Normalisation

Site: LMS UCA 2021/2022
Cours: EIIN512B - ECUE Bases de donnees relationnelles
Livre: Normalisation

Imprimé par: latapie Florian
Date: vendredi 14 janvier 2022, 00:10

Table des matières

1. Introduction

Le but essentiel de la normalisation est d'éviter les anomalies transactionnelles pouvant découler d'une mauvaise modélisation des données et ainsi éviter un certain nombre de problèmes potentiels tels que les anomalies de lecture, les anomalies d'écriture, la redondance des données et la contre-performance.

La normalisation des modèles de données permet de vérifier la robustesse de leur conception pour améliorer la modélisation (et donc obtenir une meilleure représentation) et faciliter la mémorisation des données en évitant la redondance et les problèmes sous-jacents de mise à jour ou de cohérence.

Les formes normales s'emboîtent les unes dans les autres, tant et si bien que le respect d'une forme normale de niveau supérieur implique le respect des formes normales des niveaux inférieurs.

Dans ce cours, nous nous limiterons à:

1. la première forme normale notée 1FN (1NF en anglais) ;
2. la deuxième forme normale notée 2FN (2NF en anglais) ;
3. la troisième forme normale notée 3FN (3NF en anglais) ;
4. la forme normale de Boyce Codd notée FNBC (BCNF en anglais) ;

Merci wikipedia, pour cette introduction....

2. Dépendances Fonctionnelles

Dans une relation, certains attributs en "déterminent" d'autres, i.e., il n'y a pas deux tuples ayant les mêmes valeurs pour le premier ensemble d'attributs sans avoir également les mêmes valeurs pour le deuxième ensemble. On parle alors de dépendance fonctionnelle entre les deux ensemble d'attributs.
Les dépendances fonctionnelles sont fondamentales pour éliminer les redondances. Elles sont associées au schéma et non à une instance particulière. Elles servent à déterminer les clés d'une relation et à décomposer l'information de manière normalisée pour éviter les redondances Formellement,

Soient r une instance de la relation R, X et Y deux sous-ensembles d'attributs de R.

On dit que **r satisfait la dépendance fonctionnelle $X \rightarrow Y$**

ssi $\forall\ t1,t2 \in r\ (t1.X = t2.X \Rightarrow t1.Y = t2.Y)$

Théorème:

Si une relation r satisfait la dépendance fonctionnelle $X \rightarrow Y$, et si Attr(r)-Y =U

$r = \prod_{XY}(r) \bowtie \prod_{U}(r)$

On utilisera ce théorème, pour décomposer la table r en deux tables $\prod_{XY}(r)$ et $\prod_{U}(r)$.

Parmi les dépendances fonctionnelles, on distingue les dépendances élémentaires et les dépendances triviales.

$X \rightarrow Y$ est une DFE (**dépendance fonctionnelle élémentaire**)si et seulement si :

- $X \cap Y = \emptyset$
- Pour tout sous ensemble strict Z de X , on n'a pas $Z \rightarrow Y$.

Une dépendance fonctionnelle $X \rightarrow Y$ est **triviale** si Y est inclus dans X

2.1. Dépendances Fonctionnelles déduites

Soit la relation ENREG={NumE, Pays, NomM, Classe, Date, IdDep}

Si l'on sait qu'elle respecte ces dépendances fonctionnelles

df1 : NumE,Pays → NomM,Date

df2 : NumE,Pays → Classe,IdDep

df3 : NomM,Pays,Classe →NumE

On peut en déduire qu'elle respecte aussi les dépendances fonctionnelles

df4 : NumE,Pays → NomM,Date,Classe,IdDep

df5 : NomM,Pays,Classe → NumE,Date,IdDep

Les manipulations possibles:

On peut déduire une dépendance fonctionnelle d'un ensemble de dépendances fonctionnelles si l'on applique l'une des règles suivantes

Réflexivité	$\emptyset \Longrightarrow X \rightarrow X$ ($\emptyset$ signifie que la partie droite est toujours vraie)
Augmentation	$X \rightarrow Y \Longrightarrow XZ \rightarrow YZ$
Transitivité	$X \rightarrow Y, Y \rightarrow Z \Longrightarrow X \rightarrow Z$
Addition	$X \rightarrow Y, X \rightarrow Z \Longrightarrow X \rightarrow YZ$
Projection	$X \rightarrow YZ \Longrightarrow X \rightarrow Y, X \rightarrow Z$
Pseudo transitivité	$X \rightarrow Y, YZ \rightarrow W \Longrightarrow XZ \rightarrow W$

Implication d'ensembles de dépendances fonctionnelles:
Soient DF et DF' deux ensembles de dépendances fonctionnelles définies sur un schéma de relation R.

