BDav-M1

Bases de données avancées

Cours de Cristina Sirangelo

IRIF, Université Paris Diderot

Assuré en 2021-2022 par Amélie Gheerbrant

amelie@irif.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 : raffinement ultérieur

Dans les deux cas on a besoin de :

- savoir détecter si un schéma relationnel a de "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 ces “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 qu'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 coûteuse.

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 schéma 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

Vers une définition formelle de “qualité” d’un schéma relationnel

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 informations 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 qu'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# $\rightarrow$ 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

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

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

			X				Y			
U :		A_1	…	A_m		B_1	…	B_n		
t		a_1	…	a_m		b_1	…	b_1		
u		a_1	…	a_m		b_1	…	b_1		

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

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

(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}

Remarque sur le vocabulaire

X → Y en mots : "X determine Y"

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

Exemple de mauvaise modélisation

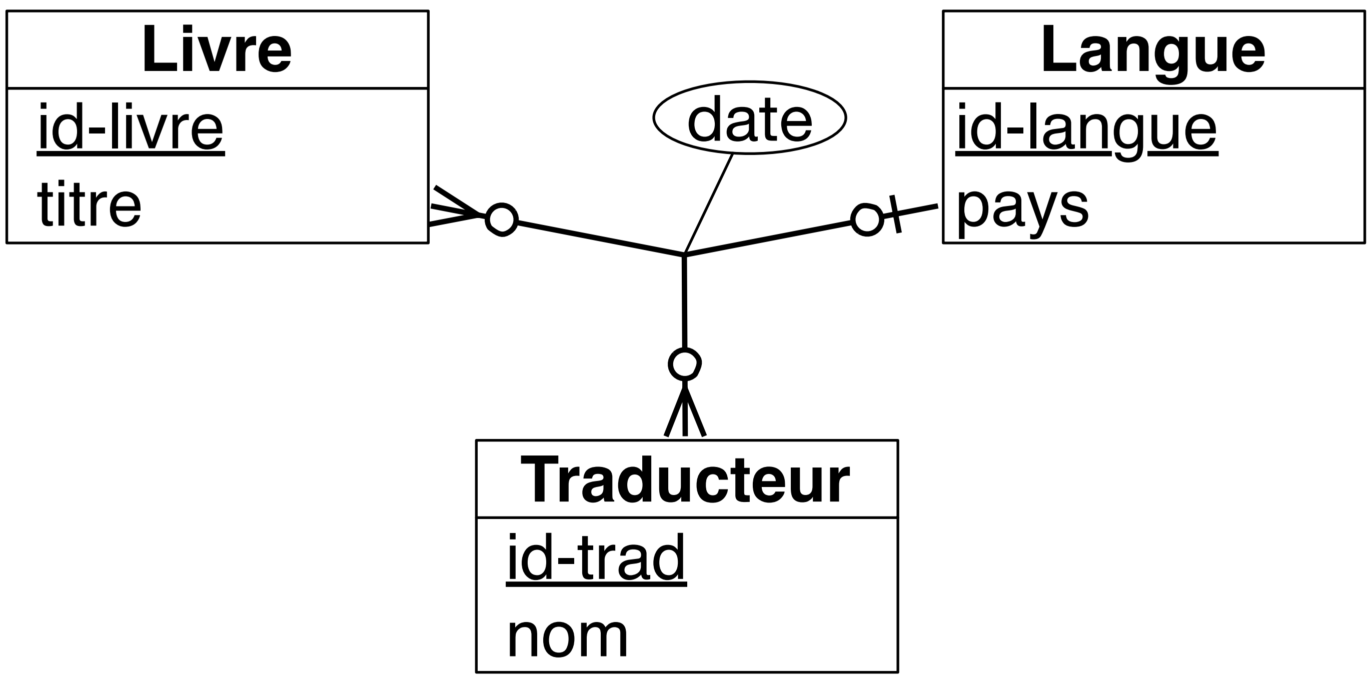


Schéma relationnel associé :

Livre (id-livre, titre)

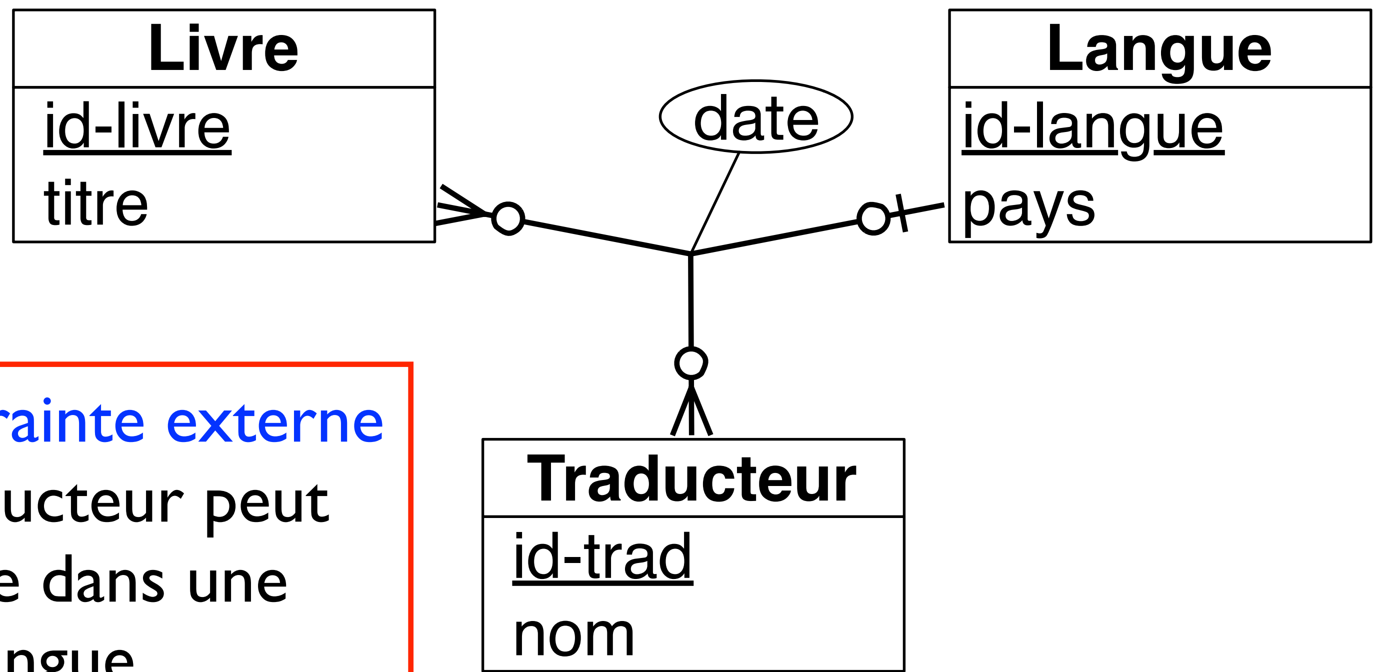
Langue (id-langue, pays)

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

    redondances et anomalies dues à la dépendance fonctionnelle

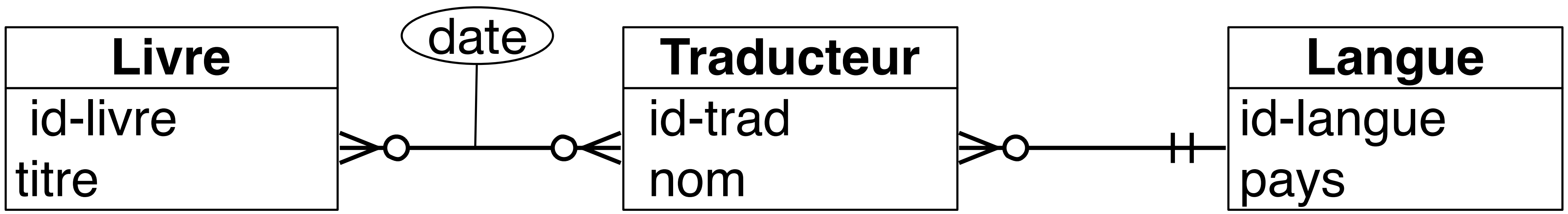
    id_trad → id_langue

Traduction

id-livre	id-trad	id-langue	date
3	5	44	02/2006
1	5	44	10/2001
2	5	44	03/1984
2	3	22	09/1998

Dépendances fonctionnelles et qualité du schéma

- Une meilleure modélisation conceptuelle produit un schéma relationnel qui n'a pas ces problèmes :



Livre (<u>id-livre</u>, titre) id-livre → titre

Langue (<u>id-langue</u>, pays) id-langue → pays

Traducteur (<u>id-trad</u>, nom, id-langue) id-trad → nom , id-trad → id-langue

Traduction (<u>id-livre, id-trad</u>, date) id-livre id-trad → date

Traduction

id-livre	id-trad	date
3	5	02/2006
1	5	10/2001
2	5	03/1984
2	3	09/1998

Traducteur

id-trad	nom	id-langue
5	Dupont	44
3	Blanc	22

Dépendances fonctionnelles et qualité du schéma

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

redondances et anomalies dues à la dépendance fonctionnelle

V# →Vnom, Vville

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

Qualité d’un schéma relationnel : formes normales

- *Formes normales* :
  - formalisent la notion de “qualité” d’un schéma par rapport à un ensemble de contraintes
  - viennent avec des techniques de *normalisation* : “corriger” un “mauvais” schéma pour le reconduire à un schéma en forme normale
  - normalisation : transformation qui opère directement sur le modèle relationnel (sans revenir sur la modélisation conceptuelle)
    - *but : éliminer les problèmes de redondance et les anomalies*

Livre (id-livre, titre)

Langue (id-langue, pays)

Traducteur (id-trad, nom)

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

normalisation

Livre (<u>id-livre</u>, titre)

Langue (<u>id-langue</u>, pays)

Traducteur (<u>id-trad</u>, nom, id-langue)

Traduction (<u>id-livre, id-trad</u>, date)

Formes normales

- Plusieurs formes normales proposées pour les schéma relationnels :
  - première, deuxième, troisième,   Boyce-Codd, ...

- *Forme normale de Boyce-Codd* (FNBC): la plus restrictive pour un schéma relationnel par rapport à un ensemble de dépendances fonctionnelles

- Sa définition nécessite d'introduire un peu de terminologie sur les dépendances fonctionnelles…

Vers la définition des formes normales: implication de DF

- Soit R(U) un schéma de relation (U ensemble d'attributs) et F un ensemble de DF sur U
  - ex. R(ABC), F= { $A \rightarrow B$, $B \rightarrow C$}
- Les DF données peuvent impliquer d'autre DF additionnelles

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

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 $X \to Y$

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

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

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

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

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

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

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

Clefs et super-clefs

Soit R(U) un schéma de relation et F un ensemble de DF sur U.

