



Institut National
Universitaire
Champollion

CHAPITRE 3

Processus de création d'une BDD : Dépendances fonctionnelles Décompositions et Formes Normales



Chapitre 3 : Plan

- Introduction
- Dépendances fonctionnelles
- Formes normales
- Décomposition en formes normales

Introduction

L'objectif du chapitre est de se donner les outils permettant d'être capable de fournir un bon système de relations permettant de construire une base de données « cohérente » et « robuste ».

Il y a plusieurs techniques pour y arriver :

- Etude des dépendances fonctionnelles.
- Traduction d'un diagramme UML en un schéma relationnel, puis normalisation.

Nous allons dans ce chapitre exposer la première méthode, puis nous détaillerons la normalisation.

Introduction

Un bon schéma relationnel doit :

- Éviter la redondance de l'information
- Ne pas perdre d'informations
- Être « robuste » : pouvoir insérer, modifier, supprimer des t-uples sans introduire d'incohérences dans les données.

Introduction : exemple

Prenons par exemple la base livraison suivante :

Sachant que les frais de livraison sont fonction de la villeClient

Immat	NomAgence	marque	modèle	VilleClient	NomClient	FraisLivraison
237E98	Agence du pont vert	Jaguar	ZE	Castres	Plouc	100
576T56	Agence Mike	Renault	Clio	Albi	De Kerziziou	300
934Y76	Agence truc	Peugeot	407	Albi	PasMoi	300
196F45	Agence Mike	Renault	Clio	Gaillac	PasToi	150

Il y a :

- Redondance d'information : le modèle clio associé à la marque Renault apparaît deux fois. De même pour les frais de livraison associés à Albi : problèmes de modification.
- Anomalies de modification : si les frais de livraison liés à Rennes changent il y a plusieurs t-uples à modifier. Si un client déménage, il faudra aller chercher les frais liés à cette ville.
- Si on supprime le premier t-uple, on perd l'information que les frais liés à Casres sont de 100 euros.
- Etc

Introduction

Concevoir une base de données c'est fournir une description des données cad des schémas ou diagrammes relationnels .

Les t-uples insérés ou modifiés doivent à tout instant vérifier toutes les contraintes exprimées par la description des données.

Dépendances fonctionnelles(DF)

- Elles permettent d'exprimer des contraintes d'intégrité sur des attributs.
- Elles expriment ainsi certaines propriétés sur les données.
- Elles sont un cas particulier de contrainte d'intégrité

L'objectif de ce paragraphe est donc d'explicitier les DF de façon optimale, afin de trouver une décomposition de l'ensemble des attributs en tables sans perte d'information.

Cela nous permettra, dans le paragraphe suivant de décliner des règles baptisées Formes Normales pour construire correctement les schémas relationnels.

Dépendances fonctionnelles

Définition :

Soit R une relation et A l'ensemble des attributs de R , et soient X et Y deux sous ensembles de A .

On dit que Y dépend fonctionnellement de X ssi :

Étant donné une valeur de X , il lui correspond au plus une valeur de Y

On note $X \rightarrow Y$,

X est la **source**, Y est le **but** de la DF.

Par exemple : modèle \rightarrow marque (et pas l'inverse !)

Dépendances fonctionnelles

Une dépendance fonctionnelle $X \rightarrow Y$ est dite **élémentaire** (dfe) ssi :

Pour tout sous ensemble X' strictement inclus dans X , il n'y a pas de DF entre X' et Y .

Par exemple la DF (immat,marque) \rightarrow modèle n'est pas élémentaire car on a aussi la DF : immat \rightarrow modèle.

On remarquera que toute DF ayant pour source un seul attribut est nécessairement élémentaire.

Dépendances fonctionnelles

DF et fonction mathématique :

Il y a une différence entre « dépendance fonctionnelle » entre deux attributs A1 et A2, et « fonction » : $A1 \rightarrow A2$

La df assure que, à un instant donné, la relation binaire entre A1 et A2 est une fonction.

Cependant , la valeur de A2 n'est pas fixée de façon absolue : par exemple, le client qui habitait Castres hier, peut habiter Albi aujourd'hui.