On dit que DF implique DF ssi pour toute instance r de la relation R, on a

r satisfait DF $\Longrightarrow$ r satisfait DF'

Exemple
DF={A → B, A → C} $\Longrightarrow$DF'= {A → BC}

Deux ensembles de dépendances fonctionnelles sont **équivalents** s'ils s'impliquent mutuellement

3. Clé candidate

Définitions :

{A1,...,An} est une **clé candidate** d'une relation r si et seulement si :

- Les attributs {A1,...,An} déterminent fonctionnement tous les autres attributs de la relation r
- Aucun sous ensemble strict de {A1,...,An} ne détermine fonctionnellement tous les autres attributs de r (la clé doit être minimale)

Super clé : ensemble d'attributs qui contient une clé candidate

Remarque :On parle de clé candidate, car il peut y en avoir plusieurs. En SQL une seule sera déclarée comme clé primaire. Mais **pour normaliser il est important de considérer toutes les clés candidates.**

On dit qu'un attribut A est **primaire** dans R si et seulement si A appartient à au moins une clé candidate de R.

On dit qu'un attribut A est **non primaire** dans R si et seulement si A n'est élément d'aucune clé candidate de R.

Soit {A1,...,An} un ensemble d'attributs et S un ensemble de dépendances fonctionnelles.

La fermeture de {A1,...,An} par S est l'ensemble d'attributs B maximal tel {A1,...,An} → B

On note la fermeture $\{A1,...,An\}^+$

Exemple: S={A→ D; A→ E; E→C} Fermeture de A par S: $\{A\}^+ = \{A,D,E,C\}$

Algorithme de saturation:

1. X ={A1,...,An}
2. Rechercher une dépendance de S de la forme B1,...,Bn →C tel que {B1,...,Bn} soient dans X mais non C . Ajouter C dans X
3. Répéter l'étape 2 jusqu'au point fixe de X (plus rien ne peut être ajouté à X)
4. X est égal à la fermeture $\{A1,...,An\}^+$

Théorème $\{A1,...,An\}^+$ est l'ensemble de tous les attributs d'une relation si et seulement si A1,...,An est une super clé de la relation considérée.

Corollaire : On peut tester si {A1,...,An} est une clé candidate d'une relation R en vérifiant que {A1,...,An} est une super clé et qu'aucun sous-ensemble strict de {A1,...,An} n'est une super clé de R

4. Première forme normale

On dit qu'un schéma relationnel R est en première forme normale (1NF) si et seulement si les valeurs des attributs sont atomiques (ni set_of, ni list_of,....)

C'est à dire si chaque attribut contient une seule valeur

Exemple de schéma non en 1NF:

Titre	Acteurs
Casablanca	Humphrey Bogart, Ingrid Bergman
Perfect World	Kevin Costner, Clint Eastwood
The Terminator	Arnold Schwarzenegger, Linda Hamilton, Michael Biehn
Die Hard	Bruce Willis

5. Deuxième forme Normale

On dit que **R est en deuxième forme normale** (2NF) ssi :

- Elle est en 1NF
- Aucun attribut non primaire ne dépend fonctionnellement d'un sous ensemble strict d'attributs d'une clé candidate

Autrement dit toute dépendance de la forme clé candidate → attribut non primaire est élémentaire

Exemple de relation qui n'est pas en 2NF

joueur(Personne, Sport, Taille, Poids)

ON a les deux dépendances fonctionnelles Personne → Taille, et Personne →Poids

Il y a donc une seule clé candidate:(Personne, Sport)

Les deux dépendances violent la 2NF. En conséquence, la table joueur contient des redondances car le même triplet (personne, taille, poids) va apparaître autant de fois que la personne pratique de sports.

Pour normaliser on applique le théorème de décomposition en suivant la dépendance fonctionnelle

$$\text{Personne} \rightarrow \text{Taille,Poids}$$

On obtient les deux tables
physique(Personne, Taille, Poids)

pratique(Personne, Sport)
où
pratique= $\prod_{\text{Personne, Sport}}$(joueur) et
physique= $\prod_{\text{Personne,Taille, Poids}}$(joueur)

On a joueur= pratique ⋈ physique et les deux tables pratique et physique vérifient toutes les deux la 2NF.

