Base de données de la rédaction d'un journal

Théo Delalande-Delarbre Thomas Minier
Benjamin Sientzoff

18 mars 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 module, 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 présenterons également les différents *triggers* créés ainsi que les fonctions stockées pour interagir avec la base.
Enfin, nous discuterons des différents atouts et modifications 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 et un numéro de téléphone. Chaque personne a un métier (photographe, rédacteur, etc.) ce qui définit un salaire de base. Un **contenu** a un titre, un type et un auteur. Dans la base de données on stockera l'emplacement des contenus sous forme d'URL. 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. La figure 2.1 présente un aperçu de notre *big table*.

idArt	titreArt	idJrnal	typeJrnal	idCont	nomCont	nomPers	nomMetier
1	Les pieuvres	1	Hebdo	1	Photo	Bertrand	Photographe
1	Les pieuvres	1	Hebdo	2	Texte	Nadine	Redacteur
2	Manger du sel	1	Hebdo	3	Schéma	Yves	Infographie
2	Manger du sel	1	Hebdo	4	Texte	Nadine	Redacteur
3	Critique film	1	Hebdo	5	Texte	Manon	Critique
4	Des poissons	3	HS	6	Texte	Nadine	Redacteur
2	Manger du sel	3	HS	1	Schéma	Yves	Infographie
2	Manger du sel	3	HS	2	Texte	Nadine	Redacteur

FIGURE 2 – Exemple de *tuples* dans la *big table* (vue partielle)

On remarque qu'il a une répétition des données relatives au journal ou au contenu : un contenu a toujours les mêmes ID, nom et auteur associé. Les journaux d'un même ID ont leur type en commun, de même pour les IDs des articles et leurs titres. Enfin, les personnes ont toujours le même métier. On peut avoir une perte d'information du fait de cette configuration car si l'on supprime par exemple tous les contenus créés par une personne, on perd les données qui lui sont associées, à savoir son nom, son prénom, son numéro, etc.

De ce fait, en prenant en compte tous les attributs nous avons dégagé un ensemble de dépendances fonctionnelles.

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 $\{dateParu, idContenu, idArticle\}$. Ces ensembles d'attributs sont des clés, car ils déterminent l'ensemble des attributs.

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 sa couverture minimale. Pour cela, on utilise la clé {idJournal, idArticle, idContenu}.

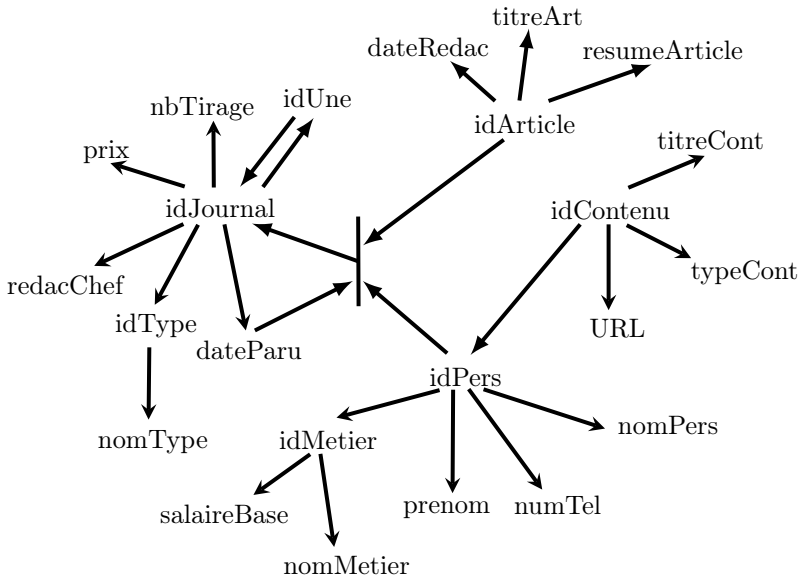

FIGURE 3 – Schéma des dépendances fonctionnelles de la base de données

Les dépendances (1), (4), (5), (6) et (7) ne sont pas élémentaires, nous avons donc dû les décomposer. Toutes les relations n'ont qu'un seul attributs à gauche, il n'est donc pas nécessaire de calculer la fermeture de ces attributs. Mais il est cependant nécessaire de le faire pour la dépendance (8) qui a plus d'un attribut en partie gauche.

