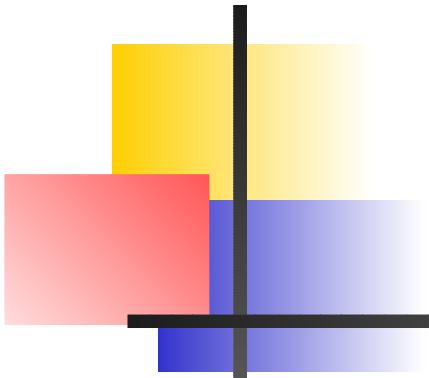

Contraintes d'intégrité Dépendances fonctionnelles

http://www.univ-orleans.fr/lifo/Members/Mirian.Halfeld

Qualité des schémas BD

- Quelques critères (informels) de qualité d'un schéma BD
  - Sémantique simple des attributs et de chaque schéma de relation.
  - Réduction des valeurs redondantes.
  - Réduction des valeurs nuls dans les relations.
  - Interdiction des n-uplets farfelus.
- Problème important de la redondance : Gaspillage de place et **anomalies des mise à jour**.

Exemple

videoId	dateAcquired	title	genre	length	rating
101	1/25/98	The Third-Nine Steps	mystery	120	R
90987	2/5/97	Elisabeth	drama	105	PG13
145	12/31/95	Lady and the Tramp	comics	93	PG
8034	4/5/98	Lady and the Tramp	comics	93	PG
90988	4/5/98	Elisabeth	drama	105	PG13
90989	3/25/86	Elisabeth	drama	105	PG13
543	5/12/95	The Third-Nine Steps	mystery	120	R
123	4/29/91	Annie Hall	comedy	120	R

Exemple

- Redondance : À chaque fois qu'un film (titre) apparaît, les valeurs pour le *genre*, *length* et *rating* apparaissent aussi.
- Mélange de la sémantique des attributs : attributs concernant la vidéo avec des attributs d'un film.
- Anomalie de mise à jour

  Que se passe t-il si la longueur (*length*) du film 90987 est mise à jour et passe de 105 à 107?

  Incohérence ou besoin de modification de plusieurs n-uplet!
- Anomalie d'insertion

  Que se passe t-il si avec l'insertion du n-uplet :

  $$\langle 102, 1/1/99, Elisabeth, drama, 110, PG13 \rangle$$

  Incohérence ou interdiction d'insertion!
- Anomalie de suppression

  Que se passe t-il si avec la suppression du vidéo numéro 123?

  Perte des informations sur le film *Annie Hall*

Contraintes de schéma $\times$ contraintes d'intégrité

Un schéma de base de données n'impose pas certaines contraintes qu'existent dans le monde réel

Exemple: PlaceAssignement [placeNb, personNb, conditions]
$\Rightarrow$ Une personne est associée à un et un seul siège. Un siège est associé à une et une seule personne.

Exemple: Film [title, year, director, type]
$\Rightarrow$ Le titre et le nom du directeur identifient de façon unique le film.

Exemple: Person [ssn, firstName, lastName, address]
$\Rightarrow$ Avec le numéro ssn, nous pouvons identifier une personne

Ces informations ne sont pas visibles à partir du schéma: Besoin des contraintes d'intégrité

Ces contraintes expriment des restrictions sémantiques

Contraintes d'intégrité: lesquelles ?

- Les **dépendances fonctionnelles** sont les contraintes les plus importantes parmi celles que nous devons manipuler pendant le projet d'une base de données relationnelle.
  Exemple: Pour exprimer qu'une personne est associée à un et un seul siège et qu' un siège est associé à une et une seule personne:

  $$\text{personNb} \longrightarrow \text{placeNb}$$
  $$\text{placeNb} \longrightarrow \text{personNb}$$

  Exemple: Pour exprimer qu'un film est identifié par son titre et son director

  $$\text{title, director} \longrightarrow \text{title, director, year, type}$$

  Exemple: Pour exprimer qu'une personne est identifié par le ssn

  $$\text{ssn} \longrightarrow \text{ssn, firstName, lastName, address}$$

- Autres contraintes existent.

Dépendances fonctionnelle: Définition(1)

Définition formelle:

Soit une relation R.
Soit un ensemble d'attributs $X \subseteq sort(R)$ et $A \in sort(R)$.
Une instance de relation $I(R)$ satisfait la dépendance fonctionnelle

$$X \rightarrow A$$

si pour chaque paire de tuples t et u de I,

si $t(X) = u(X)$ alors $t(A) = u(A)$

Nous disons que X détermine fonctionnellement A ou que A dépend de X.

