Système de Gestion de BD
PostgreSql

Hocine ABIR

27 novembre 2008

IUT de Villetaneuse
Université de Paris XIII
E-mail: abir@iutv.univ-paris13.fr

TABLE DES MATIÈRES

CHAPITRE

ONZE

Normalisation des Données

11.1 Introduction

Une Base de Données (non-normalisée) est vulnérable aux anomalies si ses données comportent des redondances. Si une donnée est enregistrée deux ou plusieurs fois et que seul une partie de ces données est modifiée alors la base est inconsistente.

Le but de la normalisation est :
- d'éliminer certaines types de redondances dans les données,
- d'organiser les données efficacement,
- d'éviter certaines erreurs potentielles durant les opérations sur les données et
- de simplement renforcer la consistence des données
- de créer une représentaion "correcte" du "monde réel"

Ce qui signifie que toutes entités de la base de données doivent être normalisées.

Une entité est normalisée si et seulement si tous les attributs non-clés :

1. sont mutuellement *indépendants*
2. dépendent totalement de la clé primaire

La normalisation est le processus qui permet de supprimer la redondance des données en décomposant une entité en plusieurs entités. Cette décomposition doit se faire sans perte d'informations et sans perte de dépendances.

11.2 Structure de Données et Terminologie

11.2.1 Intégrité des données

L'intégrité des données signifie en partie qu'un utilisateur peut naviguer et manipuler ces données de manière consistente.

On distingue en général, deux règles de base pour assurer l'intégrité des données :

Règle d'intégrité d'entité

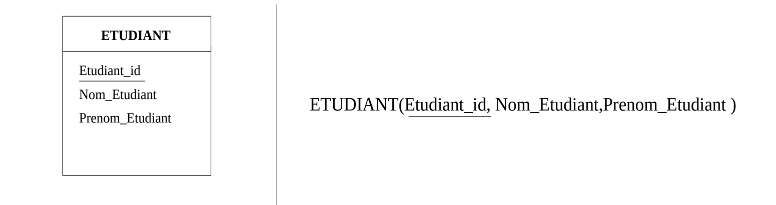
La valeur d'une clé primaire ne peut être indéterminée car son rôle est d'identifier les instances d'une entité. La règle assure que les opérations de manipulation des données (insert, update, delete) maintiennent l'unicité et la définition de la clé primaire.

Règle d'intégrité d'association

Si une entité contient une clé étrangère alors chacune de ses valeurs :
- doit être indéfinie ou
- doit correspondre à une valeur de clé primaire dans l'entité référenciée.

11.2.2 Entités vs Relations

Dans le modèle entité association, une entité est representé graphiquement par un rectangle comme par exemple :



Entité ETUDIANT (notation graphique) — Entité ETUDIANT (notation texte)

On peut aussi représenter linéairement (texte) une entité par la notation utilisé dans les modèles relationnelles. L'entité est appelée alors relation et sa description ets appelée *schéma de relation*. Dans la suite, nous utiliserons indiféremment les deux termes *entité* et *relation* tant qu'il n'y pas d'ambiguité.

Un ensemble d'instances d'une entité peut être representé sous forme tabulaire, comme par exemple :

ETUDIANT

Etudiant_id	Nom_Etudiant	Prenom_Etudiant
20060101	AMAR	Jeremy
20060102	AMAR	
20060103	CAZUC	David
20060104	TANG	Michel

11.3 Manipulation des données

11.3.1 Définition

L'opération de jointure, notée $\bowtie$, combine deux relations $R(A^1, A^2, ..., A^n)$ et $S(B^1, B^2, ..., B^m)$ pour former une relation $Q(A^1, A^2, ..., A^n, B^1, B^2, ..., B^m)$ ayant $n + m$ attributs. Les tuples de Q sont formés en concaténant les tuples de R et S qui vérifie une *condition de jointure* C. Cette opération est notée :

$$Q = R \bowtie^C S$$

11.3.2 Rôle de la Jointure

Importance de la Jointure dans un modèle de Données

- fréquemment utilisée,

- la plus coûteuse,
- la plus sensible aux données,
- utilisée pour exprimer divers liens sémantiques,
- mécanisme général : union, différence, intersection, élimination de doublets, etc.

Quand et Où la jointure devient nécessaire

Le processus de conception des bases de données génère le besoin de la jointure :

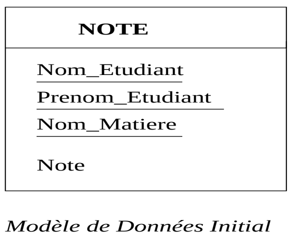


FIG. 11.1 – Modèle Non-Normalisé

En partant du modèle initial de la Figure (11.1), l'analyse décompose cette relation en la normalisant dans le but :
- de faciliter les mises à jour (pas de redondance),
- d'assurer la consistence des données.

