

# Bases de Données 1

## #4 - Schéma relationnel normalisé

Matthieu Nicolas  
Polytech S5 - II

Slides réalisées à partir de celles de Claude Godart et Malika Smaïl

# Plan

- Objectifs de la normalisation 3
- Dépendances Fonctionnelles 11
- Processus de normalisation 22

# Objectifs de la normalisation

Base de Données 1

#4 - Schéma relationnel normalisé

# Limite des systèmes à une table - 1

du produit		du dépôt		quantité produit stockée dans dépôt		
idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
p1	Parasite	14.99	d2	Laxou	6000	500
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900

**Stock**(idProd, libelle, prixUnit, idDepot, adresse, volume, qte)

# Limite des systèmes à une table - 2

- **Redondance**

- *libelle* et *prixUnit* apparaissent pour tous les tuples relatifs au même produit
- Si modification du *prixUnit* d'un produit, autant d'opérations que de tuples relatifs à ce produit

idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
p1	Parasite	14.99	d2	Laxou	6000	500
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900

# Limite des systèmes à une table - 3

- **Risque d'introduction d'incohérence**
  - Erreur de saisie lors de l'insertion d'un nouveau tuple relatif au produit **p2**

idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
p1	Parasite	14.99	d2	Laxou	6000	500
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900
p2	Mario 3D All Stars	5.49	d1	Nancy	9000	1000

# Limite des systèmes à une table - 4

- **Risque de perte d'information**
  - En cas de **suppression** du 2nd tuple, **perd** les **informations** relatives au dépôt *d2*

idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
<del>p1</del>	<del>Parasite</del>	<del>14.99</del>	<del>d2</del>	<del>Laxou</del>	<del>6000</del>	<del>500</del>
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900

# Limite des systèmes à une table - 5

- **Gêne saisie d'information**
  - On ne **peut pas saisir** les informations d'un **produit indépendamment** de son **stock** dans un dépôt
  - Ou alors table pleine de trous...

idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
p1	Parasite	14.99	d2	Laxou	6000	500
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900
p3	Tenet	14.99	NULL	NULL	NULL	NULL



# Meilleure solution

idProd	libelle	prixUnit
p1	Parasite	14.99
p2	Mario 3D All Stars	54.99

**Produit**(idProd, libelle, prixUnit)

idDepot	adresse	volume
d1	Nancy	9000
d2	Laxou	6000
d3	Nancy	2000

**Depot**(idDepot, adresse, volume)

idProd	idDepot	qte
p1	d1	300
p1	d2	500
p2	d3	900

**Stock**(idProd, idDepot, qte)

# Objectifs de la normalisation

- **Assurer** qu'un **schéma relationnel** a les **bonnes propriétés**
- **Minimiser les redondances** pour limiter les **anomalies** d'insertion, de modification et de suppression

# Dépendances Fonctionnelles

Base de Données 1

#4 - Schéma relationnel normalisé

# Processus de normalisation

- Une relation est **normalisée** si elle ne pose **pas** de problème **de redondance** ni de risques **d'anomalies** lors des mises à jour
- Consiste à **décomposer** les relations redondantes en **relations normalisées**
- **Sur quoi se baser** pour effectuer cette décomposition ?

# Dépendances fonctionnelles (DF) - 1

- Soit  $R(A)$  une relation  $R$  avec  $A$  l'ensemble de ses attributs
- Soient  $X$  et  $Y$  deux attributs ou ensembles d'attributs de la relation  $R$  ( $X, Y \subseteq A$ )
- On dit que  $Y$  *dépend fonctionnellement* de  $X$  si, **pour une valeur de  $X$ , il ne correspond qu'une valeur de  $Y$**  dans la relation  $R$
- On note alors  **$X \rightarrow Y$**

# DF dans la relation Produit - 1

idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
p1	Parasite	14.99	d2	Laxou	6000	500
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900

- Deux produits ne peuvent pas avoir le même numéro
  - **idProd → libelle**
  - **idProd → prixUnit**

# DF dans la relation Produit -

## 2

idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
p1	Parasite	14.99	d2	Laxou	6000	500
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900

- Deux dépôts ne peuvent pas avoir le même numéro
  - **idDepot** → **adresse**
  - **idDepot** → **volume**

# DF dans la relation Produit -

## 3

idProd	libelle	prixUnit	idDepot	adresse	volume	qte
p1	Parasite	14.99	d1	Nancy	9000	300
p1	Parasite	14.99	d2	Laxou	6000	500
p2	Mario 3D All Stars	54.99	d3	Nancy	2000	900

- La quantité en stock ne dépend que du produit et du dépôt
  - **idProd, idDepot → qte**



# Dépendances fonctionnelles (DF) - 2

- Pour une DF  $X \rightarrow Y$ , on dit que  $X$  est la **source** et  $Y$  la **cible**
- $\forall t1$  et  $t2$ , deux tuples de  $R$ , si  $t1[X] = t2[X]$  alors  $t1[Y] = t2[Y]$
- Dans une relation tout attribut est en DF avec la clé primaire

