Ministère de l'Education Nationale
Université de Montpellier II
Place Eugène Bataillon
34095 Montpellier Cedex 5

Projet GMIN103 Base de données avancée
Thésaurus

Rapport (Décembre 2011)

Travail réalisé par :

Baptiste Le Bail
Thibaut Marmin
Namrata Patel
Clément Sipieter
Steeve Tuvée

Introduction

Un thésaurus est un type de langage documentaire qui constitue un vocabulaire normalisé. Il regroupe de manière organisée les termes d'un même domaine de connaissance. Cet outil linguistique permet de décrire des concepts et de lever les ambiguïtés induites par les relations de synonymie, d'homonymie et de polysémie présentes dans le langage naturel.

L'outil développé lors de ce projet sera composé de termes décrivant des concepts, reliés entre eux par des relations hiérarchiques, synonymiques et associatives. L'utilisateur aura la possibilité d'explorer la hiérarchie et de gérer (ajouter / modifier / supprimer) les termes et les concepts.

Ce travail, réalisé par une équipe de cinq étudiants[1], est présenté dans ce rapport selon trois phases distinctes : analyse, conception, implémentation.

[1] Baptiste Le Bail, Thibaut Marmin, Namrata Patel, Clément Sipieter, Steeve Tuvée

Méthode de travail

Le projet à été réalisé en une durée d'environ un mois. Nous nous sommes réunis en moyenne deux fois par semaine pour mettre en commun nos travaux et réflexions.

Nous avons consacré la majeure partie du projet à la **phase d'analyse**. Celle-ci étant cruciale pour l'aboutissement du projet, tous les membres de l'équipe ont travaillé ensembles afin d'avoir plusieurs visions sur les différentes modélisations imaginées. Nous avons donc convergé vers une modélisation finale, que nous avons pris soin de justifier.

La **phase de conception** a engendré de nombreuses questions concernant l'utilisation du modèle objet-relationnel. Nous avons du faire le choix d'un SGBD et des choix concernant le modèle objet-relationnel. Ceux-ci étant généraux au projet, nous avons mis ce travail de réflexion en commun. Lors de cette phase, Baptiste et Steeve ont mis en place les vues utilisateurs de l'interface web. Clément et Namrata ont travaillé sur la mise en place de deux schémas relationnels : un schéma uniquement relationnel et un schéma objet-relationnel.

Les choix et réflextions de la phase de conception nous ont amené, lors de la **phase d'implémentation**, à développer l'application à l'aide d'un framework comportant un ORM (Object Relational Mapping). Cette implémentation à été réalisée par Thibaut, en incluant les templates HTML créés par Baptiste et Steeve sur les modèles mis en place à la phase précédente. Le projet ayant été réalisé à l'aide d'un ORM (donc sans rédaction de requêtes), nous avons tenu tout de même à travailler le modèle objet-relationnel. Clément et Namrata ont donc rédigé les scripts SQL de création et de consultation d'une base de donnée Oracle, utilisant des aspects objets.

La rédaction du rapport à été réalisée par Clément, Namrata et Thibaut.

Table des matières

Chapitre 1

Analyse

La phase d'analyse est un élément indispensable à la bonne réalisation du projet. Dans un premier temps, les fonctionnalités de l'application y sont décrites et caractérisées, notamment à l'aide d'un diagramme UML de cas d'utilisations. Une seconde partie présente les étapes successives qui nous ont permis d'arriver à la modélisation finale, sous la forme de diagrammes UML de classes et de « discussions ».

1.1 Fonctionnalités

L'application devra modéliser un thésaurus stocké dans une base de données, et devra permettre sa consultation au travers d'une interface web. L'utilisateur pourra donc consulter ou administrer les données. La consultation du thésaurus se fera par navigation dans une vue hiérarchique ou par nœud. Un outil de recherche facilitera l'accès aux données. Le module d'administration permettra quand à lui d'ajouter, de modifier et de supprimer les termes et les concepts, ainsi que les relations.