On obtient le modèle conceptuel de données de la Figure (11.2) :

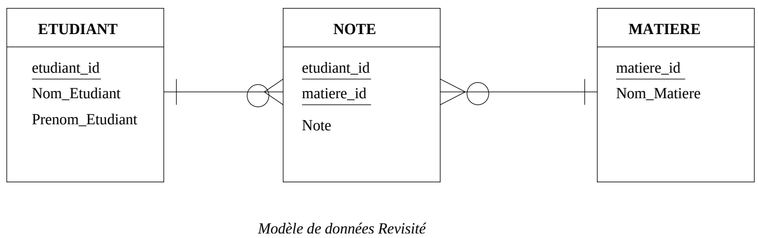


FIG. 11.2 – Modèle Normalisé

Le modèle de la Figure (11.2) est constitué de deux entités, **Etudiant** et **Matiere**, et d'une entité associative **Note** qui établit le lien : "la note obtenue par un étudiant dans une matière" ou inversement. Chaque entité a un identifiant désignant la référence (logique) de l'objet désigné par l'entité.

On peut remarquer que les identifiants **Etudiant_id** et **Matiere_id** sont utilisés dans la relation **Note**. Les attributs **Etudiant_id** et **Matiere_id** de **Note** sont appelés *Clés étrangères* de **Note**. Ils permettent d'établir un lien sémantique abstrait entre les relations *Etudiant* et *Matiere*. Ce lien sémantique est du type *clé primaire-clé étrangère*.

Si l'on veut décrire la réponse à la requête : *"Quels sont les noms des étudiants ayant une note dans au moins une matière ?"*, alors nous devons matérialiser le lien logique entre *etudiant* et *Note*. La *jointure* est un mécanisme qui permet de matérialiser ce lien :

```
SELECT Nom_Etudiant
   FROM Etudiant x, Note y
   WHERE x.Etudiant_id=y.Etudiant_id;
```

Une variété d'autres liens logiques peuvent exister entre les données d'une ou plusieurs relations : la jointure permet de déduire ces liens

- *non clé - clé étrangère* :
  Quels sont les Numéros des étudiants ayant une note identique à leur numéro ?
- *clé - non clé étrangère* :
  Quels sont les numéros des étudiants pour lesquels il y a au moins une matiere ayant le même numéro ?
- *non clé - non clé étrangère* :
  Quels sont les étudiants ayant un nom identique à un nom de matière ?

11.4 Clé Primaire

Une clé primaire est un attribut ou ensemble d'attributs (clé composite) qui détermine (ou identifie) une instance unique de l'entité. Une clé primaire doit avoir les propriétés suivantes :

- la valeur de la clé doit être définie pour chaque instance,
- la valeur de la clé doit être unique pour chaque instance,
- la valeur de la clé doit être stable c'est à dire ne change (mise à jour) pas dans le temps.

11.4.1 Définition

Une clé primaire est une contrainte structurelle sur un schéma de relation $R(p, x)$, où p et x sont des sous-ensembles d'attributs tel que :

- p détermine (ou identifie) x

$$\forall t^i, t^j \in R(t^i.p = t^j.p \Rightarrow i = j) \tag{11.1}$$

  c'est à dire :

$$\forall t^i, t^j \in R(t^i.p \neq t^j.p \vee i = j)$$

- k est minimal : il n'existe pas ($p^{'} \subset p$) qui satisfait (11.1) ci-dessus.

11.4.2 Exemple

ETUDIANT
<u>Etudiant_id</u>
Nom_Etudiant
Prenom_Etudiant

Etudiant_id	Nom_Etudiant	Prenom_Etudiant
20060101	AMAR	Jeremy
20060102	AMAR	
20060103	CAZUC	David
20060104	TANG	Michel

11.5 Qu'est ce qu'une Dépendance Fonctionnelle

11.5.1 Définition

Une Dépendance Fonctionnelle est une contrainte sur un schéma de relation $R(x, y, z)$ que l'on note sous la forme $x \longrightarrow y$, où x (appelé déterminant) et y sont des sous-ensembles d'attributs de R qui satisfont (11.2).

$$\forall t^i, t^j \in R(t^i.x = t^j.x \Rightarrow t^i.y = t^j.y) \tag{11.2}$$

c'est à dire :

$$\forall t^i, t^j \in R(t^i.x \neq t^j.x \vee t^i.y = t^j.y)$$

11.5.2 Exemple

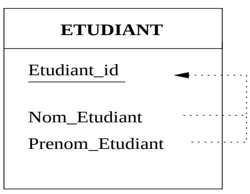
ETUDIANT

Etudiant_id	Nom_Etudiant	Prenom_Etudiant
20060101	AMAR	Jeremy
20060102	AMAR	Remi
20060103	CAZUC	David
20060104	TANG	Michel

