Licence informatique

LOGICIEL DE GESTION DE BANDES DESSINEES

Rapport de projet

Réalisé par :

MARTIN Pierre & MONNET Fabien

Année 2010/2011

Sujet proposé et suivi par M. Bouquet Fabrice
et M. Mathias Coqblin

Sommaire

Présentation

1. Le contexte

Lorsque l'on possède une collection de bandes dessinées, il est parfois utile d'avoir la possibilité d'enregistrer informatiquement ses ouvrages afin de gérer cette collection le mieux possible.
L'idée du site BDovore.com est justement de proposer une solution gratuite aux internautes pour qu'ils puissent gérer leurs collections de bandes dessinées en ligne.
C'est un site communautaire animé par des passionnés qui ont trouvé ce nouveau média pour partager avec le plus grand nombre leur amour de la bande dessinée.

Aujourd'hui, le site BDovore.com se traduit par :

- Une base de données de 91795 volumes, tenue à jour en permanence, notée et commentée par les membres.
- La possibilité de gérer simplement sa propre collection de BD en ligne, en utilisant toute l'information disponible dans la base de données commune.
- Le suivi de ses prêts, de ses futurs achats.
- Consulter les statistiques propres à sa collection mais aussi celles de la communauté dans son ensemble.
- La possibilité de mettre en ligne sa collection de bandes dessinées.
- La possibilité de rendre votre collection publique (consultable par n'importe quel internaute).
- La possibilité de suivre au plus près les sorties et l'actualité du monde de la bande dessinée.
- La possibilité en cas d'absence d'un album dans la base de données, de le proposer au site via un formulaire.
- Un forum pour discuter de tout ce qui touche à la BD mais pas seulement…

2. Notre projet

Parallèlement à ce site, il existe plusieurs projets pour permettre de diffuser le plus largement possible le travail réalisé par la communauté. Nous avons travaillé sur un logiciel permettant à une personne de pouvoir gérer sa collection de bandes dessinées tout en étant hors ligne. Ce logiciel propose un nombre plus réduit de fonctionnalités mais garde les principaux modes d'utilisation. Il conserve l'ajout, la modification et la suppression d'un ouvrage de sa collection, la recherche d'ouvrages, la visualisation des différentes informations à propos d'un ouvrage, d'une série, d'un auteur mais également la consultation des statistiques relatives à sa collection.

En revanche, le défaut majeur du logiciel est qu'il ne permet pas de synchroniser le compte utilisateur du site BDovore.com avec le compte associé au logiciel. De même, la base de données locale concernant les ouvrages doit pouvoir être mise à jour lorsque l'utilisateur le souhaite afin d'y retrouver toutes les nouveautés. Le travail de notre projet consiste alors, à partir du code existant, à réaliser les modules permettant d'effectuer cette synchronisation à travers un Web Service.

Dans ce qui suit, nous commencerons par vous présenter un état des lieux du logiciel aujourd'hui. Dans le chapitre suivant, nous présenterons les modifications et les choix que nous avons apportés à celui-ci. Dans un premier temps, nous parlerons de l'optimisation sur les informations et l'impact sur la base de données du logiciel. Dans un second temps, nous présenterons la mise en place de la synchronisation entre le logiciel et le serveur. Nous finirons par un bilan de notre travail et aborderons la question des améliorations possibles pour le logiciel.

I. Présentation et analyse du logiciel existant

Notre travail se basant sur un logiciel existant, nous allons présenter celui-ci à travers ses différentes fonctionnalités. Ensuite, nous ferons un bilan pour indiquer les parties impactées par les évolutions associées au projet.

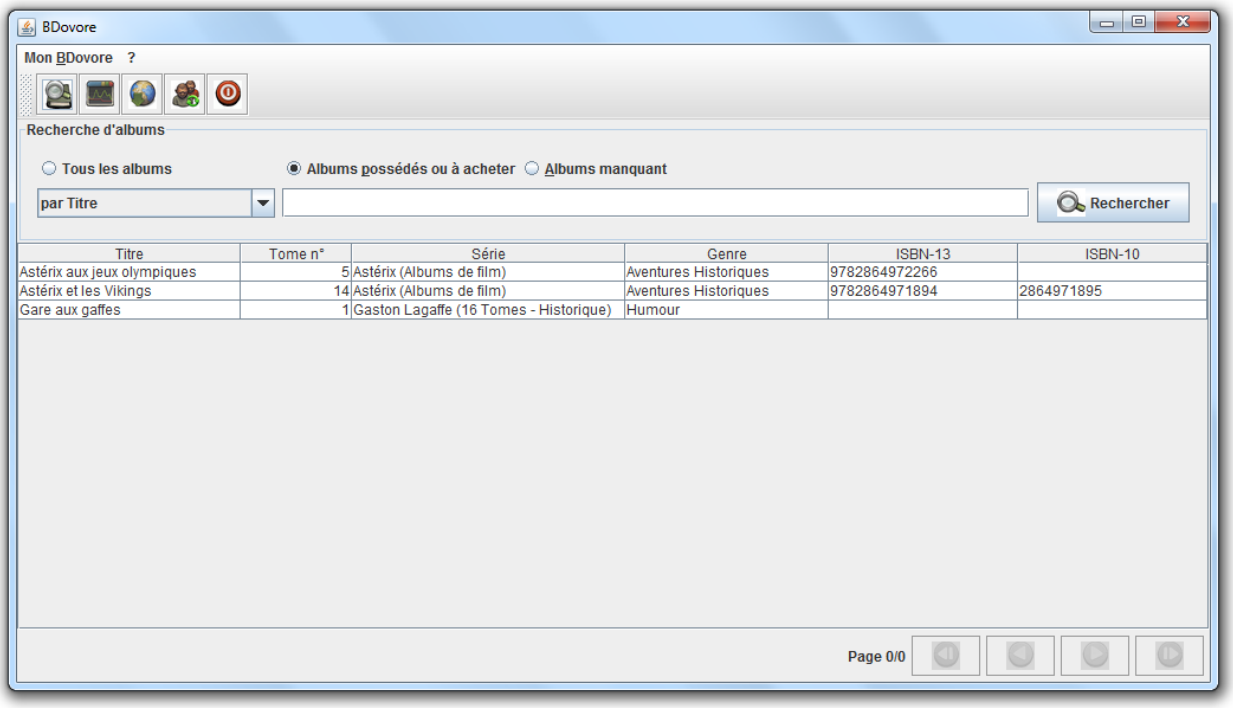
1. Modules fonctionnels du logiciel

Le logiciel existant dispose de nombreuses fonctionnalités :

1) Recherche d'ouvrages

Le logiciel permet de rechercher des ouvrages selon plusieurs critères :

- Rechercher par titre de l'ouvrage,
- Rechercher par nom de série,
- Rechercher par code barre ISBN/EAN,
- Rechercher par auteur.



