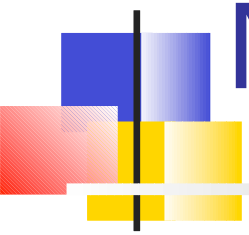


Normalisation des relations

PARTIE I

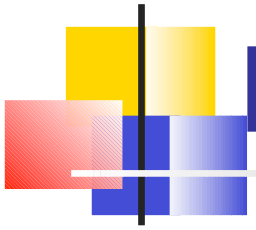
Dépendances et formes normales

Sylvie Cazalens

Patricia Serrano Alvarado

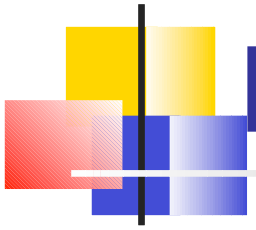
Dpt Informatique

Université de Nantes



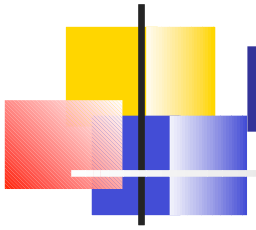
Formes normales : pourquoi ?

- Lors de la gestion de la base, certaines relations peuvent poser un certain nombre de problèmes.
- Les formes normales caractérisent les relations qui présentent moins de problèmes que d'autres.
- Les formes normales sont définies à partir de l'analyse des « liens » entre les valeurs des attributs de la relation : les dépendances.



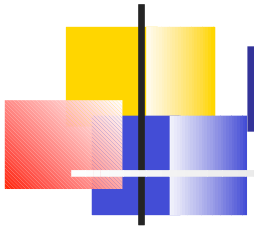
Problèmes (anomalies)

- de redondance : la même valeur d'un sous-ensemble d'attributs apparaît plusieurs fois.
- à l'ajout : l'ajout de certaines informations n'est possible que si d'autres sont présentes.
- à la suppression : la suppression de certaines informations, entraîne celle d'autres infos (que l'on aurait bien aimé conserver !)
- à la mise à jour : la modification d'une info doit être répercutée autant de fois qu'elle apparaît dans la relation.



Exemple (B. Defude, C. Date)

Produit	Quantité	Couleur	Fournisseur	Adresse
parapluie	110	rouge	Labaleine	Paris
chapeau	50	vert	Lemelon	Nantes
ceinture	65	noir	ToutCuir	Nantes
parasol	15	jaune	Labaleine	Paris
ombrelle	5	rouge	Labaleine	Paris
ceinture	25	vert	Letour	Lyon



Exemple

- L'information « le fournisseur Labaleine est sur Paris » apparaît autant de fois que l'on a commandé de produit à ce fournisseur (redondance).
- Si « Labaleine » change d'adresse, il faudra modifier autant de tuples qu'il y a de commandes (mise à jour)
- Si l'on supprime le produit « ceinture » on supprime toutes les infos relatives au fournisseur « Letour » (suppression).
- Pour insérer un nouveau fournisseur, il faut lui avoir commandé un produit (ajout).

Liens entre les attributs ?

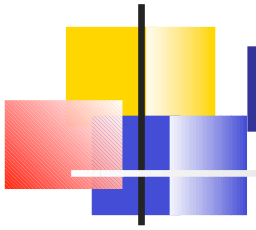
- Entre «fournisseur» et «adresse» ?

  Si on connait le fournisseur, on connaît l'adresse correspondante.

- Entre « produit » et « fournisseur » ?

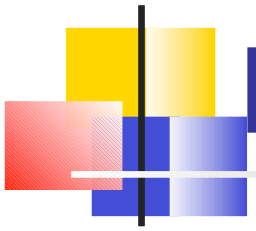
  Si on connaît le produit, on ne connaît pas forcément le fournisseur.

- Dépendance fonctionnelle entre fournisseur et adresse.



Notations et conventions

- U : ensemble d'attributs ;
- R(U) : schéma de relation ;
- Par abus de langage, R parfois appelé « relation ».
- r : un ensemble de tuples particulier (instance du schéma R(U))
- X, Y, Z, W : sous-ens d'attributs de U, non vides.
- A, B, C : attributs particuliers.



