Référent(s) du document	Relecteur(s) obligatoire(s)	Chargé(s) validation
Ekaterina Flin	Eric Maldonado	Eric Maldonado
Dispositif : CLPA Domaine fonctionnel : Préparer - Collecter - Vérifier - Stocker - Diffuser – Exploiter Date mise en application du document : 04.04.2012 Type de document : **Cahier des charges**, dossier de conception Titre du document : Outil d'intégration des bases embarquées de la CLPA Référence projet : CLPA_IntegrationDesBasesEmbarquees Référence module : Nom de ficher : CLPA_IntegrationDesBasesEmbarquees_CDC_Conception_v1.doc		

Tableau 1 : Tableau de référence du document

Relecteur	Version relue	Date de transmission ou de mise à disposition (réseau)	Date de retour ou date de modification (réseau)	Remarques
Transmission à EM le 20/04/2012, version v1				
EM				
Transmission à le				
Diffusion et mise en œuvre base version n° le				

Tableau 2 : Circuit de relecture du document

Type de vérification	Nature de la vérification	Respecté	Non respecté
Identification du fichier	Nom de fichier normalisé		x
	Numéro de version mis à jour		x
Identification du document (en-tête et pied de page)	Sigle du ou des dispositifs concernés		x
	Titre du document en en-tête		x
	Date de mise à jour		x
Forme	Utilisation d'un des modèles du projet		x
	Tables des matières mise à jour		x
	Numérotation des pages dans la table mise à jour		x
Références internes et externes	Numéro des annexes cité explicitement		x
	Points de référence précis dans le texte (absence de formule « ci-dessous... », « ci-dessus... »		x
Suivi de version	Mise à jour du circuit de relecture		x

Tableau 3 : Suivi qualité du document

Outil d'intégration
des bases embarquées de la CLPA

1. Cahier des charges

1.1. Contexte

La base de données du dispositif CLPA est alimentée par des données collectées par les chargés d'études CLPA. Chaque chargé d'études mène une enquête sur une ou plusieurs zones d'études. Au cours de l'enquête, il saisit ses données via une interface VBA mappée sur une base de données embarquée Access. A son retour, les données de la base embarquée doivent être intégrées dans la base de données centrale CLPA_PROD (base mère), gérée par le SGBD Oracle. Après une mise à jour de la base CLPA_PROD, de nouvelles bases embarquées peuvent être générées à partir de la base mère. Ces bases embarquées sont ensuite confiées aux chargés d'études pour leur permettre de poursuivre la récupération des données sur le terrain.

Historiquement, le transfert des données depuis la base embarquée sous Access vers la base mère sous Oracle a été réalisé via une interface VBA pour des cas d'utilisation simples quand les identifiants de la base embarquée étaient identiques aux identifiants de la base mère pour le plupart des tables :

\\Bipro4\bda\Informatique\BD\CLPA\EnCours\Actions\TransfertAccessOracle

\\bipro4\bda\Informatique\BD\CLPA\Valide\Actions\TransfertAccessOracle

Cette interface ne peut pas être utilisée pour le cas général des bases embarquées quand les identifiants Access sont différents des identifiants Oracle. C'est le cas des bases de données embarquées obtenues récemment à Irstea (par exemple, Queyras 2010, Allos-Praloup 2011, Tournette-Sambuy 2011). Nous avons donc besoin d'un outil qui permet d'intégrer proprement ces bases embarquées.

1.2. Objectifs

L'objectif principal de ce projet est de créer un outil « IntégrationBaseCE » permettant d'intégrer les bases embarquées récentes Access dans la base mère Oracle pour lesquelles l'utilisation de l'interface historique VBA n'est pas possible. Cet outil doit assurer la cohérence et l'intégrité de la base mère après l'importation des données même si les identifiants dans les tables correspondantes sont différents.

L'utilisation de l'outil « IntégrationBaseCE » est prévue pendant une période de 6 mois à 1 an. A l'avenir, son utilisation sera remplacée par une interface web BDA en cours de développement aujourd'hui. La conception de l'outil « IntégrationBaseCE » doit donc permettre la réutilisation de ses composants pour des projets futurs impliquant la base de données CLPA_PROD, tel que la migration des données de CLPA_PROD vers la base BDA, par exemple.

1.3. Participants

- Maîtrise d'ouvrage : Mylène Bonnefoy
- Maîtrise d'œuvre : Eric Maldonado
- Développement : Ekaterina Flin

1.4. Contraintes

Etant donné que l'utilisation de l'outil « IntégrationBaseCE » sera temporaire, on met la priorité sur la rapidité de son développement ainsi que sur la réutilisation de ses composants pour d'autres projets à l'avenir.

L'outil d'intégration des bases embarquées « IntégrationBaseCE » doit être développé le plus rapidement possible (en 2 mois environ) pour pouvoir intégrer les bases existantes et produire de nouvelles bases embarquées au printemps 2012.

L'outil ne possédera aucune interface graphique.

L'outil doit permettre la réutilisation facile de ses composants. Il doit assurer :

- l'assemblage de ses composants avec les composants Java existants du projet BDA
- le mapping des tables CLPA_PROD en objets Java en vue de leur futur transfert vers BDA
- les connexions Access et Oracle, extensible PostgreSQL en cas de migration de la base BDA sous PostgreSQL.

2. Conception

2.1. Organisation du projet

Le projet de développement est organisé en cycles itératifs rapides de quelques jours, selon les principes des méthodes Agile. A chaque cycle de développement, un prototype fonctionnel est livré et archivé avec un logiciel de gestion de versions Subversion. Les versions successives sont hébergées sur la forge logicielle Irstea aux adresses suivantes :

- Forge d'Irstea : https://forge.cemagref.fr/
- Dépôt SVN : https://svn.cemagref.fr/svn/integration-base-ce

Un cahier de travail du développeur comportant des notes techniques prises au cours du développement est disponible sur le wiki du projet :
http://wiki.grenoble.cemagref.fr/dokubda/doku.php?id=developpement_et_logiciels:cahiers_de_travail

Le projet a atteint actuellement sa 14-ème version et comporte environ 9000 lignes de code.