# Propriété des DFs - 1

- Pour une relation  $R(A)$ , avec  $X, Y, W$  et  $Z$  des ensembles d'attributs de  $R$  ( $X, Y, W, Z \subseteq A$ )
- (F1) *Réflexivité* :  $Y \subseteq X \Rightarrow X \rightarrow Y$  (en particulier,  $X \rightarrow X$ )
- (F2) *Augmentation* :  $X \rightarrow Y \Rightarrow X \cup Z \rightarrow Y \cup Z$
- (F3) *Union* :  $X \rightarrow Y$  et  $X \rightarrow Z \Rightarrow X \rightarrow Y \cup Z$

# Propriété des DFs - 2

- (F4) *Transitivité* :  $X \rightarrow Y$  et  $Y \rightarrow Z \Rightarrow X \rightarrow Z$
- (F5) *Pseudo-transitivité* :
  - $X \rightarrow Y$  et  $Y \cup W \rightarrow Z \Rightarrow X \cup W \rightarrow Z$
- (F6) *Décomposition* :  $X \rightarrow Y$  et  $Z \subseteq Y \Rightarrow X \rightarrow Z$

# Clé candidate : nouvelle définition

- Un **ensemble d'attributs**  $X \subseteq A$  est une **clé** pour la **relation**  $R(A)$  si :
  - $X \rightarrow A$
  - $X$  est **minimal** ( $X$  est le plus petit ensemble d'attributs tel que  $X \rightarrow A$ )

# Théorème de décomposition

- Soit  $R(X, Y, Z)$  une relation où  $X$ ,  $Y$  et  $Z$  sont des ensembles d'attributs, et  $X \rightarrow Y$
- Alors la **décomposition de  $R(X, Y, Z)$  en  $R_1(X, Y)$  et  $R_2(X, Z)$  est sans perte d'information**

# Processus de normalisation

Base de Données 1

#4 - Schéma relationnel normalisé

# Processus de normalisation

- Consiste à **faire évoluer** les **relations** pour **respecter** un **ensemble de propriétés**
- Il existe **plusieurs ensembles de propriétés**, de **plus en plus contraints**
- On appelle **Forme Normale** (FN, ou NF en anglais) ces ensembles de propriétés

# Relation en 1NF

- Une relation est en **1NF** si **chacun de ses attributs** est **atomique** (non-composé) et **mono-valué**
- Considérons la relation suivante :
  - **Personne**(*id*, *prenom*, *nom*, *diplomes*)
  - où *diplomes* est l'ensemble des diplômes obtenus par une personne



# Relation Personne

idPers	prenom	nom	diplomes
p1	Jean	Maire	Licence Master Doctorat
p2	David	Tor	Licence Master

- La relation **Personne** n'est pas en **1NF**

# Comment normaliser Personne en 1NF ? - 1

- Possibilité 1 :
  - **Exploser l'attribut** multi-valué en plusieurs attributs mono-valué

idPers	prenom	nom	diplome1	diplome2	diplome3
p1	Jean	Maire	Licence	Master	Doctorat
p2	David	Tor	Licence	Master	NULL

**Personne**(idPers, prenom, nom, diplome1, diplome2, diplome3)

# Comment normaliser Personne en 1NF ? - 2

- Possibilité 2 :
  - **Créer une relation** répertoriant les **diplômes** d'une **Personne**

idPers	prenom	nom
p1	Jean	Maire
p2	David	Tor

**Personne**(idPers, prenom, nom)

idPers	diplome
p1	Licence
p1	Master
p1	Doctorat
p2	Licence
p2	Master

**Diplomes**(idPers, diplome)

# Comment normaliser Personne en 1NF ? - 3

- Possibilité 1
  - Répertoire toutes les informations dans la même relation...
  - ... Mais des **attributs** n'ayant **pas de sens** pour **certains tuples** (valeur NULL)
- Possibilité 2
  - **Besoin d'accéder** aux 2 relations pour récupérer l'intégralité des données d'une **Personne**...
  - ... Mais **toute occurrence** est **sémantiquement correct**

# Relation en 2NF

- Une relation est en **2NF** si
  - Elle est en **1NF**
  - Tout attribut n'appartenant pas à la clé primaire ne dépend pas d'une partie de la clé (DF partielle)
- Considérons la relation suivante :
  - **Stock**(*idProd*, *idDepot*, *libelle*, *qte*)

# Relation Stock

idProd	idDepot	libelle	qte
p1	d1	Parasite	300
p1	d2	Parasite	500
p2	d3	Mario 3D All Stars	900

**Stock**(idProd, idDepot, libelle, qte)

- **Stock** est en **1F** mais n'est pas en **2NF**
  - **idProd, idDepot → qte**
  - **idProd → libelle**