- On a **les dépendances** :
  - $Etudiant\_id \longrightarrow Etudiant\_Nom$
  - $Etudiant\_id \longrightarrow Etudiant\_Prenom$
  - $Etudiant\_Prenom \longrightarrow Etudiant\_Nom$

- On **n'a pas la dépendance** :
  $Etudiant\_Nom \longrightarrow Etudiant\_Prenom$

Les dépendances fonctionnelles peuvent être décrites graphiqueement par des "flèches", comme par exemple :



Nom_Etudiant, Prenom_Etudiant $\longrightarrow$ Etudiant_id

11.5.3 Clés et Dépendances Fonctionnelles

La notion de clé est étroitement liée à celle de dépendance fonctionnelle. En terme de dépendance fonctionnelle, une clé primaire d'une relation $R(K, A)$ est un sous-ensemble K d'attributs de R tel que :

1. $K \longrightarrow A$: K détermine tous les attributs de R , K est dite superclé.
2. Aucun sous-ensemble de K ne peut satisfaire (1) : K est minimal (clé candidate).

11.6 Raisonner avec les Dépendances Fonctionnelles

Etant donné une relation R et un ensemble F de Dépendences Fonctionnelles (DFs) :

Peut-on déduire de F d'autres DFs

Est-ce que F contient des DFs **redondantes** ?

Est-ce K est une clé de R

Quelles sont les clés de R ?

11.6.1 Inférence de nouvelles DFs

Introduction

Dans un modèle de Données, seules les dépendences fonctionnelles évidentes et usuelles sont explicitement spécifiées. De ces dépendences fonctionnelles, plusieurs autres dépendences fonctionnelles peuvent être déduites.

On appelle fermeture F^+ d'un ensemble de dépendences F :
- l'ensemble de toutes les dépendances de $F \cup \{$les dépendances déductibles de $F\}$
- Une dépendence $X \longrightarrow Y$ est déductible de F si
  $X \longrightarrow Y$ est valide pour toutes les instances de la relation pour laquelle F est valide.
- $X \longrightarrow Y \in F^+$ si et seulement si $F \models X \longrightarrow Y$ en utilisant les règles ci-dessous.

Axiomes d'Armstrong

A1 Réflexivité :
Si $Y \subseteq X$ alors $X \longrightarrow Y$

A2 Augmentation :
Si $X \longrightarrow Y$ alors $XZ \longrightarrow YZ$ (et $XZ \longrightarrow Y$)

A3 Transitivité :
Si $X \longrightarrow Y$ et Si $Y \longrightarrow Z$ alors $X \longrightarrow Z$

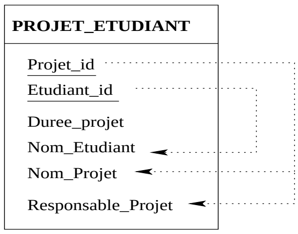
Autres règles d'inférences utiles (dérivées) :

A4 Décomposition :
Si $X \longrightarrow YZ$ alors $X \longrightarrow Y$ et $X \longrightarrow Z$

A5 Union :
Si $X \longrightarrow Y$ et $X \longrightarrow Z$ alors $X \longrightarrow YZ$

A6 Pseudo-transitivité :
Si $WY \longrightarrow Z$ et $X \longrightarrow Y$ alors $WX \longrightarrow Z$

Exemple de dérivation de DFs



{ Projet_id , Etudiant_id } $\longrightarrow$ Duree_projet
{ Projet_id , Etudiant_id } $\longrightarrow$ Nom_Etudiant
{ Projet_id , Etudiant_id } $\longrightarrow$ Nom_Projet
{ Projet_id , Etudiant_id } $\longrightarrow$ Responsable_Projet

{ Projet_id , Etudiant_id } $\longrightarrow$ { Duree_projet , Nom_Etudiant }

15 possibiltés

{ Projet_id , Etudiant_id , Duree_projet } $\longrightarrow$ 7 possibiltés
{ Projet_id , Etudiant_id , Nom_Etudiant } $\longrightarrow$ 7 possibiltés
{ Projet_id , Etudiant_id , Duree_projet , Nom_Etudiant } $\longrightarrow$ 3 possiblités

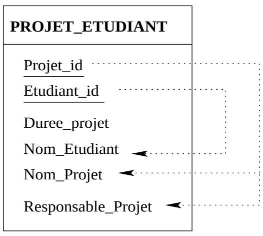
Fermeture d'un ensemble d'attributs

Définition : La fermeture X^+ d'un ensemble d'attributs X avec un ensemble F de DFs :
- est l'ensemble des attributs A tel que $X \longrightarrow A$ peut être dérivée de F avec les règles d'Armstrong
- permet de déterminer si une DF $X \longrightarrow Y$ est déductible de F

Algorithme : (Axiomes A3 et A4)

```
X+ = X ;
Repeter
{
      AncienX+ = X+ ;
      Pour Chaque DF Y ⟶ Z dans F
      {
            si (X+ ⊇ Y)
            {
                  X+ = X+ ∪ Z ;
            }
      }
} tant que (X+ = AncienX+) ;
```

Exemple :



$\{$Etudiant_id$\}^+$=$\{$Etudiant_id, Nom_Etudinat$\}$

$\{$Projet_id$\}^+$ = $\{$Projet_id, Nom_Projet, Responsable_Projet$\}$

Couvertures Minimales de DFs

Permet de déterminer les clés candidates.