2.2. Spécifications

Le projet « IntégrationBaseCE » est développé en langage Java dans l'EDI Eclipse avec un plugin Subclipse pour la gestion des versions. Il utilise les drivers ODBC et JDBC 6 pour se connecter aux bases de données Access et Oracle.

L'outil est conçu selon le patron de conception DAO/POJO (voir le tutoriel [1]). Les objets métier sont exprimés par les classes POJO (Plain Old Java Object) et correspondent globalement aux tables de la base de données. L'accès à la couche de données est assuré par les classes DAO (Data Access Object). Les connexions aux bases Access et Oracle sont réalisées selon le patron de conception Singleton qui permet d'assurer l'unicité de la connexion à chaque base.

L'insertion des données dans la base Oracle est gérée par les transactions avec la granularité de la base embarquée (fichier mdb) : soit toutes les données de la base embarquée sont intégrés dans la base mère, soit aucune donnée n'est insérée. Cette démarche permet d'éviter des données incohérentes dans la base mère suite à des tentatives d'intégration échouées. L'outil n'insère aucune donnée dans la base embarquée Access, il ne se connecte à cette base que pour la lecture.

L'outil « IntégrationBaseCE » ne détruit aucune donnée dans les bases Access et Oracle, même s'il détecte des incohérences susceptibles d'être résolues par une suppression de certaines données

incorrectes ou redondantes. En cas d'incohérences détectées, l'outil affiche un warning pour laisser à l'opérateur la décision de l'action appropriée.

2.3. Modèles des bases de données

Les bases embarquées sous Access et la base mère CLPA_PROD sous Oracle ont des modèles de données différents bien que très similaires. Le modèle physique de données (MPD) de la base embarquée représente un modèle réduit et simplifié de la base mère CLPA_PROD. Il contient uniquement une partie des tables directement concernées par le travail des chargés d'études. Les attributs des tables du même nom dans les deux bases ne sont pas forcement les mêmes. Par exemple, la table DEPARTEMENT comporte 3 attributs sous Access et 6 attributs sous Oracle.

2.3.1. Modèle de la base embarquée (Access)

La figure 1 montre le modèle de la base embarquée. La plupart des associations ont les cardinalités 1-n. Le point d'entrée est la table ENQUETE.

On peut logiquement séparer le schéma de la base de données en 3 branches en partant de la table FICHE :

- La branche « enquête » contient la table ENQUETE et les tables associées (FILIATION, TYPE_FILIATION, etc.) ainsi que la table FICHE et toutes ses tables associées avec les cardinalités n (TYPE_ENQUETE, TYPE_FICHE, etc.).
- La branche « photo » contient les tables PHOTO, OUVRAGE, TYPE_OUVRAGE et EPA.
- La branche « témoignage » contient la table TEMOIGNAGE et toutes les tables associées.

