$\text{rel(1NF)} \supseteq \text{rel(2NF)} \supseteq \text{rel(3NF)} \supseteq \text{rel(BCNF)}$$

Donc, si une relation est en forme BCNF, alors elle est aussi en forme 3NF, 2NF et 1NF.

En résumé :

- La normalisation est un processus qui consiste à transformer des schémas de relation afin qu'ils satisfassent les formes normales.
- La normalisation permet:
  - d'éviter les anomalies de mise à jour
  - de réduire la redondance

9.3.2 Usage pratique des formes normales

- Le plus souvent, les projets de conception sont élaborés à partir de conceptions de bases de données déjà en place, de conceptions de modèles hérités ou de fichiers existant.
- Dans la pratique, la conception est entreprise de manière à ce que les conceptions auxquelles on aboutit soient de haute qualité et satisfassent les propriétés précédemment évoquées.
- Bien que plusieurs formes normales supérieures aient été définies (la 4FN et la 5FN), l'utilité pratique de celles-ci devient douteuse lorsque les contraintes sur lesquelles elles reposent sont difficiles à comprendre ou à détecter par le concepteur et les utilisateurs auxquels incombe la charge de les rechercher.
- Il est possible de laisser des relations dans un état de normalisation inférieure (la 2FN, par exemple) pour des raisons de performances telles que celles évoquées à la section 9.1.2.
  - Le processus de stockage de la jointure de relations normalisées à un degré plus élevé sous forme d'une relation de base (laquelle est dans une forme normale inférieure) est appelé **dénormalisation**.

9.3.3 Définitions des clés et des attributs des clés

Une **superclé** d'un schéma relationnel $R = \{A_1, A_2, ..., A_n\}$ est un ensemble d'attributs $S \subseteq R$ caractérisés par la propriété suivante: dans aucun état relationnel r de R, deux tuples t_1 et t_2 ne seront pas tels que $t_1[S] = t_2[S]$.

ou encore, d'une façon équivalente

Un ensemble d'attributs X est une **superclé** d'une relation R(X,Y) ssi il existe une dépendance fonctionnelle $X \rightarrow Y$.

Un ensemble d'attributs X est une clé **candidate** d'une relation R(X,Y) ssi il existe une dépendance fonctionnelle minimale $X \rightarrow Y$.

Une **clé** K est une superclé dotée de la propriété supplémentaire suivante: la suppression d'un de ses attributs annule son statut de superclé (clé et clé candidate veulent donc dire la même chose).

- Une clé candidate est donc une superclé mais les superclés ne sont pas tous des clés candidates.
- Une super clé minimale est une clé candidate.
- Lors de la définition des contraintes d'intégrité dans une table, on choisit une des clé candidates comme clé primaire;
- Les autres clés candidates sont représentées par des clés uniques.

- attribut primaire : attribut qui appartient à une clé candidate
- attribut non primaire : attribut qui n'appartient pas à une clé candidate

9.3.4 Première forme normale (1NF)

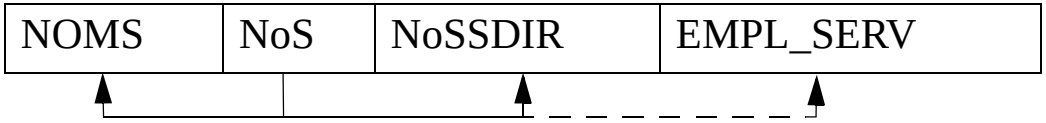
Une **relation R est en première forme normale** (1NF) ssi tous les attributs de R sont atomiques (pas d'ensemble, de tuple ou autre structure vectorielle comme type d'un attribut)

- Un schéma de relation est en première forme normale.

Exemple

- Comparaison avec de l'entité SERVICE de la figure 9.1.
  Sa clé primaire est NoS.
  On l'étend en y incluant l'attribut EMPL_SERV (figure 9.8a).
  +Supposons que chaque service puisse avoir plusieurs emplacements.
- À cause du premier tuple de la figure 9.8b
  nous voyons qu'il n'est pas conforme à INF car EMPL_SERV n'est pas un attribut atomique.

SERVICE (figure 9..8 a)



SERVICE (figure 9..8 b)

Noms	NoS	**NoSSDIR**	EMPL_SERV
Recherche	5	1551275145237	{Valbonne, Lanester, Les Ulis}
Administration	4	2410625458923	{Meyzieu}
Siège	1	1371184245901	{Les Ulis}

Les solutions

- On supprime l'attribut EMPL_SERV qui viole INF et on le place dans une relation séparée EMP_SERV avec la clé primaire NoS de SERVICE. La clé primaire de cette combinaison est {NoS, EMPL_S} comme le montre la figure suivante. Il y a un tuple distinct EMPL_SERV pour chaque emplacement d'un service. Ce procédé décompose la relation non IFN en deux relations IFN.

SERV

Noms	NoS
Recherche	5
Administration	4
Siège	1

EMP _SERV

NoS	EMPL_S
5	Valbonne
5	Lanester
5	Les Ulis
4	Meyzieu
1	Les Ulis

• On étend la clé de manière qu'elle soit un tuple distinct dans la relation SERVICE d'origine pour chaque emplacement d'un SERVICE (voir figure 9.8c). Dans le cas présent, la clé primaire devient la combinaison {NoS, EMPL_S}. Cette solution a l'inconvénient d'introduire de la redondance dans la relation.

SERVICE (figure 9..8 c)

Noms	NoS	NoSSDIR	EMPL_S
Recherche	5	1551275145237	Valbonne
Recherche	5	1551275145237	Lanester
Recherche	5	1551275145237	Les Ulis
Administration	4	2410625458923	Meyzieu
Siège	1	1371184245901	Les Ulis

- Si on sait qu'il existe un nombre maximal de valeurs pour l'attribut (par exemple, si un service ne peut pas avoir plus de trois emplacements), on peut remplacer l'attribut EMPL_S par trois attributs atomiques: EMPL_S1, EMPL_S2 et EMPL_S3. Cette solution a pour inconvénient d'introduire des valeurs nulles si la plupart des services ont moins de trois emplacements. Elle entraîne aussi une sémantique parasite pour l'ordonnancement des valeurs des emplacements qui n'était pas prévu à l'origine. Il devient également plus difficile d'écrire des requêtes relatives à cet attribut (par exemple, comment lister tous les services qui comptent «Valbonne» parmi leurs emplacements n.

SERVICE

Noms	NoS	**NoSSDIR**	EMPL_S1	EMPL_S2	EMPL_S3
Recherche	5	1551275145237	Valbonne	Lanester	Les Ulis
Administration	4	2410625458923	Meyzieu	null	null
Siège	1	1371184245901	Les Ulis	null	null

Des trois solutions précédentes,

- La première est généralement la meilleure parce qu'elle n'entraîne pas de redondance.
- De plus, elle est complètement générale puisqu'elle ne limite pas le nombre des valeurs possibles.
- En fait, si vous choisissez la seconde solution, celle-ci sera décomposée un peu plus lors des étapes suivantes de la normalisation pour aboutir en définitive à la première solution.

- La première forme normale interdit aussi les attributs multivalués qui sont euxmêmes composites. On les appelle des **relations imbriquées** parce que chaque tuple peut avoir une relation à *l'intérieur de lui-même.*

Exemple (voir figure 9.9 de votre manuel)

EMP_PROJ

NOSS	NOM	PROJS	
		NO_P	HEURES

- Chaque tuple représente une entité d'employé et une relation PROJS (No_P, HEURES) à l'intérieur de chaque tuple représente les projets de l'employé et les heures hebdomadaires qu'il consacre à chaque projet.
- Le schéma de cette relation EMP _PROJ peut être représenté de la manière suivante:
  EMP_PROJ(NoSS, NOM, **{**PROJS(No_P, HEURES)**}**)
  - Les accolades { } indiquent que l'attribut est multivalué et les attributs des composants formant PROJS sont listés entre parenthèses ( ).
  - Notez que NoSS est la clé primaire de la relation EMP _PROJ, tandis que No_P est la clé partielle de la relation imbriquée.

- Lors de la prise en charge d'objets complexes et de données XML , on utilise le modèle relationnel pour autoriser et formaliser des relations imbriquées à l'intérieur de systèmes de bases de données relationnels.

