MODIFICATIONS DE LA BASE DE DONNEES AREL

Boisson Matthieu
Chouzenoux Pierrick
Samson Julien
Wagner Guillaume
Willems Bruno

TABLE DES MATIERES

AVANT-PROPOS

Avant toute chose, il est important de bien comprendre le but de ce document : il s'agit non seulement de savoir de quelle façon la base de données a été modifiée mais également la norme qui a été mise en place pour harmoniser la base.

Afin de faciliter la lecture de ce document et de faire le lien avec l'ancienne base, il est donc important de connaître globalement quelles ont été les tables modifiées, supprimées, celles dont le nom a été changée…

Et il est tout aussi important de comprendre la norme mise en place, afin qu'elle soit toujours respectée par la suite, pour permettre de reprendre plus facilement le code SQL.

Enfin, on retrouve au cours de ce document quelques suggestions qui n'ont pu être mis en place faute de temps et de distribution aux autres étudiants.

LEGENDE

Pour une meilleure compréhension des modifications appliquées à la base de données, veuillez vous rapporter à la légende suivante :

TABLE INCHANGEE

TABLE MODIFIEE

TABLE SUPPRIMEE

TABLE AJOUTEE

Certaines tables ont également été renommées ou fusionnées avec d'autres, elles sont alors indiquées de la manière suivante :

ANCIEN NOM DE LA TABLE ➧ NOUVEAU NOM DE LA TABLE

TABLE FUSIONNEE ➧ TABLE DANS LAQUELLE ELLE EST FUSIONNEE

Notez toutefois que pour une meilleure lisibilité de la base, l'ensemble des tables s'est vu raccourcir du fameux A_ qui n'a plus lieu d'être. Cette modification n'est pas prise en compte comme un changement du nom de la table.

Enfin, les modifications des noms des champs ne sont pas prises en compte comme modification d'une table, car l'intégralité des tables a été modifiée afin de respecter une norme.

Pour la description des tables, on détaille chaque champ. Afin de faciliter la compréhension de la base, on applique la même légende que l'on a applique aux tables.

Si les modifications de noms dues aux changements de norme n'ont pas été prises en compte dans la légende des tables, il en va de même pour les champs. Ainsi un champ peut ne pas être nommé tout à fait de la même façon que dans l'ancienne base et ne pas être indiqué en violet, tant que la correspondance est évidente.

Place maintenant à un résumé rapide des tables de la base.

TABLES

ABSENCES_JUSTIFIEES

ACTEURS

ACTEURS_LANGUES → ACTEURS

ACTEURS_MAIL → ACTEURS

ACTIONS

ACTIONS_ACTEURS → ACTIONS_STATUTS

ACTIVITES

ACTIVITES_TYPES → RELS

ACTIVITE_PREREQUIS

ADRESSES_MAIL

ALERTES

ANNEES

ARCHIVE_ETUDIANTS

ARCHIVE_TRAVAUX

ARTICLES

CHOIX_PROGRAMMES

CORRECTIONS → DROITS_ACTEURS

CRENEAUX

CYCLES

DEPARTEMENTS

DOCUMENTS

DOC_TYPES → RELS

DOC_TYPES_XML → XML_TYPES

DROITS_ABSENCES

EDT

EDT_CL_SA_PR

EDT_GEN → EDT

ESPACES

ETUDIANTS

ETUDIANTS_ABSENCES

EVTS

FICHIERS

FILIERES

FONCTIONS

GROUPES

GROUPE_GROUPE → GROUPES_PARENTES

GS_ETUDIANTS

GS_GROUPES

INSCRIPTIONS_SESSIONS

LOG → LOGS

MATIERES

MATIERES_ETUDIANTS

MATIERES_GESSCOL

MENU_PRINCIPAL → MENUS

MENU_PRINCIPAL_ACTEUR → MENUS_ACTEURS

MESSAGES

OPTIONS

PARCOURS

PARCOURS_ACTIVITES

PARCOURS_DATES

PARCOURS_DATES_ACTIVITES

PARENTS

PARENTS_ETUDIANTS

PAR_TYPES → RELS

PREREQUIS_PROGRAMMES

PROFESSEURS

PROFESSEURS_DEP

PROFESSEURS_FILIERES

PROFESSEURS_OPTIONS

PROFESSEURS_RELS → DROITS_ACTEURS

PROFS_GESSCOL

PROGRAMMES

PROGRAMMES_MATIERES

PROGRAMMES_SESSIONS

PROMOS

PS_SEMESTRES

QCMS

QCMS_QUESTIONS

QCMS_REPONSES

QUESTIONS

QUESTIONS_SUIVI → QUESTIONS_SUIVIS

QUESTIONS_TYPES

RELS

RELS_INDEXATION → RELS

RELS_MAJ → RELS

RELS_PREFEREES

RELS_RELS → RELS_RELATIONS

RELS_SAV

RELS_SOUS_TYPE → RELS

RENDU

RENDU_QCMS → QCMS_RENDUS

REPONSES

RUBRIQUES

RUBRIQUES_CRITERES

RUBRIQUES_RELS

SALLES

SEMAINES

SEMAINIERS

SEMESTRES

SITES

STATUTS

TRAVAUX

TRAVAUX_ETUDIANTS

TYPE_PROGRAMMES

TYPES_GROUPES

UE

VALEURS_CRITERE → VALEURS_CRITERES

VISIBILITES

NORME

Afin d'harmoniser la base et de simplifier sa lecture, l'ensemble des noms de contraintes et champs a été normalisé. Il est important que cette norme soit respectée par la suite, afin d'assurer la cohésion de la base. La norme est la suivante :

Champ basique (pas de clé) :	[nomchamp]
Clé primaire :	p_[nomchamp]
Clé étrangère :	f_[nomchamp]
Clé primaire et étrangère :	pf_[nomchamp]
Contrainte de clé primaire :	pk_[nomtable]_[number]
Contrainte de clé étrangère :	fk_[nomtable]_[number]
Contrainte d'unicité :	un_[nomtable]_[number]
Contrainte check :	ck_[nomtable]_[number]

Toujours dans un souci de lisibilité, l'ensemble des champs doit se trouver dans l'ordre suivant : tout d'abord les clés primaires, suivi des clés étrangères, et enfin les autres champs.

De même, les contraintes doivent se trouver dans l'ordre suivant : contrainte de clé primaire, puis contrainte d'unicité et enfin contrainte de check. Chacune des contraintes doit ensuite apparaître dans le même ordre que les champs auquel elles sont associées.

Pour finir, on sépare les contraintes de clés étrangères dans un autre script afin de permettre la construction des tables sans problème de contrainte étrangère.

GESTION DES RESSOURCES E-LEARNING

Tout d'abord, il est important de définir ce qu'est une ressource e-learning (on parle également de rel). On considère comme étant une rel tout entité créée et gérée par les formateurs du site AREL.

Le système mis en place pour la gestion des rels est assez particulier, puisqu'il s'agit d'une sorte d'héritage en base de données. En effet, chaque table étant également une rel (et donc considéré comme une extension de la table rel) se voit attribuer une clé étrangère sur sa clé primaire, pointant vers la table rel. De cette manière, l'id d'une matière, activité (ou autre…) se révèle également être l'id d'une rel. Il est ainsi précisé dans ce document lorsqu'une table se révèle être une rel.

RELS_INDEXATION

Dans l'ancienne base, pas mal de modification des rels ont eu lieu. Faute de temps, ces modifications ont eu lieu par extension de la table RELS. Un premier travail a donc consisté à fusionner les tables d'extension avec la table RELS afin d'en faire une seule et de clarifier la base. RELS_INDEXATION fait partie de ces tables qui ont donc été supprimées.

RELS_MAJ

De la même façon, RELS_MAJ a également été fusionnée avec RELS.

RELS_SOUS_TYPE

De la même façon, RELS_SOUS_TYPE a également été fusionnée avec RELS.

RELS_SAV

Dans l'ancienne base de données, cette table servait à sauvegarder les rels. Le système d'archivage a été retiré temporairement, le temps de réfléchir à un système plus efficace. De cette façon, la table RELS_SAV a été supprimée.

RELS_RELATIONS

Cette table permet de modéliser les relations entre les rels. Elle n'a subie aucune modification si ce n'est qu'elle a été renommée pour plus de compréhension.

Ses champs sont définis de la façon suivante :

- **pf_pere** : la rel père dans la relation. Il s'agit d'une clé étrangère vers la table RELS.
- **pf_fils** : la rel fils dans la relation. Il s'agit d'une clé étrangère vers la table RELS.
- **relation** : définit le type de relation entre les deux rels. Suggestion : ce champ pourrait également être stocké dans une nouvelle table, et donc se transformer en clé étrangère.