Dépendances fonctionnelle: Définition(2)

- NOTATION:
  Soit un ensemble d'attributs U (univers).
  Soient X et Y des sous-ensemble de U

  $X \rightarrow Y$ est une dépendance fonctionnelle sur U
  Nous pouvons aussi dire que $X \rightarrow Y$ est une dépendance fonctionnelle sur R, étant donné le schéma de relation $R[U]$.

- Si un ensemble d'attributs détermine fonctionnellement plus d'un attribut nous disons que

$$X \rightarrow A_1$$
$$\ldots$$
$$X \rightarrow A_n$$

  et nous résumons en écrivant

$$X \rightarrow A_1, \ldots, A_n \text{ ou}$$
$$X \rightarrow Y \text{ où } Y = \{A_1, \ldots, A_n\}$$

Exemple

Instance I

Movie

title	year	length	filmType	studioName	starName
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Esteves
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

I satisfait la dépendance

title year $\rightarrow$ length, filmType, studioName

mais ne satisfait pas la dépendance

title year $\rightarrow$ starName

Exemples

- Les dépendances fonctionnelles sont des contraintes que nous voulons imposer à la base.

$$Customer[accountId, lastName, firstName, street, city, state, zipCode]$$

$$\begin{array}{lcl} zipCode & \rightarrow & city, state \\ street, city, state & \rightarrow & zipCode \end{array}$$

- Des modifications sur *city*, *state* ou *zipCode* de *Customer* doivent être vérifiées pour ne pas violer les contraintes! Cette vérification peut être très chère.
- Redondance dans la relation *Customer* : répétition de *state* et *city* pour chaque *zipCode*.

La maintenance des DFs

- Une manière de concevoir une BD :
  - Lister tous les attributs qui feront partie de la base.
  - Lister toutes les DFs (contraintes) à imposer.
- Les DFs sont des contraintes qui doivent être respectées : **les utilisateurs veulent l'assurance de la cohérence de la base**.

  Base cohérente = BD qui respecte les contraintes d'intégrité

- **Toute modification qui viole la cohérence de la base doit être INTERDITE**.
- Grand enjeu dans la conception des bases de données : trouver un moyen de rendre la maintenance des DF efficace et moins chère.

Clés (1)

Nous disons qu'un ensemble d'attributs $\{A_1, A_2, \ldots, A_n\}$ est une clé pour une relation si:

1. Les attributs déterminent fonctionnellement **tous** les autres attributs de la relation. En d'autres mots, il est impossible d'avoir deux tuples différents en R ayant les mêmes valeurs pour les attributs $A_1, A_2, \ldots, A_n$.
2. Il n'existe pas un **sous ensemble** de $\{A_1, A_2, \ldots, A_n\}$ qui détermine fonctionnellement tous les autres attributs de la relation.

Clés (2)

Une relation peut avoir plusieurs clés

Super clé: ensemble d'attributs qui contient une clé

Movie

title	year	length	filmType	studioName	starName
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Esteves
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

Les attributs $\{title, year, starName\}$ forment une clé pour Movie.

Comment prouver qu'ils détermine fonctionnellement tous les autres attributs?

Implication sémantique

- Soit F un ensemble de dépendances fonctionnelles sur $R[U]$.
- Soit $X \rightarrow Y$ une df (pas forcement en F).
- Nous disons que

$$F \models X \rightarrow Y \ (F \text{ implique } X \rightarrow Y)$$

  si toute instance de relation sur R qui satisfait les dépendances dans F satisfait aussi $X \rightarrow Y$.
- **Exemple:** Soit $F = \{A \rightarrow B, B \rightarrow C\}$ sur $R[ABC]$. Soit la dépendance fonctionnelle $A \rightarrow C$.

$$F \models A \rightarrow C$$

Fermeture d'un ensemble de dépendances fonctionnelles

$$F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$$

F^+ contient toutes les df impliquées par F.

Clés: Définition formelle

Soient un schéma $R[A_1, \ldots, A_n]$ et $X \subseteq \{A_1, \ldots, A_n\}$.

Soit un ensemble F de dépendances fonctionnelles sur R.
X est clé de R si:

1. $X \rightarrow A_1, \ldots, A_n$ est dans F^+.
2. Il n'existe pas un sous ensemble $Y \subset X$ tel que $Y \rightarrow A_1, \ldots, A_n$ est dans F^+.

Les clés: commentaires

- **Grande avantages des clés sur les DFs :** Les SGDB fournissent le contrôle des clés alors qu'ils ne fournissent pas le contrôle des DFs.
- **Notre but est donc de transformes les DF en clés!**
- Technique de base : identifier les tables avec des DFs qui ne sont pas des clés et les **décomposer** en tables plus petites.
- Pour cela nous devons connaître certaines propriétés des DFs.