Solution

EMP_PROJ

NOSS	NOM	PROJS	
		NO_P	HEURES

EMP_PROJ1

NOSS	NOM

EMP_PROJ2

NOSS	NO_P	HEURES

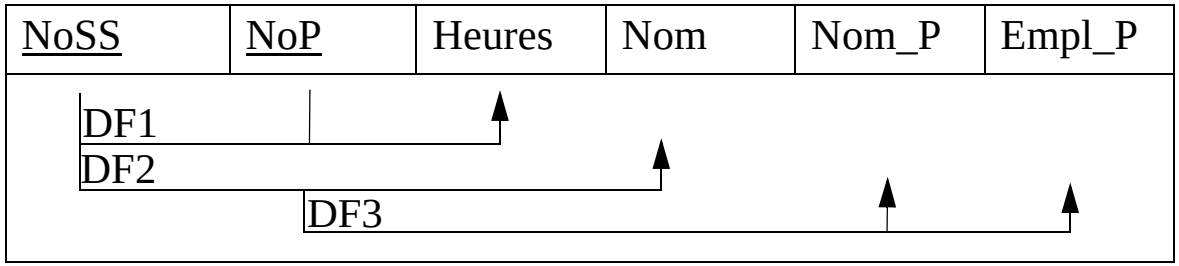
- Pour normaliser cette relation de manière conforme à IFN, on supprime les attributs de la relation imbriquée pour en faire une nouvelle relation et *propager la clé primaire* à l'intérieur de cette dernière. La clé primaire de la nouvelle relation combinera la clé partielle et la clé primaire de la relation de départ. De la décomposition et de la propagation de la clé primaire résultent les schémas EMP _PROJ1 et EMP _PROJ2 de la figure 9.9c.
- On peut appliquer de manière récursive cette procédure à une relation qui contient une imbrication à plusieurs niveaux afin de la « désimbriquer » et obtenir un ensemble de relations IFN. Cette procédure est utile pour convertir en relations IFN un schéma relationnel non normalisé et comportant plusieurs niveaux d'imbrication.

9.3.5 Deuxième forme normale (2NF)

Définition 9.9 Une dépendance fonctionnelle $X \rightarrow Y$ est complète
ssi pour tout $A \in X$, $\neg(X-\{A\} \rightarrow Y)$

- Une dépendance fonctionnelle $X \rightarrow Y$ est une dépendance fonctionnelle complète si la suppression d'un des attributs A de X annule la dépendance fonctionnelle.
  Une dépendance fonctionnelle complète et une dépendance fonctionnelle minimale sont en réalité la même notion.

EMP_PROJ



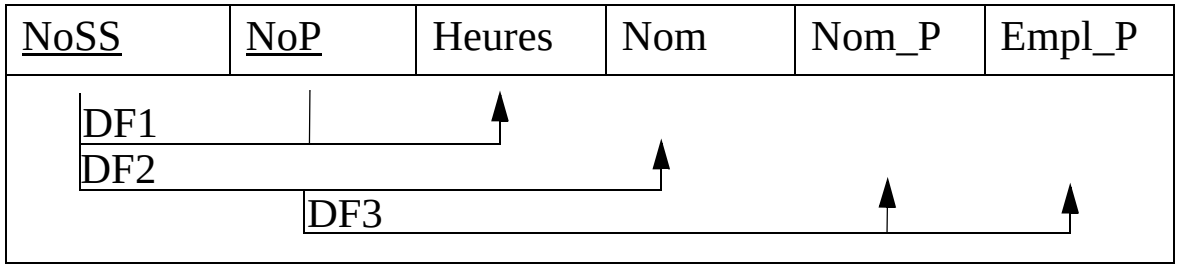
- Dans la figure {NoSS, No_P} → HEURES est une dépendance fonctionnelle complète puisque
  - NoSS → HEURES n'est pas vraies et
    No_P → HEURES n'est pas vraies.

Mais la dépendance {NoSS, No_P} → NOM n'est que partielle car NoSS → NOM est vraie.

Un **schéma relationnel R est en 2FN** si chaque attribut non primaire *A* de *R* est *complètement dépendant fonctionnellement* de la clé primaire de *R.*

- Le test de la conformité à 2FN implique le test des dépendances fonctionnelles dont les attributs de la partie gauche font partie de la clé primaire.
  - Si la clé primaire ne contient qu'un attribut, il est inutile d'appliquer le test. .

EMP_PROJ



- La relation EMP _PROJ de la figure 9.3b est en IFN mais pas en 2FN.
  L'attribut non primaire NOM viole 2FN en raison de DF2, de même que les attributs non primaires NOM_P et EMP _P à cause de DF3.
- Si un schéma relationnel n'est pas en 2FN, il peut être « normalisé en 2FN » grâce à sa décomposition en plusieurs relations 2FN.

EMP_PROJ1

NoSS	NoP	Heures

-

EMP_PROJ2

NoSS	Nom

EMP_PROJ3

NoP	Nom_P	Empl_P

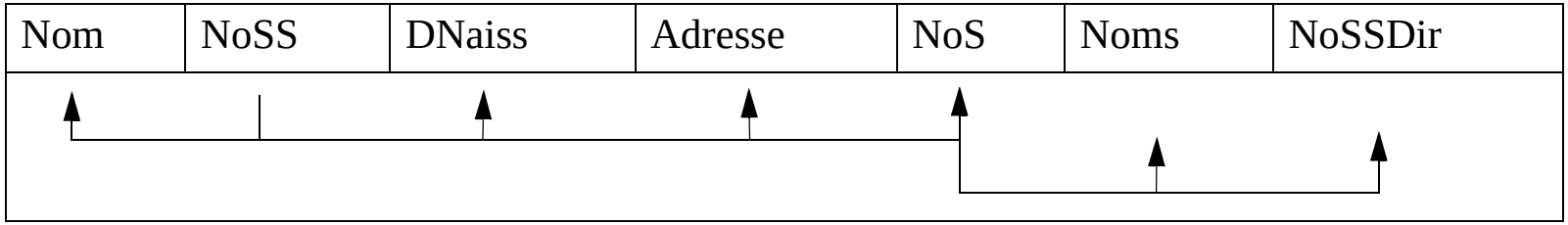
9.3.6 Troisième forme normale (3NF)

Une relation R est en troisième forme normale (3NF) ssi pour toute dépendance fonctionnelle non-triviale X $\rightarrow$ A de R, une des conditions suivantes est satisfaite:

1. X est une super clé
2. A est un attribut premier

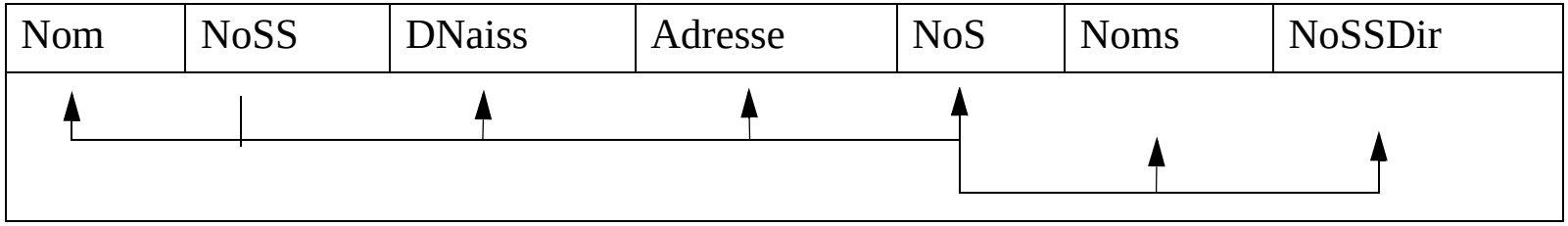
La troisième forme normale (3NF) repose sur le concept de *dépendance transitive*. Une dépendance fonctionnelle X $\rightarrow$Y d'un schéma relationnel *R* est une dépendance transitive s'il y a un ensemble d'attributs Z qui n'est ni une clé candidate ni un sous-ensemble d'une des clés de *R*, et que X $\rightarrow$ Z et Z $\rightarrow$ Y sont toutes les deux vraies.

EMP_SERV



• La dépendance NoSS $\rightarrow$ NoSSDIR est transitive *via* NoS dans EMP _SERV parce que
- NoSS $\rightarrow$ NoS est vrai
NoS $\rightarrow$ NoSSDIR est vrai et que
NoS n'est ni une clé ni un sous-ensemble de la clé de EMP _SERV.