Introduction : Deux ensembles de DFs F et G sont équivalents si :
- Chaque DF de F peut être dérivée de G et
- Chaque DF de G peut être dérivée de F

F et G sont équivalents si $F^+ = G^+$.

Définition : Un ensemble F de dépendences fonctionnelles est minimal si

(1) Chaque DF de F a comme membre droit un seul attribut,

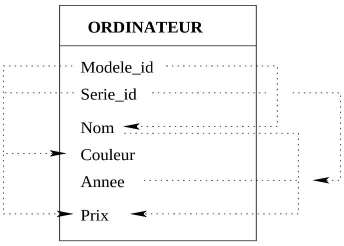
(2) Aucun sous-emsemble de F n'est équivalent à F

(3) Aucune dépendance $X \longrightarrow A$ de F ne peut être remplacée par $Y \longrightarrow A$ avec $Y \subset X$ en produisant un ensemble de DFs équivalent à F.

Algorithme : Couverture minimal G d'un ensemble de Dfs F :

G=F ;
Pour chaque DF ($X \longrightarrow A1, ..., AN \in G$)
{
    Remplacer $X \longrightarrow A1, ..., AN$ par N DFs :
        $X \longrightarrow A1$
        $X \longrightarrow ...$
        $X \longrightarrow AN$;
}
Pour chaque DF ($X \longrightarrow A \in G$)
{
    Pour chaque attribut B de X
    {
        si ($\{G - \{X \longrightarrow A\}\} \cup \{(X - \{B\}) \longrightarrow A\}$ est équivalent à G)
        {
            Remplacer $X \longrightarrow A$ par $(X - \{B\}) \longrightarrow A$;
        }
    }
}
Pour chaque DF ($X \longrightarrow A \in G$)
{
    si ($G - \{X \longrightarrow A\}$ est équivalent à G)
    {
        G=$G - \{X \longrightarrow A\}$;
    }
}

Exemple :



Modele_id , Serie_id $\longrightarrow$ Couleur

Modele_id , Serie_id $\longrightarrow$ Prix

Modele_id $\longrightarrow$ Nom
Serie_id $\longrightarrow$ Annee
Nom , Annee $\longrightarrow$ Prix

Couvertue Minimale :

Modele_id , Serie_id $\longrightarrow$ Couleur

Modele_id $\longrightarrow$ Nom

Serie_id $\longrightarrow$ Annee

Nom , Annee $\longrightarrow$ Prix

11.7 Dépendances Fonctionnelles et Décomposition

11.7.1 Problèmes potentiels

Jointure

Performance : Relations plus petites $\longrightarrow$ plus de jointure dans les requêtes

Perte d'Informations

Perte de Dépendances Fonctionnelles

Une décomposition n'éliminent pas toujours ces 3 problèmes.

11.7.2 Décomposition Sans Perte d'Informations (SPI)

Définition

Soit $R(X)$ un schéma de relation et F un ensemble de dépendences sur R. La décomposition de R en deux relation $R1(X1)$ et $R2(X2)$ est sans perte d'information si une des conditions suivantes est satisfaite :

(a) $X1 \cap X2 \longrightarrow X1$

(b) $X1 \cap X2 \longrightarrow X2$

c'est à dire que les attributs $X1 \cap X2$ intervenants dans la jointure naturelle constituent une clé candidate pour au moins une des relations.