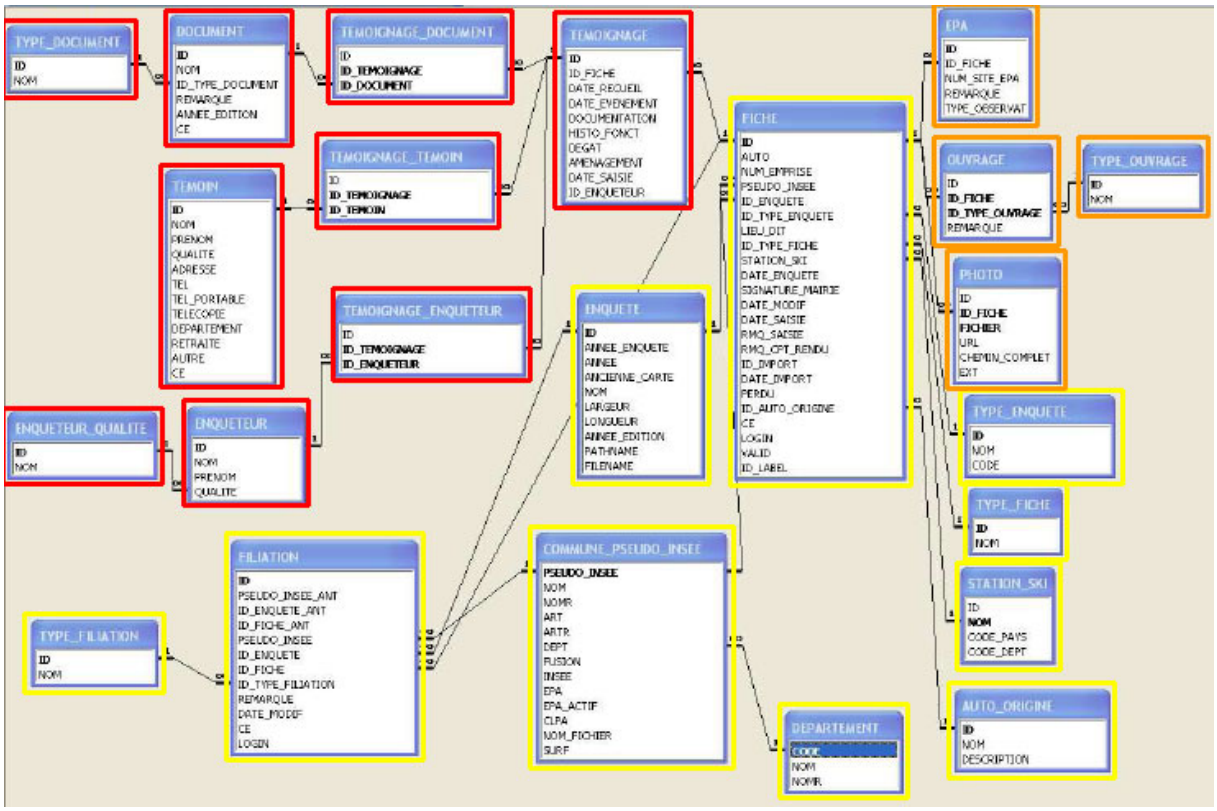


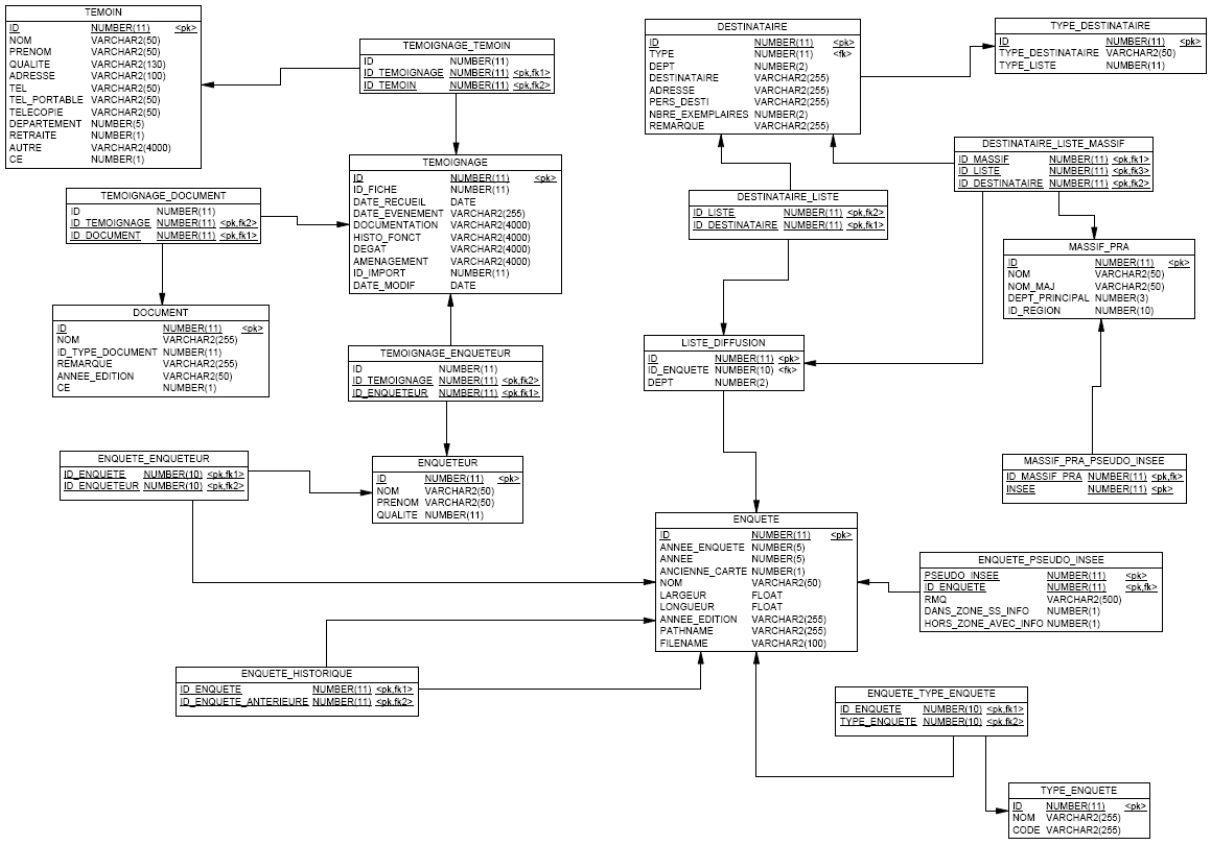
Figure1. Modèle de données de la base embarquée.

La table TEMOIGNAGE_ENQUETEUR n'est pas réellement utilisée dans la base embarquée. Le lien est assuré par l'attribut ID_ENQUETEUR dans la table TEMOIGNAGE. On a donc la cardinalité 1-n à la place de n-n.

Le chargé d'étude travaille principalement avec les tables ENQUETE, FICHE, FILIATION, TEMOIGNAGE, DOCUMENT, TEMOIN, PHOTO et leurs associations. Les autres tables ne changent que très rarement, voire pas du tout.

2.3.2. Modèle de la base mère CLPA_PROD (Oracle)

La figure 2 montre le MPD de la base CLPA_PROD.