EMP_SERV



- Le schéma relationnel EMP _SERV est en 2FN puisqu'il n'y a pas de dépendances partielles vis-à-vis d'une clé.
- EMP _SERV n'est pas en 3FN en raison de la dépendance transitive de NoSSDIR et nom
  NoSS → NoS est vrai
  NoS → {NoSSDIR, NOMS}
- On peut normaliser EMP _SERV en la décomposant en deux schémas relationnels 3FN ED1 et ED2

ED1

Nom	<u>NoSS</u>	DNaiss	Adresse	NoS

ED2

<u>NoS</u>	Noms	NoSSDir

- (voir figure 9.10b). Intuitivement, on se rend compte que ED1 et ED2 représentent des entités factuelles indépendantes par rapport aux employés et aux services. Une opération JOINTURE NATURELLE sur ED1 et ED2 permet de retrouver la relation d'origine EMP - SERV sans générer de tuples parasites.

Résumé

Formes normales basées sur des clés primaires et normalisations correspondantes.

Forme normale	Test	Remède (normalisation)
Première (1 FN)	La relation ne doit pas comporter d'attributs non atomiques ou de relations imbriquées.	Former de nouvelles relations pour chaque attribut non atomique ou relation imbriquée.
Deuxième (2FN)	Pour les relations dans lesquelles la clé primaire contient plusieurs attributs, il ne doit pas y avoir d'attribut non-clé fonctionnellement dépendant d'une partie de la clé primaire.	Décomposer et construire une nouvelle relation pour chaque clé partielle et son (ses) attribut(s) dépendant(s). Il faut s'assurer de préserver une relation avec la clé primaire d'origine et tous les attributs qui dépendent fonctionnellement d'elle.
Troisième (3FN)	La relation ne doit pas avoir d'attribut non-clé fonctionnellement déterminé par un autre attribut non-clé (ou par un ensemble de plusieurs attributs non-clés). Autrement dit, il ne doit pas y avoir de dépendance transitive d'un attribut non-clé vis-à-vis de la clé primaire.	Décomposer et construire une relation qui inclut le ou les attributs non-clés qui déterminent fonctionnellement le ou les autres attributs non-clés.

9.4 Définitions générales des deuxième et troisième formes normales

- Les étapes de la normalisation en relations 3FN examinées jusqu'ici interdisent des dépendances partielles et transitives sur la clé primaire.
- Mais ces définitions ne tiennent pas compte des autres clés candidates éventuelles d'une relation.
- Dans cette section, nous donnons les définitions plus générales de 2FN et de 3FN qui prennent en compte toutes les clés candidates.

De manière plus générale, on considère comme un **attribut primaire** tout attribut qui fait partie d'une des clés candidates.

De leur côté, les *dépendances fonctionnelles partielles et complètes* seront désormais envisagées en tenant compte de toutes les clés candidates d'une relation.

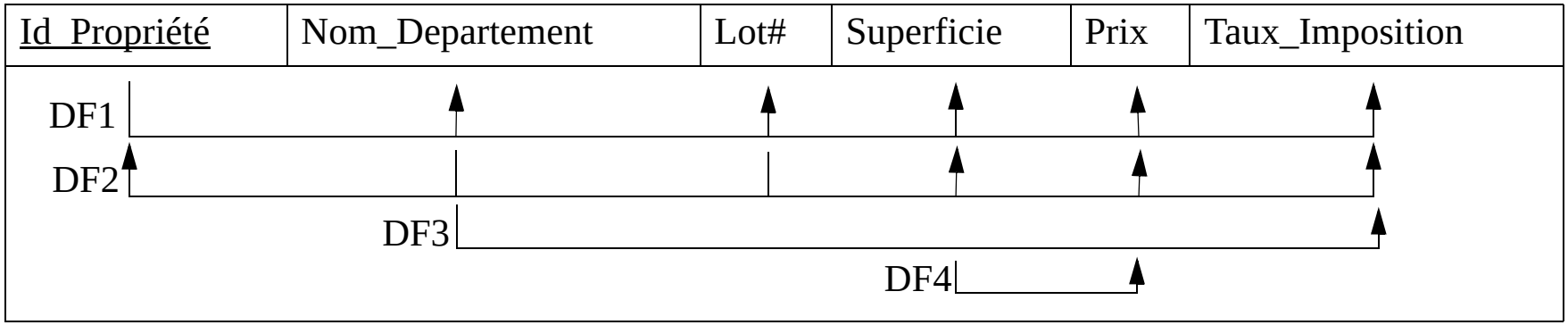
9.4.1 Définition générale de la deuxième forme normale

Un **schéma relationnel R est en deuxième forme normale (2FN)** si chacun des attributs non primaires *A* de *R* n'est pas partiellement dépendant *d'une des* clés de *R.*

Un **schéma relationnel R est en deuxième forme normale (2FN)** si chaque attribut non primaire *A* de *R* est fonctionnellement complètement dépendant de chacune des clés de *R*.

- Le test de la conformité à 2FN implique le test des dépendances fonctionnelles dont les attributs de la partie gauche font partie d'une clé de R.
  - Si la clé ne contient qu'un attribut, il est inutile d'appliquer le test.

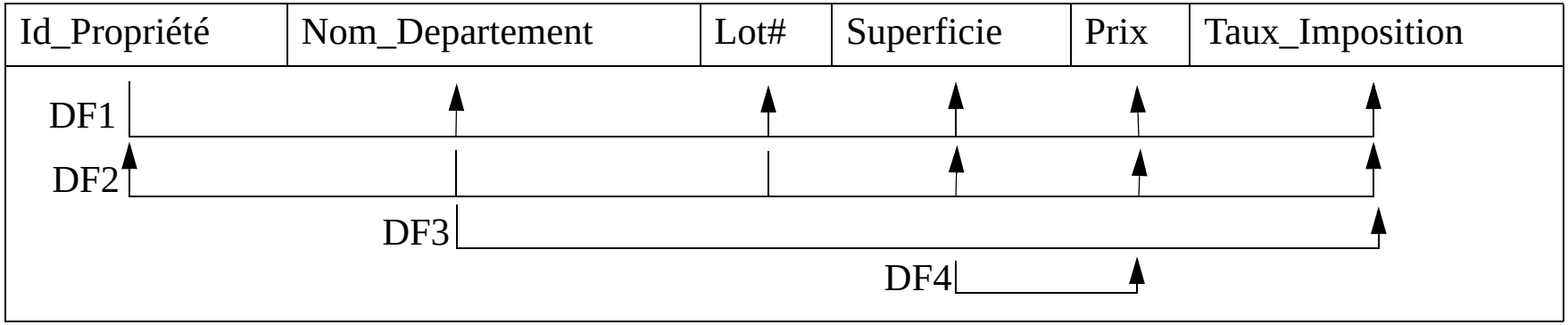
LOTS



- il y a deux clés candidates: ID_PROPRIETE et {NOM_DEPARTEMENT, LOT#}.
- ID_PROPRIETE comme clé primaire.
- Supposons que les deux dépendances fonctionnelles suivantes soient vraies dans LOTS:
  - DF3: Nom_Departement → Taux_Imposition
  - DF4: Superficie → Prix
- Le schéma relationnel LOTS viole la définition générale de 2FN parce que TAUX_IMPOSITION est partiellement dépendant de la clé candidate {NOM_DEPARTEMENT, LOT#} en raison de DF3.

- Pour normaliser LOTS en 2FN, on le décompose en deux relations LOTS1 et LOTS2

LOTS

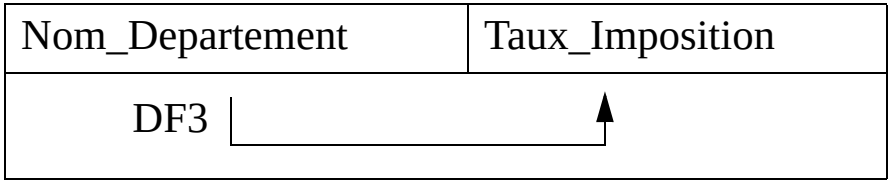


-

LOTS1

-

LOTS2



9.4.2 Définition générale de la troisième forme normale

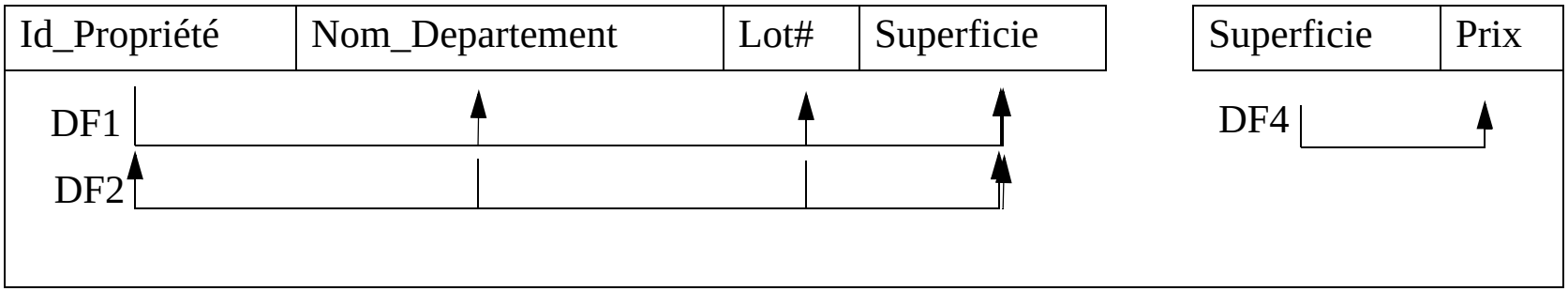
Une relation R est en troisième forme normale (3NF) ssi pour toute dépendance fonctionnelle non-triviale X $\rightarrow$ A de R, une des conditions suivantes est satisfaite:

1. X est une superclé
2. A est un attribut primaire de R

LOTS1

- Selon cette définition, DF4 de LOTS1 viole 3FN car
  - SUPERFICIE n'est pas une superclé et
  - PRIX n'est pas un attribut primaire dans LOTS1.
- Pour normaliser LOTSl en 3FN, on la décompose en schémas relationnels LOTS lA et LOTSlB

LOTS1A **LOTS1B**



•

Il convient de relever deux points à propos de cet exemple et de la définition générale de la 3 FN :

- LOTS1 viole 3FN parce que PRIX dépend transitivement de chacune des clés candidates de LOTS1 via l'attribut non primaire SUPERFICIE.
- Cette définition générale peut être directement appliquée pour tester la conformité à 3FN d'un schéma relationnel. Il n'est pas nécessaire de passer au préalable par la normalisation 2FN. Si on applique la définition de 3FN donnée ci-dessus à LOTS et à ses dépendances DFl à DF4, on s'aperçoit qu'aussi bien DF3 que DF4 violent 3FN. On peut en conséquence directement décomposer LOTS en LOTS1A, LOTS1 B et LOTS2. Il en résulte que les dépendances partielles et transitives qui contreviennent à 3FN peuvent être supprimées dans n'importe quel ordre.

9.4.3 Interprétation de la définition générale de la troisième forme normale

- Un schéma relationnel R contrevient à la définition générale de 3FN
  - si une dépendance fonctionnelle X $\rightarrow$ A vraie dans R ne respecte pas les deux conditions 1 et 2 de 3FN.
- La violation de la condition 2 signifie que A n'est pas un attribut primaire.
- La violation de la condition 1 signifie que X n'est pas un super-ensemble d'une des clés de R.
- Il en résulte que X peut être non primaire ou un sous-ensemble d'une clé de R.
- Si X est non primaire, on a habituellement affaire à une dépendance transitive qui contrevient à 3FN,
- Tandis que si X appartient à un sous-ensemble d'une clé de R,
  on a affaire à une dépendance partielle qui viole 3FN (et aussi 2FN).

En conséquence, on peut énoncer une autre définition générale de 3FN :

Un schéma relationnel R est en 3FN
si chacun des attributs non primaires de R satisfait les 2 conditions suivantes:

1. Il est fonctionnellement dépendant de chacune des clés de R
2. Il est non transitivement dépendant de chacune des clés de R.

9.5 Forme normale de Boyce-Codd

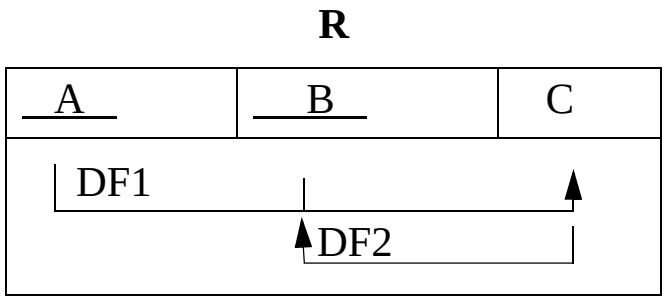
Rappel :
Une relation R est en troisième forme normale (3NF) ssi pour toute dépendance fonctionnelle non-triviale $X \rightarrow A$ de R, une des conditions suivantes est satisfaite:

1. X est une superclé
2. A est un attribut primaire de R

Définition de forme normale de Boyce-Codd
Une relation R est en forme normale de Boyce-Codd (FNBC) ssi pour toute dépendance fonctionnelle non-triviale $X \rightarrow A$ de R, la condition suivante est satisfaite:

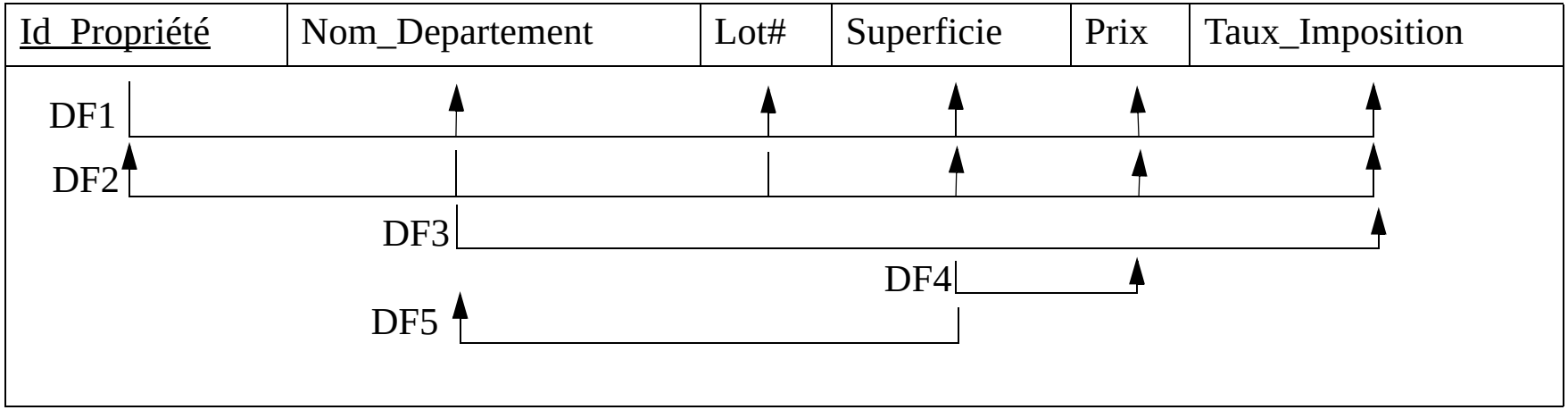
1. X est une superclé

- La FNBCest une forme normale plus stricte que la 3FN puisqu'elle ne prend que la condition 1
- Autrement dit, toute relation en FNBC est aussi en 3FN, mais une relation en 3FN n'est pas nécessairement en FNBC.
- Dans la pratique, la plupart des schémas relationnels qui sont en 3FN sont aussi en FNBC. Ce n'est que si $X \rightarrow A$ est vrai dans un schéma relationnel R, X n'étant pas une superclé et A étant un attribut primaire, que R sera en 3FN mais pas en FNBC.

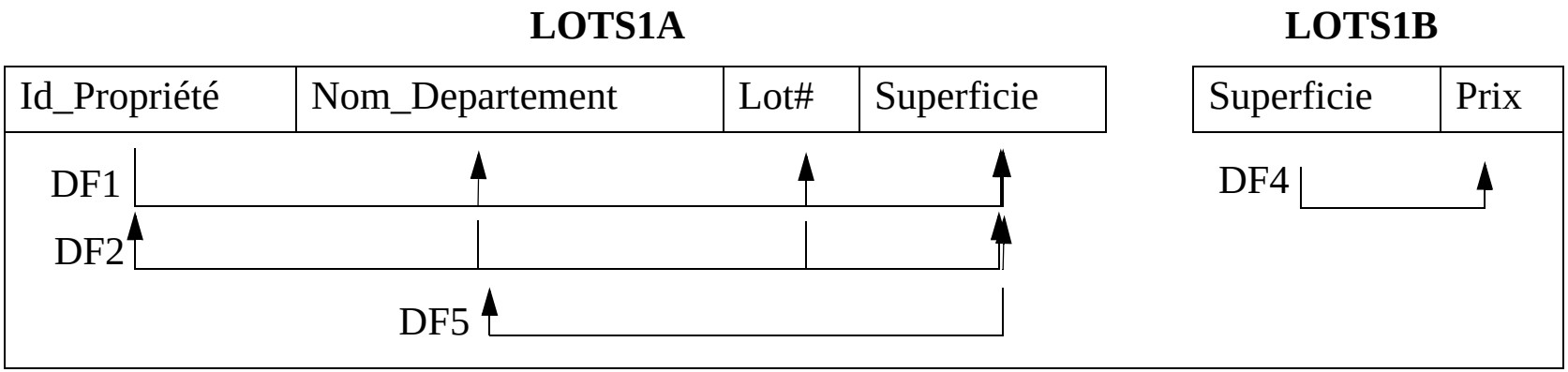


9.5.1 Nécessité d'une forme normale plus stricte

LOTS



On ajoute DF5, qui permet d'identifier le département à partir de la superficie, i.e. un dept X a des lots dont la superficie mesure un certain nombre d'unité (pas de fraction) entre deux valeurs assez rapprochées, donc connaissant la superficie on peut identifier le dept.
DF5 : SUPERFICIE $\rightarrow$ NOM_DEPARTEMENT.



