BDav-M1
Bases de données avancées

Cristina Sirangelo
IRIF, Université Paris Diderot
cristina@liafa.univ-paris-diderot.fr

Modélisation de BD relationnelles : théorie de la normalisation

Sources (quelques slides empruntés et réadaptés) :

- cours *Database systems principles* - V. Vianu, UCSD
- cours *Introduction to databases* - C.Re, Stanford Univ.

Modélisation de BD relationnelles

Conception du modèle relationnel (schéma) à partir du réel

Rappel : deux approches

Approche "brute - force" :

- Identifier des attributs d'intérêt
- repartir les attributs dans plusieurs relations

Approche modélisation conceptuelle :

- production d'un modèle conceptuel
- traduction en relationnel (automatique)
- potentiellement : ultérieure raffinement

Dans les deux cas on a besoin de :

- savoir détecter si un schéma relationnel a des "bonnes propriétés" ou pas
- si ce n'est pas les cas :

  des techniques pour le reconduire à un "bon" schéma (*forme normale*)

Qualité d'un schéma relationnel

Quelles sont de "bonnes propriétés" d'un schéma relationnel?

Exemple:

Attributs relatifs à des vendeurs, produits, et fournitures

V#: numéro de vendeur

Vnom: nom du vendeur

Vville: ville du vendeur

P#: numéro du produit

Pnom: nom du produit

Pville: ville où le produit est stocké

Qte: quantité de produit fournie au vendeur

Qualité d'un schéma relationnel

- Un schéma relationnel possible : une seule relation "fourniture" avec tous les attributs

R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

- C'est une mauvaise modélisation! Pourquoi?

1) Redondance

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	...	...	...	...
3	MagicV	Paris	...	...	...	...
2	IdealB	Lyon	...	...	...	...
2	IdealB	Lyon	...	...	...	...

Ex: Vnom et Vville sont **déterminés** par V#, i.e.

si deux fournitures ont le même V# , elles ont aussi le même Vville et le même Vnom

On représente l'information que le vendeur 3 est MagicV et il est à Paris, une fois pour chaque fourniture : **redondant**

Qualité d'un schéma relationnel

R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

C'est une mauvaise modélisation! Pourquoi?

1) Redondance

2) Anomalies de mise à jour

Vnom ou Vville pourrait être mis à jour dans une fourniture et pas dans une autre, ce qui donnerait une incohérence. Pour éviter cela : mise à jour plus couteuse.

3) Anomalies d'insertion

On ne peut pas stocker un vendeur s'il ne reçoit pas de fourniture

4) Anomalies de suppression

Si on supprime toutes les fournitures d'un vendeur, on perd toute l'info sur ce vendeur

Qualité d'un schéma relationnel

Solution : un "bon" schéma

Vendeur (V#, Vnom, Vville) Clef: V#
Produit (P#, Pnom, Pville) Clef: P#
Fourniture(V#, P#, Qte) Clef: V# P#

Plus d'anomalie! Comment y arriver?

La théorie de la normalisation des bd relationnelles nous donne:

- Des formes normales :
  - propriétés d'un schémas qui garantissent absence (ou réduction) de redondance, et des anomalies qui en dérivent
  - définies par rapport à un ensemble de contraintes (appelés dépendances)

- Des techniques de normalisation : passage d'un schéma arbitraire (mauvais) à un schéma en forme normale (typiquement par décomposition)

Contraintes d'intégrité et dépendances

Contrainte d'intégrité sur un schéma

Une propriété que les instances du schéma sont censées satisfaire pour être valides

- e.g. contrainte de clef: NSS est une clef pour la relation Personne (NSS, nom, adresse)

- c'est la réalité qu'on modélise qui impose les contraintes

Le processus de modélisation doit identifier non-seulement les information à représenter, mais également les contraintes qui existent sur celles-ci

⇒ Notre point de départ : un schéma relationnel (potentiellement à raffiner) avec un ensemble de contraintes identifiées

Contraintes d'intégrité et dépendances

- Dépendances fonctionnelles : Une forme particulière de contraintes d'intégrité
- Exemple

  Schéma : R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

  Un ensemble de dépendances fonctionnelles qu'on peut raisonnablement supposer :

  V# → Vnom Vville
  P# → Pnom Pville
  V# P# → Qte

- Sémantique (intuition) : pour que une instance J de la relation R soit valide, J doit satisfaire :

  - si deux tuples dans J ont la même valeur de V#

    alors ils ont la même valeurs de Vnom et de Vville

  - si deux tuples dans J ont la même valeur de P#

    alors ils ont la même valeurs de Pnom et de Pville

  - si deux tuples dans J ont la même valeur de V# et la même valeur de P#

    alors ils ont la même valeurs de Qte

Dépendances fonctionnelles

Exemple

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	322	manteau	Lille	2
1	StarV	Rome	546	veste	Rome	1
3	MagicV	Paris	322	manteau	Lille	5
2	IdealB	Lyon	145	jupe	Paris	7
2	IdealB	Lyon	234	jupe	Lille	1