RELS

Cette table représente les rels. Elle a été modifiée par rapport à l'ancienne base et présente les champs suivants :

- **p_id** : id de la rel.
- frel_dep : une rel n'appartient plus à un département particulier.
- frel_ue : les études d'étude n'existant plus, ce champ n'a plus de raison d'être.
- frel_annee : une rel n'a pas de raison d'être associée à une année.
- visibilite ➡ f_visibilite : visibilité dont dispose la rel. Elle a été modifiée d'un varchar2 à une clé étrangère vers la table VISIBILITES.
- **libelle** : nom de la rel.
- **rels_type** : type de la rel. En fait, comme une rel est forcément attaché à une entité d'une autre table, il s'agit ici du nom de la table concernée. Par exemple, une activité sera attachée à une rel de type activité.
- sous_type : précision du type de la rel concernant directement un type dans la table associée. Par exemple, une activité peut être attachée à une rel de sous-type TP ou TD. Ce champ a été en fait été remonté des tables descendant de rels.
- indexation : xml contenant l'ensemble de l'indexation de la rel. Sa structure exacte et son schéma restent encore à définir. Ce champ vient de l'ancienne table RELS_INDEXATION.
- date_maj : dernière date de mise à jour de la rel. Ce champ vient de l'ancienne table RELS_MAJ.
- nouvelle : précise si la rel est nouvelle ('y') ou pas ('n'). Ce champ vient de l'ancienne table RELS_MAJ.
- date_cre : date de création de la rel. Ce champ vient de l'ancienne table RELS_MAJ.
- etat : état de la rel. Ce champ vient de l'ancienne table RELS_MAJ.
- **libelle_court** : nom raccourci de la rel.
- date_publication : date à laquelle la rel sera publiée.
- option_publication : précise si la rel est à l'état de brouillon ('BROUILLON'), si elle est publiée ('PUBLIEE') ou si elle est en attente de publication ('ATTENTE').
- rels_publi_externe : précise si le stockage de la rel est gérée selon l'ancien système de stockage ou pas ('y' ou 'n'). Il est prévu de retirer ce champ par la suite. Ce champ vient de l'ancienne table RELS_MAJ.

RELS_PREFEREES

Cette table représente les rels favorites d'un acteur. La seule modification de cette table consiste en la modification de la clé étrangère vers acteurs en une clé vers groupes. Ceci est dû à la gestion des acteurs et groupes proposé dans la nouvelle base (voir gestion des acteurs pour plus d'informations).

Ses champs sont définis de la façon suivante :

- facteur_id : un favori n'est plus associé directement à un acteur.
- pf_groupe : un favori est désormais associé à un groupe (qui correspond à l'acteur). Il s'agit d'une clé étrangère vers la table GROUPES.
- **pf_rel** : la rel favorite de l'acteur. Il s'agit d'une clé étrangère vers la table RELS.
- **date_consultation** : la date de la dernière consultation de la rel.
- **score** : détermine l'importance du favori pour l'acteur.

GESTION DES INFORMATIONS

Il existe deux types d'informations disponibles sur AREL. En effet, on a d'une part tout ce qui concerne l'information pure : on parle alors d'espaces d'informations. Mais il ne faut pas oublier les emplois du temps, qui permettent également une prise d'information sur le site AREL.

GESTION DES ESPACES

Les espaces d'AREL représentent des espaces d'informations. Chacun de ces espaces est composé d'un ensemble de rubriques composées elle-même de rels. Une rubrique possède des critères particuliers, qui sont en fait des critères de recherche et qui vont ainsi servir lors d'une recherche sur le site. Chaque rubrique peut également contenir d'autres rubriques, formant ainsi une arborescence. Si l'on prend l'exemple de l'espace d'échanges actuellement visible sur AREL, il s'agit typiquement d'une arborescence de rubriques.

ESPACES

Cette table représente les espaces d'information. Ses champs sont définis de la façon suivante :

- **p_id** : id de l'espace.
- **presentation** : xmltype présentant l'espace d'information.
- frub_id : un espace d'informations n'est pas associé à une mais à plusieurs rubriques. Ce champ n'a donc pas de raison d'être.

RUBRIQUES

Cette table représente les rubriques présentes dans les espaces d'informations. Ses champs sont définis de la façon suivante :

- **p_id** : id de la rubrique.
- **f_pere** : rubrique père de la rubrique actuelle. C'est une clé étrangère sur la table elle-même.
- f_espace : espace auquel appartient cette rubrique. C'est une clé étrangère sur la table ESPACES.
- f_visibilite : visibilité de la rubrique. C'est désormais une clé étrangère sur la table VISIBILITES.
- **libelle** : nom de la rubrique.
- **rang** : rang de la rubrique au sein de l'espace.

RUBRIQUES_RELS

Cette table représente les liens entre les rubriques et les rels qu'elles contiennent. Ses champs sont définis de la façon suivante :

- pf_rubrique : rubrique contenant la rel. C'est désormais une clé étrangère vers la table RUBRIQUES.
- pf_rel : rel contenue dans la rubrique. C'est désormais une clé étrangère vers la table RELS.

VALEURS_CRITERES

Cette table représente l'ensemble des critères de rubrique. Ses champs sont définis de la façon suivante :

- **p_id** : id du critère.
- f_critere_id ➡ nom : ce n'est désormais plus une clé étrangère vers RUBRIQUES_CRITERES. Il s'agit du nom du critère. Champ supprimé puis rajouté après les soutenances
- valeur : valeur du critère.

Suggestion : Cette table devrait en réalité s'appeler CRITERES.

RUBRIQUES_CRITERES

Cette table fait désormais l'association entre les critères et les rubriques. Ses champs sont définis de la façon suivante :

- pf_rubrique : rubrique possédant le critère. C'est une clé étrangère vers la table RUBRIQUES.
- pf_critere : critère de la rubrique. C'est désormais une clé étrangère vers la table CRITERES.
- critere_id : étant désormais une table d'association, ce champ n'a plus de raison d'être.

GESTION DES EMPLOIS DU TEMPS

Les emplois du temps sont stockés dans la base de données sous la forme de différents créneaux. Ces créneaux contiennent toutes les informations que nécessitent les étudiants pour consulter leur emploi du temps.

SALLES

Cette table permet de représenter les salles de l'EISTI, qu'elles soient à Cergy ou Pau. Ses champs sont définis de la façon suivante :

- **p_id** : id de la salle.
- f_site : site auquel appartient la salle. C'est désormais une clé étrangère vers la table GROUPES (c'est donc un groupe de type SITE).
- libelle : nom de la salle.

EDT_CL_SA_PR

Cette table n'a plus de raison d'être du fait de la disparition du type CL_SA_PR anciennement présent dans la base. Il a été préféré une table d'association à l'instauration de ce type complexe. Cette table a donc été supprimée.

EDT

Sous le nom de cette table se cachent 2 anciennes tables de la base. En effet, les tables EDT et EDT_GEN étaient 2 versions de la table de stockage des emplois du temps. EDT_GEN étant la version la plus récente, c'est elle qui a servi de base à la nouvelle.

Ses champs sont définis de la façon suivante :

- **p_id** : id de l'edt
- f_rel : edt est désormais associé à une rel (clé étrangère) et plus à une matière.
- depositaire → f_site : ce champ représentant le site (cergy ou pau) auquel est attribué cet edt, il a été renommé pour plus de clarté. Par la même occasion, il a été changé en clé étrangère vers la table GROUPES (c'est un groupe de type site).
- fpromo_id : un edt n'est désormais plus associé à une promo (table supprimée).
- **hd** : heure de début de l'edt.
- **hf** : heure de fin de l'edt.
- **jour** : numéro du jour dans la semaine (1 à 7) de l'edt.
- **semaine** : numéro de la semaine dans l'année (1 à 52) de l'edt.
- **annee_civile** : numéro de l'année civile de l'edt.
- **date_saisie** : date de la saisie de l'edt.
- matiere : un edt n'est désormais plus associé à une matière mais à une rel.
- mat_libelle_court : champ associé à la matière, donc plus de raison d'être.
- groupe : cette relation est désormais dans la table d'association créneaux.
- login : ce champ n'a aucune raison d'être.
- sa_pr_cl : ce type complexe ayant été supprimé, ce champ l'est également.

CRENEAUX

Un créneau au sens propre du terme correspond en fait à un EDT. Cette nouvelle table fait son apparition pour gérer la présence de plusieurs divisions dans un même EDT. Ainsi pour un même créneau nous retrouvons plusieurs ensembles de salles, formateurs, apprenants. C'est donc elle qui remplace l'ancien type CL_SA_PR. Ses champs sont définis de la façon suivante :

- pf_edt : edt dont ce créneau est la division. C'est une clé étrangère vers la table EDT.
- pf_apprenants : groupe d'apprenants concerné par ce créneau. C'est une clé étrangère vers la table GROUPES.
- pf_formateurs : groupe de formateurs concerné par ce créneau. C'est une clé étrangère vers la table GROUPES.
- pf_salle : salle dans laquelle a lieu ce créneau. C'est une clé étrangère vers SALLES.

GESTION DES ACTEURS

Il s'agit sans doute de la partie qui a subie le plus de modifications. En effet, il faut savoir que le principe même des acteurs a été entièrement revu. Ainsi, même si un acteur est enregistré dans la table ACTEURS, cette table ne sera (presque) jamais référencée par les tables concernant des acteurs.

En fait, chaque acteur est associé à un groupe qui lui est propre : ce groupe contient donc uniquement cet acteur, et porte d'ailleurs un libellé propre à l'acteur dont il est issu. On référence alors la table GROUPES au lieu de référencer la table ACTEURS.

De cette façon, on permet la généralisation des actions : au lieu d'avoir un acteur qui pose une question, on a un groupe qui peut au choix être un unique acteur, ou bien un ensemble de groupes.

GESTION DES GROUPES

GS_GROUPES

Cette table fonctionnait avec l'ancien logiciel de gestion de la scolarité. Etant donné que celui-ci n'est plus utilisé, cette table n'a plus de raison d'être.