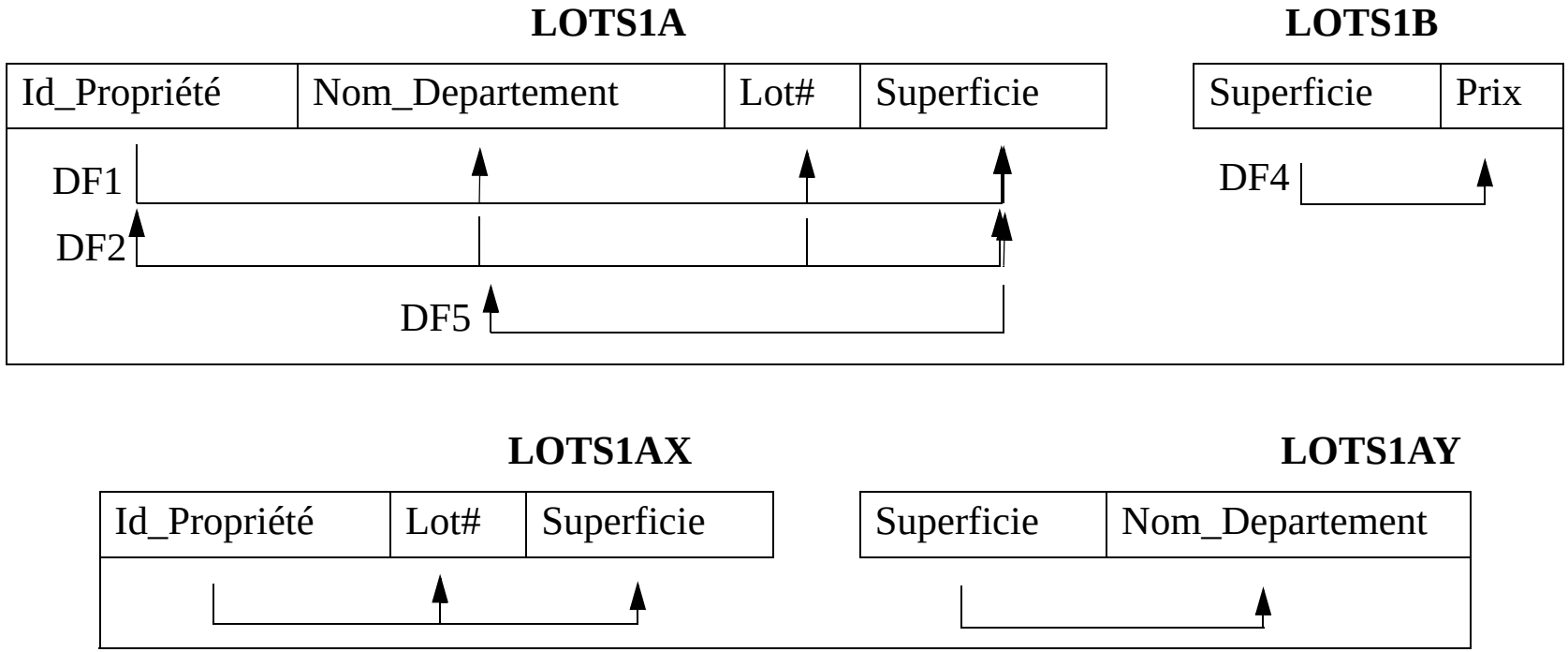
Si nous l'ajoutons aux autres dépendances, le schéma relationnel LOTS1A demeure en 3FN parce que NOM_DEPARTEMENT est un attribut premier.

La superficie d'un lot, en ce qu'elle détermine le département, comme le spécifie DF5, peut être représentée par un nombre fini de tuples dans une relation distincte R(SUPERFICIE, NOM_DEPARTEMENT) puisque SUPERFICIE n'a qu'un nombre fini (supposons 16) de valeurs possibles. Cette représentation réduit la redondance qu'entraîne la répétition des mêmes informations dans les milliers de tuples LOTS1A.

- FNBC est une forme normale supérieure qui interdirait LOTS1A et suggèrerait sa décomposition.

Solution

On peut décomposer LOTS1A en deux relations FNBC: LOTS1AX et LOTS1AY (voir figure 9.12a).



- Cette décomposition fait perdre la dépendance fonctionnelle DF2 parce que ses attributs ne coexistent plus dans la même relation après la décomposition.

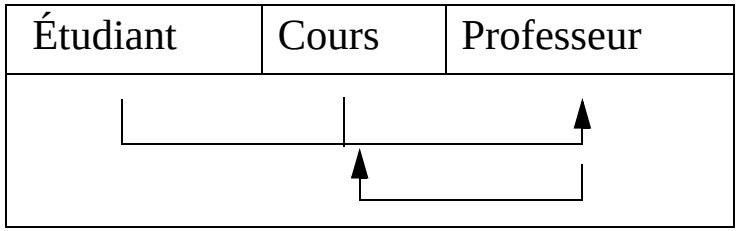
Autre exemple

ENSEIGNEMENT

Étudiant	Cours	Professeur
Narayan	Bases de données	Marchand
Ferrand	Bases de données	Navarre
Ferrand	Systèmes d'exploitation	Ammar
Ferrand	Théorie	Schulman
Lefèvre	Bases de données	Mark
Lefèvre	Systèmes d'exploitation	Ahamad
Wong	Bases de données	Omiecinski
Zelaya	Bases de données	Navarre

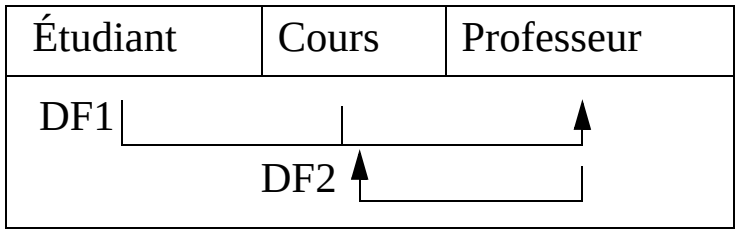
- FD1: {ETUDIANT, COURS} $\rightarrow$ PROFESSEUR
- FD2: PROFESSEUR $\rightarrow$ COURS

ENSEIGNEMENT



La solution

ENSEIGNEMENT



Il y a plusieurs solutions possibles

1. {ETUDIANT, PROFESSEUR} et {ETUDIANT, COURS}.
2. {COURS, PROFESSEUR} et {COURS, ETUDIANT}.
3. {PROFESSEUR, COURS} et {PROFESSEUR, ETUDIANT}.

- Les trois décompositions font perdre la dépendance fonctionnelle DF1
- Seule la troisième est *souhaitable* car elle ne génère pas de tuples parasites après une jointure.

9.6 Propriété de la décomposition relationnelle

Dans les sections qui suivent nous décrirons quelques algorithmes de normalisation ainsi que de nouveaux types de dépendances. Nous décrirons aussi 2 propriétés que doivent respecter les schémas relationnels : (1) la non perte de jointure ; (2) la préservation des dépendances.
Les algorithmes de normalisation commencent typiquement en synthétisant un seul schéma relationnel, appelé la **relation universelle**, laquelle est une relation théorique incluant tous les attributs de la BD. Ensuite cette relation est décomposé en de schéma de plus en plus petit jusqu'au moment où ce n'est plus possible ou que ce n'est plus désirable de le faire en fonction des fonctionnalités ou des dépendances spécifiées par le client de la BD. Soit R = $\{A_1 \ldots A_n\}$ un schéma de relation universelle.