Raisonner avec les dépendances fonctionnelles

Soit F un ensemble de dépendances fonctionnelles sur une relation.

- Sans connaître l'instance de la relation, nous pouvons déduire que la relation doit aussi satisfaire un autre ensemble de dépendances fonctionnelles.
- L'habilitée de découvrir des dépendances fonctionnelles "en plus" est TRÈS importante pendant le projet d'une base de données relationnelle.
- Nous avons des règles d'inférence capables de nous dire comment dériver une dépendance fonctionnelle f à partir d'un ensemble de dépendances fonctionnelles F
- Cet ensemble des règles d'inférence est connu par le nom de **Axiomes d'Armstrong**

Axiomes d'Armstrong (1)

Soit F un ensemble de dépendances fonctionnelles qui s'applique à un ensemble d'attributs U.

Réflexivité:
Si $Y \subseteq X \subseteq U$ alors $F \models X \rightarrow Y$.

Cette règle d'inférence fournit les *dépendances triviales*.

Ces dépendances sont valides pour toutes les relations, car elles disent que "deux tuples ayant les mêmes valeurs sur les attributs X ont les mêmes valeurs sur un sous ensemble de X".

$$AB \rightarrow AB; AB \rightarrow A; AB \rightarrow B$$

Axiomes d'Armstrong (2)

Augmentation:

Si $X \rightarrow Y$ alors $XZ \rightarrow YZ$ où $Z \subseteq U$. En d'autres termes

$$X \rightarrow Y \models XZ \rightarrow YZ.$$

Si $AB \rightarrow C$ alors $ABC \rightarrow C$ Si $AB \rightarrow D$ alors $ABC \rightarrow CD$

Axiomes d'Armstrong (3)

Transitivité:

Si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$. En d'autres termes:

$$X \rightarrow Y, Y \rightarrow Z \models X \rightarrow Z.$$

Si $AB \rightarrow C$ et $C \rightarrow D$ alors $AB \rightarrow D$

Lemme

Les axiomes d'Armstrong sont corrects, c'est-à-dire, étant donné une dépendance fonctionnelle f et un ensemble de dépendances fonctionnelles F:

$$\text{Si } F \vdash f \text{ alors } F \models f.$$

□

Il est donc possible d'obtenir un système d'inférence qui est connu sous le nom de **système d'Armstrong**.

Système d'Armstrong

Un ensemble de dépendances fonctionnelles F engendre f

$$F \vdash f$$

s'il existe une suite de dépendances fonctionnelles

$$f_1, f_2, \ldots, f_n$$

avec les propriétés suivantes:

1. $f = f_n$
2. Pour tout $i = 1, \ldots, n$ soit f_i est dans F soit elle est engendrée par l'ensemble $f_1, f_2, \ldots f_{i-1}$ en se servant des axiomes d'Armstrong.

La suite $f_1, f_2, \ldots f_{i-1}$ est alors appelée une dérivation (ou encore une démonstration) de f à partir de F

Exemple

Soit $F = \{A \rightarrow C, B \rightarrow D\}$.

La dépendance $f : AB \rightarrow CD$ est elle engendrée par F?

Démonstration ?

D’autres règles d’inférence

Les règles suivantes sont des conséquences immédiates du système d’Armstrong et simplifient souvent les démonstrations.

1. **Union**: $\{X \rightarrow Y,\ X \rightarrow Z\} \vdash X \rightarrow YZ$
2. **Décomposition**: Si $X \rightarrow Y$ et $Z \subseteq Y$ alors $X \rightarrow Z$.
3. **Pseudo-transitivité**: $\{X \rightarrow Y, WY \rightarrow Z\} \vdash WX \rightarrow Z$

Fermeture d'un ensemble de DF F

- La fermeture de F (F^+) est le plus grand ensemble de DF qui peut être obtenu par l'application répétée des règles d'inférence sur l'ensemble original.
- F^+ décrit toutes les dépendances qui peuvent être inférées à partir des DF imposées sur le schéma.
- F^+ va nous permettre d'évaluer le schéma et de l'améliorer.

Fermeture d'un ensemble d'attributs (1)

Comment répondre à la question F engendre-t-il f?

- Chercher une démonstration: *pas toujours facile*!
- Algorithme efficace pour répondre à cette question de manière mécanique

Fermeture d'un ensemble d'attributs (2)

Soient:

- U: un ensemble d'attributs
- F: un ensemble de dépendances fonctionnelles sur U
- $X \subseteq U$

La **fermeture de X par rapport à F**, notée X^+ est l'ensemble maximal Y tel que $F \vdash X \rightarrow Y$, c'est-à-dire

$$X^+ = max(\{Y \subseteq U \mid F \vdash X \rightarrow Y\})$$

Un tel ensemble existe et est unique.

