

La normalisation du modèle relationnel (et du MCD)

L'objectif de la normalisation est de construire un schéma de base de données cohérent. Un mauvais schéma logique peut conduire à un certain nombre d'anomalies pendant la phase d'exploitation de la base de donnée. Nous allons voir ces anomalies dans une première partie. Pour qu'un modèle relationnel soit normalisé, il faut qu'il respecte certaines contraintes appelées les formes normales. Les formes normales s'appuient sur les dépendances fonctionnelles entre attributs.

I. Rappels sur la notion de dépendance fonctionnelle

La construction du MCD mais également du modèle relationnel correspondant, repose presque entièrement sur le concept de dépendance fonctionnelle. C'est ce concept qui permet de passer d'un ensemble de propriétés non structuré à un modèle conceptuel des données formé d'entités et d'associations et au modèle relationnel correspondant.

Définition

On dit que b est en dépendance fonctionnelle (DF) de a si à une valeur quelconque de la propriété a, on ne peut faire correspondre qu'**une seule** valeur au plus de la propriété b.

On note $a \xrightarrow{\text{(source)}} b$
(source) \rightarrow (but)

Autrement dit, si on connaît la valeur de a, on peut en déduire **une seule valeur de b**. Mais la réciproque n'est pas vrai (si on connaît b, on ne peut pas en déduire a).

Exemple :

Num client $\xrightarrow{\quad} \text{Nom client}$

Il existe une DF entre num client et Nom client, car si on connaît une valeur de la propriété num client (ex : 4553), il ne peut lui correspondre qu'une seule valeur de la propriété nom (ex : Duval).

La réciproque est fausse :

Nom client $\xrightarrow{\quad} \text{Num client}$
n'est pas une DF

Si l'on connaît la valeur de la propriété Nom client, on ne peut pas en déduire la propriété Num client, car il peut y avoir des homonymes.

Terminologie

Si on a une dépendance fonctionnelle $a \xrightarrow{\quad} b$, on peut employer les expressions suivantes de façon équivalente :

- ? Il y a une dépendance fonctionnelle de a vers b
- ? b est en dépendance fonctionnelle de a
- ? b dépend fonctionnellement de a
- ? a est la source et b est le but (ou la cible) de la dépendance fonctionnelle

DF à partir de propriétés concaténées (partie gauche composée de plusieurs attributs)

Il peut exister des dépendances fonctionnelles à partir de propriétés concaténées, c'est-à-dire qui forment un tout indissociable, comme si elles étaient soudées.

On note par un + cette concaténation.

Exemple : Considérons une commande qui comporte plusieurs produits
 Num_Commande + Ref_Produit \longrightarrow quantité commandée

Si on n'a seulement le numéro de la commande, on ne peut pas en déduire la quantité commandée, car il faut aussi savoir de quel produit. De même, on ne peut pas savoir la quantité commandée d'un produit si on ne sait pas de quelle commande. Il faut bien connaître à la fois la commande et le produit (leurs identifiants respectifs) pour en déduire la quantité commandée.

Les dépendances fonctionnelles dont la source est formée de plusieurs propriétés doivent être **élémentaires**, c'est-à-dire ne pas être créées artificiellement.

Ex : Num Commande + Num Client $\not\rightarrow$ date commande

n'est pas une DF élémentaire car on n'a pas besoin du numéro de client pour connaître la date de commande, il suffit de connaître le numéro de la commande. La propriété Num Client ne sert à rien dans cette DF.

$\not\rightarrow$ Propriétés des dépendances fonctionnelles

Les dépendances fonctionnelles ont les propriétés suivantes :

○ Union :

Si on a deux DF ayant la même source, on peut les rassembler en une seule, en séparant les cibles par une virgule.

Si $a \rightarrow b$ et $a \rightarrow c$ alors on peut écrire que $a \rightarrow b, c$

Ex : Référence \rightarrow Désignation et Référence \rightarrow Prix de vente unitaire

Alors par union on a : Référence \rightarrow Désignation, Prix de vente unitaire

Lors du tracé du graphe des dépendances fonctionnelles, l'union permet de regrouper sur une seule ligne toutes les dépendances fonctionnelles ayant la même source.

○ Transitivité :

Si $a \rightarrow b$ et $b \rightarrow c$ alors on a $a \rightarrow c$

Ex : Num Médecin \rightarrow Code Service et Code Service \rightarrow Num Hopital

Alors on a Num Médecin \rightarrow Num Hopital

Les DF qui peuvent être déduites par transitivité de deux autres DF (qui ne sont pas directes) doivent être éliminées car elles sont alors redondantes.

Il ne reste alors que les **DF directes**, c'est-à-dire celles qui ne peuvent pas être retrouvées par transitivité.

$\not\rightarrow$ Attention toutefois à la signification des dépendances. Une dépendance fonctionnelle qu'on peut retrouver par transitivité ne doit pas être supprimée si elle n'a pas le même sens que la transitivité des deux autres, car il y aurait perte d'information.

II. L'intérêt de la normalisation

Pour vous montrer l'intérêt de la normalisation d'une base de donnée relationnelle, voyons les problèmes que peuvent poser l'utilisation d'une base de donnée basée sur un modèle relationnel non normalisé.