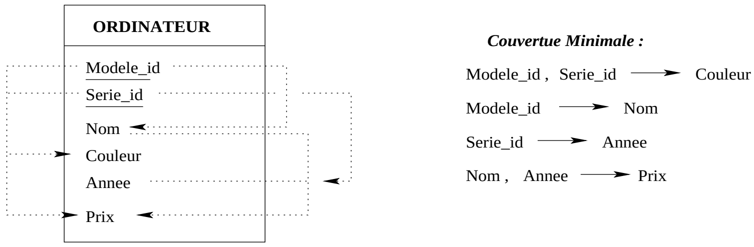
Algorithme de décomposition SPI

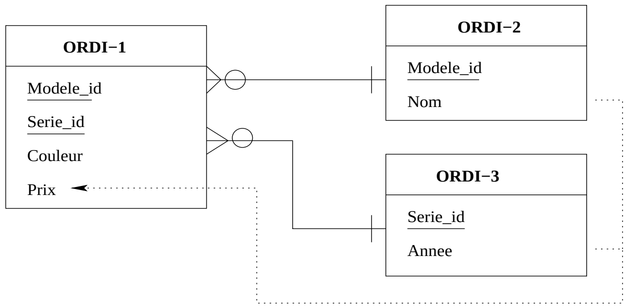
Décomposition SPI d'une relation R en $D = R1, R2, ..., Rm$ par rapport à un ensemble de DFs F tel que chaque Ri est sous forme FNBC.

```
D = {R} ;
Tant que ( ∃Q(Z) ∈ D tel que Q(Z) n'est pas FNBC)
{
        Choisir une DF X ⟶ Y qui viole FNBC ;
        Remplacer Q(Z) par deux relations :
                (1) Qi(Z − Y) et
                (2) Qj(X ∪ Y) ;
}
```

La Préservation des Dépendances Fonctionnelles n'est pas garantie.

Exemple





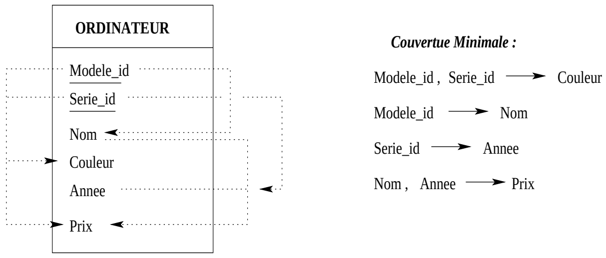
11.7.3 Décomposition Préservant les Dépendances Fonctionnelles (PDF)

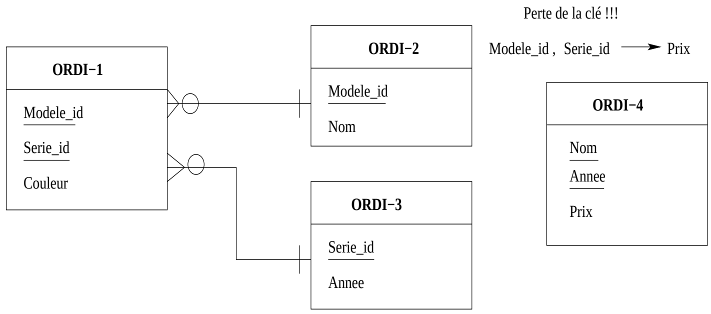
Soit un schéma de relation R avec un ensemble de DFs F et une décomposition Q de R en m relations $D = R1, R2, ..., Rm$:

- Si la décomposition est PDF alors chaque dépendence $X \longrightarrow Y$ de F apparaît explicitement ou dérivable dans une relation Ri
- Sinon pour préserver l'information, des DFs inter-relations sont nécessaires (cad dépendances valides sur la jointure de plusieurs relations)

Dans une décomposition PDF, les contraintes F sont toutes vérifiées dans les Ri.

Exemple





Définition

- La projection $\Pi^F(R^i)$ de F sur R^i est l'ensemble des dépendances $X \longrightarrow Y$ de F^+ tel que $X \in Ri$ et $Y \in Ri$ (les parties gauche et droite de la DF sont des attributs de Ri.
- Une décomposition $D = R1, R2, ..., Rm$ est PDF (préserve les dépendances) si $(\Pi^F(R^1) \cup ... \cup \Pi^F(R^m))^+ = F^+$.

Algorithme

produit une décomposition SPI et PDF (3FN)

Déterminer une couverture minimal G de F ;
Pour Chaque X des DFs $X \longrightarrow Y \in G$
{
  Créer une relation Ri dans D avec les attributs $X \cup A1 \cup ... \cup Ak$
  où $X \longrightarrow A1 ... X \longrightarrow Ak$ sont les seules Dfs de G
  ayant X en partie gauche (X est la clé de Ri) ;
}
Si ($\exists C1... \cup CL$ des attributs de G qui n'apparaissent dans aucune Ri)
{
  créer une relation ayant pour clé l'ensemble de toutes les clés des Ri
  et comme autres atributs $C1... \cup CL$;
}

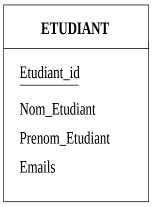
11.7.4 Dépendances Fonctionnelles et Redondances

Une Forme Normale est une contrainte d'intégrité sur une entité qui garantit que cette entité est sous une certaine forme qui préserve l'entité de certaines types d'anomalies. On distingue différentes formes normales, chacune d'entre est caractérisée par un type d'anomalie.

