LES FORMES NORMALES

Modèle relationnel

La théorie de la normalisation

- elle met en évidence les relations "indésirables"
- elle définit les critères des relations "désirables" appelées **formes normales**
- Propriétés indésirables des relations
  - Redondances
  - Valeurs NULL
- elle définit le processus de normalisation permettant de décomposer une relation non normalisée en un ensemble équivalent de relations normalisées

Normalisation

- **Le processus de transformation d'une relation posant des problèmes lors des mises à jour en relations n'ayant pas ces problèmes, est appelé processus de normalisation.**

- **On mesure la qualité d'une relation (ou sa capacité à représenter le monde réel sans générer des problèmes) par son degré de normalisation.**

- **Une relation peut être (de la moins bonne à la meilleure) en 1ère forme normale, en 2ème forme normale, en 3ème forme normale, en forme normale de Boyce Codd, en 4ème forme normale ... (chaque forme normale implique les précédentes).**

Normalisation

- **Normaliser** une relation qui pose des problèmes de **mise-à-jour** (relation non normalisée), consiste à **décomposer** cette relation en relations sans problèmes (**relations normalisées**). La méthode à suivre est la suivante:

- **1/** vérifier que la relation est en **première forme normale**

- **2/** établir son graphe minimum des dépendances;

- **3/** déterminer, à l'aide du graphe, tous ses identifiants;

- **4/** déterminer, à l'aide du graphe, sa **forme normale**

- **5/** si la relation n'est pas normalisée, **décomposer**, à l'aide du graphe, la relation en relations mieux normalisées

Les formes normales

5 FN, les critères sont de plus en plus restrictifs

$$FNj \Rightarrow FNi \quad (j > i)$$

- **Notion intuitive de FN**

  une « *bonne relation* » peut être considérée comme une fonction de la clé primaire vers les attributs restants

La décomposition

Objectif:

- décomposer les relations du schéma relationnel sans perte d'informations
- obtenir des relations canoniques ou de base du monde réel
- aboutir au schéma relationnel normalisé

- Le schéma de départ est le schéma universel de la base

- Par raffinement successifs ont obtient des sous relations sans perte d'informations et qui ne seront pas affectées lors des mises à jour (non redondance)

Dépendances fonctionnelles et graphe des dépendances fonctionnelles

- **Définition**
- **Etant donné une relation, R (X, Y, Z), il existe une dépendance fonctionnelle, ou "DF", de Y vers Z, notée Y→Z, si, étant donné deux tuples quelconques de R, s'ils ont même valeur pour Y, alors ils ont nécessairement même valeur pour Z. Y source de la DF, et Z cible de la DF**

- **Exemple: La relation Produit (NP, NomP, Poids, Couleur), il y a les DF suivantes en supposant qu'il n'existe pas deux produits de même nom: NP →NomP, NomP → NP, NP →Poids, NomP → Poids, NP →Couleur, NomP → Couleur NP →(NomP, Poids, Couleur), (NP, NomP) →Poids, (NP, NomP) →Couleur**

Dépendances fonctionnelles et graphe des dépendances fonctionnelles

- **Définition :**

une DF, X → B, est une dépendance fonctionnelle élémentaire si B est un attribut unique, et si X est un ensemble minimum d'attributs (ou un attribut unique).

Exemples : Dans la relation Produit, les DF :

NP → (couleur, poids) et (NP, NomP) →Poids

ne sont pas élémentaires.

Mais les DF : NP → Couleur, NP → Poids, NP →NomP , NomP → Couleur, NomP → Poids, NomP →NP

sont élémentaires.

Relation: Livraison (NP, NF, Date, Qté)

La DF : (NP, NF, date) → Qté est élémentaire

Dépendances fonctionnelles et graphe des dépendances fonctionnelles

- **Chaque DF traduit un fait du monde réel.**
- **Les DF élémentaires traduisent des faits atomiques.**
- **La DF, NP → Couleur, signifie que chaque produit, identifié par son numéro, est d'une seule couleur.**
- **La DF, (NP, NF, date) → Qté, signifie qu'un même fournisseur ne peut livrer plusieurs fois le même jour le même produit avec des quantités différentes.**

Dépendances fonctionnelles et graphe des dépendances fonctionnelles

- **Propriété des dépendances fonctionnelles:**

Si dans une relation, on a les deux dépendances fonctionnelles, X→Y et Y→Z, alors on a aussi la dépendance fonctionnelle X→Z qui est dite déduite des deux autres.

