

BASES DE DONNÉES

NORMALISATION ET DÉPENDANCES FONCTIONNELLES



Mickaël Coustaty
Jean-Loup Guillaume

Laboratoire Informatique Image Interaction (L3i)

Université de La Rochelle - Pôle Sciences et Technologie - Avenue Michel Crépeau - 17042 LA ROCHELLE CEDEX 1 France

Tél : +33 (0)5 46 45 82 62 – Fax : 05.46.45.82.42 – Site internet : <http://l3i.univ-larochelle.fr/>

OBJECTIFS DU COURS 2

Maîtriser les principes de la normalisation

- Pourquoi normaliser ?
- Concept de dépendance fonctionnelle
- Formes normales 1 à 3 + FNBC
- Notion de clé

+ Notions sur :

- Quand dénormaliser ?

PRÉREQUIS ET LIENS AVEC D'AUTRES COURS

Prérequis

- Principes de la modélisation
- SQL, clés primaires et étrangères

Liens avec d'autres cours

- C1 = requêtes et notamment jointures
- C3 = fonctions (triggers) pour mise à jour automatique
- C6 indexation = notion de clé
- C7 optimisation = normalisation et dénormalisation
- C9/10 NoSQL = bases « sans tables »

NORMALISATION — EXEMPLE

Paléontologie

- Informations sur les dinosaures :
 - Nom, lieu de vie, poids, nourriture...
- Informations sur les périodes (crétacé, jurassique...)
 - Période (crétacé, jurassique, trias), début, fin

Problèmes ?

Nom	Période	Début	Fin	Lieu	Nourriture	Poids
Allosaurus	Jurassique	200	140	Terre	Carnivore	2.09
Ptéranodon	Crétacé	140	65	Air	Carnivore	NULL
Brontosaurus	Jurassique	200	149	Lac	Herbivore	32.48
Tyrannosaurus	Crétace	140	65	Terre	Carnivore	6.89

NORMALISATION — EXEMPLE

Problèmes

- **Redondance** : Données sur les périodes répétées (espace de stockage)

Nom	Période	Début	Fin	Lieu	Nourriture	Poids
Allosaurus	Jurassique	200	140	Terre	Carnivore	2.09
Ptéranodon	Crétacé	140	65	Air	Carnivore	NULL
Brontosaurus	Jurassique	200	149	Lac	Herbivore	32.48
Tyrannosaurus	Crétace	140	65	Terre	Carnivore	6.89

NORMALISATION — EXEMPLE

Problèmes

- **Redondance** : Données sur les périodes répétées (espace de stockage)
- **Incohérence** : Deux dates de fin pour le Jurassique, Crétacé ou Crétace ?

Nom	Période	Début	Fin	Lieu	Nourriture	Poids
Allosaurus	Jurassique	200	140	Terre	Carnivore	2.09
Ptéranodon	Crétacé	140	65	Air	Carnivore	NULL
Brontosaurus	Jurassique	200	149	Lac	Herbivore	32.48
Tyrannosaurus	Crétace	140	65	Terre	Carnivore	6.89

NORMALISATION — EXEMPLE

Problèmes

- **Redondance** : Données sur les périodes répétées (espace de stockage)
- **Incohérence** : Deux dates de fin pour le Jurassique, Crétacé ou Crétace ?
- **Incomplétude** : Pas d'information sur le Trias si pas de dinosaure associé

Nom	Période	Début	Fin	Lieu	Nourriture	Poids
Allosaurus	Jurassique	200	140	Terre	Carnivore	2.09
Ptéranodon	Crétacé	140	65	Air	Carnivore	NULL
Brontosaurus	Jurassique	200	149	Lac	Herbivore	32.48
Tyrannosaurus	Crétace	140	65	Terre	Carnivore	6.89

Trias ?

NORMALISATION — EXEMPLE

Problèmes

- **Redondance** : Données sur les périodes répétées (espace de stockage)
- **Incohérence** : Deux dates de fin pour le Jurassique, Crétacé ou Crétace ?
- **Incomplétude** : Pas d'information sur le Trias si pas de dinosaure associé
- Comment mettre à jour simplement ?

Nom	Période	Début	Fin	Lieu	Nourriture	Poids
Allosaurus	Jurassique	200	140	Terre	Carnivore	2.09
Ptéranodon	Crétacé	140	65	Air	Carnivore	NULL
Brontosaurus	Jurassique	200	149	Lac	Herbivore	32.48
Tyrannosaurus	Crétace	140	65	Terre	Carnivore	6.89

NORMALISATION — EXEMPLE

Solution : décomposer la table D en 2 tables + une clé étrangère

- Chaque information n'apparaît qu'une fois
- Validation automatique de la période grâce à la clé étrangère

Dinosaures	Nom	Période #	Lieu	Nourriture	Poids
	Allosaurus	Jurassique	Terre	Carnivore	2.09
	Ptéranodon	Crétacé	Air	Carnivore	NULL
	Brontosaurus	Jurassique	Lac	Herbivore	32.48
	Tyrannosaurus	Crétacé	Terre	Carnivore	6.89

Périodes	Période	Début	Fin
	Jurassique	200	140
	Crétacé	140	65
	Trias	230	200

NORMALISATION - OBJECTIFS

Normalisation : aide à la conception d'une « bonne » base de données

- Pas de redondance
- Mise à jour facilitée
- Cohérence mieux assurée
- Compréhension plus aisée

Principe → décomposer les tables quand c'est pertinent

- Comment faire ?
- Quand s'arrêter ?

