le cnam

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

PARIS

MEMOIRE

présenté en vue d'obtenir

le DIPLOME D'INGENIEUR CNAM

SPECIALITE : Informatique

OPTION : Systèmes d'information

par

Stéphane Arenas

Mise en place d'un suivi dynamique de base de données :

DaPCol

Soutenu le 3 décembre 2012

ST Groupe
INGÉNIERIE
CONSULTING

sacem

JURY

PRESIDENT : Elisabeth Métais

MEMBRES : Cédric du Mouza
Nicolas Travers
Robert Vesoul
Franck Livernette

Remerciements

Je tiens, dans un premier temps, à adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de ce cursus au Cnam.

Je tiens à remercier sincèrement Monsieur Nicolas Travers, assistant professeur au Cnam qui s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi que pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer et grâce à qui ce mémoire a pu prendre jour sous cette forme.

J'exprime ma gratitude aux équipes dans lesquels j'ai travaillé, non seulement à la Sacem pour la réalisation de ce mémoire mais aussi à Ommic lors du début de ma formation. A Messieurs Peter Frijlink et Daniel Rondi qui ont cru en mon potentiel et m'ont permis de me lancer dans cette aventure. A Messieurs Christian Clarenc, Franck Livernette et Serge Heritier qui m'ont accordé leur confiance et leur soutien sur ce projet.

Mes remerciements s'adressent également à Mademoiselle Mélanie Berthelot qui m'a soutenu et encouragé au cours de la réalisation de ce mémoire et qui a eu la gentillesse de lire et corriger ce travail.

Merci à tous et à toutes.

Liste des abréviations

AJAX : Asynchronous Javascript and XML
BAM : Business Activity Monitoring
BI : Business Intelligence
BPM : Business Process Management
BSM : Business Service Management
CD : Compact Disc
DaPCol : Data Progress Collector
DOSIT : Département de l'Organisation, des Systèmes d'Information et des Technologies
DRC : Département des Relations avec la Clientèle
DRS : Département des Relations avec les Sociétaires
DTD : Document Type Definition
DVD : Digital Versatile Disc
eBS : eBusiness Suite d'Oracle
ERP : Enterprise Resource Planning
FAP : Facturation a priori
FTP : File Transfer Protocol
RDO : Reprise de données
Sacem : Société des Auteurs, Compositeurs et Éditeurs de Musique
SDRM : Société pour l'administration du Droit de Reproduction Mécanique des auteurs, compositeurs et éditeurs
SELDR : SELECT pour le réseau
SELECT : Projet d'implémentation de l'eBS pour la DRC
SELOL : SELECT pour le droits de diffusion en ligne
SELVP : SELECT pour les droits Phono-Vidéo
TLG : Télégestion
XML : Extensible Markup Language
SOA : Service-oriented architecture

Glossaire

Fichiers plats : Fichiers contenant un ensemble de données accessible de façon séquentielle et n'ayant pas de format d'application spécifique.

Tables standards : Tables fournies avec l'eBusiness Suite d'Oracle

Tables spécifiques : Tables mises en place pour l'adaptation de l'eBS

Tables intermédiaires : Tables destinatrices du chargement des fichiers plats

Tables consolidées : Tables contenant les données construites à partir des tables intermédiaires (modèle source) afin de pouvoir permettre l'injection dans les tables standards (modèle cible).

Paquetage : ensemble de fonction et procédure regroupé au sein d'un même composant afin de faciliter le travail d'équipe, de permettre de réutiliser des composants existants et de faciliter la maintenance.

Trigger : déclencheur positionné sur une table et exécuté lors de la mise à jour de celle-ci.

Patchs : Section de code ayant pour vocation de corriger certaines données suite à un disfonctionnement ou de préparer des données en vue d'une évolution conceptuelle.

Table des matières

Remerciements 3
Liste des abréviations 4
Glossaire 5
Table des matières 6
Introduction 7
I LA SACEM 9
I.1 L'HISTOIRE DE LA SACEM 9
I.2 LES MISSIONS DE LA SACEM 10
I.3 L'ORGANISATION DE LA SACEM 12
I.3.A Le Conseil d'Administration 13
I.3.B Les Commissions 14
I.3.C Le Directoire 14
II LA MIGRATION RESEAU DU PROJET SELECT 16
II.1 CONTEXTE 16
II.2 L'EQUIPE 18
II.2.A La reprise de données 18
II.2.B L'interface entre l'eBS et la FAP 20
II.2.B.a L'interface de l'eBS vers la FAP 20
II.2.B.b L'interface de la FAP vers la l'eBS 20
II.2.C Le besoin d'un nouvel outil 21
III SOLUTIONS ENVISAGEES 24
IV REALISATION DE L'OUTIL : DAPCOL 28
IV.1 LE MECANISME DE COLLECTE : DAPCOL – COLLECT 29
IV.1.A Schéma conceptuel 30
IV.1.B Implémentation 31
IV.1.B.a La création et la gestion des tables 32
IV.1.B.b L'exécution des audits 33
IV.1.B.b.1 Déclencheur et subdivision 34
IV.1.B.b.2 Stockage des résultats 34
IV.1.B.b.3 Détermination des compléments 35
IV.1.B.c Planification des audits 42
IV.2 L'INTERFACE D'ADMINISTRATION : DAPCOL – ADMIN 44
IV.2.A Gestion des audits 44
IV.2.B Gestion de la planification des audits 48
IV.2.C Gestion des DB links 50
IV.2.D Suivi de l'exécution des audits et exécution manuelle 51
IV.2.E Taille des résultats 53
IV.3 LA GENERATION DE RAPPORT : DAPCOL – REPORTS 54
IV.3.A Le fichier de configuration générale 56
IV.3.B Les classes de l'interface 56
IV.3.C Gestion des paramètres utilisateurs 57
IV.3.D Fichiers des paramètres pour la génération automatique 58
IV.3.E Définition du modèle Jasper 59
IV.3.F Requêtes d'alimentation du rapport 61
V EXPLOITATION 65
Conclusion 68
Bibliographie 70
Ouvrages imprimés 70
Chapitre dans un ouvrage imprimé 70
Sites web 70
Table des annexes 73
Liste des figures 142

Introduction

Avec l'évolution des technologies de l'information, de nombreuses entreprises font désormais migrer leurs différents systèmes informatiques au sein d'un seul et unique système progiciel de gestion. Les objectifs sous-jacents de cette approche sont multiples : décloisonner les différents services de l'entreprise, harmoniser et capitaliser les méthodes de travail, renforcer la prise de décision grâce à un meilleur accès à l'information, rationnaliser les systèmes informatiques et les ressources nécessaires pour leur fonctionnement. C'est ainsi qu'à la Sacem (Société des Auteurs, Compositeurs et Éditeurs de Musique), le DRC (Département des Relations avec la Clientèle) a décidé de se doter d'un ERP (*Enterprise Resource Planning*) et que le projet SELECT a commencé.

Les progiciels de gestion lèvent de nouveaux défis auxquels les informaticiens doivent faire face, à la fois lors des migrations vers le nouveau système mais aussi dans la gestion quotidienne de celui-ci. En effet, garant de la mise en place de ces systèmes dans l'entreprise, les informaticiens ont pour mission d'intégrer l'ensemble des contraintes des différents processus opérationnels de l'entreprise au sein d'un même système informatique. Cette intégration doit s'effectuer en permettant la mutualisation et la propagation de l'information au sein des différents modules qui composent le progiciel de gestion. Ces contraintes peuvent notamment être dues à la structure de l'entreprise[1], l'organisation des différents services[2], le « dialecte » utilisé[3] mais aussi aux différents besoins techniques[4].

---

[1] D'un point de vue hiérarchique, chaque service de l'entreprise à une organisation qui lui est propre et les personnes de ces services peuvent donc avoir des périmètres d'activités différents.

[2] L'ordonnancement des tâches au sein de chaque service peut aboutir à la mise en place de cycles de vie spécifiques sur des composants communs à différents services.

[3] Les différents services peuvent avoir un jargon spécifique avec des notions propres que le nouveau système ne doit pas redéfinir.

[4] Il peut s'avérer nécessaire de recevoir et/ou de transférer de l'information vers des systèmes tiers aux règles différentes, de fournir certains services non prévus par le progiciel de gestion, de permettre l'accès à des ressources supplémentaires durant des phases d'activités ponctuelles de certains services sans dégrader la qualité de service du système dans son ensemble.

Comment pouvons-nous nous assurer de la bonne interopérabilité entre chaque mécanisme et qu'aucun cas particulier n'a été oublié ? Avec la complexité accrue des règles de gestion, il devient impossible de réaliser l'intégralité des tests nécessaires pour garantir le bon fonctionnement de l'ensemble de ces systèmes. De plus, le non-respect des règles de gestion par un système informatique ne produit pas forcément une erreur informatique. Nous risquons donc après plusieurs jours (voire plusieurs mois), de trouver des données dans un statut non-géré ou mal-géré par le système. En outre, lorsqu'un dysfonctionnement est constaté, les données sont dans leur état final. Il est donc particulièrement délicat de savoir rapidement si ce dysfonctionnement est le résultat d'un problème latent (et non détecté auparavant) ou s'il résulte de l'installation d'une évolution ou d'un correctif.

Ce mémoire développe l'approche abordée lors de la réalisation du projet SÉLECT pour la SACEM. Il a pour objectif, la mise en place d'un outil permettant de suivre l'évolution régulière des données de l'ERP afin de permettre la détection de dysfonctionnement le plus rapidement possible, de disposer de suffisamment de données pour comprendre l'enchaînement des mécanismes en cause et de garantir que les corrections apportées ont bien résolu l'ensemble des cas.

Après une description du contexte de réalisation de ce mémoire, nous aborderons les différentes solutions envisagées. Nous détaillerons ensuite les différentes composantes de la solution retenue pour terminer enfin par les résultats d'exploitation de ce nouvel outil.

I La Sacem

La Sacem : Société des auteurs, compositeurs et éditeurs de musique, est une société civile à but non lucratif ayant pour objectifs la collecte et la répartition des droits d'auteur des œuvres musicales de son catalogue. En plus des œuvres musicales au sens commun, le catalogue de la Sacem se compose aussi :

- des musiques d'œuvres audiovisuelles
- des musiques de publicité
- des sketches humoristiques
- des poèmes
- des documentaires musicaux
- des vidéo-clips
- des textes de doublages et sous-titrages de films, téléfilms et séries étrangères
- des extraits d'œuvres dramatiques et dramatico-musicales

I.1 L'histoire de la Sacem

Il faut attendre la fin du 19^{e} siècle et l'affaire du café-concert « Les Ambassadeurs » pour que les auteurs et compositeurs soit payés non seulement lors de la commande de leurs œuvres mais aussi lors de la diffusion de celles-ci. En effet, c'est en mars 1847 que trois auteurs compositeurs connus (Ernest Bourget, Paul Henrion et Victor Parizot) refusent de payer leurs consommations dans le café-concert parisien « Les Ambassadeurs » car celui-ci diffuse leurs œuvres sans les rétribuer. En se basant sur les textes révolutionnaires, ils arrivent à faire reconnaitre devant les tribunaux ce droit à être rétribué pour la diffusion de leurs œuvres. Grâce à cette décision, naît un syndicat des auteurs et compositeurs en 1850 regroupant 221 adhérents. Mais c'est l'année suivante en 1851 que ce syndicat se transforme en société civile, seule structure juridique permettant de rétribuer ses membres sans avoir de vocation commerciale ; la Sacem est née. En 1858, la Sacem prend une envergure nationale grâce à l'implantation de 181 agences. Un siècle plus tard, en 1950, la Sacem compte 15 000 auteurs, compositeurs et éditeurs de musique et répartit 25 millions de titres. Aujourd'hui, c'est un répertoire de 40 millions de titres pour 132 000 sociétaires que gère et protège la Sacem en France mais aussi à l'international soit directement grâce à ses sociétés ou agences étrangères (comme c'est le cas par exemple au Luxembourg ou au

Liban), soit indirectement par le biais d'accords de représentation avec les sociétés d'auteurs locales (comme c'est le cas par exemple avec la GEMA *Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte* société de gestion des droits d'auteur en Allemagne).

I.2 Les missions de la Sacem

Les principales missions de la Sacem sont de protéger, percevoir et répartir les droits d'auteurs des œuvres des auteurs/compositeurs de son répertoire en France mais aussi à travers le monde.
Pour ce faire la Sacem, agit suivant 3 axes :

1. Autoriser la diffusion publique et percevoir les droits d'auteurs :

   Par l'intermédiaire de la SDRM (Société pour l'administration du droit de reproduction mécanique des auteurs, compositeurs et éditeurs), la Sacem autorise, sous forme de contrats, la reproduction des œuvres de son répertoire sur les supports tel que les CD (Compact Disc), les DVD (Digital Versatile Disc), les vinyles et les cassettes à bande magnétique.

   De par les droits que lui accordent ses sociétaires, la Sacem autorise, sous forme de contrats, toute diffusion publique des œuvres de son répertoire. Lorsque la musique est essentielle à l'activité du client, la Sacem perçoit un pourcentage de recette provenant de l'exploitation de la musique (c'est notamment le cas des discothèques et des radios). Lorsque la musique n'a qu'un rôle accessoire dans l'activité de son client, les droits sont calculés forfaitairement (c'est notamment le cas des restaurants et des salons de coiffure).

2. Identifier les œuvres diffusées et répartir les droits perçus :

   Afin de pouvoir effectuer une répartition des droits perçus au plus proche des diffusions effectuées, la Sacem s'appuie principalement sur l'article L.132-2 du Code de la propriété intellectuelle :

   "L'entrepreneur de spectacles est tenu de déclarer à l'auteur ou à ses représentants le programme exact des représentations ou exécutions publiques et de leur fournir un état justifié de ses recettes."

Ainsi, les radios, les télévisions, les organisateurs de concerts et de spectacles, mais aussi les producteurs de disques et de vidéos fournissent régulièrement leurs programmes ainsi que leurs recettes.

Cela permet à la Sacem de garantir que plus de 80% des droits sont répartis selon l'utilisation réelle des œuvres.

En plus de ces informations, la Sacem réalise plus de 7 000 heures de relevés d'écoute dans les discothèques, les bals et les discomobiles (mariage, crémaillère, anniversaire, communion…) qui lui permette de repartir 13% des droits collectés.

Pour les 7% restants, la Sacem se base sur des statistiques de consommation musicale dans les lieux et manifestations considérés.

3. <u>Protéger son répertoire à l'international :</u>

Afin de permettre de contrôler l'utilisation de son répertoire à l'étranger et de percevoir les droits qui lui sont dus, la Sacem peut :

_ soit conclure un accord de représentation avec une société d'auteurs locale. Ainsi celle-ci pourra identifier les œuvres du répertoire de la Sacem, autoriser leur diffusion publique et percevoir les droits d'auteurs correspondant, la Sacem gardant la charge de répartir les sommes ainsi collectées à ses sociétaires.

_ soit créer directement une agence ou une société d'auteur.

I.3 L'organisation de la Sacem

Le principe d'une gestion collective des droits d'auteur assurée par la Sacem s'inscrit dans un fonctionnement axé autour des sociétaires.

Ces sociétaires, auteurs, compositeurs et éditeurs de musique jouent un rôle prépondérant dans la constitution des principaux organes de la Sacem. Effectivement, en élisant les membres du Conseil d'Administration et des Commissions de contrôle, ils prennent une part déterminante dans la définition des orientations politiques et budgétaires de la société comme le montre la figure ci-dessous.

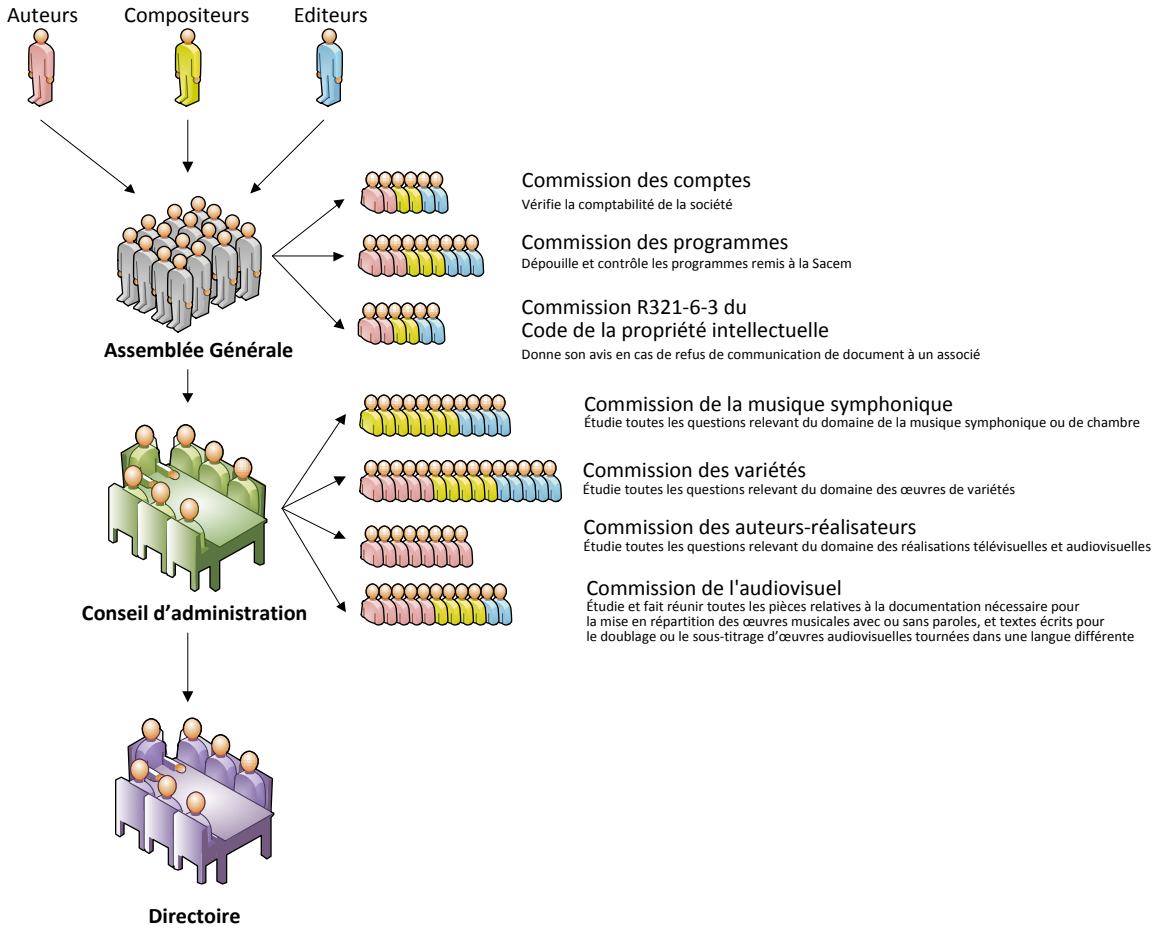

Figure 1 – Organisation de la Sacem

Les différents organes de la Sacem dans leurs rôles et leurs modalités de fonctionnement sont décrits ci-après.