Théorème 1

La dépendance fonctionnelle $X \rightarrow Y$ est engendrée à partir de F si et seulement si $Y \subseteq X^+$, où X^+ est la fermeture de X par rapport à F.

En d'autres termes,

$$F \vdash X \rightarrow Y \text{ ssi } Y \subseteq X^+$$

Algorithme

F engendre-t-il f ?

function Engendre(F: EnsembleDF; $X \rightarrow Y$: DepFun) : boolean;
begin
    if $Y \subseteq X^+$
    then Engendre := true
    else Engendre := false;
end

Correction et complètude des axiomes d'Armstrong

- Nous voulons montrer que la notion de dérivation est équivalente à celle de l'implication.
- **Le système d'Armstrong est correct**: il n'engendre pas des dépendances fonctionnelles fausses
- **Le système d'Armstrong est complet**: Étant donnée un ensemble F de dépendances fonctionnelles, le système d'Armstrong nous permet d'engendrer toutes les dépendances de F^+.

Ainsi, si $X \rightarrow Y$ ne peut pas être prouvée à partir de F en utilisant les axiomes d'Armstrong, alors il existe une instance de relation où les dépendances de F sont satisfaites, mais où $X \rightarrow Y$ est fausse.

Théorème 2: Correction et Complétude du Système d'Armstrong

Le système d'Armstrong est correct (c'est-à-dire: si $F \vdash f$ alors $F \models f$) et complet (c'est-à-dire: si $F \models f$ alors $F \vdash f$).

En d'autres termes,

$$F \vdash f \text{ ssi } F \models f$$

Nous pouvons ainsi répondre à la question F engendre t-il f de manière mécanique.

Théorème 3

Soit F un ensemble de dépendances fonctionnelles sur U.
Soit X un ensemble d'attributs, tel que $X \subseteq U$.

L'algorithme suivant termine et si ref_f est la valeur finale de la variable res, alors $res_f = X^+$ par rapport à F.

$res := X$
repeat for each $Y \rightarrow Z$ in F do
    if $Y \subseteq res$ then $res := res \cup Z$
until (res does not change) or ($res = U$)

Équivalence entre des ensembles de dépendances fonctionnelles

Soient F et G deux ensembles de dépendances fonctionnelles.

- $F \equiv G$ (F est équivalent à G) si $F^+ = G^+$.
- Pour vérifier si F et G sont équivalents:
  1. Pour chaque df $X \rightarrow Z$ dans F:
     vérifier si $X \rightarrow Z$ est dans G^+ (calculer X^+ par rapport à G et vérifier si $Z \subseteq X^+$).
  2. De façon similaire, pour chaque df $X \rightarrow Z$ dans G:
     vérifier si $X \rightarrow Z$ est dans F^+.
- Afin de montrer que F n'est pas équivalent à G il suffit de trouver une instance de relation I telle que I satisfait F mais pas G (ou vice-versa).

Comment trouver les clés à partir des DF?

- Les clés sont des DF.
- Une manière de trouver les clés consiste à spécifier toutes les DFs et ensuite les utiliser pour trouver les clés.
- Définition formelle des clés.

F^+ contient toutes les DF impliquées par F.
Soient un schéma $R[A_1, \ldots, A_n]$ et $X \subseteq \{A_1, \ldots, A_n\}$.
Soit un ensemble F de dépendances fonctionnelles sur R.

X est clé de R si:

1. $X \rightarrow A_1, \ldots, A_n$ est dans F^+.
2. Il n'existe pas un sous ensemble $Y \subset X$ tel que $Y \rightarrow A_1, \ldots, A_n$ est dans F^+.

Exemple

$PurchaseInfo$
$[pOId, supplierId, supplierName, str, city, st, zip, movieId, title, qty, date]$

Comment obtenir les clés suivantes à partir des DFs données?
K1 : $pOId, movieId$
K2 : $pOId, title$

Dans cet exemple la découverte des clés à partir des DFs nous montre que la $pOId$ determine toutes les propriétés des l'ordre (*purchase order*) alors que $movieId$ (ou $title$) détermine les propriétés des vidéos.

Est-ce qu'il serait intéressant de les séparer?

