Bases de Données

Amélie Gheerbrant

Université Paris Diderot
UFR Informatique
Laboratoire d'Informatique Algorithmique : Fondements et Applications

amelie@liafa.univ-paris-diderot.fr

9 décembre 2014

Contraintes et conception de schéma

- On sait que les bases de données doivent typiquement satisfaire certaines contraintes d'intégrité
- Les plus communes sont les dépendance fonctionnelles et d'inclusion
- Nous allons étudier certaines propriétés de ces contraintes d'intégrité
- Nous allons voir comment elles influencent la conception de BD, en particulier, comment elles nous permettent de déterminer quel type d'information devrait être stocké dans quelle relation

Les dépendances fonctionnelles et les clefs

- Une dépendance fonctionnelle est de la forme $X \rightarrow Y$ où X, Y sont des séquences d'attributs. Elle vaut dans une relation R si pour tous tuples t_1, t_2 de R :

$$\pi_X(t_1) = \pi_X(t_2) \text{ implique } \pi_Y(t_1) = \pi_Y(t_2)$$

- Un cas spécial très important : les clefs
- Soit K un ensemble d'attributs de R, et U l'ensemble de tous les attributs de R. Alors K est une clef si R satisfait la dépendance fonctionnelle $K \rightarrow U$.
- i.e., un ensemble d'attributs K est une clef de R si pour tous tuples t_1, t_2 de R,

$$\pi_K(t_1) = \pi_K(t_2) \text{ implique } t_1 = t_2$$

- Une clef est donc un ensemble d'attributs qui définit de manière unique un tuple dans une relation.

Problèmes

- Bonnes vs Mauvaises contraintes : certaines contraintes peuvent être indésirables car susceptibles de mener à des problèmes.
- Problème de l'implication : supposons qu'on nous donne un ensemble de contraintes, est-ce qu'elles en impliquent d'autres ?
  Important, car on n'a jamais la liste de **toutes** les contraintes qui valent dans une BD (trop grande, ou tout simplement inconnue du concepteur)
  Toutes les contraintes connues peuvent avoir l'air ok, mais impliquer de mauvaises contraintes
- Axiomatisation de contraintes : moyen simple de formuler l'implication de contraintes.

Mauvaises contraintes : exemple

DRE	Dept	Res	Empl
	D1	Smith	Jones
	D1	Smith	Brown
	D2	Turner	White
	D3	Smith	Taylor
	D4	Smith	Clarke
	...	...	...

ES	Empl	Salaire
	Jones	10
	Brown	20
	White	20
	...	...

On observe les dépendances fonctionnelles suivantes :

- dans DME : Departement $\rightarrow$ Responsable, mais pas :
  - Dept $\rightarrow$ Empl
  - Resp $\rightarrow$ Dept
  - Resp $\rightarrow$ Empl
- dans ES : Empl $\rightarrow$ Salaire (Employe est une clef de ES), mais pas Salaire $\rightarrow$ Empl

Anomalies de mise à jour

- **Anomalies d'insertion** : une entreprise engage un nouvel employé, mais ne l'assigne pas tout de suite à un département. On ne peut pas enregistrer ce fait dans la relation DME.
- **Anomalies de suppression** : l'employé White quitte l'entreprise. Il nous faut donc supprimer un tuple de DME. Mais ceci entraine aussi la suppression du responsable Turner, alors qu'il n'est pas parti !
- **Anomalies de modification** : imaginons que Smith soit remplacé à la tête de D1 par Turner. Il faut alors modifier tous les triplets de DRE correspondant.

Anomalies de mise à jour

- **Raisons** : association entre employés et responsables dans la même relation que association entre responsables et départements + le même fait (département D dirigé par R) peut être représenté plus d'une fois.
- Dans le langage des dépendances fonctionnelles :

$$Dept \rightarrow Resp, \text{ mais } Dept \text{ n'est pas une clef}$$

- En général on essaie d'éviter ce genre de situations.

Dépendances et implications

- Soit une relation R avec trois attributs A, B, C
- On nous dit que R satisfait A $\rightarrow$ B et B $\rightarrow$ {A,C}
- Ressemble au mauvais exemple de tout à l'heure : on a A $\rightarrow$ B sans mention de C
- Mais en fait, A est une clef :

$$\text{si } \pi_A(t_1) = \pi_A(t_2), \text{ alors } \pi_B(t_1) = \pi_B(t_2)$$

  et comme B est une clef, $t_1 = t_2$
- Donc, même sans savoir qu'A était une clef, nous avons pu le dériver, car les autres dépendances l'impliquaient.
- En utilisant l'implication, on peut trouver toutes les dépendances et déterminer s'il y a un problème de conception.