L'**hypothèse de relation universelle** affirme que les noms d'attributs dans un schéma de relation universelle sont unique.

Nous supposerons cette dernière hypothèse toujours vraie.

Soit F un ensemble de dépendances fonctionnelles sur les attributs de R

En utilisant les dépendances fonctionnelles, l'algorithme de normalisation décompose R dans un ensemble de schéma relationnel D = $\{R_1 \ldots R_m\}$. D deviendra le schéma de base de donnée relationnel.

D est appellé une **décomposition** de R.

La **condition de préservation des attributs d'une décomposition** est $\bigcup_{i=1}^{m} R_i = R$

si cette condition est vraie on dit que D préserve les attributs

Cette condition doit être vraie pour nous assurer que nous n'avons pas perdu d'attributs.

Évidemment chacun des R_i doit être en 3FN ou FNBC.

9.6.1 Décomposition et préservation des dépendances fonctionnelles

Soit F un ensemble de dépendances fonctionnelles sur R, et $R_i \subseteq R$.

La **projection de F sur R_i**, notée $\pi_F(R_i)$, est un ensemble F' de dépendances fonctionnelles défini comme suit:

$$\pi_F(R_i) = F' = \{X \rightarrow Y \mid (X \rightarrow Y) \in F^+ \text{ et } (X \cup Y) \subseteq R_i\}$$

Une décomposition $D = \{R_1, \ldots, R_n\}$ de R préserve les dépendances fonctionnelles F dans R si et seulement si

$$\left(\bigcup_{i=1}^{n} \pi_F(R_i) \right)^+ = F^+$$

où R est le schéma relationnel utilisé pour calculer la fermeture transitive des dépendances fonctionnelles appartenant à l'ensemble de gauche .

- Soit F un ensemble de dépendances fonctionnelles sur un schéma relationnel, R.
  Il est toujours possible de trouver une décomposition $D= \{R_1, \ldots, R_n\}$ de R,
  tel que chaque R_i soit en 3FN.
  La preuve n'est pas au programme de cette session.
- Voir l'algorithme "décomposition en 3NF avec préservation des dépendances fonctionnelles" à la page 81

9.6.2 Décomposition satisfaisant la propriété de non perte de jointure

Une décomposition D = $\{R_1, \ldots, R_n\}$ de R satisfait la propriété de non perte de jointure (jointure non additive) par rapport à F ssi,
pour tout état r de R satisfaisant F

$$\Pi_{R1}(r) \bowtie \Pi_{R2}(r) \bowtie \ldots \bowtie \Pi_{Rn}(r) = r$$

$\bowtie$ est l'opérateur de jointure naturel, il permet de faire le produit cartésien en ne conservant que les lignes où les attributs de même nom ont la même valeur.
Dans la programmation SQL, ça se fait avec le WHERE dans un SELECT

9.6.3 La décompositon des relations et l'insuffisance des formes normales

EMP_SITE

Nom	EmplProjet
Bernard, A, Schmidt	Valbonne
Bernard, A Schmidt	Lanester
Robert, E, Nathan	Les Ulis
Albertine, F, Anglais	Valbonne
Albertine, F, Anglais	Lanester
Thierry B, Wong	Lanester
Thierry, B, Wong	Les Ulis
Thierry, B, Wong	Meyzieu
Jeanne C Zaoui	Meyzieu
Victor, Jabare	Meyzieu
Séverin, D, Waftwiller	Meyzieu
Séverin,Wattwiller	Les Ulis
Etienne H.Borg	Les Ulis

PROJET

NomProjet	NumeroProjet	EmplProjet	Nums
ProduitX	1	Valbonne	1
ProduitY	2	Lanester	1
ProduitZ	3	Les Ulis	1
ProduitX	1	Valbonne	1
ProduitX	2	Lanester	1
ProduitY	2	Lanester	1
ProduitZ	3	Les Ulis	1
Informatisation	10	Meyzieu	2
Réorganisation	20	Les Ulis	5
Innovation	30	Meyzieu	3
Informatisation	10	Meyzieu	2
Informatisation	10	Meyzieu	2
Innovation	30	Meyzieu	3
Innovation	30	Meyzieu	3
Informatisation	20	Les Ulis	2
Informatisation	20	Les Ulis	2

Problème

- La relation PROJET et la relation EMP_SITE sont tous les deux en FNBC mais leur jointure naturelle donne lieu à des tuples supplémentaire.
- Le problème vient de la sémantique de EMP_SITE

- La solution dans les sections qui suivent.

9.7 Les dépendances multivaluées et la 4 ième forme normale

Les dépendances multivaluées sont une conséquence de la normalisation en 1FN. La 1FN ne permet pas à un attribut d'avoir un ensemble de valeur.
Si nous avons 2 ou plus de 2 attributs indépendants multivalués dans le même schéma relationnel, nous devons répéter dans un état de la relation chaque valeur d'un des attributs avec toutes les combinaisons de valeurs des autres attributs pour garder cet état de la relation cohérent et pour maintenir l'indépendance entre les attributs impliqués.
Exemple

EMPLOYE

NomEmployé	NomProjet	NomDépendant
Didier	X	Jean
Didier	Y	Anne
Didier	X	Anne
Didier	Y	Jean

9.7.1 Définition formelle de la dépendance multivaluée

Une **dépendance multivaluée X $\twoheadrightarrow$ Y** spécifiée sur un schéma relationnel R, où X $\subseteq$ R et Y $\subseteq$ R, spécifie la contrainte suivante sur n'importe quel état r de R :

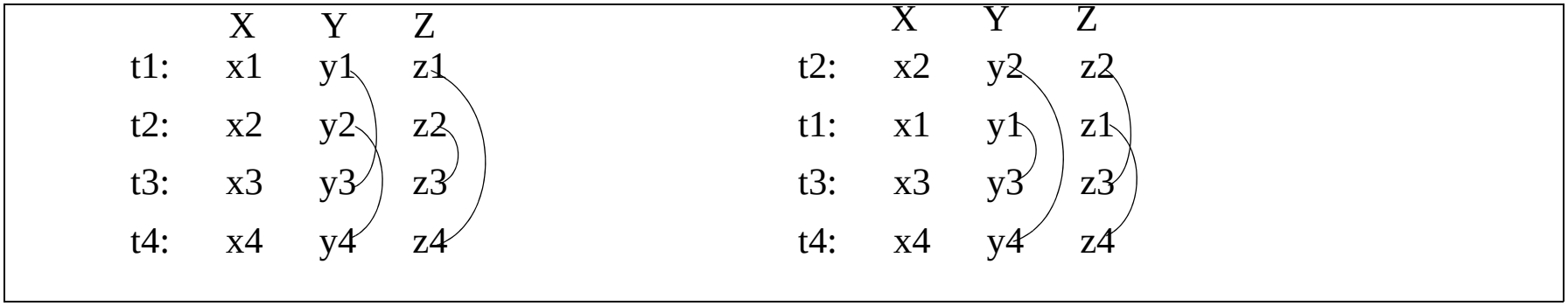
Si deux tuples t_1 et t_2 existent dans r tel que $t_1[X] = t_2[X]$ alors il existe aussi 2 couples t_3 et t_4[1] dans r avec les propriétés suivantes (où Z = R\(X $\cup$ Y)) :

1. $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
2. $t_3[Y] = t_1[Y]$ et $t_4[Y] = t_2[Y]$
3. $t_3[Z] = t_2[Z]$ et $t_4[Z] = t_1[Z]$

- Pour chaque deux couples t_1 et t_2 tel que $t_1[X] = t_2[X]$, je dois trouver t_3 et t_4, mais t_3 et t_4 peuvent être t_1 et t_2.

Si X $\twoheadrightarrow$ Y alors nous disons que **X multidétermine Y**.

- À cause de la symétrie dans la définition nous avons nécessairement que
  si X $\twoheadrightarrow$ Y alors X $\twoheadrightarrow$ Z
  - nous écrivons parfois X $\twoheadrightarrow$ Y|Z



1. Les tuples t_1, t_2, t_3 et t_4 ne sont pas nécessairement distincts

- Ce que la définition formelle spécifie que pour une valeur donnée de X, l'ensemble des valeurs de Y déterminées par cette valeur de X est complètement déterminé par cette valeur de X, i.e. elle ne dépend aucunement des autres attributs (Z). En conséqence, lorsque 2 tuples ont des valeurs distinctes de Y pour la même valeur de X, alors ces valeur de Y doivent être répétées dans des tuples séparés avec chaque valeur de Z distincte qui occure avec cette même valeur de X.

