Base de données de la rédaction d'un journal

Théo Delalande-Delarbre Thomas Minier
Benjamin Sientzoff

30 janvier 2015

Université de Nantes - Licence 3 Informatique
X6I0050 Base de données 2

Table des matières

Introduction

Dans le cadre du cours *Base de données 2*, il nous a été demandé de réaliser, par groupe de trois, une base de données réaliste et réalisable en entreprise. Nous avons choisi de modéliser la rédaction d'un journal. Ensuite tout au long du modulee, nous devons ajouter dans notre base de données des applications des concept vu en cours.

Dans une première partie, nous détaillerons l'organisation de notre *trinôme* pour la réalisation de ce projet. Ensuite, nous présenterons la conception de notre base de données en partant d'une *Big table*. Nous déterminerons les différentes dépendances fonctionnelles, puis nous appliquerons deux algorithmes de normalisation sur notre table : l'algorithme de Bernstein et l'algorithme de décomposition, tous deux vus en cours.

Nous présenterons également les différentes vues que nous avons mises en place pour visualiser notre base. Nous traiterons aussi les différents *triggers* créés ainsi que les fonctions stockées pour interagir avec la base.
Enfin, nous discuterons des différents ajouts et améliorations possibles de notre base de données.

semaine	Tâche	Responsable
S4	Big table	groupe complet
S5	Rédaction rapport	groupe complet
S5	Algo de décomposition	Théo DD
S5	Algo de Bernstein	Thomas M
S5	Planning	Benjamin S
S6	Implémentation SQL des tables	groupe complet
S6	Insertion données dans BD	groupe complet
S7	Définissions des requêtes stockées	Théo DD
S8	Procédures PL/SQL	Thomas M, Benjamin S
S8	Mise à jour du planning	Groupe complet

FIGURE 1 – Répartition des tâches dans le groupe

1 Base du projet

1.1 Répartition des tâches

Le projet étant réalisé en *trinôme*, il est alors indispensable de répartir au mieux les tâches. Le tableau à la figure 1 présente la répartition des tâches dans le groupe ainsi que les dates prévisionnelles de leur réalisation.

Remarque Ce planning n'est ni complet ni exhaustif. En effet, on ne peut pas faire de spéculations sur ce qu'on va mettre dans notre base de données avant d'avoir eu les cours en relation.

1.2 Définition de la *big table*

Notre base de données va représenter la gestion des numéros d'un journal. Chaque **édition** a un numéro, une une, un prix, une date de parution, un type, un rédacteur en chef. Chaque **personne** a un numéro d'identification, un nom, un prénom, un salaire de base et un numéro de téléphone. Chaque personne a un métier (photographe, rédacteur, etc.). Un **contenu** a un titre, un type et un auteur. Dans la base de données on stockera l'emplacement des contenus. Un **article** a une date de rédaction, un titre et un résumé. Un **article** est composé d'un ensemble de **contenus** et une **édition** est composée d'un ensemble d'articles.
Le schéma à la figure 2.1 présente toutes les dépendances fonctionnelles de notre base de données.

2 Conception de la base de données

2.1 Pourquoi décomposer la table ?

Il y a au début une unique table contenant une vingtaine d'attributs. Pour illustrer la redondance et les dépendances fonctionnelles, cet exemple de tuples ne sera centré que sur quelques attributs pour des raisons de lisibilité

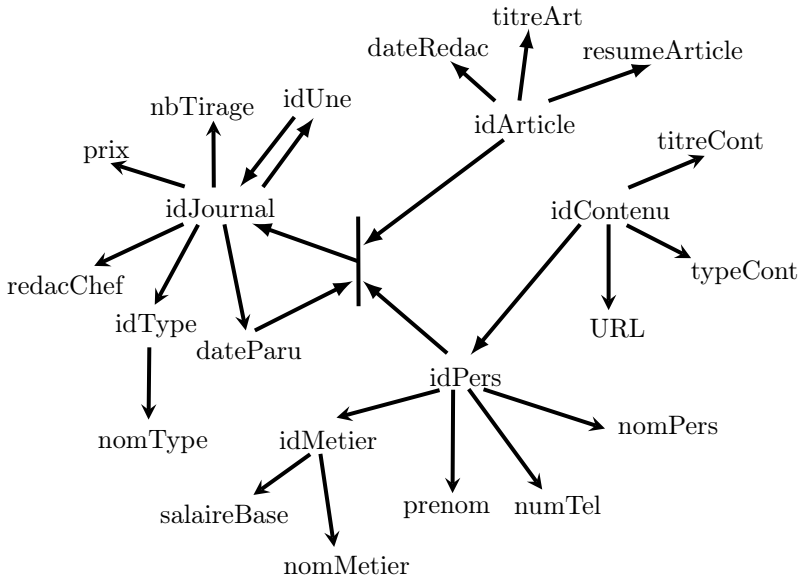

FIGURE 2 – Schéma des dépendances fonctionnelles de la base de données

Dépendances fonctionnelles Les dépendances fonctionnelles de la figure 2.1 peuvent être retranscrites de la manière suivante.

(1) idJournal → prix, nbTirage, idUne, idType, redacChef, dateParu

(2) idType → nomType

(3) idUne → idJournal

(4) idArticle → dateRedac, titreArt, resumeArt

(5) idContenu → URL, titreCont, typeCont

(6) idPers → nomPers, prenom, numTel, idMetier

(7) idMetier → salaireBase, nomMetier

(8) idArticle, idPers, dateParu → idJournal

Clés De ce schéma, on peut en déduire trois clés $\{idJournal, idArticle, idContenu\}$, $\{idUne, idArticle, idContenu\}$ et $\{dateParu, idContenu, idArticle\}$.