L'implication pour les dépendances fonctionnelles

- On essaie seulement de trouver les dépendances non triviales
  Une dépendance triviale c'est $X \rightarrow Y$ avec $Y \subseteq X$
- Supposons donné l'ensemble U des attributs d'une relation, un ensemble F de dépendances fonctionnelles (DFs), et une DF f sur U. Alors F implique f, noté

$$F \vdash f$$

  ssi pour toute relation R sur les attributs U, si R satisfait toutes les FDs de F, alors elle satisfait aussi f.
- Problème de l'implication : Etant donné U et F, trouver toutes les DFs non triviales telles que $F \vdash f$.

L'implication pour les dépendances fonctionnelles

- Ensemble clos par rapport à F : sous ensemble V de U t.q. pour tout X $\rightarrow$ Y dans F,

$$\text{si } X \subseteq V, \text{ alors } Y \subseteq V$$

- Propriétés des FDs : pour tout ensemble V, il existe un unique ensemble $C_F(V)$, appelé clôture de V par rapport à F, tel que
  - $C_F(V)$ est clos
  - $V \subseteq C_F(V)$
  - Pour tout ensemble clos W : $V \subseteq W$ implique $C_F(V) \subseteq W$.
- Solution du problème de l'implication :

$$\text{Une FD X} \rightarrow \text{Y est impliquée par F}$$
$$\text{si et seulement si}$$
$$Y \subseteq C_F(X)$$

Implication et clôture

- Pour résoudre le problème de l'implication, il suffit de trouver la clôture de chaque ensemble d'attributs.
- Approche naïve pour trouver $C_F(X)$: vérifier tous les sous ensembles $V \subseteq U$, vérifier qu'ils sont clos, et sélectionner le plus petit ensemble clos contenant X.
- Problème : c'est trop coûteux.
- Si U a n elements et X a $m < n$ éléments, alors il y'a 2^{n-m} sous ensembles de U qui contiennent V.
- En fait on va considérer un algorithme beaucoup plus efficace.

Implication et clôture

Algorithme CLOTURE

Entrée : un ensemble F de FDs, et un ensemble X d'attributs
Sortie : $C_F(X)$

1. *inutilisé* := F
2. *clôture* := X
3. répéter jusqu'à ce que plus de changement :
   si $Y \rightarrow Z \in$ *inutilisé* et $Y \subseteq$ *clôture* alors
   (i) *inutilisé* := *inutilisé* - $\{Y \rightarrow Z\}$
   (ii) *clôture* := *clôture* $\cup Z$
4. Sortie : *clôture*.

Clôture : exemple

- Pratique commune : écrire AB pour l'ensemble $\{A, B\}$, BC pour $\{B, C\}$ etc
- $U = \{A, B, C\}$, $F = \{A \rightarrow B, B \rightarrow AC\}$
- Clôture :
  $C_F(\emptyset) = \emptyset$
  $C_F(C) = C$
  $C_F(B) = ABC$
  d'où $C_F(X) = ABC$ pour tout X qui contient B
  $C_F(A) = ABC$
  d'où $C_F(X) = ABC$ pour tout X qui contient A

Clôture : exemple pratique

Soit $U = \{NomFournisseur, Adresse, Produit, Prix, FraisTransport\}$ et $F = \{F_1, F_2, F_3\}$, avec :

- $F_1 = NomFournisseur \rightarrow Adresse$
- $F_2 = NomFournisseur, Produit \rightarrow Prix$
- $F_3 = Adresse, Produit \rightarrow FraisTransport$

Calcul de $C_F(NomFournisseur, Produit)$:

1. inutilisé=$\{F_1, F_2, F_3\}$, clôture=$\{NomFournisseur, Produit\}$
2. inutilisé=$\{F_3\}$, clôture=$\{NomFournisseur, Produit, Adresse, Prix\}$
3. inutilisé=$\emptyset$,
   clôture=$\{NomFournisseur, Produit, Adresse, Prix, FraisTransport\}$
4. Fin (plus de DF à appliquer) :
   $C_F(NomFournisseur, Produit)$=clôture