GROUPES

Cette table représente à la fois les acteurs et les regroupements d'acteurs. Elle a subie une légère modification par rapport à l'ancienne base. En effet il a fallut procéder à l'externalisation des types de groupes dans une autre table afin de rendre la base plus propre. Ses champs sont donc définis de la façon suivante :

- **p_id** : id du groupe
- f_type : le type du groupe est passé d'un varchar2 à une clé étrangère vers la nouvelle table TYPES_GROUPES.
- **libelle** : libellé du groupe (dans le cas d'un acteur, il s'agit de son nom raccourci)

TYPES_GROUPES

Cette table fait son apparition afin de permettre de définir un certain nombre de types de groupes, qui seront les seuls utilisés pour les groupes. Ses champs sont définis de la façon suivante :

- p_id : id du type
- libelle : libellé du type

NB : Afin de faire la correspondance avec l'ancienne base, cette table doit être au minimum remplie de tous les types de groupes présents dans l'ancienne base. De plus, les types suivants devront être directement ajoutés : 'PROMO', 'PARENT', 'APPRENANT', 'FORMATEUR', 'REGROUPEMENT', 'SITE', 'FONCTION'.

GROUPES_PARENTES

Cette table est une des rares tables à n'avoir subie aucune modification (si ce n'est son nom). Elle permet d'exprimer une relation d'encapsulation des groupes. Ses champs sont définis de la façon suivante :

- **pf_pere** : le groupe englobant de la relation. C'est une clé étrangère vers GROUPES.
- **pf_fils** : le groupe englobé de la relation. C'est une clé étrangère vers GROUPES.

BASE DES ACTEURS

ACTEURS_LANGUES

Dans l'ancienne base, pas mal de modification des acteurs ont eu lieu. Faute de temps, ces modifications ont eu lieu par extension de la table ACTEURS. Un premier travail a donc consisté à fusionner les tables d'extension avec la table ACTEURS afin d'en faire une seule et de clarifier la base.

ACTEURS_MAILS

De la même façon, ACTEURS_MAILS a été fusionnée avec ACTEURS.

ADRESSES_MAILS

Cette table était en réalité une erreur d'inattention, car il s'agit clairement d'un doublon de la table ACTEURS_MAILS. Elle a donc été supprimée.

ACTEURS

Cette table représente l'ensemble des acteurs du site AREL. Elle a subit de nombreuses modifications par rapport à l'ancienne base. Outre le fait que des tables sous-jacentes aient été fusionnées avec, il s'avère que le système d'acteurs a bien changé : chaque acteur est en effet associé à un groupe, mais un groupe qui lui correspond. Ses champs sont définis de la façon suivante :

- facteur_id ➧ pf_groupe : c'est non seulement la clé primaire de l'acteur mais désormais c'est également une clé étrangère vers le groupe qui correspond à l'acteur (table GROUPES).
- f_statut : les statuts sont désormais stockés dans une nouvelle table. Ce champ a donc été modifié d'un varchar2 en une clé primaire vers la table STATUTS.
- **login** : login de l'acteur.
- **nom** : nom de l'acteur.
- **prenom** : prénom de l'acteur.
- langue **:** langue choisie par l'acteur.
- email : email de l'acteur.
- **photo** : photo de l'acteur (stockée sur le serveur de l'école).

GESTION DES ETUDIANTS

GS_ETUDIANTS

Cette table fonctionnait avec l'ancien logiciel de gestion de la scolarité. Etant donné que celui-ci n'est plus utilisé, cette table n'a plus de raison d'être.

ARCHIVE_ETUDIANTS

Dans l'ancienne base de données, cette table servait à sauvegarder les étudiants. Le système d'archivage a été retiré temporairement, le temps de réfléchir à un système plus efficace. De cette façon, la table ARCHIVE_ETUDIANTS a été supprimée.

ETUDIANTS

La table ETUDIANTS représente l'ensemble des étudiants de la base. Il s'agit désormais d'une simple extension de la table ACTEURS pour déterminer le type d'un acteur. Cette distinction est utilisée par certaines tables qui référencent un acteur, mais de type étudiant.

Or, ce type est déjà défini dans la table GROUPES. En effet un acteur est associé à un groupe, et celui-ci peut-être de type 'APPRENANT'. On peut donc se poser la question de savoir s'il est utile de garder cette table, et s'il ne vaut mieux pas la supprimer et changer les références sur cette table par une référence sur la table GROUPES (avec un trigger vérifiant que le groupe référencé est de type 'APPRENANT').

Elle est donc composée d'un unique champ :

- **pf_id** : id de l'étudiant. C'est une clé étrangère vers ACTEURS.
- fetudiant_promo : la table PROMOS ayant été supprimée, ce champ l'est aussi.
- fetudiant_annee : l'année de l'étudiant est retrouvée par l'intermédiaire du programme auquel il est inscrit.
- fetudiant_cycle : le cycle de l'étudiant est retrouvé par l'intermédiaire de l'année.
- fetudiant_filiere : la table FILIERES ayant été supprimée, ce champ l'est aussi.
- fetudiant_option : la table OPTIONS ayant été supprimée, ce champ l'est aussi.
- etudiant_scolarite_ant : un étudiant étant inscrit à des programmes, ses antécédents de scolarité peuvent être retrouvés par cette intermédiaire.
- etudiant_fonction : la fonction de l'étudiant est désormais renseignée dans un groupe.
- etu_gesscol_promo : ce champ fonctionnait avec l'ancien logiciel de gestion de la scolarité. Il n'a donc plus de raison d'être.
- etu_gesscol_num_ordre : il en va de même que pour etu_gesscol_promo.

ETUDIANTS_ABSENCES

Cette table représente les absences des étudiants. La majeure modification de cette table réside dans le fait qu'on utilise les créneaux de la table EDT. Du coup, nombre de champs disparaissent pour être remplacés par l'edt. Ses champs sont donc définis de la façon suivante :

- facteur_id ➧ pf_etudiant : id de l'étudiant. C'est actuellement une clé étrangère vers ETUDIANTS, mais il est éventuellement possible de la changer pour une clé vers GROUPES.
- **f_edt** : créneau edt durant lequel l'étudiant était absent. C'est une clé étrangère vers EDT.
- **type_abs** ➧ justification : précise si l'absence est justifiée ('aj') ou non justifiée ('anj').
- ffmat_id : la matière (ou plutôt, la rel maintenant) étant accessible par l'edt, ce champ n'a aucune raison d'être.
- num_semaine : le numéro de la semaine étant accessible par l'edt, ce champ n'a aucune raison d'être.
- annee : l'année étant accessible par l'edt, ce champ n'a aucune raison d'être.
- jour : le jour étant accessible par l'edt, ce champ n'a aucune raison d'être.
- hd : l'heure de début étant accessible par l'edt, ce champ n'a aucune raison d'être.
- hf : l'heure de fin étant accessible par l'edt, ce champ n'a aucune raison d'être.

ABSENCES_JUSTIFIEES

Cette table représente les justifications des étudiants pour leurs absences. Ses champs sont donc définis de la façon suivante :

- **p_id** : id de la justification.
- f_etudiant : étudiant justifiant son absence. C'est désormais une clé étrangère vers la table ETUDIANTS.
- **date_debut** : date de début de l'absence.
- **date_fin** : date de fin de l'absence.
- **motif** : motif de l'absence.

GESTION DES PROFESSEURS

PROFS_GESSCOL

Cette table fonctionnait avec l'ancien logiciel de gestion de la scolarité. Etant donné que celui-ci n'est plus utilisé, cette table n'a plus de raison d'être.

PROFESSEURS_OPTIONS

La table OPTIONS ayant été supprimée, il n'y a plus lieu de garder cette table.

PROFESSEURS_FILIERES

La table FILIERES ayant été supprimée, il n'y a plus lieu de garder cette table.

PROFESSEURS_DEP

Les départements ne sont désormais plus associés à un acteur mais à un programme. Si un professeur est associé à un programme, alors il appartient indirectement aux départements du programme en question. Cette table est donc supprimée.

PROFESSEURS

La table PROFESSEURS représente l'ensemble des formateurs de la base. A l'instar de la table ETUDIANTS, il s'agit désormais d'une simple extension de la table ACTEURS pour déterminer le type d'un acteur. Cette distinction est utilisée par certaines tables qui référencent un acteur, mais de type formateur.

Or, ce type est déjà défini dans la table GROUPES. En effet un acteur est associé à un groupe, et celui-ci peut-être de type 'FORMATEUR'. On peut donc se poser la question de savoir s'il est utile de garder cette table, et s'il ne vaut mieux pas la supprimer et changer les références sur cette table par une référence sur la table GROUPES (avec un trigger vérifiant que le groupe référencé est de type 'FORMATEUR').

Elle est donc composée d'un unique champ :

- **pf_id** : id du professeur. C'est aussi une clé étrangère vers la table ACTEURS.
- professeur_responsabilite : La responsabilité du professeur est désormais retrouvable dans la table DROITS_ACTEURS.

GESTION DES PARENTS

PARENTS

Cette table représente les parents des étudiants. Ses champs sont définis de la façon suivante :

- **pf_id** : id du parent. C'est une clé étrangère vers la table ACTEURS.
- **password** : mot de passe du parent.
- **civilite** : civilité du parent. On pourrait y ajouter une contraint d'unicité.

PARENTS_ETUDIANTS

Cette table représente les relations entre les parents et les étudiants. Ses champs sont définis de la façon suivante :

- **pf_parent** : parent de la relation. C'est une clé étrangère vers la table PARENTS.
- **pf_etudiant** : étudiant de la relation. C'est une clé étrangère vers la table ETUDIANTS, mais il est éventuellement possible de la changer pour une clé vers GROUPES.