I.3.A Le Conseil d'Administration

Les membres du Conseil d'Administration sont élus par tiers et par catégorie lors d'une Assemblée Générale des sociétaires ayant lieu chaque année le troisième mardi du mois de juin.

Le Conseil d'Administration est composé de 6 auteurs, 6 compositeurs, 6 éditeurs et un auteur-réalisateur.

Pour 2010 – 2011 le Conseil d'Administration est composé de :

- Claude LEMESLE **Président** *auteur*
- Jean-Pierre SPIÈRO **Vice-Président** *auteur-réalisateur*
- Nelly QUEROL **Vice-Président** *éditeur*
- Alain CHAMFORT **Vice-Président** *compositeur*
- Patrick LEMAITRE **Trésorier** *compositeur*
- Alain GORAGUER **Trésorier Adjoint** *compositeur*
- Arlette TABART **Secrétaire Général** *auteur*
- Christian de RONSERAY **Secrétaire Adjoint** *éditeur*
- Gilles AMADO **Administrateur** *auteur-réalisateur*
- Jean-Pierre BOURTAYRE **Administrateur** *compositeur*
- Thierry COMMUNAL **Administrateur** *éditeur*
- Jean FAUQUE **Administrateur** *auteur*
- Christian GAUBERT **Administrateur** *compositeur*
- Pierre GRILLET **Administrateur** *auteur*
- Caroline MOLKO **Administrateur** *éditeur*
- Laurent PETITGIRARD **Administrateur** *compositeur*
- Jean-Max RIVIERE **Administrateur** *auteur*
- Jean-Marie SALHANI **Administrateur** *éditeur*
- David SECHAN **Administrateur** *éditeur*
- Richard SEFF **Administrateur** *auteur*

- Jacques DEMARNY **Président d'honneur** *auteur*
- Gérard DAVOUST **Président d'honneur** *éditeur*
- Gérard CALVI **Président d'honneur** *compositeur*

Le Conseil d'Administration se réunit périodiquement à la demande de son président ou du gérant. La moitié des membres le composant doivent être présents pour qu'une réunion soit valable. Les décisions se prennent à la majorité des administrateurs présents. En cas de partage des voix, celle du président est prépondérante.
Le Conseil d'Administration décide des grandes orientations politiques et budgétaires de la Sacem. C'est lui qui établit les modalités de répartition des droits perçus et qui a la charge de la nomination et la révocation des cadres supérieurs et des directeurs régionaux de la Sacem. Le Conseil d'Administration peut convoquer des Assemblées Générales exceptionnelles afin de soumettre une problématique spécifique aux sociétaires.
Afin de pouvoir prendre ces décisions, le Conseil d'Administration s'appuie sur les Commissions et pour en assurer l'exécution, il nomme le directoire.

I.3.B Les Commissions

Les commissions, exclusivement composées de sociétaires, ont pour missions d'étudier les questions relevant de leur compétence, et de proposer les solutions appropriées (voir figure 1).
Les membres des commissions sont, soit élus par l'assemblée générale (cas des commissions statutaires), soit nommés par le conseil d'administration (cas des commissions réglementaires). S'il estime en avoir besoin, le conseil d'administration peut décider de constituer une nouvelle commission dont il définira alors la mission.

I.3.C Le Directoire

Organe exécutif du Conseil d'Administration, le directoire conduit la politique initiée par les administrateurs. Le président du Directoire est le gérant officiel de la société, il a une compétence générale pour le groupe et toutes les sociétés le constituant, ainsi que tous les organismes internationaux dont elles sont membres.

Pour 2010 – 2011 le Directoire se compose de la façon suivante :

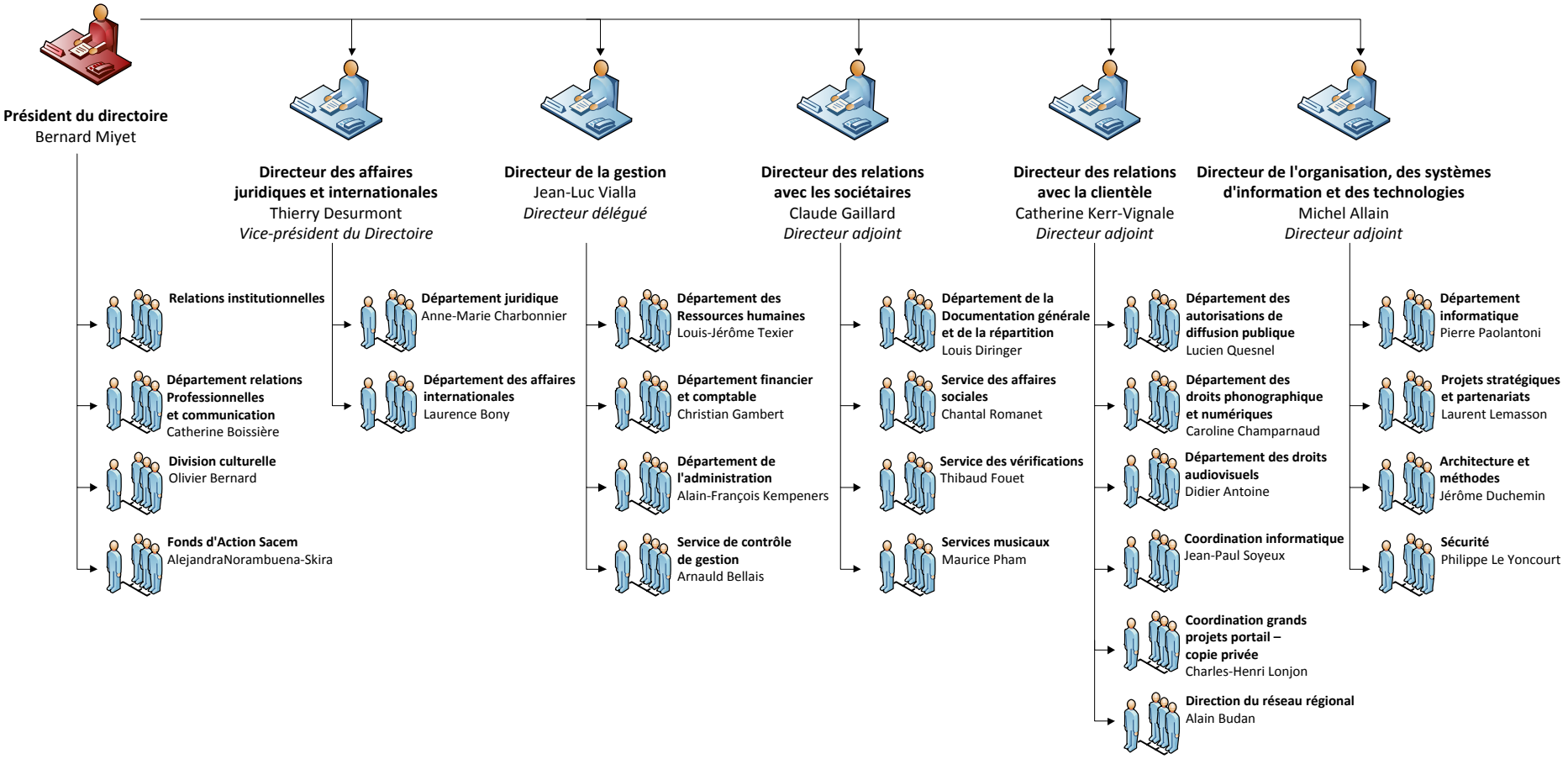

Figure 2 – Composition du directoire

II La migration Réseau du projet SELECT

Ce mémoire a été réalisé au sein du département informatique et s'inscrit dans le cadre du projet SELECT. Dans ce chapitre nous aborderons le contexte de réalisation de ce mémoire, avec dans un premier temps une description du projet SELECT dans son ensemble puis des deux composantes et de l'équipe de ce projet que je coordonne. Enfin nous aborderons les différents raisons ayant nécessité la mise en place d'un nouvel outil et les caractéristiques attendues par celui-ci.

II.1 Contexte

Le projet SELECT a été initié en 2003 avec pour objectif de moderniser les outils informatiques de la DRC (Département des Relations avec la Clientèle). La DRC, sous la responsabilité de Catherine Kerr-Vignale (membre du directoire et directrice adjointe de la Sacem), souhaitait que ce nouvel outil puisse rapidement être mis en place et qu'il permette de disposer d'une vision à 360° du client.

Le but de cette vision était de disposer d'un référentiel client unique permettant, ainsi, une meilleure analyse des besoins du métier et une meilleure réactivité face au développement des nouvelles activités de diffusion.

En effet, la perception des droits s'effectue suivant trois axes principaux :

- SELVP / SELECT Phono-Video : Droits liés à la reproduction des œuvres sur des supports physiques
- SELDR / SELECT Réseaux : Droits liés aux autorisations de diffusion publique à toutes personnes physiques ou morales ayant besoin pour son activité ou de façon ponctuelle de diffuser des œuvres protégées par la Sacem
- SELOL / SELECT On-Line : Droits liés aux ventes dématérialisées de musique

Chacun de ces axes, faisant l'objet d'outils informatiques spécifiques, les clients n'étaient donc pas connus de façon unique mais sous différentes formes suivant les besoins qui leurs étaient liés. De plus, dans chacun de ces outils, certaines fonctionnalités identiques avaient été développées de façon spécifique (notamment la facturation et l'édition).

La solution technique choisie pour ce projet a été l'ERP eBusiness Suite d'Oracle (eBS), version 11.5.9, car il comportait de nombreuses fonctionnalités déjà implémentées et identifiées comme nécessaires par la DRC permettant ainsi une mise en œuvre plus rapide.

En outre, il s'appuie sur la technologie d'Oracle déjà utilisé au sein de l'entreprise. Afin de répondre correctement au besoin spécifique de la DRC, il a tout de même nécessité le développement d'éléments complémentaires spécifiques.
La mise en place du projet SELECT mobilise aujourd'hui une cinquantaine de personnes réparties en huit équipes.

En juin 2009, a été mise en production la partie Réseau (SELDR) du projet SELECT avec un premier déploiement de 3 délégations pilotes. Le Réseau de la Sacem se compose de 86 délégations réparties dans toute la France (outremer et Monaco inclus).

Avant SELECT, les délégations utilisaient principalement deux outils pour gérer leur activité :

- Le premier nommé TLG (TéLéGestion), leur permettant de gérer les contrats des clients (création, tarification, résiliation), était installé sur des serveurs locaux présents dans chaque délégation.
- Le deuxième nommé FAP (Facturation A Priori), leur permettant de gérer les aspects financiers liés aux contrats (facturation, encaissement, mise en plan de règlement, mise en contentieux), était installé sur deux Mainframe situés au siège de la Sacem.

En Janvier 2009, afin de permettre une mise à disposition plus rapide du nouvel outil, il a été décidé que la migration dans SELECT s'effectuerait en deux phases :

- La première phase a consisté à réaliser la migration des aspects contractuels ainsi que les calendriers de facturation des clients ce qui a donc impliqué la reprise complète des données des serveurs TLG et une reprise partielle des données des Mainframe FAP. Cette phase s'est déroulée de septembre 2010 à mai 2011 sur la base d'un planning de déploiement prévoyant la migration de 4 délégations par semaine. A l'issue de cette phase, les serveurs TLG pourraient être définitivement arrêtés, en revanche les Mainframe FAP devront être maintenus.
- La seconde phase consistera en la migration complète de la comptabilité individuelle des clients. Cette phase devrait se dérouler à partir de septembre 2012 et se soldera par l'arrêt des Mainframe FAP.

Ce mémoire s'inscrit donc dans le cadre de la réalisation de cette première phase de migration avec la réalisation de la reprise de données et la mise en place d'une interface avec le Mainframe de facturation.

II.2 L'équipe

L'équipe que je coordonne se compose de 7 personnes et a deux missions :

- réaliser la première phase de la migration
- mettre en place et assurer le suivi de l'interface entre l'eBS et la FAP.

II.2.A La reprise de données

Afin de réaliser cette première phase de migration, nous avons développé des programmes spécifiques permettant de réaliser la reprise des données contractuelles et la reprise des calendriers de facturation des clients. Nous aurons ensuite la charge d'effectuer la migration de chacune des 86 délégations de la Sacem.

Les objectifs de cette première phase de reprise de données sont les suivants :

- Garantir l'intégralité de la reprise des données de chaque client avant leur mise à disposition auprès des délégations.
- Avoir un taux de reprise finalisée de 97% des clients d'une délégation au moment où celle-ci commencera à travailler sur l'eBS (c.à.d. trois jours après le début de la reprise de données).
- Ne pas perturber le travail des utilisateurs de l'eBS durant le déploiement des délégations non migrées.

Durant la phase de développement, nous avons dû faire face à différents problèmes.
En effet, les systèmes sources (TLG et FAP) n'étant que très peu documenté, il a été particulièrement délicat de définir les différentes règles de reprise de données. Nous avons donc dû faire un travail de rétro-ingénierie et solliciter des référents techniques et fonctionnels afin de définir correctement les règles de migrations.
De plus, le système source TLG était beaucoup moins contraint que le système cible eBS et était basé sur une architecture non centralisée. Nous avons donc dû faire face à des typologies de cas spécifiques suivant l'organisation de chacune des 86 délégations.
Dans ces conditions, pour garantir une reprise de données assez rapide, nous avons dû prévoir un certain nombre de mécanismes automatiques dégradant la qualité fonctionnelle de la reprise. Ces mécanismes ont été prévus pour permettre de réaliser la reprise de

données en garantissant le minimum requis pour le nouveau système eBS et en permettant aux utilisateurs de consulter les données d'origine.
Enfin, nous avons dû mettre en place une communication spécifique avec les délégations afin d'obtenir des informations complémentaires nécessaires à l'eBS mais absentes des données TLG et FAP. Nous avons aussi traité les demandes d'information émanant des délégations.

Du point de vue technique, la méthode choisie pour effectuer cette migration est la suivante :

1. Extraction des données sources des serveurs TLG et des Mainframe FAP sous forme de 82 fichiers plats.
2. Chargement de ces données dans 51 tables intermédiaires au sein de l'eBS.
3. Blocage des données répondant aux règles d'épuration (ne devant donc pas faire l'objet d'une reprise).
4. Consolidation des données dans 21 tables consolidées.
5. Injection des données consolidées dans 26 tables standards eBS et 22 tables spécifiques de la solution SELDR.

Les étapes 1, 2 et 3 s'effectuent grâce à 5 programmes et ne sont exécutées qu'une seule fois par groupe de délégations déployées (soit deux fois par semaine le lundi et le mercredi).
Les étapes 4 et 5 s'effectuent grâce à 16 programmes qui sont exécutés tous les soirs.
De plus, pour réaliser nos tests, nous avons aussi développé 11 programmes nous permettant d'annuler les consolidations et/ou les injections.

Afin de pouvoir garantir le traitement complet des données, les programmes ont été construits de façon à traiter systématiquement toutes données non précédemment traitées ou étant restées en anomalie suite aux précédents programmes. Ainsi à la fin de chaque programme de reprise, l'intégralité des données sources nécessaires est marquée comme étant traitée ou non (et si non avec une indication indiquant le code de l'anomalie).

Nous allons maintenant nous intéresser à l'interface entre l'eBS et le Mainframe FAP, qui nous a permis de réaliser cette migration en deux phases.

II.2.B L'interface entre l'eBS et la FAP

Afin de permettre aux utilisateurs de bénéficier du nouvel outil, plus rapidement, il a été décidé que la reprise de données s'effectuerait en deux phases : première phase sur les aspects contractuels, deuxième phase sur les aspects comptables.
Cette décision implique qu'à l'issue de la première phase de cette migration, nous devons donc maintenir les Mainframes FAP. En effet, ce sont eux qui gèrent la comptabilité client individuelle. Cette décision a eu pour conséquence de nécessiter la mise en place d'une interface entre le nouveau système SELECT et les Mainframes FAP.