Comment
guider la
décomposition ?

**Dépendances
Fonctionnelles**

Formes Normales

Jusqu'où
décomposer ?

DÉPENDANCES FONCTIONNELLES

DÉPENDANCES FONCTIONNELLES

Les dépendances fonctionnelles (DFs) expriment

- Les liens sémantiques entre les attributs
- Les choix de représentation de la réalité

ensemble d'attributs $A \rightarrow$ ensemble d'attributs B

B dépend fonctionnellement de A (ou A détermine B)

si et seulement si

Pour une valeur de A correspond une unique valeur de B

On appelle A le déterminant, B le dépendant

DÉPENDANCES FONCTIONNELLES - EXEMPLES

N°sécu → Nom ?

✓ Au N° SS 101 correspond
toujours le même nom (Durand)

Immat → Modèle ?

✓ AAA 100 AB → Clio
ABC 111 CD → Scénic
(sauf si réutilisation Immatriculation)

Couleur → Immat ?

✗ Rouge → AAA 100 AB
Rouge → ABC 111 CD

DÉPENDANCES FONCTIONNELLES

Attention au sens de la DF :

- La partie de droite dépend de la partie gauche, pas l'inverse

Si on suppose qu'un étudiant n'a qu'une moyenne par cours :

- Durand, numéro d'étudiant 100 a obtenu 14 en Droit et en Informatique

Matr, Note \rightarrow Cours ?

- 100, 14 \rightarrow Droit ; 100, 14 \rightarrow Informatique

Matr, Cours \rightarrow Note ?

- 100, Droit \rightarrow 14 ; 100, Informatique \rightarrow 14

DÉPENDANCES FONCTIONNELLES

Attention : regarder dans la base ne suffit pas

- Une DF est une contrainte métier
- On peut observer qu'à chaque valeur de A correspond une unique valeur de B sans que ce soit une DF
 - Lieu → Nourriture est peut-être une DF mais il faudrait voir un expert
- Par contre si pour une valeur de A on a plusieurs valeurs de B alors $A \rightarrow B$
 - Période → Lieu n'est pas une DF

Nom	Période	Début	Fin	Lieu	Nourriture	Poids
Allosaurus	Jurassique	200	140	Terre	Carnivore	2.09
Ptéranodon	Crétacé	140	65	Air	Carnivore	NULL
Brontosaurus	Jurassique	200	149	Lac	Herbivore	32.48
Tyrannosaurus	Crétace	140	65	Terre	Carnivore	6.89

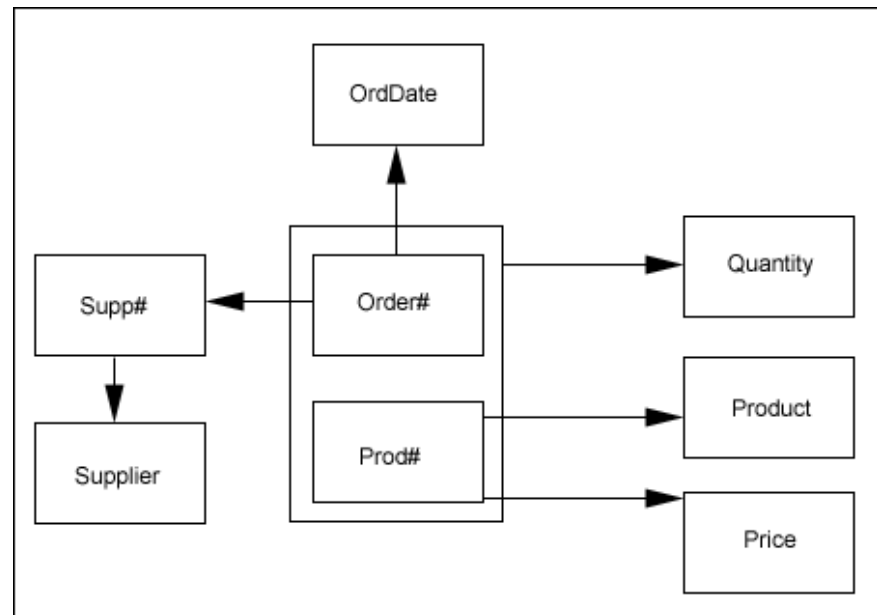
SCHÉMA DE REPRÉSENTATION

Chaque attribut ou groupe d'attribut est encadré

On relie les attributs avec des flèches pour indiquer les DFs

Exemple :

- $\text{Supp} \rightarrow \text{Supplier}$
- $\text{Order} \rightarrow \text{Supp}, \text{OrdDate}$
- $\text{Order}, \text{Prod} \rightarrow \text{Quantity}$
- ...