On en déduit que :

$F \vdash NomFournisseur, Produit \rightarrow Adresse, Prix, FraisTransport$

Propriétés de la clôture

- $X \subseteq C_F(X)$
- $X \subseteq Y$ implique $C_F(X) \subseteq C_F(Y)$
- $C_F(C_F(X)) = C_F(X)$

On peut utiliser l'algorithme de clôture pour déterminer si un ensemble de DFs F implique une DF f :

$$F \vdash Y \rightarrow Z \Leftrightarrow Z \in C_F(Y)$$

ou pour déterminer étant donné F, la clef d'un schéma de relation R, on supprime tous les attributs de R jusqu'à arriver à un ensemble $K \subseteq R$ t.q. :

- $C_F(K) = R$
- $\forall A \in K, C_F(K - \{A\}) \neq R$ (i.e., K est minimal)

Clefs, clefs candidates et attributs primaires

- Un ensemble X d'attributs est une clef pour F si $X \rightarrow U$ est impliqué par F (i.e., si X détermine tous les attributs)
- C'est-à-dire, X est une clef si $C_F(X) = U$
- Dans le premier exemple, tout ensemble qui contient A ou B est une clef.
- Clefs candidates : plus petites clefs, i.e., clefs X t.q. pour tout $Y \subseteq X$, Y n'est pas une clef.
- Dans l'exemple précédent : A et B
- Attribut primaire : attribut d'une clef candidate.

Attention : on parle parfois de superclefs (au lieu de clefs) versus clefs (au lieu de clef candidates). Il y a un flottement terminologique dans la littérature...

Dépendances d'inclusion

- Les dépendances fonctionnelles et d'inclusion sont les plus fréquentes dans les applications.
- Rappel : $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$ veut dire que pour tout tuple t dans R, il y a un tuple t' dans S t.q.

$$\pi_{A_1,\ldots,A_n}(t) = \pi_{B_1,\ldots,B_n}(t')$$

- Supposons que l'on ait un ensemble de noms de relations et de dépendances d'inclusion (DIs) sur eux. Peut-on dériver toutes les DIs qui sont valides ?
- Formellement, soit G un ensemble de DIs et g une DI. On dit que G implique g, noté

$$G \vdash g$$

si tout ensemble de relations qui satisfait toutes les IDs dans G satisfait aussi g.

- La réponse est positive, comme pour les FDs.

Implication et dépendances d'inclusion

Il existe un ensemble de règles simples :

1. $R[A] \subseteq R[A]$
2. $$\text{si } R[A_1, \ldots, A_n] \subseteq R[B_1, \ldots, B_n]$$
   $$\text{alors } R[A_{i_1}, \ldots, A_{i_n}] \subseteq R[B_{i_1}, \ldots, B_{i_n}]$$
   où $\{i_1, \ldots, i_n\}$ est une permutation de $\{1, \ldots, n\}$
3. $$\text{si } R[A_1, \ldots, A_m, A_{m+1}, \ldots, A_n] \subseteq R[B_1, \ldots, B_m, B_{m+1}, \ldots, B_n]$$
   $$\text{alors } R[A_1, \ldots, A_m] \subseteq R[B_1, \ldots, B_m]$$
4. si $R[X] \subseteq S[Y]$ et $S[Y] \subseteq T[Z]$ alors $R[X] \subseteq T[Z]$

(Attention, voir transparent précédent, ici les attributs sont ordonnés !)

Implication et dépendances d'inclusion

- Une DI g est impliquée par G ssi elle peut être dérivée par des applications répétées des quatre règles précédentes
- Donne immédiatement un algorithme coûteux : appliquer toutes les règles jusqu'à ce qu'il n'y en ait plus d'applicables
- Et en fait, le problème est *vraiment* dur : on ne peut pas trouver d'algorithme raisonnable pour le résoudre

Une restriction du problème :

- DIs unaires : de la forme $R[X] \subseteq S[Y]$
- L'implication $G \vdash g$ peut être testée en temps polynomial pour les DIs unaires.

