

# Base de données de la rédaction d'un journal

Théo Delalande-Delarbre      Thomas Minier  
Benjamin Sientzoff

18 mars 2015



# Table des matières

|   |          |
|---|----------|
| <b>Introduction</b>                                   | <b>3</b> |
| <b>1 Base du projet</b>                               | <b>4</b> |
| 1.1 Répartition des tâches . . . . .                  | 4        |
| 1.2 Définition de la <i>big table</i> . . . . .       | 4        |
| <b>2 Conception de la base de données</b>             | <b>5</b> |
| 2.1 Pourquoi décomposer la table? . . . . .           | 5        |
| 2.2 Algorithme de Bernstein . . . . .                 | 6        |
| 2.3 Algorithme de décomposition . . . . .             | 8        |
| <b>3 Procédures stockées, vues et <i>triggers</i></b> | <b>9</b> |
| 3.1 Procédures stockées . . . . .                     | 9        |
| 3.2 Vues de la base de données . . . . .              | 9        |
| 3.3 Traitements automatiques . . . . .                | 9        |
| <b>4 Critique de la base données</b>                  | <b>9</b> |
| 4.1 Droits d'accès à la base de données . . . . .     | 9        |

## 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          |
|---------|------------------------------------|----------------------|
| 4       | Big table                          | groupe complet       |
| 5       | Rédaction rapport                  | groupe complet       |
| 5       | Algo de décomposition              | Théo DD              |
| 5       | Algo de Bernstein                  | Thomas M             |
| 5       | Planning                           | Benjamin S           |
| 6       | Rendu rapport partiel              | groupe complet       |
| 6       | Implémentation SQL des tables      | groupe complet       |
| 6       | Insertion données dans BD          | groupe complet       |
| 7       | Définitions des requêtes stockées  | Théo DD              |
| 8       | Procédures PL/SQL                  | Thomas M, Benjamin S |
| 8       | Mise à jour du planning            | Benjamin S           |
| 9       | Correction du rapport              | Thomas M             |
| 10      | Correction rôle pour droit d'accès | Benjamin S           |
| 11      | Procédures PL/SQL                  | Théo DD, Thomas M    |
| 12      | Démonstration                      | groupe complet       |
| 13      | Rendu rapport final                | 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.

| 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)

## 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*.

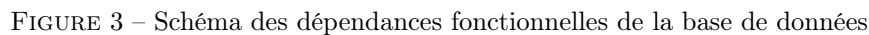
On remarque qu'il y a une répétition des données relatives au journal ou au contenu : un contenu a toujours les mêmes ID, nom et auteur associé. Les journaux d'un même ID ont leur type en commun, de même pour les IDs des articles et leurs titres. Enfin, les personnes ont toujours le même métier. On peut 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)  $\text{idJournal} \rightarrow \text{prix}, \text{nbTirage}, \text{idUne}, \text{idType}, \text{redacChef}, \text{dateParu}$
- (2)  $\text{idType} \rightarrow \text{nomType}$
- (3)  $\text{idUne} \rightarrow \text{idJournal}$
- (4)  $\text{idArticle} \rightarrow \text{dateRedac}, \text{titreArt}, \text{resumeArt}$
- (5)  $\text{idContenu} \rightarrow \text{URL}, \text{titreCont}, \text{typeCont}$
- (6)  $\text{idPers} \rightarrow \text{nomPers}, \text{prenom}, \text{numTel}, \text{idMetier}$
- (7)  $\text{idMetier} \rightarrow \text{salaireBase}, \text{nomMetier}$
- (8)  $\text{idArticle}, \text{idPers}, \text{dateParu} \rightarrow \text{idJournal}$

**Clés** De ce schéma, on peut en déduire trois clés  $\{\text{idJournal}, \text{idArticle}, \text{idContenu}\}$ ,  $\{\text{idUne}, \text{idArticle}, \text{idContenu}\}$  et  $\{\text{dateParu}, \text{idContenu}, \text{idArticle}\}$ . Ces ensembles d'attributs sont des clés, car ils déterminent l'ensemble des attributs.



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  $\{\text{idJournal}, \text{idArticle}, \text{idContenu}\}$ .

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

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

(1.1)  $\text{idJournal} \rightarrow \text{prix}$

- (1.2)  $\text{idJournal} \rightarrow \text{nbTirage}$
- (1.3)  $\text{idJournal} \rightarrow \text{idUne}$
- (1.4)  $\text{idJournal} \rightarrow \text{idType}$
- (1.5)  $\text{idJournal} \rightarrow \text{redacChef}$
- (1.6)  $\text{idJournal} \rightarrow \text{dateParu}$
- (2)  $\text{idType} \rightarrow \text{nomType}$
- (3)  $\text{idUne} \rightarrow \text{idJournal}$
- (4.1)  $\text{idArticle} \rightarrow \text{dateRedac}$
- (4.2)  $\text{idArticle} \rightarrow \text{titreArt}$
- (4.3)  $\text{idArticle} \rightarrow \text{resumeArt}$
- (5.1)  $\text{idContenu} \rightarrow \text{URL}$
- (5.2)  $\text{idContenu} \rightarrow \text{titreCont}$
- (5.3)  $\text{idContenu} \rightarrow \text{typeCont}$
- (6.1)  $\text{idPers} \rightarrow \text{nomPers}$
- (6.2)  $\text{idPers} \rightarrow \text{prenom}$
- (6.3)  $\text{idPers} \rightarrow \text{numTel}$
- (6.4)  $\text{idPers} \rightarrow \text{idMetier}$
- (7.1)  $\text{idMetier} \rightarrow \text{salaireBase}$
- (7.2)  $\text{idMetier} \rightarrow \text{nomMetier}$
- (8)  $\text{idArticle}, \text{idPers}, \text{dateParu} \rightarrow \text{idJournal}$

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  $\langle R_i(U_i), DF_i \rangle$  pour chaque  $DF_i$ , où  $U_i$  est l'ensemble des attributs apparaissant dans  $DF_i$ .

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

Pour terminer, comme la clé entière n'étant pas présente dans le schéma, on l'ajoute le schéma suivant  $\langle R_9(idJournal, idArticle, idContenu), \{\} \rangle$ .

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 choisis 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 utilisée 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 fallu que les autres membres de l'équipe puissent voir et avoir accès à ce qu'a fait un autre membre. Pour cela, on a joué 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, ce dernier attribuera ensuite des rôles précis à d'autres personnes.

Typiquement pour notre projet, Nous avons deux rôles. Le rôle **resp** et **reader**. Un responsable (**resp**) a tous les droits sur les tables du projet alors qu'un lecteur (**reader**) n'a que le droit de consulter les tables. 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 pour le rôle **resp** au utilisateurs L3\_3 et L3\_13. L'utilisateur L3\_4 n'a que le droit de consulter la table **articles**.

```
CREATE ROLE resp;  
CREATE ROLE reader;  
  
GRANT ALL ON articles TO resp;  
  
GRANT SELECT ON articles TO reader;  
  
GRANT resp TO L3_3, L3_13;  
GRANT reader TO L3_4;
```