Dépendances fonctionnelles

DF intra table et DF inter table :

Les DF étant des dépendances entre les valeurs des attributs, il y a deux cas possibles :

- soit les valeurs des attributs concernent le même tuple (df **INTRA**)
- soit les valeurs des attributs concernent des tuples différents (df **INTER**)

Dépendances fonctionnelles

L'exemple typique de DF intra table est :

PK (tuple) ---> autre attribut du tuple

codage : contrainte **PK**

L'exemple typique de DF inter table est :

FK (tuple1) ---> attribut (tuple2)

avec $FK (tuple1) = PK (tuple2)$

codage : contrainte **FK**

Dépendances fonctionnelles

Exemple de DF inter table :

Vol(noVol, aeroportDepart), aeroportDepart référence nom.
Aeroport(nom, capacité)

Il y a bien une DF inter table entre aeroportDepart et nom :

En effet à chaque attribut aéroportDepart est associé au plus un nom, en revanche un nom peut être associé à plusieurs aeroportDepart.

Dépendances fonctionnelles


Une **DF forte** correspond à une **fonction totale**

codage : contrainte **NN** sur le but de la DF

Représentation : 

Une **DF faible** correspond à une fonction non totale

codage : aucun puisqu'aucune contrainte

Représentation : 

Dépendances fonctionnelles

Exemples

Dans la relation :

Voiture(nom, marque NN , modèle NN, nomPropriétaire)

Les DF intra tables :

nom \longrightarrow marque et nom \longrightarrow modèle sont FORTES

Mais la DF intra table : nom-----> nomPropriétaire est FAIBLE

Dépendances fonctionnelles

Une df INTRA : $A \multimap B$

est **directe** s'il n'existe aucun attribut non clé (ou ensemble d'attributs non clé) C différent de B tel que :

$A \multimap C$ et $C \multimap B$

Autrement dit la DF $A \multimap B$ ne peut pas s'obtenir par transitivité

Dépendances fonctionnelles

Par exemple, Dans la relation :

Voiture(noIm, marque NN , modèle NN, nomPropriétaire)

La DF NoIm \longrightarrow marque n'est pas directe,

Car il y a aussi une DF modèle \longrightarrow marque !!

Par contre la DF noIm \dashrightarrow nomPropriétaire est bien directe

Dépendances fonctionnelles

Axiomes d'Armstrong
(dépendances fortes ou faibles)

la réflexivité :

Si : $B \subseteq A$ alors : $A \twoheadrightarrow B$

Augmentation :

Si $A \twoheadrightarrow B$, alors $\forall Z \quad AZ \twoheadrightarrow BZ$

Transitivité :

Si $A \twoheadrightarrow B$ et $B \twoheadrightarrow C$ alors $A \twoheadrightarrow C$

Dépendances fonctionnelles

Axiomes d'Armstrong
(dépendances fortes ou faibles)

Pseudotransitivité :

Si $A \twoheadrightarrow B$ et $BC \twoheadrightarrow E$ alors : $AC \twoheadrightarrow E$

Décomposition :

Si $A \twoheadrightarrow B$ et $C \subseteq B$, alors $A \twoheadrightarrow C$

Union :

Si $A \twoheadrightarrow B$ et $A \twoheadrightarrow C$, alors $A \twoheadrightarrow BC$

Dépendances fonctionnelles

Par les axiomes d'Armstrong certaines dépendances fonctionnelles se déduisent d'autres.

On distingue donc les dépendances explicitées DF et les dépendances déduites DF^+ .

DF^+ est la fermeture transitive de DF .

On dit que deux ensembles de DF , $DF1$ et $DF2$ sont équivalents ssi ils ont même fermeture transitive.

Dépendances fonctionnelles

L'objectif est d'obtenir une « couverture minimale » des DF nécessaires DF1, c'est à dire le plus petit ensemble DF' équivalent à DF1 , DF' vérifiera :

- Toutes les parties droites (but) de DF' sont réduites à un élément.
- Aucune partie gauche (source) de DF' ne contient d'élément redondant (ou inutile pour obtenir le même but) :
 $\forall A \rightarrow B \in DF1 \text{ et } A' \subset A, (DF' \setminus \{A \rightarrow B\} \cup \{A' \rightarrow B\})^+ \neq DF'^+$
- Il n'y a pas de dépendances superflues :
 $\forall A \rightarrow B \in DF1, (DF1 \setminus \{A \rightarrow B\})^+ \neq DF'^+$

Dépendances fonctionnelles

Algorithme d'obtention d'une couverture minimale :

1. Pour enlever les parties droites à un élément on expose toutes les DF $A \rightarrow B_1, \dots, B_n$ en $A \rightarrow B_1, \dots, A \rightarrow B_n$
2. Pour enlever les éléments redondants de gauche on essaye de les enlever en les prenant tour à tour
3. Pour enlever les dépendances superflues on cherche à en enlever tour à tour.