Soit le schéma de relation

FOURNISSEUR (NomFournisseur, AdresseFournisseur, Produits, Prix)

Modèle en extension

NomFournisseur	AdresseFournisseur	Produit	Prix
Lebras	10, Rue des Gras - Clermont	Chaise	20
		Table	35
Dupont	86, Rue de la République - Moulins	Bureau	60
Lajoie	26, Rue des Dômes - Vichy	Lit	50
Dupont	39, Rue des Buttes - Moulins	Lampe	18
		Table de chevet	25

1° problème :

Il n'y a pas de clé primaire : on ne sait pas si les deux Dupont sont différents ou pas (si c'est le même Dupont, il y a une des deux adresses qui est fausse).

2° problème :

L'adresse n'est pas décomposée. Si on veut par exemple rechercher tous les fournisseurs qui habitent la même ville, ça ne va pas être possible

3° problème :

Une relation (table) correspondant à ce schéma pourra éventuellement contenir plusieurs produits pour un même fournisseur.

Dans ce cas, il faudra faire face à un certain nombre de problèmes :

- ✍ l'adresse du fournisseur sera dupliquée dans chaque n-uplet (redondance),
- ✍ si on souhaite modifier l'adresse d'un fournisseur, il faudra rechercher et mettre à jour tous les n-uplets correspondant à ce fournisseur,
- ✍ si on insère un nouveau produit pour un fournisseur déjà référencé, il faudra vérifier que l'adresse est identique,
- ✍ si on veut supprimer un fournisseur, il faudra retrouver et supprimer tous les n-uplets correspondant à ce fournisseur (pour différents produits) dans la table.

✍ **La normalisation élimine les redondances**, ce qui permet :

- une diminution de la taille de la base de donnée sur le disque
- une diminution des risques d'incohérence
- d'éviter une mise à jour multiple des mêmes données

III. Les 3 formes normales

A. 1° forme normale :

Une relation est normalisée en première forme normale si :

- 1) elle possède une clé identifiant de manière unique et stable chaque ligne
- 2) chaque attribut est monovalué (ne peut avoir qu'une seule valeur par ligne)
- 3) aucun attribut n'est décomposable en plusieurs attributs significatifs

Contre-exemple :

EMPLOYE (Nom, Prénom, Enfants, Diplômes)

Cette relation n'est pas en première forme normale

NOM	DIPLOMES		ENFANTS	
	Nature	Année	Prénom	Année de naissance
DUPONT	Bac	1975	Sophie	1985
	Licence	1980	Simon	1993
			Lucie	1996

Un employé peut avoir plusieurs enfants et plusieurs diplômes. En outre, ces attributs sont décomposables : diplôme est décomposable en Nature et Année, et Enfants est décomposable en Prénom et Année de Naissance.

B. 2° forme normale

Une relation R est en *deuxième forme normale* si et seulement si :

✍ elle est en 1FN

✍ et tout attribut non clé est totalement dépendant de toute la clé.

Autrement dit, aucun des attributs ne dépend que d'une partie de la clé.

La 2FN n'est à vérifier que pour les relations ayant une clé composée. Une relation en 1FN n'ayant qu'un seul attribut clé est toujours en 2FN

Contre-exemple :

LIGNE_COMMANDE(#Num_cde, #RéférenceProd, DésignationProd, Quantité)

Cette relation est en première forme normale (existence d'une clé valide et aucun attribut n'est décomposable)

MAIS elle n'est pas en 2° forme normale car on a DésignationProd ne dépend pas de toute la clé mais seulement de RéférenceProd:

RéférenceProd → DésignationProd

pour connaître l'attribut désignationProd, on n'a pas besoin de connaître le numéro de commande.

C. 3° forme normale

Une relation est en 3° forme normale si et seulement si :

✍ elle est en 2° forme normale

✍ et tout attribut doit dépendre **directement** de la clé, c'est-à-dire qu'aucun attribut ne doit dépendre de la clé par transitivité.

Autrement dit, aucun attribut ne doit dépendre d'un autre attribut non clé.

Contre-exemple :

CLIENT(Num_client, Nom_client, code_categ, nom_categ)

Cette relation n'est pas en 3FN car $\text{num_client} \longrightarrow \text{nom_categ}$ n'est pas une dépendance directe. En effet, on a aussi $\text{num_client} \longrightarrow \text{num_categ} \longrightarrow \text{nom_categ}$

D. Application des règles





Si l'une des 3 règles n'est pas vérifiée, cela indique une erreur dans le modèle relationnel et il faut alors modifier pour que les 3 règles soient vérifiées pour toutes les relations.

On vérifie les règles dans l'ordre. Si la première forme normale n'est pas respectée, pas la peine de vérifier la 2FN. Et si la 2FN n'est pas vérifiée, inutile de vérifier la 3FN.

Il existe d'autres formes normales mais on admet couramment que ces 3 premières formes normales sont suffisantes pour permettre de construire des modèles fiables et cohérents. (De plus, ces autres formes normales ne sont pas au programme du BTS)

Résumé

Modèle normalisé = relations avec

-  **une clé, qui permet de distinguer chaque occurrence**
-  **des attributs élémentaires (1FN)**
-  **en dépendance de TOUTE la clé (2FN),**
-  **et RIEN QUE de la clé (3FN)**

On parle aussi de normalisation pour un MCD. Un MCD qui donne un MR normalisé est qualifié aussi de normalisé.