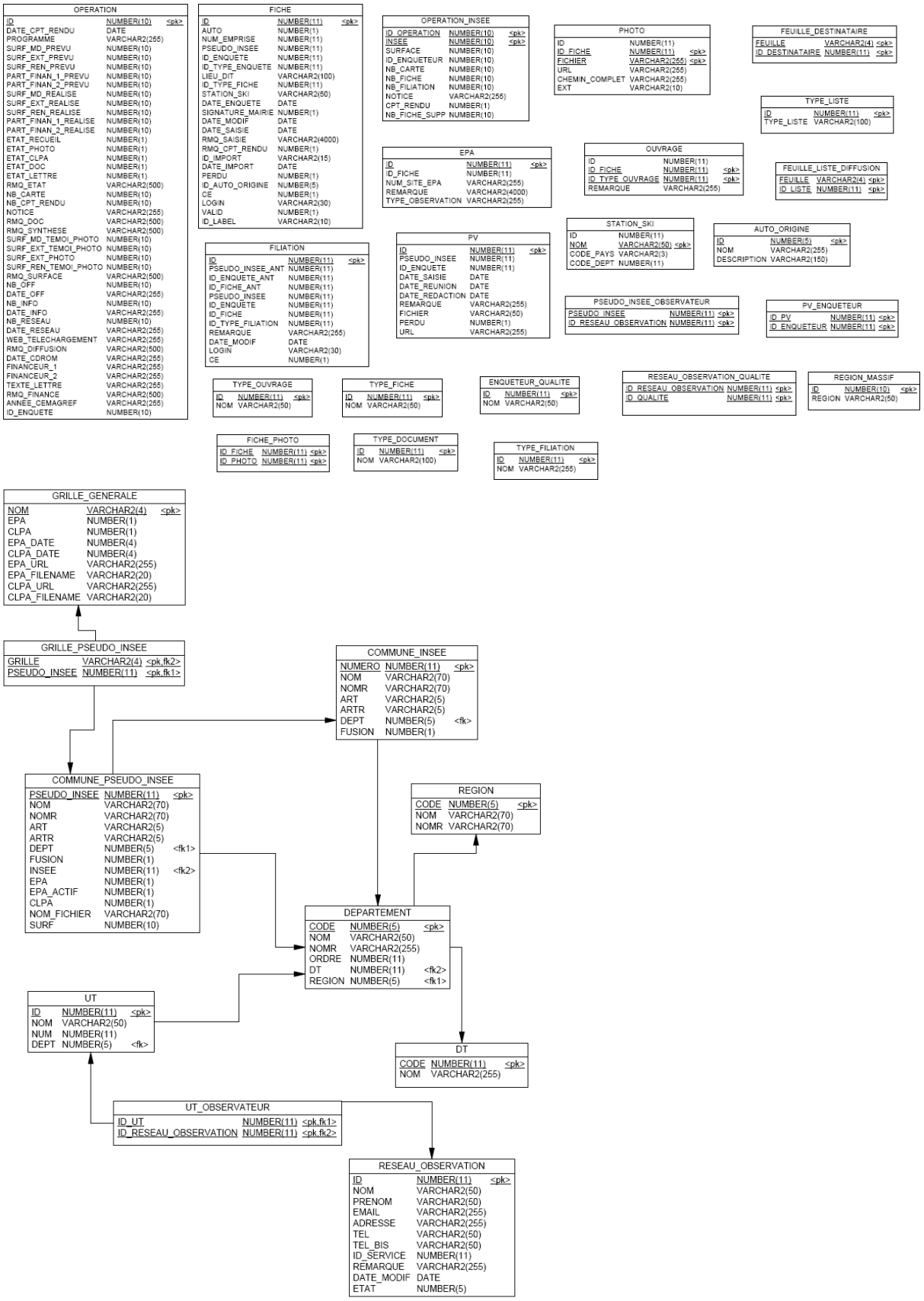


Figure 2. Modèle physique de données de la base CLPA_PROD.

On peut remarquer que certains liens sur les clés étrangères qui doivent logiquement être présents dans le schéma ne sont pas exprimés. Cela vient du fait que les contraintes sur ces clés étrangères ne sont pas définies explicitement dans la base de données.

La base de données CLPA_PROD contient plus de tables que la base embarquée. Elle contient également des triggers et des associations supplémentaires. Par exemple, la table FICHE_PHOTO représente une association entre FICHE et PHOTO avec les cardinalités n-n tandis que dans la base embarquée les cardinalités sont 1-n et la table FICHE_PHOTO n'existe pas. Un trigger permet de remplir automatiquement la table FICHE_PHOTO dès la mise à jour de la table PHOTO.

La table TEMOIGNAGE_ENQUETEUR représente une association entre TEMOIGNAGE et ENQUETEUR. Elle est utilisée et doit être remplie lors de l'importation des données. Par conséquent, la table TEMOIGNAGE ne comporte pas d'attribut ID_ENQUETEUR.

Les attributs de type booléen dans la base embarquée ont systématiquement le type entier dans la base CLPA_PROD.

2.4. Structure du projet java

2.4.1. Description des paquetages

Le programme est organisé en 7 paquetages représentés sur la figure 3.

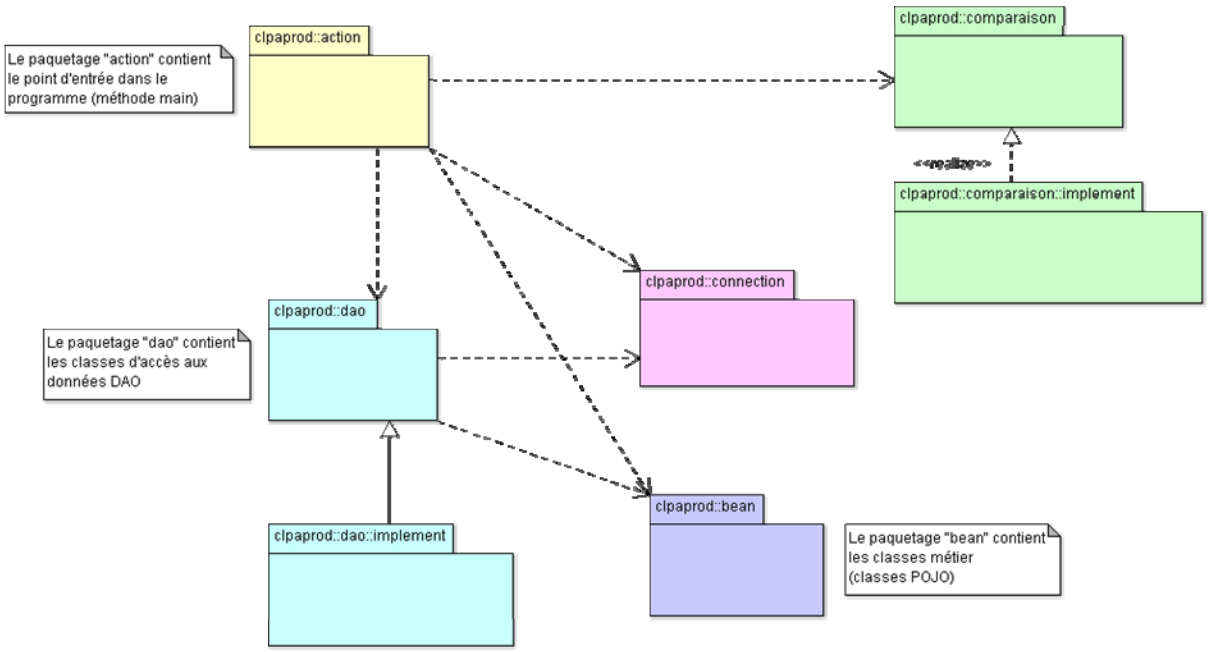


Figure 3. Diagramme des paquetages.

Le paquetage « action » contient les classes qui réalisent les algorithmes d'intégration des données et les vérifications associées (par exemple, un bilan des filiations). Ce paquetage est le seul point d'entrée dans le programme avec la méthode *main* de la classe TransfertAccessOracle.

Le paquetage « connection » contient les classes de connexion aux bases de données Access et Oracle.

Le paquetage « bean » contient les classes métier.

Le paquetage « dao.implement », généralisé par le paquetage « dao », contient les classes d'accès aux données DAO.