GESTION DES DROITS DES ACTEURS

CORRECTIONS

Les corrections entrent dans les types de droits sur les rels. C'est pourquoi cette table a été supprimée, étant donné qu'il existe déjà DROITS_ACTEURS.

DROITS_ACTEURS

Anciennement PROFESSEURS_RELS, cette table sert à définir les droits d'un acteur sur une rel. Ses champs sont définis de la façon suivante :

- facteurs_rels_prof_id ➡ pf_acteur : acteur qui possède le droit sur la rel. C'est maintenant une clé étrangère vers GROUPES et plus PROFESSEURS, car les étudiants peuvent aussi avoir une responsabilité sur un rel (correcteur pour un travail par exemple).
- **pf_rel** : rel sur lequel le droit existe. C'est une clé étrangère vers RELS.
- **f_fonction** : droit de l'acteur sur la rel. C'est une clé étrangère vers la nouvelle table FONCTIONS.

FONCTIONS

Cette table a été créée pour permettre de stocker les fonctions que peuvent avoir les acteurs pour les rels. Ses champs sont définis de la façon suivante :

- p_id : id de la fonction.
- libelle : libellé de la fonction.

NB : Afin de faire la correspondance avec l'ancienne base, cette table doit être au minimum remplie de tous les droits sur les rels présents dans l'ancienne base. De plus, les types suivants devront être directement ajoutés : 'CORRECTEUR', 'ENSEIGNANT', 'RESPONSABLE'.

GESTION DU MENU

Le menu du site d'AREL permet de naviguer sur l'ensemble du site et n'est pas le même selon les acteurs connectés. Chaque objet du menu est ainsi considéré comme une action, ces actions étant stockées dans la base. Chaque acteur de la base possède un statut, et en fonction de son statut il lui est possible d'avoir accès à telles actions.

Important : Il est à noter que plusieurs questions demeurent quant à la gestion des menus. En effet, le lien entre les actions et les menus est assez flou. Il semble que les actions représentent les actions possibles dans le menu de gauche, et que le menu représente le menu de navigation en haut. Cependant, on peut se demander pourquoi un menu possède une action, ce qui reste inexpliqué pour l'instant.

ACTIONS

La table action définit les actions associées à un objet du menu, et de la même façon établit un lien entre les menus et les statuts. Ses champs sont définis de la façon suivante :

- **p_id** : l'id de l'action
- **action** : ce champ fait partie de ceux qui n'ont pas été identifiés.
- **icone** : l'icône de l'action, qui est stockée directement sur le serveur. On peut se demander s'il ne vaudrait pas mieux passer ce champ en CLOB.
- **libelle_court** : nom raccourci de l'action.
- **libelle_long** : nom complet de l'action.
- **rubrique** ➧ regroupement : groupe de l'action. On peut se demander s'il ne vaudrait pas mieux changer ce champ en une clé étrangère vers une nouvelle table.

ACTIONS_STATUTS

Cette table fait le lien entre les actions et les statuts. Ses champs sont définis de la façon suivante :

- pf_action : C'est désormais une clé étrangère vers la table ACTIONS.
- pf_statut : C'est désormais une clé étrangère vers la table STATUTS.

STATUTS

Cette table représente les statuts des acteurs, qui permettent de faire le lien entre des acteurs et des actions. Ses champs sont définis de la façon suivante :

- p_id : id du statut.
- libelle : libellé du statut.

MENUS

Cette table représente l'ensemble des menus présents sur le site. Un menu peut être composé de sous-menus. Dans la dernière version du script proposé aux étudiants, le menu avait obligatoirement un sous menu, ce qui n'est pas vrai. Ainsi, le champ du sous menu doit passer en not null. Ses champs sont définis de la façon suivante :

- **p_id** : id du menu
- f_menu_pere : menu père de l'objet menu courant. C'est une clé étrangère vers la table MENUS. Ce champ a été changé après les soutenances, il était auparavant nommé f_sous_menu, mais dans ce sens il n'était pas possible de construire une arborescence
- f_action : action associée à cet objet menu. C'est désormais une clé étrangère vers la table ACTIONS.
- **icone** : icône du menu.
- **libelle_court** : nom raccourci de l'objet menu.
- **libelle_long** : nom complet de l'objet menu.
- lien_cible : lien vers lequel l'objet menu pointe.

MENUS_ACTEURS

Cette table sert à faire le lien entre les menus et les acteurs. Plus exactement, entre les menus et les statuts, puisqu'un acteur possède un statut. Or, on peut se demander si dans ce cas il ne serait pas plus judicieux d'appeler cette table MENUS_STATUTS. Ses champs sont définis de la façon suivante :

- pf_menu : C'est désormais une clé étrangère vers la table MENUS.
- pf_statut : C'est désormais une clé étrangère vers la table STATUTS.

GESTION DES PARCOURS

Avant d'aborder la notion de parcours, il est important de connaître celle de semainier. Chaque semainier comporte une date de début et une date de fin. A partir de ces dates, on construit, grâce à un logiciel de scolarité, les semaines (qui peuvent être fériées !). Ces deux entités (semaine et semainier) sont donc stockées en base de données.

Les parcours quant à eux, permettent de définir une liste d'activités ordonnée, et ce de manière intemporelle. Ainsi, un parcours reste le même quelque soit l'année de mise en pratique. A cela, on ajoute la possibilité d'instancier ce parcours, sous la forme d'un parcours daté. Il s'agit du même parcours, mais pour un semainier donné. Pour chacune des activités du parcours daté, on affecte donc également une semaine de début et une semaine de fin (qui sera la plupart du temps la même).

Plusieurs contraintes apparaissent alors : il est bien entendu évident que les semaines de chaque activité doivent respecter l'ordre dans lequel elles sont définies dans le parcours associé au parcours daté. De même, chacune des semaines des activités doit appartenir au semainier du parcours daté associé à ces activités. Enfin, chaque activité est associée à une matière, et chaque matière appartient à plusieurs programmes, qui sont instanciés sous forme de sessions. Il est alors important de vérifier que le semainier du parcours daté corresponde au semainier de la session du programme dans lequel on souhaite présenter le parcours daté.

Il résulte de l'unique solution trouvée un énorme maillage de tables, demandant nombre de triggers pour vérifier les contraintes avant ajout/modification… Certaines zones de cette partie restent même incertaines quant à la possibilité d'un réel contrôle sur les données. Ainsi, il apparaît évident que la structure de cette partie de la base est à retravailler. Mais il faut alors tenir compte de plusieurs autres structures de la base, à savoir notamment les programmes et les activités.

PAR_TYPES

Les types des tables descendant de rels sont désormais stockés dans la table RELS dans le champ sous-type. Cette table n'a donc plus de raison d'être.

SEMAINIERS

Cette table représente l'ensemble des semainiers de la base. Un semainier est en fait une entité composée d'une date de début et d'une date de fin, à laquelle sont automatiquement allouées des semaines remplissant ce semainier. Ses champs sont définis de la façon suivante :

- p_id : id du semainier.
- nom : nom du semainier permettant de le distinguer.
- date_debut : date de début du semainier.
- date_fin : date de fin du semainier.

SEMAINES

Cette table représente l'ensemble des semaines contenues dans des semainiers. Ses champs sont définis de la façon suivante :

- p_id : id de la semaine.
- f_semainier : semainier auquel appartient la semaine. C'est une clé étrangère vers la table SEMAINIERS.
- numero : numéro de la semaine dans l'année (1 à 52).
- date_debut : date de début de la semaine.
- date_fin : date de fin de la semaine.
- annee : année civile de la semaine.
- feriee : précise si la semaine est fériée ou pas.

PARCOURS

Cette table représente les parcours. Un parcours est un ensemble d'activités ordonnées, aussi elle n'est composée que d'un unique champ, son id :

- **pf_id** : id du parcours. C'est aussi une clé étrangère vers la table RELS.
- fmat_id : un parcours étant composé d'activités, on peut par leur intermédiaire retrouver la matière associée à ce parcours daté.
- activites_parcours : les activités du parcours sont désormais reliées à lui par une table d'association.
- date_maj : la date de mise à jour est déjà disponible au sein de la table RELS.
- fparcours_type_id : les types de parcours sont désormais renseignés dans le champ sous-type de la table RELS.
- parcours_etat_redaction : ce champ est désormais disponible au sein de la table RELS.

PARCOURS_ACTIVITES

Cette table fait son apparition suite à la suppression du xmltype activites_parcours de la table PARCOURS. C'est donc une table d'association entre les parcours et les activités. Ses champs sont donc définis de la façon suivante :

- pf_parcours : parcours dans la relation. C'est une clé étrangère vers la table PARCOURS.
- pf_activite : activité dans la relation. C'est une clé étrangère vers la table ACTIVITES.
- rang : ordre dans lequel l'activité apparaît dans le parcours.

PARCOURS_DATES

Cette table représente toutes les instanciations d'un parcours sur une période (semainier) donnée. Un parcours daté est indirectement lié à la session du programme dont sont tirées ses activités, puisque son semainier doit être inclus dans cette session. Ses champs sont définis de la façon suivante :

- pf_id : id du parcours daté. C'est aussi une clé étrangère vers la table RELS.
- f_parcours : parcours intemporel dont le parcours daté est l'instance. C'est une clé étrangère vers la table PARCOURS.
- f_semainier : semainier associé à ce parcours daté. C'est une clé étrangère vers la table SEMAINIERS.