Dépendance fonctionnelle (df)

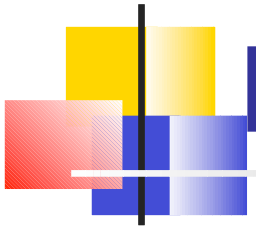
Soit R(U) un schéma de relation ;
soient X et Y deux sous-ens de U.

La dépendance fonctionnelle
X->Y est vraie sur R ssi
quelle que soit r instance de R,
X->Y est vrai dans r.

Dépendance fonctionnelle

- On dit aussi X *détermine* Y, Y *dépend fonctionnellement* de X.
- Une df est une propriété qui doit être extraite de la connaissance que l'on a de l'application.
- L'ensemble des dépendances fonctionnelles vérifiées constitue des informations à rajouter au schéma pour affiner la caractérisation.

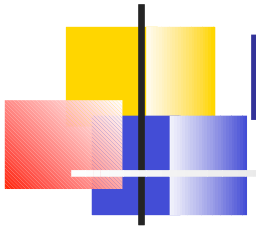
  R(U), DF



Dépendance élémentaire (dfe)

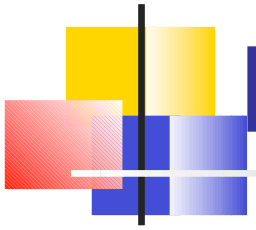
Soit R(U), DF et soit la dépendance fonctionnelle X->A appartenant à DF, avec A non inclus dans X (i.e. pas trivial).

X->A est *élémentaire*, s'il n'existe pas de sous-ensemble de X qui détermine A. (On dit que A dépend pleinement de X)



Propriétés des dfs

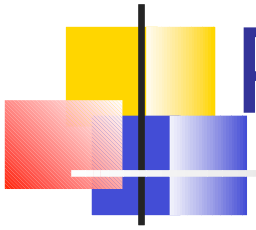
- Quelques règles
- Split
  - Si A->BC alors A->B et A->C
- Combination
  - Si A->B et A->C alors A->BC
- Trivial
  - Si X->Y et Y est sous-ens de X alors X->XUY



Propriétés des dfs

Règles de dérivation d'Armstrong

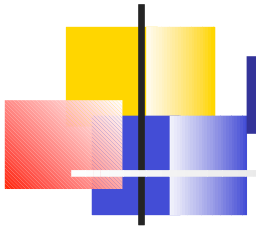
- Réflexivité
  - Si X est inclus dans Y alors Y ->X
- Transitivité
  - Si X ->Y et Y -> Z alors X-> Z
- Augmentation
  - Si X-> Y alors X,Z-> Y,Z



Propriétés des dfs

Règles déduites (exemple)

- Union
  - Si X ->Y et X -> Z alors X->Y,Z
- Pseudo-transitivité
  - Si X ->Y et W,Y -> Z alors W,X -> Z

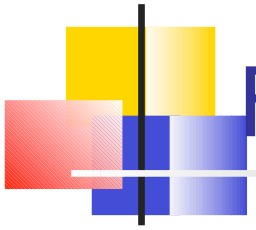


Fermeture d'un ensemble de df

- Soit DF un ensemble de df.

La fermeture (ou clôture) DF^+ de DF est l'ensemble de toutes les df déductibles à partir de DF avec les règles de dérivation.

- La fermeture comprend toutes les dfs déductibles, y compris les plus triviales.



Fermeture d'un ensemble d'attributs

- Etant donné un ensemble d'attributs X, quels sont tous les autres attributs qui sont déterminés par X?
- Soit R(U), DF ; soit X inclus dans U,

X^+ est l'ensemble des attributs A tel que X -> A est déductible de DF.