– Super-clef : $X \subseteq U$ tel que $F \models X \rightarrow U$

X détermine tous les attributs de R

– Clef (ou clef candidate) : $X \subseteq U$ tel que X est une super-clef et il n'existe pas

$Y \subsetneq X$ tel que Y est une super-clef

Exemple : R(ABC) $F = \{A \rightarrow B, B \rightarrow C\}$

Super-clefs : A, AB, AC, ABC

Clef : A (la seule)

Forme normale de Boyce-Codd (FNBC)

Forme normale de Boyce-Codd

Un schéma de relation R(U) est en FNBC par rapport à un ensemble de DF F sur R(U),

ssi pour tout $X \rightarrow A \in F^+$ tel que $A \notin X$, X est une super-clef pour R

C'est à dire les seules DF non-triviales sont celles induites par des super-clefs

FNBC : Exemple 1

Exemple des traductions :

schéma 1 (venant de l'association ternaire) :

Livre (id-livre, titre) {id-livre → titre}

Langue (id-langue, pays) {id-langue → pays}

Traducteur (id-trad, nom) {id-trad → nom}

Traduction (id-livre, id-trad, id-langue, date) : {id-livre id-trad id-langue → date

id-livre id-trad → id-langue

id-trad → id-langue}

La relation Traduction n'est pas en FNBC par rapport à ses DF

FNBC : Exemple 1

Exemple des traductions :

schéma 2 (venant des deux associations binaires) :

Livre (<u>id-livre</u>, titre) {id-livre → titre}

Langue (<u>id-langue</u>, pays) {id-langue → pays}

Traducteur (<u>id-trad</u>, nom, id-langue) {id-trad → nom , id-trad → id-langue}

Traduction (<u>id-livre, id-trad</u>, date) {id-livre id-trad → date}

Chaque relation est en FNBC par rapport à ses DF

FNBC : Exemple 2

schéma 1

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

F = V# → Vnom Vville

P# → Pnom Pville

V# P# → Qte

R n'est n'est pas en FNBC par rapport à F

En effet ni V# ni P# ne sont des super-clefs pour R (ni V# ni P# ne déterminent Qte)

FNBC : Exemple 2

schéma 2

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

(V# est une super-clef pour *Vendeur*

P# est une super-clef pour *Produit*

V# P# est une super-clef pour *Fourniture*)

FNBC =

absence de redondance (et anomalies associées) dues aux DFs

Une vue (simplifiée) de la modélisation de schéma relationnel avec des DF

1. - Choisir les attributs d'intérêt U et produire un schéma de relation R(U)

- Alternative : utiliser une étape de modélisation conceptuelle (e.g E/R) et traduire en un schéma relationnel $R_1(U_1)$ ... $R_k(U_k)$.

2. Spécifier toutes les DF pour R (ou pour R_1.. R_k)

- rappel : un schéma E/R peut exprimer des DFs par les contraintes d'identification, les associations, les contraintes de cardinalité et les contraintes externes

3. Si R (ou une Ri) n'est pas dans une forme normale souhaitée (FNBC par exemple)*

- normaliser R (ou chaque Ri)

- alternative :"corriger" la modélisation conceptuelle et revenir à l'étape 2.

* **Remarque.** Si on est passé par une étape de modélisation conceptuelle, $\{R_1..R_k\}$ a des chances d'être déjà en forme normale, mais cela n'est pas garanti.

Normalisation

- Donné R(U) , F

  on apprendra à "normaliser" R(U) par rapport à F
- L'algorithme de normalisation dépend de la forme normale souhaitée
- Dans tous les cas : normalisation par décomposition
- Avant d'étudier ces algorithmes on a besoin de comprendre l'implication de DF

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

$X \rightarrow Y$ peut être dérivé de F par applications 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 vue 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 $X \to A$ est impliqué par un ensemble F de DF :

- on pourrait utiliser les axiomes de Armstrong (et les autres règles dérivables) pour essayer de dériver $X \to A$ 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

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

Exemple.

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

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 : $F \models X \rightarrow A$ iff $A \in X^+$

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) $F = \{A \rightarrow C,\ BC \rightarrow D,\ AD \rightarrow E\}$ $X = AB$

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 :

A 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) $F = \{A \to C,\ BC \to D,\ AD \to E\}$ $X = AB$

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 :

A B C D (AD→E)

Exemple de calcul de la clôture

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

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

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) $F = \{A \to C,\ BC \to D,\ AD \to E\}$ $X = AB$

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 $Z \nsubseteq X_c$
>
> $X_c := X_c \cup Z$
>
> 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 $X \rightarrow A$ $\Rightarrow$ $A \notin X^+$

Rappel : Normalisation

- Donné R(U) , F

  on apprendra à “normaliser” R(U) par rapport à F
- L’algorithme de normalisation dépend de la forme normale souhaitée
- Dans tous les cas : normalisation par décomposition

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 { $R_1(S_1), \ldots, R_k(S_k)$ } 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)

Propriétés d’une décomposition

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

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

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

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

V#	Vnom	Vville
3	MagicV	Paris
2	IdealB	Lyon

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

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 (*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,

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

Décomposition sans perte d'information

Dans l'exemple, 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-ce vrai ?

Dans l'exemple de J donné, oui. Mais pour d'autres 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 loin)

la propriété de décompositions sans perte d'information dépend

des dépendances fonctionnelles

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 : un algorithme simple existe

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)

Un algorithme pour tester si

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

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 (par définition de $\bowtie$) :

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

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

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

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 (par définition de $\bowtie$ et de projection) :

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

"--"
dénote l'existence
d'une valeur

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 (par définition de $\bowtie$ et de projection) :

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

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

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

← motif dans J

⇕

← 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	$\mathbf{a_5}$	$\mathbf{a_6}$	z_8	← motif dans J
Fourniture	a_1	a_2	a_3	a_4	$\mathbf{a_5}$	$\mathbf{a_6}$	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
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

← motif dans J

⇕

← 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 ⇒ $t \in J$

donc :

$\pi_{V\#, Vnom, Vville}(J) \bowtie \pi_{P\#, Pnom, Pville}(J) \bowtie \pi_{S\#, P\#, Qte}(J) \subseteq J$ ⇒ 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 a ou à défaut, des *z**)

4. **Output :** OUI ssi le tableau résultat 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 :

	A1	A2	An
R1			
R2			
Rk			
	a_1	a_2	a_n

motif dans J

$\Updownarrow$

$t \in \pi_{S1}(J) \bowtie \ldots \bowtie \pi_{Sk}(J)$

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

$\Rightarrow$ pour tout J valide, $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie \ldots \bowtie \pi_{Sk}(J) \subseteq J \Rightarrow$ 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 :

	A1	A2	An
R1			
R2			
Rk			
	a_1	a_2	a_n

motif dans J

$\Updownarrow$

$t \in \pi_{S1}(J) \bowtie \ldots \bowtie \pi_{Sk}(J)$

Si l'algorithme dit NON : $t \in \pi_{S1}(J) \bowtie \ldots \bowtie \pi_{Sk}(J)$ 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 $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie \ldots \bowtie \pi_{Sk}(J) \not\subseteq J \Rightarrow$ perte d'info.

Rappel : Propriétés d’une décomposition

- On ne peut pas décomposer arbitrairement

- Conditions pour une décomposition “raisonnable” :

  - Décomposition sans perte d’information

  - Décomposition sans perte de dépendances fonctionnelles

Pourquoi préserver les DFs?

- Rappel : les DFs sur un schéma sont des contraintes d'intégrité
- Le SGBD est censé vérifier que les contraintes ne sont pas violées par les mises à jour

R (Ville, Rue, Numero, CP)

INSERT into R

values ('Paris', 'rue Monge', 3, 75005)

F = {

Ville, Rue, Numero $\rightarrow$ CP

CP $\rightarrow$ Ville }

INSERT into R

values ('Paris', 'rue Monge', 3, 75013) ⟵ refusé

violation de : Ville, Rue, Numero $\rightarrow$ CP

Si on décompose R en

{ Code_Postal(CP, Ville) , Adresse (Rue, Numero, CP) }

Vérification des contraintes suite à une mise à jour :

- CP $\rightarrow$ Ville : sur le contenu de la table Code_Postal
- Ville, Rue, Numero $\rightarrow$ CP : une jointure Code_Postal $\bowtie$ Adresse est nécessaire

Pourquoi préserver les DFs?

- Rappel : les DFs sur un schéma sont des contraintes d'intégrité
- Le SGBD est censé vérifier que les contraintes ne sont pas violées per les mises à jour

- Il est souhaitable de pouvoir vérifier les dépendances localement (jointure couteuse)

⇒ On cherche une décomposition sans perte de dépendances fonctionnelles

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

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

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

- DFs locales pour *Vendeur*: V# → Vnom Vville
- DFs locales pour *Produit*: P# → Pnom Pville
- DFs locales pour *Fourniture*: V# P# → Qte

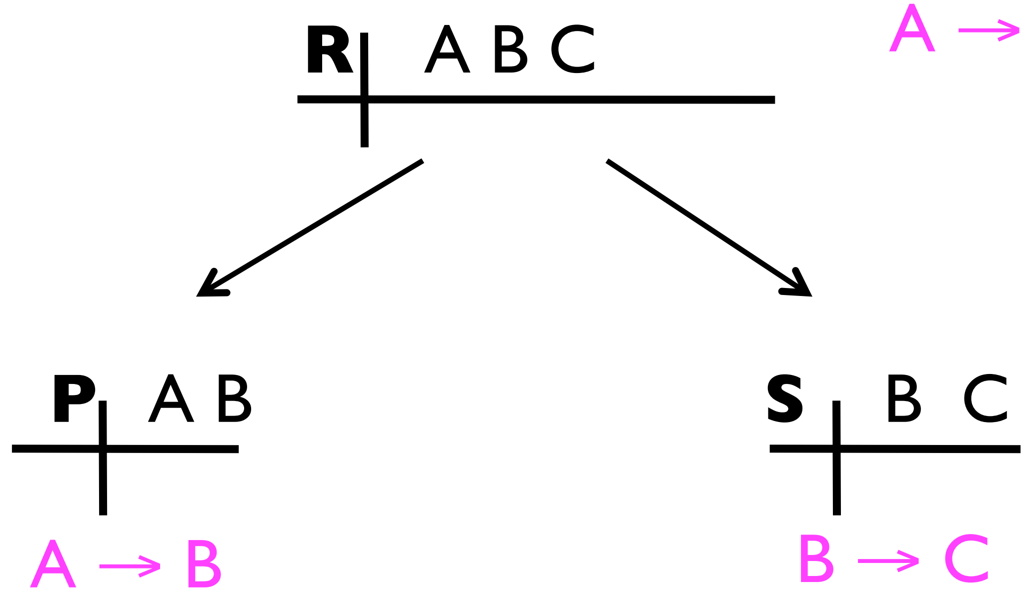
Rien n'est perdu : les DF locales suffisent pour vérifier toutes les DFs d'origine

Décomposition sans perte de DF

Pas toujours aussi simple de vérifier que les DF sont préservées

Exemple:

$A \rightarrow B,\ A \rightarrow C,\ B \rightarrow C$



DF locales : $A \rightarrow B,\ B \rightarrow C$ ne coïncident pas avec les DF d'origine