DÉFINITIONS ET PROPRIÉTÉS

Les trois axiomes d'Armstrong sont les suivants

- Si Y est contenu dans X alors $X \rightarrow Y$ (réflexivité)
- Si $X \rightarrow Y$ alors $XZ \rightarrow YZ$ (augmentation)
- Si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$ (transitivité)

DF **triviales** : On a toujours $X \rightarrow X$ et $X, Y \rightarrow X$ (par réflexivité)

DF **partielles** : $X, Y \rightarrow Z$ est partielle si elle est non triviale et si $X \rightarrow Z$

- C'est-à-dire que Y n'est pas « utile » pour cette DF

DF **élémentaires** : $X \rightarrow Y$ est élémentaire si elle n'est ni triviale ni partielle et Y est un seul attribut

Si $X \rightarrow Y$ est une DF et X **minimal** alors X est une **clé candidate** de la table (X, Y)

- Si X n'est pas minimal ou parle de surclé

DÉFINITIONS ET PROPRIÉTÉS

Fermeture transitive

- La fermeture transitive F^+ d'un ensemble F de DFE est l'ensemble des DFE pouvant être obtenues par transitivité (Si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$)
- Si on pense graphe orienté alors $F^+ =$ ensemble des paires $X \rightarrow Y$ telles qu'il existe un chemin de X à Y
- Ex : Si $F = \{A \rightarrow B, B \rightarrow C, B \rightarrow D, A \rightarrow E\}$, que vaut F^+ ?
 - $F^+ = \{A \rightarrow B, B \rightarrow C, B \rightarrow D, A \rightarrow E, A \rightarrow C, A \rightarrow D\}$

Couverture minimale

- Un sous-ensemble minimum de DFE permettant d'obtenir toutes les autres par transitivité
- Ex : Si $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$ que vaut $CM(F)$?
 - $CM = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$ ou $CM = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$

DÉCOMPOSITION - EXEMPLE

Comment utiliser les DF pour décomposer ?

Produit	Ref	Qté	Fournisseur	CP	Ville
Parapluie	501	50	Labalaine	75000	Paris
Chapeau	530	100	Lemelon	17000	La Rochelle
Parasol	510	100	Labaleine	75000	Paris

Supposons que

- Les attributs CP et Ville dépendent de l'attribut Fournisseur
 - Fournisseur \rightarrow CP, Ville
- L'attribut Produit dépend de Ref
 - Ref \rightarrow Produit
- Les attributs Qté et Fournisseur dépendent de Produit
 - Produit \rightarrow Qté, Fournisseur

DÉCOMPOSITION - EXEMPLE

On peut décomposer la relation initiale:

- Fournisseur → CP, Ville

Produit	Ref	Qté	Fournisseur	CP	Ville
Parapluie	501	50	Labalaine	75000	Paris
Chapeau	530	100	Lemelon	17000	La Rochelle
Parasol	510	100	Labaleine	75000	Paris

Produit	Ref	Qté	Fournisseur#
Parapluie	501	50	Labalaine
Chapeau	530	100	Lemelon
Parasol	510	100	Labaleine

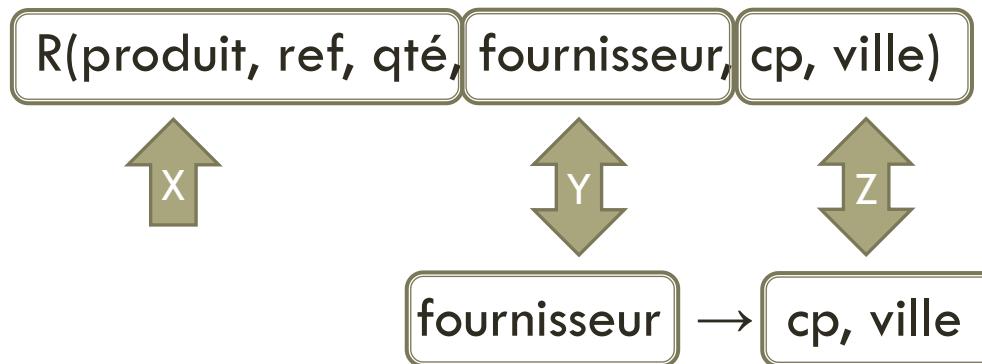
Fournisseur	CP	Ville
Labalaine	75000	Paris
Lemelon	17000	La Rochelle

DÉCOMPOSITION — RÈGLE GÉNÉRALE

Règle générale

- Si $R(X, Y, Z)$ est une table, où X , Y et Z sont des ensembles d'attributs
- Et si $Y \rightarrow Z$ est une DF de R
- Alors $R(X,Y,Z) = \text{jointure de } R(X,Y) \text{ et de } R(Y,Z)$

C'est-à-dire qu'on peut sortir de la table la partie droite de la DF



DÉCOMPOSITION - REMARQUES

Décomposer suivant une DF ne perd pas d'information

- La jointure entre les deux tables issues de la décomposition contient les mêmes informations que la table originale.