PARCOURS_DATES_ACTIVITES

Cette table représente les associations entre les parcours datés et les activités. Ses champs sont définis de la façon suivante :

- pf_parcours_date : parcours daté de la relation. C'est une clé étrangère vers la table PARCOURS_DATES.
- pf_activite : activité de la relation. C'est une clé étrangère vers la table ACTIVITES.
- f_semaine_debut : semaine de début de l'activité dans le parcours daté. C'est une clé étrangère vers la table SEMAINES.
- f_semaine_fin : semaine de fin de l'activité dans le parcours daté. C'est une clé étrangère vers la table SEMAINES.

GESTION DES FORMATIONS

La gestion des formations représente sans aucun doute le cœur et le noyau d'AREL. En effet, il s'agit bien d'un site créé par les étudiants pour les étudiants. Il incorpore donc une gestion interne des formations, permettant de définir des programmes, des matières avec des activités, des parcours d'activités… Tous les types de formation ont besoin d'être définis, mais les parcours étant la grande nouveauté d'AREL, ils sont présentés dans une rubrique consacrée.

GESTION DES ANNEES, CYCLES

Le découpage des promos étant nécessaire, il a bien fallut représenter les différentes années du cursus de l'EISTI. C'est à cela que servent les tables années et cycles. Nombre de tables permettaient également des représentations de promotions, filières ou options, mais ces informations étant récupérables autrement, elles ont été supprimées.

PROMOS

Une promotion étant déterminée soit par l'année (ANNEES), soit par un groupe de type 'PROMO' (GROUPES), cette table n'a plus de raison d'être.

OPTIONS

Une option s'apparentant à un programme (voir PROGRAMMES), cette table n'a pas de raison d'être.

FILIERE

Une filière s'apparentant à un programme (voir PROGRAMMES), cette table n'a pas de raison d'être.

CYCLES

Cette table permet de représenter l'ensemble des cycles du cursus à l'EISTI. Un cycle est en réalité un ensemble d'années adjacentes. Ses champs sont définis de la façon suivante :

- **p_id** : id du cycle.
- **libelle** : nom du cycle ('ING', 'CPI').

ANNEES

Cette table permet de représenter l'ensemble des années du cursus à l'EISTI. Chaque année appartenant à un cycle, ses champs sont donc définis de la façon suivante :

- **p_id** : id de l'année.
- **f_cycle** : cycle auquel l'année appartient. C'est une clé étrangère vers la table CYCLES.
- **libelle** : nom de l'année ('CPI1', 'CPI2', 'ING1', 'ING2', 'ING3').

GESTION DES MATIERES

On entend bien sûr par matières toutes celles enseignées à l'EISTI. Une matière est un certain enseignement appartenant à un département (si l'on prend l'exemple de l'EISTI, les matières MDA, Programmation Système ou encore Réseaux font parties du département informatique). Une matière est bien sur composée d'activités, et étant une rel, contient par l'intermédiaire de la table RELS_RELATIONS des rels.

MATIERES_ETUDIANTS

Cette table servait à représenter les relations entre matières et étudiants. Elle est en fait inutile de par le fait qu'il existe déjà un lien entre les étudiants et les matières par l'intermédiaire des programmes.

MATIERES_GESSCOL

Cette table fonctionnait avec l'ancien logiciel de gestion de la scolarité. Etant donné que celui-ci n'est plus utilisé, cette table n'a plus de raison d'être.

MATIERES

Cette table représente donc les enseignements proposés à l'EISTI. Ses champs sont définis de la façon suivante :

- **pf_id** : id de la matière. C'est aussi une clé étrangère vers la table RELS.
- mat_semestre : la table SEMESTRES n'existant plus, ce champ n'a plus de raison d'être.
- mat_date_debut : on a supprimé les dates de la matière, car c'est un objet intemporel.
- mat_date_fin : on a supprimé les dates de la matière, car c'est un objet intemporel.
- mat_libelle_court : ce champ est déjà présent dans la table RELS.
- f_departement : département auquel appartient la matière. C'est une clé étrangère vers la table DEPARTEMENTS.
- **presentation** : brève présentation de la matière.

DEPARTEMENTS

Cette table représente les départements de l'EISTI, qui sont en fait des regroupements de matières. Chaque département possède un groupe de formateurs qui sont responsables de ce département et des matières enseignées dans ce département. Ses champs sont définis de la façon suivante :

- **p_id** : id du département.
- **f_groupe** : responsables du département. C'est une clé étrangère vers la table GROUPES.
- dep_f_filiere : la table FILIERES n'existant plus, ce champ n'as plus de raison d'être.
- **libelle** : nom complet du département.
- **court** : nom raccourci du département.

GESTION DES PROGRAMMES

Les programmes sont un ensemble de matières, qui permettent en quelque sorte de définir une filière. Par exemple, les programmes d'ING2 GI ou GM sont 2 programmes différents. De même, le programme ING2 tronc commun en est un 3eme.

Un programme est par définition intemporel. Il reste effectivement le même quelque soit l'année civile en cours (du moins, il peut être modifié mais cela reste inhabituel). Un apprenant n'est donc pas associé à un programme, mais à une session de ce programme. On dira alors qu'il est inscrit à cette session. De plus, un programme peut nécessiter des pré-requis, qu'il faut avoir assimilé avant de pouvoir entamer le programme.

TYPE_PROGRAMMES

Les types des tables descendant de rels sont désormais stockés dans la table RELS dans le champ sous-type. Cette table n'a donc plus de raison d'être.

PROGRAMMES

Cette table représente donc les programmes, qui sont un ensemble de matières. Ses champs sont définis de la façon suivante :

- **p_id** : id du programme.
- f_annee : année pour laquelle ce programme est disponible. C'est désormais une clé étrangère vers la table ANNEES.
- **objectif** : but du programme.
- **descriptif** : description de programme.
- **nbchoix** : ce champ fait partie de ceux qui n'ont pas été identifiés : il semble que cela soit le nombre de choix pour les programmes suivant celui-ci, mais rien n'est sur.
- **url** : ce champ fait partie de ceux qui n'ont pas été identifiés : on ne sait pas pourquoi une url est associée à un programme.
- type : un programme n'a pas de type particulier, c'est pourquoi ce champ a été supprimé.
- id_fcycle : le cycle étant récupérable par l'année, ce champ est inutile.

PREREQUIS_PROGRAMMES

Cette table modélise les pré-requis que peuvent nécessiter certains programmes. Ses champs sont définis de la façon suivante :

- pf_pere : pré-requis nécessaire au programme. C'est désormais une clé étrangère vers la table PROGRAMMES.
- pf_fils : programme qui nécessite le pré-requis. C'est désormais une clé étrangère vers la table PROGRAMMES.

PROGRAMMES_MATIERES

Cette table représente les appartenances des matières à des programmes. Chaque programme contient plusieurs matières, mais il faut savoir qu'une matière peut également apparaître dans plusieurs programmes. Ses champs sont donc définis de la façon suivante :

- **pf_programme** : programme contenant la matière. C'est une clé étrangère vers la table PROGRAMMES.
- pf_matiere : matière contenue dans le programme. C'est désormais une clé étrangère vers la table MATIERES.

PROGRAMMES_SESSIONS

Cette table représente les instances temporelles d'un programme. En effet, en soit un programme est une entité intemporelle, qui reste la même au fil du temps. Pour qu'un apprenant puisse suivre la formation d'un programme, il faut donc qu'il s'inscrive à ce qu'on appelle une session de ce programme, une session n'étant rien d'autre que l'attribution d'un semainier à un programme. Ses champs sont donc définis de la façon suivante :

- **p_id** : id de la session.
- f_programme : programme auquel on associe une entité temporelle, soit un semainier ici. C'est désormais une clé étrangère vers la table PROGRAMMES.
- f_semainier : semainier attribué au programme. C'est une clé étrangère vers la table SEMAINIERS.
- date_debut : ce champ a été supprimé pour être remplacé par le semainier.
- date_fin : ce champ a été supprimé pour être remplacé par le semainier.

INSCRIPTIONS_SESSIONS

Cette table représente l'inscription des apprenants à des sessions de programme. Ses champs sont définis de la façon suivante :

- **pf_session** : session à laquelle l'apprenant s'inscrit. C'est une clé étrangère vers la table PROGRAMMES_SESSIONS.
- fetudiant_id ➡ pf_groupe : apprenant souhaitant s'inscrire à la session. C'est désormais une clé étrangère vers la table GROUPES (voir gestion des groupes).
- **valide** : précise si l'inscription est valide ou non. On peut se demander l'utilité d'un tel champ étant donné qu'une inscription non valide n'a aucune raison d'être stockée en base.

CHOIX_PROGRAMMES

Cette table représente les choix des élèves sur le programme qu'ils veulent suivre par la suite. Ses champs sont définis de la façon suivante :

- pf_groupe : apprenant(s) qui souhaite(nt) suivre le programme donné. C'est désormais une clé étrangère vers la table GROUPES.
- pf_programme : programme choisi. C'est désormais une clé étrangère vers la table PROGRAMMES.

- **p_ordre** : rang avec lequel le programme est choisi.

GESTION DES ACTIVITES

Les activités modélisent toute activité proposée aux étudiants par les formateurs. Cela va donc des TP aux TD, en passant par les projets de pôles. Ces différentes activités sont liées à une unique matière, et peuvent être organisées à l'intérieur de parcours et de parcours datés.

ACTIVITES_TYPES

Les types des tables descendant de rels sont désormais stockés dans la table RELS dans le champ sous-type. Cette table n'a donc plus de raison d'être.

