UNIVERSITÉ de la NOUVELLE-CALÉDONIE

Chapitre 2: Définir et structurer les bases de données

Inférence des DF et normalisation

(dérivé du cours du Pr. Jeffrey Ullman, Stanford University
et du cours du Pr. Christian Retoré, Université de Bordeaux)

Université de la Nouvelle-Calédonie
frederic.flouvat@univ-nc.nc

PLAN

- **Inférence des DF**
- Normalisation de bases de données

Rappels sur les Dépendances Fonctionnelles

- La DF X -> Y est satisfaite dans la relation R ssi 2 tuples égaux sur les attributs X sont aussi égaux sur les attributs Y
  - p.ex. name -> addr favBeer
  - Une clé primaire représente un cas particulier de DF

- Intérêt des DF: améliorer le schéma des bases de données en évitant des anomalies
  - Anomalie de mise à jour et anomalie de suppression

- Des DF peuvent être déduites en analysant le problème (p.ex. heure salle -> cours)

➢ Problème: des DF implicites pouvent échappées à cette analyse
  - Problème de l'inférence des DF, i.e. à partir d'un ensemble de DF connues, comment trouver toutes les DF satisfaites par une relation

Déduire des DF

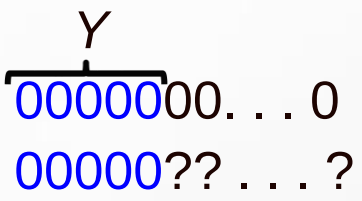
- On se donne F un ensemble de DF: X_1 -> A_1, X_2 -> A_2,…, X_n -> A_n , et on souhaite savoir si une DF *Y* -> *B* est la conséquence sémantique de F
  - c'est-à-dire *Y* -> *B* est satisfaite dans tout modèle satifaisant F.

- Exemple:
  - Si *A* -> *B* et *B* -> *C* sont vraies, sans doute que *A* -> *C* aussi, même si on ne le dit pas.

- Important pour la conception de bons schémas relationnels.

Test d'inférence

- Pour tester si *Y -> B*, commencer par supposer que deux tuples sont égaux sur tous les attributs de Y



- Utiliser l'ensemble de DF de départ pour en déduire que les tuples sont égaux sur d'autres attributs

  - Si B est l'un des attributs pour lesquels il y a égalité alors *Y -> B* est vrai

  - Sinon les 2 tuples, avec les égalités induites par les dépendances, forment un contre-exemple demontrant que *Y -> B* n'est pas une conséquence des DF de départ

Test d'inférence

- Exemple :
  - DF données { C->B , AB->D }
  - AC-> D ?

A	B	C	D	E
a′	b′	c′	d′	e′
a′		c′		

A	B	C	D	E
a′	b′	c′	d′	e′
a′	**b′**	c′		

C->B

A	B	C	D	E
a′	b′	c′	d′	e′
a′	b′	c′	**d′**	

AB->D

Donc AC->D est satisfait

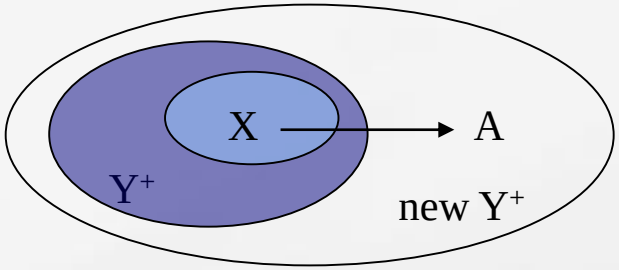
  - AB-> C ?