Couverture (ou réduction) minimale

Une couverture minimale G est un ensemble de dépendances fonctionnelles équivalent à F qui respecte les propriétés suivantes:

- Toute dépendance fonctionnelle $X \rightarrow Y$ dans G est réduite à droite, c'est-à-dire, il n'existe pas de sous-ensemble strict Y' de Y tel que $G \vdash X \rightarrow Y'$.
  Notons que la dépendance $X \rightarrow Y$ est réduite à droite ssi Y est un singleton.
- Toute dépendance fonctionnelle $X \rightarrow Y$ dans G est réduite à gauche, c'est-à-dire, il n'existe pas de sous-ensemble strict X' de X tel que $G \vdash X' \rightarrow Y$.
- G est non redondant, c'est-à-dire, il n'existe pas f dans G tel que $G \setminus \{f\} \vdash f$.

Comment trouver une couverture minimale?

1. **Réduction à droite:**
   Pour chaque dépendance fonctionnelle $X \rightarrow A_1 \ldots A_n$ de F faire remplacer $X \rightarrow A_1 \ldots A_n$ par $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$.
   F_1: l'ensemble de df qui en résulte.

2. **Réduction à gauche:**
   Pour chaque dépendance fonctionnelle $X \rightarrow A$ de F_1 trouver un sous-ensemble strict X' de X tel que $F_1 \equiv ((F_1 \setminus \{X \rightarrow A\}) \cup \{X' \setminus A\})$
   et remplacer $X \rightarrow A$ par $X' \rightarrow A$.
   Nota: Si un tel sous-ensemble X' n'existe pas alors la dépendance $X \rightarrow A$ est réduite à gauche et s'il en existe plusieurs alors nous aurons éventuellement plusieurs réductions.
   F_2: l'ensemble des df qui résulte de cette étape

3. **Réduire les redondances**
   Pour chaque dépendance f de F_2 , si $(F_2 \setminus \{f\}) \vdash f$ alors supprimer f de F_2..
   Nota: Suivant l'ordre dans lequel nous considérons les dépendances F, nous pouvons avoir plusieurs réductions.
   F_3: l'ensemble des df qui résulte de cette étape.

Comment décomposer les attributs?

- Pour éviter les anomalies, la technique de base est la décomposition.
- Mais... il faut savoir décomposer!
- Soit U (Univers) l'ensemble d'attributs d'une base de données.
  Soit une instance de relation sur U
  Il est souvent souhaitable de décomposer les relations sur l'univers U.
  Cependant, pour que tel remplacement soit acceptable, on devrait pouvoir reconstruire, à tout moment, la relation de départ.

Exemple

Soient un schéma de relation $R[ABC]$ et une instance $I(R)$.

Décomposition $\rho = \{R_1[AB], R_2[BC]\}$.

Nous devons avoir $\Pi_{AB}(I) \bowtie \Pi_{BC}(I) = I$.

Détails :

- La décomposition nous donne deux instances de relation:
  1. L'instance J_1, sur $sort(R) = \{A, B\}$
  2. L'instance J_2, sur $sort(S) = \{B, C\}$.
- À partir de la décomposition nous voulons reconstruire l'instance I de départ sur $R[ABC]$.
  Pour cela nous composons des n-uplets qui ont la même valeur pour l'attribut B (car $\{A, B\} \cap \{B, C\} = \{B\}$). Cette opération est appelée *jointure*.
- Le résultat de la dernière opération doit donner I comme résultat.

A	B	C
a	b	c
a_1	b	c_1

A	B
a	b
a_1	b

B	C
b	c
b	c_1

Décomposition sans perte d'information

- Décomposer $R = \{A_1, A_2, \ldots, A_n\}$ consiste à le remplacer par une collection $\rho = \{R_1, R_2, \ldots, R_k\}$ de sous-ensembles de R (par forcement disjoint) tels que $sort(R) = sort(R_1) \cup sort(R_2) \cup \ldots \cup sort(R_k)$
- Soient R un schéma de relation, $\rho = \{R_1, R_2, \ldots, R_k\}$ une décomposition et F un ensemble de dépendances fonctionnelles. La décomposition ρ est *sans perte d'information* (SPI) par rapport à F si pour toute instance I sur R qui satisfait F, nous avons