II.2.B.a L'interface de l'eBS vers la FAP

Pour réaliser l'interface de l'eBS vers la FAP, nous avons conservé le même mécanisme que l'interface utilisé entre la TLG et la FAP. Ainsi, les modifications effectuées au sein de l'eBS sont transférées à la FAP sous forme de transactions au sein d'un fichier plat. Cette opération s'effectue grâce à un ensemble de quatre programmes. Le premier programme, nous permet d'identifier les interlocuteurs eBS ayant été mis à jour ou devant faire l'objet d'une facturation. Le deuxième programme, met à jour les tables spécifiques en comparant pour chaque interlocuteur les données précédemment envoyées à la FAP et celles de l'eBS. Le troisième programme, nous permet d'effectuer le recyclage de transactions précédemment envoyées à la FAP mais rejetées par celle-ci. Et enfin, le dernier programme génère les transactions, les écrits dans un fichier plat et le transfert via une liaison FTP (File Transfer Protocol) à la FAP.

II.2.B.b L'interface de la FAP vers la l'eBS

Pour remonter les informations de la FAP dans l'eBS (principalement les numéros de factures, les pénalités de retard et la situation comptable), nous avons choisi de nous baser sur les mêmes mécanismes que ceux utilisés dans le cadre de la reprise de données. Ainsi, l'intégralité des données comptables des 86 délégations est extraite de la FAP sous forme de 7 fichiers plats. Ceux-ci sont ensuite chargés au sein de 4 tables intermédiaires. A partir de celles-ci, les nouvelles données sont consolidées puis injectées dans l'eBs, alors que les

données précédemment traitées sont mises à jour. L'ensemble de ces opérations est effectuée grâce à 7 programmes exécutés tous les soirs.

II.2.C Le besoin d'un nouvel outil

C'est donc un total de 43 programmes manipulant des données sur 144 tables que nous devons suivre quotidiennement. Un simple suivi des rapports/journaux de ces traitements ne serait pas suffisant. En effet, cela nous prendrait trop de temps et ne permettrait pas d'avoir une analyse pertinente de l'ensemble des anomalies.
Les anomalies remontées par les programmes de la reprise de données et de l'interface eBS-FAP peuvent avoir des origines diverses :

- Cas 1 : Les données ne répondent pas aux règles fonctionnelles car certains cas de figure créés par les utilisateurs n'ont pas été prévus.
  Par exemple : Pour un de ses clients, la délégation de Lyon a utilisé un modèle de contrat « Cars sonorisés ». Ce cas n'était pas prévu car ce modèle était censé être utilisé uniquement dans le cadre d'une gestion nationale et non régionale.
- Cas 2 : Pour qu'un programme traite avec succès ses données, il peut être nécessaire qu'un précédent programme ait réussi à traiter avec succès les données concomitantes.
  Par exemple : Chaque contrat est attaché à des comptes-clients. Donc dans le cadre de la reprise de données, le traitement d'injections de contrats pourra créer avec succès uniquement les contrats pour lesquels les comptes-clients ont pu être créés par le traitement d'injections de comptes-clients.
  Autre exemple : Dans le cadre de l'interface eBS-FAP, nous ne pouvons générer la transaction de création d'un contrat que si nous avons précédemment généré la transaction de création des comptes-clients associés à ce contrat.
- Cas 3 : Les règles fonctionnelles appliquées par un des programmes ne permettent pas aux programmes suivants de traiter correctement l'intégralité de ses données.
  Par exemple : L'application des règles de reprise des contrats pour un client nous conduit à reprendre son contrat dans l'eBS avec une date de fin au 31/12/2000. Cependant dans les données comptables du même client, nous trouvons trace de facturations pour les années 2001 et 2002. Dans ce cas, la stricte application des

règles de reprise des contrats ne nous permet pas de réaliser la reprise des données de facturation 2001 et 2002 du client.

- Cas 4 : Malgré la migration en cours, la DRC a demandé un certain nombre d'évolutions dans SELECT. Nous avons donc rencontré des anomalies suite aux évolutions dans les règles de gestion du modèle de données ou aux évolutions du paramétrage de l'application. Ceci était dû au fait que les programmes de reprises de données ou d'interface n'avaient pas fait l'objet d'une évolution dans le même sens, ils ne pouvaient donc plus traiter correctement les données.
- Cas 5 : Dans le cadre de la reprise de données, certaines informations nécessaires à l'eBS ne sont pas disponibles dans les données FAP et TLG. Pour ces cas de figure, il est nécessaire de demander des informations complémentaires auprès des utilisateurs. Tant que ces informations ne sont pas fournies, les données associées restent en anomalies et ne peuvent donc pas être reprises.

Outre l'origine des anomalies, lorsque des corrections sont apportées aux programmes, il faudrait non seulement s'assurer que le curatif ait été corrigé, mais aussi que les corrections apportées soient correctes et ne génèrent pas d'effets de bord sur d'autres programmes. Un simple suivi des traces ne nous permettrait pas non plus de répondre à ce besoin. Enfin, nous ne pouvons pas nous permettre d'interrompre la migration ou de devoir arrêter la facturation de nos clients. Il nous faut donc un nouvel outil qui nous permette non seulement de suivre au quotidien le traitement de nos données, mais aussi l'évolution des volumétries des différents cas d'anomalies, afin de garantir la meilleure maitrise de notre système.

Les principales caractéristiques attendues de cet outil sont :

- **Pour l'aspect de collecte d'informations (audit)**
  - **Fonctionner avec différentes bases** (Dev., Recette, Prod.) : pouvoir détecter les problèmes au plus tôt et vérifier que les réponses obtenues sur un environnement sont similaires à celles obtenues sur un autre.
  - **Lancer les mécanismes d'audit automatiquement** de façon planifiée avec possibilité de réaliser un pré-contrôle pour connaître la pertinence de la

collecte : dans le cas de la reprise de données il peut s'avérer judicieux de ne faire la collecte qu'en cas d'évolution des données de reprise.

- **Pouvoir définir des groupes d'audit avec des caractéristiques similaires** et pouvant donc être synthétisées au sein d'un même rapport : l'idée est de pouvoir segmenter un même audit de façon à mener l'étude uniquement sur la partie de l'audit qui a évolué.
- **Pouvoir supporter la volumétrie d'une année de suivi** : par exemple, pour la RDO ~ 30 000 lignes de données par collecte à terme, pour l'interface eBS-FAP ~15 000 lignes de données par collecte à terme.
- **Simplicité et rapidité de mise en place de nouveaux audits** : en cas de détection d'une anomalie ou de mise en place d'une évolution, il est important de pouvoir disposer le plus rapidement possible d'un nouvel audit afin d'analyser et de suivre le comportement du système sur la totalité des données et pas seulement sur quelques cas identifiés.

- **Pour l'aspect de présentation des résultats (reporting)**
  - **Pouvoir être consulté par plusieurs personnes** (~10 pers)
  - **Avoir des rapports prédéfinis** : ne pas avoir à reconfigurer les rapports journaliers dont on a besoin.
  - **Pouvoir générer de nouveaux rapports « facilement »**
  - **Les rapports doivent pouvoir faire des agrégations de données** : avoir sur le même rapport aussi bien le détail pour chaque délégation que le total toutes délégations confondues.
  - **Avoir une symbolique explicite** pour mettre en avant les évolutions : les tableaux risquant de contenir beaucoup de données, il faut que les évolutions entre les différentes collectes puissent être rapidement visibles.
  - **Pouvoir définir un code couleur entre les différents éléments mesurés** au sein d'une même collecte : comme pour le point précédent l'idée est de pouvoir lire un rapport de façon plus rapide.

Dans le cadre de la réalisation de ce projet, j'aurai en charge la conception de l'architecture du système, la recherche et la sélection des technologies logicielles nécessaires à la mise en place de ce nouvel outil ainsi que la réalisation du développement, de la recette et du déploiement du nouvel outil.

III Solutions envisagées

L'outil dont nous avons besoin entre dans le concept de BAM (Business Activity Monitoring).

Le BAM a pour objectifs non seulement de fournir des indicateurs sur les processus et activité métier de l'entreprise, mais aussi d'être capable d'établir un diagnostic de disfonctionnement, d'en évaluer les impacts voire même de proposer une solution corrective. Souvent intégré au sein d'une solution BI (Business Intelligence) ou BPM (Business Process Management), le BAM est destiné aussi bien aux responsables métier qu'aux responsables opérationnels. Les premiers pourront ainsi disposer d'une vision d'ensemble de l'évolution de leurs activités et des processus métiers qui sont sous leurs responsabilités. Les seconds pourront déterminer au mieux l'impact des incidents techniques sur les processus métiers et prioriser les actions pour respecter le niveau de service délivré. Le BAM est plus englobant que le concept de BSM (Business Service Management). En effet, ce dernier est plus orienté sur le suivi des ressources. Il a pour objectif, en surveillant les métriques des systèmes des services informatiques, de mieux appréhender les dépendances entre les services métiers et les ressources informatiques/métiers, de connaître en temps réel l'impact d'un dysfonctionnement technique sur les processus métiers et d'aider à la décision d'allocation de ressources supplémentaires, en fonction du niveau de criticité des processus métiers.

L'architecture des solutions BAM peut se résumer à l'intégration des trois couches de services suivants (cf. Figure 3 – Architecture BAM) :

- Une couche de collecte des données : capable d'obtenir des données à partir de sources d'informations variées (bases de données, mainframes, réseaux informatiques, outils de supervision système, progiciels…)
- Une couche d'analyse et de corrélation : capable, à partir des données collectées, des règles métiers et des règles de corrélation, de calculer les indicateurs, de les analyser et de détecter une tendance
- Une couche de restitution : capable non seulement de générer des tableaux de bord mais aussi de générer des alertes

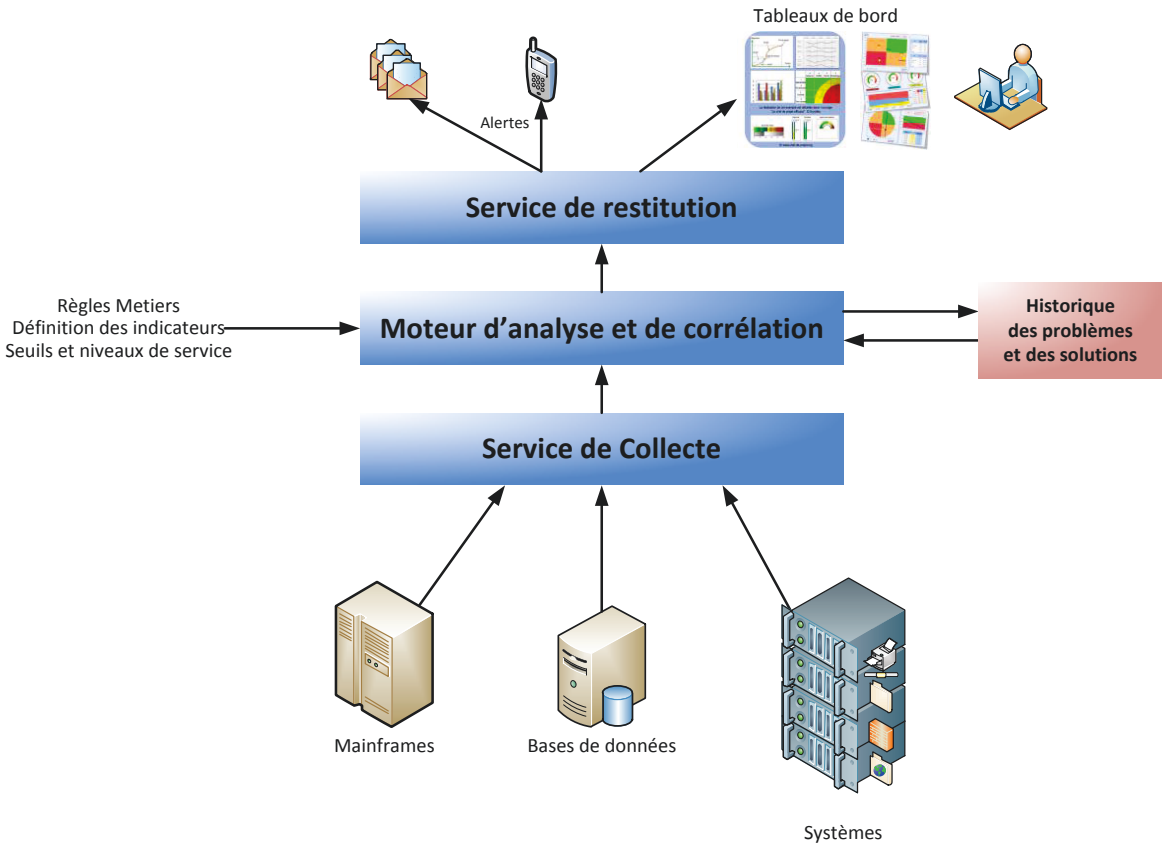

Figure 3 – Architecture BAM

Il existe de nombreux outils BAM sur le marché, les plus connus sont : Synchrony Sentinel d'AXWAY[5], BusinessBridge de SYSTAR[6], Oracle BAM[7], BusinessEvents de TIBCO[8], WebMethods Optimize de SOFTWARE AG[9]. Ces outils étant tous payants, ils ne peuvent pas correspondre à notre besoin.

Nous pourrions alors nous rapprocher de solutions librement et gratuitement accessibles, et en particulier des outils Open Source.

---

[5] http://www.axway.fr/produits-solutions/ps-overview/axway-synchrony/sentinel

[6] http://www.systar.fr/products/businessbridge/

[7] http://www.oracle.com/technetwork/middleware/bam/overview/index.htm

[8] http://www.tibco.com/products/business-optimization/complex-event-processing/businessevents/default.jsp

[9] http://www.softwareag.com/corporate/products/wm/bam/default.asp#

Parmi les outils Open Source les plus adaptés et les plus proches de l'outil dont la mise en place est requise par l'entreprise, mon recensement conduit à étudier plus précisément l'outil WSO2 BAM[10].

WSO2 est une entreprise fondé en 2005, qui fournit des solutions middleware Open Source sous licence Apache Version 2.0. Bien que développé pour fonctionner dans un environnement SOA (Service-oriented architecture), WSO2 BAM est capable de s'intégrer au sein d'environnements non-SOA (cf. Figure 4 – WSO2 BAM in Action), en collectant des données des bases de type : Oracle, DB2, MySQL, MSSQL, Derby, H2, DB2 et PostgreSQL. La génération des rapports et des tableaux de bord peut s'effectuer à l'aide de JSP (Java Server Pages) et/ou de WSO2 Gadget Server.

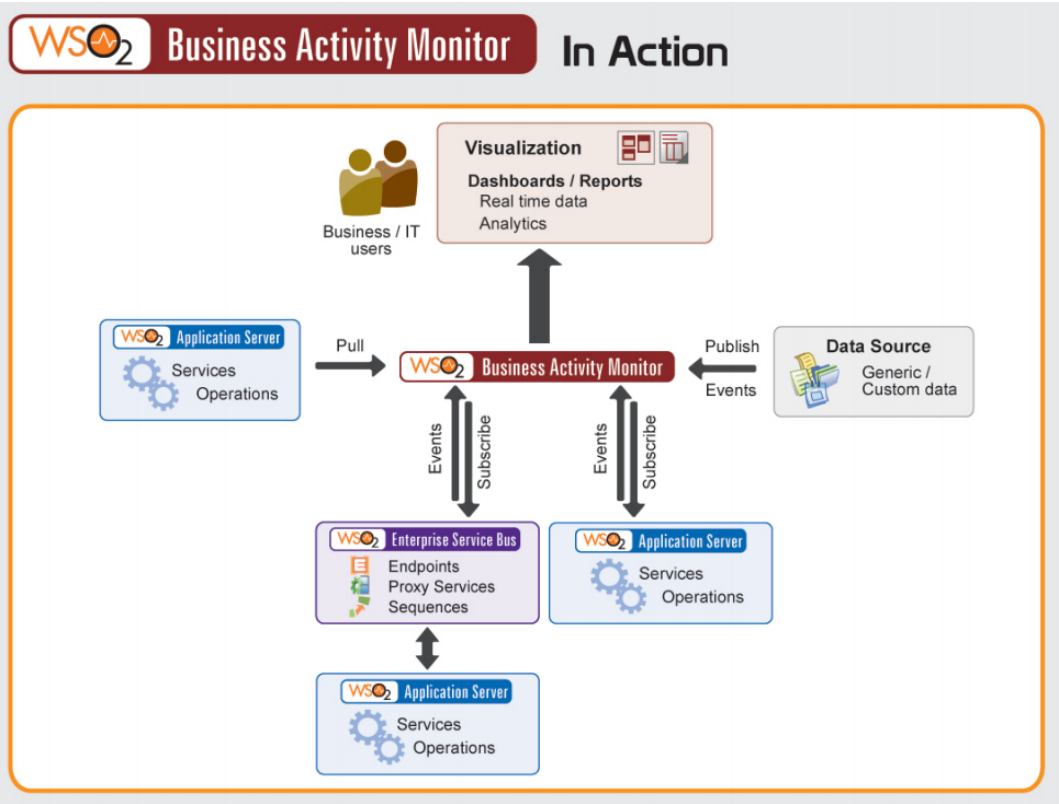

Figure 4 – WSO2 BAM in Action[11]

[10] http://wso2.com/products/business-activity-monitor/

[11] BAM Features datasheet : http://wso2.com/wp-content/datasheets/BAM.pdf

Bien que cet outil se montre particulièrement intéressant, je ne l'ai pas retenu pour des raisons de complexité. En effet, la mise en place d'un flux complet dans cet outil, soit de l'extraction de données à la présentation des résultats, requiert une somme d'actions techniques importantes et spécifiques à réaliser. Ces dernières ne peuvent être réalisées que par une personne maitrisant cet outil et sa technicité.