ACTIVITES_PREREQUIS

Cette table représente les pré-requis que peuvent nécessiter certaines activités. En effet, il peut être nécessaire de maîtriser une certaine rel avant de pouvoir entamer une activité. Ses champs sont donc définis de la façon suivante :

- **pf_activite** : activité nécessitant le pré-requis. C'est une clé étrangère vers la table ACTIVITES.
- **pf_prerequis** : rel dont la maîtrise est nécessaire pour pouvoir commencer l'activité. C'est une clé étrangère vers la table RELS.

ACTIVITES

Cette table représente les activités proposées par les formateurs aux étudiants. Elle permet de représenter entre autres les TD, TP, projets de pôle. Ses champs sont définis de la façon suivante :

- **pf_id** : id de l'activité.
- **f_matiere** : matière pour laquelle cette activité est proposée. C'est une clé étrangère vers la table MATIERES.
- factivite_type_id : le type de l'activité est désormais dans la table RELS.
- **organisation_travail** : précise si l'activité est adaptée à un travail individuel, en groupe ou par promo.
- **description** : description de l'activité en elle-même.
- **mode_evaluation** : précise si l'activité est évaluée ou pas.
- **mot_cle** : on peut se poser la question de l'utilité de ce champ, étant donné que la table RELS comporte un champ indexation.
- **etat_redaction** : précise l'état de rédaction de l'activité. On peut supposer qu'une contrainte de check doive être ajoutée.

GESTION DES DOCUMENTS

Les documents d'AREL se présentent sous plusieurs formes : on retrouve en effet non seulement des fichiers normaux, mais également des documents xml, des fichiers de logs ou encore des articles sous forme de xmltype. Les documents xml occupent une place importante dans l'architecture d'AREL. Il existe en effet plusieurs xsl dans la base pour un même xml, qui selon l'utilisateur vont permettre d'afficher ou cacher certaines parties des données du xml.

ARTICLES

Cette table représente les articles qui sont postés sur AREL. Ses champs sont définis de la façon suivante :

- pf_id : id de l'article. C'est désormais également une clé étrangère vers la table RELS.
- **contenu** : contenu de l'article sous forme de xmltype.

FICHIERS

Cette table représente les fichiers simples d'AREL. Ses champs sont donc définis de la façon suivante :

- **pf_id** : id du fichier. C'est aussi une clé étrangère sur la table RELS.
- **filename** : nom du fichier.

LOGS

Cette table représente les fichiers de log d'AREL. Ils permettent un traçage des actions effectuées sur le site, mais c'est la table la plus floue de la base. Ses champs sont définis de la façon suivante :

- **p_id** : id du log.
- f_acteur → f_groupe : acteur concerné par le log. C'est désormais une clé étrangère vers la table GROUPES.
- **f_rel** : rel concernée par le log. C'est une clé étrangère vers la table RELS.
- **adresse_ip** : adresse ip de l'utilisateur qui a déclenché le log.
- **log_date** : date à laquelle le log s'est déclenché.
- **temps_connexion** : temps de connexion sur la rel avant déclenchement du log.
- **url** : ce champ fait partie de ceux qui n'ont pas été identifiés.
- **parametre** : ce champ fait partie de ceux qui n'ont pas été identifiés.
- **log_exception** : précise le type d'exception rencontrée.
- **message_erreur** : message affiché en cas d'erreur.

DOC_TYPES

Cette table ayant fusionnée avec RELS, il est inutile de la garder.

DOCUMENTS

Cette table représente les documents de type xml postés sur AREL. Chaque document de ce type doit posséder au moins une transformation xml. Ses champs sont donc définis par :

- **pf_id** : id du document. C'est aussi une clé étrangère vers la table RELS.
- **f_type_xml** : précise le type de document xml dont il s'agit. C'est une clé étrangère vers la table XML_TYPES.
- fdoc_type_id : le type du document est désormais renseigné dans le champ sous-type de RELS.
- md5 : cryptage md5 du fichier.
- **content** : décrit brièvement le contenu du fichier.
- doc_auteur : possibilité de le renseigner dans la table DROITS_RELS.
- doc_content_bin : champ inutile.
- **filename** : nom du fichier.
- **mot_cle** : ce champ fait partie de ceux qui n'ont pas été identifiés : sa présence est d'autant plus étrange qu'un xmltype indexation se trouve dans RELS. Il est donc à priori à supprimer.
- **concepts** : ce champ fait partie de ceux qui n'ont pas été identifiés.
- **etat_redaction** : précise l'état de rédaction dans lequel se trouve ce document. Les valeurs que ce champ peut prendre n'ont pas été identifiées.

XML_TYPES

Cette table représente un type de document xml sous la forme d'une DTD à vérifier et un XSL pour l'affichage. Ses champs sont définis de la façon suivante :

- **p_id** : id du type xml.
- **libelle** : nom du type xml.
- **dtd** : DTD du type de document.
- **xsl** : xsl du type de document.

GESTION DES COMMUNICATIONS

On distingue essentiellement deux types de communication sur AREL : la première est une messagerie, et la deuxième un système de questions/réponses. Ces deux outils sont désormais en mesure de subir une révolution : en effet la modification majeure des acteurs et des groupes permet d'envoyer un mail à plusieurs expéditeurs.

MESSAGES

Cette table représente l'ensemble des messages échangés sur AREL via la messagerie. Un message peut posséder un autre message, qui est sa réponse. On constitue ainsi l'arborescence des messages. Ses champs sont définis de la façon suivante :

- **p_id** : id du message.
- **f_rel** : rel concernée par le message. C'est une clé étrangère vers la table RELS.
- **f_exped** : expéditeur(s) du message. C'est une clé étrangère vers la table GROUPES.
- **f_dest** : destinataire(s) du message. C'est une clé étrangère vers la table GROUPES.
- f_reponse : réponse éventuelle au message. C'est désormais une clé étrangère vers la table REPONSES.
- **objet** : objet du message.
- **content** : Corps du message.
- **date_exped** : date d'envoi du message.
- **etat** : détermine si le message a été lu ou pas encore.

QUESTIONS

Cette table représente les questions que peuvent poser les apprenants aux formateurs par rapport à une rel. Ses champs sont définis de la façon suivante :

- **p_id** : id de la question
- **f_rel** : rel concernée par la question. C'est une clé étrangère vers la table RELS.
- fqst_auteur ➡ f_groupe : expéditeur de la question. C'est maintenant une clé étrangère vers la table GROUPES.
- **qst_date** : date à laquelle a été posée la question.
- **libelle** : objet de la question.
- **contenu** : contenu de la question.
- **nb_reponses** : nombre de réponses à la question. On peut se demander l'utilité de ce champ.

QUESTIONS_SUIVIS

Cette table représente quelles questions sont suivies par quels acteurs. Ses champs sont définis de la façon suivante :

- **pf_question** : question suivie par l'acteur. C'est une clé étrangère vers la table QUESTIONS.
- **pf_groupe** : acteur qui suit la question. C'est une clé étrangère vers la table GROUPES.

REPONSES

Cette table représente les réponses aux questions posées sur AREL. Ses champs sont définis de la façon suivante :

- **p_id** : id de la réponse.
- **f_question** : question à laquelle répond cette réponse. C'est une clé étrangère vers la table QUESTIONS.
- frep_auteur → f_groupe : auteur de la réponse. C'est une clé étrangère vers la table GROUPES.
- **contenu** : contenu de la réponse.
- **rep_date** : date à laquelle la réponse a été postée.

GESTION DES RENDUS

Il existe deux types de rendus sur AREL. Les rendus actuellement connus de tous, qui concernent les travaux demandés lors des activités, sont les premiers. Mais il existe également une autre entité de rendu, qui est en fait une fonctionnalité à venir, et qui n'est rien d'autre qu'un système de QCMS.

GESTION DES QCMS

Ainsi donc, il sera bientôt possible de répondre à des QCMS sur AREL. Pour rappel, un QCM est un ensemble de questions à choix multiples. Ceux-ci pourront être générés aléatoirement sur AREL, et certains d'entre eux pourront être générés par les formateurs pour évaluer leurs étudiants.

QCMS

Cette table sert à représenter une future fonctionnalité disponible sur AREL. Il sera en effet bientôt possible de répondre à des QCMS en ligne, qu'ils soient mis en place pour des examens ou bien généré aléatoirement pour des révisions. Ses champs sont définis de la façon suivante :

- **pf_id** : id du QCM. C'est aussi une clé étrangère vers la table RELS.
- f_rel : rel associée au QCM. C'est une clé étrangère vers la table RELS.
- f_visibilite : visibilité du QCM. C'est une clé étrangère vers la table VISIBILITES.
- **nb_questions** : nombre de questions présentes dans le QCM.
- note : la note étant propre à l'élève, il est absurde que celle-ci se trouve ici.
- reponse_bonne : le nombre de bonnes réponses est retrouvable par une requête sur la table QCMS_REPONSES.
- reponse_partielle : il en est de même pour le nombre de réponses partielles.
- reponse_aucune : il en est de même pour le nombre de questions sans réponses.
- reponse_mauvaise : il en est de même pour le nombre de réponses mauvaises.

QCMS_REPONSES

Cette table représente les réponses possibles aux questions d'un QCM. Une question de QCM peut avoir un nombre indéfini de réponses, en revanche une réponse n'appartient qu'à une question. Ses champs sont définis de la façon suivante :

- **pf_id** : id de la réponse. C'est aussi une clé étrangère vers la table RELS.
- f_question : question à laquelle est attachée la réponse. C'est une clé étrangère vers la table QCMS_QUESTIONS.
- **reponse_correcte** → valide : précise si la réponse est bonne ou pas.
- **libelle** : texte de la réponse.
- **contenu** → infos : informations complémentaires sur la réponse.