On peut toujours décomposer une relation lorsqu'il existe une dépendance fonctionnelle

On ne peut pas décomposer une relation s'il n'y a pas de dépendance fonctionnelle

ALGORITHME DE DÉCOMPOSITION SIMPLE

Entrée

- Une table R à décomposer,
- Un ensemble de DF sur R

Sortie

- Plusieurs tables (en 3FN) dont la jointure redonne R

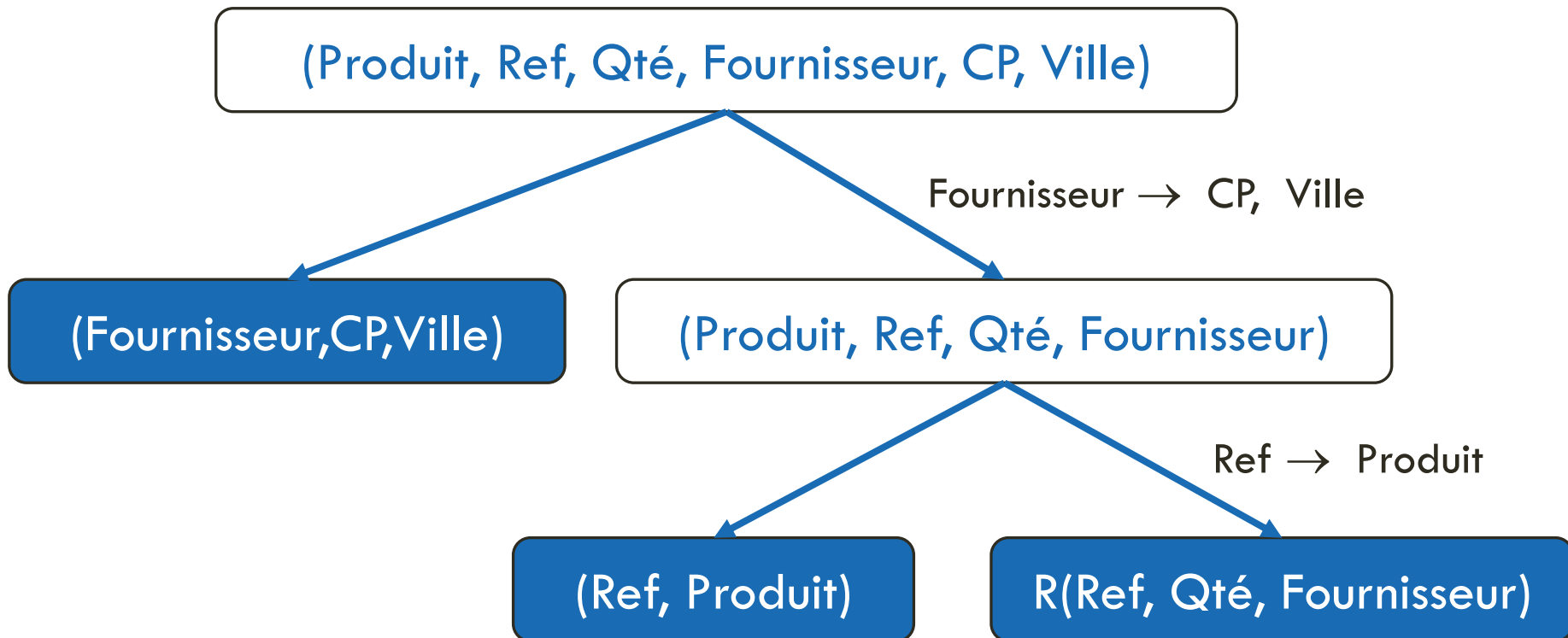
Etape récursive de décomposition

- Nœud courant: une table $R(X, Y, Z)$ (qui n'est pas en 3FN)
- Choisir une DF $Y \rightarrow Z$
- Ajouter la table $R(Y, Z)$ en fils gauche de R
- Ajouter la table $R(X, Y)$ en fils droit de R si X non vide
- Décomposer récursivement $R(X, Y)$ et $R(Y, Z)$ si nécessaire

EXEMPLE

Entrée :

- Table : (Produit, Ref, Qté, Fournisseur, CP, Ville)
- DF : Fournisseur \rightarrow CP, Ville ; Ref \rightarrow Produit ; Produit \rightarrow Qté, Fournisseur



ALGORITHME DE DÉCOMPOSITION

Non déterministe

- Le résultat dépend de l'ordre de traitement des DF
- Les DF ne sont pas toujours préservées (mais on ne perd pas d'information)

Préservation des DF

- Une décomposition préserve les DF si la fermeture transitive de la relation initiale est égale à l'union des fermetures transitives des relations obtenues
- Si préservation alors on ne perd pas de contraintes d'intégrité

Exemple

- $R(A,B,C,D,E)$ avec $A \rightarrow E$; $A,B \rightarrow D$; $B,C \rightarrow A$; $B,D \rightarrow C$
- Si on décompose $B,C \rightarrow A$: (B,C,A) , (B,C,D,E) on perd $A \rightarrow E$

Comment
guider la
décomposition ?