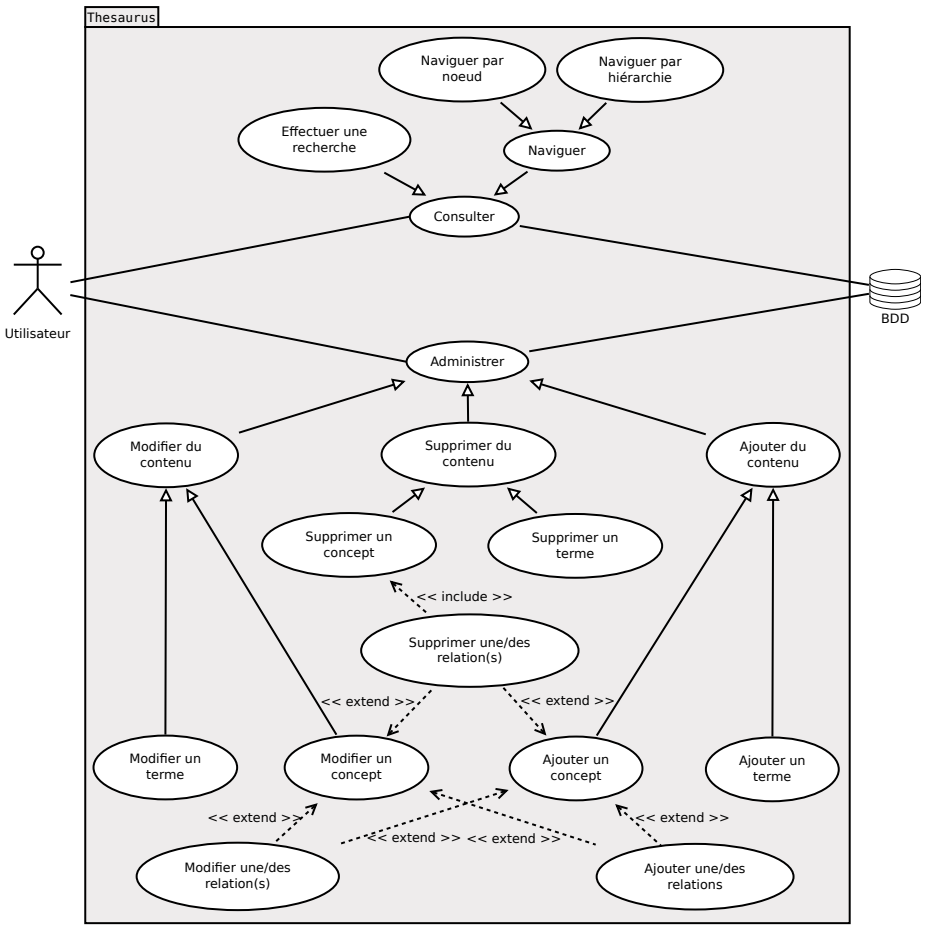

Fig. 1.1 – Diagramme de cas d'utilisations

1.2 Modélisation

Un thésaurus est composé de deux types d'entités : les Termes et les Concepts.

Concepts : ils sont décrits par un descripteur (terme vedette), et sont organisés hiérarchiquement par relations de concepts généraux (parents) et spécifiques (fils). Les concepts peuvent aussi être mis en relation par simple association.

Termes : ils sont définis par un libellé et sont en relation de synonymie entre eux.

1.2.1 Une première piste...

Pour notre première modélisation, nous avons fait le choix de respecter la définition stricte du thésaurus rappelée ci-dessus.

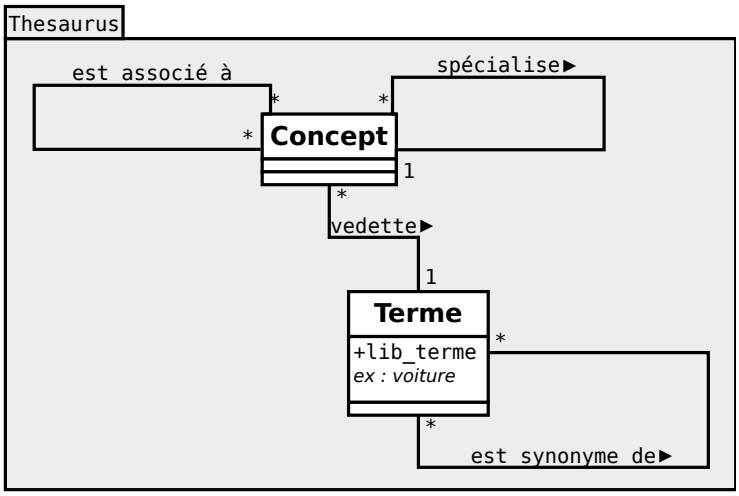

FIG. 1.2 – Première version du diagramme de classes.

Bien que la définition soit rigoureusement respectée, cette modélisation est limitée : il est impossible de lever le problème de polysémie sur les termes. En effet prenons le terme « avocat ». Celui-ci peut désigner un fruit ou un métier. La relation de synonymie est donc ambigüe bien que cette modélisation permet la création de deux concepts ayant comme le même terme vedette.