Mais elles sont suffisantes pour vérifier les DF d'origine, parce que

$$A \rightarrow B,\ B \rightarrow C \models A \rightarrow C$$

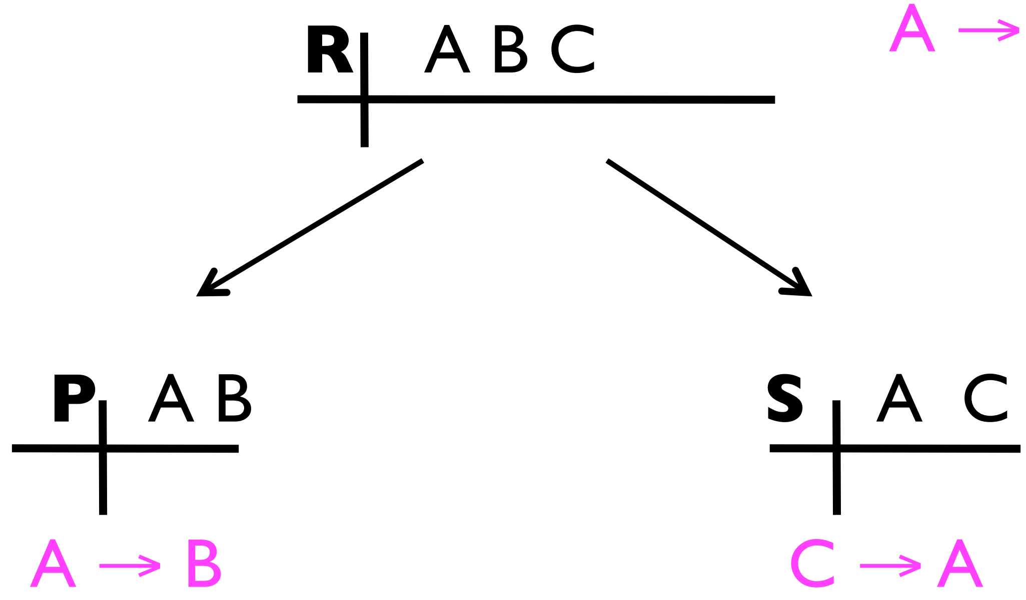
Cette décomposition est donc sans perte de dépendances fonctionnelles

Décomposition sans perte de DF

Pas toujours aussi simple de trouver les DF locales :

Exemple:

$A \to B,\ B \to C,\ C \to A$

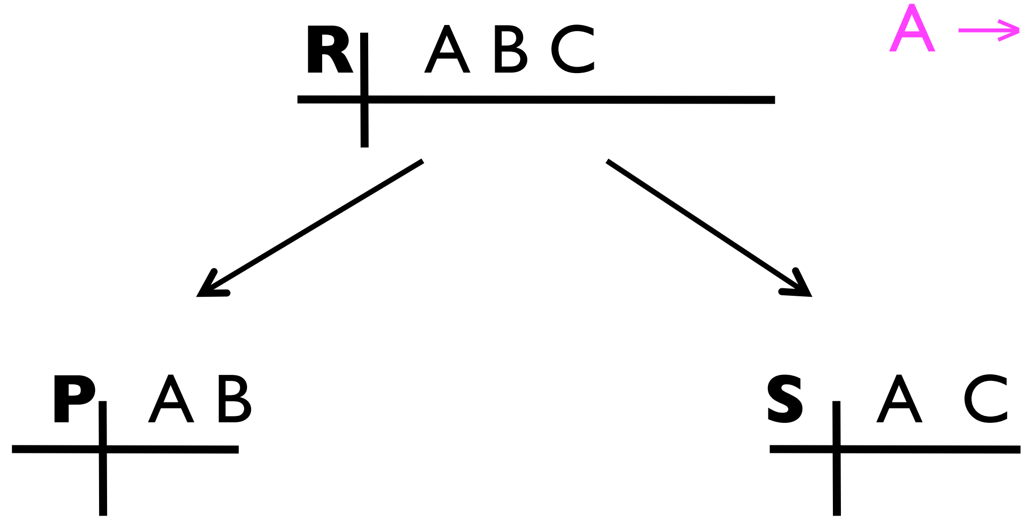


Décomposition sans perte de DF

Pas toujours aussi simple de trouver les DF locales :

Exemple:

$A \rightarrow B,\ B \rightarrow C,\ C \rightarrow A$



$A \rightarrow B$ $\quad$ $C \rightarrow A$

$B \rightarrow A$ $\quad$ $A \rightarrow C$

$B \rightarrow A$ et $A \rightarrow C$ impliquées par $A \rightarrow B, B \rightarrow C, C \rightarrow A$

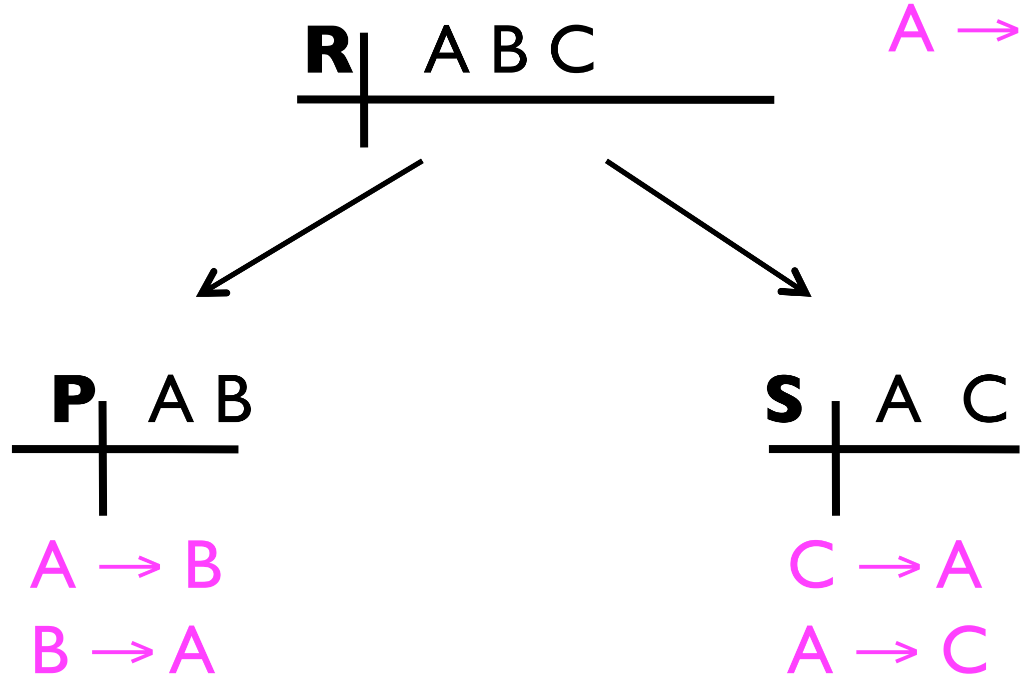
Question : cette décomposition préserve-t-elle toutes les DF d'origine?

Décomposition sans perte de DF

Pas toujours aussi simple de trouver les DF locales :

Exemple:

$A \rightarrow B,\ B \rightarrow C,\ C \rightarrow A$



$B \rightarrow A$ et $A \rightarrow C$ impliqués par $A \rightarrow B, B \rightarrow C, C \rightarrow A$

Question : cette décomposition préserve-t-elle toutes les DF d'origine?

OUI : $B \rightarrow A\ , A \rightarrow C \models B \rightarrow C$

Décomposition : DF locales

Définition de DF "locales"

si F est un ensemble de DF sur R(U) et $X \subseteq U$ alors

$$\pi_X(F^+) = \{ V \to W \mid F \models V \to W \text{ et } V, W \subseteq X \}$$

I.e : les DF impliquées par F qui s'appliquent à l'ensemble d'attributs X ("locales" à X)

Caractérisation :

Pour $V \subseteq X$ $\quad V \to W \in \pi_X(F^+)$ ssi $W \subseteq V^+ \cap X$

Dans l'exemple précédent :

R(ABC) $\quad F = \{ A \to B,\ B \to C,\ C \to A \}$ $\quad \pi_{AB}(F^+) = \{ A \to B,\ B \to A \}$

(DF triviales pas représentées)

$B \to A \in \pi_{AB}(F^+)$ puisque $A \in B^+ \cap AB$

Décomposition : DF locales

Calcul des DF locales $\pi_X(F^+)$: (on calcule un ensemble équivalent à $\pi_X(F^+)$)

$P_X := \{ \}$

pour tout $Y \subset X$ non vide

  $Z := Y^+ \cap X$

  ajouter $Y \rightarrow Z$ à P_X

retourner P_X

Très couteux! (considère tous les sous-ensembles de X, exponentiel)

En pratique on évitera de calculer explicitement $\pi_X(F^+)$ sauf si X est de petite taille

Décomposition : DF locales

Example. R(ABCDE) $F = \{ A \to C,\ BC \to D,\ AD \to E \}$

Les DF dans F^+ qui sont

- locales à AC: $\{ A \to C \}$ $A^+ = AC$ $C^+ = C$
- locales à ABD: $\{ AB \to D \}$ $A^+ = AC$ $B^+ = B$ $D^+ = D$
  $AB^+ = ABCDE$ $AD^+ = ADCE$ $BD^+ = BD$
- locales à ABCE: $\{ A \to C,\ AB \to CE,\ AE \to C,$
  $ABC \to E,\ ABE \to C \}$

$A^+ = AC$ $B^+ = B$ $C^+ = C$ $E^+ = E$

$AB^+ = ABCDE$, $AC^+ = AC$, $AE^+ = AEC$, $BC^+ = BCD$, $BE^+ = BE$, $CE^+ = CE$

$ABC^+ = ABCDE$ $ABE^+ = ABECD$ $BCE^+ = BCED$ $ACE^+ = ACE$

Calcul "à la main" raisonnable uniquement pour max 3 ou 4 attributs locaux!!!

Décomposition sans perte de DF

Définition:

Soit ρ = ($R_1(S_1)$, …, $R_k(S_k)$) une décomposition pour R, et soit F un ensemble de DF sur R. Alors ρ préserve F ssi

L'ensemble des DF locales $\bigcup_{i=1}^{k} \pi_{S_i}(F^+)$ est équivalent à F

En d'autres termes, toutes les DF dans F sont impliquées par les DF locales

Une extension de l'algorithme de clôture permet de tester si une décomposition est sans perte de dépendances fonctionnelles (sans devoir calculer les $\pi_{S_i}(F^+)$)

Tester la préservation des dépendances

- **Méthode naïve :**
  1. Calculer $G = \cup^k_{i=1}\ \pi_{Si}\,(F^+)$ % DF locales
  2. Tester $G \vDash F$

  Inconvénient : pas pratique, la taille de G peut être exponentielle dans celle de F
- **Méthode améliorée:**

pour tout $X \rightarrow Y$ dans F % on teste si $G \vDash X \rightarrow Y$ (sans calculer G !)

 $Z := X$ % Z contiendra à terme la clôture de X par rapport à G

 tant que Z change

  pour tout i := 1 à k faire

   $Z := Z \cup (\,(Z \cap S_i)^+ \cap S_i\,)$ % la clôture de Z par rapport au DF locales à Si

  fin pour tout

 fin tant que

 Si $Y \not\subseteq Z$ renvoyer "NON" et arrêter % Y n'est pas dans la clôture de X par
 % rapport à G donc $X \rightarrow Y$ n'est pas impliquée par G

fin pour tout

renvoyer "OUI" (toutes les DF sont préservées)

($^+$ est par rapport à F)

Exemple

$R(A\ B\ C\ D)$ $\rho = \{ R_1(AB) \quad R_2(BC) \quad R_3(CD) \}$

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

Est-ce que ρ préserve F? i.e. est-ce que l'ensemble des DF locales implique F ?