De telles contraintes sont incompatibles avec le besoin « Simplicité et rapidité de mise en place de nouveaux audits » explicité au chapitre II.2.C page 21, sur lequel nous souhaitons nous appuyer pour l'outil à utiliser.

IV Réalisation de l'outil : DaPCol

Compte tenu des caractéristiques attendues et solutions envisagées dans le chapitre III, j'ai décidé de développer moi-même notre outil de suivi en me basant sur l'architecture présenté en Figure 5. Cet outil, que j'ai appelé DaPCol (Data Progress Collector), permet d'exécuter des requêtes SQL sur des environnements tiers, d'en sauvegarder les résultats et de générer des rapports sur l'évolution de ses résultats dans le temps. Les données étant centralisées, nous pouvons aussi comparer les données entre les différents environnements.

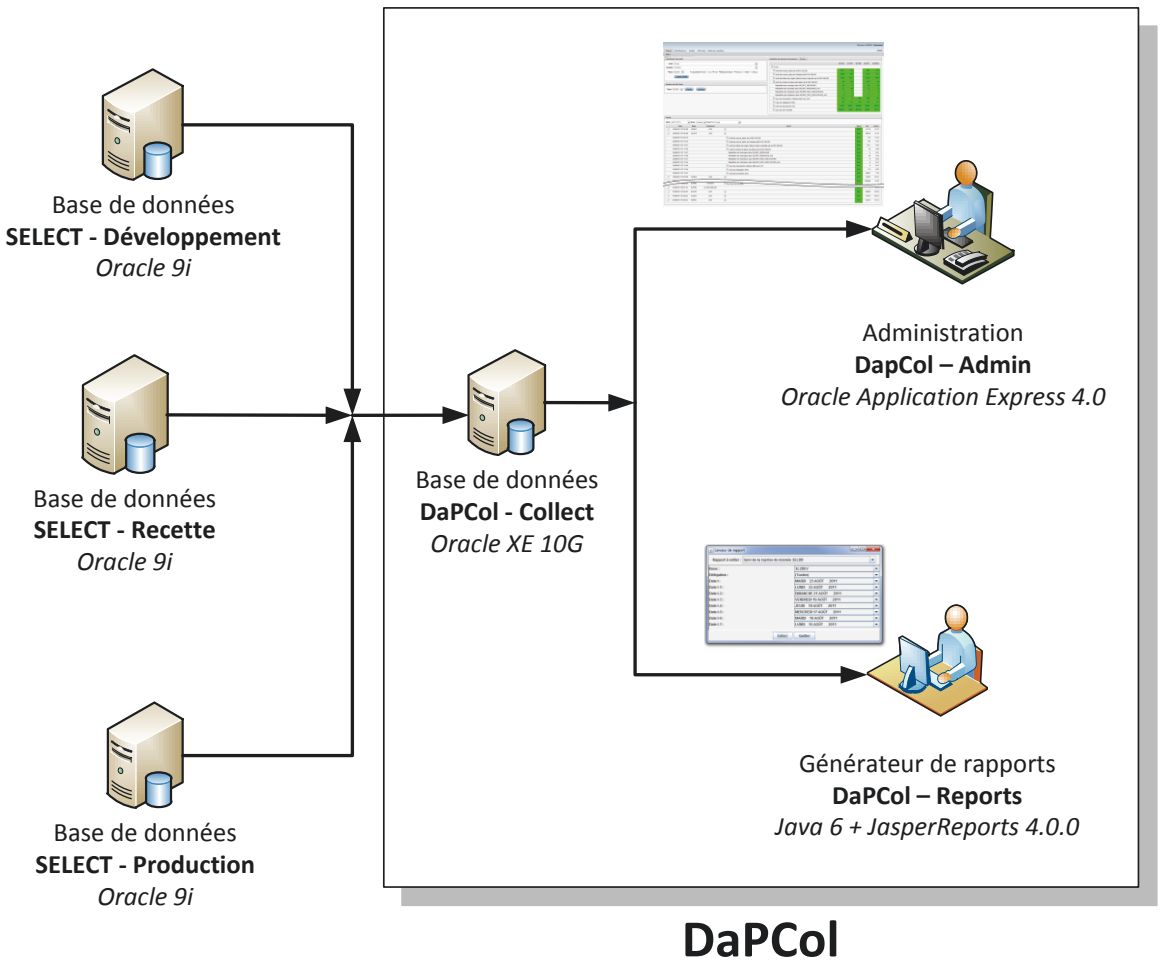

Figure 5 – Architecture DaPCol

L'architecture retenue peut se décrire de la façon suivante :

- Grâce à une base de données sous Oracle XE 10G, j'ai implémenté un mécanisme de collecte (DaPCol – Collect) permettant d'analyser et d'historiser les données

provenant des différentes bases de données (développement/recette/production) du projet SELECT.

- Pour configurer et exploiter ce mécanisme au quotidien, j'ai développé une interface web (DaPCol – Admin) à l'aide d'Oracle Express Application fourni avec la base de données Oracle XE 10G.
- Enfin, pour générer les rapports d'analyse, je me suis basé sur JasperReports et j'ai développé une application Java afin de fournir une interface et des fonctionnalités correspondantes à nos besoins (DaPCol – Reports).

Nous allons donc dans un premier temps aborder le mécanisme de collecte que j'ai implémenté. Nous appréhenderons ensuite l'interface d'administration mise en place afin de réaliser la gestion quotidienne de ce mécanisme de collecte. Enfin, nous terminerons ce chapitre par la présentation de l'outil de reporting qui nous permet de générer les différents rapports d'analyse à partir des données collectées.

IV.1 Le mécanisme de collecte : DaPCol – Collect

Le mécanisme de collecte a pour objectif, l'exécution de requête SQL sur des environnements tiers (de façon ponctuelle ou planifier) et le stockage des résultats de ces requêtes.

Pour cette partie, il a fallu choisir une base de données relationnelle, le choix de ce produit est basé sur plusieurs critères. En effet, pour cet aspect du projet nous voulions utiliser une base de données gratuite, nous avons donc envisagé d'utiliser MySQL. Cependant nous voulions aussi que cette base de données puisse être mise en place rapidement et simplement. La reprise de données SELDR devant commencer en septembre 2010 (cf. §II.2.A p18), nous voulions donc que l'ensemble du système donne ses premiers résultats en 5 mois. Enfin nous voulions aussi garder comme possibilité de pouvoir migrer rapidement vers une version professionnelle, si nous avions besoin de plus de performance. La Sacem utilisant déjà, depuis plusieurs années, des bases de données Oracle (notamment

dans le cadre du projet SELECT), nous avons donc opté pour baser le mécanisme de collecte sur une base de données Oracle Express 10g.
Oracle Express 10g[12] est une version légère mais gratuite de la base de données Oracle 10g qui autorise jusqu'à 4Go de stockage de données utilisateur, fonctionne sur un seul processeur et utilise au maximum 1Go de Mémoire vive (Il est à noter que la version Oracle Express 11g autorisera quant à elle jusqu'à 11Go de données utilisateur[13]).

IV.1.A Schéma conceptuel

Par convention, nous avons décidé de considérer que le mécanisme de collecte se composait de plusieurs audits. Chaque audit exécutant sur la base de données auditée des requêtes de comptage et celles-ci déclenchant suivant les résultats remontés des requêtes d'analyses complémentaires. Nous avons donc abouti au schéma conceptuel représenté en Figure 6.

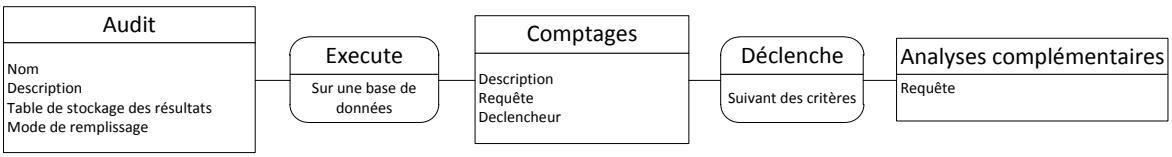

Figure 6 – Schéma conceptuel du mécanisme de collecte

Afin de limiter l'exécution des requêtes de comptage, potentiellement consommatrices de ressources, il a été prévu de pouvoir conditionner leur exécution à un test préalable. Ce test préalable, que nous appellerons déclencheur, consiste en l'exécution d'une requête plus légère ayant pour vocation de se comporter de façon booléenne (ex : le programme X s'est-il exécuté depuis le dernier comptage ?). La mise en place de ce mécanisme implique, soit que le comptage permette d'obtenir les informations nécessaires au prochain déclencheur, soit qu'un autre audit permette de récupérer ces informations.

---

[12] Oracle Database Express Edition Home Page : http://www.oracle.com/technetwork/database/express-edition/overview/index.html

[13] Oracle Database 11g Express Edition R2 Online Documentation : http://www.oracle.com/pls/xe112/homepage

L'idée d'effectuer des analyses complémentaires vient du constat suivant : lorsque certaines données n'ont pu être traitées, les traitements nous indiquent généralement un code anomalie qui permet de savoir à quelle étape le problème est survenu. La première fois que ce type de problème survient, nous effectuons souvent une analyse plus poussée afin de déterminer la cause exacte de l'anomalie et de corriger le traitement. Cependant les corrections ne peuvent être directement appliquées sur les environnements de production (sauf cas de force majeure). Notre système doit donc continuer à travailler avec le programme défaillant quelques semaines durant lesquelles des nouvelles données vont devoir être traitées. Le nombre d'anomalies va donc naturellement augmenter. Cependant une question se pose : si le nombre d'anomalies change, la cause de celles-ci est-elle toujours la même ? Il se pourrait que la première analyse n'ait relevé qu'une cause partielle de la défaillance. Il serait donc pertinent en cas d'anomalies de pouvoir vérifier, jusqu'à ce que le correctif soit installé, que notre première analyse est toujours correcte. Ce mécanisme a été particulièrement utilisé dans le cadre de la reprise de données et nous a permis un suivi au plus près des anomalies malgré un rythme de déploiement soutenu.

IV.1.B Implémentation

Nous aborderons dans ce chapitre la façon dont le mécanisme de collecte a été implémenté : choix de la base de données, tables créées et procédures associées.

Pour mettre en place le mécanisme de collecte, nous avons d'abord installé la base de données Oracle Express 10g[14] sur un PC équipé de Windows XP (PC Dell Optiplex 755 équipé d'un processeur Intel Core 2 Duo 1.8GHz et 4 Go de RAM).
L'utilisation d'une machine indépendante pour ce mécanisme nous offre plusieurs avantages :

- Ne pas surcharger les différentes bases de données auditées lors des générations de rapports.
- Stocker les données d'audit à part et donc pouvoir aussi effectuer des comparaisons entre les bases de données.

[14] Installation d'Oracle Express 10g : http://www.rci-informatique.fr/oracle_xe/

- Ne pas perturber le rythme d'installations des bases auditées en nécessitant des installations uniquement pour le mécanisme de collecte.
- Disposer d'un seul et unique point de configuration et d'administration.

Nous avons ensuite créé un utilisateur oracle « AUDITU » auquel nous avons donné les droits de consulter les tables et vues « DBA » et « ALL » et d'exécuter les packages « DBMS ».
Nous avons également créé deux tablespaces, le premier « AUDITD » de 500 Mo dédié aux données et le deuxième « AUDITX » de 100 Mo dédié aux index.
Enfin, nous avons créé les différentes tables à partir du schéma conceptuel (cf. Figure 6 page 30).

IV.1.B.a La création et la gestion des tables

Pour que le mécanisme de collecte fonctionne, nous avons besoin de créer et d'administrer un certain nombre de tables (notamment pour stocker les requêtes à exécuter). Plutôt que d'avoir à gérer ces tables à partir d'un script de création et de faire ensuite des scripts de modifications au fur-et-à-mesure des évolutions (ajout d'un champ, d'une contrainte, modification du type de données d'un champ), j'ai décidé de mettre en place un mécanisme de gestion dynamique. L'idée est d'avoir uniquement à décrire la structure de la table voulue. Ensuite une procédure sera en charge d'analyser la base de données et d'exécuter les ordres nécessaires pour effectuer la mise à niveau.
Pour cette procédure, j'ai défini dans le paquetage « PKG_AUDIT_COMMUN » les structures permettant de définir une table et ses colonnes. J'ai ensuite créé une procédure « SP_DEFINE_TABLE »[15] permettant de gérer la création de la table avec ses options de construction, la déclaration ou la modification des clés primaires et secondaires, l'ajout ou la suppression de colonnes, les changements de types de données (s'ils sont autorisés par la base) ainsi que l'ajout ou la suppression de contraintes ou de valeurs par défaut sur les colonnes.

[15] Annexe 1 - Déclarations et procédure pour la gestion des tables dans le paquetage PKG_AUDIT_COMMUN (page 73)

A partir de cette procédure et du schéma conceptuel, j'ai alors créé les tables suivantes[16] :

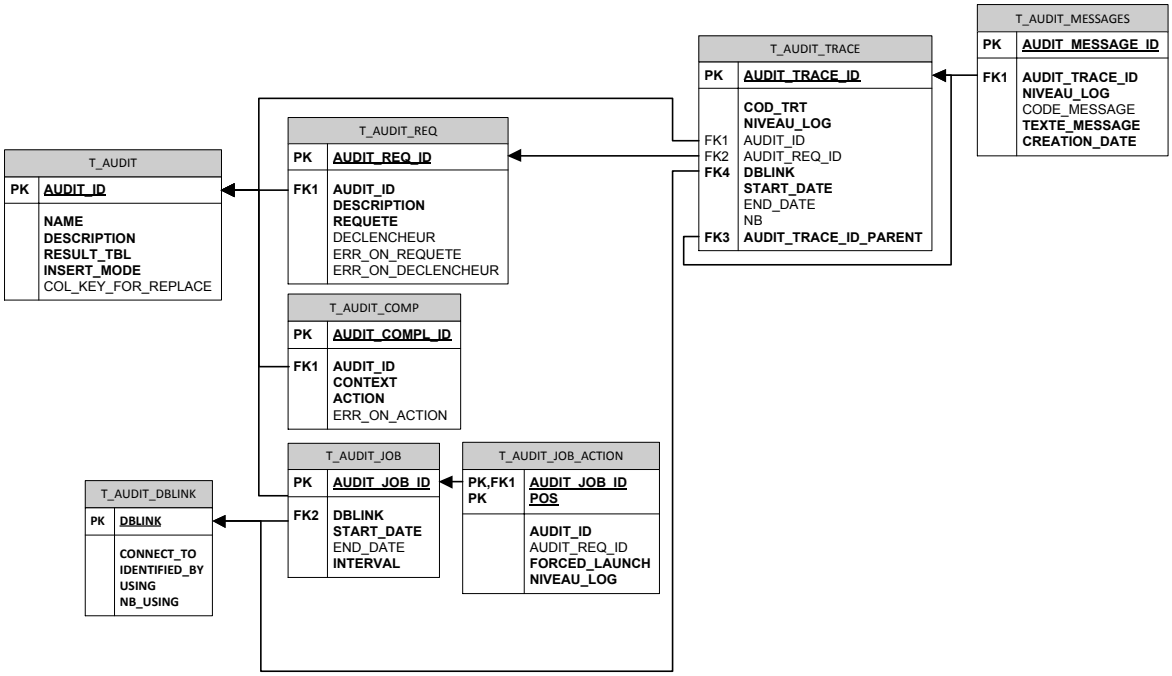

Figure 7 – Schéma des tables

Les tables « T_AUDIT, T_AUDIT_REQ » et « T_AUDIT_COMP » servent à définir les audits.

La table « T_AUDIT_DBLINK » permet de définir les connexions vers les bases de données qui seront auditées.

Les tables « T_AUDIT_JOB » et « T_AUDIT_JOB_ACTION » permettent de gérer la planification des audits.

Enfin les tables « T_AUDIT_TRACE » et « T_AUDIT_MESSAGES » permettent de conserver la trace de chaque exécution d'audit avec son rapport associé.

IV.1.B.b L'exécution des audits

Le mécanisme de collecte est basé sur le principe d'exécution d'une requête sur des bases de données tierces, et du stockage des résultats afin de pouvoir les comparer dans le temps ou entre bases de données. Nous aborderons dans ce chapitre les différents concepts exploités pour réaliser les audits de façon efficiente.

---

[16] Annexe 2 - Exemple de script de création de table (page 82)

IV.1.B.b.1 Déclencheur et subdivision

La mise en place d'un mécanisme de collecte peut vite conduire à générer un stress important sur les bases de données auditées, notamment dû aux ressources nécessaires pour exécuter les requêtes d'analyses. Afin de limiter ce problème, nous avons mis en place deux mécanismes : le test d'un déclencheur et la subdivision des audits.

La subdivision des audits consiste à éviter d'avoir une requête d'audit trop imposante et qui souvent est la somme d'union entre plusieurs sous-requêtes. Nous pouvons donc à la place définir *n* sous-audits correspondant à chacune des sous-requêtes minimales (c.à.d. ne pouvant pas être réduites).

La subdivision prend tout son intérêt grâce au système du déclencheur. L'idée du test d'un déclencheur est d'associer à chaque requête de sous-audit une autre requête. Cette requête nettement moins coûteuse doit permettre de déterminer l'opportunité ou non d'exécuter la requête de sous-audit.