Dépendances fonctionnelles et d'inclusion

- Pour la majorité des applications on a besoin de considérer seulement deux types de dépendances : DFs et DIs
- Le problème de l'implication peut être résolu pour les DFs et les DIs
- Est-ce qu'il peut être résolu pour les DFs et les DIs ensemble ?
- I.e., est-ce que, étant donné un ensemble de DFs F et un ensemble de DIs G, une DF est une DI g, on peut déterminer si

$$F \cup G \vdash f$$

$$F \cup G \vdash g$$

  i.e., si toute BD qui satisfait F et G doit aussi satisfaire f (ou g)
- En fait, aucun algorithme ne peut résoudre ce problème

Encore plus de restrictions

- Les relations sont typiquement déclarées avec juste des clefs primaires et les dépendances d'inclusion occurrent dans les déclaration de clef étrangères.
- Est-ce que l'implication peut être décidée algorithmiquement juste pour les clefs primaires et les clefs étrangères ?
- La réponse est toujours NON
- Il est donc intrinsèquement dur de raisonner à propos des classes de contraintes les plus basiques en BD relationnelles.

Quand l'implication peut-elle être résolue pour les DFs et les DIs ?

- Rappel : une DI unaire est de la forme $R[X] \subseteq S[Y]$
- On peut tester l'implication pour les DIs unaires et les DFs arbitraires
- De plus, on peut le faire en temps polynomial
- Cependant, l'algorithme en question est plutôt complexe.

Dépendances : résumé

- **DFs** : $X \to Y$
- **Clefs** : $X \to U$, où U est l'ensemble de tous les attributs de la relation.
- **Clef candidate** : clef minimale
  (tout sous ensemble de X est une clef)
- **Dépendance d'inclusion** : $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$
  (unaire si n=1)
- **Clef étrangère** : $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$ et $\{B_1, \ldots, B_n\}$ est une clef
- **Problème de l'implication** :
  facile pour les DFs seules
  difficile mais décidable pour les DIs seules
  décidable par un algorithme complexe pour DFs + DIs unaires
  indécidable pour DFs + DIs, et même
  indécidable pour les clefs primaires et les clefs étrangères

La conception de bases de données

- Trouver des schémas de bases de données avec de bonnes propriétés
- Bonnes propriétés :
  - pas d'anomalie de mise à jour
  - pas de redondance
  - pas de perte d'information
- Entrée : liste d'attributs et de contraintes (d'habitude, des dépendances fonctionnelles)
- Sortie : liste de relations et de contraintes

Exemple : mauvaise conception

- Attributs :
  titre, réalisateur, cinéma, adresse, téléphone, heure, prix
- Contraintes :
  - FD1 cinéma $\rightarrow$ adresse, téléphone
  - FD2 cinéma, heure, titre $\rightarrow$ prix
  - FD3 titre $\rightarrow$ réalisateur
- Mauvaise conception : tout mettre dans la même relation
  MAUVAIS[titre, réalisateur, cinéma, adresse, téléphone, heure, prix]

Pourquoi Mauvais est-elle mauvaise ?

- **Redondance** : beaucoup de faits sont répétés
  - réalisateur est déterminé par titre

    Pour toute séance, on liste à la fois le réalisateur et le titre.
  - adresse est déterminé par cinéma

    Pour toute film joué, on répète l'adresse.
- **Anomalies de mise à jour** :
- si adresse change dans un tuple, on obtient de l'incohérence, car il faut la changer dans tous les tuples correspondant à chaque film et séance
- si un film cesse d'être joué, on perd l'association entre titre et réalisateur
- on ne peut pas ajouter un film avant qu'il commence à être joué

Bonne conception

Séparer Mauvais en 3 relations

- Schéma relationnel Bon :

Table	attributs	contraintes
T1	cinéma, adresse, téléphone	FD1 : cinéma $\rightarrow$ adresse, téléphone
T2	cinéma, titre, heure, prix	FD2 : cinéma, heure, titre $\rightarrow$ prix
T3	titre, réalisateur	FD3 : titre $\rightarrow$ réalisateur

Pourquoi Bon est-il bon ?

- Pas d'anomalie de mise à jour : toute FD est une clef
- Pas de perte d'information :
  - $T1 = \pi_{cinema,adresse,telephone}(Mauvais)$
  - $T2 = \pi_{cinema,titre,heure,prix}(Mauvais)$
  - $T3 = \pi_{titre,realisateur}(Mauvais)$
  - $Mauvais = T1 \bowtie T2 \bowtie T3$
- Aucune contrainte n'est perdue
  FD1, FD2, FD3 apparaissent toutes comme contraintes de T1, T2, T3

Impact sur les requêtes (bémol) : le temps d'exécution peut maintenant augmenter lorsqu'il y a des jointures à faire.