Nous pouvons rechercher soit :

- Dans toute la base des ouvrages,
- Dans les ouvrages possédés par l'utilisateur,
- Dans les ouvrages manquants (c'est-à-dire, soit à acheter, soit faisant partie d'une série dont l'utilisateur possède un ou plusieurs ouvrages

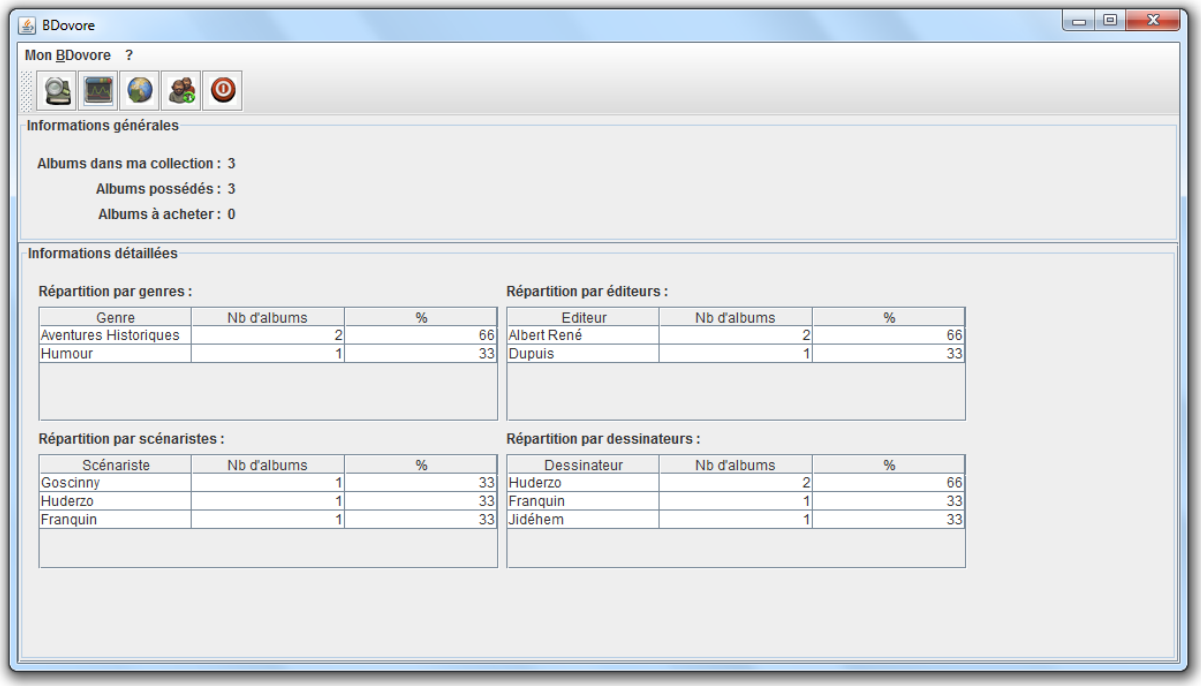
2) Visualisation d'une fiche sur un ouvrage

Lorsque l'utilisateur effectue une recherche, il peut ensuite visualiser la fiche correspondant aux ouvrages générés par cette recherche. Cette fiche fait apparaître les informations principales sur l'ouvrage comme le titre, la série associée, le numéro de tome, la liste des auteurs, la liste des éditions et les informations utilisateur (ouvrage possédé, prêté, dédicacé, à acheter ou non possédé) ainsi que l'image de couverture relative à l'édition sélectionnée.
Sur cette fiche, nous pouvons donc sélectionner l'édition que l'on veut voir dans une liste ainsi que changer son statut par rapport à l'utilisateur (Possédé, à acheter, …). Les informations seront enregistrées, si nécessaire lorsque l'on sélectionne une autre édition ou encore lors de la fermeture de la fenêtre.

3) Visualisation des statistiques de la BD thèque utilisateur

Le logiciel propose également de visualiser les statistiques sur sa BD thèque. Les statistiques proposées actuellement sont :

- Répartition par genres,
- Répartition par éditeurs,
- Répartition par scénaristes,
- Répartition par dessinateurs.



4) Importer ISBN

Cette fonctionnalité permet à l'utilisateur, grâce à un fichier texte qu'il aura préalablement rempli de codes ISBN mis les uns à la suite des autres, d'insérer dans sa collection tous les ouvrages correspondants aux différents codes ISBN présents dans ce fichier. C'est une méthode plus rapide pour l'utilisateur qui veut entrer un grand nombre d'ouvrages dans sa collection sans pour autant être obligé de rechercher chaque ouvrage puis de les ajouter.

2. Bilan

Les modules existants fonctionnent correctement et mis à part une réadaptation pour correspondre avec la nouvelle base de données que nous allons proposer, nous n'avons pas vu de point important qui nous aurait fait changer tel ou tel module.

En revanche, nous pouvons agrandir le nombre de module. Par exemple, créer une fiche pour visualiser les informations d'une série ou bien d'un auteur. Dans le code du logiciel, ces modules ont déjà été commencés à implémenter. Il nous suffira donc de les compléter pour arriver à quelque chose de cohérent.

La base de données associée au logiciel utilise le Système de Gestion de Base de Données H2. C'est un SGBD qui est relativement léger (environ 1 Mo) et le logiciel utilise JDBC pour se connecter à cette base.

Actuellement, le modèle relationnel de la base de données est calqué sur celui du site BDovore.com. Cependant, ce modèle n'est pas cohérent avec le modèle de données pour une gestion de bandes dessinées. Par exemple, dans la table tome, on retrouve des champs comme l'id. de l'éditeur ou encore la date de parution qui sont des données propres à l'édition et non pas au tome. Nous avons donc décidé de reconstruire une base plus claire afin qu'elle corresponde plus à la réalité ainsi qu'à nos besoins pour le logiciel. Cela implique une profonde modification du code mais nous justifions notre choix par le fait que le logiciel étant basé sur sa base de données, il faut faire en sorte de partir sur quelque chose de solide. De plus, les développeurs du site BDovore.com étant eux-mêmes en train de modifier leur propre base de données, il n'y a plus de raisons de garder ce modèle qui comporte de nombreuses maladresses.

II. Refonte de la base de données

Comme dit précédemment, nous avons décidé de reconcevoir le modèle relationnel de la base de données car celle-ci comporte un nombre important d'informations obsolètes mais aussi erronées. Nous allons donc vous présenter notre analyse ainsi que le travail réalisé sur cette base de façon à l'optimiser et la rendre plus claire.

1. La base de données locale existante

Le logiciel sert en premier lieu à construire sa BD thèque. C'est-à-dire, ajouter les ouvrages que l'on possède afin d'avoir une bibliothèque virtuelle représentant l'ensemble des ouvrages possédés.
Il ne s'agit donc pas de faire du logiciel une copie du site mais une version plus réduite utilisable hors-ligne tout en restant ergonomique. Partant de ce constat, il n'est alors pas utile de posséder toutes les informations sur chacun des ouvrages, chacun des auteurs ou encore chacune des série mais de réussir à ne garder que les informations intéressantes pour que l'utilisateur puisse avoir des repères et si nécessaire, aller chercher des informations supplémentaires s'il est connecté.