Théorême :
Soit R un schéma de relation et $X \subseteq R$ et $Y \subseteq R$.
Il existe une *dépendance mulitvaluée (plurivalente) entre X et Y*, notée $X \twoheadrightarrow Y$ ssi,
pour tout état r de R, nous avons

$$\Pi_{X \cup Y}(r) \bowtie \Pi_{R \setminus Y}(r) = r$$

Π est l'opérateur de projection, il permet de construire une nouvelle relation en sélectionnant certain attribut
$\Pi_{liste\_attributs}$ (relation) = nouvelleRelation
C'est l'équivalent de faire un SELECT sur une même relation

$\bowtie$ est l'opérature de jointure naturel, il permet de faire le produit cartésien en ne conservant que les lignes où les attributs de même nom ont la même valeur.
Ça se fait avec le WHERE dans un SELECT

Preuve de $\Rightarrow$

1. Supposons $X \twoheadrightarrow Y$
   Soit $Z = R \setminus (X \cup Y)$
   Soit r un état de R,
   Soit R1(X,Y) et r1 un état de R1 correspondant à $\Pi_{X \cup Y}(r)$]
   Soit R2(X,Z) et r2 un état de R2 correspondant à $\Pi_{R \setminus Y}(r)$]

2. Supposons $(x_a, y_a, z_b) \in (\Pi_{X \cup Y}(r) \bowtie \Pi_{R \setminus Y}(r))$

3. Alors il existe $(x_a, y_a) \in r1$ et $(x_a, z_b) \in r2$

4. Alors il existe t1= $(x_a, y_a, z_a) \in r$ et t2 = $(x_a, y_b, z_b) \in r$

5. $(x_a, y_a, z_a) \in r$
   $(x_a, y_b, z_b) \in r$
   $(x_a, y_a, z_b) \in r$
   $(x_a, y_b, z_a) \in r$ par définition de $X \twoheadrightarrow Y$

6. Donc $(x_a, y_a, z_b) \in (\Pi_{X \cup Y}(r) \bowtie \Pi_{R \setminus Y}(r))$ $\Rightarrow$ (xa, ya, zb)$\in$r

7. $(x_a, y_a, z_b) \in r$ $\Rightarrow$ $x_a, y_a, z_b) \in (\Pi_{X \cup Y}(r) \bowtie \Pi_{R \setminus Y}(r))$ est trivial

8. $\Pi_{X \cup Y}(r) \bowtie \Pi_{R \setminus Y}(r) = r$

Preuve de $\Leftarrow$ en exercice

Exemple

EMPLOYE

NomEmployé	NomProjet	NomDépendant
Didier	X	Jean
Didier	Y	Anne
Didier	X	Anne
Didier	Y	Jean

NomEmployé $\twoheadrightarrow$ NomProjet
NomEmployé $\twoheadrightarrow$ NomDépendant

la dépendance multivaluée (suite)

Une dépendance multivalué (MVD) $X \twoheadrightarrow Y$ est dite **triviale** si $Y \subseteq X$ ou si $X \cup Y = R$.

Une MVD qui n'est pas triviale est dite **non-triviale**.

EMPLOYE

NomEmployé	NomProjet	NomDépendant
Didier	X	Jean
Didier	Y	Anne
Didier	X	Anne
Didier	Y	Jean

- Comme on l'a constaté les MVD non triviale entraîne de la redondance.
- Pourtant la relation EMPLOYE est en FNBC

9.7.2 Les règles d'inférences pour les dépendances fonctionnelles et multivaluées

Soit *W, X, Y, Z* des ensembles d'attributs.

1. réflexivité : si $X \supseteq Y$, alors $X \rightarrow Y$
2. augmentation : si $X \rightarrow Y$, alors XZ[1] $\rightarrow YZ$
3. transitivité : si $X \rightarrow Y$ et $Y \rightarrow Z$, alors $X \rightarrow Z$
4. décomposition : si $X \rightarrow YZ$, alors $X \rightarrow Y$ et $X \rightarrow Z$
5. union : si $X \rightarrow Y$ et $X \rightarrow Z$, alors $X \rightarrow YZ$
6. pseudo-transitivité : si $X \rightarrow Y$ et $WY \rightarrow Z$, alors $WX \rightarrow Z$
7. complémentarité pour MVD : si $X \twoheadrightarrow Y$ alors $X \twoheadrightarrow R\backslash(X \cup Y)$
8. augmentation pour MVD : si $X \twoheadrightarrow Y$ et $W \supseteq Z$ alors $WX \twoheadrightarrow YZ$
9. transitivité pour MVD : si $X \twoheadrightarrow Y$ et $Y \twoheadrightarrow Z$, alors $X \twoheadrightarrow Z$
10. reproduction de DF à MVD : si $X \rightarrow Y$ alors $X \twoheadrightarrow Y$
11. coalescence de DF et MVD :
    si $X \twoheadrightarrow Y$ et il existe W avec les propriété a) $W \cap Y = \varnothing$, b) $W \rightarrow Z$, et c) $Y \supseteq Z$
    alors $X \rightarrow Z$

- Les 6 première sont spécifiques aux DF.
- Les règles 7, 8 et 9 sont spécifiques aux MVD
- Les règles 10 et 11 font la jonction entre les DF et les MVD

- Les preuves et les exercices seront pour l'année prochaine

---

1. Notation : XZ est une abbréviation de X U Z.

9.7.3 La quatrième forme normale

Un **schéma relationnel R est en 4FN** par rapport à F, un ensemble de dépendances fonctionnelles et multivaluées, si, pour chaque dépendance multivaluée non-triviale X $\twoheadrightarrow$ Y dans F^+, X est une superclé de R

Exemple

EMPLOYE		
Nom Employé	Nom Projet	Nom Dépendant

EMP_PROJET	
Nom Employé	Nom Projet

EMP_DEPENDANT	
Nom Employé	Nom Dépendant

Le schéma relationnel EMPLOYE n'est pas en 4FN parce que

- NomEmployé $\twoheadrightarrow$ NomProjet et NomEmployé $\twoheadrightarrow$ NomDépendant sont deux MVD non triviales et que NomEmployé n'est pas une superclé de EMPLOYE

Solution

- décomposer EMPLOYE en deux schéma en 4FN : EMP_PROJET et EMP_DEPENDANT puisque
  - dans EMP_PROJET, NomEmployé $\twoheadrightarrow$ NomProjet est une MVD triviale et, il n'y a pas d'autres dépendances fonctionnelles ou multivaluées
  - dans EMP_DEPENDANT NomEmployé $\twoheadrightarrow$ NomDépendant est une MVD triviale et, il n'y a pas d'autres dépendances fonctionnelles ou multivaluées

- La prochaine acétate illustre l'importance de la 4FN pour
  - éviter la redondance dans la BD et
  - éviter les anomalies de mise à jour (que faut-il faire si Pigot travaille pour un nouveau projet)
- Lorsque deux associations 1:N indépendantes dans un diagramme ER sont placés dans le même schéma relationnel, ce schéma a toutes les chances de ne pas être en 4 FN.

Problème et solution (explication acétate précédente)

EMPLOYE		
Nom Employé	Nom Projet	Nom Dépendant
Didier	X	Jean
Didier	Y	Anne
Didier	X	Anne
Didier	Y	Jean
Pigot	W	Lise
Pigot	X	Lise
Pigot	Y	Lise
Pigot	Z	Lise
Pigot	W	Louis
Pigot	X	Louis
Pigot	Y	Louis
Pigot	Z	Louis
Pigot	W	Luc
Pigot	X	Luc
Pigot	Y	Luc
Pigot	Z	Luc

EMP_PROJET	
Nom Employé	Nom Projet
Didier	X
Didier	Y
Pigot	W
Pigot	X
Pigot	Y
Pigot	Z

EMP_DEPENDANT	
Nom Employé	Nom Dépendant
Didier	Jean
Didier	Anne
Pigot	Lise
Pigot	Louis
Pigot	Luc

9.7.4 Décomposition sans perte de jointure (non addition de tuple)

À chaque fois que nous décomposons un schéma relationnel R en un schéma R1 = (X∪Y) et R2= R\Y basé sur une MVD $X \twoheadrightarrow Y$ vraie sur R, la décomposition possède la propriété de jointure non-additive.