Les paquetages « comparaison » et « comparaison.implement » permettent d'effectuer une comparaison de données entre les bases Access et Oracle.

2.4.2. Description des classes métier et DAO

Les classes métier sont représentées par des classes java simples (classes POJO) pour modéliser les tables communes des bases de données Access et Oracle. De manière générale, les attributs des classes métier reprennent les attributs des tables correspondantes. Chaque objet métier correspond à un tuple de données dans une table. Dans certains cas, les mêmes tables dans les bases Access et Oracle n'ont pas des attributs identiques comme c'est le cas, par exemple, de la table TEMOIGNAGE. Certains attributs peuvent être obsolètes (par exemple, l'attribut ID_IMPORT de la table TEMOIGNAGE dans CLPA_PROD). Dans ces cas particuliers, les classes métier sont modélisées uniquement sur les attributs communs des deux bases. On ne modélise pas les attributs obsolètes ou non utilisées. La figure 4 montre un exemple de modélisation de la classe métier Temoignage à partir de la table TEMOIGNAGE des bases Access et Oracle. Les types des attributs se rapprochent au maximum des types correspondants dans la base CLPA_PROD. Le type de données DATE du SGBD Oracle est modélisé par la classe Date du paquetage standard java.sql.

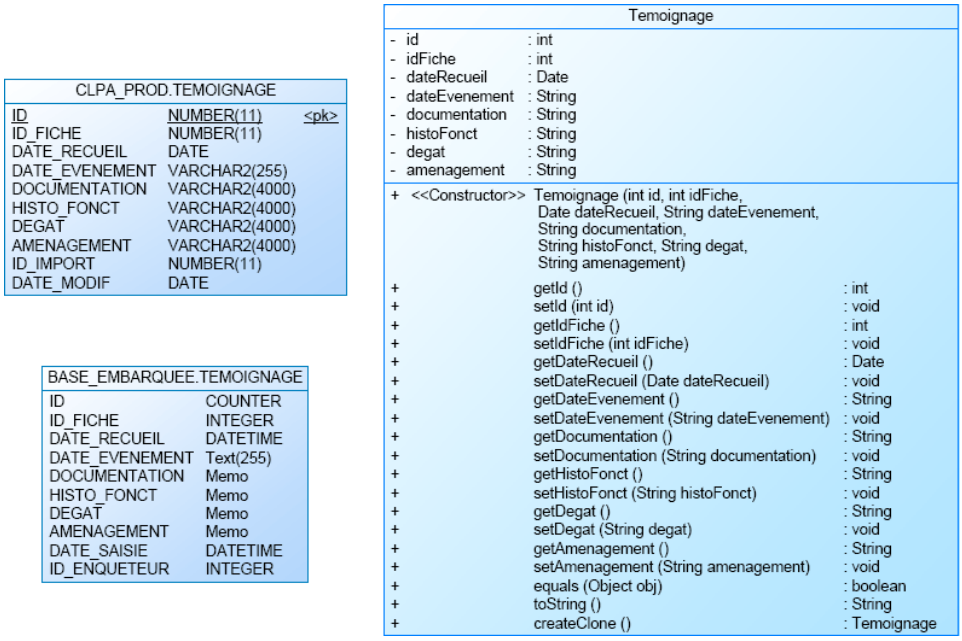


Figure 4. Exemple de modélisation de la classe Temoignage à partir de la table TEMOIGNAGE de la base embarquée et de la base CLPA_PROD.

Les tables de la base de données correspondant aux associations sont également modélisées par des classes POJO. Par exemple, la table TEMOIGNAGE_TEMOIN qui représente une association entre deux entités TEMOIGNAGE et TEMOIN est modélisée par une classe java TemoignageTemoin.

Les classes d'accès aux données DAO sont destinées à faire le lien entre les données contenues dans les bases de données et les classes métier. A chaque classe métier correspond une classe DAO. La figure 5 montre un diagramme de classes pour trois classes DAO TypeFicheDao, TemoinDao et EnqueteDao ainsi que leurs relations avec les classes métier. Toutes les classes DAO

doivent spécialiser une classe abstraite Dao. Les classes qui se chargent de l'insertion de données dans la base de données doivent en plus réaliser une interface Insert.