- **Définition :**

Etant donné une relation et un ensemble F de DF portant sur les attributs de cette relation, on appelle graphe minimum des DF de la relation, tout ensemble de DF élémentaires non déduites, équivalent à F en ce sens que toute DF de F peut être déduite des DF du graphe.

Dépendances fonctionnelles et graphe des dépendances fonctionnelles

- Une méthode pour savoir si une DF, X→Y, est déduite des autres est la suivante:
- établir un graphe (non minimum) des DF,
- supprimer la DF X→Y du graphe,
- parcourir tous les chemins possibles partant de X et suivant les DF. La DF, X→Y, est déduite si un (ou plusieurs) de ces chemins atteint Y.

Le graphe minimum des DF sert essentiellement à définir des relations normalisées.

Tout graphe (minimum ou pas) des DF peut être employé pour la recherche des identifiants des relations de la façon suivante: chercher, sur le graphe des DF de la relation, tout ensemble minimum d'attributs, X, tel que tous les chemins partant de X et suivant les DF atteignent tous les autres attributs du graphe. Alors X est un identifiant de la relation

Décomposition d'une relation

- **Objectif : étant donné une relation non satisfaisante, en ce sens qu'elle implique des répétitions au niveau de sa population et qu'elle pose des problèmes lors des insertions/modifications/suppressions de tuples. Exemple: Livraison (NP, NF, Date, TélF, Qté)**

exemples de problèmes « *Livraison* » : s'il n'y a plus de livraison pour un fournisseur son numéro de téléphone est perdu. S'il existe N livraisons pour un fournisseur, le numéro TélF (téléphone du fournisseur) est répété N fois.

trouver un ensemble de relations satisfaisantes et qui décrive les mêmes informations

La méthode consiste à décomposer la relation en deux (ou plusieurs) relations. Pour cela, il est nécessaire de disposer de deux opérations qui permettent:

*** l'une, de découper une relation en sous relations (projection),**

*** et l'autre, de recomposer la relation à partir de ses sous-relations (la jointure).**

Décomposition d'une relation

- **On peut décomposer une relation en un ensemble de relations projetées, si à partir des relations projetées, on peut, en faisant leur jointure, retrouver la relation initiale.**
- **Ainsi toute requête, qu'elle soit posée à la relation initiale ou aux relations issues de la décomposition, donnera le même résultat. La relation initiale et les relations issues de la décomposition constituent deux bases de données équivalentes.**

- **Définition: Une décomposition d'une relation R(X,Y,Z) en deux relations R1=π[X,Y]R et R2=π[X,Z] est dite "sans perte d'information" si R = R1*R2.**

Exemples sur la base FormaPerm:

• La décomposition de la relation Personne en:
π[n°P, nom] Personne et π[n°P, adr] Personne
est une décomposition sans perte d'information, car:
(π[n°P, nom] Personne) * (π[n°P, adr] Personne) = Personne.
On remarquera cependant que cette décomposition est inutile, car la relation Personne est "bonne".

• La décomposition de la relation Personne en:
π[n°P, nom] Personne et π[nom, adr] Personne
est une décomposition avec perte d'information, car:
(π[n°P, nom] Personne) * (π[nom, adr] Personne) =

n°P	nom	adr
1111	Rochat	Lausanne
1111	Rochat	Renens
6666	Walter	Lausanne
5555	Bernard	Yverdon
2222	Rochat	Lausanne
2222	Rochat	Renens
3333	Muller	Morges

La jointure sur l'attribut nom a multiplié les tuples Rochat. On remarquera que l'on parle de "perte d'information" alors qu'il y a plus de tuples qu'auparavant. En effet, on ne sait plus quels sont les "bons" tuples et les "faux" tuples. Les décompositions sans perte d'information assurent que toute requête posée à la base de données initiale (avant décomposition) et la requête équivalente posée sur la base de données issue de la décomposition donnent le même résultat.