$$I = \pi_{R_1}(I) \bowtie \pi_{R_2}(I) \bowtie \ldots \bowtie \pi_{R_k}(I).$$

Exemple

- Soient F (ens. DF) et $R[ABC]$.
  Soit I une instance sur R.
  Soit la décomposition $\rho = \{R_1[AB], R_2[BC]\}$.
- **Sous quelles conditions sur F avons nous $\Pi_{AB}(I) \bowtie \Pi_{BC}(I) = I$, c'est-à-dire sous quelles conditions ρ est SPI?**
  - Nous essayons de trouver les conditions sur F pour que tout tuple $t = \langle abc \rangle$ dans $\Pi_{AB}(I) \bowtie \Pi_{BC}(I)$ soit aussi dans I.
  - Si $t = \langle abc \rangle$ est dans $\Pi_{AB}(I) \bowtie \Pi_{BC}(I)$ alors il existe des tuples t_1 et t_2 dans I de forme que $t_1 = \langle abx \rangle$ et $t_2 = \langle ybc \rangle$.
  - Il en suit qu'une condition suffisante pour que t soit dans I est $t = t_1$ ou $t = t_2$, c'est-à-dire, $x = c$ ou $y = a$.
  - Comme t_1 et t_2 sont dans I et comme I doit satisfaire F:
    Si $F \vdash B \rightarrow C$ alors $x = c$ et si $F \vdash B \rightarrow A$ alors $y = a$
  - Donc $F \vdash B \rightarrow C, B \rightarrow A$ est une condition suffisante pour que $\rho = \{R_1[AB], R_2[BC]\}$ soit SPI.
- Ce raisonnement conduit à un algorithme pour vérifier si une décomposition est SPI.

Algorithme CHASE

Entrée

- Un schéma de relation R$[A_1 \ldots A_n]$ (ou un univers $U = \{A_1 \ldots A_n\}$).
- Un ensemble de dépendances fonctionnelles F sur R (ou U).
- Une décomposition $\rho = \{R_1, R_2, \ldots, R_k\}$.

Sortie: Oui ou non la décomposition est SPI par rapport à F.

Algorithme CHASE

1. Construire une table avec n colonnes et k lignes.
   La colonne j correspond à attribut A_j et la ligne i correspond à la relation R_i.
2. Sur la position (ligne i, colonne j):
   `if` A_j appartient à R_i
   - `then` placer le symbole a_j (une constante)
   - `else` placer le symbole b_{ij} (une variable)
3. Considérer les DF de F de la manière suivante:
   `repeat`
   `for` (chaque dépendance fonctionnelle $X \rightarrow Y$ dans F) `do`
   `If` (il existe des tuples avec les mêmes valeurs sur les attributs de X)
   `then` rendre égales les valeurs sur Y pour ces tuples.
   `until` (aucun changement est possible sur la table);
4. `If` (il existe une ligne $a_1 \ldots a_n$ dans la table)
   `then` SPI
   `else` non SPI;

SPI - Cas simple

Test pour une décomposition avec deux schémas de relation

Soient une décomposition $\rho = \{R_1, R_2\}$ et un ensemble de DFs F. La décomposition ρ est SPI ssi

$$(sort(R_1) \cap sort(R_2)) \rightarrow (sort(R_1) \setminus sort(R_2))$$

ou

$$(sort(R_1) \cap sort(R_2)) \rightarrow (sort(R_2) \setminus sort(R_1))$$

sont des dépendances dans F^+.

Pour plusieurs schémas : algorithme **Chase**

Exemple

Soit $F = \{B \rightarrow C\}$ un ensemble de dépendances fonctionnelles sur $U = \{ABC\}$.

Décompositions: (SPI?)

1. $\rho = \{R_1[AB], R_2[BC]\}$
2. $\rho = \{R_1[AB], R_2[AC]\}$
3. $\rho = \{R_1[AC], R_2[BC]\}$

Décompositions sans perte de dépendances

- Nous décidons de stocker les données d'une base suivant une décomposition SPI ρ de U.
- Lors de la mise à jour de la base, nous devons vérifier que la base modifiée reste cohérente, c'est-à-dire, qu'elle satisfait les dépendances de F (sinon la mise à jour doit être refusée).
- Pour ce faire: à chaque mise à jour, reconstituer la relation de départ (sur U), car les dépendances de F sont énoncées sur U.
- **Cette reconstruction est très coûteuse!**

Est-ce qu'il est possible de vérifier la cohérence de la base sans reconstruire la relation sur U?

Décompositions sans perte de dépendances

- Une décomposition ρ est SPD par rapport à F s'il existe un ensemble de DFs G tel que
  1. $G \equiv F$
  2. Pour toute dépendance $X \rightarrow Y$ dans G, il y a R_i dans ρ tel que $XY \subseteq sort(R_i)$
- Une dépendance $X \rightarrow Y$ est applicable sur un schéma de relation R si $XY \subseteq sort(R)$.

Projection de l'ensemble de F sur R

L'ensemble de toutes les dépendances impliquées par F et qui sont applicable sur R.

$$F_R = \{X \rightarrow Y \mid F \vdash X \rightarrow Y \text{ et } XY \subseteq sort(R)\}$$

ou

$$F_R = \{X \rightarrow Y \mid X \rightarrow Y \text{ est dans } F^+ \text{ et } XY \subseteq sort(R)\}$$

Décomposition SPD

Soit une décomposition $\rho = \{R_1, R_2, \ldots, R_n\}$.