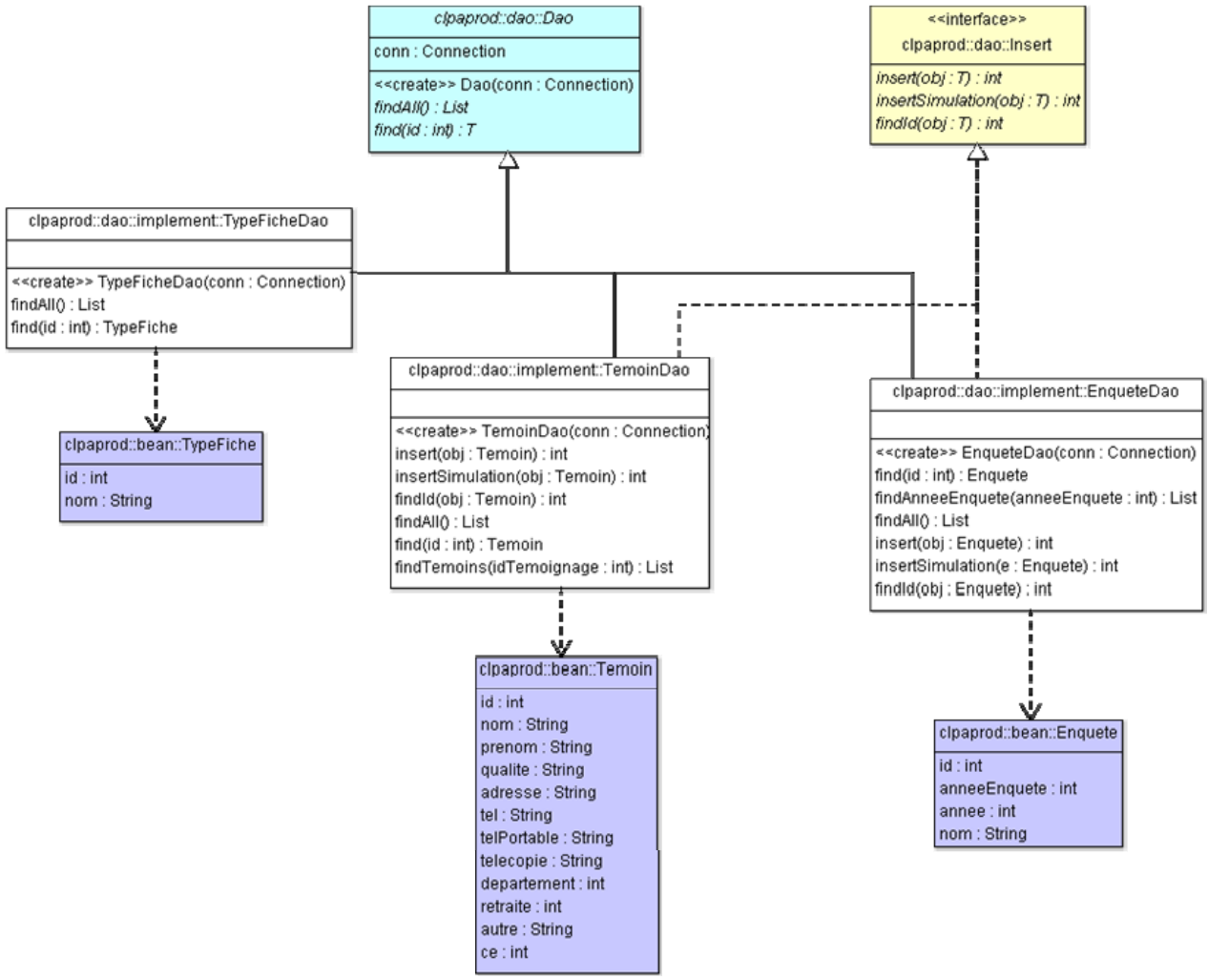


Figure 5. Diagramme des classes DAO.

Au cours de l'intégration de la base embarquée, l'outil « IntégrationBaseCE » effectue l'insertion de données dans les 11 tables de la base CLPA_PROD suivantes :

Table	**Classe POJO**	**Classe DAO**
ENQUETE	Enquete	EnqueteDao
FICHE	Fiche	FicheDao
FILIATION	Filiation	FiliationDao
PHOTO	Photo	PhotoDao
TEMOIGNAGE	Temoignage	TemoignageDao
TEMOIN	Temoin	TemoinDao
TEMOIGNAGE_TEMOIN	TemoignageTemoin	TemoignageTemoinDao
DOCUMENT	Document	DocumentDao
TEMOIGNAGE_DOCUMENT	TemoignageDocument	TemoignageDocumentDao
ENQUETEUR	Enqueteur	EnqueteurDao
TEMOIGNAGE_ENQUETEUR	TemoignageEnqueteur	TemoignageEnqueteurDao

Tableau 4. Liste des tables dans lesquels l'outil d'intégration insère de nouvelles données.

2.4.3. Description des classes dédiées à la connexion aux SGBD

La connexion aux bases de données dans le programme « IntégrationBaseCE » est réalisée selon le patron de conception Singleton. Le paquetage « connection » contient deux classes AccessConnection et OracleConnection qui permettent de se connecter au SGBD correspondant (figure 6). Pour l'intégration de données, on a besoin d'une instance de connexion à la base embarquée et une instance de connexion à la base CLPA_PROD. Le design pattern Singleton permet de s'assurer qu'on ne crée pas plus d'une seule connexion à chaque base.

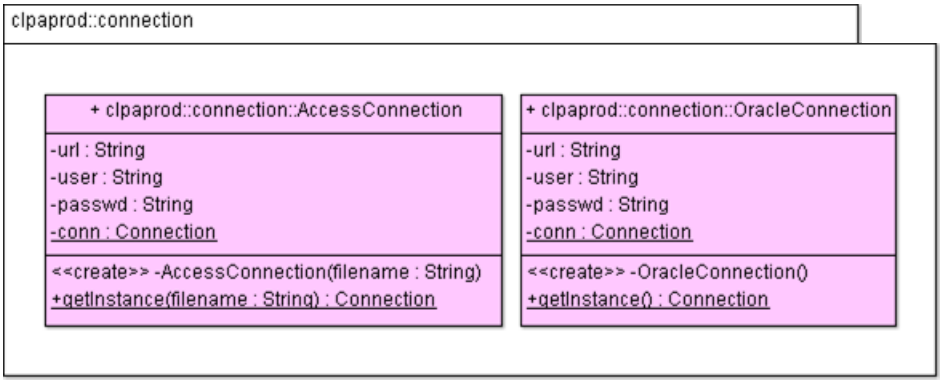


Figure 6. Classes de connexions aux bases de données Access et Oracle.

2.5. Algorithme d'intégration d'une base embarquée

Pour définir un algorithme d'intégration d'une base embarquée, nous avons repris la procédure de travail avec l'outil historique (interface VBA) :

\\bipro4\BDA\Informatique\BD\CLPA\Valide\Global\DocsTechniques\CLPA_Notice_TransfertAccessOracle_v1.doc

D'après cette procédure, l'opérateur copiait d'abord les photos prises par le chargé d'études sur un serveur commun (\\Bipro4\BDA\Images\CLPA\Photo\Valide). Il devait ensuite réaliser un bilan des filiations de la base embarquée, effectuer l'insertion de nouvelles données à travers l'interface VBA, mettre à jour le chemin d'accès de chaque photo dans la base mère et terminer ensuite par un bilan des filiations de la base mère. Finalement, l'opérateur comparait les deux bilans pour juger si l'insertion s'était bien passée ou non. Ces opérations, sauf l'insertion de données, se faisaient manuellement pour chaque nouvelle enquête.

Dans le nouvel outil « IntégrationBaseCE », nous avons structuré la procédure d'intégration en trois actions, similaires aux actions de l'opérateur : faire un bilan, rechercher une photo, insérer des données. L'action « faire un bilan » s'applique deux fois, une fois à la base embarquée et une autre fois à la base mère après l'insertion des données. Toutes ces actions se font automatiquement, y compris la comparaison des deux bilans à la fin de l'intégration. La seule action manuelle reste la copie des photos sur le serveur commun.