QCMS_QUESTIONS

Cette table permet de représenter toutes les questions présentes dans les QCMS. Ces questions peuvent n'appartenir à aucun QCM et exister dans le but de générer des QCMS dynamiquement. Ses champs sont définis de la façon suivante :

- pf_id : id de la question. C'est aussi une clé étrangère vers la table RELS.
- freponse_id : ce champ a été supprimé, du fait qu'il est absurde qu'une question de QCM n'ait qu'une seule réponse possible.
- f_type : type de la question. C'est une clé étrangère vers la table QUESTIONS_TYPES.
- f_visibilite : visibilité de la question. C'est une clé étrangère vers la table VISIBILITES.
- **contenu** ➤ libelle : texte de la question.
- infos : informations complémentaire pour la question.

QUESTIONS_TYPES

Cette table représente les différents types de question que l'on peut retrouver en QCM. Il n'a pas encore été précisé de quelle façon elle allait être remplie. Ses champs sont définis de la façon suivante :

- **p_id** : id du type de question.
- **libelle** : nom du type de question.

QCMS_RENDUS

Cette table représente les rendus de QCMS que peuvent déposer les apprenants pour des examens QCMS. Chaque entrée de la table associe en réalité un acteur de type apprenant à une réponse, une question et un qcm. Le système n'est donc clairement pas optimisé, puisqu'on aura autant d'entrées que de réponses sélectionnées par chaque apprenant pour chaque question de chaque qcm… Autant dire beaucoup. Il faudrait donc réfléchir à un système de gestion de rendu de QCMS plus efficace.

De plus, la clé primaire était composite sur 5 champs. Afin d'harmoniser la base, cette clé a été modifiée, et un id a été implanté dans la table. Ses champs sont définis de la façon suivante :

- p_id : id du rendu.
- f_qcm : QCM pour lequel il y a rendu. C'est une clé étrangère vers la table QCMS.
- facteur_id ➤ f_groupe : groupe (de type apprenant) qui représente l'acteur répondant au QCM. C'est désormais une clé étrangère vers la table GROUPES.
- **f_question** : question à laquelle l'apprenant répond. C'est une clé étrangère vers la table QCMS_QUESTIONS.
- f_reponse : l'une des réponses choisies (s'il y en a plusieurs) par l'étudiant pour la question donnée. C'est désormais une clé étrangère vers la table QCMS_REPONSES.
- **reponse_ouverte** : si la question est de type réponse ouverte, alors la réponse de l'étudiant est stockée dans ce champ.

GESTION DES TRAVAUX

Les travaux sur AREL représentent plus exactement des travaux d'activités. En effet, une activité peut donner lieu à un travail qu'il conviendra de rendre dans les délais imposés. Ainsi les travaux sont-ils assignés à une activité. Un travail possède également un nombre maximum et minimum d'étudiants pouvant y participer ensemble.

RENDU

N'ayant pas trouvé d'utilité propre à cette table et compte tenu de sa similarité avec TRAVAUX, il semble que cette table en soit en fait un doublon. C'est pourquoi elle a été supprimée.

ARCHIVES_TRAVAUX

Dans l'ancienne base de données, cette table servait à archiver les travaux. Le système d'archivage a été retiré temporairement, le temps de réfléchir à un système plus efficace. De cette façon, la table ARCHIVES_TRAVAUX a été supprimée.

TRAVAUX

Cette table représente l'ensemble des travaux qui peuvent être demandés aux apprenants en corrélation avec les activités sur lesquelles ils sont formés. Les travaux sont effectués individuellement ou en groupe. Il faut également savoir qu'un travail est bien entendu une rel. Ses champs sont donc définis de la façon suivante :

- **pf_id** : id du travail. C'est également une clé étrangère vers la table RELS.
- ftravaux_matiere_id : une activité étant liée à une matière, il est possible de récupérer cette dernière par l'intermédiaire de l'activité.
- ftravaux_activite_id : ce champ n'a plus d'utilité, du fait de l'existence de la table RELS_RELATIONS qui permet de relier deux rels.
- date_debut : date à laquelle le travail est assigné aux apprenants.
- **date_limite** : date limite pour rendre ce travail.
- nombre_min : nombre minimum d'apprenants par groupe de travail.
- nombre_max : nombre maximum d'apprenants par groupe de travail.
- **libelle** : nom exact du travail.
- **description** : description précise du travail.

TRAVAUX_ETUDIANTS

Cette table représente la liaison entre les travaux et les apprenants. C'est par son intermédiaire que les travaux sont assignés aux apprenants, et que le rendu de chacun est contrôlé. Ses champs sont définis de la façon suivante :

- **fetudiant_travaux_id** ➤ pf_groupe : apprenant ou groupe d'apprenants concerné par le travail. C'est une clé étrangère vers la table GROUPES.
- **pf_travaux** : travail à rendre par le groupe d'apprenants. C'est une clé étrangère vers la table TRAVAUX.
- f_visibilite : visibilité du rendu. C'est désormais une clé étrangère vers la table VISIBILITES.
- **rendu** : précise si le travail a été rendu par le groupe ou pas.
- **date_rendu** : précise la date du rendu s'il a été effectué.
- **retard** : précise si le travail a été rendu en retard ou pas.
- **penalite** : précise la pénalité affectée au groupe pour le retard.
- **corrige** : précise si le rendu a été corrigé.
- correction : url vers le fichier de correction du rendu.
- **commentaire** : commentaire éventuel laissé par le correcteur.
- **nomfichier** : nom du fichier du rendu.
- **note** : note attribuée au groupe pour ce travail.
- **contenu** : on peut se demander l'utilité d'un tel champ, étant donné que le contenu du rendu se trouve être un fichier et qu'un champ précise déjà son nom.

DIVERS

ALERTES

Cette table n'est pas encore exploitée et représente une future fonctionnalité d'AREL. Ainsi l'entité qu'elle représente n'existe pas encore, et demeure floue. Ses champs sont définis de la façon suivante :

- **p_id** : id de l'alerte.
- facteur_id ➡ f_groupe : groupe averti par l'alerte. C'est désormais une clé étrangère vers la table GROUPES.
- **f_rel** : rel concernée par l'alerte. C'est une clé étrangère vers la table RELS.
- **action** : action déclenchée par l'alerte.
- **alerte_date** : date à laquelle l'alerte a été levée.

EVTS

Cette table représente l'ensemble des événements de l'EISTI. On peut notamment apercevoir ces dit-événements sur la page d'accueil d'AREL. Ses champs sont définis de la façon suivante :

- **p_id** : id de l'événement.
- **date_debut** : date de début de l'événement.
- **date_fin** : date de fin de l'événement.
- **filename** : nom du fichier associé à l'événement.
- **libelle** : nom de l'événement.
- **url** : url de l'image associée à l'événement.
- **afficher_date** : précise si la date de l'événement doit être affichée ou pas.

VISIBILITES

Cette table représente l'ensemble des visibilités associées à diverses ressources de la base de données d'AREL. Ses champs sont définis de la façon suivante :

- **p_id** : id de la visibilité.
- **libelle** : nom de la visibilité.

Elle sera notamment remplie des valeurs suivantes : 'PUBLIQUE', CACHEE', 'SEMI-PRIVEE', 'PRIVEE'.

DROITS_ABSENCES

Cette table étant inachevée (un unique champ) et n'ayant aucune utilité, elle a été supprimée.

SEMESTRES

La gestion de semestres n'ayant actuellement aucun intérêt sur AREL, cette table a été supprimée.

PS_SEMESTRES

Au vu de la suppression de la table SEMESTRES, cette table n'a plus de raison d'être. Elle a donc été supprimée également.

SITES

Cette table permettait de représenter les sites (Cergy ou Pau). Désormais, les sites sont représentés dans la table groupe, pour un groupe de type site, ce qui permet par la même occasion d'englober tous les groupes d'acteurs concernés dans le groupe d'un site. La table SITES n'a donc plus de raisons d'être.

UE

Cette table représentait les unités d'étude. Son utilité étant inexistante, elle a été supprimée.

CE QU'IL RESTE A FAIRE

Après d'aussi grosses modifications, il n'est pas d'illusions possibles quant à la qualité actuelle de la base de données proposée : certes celle-ci s'est vu corrigée de nombreuses erreurs et autant d'oublis, mais il n'en reste pas moins que le nombre de modifications apportées a sans doute laissé au passage d'autres erreurs et oublis (en bien moindre quantité bien sur !). Ainsi, si la base de données proposée a subie une telle refonte, il est exclu que celle-ci soit parfaite et utilisable directement sans de légères corrections.

De plus, il est un problème qui lui est d'origine plus analytique, puisqu'il nécessite une refonte d'une petite partie de la base : en effet les entités intemporelles qui doivent être instanciées posent un certain problème de maillage des tables qui n'est absolument pas optimisé.

Enfin, de par le temps accordé à cette refonte, il existe actuellement deux entités qui n'aient pas été créées mais sont indispensables au bon fonctionnement de la base : les triggers et les vues.

CORRECTIONS A APPORTER

Certaines erreurs présentes dans la base de données actuelle ont été décelées après les soutenances de ce projet. Afin de permettre la compréhension du code de certains modules, il a été décidé de conserver la base telle qu'elle a été proposée aux étudiants. Cette section fait donc office de récapitulatif des erreurs reconnues de la base proposée, qu'il faudra corriger par la suite.