Par exemple :

> Dans le cadre de la reprise de données « SELDR », j'ai mis en place un audit pour suivre l'évolution des « RECORD_STATUS » (état de traitement de chacune des lignes des tables). Cet audit est en fait subdivisé en 15 sous-audits propres aux différents traitements de cette reprise de données. Ce faisant, nous avons aussi pu définir 15 déclencheurs permettant de déterminer si chacun des traitements impactant les données a été exécuté depuis le précèdent audit. Ainsi, les bases de données peuvent être auditées plusieurs fois par jour tout en limitant le stress résultant.

Bien sûr, la définition des déclencheurs n'étant pas toujours possible, les requêtes de déclenchement sont optionnelles. En outre, nous avons aussi mis en place dans le mécanisme de collecte, un mode de lancement dit « forcé » permettant d'exécuter un audit sans tester les déclencheurs.

IV.1.B.b.2 Stockage des résultats

De façon à réduire au minimum les tâches d'administration sur la base d'audit, les tables de stockage des résultats sont gérées dynamiquement. En effet, les résultats de chaque exécution d'audits sont d'abord stockés dans des tables temporaires grâce au mécanisme ORACLE de création dynamique de table : « CREATE TABLE *XXX* AS SELECT… » .

Une fois la table temporaire créée à partir de la requête de sous-audit, nous y ajoutons un certain nombre de champs de gestion qui seront donc présents dans toutes les tables de résultats : la date de l'audit, l'identifiant de la requête d'audit, l'identifiant du lancement de l'audit, le nom de la base de données auditée, le résultat des actions complémentaires et la date de fin de validité de l'audit.

Après le calcul des champs de gestion, les résultats sont transférés dans la table définitive. Si celle-ci n'existe pas alors, il suffit de renommer la table temporaire. En revanche, si la table existe, il faut d'abord s'assurer de la compatibilité des champs entre la table temporaire et la table définitive. Cette contrainte impose qu'en cas de subdivision, tous les champs de même nom doivent être de même type. Pour les champs de type texte, Oracle adapte la taille des champs automatiquement au plus grand texte remonté par la requête de sous-audit, j'ai donc également ajouté un mécanisme qui permet d'agrandir les champs texte de la table définitive au-fur-et-à-mesure des sous-audits.

Enfin, en cas d'erreur lors de l'exécution de la requête du déclencheur ou de la requête de sous-audit un drapeau « ERR_ON_REQUETE » ou « ERR_ON_DECLENCHEUR » est positionné sur la table « T_AUDIT_REQ » (cf. §IV.1.B.b.3.4)[17].

IV.1.B.b.3 Détermination des compléments

Pour chaque enregistrement remonté dans la table temporaire, le mécanisme d'audit nous permet de chercher des informations complémentaires, grâce à des requêtes contextuelles spécifiques. Ce mécanisme a pour vocation d'ajouter un niveau de raffinement supplémentaire dans l'analyse des erreurs. En effet, la requête d'audit peut, par exemple, nous indiquer que *n* enregistrements ne peuvent être traités par le processus de reprise de données car un paramétrage est manquant. Mais quel paramétrage exact est manquant ? Est-ce le même pour les *n* enregistrements ? Pour répondre à ces deux questions, il faudrait soit complexifier la requête de sous-audit (ce qui nous pénaliserait que l'erreur soit présente ou non), soit pouvoir exécuter une analyse complémentaire uniquement lorsque l'erreur se présente. Nous avons choisi de proposer cette deuxième optionalité. Bien sûr, il est nécessaire de faire une analyse minutieuse avant d'exploiter cette fonctionnalité car le

[17] Cf. Figure 7 – Schéma des tables (page 26)

déclenchement systématique d'une analyse complémentaire peut s'avérer plus couteux que de complexifier la requête de sous-audit.

La définition des compléments d'audit est gérée par la table « T_AUDIT_COMP »[18]. Cette table ce compose principalement des deux champs : « CONTEXT » et « ACTION ».

IV.1.B.b.3.1 Définition du contexte

Pour déclencher le lancement d'une analyse complémentaire, nous avons besoin de définir quel est le champ d'application de cette analyse (c.à.d. quand déclencher l'exécution de la requête complémentaire). Comme les tables de résultats, ne comporte pas toutes les mêmes champs, nous ne pouvons pas nous baser sur une structure simple pour décrire le contexte d'exécution. J'ai donc choisi de me baser sur un champ « CONTEXT » de type XMLTYPE[19]. Cela nous permet de constituer une structure XML définissant les champs et leurs valeurs requises pour l'exécution de la requête complémentaire.

L'élément racine du champ « CONTEXT » est toujours le terme « CONTEXTES ». Ensuite chaque feuille correspond à un des champs de la table des résultats d'audit. La valeur de la feuille correspond aux contraintes demandées sur le champ des résultats d'audit.

Par exemple :

```
<CONTEXTES>
     <TRT>'SELDR0025'</TRT>
     <SRC>'SELDRTI%'</SRC>
     <RECORD_STATUS>'ANO'</RECORD_STATUS>
     <CODE_ANO>'SELDR0025_ERR_NOCTRTYP_MAPPE'</CODE_ANO>
</CONTEXTES>
```

Ce contexte nous indique que ce complément d'audit est valable si le champ « TRT » vaut « SELDR0025 », le champ « RECORD_STATUS » vaut « ANO », le champ « CODE_ANO » vaut « SELDR0025_ERR_NOCTRTYP_MAPPE » et que le champ « SRC » commence par « SELDRTI ».

Grâce à cette structure XML et aux fonctionnalités fournies par Oracle permettant d'exploiter le format XML directement dans les requêtes, nous pouvons déterminer rapidement quels sont les compléments à exécuter à partir des résultats de chaque sous-audit (cf. §IV.1.B.b.3.3 p38).

---

[18] Cf. Figure 7 – Schéma des tables (page 26)

[19] Using XMLType : http://download.oracle.com/docs/cd/B10501_01/appdev.920/a96620/xdb04cre.htm

IV.1.B.b.3.2 Définition des actions

Le champ « ACTION » lui aussi de type XMLTYPE est constitué d'une structure XML qui permet de définir les actions à réaliser pour ce complément. Nous avons défini deux types d'actions :

Les actions de type message « MSG » qui permettent d'associer un message fixe un peu plus explicite que les codes anomalies remontés par les requêtes.

Les actions de type requête « REQ » qui sont exécutées avec les valeurs de certains champs des résultats d'audit.

L'élément racine du champ « ACTION » est le terme « ACTIONS ». Ensuite chaque feuille « MSG » ou « REQ » sera exécutée de façon séquentielle. Nous pouvons donc définir plusieurs « MSG » ou plusieurs « REQ » pour analyser correctement les enregistrements des résultats d'audit.

Par exemple :

```
<ACTIONS>
     <MSG>Aucun type de contrat n'est indiqué dans le mapping teta pour le teta</MSG>
     <REQ>
     SELECT  COUNT(*)  ||  '  cas  où  le  TETA  est  '||  LPAD(TYP_ETAB,3,0)||  '-'  ||
LPAD(NO_ETAB,2,0) || '-' || LPAD(TYP_AUD,3,0)|| '-' || LPAD(NO_AUD,2,0)
     FROM   SELDRTI_CONTRAC@DATABASE
     WHERE  GTS0025 = 'ANO'
     AND    CODE_ANO_GTS0025 = 'SELDR0025_ERR_NOCTRTYP_MAPPE'
     AND    DRDL=NVL(:NO_DRDL,DRDL)
     GROUP BY TYP_ETAB,NO_ETAB,TYP_AUD,NO_AUD
     ORDER BY COUNT(*) DESC
     </REQ>
</ACTIONS>
```

Donnera comme résultat :

Aucun type de contrat n'est indiqué dans le mapping teta pour le teta

6 cas où le TETA est 272-01-229-01

1 cas où le TETA est 203-01-201-01

1 cas où le TETA est 272-01-229-02

Après substitution de « :NO_DRDL » par la valeur du champ « NO_DRDL » présent dans la table temporaire.

IV.1.B.b.3.3 Identification des compléments

Après l'exécution de la requête de sous-audit et la création de la table temporaire, il faut donc identifier pour chaque résultat quel complément s'applique. Ceci revient à chercher à quel contexte correspondent les différents champs de chaque résultat.

Pour ce faire, nous avons utilisé le fait que les contextes des compléments étaient stockés sous forme XML. Ainsi, l'utilisation de la fonction « EXTRACTVALUE »[20] nous permet d'exécuter dynamiquement une requête du type :

```
SELECT (
  SELECT  insertXMLbefore(ACTION,'/ACTIONS/REQ[1]',XMLTYPE('<ID>' || AUDIT_COMP_ID ||
'</ID>'))
  FROM   T_AUDIT_COMP
  WHERE  AUDIT_ID = PN_AUDIT_ID
  AND    (EXTRACTVALUE(CONTEXTE, '/CONTEXTES/CHAMP1') IS NULL OR
NVL2(A.CHAMP1,A.CHAMP1,'''NULL''') LIKE EXTRACTVALUE(CONTEXTE, '/CONTEXTES/CHAMP1'))
  AND    (EXTRACTVALUE(CONTEXTE, '/CONTEXTES/CHAMP2') IS NULL OR
NVL2(A.CHAMP1,A.CHAMP2,'''NULL''') LIKE EXTRACTVALUE(CONTEXTE, '/CONTEXTES/CHAMP2'))
  AND    (EXTRACTVALUE(CONTEXTE, '/CONTEXTES/CHAMP3') IS NULL OR
NVL2(A.CHAMP1,A.CHAMP3,'''NULL''') LIKE EXTRACTVALUE(CONTEXTE, '/CONTEXTES/CHAMP3'))
  ...
  AND ROWNUM = 1
  ) ACTION
  ,A.ROWID
FROM TABLE_TEMPORAIRE A
```

En ajoutant une feuille « ID » contenant l'identifiant du complément cela nous permet en cas d'erreur lors de l'exécution des actions de positionner un drapeau « ERR_ON_ACTION » sur la table « T_AUDIT_COMP » (voir §IV.1.B.b.3.4).

IV.1.B.b.3.4 Gestion des traces et des erreurs

[20] EXTRACTVALUE :
http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/functions054.htm

Une des caractéristiques principales de ce mécanisme de collecte, est le fait que les audits soient planifiés. Ceci implique que les audits s'exécuteront sans intervention humaine, il est donc primordial dans ce contexte d'avoir une parfaite gestion des traces. Lorsque tout se passe bien, cela permet de savoir exactement quels sous-audits ont été exécutés (notamment suite à l'utilisation des déclencheurs §IV.1.B.b.1 p34). En cas d'erreurs, cela permet d'avoir assez d'information pour analyser correctement le dysfonctionnement et minimiser au maximum la maintenance.

Les traces sont gérées aux travers de deux tables dédiées : « T_AUDIT_TRACE » et « T_AUDIT_MESSAGES »[Erreur ! Signet non défini.].

La table « T_AUDIT_TRACE » permet d'enregistrer les lancements de chaque traitement ainsi que l'état de sortie de celui-ci. Elle se compose des champs suivants :

- AUDIT_TRACE_ID : Identifiant du traitement
- COD_TRT : Libellé court du traitement
- NIVEAU_LOG : Niveau de trace demandé (Normal, Détail, Debug)
- AUDIT_ID : Identifiant de l'audit
- AUDIT_REQ_ID : Identifiant de la requête d'audit
- DB_LINK : Identifiant de la base de données
- START_DATE : Date de début du traitement
- END_DATE : Date de fin du traitement
- STATUT : Statut de sortie du traitement (Succès, Avertissement, Anomalie)
- NB : Nombre d'élément traité
- AUDIT_TRACE_ID_PARENT : Identifiant du traitement père (dans le cas de traitement récursif)

Cette table permet donc d'identifier rapidement tout traitement et de savoir s'il s'est terminé en anomalie. Elle peut aussi rapidement permettre de comparer l'évolution de la durée du traitement et/ou du nombre de données traitées.

Le renseignement de cette table s'effectue grâce à la fonction « FN_START_TRACE » et à la procédure « SP_END_TRACE » définie dans le paquetage « PKG_AUDIT_COMMUN ».

Chaque traitement (y compris les patchs) peut donc implémenter, l'appel à ces deux fonctions/procédures pour garantir la conservation de l'historique des actions effectuées sur la base de collecte.

La table « T_AUDIT_TRACE » ne conserve comme information que le fait qu'un programme ait été lancé, et si celui-ci a rencontré une erreur ou non. Cette table seule ne permet donc pas d'avoir le détail de l'exécution de chaque programme. Pour ce faire la table « T_AUDIT_MESSAGES »[21] permet d'enregistrer le compte-rendu d'exécution de chaque programme. Cette table nous permet donc de disposer du journal d'exécution de chaque programme.

Elle se compose des champs suivants :

- AUDIT_MESSAGE_ID : Identifiant du message
- AUDIT_TRACE_ID : Identifiant de la trace
- NIVEAU_LOG : Niveau de trace du message (Normal, Détail, Debug)
- CODE_MESSAGE : Code d'identification du message
- TEXTE_MESSAGE : Texte du message
- CREATION_DATE : Date du message

Le renseignement de cette table s'effectue à partir de la procédure « SP_ECRIRE_MSG » définie dans le paquetage « PKG_AUDIT_COMMUN ». Cette procédure utilise comme paramètres d'entrée : AUDIT_TRACE_ID, CODE_MESSAGE, TEXTE_MESSAGE et NIVEAU_LOG. En comparant au moment de son exécution le niveau de trace du message et celui demandé lors du lancement du programme, ce mécanisme permet d'obtenir un compte-rendu d'exécution plus ou moins détaillé sans devoir modifier le code du traitement lui-même. Bien sûr, tous les messages d'erreur doivent impérativement se trouver sur le niveau de trace « normal » de façon à être systématiquement présents quel que soit le niveau de trace demandé. De plus, la possibilité de pouvoir ajouter un code d'identification du message permet de trouver plus rapidement la partie du code sur laquelle l'erreur est survenue.

Pour garantir au mieux la compréhension de l'enchaînement des différentes fonctions et procédures, un mécanisme de gestion de contextes a été mis en place.

A chaque entrée dans une fonction/procédure, un appel à la procédure « SP_ENTER_CONTEXT » définie dans le paquetage « PKG_AUDIT_COMMUN »

[21] Cf. Figure 7 – Schéma des tables (page 26)

permet d'empiler le nom de la fonction/procédure. En sortie de la fonction/procédure un appel à « SP_LEAVE_CONTEXT » défini dans le paquetage « PKG_AUDIT_COMMUN » permet de dépiler le nom de la fonction/procédure. Grâce à ce mécanisme de contexte, lors de l'appel à la procédure « SP_ECRIRE_MSG », le nom de la fonction/procédure est automatiquement ajouté au message avec un mécanisme d'indentation en fonction de la taille de la pile des contextes[22].

Compte tenu du mécanisme de planification et du lancement contextuel des requêtes, nous n'avons jamais l'intégralité des requêtes qui sont exécutées au sein d'un seul audit. Ceci implique qu'une erreur peut survenir lors de l'exécution d'une requête sur un audit mais ne pas se reproduire au prochain lancement du même audit sur la même base de données. En conséquence, si nous nous préoccupons uniquement de la dernière trace du lancement d'un audit, nous risquons de ne pas identifier qu'un sous-audit ou qu'un complément n'est pas correct. Afin de permettre de mieux identifier rapidement ce genre d'erreur et d'éviter de devoir dépouiller systématiquement chaque compte-rendu de chaque traitement, j'ai donc créé les champs « ERR_ON_REQUETE » et « ERR_ON_DECLENCHEUR » sur la table « T_AUDIT_REQ » et le champ « ERR_ON_ACTION » sur la table « T_AUDIT_COMP ». Ces colonnes sont renseignées dès qu'une erreur survient et ne sont effacées qu'après correction des requêtes.

[22] Annexe 4 – Exemple de trace du mécanisme de collecte (page 104)

IV.1.B.c Planification des audits

La planification des audits s'effectue grâce aux fonctions du paquetage standard « DBMS_SCHEDULER »[23] et à partir des deux tables : « T_AUDIT_JOB » et « T_AUDIT_JOB_ACTION »[24].

La table « T_AUDIT_JOB » gère la planification en tant que tel. Elle contient donc les champs suivants :

- AUDIT_JOB_ID : Identifiant de la tâche
- DB_LINK : Identifiant de la base de données à auditer
- START_DATE : Date de début de la planification de la tâche
- END_DATE : Date d'expiration de la planification de la tâche
- INTERVAL : Intervalle entre chaque lancement

A chaque création, suppression ou modification effectuée dans cette table, des triggers répercutent la planification du lancement de la procédure « SP_LAUNCH_JOB » définie dans le paquetage « PKG_AUDIT_COMMUN » grâce aux procédures « CREATE_JOB » et « DROP_JOB » du paquetage « DBMS_SCHEDULER ».

Lors de son exécution, la procédure « SP_LAUNCH_JOB » va parcourir la table « T_AUDIT_JOB_ACTION » pour enchaîner le lancement des différents audits demandés pour la tâche planifiée. La table contient donc les champs suivants :

- AUDIT_JOB_ID : Identifiant de la tâche
- POS : Position d'exécution de l'action
- AUDIT_ID : Identifiant de l'audit à lancer
- AUDIT_REQ_ID : Identifiant de la requête de l'audit (si nous ne voulons pas lancer l'audit dans son intégralité)
- FORCED_LAUNCH : Lancement en mode forcé OUI/NON
- NIVEAU_LOG : Niveau de trace à utiliser pour le lancement

---

[23] DBMS_SCHEDULER : http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14258/d_sched.htm

[24] Cf. Figure 7 – Schéma des tables (page 26)

Ce mécanisme nous permet d'obtenir une grande souplesse de gestion puisqu'il est non seulement possible de planifier des lancements à des heures différentes suivant les environnements mais aussi d'en adapter totalement le contenu. De plus, les lancements étant gérés par l'ordonnanceur Oracle, les audits peuvent aussi s'effectuer en parallèle sur les différentes bases. Pour ne pas avoir d'erreur d'ordonnancement, les audits sont exécutés de façon séquentielle au sein de chaque lancement et une contrainte a été ajoutée afin d'empêcher d'avoir deux lancements simultanés sur la même base.