La décomposition ρ est SPD par rapport à F si l'union de toutes les dépendances en F_{R_i} $(i = 1, 2, \ldots, n)$ implique F:

$$\bigcup_{i=1}^{i=n} F_{R_i} \vdash F$$

Exemple

Est-ce ρ est SPD?

- $U = \{C, R, Z\}$
- $F = \{CR \rightarrow Z,\ Z \rightarrow C\}$
- $\rho = \{RZ, CZ\}$

Comment savoir si ρ est SPD?

Solution naïve

1. Calculer F^+
2. Calculer $G = \bigcup_{i=1}^{i=n} F_{R_i}$
3. Vérifier si $G \equiv F$.

Comment savoir si ρ est SPD?

Autre solution

- Définissons $G = \bigcup_{i=1}^{i=n} F_{R_i}$
- Nous voulons savoir si $G \equiv F$.
- Pour cela nous considérons chaque $X \rightarrow Y$ dans F et nous calculons X^+ (à partir de G)
- Pour calculer X^+ sans connaître G, nous considérons X^+ en fonction de F_{R_i} ($i = 1, 2, \ldots, n$)
- En d'autres termes nous définissons l'opération

$$Z := Z \cup ((Z \cap R_i)^+ \cap R_i)$$

où Z est un ensemble d'attribut et $(Z \cap R_i)^+$ est la fermeture par rapport à F.

- Nous pouvons ainsi calculer X^+ en relation à G (on commence par X et on considère tous les R_i).

Pour tester si une décomposition est SPD

Entrée

- Un schéma de relation R$[A_1 \ldots A_n]$ (ou un univers $U = \{A_1 \ldots A_n\}$).
- Un ensemble de dépendances fonctionnelles F sur R (ou U).
- Une décomposition $\rho = \{R_1, R_2, \ldots, R_k\}$.

Sortie Oui ou non la décomposition est SPD par rapport à une dépendance $X \rightarrow Y$ dans F.

1. Calculer X^+.
   - Soit Z un ensemble d'attributs.
   - $Z := X$;
   - `while` (Z change) `do`
     `for` i:=1 `to` n `do`
     $Z := Z \cup ((Z \cap R_i)^+ \cap R_i)$
2. `If` ($Y \in Z$)
   `then` SPD par rapport à $X \rightarrow Y$
   `else` non SPD;

Normalisation

- Une **normalisation** consiste en transformer des objets dans une forme structurale qui satisfait une collection des règles.
- Un schéma dans une **forme normale** possède certaines caractéristiques de qualité.
- Plusieurs travaux de recherche ont été nécessaires pour permettre la proposition des **formes normales** et des règles pour leur obtention.

La forme normale Boyce-Codd

Un schéma de relation R (ou univers U) est en FNBC par rapport à F si, pour tout $X \rightarrow A$ tel que $F \vdash X \rightarrow A$ (F satisfait $X \rightarrow A$):

1. Soit $X \rightarrow A$ est triviale
2. Soit X est sur clé

Une décomposition $\rho = \{R_1, R_2, \ldots, R_k\}$ de R est FNBC par rapport à F si R_i est FNBC par rapport à F_{R_i}, pour $i = 1, 2, \ldots, k$.

Une décomposition FNBC est toujours SPI, mais pas forcement SPD

Trouver une décomposition FNBC

Entrée

- Un schéma de relation R$[A_1 \dots A_n]$ (ou un univers $U = \{A_1 \dots A_n\}$) qui n'est pas FNBC.
- Un ensemble de dépendances fonctionnelles F sur R (ou U).

Sortie: Une décomposition $\rho = \{R_1, R_2, \dots, R_k\}$ FNBC

1. Soit $X \rightarrow Y$ une dépendance non triviale telle que $F \vdash X \rightarrow Y$ et X n'est pas sur clé (une telle dépendance existe puisque U n'est pas FNBC).
2. Décomposer U en XY et $U \setminus Y$
3. Si XY (ou $U \setminus Y$) n'est pas en FNBC par rapport à F_{XY} (ou $F_{U \setminus Y}$) alors aller au pas 1.

