



# ESIR1 BD

## Bases de données

normalisation

Olivier Ridoux

# Plan

- Formes normales
- Normalisation



# Formes normales

normalisation

# Formes normales

- Propriété structurelle d'un schéma qui entraîne un niveau de qualité déterminé

**structurel**

≡

**qui dépend seulement du schéma**

# 1<sup>ère</sup> forme normale (1)

- **La valeur de tout attribut est atomique**
- 1NF
- Difficile de faire autrement, mais ...

## 1<sup>ère</sup> forme normale (2)

- On voudrait

Livre( num-livre, titre, **auteurs**, éditeur, année )

- On doit faire

Livre( num-livre, titre, éditeur, année )

**Écrit( num-auteur, num-livre )**

## 2<sup>nd</sup>e forme normale (1)

- 1<sup>ère</sup> forme normale + ...
- ... pour tout  $R(\underline{K}, X)$ ,  $K$  est minimale aussi pour les  $Y \subset X$
- 2NF
- **Pas de clé trop grosse pour une projection de la relation**

## 2<sup>nd</sup>e forme normale (2)

- On voudrait

Client( prénom, nom, fête-à-souhaiter, ...)

- On doit faire

Client( prénom, nom, ...)

**Fête( prénom, fête-à-souhaiter )**



# Remarque : 2<sup>nd</sup>e forme normale

- Si une relation est 1NF et n'a que des clés simples (1 attribut)...

...alors elle est 2NF

# 3<sup>ème</sup> forme normale (1)

- 2<sup>nde</sup> forme normale + ...

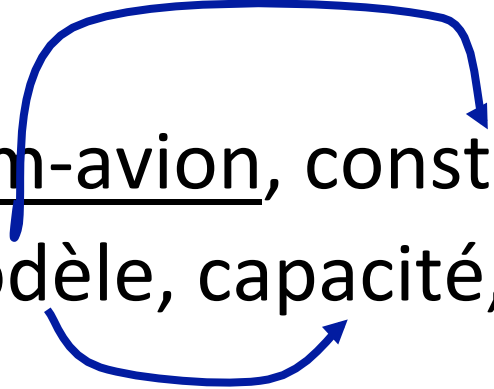
... pour tout  $R(\underline{K}, A, B, X)$ , on n'a pas  $A \rightarrow B$

- 3NF
- **Pas de DF dissimulée dans une relation**

## 3<sup>ème</sup> forme normale (2)

- On voudrait

Flotte( num-avion, constructeur,  
modèle, capacité, ... )



- On doit faire

Flotte( num-avion, num-modèle, ...  
**Modèle( num-modèle,  
constructeur, capacité, ...)**