11.8 Première Forme Normale

11.8.1 Anomalies

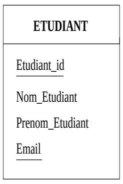
Cas 1 : Attribut Multi-valué



ETUDIANT

Etudiant_id	Nom_Etudiant	Prenom_Etudiant	Emails
20060101	AMAR	Jeremy	{jer@free.fr, ama@hotmail.fr}
20060103	CAZUC	David	caz@aol.fr
20060104	TANG	David	{tan@frre.fr, dav@aol.fr}

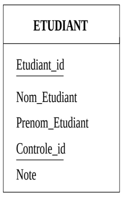
Normalisation FN-1 :



ETUDIANT

Etudiant_id	Nom_Etudiant	Prenom_Etudiant	Email
20060101	AMAR	Jeremy	jer@free.fr
20060101	AMAR	Jeremy	ama@hotmail.fr
20060103	CAZUC	David	caz@aol.fr
20060104	TANG	David	tan@frre.fr
20060104	TANG	David	dav@aol.fr

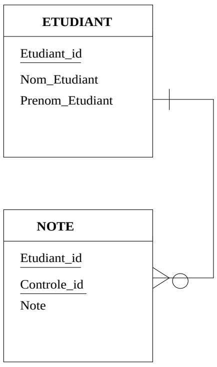
Cas 2 : Relations Imbriquées



ETUDIANT

Etudiant_id	Nom_Etudiant	Prenom_Etudiant	Controle_id	Note
20060101	AMAR	Jeremy	110101	12
20060101	AMAR	Jeremy	110102	11
20060103	CAZUC	David	110101	11
20060104	TANG	David	110101	13
20060104	TANG	David	110102	12

Normalisation FN-1 :



ETUDIANT

Etudiant_id	Nom_Etudiant	Prenom_Etudiant
20060101	AMAR	Jeremy
20060103	CAZUC	David
20060104	TANG	David

NOTE

Etudiant_id	Controle_id	Note
20060101	110101	12
20060101	110102	11
20060103	110101	11
20060104	110101	13
20060104	110102	12

11.8.2 Définiton

La Première Forme Normale est une contrainte sur un schéma de relation R tel que :

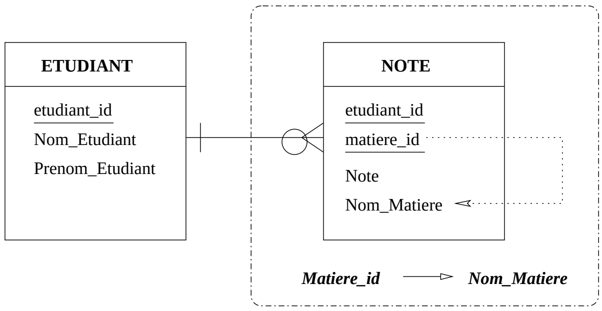
Si a est un attribut de R alors le domaine de a est atomique

11.8.3 Remarques

- *rôle* : permet de différencier {a,b} de {b,a}
- *conséquence* : Dépendences Multi-valuées.

11.9 Deuxième Forme Normale

11.9.1 Anomalies



NOTE

Etudiant_id	Mariere_id	Note	Nom_matiere
20060101	1101	12	CA−BD
20060102	1101	13	CA−BD
20060103	1101	10	CA−BD
20060104	1101	14	CA−BD

INSERT : On peut associer des noms de matiere différents **Nom_Matiere** pour un même **Matiere_id**.

DELETE : Si on supprime toutes les notes d'une matiere, on perd son numéro **Matiere_id** et son nom **Nom_Matiere**.

UPDATE : On peut modifier partiellement les différentes ocurrences du nom **Nom_Matiere** pour un même **Matiere_id**.

11.9.2 Définiton

La Deuxième Forme Normale (2FN) est une contrainte sur un schéma de relation $R(p, x)$ où p est une clé primaire tel que :

$$\forall x^i \in x, \forall p^j \subset p \quad (p \longrightarrow x^i \wedge \neg(p^j \longrightarrow x^i)) \tag{11.3}$$

c'est à dire que chaque attribut x^i de x est entièrement déterminé par p (et pas une partie de p).

Redondance :

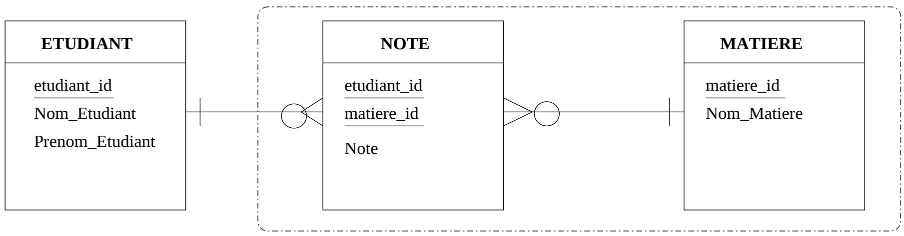
Si un attribut $x^i \in x$ dépend seulement d'une partie de la clé $p^j \subset p$ alors la valeur de x^i est répétée pour chaque valeur p^j .