Calcul de la fermeture des attributs en partie gauche de la dépendance fonctionnelle (8) :

- $idArticle+ = \{dateRedac, titreArt, resumeArticle\}$
- $idPers+ = \{nomPers, numTel, prenom, idMetier, nomMetier, salaireBase\}$
- $dateParu+ = \{\}$

Comme aucun élément des trois fermetures n'apparaît en partie gauche de la dépendance (8), cette dernière n'a pas besoin d'être modifiée.

Couverture minimale

(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 regroupons maintenant les dépendances ayant même partie gauche comme ci-dessous.

(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) $DF_8 = \{(8)\}$

On construit à présent les schémas $< R_i(U_i), DF_i >$ pour chaque 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'étant pas présente dans le schéma, on l'ajoute le schéma suivant $< R_9(idJournal, idArticle, idContenu), \{\} >$.

Après avoir appliqué de l'algorithme de Bernstein sur notre table de départ, nous obtenons neuf relations. Ce schéma est sans perte d'informations, car la dernière relation permet de relier toutes les tables entre elles grâce à la clé, sans perte de dépendances fonctionnelles, car nous n'avons perdu aucune dépendance fonctionnelle par rapport au modèle de base et est en troisième forme normale, car l'algorithme nous le garantit.

Mais qu'en est-il pour l'algorithme de décomposition ? Donne-t il le même résultat ?

2.3 Algorithme de décomposition

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

On décompose la relation avec l'algorithme de décomposition vu en cours. 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 :

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

Choix du schéma Nous venons de voir deux décompositions différentes de notre table de départ. Il faut à présent choisir l'une des deux pour l'exploitation de nos données dans un système de gestion de base de données.
Nous choisissons le résultat de l'algorithme de Bernstein car nous avons obtenu une décomposition sans perte d'information, sans perte de dépendances fonctionnelles et en troisième forme normale contrairement au second algorithme. Mais ce choix reste à approfondir en considérant quelles requêtes seront les plus fréquentes dans notre base de données. On se donne un peu plus de temps pour adopter le choix le plus judicieux.

3 Procédures stockées, vues et *triggers*

3.1 Procédures stockées

Ajout d'article On souhaite simplifier la possibilité d'ajouter un article pour

les utilisateurs : ceux-ci ne connaissent pas forcément leur identifiant dans la base de données et certains champs peuvent être remplis automatiquement, tel que celui de date de rédaction. ...

Contenu d'article La plupart du temps, un contenu ajouté à la base de données est utilisé dans un article, on a donc créé une procédure qui permet d'ajouter un contenu à un article. Cette procédure a double effet : d'une part ajouter le nouveau contenu à la table et d'autre part associer ce contenu à l'article dans la table d'association ...

3.2 Vues de la base de données

Pour faciliter la visualisation des données dans notre base, nous avons choisit de mettre en place quelques vues. Elles nous permettront d'avoir accès rapidement à certaines informations sans avoir à utiliser des requêtes complexes.

Vues liées aux métiers La table `Metiers` contient une entrée pour chaque métier existant, et est liée par une clé étrangère à la table Personnes, pour associer une personne à son métier. Il est donc intéressant d'avoir accès rapidement à toutes personnes ayant le même métier.

Pour ce faire, nous créons une vue par métier recensé dans la table `Metiers`. Chacun d'entre elle contiendra une requête sur la table `Personnes` qui sélectionnera les personnes ayant un id_metier correspondant à l'id du métier géré par la vue. Par exemple, la vue `Illustrateurs` sélectionne les personnes ayant un id_metier égal à six, car c'est l'identifiant associé au métier d'illustrateur.

Toutes ces vues sont modifiables, et cela pour plusieurs raisons. En effet, elles ne sélectionnent que des attributs NOT NULL et elles n'opèrent que sur une seule table qui n'est pas utilisé dans une sous-requête.

3.3 Traitements automatiques

4 Critique de la base données

4.1 Droits d'accès à la base de données

Comme nous travaillons à trois sur la base de données, il a fallut que les autres membres de l'équipe puissent voir et avoir accès à ce qu'a fait un autre membre. Pour cela, on a jouer avec les droits d'accès de la base de données Oracle SQL. Pour simplifier la gestion des droits, on a créé toutes les tables sur un même compte. Ensuite, donne les différents droits sur ces tables aux autres membres. Le code PL/SQL ci-dessous donner un exemple d'accord de droits sur la table `articles` des droits de sélection, mise à jour, insertion et suppression des données.

```
GRANT SELECT, INSERT, UPDATE, DELETE, REFERENCES
    ON articles TO L3_3, L3_13;
```