- Les DF locales impliquent $A \to B$, $B \to C$, $C \to D$ (elle sont DF locales).
- Est-ce que les DF locales impliquent $D \to A$?

Décomposition: AB BC CD

Commencer avec $Z=D$

$(D \cap CD)^+ \cap CD = D^+ \cap CD = DABC \cap CD$, ajouter C à Z ($D \to C$ est locale à CD)

$(DC \cap BC)^+ \cap BC = C^+ \cap BC = CDAB \cap BC$, ajouter B à Z ($C \to B$ est locale à BC)

$(DCB \cap AB)^+ \cap AB = B^+ \cap AB = BCDA \cap AB$, ajouter A à Z ($B \to A$ est locale à AB)

⇧

succès !

$D \to A$ est préservée

Décomposition FNBC

Soit R(U) un schéma de relation et F un ensemble de DF sur R(U)

Si R n'est pas en forme normale de Boyce-Codd par rapport à F

on cherche une décomposition $\{ R_1(S_1), \ldots, R_k(S_k) \}$ de R(U)

telle que chaque Ri est en FNBC par rapport à ses DF locales $\pi_{S_i}(F^+)$

➡ Idéalement sans perte d'information ni de DF.

On verra que

- il est toujours possible de trouver un décomposition FNBC de R(U) sans perte d'information
- mais pas toujours possible d'en trouver une sans perte de DF
- une décomposition dans une autre forme normale sera alors cherchée

Décomposition FNBC sans perte d'information

- Chaque schéma de relation a une décomposition en un ensemble de schémas de relations FNBC sans perte d'information

cette décomposition ne préserve pas toujours les DF

- **Exemple**

R (Ville, CP, Rue, Numero)

F = Ville, Rue, Numero $\rightarrow$ CP, CP $\rightarrow$ Ville

- R n'est pas en FNBC : CP n'est pas une super-clef
- R n'a pas de décomposition en schémas FNBC, qui préserve F :
  - Soit ρ une décomposition qui ne contient pas R, aucune relation de ρ ne peut avoir une DF locale de la forme X $\rightarrow$ CP (sinon Ville, Rue, Numero $\subseteq$ X)
  - $\Rightarrow$ les DFs locales n'impliquent pas Ville, Rue, Numero $\rightarrow$ CP

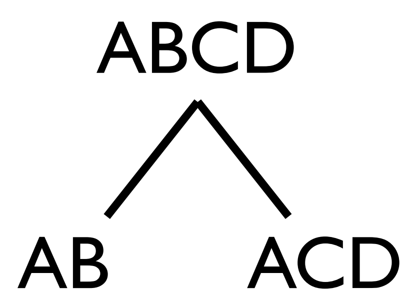
Algorithme de décomposition FNBC de R par rapport à F sans perte d'information

<u>Étape de base (exemple):</u>

- Supposer qu'on a obtenu une décomposition de R, contenant P(S), P pas en FNBC
- Supposer : S = ABCD et une seule DF locale : $A \rightarrow B$

(violation de FNBC : A n'est pas une super-clef pour P)

étape de décomposition pour éliminer la violation



cette étape est sans perte d'information:
appliquer le test en utilisant la DF $A \rightarrow B$

A	B	C	D
a	b	z_1	z_3
a	z_2	c	d
a	b	c	d

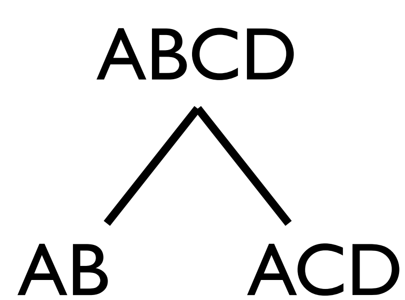
Algorithme de décomposition FNBC de R par rapport à F sans perte d'information

<u>Étape de base (exemple):</u>

- Supposer qu'on a obtenu une décomposition de R, contenant P(S), P pas en FNBC
- Supposer : S = ABCD et une seule DF locale : $A \rightarrow B$

(violation de FNBC : A n'est pas une super-clef pour Ri)

étape de décomposition pour éliminer la violation



cette étape est sans perte d'information:
appliquer le test en utilisant la DF $A \rightarrow B$

A	B	C	D
a	b	z_1	z_3
a	b	c	d
a	b	c	d

un seul coup suffit

Algorithme de décomposition FNBC de R par rapport à F sans perte d'information

Commencer avec une décomposition $\rho = \{ R(U) \}$

Tant que ρ contient un schéma de relation P(S) qui n'est pas en FNBC

soit $X \rightarrow A$ une violation de FNBC dans P

dans ρ remplacer P(S) par $P_1(V)$ et $P_2(W)$ où

$V = XA$

$W = X(S - V)$

Remarque. une telle décomposition est toujours sans perte d'information (lossless join)

S

XA X (S-XA)

en utilisant :

$X \rightarrow A \in F^+$

X	A	S-XA
$a_1 \ldots a_{k-1}$	a_k	$z_{k+1} \ldots z_n$
$a_1 \ldots a_{k-1}$	z_0	$a_{k+1} \ldots a_n$
$a_1 \ldots$		$\ldots a_n$

Algorithme de décomposition FNBC de R par rapport à F sans perte d'information

Commencer avec une décomposition ρ = { R(U) }

Tant que ρ contient un schéma de relation P(S) qui n'est pas en FNBC

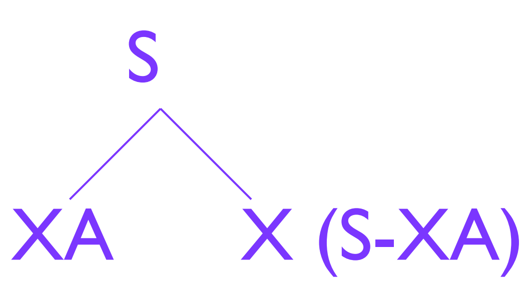
soit $X \rightarrow A$ une violation de FNBC dans P

dans ρ remplacer P(S) par $P_1(V)$ et $P_2(W)$ où

V = XA

W = X(S – V)

Remarque. une telle décomposition est toujours sans perte d'information (lossless join)



en utilisant :

$X \rightarrow A \in F^+$

X	A	S-XA
$a_1 \ldots a_{k-1}$	a_k	$z_{k+1} \ldots . z_n$
$a_1 \ldots a_{k-1}$	a_k	$a_{k+1} \ldots . a_n$
$a_1 \ldots$		$\ldots a_n$

Algorithme de décomposition FNBC de R par rapport à F sans perte d'information

Commencer avec une décomposition $\rho = \{ R(U) \}$

Tant que ρ contient un schéma de relation P(S) qui n'est pas en FNBC

soit $X \subseteq S$ t.q. $X \subsetneq X^+ \cap S \subsetneq S$ et soit $A \in X^+ \cap S$, $A \notin X$ } i.e. $X \rightarrow A$ violation de FNBC dans P

dans ρ remplacer P(S) par $P_1(V)$ et $P_2(W)$ où

$V = XA$

$W = X(S - V)$

- Preuve de l'equivalence :

il existe $X \rightarrow A$ violation de FNBC dans P

ssi

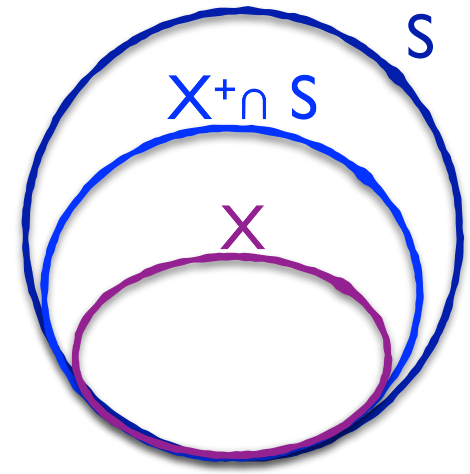
$\exists X,A$ $\underbrace{X \rightarrow A \text{ est une DF locale à P}}_{\text{ssi}}$, $\underbrace{X \rightarrow A \text{ n'est pas triviale}}_{\text{ssi}}$, $\underbrace{X \text{ n'est pas superclef pour P}}_{\text{ssi}}$

$\exists X,A$ $\underbrace{A \in X^+ \cap S \qquad A \notin X}_{\text{ssi}}$ $\qquad X^+ \cap S \subsetneq S$

$X \subsetneq X^+ \cap S$

Algorithme de décomposition FNBC

Chercher une violation de FNBC dans P(S) en pratique:



- Chercher un sous-ensemble X de S tel que

  - X^+ contient un attribut A de S pas dans X

  - X^+ ne contient pas tous les attributs de S

- Couteux de tester tous les sous-ensembles X de S, mais quelques astuces :

  - vérifier uniquement les sous-ensembles qui contiennent la partie gauche d'une DF de F

  - pas besoin de vérifier les sous-ensembles de S de taille |S|-1 ou |S|

    (Si X est de taille |S|-1 : soit $X^+ \cap S = S$, soit $X^+ \cap S = X$

    Si X est de taille |S| : $X^+ \cap S = X = S$)

  - si S est de taille 2 il n'y a pas de violations

Algorithme de décomposition FNBC - Exemple

R = C T H S E N (Présences aux séances de cours et notes)

C = cours

T = enseignant

H = horaire

S = salle

E = étudiant

N = note

F: $C \rightarrow T$

$HS \rightarrow C$

$HT \rightarrow S$

$CE \rightarrow N$

$HE \rightarrow S$

Algorithme de décomposition FNBC - Exemple

R = C T H S E N (Présences aux séances de cours et notes)

C = cours

T = enseignant

H = horaire

S = salle

E = étudiant

N = note

Quelles sont les clefs ?