(équiv. à X ->A appartient à DF^+)

Algorithme de calcul de X^+

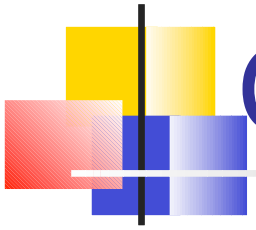
X^+ <- X

Répéter

    pour chaque df Y -> Z de DF faire

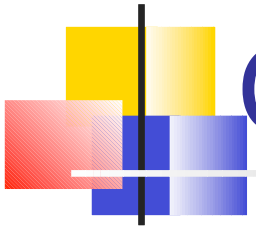
        si Y inclus dans X^+, alors X^+ <- X^+ U Z

Jusqu'à ce que X^+ ne soit plus modifiable



Clés

- Ensemble minimal d'attributs qui détermine tous les autres.
- X est une clé de R(U), DF ssi :
  - X -> U est déductible de DF ($X^+ = U$)
  - Il n'existe pas Y, Y strictement inclus dans X tel que Y ->U (X-> U dfe)
- Il peut y avoir plusieurs clés. Un attribut qui appartient à une clé est parfois appelé « attribut clé ».



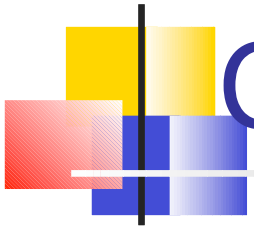
Couverture minimale de DF

Sous ensemble minimal de DF permettant de générer toutes les autres.

Soit DF un ensemble de df. Une couverture minimale de DF, noté CV(DF) est un ensemble de df tel que :

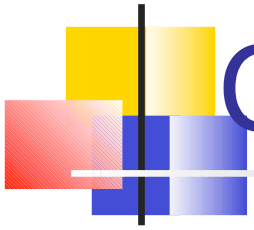
- CV(DF)+ = DF+
- Toutes les df de CV(DF) sont élémentaires
- Quelle que soit X-> A de CV(DF),

(CV(DF) – {X->A})$^{+}$ ≠ DF^{+}



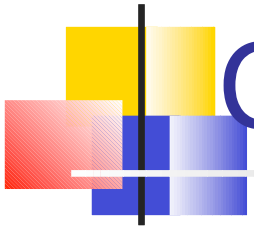
Calcul de CV(DF)

1) Toutes les dépendances doivent être élémentaires ; les décomposer si nécessaire.
2) Eliminer les attributs superflus du coté gauche de la df
3) Eliminer les dfs redondantes



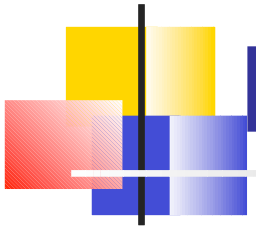
Comment faire le pas 2

- Prendre une df avec coté gauche > 1 attributs (supposons 2 ici)
- Choisir un attribut et calculer sa fermeture :
  - on peut éliminer l'autre attribut si
    - l'attribut de droite de la df apparaît dans le résultat
    - il apparaît dans le résultat de la fermeture.



Comment faire le pas 3

- On peut éliminer une df si par transitivité elle est préservée (donc elle n'est pas nécessaire)
- Ensuite, analyser DF, une df à la fois
  - Calculer la fermeture d'un attribut du coté gauche sans prendre en compte la df en cours d'analyse
    - Si dans le résultat apparaît l'attribut de droite alors la df est éliminée (car elle est préservée même si la df en analyse n'existe pas)

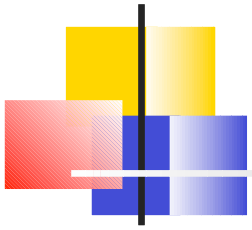


Formes normales 1 et 2

- 1FN : tous les attributs sont atomiques
  - Il ne peut y avoir de listes de valeurs

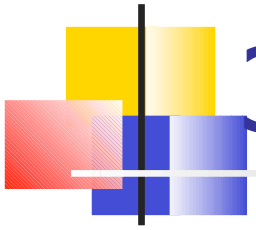
- 2FN : 1FN + tous les attributs non clés ne dépendent pas d'une partie d'une clé

  => les attributs non clé dépendent pleinement d'une clé.



2FN

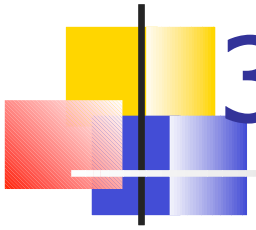
- La 2FN permet d'assurer l'élimination de certaines redondances en garantissant qu'aucun attribut n'est déterminé seulement par une partie de la clé.