2.2 Algorithme de Bernstein

Une fois nos dépendances fonctionnelles établies, nous commençons par appliquer l'algorithme de Bernstein pour décomposer notre grande table. Tout d'abord, nous déterminons la couverture minimale. On utilisera la clé suivante : {idJournal, idArticle, idContenu}.

Les relations (1), (4), (5), (6) et (7) ne sont pas en forme élémentaires, nous les décomposons donc. De plus, à part la relation (8), toutes les autres relations sont qu'un seul attributs à gauche. il n'est donc pas nécessaire de calculer la fermeture de ces attributs. Il est cependant nécessaire de le faire pour la relation (8).

Calcul de la fermeture des attributs en partie gauche de la relation (8) :
- $idArticle+ = \{dateRedac, titreArt, resumeArticle\}$
- $idPers+ = \{nomPers, numTel, prenom, idMetier, nomMetier, salaireBase\}$
- $dateParu+ = \{\}$

Étant donné qu'aucun élément de chacune des trois fermetures n'apparaît en partie gauche de la relation (8), cette dernière peut être gardée telle quelle.

Au final, la couverture minimale est la suivante :

(1.1) idJournal → prix
(1.2) idJournal → nbTirage
(1.3) idJournal → idUne
(1.4) idJournal → idType
(1.5) idJournal → redacChef
(1.6) idJournal → dateParu
(2) idType → nomType
(3) idUne → idJournal
(4.1) idArticle → dateRedac
(4.2) idArticle → titreArt
(4.3) idArticle → resumeArt
(5.1) idContenu → URL
(5.2) idContenu → titreCont
(5.3) idContenu → typeCont
(6.1) idPers → nomPers
(6.2) idPers → prénom
(6.3) idPers → numTel
(6.4) idPers → idMetier
(7.1) idMetier → salaireBase
(7.2) idMetier → nomMetier
(8) idArticle, idPers, dateParu → idJournal

Nous créons ensuite nos groupes, en regroupant les dépendances ayant même partie gauche dans chaque groupe :

(1) $DF_1 = \{(1.1), (1.2), (1.3), (1.4), (1.5), (1.6)\}$
(2) $DF_2 = \{(2)\}$
(3) $DF_3 = \{(3)\}$

(4) $DF_4 = \{(4.1), (4.2), (4.3)\}$
(5) $DF_5 = \{(5.1), (5.2), (5.3)\}$
(6) $DF_6 = \{(6.1), (6.2), (6.3), (6.4)\}$
(7) $DF_7 = \{(7.1), (7.2)\}$
(8) $DF_8 = \{(8)\}$

Puis, nous construisons un schéma $< R_i(U_i), DF_i >$ pour chaque groupe DF_i, où U_i est l'ensemble des attributs apparaissant dans DF_i :

(1) $< R_1(idJournal, nbTirage, idUne, idType, redacChef, dateParu), DF_1 >$
(2) $< R_2(idType, nomType), DF_2 >$
(3) $< R_3(idUne, idJournal), DF_3 >$
(4) $< R_4(idArticle, dateRedac, titreArt, resumeArt), DF_4 >$
(5) $< R_5(idContenu, URL, titreCont, typeCont), DF_5 >$
(6) $< R_6(idPers, nomPers, prenom, numTel, idMetier), DF_6 >$
(7) $< R_7(idMetier, salaireBase, nomMetier), DF_7 >$
(8) $< R_8(idArticle, idPers, dateParu, idJournal), DF_8 >$

Pour terminer, comme la clé entière n'est présente dans aucun schéma, on rajoute un dernier schéma de la forme : $< R_9(idJournal, idArticle, idContenu), \{\} >$

Nous obtenons donc, par application de l'algorithme de Bernstein, un ensemble de relations *spi*, *spdf* et en 3ème forme normale.

2.3 Algorithme de décomposition

On considère la relation de départ :
$R(idJournal, idContenu, idArticle, titreCont, typeCont, URL, dateRedac,$
$titreArt, resumeArticle, idPers, nomPers, numTel, prenom, idMetier,$
$salaireBase, nomMetier, dateParu, idUne, nbTirage, prix,$
$redacChef, idType, nomType),$
$DF = \{(1), (2), (3), ...(8)\}$

On décompose la relation avec l'algorithme. Plusieurs résultats sont possibles : dans cette exécution nous avons arbitrairement choisi l'ordre nous permettant de conserver les sous-ensembles les plus proches de la réalité de notre table. Les tables obtenues sont celles-ci :

$R_1(idMetier, salaireBase, nomMetier), DF_1 = \{(7)\}$
$R_{1.1}(idType, nomType), DF_{1.1} = \{(2)\}$
$R_{2.1}(idPers, nomPers, prenom, numTel, idMetier), DF_{2.1} = \{(6)\}$
$R_{3.1}(idContenu, URL, idPers, titreCont, typeCont), DF_{3.1} = \{(5)\}$
$R_{4.1}(idJournal, idUne, nbTirage, prix, redacChef, idType, dateParu),$
$DF_{4.1} = \{(41)\}$
$R_{5.1}(idArticle, dateRedac, titreArt, resumeArticle), DF_{5.1} = \{(4)\}$
$R_{5.2}(idJournal, idUne, nbTirage, prix, redacChef, idType, dateParu),$
$DF_{5.2} = \{(1)\}$

Conclusion

Table des figures