F: $C \rightarrow T$

$HS \rightarrow C$

$HT \rightarrow S$

$CE \rightarrow N$

$HE \rightarrow S$

Algorithme de décomposition FNBC - Exemple

R = C T H S E N (Présences aux séances de cours et notes)

C = cours

T = enseignant

H = horaire

S = salle

E = étudiant

N = note

Quelles sont les clefs ?

une seule : HE

F: $C \rightarrow T$

$HS \rightarrow C$

$HT \rightarrow S$

$CE \rightarrow N$

$HE \rightarrow S$

Algorithme de décomposition FNBC - Exemple

R = C T H S E N (Présences aux séances de cours et notes)

C = cours

T = enseignant

H = horaire

S = salle

E = étudiant

N = note

Quelles sont les clefs ?

une seule : HE

F: $C \rightarrow T$

$HS \rightarrow C$

$HT \rightarrow S$

$CE \rightarrow N$

$HE \rightarrow S$

- HE super-clef : $HE^+ = HESCTN$

- HE la seule super-clef minimale :

H et E doivent être dans toutes les super-clefs,
puisque il ne sont pas déterminés par d'autres attributs

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

$\boxed{CTHSEN}$

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

$\boxed{CTHSEN}$

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

CEN CETHS

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

CEN

pas de violation:
$C^+ = CT$, $E^+ = E$, $N^+ = N$

CETHS

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

CEN

pas de violation:

$C^+ = CT$, $E^+ = E$, $N^+ = N$

CETHS

violation: $C \rightarrow T$ $C^+ \cap CETHS = CT$

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

CEN

pas de violation:

$C^+ = CT$, $E^+ = E$, $N^+ = N$

CETHS

violation: $C \rightarrow T$ $C^+ \cap CETHS = CT$

CT

CHSE

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

CEN

pas de violation:

$C^+ = CT$, $E^+ = E$, $N^+ = N$

CETHS

violation: $C \rightarrow T$ $C^+ \cap CETHS = CT$

CT

pas de violation:

2 attributs

CHSE

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

$\boxed{CTHSEN}$

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

$\boxed{CEN}$

pas de violation:

$C^+ = CT$, $E^+ = E$, $N^+ = N$

$\boxed{CETHS}$

violation: $C \rightarrow T$ $C^+ \cap CETHS = CT$

$\boxed{CT}$

pas de violation:

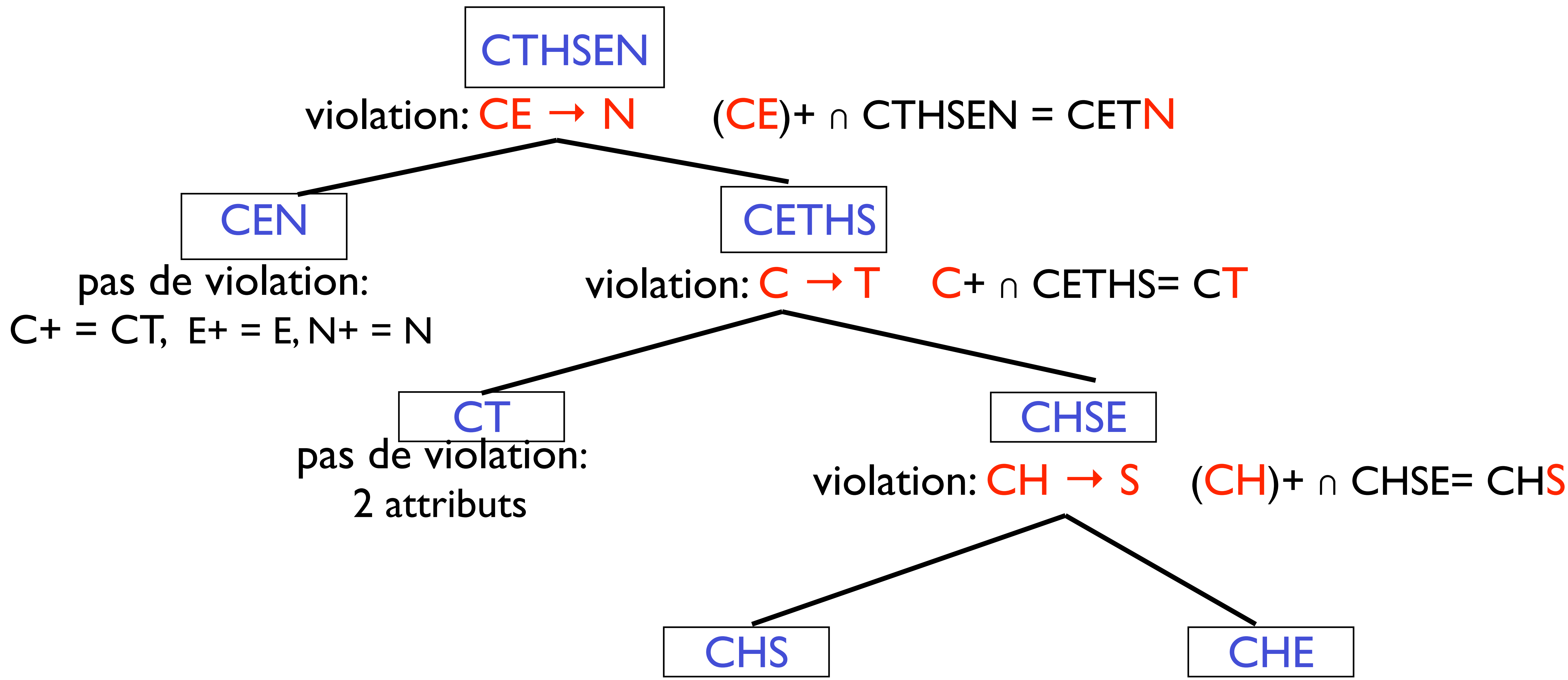
2 attributs

$\boxed{CHSE}$

violation: $CH \rightarrow S$ $(CH)^+ \cap CHSE = CHS$

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$



Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

CEN

pas de violation:
$C^+ = CT$, $E^+ = E$, $N^+ = N$

CETHS

violation: $C \rightarrow T$ $C^+ \cap CETHS = CT$

CT

pas de violation:
2 attributs

CHSE

violation: $CH \rightarrow S$ $(CH)^+ \cap CHSE = CHS$

CHS

CHE

pas de violation: $C^+ = CT$, $H^+ = H$, $S^+ = S$, $E^+ = E$

Algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: $CE \rightarrow N$ $(CE)^+ \cap CTHSEN = CETN$

CEN

pas de violation:
$C^+ = CT$, $E^+ = E$, $N^+ = N$

CETHS

violation: $C \rightarrow T$ $C^+ \cap CETHS = CT$

CT

pas de violation:
2 attributs

CHSE

violation: $CH \rightarrow S$ $(CH)^+ \cap CHSE = CHS$

CHS

CHE

pas de violation: $C^+ = CT$, $H^+ = H$, $S^+ = S$, $E^+ = E$

Décomposition obtenue: CEN, CT, CHS, CHE

Correction de l'algorithme de décomposition FNBC

Correction de l'algorithme de décomposition FNBC

1) l' algorithme termine :

- à chaque étape S est décomposé en V et W, qui sont strictement inclus dans S :

  - $V = XA \subseteq X^+$ alors que $S \not\subseteq X^+$ $\Rightarrow$ V ne peut pas être égale à S

  - $W = X(S - XA) = S\text{-}A \subsetneq S$

- $\Rightarrow$ si l'algorithme n'a pas terminé avant, après un nombre fini d'étapes toutes les relations de la décomposition auront au plus deux attributs

- si cela arrive, la décomposition est FNBC et l'algorithme termine

Correction de l'algorithme de décomposition FNBC

2) La décomposition obtenue est FNBC

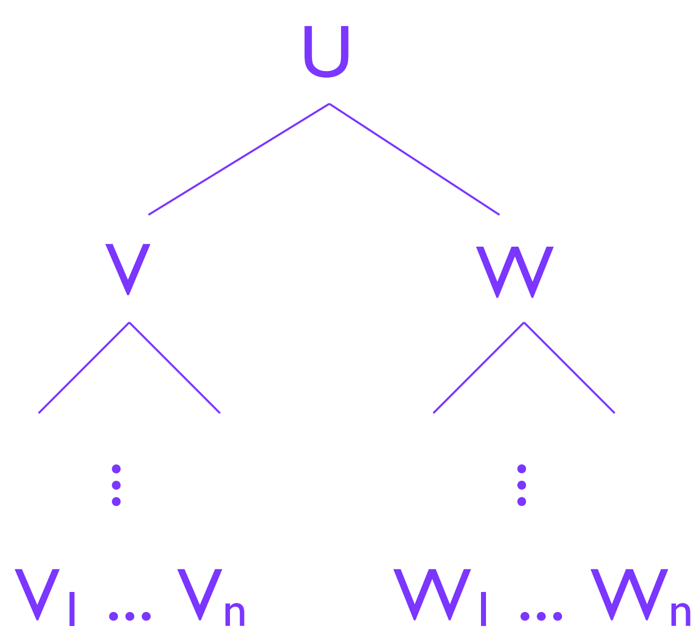
puisque c'est la condition de terminaison de l'algorithme

3) La décomposition obtenue est sans perte d'information

Intuitivement, puisque chaque étape l'est.

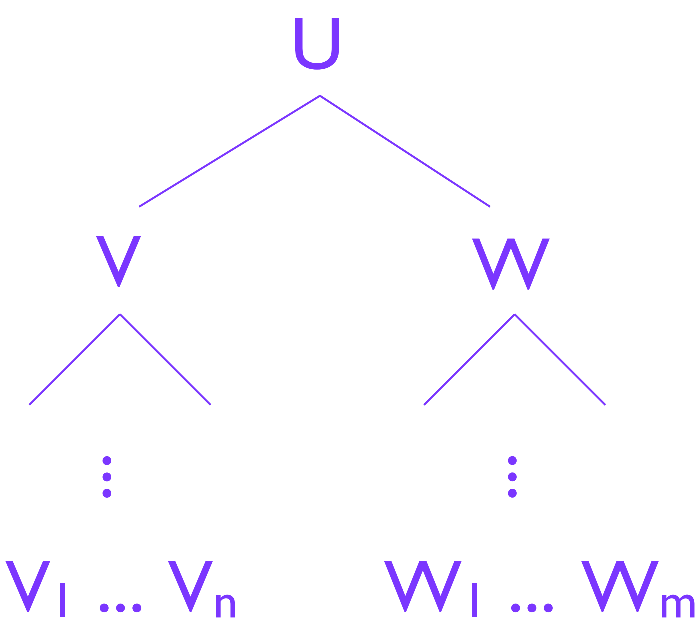
Plus précisément : par induction sur le nombre d'étapes

- S'il y a une seule étape, elle est *lossless join*
- S'il y a plusieurs étapes :



Correction de l'algorithme de décomposition FNBC

Supposer par induction $V_1..V_n$ *lossless-join* pour V et $W_1..W_m$ *lossless-join* pour W (par rapport aux DF locales respectives)



Pour tout instance J avec attributs U qui satisfait F

$\pi_V(J)$ (resp. $\pi_W(J)$) satisfait les DF locales à V (resp. à W)

alors $\pi_V(J) = \pi_{V_1}(\pi_V(J)) \bowtie \ldots \bowtie \pi_{V_n}(\pi_V(J)) =$

$= \pi_{V_1}(J) \bowtie \ldots \bowtie \pi_{V_n}(J)$

et de façon similaire :

$\pi_W(J) = \pi_{W_1}(J) \bowtie \ldots \bowtie \pi_{W_m}(J)$

V, W est une décomposition de U *lossless-join* par rapport à F alors

$J = \pi_V(J) \bowtie \pi_W(J) = \pi_{V_1}(J) \bowtie \ldots \bowtie \pi_{V_n}(J) \bowtie \pi_{W_1}(J) \bowtie \ldots \bowtie \pi_{W_n}(J)$

$\Rightarrow V_1..V_n\ W_1...W_m$ est une décomposition de U *lossless-join* par rapport à F

Variante de l'algorithme de décomposition FNBC

La correction de l'algorithme est préservée si on le modifie comme suit :

Commencer avec une décomposition $\rho = \{ R(U) \}$

Tant que ρ contient un schéma de relation $P(S)$ qui n'est pas en FNBC

soit $X \subseteq S$ t.q. $X \subsetneq X^+ \cap S \subsetneq S$ } violation de FNBC dans P

dans ρ remplacer $P(S)$ par $P_1(V)$ et $P_2(W)$ où

$V = X^+ \cap S$

$W = X(S - V)$

Ce qui permet parfois d'obtenir moins d'étapes de décomposition

Préférable !

Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

$\boxed{CTHSEN}$

Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

$\boxed{CTHSEN}$

violation: CE $(CE)+ = CETN$

Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: CE (CE)+ = CETN

CETN CEHS

Variante de l’algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

$\boxed{\text{CTHSEN}}$

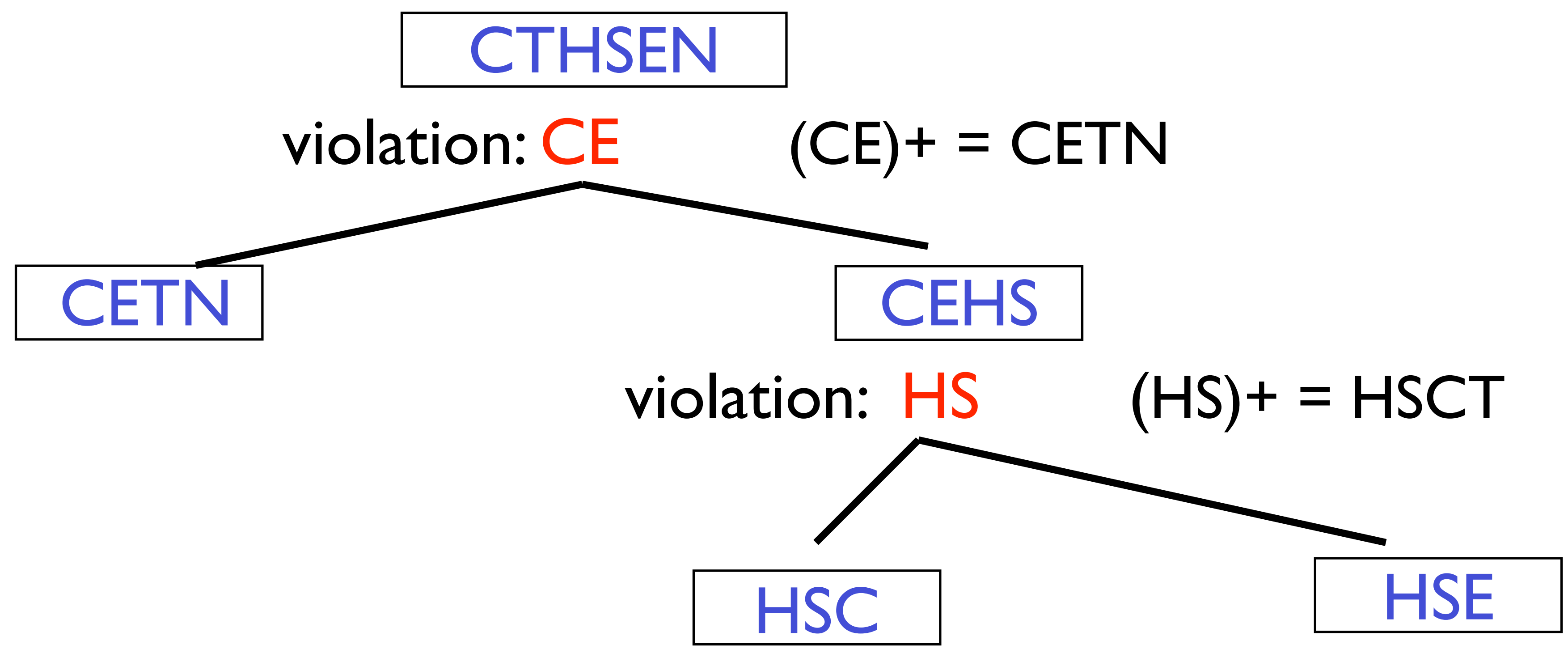
violation: CE (CE)+ = CETN

$\boxed{\text{CETN}}$ $\boxed{\text{CEHS}}$

violation: HS (HS)+ = HSCT

Variante de l'algorithme de décomposition FNBC - Exemple

DF: C→T, CE →N, HS →C, HE →S, TH →S



Variante de l'algorithme de décomposition FNBC - Exemple

DF: C→T, CE →N, HS →C, HE →S, TH →S

CTHSEN

violation: CE (CE)+ = CETN

CETN

CEHS

violation: HS (HS)+ = HSCT

HSC

HSE

pas de violation:
C+ = CT, H+ = H,
S+ = S

Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: CE (CE)+ = CETN

- CETN
- CEHS

  violation: HS (HS)+ = HSCT

  - HSC

    pas de violation:
    C+ = CT, H+ = H,
    S+ = S

  - HSE

    pas de violation:
    H+ = H, S+ = S,
    E+ = E

Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \to T$, $CE \to N$, $HS \to C$, $HE \to S$, $TH \to S$

- **CTHSEN**
  violation: CE (CE)+ = CETN
  - **CETN**
    violation: C
    (C)+ = CT
  - **CEHS**
    violation: HS (HS)+ = HSCT
    - **HSC**
      pas de violation:
      C+ = CT, H+ = H,
      S+ = S
    - **HSE**
      pas de violation:
      H+ = H, S+ = S,
      E+ = E

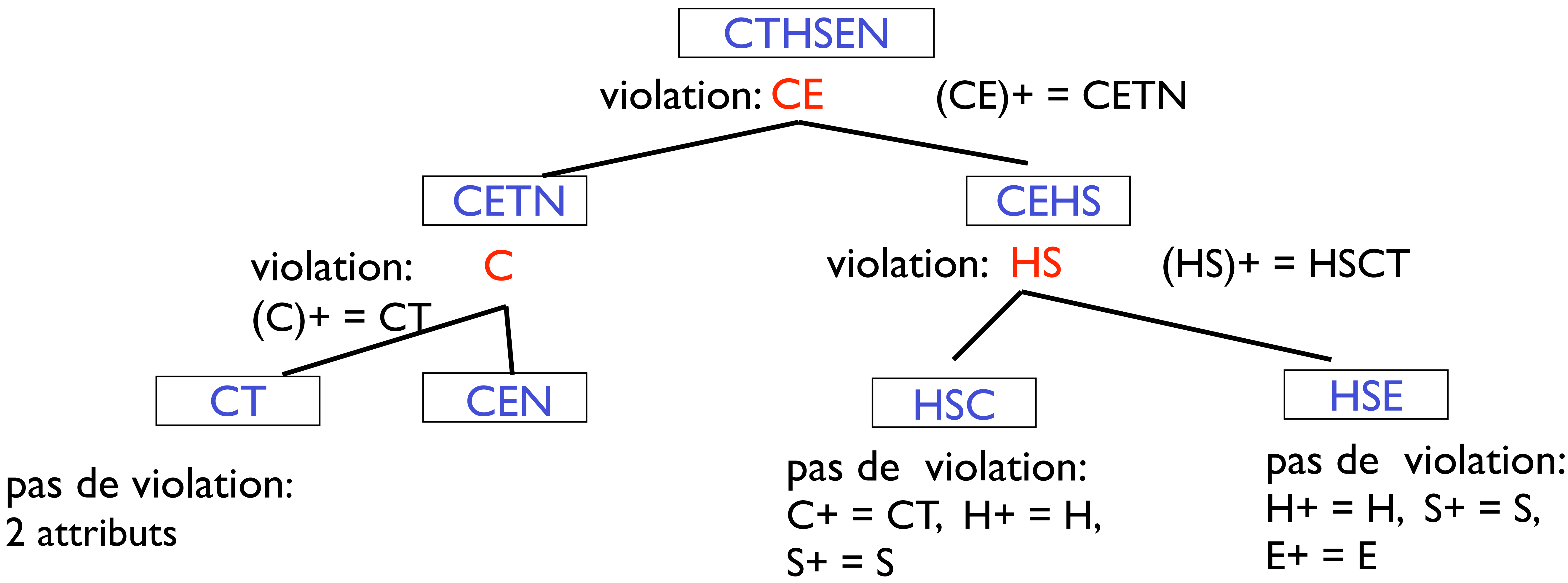
Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

- CTHSEN
  violation: CE (CE)+ = CETN
  - CETN
    violation: C
    (C)+ = CT
    - CT
    - CEN
  - CEHS
    violation: HS (HS)+ = HSCT
    - HSC
      pas de violation:
      C+ = CT, H+ = H,
      S+ = S
    - HSE
      pas de violation:
      H+ = H, S+ = S,
      E+ = E

Variante de l'algorithme de décomposition FNBC - Exemple

DF: C→T, CE →N, HS →C, HE →S, TH →S



Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: CE (CE)+ = CETN

CETN

violation: C

(C)+ = CT

CEHS

violation: HS (HS)+ = HSCT

CT

pas de violation:
2 attributs

CEN

pas de violation:
C+ = CT, E+ = E,
N+ = N

HSC

pas de violation:
C+ = CT, H+ = H,
S+ = S

HSE

pas de violation:
H+ = H, S+ = S,
E+ = E

Variante de l'algorithme de décomposition FNBC - Exemple

DF: $C \rightarrow T$, $CE \rightarrow N$, $HS \rightarrow C$, $HE \rightarrow S$, $TH \rightarrow S$

CTHSEN

violation: CE (CE)+ = CETN

CETN

violation: C
(C)+ = CT

CT

pas de violation:
2 attributs

CEN

pas de violation:
C+ = CT, E+ = E,
N+ = N

CEHS

violation: HS (HS)+ = HSCT

HSC

pas de violation:
C+ = CT, H+ = H,
S+ = S

HSE

pas de violation:
H+ = H, S+ = S,
E+ = E

Décomposition obtenue: CEN, CT, CHS, HSE

Inconvénients de la décomposition FNBC (les deux variantes)

- La décomposition n'est pas unique (elle dépend du choix de X à chaque étape)
  - Un choix plutôt qu'un autre affecte la qualité de la décomposition (la taille de la décomposition, ou la préservation de DF)
  - Dans l'ex. : CHSE peut être décomposé en { CHS, CHE } ou { CHS, HES }
- La décomposition obtenue peut ne pas préserver toutes les DF

Exemple la décomposition FNBC obtenue dans le premier cas:

CEN CT CHS CHE

ne préserve pas $HT \rightarrow S$:

- exécutons l'algorithme pour tester la préservation des DF sur

$$F = C \rightarrow T,\quad CE \rightarrow N,\quad HS \rightarrow C,\quad HE \rightarrow S,\quad HT \rightarrow S$$

- la clôture locale de HT est HT :

$$(HT \cap CEN)^+ = \varnothing \qquad (HT \cap CT)^+ = T \quad (HT \cap CHS)^+ = H \quad (HT \cap CHE)^+ = H$$

- donc $HT \rightarrow S$ n'est pas préservée
- $HT \rightarrow S$ serait perdu également si on avait HES à la place de CHE

Efficacité de l'algorithme de décomposition FNBC

Existe-t-il un algorithme efficace pour la décomposition FNBC ?