Nous avons voulu recréer une base avec des tables plus légères en données mais également plus lisibles. Les tables AUTEUR, SERIE, EDITION, EDITEUR, TOME et GENRE se verront donc réduites de par leurs nombres de champs et pourrons avoir une table de détails associée que l'on rempliera si l'utilisateur veut plus d'informations sur ces différentes données. Ceci pour ne pas à avoir des informations qui ne seraient pas utiles à la première utilité du logiciel qui est d'enregistrer les ouvrages possédés.

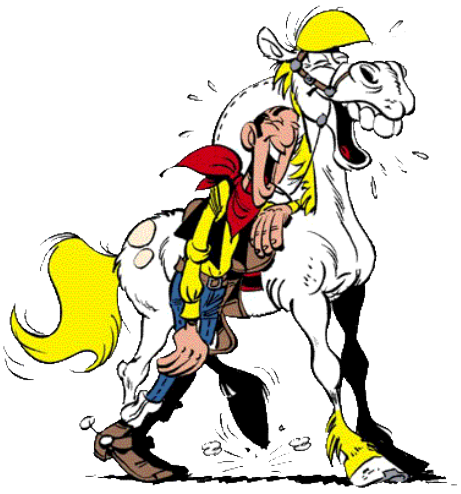
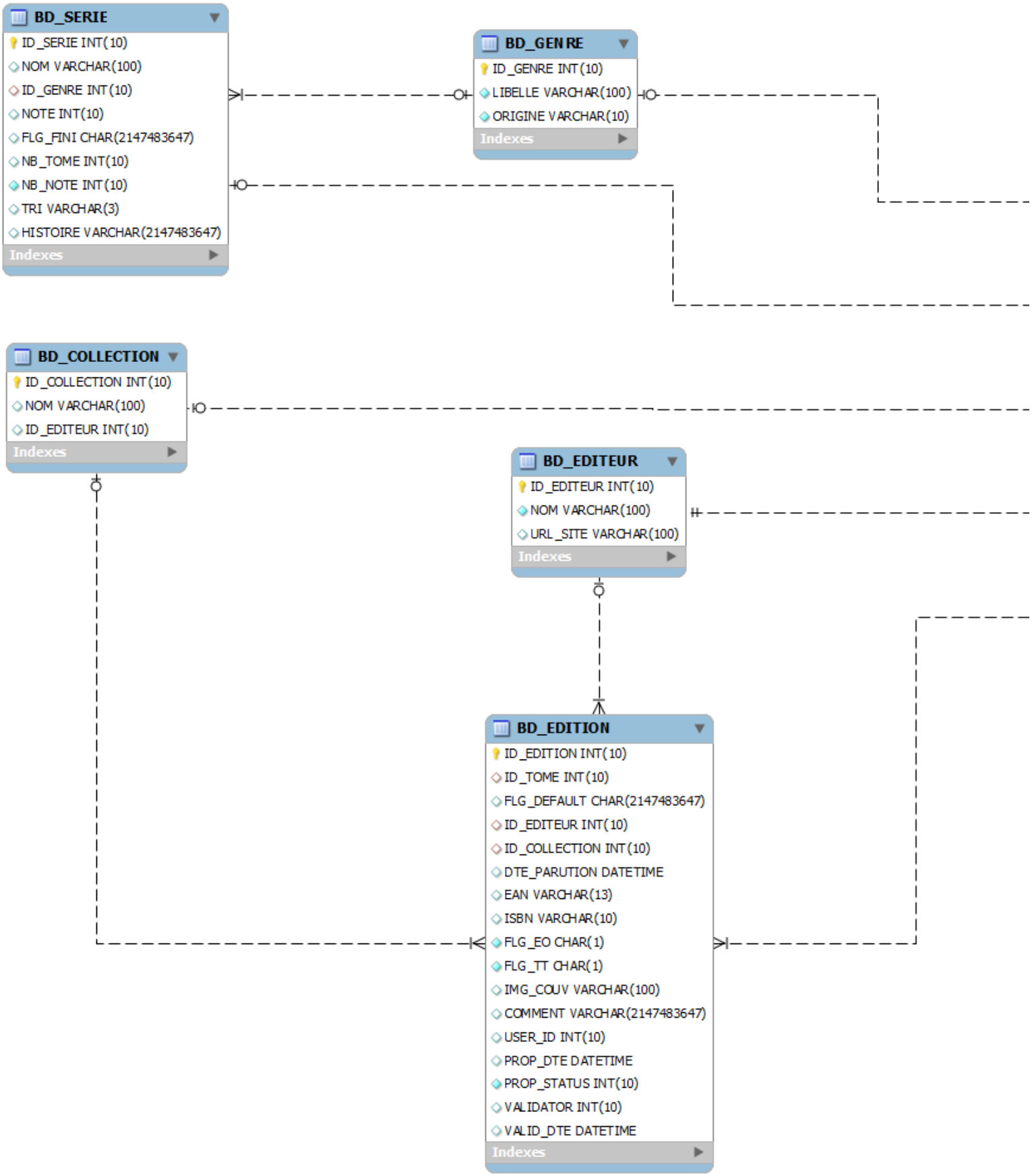
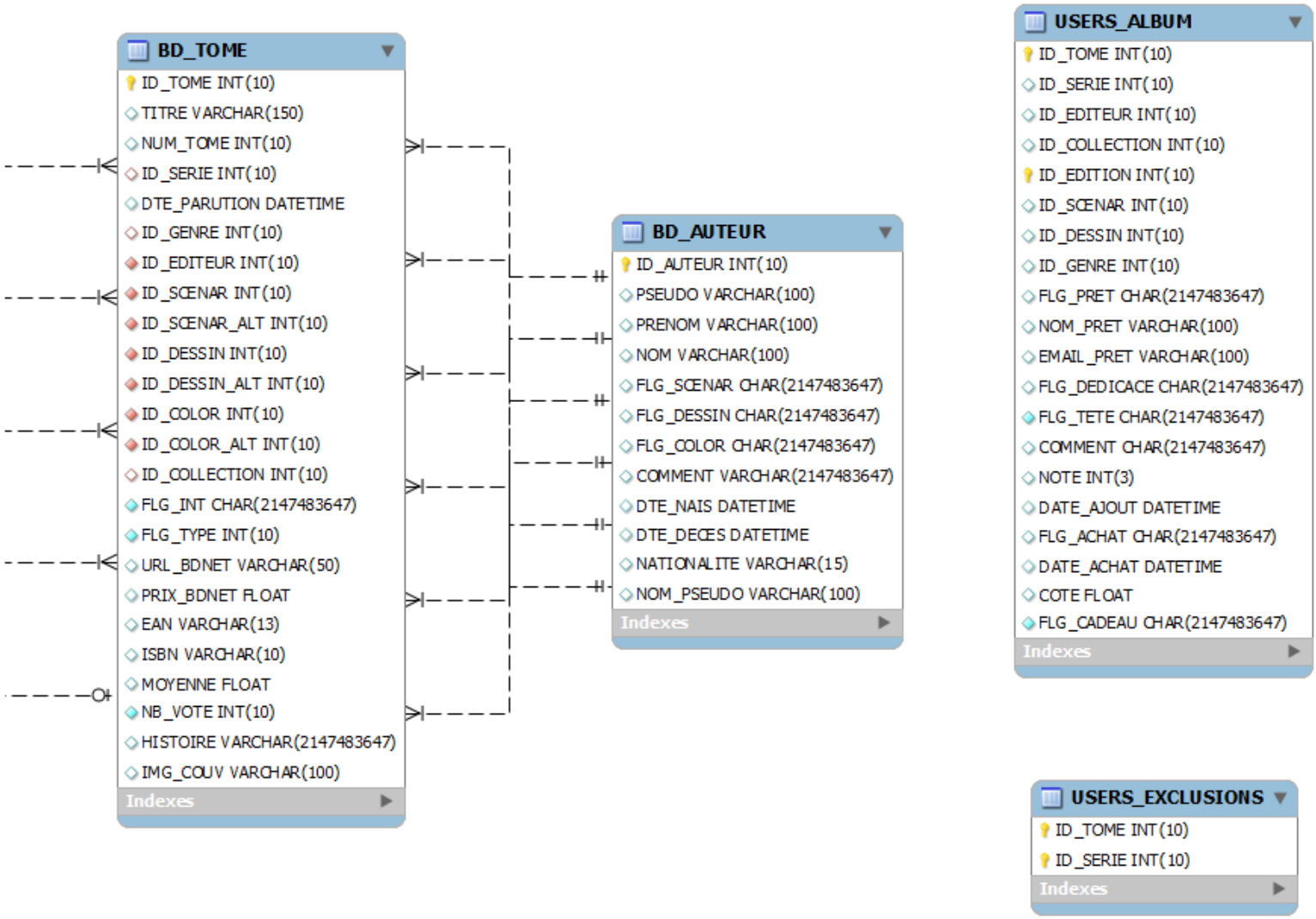