Décomposition d'une relation

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

- **Principe de la démonstration:**

- **R1*R2 contient au moins tous les tuples de R, puisque tout tuple xyz de R crée un tuple xy dans R1 et un tuple xz dans R2, qui sont concaténés par la jointure en xyz.**
- **- R1*R2 ne peut pas contenir de tuples en plus de ceux de R. Ceci est démontré par l'absurde. Soit xyz un tuple de R1*R2, qui n'appartient pas à R. xyz provient de deux tuples de R: xyz' et xy'z. Etant donné que xyz n'appartient pas à R, on a: z'≠z et y'≠y, ce qui est contraire à la DF X→Y.**

1ère Forme Normale 1FN

Une relation est en 1FN si tout attribut est atomique (non décomposable) **Une relation est en première forme normale si chaque valeur de chaque attribut de chaque tuple est une valeur simple (tous les attributs sont simples et monovalués).**

Contre-exemple

ELEVE (**no_elv**, nom, prenom, liste_notes)

Un attribut ne peut pas être un ensemble de valeurs

Décomposition

ELEVE (**no_elv**, nom, prenom)

NOTE (no_elv, no_matiere, note)

2ème Forme Normale 2FN

si chaque attribut qui ne fait partie d'aucun identifiant dépend de tout identifiant entier (et non pas d'un morceau d'identifiant).

Une relation est en 2FN si
- elle est en 1FN
- si tout attribut n'appartenant pas à la clé ne dépend pas d'une partie de la clé

- C'est la phase d'identification des clés
- Cette étape évite certaines redondances
- Tout attribut doit dépendre fonctionnellement de la totalité de la clé

2ème Forme Normale 2FN

Contre-exemple

une relation en 1FN qui n'est pas en 2FN

COMMANDE (**date**, **no_cli**, **no_pro**, qte, prixUHT)

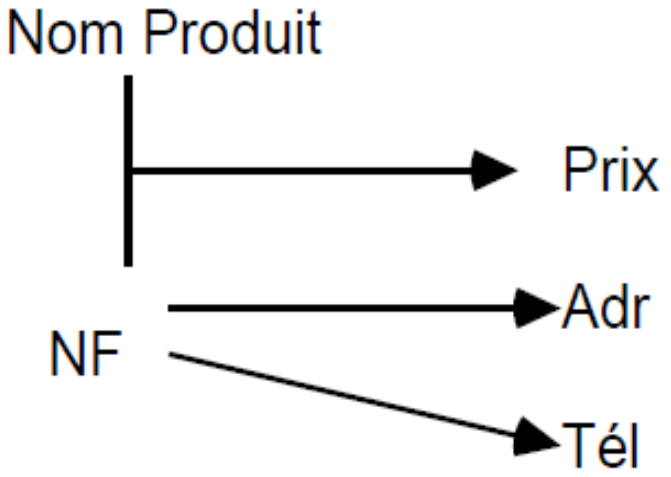
elle n'est pas en 2FN car la clé **= (date, no_cli, no_pro)**, et le prixUHT ne dépend que de no_pro

Décomposition

COMMANDE (**date**, **no_cli**, **no_pro**, qte)
PRODUIT (**no_pro**, prixUHT)

2ème Forme Normale 2FN

- Soit une relation qui est en 1ère forme normale, mais qui n'est pas en 2ème forme normale:
- Fournisseur1 (NF, NomProduit, Adr, Tel, Prix)
- Graphe minimum des DF de Fournisseur1:
- Une telle relation pose des problèmes :



- **Redondances**: s'il existe 100 produits pour un fournisseur on va répéter 100 fois le nom, l'adres, téléph du Fourns
- **Problème m-à-j pour les insertions**: quand on veut rajouter un produit, il faut rentrer à nouveau l'adrs et télph du Fourn
- **Problème pour les suppressions**: si on supprime (momentanément) la liste des produits d'un fournisseur, alors on supprime aussi le fournisseur.
- **Problème de mise à jour des tuples**: si un fournisseur change d'adresse ou de téléphone, il faut faire cette mise à jour sur tous les 100 tuples !

2ème Forme Normale 2FN

- Ces problèmes sont dus au fait que la relation n'est pas en 2ème forme normale.

- On décompose Fournisseur1 en deux relations de la façon suivante : pour chaque source de DF, on crée une relation ayant pour attributs la source et tous les attributs en dépendance fonctionnelle directe de cette source, en s'assurant qu'une (au moins) des deux sources est entièrement contenue dans les attributs communs aux deux relations ainsi créées (*théorème de Heath*). On obtient ainsi

- Fournisseur (NF, Adr, Tel)
- Catalogue (NF, NomProduit, Prix)
- qui sont en 2ème forme normale.