Très probablement NON:

Décider si un schéma de relation R est en FNBC par rapport à un ensemble de DF est NP-complet

$$\Downarrow$$

Tout algorithme de décomposition FNBC, qui a la propriété de ne jamais décomposer une relation déjà en FNBC, sera très probablement exponentiel

Troisième Forme Normale (3NF)

Problème avec FNBC:

Tous les schémas de relation ne peuvent pas être décomposés en un ensemble de schémas FNBC qui préservent *à la fois* l'information *et* les dépendances fonctionnelles

voir exemple R(Ville, Rue, Numero, CP)

⇓

Troisième forme normale (3NF)

Un schéma de relation R est en troisième forme normale par rapport à un ensemble F de DFs sur R si pour tout $X \rightarrow A \in F^+$ tel que $A \notin X$

soit X est-une super-clef de R **soit** A **appartient à une clef de** R

3NF est plus "faible" que FNBC :

FNBC implique 3NF mais pas vice-versa

3NF admet donc une certaine forme de redondance et des anomalies, mais considérées acceptables

Troisième Forme Normale - Exemple

Exemple

R (Ville, CP, Rue, Numero)

F = Ville Rue Numero $\rightarrow$ CP, CP $\rightarrow$ Ville

Violation de FNBC: CP $\rightarrow$ Ville

Néanmoins, *Ville* appartient à la **clef** *Ville Rue Numero*

donc R est en 3NF par rapport à F

R est en 3NF mais pas en FNBC

Attention : l'attribut à droite doit appartenir à une clef (donc une superclef minimale, **attention à la minimalité**!)

Décomposition 3NF sans perte d'information et sans perte de DF

Deux étapes

- Simplifier l'ensemble de DFs (éliminer les redondances) - cf. prochains transparents

- Construire la décomposition 3NF à partir des DF restantes - cf. prochains transparents

On obtient une décomposition qui préserve les DF et est sans perte d'information

Eliminer les redondances dans les DF

- Réécrire les DF avec un seul attribut sur les partie droites

  ex: remplacer $AB \rightarrow CD$ par $AB \rightarrow C$ et $AB \rightarrow D$

- **Répéter**

  1. S'il existe une DF redondante (i.e impliquée par les autres) l'éliminer

  ex: $F = \{A \rightarrow C,\ A \rightarrow B,\ B \rightarrow C\}$ on obtient $\{A \rightarrow B,\ B \rightarrow C\}$

  $A \rightarrow C$ est redondante (elle est impliquée par $A \rightarrow B$ et $B \rightarrow C$)

  2. S'il existe un attribut redondant dans une partie gauche d'une DF, réduire cette partie gauche en éliminant cet attribut

  ex: $F = \{AD \rightarrow B,\ ABD \rightarrow C\}$ on obtient $\{ AD \rightarrow B,\ AD \rightarrow C \}$

  B est redondant dans $ABD \rightarrow C$ puisque $AD \rightarrow C$ est impliqué par F (et donc AD tout seul détermine C)

  Jusqu'à ce que les DFs ne changent plus (couverture minimale de F)

Remarque. On peut démontrer qu'il suffit d'éliminer d'abord toutes les redondances de type 2., puis toutes les redondances de type 1. Mais pas vice-versa.

Décomposition 3NF

R un schéma de relation

F un ensemble minimal de DF sur R (- chaque partie droite = attribut simple,

- pas de redondances, ni de type 1. ni de type 2.)

Décomposition ρ en 3NF:

a) un schéma de relation avec attributs XA pour chaque $X \rightarrow A \in F$

b) Si aucun des schémas de relation de l'étape a) n'est une super-clef pour R, ajouter un autre schéma de relation qui a pour attributs une **clef de R**.

- Exemple
  - R = CTHSEN (voir exemple précèdent)
  - F = $C \rightarrow T$ $CE \rightarrow N$ $HS \rightarrow C$ $HE \rightarrow S$ $HT \rightarrow S$ (minimal)
  - Alors ρ = { CT, CEN, HSC, HES , HTS } (HES super-clef)

Un autre exemple de décomposition 3NF

- R (ABCDE) F = { $A \to C$, $BC \to D$ $AD \to E$ } (minimal)

- décomposition initiale : AC, BCD, ADE

- Aucune entre AC, BCD, ADE n'est une super-clef:

  $AC^+ = AC$, $BCD^+ = BCD$, $ADE^+ = ADEC$

- AB est une clef (attention à la minimalité) on l'ajoute à ρ

  ρ = { $R_1(AC)$ $R_2(BCD)$ $R_3(ADE)$ $R_4(AB)$ }

Décomposition 3NF

Théorème. la décomposition ρ de R obtenue par l'algorithme de decomposition 3NF préserve F. Chaque R_i (S_i) dans ρ est en 3NF par rapport à $\pi_{Si}(F^+)$. De plus ρ est sans perte d'information.

Décomposition 3NF

Théorème. la décomposition ρ de R obtenue par l'algorithme de decomposition 3NF préserves F. Chaque R_i (S_i) dans ρ est en 3NF par rapport à $\pi_{S_i}(F^+)$. De plus ρ est sans perte d'information.

- Pourquoi F est préservé ? chaque DF de F est trivialement locale
- Pourquoi chaque relation est en 3NF? **Idée de la preuve** :

Soit Ri(XA) dans ρ construit de $X \rightarrow A \in F$

Remarque : X est une clef de Ri (sinon $X \rightarrow A$ aurait des attributs redondants dans la partie gauche).

Soit $Y \rightarrow B$ une DF locale à XA, non-triviale. Donc $YB \subseteq XA$, $B \notin Y$. Deux cas:

1. $B \neq A$. Alors $B \in X$. Alors B appartient à une clef de Ri
2. $B = A$. Alors $Y \subseteq X$. Mais si $Y \subset X$ les attributs $X - Y$ sont redondant en $X \rightarrow A$

   Alors $Y = X$ et Y est une super-clef pour Ri

Soit Rj(K) obtenue d'une clef K de R. Alors Rj n'a pas de DF locale non-triviale (par minimalité de K)

Décomposition 3NF

- Pourquoi ρ est sans perte d'information ? **Idée de la preuve**:

La *chase* du tableau pour ρ produit une ligne de symboles a_i

dans la ligne de la clef (ou super-clef) K :

Appliquer les DF dans le même ordre utilisé pour le calcul de K^+

(Si $X \rightarrow Ai$ est utilisé dans le calcul de K^+, utiliser $X \rightarrow Ai$ entre la ligne de K et la ligne de XAi pour remplacer zi avec ai)

	A1	A2					Ai		An
...									
XAi		...					**a_i**	...	
...									
K	a_1	a_2	...	a_k	z_{k+1}	...	**z_i**	...	z_n
	a_1	a_2							a_n

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) F = { $A \rightarrow C$, $BC \rightarrow D$ $AD \rightarrow E$ }
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC) \quad R_2(BCD) \quad R_3(ADE) \quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

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

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	z_8	z_9	z_{10}
	a	b	c	d	e

Chase avec $A \rightarrow C$, $BC \rightarrow D$, $AD \rightarrow E$ (dans l'ordre)

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) $F = \{A \to C,\ BC \to D\ \ AD \to E\}$
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC)\quad R_2(BCD)\quad R_3(ADE)\quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

$A \to C$, $BC \to D$, $AD \to E$

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	z_8	z_9	z_{10}
	a	b	c	d	e

Chase avec $A \to C$

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) F = { $A \rightarrow C$, $BC \rightarrow D$ $AD \rightarrow E$ }
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC) \quad R_2(BCD) \quad R_3(ADE) \quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

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

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	c	z_9	z_{10}
	a	b	c	d	e

Chase avec $A \rightarrow C$

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) F = { $A \rightarrow C$, $BC \rightarrow D$ $AD \rightarrow E$ }
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC) \quad R_2(BCD) \quad R_3(ADE) \quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

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

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	c	z_9	z_{10}
	a	b	c	d	e

Chase avec $BC \rightarrow D$

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) F = { A→C, BC→D AD→E }
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC) \quad R_2(BCD) \quad R_3(ADE) \quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

A → C, BC → D, AD → E

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	c	**d**	z_{10}
	a	b	c	d	e

Chase avec BC → D

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) F = { $A \rightarrow C$, $BC \rightarrow D$ $AD \rightarrow E$ }
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC) \quad R_2(BCD) \quad R_3(ADE) \quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

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

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	c	d	z_{10}
	a	b	c	d	e

Chase avec $AD \rightarrow E$

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) F = { $A \rightarrow C$, $BC \rightarrow D$ $AD \rightarrow E$ }
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC) \quad R_2(BCD) \quad R_3(ADE) \quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

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

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	c	d	**e**
	a	b	c	d	e

Chase avec $AD \rightarrow E$

3NF sans perte d'information : exemple

- On prend l'exemple précèdent R (ABCDE) $F = \{ A \to C,\ BC \to D\ \ AD \to E \}$
- Et sa décomposition 3NF trouvée avec l'algorithme

$$\rho = \{ R_1(AC)\quad R_2(BCD)\quad R_3(ADE)\quad R_4(AB) \}$$

- On montre avec la *chase* que ρ est sans perte d'information

DFs utilisées dans le calcul de AB^+ (dans l'ordre) :

$A \to C$, $BC \to D$, $AD \to E$

AB ABC ABCD ABCDE

	A	B	C	D	E
AC	a	z_1	c	z_2	z_3
BCD	z_4	b	c	d	z_5
ADE	a	z_6	z_7	d	e
AB	a	b	c	d	e
	a	b	c	d	e

lossless join !

Variante de l'algorithme de decomposition 3NF

La correction de l'algorithme est préservée si on le modifie comme suit :

R un schéma de relation

F un ensemble minimal de DF sur R (- chaque partie droite = attribut simple,

- pas de redondances, ni de type 1. ni de type 2.)

F' obtenu de F en fusionnant toutes les DF avec la même partie gauche
($X \to A1, \ldots, X \to An$ devient $X \to A1 \ldots An$)

Décomposition ρ :

a) produire un schéma de relation avec attributs XY pour chaque $X \to Y \in F'$

b) Si aucun des schémas de relation de l'étape a) n'est une super-clef pour R, ajouter un autre schéma de relation qui a pour attributs une **clef de R**.

Préférable !

Ce qui permet en general d'obtenir moins de relations dans la décomposition

Amélioration de la décomposition

Une décomposition (FNBC ou 3NF) obtenue avec les algorithmes montrés peut être améliorée ensuite.

Donnée une décomposition $\rho = \{ R_1, \ldots R_k \}$ de R en 3NF (resp. FNBC) :

- Éliminer une relation Ri(Si) s'il existe une relation Rj(Sj) $Si \subseteq Sj$

  cela n'altère aucunes des propriété suivantes de ρ :

  - 3NF (resp. FNBC)
  - lossless-join
  - préservation de DF

- Fusionner deux relations Ri(Si) et Rj(Sj) en R'(Si Sj) n'altère pas

  - lossless join
  - préservation de DF

  fusionner uniquement si R' est encore en 3NF (resp. FNBC)

Rappel : une vue (simplifiée) de la modélisation de schéma relationnels avec des DF

1. - Choisir les attributs d'intérêt U et produire un schéma de relation R(U)

   - Alternative : utiliser une étape de modélisation conceptuelle (e.g E/R) et traduire en un schéma relationnel $R_1(U_1)$ ... $R_k(U_k)$.