Dépendances
Fonctionnelles

Formes Normales

Jusqu'où
décomposer ?

FORMES NORMALES

POURQUOI NORMALISER ?

Normaliser un schéma relationnel c'est le remplacer par un schéma équivalent où toutes les tables vérifient certaines propriétés

Ces propriétés sont basées sur l'analyse des dépendances fonctionnelles à l'intérieur de chaque table

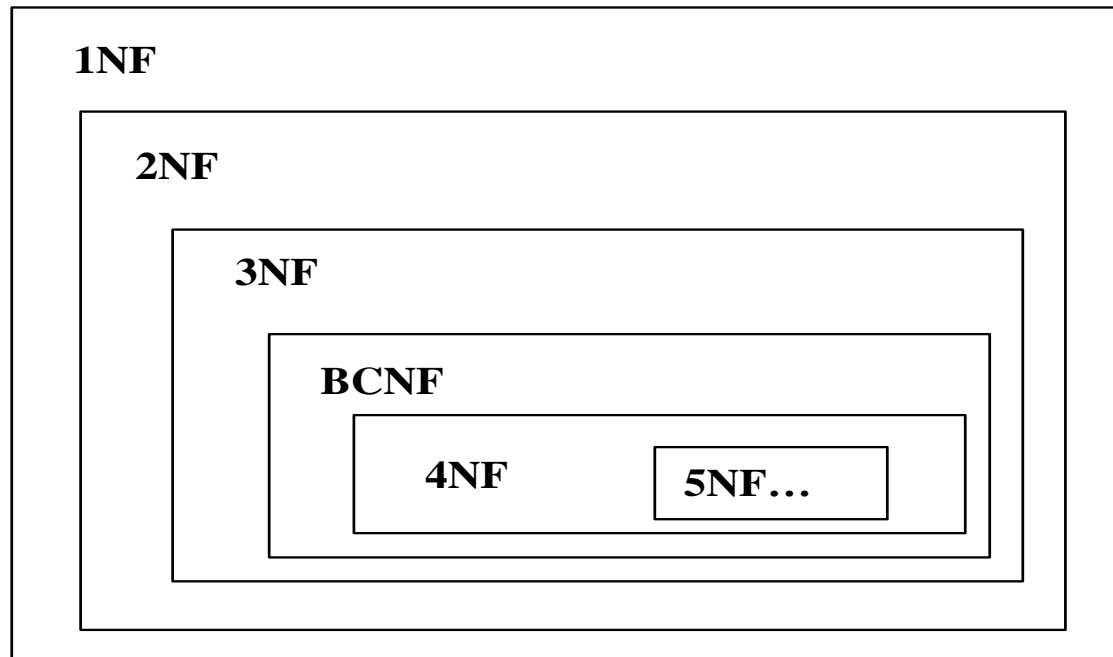
La normalisation est utile pour

- limiter les redondances de données
- minimiser l'espace de stockage
- limiter les pertes de données
- limiter les incohérences
- améliorer les performances des traitements (cf cours optimisation S6)
- éviter les problèmes de mises à jour

FORMES NORMALES

Il existe différents niveaux de normalisation

- 1ere, 2eme et 3eme formes normales (1FN, 2FN, 3FN)
- Forme normale de Boyce-Codd (BCNF)
- Les 4FN et suivantes sont plus complexes




NORMALISATION : 1FN

Une relation est en 1FN si tous les attributs

- Sont atomiques (pas décomposables) / non répétitifs
- (Sont constants dans le temps)

Quels problèmes dans cette base ?

<u>N° SS</u>	Patronyme	Enfants	Adresse	Age
101	Alain Durand	Louise, Naïma	23, Av de Verdun, Meylan, 38240	35
122	Gisèle Dupont	Maëva, Jeanne	15, Bd Pommery, Reims, 51100	44
120	André Remy	Laurent, Pierre	13, R. des Ecoles, Paris, 75005	26



<u>N° SS</u>	Nom	Prénom	Enfant 1	Enfant 2	Adr	Ville	CodeP	ddn
101	Durand	Alain	Louise	Naïma	23, Av de Verdun	Meylan	38240	12/12/79

NORMALISATION : 1FN

La relation Propriétaire est-elle en 1FN ?