Forme normale de Boyce-Codd (FNBC)

- Qu'est-ce qui cause les anomalies de mise à jour ?
- Les dépendances fonctionnelles $X \to Y$ où X n'est pas une clef
- Une relation est en forme normale de Boyce-Codd (FNBC) si pour toute DF non triviale $X \to Y$, X est une clef
- Une base de données est en FNBC si toutes ses relations sont en FNBC

Décompositions : critères de bonnes conception

Etant donné un ensemble d'attributs U et un ensemble de dépendances fonctionnelles, une décomposition de (U, F) est un ensemble

$$(U_1, F_1), \ldots, (U_n, F_n)$$

où $U_i \subseteq U$ et F_i est un ensemble de FDs sur les attributs U_i

Une décomposition est FNBC si chaque (U_i, F_i) est en FNBC

Décompositions : critères de bonnes conception

- **Sans perte d'information (SPI)** si pour toute relation R sur U qui satisfait toutes les DFs dans F,
  tout $\pi_{U_i}(R)$ satisfait F_i et

$$R = \pi_{U_1}(R) \bowtie \pi_{U_2}(R) \bowtie \ldots \bowtie \pi_{U_n}(R)$$

- **Sans perte de dépendances (SPD)** si

$$F \text{ et } F^* = \bigcup_i F_i \text{ sont équivalents}$$

  i.e., pour toute FD f,

$$F \vdash f \Leftrightarrow F^* \vdash f$$

  ou

$$\forall X \subseteq U, \quad C_F(X) = C_{F^*}(X)$$

Projection des DFs

- Soit U un ensemble d'attributs et F un ensemble de DFs
- Soit $V \subseteq U$
- Alors on définit

$$\pi_V(F) = \{X \to Y \mid X, Y \subseteq V,\ Y \subseteq C_F(X)\}$$

- ce sont toutes les DFs sur V qui sont impliquées par F :

$$\pi_V(F) = \{X \to Y \mid X, Y \subseteq V,\ F \vdash X \to Y\}$$

- Même si tel qu'on l'a définit $\pi_V(F)$ pourrait être très grand, on peut souvent le représenter de manière compacte par un ensemble F' de DFs sur V t.q. :

$$\forall X, Y \subseteq V : \quad F' \vdash X \to Y \Leftrightarrow F \vdash X \to Y$$

Projection des DFs : exemple

Attention, on peut décomposer SPD $U = \{A, B, C, D, \}$ et $F = \{F_1 = A \to B, F_2 = B \to C, F_3 = C \to D, F_4 = D \to A\}$ comme suit :

- $S_1 = (\{AB\}; F_1)$
- $S_2 = (\{BC\}; F_2)$
- $S_3 = (\{CD\}; F_3)$

En fait nous n'avons pas perdu F_4, la décomposition et bien SPD, car :

- $F \vdash B \to A, D \to C, C \to B,$
- $B \to A \in \pi_{AB}(F)$, $C \to B \in \pi_{BC}(F)$, $D \to C \in \pi_{CD}(F)$
- $B \to A, D \to C, C \to B \vdash D \to A$

Algorithme de décomposition

Entrée : un ensemble d'attributs U et un ensemble de DFs F
Sortie : un schéma de BD $S = \{(U_1, F_1), \ldots, (U_n, F_n)\}$

1. $S := \{(U, F)\}$
2. Tant que S n'est pas en FNBC faire :
   (a) Choisir $(V, F') \in S$ pas en FNBC
   (b) Choisir X, Y, Z disjoints non vides t.q. :
      (i) $X \cup Y \cup Z = V$
      (ii) $Y = C_{F'}(X) - X$
      (i.e., $F' \vdash X \to Y$ et $F' \nvdash X \to A$ pour tout $A \in Z$)
   (c) Remplacer (V, F') par

$$(X \cup Y, \pi_{X \cup Y}(F')) \text{ et } (X \cup Z, \pi_{X \cup Z}(F'))$$

   (d) S'il existe (V', F') et (V'', F'') dans S t.q. $V' \subseteq V''$ alors retirer (V', F')