IV.1.C Conclusion

Grâce à l'utilisation d'Oracle EXPRESS, j'ai pu mettre en place DaPCol – Collect en moins de quatre mois et donc remplir mon objectif. Durant cette période, j'ai également pu préparer les différentes requêtes lors de la recette de la reprise de données et de l'interface eBS-FAP. J'ai ainsi mis en place 8 audits différents, 32 requêtes d'audits, 26 déclencheurs et 188 compléments. L'exécution automatique de l'ensemble de ces requêtes, nous a permis de disposer d'une vue d'ensemble de l'état des différents traitements. Mais afin de mettre en place l'ensemble de ces audits, une interface d'administration s'est avérée nécessaire.

IV.2 L'interface d'administration : DaPCol – Admin

Afin d'éviter au maximum de devoir intervenir directement sur la base de données pour mettre en place, modifier ou utiliser un audit, il était nécessaire de disposer d'une interface d'administration.

En plus de la base de données, Oracle fournit avec Oracle Express un client WEB permettant de développer rapidement des applications. J'ai donc mis en place l'interface d'administration grâce à cet utilitaire nommé « APEX » (Oracle Application Express)[25]. Pour ce faire, j'ai utilisé la version 4.0 qui permet de disposer de l'intégration d'AJAX (Asynchronous Javascript and XML) et des graphiques Adobe Flash[26].

IV.2.A Gestion des audits

La gestion des audits s'effectue à partir de quatre pages web regroupées sous l'onglet « Audits ».

La première (cf. Figure 8 – Liste des audits) permet d'obtenir la liste de l'ensemble des audits.

Figure 8 – Liste des audits

En cliquant sur l'icône de modification, nous avons accès à la deuxième page web (cf. Figure 9 – Modification d'un audit).

[25] Site officiel de l'APEX : http://apex.oracle.com/

[26] Building Charts, Gantts and Maps with Oracle Application Express 4.0 : http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/apex/r40/apexcharts/apexcharts_ll.htm

Cette page est composée de trois zones :

- La première zone : « Modification de l'audit », permet de renseigner les informations de la table « T_AUDIT ». C'est dans cette zone que l'utilisateur définit le nom et la description de l'audit, le mode d'audit et la table dans laquelle seront stockées les données de l'audit. En mode « ajout/append », les données de chaque audit sont sauvegardées ; en mode « remplacement/replace » un dé-doublonnage des données est effectué pour chaque audit afin de ne conserver que les données les plus récentes.
- La deuxième zone : « Liste des requêtes » fournit la liste complète des requêtes qui composent l'audit. Il est à noter que si une erreur est survenue lors de l'exécution d'une requête ou d'un déclencheur, la zone correspondante sera affichée en orange.
- La dernière zone : « Liste des compléments » fournit la liste complète des compléments permettant d'effectuer une analyse plus détaillée. Il est à noter qu'ici aussi si une erreur est survenue lors de l'exécution d'un des compléments, celui-ci sera alors affiché en orange.

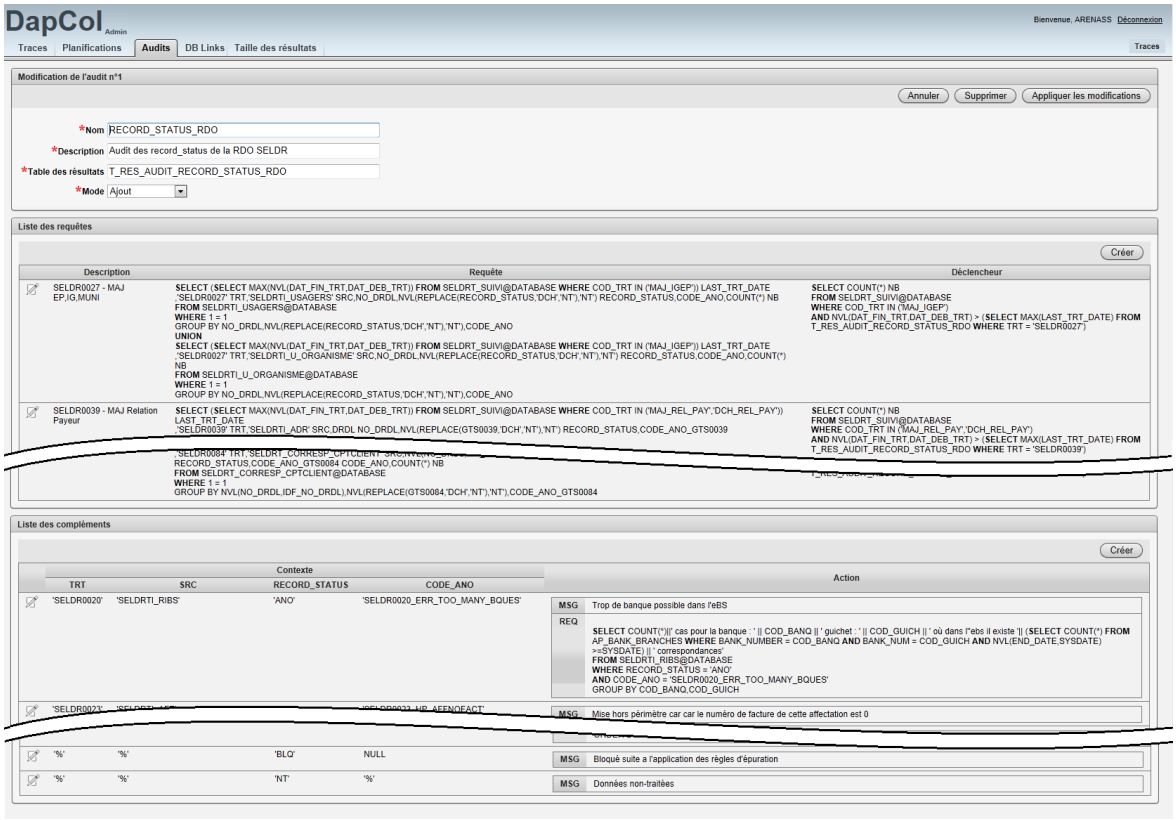

Figure 9 – Modification d'un audit

En cliquant sur l'icône de modification zone : « Liste des requêtes », nous avons accès à la troisième page de gestion des audits (cf. Figure 10 – Modification d'une requête d'audit).

Cette page permet de créer ou de modifier les requêtes (ainsi que leur déclencheur) qui composent les audits.

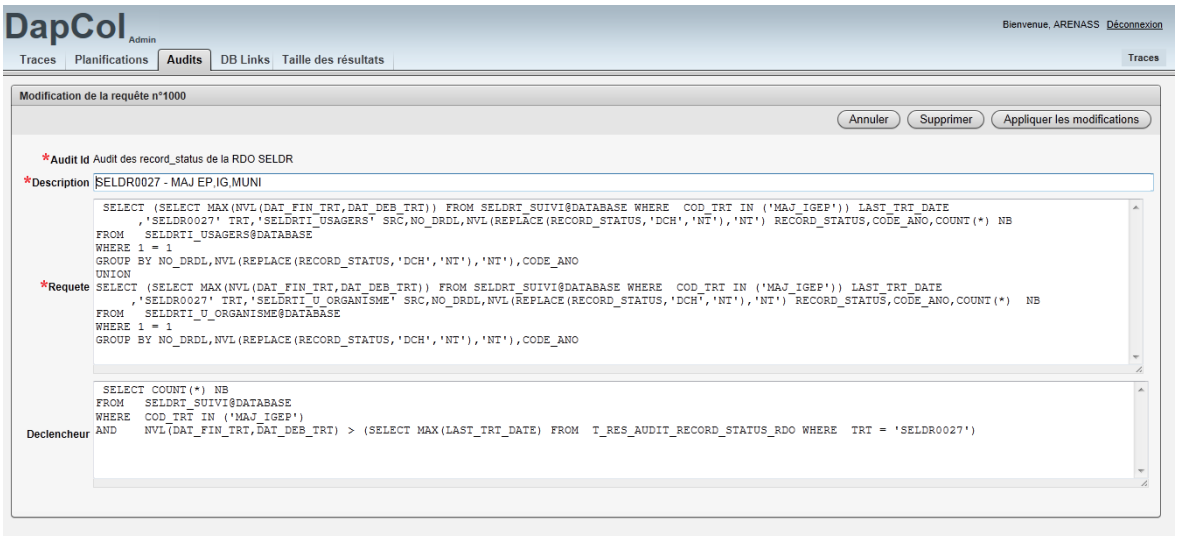

Figure 10 – Modification d'une requête d'audit

Enfin, en cliquant sur l'icône de modification zone : « Liste des compléments », nous avons accès à la dernière page de gestion des audits (cf. Figure 11 – Modification d'un complément d'audit).

Cette page permet de créer ou de modifier les compléments à exécuter lors des audits (cf. §IV.1.B.b.3 p35 & IV.1.B.b.3.4 p38).

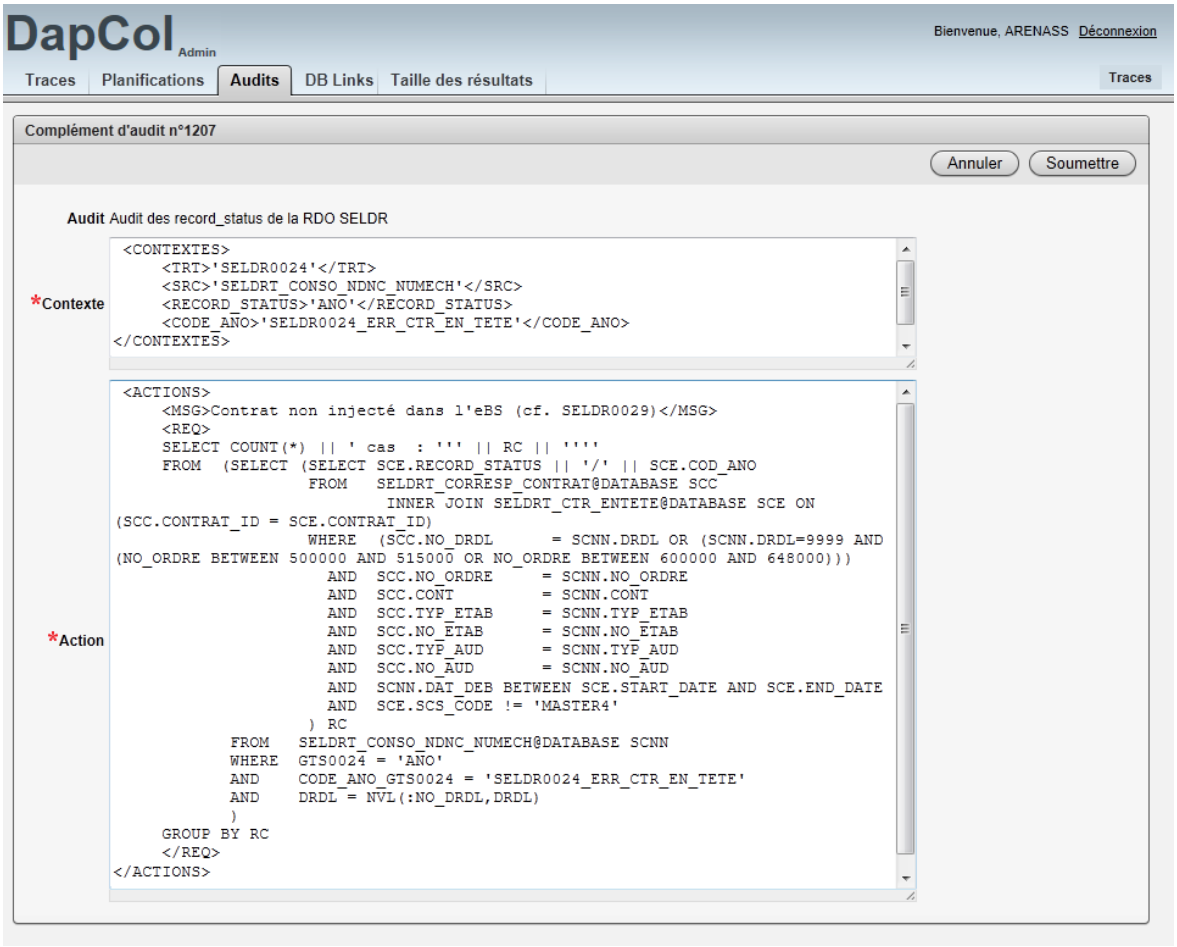

Figure 11 – Modification d'un complément d'audit

IV.2.B Gestion de la planification des audits

La gestion de la planification des audits s'effectue à partir de trois pages web regroupées sous l'onglet « Planifications ».

La première (cf. Figure 12 – Liste des planifications) permet d'obtenir la liste de l'ensemble des planifications.

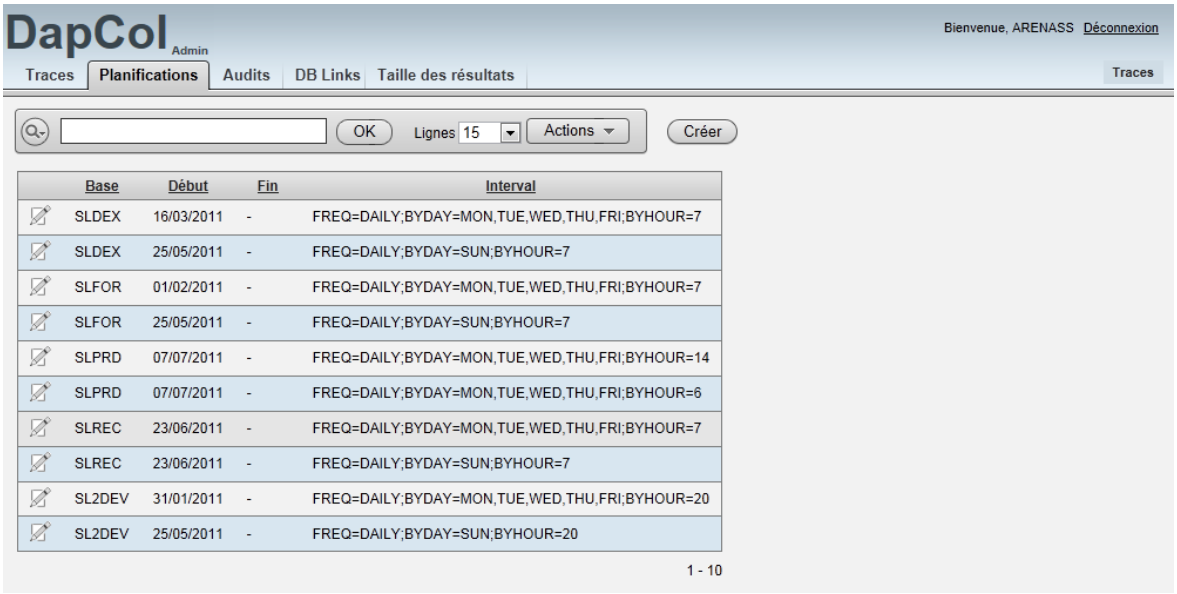

Figure 12 – Liste des planifications

En cliquant sur l'icône de modification , nous avons accès à la deuxième page web (cf. Figure 13 – Modification d'une planification). Cette page est composée de deux zones. La première zone : « Modification de la planification », permet de renseigner les informations de la table « T_AUDIT_JOB » (Base de données auditée, date de début, date de fin et intervalle d'audit). La deuxième zone : « Liste des actions » fournit la liste complète des actions qui seront lancées à l'exécution de cette planification.

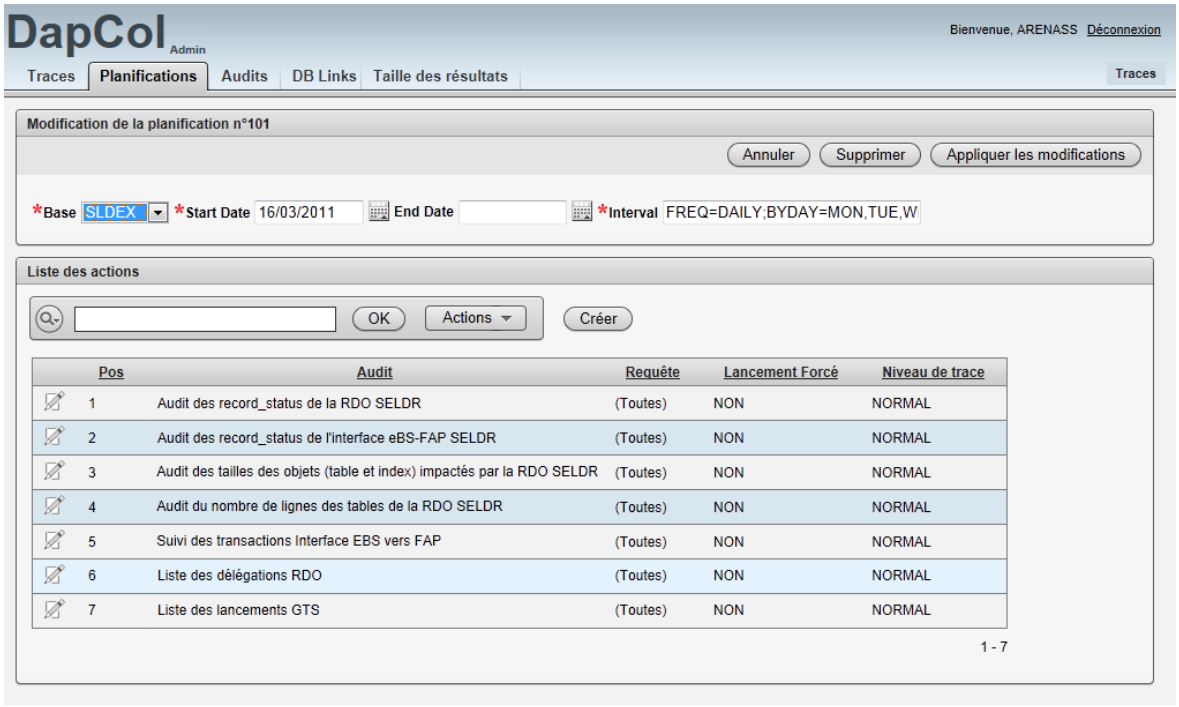

Figure 13 – Modification d'une planification

En cliquant sur l'icône de modification zone : « Liste des actions », nous avons accès à la troisième page de la gestion des planifications (cf. Figure 14 – Modification des actions d'une planification).