N°SS	Immat	Nom	Prénom	Type	Marque	Puis	Coul
101	100RH38	Durand	Alain	R5	Renault	5	Rouge
101	130ZE38	Durand	Alain	2CV	Citroën	2	Jaune
102	200HR51	Noël	Anne	2CV	Citroën	2	Bleue

Oui !! Mais les redondances persistent

- Alain Durand a eu deux voitures
- Alain Durand et Anne Noël ont eu la même voiture

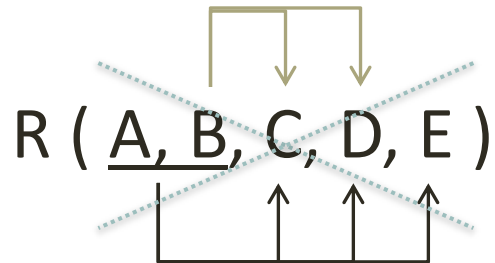
NORMALISATION : 2FN

Une relation 1FN est en 2^e forme normale si

- Aucun attribut non-clé ne dépend que d'une partie d'une clé
- Donc si les clés (attention il peut y en avoir plusieurs) sont simples, la table est en 2FN

Dit autrement on interdit donc la configuration suivante

- $A, B \rightarrow C, D, E$; $B \rightarrow C, D$ (B est une partie d'une clé dont dépendent C, D)



Passage en 2FN : on coupe en $\{\underline{A}, B, E\}$ et $\{B, C, D\}$

NORMALISATION : 2FN

La relation R (N°SS, Nom, NumProj, Heures) est-elle en 2FN ?

<u>N°SS</u>	Nom	<u>NumProj</u>	Heures
101	Durand	1	18
122	Dupont	2	6.5
120	Remy	2	8.5

DFs

- **NSS → Nom**
- NSS, NumProj → Heures

<u>N°SS</u>	Nom
101	Durand
122	Dupont
120	Remy

<u>N°SS</u>	<u>NumProj</u>	Heures
101	1	18
122	2	6.5
120	2	8.5

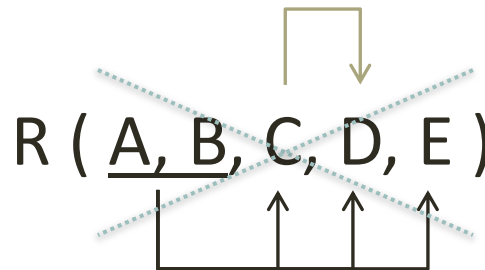
NORMALISATION : 3FN

Une relation 2FN est en 3^e forme normale si

- Il n'existe aucune DF entre les attributs non-clés
- Donc si un seul attribut non clé la table est en 3FN

On interdit la configuration suivante

- $A, B \rightarrow C, D, E$; $C \rightarrow D$ (C et D ne sont pas dans une clé)



Passage en 3FN : on coupe en $\{A, B, C, E\}$ et $\{C, D\}$

NORMALISATION : 3FN

Les relations (N°SS, Nom) et (N°SS, NumProj, Heures) sont-elles 3FN ?

$NSS \rightarrow Nom$

$NSS, NumProj \rightarrow Heures$

<u>N°SS</u>	Nom
101	Durand
122	Dupont
120	Remy

<u>N°SS</u>	<u>NumProj</u>	Heures
101	1	18
122	2	6.5
120	2	8.5

NORMALISATION : 3FN

<u>N°SS</u>	<u>Immat</u>	Nom	Prénom	Type	Marque	Couleur
101	123AF43	Durand	Laurent	Clio	Renault	Rouge
122	478FT12	Dupont	Mélanie	Mégane	Renault	Vert
120	119QN56	Marchand	Maxime	106	Peugeot	Bleu

1FN :

- Propriétaire (N°SS, Immat, Nom, Prénom, Type, Marque, Couleur)
- Pas 2FN car **N°SS → Nom, Prénom**

2FN :

- Personnes (N°SS, Nom, Prénom)
- Voitures (Immat, N°SS, Couleur, Type, Marque)
- Pas 3FN car **Type → Marque**

3FN :

- Personnes (N°SS, Nom, Prénom)
- Voitures (Immat, N°SS, Type, Couleur)
- Modèles (Type, Marque)

NORMALISATION : 3FN

Caractéristiques de la troisième forme normale

- Elle préserve les dépendances fonctionnelles
 - La fermeture transitive des DF est la même que celle de l'union des décomposées
- Elle est sans perte

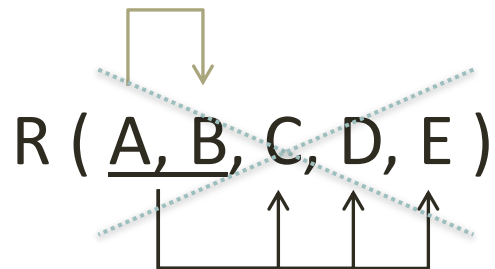
FORME NORMALE DE BOYCE-CODD

Une relation 3FN est en FNBC si

- Tout attribut d'une clé ne dépend pas d'un autre attribut d'une clé
 - Donc si les clés sont toutes simples, la table est en FNBC
- De manière équivalente (sans passer par 2FN et 3FN) : une relation est FNBC si pour toute DF non triviale $X \rightarrow Y$, X est une surclé

On interdit la configuration suivante

- $A, B \rightarrow C, D, E$; $A \rightarrow B$ (A et B sont tous deux dans une clé)