# Comment normaliser Stock en 2NF ?

- **Décomposer Stock** en deux relations

idProd	libelle
p1	Parasite
p2	Mario 3D All Stars

**Produit**(idProd, libelle)

idProd	idDepot	qte
p1	d1	300
p1	d2	500
p2	d3	900

**Stock**(idProd, idDepot, qte)

- Chacune de ces relations est en **2NF**

# Comment normaliser en 2NF ?

- Isoler la DF qui pose problème en créant une nouvelle relation
- Éliminer l'attribut cible de la DF de la relation initiale



# Relation en 3NF

- Une relation est en **3NF** si
  - Elle est en **2NF**
  - Tout attribut n'appartenant pas à la clé primaire ne dépend pas d'un autre attribut n'appartenant pas à la clé
- Considérons la relation suivante :
  - **Avion**(*idAvion*, *constructeur*, *type*, *capacite*, *proprietaire*)

# Relation Avion

idAvion	constructeur	type	capacite	proprietaire
AH32	Boeing	B747	C1	Air France
FM34	Airbus	A320	C2	British Airways
BA45	Boeing	B747	C1	Egypt Air

***Avion**(idAvion, constructeur, type, capacite, proprietaire)*

- **Avion** est en **2NF** mais n'est pas en **3NF**
  - **type** → **capacite**
  - **type** → **constructeur**

# Comment normaliser Avion en 3NF ?

- **Décomposer Avion** en deux relations

idAvion	type	proprietaire
AH32	B747	Air France
FM34	A320	British Airways
BA45	B747	Egypt Air

**Avion**(idAvion, type, proprietaire)

idType	constructeur	capacite
B747	Boeing	C1
A320	Airbus	C2

**Type**(idType, constructeur, capacite)

- Chacune de ces relations est en **3NF**

# Comment normaliser en 3NF ?

- Isoler la DF qui non directe en créant une nouvelle relation
- Éliminer l'attribut cible de la DF de la relation initiale

# Et après ?

- Maintenant que toutes nos relations sont en **3NF**, il n'y a plus de redondance ?
- Si seulement...
- Considérons la relation suivante :
  - **Inscription**(*noEtudiant*, *module*, *enseignant*)

# Relation Inscription - 1

noEtudiant	module	enseignant
101	Java	Pr. Java
101	C#	Pr. Chash
102	Java	Pr. Java2
103	Java	Pr. Java

**Inscription**(noEtudiant, module, enseignant)

- **Inscription** est en **3NF**
  - noEtudiant, module → *tout*
  - Pas de dépendances partielles ou transitives

# Relation Inscription - 2

noEtudiant	module	enseignant
101	Java	Pr. Java
101	C#	Pr. Chash
102	Java	Pr. Java2
103	Java	Pr. Java

**Inscription**(noEtudiant, module, enseignant)

- Mais redondance
  - **enseignant → module**
  - Répète le module pour toutes les occurrences d'un enseignant

# Comment normaliser Inscription ?

- On a une Forme Normale pour ça !
  - **Forme Normale Boyce-Codd (BCNF)**
- Mais nous on va s'arrêter là...



# Formes Normales restantes

- Il (vous) reste plein de Formes Normales à découvrir !
  - 8 Formes Normales dans le modèle relationnel
- Mais globalement, les **3 premières** sont les **plus connues et utilisées**

# La dénormalisation

Base de Données 1

#4 - Schéma relationnel normalisé

# Normaliser, c'est cool...

- Normaliser permet de minimiser les redondances
  - **Réduit la volumétrie** globale de notre base de données
  - **Limite les risques d'anomalies** d'insertion, de modification et de suppression
  - **Réduit le nombre**, et donc **la durée**, d'opérations de mise à jour

# ... mais possède aussi des inconconvénients

- **Normaliser augmente le nombre de relations** dans notre schéma relationnel
- Besoin d'accéder, de parcourir et de **recroiser** les données de ces relations lors de requêtes
  - Génère un **surcoût**

# Dénormalisation

- Consiste à **réintroduire de la redondance, maîtrisée**, dans le schéma relationnel **pour améliorer les performances**
  - Besoin d'**identifier les données à dupliquer** d'une relation à l'autre
  - Besoin de **mesurer le gain** de performances

# Remarques sur la dénormalisation

- Généralement, se limiter à la **3NF** est un bon compromis pour une BD
- **Seulement pour les BDs volumineuses où on observe que les performances sont problématiques**, on considère la dénormalisation

# Résumé

- Un **schéma relationnel mal conçu** peut entraîner une **redondance** de données
- Redondance de données introduit des **risques d'anomalies**
- Le processus de normalisation permet d'**éliminer** de plus en plus de **redondance**
- Repose sur l'**identification des DFs** entre attributs
- **Mettre ses relations en 3NF** est souvent suffisant