J satisfait V# → Vnom Vville et P# → Pnom Pville

Dépendances fonctionnelles

Exemple

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	322	manteau	Lille	2
1	StarV	Rome	546	veste	Rome	1
3	MagicV	Paris	322	manteau	Lille	5
2	IdealB	Lyon	145	jupe	Paris	7
2	IdealB	Lyon	234	jupe	Lille	1

J viole V# P# → Qte

Dépendances fonctionnelles (DF)

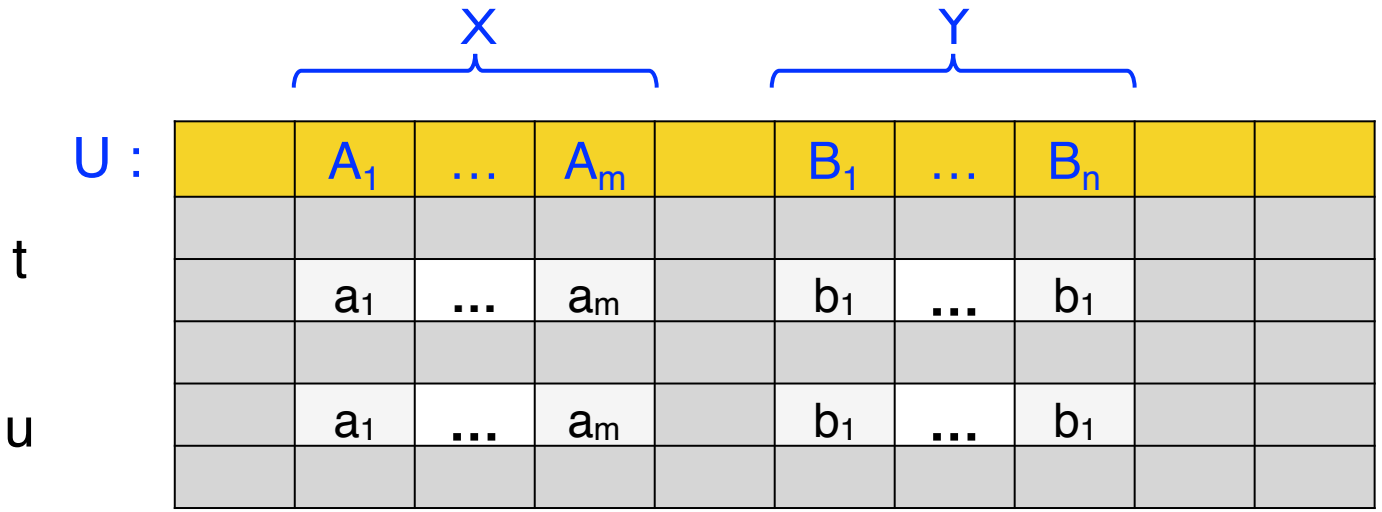
Soit R(U) un schéma de relation avec U : ensemble d'attributs

Une dépendance fonctionnelle est une expression : $X \rightarrow Y$, avec $X, Y \subseteq U$

Une instance J de R(U) satisfait $X \rightarrow Y$ si pour toute paire de tuples t, u dans J

$$t[X] = u[X] \Rightarrow t[Y] = u[Y]$$

(si t et u sont en accord sur X alors t et u sont en accord sur Y)



J satisfait un ensemble F de DF, si J satisfait chaque DF dans F

Dépendances fonctionnelles (DF)

Soit R(U) un schéma de relation avec U : ensemble d'attributs

Une dépendance fonctionnelle est une expression : $X \rightarrow Y$, avec $X, Y \subseteq U$

Une instance J de R(U) satisfait $X \rightarrow Y$ si pour toute paire de tuples t, u dans J

$$t[X] = u[X] \Rightarrow t[Y] = u[Y]$$

(si t et u sont en accord sur X alors t et u sont en accord sur Y)

Remarque sur la notation.

Par la suite un ensemble d'attributs {A1,…,An} sera dénoté par A1…An

Donc A1…An → B1..Bn dénotera la DF {A1, …,An} → {B1, …, Bn}

Dépendances fonctionnelles et modélisation E/R

S'il y a eu un phase de modélisation E/R, les contraintes d'identification, les associations, les contraintes de cardinalité et les contraintes externes du schéma E/R impliquent des DF sur le schéma relationnel