3e forme normale (v2)

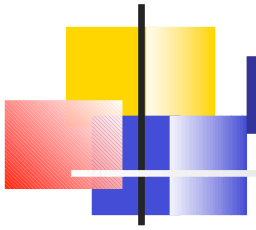
2FN+ tout attribut non clé ne dépend pas d'un autre attribut non clé

=> les attributs non clés ne dépendent pas entre eux.



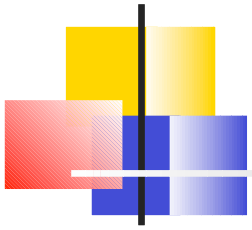
3FN n'est pas suffisant

- Quid des dépendances entre parties de clés entre elles et entre attributs non clés et parties de clés



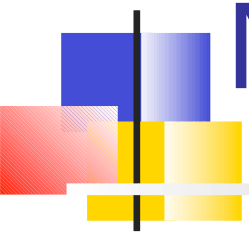
Boyce-Cood-Kent (FNBCK ou BCNF)

- R(U), DF est en FNBCK si chaque fois que la df X-> A (A n'appartenant pas à X) est vérifiée, X est une clé de R.
- Remarque :
  - FNBCK -> 3FN ; 3FN -> 2FN ; 2FN -> 1FN



Quoi faire si une relation n'est pas normalisée ?

Découper la relation en plusieurs relations à l'aide des algorithmes de normalisation.

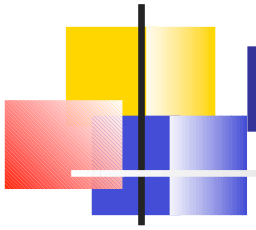


Normalisation des relations

PARTIE II

Algorithmes de Normalisation

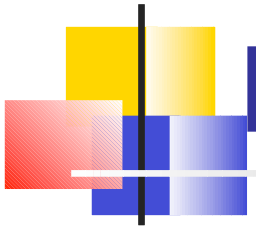
rédaction : sylvie.cazalens@univ-nantes.fr



Normalisation ?

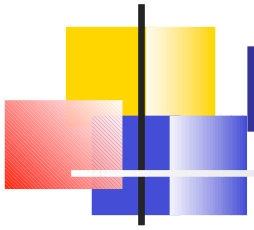
- Une constatation : une relation qui est en forme normale (3FN, FNBCK) présente moins de problèmes d'ajout, de redondance, de suppression, de mise à jour...
- Que faire si une relation ne vérifie pas une forme normale désirée ?

Normaliser



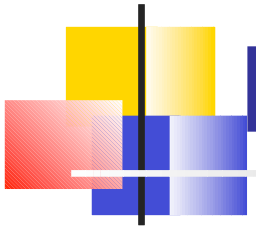
Normalisation d'une relation

La normalisation dans une forme normale «*nf* » d'une relation R est le remplacement de la relation par plusieurs relations qui sont dans la forme normale «*nf* » et qui, « globalement » conservent exactement les informations contenues dans R.



Plan

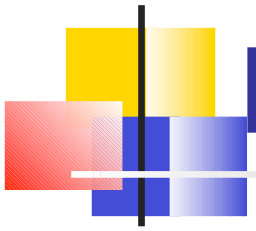
- Décomposition d'une relation
- Décomposition sans perte de df (spdf)
- Décomposition sans perte d'information (spi)
- Algorithmes de normalisation
  - Par synthèse
  - Par décomposition



Restriction d'un ens. de df

Soit le schéma de relation <R(U), DF> et soit X un ensemble d'attributs de U

La restriction de DF à l'ensemble X, notée $DF_{|X}$ est constituée de l'ensemble des dfs de DF^{+} formées d'attributs de X uniquement.