# Normalisation

normalisation

12

# Jonction conservatrice (1)

- Soit  $R(X)$  une relation,  
...peut-on décomposer  $X$  en

$UCX$  et  $VCX$

tel que

$$\pi_U(R) \bowtie \pi_V(R) = R$$

...principe d'une **décomposition inversible**

## Jonction conservatrice (2)

- Une **décomposition inversible** est possible

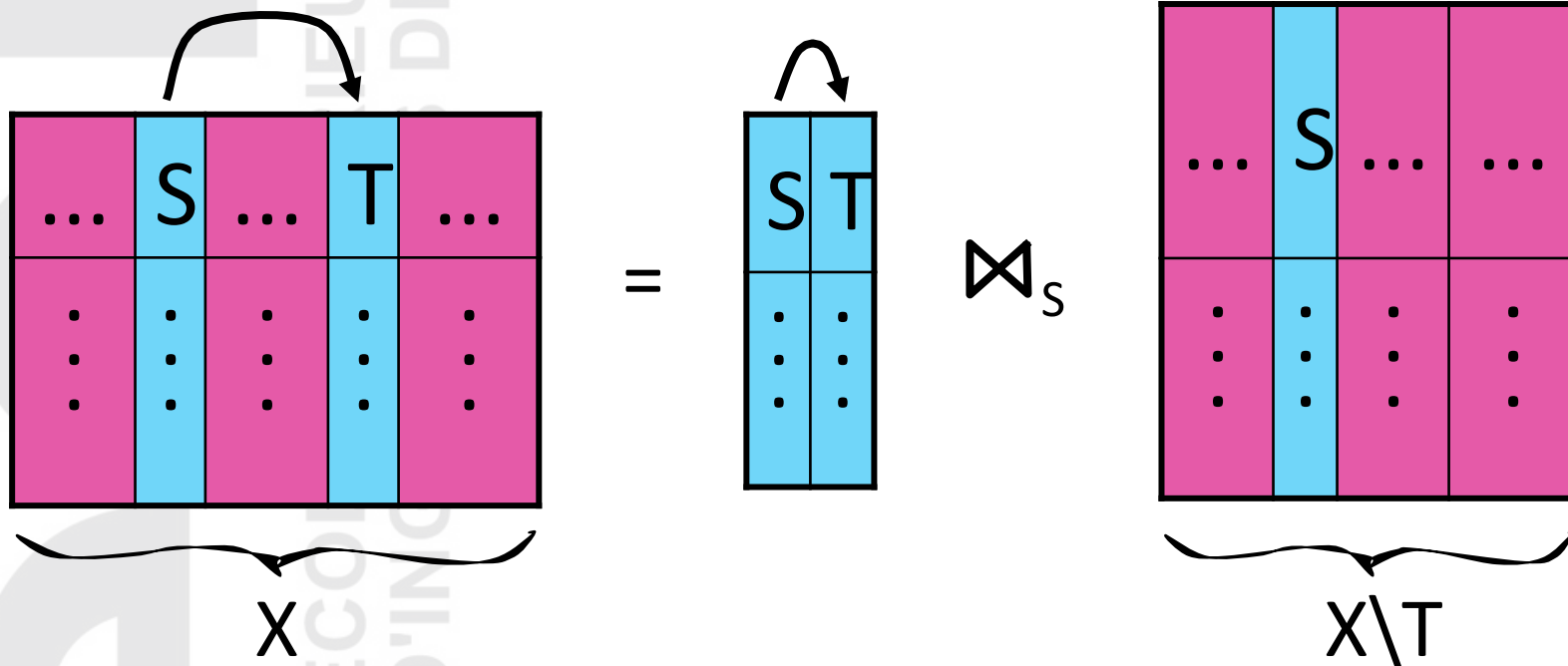
- Si  $S \rightarrow T$  une DF de  $R(X)$  ...  
...alors

$$\Pi_{S,T}(R) \bowtie_S \Pi_{X \setminus T}(R) = R$$

- C'est la **jonction conservatrice**

## Jonction conservatrice (2)

- $R(X) = \Pi_{S,T}(R) \bowtie_S \Pi_{X \setminus T}(R)$



- Quel que soit le contenu des tables

# Exemple - Jonction conservatrice

num-four	ville	frais
...	...	...
f1	<b>Paris</b>	<b>100</b>
f2	<b>Paris</b>	<b>100</b>
f3	Rennes	50
f4	Redon	50

=

ville	frais
...	...
<b>Paris</b>	<b>100</b>
Rennes	50
Redon	50

$\bowtie$  ville

num-four	ville
...	...
f1	<b>Paris</b>
f2	<b>Paris</b>
f3	Rennes
f4	Redon



# Préservation des DF

- Soit  $F$  un ensemble de DF sur  $R(X)$
- Soit  $X_1, X_2, \dots, X_n$  une décomposition de  $X$  à jonction conservatrice
- Soit  $F_i \subset F+$  tq les DF de  $F_i$  sont définies sur  $X_i$
- **$F$  est préservé par la décomposition**

$$\text{ssi } F+ = (\cup F_i)+$$

## 3<sup>ème</sup> forme normale (3)

- Théorème :

Toute relation peut être décomposée...  
...sous forme **3NF** à jonction conservatrice,  
...et avec **préservation** des DF

# Algorithme de synthèse (1)

- Entrée :  $R(X)$  et DF
  - Sortie :  $\{R_1, \dots, R_n\}$  en 3NF
1.  $C$  = couverture irredondante de DF
  2.  $F_1, \dots, F_n = C$  groupée par partie gauche
  3. Pour chaque  $F_i$   
construire  $R_i = \Pi_{F_i}(R)$