Le tout sans évidemment modifier la fermeture transitive de l'ensemble de DF initial.

Dépendances fonctionnelles

Fermeture d'un ensemble d'attributs :

Soit une relation définie par A un ensemble d'attributs et DF un ensemble de dépendances fonctionnelles.

La fermeture B^+ d'une partie $B \subset A$ relativement à $DF1 \subset DF$ est l'ensemble des attributs impliqués par $DF1$:

$$B^+ = \{a \in A \text{ tq } B \rightarrow a \in DF1^+\}$$

Intérêt :

si $Y \in B^+$, alors la dépendance $A \rightarrow Y$ se déduit de $DF1$

Exemple : $A = \{B, C, D, E\}$ et $DF1 = \{B \rightarrow C, C \rightarrow D, C \rightarrow E\}$

$\{B\}^+ = \{B, C, D, E\}$, $\{C\}^+ = \{C, D, E\}$, $\{D\}^+ = \{D\}$ et $\{E\}^+ = \{E\}$

On en déduira ici que B est une clé potentielle de A .

Dépendances fonctionnelles

Algorithme d'obtention de B^+ :

Données : $R(A, DF)$, $B \subset A$ et $DF1 \subset DF$.

$B' = B$

Répéter

$B_{tmp} = B'$

Pour chaque $df\ y \rightarrow z$ de DF **Faire**

Si $y \in B$ **Alors** $B' = B' \cup \{z\}$

FinPour

Jusqu'à $B_{tmp} = B'$ ou $B' = A$

Dépendances fonctionnelles

Clé d'une relation :

Soit une relation définie par A l'ensemble des attributs et DF l'ensemble des DF la caractérisant. On dit que $K \subset A$ est une clé de la relation SSI :

$$K \rightarrow A \subset DF$$

C'est à dire qu'une clé détermine chaque t-uple de façon unique.

Dépendances fonctionnelles

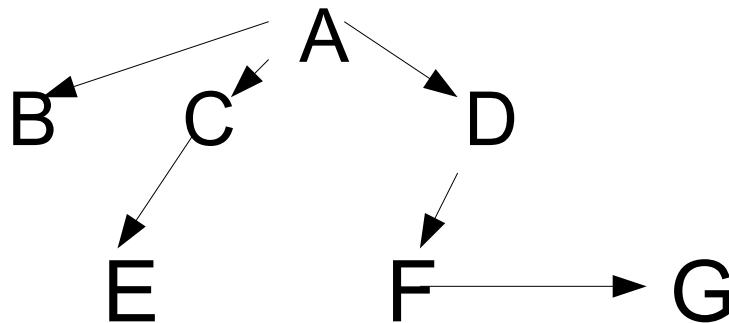
Graphe des dépendances fonctionnelles :

Pour chaque relation, on recense toutes les DF élémentaires qui ne se déduisent pas des autres.

On les représente sous forme d'un graphe orienté

Une relation peut avoir plusieurs graphes minimums : ils sont équivalents et représentent un ensemble de DF équivalentes.

Exemple :