Test de fermeture

- Une façon plus simple pour inférer des DF est de calculer la fermeture de Y, noté Y^+
  - Etant donné un ensemble F de DF et X est un ensemble d'attributs, on note X^+ l'ensemble des attributs A tels que X->A est conséquence de F

➢ **Fermeture d'un ensemble d'attributs**

- Initialisation: $Y^+ = Y$.
- Induction: Rechercher une partie gauche de DF dans F qui soit inclue dans le Y^+ courant. Si la DF est X->A, ajouter A à Y^+. Répéter ces opérations jusqu'à ce que l'on n'ajoute plus de nouveaux attributs dans Y^+.
  - Autrement dit, $Y^+_{n+1}=Y^+_n$ U {A| Z->A et Z inclus dans Y^+_n}
  - Arrêt: stabilité $Y^+_{n+1}=Y^+_n$



Test de fermeture

- Exemple :
  - Soit F = { AB->C, C->A, BC->D, ACD->B, D->EG, BE->C, CG->BD, CE->AG }, quelle est la fermeture de BD ?
  - Initialisation: $BD^+=\{B,D\}$
  - Itération 1:
    - rechercher les DF dont la partie gauche est dans BD^+: D->EG
    - ajouter leur partie droite dans la fermeture: $BD^+=\{B,D,E,G\}$
  - Itération 2:
    - rechercher les DF dont la partie gauche est dans le nouveau BD^+: D->EG, BE->C
    - ajouter leur partie droite dans la fermeture: $BD^+=\{B,D,E,G,C\}$
  - Itération 3:
    - rechercher les DF dont la partie gauche est dans le nouveau BD^+: C->A, BC->D, D->EG, BE->C, CE->AG
    - ajouter leur partie droite dans la fermeture: $BD^+=\{B,D,E,G,C,A\}$
  - …

Idée simple pour trouver toutes les DF

- Commencer à partir d'un ensemble de DF connues et trouver toutes les DF non triviales qui découle de cet ensemble de DF
  - non triviale = partie droite non inclue dans la partie gauche
- **Fermeture d'un ensemble de DF**

- Un algorithme simple, mais exponentiel
  1. Pour chaque ensemble d'attribut *X* de la relation, on calcule X^+.
  2. Ajouter *X ->A* pour tout *A* dans X^+ - *X*.
  3. Supprimer *XY ->A* si on découvre *X ->A*.
     car *XY ->A* découle de *X ->A* quelle que soit la manière dont on projette

- Astuces:
  - Inutile de calculer la fermeture de l'ensemble contenant tous les attributs et celle de l'ensemble vide
  - Si on trouve X^+ = tous les attributs, alors il en va de même de tout X' contenant *X*

Idée simple pour trouver toutes les DF

- Exemple:
  - Soit la relation ayant pour schéma {*A,B,C*} avec les DF *A ->B* et *B ->C*, cad F={A->B, B->C}

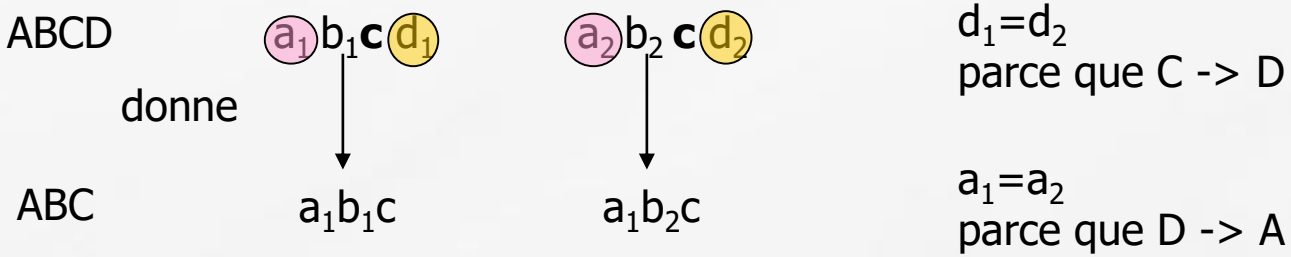
  - $A^+ = ABC$; donne *A ->B*, *A ->C*
  - $B^+ = BC$; donne *B ->C*
  - $C^+ = C$; ne donne rien
  - AB^+ ; inutile de calculer car A^+= tous les attributs
  - AC^+ ; inutile de calculer car A^+= tous les attributs
  - $BC^+ = BC$; ne donne rien

  - DF obtenues:
    F^+={*A ->B*, *A ->C*, *B ->C*}

Objectif de l'inférence des DF

- Motivation: "normalisation", le processus où le schéma d'une relation est divisé en plusieurs schémas.