Figure 1 - Base de données existante





(Pour des raisons de lisibilité, les tables USERS_ALBUMS et USERS_EXCLUSIONS ne sont pas reliées aux autres)

2. Choix par rapport à la base de données existante

La table Genre

Nous avons décidé de garder uniquement les champs id et libelle. Le champ origine n'étant pas très utile à l'utilisateur.

La table Collection

Nous avons supprimé cette table car le fait qu'un tome fasse partie d'une collection n'est pas une information à laquelle l'utilisateur se référencera pour ajouter ses ouvrages dans sa BD thèque.

La table Editeur

Sur cette table, nous avons gardé tous les champs existants. L'id et le nom étant des éléments incontournables, l'URL est également une information supplémentaire qui peut être utilisée dans le logiciel. Cette table est indispensable à l'utilisateur pour qu'il puisse savoir quelle édition (qui en revient au nom de l'éditeur) il possède.

La table Série

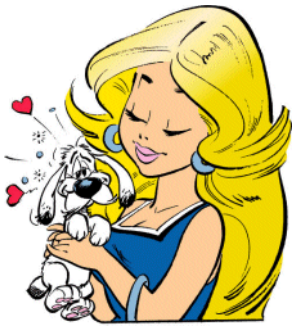
Cette table peut s'avérer utile pour donner à l'utilisateur une information supplémentaire sur les ouvrages. Par exemple, s'il possède un ouvrage d'une série donnée, il pourra ainsi voir les autres ouvrages correspondants à cette série. Au niveau des informations utiles dans cette table, l'id, le nom et le genre sont les principales données à garder. Puisque l'utilisateur peut visualiser la fiche d'une série dans le logiciel, il faut également être en mesure de lui fournir quelques informations supplémentaires comme l'histoire, le nombre de tomes dans la série ainsi que le statut de la série (en cours, finie, interrompue). C'est pourquoi nous avons créé une table supplémentaire nommée DETAILS_SERIE que l'on remplira si possible, via le web service, lorsque l'utilisateur demandera à voir la fiche de la série correspondante.

La table Users_Album

Dans la table utilisateur, apparaissent les éditions soit possédées par l'utilisateur, soit dites « à acheter » (car l'utilisateur peut faire une sélection d'ouvrage qu'il voudrait acheter. Dans ce cas, l'édition est contenue dans la table utilisateur avec l'informations qu'elle est « à acheter » et non « possédée » à proprement parler).
Cette table qui est en soit indispensable reflète bien le problème de la base de données actuelle. Il y a une grande duplication des données. Par exemple, pour une édition donnée, dans la table utilisateur, nous retrouvons : le tome associé, l'éditeur, la collection, les auteurs, la série associée et le genre. Ces informations sont inutiles dans cette table puisque déjà existantes dans les différentes autres tables. Les informations importantes dans cette table sont les informations propres à l'utilisateur : le champ FLG_DEDICACE (qui renseigne si l'ouvrage est dédicacé), le champ FLG_PRET (qui renseigne si l'utilisateur à prêté cet ouvrage), le champ FLG_ACHAT (qui renseigne si l'ouvrage est à acheter et non pas possédé) ainsi que la date d'ajout (pour avoir une trace sur l'écriture des données de l'utilisateur et permettre de faire des statistiques plus évoluées). Nous avons donc uniquement gardé ces informations ainsi que l'id de l'édition. Nous n'avons pas voulu garder d'autres données pour ne pas surcharger la table. Les champs NOM_PRET et EMAIL_PRET qui renseignent sur le nom et l'email de la personne à laquelle l'utilisateur a prêté l'ouvrage ainsi que la date d'achat peuvent s'avérer des informations utiles, cependant, nous avons préféré ne pas les sauvegarder dans un souci de garder une base assez légère en données.

La table Users_Exculsions

Le site propose aux utilisateurs de les informer des albums qui leur manquent (par exemple, si l'utilisateur possède un ou plusieurs tomes d'une série, et que la série a de nouveaux tomes, on lui propose).
On compose donc cette liste et il est proposé d'exclure des titres, c'est-à-dire de décider que dans cette liste, un ne voudrait pas de certains tomes. La table sert donc à enregistrer les exclusions. On génère ensuite la liste des albums manquant d'une série en prenant la liste des tomes, et en éliminant ceux possédés et ceux exclus.
Dans le logiciel, rien n'implémentait une fonction qui permettait de faire ces opérations. Ceci n'étant pas un des objectifs principaux du logiciel, nous avons plutôt décidé de ne pas garder cette table.