- Cette décomposition est :
- sans perte d'information (NF est l'identifiant de la relation Fournisseur)
- sans perte de dépendance fonctionnelle (les DF sont soit dans l'une, soit dans l'autre des deux relations décomposées).

3ème Forme Normale 3FN

Une relation est en 3FN si
- elle est en 2FN
- si tout attribut n'appartenant pas à la clé ne dépend pas d'un attribut non clé

Ceci correspond à la non transitivité des D.F. ce qui évite les redondances.

En 3FN une relation préserve les D.F. et est sans perte.

une relation est en 3FN si elle est en première forme normale et si chaque attribut qui ne fait partie d'aucun identifiant dépend uniquement des identifiants entiers.

NB : On peut toujours décomposer en 3FN sans perte ni d'information, ni de DF; ce qui n'est pas vrai des formes normales plus poussées. D'où l'intérêt de cette 3FN

3ème Forme Normale 3FN

Contre-exemple

une relation en 2FN qui n'est pas en 3FN

VOITURE (**matricule**, marque, modèle, puissance)

on vérifie qu'elle est en 2FN ; elle n'est pas en 3FN car la clé = **matricule**, et la puissance dépend de (marque, modèle)

Décomposition

VOITURE (**matricule**, marque, modèle)

MODELE (**marque, modèle**, puissance)

3ème Forme Normale 3FN

- Soit une relation qui est en deuxième forme normale :
- Fournisseur2 (N°F, Pays, Ville)
- Il existe les DF : NF → Ville et Ville → Pays
- car on suppose qu'on n'a dans la base de données que des grandes villes de différents noms.
- La DF : NF → Pays est une DF déduite.
- Graphe minimum des DF de Fournisseur2 :

- Dans la relation Fournisseur2, il y a redondance : le pays d'une ville est répété, ce qui cause des problèmes de mise à jour. On décompose donc en :
- Fourn (NF, Ville) Géo (Ville, Pays)
- Cette décomposition est sans perte d'information (Ville est identifiant pour Géo), et sans perte de DF (les DF non déduites sont soit dans Fourn, soit dans Géo).

3ème Forme Normale de BOYCE-CODD BCNF

Relations à plusieurs identifiants

Une relation est en BCFN :
- elle est en 1FN et
- ssi les seules D.F. élémentaires sont celles dans lesquelles une clé détermine un attribut

- BCNF signifie que l'on ne peut pas avoir un attribut (ou groupe d'attributs) déterminant un autre attribut et distinct de la clé

- Ceci évite les redondances dans l'extension de la relation: mêmes valeurs pour certains attributs de n-uplets différents

- BCNF est plus fin que FN3 : BCNF $\Rightarrow$ FN3

3ème Forme Normale de BOYCE-CODD BCNF

Contre-exemple

une relation en 3FN qui n'est pas BCNF

CODEPOSTAL (**ville**, **rue**, code)

on vérifie qu'elle est FN3, elle n'est pas BCNF car la clé = (ville, rue) (ou (code, ville) ou (code, rue)), et code → ville

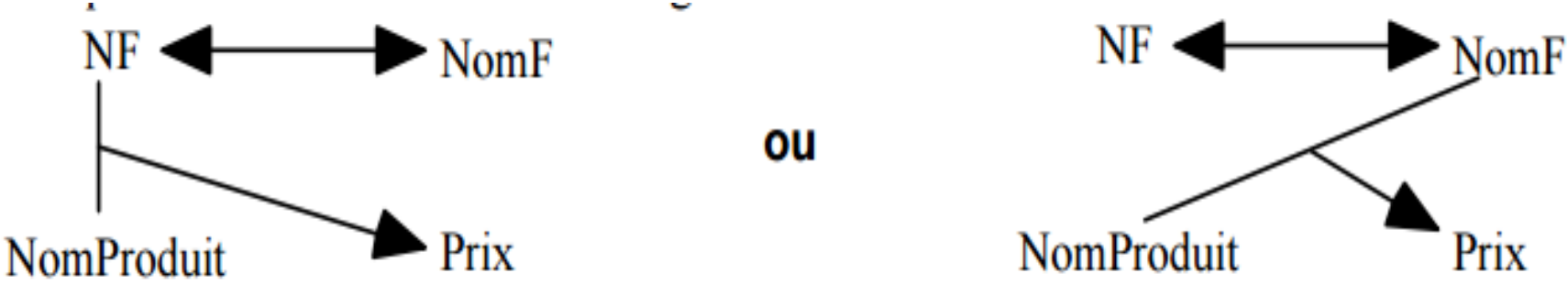
3ème Forme Normale de BOYCE-CODD BCNF

Relations à plusieurs identifiants

- Soit une relation qui est en troisième forme normale (on suppose qu'il n'y a pas mêmes noms de fournisseurs) : Catalogue3 (NF, NomF, NomProduit, Prix)

Identifiants : (NF + NomProduit) , (NomF + NomProduit)

- Cette relation possède deux graphes minimum des DF. Ils sont équivalents et qui conduisent à des décompositions équivalentes.