Rappel : exemple de mauvaise modélisation

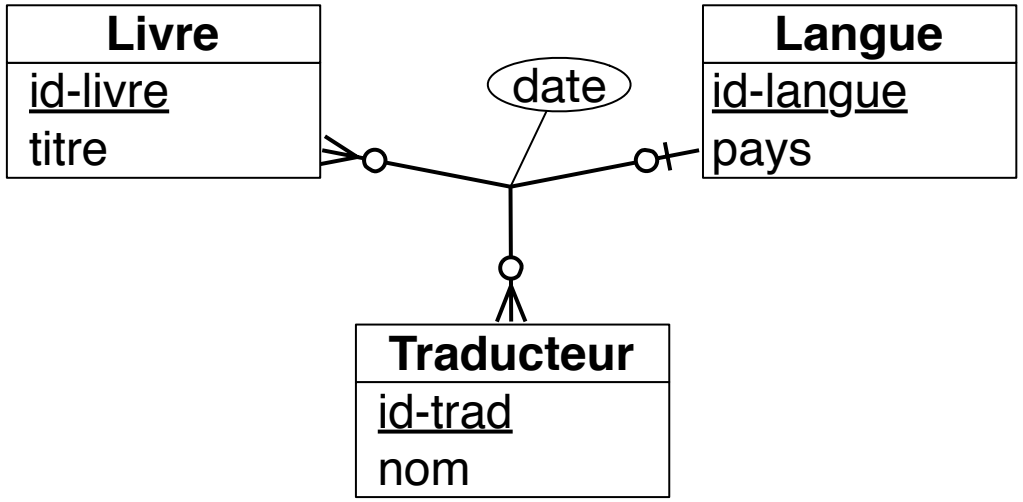


Schéma relationnel associé :

Livre (id-livre, titre)

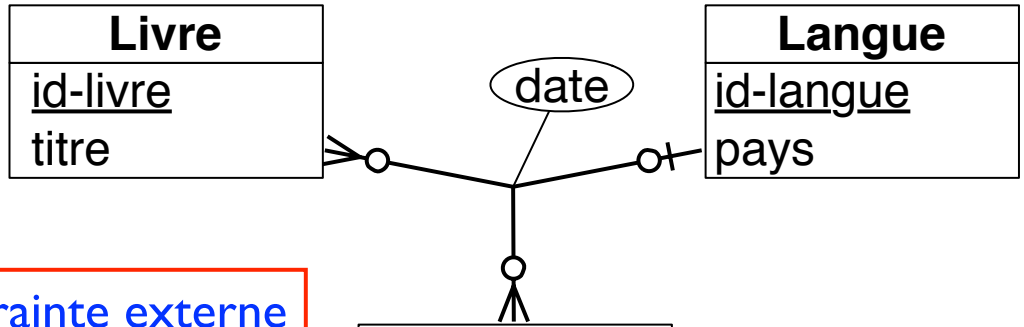
Langue (id-langue, pays)

Traducteur (id-trad, nom)

Traduction (id-livre, id-trad, id-langue, date)

+ contrainte externe
un traducteur peut traduire dans une seule langue

Dépendances fonctionnelles et modélisation E/R



+ contrainte externe
un traducteur peut traduire dans une seule langue

Schéma relationnel associé :

Livre (id-livre, titre)

Langue (id-langue, pays)

Traducteur (id-trad, nom)

Traduction (id-livre, id-trad, id-langue, date)

DF sur le schéma relationnel:

sur Livre : id-livre → titre — par la contrainte d'id. sur l'entité Livre

sur Langue : id-langue → pays — par la contrainte d'id. sur l'entité Langue

sur Traducteur : id-trad → nom — par la contrainte d'id. sur l'entité Traducteur

sur Traduction : id-livre id-trad id-langue → date — par l'association Traduction

id-livre id-trad → id-langue — par la contrainte de card. max=1 sur Traduction

id-trad → id-langue — par la contrainte externe

Dépendances fonctionnelles et qualité du schéma

- Un schéma relationnel est "bon" ou pas, selon les contraintes qui y sont associées

  – **Exemple 1 :** **R** (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

  redondances et anomalies dues par exemple à la dépendance fonctionnelle

  V# →Vnom, Vville

  – **Exemple 2.** **Traduction** (id-livre, id-trad, id-langue, date)

  redondances et anomalies dues à la dépendance fonctionnelle

  id_trad → id_langue

- Donné R(U) , F

  on apprendra à "normaliser" R(U) par rapport à F

Décomposition d'un schéma de relation

L'outil indispensable pour arriver à une forme normale

- Soit R(U) un schéma de relation
- Une décomposition de R(U) est un ensemble { $R_1(S_1)$, …, $R_k(S_k)$ } de schémas de relation tels que:

$$U = \bigcup_{i=1}^{k} Si$$

- Exemple

{ Vendeur (V#, Vnom, Vville),
Produit (P#, Pnom, Pville),
Fourniture(V#, P#, Qte) }

est une décomposition de
R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

Décomposition

- Objectif de la décomposition :

  Eliminer (ou réduire) les redondances et les anomalies associées (i.e obtenir un nouveau schéma en "forme normale")

- Mais on ne peut pas décomposer arbitrairement

- Conditions pour une décomposition "raisonnable" :