La table Edition

Cette table est l'une des tables les plus importantes de la base de données car c'est la version concrète du tome (le tome étant une idée plus abstraite). Lorsque l'utilisateur dit posséder un ouvrage, c'est d'abord une édition qui est enregistrée. Dans cette table, beaucoup d'informations sont également de trop. Nous avons choisi de garder uniquement les informations utiles au logiciel comme l'id de l'édition, l'id du tome correspondant, l'id de l'éditeur, la date de parution, le flag par défaut (pour des soucis d'affichage des résultats de recherche) ainsi que l'ISBN. Sur ce dernier point, lorsque l'on récupérera l'ISBN (au moment de la mise à jour de la base de données locales via le web service), nous chercherons à récupérer en premier l'ISBN10 s'il existe, puis sinon, l'ISBN13. L'ISBN10 étant facilement transformable en ISBN13, il est plus pertinent de le récupérer en premier lieu puis de faire la conversion si nécessaire. De plus, les ouvrages n'ont pas tous un ISBN13, celui-ci étant en vigueur pour les nouveaux ouvrages depuis 2007.
Nous avons choisi de créer une table DETAILS_EDITION afin d'enregistrer si besoin le nom de l'image de couverture de l'édition. En effet, nous n'allons pas enregistrer chacun des noms des images de couverture. Ce n'est pas une information de premier plan. L'utilisateur se référera au nom de l'éditeur et à la date de parution pour sélectionner l'édition qu'il possède. L'image étant là pour l'affichage et donc plus pour un souci d'ergonomie. Nous téléchargerons donc l'image si l'utilisateur demande la fiche d'un ouvrage.

La table Auteur

Dans cette table, nous avons également décidé de garder uniquement les informations naturelles. C'est-à-dire, l'id de l'auteur, le nom, le prénom ainsi que le pseudo de l'auteur. Lorsque l'utilisateur demande à visualiser une fiche sur un auteur, on télécharge via le web service (si possible) les informations supplémentaires comme la date de naissance, la date de décès (si elle existe) et la nationalité dans une table nommée DETAILS_AUTEUR.

La table Tome

Dans la table Tome, il y a des informations redondantes. On y retrouve l'id du genre déjà présent dans la table Série. Nous ne l'avons donc pas gardé. Il y a également des informations erronées que l'on ne devrait pas retrouver dans cette table : la date de parution, l'ISBN et l'EAN mais également le nom de l'image de couverture. Pour cette version de la base de données, il est possible d'enregistrer uniquement 2 scénaristes, 2 dessinateurs ainsi que 2 coloristes (marqués par leur id qui fait référence à la table Auteur). Dans notre nouvelle version, nous avons décidé de créer un table nommée TJ_TOME_AUTEUR définie par trois champs, l'id du tome, l'id de l'auteur et le rôle (scénariste, dessinateur ou coloriste) qui font tous les trois partie de la clé primaire de cette table. Ce sera cette table qui fera office de détails du tome. Elle correspond en fait, pour un

id de tome donné, aux différents auteurs de ce tome ainsi que le rôle de chacun. Cette méthode permet de ne pas borner le nombre d'auteur pour un tome.

Une nouvelle table, la table transaction

Pour faciliter le mécanisme de synchronisation avec le compte utilisateur du site internet, nous avons choisi de créer une table supplémentaire nommée Transaction. Cette table enregistrera les modifications apportées à la table utilisateur. Son fonctionnement en décrit dans la partie suivante.
Cette table se compose de trois champs, l'id de l'édition, le type de modification effectuée (ajout, modification, suppression) ainsi que la date de modification.

3. Adaptation du code avec la nouvelle base de données

Le travail sur la réadaptation du code afin qu'il corresponde à la nouvelle base de données a été un travail relativement long. Nous avons dû changer toutes les requêtes de recherche, de récupérations d'informations sur les différents éléments de la base ainsi que les requêtes de statistiques. Puis il a fallu également modifier les traitements lors de la réception des données de la base. Nous avons aussi implémenté la méthode permettant de tenir à jour la table des transactions. Cette méthode sera décrite dans la partie suivante.

Dans notre nouvelle version, la recherche par auteur à été supprimée car devenue trop limitée. En effet, les auteurs ne sont pas d'emblée associés aux ouvrages, une recherche par auteur ne nous retournerait alors qu'une réponse partielle.

Sur la fiche d'un album, dans la version précédente, nous pouvions afficher uniquement 2 artistes pour chaque rôle (Dessinateur, scénariste ou coloriste). Dans notre version, nous avons effectué des changements afin qu'il soit possible, si nécessaire, d'en afficher le nombre que l'on souhaite. Ceci va de pair avec le fait que notre nouvelle base ne cantonne pas le nombre d'artiste à 2 par tome selon leur rôle.

Nous avons également réalisé les fiches de visualisation d'un artiste et d'une série qui étaient partiellement faites sur la version précédente. Ces fiches sont accessibles depuis la fiche de visualisation d'un album. L'on peut maintenant cliquer sur le nom de la série et sur le nom des différents auteurs pour voir les détails sur la fiche correspondante.