- Exemple: Soit une relation R ayant pour schéma {A,B,C,D} avec l'ensemble de DF {AB->C, C->D, D->A}
  - Décomposer en 2 schémas {A,B,C} et {A,D}.
  - Quelles DF doivent être satisfaites dans {A,B,C} ? non seulement AB->C mais aussi C->A !



Ainsi, des tuples issus de la projection avec les mêmes valeurs pour C ont la même valeur pour A, cad C -> A.

PLAN

- ✓ Inférence des DF
- ➢ **Normalisation de bases de données**

Design de schéma relationnel

- L'objectif du design de schéma relationnel est d'éviter les anomalies et les redondances
  - *Anomalie de mise à jour* : une occurrence d'une information est modifiée et pas les autres
  - *Anomalie de suppression* : une information pertinente est perdue en détruisant un n-uplet.

- Exemple de mauvais design:

  Drinkers(name, addr, beersLiked, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	???	WickedAle	Pete's	???
Spock	Enterprise	???	A.B.	Bud

Les données sont redondantes, car chaque **???** peut être retrouvé en utilisant les DF name -> addr favBeer et beersLiked -> manf.

Design de schéma relationnel

- Ce mauvais schéma fait aussi ressortir des anomalies

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

- Anomalie de mise à jour: si Janeway part pour l'*Intrepid*, pensera-t-on à changer tous les tuples?

- Anomalie de suppression: si personne n'aime Bud, on perd le fait que son fabricant soit Anheuser-Busch.

➢ Besoin de propriétés, de règles, de méthodes permettant de concevoir de bons schémas

Forme Normale de Boyce-Codd (BCNF)

- Une relation R est dite en *BCNF* ssi pour toute Dépendance Fonctionnelle non triviale *X ->A* sur les attributs de R, X est une super clé.
  - non triviale = X ne contient pas A
  - super clé = clé (minimale) ou sur-ensemble d'une clé (minimale)

- Exemple:

  Drinkers(name, addr, beersLiked, manf, favBeer)

  DF: name->addr favBeer, beersLiked->manf

  - Une seule clé minimale {name, beersLiked}.
  - Pour chaque DF: la partie gauche n'est pas une super clé
  - *Drinkers* n'est pas en BCNF (prendre l'une des deux DF au choix)

Forme Normale de Boyce-Codd (BCNF)

- Autre exemple:

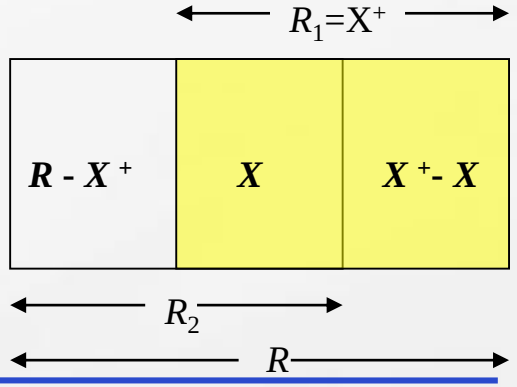
  Beers(name, manf, manfAddr)

  DF: name->manf, manf->manfAddr

  - Une seule clé minimale {name} .
  - name->manf ne contredit pas BCNF, mais par contre la relation n'est pas BCNF à cause de manf->manfAddr .