  - Décomposition sans perte d'information

  - Décomposition sans perte de dépendances fonctionnelles

Décomposition sans perte d'information

Idée : Si on remplace R par {Vendeur, Produit, Fournitures}

notre BD, au lieu de stocker une instance J de R stockera ses projections

$\pi_{V\#,Vnom,Vville}(J)$ $\pi_{P\#,Pnom,Pville}(J)$ $\pi_{V\#,P\#,Qte}(J)$

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	5	jupe	Paris	5
3	MagicV	Paris	6	veste	Lille	2
2	IdealB	Lyon	12	manteau	Lyon	1
2	IdealB	Lyon	13	jupe	Paris	1

$\pi_{V\#,Vnom,Vville}(J)$

V#	Vnom	Vville
3	MagicV	Paris
2	IdealB	Lyon

$\pi_{V\#,P\#,Qte}(J)$

V#	P#	Qte
3	5	5
3	6	2
2	12	1
2	13	1

$\pi_{P\#,Pnom,Pville}(J)$

P#	Pnom	Pville
5	jupe	Paris
6	veste	Lille
12	manteau	Lyon
13	jupe	Paris

Décomposition sans perte d'information

Idée

- La décomposition doit garantir que pour toute instance J de R, les projections de J contiennent la "même information" que J

- C'est à dire on doit pouvoir reconstruire une instance J de R à partir de ses projections

- Comment tenter de reconstruire l'instance à partir de ses projections?
  Jointure naturelle

$$\Pi_{V\#, Vnom, Vville}(J) \bowtie \Pi_{P\#, Pnom, Pville}(J) \bowtie \Pi_{V\#, P\#, Qte}(J)$$

Décomposition sans perte d'information

Rappel. Jointure naturelle de deux instances de relation:
I avec ensemble d'attributs X, et J avec ensemble d'attributs Y

$I \bowtie J$

retourne l'ensemble des tuples t sur attributs $X \cup Y$ telles que $t[X] \in I$ et $t[Y] \in J$

X= {A, B}

Y= {B,C}

I

A	B
1	2
4	2
6	6
7	7

J

B	C
2	3
2	5
9	1
8	8

$I \bowtie J$

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

Décomposition sans perte d'information

Propriété souhaitée pour notre décomposition :

$$J = \pi_{V\#, Vnom, Vville}(J) \bowtie \pi_{P\#, Pnom, Pville}(J) \bowtie \pi_{V\#, P\#, Qte}(J)$$

pour toute instance valide J de R

Est cela vrais?

Dans l'exemple de J donné, oui. Mais pour d'autre J valides?

Intuitivement, Oui : puisque en partant de la fourniture (V#, P#, Qte)

- V# nous permet de récupérer toutes les infos sur un unique vendeur
  (grâce à la DF V# → Vnom Vville)
- P# nous permet de récupérer toutes les infos sur un unique produit

  (grâce à la DF P# → Pnom Pville)

(une procédure plus rigoureuse pour ce test plus tard)

la propriété de décompositions sans perte d'information dépend

des dépendances fonctionnelles

Décomposition sans perte d'information (*lossless join*)

Définition.

Soir R(U) une schéma de relation et F un ensemble de DFs sur R.

Une décomposition $\{ R_1(S_1), \ldots, R_k(S_k) \}$ de R est

sans perte d'information par rapport à F

ssi, pour toute instance J de R qui satisfait F,

$$J = \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$$

Un exemple de décomposition avec perte d'information

R(A, B, C) décomposition : { R1(A, B), R2(B, C) }

F= {AB → C}

Il existe une instance J de R qui satisfait F, mais qu'on ne peut pas reconstruire à partir de ses projections:

J

A	B	C
1	2	3
4	2	5

$\pi_{AB}(J)$

A	B
1	2
4	2

$\pi_{BC}(J)$

B	C
2	3
2	5

$\pi_{AB}(J) \bowtie \pi_{BC}(J)$

A	B	C
1	2	3
4	2	5
1	2	5
4	2	3

Décomposition sans perte d'information (*lossless join*)

Pour une instance J arbitraire, quelle est la connexion entre

J et $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$?

Décomposition sans perte d'information (*lossless join*)

Pour une instance J arbitraire, quelle est la connexion entre

J et $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$?

- Pour tout J, $J \subseteq \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$

  Par la définition de jointure naturelle et projection :

  $t \in J \Rightarrow t[Si] \in \pi_{Si}(J)$ pour tout i $\Leftrightarrow t \in \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$

- le seule problème est donc que les jointures peuvent générer des tuples en plus (voir exemple précédent)

- Mais J n'est pas arbitraire : J satisfait des DFs, cela peut garantir l'inclusions inverse dans certains cas

Tester si un décomposition est sans perte d'information

Sur l'exemple des fournitures d'abord.

R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

DFs
V# → Vnom Vville
P# → Pnom Pville
V# P# → Qte

Décomposition :

Vendeur (V#, Vnom, Vville), Produit (P#, Pnom, Pville), Fourniture(V#, P#, Qte)

Question :

$J = \pi_{V\#,Vnom,Vville}(J) \bowtie \pi_{P\#,Pnom,Pville}(J) \bowtie \pi_{V\#,P\#,Qte}(J)$

pour toute instance J de R qui satisfait F ?