Passage en FNBC : on coupe en $\{\underline{A}, C, D, E\}$ et $\{\underline{A}, B\}$

FORME NORMALE DE BOYCE-CODD

<u>Commande</u>	<u>Ref Produit</u>	<u>Produit</u>	Qté
10001	501	Parapluie	50
10001	530	Chapeau	100
10003	510	Parasol	100

Commande, RefProduit, Produit → Qté

Ref_Produit → Produit

<u>Commande</u>	<u>Ref Produit</u>	Qté
10001	501	50
10001	530	100
10003	510	100

<u>Ref Produit</u>	<u>Produit</u>
501	Parapluie
530	Chapeau
510	Parasol

DÉCOMPOSITION EN BCNF - ALGORITHME

Entrées : une table R + dépendances fonctionnelles sur R

Sortie : décomposition en R en tables FNBC sans pertes

Tant qu'il existe une table R' qui n'est pas en FNBC

- Chercher une dépendance $A \rightarrow B$ qui ne respecte pas la contrainte
- Décomposer R' en R1(A,B) et R2(A, reste)

Le résultat dépend de l'ordre dans lequel on traite les DFs

- Pas de perte d'information mais perte de DFs éventuelles

POURQUOI ÇA NE SUFFIT PAS ?

Exemple : Un magasin livre

- Différentes pizzas
- Différentes zones
- Toutes les pizzas dans toutes les zones de livraison

FNBC car

- Pas d'attribut non clé
- La table est bien en 3FN

Il peut rester des redondances

<u>Restaurant</u>	<u>Pizza</u>	<u>Zone</u>
A1 Pizza	Thick Crust	Springfield
A1 Pizza	Thick Crust	Shelbyville
A1 Pizza	Thick Crust	Capital City
A1 Pizza	Stuffed Crust	Springfield
A1 Pizza	Stuffed Crust	Shelbyville
A1 Pizza	Stuffed Crust	Capital City
Elite Pizza	Thin Crust	Capital City
Elite Pizza	Stuffed Crust	Capital City
Vincenzo's Pizza	Thick Crust	Springfield
Vincenzo's Pizza	Thick Crust	Shelbyville
Vincenzo's Pizza	Thin Crust	Springfield
Vincenzo's Pizza	Thin Crust	Shelbyville

NORMALISATION - SYNTHÈSE

1FN

- Pas d'attribut non atomique

2FN

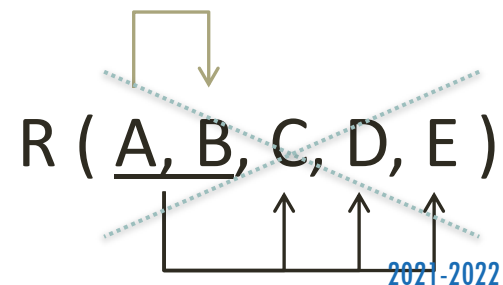
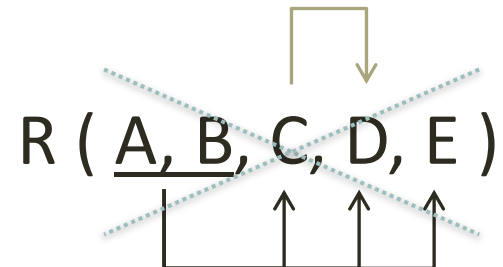
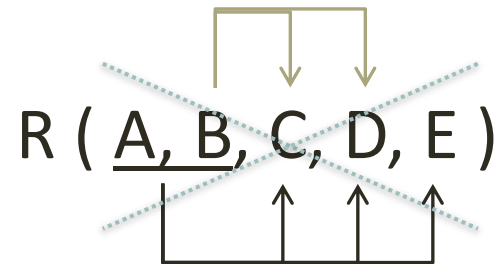
- Pas de DF sur une partie d'une clé

3FN

- 2FN + pas de DF hors d'une clé

FNBC

- 3FN + pas de DF dans une clé



NORMALISATION - SYNTHÈSE

Avantage : limite les redondances de données donc

- L'espace disque nécessaire
- Les incohérences de données qui pourraient les rendre inutilisables
 - Notamment les 4FN et 5FN qui permettent d'exprimer des contraintes plus complexes
- Les mises à jour multiples

Inconvénients

- Temps d'accès potentiellement supérieur car jointures sur plus de tables
 - Mais chercher dans une grande table ça peut coûter aussi
- Non redondance \Rightarrow plus grande fragilité des données
- Difficulté de réorganiser la structure (si on ne connaît pas à l'avance les liens entre entités, ce qui est à éviter)



OPTIMISATION

NORMALISATION ET DÉNORMALISATION

La normalisation c'est bien

- Bases plus structurées, moins (pas ?) de redondances, moins de risques de données erronées

Mais où s'arrêter ?

- FN3 ou FNBC peu contraignantes et évitent les redondances de base
- 5FN si nécessaire

Dénormalisation = enfreindre les règles de normalisation

- A faire uniquement si les tables sont normalisées
- Et que l'on rencontre un problème de performances ou de complexité
 - “Premature optimization is the root of all evil”
- Pas juste par flemme de faire des jointures
 - Car dénormaliser implique des tables plus grosses, des redondances donc obligation d'ajouter des vérifications, etc.