Explications sur le fonctionnement du logiciel en mode connecté

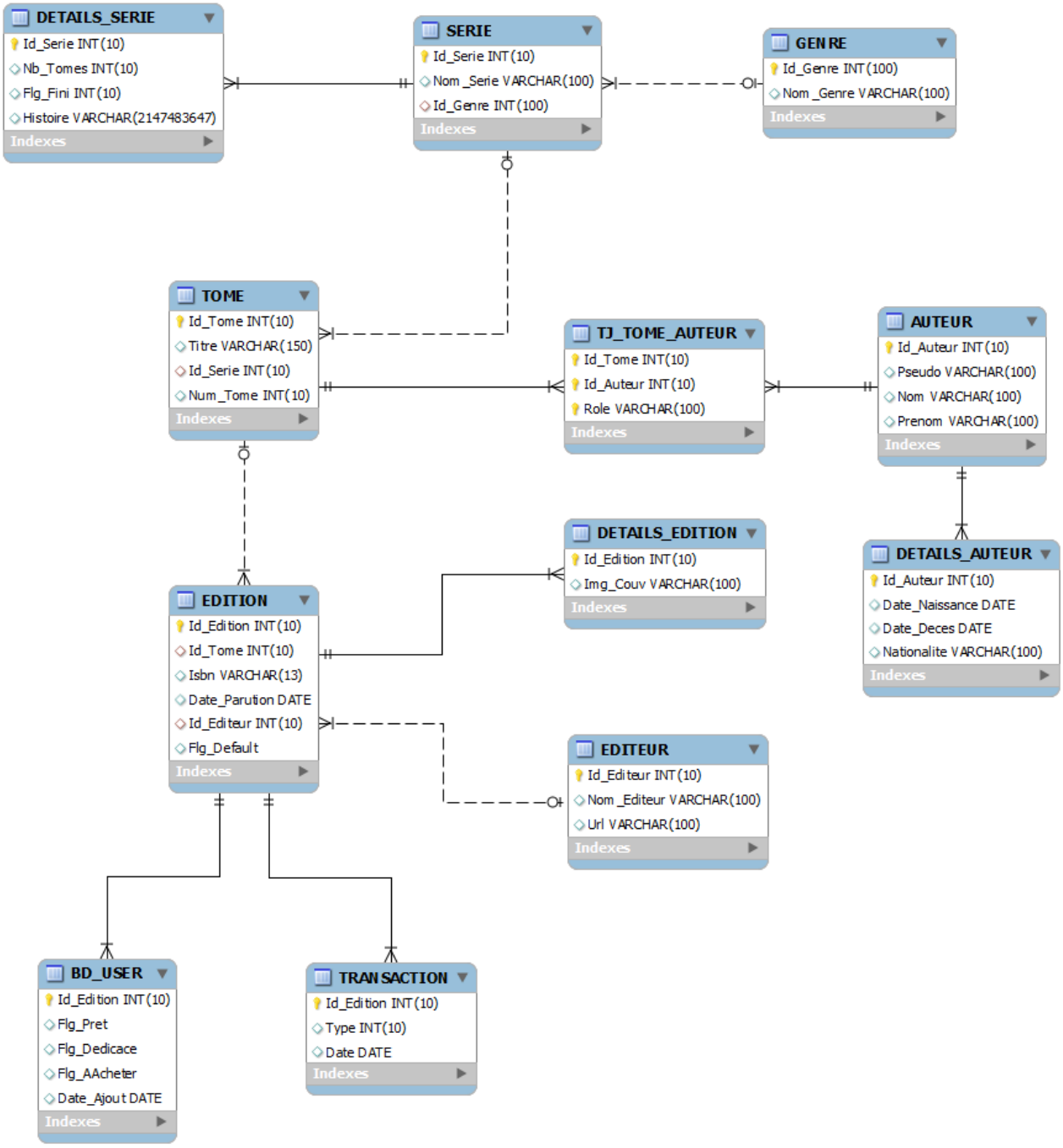
Dans notre version du logiciel, le principe sera de récupérer les informations complémentaires nécessaires à l'affichage des différentes fiches si le logiciel est en mode connecté (c'est-à-dire qu'il peut se connecter au web service ainsi qu'à internet pour l'image):

- Si l'utilisateur demande à voir la fiche d'une série, on télécharge les informations complémentaires à la série pour lui afficher (nombres de tomes, statut, histoire).
- Si l'utilisateur demande à voir la fiche d'un auteur, on télécharge les informations complémentaires à l'auteur pour lui afficher (date de naissance, date de décès, nationalité).
- Si l'utilisateur demande à voir la fiche d'un album, on télécharge les informations complémentaires à l'album pour lui afficher (les auteurs, l'image de l'édition sélectionnée).

De même, si l'utilisateur dit posséder une édition, on se doit de télécharger les informations sur les auteurs du tome correspondant (leur rôle) ainsi que l'image de l'édition.

Toutes les informations collectées sont insérées dans la base de données. Si, à un certain moment, l'utilisateur trouve que sa base de données prend trop de place, nous avons créé une fonction de purge afin que lorsque cette fonction est appelée, cela supprime de la base toutes les informations qui ne sont pas relatives à une édition possédée. En revanche, cette fonction ne permet pour l'instant pas de supprimer les photos téléchargées.

Figure 2 - Nouvelle base de données



III. Mise à jour et synchronisation

Grâce au logiciel, l'utilisateur peut donc ajouter, supprimer ou effectuer des modifications sur chacun des ouvrages qu'il possède sans être obligatoirement connecté à Internet. Cependant, si le logiciel ne veut pas être dépassé, le logiciel doit pouvoir se mettre à jour afin de proposer les nouveaux ouvrages qui ont pu être ajoutés sur le site BDovore.com. Lorsque l'utilisateur modifie sa collection, les modifications sont enregistrées en local, sur son disque dur. En parallèle, il est toujours possible d'utiliser le site internet pour effectuer ces modifications, le but de la synchronisation est donc de répercuter les modifications effectuées en local sur la base de données du site et inversement, du site vers le local.
On considère qu'une fois la synchronisation effectuée, les données du compte utilisateur sont exactement les mêmes en local que sur le site, chaque synchronisation se base sur ce qui a été modifié depuis la dernière synchronisation.

Néanmoins, nous n'avons pas accès directement à la base de données du site, le logiciel communique donc avec un Web Service, afin de télécharger les dernières modifications effectuées sur le compte utilisateur et de récupérer les dernières mises à jour effectuées sur la base de données du site.

1. La communication avec le Web Service

Le Web Service avec lequel nous communiquons est un script. Il n'est pas implémenté pour l'instant. Nous avons donc développé notre synchronisation de manière indépendante, en normalisant les fonctions de communication grâce au langage WSDL afin de donner des indications sur les méthodes que devra implémenter le web-service lorsqu'il sera développé.

Définition de WSDL (inseadima.com) : « *le WSDL (Web Services Description Language) est un langage de description de Web Services, au format XML. Il permet de séparer la description des fonctionnalités abstraites offertes par un service, des détails concrets d'une description de service, tels que "comment" et "où" cette fonctionnalité est proposée. C'est donc un langage décrivant les fonctionnalités abstraites d'un service ainsi que l'architecture décrivant les détails concrets de la description de service.*
En clair, il définit, de manière abstraite et indépendante du langage, l'ensemble des opérations et des messages qui peuvent être transmis vers et depuis un service web donné. »

Le WSDL décrit quatre ensembles de données importants :

- information d'interface décrivant toutes les fonctions disponibles publiquement,
- information de type de données pour toutes les requêtes de message et requêtes de réponse,
- information de liaison sur le protocole de transport utilisé,
- information d'adresse pour localiser le service spécifié.

Dans notre cas, le format XML des requêtes sert uniquement à encapsuler les données. Les informations sont contenues dans une chaîne de caractère qui reprend le format csv (une ligne par édition, les champs sont séparés par des virgules).

2. La mise à jour de la base de données

La base de données du site est en constante évolution. En effet le site offre à ses utilisateurs la possibilité de proposer de nouveaux ouvrages, ce qui implique de nouvelles séries, de nouveaux auteurs, pourquoi pas de nouveaux genres.
Lors de la toute première connexion du logiciel au Web Service, le logiciel va récupérer l'ensemble des données présentes dans la base de données du site et remplir, dans un ordre bien précis (afin de respecter les contraintes d'intégrité de la base), les tables associées dans la base locale. Puis, lors de chaque nouvelle synchronisation ou sur demande de l'utilisateur, le logiciel téléchargera uniquement les informations manquantes (nouvelles données depuis la dernière synchronisation).

Nous avons choisi de ne pas tenir compte des éventuelles modifications sur la base de site car pour réaliser ceci, il aurait fallu comparer toutes les entrées de la base locale avec la base distante, ce qui implique un très grand nombre de calcul et de grand téléchargement lors de chaque synchronisation. De plus, les membres de l'équipe du site portant une grande attention à la qualité des informations qu'ils enregistrent dans leur base de données, le nombre de modifications apportée est donc négligeable.