Cette page permet de créer ou de modifier les audits qui composent la planification. Ainsi nous pouvons planifier tout ou partie d'un audit, choisir ou non de tester les déclencheurs ainsi que le niveau de trace à utiliser.

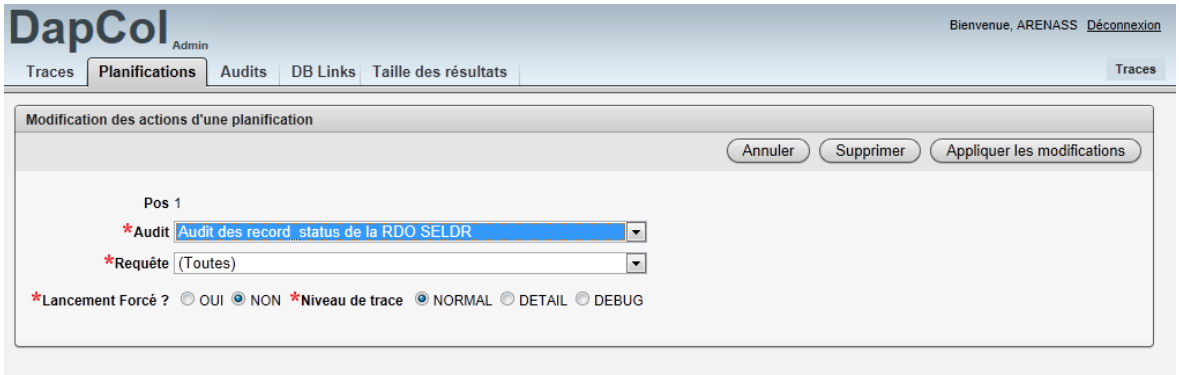

Figure 14 – Modification des actions d'une planification

IV.2.C Gestion des DB links

La gestion des DB links s'effectue à partir d'une page web (cf. Figure 15 – Liste des DB Links) sous l'onglet « DB Link ».

A partir de cette page nous pouvons définir le nom des bases de données à auditer, l'identifiant, le mot de passe et le TNS à utiliser pour créer les liens vers celles-ci.

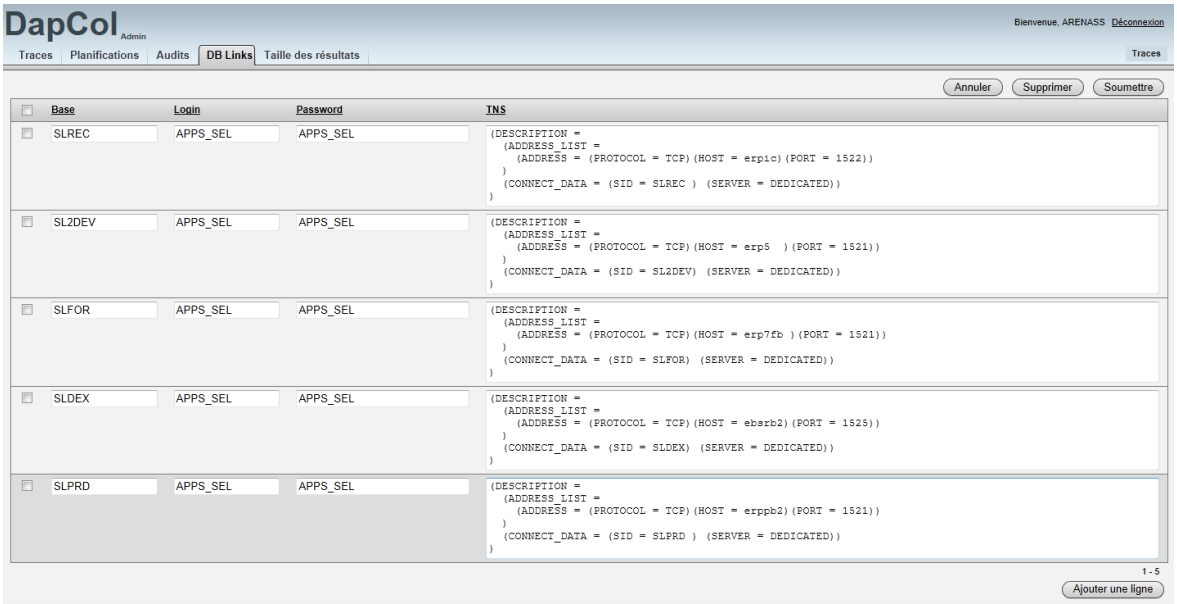

Figure 15 – Liste des DB Links

IV.2.D Suivi de l'exécution des audits et exécution manuelle

La page d'accueil de l'interface d'administration (cf. Figure 16 – Suivi des exécutions) se compose de quatre zones permettant d'avoir une vision synthétique sur l'exécution des différents audits mais aussi d'interagir directement avec le mécanisme de collecte.

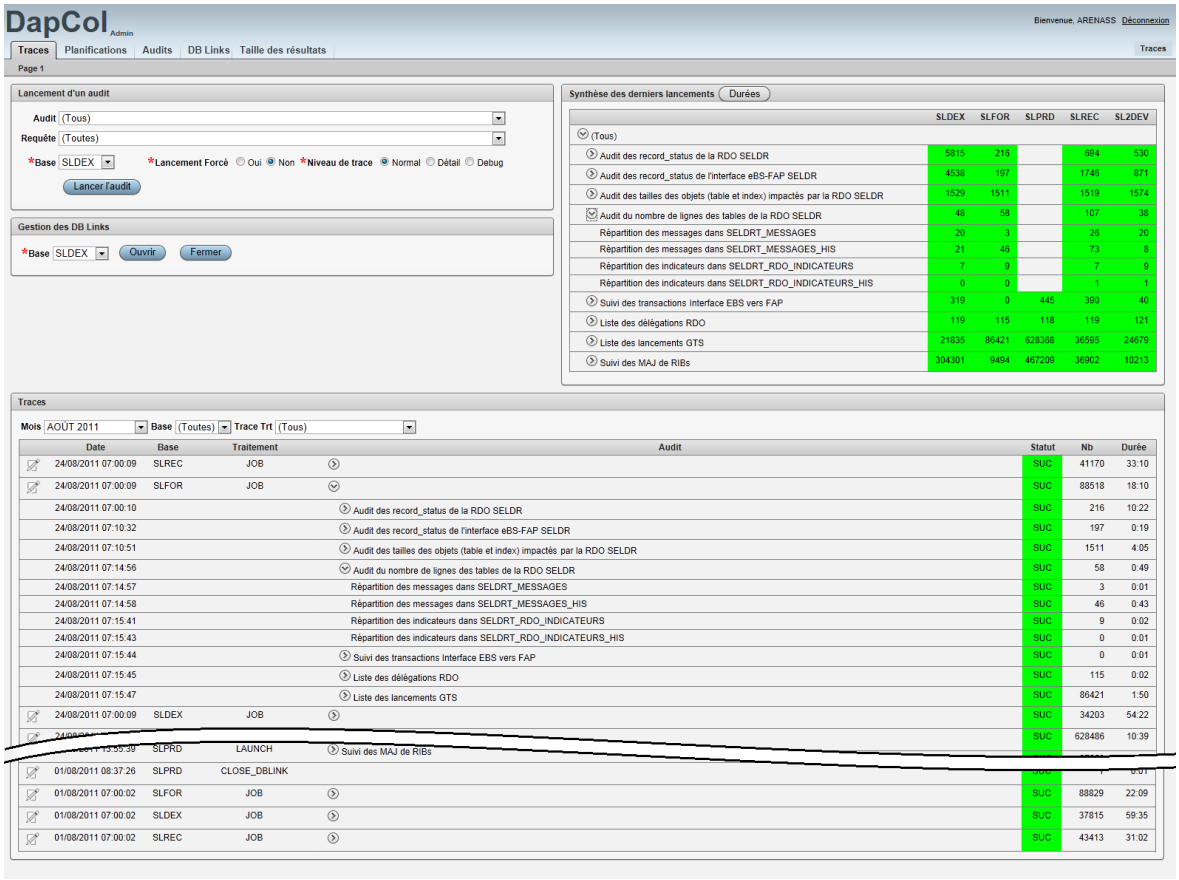

Figure 16 – Suivi des exécutions

La première zone : « Lancement des audits » permet de lancer instantanément un audit sans passer par le mécanisme de planification. Cette fonctionnalité permet non seulement la mise en place plus rapide des nouveaux audits mais aussi de vérifier après les opérations de rattrapage que les corrections effectuées ont bien eu les effets escomptés.

La deuxième zone : « Gestion des DB links » permet d'ouvrir ou de fermer manuellement les différents DB Links.

La troisième zone : « Synthèse des derniers lancements » présente sous forme synthétique l'état d'exécution du dernier lancement de chaque audit (et de chacune de ses requêtes) pour chacune des bases de données auditées. Cette synthèse peut être vue en nombre d'enregistrements remontés mais aussi en durée d'exécution. L'affichage des données pour

les requêtes de chaque audit s'effectue simplement en cliquant sur l'icône ce qui dévoile les lignes des requêtes grâce à un code JavaScript ajouté à cette page web.

Enfin la dernière zone : « Traces » permet d'avoir la liste de l'ensemble des procédures exécutées par le mécanisme de collecte.

On y retrouve principalement :

- les « JOB » qui correspondent aux lancements d'audits effectués par le mécanisme de planification.
- les « LAUNCH » qui correspondent aux lancements d'audits effectués manuellement depuis cette page web.

mais aussi toutes les procédures d'administration et de maintenance ayant implémenté les mécanismes de trace décrits en §IV.1.B.b.3.4 Gestion des traces et des erreurs (p38).

Pour chacune des traces nous avons : la date de lancement, la base de données concernée, le code du traitement effectué, le statut du traitement, sa durée et éventuellement l'audit concerné ainsi que le nombre d'enregistrements traités. De la même façon que précédemment, un clic sur l'icône permet d'avoir accès aux sous-traitements et d'obtenir ainsi un niveau de détail supplémentaire.

Enfin en cliquant sur l'icône, nous avons accès à une page web (cf. Figure 17 – Exemple de traces) contenant l'ensemble de la trace générée par le traitement.

Date	Traitement	Base	Audit	Requête	Niveau Log	Statut	Nb	Duree
24-08-2011 06:00:03	JOB	SLPRD	(NULL)	(NULL)	0	SUC	628486	10min39s

1

Code	Texte
1s	SP_LAUNCH_JOB ----------
1s	SP_LAUNCH_JOB Début du lancement du job : 161
1s	SP_LAUNCH_JOB ----------
2s	SP_LAUNCH_AUDIT ----------
2s	SP_LAUNCH_AUDIT Début de l'audit : DATE_REPRISE_RDO (Liste des délégations RDO) sur SLPRD
2s	SP_LAUNCH_AUDIT ----------
5s	SP_LAUNCH_AUDIT ----------
5s	SP_LAUNCH_AUDIT Requête des délégations RDO
1min15s	SP_LAUNCH_AUDIT ----------
1min15s	SP_LAUNCH_AUDIT Fin de l'audit : DATE_REPRISE_RDO (Liste des délégations RDO) sur SLPRD
1min15s	SP_LAUNCH_AUDIT ----------
1min15s	SP_LAUNCH_AUDIT ----------
1min15s	SP_LAUNCH_AUDIT Début de l'audit : GTS (Liste des lancements GTS) sur SLPRD
1min15s	SP_LAUNCH_AUDIT ----------
1min47s	SP_LAUNCH_AUDIT ----------
1min47s	SP_LAUNCH_AUDIT Requête des lancements GTS
10min13s	SP_LAUNCH_AUDIT ----------
10min13s	SP_LAUNCH_AUDIT Fin de l'audit : GTS (Liste des lancements GTS) sur SLPRD
10min13s	SP_LAUNCH_AUDIT ----------
10min39s	SP_LAUNCH_JOB ----------
10min39s	SP_LAUNCH_JOB Fin du lancement du job : 161
10min39s	SP_LAUNCH_JOB ----------

1

Fermer

Figure 17 – Exemple de traces

IV.2.E Taille des résultats

Les bases de données Oracle Express étant limitées en terme de stockage de données utilisateurs, j'ai donc créé une page web (cf. Figure 18 – Tailles des résultats des audits) nous permettant d'avoir une vue rapide de la taille des données occupées par le mécanisme d'audit. Sur cette page nous retrouvons l'espace de stockage total utilisée ainsi que deux graphiques. Le premier graphique montre la taille dédiée à chacun des audits ainsi qu'au mécanisme de trace. Le deuxième permet en sélectionnant un élément du premier graphique, de voir le détail de l'espace utilisé pour chaque table et index de l'élément.

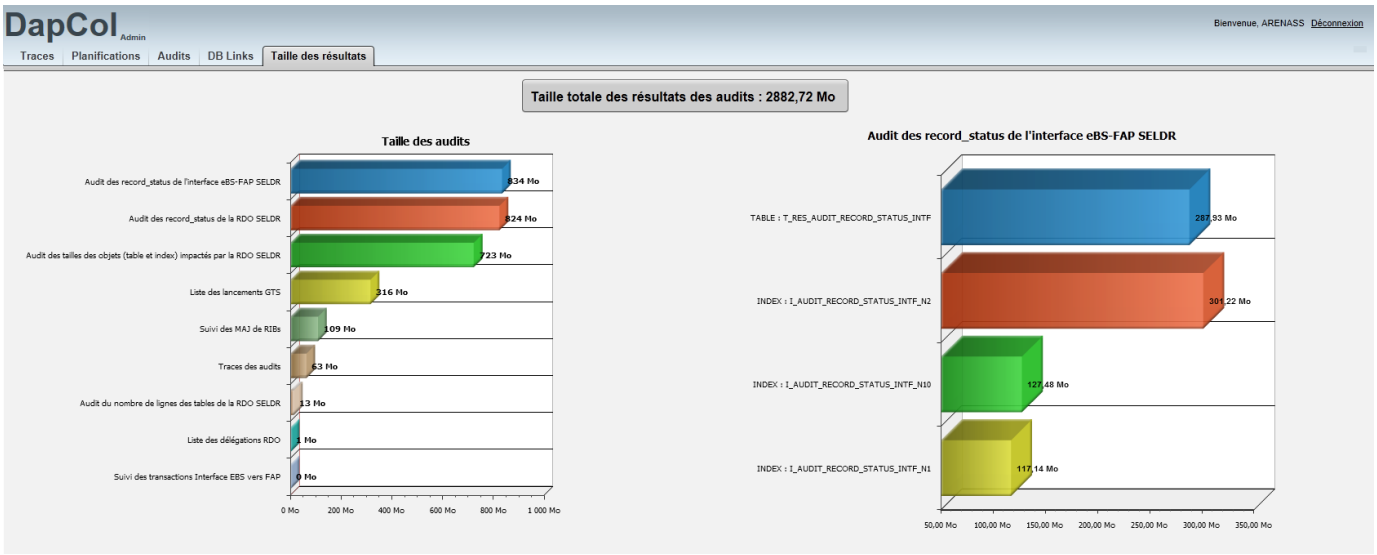

Figure 18 – Tailles des résultats des audits

IV.3 La génération de rapport : DaPCol – Reports

La troisième partie de ce mémoire se focalise sur la génération de rapports basés sur les résultats obtenus grâce à DaPCol – Collect.
Pour la génération des rapports, les objectifs étaient les suivants :

- Pouvoir disposer de modèles prédéfinis permettant de générer automatiquement des rapports le matin, nous permettant d'avoir une analyse rapide de l'évolution des données suite à l'exécution des traitements de nuit.
- Pouvoir générer des rapports complémentaires dans la journée.
- Pouvoir disposer d'un symbolique et d'une mise en forme explicite, de façon à simplifier la lecture des rapports.
- Pouvoir être utilisé par plusieurs personnes sans nécessiter de connaissances poussées sur le fonctionnement de DaPCol.