6. Troisième forme normale

Intuition : Une relation est en troisième forme normale si et seulement si elle est en 2NF et que les attributs non primaires sont mutuellement indépendants

Définition 1:

On dit qu'un schéma relationnel R est en troisième forme normale (3NF) si et seulement si :

Chaque dépendance fonctionnelle X → {A} (on peut toujours se ramener à un seul attribut à droite)vérifie au moins une de ces conditions :

- X est une superclé
- A ∈X (dépendance fonctionnelle triviale)
- A est primaire

Définition 2 : (équivalente à la première bien sur)
On dit qu'un schéma relationnel R est en troisième forme normale (3NF) si et seulement si si :

Chaque dépendance fonctionnelle X → Y, (cette fois ci Yest un ensemble d'attribut) vérifie au moins une de ces conditions :

- La dépendance fonctionnelle est triviale
- X est une superclé
- Tous les attributs de Y-X sont primaires

Exemple de table en 2NF mais pas en 3NF

VOITURE (NVH, TYPE, MARQUE, PUISS, COULEUR)

Une clé (NVH), et aussi la dépendance fonctionnelle TYPE → MARQUE , PUISS

En décomposant selon la dépendance fonctionnelle, on obtient deux tables en 3NF

MODELE (TYPE, MARQUE, PUISS)

VEHICULE (NVH, TYPE, COULEUR)

7. BCNF

La 3NF n'élimine pas toutes les redondances:

Exemple :

RELATION VIN (CRU, PAYS, REGION)

CRU	PAYS	REGION
CHENAS	France	BEAUJOLAIS
JULIENAS	France	BEAUJOLAIS
CHABLIS	France	BOURGOGNE
CHABLIS	USA	CALIFORNIE

avec les dépendances fonctionnelles: CRU,PAYS → REGION et REGION → PAYS

Il y a deux clés candidates CRU,PAYS et CRU,REGION

La relation est en 3NF car CRU PAYS est une clé candidate et PAYS est primaire.

Cependant il y redondace de l'information Beaujolais est en France par exemple

Pour éviter ce type de redondance on défini la BCNF

Intuition :

Une relation R est sous forme normale de Boyce-Codd si et seulement si chacun des attributs ne dépend fonctionnellement que des clés entières.

Définition formelle:

Pour chaque dépendance fonctionnelle X → {A}, (un seul attribut à droite) au moins une de ces conditions est vraie:

- X est une superclé
- A∈X (DF triviale)

VIN (CRU, PAYS, REGION))eut etre décomposé en deux tables en BCNF

1. CRUS (CRU, REGION)

2. REGIONS (REGION, PAYS)

On a perdu la DF: (CRU, PAYS) → REGION

Remarque : pour qu'une relation soit en 3NF mais pas en BCNF il faut qu'elle ait au moins deux clés candidate

8. Algorithmes de normalisation

Objectif : décomposer une relation non normalisée en sous relations normalisées, si possible sans perte :

- La jointure naturelle des sous relations est la relation de départ
- On préfèrerait ne pas perdre les dépendances fonctionnelles

8.1. Algorithme de normalisation en 3NF

Algorithme de normalisation en 3NF

Soit R(A1,A2,...An) satisfaisant l'ensemble de Dépendances Fonctionnelles S

- Calculer Smin équivalent à S qui est une couverture minimale de S(voir algo un peu plus loin)
- Regrouper toutes les dépendances ayant la même partie gauche en une seule
- Créer une relation par dépendance (dont les attributs sont tous ceux de la dépendance)
- Si aucune de ces relations ne contient une des clés candidates de R, ajouter une dernière relation dont les attributs sont ceux d'une clé candidate de R

Le dernier point est indispensable pour assurer que la jointure des tables produites soit bien égale à la table initiale.
Il suffit qu'une des clés candidates soit incluse entièrement dans une des tables du résultat pour que ce soit le cas. Cet algorithme n'utilise pas le théorème de décomposition selon une dépendance fonctionnelle

Théorème :
L'algorithme de décomposition en 3NF est sans perte d'information , et sans perte de dépendances fonctionnelles.