1.2.2 Évolution

Pour résoudre ce problème d'ambiguïté des synonymes, il est nécessaire de distinguer ce type de relation en ajoutant une dépendance à un concept.

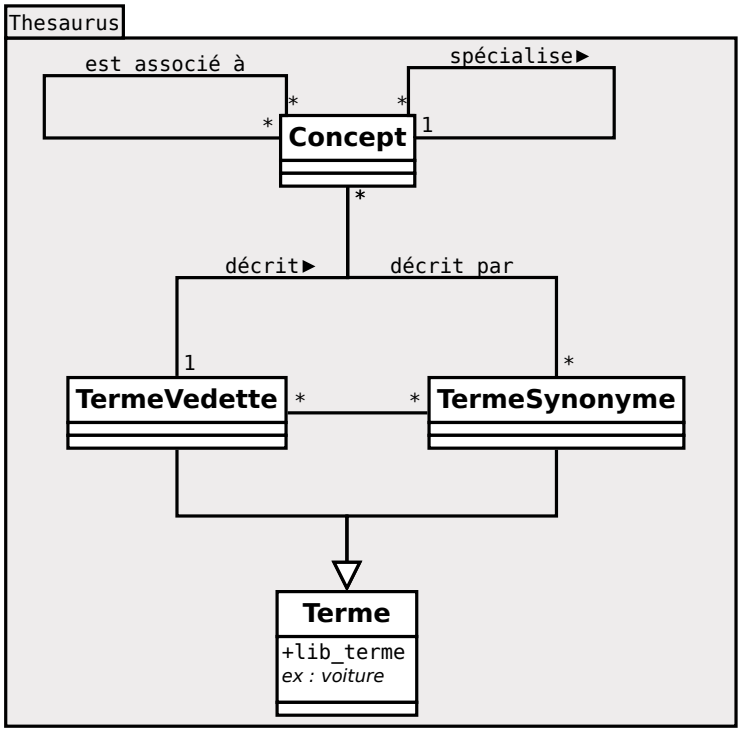

FIG. 1.3 – Évolution du diagramme de classes.

Ce modèle est légèrement éloigné de la définition d'un thésaurus car les termes ne sont plus en relation de synonymie entre eux, mais avec des concepts. Nous avons donc choisi de spécialiser la classe Terme en deux classes filles `TermeVedette` et `TermeSynonyme`. Un concept est maintenant décrit à la fois par plusieurs termes : un terme vedette de type `TermVedette` et des synonymes de type `TermeSynonyme`.

1.2.3 Décision finale

Pour cette dernière étape de réflexion, nous avons choisi de simplifier la modélisation.

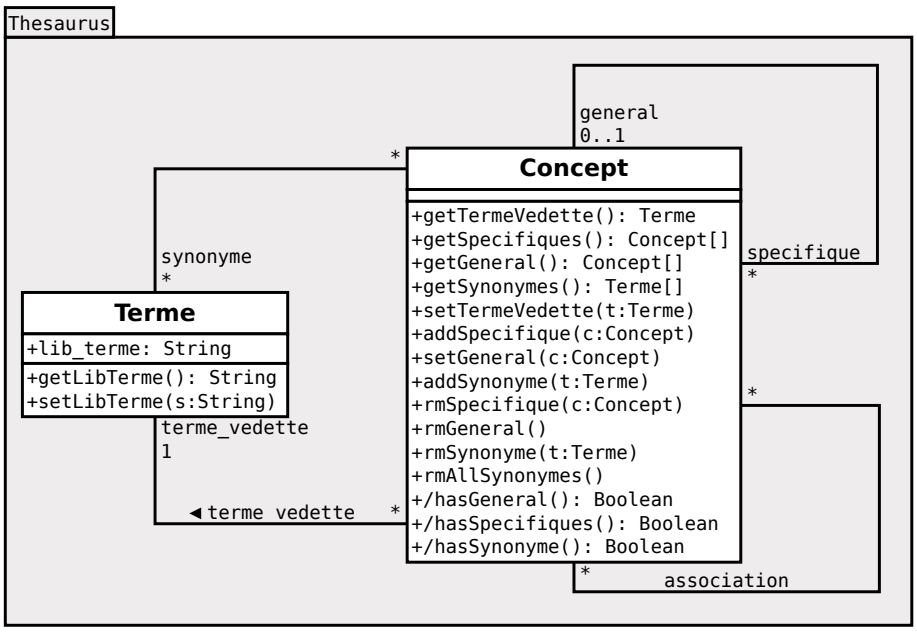

FIG. 1.4 – Version finale du diagramme de classes.