Tester si un décomposition est sans perte d'information

- Soit J une instance de R qui satisfait F

un tuple t :

V#	Vnom	Vville	P#	Pnom	Pville	Qte
a_1	a_2	a_3	a_4	a_5	a_6	a_7

est dans $\pi_{V\#,Vnom,Vville}(J) \bowtie \pi_{P\#,Pnom,Pville}(J) \bowtie \pi_{V\#,P\#,Qte}(J)$

ssi :

(a1 a2 a3) $\in \pi_{V\#,Vnom,Vville}(J)$ ssi (a1 a2 a3, --, --, --, --) $\in$ J

(a4 a5 a6) $\in \pi_{P\#,Pnom,Pville}(J)$ ssi (--, --, --, a4, a5, a6, --) $\in$ J

(a1, a4, a7) $\in \pi_{S\#,P\#,Qte}(J)$ ssi (a1, --, --, a4, --, --, a7) $\in$ J

"--" dénote l'existence d'une valeur

- On résume dans un tableau (le tableau de la requête de jointure!)
  - une ligne par relation de la décomposition
  - on utilise des variables z_i distinctes pour les valeurs inconnus

Tester si un décomposition est sans perte d'information

- Soit J une instance de R qui satisfait F

F = {V# → Vville Vnom
P# → Pnom Pville
P#V# → Qte}

	V#	Vnom	Vville	P#	Pnom	Pville	Qte	
Vendeurs	a_1	a_2	a_3	z_1	z_2	z_3	z_4	
Produit	z_5	z_6	z_7	a_4	a_5	a_6	z_8	← motif dans J
Fourniture	a_1	z_9	z_{10}	a_4	z_{11}	z_{12}	a_7	⇕
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	← t : tuple dans la jointure

- J satisfait F ⇒ le motif ne peut pas être arbitraire
- En utilisant les DF on déduit des égalités entre les zi, et entre les ai et les zi

Procédure appelée ***chase***

Tester si un décomposition est sans perte d'information

- Soit J une instance de R qui satisfait F

F = {V# → Vville Vnom
P# → Pnom Pville
P#V# → Qte}

	V#	Vnom	Vville	P#	Pnom	Pville	Qte	
Vendeurs	a_1	**a_2**	**a_3**	z_1	z_2	z_3	z_4	
Produit	z_5	z_6	z_7	a_4	a_5	a_6	z_8	← motif dans J
Fourniture	a_1	**z_9**	**z_{10}**	a_4	z_{11}	z_{12}	a_7	⇕
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	← t : tuple dans la jointure

- J satisfait F ⇒ le motif ne peut pas être arbitraire
- En utilisant les DF on déduit des égalités entre les zi, et entre les ai et les zi

Procédure appelée ***chase***

Tester si un décomposition est sans perte d'information

- Soit J une instance de R qui satisfait F

F = {V# → Vville Vnom
P# → Pnom Pville
P#V# → Qte}

	V#	Vnom	Vville	P#	Pnom	Pville	Qte	
Vendeurs	a_1	**a_2**	**a_3**	z_1	z_2	z_3	z_4	
Produit	z_5	z_6	z_7	a_4	a_5	a_6	z_8	← motif dans J
Fourniture	a_1	**a_2**	**a_3**	a_4	z_{11}	z_{12}	a_7	⇕
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	← t : tuple dans la jointure

- J satisfait F ⇒ le motif ne peut pas être arbitraire
- En utilisant les DF on déduit des égalités entre les zi, et entre les ai et les zi

Procédure appelée ***chase***

Tester si un décomposition est sans perte d'information

- Soit J une instance de R qui satisfait F

F = {V# → Vville Vnom
P# → Pnom Pville
P#V# → Qte}

	V#	Vnom	Vville	P#	Pnom	Pville	Qte	
Vendeurs	a_1	a_2	a_3	z_1	z_2	z_3	z_4	
Produit	z_5	z_6	z_7	a_4	**a_5**	**a_6**	z_8	← motif dans J
Fourniture	a_1	a_2	a_3	a_4	**z_{11}**	**z_{12}**	a_7	⇕
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	← t : tuple dans la jointure

- J satisfait F ⇒ le motif ne peut pas être arbitraire
- En utilisant les DF on déduit des égalités entre les zi, et entre les ai et les zi

Procédure appelée ***chase***

Tester si un décomposition est sans perte d'information

- Soit J une instance de R qui satisfait F

F = {V# → Vville Vnom
P# → Pnom Pville
P#V# → Qte}

	V#	Vnom	Vville	P#	Pnom	Pville	Qte	
Vendeurs	a_1	a_2	a_3	z_1	z_2	z_3	z_4	
Produit	z_5	z_6	z_7	a_4	**a_5**	**a_6**	z_8	← motif dans J
Fourniture	a_1	a_2	a_3	a_4	**a_5**	**a_6**	a_7	⇕
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	← t : tuple dans la jointure