La récupération de toutes ces informations étant assez longues au vu de la taille imposante de la base de données, nous demandons au Web Service uniquement les derniers ajouts de chaque par rapport à la base de données locale. Ce qui revient à lui envoyer le dernier id de chaque table afin qu'il nous renvoie les données manquantes. Le résultat renvoyé par le Web Service est une chaîne de caractère sous la forme d'un fichier csv, à chaque ligne correspond une entrée, chaque champ est séparé par une virgule, l'ordre des champs étant défini à l'avance dans le fichier WSDL.

Fonction récupérant le dernier id d'une table :

```
public synchronized int getLastID(String table) throws SQLException {
        int id = 0;

        if (!Tables.ids.containsKey(table)) return 0;
        if (!table.startsWith("BD_")) return 0;

        String sql = "SELECT MAX(" + Tables.ids.get(table) + ") AS id FROM " + table;
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql);
        if (rs.next()) {
                id = (rs.getObject(1) == null) ? 0 : (Integer) rs.getObject(1);
        }
        st.close();
```

```
        return id;
}
```

Fonction définie dans le WSDL envoyé au web-service :

```
<message name="getNewEntries">
        <part name="userId" type="xsd:string"/>
        <part name="userPass" type="xsd:string"/>
        <part name="nomTable" type="xsd:string"/>
        <part name="lastId" type="xsd:integer" />
</message>
<message name="getNewEntriesResponse">
        <part name="listNewEntries" type="xsd:string"/>
</message>
```

3. Synchronisation du compte de l'utilisateur

L'utilisateur pouvant effectuer des changements sur sa collection tant sur le site que le logiciel, il faut que les données de chaque coté soit comparées afin que les comptes de chaque coté possède les mêmes éditions avec les mêmes options.
Chaque synchronisation se basant sur les changements effectués par l'utilisateur depuis sa dernière synchronisation, il faut garder une trace de chaque changement effectué en local. En effet, la table utilisateur ne nous dit pas tout. Par exemple, si l'utilisateur passe un ouvrage de « possédé » à non « possédé », sur le logiciel, il n'y aura aucune trace puisque l'ouvrage à été supprimé de la table utilisateur. Lorsque l'on voudra effectuer la synchronisation, on ne saura pas si l'utilisateur l'a ajouté sur le site ou alors supprimé sur le logiciel. C'est pourquoi il faut donc garder une trace ''intelligente'' des modifications effectuées par l'utilisateur sur le logiciel (à défaut de pouvoir le faire également sur le site).

Principe des transactions – Fonction de la table de transactions

Nous avons donc utilisé comme méthode ce que nous avons appelé des ''transactions''. Une transaction est en fait un triplet (entier, entier, date) qui est renseigné dès lors qu'une modification est effectuée par l'utilisateur sur le logiciel. Ce triplet sera alors inséré dans la table de notre nouvelle base de données intitulée « transaction ».

Le triplet d'une transaction se compose comme suit :

- Le premier entier correspond à l'id de l'édition pour laquelle il y a eu des changements de statut.
- Le second entier correspond au type de la transaction, il existe trois types de transaction qui correspondent aux trois types de modifications possibles : 1 pour Ajout, 2 pour Modification, 3 pour Suppression.
- La date correspond à la date à laquelle le statut de cet ouvrage a été changé.

Pour chaque édition, au regard de la table transaction, il nous est possible de savoir quelles modifications ont été effectuées, et comment les interpréter lors de la suivante synchronisation.

Dès que l'utilisateur effectuera un changement sur une édition qu'il possède ou ne possède pas une transaction sera créée. Si l'utilisateur effectue plusieurs changements pour une même édition, on n'ajoute pas une nouvelle transaction mais on modifie la transaction existante. Ainsi il n'y aura jamais plus d'une transaction pour une édition donnée. Cela permettra un traitement plus facile à mettre en œuvre et plus efficace lors de la synchronisation.

La première chose que nous avons eu à penser a été de lister tous les cas possibles d'enchainement de transaction afin de prévoir chaque manipulation de l'utilisateur. Par exemple si un ouvrage est ajouté, on ajoute une transaction de type « ajout », puis s'il est modifié par la suite, la transaction reste de type « ajout ».

Voici un tableau récapitulatif de tous les enchaînements possibles de transaction et les modifications à réaliser sur la table transaction :

Type de transaction existante (pour un ouvrage donnée)	**Changement effectué sur l'ouvrage**	**Modification de la transaction**
Ajout	Ajout	Cas impossible
	Modification	Ajout
	Suppression	On supprime la transaction existante
Modification	Ajout	Cas impossible
	Modification	Modification
	Suppression	Pas de transaction (On supprime la transaction existante)
Suppression	Ajout	Cas impossible
	Modification	Cas impossible
	Suppression	Cas impossible
Transaction inexistante	Ajout	Pas de transformation, on insère une transaction de type Ajout
	Modification	Pas de transformation, on insère une transaction de type Modification
	Suppression	Pas de transformation, on insère une transaction de type Suppression

Partie du code de cette méthode de mise à jour de la table transaction :

```
public void setTransaction(int idEdition, int typeModif) throws SQLException {
        String sql = "SELECT TYPE " +
        "FROM TRANSACTION T " +
        "WHERE T.ID_EDITION=" + idEdition;

        int typeModifOld = getTypeTransaction(sql);
        sql ="";
        switch(typeModifOld){

                case Edition.INSERT : // Si pour l'edition donnee, il ya une transaction de type
        insert
                        switch (typeModif){
                                case Edition.INSERT:
                                        break;
                                case Edition.UPDATE:
                                        sql = "UPDATE TRANSACTION " +
                                        "SET TYPE = 1, DATE = NOW() " +
                                        "WHERE ID_EDITION = "+ idEdition;
                                        break;
                                case Edition.DELETE:
                                        sql = "DELETE FROM TRANSACTION " +
                                        "WHERE ID_EDITION = " + idEdition;
                                        break ;
        […]
}
```

Cette méthode est appelée dès lors que l'utilisateur effectue une modification sur une édition, on l'appelle en lui passant en paramètre l'id de l'édition à modifier ainsi que le nouveau type de la transaction.
La fonction regarde dans la table transaction s'il existe une entrée pour l'édition donnée, s'il n'en existe pas, on insert une entrée avec l'id de l'édition, la valeur de la transaction ainsi que la date courante et s'il existe une entrée pour cet édition, on met à jour cette entrée dans la table conformément au tableau vu précédemment.

Protocole de synchronisation

Du coté du site internet, nous n'avons pas la possibilité de savoir les différentes actions que l'utilisateur a effectuées sur sa collection.
C'est pourquoi, lors de la synchronisation nous parcourons la liste de toutes les éditions possédées par l'utilisateur en local, que nous comparons à la liste des éditions possédées sur le compte utilisateur du site (retourné par le Web Service sous format csv).
Nous listons donc tous les cas possibles en fonction de la présence ou non d'une édition sur le logiciel et sur le site (sur le compte utilisateur associé) ainsi que la présence ou non de transaction pour cette édition.