Dépendances fonctionnelles

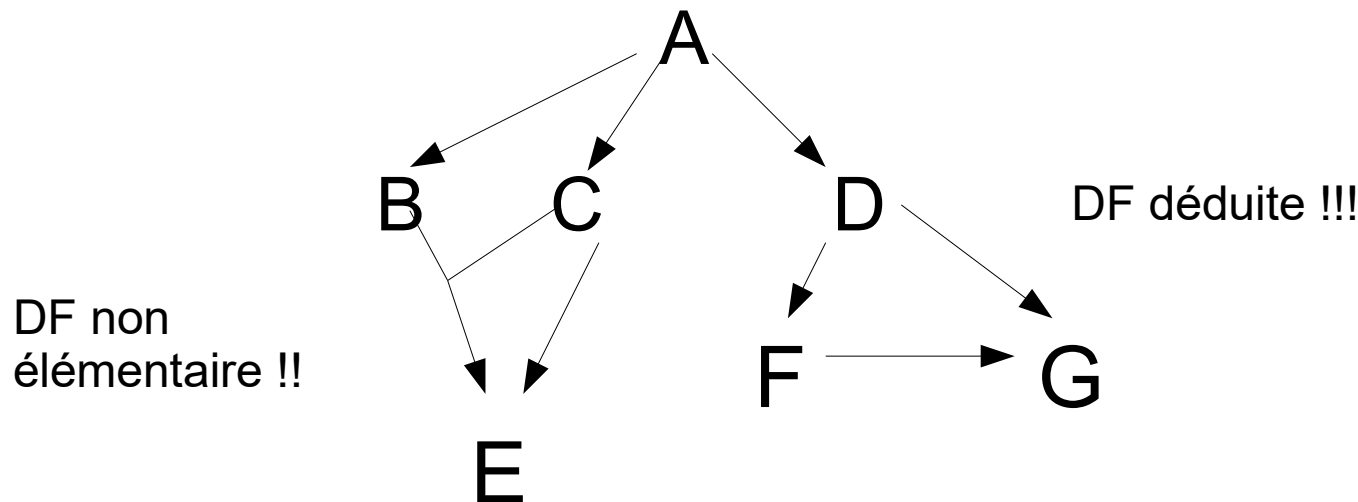
Graphe des dépendances fonctionnelles :

Il faut vérifier que le graphe est bien minimum.

Il permet de trouver les clés des relations

Il permet de tester si la décomposition est bonne, en en trouver une si nécessaire

Exemple de graphe non minimum :



Dépendances fonctionnelles

On obtiendra une décomposition sans perte d'information d'une relation $R(A_1, \dots, A_n, DF)$ en $R(U_1)$ et $R(U_2)$ si :

$$U_1 \cup U_2 = \{A_1, \dots, A_n\} \text{ et } R(U_1) \bowtie R(U_2) = R(A_1, \dots, A_n)$$

Ou bien de façon équivalente :

$$U_1 \cap U_2 \rightarrow U_1 - U_2 \in DF^+ \text{ ou } U_1 \cap U_2 \rightarrow U_2 - U_1 \in DF^+$$

Cela veut dire plus concrètement que l'on garde la possibilité de faire les mêmes requêtes avec la décomposition.

(via une jointure qui exprime un contrôle de cohérence).

Dépendances fonctionnelles

On obtiendra une décomposition préservant les DF si :

$$DF_1+ \cup DF_2+ = DF+$$

Un choix de décomposition sans perte d'information et préservant les dépendances permet uniquement de limiter les contrôles de cohérence à faire lorsque l'on modifie les tables.

Dépendances fonctionnelles

Comment faire une décomposition valide ?

Tout schéma relationnel $R(A,B,C)$ où A, B et C sont des ensembles d'attributs est décomposable en $R_1 = \pi_{A,B}(R)$ et $R_2 = \pi_{A,C}(R)$ s'il existe une dépendance fonctionnelle $A \rightarrow B$.

On aura : $R(A,B,C) = R_1(A,B) \bowtie R_2(A,C)$

Les requêtes posées sur R et celles posées sur $R_1 \bowtie R_2$ donneront le même résultat.

Dépendances fonctionnelles

Exemple : préservation des dépendances

sur la relation $R(F,V,C)$ avec $DF = (F \rightarrow V, V \rightarrow C)$

La décomposition $\{FV, VC\}$ préserve t-elle les dépendances ?

$$DF[\{FV\}] = \{F \rightarrow V\}$$

$$DF[\{VC\}] = \{V \rightarrow C\}$$

donc on a bien $DF[\{FV\}]^+ \cup DF[\{VC\}]^+ = DF^+$

Dépendances fonctionnelles

Ce choix de décomposition valide qui préserve les dépendances permet de limiter les contrôles de cohérence à effectuer lors des modifications sur les tables.

Ces deux critères ne sont cependant pas suffisants pour assurer que une telle décomposition évite des anomalies de mise à jour.

Par exemple : $R(F,V,C,B)$ et $DF = \{F \rightarrow V, V \rightarrow C, B \rightarrow F\}$ décomposée en $\{F,V,C\}$ et $\{B,F\}$ est une décomposition valide et préserve les DF, mais on va voir qu'elle n'est pas en 3NF.