- J satisfait F ⇒ le motif ne peut pas être arbitraire
- En utilisant les DF on déduit des égalités entre les zi, et entre les ai et les zi

Procédure appelée ***chase***

Tester si un décomposition est sans perte d'information

- Soit J une instance de R qui satisfait F

F = {V# → Vville Vnom
P# → Pnom Pville
P#V# → Qte}

	V#	Vnom	Vville	P#	Pnom	Pville	Qte	
Vendeurs	a_1	a_2	a_3	z_1	z_2	z_3	z_4	
Produit	z_5	z_6	z_7	a_4	a_5	a_6	z_8	← motif dans J
Fourniture	a_1	a_2	a_3	a_4	a_5	a_6	a_7	⇕
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	← t : tuple dans la jointure

- J satisfait F ⇒ le motif ne peut pas être arbitraire
- En utilisant les DF on déduit des égalités entre les zi, et entre les ai et les zi
- Si on obtient une ligne avec uniquement des ai ⇒ t ∈ J ⇒

$\pi_{V\#,Vnom,Vville}(J) \bowtie \pi_{P\#,Pnom,Pville}(J) \bowtie \pi_{S\#,P\#,Qte}(J) \subseteq J$ ⇒ lossless join

Tester si un décomposition est sans perte d'information

L'algorithme dans le cas général

Input: $R(A_1, ..A_n)$, une décomposition $\{R_1, \ldots, R_k\}$, un ensemble F de DFs

1. Construire un tableau dont les colonnes sont les attributs de R

> le tableau a une ligne pour chaque Ri
>
> - cette ligne a un symbole a_k en correspondance de chaque attribut A_k de R_i
>
> les autres positions du tableau sont remplies avec des variables z_i distinctes

3. (*Chase* du tableau avec F)

> répéter tant que possible :
>
> s'il existent une DF $X \rightarrow Y$ dans F et deux lignes du tableau en accord sur X
>
> égaliser ces deux lignes sur Y (en remplaçant des *z* par des autres *z* ou par des *a* *)

4. **Output :** OUI ssi le tableau résultant a une ligne de *ai*

* *chaque remplacement doit être effectué partout dans le tableau*

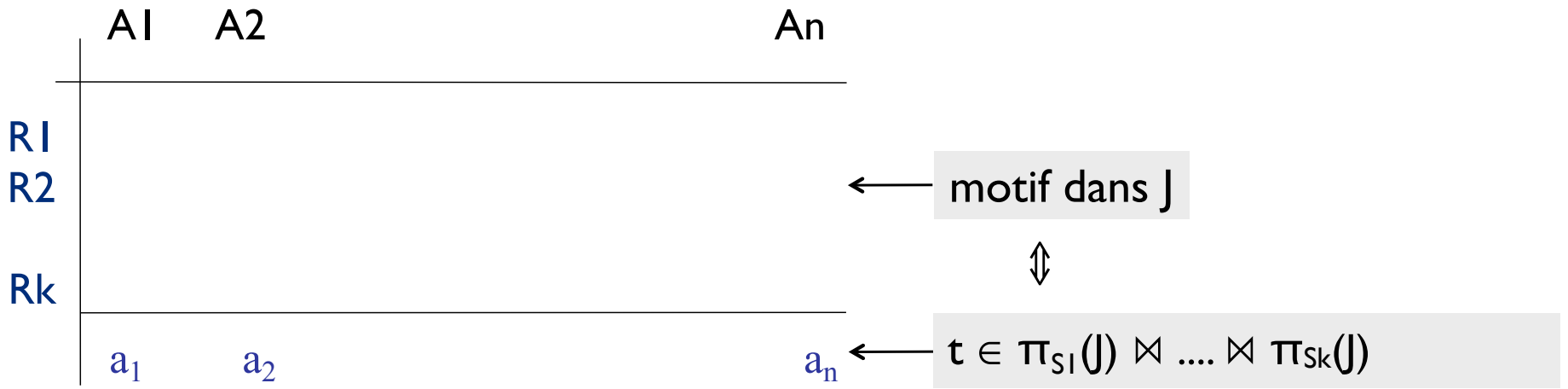
Tester si un décomposition est sans perte d'information

Correction de l'algorithme :

L'algorithme renvoie OUI ssi $\{R_1, \ldots, R_k\}$ est sans perte d'information par rapport à F

Idée de la preuve (généralisation de l'exemple vu)

Chaque étape de *chase* montre que pour tout *t* et tout J qui satisfait F :



Si l'algorithme dit OUI : $t \in \pi_{S1}(J) \bowtie .... \bowtie \pi_{Sk}(J) \Rightarrow t \in J$, pour tout t et tout J valide

$\Rightarrow$ pour tout J valide, $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J) \subseteq J \Rightarrow$ Lossless join