Algorithme de décomposition : exemple

Considérons l'ensemble d'attributs et les contraintes suivants :
$Mauvais = \{titre, realisateur, cinema, adresse, telephone, heure, prix\}$
$F = \{FD1, FD2, FD3\}$

Initialisation : $S = (Mauvais, FD1, FD2, FD3)$

Boucle While :

- Etape 1 : on sélectionne Mauvais ; qui n'est pas en FNBC car
  cinema $\rightarrow$ adresse, telephone, mais
  cinema $\not\rightarrow$ titre, réalisateur, heure, prix

  Nos ensembles sont :
  $X = \{cinema\}$, $Y = \{adresse, telephone\}$,
  $Z = \{titre, realisateur, heure, prix\}$

Algorithme de décomposition : exemple

- Etape 1, suite :
  $\pi_{X \cup Y}(\{FD1, FD2, FD3\}) = FD1$
  $\pi_{X \cup Z}(\{FD1, FD2, FD3\}) = \{FD2, FD3\}$

Après l'étape 1, nous avons obtenu deux schémas :

$S_1 = (\{cinema, adresse, telephone\}, FD1)$,
$S_1' = (\{cinema, titre, ralisateur, heure, prix\}, FD2, FD3)$

- Etape 2 : S_1 est en FNBC, rien à faire donc.
  S_1' n'est en FNBC : titre n'est pas une clef
  Soit $X = \{titre\}$, $Y = \{realisateur\}$,
  $Z = \{cinema, heure, prix\}$
  $\pi_{X \cup Y}(\{FD1, FD2, FD3\}) = FD3$
  $\pi_{X \cup Z}(\{FD1, FD2, FD3\}) = \{FD2\}$

Algorithme de décomposition : exemple

- Après l'étape 2, nous avons :

  $S_1 = (\{cinema, adresse, telephone\}, FD1)$,
  $S_2 = (\{cinema, titre, heure, prix\}, FD2)$
  $S_3 = (\{titre, realisateur\}, FD3)$

- S_1, S_2, S_3 sont toutes en FNBC, fin.

Propriétés de l'algorithme de décomposition

- Pour tout schéma relationnel, l'algorithme de décomposition génère un schéma relationnel qui est
  - en FNBC
  - SPI (sans perte d'information)
- Cependant, la sortie n'est pas garantie SPD (sans perte de dépendance)

La FNBC et la préservation des dépendances

- Schéma $Adresse[(C)odePostal, (V)ille, (R)ue]$
- Ensemble de FDs $F = \{C \to V, VR \to C\}$
- $(Adresse, F)$ n'est pas en FNBC : $C \to V \in F$, mais C n'est pas une clef.
- On applique l'algo de décomposition FNBC : $X = \{C\}$, $Y = \{V\}$, $Z = \{R\}$
- Sortie : $(\{C, V\}, C \to V)$, $(\{C, R\}, \emptyset)$
- On perd $VR \to C$!
- En fait, il n'existe pas de schéma relationnel équivalent à *Adresse* qui soit à la fois en FNBC, SPI et SPD.
- Preuve : il suffit de vérifier tous les schémas sur trois attributs.

3ème forme normale (3FN)

- Si l'on veut une décomposition SPI et SPD, on ne peut pas toujours recourir à la FNBC
- Il nous faut une condition un peu plus faible
- Rappels :
  - une clef candidate est une clef qui n'est incluse dans aucune autre clef
  - Un attribut est primaire s'il appartient à une clef candidate
  - (U, F) est en FNBC si pour toute DF $X \to A$, où $A \notin X$ est un attribut, $F \vdash X \to A$ implique que X est une clef
- (U, F) est en 3FN si pour toute DF $X \to A$ où $A \notin X$ est un attribut, $F \vdash X \to A$ implique que :
  - soit X est une clef,
  - ou bien A est primaire.

3ème forme normale (3FN)

- Différences principales entre FNBC et 3FN : en 3FN les DFs non clefs sont OK, tant qu'elles n'impliquent que des attributs primaires.
- $\{NomFournisseur, Adresse, Produit, Prix\}$;
  $\{NomFournisseur \rightarrow Adresse; NomFournisseur, Produit \rightarrow Prix\}$
  nest pas en 3FN :
  - $NomFournisseur, Produit$ est clef candidate ; mais $Adresse$ n'est pas primaire.
- Mais $Adresse[C, V, R], F = \{C \rightarrow V, VR \rightarrow C\}$ est en 3FN :
  - VR est clef candidate, donc V est un attribut primaire.
- Plus de redondance qu'avec la FNBC : chaque fois qu'un code postal apparaît dans un tuple, le nom de la ville est répété.
- On tolère cette redondance parce qu'il n'existe pas de décomposition FNBC.