## Algorithme de synthèse (2)

- $R(A, B, C, D, E)$  et

- $A \rightarrow B$

- $A \rightarrow C$

- $C, D \rightarrow E$

- $B \rightarrow D$

donne

•  $R_1(\underline{A}, B, C)$

•  $R_2(\underline{C}, \underline{D}, E)$

•  $R_3(\underline{B}, D)$

# Algorithme de synthèse (3)

- Remarque :

$$A \rightarrow D \notin F_1 \cup F_2 \cup F_3$$

...mais

$$A \rightarrow D \in (F_1 \cup F_2 \cup F_3)^+$$

# Forme normale de Boyce-Codd (1)

- $X \rightarrow Y$  élémentaire entraîne que...
  - ... $X$  est une clé
- BCNF
- Les clés sont **adéquates**

# Forme normale de Boyce-Codd (2)

- BCNF  $\Rightarrow$  3NF
- On peut toujours décomposer une relation en BCNF,  
...mais parfois au pris de **perdre des DF**

# Forme normale de Boyce-Codd (3)

- Soit R...

**$R(\underline{A}, B, C)$  et  $A, B \rightarrow C$  et  $C \rightarrow B$**

...R est 3NF

...mais pas BCNF à cause de  $C \rightarrow B$

- **$\pi_{A,B}(R)$  et  $\pi_{C,B}(R)$  est BCNF mais...**

...perd  **$A, B \rightarrow C$**



# Algorithme de décomposition

- Entrée :  $R(X)$  et DF
- Sortie :  $\{R_1, \dots, R_n\}$  en BCNF

1.  $Res = \{R\}$

2. Itérer : soit  $R_i \in Res$  et  $R_i$  pas BCNF

soit  $X \rightarrow Y$  une DF de  $R_i$  qui n'est pas clé

remplacer  $R_i$  par  $R_i \setminus Y$  et  $XY$

## 3NF et BCNF

- Si BCNF mais perte DF préférer 3NF...  
...sinon préférer BCNF
- Commencer par l'algorithme de synthèse ( $\rightarrow$  3NF), puis appliquer l'algorithme de décomposition si besoin (aux composants  $\neg$ BCNF)
- Il existe d'autres formes normales

# xNF et BCNF

- Si mises à jour **fréquentes**, préférer niveau de normalisation **élevé**
- Si mises à jour **rare**s, envisager niveau de normalisation **faible**
- On peut **dénormaliser** exprès pour éviter des jointures répétitives

# En résumé

- Éviter les redondances
- Représenter fidèlement les dépendances fonctionnelles

→ devient **R(...)**

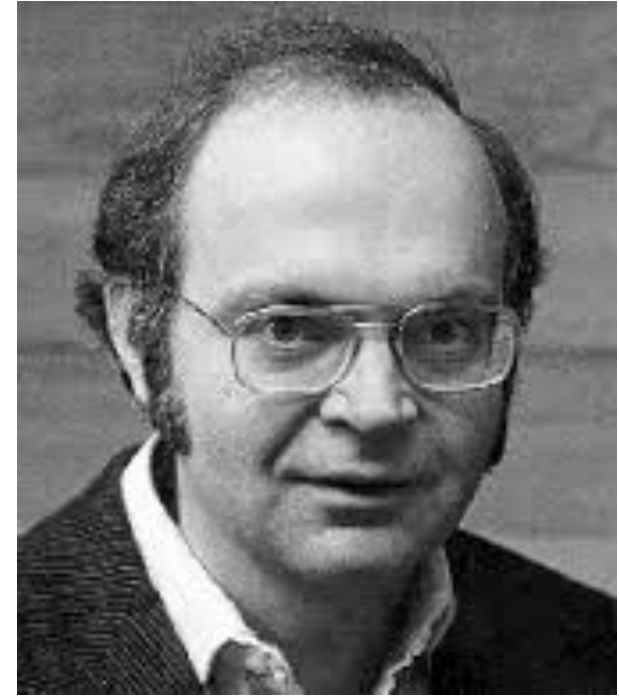
- On peut ne pas normaliser exprès pour des raisons d'efficacité



Attention !

**Attention de ne pas tenter  
d'améliorer un système dont on  
ne connaît pas les performances !**

# Donald Knuth a dit



*There is no doubt that the grail of efficiency leads to abuse[...]:*

***premature optimization is the root of all evil.***

*A good programmer [...] will be wise to look carefully at the **critical** code;*

***but only after that code has been identified.***

# Codd a dit

*The data in a record depends on*

***the Key to the record,  
the Whole Key,  
and nothing but the Key***

@ IBM