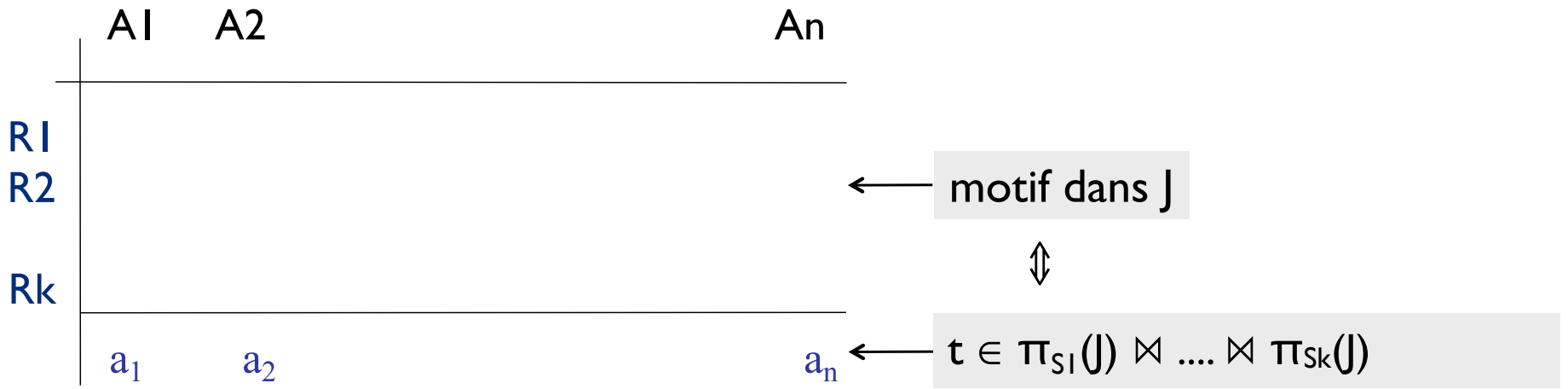
Tester si un décomposition est sans perte d'information

Correction de l'algorithme :

L'algorithme renvoie OUI ssi $\{R_1, \ldots, R_k\}$ est sans perte d'information par rapport à F

Idée de la preuve (généralisation de l'exemple vu)

Chaque étape de *chase* montre que pour tout *t* et tout J qui satisfait F :



Si l'algorithme dit NON : $t \in \pi_{S1}(J) \bowtie .... \bowtie \pi_{Sk}(J)$ et

le motif à la dernière étape est une instance J de R qui satisfait F mais ne contient pas *t*

i.e. il existe une J valide tel que $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J) \not\subseteq J \Rightarrow$ perte d'info.

Rappel : décomposition

- Objectif de la décomposition :

  Eliminer (ou réduire) les redondances et les anomalie associées (i.e obtenir un nouveau schéma en "forme normale")

- Mais on ne peut pas décomposer arbitrairement

- Conditions pour une décomposition "raisonnable" :

  - Décomposition sans perte d'information

  - Décomposition sans perte de dépendances fonctionnelles

Vers la préservation des DF: implication de DF

On a besoin de comprendre l'implication de DF pour vérifier la préservation des DF

(mais aussi pour calculer les clefs, vérifier la satisfaction de formes normales, etc.)

- Les DF données peuvent impliquer d'autre DF additionnelles

- Exemple: $A \rightarrow B$ et $B \rightarrow C$ implique $A \rightarrow C$

C'est à dire
toute instance de relation qui satisfait $A \rightarrow B$ et $B \rightarrow C$
satisfait également $A \rightarrow C$

- Un autre exemple :

$A \rightarrow C$, $BC \rightarrow D$, $AD \rightarrow E$ implique $AB \rightarrow E$

Implication de DF

Définition.

Un ensemble F de DF implique une autre DF $X \rightarrow Y$

si toute instance de relation qui satisfait F satisfait également $X \rightarrow Y$

Notation pour " F implique $X \rightarrow Y$" : $F \models X \rightarrow Y$

Toutes les DF impliqués par F: $F^+ = \{ X \rightarrow Y \mid F \models X \rightarrow Y \}$

Exemple : $\{ A \rightarrow B, B \rightarrow C \}^+$ inclut les dépendances suivantes :

$A \rightarrow B$, $B \rightarrow C$, $A \rightarrow C$ $AB \rightarrow C$, ...

mais aussi des DF "triviales" (i.e. satisfaites par toute instance)

$A \rightarrow A$, $AB \rightarrow A$, $ABC \rightarrow A$, $B \rightarrow B$, $AB \rightarrow B$, etc.

Implication de DF : Axiomes de Armstrong

Trois règles d'inférence (dont la correction est facile à vérifier) :

Pour un schéma de relation R(U) , et $X, Y, Z \subseteq U$

1) *Transitivité :* $\{ X \rightarrow Y,\ Y \rightarrow Z \} \models X \rightarrow Z$

2) *Augmentation :* $X \rightarrow Y \models X\,Z \rightarrow Y\,Z$

3) *Réflexivité :* $\models XY \rightarrow X$ (appelée DF triviale)

Ces règles ne sont pas seulement correctes , il s'agit d'axiomes , i.e

$F \models X \rightarrow Y$ ssi

$X \rightarrow Y$ peur être dérivé de F par application successives des trois règles ci-dessus

Implication de DF : d'autres règles

Plusieurs autres règles correctes, mais pas nécessaires pour former des axiomes (dérivables des axiomes) :

Union : $\{ X \rightarrow Y,\ X \rightarrow Z \} \models X \rightarrow YZ$

Séparation : $X \rightarrow YZ \models X \rightarrow Y$

et d'autres encore ...