8.2. Algorithme de recherche d'une couverture minimale

Algorithme de recherche d'une couverture minimale

- Minimiser à droite (remplacer chaque dépendance fonctionnelle par plusieurs dépendances fonctionnelles chacune avec un seul attribut à droite)
- Minimiser à gauche chaque DF (peut on réduire la partie gauche ?)
- Elimination des redondances: si on peut déduire une dépendance fonctionnelle des autres on l'enlève

Exemple d'application de l'algorithme

S= {A → B; B C → D ;A C → B D E ;D → E}

Etape 1 : minimiser a droite

A → B ;B C → D ;A C → B ;A C → D ;A C → E ;D → E

Etape 2 : minimiser à gauche

A → B ;B C → D ; A C → D ;A C → E ;D → E

A C → B est éliminé car peut etre réduite à A → B que l 'on a déjà

Etape 3: élimination des redondances:

A → B ;B C → D ;D → E ;

En effet de A → B ;B C → D on peut déduire A C →BC→ D et

de A → B ;B C → D ;D → E on peut déduire A C → E.

8.3. Exemples de mises en 3NF

Premier exemple:

R(Cours,Prof, Heure,Salle,Eleve,Note) avec les dépendances fonctionnelles

- Cours → Prof
- Heure, Salle→ Cours
- Heure,Prof → Salle
- Cours, Eleve → Note
- Heure, Eleve → Salle

Une unique clé candidate : Eleve, Heure

- Ces dépendances fonctionnelles forment une couverture minimale.
- Il n'y a rien à regrouper
- On décompose la relation en 5 relations
- Pas besoin d'une sixième car une des tables contient la clé candidate

Deuxième exemple

R(A,B,C) avec deux dépendances fonctionnelles

AB → C et B → C

Une seule clé candidate : AB

Minimisation des dépendances B → C

On obtient R1(BC)

On ajoute R2(AB) car R1 ne contient pas la clé candidate.

9. Algorithme de normalisation en BCNF

Algorithme de normalisation en BCNF

D←{R}

Tant qu'il existe Ri dans D qui n'est pas en BCNF

Trouver X -> Y une dépendance non triviale sur Ri qui viole la BCNF

Décomposer Ri en deux relations:

- Ri1(X U Y)
- Ri2(Attr(Ri)-Y)

D←D-{Ri} U {Ri1, Ri2}

Théorème : Il meut y avoir des pertes de dépendances et plusieurs résultats sont possible selon l'ordre de décomposition choisi.

En effet, comme on décompose toujours selon une Dépendance Fonctionnelle, la jointure des nouvelles tables est toujours égale à la table initiale. Mais il est possible qu'il n'y ai plus de table pour « porter » une DF, c'est à dire pas de table comportant tous les attributs de la DF.

9.1. Exemple d'utilisation de l'algorithme

Soit la relation R(A,B,C,D,E) qui vérifie les dépendances fonctionnelles

A → B ;A → C ;CD → E ;B → D

Cette relation a une seule clé candidate: A

Cette relation n'est pas en BCNF(pas en 3NF en fait) a cause des deux dernières dépendances fonctionnelles

On choisit de commencer la décomposition selon la dépendance fonctionnelle B → D.

On obtient les deux tables

- R1(B,D) qui est en BCNF [son unique clé est B] c'est B → D qui le dit
- R2(A,B,C,E) dont l'unique clé est A. A cause de CB → E (et oui à cause de CB→ CD → E, et oui il faut vraiment s'interroger sur les DF vérifiées par les nouvelles tables) cette table n'est pas en BCNF

On va donc a nouveau décomposer R2 selon CB → E
et obtenir les deux tables
- R21(B,C,E) cle BC en BCNF
- R22(A,B,C) clé A en BCNF

Au final on a décomposé R en trois tables dont on a souligné la clé candidate (elles en on une seule chacune)

- R1(B,D)
- R21(B,C,E)
- R22(A,B,C)

Remarque 1: la dépendance fonctionnelle CD→ E est perdue

Remarque 2 : Une autre décomposition était possible
On considère d'abord CD → E : on obtient

- R'1(C,D,E), en BCNF
- R'2(A,B,C,D) clé A (et pas d'autres) pas en BCNF à cause de B → D,

on décompose donc R'2 en
- R'21(BD) en BCNF
- R'22(ABC) en BCNF

Cette décomposition ne perd aucune dépendance fonctionnelle: c'est aussi celle qu'aurait donné l'algo de normalisation en 3NF