11.9.3 Solution : decomposition en 2FN

Soit un schéma de relation de la forme $R(p^i, p^j, x^i, x^j)$ où p^i, p^j est la clé primaire et x^i et x^j sont des sous-ensemble d'attributs de R tel que $p^i \longrightarrow x^i$. La décomposition en 2FN de R consiste à remplacer R par deux relations :

1. $R^i(p^i, x^i)$ où p^i est la clé primaire de R^i
2. $R^j(p^i, p^j, x^j)$ où p^i, p^j est la clé primaire de R^j

Cette décomposition est sans perte d'information.

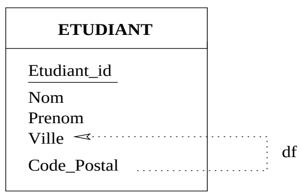


11.9.4 Remarques

- Seules les relations ayant une clé primaire composite peuvent ne pas être en deuxième forme normale càd présenter ce type d'anomalie.
- Chaque attribut non-clé doit dépendre totalement (entièrement) de la clé primaire.

11.10 Troisième Forme Normale

11.10.1 Anomalies



Code Postal $\longrightarrow$ **Ville**

ETUDIANT

Etudiant_id	Nom	Prenom	Ville	Code_Postsal
20060101	AMAR	Jeremy	Villetaneuse	93630
20060102	AMAR	Remi	Villetaneuse	93630
20060103	CAZUC	David	Villetaneuse	93630
20060104	TANG	Michel	Villetaneuse	93630

df

INSERT : On peut associer des noms de ville différents **Ville** pour un même **Code_Postal** et inversement plusieurs codes postaux pour la même ville.

DELETE : Si on supprime tous les étudiants d'une ville, on perd son code postal **Code_Postal** et son nom **Ville**.

UPDATE : On peut associer des noms de ville différents **Ville** pour un même **Code_Postal** et inversement plusieurs codes postaux pour la même ville.

11.10.2 Définiton

La Deuxième Forme Normale (3FN) est une contrainte sur un schéma de relation $R(p, x)$ où p est une clé primaire telque :

$$\forall x^i \in x, \forall x^j \in x \quad \neg(p \longrightarrow x^i \wedge x^i \longrightarrow x^j) \tag{11.4}$$

c'est à dire que chaque attribut x^j de x n'est pas déterminé transitivement par un autre autre attribut x^i.

Redondance :

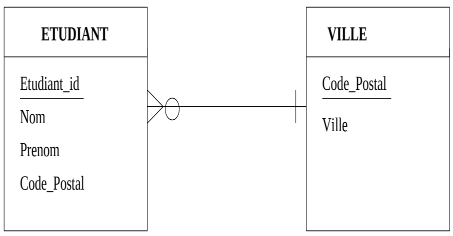
Si un attribut $x^j \in x$ dépend transitivement de x^i alors la valeur de x^j est répétée pour chaque valeur x^i .

11.10.3 Solution : decomposition en 3FN

Soit un schéma de relation de la forme $R(p, x^i, x^j, y)$ où p est la clé primaire et x^i et x^j sont des attributs de R tel que $x^i \longrightarrow x^j$. La décomposition en 3FN de R consiste à remplacer R par deux relations :

1. $R^i(p, x^i, y)$ où p est la clé primaire de R^i
2. $R^j(x^i, x^j)$ où x^i est la clé primaire de R^j

Cette décomposition est sans perte d'information.



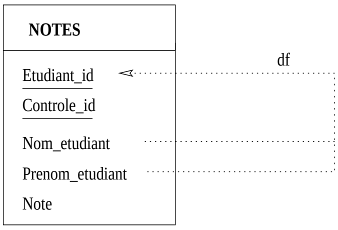
11.10.4 Remarques

- Dans une relation 3FN , tous les attributs non-clés sont mutuellement indépendants.

11.11 Forme normale de Boyce Codd

11.11.1 Anomalies

Cas 1 : DF entre deux clés candidates



Nom_etudiant, Prenom_etudiant ⟶ **Etudiant_id**

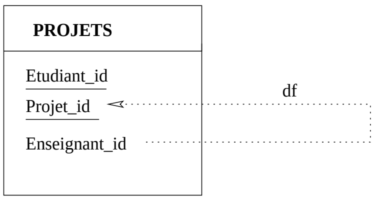
A Chaque **Etudiant_id**, les **Nom_etudiant** et **Prenom_etudiant** sont répétés. le couple déterminant (**Nom_etudiant**, **Prenom_etudiant**) n'est pas *clé candidate*.