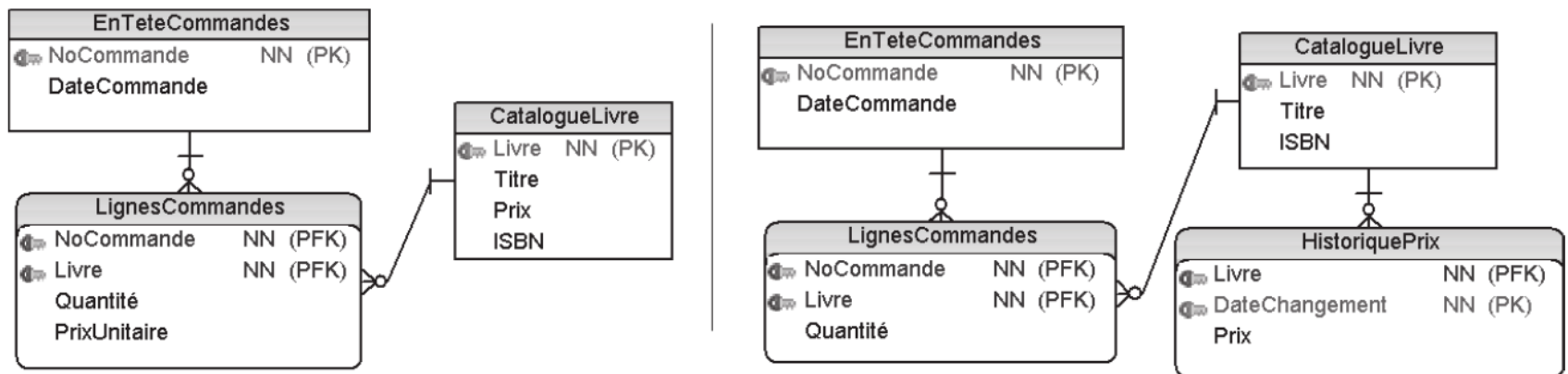
EXEMPLE — HISTORIQUE DE PRIX

Prix des livres + changements de prix

- V1 : les prix sont copiés dans chaque commande
- V2 : on stocke l'historique des prix dans une table annexe

Questions :

- Comment connaître toutes les évolutions du prix d'un livre dans le cas 1 ?
- Comment trouver le prix d'un produit dans le cas 2 ?



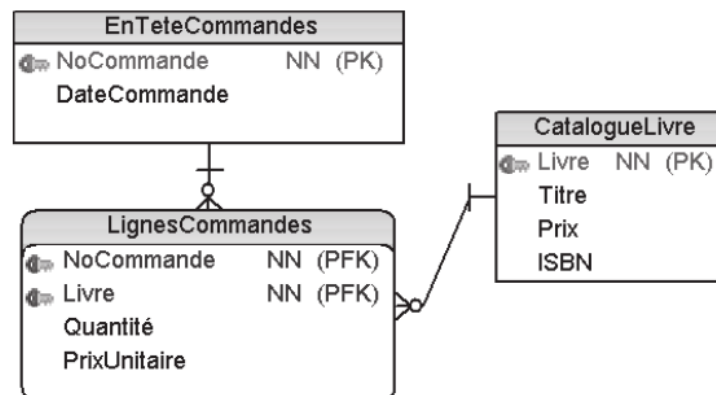
EXEMPLE — PRIX TOTAL D'UNE COMMANDE

Comment calculer le prix total de la commande ?

- `SELECT SUM(Quantité * PrixUnitaire) FROM ...`

Une fois qu'une commande est créée, va-t-elle changer souvent ?

- Si oui alors il vaut mieux ne rien faire et recalculer
- Si non alors on peut ajouter un prix total dans la commande
 - Avec éventuellement un trigger si on veut mettre à jour le total



EXEMPLE — ANALYSE DE L'EXECUTION

```
EXPLAIN ANALYZE
SELECT first_name, last_name, postal_code
FROM customer c
JOIN address a on c.address_id = a.address_id
```

```
"Hash Join (cost=20.48..41.76 rows=599 width=18) (actual time=0.152..0.261
rows=599 loops=1)"
"  Hash Cond: (a.address_id = c.address_id)"
"    -> Seq Scan on address a (cost=0.00..13.03 rows=603 width=7) (actual
time=0.010..0.035 rows=603 loops=1)"
"    -> Hash (cost=12.99..12.99 rows=599 width=15) (actual time=0.135..0.135
rows=599 loops=1)"
"          Buckets: 1024  Batches: 1  Memory Usage: 37kB"
"          -> Seq Scan on customer c (cost=0.00..12.99 rows=599 width=15)
(actual time=0.006..0.058 rows=599 loops=1)"
"Planning time: 0.076 ms"
"Execution time: 0.297 ms"
```