2. Spécifier toutes les DF pour R (ou pour R_1.. R_k)

   - rappel : un schéma E/R peut exprimer des DF par les les contraintes d'identification, les associations, les contraintes de cardinalité et les contraintes externes

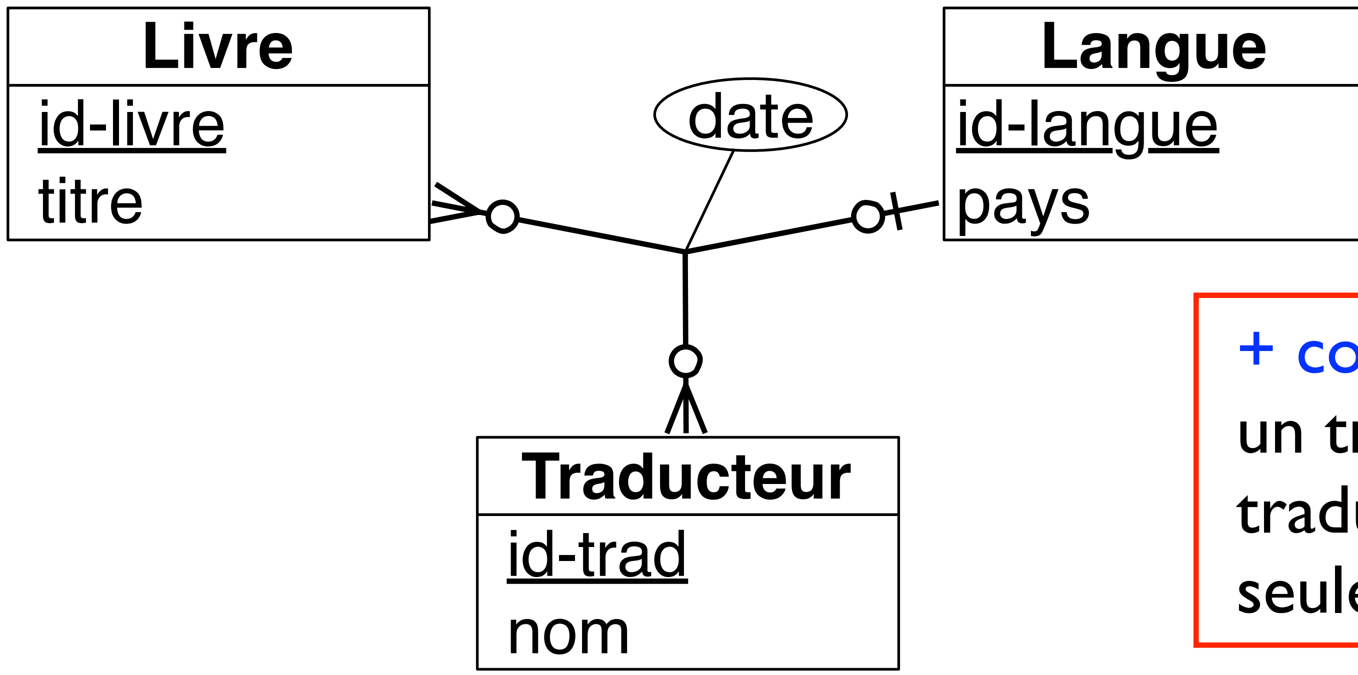
3. Si R (ou R_1.. R_k) n'est pas en FNBC*

   - Trouver une décomposition de R (ou de chaque Ri) qui est sans perte d'information et sans perte de DF en une forme normale : FNBC si possible, 3NF sinon

   - Alternative : "corriger" la modélisation conceptuelle et revenir à l'étape 2.

* **Remarque.** Si on est passé par une étape de modélisation conceptuelle, R_1..R_k a des chances d'être déjà en forme normale, mais cela n'est pas garanti.

Modélisation de schéma relationnels avec des DF - Exemple complet



+ contrainte externe
un traducteur peut traduire dans une seule langue

Le schéma relationnel avec toutes les DF :

Livre (id-livre, titre) id-livre → titre

Langue (id-langue, pays) id-langue → pays

Traducteur (id-trad, nom) id-trad → nom

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

id-livre id-trad → id-langue

id-trad → id-langue

violation de FNBC

Modélisation de schéma relationnels avec des DF - Exemple complet

Première approche : normalisation par décomposition

Livre (id-livre, titre)   id-livre → titre

Langue (id-langue, pays) id-langue → pays

Traducteur (id-trad, nom)   id-trad → nom

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

  id-livre id-trad id-langue → date
  id-livre id-trad → id-langue
  id-trad → id-langue

Tous les schémas de relation sont en FNBC sauf Traduction

  violation : id-trad → id-langue

une étape de décomposition FNBC sur Traduction

id-livre, id-trad, id-langue, date   violation : X = id-trad

id-trad, id-langue   id-trad, id-livre, date

Modélisation de schéma relationnels avec des DF - Exemple complet

Le nouveau schéma relationnel en FNBC :

Livre (id-livre, titre) id-livre → titre

Langue (id-langue, pays) id-langue → pays

Traducteur (id-trad, nom) id-trad → nom

Trad-Langue (id-trad, id-langue) id-trad → id-langue

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

DF locales

Amélioration de la décomposition : on peut fusionner Traducteur et Trad-Langue sans perdre la propriété FNBC.

Livre (id-livre, titre) id-livre → titre

Langue (id-langue, pays) id-langue → pays

Traducteur (id-trad, nom, id-langue) id-trad → nom , id-trad → id-langue

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

Modélisation de schéma relationnels avec des DF - Exemple complet

On identifie les clefs (super-clefs minimales) par les DFs. On choisit une clef par relation, qui devient la clef primaire

Livre (<u>id-livre</u>, titre)   id-livre → titre

Langue (<u>id-langue</u>, pays)   id-langue → pays

Traducteur (<u>id-trad</u>, nom, id-langue)   id-trad → nom , id-trad → id-langue

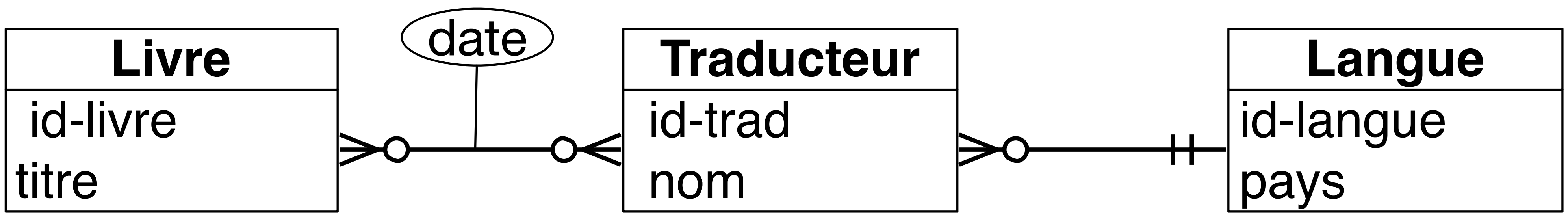
Traduction (<u>id-trad, id-livre</u>, date)   id-livre id-trad → date

Les autres clefs candidates se transforment en contraintes d'unicité

(UNIQUE en SQL)

Modélisation de schéma relationnels avec des DF - Exemple complet

Deuxième approche : on revient sur la modélisation conceptuelle (voir plus haut)



Livre (id-livre, titre) id-livre → titre

Langue (id-langue, pays) id-langue → pays

Traducteur (id-trad, nom, id-langue) id-trad → nom , id-trad → id-langue

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

On répète le test FNBC : succès

Au delà des DF

Exemple

Production | film | réalisateur | acteur

– Supposer que chaque film puisse avoir plusieurs réalisateurs et plusieurs acteurs

– Alors il n'y a pas de DF (non triviales) $\Rightarrow$ ce schéma de relation est en FNBC

– Mais il y a tout de même de la redondance…

Réalisateurs et acteurs sont des informations **indépendantes** pour un même film

Si un film est réalisé par un réalisateur, ce réalisateur est répété une fois pour chaque acteur dans le film, et idem pour les acteurs

Au delà des DF

Exemple

Production	film	réalisateur	acteur

Une meilleure modélisation:

Réalisateurs	film	réalisateur

Acteurs	film	acteur

Dépendance multi-valuée DMV (une généralisation de la contrainte de DF) :

affirme que deux attributs (ou ensembles d'attributs) sont "indépendants" l'un de l'autre

Définition de DMV

Soit R(XYZ) un schéma de relation, X, Y, Z ensembles disjoints d'attributs

Dépendance multi-valuée : $X \twoheadrightarrow Y$
Une instance J de R satisfait $X \twoheadrightarrow Y$ ssi $J = \pi_{XY}(J) \bowtie \pi_{XZ}(J)$

(c-à-d., si dans J une valeur de X apparait associée à une valeur de Y, et apparait également associée à une valeur de Z, alors elle doit apparaitre associé aux deux ensemble)

J

-X-	-Y-	-Z-
t	a	
t		b

$\Rightarrow$

J

-X-	-Y-	-Z-
t	a	b

Définition de DMV

- **Remarque**. MVD est une généralisation de DF

J satisfait $X \rightarrow Y$ implique J satisfait $X \twoheadrightarrow Y$

En d'autres termes : $X \twoheadrightarrow Y$ admets que deux tuples qui sont en accord sur X aient des valeurs different de Y, mais dans ce cas ces valeurs de Y doivent être interchangeables

J

-X-	-Y-	-Z-
t	a	
t		b

$\Rightarrow$

J

-X-	-Y-	-Z-
t	a	b

DMV exemple

Exemple

Production	film	réalisateur	acteur

A une dépendance multi-valuée **film $\twoheadrightarrow$ réalisateur**

I.e. pour chaque instance J de Production : $J = \pi_{\text{film, realisateur}}(J) \bowtie \pi_{\text{film, acteur}}(J)$

Implication de DMV et DF

Donné un schéma de relation R(U) et F : un ensemble de DMV et de DF

Quelles autres DF et DMV sont impliquées ? (dénoté toujours F^+)

Quelques règles d'inférence immédiate :

- toutes les règles d'inférence pour les DF
- $X \twoheadrightarrow Y$ implique $X \twoheadrightarrow Z$ (où $Z = U \setminus XY$)
- $X \rightarrow Y$ implique $X \twoheadrightarrow Y$ (mais pas vice-versa)
- DMV triviales : $XY \twoheadrightarrow Y$, $X \twoheadrightarrow U \setminus X$

Et d'autres :

- des règles qui infèrent des DF à partir de DMV et DF
- une forme de transitivité et augmentation pour les DMV
- et d'autres règles encore (on ne rentrera pas dans le détail...)

Eliminer les redondances dues aux DMV : 4NF

Idée similaire à la FNBC : si $X \twoheadrightarrow Y$ est une DMV non triviale, X doit être une superclef

Ex. la décomposition

Réalisateurs (film, réalisateur) , Acteurs (film, acteur)

évite la redondance due à la DMV : film $\twoheadrightarrow$ réalisateur (qui devient triviale)

Soit R(U) un schéma de relation et F un ensemble de DMV et de DF sur R(U)

R(U) est en quatrième forme normale (4NF)

si pour toute DMV non triviale $X \twoheadrightarrow Y \in F^+$, X est une super-clef

Remarque. 4NF $\Rightarrow$ FNBC (puisque toute DF est également une DMV)

Algorithme de décomposition 4NF : similaire à la décomposition FNBC