Dans la relation Catalogue3, il y a redondance entre NF et NomF, ce qui génère des problèmes lors des mises à jour. Par exemple si un fournisseur change de nom, il faut mettre à jour son nom dans tous les tuples correspondants à ses produits. On décompose donc Catalogue3 en deux relations de forme normale de Boyce Codd, par exemple en:

Fournisseur (NF , NomF) qui a deux identifiants : (NF) et (NomF)

Catalogue (NF, NomProduit, Prix) qui a un identifiant composé:(NF+ NomProduit)

3ème Forme Normale de BOYCE-CODD BCNF

Relations à plusieurs identifiants

- **Définition :**

- **Une relation est en forme normale de Boyce Codd si elle est en première forme normale et si toute source complète de DF est un identifiant entier.**

- **La relation Catalogue3 n'est pas en forme normale de Boyce Codd parce que l'attribut NF est source complète de DF (NF → NomF) et n'est pas un identifiant entier. Il en est de même pour NomF.**
- **On remarque qu'une relation en forme normale de Boyce Codd, est en troisième forme normale.**

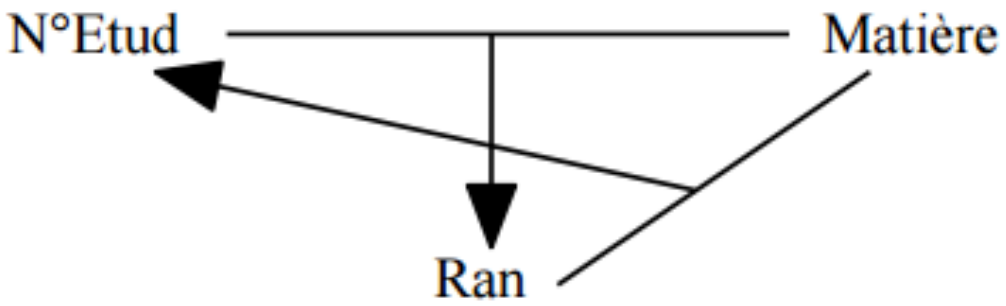
3ème Forme Normale de BOYCE-CODD BCNF

Relations à plusieurs identifiants

- **Autre exemple La relation: Place (N°Etud, Matière, Rang) représente le rang obtenu par chaque étudiant pour chaque matière. Il y a deux identifiants :**

(N°Etud + Matière) , (Matière + Rang).

Graphe minimum des DF de Place :



Est-on en troisième forme normale ? Oui, car il n'y a pas d'attribut qui ne fasse pas partie d'un identifiant.
On est aussi en Boyce Codd. On ne décompose donc pas

3ème Forme Normale de BOYCE-CODD BCNF

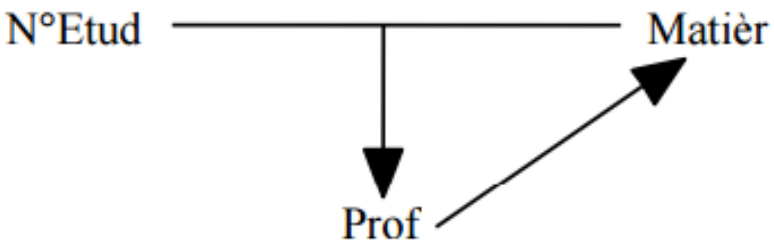
Relations à plusieurs identifiants

Autre exemple

Enseignement (N°Etud, Matière, Prof)

On suppose que chaque professeur enseigne une seule matière, et que pour chaque matière, chaque étudiant suit les cours d'un seul professeur.