Remarque : pour simplifier la notation, on peut omettre {…} pour les ensembles de DF :

$X1 \rightarrow Y1, \ldots, Xn \rightarrow Yn$ dénote l'ensemble de DF : $\{ X1 \rightarrow Y1, \ldots, Xn \rightarrow Yn \}$

Implication de DF : équivalence

Ensembles équivalents de DF

Soit F et G deux ensembles de DF sur R(U)

F est équivalent à G si $F \models G$ et $G \models F$

(i.e. ssi $F^+ = G^+$)

On peut toujours remplacer un ensemble de DF avec un ensemble équivalent

Remarque : Par les règles d'*Union* , *Séparation* et *Réflexivité*:

- $X \rightarrow A_1, \ldots A_n$ équivalent à $X \rightarrow A_1, \ldots, X \rightarrow A_n$
- $XY \rightarrow YZ$ équivalent à $XY \rightarrow Z$

Implication de DF

Question principale d'un point de vu algorithmique:

Comment vérifier si un ensemble F de DF implique une DF $X \rightarrow Y$?

Ou bien, par les équivalences du slide précédent :

Comment vérifier si un ensemble F de DF implique une DF $X \rightarrow A$?

(X :ensemble, A: attribut)

Implication de DF : Clôture d'un ensemble d'attributs

Vérifier si $X \rightarrow A$ est impliqué par un ensemble F de DF :

- on pourrait utiliser les axiomes de Armstrong (et les autres règles dérivables) pour essayer de dériver $X \rightarrow A$ à partir de F
- souvent plus utile de penser en termes de clôture de X

Clôture de X (par rapport à F): l'ensemble d'attributs "déterminés" par X

Définition.

La clôture d'un ensemble d'attributs X par rapport à un ensemble F de DF est

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

Exemple.

R (ABCDE) F = { $AB \rightarrow C$, $C \rightarrow D$, $E \rightarrow D$ } $(AB)^+ = ABCD$

Implication de DF : Clôture d'un ensemble d'attributs

Vérifier si $X \rightarrow A$ est impliqué par un ensemble F de DF :

- on pourrait utiliser les axiomes de Armstrong (et les autres règles dérivables) pour essayer de dériver $X \rightarrow A$ à partir de F
- souvent plus utile de penser en termes de clôture de X

Clôture de X (par rapport à F): l'ensemble d'attributs "déterminés" par X

Définition.

La clôture d'un ensemble d'attributs X par rapport à un ensemble F de DF est

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

Caractérisation : $F \models X \rightarrow A \quad \text{iff} \quad A \in X^+$

Vérifier si $X \rightarrow A$ est impliqué par F : se réduit à calculer une clôture

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

A B

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

Ⓐ B (A→C)

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

A B C

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

A (B C) (BC→D)

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

A B C D

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

Ⓐ B C Ⓓ (AD→E)

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

A B C D E

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples sont en accord sur X=AB

alors elles sont aussi en accord sur :

A B C D E

X^+ = ABCDE

Calcul de la clôture d'un ensemble d'attributs

Algorithme général:

Soit F un ensemble de DF sur R(U) et $X \subseteq U$.

L'algorithme suivant calcule la clôture X^+ de X par rapport à F

$X_c := X$

tant que il existe $V \rightarrow Z$ dans F tel que $V \subseteq X_c$ et $Z \notin X_c$

$X_c := X_c \cup Z$

renvoyer X_c

X_c grandit à chaque itération

Comme U est fini l'algorithme termine en au plus |U| itérations

Correction de l'algorithme de clôture

1) L'algorithme calcule uniquement des attributs dans la clôture (i.e. $X_c \subseteq X^+$)

Idée (intuition donnée sur l'exemple de clôture) :

Si on suppose qu'une instance de R satisfait F et que deux tuples sont en accord sur X,

l'algorithme ajoute uniquement des attributs sur lesquels les deux tuples sont en accord

2) L'algorithme calcule tous les attributs dans la clôture
(i.e. $X^+ \subseteq X_c$ quand l'algorithme termine)

Preuve. Supposer $A \notin X_c$ quand l'algorithme termine

- quand l'algorithme termine, pour toute DF $V \rightarrow Z$ telle que $V \subseteq X_c$ on a $Z \subseteq X_c$

$\Rightarrow$ J

- X_c -	A	...
a a ... a	c	c c ... c
a a ... a	d	d d ... d

satisfait F

- mais J ne satisfait pas $X \rightarrow A$ $\Rightarrow$ $A \notin X^+$