INSERT : On peut insérer deux etudiants **Etudiant_id** avec le même nom **Nom_etudiant** et prenom **Prenom_etudiant**.

DELETE : Si on supprime les notes d'un étudiants, on les informations sur son numéro **Etudiant_id**, son nom **Nom_etudiant** et son prénom **Prenom_etudiant**.

UPDATE : On peut modifiers deux etudiants **Etudiant_id** avec le même nom **Nom_etudiant** et prenom **Prenom_etudiant** ou avoir plusieurs noms et prénoms pour le même étudiant.

Cas 2 : DF entre un attribut non-clé et une clé



Enseignant_id ⟶ **Projet_id**

A Chaque **Projet_id**, les **Enseignant_id** sont répétés. le déterminant **Enseignant_id** n'est pas *clé candidate*.

INSERT : On insérer un même enseignant **Enseignant_id** avec deux projets différents **Projet_id**.

DELETE : Si on supprime un projet **Projet_id**, on perd l'enseignant **Enseignant_id** associé.

UPDATE : On peut affecter un enseignant **Enseignant_id** à plusieurs projets **Projet_id**

.

11.11.2 Définiton

La Forme normale de Boyce Codd (FNBC) est une contrainte sur un schéma de relation $R(p, x)$ où p est une clé primaire telque :

$$\forall x^i \in x, \forall x^j \subset x(x^j \longrightarrow x^i \Rightarrow x^j = p) \tag{11.5}$$

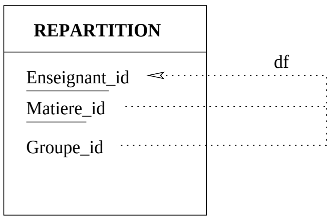
c'est à dire que toutes les contraintes de dépendances du schéma de relation $R(p, x)$ sont de la forme $p \longrightarrow x^i$ où x^i est un attribut de x.

Redondance :

Si un attribut $x^i \in x$ détermine une partie de la clé $p^j \subset p$ alors la valeur de p^j est répétée pour chaque valeur x^i .

Exemple :

Déterminant de la DF est clé candidate



Groupe_id, Matiere_id $\longrightarrow$ **Enseignant_id**

REPARTITION

Enseignant	Matiere	Groupe
Abir	CA−BD	G1
Lebbah	CA−BD	G2
Gerard	CA−BD	G3
Levy	CA−BD	Z

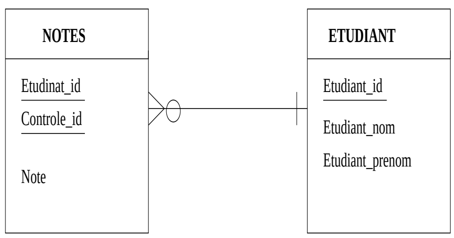
df

A Chaque **Enseignant_id**, les attributs **Matiere_id** et **Groupe_id** sont répétés.

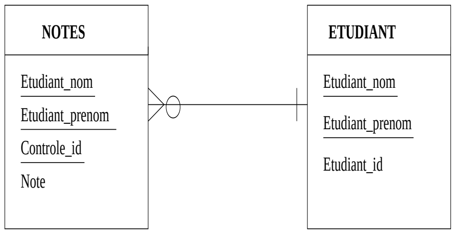
11.11.3 Solution : decomposition en FNBC

Cas 1 : deux solutions

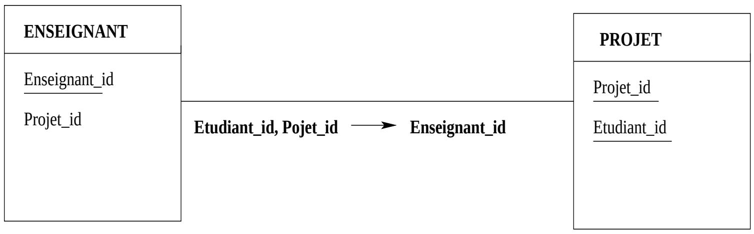
Solution 1



Solution 2



Cas 2 :



La dépendance **Etudiant_id**, **Projet_id** $\longrightarrow$ **Enseignant_id** est perdue. Les mises à jour des relations **Enseignant** et **Projet** sont liées par une contrainte qui sera traduite sous forme de contrainte inter-relation. .

11.11.4 Remarques

- la FNBC gère les inter-dépendences entre les attributs non-primes mais qui font partie d'une clé candidate.
- Une relation est FNBC si tous les déterminants des DFs sont des clé candidates.
- Une relation n'est pas FNBC si deux de ses clés candidates partagent au moins un attribut.
  1. elle possède plusieurs clés candidates,
  2. les clés sont composées de plusieurs attributs,
  3. il y a des attributs communs entre les clés

11.12 Processus de Normalisation