Graphe minimum des DF de Enseignement :



La relation Enseignement a deux identifiants : (N°Etud + Matière) et (N°Etud + Prof). Elle est en 3ème forme normale puisque tout attribut fait partie d'un identifiant. Mais elle n'est pas en forme normale de Boyce Codd puisque l'attribut Prof est source de DF et ne constitue pas un identifiant entier.

3ème Forme Normale de BOYCE-CODD BCNF

Relations à plusieurs identifiants

Cette relation présente des inconvénients (redondances, problèmes de mise à jour) dus au fait que la DF, Prof → Matière, n'est pas traduite par la relation. Notamment si un professeur change de spécialité, il faut penser à faire cette mise à jour dans tous les tuples où apparaît ce professeur.
Cependant on ne peut pas décomposer la relation Enseignement sans perdre de DF. Par exemple, la décomposition selon Heath:

(Prof, Matière)
(N°Etud, Prof)

perd la DF: (N°Etud, Matière) → Prof. Cette décomposition permet d'insérer le fait qu'un même étudiant suit deux cours portant sur la même matière avec deux professeurs différents, ce qu'interdisait la relation Enseignement.

Il n'y a pas de solution idéale dans ce cas. Pratiquement on conserve la relation initiale et on rajoute une contrainte d'intégrité spécifiant le fait qu'un professeur ne peut enseigner qu'une seule matière.

Méthodes de décomposition

- **Pour décomposer une relation non normalisée en un ensemble de relations normalisées, plusieurs méthodes conduisant à des algorithmes ont été proposées, mais aucune d'entre elles, appliquée de façon purement automatique, n'est totalement satisfaisante.**

- **En effet, elles peuvent proposer en résultat des relations normalisées, mais qui ne sont pas sémantiquement significatives.**

1- Décomposition en Boyce Codd sans perte d'information

- **Il est possible de décomposer toute relation en forme normale de Boyce Codd sans perte d'information en appliquant récursivement le théorème de Heath : *Tant que dans une des relations obtenues par décomposition, R(X,Y,Z), il existe une DF X→Z, on décompose R en π[X,Y]R et π[X,Z]R*.**

- **Cette méthode présente plusieurs inconvénients :**

***elle peut conduire à trop décomposer, notamment en décomposant une relation qui est déjà en Boyce Codd.**

Une décomposition n'est utile que si la relation initiale n'est pas assez normalisée et que les relations issues de la décomposition sont plus normalisées. Un schéma trop décomposé complique l'emploi de la base de données, car les requêtes seront plus complexes (nombreuses jointures).

*** elle peut perdre des DFs, voir par exemple le cas de la relation: Enseignement (N°Etud, Matière, Prof). Les dépendances perdues doivent alors être ajoutées au schéma décomposé sous la forme de contraintes d'intégrité.**

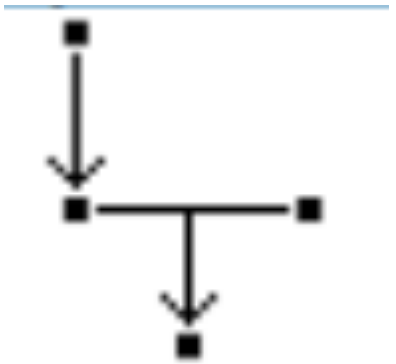
2- Décomposition en troisième forme normale sans perte d'information ni de dépendance fonctionnelle

- **Il est possible de décomposer toute relation R en troisième forme normale sans perte d'information ni de DF, en suivant la méthode ci-dessous :**

***créer pour chaque source de DF une relation comprenant comme attributs la source et toutes les cibles de cette source;**

*** si aucun des identifiants de R n'est présent dans au moins une des relations créées, ajouter une relation supplémentaire, constituée des attributs composant un des identifiants de R. Cette relation supplémentaire supprimera les tuples excédentaires lors de la jointure des relations issues de la décomposition pour reconstituer la relation initiale.**

Par exemple, le cas suivant décomposé en deux relations (sans relation contenant l'identifiant de la relation initiale), ne satisfait pas le théorème de Heath et génère lors de la jointure des tuples exédentaires.



2- Décomposition en troisième forme normale sans perte d'information ni de dépendance fonctionnelle

- **Cette méthode de décomposition a cependant l'inconvénient de générer parfois des décompositions redondantes, par exemple dans le cas de la relation Enseignement (N°Etud, Matière, Prof).**

Il faut donc ensuite supprimer du résultat les relations qui sont incluses dans (c'est-à-dire égales à une projection d') une autre relation.