NOTION D'ACTEUR

Une première erreur qui est née d'une incompréhension se trouve dans la notion d'acteur. En effet, si ceux-ci sont désormais reliés à un groupe, ce groupe ne prend pas pour autant le type 'APPRENANT', 'PARENT', ou encore 'FORMATEUR'. Il n'existe en réalité pour ces trois entités qu'une unique notion : un groupe de type 'ACTEUR'.

La différenciation se fait ensuite de par le statut que l'on trouve dans la table ACTEURS. Ainsi, toute relation décrite dans ce document et liant des tables à des groupes de type 'APPRENANT, 'PARENT' ou 'FORMATEUR' est à remplacer par une relation par un groupe de type 'ACTEUR' associé à un acteur du statut approprié.

ACTIONS

Une autre erreur ayant été repérée concerne les actions dans les menus. Il apparaît en effet illogique qu'une action (représentant en réalité le menu à gauche lorsque l'on est sur AREL) soit associée à un objet menu. Il s'agit là encore d'une incompréhension puisque c'est en fait le lien vers lequel le menu pointe. Et le champ alors nommé lien_cible représenterait en fait une ancre sur la page.

Toutefois, cette explication est un peu floue, puisque selon toute logique l'ancre devrait être associée à une action et non pas à un menu. Ainsi, si la correction à apporter serait de changer f_action de la table MENUS en un lien_cible et de déplacer le lien_cible actuel vers la table ACTIONS, il n'en reste pas moins qu'une réflexion sur cet ensemble s'avère nécessaire.

TYPES_PROGRAMMES

Enfin, la dernière erreur reconnue de la base de données n'est pas des moindres. En effet, il s'agit là de l'erreur à corriger au plus vite, étant donné qu'elle met en péril certaines fonctionnalités de la base. La table TYPES_PROGRAMMES, à l'origine supprimée par doublon avec le type présent dans la table RELS, s'avère en fait être plus précise que ce dernier.

Elle permettait en effet de déterminer un type de programme (par exemple une option, un tronc commun, etc...) et donc d'affecter un nombre de choix possibles quant à la demande d'un apprenant à s'inscrire à un certain type de programmes (2 choix pour les options, 1 seul pour la spécialité d'ING2, etc...).

Il faut donc à l'avenir réintégrer cette table et sa relation avec la table PROGRAMMES.

TRIGGERS ET VUES

L'ensemble des triggers et des vues représente l'essentiel du travail qui reste à fournir sur la nouvelle base de données.

L'intérêt des vues réside dans le fait qu'une modification de la base de données a peu de chances d'affecter l'ensemble des données de la vue. Le déplacement d'une variable d'une table à une autre est par exemple le genre de choses qui est relativement transparent lorsque l'on utilise des vues : si les tables sont modifiées, la vue elle conserve l'ensemble de ses champs et une simple modification de sa requête SELECT suffit. Ainsi le code n'a pas besoin d'être retouché, étant donné que la vue conserve les mêmes noms de champs avec les mêmes données.

L'ensemble des vues à fournir se fait d'un commun accord avec l'ensemble des groupes travaillant les modules d'AREL. Ainsi, il faut analyser le besoin de chaque groupe concernant les données et chaque groupe doit déterminer les vues que son module nécessite.

Concernant les triggers, de nombreuses indications données dans ce document permettront leur bonne mise en place. Cependant, certains d'entres eux apparaissent déjà évidents.

TYPE DE GROUPE ET STATUT

Un premier trigger qui apparait évident est bien entendu celui concernant les types de groupes et les statuts. En effet, dans certaines tables il est précisé une clé étrangère vers la table GROUPES. Seulement les groupes peuvent désormais être de types différents. Or, la plupart du temps, nous souhaitons qu'un unique type de groupe puisse être ainsi référencé. Il est donc important de mettre en place des triggers qui vérifient la bonne insertion dans ces tables. De plus, s'il s'agit d'un groupe de type 'ACTEUR', il faut parfois de la même façon vérifier que le statut de l'acteur en question correspond.

Par exemple, pour la table PARENTS_ETUDIANTS, il faut non seulement vérifier que chacun des groupes de cette table est de type 'ACTEUR', mais également que le statut de l'acteur associé au pf_parent soit 'PARENT' et le type de celui associé au pf_etudiant soit 'APPRENANT'.

INSTANCES D'ENTITES INTEMPORELLES

Les instances d'entités intemporelles représentent l'autre grande famille de triggers à mettre en place. En fait, le maillage actuel de la base concernant ces entités impose un certain nombre de triggers. Si l'on prend l'exemple des parcours datés, ils nécessitent à eux seuls la mise en place d'au moins 4 triggers ! En effet, il faut vérifier les cas suivants :

- les activités associées au parcours daté appartiennent bien au parcours dont il est l'instance.
- les semaines attribuées pour une activité dans l'association avec un parcours daté appartiennent bien au semainier de ce parcours daté.
- les semaines attribuées pour toutes les activités d'un parcours daté sont bien respectent bien l'ordre qu'elles ont dans le parcours.
- le semainier du parcours daté appartient bien au semainier de la session du programme auquel il est attaché.

Ces cas sont donc autant de triggers à mettre en place, à moins bien entendu d'effectuer une refonte de cette partie de la base que sont les entités intemporelles et leurs instances.

LE MOT DE LA FIN

Ce document a donc un objectif double : il donne d'une part un nombre conséquent d'informations sur la base de données AREL, et de par la clarté que nous avons tenté d'y apporter nous espérons qu'il saura aider à une meilleure compréhension de celle-ci.

D'autre part, si ce document n'est donc pas la panacée et ne propose pas une base de données miracle, il s'agit là d'une première étape avec une normalisation et refonte complète qu'il est intéressant de poursuivre. Nous espérons donc qu'il saura également aider les promotions futures à pouvoir modeler la base plus facilement selon les besoins et les évolutions du site AREL.

HIBERNATE

INTRODUCTION

Hibernate est un framework de persistance d'objets en Base de Données. Il fonctionne comme un *middleware*, en s'intercalant entre l'application Java et la BDD. Le principal avantage d'Hibernate se situe dans sa configuration relativement simple, bien qu'un peu lourde, et les nombreuses couches d'abstraction qu'il nous permet de mettre en place.

Dans cette partie, nous allons étudier la mise en place du framework dans le cadre d'AREL, des difficultés rencontrées et des évolutions à prévoir pour l'avenir.

INSTALLATION

LIBRAIRIES ET FICHIERS

Actuellement, toutes les librairies nécessaires au bon fonctionnement d'Hibernate sont déjà installées. Les classes mappées sur le BDD ont également été crées, il y a 3 classes par table:

- AbstractMaClasse.java: les informations de base de la classe (attributs, constructeur, méthodes diverses…)
- MaClasse.java: hérite de AbstractMaClasse.java, permet de rajouter des méthodes tout en gardant un code clair.
- MaClasseDAO.java: classe qui agit comme une fabrique d'objet, en proposant des requêtes préparées en BDD (findAll(), findById(), save(), delete(),…)

Le fichier de configuration global hibernate.cfg.xml est également finalisé, ainsi que chaque fichier de mapping (MaClasse.hbm.xml).

A noter que ces fichiers ont été générés avec l'IDE MyEclipse, version modifiée d'Eclipse, qui est optimisé pour la mise en place de framework, dont Hibernate. Une documentation d'utilisation de MyEclipse est disponible sur http://www.eisti.fr/~vg/Documents/ProjetJEE/BACSWW_Hibernate_Jaxb_Spring/index.html.

UTILISATION

Après installation, l'utilisation d'Hibernate du point de vue du code est très simple. Il suffit d'instancier un objet de type DAO qui s'occupera d'exécuter les requêtes que l'on souhaite, s'occupant au passage de la connexion et de la déconnexion à la BDD. Dans le cadre d'une transaction (delete, insert, update), il faudra cependant penser à effectuer un commit pour valider les modifications en BDD.

Exemple:

```
TypesGroupesDAO dao = new TypesGroupesDAO(); //L'objet de DAO
TypesGroupes groupe = new TypesGroupes(); //L'objet à persister

groupe.setNom("Groupe de test"); //On modifie ses attributs
dao.save(groupe); //On sauvegarde
dao.getSession().beginTransaction().commit(); //On commit
```

PROBLEMES RENCONTRES

Le plus gros souci auquel nous avons été confronté est le manque visible de travail effectué sur le couple Oracle+Hibernate. En effet, ces deux solutions semblent souffrir de lourd problèmes de compatibilité entre eux, notamment au niveau de la configuration de l'autocommit, qui n'est tout simplement pas fonctionnel, de la gestion des tables d'association, qui est vraiment laborieuse par rapport à l'utilisation d'Hibernate avec un SGBD tel que MySQL. On peut donc se poser la question du choix de la technologie pour des applications telles qu'AREL, qui n'ont pas forcément besoin d'exploiter à fond les capacités énormes du monstre qu'est Oracle, alors que MySQL pourrait s'avérer beaucoup plus simple et largement suffisant.

Il faut également noter qu'Hibernate n'est compatible qu'avec des builds très précis d'Oracle XE (dont le fichier d'installation est présent sur AREL).

Enfin, nous avons du effectuer quelques modifications dans les fichiers générés par MyEclipse:

- L'attribut default-lazy="false" au début de chaque fichier de mapping, pour permettre le chargement d'objets en cascade.
- Le remplacement des BigDecimal par des int (Oracle n'utilisant pas des INTEGER mais des NUMBER, MyEclipse ne les reconnaît pas comme des int classiques).
- Les clefs primaires sont en auto-incrémentation.