Exemple

Soit le schéma $R[CTHRSG]$ et l'ensemble de dépendances fonctionnelles F (couverture minimale) ci-dessous. Trouver une décomposition FNBC. Est-ce que cette décomposition préserve les dépendances?

$$C \rightarrow T \quad HR \rightarrow C \quad HT \rightarrow R$$
$$CS \rightarrow G \quad HS \rightarrow R$$

Troisième Forme Normale

Un schéma de relation R (ou univers U) est en FN3 par rapport à F si, pour tout $X \rightarrow A$ tel que $F \vdash X \rightarrow A$ (F satisfait $X \rightarrow A$):

1. Soit $X \rightarrow A$ est triviale
2. Soit X est sur clé
3. Soit A fait partie d'une clé

Trouver une décomposition FN3

Entrée :

- Un schéma de relation R$[A_1 \ldots A_n]$ (ou un univers $U = \{A_1 \ldots A_n\}$) qui n'est pas FN3.
- Un ensemble de dépendances fonctionnelles F sur R (ou U). F *est une couverture minimale.*

Sortie : Une décomposition $\rho = \{R_1, R_2, \ldots, R_k\}$ FN3 qui préserve les dépendances.

1. Si un attribut en U n'apparaît pas en F alors cette attribut peut rester tout seul.
2. Si une dépendance fonctionnelle de F contient tous les attributs $A_1 \ldots A_n$,
   alors R$[A_1 \ldots A_n]$ est le résultat
   sinon la décomposition ρ est composée par le XA pour chaque dépendance $X \rightarrow F$.

FN3

- Si $X \rightarrow A_1, \ldots X \rightarrow A_n$ sont des dépendances d'une couverture minimale, alors nous pouvons avoir un schéma avec les attributs $XA_1 \ldots A_n$ (à la place de n schémas $XA_1 \ldots XA_n$).
- Soit une décomposition FN3 ρ de R obtenue par l'algorithme et soit X une clé de R.

  Alors $\rho_1 = \rho \cup \{X\}$ est une décomposition FN3 de R. Cette décomposition préserve l'information et les dépendances.

Exemple

Soit le schéma $R[CTHRSG]$ et l'ensemble de dépendances fonctionnelles F (couverture minimale) ci-dessous. Trouver une dépendances FN3 qui préserve les informations.

$$C \rightarrow T \quad HR \rightarrow C \quad HT \rightarrow R$$
$$CS \rightarrow G \quad HS \rightarrow R$$

Dépendances d'inclusion

STUDENTS

StudNb	StudName	Address	Phone
102	Mary Smith	78 rue d'Alésia, Paris	01 45 67 87 23
103	Pedro Ferrari	90 bd Saint Michel, Paris	01 23 67 34 23
104	Andre Dupont	340 bd Saint Michel, Paris	01 45 67 56 23
…			

COURSES

CourseNb	CourseName	Hours
CS101	Databases	20h
CS101	Databases	20h
CS102	Compilers	20h
…		

Dépendances d'inclusion

INSCRIPTION

Course	Student
CS101	102
CS101	103
CS101	104
CS102	103
…	

Dépendances d'inclusion

- Soit **R** un schéma de base de données. Une dépendance d'inclusion sur **R** est une expression de la forme

$$R(A_1, \ldots, A_m) \subseteq S(B_1, \ldots, B_m)$$

où:

1. R et S sont des noms de relations dans **R** (éventuellement identiques)
2. $A_1, \ldots, A_m$ est une suite d'attributs distincts appartenant à $sort(R)$
3. $B_1, \ldots, B_m$ est une suite d'attributs distincts appartenant à $sort(S)$

- Une instance **I** de **R** satisfait une dépendance d'inclusion $R(A_1, \ldots, A_m) \subseteq S(B_1, \ldots, B_m)$ si

$$\mathbf{I}(R)(A_1, \ldots, A_m) \subseteq \mathbf{I}(S)(B_1, \ldots, B_m)$$

Règles d'inférence

Soient X, Y et Z des ensembles d'attributs et R, S and T des schémas de relation.

- **Réflexivité**: $R[X] \subseteq R[X]$
- **Projection et Permutation**: Si $R[A_1, \ldots, A_n] \subseteq S[B_1, \ldots, B_n]$ alors $R[A_{i_1}, \ldots, A_{i_k}] \subseteq S[B_{i_1}, \ldots, B_{i_k}]$ pour chaque séquence d'entiers distincts $i_1, \ldots, i_k$ dans $\{1, \ldots, m\}$
- **Transitivité**: Si $R[X] \subseteq S[Y]$ et $S[Y] \subseteq T[Z]$, alors $R[X] \subseteq T[Z]$.

Clés étrangères

- Exemples de dépendances d'inclusion
- Très utilisées!
- Contrainte de clé étrangère:

$$R[X] \subseteq S[Y]$$

et Y **est la clé primaire de** S.