A chaque action correspond une classe. La figure 7 montre l'ensemble des classes impliquées dans l'algorithme d'intégration d'une base embarquée. La classe Bilan permet de faire un bilan des filiations. La classe RecherchePhoto détecte l'emplacement des photos sur le serveur commun. La classe Insertion intègre les nouvelles données dans la base CLPA_PROD. Ces actions sont regroupées dans la classe principale TransfertAccessOracle qui représente le seul point d'entrée dans le programme.

Un chargé d'études lors de son travail renseigne des données dans 11 tables de la base embarquée qui feront l'objet d'une insertion dans la base mère par la suite (voir le tableau 4). En ce qui concerne les autres tables, appelées ici tables secondaires, elles restent à priori inchangées et strictement identiques à la base Oracle, même pour ses identifiants. Dans notre algorithme, on part du principe que ces tables ne changent pas. Toutefois, on vérifie cette hypothèse à chaque intégration en comparant les données de chaque table secondaire entre la base embarquée et la base mère. La comparaison des tables secondaires est réalisée dans le programme selon le design pattern Strategy.

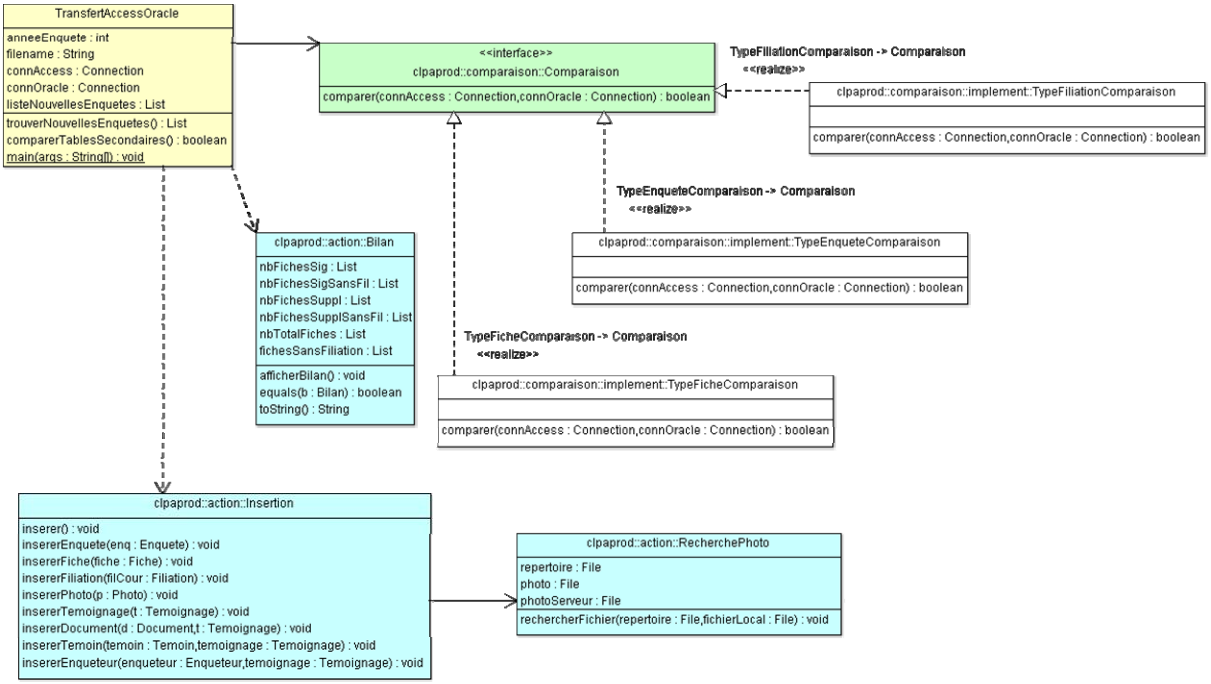


Figure 7. Diagramme des classes applicatives qui réalisent l'algorithme d'intégration d'une base embarquée.

2.5.1. Intégration de la branche « enquête »

L'algorithme d'intégration d'une base embarquée commence par la recherche de nouvelles enquêtes crées par le chargé d'études et par la vérification des tables secondaires. Pour chaque nouvelle enquête on dresse une liste des fiches signalétiques attachées à cette enquête ainsi que leurs filiations.

Chaque fiche peut avoir plusieurs filiations. L'information sur les filiations est contenue dans la table FILIATION. Pour chaque fiche ID_FICHE on peut trouver aucune ou plusieurs fiches antérieures ID_FICHE_ANT. La filiation s'applique sur les fiches et non sur les enquêtes. L'information sur les enquêtes ID_ENQUETE et ID_ENQUETE_ANT dans la table FILIATION est récupérée dans les attributs des fiches filées. Il s'agit d'une information dérivée : deux enquêtes sont considérées comme filées si une enquête contient au moins une fiche filée à une autre fiche appartenant à la deuxième enquête.

Pour reconstituer l'arbre des fiches filées, on utilise une structure de données de type file d'attente FIFO. Au début on met la fiche courante dans la file d'attente. On récupère le premier élément de la file d'attente. Pour cet élément, on cherche ses fiches antérieures et on les rajoute une à une dans la file d'attente. On passe au deuxième élément dans la file d'attente et ainsi de suite. Quand la file d'attente est vide, cela veut dire qu'on a parcouru toutes les fiches de la plus récente aux plus anciennes. Ce mécanisme est utilisé pour créer un bilan des filiations.

Les nouvelles enquêtes, fiches et filiations sont insérées dans la base CLPA_PROD, soit avec le même identifiant s'il est disponible, soit avec un nouveau identifiant attribué automatiquement par Oracle.

2.5.2. Intégration de la branche « photo »

La table PHOTO porte de nouvelles données qui doivent être insérées dans la base CLPA_PROD. Les photos dans la base embarquée ont un chemin du PC local du chargé d'études. Elles doivent d'abord être copiées dans le répertoire \\bipro4\BDA\Images\CLPA\Photo\Valide du serveur commun, en créant un nouveau répertoire <enquete.annee>_<enquete.nom>. Les photos y sont placées manuellement, en dehors du programme Java.