Décomposer un schéma en BCNF

- Soit une relation *R* avec un ensemble F de DF
- Chercher les DF *X ->B* telles que X ne soit pas une clé
  - Si R pas BCNF, il y en a au moins une.
- Calculer X^+.
  - Rq: X^+ ne contient pas tous les attributs, sinon X serait une clé.
- Décomposer *R* en suivant X->B
  - Remplacer *R* par par deux relations dont les attributs sont:
    - $R_1 = X^+$
    - $R_2 = R - (X^+ - X)$
  - *Projeter* les DF de la fermeture de F sur ces deux nouveaux schémas



Décomposer un schéma en BCNF

- Exemple :

  Drinkers(name, addr, beersLiked, manf, favBeer)

  F = { name->addr, name -> favBeer, beersLiked->manf }

  - Trouver une DF qui fait que Drinkers n'est pas en BCNF:
    name->addr

  - Calcule de la fermeture des attributs de la partie gauche:
    $\{name\}^+ = \{name, addr, favBeer\}$

  - On obtient deux relations:
    1. Drinkers1(name, addr, favBeer) = $\{name\}^+$
    2. Drinkers2(name, beersLiked, manf)
       = {name, addr, beersLiked, manf, favBeer} - {addr, favBeer}

Décomposer un schéma en BCNF

- Exemple (suite) :
  - Ce n'est pas fini; nous devons vérifier que Drinker1 et Drinker2 sont en BCNF
  - Projeter les DF est assez facile pour ce cas (car $F=F^+$)
    - F^+ = { name->addr, name -> favBeer, beersLiked->manf }
    - Pour Drinkers1(name, addr, favBeer), les DF pertinentes sont name->addr et name->favBeer.
    - {name} est donc la seule clé et Drinkers1 est en BCNF.
    - Pour Drinkers2(name, beersLiked, manf), la seule DF est beersLiked->manf, et la seule clé est {name, beersLiked}.
    - ➢ **Pas BCNF, on recommence** la décomposition pour Drinkers2.

Décomposer un schéma en BCNF

- Exemple (suite) :
  - La DF qui fait que Drinkers2 n'est pas en BCNF est beersLiked->manf
  - Calcule de la fermeture {beersLiked}$^+$ = {beersLiked, manf}
  - On décompose donc *Drinkers2* en
    1. Drinkers3(beersLiked, manf)=X^+
    2. Drinkers4(name, beersLiked)={name,beersLiked,manf} - {manf}
  - Projeter les DF de **l'ensemble F$^+$** dans ces nouveaux schémas
    - Pour Drinkers3(beersLiked, manf), la seule DF pertinente est beersLiked->manf
    - {beersLiked} est donc la seule clé et Drinkers3 est en BCNF
    - Pour Drinkers4(name, beersLiked), aucune DF (implicite ou pas) n'est pertinente
    - {name, beersLiked} est clé et Drinkers4 est en BCNF

Décomposer un schéma en BCNF

- Exemple (fin) :
  - Décomposition de *Drinkers* :
    1. Drinkers1(name, addr, favBeer)
    2. Drinkers3(beersLiked, manf)
    3. Drinkers4(name, beersLiked)

- Rq: *Drinkers1* décrit les personnes, *Drinkers3* décrit les bières, et *Drinkers4* décrit la relation entre les personnes et les bières.

Drinkers1

name	addr	favBeer
Janeway	Voyager	WickedAle
Spock	Enterprise	Bud

Drinkers3

beersLiked	manf
Bud	A.B.
WickedAle	Pete's

Drinkers4

name	beersLiked
JaneWay	Bud
JaneWay	WickedAle
Spock	Bud

Problème de la décomposition en BCNF

- Certaines configurations de DF posent problème lorsque l'on essaye de décomposer un schéma en BCNF

- Exemple:
  - *AB ->C* et *C ->B* avec *A* = street, *B* = city, et *C* = zip
  - Il y a deux clés, {*A,B* } et {*A,C* }
  - *C ->B* contredit BCNF, il faudrait décomposer en {*A,C*} et {*B,C*}
  - **Problème**: si nous utilisons ces schémas, nous ne retrouvons plus la DF *AB ->C* à partir des DF projetées

Exemple de DF non préservée

street	zip
545 Tech Sq.	02138
545 Tech Sq.	02139

city	zip
Cambridge	02138
Cambridge	02139

faire une jointure sur le code postal

street	city	zip
545 Tech Sq.	Cambridge	02138
545 Tech Sq.	Cambridge	02139

Bien qu'aucune DF ne soit violée dans chacune des relations décomposées, la DF street city -> zip est violée dans la base de données dans son ensemble