La suppression de l'héritage de terme nous rapproche du modèle initial, tout en conservant la résolution du problème lié à la polysémie. En effet, la distinction entre les termes vedette et les termes synonymes peut être ignorée.

Reprenons le problème de l'« avocat » : le terme sera stocké une fois en base, mais sera terme vedette de deux concepts : celui d'« avocat » le fruit, et celui d'« avocat » le métier. Le premier concept (fruit) pourra ne contenir aucun synonyme, tandis que le deuxième concept pourra avoir comme synonyme « défenseur ».

Chapitre 2

Conception

Schéma objet-relationnel / 100% relationnel.

Discussion sur pourquoi 100% relationnel ? ! ORM ? ! Doctrine / Symfony ?

Présentation des templates / formulaires

2.1 Objet-relationnel. . .

2.2 Pourquoi un ORM ?

2.3 Décisions

2.3.1 Templates

2.3.2 Framework

Chapitre 3

Implémentation

Classes réellement implémentées dans symfony2 + schema relationnel créé par doctrine.

Aperçus d'écran finaux

3.1 Présentation de Symfony2

3.2 Entities implémentées

3.3 Schéma relationnel généré

3.3.1 Ajout d'un déclencheur

```
1  /* Active PL/PGSQL */
2  CREATE LANGUAGE plpgsql;
3
4  /* Creation de la fonction associee au trigger de gestion de la reflexivite des associations */
5  CREATE OR REPLACE FUNCTION fc_gestion_relations() RETURNS trigger as $tgr_gestion_relations$
6  DECLARE
7  nb INTEGER;
8  BEGIN
9  IF (TG_OP = 'DELETE') THEN
10         SELECT count(*) INTO nb FROM concept_concept WHERE concept1_id = OLD.concept2_id AND
               concept2_id = OLD.concept1_id;
11         IF (nb > 0) THEN
12             DELETE FROM concept_concept WHERE (concept1_id=OLD.concept2_id AND
                   concept2_id=OLD.concept1_id);
13         END IF;
14         RETURN OLD;
15 ELSEIF (TG_OP = 'UPDATE') THEN
16         SELECT count(*) INTO nb FROM concept_concept WHERE concept1_id = OLD.concept2_id AND
               concept2_id = OLD.concept1_id;
17         IF (nb > 0) THEN
18         UPDATE concept_concept SET concept1_id = NEW.concept2_id, concept2_id = NEW.concept1_id
               WHERE concept1_id=OLD.concept2_id AND concept2_id = OLD.concept1_id;
19         END IF;
20         RETURN NEW;
21 ELSEIF (TG_OP = 'INSERT') THEN
22         SELECT count(*) INTO nb FROM concept_concept WHERE concept1_id = NEW.concept2_id AND
               concept2_id = NEW.concept1_id;
```

```
23          IF (nb = 0) THEN
24          INSERT INTO concept_concept VALUES(NEW.concept2_id,NEW.concept1_id);
25          END IF;
26          RETURN NEW;
27  END IF;
28  RETURN NULL;
29  END;
30  $tgr_gestion_relations$ LANGUAGE plpgsql;
31
32  /* Creation de la fonction associee au trigger de verification des racines */
33  CREATE OR REPLACE FUNCTION fc_verif_root() RETURNS trigger as $tgr_verif_root$
34  DECLARE
35  nb INTEGER;
36  BEGIN
37  IF (OLD.concept_general_id IS NULL) THEN
38          SELECT count(*) INTO nb FROM concept WHERE concept_general_id IS NULL;
39          IF (TG_OP = 'DELETE' AND nb < 1) THEN
40                  RAISE EXCEPTION 'Au_moins_une_racine_doit_etre_presente.';
41          ELSEIF (TG_OP = 'UPDATE' AND nb < 1 AND NEW.concept_general_id IS NOT NULL ) THEN
42                  RAISE EXCEPTION 'Au_moins_une_racine_doit_etre_presente.';
43          END IF;
44  END IF;
45  RETURN NULL;
46  END;
47  $tgr_verif_root$ LANGUAGE plpgsql;
48
49
50  /* Creation du declencheur tgr_gestion_relations */
51  CREATE TRIGGER tgr_gestion_relations AFTER DELETE OR UPDATE OR INSERT
52  ON concept_concept FOR EACH ROW
53  EXECUTE PROCEDURE fc_gestion_relations();
54
55  /* Creation du declencheur tgr_verif_root */
56  CREATE TRIGGER tgr_verif_root AFTER DELETE OR UPDATE
57  ON concept FOR EACH ROW
58  EXECUTE PROCEDURE fc_verif_root();
```

3.4 Templates finaux