Il peut être montré que ceci est une condition suffisante et nécessaire pour décomposer un schéma en deux schéma qui possèdent la propriété de jointure non additive.

En d'autres termes **la propriété de jointure sans perte** s'énonce comme suit :
Les relations R_1 et R_2 forment une décomposition sans perte de jointure (jointure non additive) par rapport à un ensemble F de dépendances fonctionnelles et multivaluées si et seulement si
$R_1 \cap R_2 \twoheadrightarrow R_1 \backslash R_2$ ou de façon symétrique $R_1 \cap R_2 \twoheadrightarrow R_2 \backslash R_1$

Algorithme de décomposition en schémas relationnels 4FN avec la propriété de jointure non-additive

```
Soit R, un schéma relationnel universel
Soit F, un ensemble de dépendances fonctionnelles ou multivaluées

1.D := {R}

2.tant qu'il existe un schéma de relation Q dans D non en 4NF faire
    {trouver X ↠ Y une DP en Q qui viole 4FN
    remplacer Q dans D par les deux schémas X∪Y et Q\Y}.
```

Attention ce dernier algorithme ne préserve pas les dépendances fonctionnelles.

9.8 Dépendance de jointure et la cinquième forme normale

Nous avons vu une condition nécessaire et suffisante pour qu'un schéma relationnel soit décomposé en deux schémas et où la décomposition possède la propriété de jointure non-additive.
Cependant, dans certains cas, il peut ne pas exister de décomposition sans perte de jointure de R en deux relations mais il peut exister une décomposition sans perte de jointure en plus de deux schémas.
De plus, il peut ne pas exister de dépendances fonctionnelles dans R qui violent les formes normales jusqu'à FNBC, et il peut ne pas y avoir de MVD non-triviale dans R qui violent la 4FN. Dans ces cas, nous utilisons un autre type de dépendance, appelé dépendance de jointure, et si elle est présente, nous effectuons une décomposition à plusieurs branches vers une 5FN. Il est important de noter que ce type de dépendance une une contrainte sémantique particulière qui est très difficile à détecter en pratique ; en conséquence, la normalisation jusqu'à la 5FN est rarement faite en pratique.

9.8.1 Définition formelle de la dépendance de jointure

Soit R un schéma relationnel
Soit F un ensemble de dépendances fonctionnelles définies sur R,
Soit D = $\{R_1, \ldots, R_n\}$ une décomposition de R.
Il existe une dépendance de jointure, notée JD = $\{R_1, \ldots, R_n\}$ dans R
ssi, pour tout état r de R satisfaisant F

$$\Pi_{R1}(r) \bowtie \Pi_{R2} \bowtie \ldots \bowtie \Pi_{Rn}(r) = r$$

Notons qu'une MVD est un cas particulier de dépendance de jointure avec n=2.

9.8.2 Dépendance de jointure triviale

Une dépendance de jointure, DJ(R_1, R_2, …, R_n), spécifiée sur un schéma relationnel R, est **une DJ triviale** ssi un des schémas relationnels $\exists R_i \in \{R_1, R_2, \ldots, R_n\}$ tel que $R_i = R$.

- Une telle dépendance est appelée triviale parce qu'elle confère la propriété de jointure non-additive à n'importe quel état de r de R et par conséquent elle ne spécifie par le fait même aucune contrainte sur R.

9.8.3 définition de la 5FN

Un schéma de relation est en 5FN par rapport à un ensemble F de dépendances fonctionnelles, multivaluées ou de jointure si, pour chaque dépendance de jointure non triviale, DJ(R_1, R_2, …, R_n), dans F+ (i.e. qui découlent de F) chaque R_i est une superclé de R.

9.8.4 exemple :

FOURNISSEUR		
Fournisseur	Pièce	Projet
Pigot	boulon	ProjX
Pigot	vis	ProjY
Girard	boulon	ProjY
Lebeau	vis	ProjZ
Girard	clou	ProjX
Girard	boulon	ProjX
Pigot	boulon	ProjY

- Supposons que nous avons la contrainte suivante :
  À chaque fois, (1) qu'un fournisseur *f* fournit une pièce *p*,
  (2) qu'un projet *j* utilise une pièce *p*,
  (3) que le fournisseur *f* fournit une pièce quelconque au projet *j*,
  alors le fournisseur *f* fournit aussi la pièce p au projet *j* .
- Cette contrainte peut être reformulée d'une autre façon et spécifiée une dépendance de jointure JD(R1,R2, R3) entre les trois projections R1(Fournisseur, Pièce), R2(Fournisseur, Projet) et R3(Pièce, Projet) de fournisseur.
- Si cette contrainte est vraie, les couples en-dessous du trait gras doivent exister dans tous les états légaux de la relation FOURNISSEUR qui contiennent aussi les tuples au-dessus du trait gras.

Solution

FOURNISSEUR		
Fournisseur	Pièce	Projet
Pigot	boulon	ProjX
Pigot	vis	ProjY
Girard	boulon	ProjY
Lebeau	vis	ProjZ
Girard	clou	ProjX
Girard	boulon	ProjX
Pigot	boulon	ProjY

R1	
Fournisseur	Pièce
Pigot	boulon
Pigot	vis
Girard	boulon
Lebeau	vis
Girard	clou

R2	
Fournisseur	Projet
Pigot	ProjX
Pigot	ProjY
Girard	ProjY
Lebeau	ProjZ
Girard	ProjX

R3	
Pièce	Projet
boulon	ProjX
vis	ProjY
boulon	ProjY
vis	ProjZ
clou	ProjX

- L’application de la jointure naturelle à une paire de ces relations produit des tuples erronés mais la jointure des trois ensemble n’en produit pas.

9.9 Algorithmes de décomposition de schéma

1. décomposition en 3NF avec préservation des dépendances fonctionnelles (algorithme 13.1)
2. décomposition en BCNF avec jointure non additive (algorithme 13.3)
3. décomposition en 3NF avec préservation des dépendances fonctionnelles et jointure non additive (algorithme 13.4)
4. décomposition en 4NF avec jointure non additive (algorithme 13.5)
5. décomposition en 5NF avec jointure non additive

9.9.1 décomposition en 3NF avec préservation des dépendances fonctionnelles

Soit R le schéma relationnel à décomposer

1. Trouver F un ensemble de dépendances fonctionnelles minimal sur R.
2. Soit D = {}
3. Pour chaque X tel que $\exists A : X \rightarrow A \in F$ faire
   ajouter à D le schéma de relation $R_X(X, A_1, \ldots, A_m)$,
   où $X \rightarrow A_i \in F$
4. Ajouter à D une relation $R_B(B_1, \ldots, B_n)$,
   où B_i est un attribut de R qui n'a pas été inclus
   dans aucune relation à l'étape précédente l
   (cette étape est nécessaire pour préserver l'ensemble des attributs)

La décomposition est l'ensemble des schémas relationnels créés
i.e. D contient les R_X et le R_B

9.9.2 décomposition en BCNF avec jointure non additive

Soit R le schéma relationnel à décomposer

1.D := {R}

2.tant qu'il existe un schéma de relation Q dans D non en BCNF faire

soit X $\rightarrow$Y une DF en Q qui viole BCNF

dans D, remplacer Q par les deux schémas X $\cup$Y et Q - Y.

9.9.3 décomposition en 3NF avec préservation des dépendances fonctionnelles et jointure non additive

Soit R le schéma relationnel à décomposer

1. Trouver F un ensemble de dépendances fonctionnelles minimal sur R.
2. Soit D = {}
3. Pour chaque X tel que $\exists A : X \rightarrow A \in F$ faire
   ajouter à D le schéma de relation $R_X(X, A_1, \ldots, A_m)$,
   où $X \rightarrow A_i \in F$
4. Si aucun des schémas ne comporte une clé pour R,
   ajouter un schéma de relation avec
   des attributs qui forment une clé pour R

- C'est la quatrième étape que la jointure des schémas relationnels créés n'ajoutera pas de tuples superflus.
- Les trois premières étapes sont similaires à celle de "décomposition en 3NF avec préservation des dépendances fonctionnelles" à la page 81.
  La quatrième étape de ce dernier algorithme n'est plus nécessaire parce que les attributs n'ont pris en compte étaient nécessairement des attributs primaires.

9.9.4 Algorithme de décomposition en 5NF avec jointure non additive

```
D := {R}
tant qu'il existe un schéma de relation Q dans D non en 5NF faire
  soit JD(Q1,…,Qm) une DP en Q qui viole 5NF
  remplacer Q par les schémas Q1,…,Qm.
```