La 3NF évite le problème de non préservation des DF

- 3e Forme Normale (3NF) assouplit la condition de BCNF pour garantir une décomposition préservant les DF
- Un attribut est dit *premier* s'il fait partie d'une clé **minimale**.
- Une relation n'est pas en 3NF ssi on peut trouver une DF *X ->A* telle que
  - *X* n'est pas une clé et
  - A n'est pas premier (ne fait pas partie d'une clé minimale)
- Exemple :
  - Dans l'exemple précédent avec la relation ayant pour schéma {A,B,C} et les DF *{AB ->C, C ->B}*
  - Les clés minimales sont AB et AC
  - Chaque attribut *A*, *B*, ou *C* est premier
  - Bien que *C ->B* contredise BCNF, ce schéma est en 3NF

Couverture minimale des DF

- Besoin de calculer une couverture minimale des DF pour décomposer en 3NF
  - Toutes DF a un seul attribut à droite
  - Aucune DF ne peut être retirée
    - si on en retire une, la fermeture de la couverture minimale n'est plus égale à celle de l'ensemble de DF de départ
  - Aucun attribut ne peut être enlevé
    - sans changer le résultat de la fermeture

  ➢ Plus petit ensemble de DF équivalent
    ➢ leurs fermetures sont égales

- Méthode:
  1. Décomposer chaque DF pour avoir un seul attribut à droite
  2. Supprimer les attributs en surnombre à gauche
  3. Supprimer les DF redondantes

Exemple de couverture minimale des DF

- Soit F={A->B, ABCD->E, EF->G, EF->H, ACDF->EG}

- Décomposition des DF pour avoir un seul attribut à droite
- ACDF->EG devient ACDF->E et ACDF->G
- ➢ F={A->B, ABCD->E, EF->G, EF->H, **ACDF->E,ACDF->G**}

- Suppression des attributs en surnombre à gauche (tester chaque attribut de chaque DF)
- ABCD->E peut être remplacé par ACD->E car $ABCD^+=ACD^+$ (grâce à A->B)
- ➢ F={A->B, **ACD->E**, EF->G, EF->H, ACDF->E,ACDF->G}

- Suppression les DF redondantes
  - ACDF->G peut être supprimé car cette dépendance est impliquée par ACD->E et EF->G
  - idem pour ACDF->E
  - ➢ F={A->B, ACD->E, EF->G, EF->H}

Construction d'une décomposition 3NF

- Soit une relation *R* et un ensemble *F* de DF
- Calculer une couverture minimale de *F*
- Pour chaque DF X->A dans cette couverture minimale, créer une relation ayant pour schéma {X,A}
- Si la clé (minimale) n'est pas contenue dans aucune DF, alors ajouter une relation avec pour schéma la clé
- Exemple :
  - Soit la relation R avec pour schéma {A,B,C}, pour ensemble de DF F={A->B, C->B}, et pour clé minimale {A,C}
  - La couverture minimale est {A->B, C->B}
  - Création de deux relations à partir des DF: R_1={A,B} et R_2={C,B}
  - Création d'une relation à partir de la clé de R: R_3={A,C}

Propriétés importantes pour les décompositions

- Préservation des dépendances fonctionnelles
  - On peut vérifier dans les relations projetées que les dépendances originales sont préservées.
  - Contre-exemple: décomposition BCNF de street-city-zip où street city -> zip n'est plus forcément vérifié

- Décomposition Sans Perte d'Information (SPI)
  - On peut projeter la relation de départ sur chacune des composantes et reconstruire la relation de départ.
  - une décomposition BCNF vérifie SPI
  - Exemple: décomposition de Drinkers en Drinkers1, Drinkers3 et Drinkers4

Propriétés des décompositions 3NF et BCNF

	BCNF	3NF
Décomposition Sans Perte d'Information	**oui**	**oui**
Préservation des dépendances fonctionnelles	Pas forcément (ex: street-city-zip)	**oui**