Formes Normales

Normalisation d'un schéma relationnel

Il s'agit de transformer un schéma relationnel $R1$ et un autre $R2$ tel que :

- $R1$ et $R2$ sont équivalents (même contenu)
- Les mises à jour sont simple (cad qu'une mise à jour d'un changement élémentaire dans le monde réel ne conduise qu'à la modification d'un t-uple)

Formes Normales

Une relation 1NF représente un ensemble de tuples associés à un ensemble d'attributs, donc possède une « **clé** » permettant de repérer un tuple et un seul (à la limite le t-tuple entier).

Une **clé « primaire »** est **minimale**, c'est-à-dire, comporte un nombre minimal d'attributs.

Chaque relation 1NF possède une clé primaire, qui, dans les cas simples, sera la seule clé minimale. Cette clé primaire peut être mono-attribut ou multi-attribut. Un attribut de la clé primaire est appelé **attribut-clé**, un autre attribut est appelé **attribut non clé**.

1NF = valeurs atomiques typées + clé + aucun ordre, ni sur les tuples, ni sur les attributs

Formes Normales

Problème non résolu : dans le cas d'une clé multiattribut (Immat,NomClient) : DF non élémentaire

Immat	NomClient	marque	modèle	VilleClient	FraisLivraison
237E98	Plouc	Jaguar	ZE	Grand Champ	100
576T56	De Kerziziou	Renault	Clio	Rennes	300
934Y76	DuSchmoll	Peugeot	407	Rennes	300
196F45	Dupont	Renault	Clio	Auray	150
27OP2	De Kerziziou	peugeot	408	Rennes	300

Problème : redondance d'information.

La ville Client du Client De Kerziziou est mentionnée deux fois, ce qui pose problème si ce client change d'adresse : la modification est moins aisée.

Formes Normales

Une relation **2NF** (**en deuxième forme normale**) est une relation 1NF telle que :

les dépendances fonctionnelles non triviales de source égale à la clé primaire **sont toutes élémentaires**.

Un schéma est 2NF si toutes ses relations sont 2NF.

Formes Normales

Problème : la transitivité

Immat	marque	modèle
237E98	Jaguar	ZE
576T56	Renault	Clio
934Y76	Peugeot	407
196F45	Renault	Clio

NomClient	VilleClient	FraisLivraison
Plouc	Grand Champ	100
De Kerziziou	Rennes	300
DuSchmoll	Rennes	300
Dupont	Auray	150

Imaginons que les frais de livraison changent sur la ville de Rennes.

Encore une fois la mise à jour ne sera pas aisée sur la seconde table.

De même pour la première table : imaginons que Renault fasse un nouveau modèle, la clio2, du coup les clios deviennent clio1 ...

Formes Normales

Une relation **3NF (en troisième forme normale)** est une relation 2NF telle que :

les dépendances fonctionnelles non triviales de source égale à la clé primaire **sont toutes directes** (pas de transitivité).

Un schéma est 3NF si toutes ses relations sont 3NF.

Formes Normales

Solution :

Immat	modèle
237E98	ZE
576T56	Clio
934Y76	407
196F45	Clio

modèle	marque
ZE	Jaguar
Clio	Renault
407	Peugeot

NomClient	VilleClient
Plouc	Grand Champ
De Kerziziou	Rennes
DuSchmoll	Rennes
Dupont	Auray

VilleClient	FraisLivraison
Grand Champ	100
Rennes	300
Auray	150

Formes Normales

Problème : la relation Commande (qui est en 3NF)

<u>Immat</u>	<u>NumClient</u>	DateLivraison	AdresseLivraison
237E98	1610	12/05/2012	10 rue Henri IV Grand Champ
576T56	2012	08/04/2012	34 rue des élections Rennes
934Y76	1789	01/04/2012	12 rue de la révolution Rennes
196F45	2012	15/04/2012	34 rue des élections Rennes

IL y a encore ici redondance d'informations (entre le NumClient et l'adresse de livraison) :
Il y a une DF entre AdresseLivraison et NumClient , cad entre un attribut non clé et une partie de la clé.

Formes Normales

Une relation est en BCNF (Forme Normale de Boyce-Codd) ssi :

les seules dépendances fonctionnelles sont celles dans lesquelles une clé détermine un attribut

tous les attributs non-clé ne sont donc pas source de DF vers une partie de la clé