Pour chaque fiche signalétique, le programme Java cherche des photos correspondant à cette fiche. Il récupère le nom de la photo dans la base embarquée et cherche le chemin complet de cette photo dans son emplacement final sur bipro4 :

\\bipro4\BDA\Images\CLPA\Photo\Valide\<enquete.annee>_<enquete.nom>.

Il remplit ensuite la table PHOTO de la base CLPA_PROD.

2.5.3. Intégration de la branche « témoignage »

Une fiche signalétique peut comporter plusieurs témoignages. A leur tour, les témoignages peuvent être associés à plusieurs documents. Un témoignage peut être signé par un ou plusieurs témoins et enregistré par un enquêteur.

Le programme insère chaque témoignage et récupère son ID dans la base CLPA_PROD. Ensuite, il vérifie si les documents, les témoins et l'enquêteur associés à chaque témoignage sont déjà présents dans la base CLPA_PROD. Si c'est le cas, il récupère leur identifiants et remplit les tables d'associations correspondantes. Sinon, il remplit également les tables DOCUMENT, TEMOIN et ENQUETEUR.

2.5.4. Gestion des transactions

L'insertion de données dans la base CLPA_PROD s'effectue en mode transaction qui préserve au maximum l'intégrité des données en cas d'accès multiples. Par défaut, Oracle assure les principes ACID pour toute transaction classique via la sérialisation des accès (voir [2]).

L'entrée en mode transaction est gérée au niveau du programme Java par les classes DAO pour chaque méthode qui réalise une insertion. A la fin de l'algorithme d'insertion, le programme principal TransfertAccessOracle décide si la suite des insertions effectuées peut être validée par COMMIT ou non. Pour cela, il dresse deux bilans de filiations des bases Access et Oracle. Dans le cas où ces bilans sont identiques, les insertions sont validées par COMMIT. Dans le cas contraire ou si le programme rencontre une quelconque erreur au cours de son exécution, toutes les insertions sont annulées par ROLLBACK.

2.5.5. Gestion des identifiants

Dans le cas général, les identifiants attribués aux mêmes données dans la base embarquée et dans la base CLPA_PROD sont différents. Pour éviter des doublons, toute insertion dans la base CLPA_PROD commence par une vérification si les données qu'on cherche à insérer existent déjà dans la base mère sous un autre identifiant.

Dans la mesure du possible, on essaie d'insérer les tuples de nouvelles données dans la base CLPA_PROD avec les mêmes identifiants que dans la base embarquée. Si un identifiant n'est pas disponible, l'insertion se fait avec un identifiant automatique attribué par Oracle.

La correspondance entre les identifiants Access et Oracle est stockée dans les attributs tuple<nom> de la classe Insertion. Par exemple, tupleEnquete, tupleFiche, tupleFiliation, etc. Ils sont réalisés avec des collections de type HashMap où la clé correspond à l'identifiant Access et la valeur correspond à l'identifiant Oracle.

2.5.6. Gestions des valeurs NULL

Quand on récupère des données de type NULL (dans le sens SQL) depuis une base de données embarquée via JDBC pour les stocker en objets Java, des transformations de types s'appliquent par défaut selon les règles suivantes :

- Si un attribut Java est de type int, il prend la valeur 0 ;
- Si un attribut Java est de type String, sa référence devient null (dans le sens objet Java) ;
- Si un attribut Java est de type Date (java.sql), sa référence devient également null.

Le programme doit tenir compte de ces transformations au moment de l'insertion de ces données dans la base CLPA_PROD à partir des objets Java. Si la transformation systématique des références null Java en NULL SQL paraît tout à fait raisonnable, il n'en est pas de même pour les valeurs 0 des attributs de type int.

Par exemple, si un objet « :Filation » a son attribut « idFicheAnt » égal à 0, au moment de l'insertion on doit le transformer en NULL car la valeur 0 signifie ici qu'il n'y a pas de fiche antérieure pour cet objet. La requête correspondante sera :

INSERT INTO FILIATION (ID_FICHE_ANT) VALUES (NULL) ;

Au même temps, si l'attribut « ce » d'un objet « :Temoin » est égal à 0, on ne peut pas le transformer en NULL car il s'agit ici d'une vraie valeur 0 (qui signifie que ce témoin n'est pas un chargé d'études) :

INSERT INTO TEMOIN (CE) VALUES (0) ;

La décision sur la transformation de 0 en NULL se fait au cas par cas pour chaque attribut. Elle est gérée par les classes DAO. De manière générale, on applique la transformation de 0 en NULL pour les clés étrangères et on garde la valeur 0 pour les autres attributs.

3. Test

Pour tester l'outil « IntégrationBaseCE », on a appliqué des tests unitaires et des tests fonctionnels. Des copies d'instances des bases embarquées de 2010 et de 2011 ont été utilisées pour construire les vecteurs d'entrée des tests correspondants. Pour tester les méthodes d'insertion des classes DAO, nous avons utilisé une copie d'instance de la base CLPA_PROD sur sandbox-

bda.grenoble.cemagref.fr. Les tests ont été exécutés manuellement en comparant les sorties de l'OST (objet sous test) et les sorties des interfaces VBA existantes.

Le logiciel est actuellement en phase de test d'intégration sur sandbox-bda.grenoble.cemagref.fr. Nous continuons également des tests aux limites. Les notes concernant ces tests sont disponibles sur le wiki du projet :

- http://wiki.grenoble.cemagref.fr/dokubda/doku.php?id=developpement_et_logiciels:bases_a_integrer
- http://wiki.grenoble.cemagref.fr/dokubda/doku.php?id=developpement_et_logiciels:remarques_sur_l_integration_des_bases_embarquees

4. Bibliographie

[1] Tutoriel d'utilisation de JDBC avec le patron de conception DAO (voir les chapitres 7, 9 et 10) :
http://www.siteduzero.com/tutoriel-3-119239-programmation-en-java-api.html

[2] Utilisation des transactions dans le SGBD Oracle avec JDBC :
http://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html