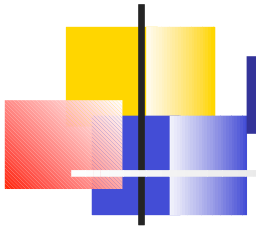


Décomposition

Une décomposition de <R(U), DF> est le remplacement de <R(U), DF> par un ensemble de schémas

$<R_1(U_1), DF_{|U1}>, \ldots, <R_n(U_n), DF_{|Un}>$ où

- pour tout i, U_i est sous-ensemble de U.
- $R(U_i)$ obtenu par projection de R(U) sur U_i
- l'union des U_i est égale à U.



Décomposition sans perte de dfs

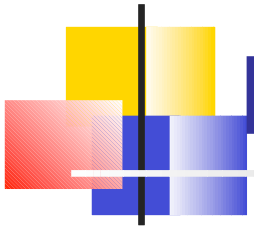
Définition :

La décomposition de <R(U), DF> en

$<R_1(U_1), DF_{|U1}>, \ldots, <R_n(U_n), DF_{|Un}>$

est sans perte de dépendances fonctionnelles (spdf) ssi

la fermeture de l'union des $DF_{|Ui}$ $(1<=i<=n)$ est égale à DF^+



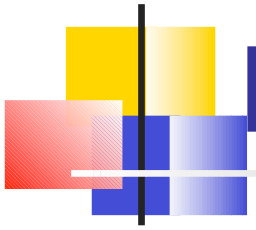
Décomposition sans perte d'info

Définition :

La décomposition de <R(U), DF> en

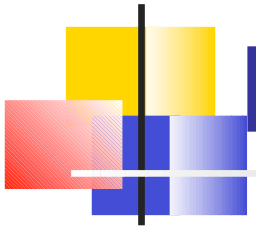
$<R_1(U_1), DF_{|U1}>, \ldots, <R_n(U_n), DF_{|Un}>$

est sans perte d'information (spi) ssi quelle que soit l'instance r de R, r est égale à la jointure naturelle des r_i ($1 <= i <= n$), chaque r_i étant obtenu par projection de r sur R_i.



Liens spdf et spi

- Il n'y a aucun lien entre les propriétés « sans perte d'information » et « sans perte de dépendance ».
- Une décomposition spi n'est pas forcément spdf
- et vice-versa.



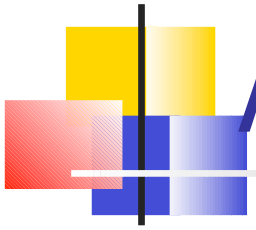
Normalisation : résultats

- Théorème 1

  Toute relation en 1FN admet une décomposition en 3FN qui est à la fois spi et spdf.

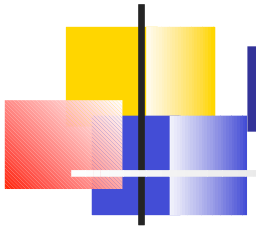
- Théorème 2

  Toute relation en 1FN admet une décomposition en FNBCK qui est spi.



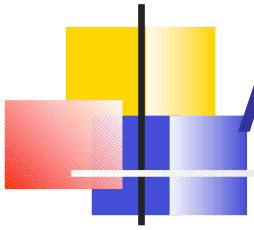
Algorithmes de normalisation

- Objectif : Obtenir de manière automatique un ensemble de relations vérifiant la forme normale souhaitée pour remplacer une relation qui ne la vérifie pas.
- Deux types d'algorithmes :
  - Normalisation « par synthèse »:3FN
  - Normalisation « par décomposition » : BCNF



Normalisation par synthèse

- On synthétise des schémas de relations à partir de l'ensemble des attributs et des dépendances fonctionnelles.
- Algorithme de Bernstein est le plus utilisé.

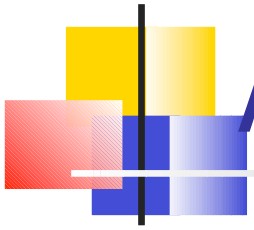


Algorithme de Bernstein

Entrée : <R(U), DF>

Sortie : une décomposition

$<R_1(U_1), DF_{|U1}>, \ldots, <R_n(U_n), DF_{|Un}>$, à la fois spi et spdf où toutes les relations sont en 3FN.