Décompositions 3FN : couvertures minimales

- Besoin pour l'algo d'un petit ensemble représentant toutes les DFs d'un ensemble F
- On parle de couverture minimale. Formellement :
  - F' est une couverture de F ssi $C_F = C_{F'}$, i.e., pour tout f :

$$F \vdash f \Leftrightarrow F' \vdash f$$

  - F' est une couverture minimale si
    - F' est une couverture
    - aucun sous ensemble $F'' \subset F'$ n'est une couverture de F,
    - toute FD dans F' est de la forme $X \rightarrow A$, où A est un attribut
    - pour tout $X \rightarrow A \in F'$ t.q. $X' \subset X$, on a $A \notin C_F(X')$

Décompositions 3FN : couvertures minimales

- Une couverture minimale est un petit ensemble de FDs qui nous donne exactement la même information que F.
- Exemple :

$$F = \{A \to AB, A \to AC, A \to B, A \to C, B \to BC\}$$

- Clôture :

$$C_F(A) = ABC,$$
$$C_F(B) = BC,$$
$$C_F(C) = C, \text{ et donc}$$

- Couverture minimale :

$$F' = \{A \to B, B \to C\}$$

Décompositions 3FN : algorithme

Entrée : un ensemble d'attributs U et un ensemble de DFs F
Sortie : un schéma de BD $S = \{(U_1, F_1), \ldots, (U_n, F_n)\}$

Etape 1 Trouver une couverture minimale F' de F

Etape 2 S'il existe $X \to A$ dans F' t.q. $XA = U$, alors sortie $= (U, F')$.
Sinon, choisir une clef K et sortir :
(2a) $(XA, X \to A)$ pour tout $X \to A \in F'$, et
(2b) $(K, \emptyset)$

Décompositions 3FN : exemple

- $F = \{A \to AB, A \to AC, A \to B, A \to C, B \to BC\}$
- $F' = \{A \to B, B \to C\}$
- A est une clef, donc sortie :

$$(A, \emptyset), \quad (AB, A \to B) \quad (BC, B \to C)$$

- Simplification : si un des ensembles d'attributs produits en (2a) est une clef, alors pas besoin de (2b)
- Ce qui donne comme résultat :

$$(AB, A \to B) \quad (BC, B \to C)$$

- Autre type de simplification possible :
  remplacer $(XA_1, X \to A_1), \ldots, (XA_k, X \to A_k)$ dans la sortie, par $(XA_1 \ldots A_k, X \to A_1 \ldots A_k)$

Décompositions 3FN

- L'algorithme de décomposition produit un schéma :
  - en 3FN
  - SPI
  - SPD
- Complexité de l'algorithme ?
- Etant donnée la couverture minimale F', en temps linéaire.
- Mais à quel point est-ce dur de trouver une couverture minimale ?
- Approche naïve : essayer tous les ensembles F', vérifier s'ils sont des couvertures minimales.
- Trop coûteux (exponentiel), peut-on faire mieux ?
- Il y a un algorithme en temps polynomial.

Algorithme de calcul de couverture minimale

Entrée : un ensemble de DFs F
Sortie : une couverture minimale de F

1. Mettre F sous forme canonique, i.e., remplacer toute DF $X \to ABC$ par l'ensemble de DFs $F = \{X \to A, X \to B, X \to C, \ldots\}$
2. Répéter jusqu'à ce qu'il n'y ait plus de changement : s'il existe f t.q. $F \setminus \{f\} \vdash f$, alors remplacer F par $F \setminus \{f\}$
3. Répéter jusqu'à ce qu'il n'y ait plus de changement : s'il existe $f = X \to A \in F$ et $Y \subset X$ t.q. $A \in C_F(Y)$, alors remplacer $X \to A$ par $Y \to A$ dans F.

Conception de schéma : démarche

- Choisir un ensemble d'attributs U
- Choisir l'ensemble de FDs F
- Trouver un décomposition SPI et SPD qui soit :
  - FNBC, s'il en existe une
  - 3FN, s'il n'existe pas de décomposition FNBC