Le tableau suivant résume tous les cas possibles ainsi que la tâche à accomplir pour chaque cas :

Local	Site	Type de transaction	Action à effectuer
Présent	Présent	Ajout	On vérifie si les données sont les même
		Modification	On vérifie si les données sont les même
		Suppression	**X**
		Pas de transaction	On vérifie si les données sont les même
	Absent	Ajout	On ajoute l'édition sur le site
		Modification	Si édition modifiée en local et supprimée sur le site, conflit : On demande l'avis de l'utilisateur
		Suppression	**X**
		Pas de transaction	On supprime l'édition en local
Absent	Présent	Ajout	**X**
		Modification	**X**
		Suppression	On supprime l'édition sur le site
		Pas de transaction	On ajoute l'édition en local
	Absent	Ajout	**X**
		Modification	**X**
		Suppression	Rien à faire
		Pas de transaction	Rien à faire

Pour l'implémentation de ce tableau, on effectue un parcours de toutes les éditions contenues dans la table BD_USER joint avec la table transaction. Voici la requête :

```
SELECT *
FROM BD_USER LEFT OUTER JOIN TRANSACTION
ON BD_USER.ID_EDITION=TRANSACTION.ID_EDITION
```

L'utilisation de la jointure externe à gauche permet de récupérer les données de la table BD_USER même lorsqu'aucune transaction n'existe pour cette édition.

Cette requête donne des résultats sous la forme :

IdEdition	FLG_Pret	FLG_Dedicace	FLG_AAcheter	Date	TypeTransaction	DateTransaction
45	0	0	0	10/08/09	null	null
46	0	0	1	10/08/09	null	null
47	0	1	1	10/09/10	1	10/09/10
49	1	0	0	10/11/09	2	10/09/10
54	1	0	0	10/05/09	null	null

Nous comparons chacune des informations contenue dans la table utilisateur du logiciel, ce qui fait trois comparaisons par édition, pour les flags de prêt, de dédicace et de « à acheter ».

Pour chaque tuples résultant de cette requête, on regarde dans le fichier csv envoyé par le Web Service représentant l'ensemble des éditions contenues dans le compte utilisateur sur le site s'il existe une édition ayant le même id.
Pour chaque ligne ayant un équivalent dans le fichier csv, on effectue le traitement approprié puis on supprime la ligne correspondante dans le fichier csv.
Ainsi, une fois que l'on aura parcouru toutes les éditions contenues en local, il ne restera dans le fichier csv que des éditions n'étant pas présentes en local. Il suffira alors de parcourir le reste du fichier csv, et d'ajouter les ouvrages dans la table utilisateur du logiciel.

A la fin de la synchronisation, on supprime toutes les données de la table de transaction afin d'avoir le même raisonnement pour la prochaine synchronisation.

Conclusion

Point sur le projet au jour d'aujourd'hui

La première chose a été la phase d'analyse du logiciel existant, compréhension de la structure du projet, gestion de l'interface graphique, connexion à la base de données. Cette analyse a abouti à la refonte de la base de données, dans laquelle il fallait réussir à ne conserver que les informations réellement utiles au fonctionnement du logiciel tout en construisant une base plus propre et plus claire.

Une fois le nouveau modèle de la base réalisé, il a fallu adapter le code, modifier les différentes requêtes de recherche, de statistique et de récupération d'informations. Puis également adapter les fonctions se référant à la base utilisateur du logiciel.

Le code modifié et de nouveau fonctionnel avec notre base de données, nous nous sommes attaqués au protocole de communication avec le Web Service ainsi que le traitement des données reçues. Nous avons mis en place la table transaction, fonctionnant de pair avec la table bd_user, puis listé tous les cas d'enchaînement de transactions possibles, planifié les différentes actions à effectuer selon la présence ou non d'un ouvrage dans les deux bases locales et distantes et la valeur de la transaction associée.

Ensuite nous avons normalisé l'implémentation des fonctions communicantes avec le Web Service à l'aide du langage WSDL, et nous avons commencé à implémenter les fonctions effectuant les traitements expliqués dans la deuxième partie conformément à notre description dans le fichier WSDL.

La dernière chose faite a été de créer des fiches auteur et série sur le même principe que la fiche de visualisation d'un ouvrage et de les intégrer aux fonctions assurant l'affichage.

Perspectives

La fonction de recherche a été modifiée suite à la refonte de la base de données, elle ne fonctionne plus de la même manière qu'auparavant, c'est une simple recherche sur la valeur exacte d'une chaine de caractère, la fonction ne gère plus la recherche en ''full text'' car nous n'avons pas réussi à réutiliser le moteur de recherche lucene.

De plus nous n'avons pas réalisé réellement la connexion avec le Web Service utilisant le protocole SOAP, mais il sera facile de le faire par la suite grâce au fichier WSDL ainsi qu'a l'interface qu'il suffit d'implémenter pour que le logiciel puisse communiquer avec le web service.

Nous voulions également gérer la possibilité pour l'utilisateur de proposer de nouveaux ouvrages par le biais d'un formulaire de soumission comme cela ce fait actuellement sur le site.

Bilan personnel

Ce projet nous a tout d'abord permis d'apprendre comment gérer un projet à plusieurs. Il a fallut que l'on définisse clairement les tâches de chacun. De plus, nous avons découvert durant ce projet l'outil SVN pour travailler en équipe sur les mêmes fichiers.

Au niveau du développement, nous avons vu comment les fichiers sources d'un projet de cette ampleur étaient organisés (interface graphique séparée du traitement des données). Nous avons découvert la base de données H2 ainsi que les moyens possible de se connecter à cette base. Dans ce projet, nous avons également vu comment utiliser le format csv puis comment se connecter à un web service à partir d'une application réalisée en JAVA.

Pour nous, cela a été un grand apprentissage ou l'on a dût reprendre un code existant et le comprendre (ce qui n'a pas été une tache facile et ce qui a pris du temps), ensuite, faire face à un certains nombre de choix (comme par exemple refonder la base de données existante), puis insérer les nouveaux modules développés dans le code. Nous avons là bien compris l'utilité d'une bonne analyse avant de se lancer dans le développement.

En résumé, ce projet était un projet ambitieux qui nous a beaucoup appris tant au niveau de la gestion du projet que de la partie de développement. Nos difficultés ont été de comprendre l'existant ainsi que de voir l'impact sur le logiciel de notre projet. C'est un projet qui peut encore évoluer car il y a beaucoup de fonctionnalités que l'on pourrait ajouter afin de l'améliorer encore et encore.

Nous tenons à remercier nos tuteurs pour le temps qu'ils nous ont accordé et leur aide précieuse tout au long de ce projet.