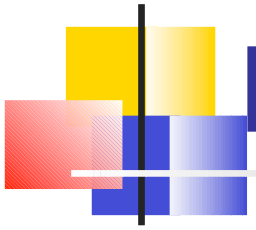
Algorithme de Bernstein

(1) Calculer CV(DF), et les clés. Si R est en 3FN on s'arrête.

(2) Partitionner CV(DF) en groupes DFi (1<=i<=k) tels que toutes les df d'un même groupe aient même partie gauche.

(3) Construire un schéma <Ri(Ui), DFi> pour chaque groupe DFi, où Ui est l'ensemble des attributs apparaissant dans DFi.

(4) Si aucun des schémas définis ne contient de clé X de R, rajouter un schéma <Rk+1(X), {}>.

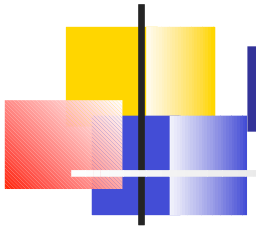


Remarque

Certains schémas définis doivent être regroupés, lorsque l'ensemble d'attributs de l'un est inclus dans l'ensemble d'attributs de l'autre.

R(A, B, C, D)

DF = { A, B -> C,
C -> A,
B -> D}



Exemple

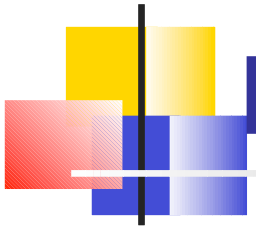
R(C, E, P, N, J, H, S)

DF = { C, E, P -> N

J, H, S -> P, C

E, P -> C

E, P, S -> C }



Normalisation par décomposition

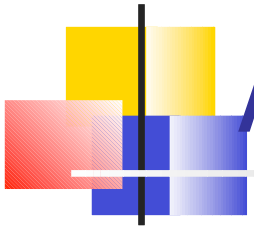
- On part du schéma initial et on le remplace par 2 autres schémas, contenant moins d'attributs et ainsi de suite jusqu'à n'avoir que des schémas dans la forme normale désirée.
- Formation d'un arbre de décomposition.
- Le principe peut être utilisé lorsqu'on a à la fois des dépendances multi-valuées et des dépendances fonctionnelles.

Algorithme decomp()

- Entrée : <R(U), DF>
- Sortie : une décomposition

  $<R_1(U_1), DF_{|U1}>, \ldots, <R_n(U_n), DF_{|Un}>$, sans perte d'information où tous les schémas sont en FNBCK

- On crée un arbre de décomposition



Algorithme decomp()

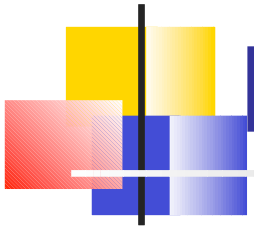
(1) Choisir une dépendance non triviale de DF X -> Y, X ne contenant pas de clé.

(2) Décomposer en R1(X,Y) ; R2= U- (Y-X)

(3) Pour i = 1, 2 faire

- Si <Ri, DF|Ri> est en FNBCK alors il devient feuille ;
- Sinon decomp(<Ri, $DF_{|Ri}$>)

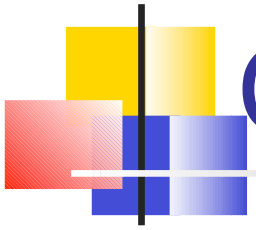
(1) Renvoyer l'ensemble de toutes les feuilles.



Remarque

- Les algorithmes par décomposition n'imposent pas de travailler avec la couverture minimale. D'où par exemple la précaution R2 = U – (Y-X)

- Dans le cas d'une couverture minimale, on mettrait juste R2 = U – Y.



Conclusion

- Il faut faire un compromis entre performance et absence de problèmes.
- Exemple : R(A, B, C, D)

  DF = {A,B -> C ; A,B -> D ; C -> A}

  est 3FN mais pas FNBCK. Une décomposition en FNBCK est-elle souhaitable sachant qu'elle génère 2 relations (et qu'il faudra faire des jointures pour retrouver certaines informations ?)