Au lieu de développer entièrement un outil de génération de rapport, nous avons choisi de nous appuyer sur JasperReport. Les défis de cette partie ont été, d'une part, de s'approprier ce nouvel outil, et d'autre part de trouver les solutions les plus optimums et permettant de mutualiser au maximum le travail effectué pour mettre en place chaque rapport.
JasperReport[27] est un projet Java Open Source sous licence LGPLv3[28] dédié à la génération de rapport. Il permet de générer des rapports à condition de lui fournir : un modèle de document, les données à utiliser pour le rapport et l'ensemble des paramètres nécessaires au modèle. Bien que les modèles de documents puissent être mis en place à partir de n'importe quel éditeur de texte (puisque ce sont des fichiers XML) JasperSoft fournit iReport Designer[29] : une interface utilisateur permettant de définir plus facilement des modèles de documents. Grâce à iReport, nous pouvons rapidement mettre en place un modèle de rapport contenant aussi bien : des tableaux, des tableaux croisés, des graphiques et même des images. Il est aussi possible de définir des mises en forme conditionnelles afin de permettre de faire mieux ressortir certaines données.

---

[27] JasperReport Home Page : http://jasperforge.org/projects/jasperreports

[28] Licence GPL V3 : http://www.gnu.org/licenses/lgpl-3.0.html

[29] iReport Home Page : http://jasperforge.org/projects/ireport

Pour minimiser le nombre de modèles à définir, générer les rapports automatiquement et permettre à des utilisateurs de générer des rapports complémentaires, nous avons créé une nouvelle interface « DaPCol – Reports » permettant de générer l'ensemble des rapports escomptés.

Oracle Express ne permettant pas d'utiliser du code Java et JasperReport étant un projet Java, nous avons donc naturellement choisi de développer cette interface complètement en Java suivant l'architecture présentée sur la Figure 19.

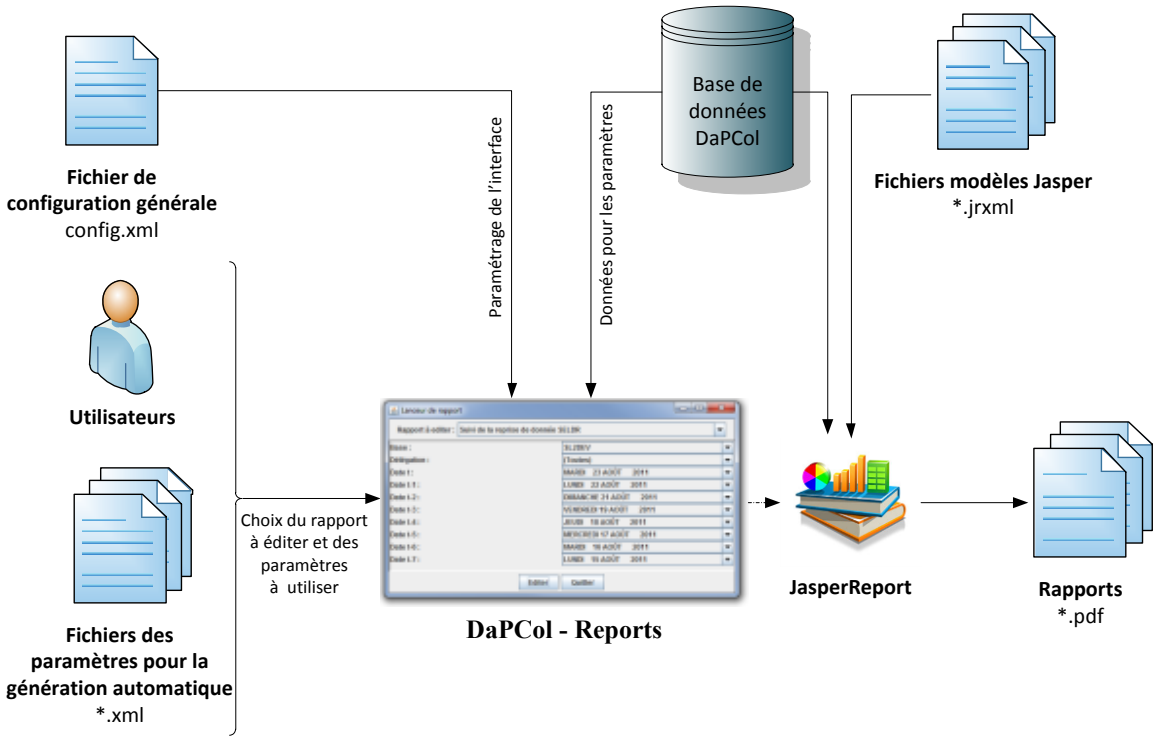

Figure 19 – Architecture DaPCol - Reports

Un fichier de configuration générale permet de définir pour chaque rapport, le modèle Jaspers à utiliser, les requêtes nécessaires pour récupérer les données sur la base de données DaPCol et les paramètres que l'utilisateur va pouvoir définir. Une fois que l'utilisateur aura sélectionné les paramètres du rapport, l'ensemble des informations sera transmis au moteur JasperReport qui générera le rapport sous forme d'un document pdf. Pour les exécutions automatiques, la saisie utilisateur sera remplacée par un fichier de définition des paramètres.

IV.3.A Le fichier de configuration générale

Le fichier de configuration générale `config.xml` est l'élément central de cette interface.
En effet, c'est celui-ci qui regroupe l'ensemble des informations permettant d'exploiter les modèles Jasper afin de générer les différents rapports.
Ce fichier XML contient :

- Les paramètres de connexion à la base de données des audits (Pilote à utiliser, chaine de connexion, identifiant et mot de passe).
- Les paramètres de configuration pour chaque rapport
    - L'identifiant du rapport
    - Le titre du rapport
    - Le modèle Jasper à utiliser
    - La liste des paramètres définis dans le modèle Jasper
    - La requête principale
    - Les différentes requêtes nécessaires pour les graphiques

IV.3.B Les classes de l'interface

DaPCol – Reports est basé sur 8 classes Java :

- Les classes Trace et Context permettent comme dans le mécanisme de collecte[30] d'obtenir une trace plus ou moins détaillée lors de l'exécution de l'interface.

[30] Cf. §IV.1.B.b.3.4 : Gestion des traces et des erreurs (page 32)

- La classe Main, classe d'entrée du programme.
  Cette classe ne contient qu'une seule procédure `main` qui a pour but de traiter les paramètres de lancement, charger le fichier de configuration et lancer une instance de la classe IHM (interface homme machine).

- La classe IHM[31] est la classe création et de gestion de l'interface. Elle a pour but de créer une fenêtre composée de trois zones :
  - La première zone contient une seule liste déroulante avec l'intégralité des rapports disponibles.
  - La deuxième zone contient l'ensemble des paramètres devant être saisis par l'utilisateur, elle est contextualisée en fonction de la première zone.
  - la dernière zone contient un bouton pour générer un nouveau rapport et un autre pour quitter l'interface.

  La génération des rapports se fait dans le répertoire temporaire par défaut. Ils sont ensuite automatiquement ouverts avec le programme dédié au type de document généré. Pour l'instant, l'interface ne génère que des documents de type PDF mais JasperReport propose en natif de générer aussi des documents de type Excel, CSV HTML ou XML.

- Les classes JComboBoxSQL[32], JComboBoxDependent[33] et Item permettent de gérer les paramètres de type liste de valeurs basés soit sur une requête soit sur une autre liste de valeurs.

IV.3.C Gestion des paramètres utilisateurs

Au sein de DaPCol – Report, la gestion des paramètres est calquée sur la gestion des paramètres définis dans le modèle Jasper à utiliser.

---

[31] Cf. Annexe 8 – DaPCol-Reports : Classe IHM (page 115)

[32] Cf. Annexe 9 – DaPCol-Reports : Classe JComboBoxSQL (page 134)

[33] Cf. Annexe 10 – DaPCol-Reports : Classe JComboBoxDependent (page 137)

Nous avons simplement rajouté des fonctionnalités supplémentaires non-gérées en natif et correspondant à nos besoins.

- La possibilité de définir un paramètre Jasper non visible avec une valeur par défaut pour mutualiser les modèles.
- La possibilité de proposer une liste de valeur pour chaque paramètre. Cette liste peut être soit indépendante, soit dépendante d'un ou plusieurs autres paramètres.
- Si plusieurs paramètres dépendent d'une même liste de valeurs mais avec une notion d'ordre (Exemple : Date J1 < Date J2 < Date J), plutôt que de solliciter plusieurs fois la base de données, un mécanisme permet aux paramètres dépendants (ici Date J2 et Date J3) d'exploiter directement la liste de valeurs de leur référent. Ceci permet de réduire les sollicitations à la base de données DaPCol et d'améliorer la réactivité de l'interface.

IV.3.D Fichiers des paramètres pour la génération automatique

Afin de pouvoir disposer automatiquement de certains rapports sans nécessiter une quelconque intervention humaine, il a fallu mettre en place un mécanisme d'automatisation. Ce mécanisme a été mis en place simplement en permettant à l'interface de recevoir en paramètre un fichier XML contenant les paramètres du rapport demandé.
Ces fichiers XML de génération automatique doivent contenir :

- L'identifiant du type rapport à générer
- Les paramètres à définir (Pour les listes de valeur, si le paramètre n'est pas défini alors celui-ci prendra la première valeur de la liste)
- Le chemin et le nom d'enregistrement du rapport.

IV.3.E Définition du modèle Jasper

Un des besoins principaux est de fournir un suivi de l'évolution dans le temps des données traitées et plus particulièrement des anomalies. J'ai envisagé, dans un premier temps, l'utilisation des tableaux croisés mais il s'est rapidement avéré que cette solution ne permettait pas de répondre à notre besoin. En effet, pour utiliser les tableaux croisés il faut une correspondance exact entre les données mises en colonne et celles mises en ligne. Or, de par les mécanismes de déclencheurs et de subdivisions des audits (cf. §IV.1.B.b.1) nous n'avons pas forcément pour chaque lancement d'audit l'intégralité des requêtes qui sont exécutées :

Par exemple, supposons un audit composé des deux requêtes REQ1 et REQ2.
Le jour J1 à 22h00 les deux requêtes sont exécutées grâce au mécanisme de planification.
Suite à diverses actions effectuées sur la base de données auditée, nous décidons de ré-exécuter la requête REQ1 le jour J2 à 12h00 par lancement manuel.
Le jour J2 à 22h00 lors de l'exécution de la planification, l'analyse des déclencheurs n'exécute que la REQ2.
Enfin le jour J3 les deux requêtes sont exécutées grâce au mécanisme de planification.
Dans ce contexte si nous utilisions les tableaux croisés le rapport obtenu serait comme-ceci :

		J1 22h	J2 12h	J2 22h	J3 22h
Comptage remonté par la requête REQ1	Anomalie	150	100	*Vide*	250
	Succès	2000	2050	*Vide*	4500
Comptage remonté par la requête REQ2	Anomalie	200	*Vide*	100	170
	Succès	3000	*Vide*	3100	5100

Alors que le rapport escompté est :

		J1	J2	J3
Comptage remonté par la requête REQ1	Anomalie	150	100	250
	Succès	2000	2050	4500
Comptage remonté par la requête REQ2	Anomalie	200	100	170
	Succès	3000	3100	5100

De plus, nous pourrions avoir besoin de voir l'évolution des données, non pas pour chaque date d'audit mais par semaine ou par mois. Dans ce cas aussi les tableaux croisés ne permettent pas d'obtenir le résultat escompté.

Nous avons donc choisi de générer nos rapports à partir d'un modèle sous forme de tableau simple avec un nombre de colonne fixe. Dans la mesure où le nombre de colonne fixe ne permet pas de retracer l'historique complet de l'évolution des données, nous avons exploité la zone « résumé » du modèle afin de fournir une série de graphique complémentaire (cf. Figure 20 – Modèle de rapport sur 8 dates et Figure 21 – Modèle de rapport sur 3 dates avec les compléments pour la date la plus récente).

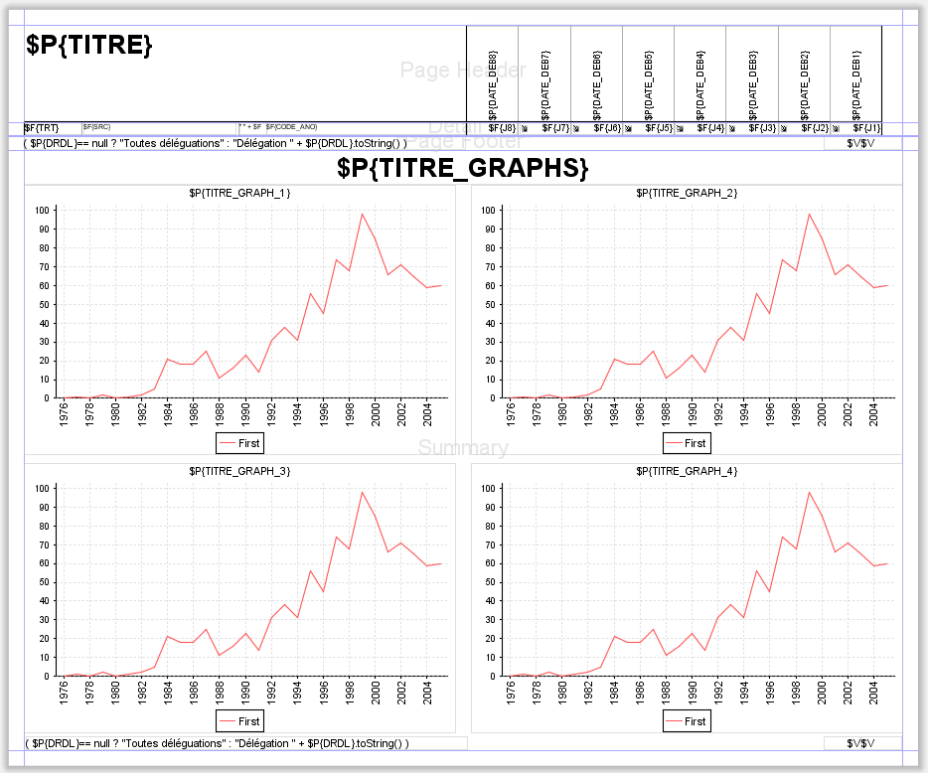

Figure 20 – Modèle de rapport sur 8 dates

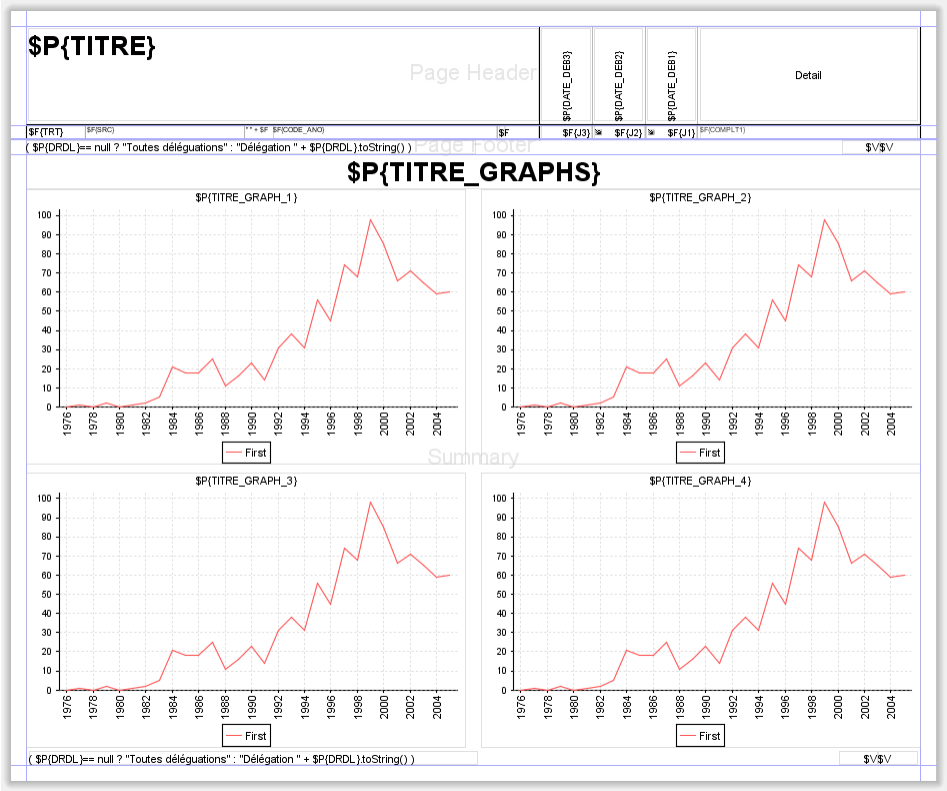

Figure 21 – Modèle de rapport sur 3 dates avec les compléments pour la date la plus récente

L'utilisation de JasperReport nous permet ici non seulement de répondre à notre besoin actuel, mais aussi de nous laisser une grande marge de manœuvre quant à la typologie de rapport réalisable dans le futur.

IV.3.F Requêtes d'alimentation du rapport

Ne pas utiliser les tableaux croisés disponibles dans JasperReport a un impact direct sur la requête qui alimente le rapport. En effet, avec un tableau croisé dynamique cette requête aurait été simple.

Exemple :

```
SELECT AUDIT_EXTRACT_DATE,REQUETE,ELEMENT_COMPTE,NOMBRE
FROM   TABLE_RESULTAT_AUDIT
WHERE  BASE_AUDITE = 'MaBase';
```

Ce serait ensuite JasperReport qui aurait consolidé les données pour les présenter sous forme de tableaux.

Mais puisqu'il a été choisi un modèle tabulaire simple, il faut impérativement que la requête remonte les données directement sous la forme définitive, c'est à dire : `REQUETE, ELEMENT_COMPTE, NOMBRE_DATE1, NOMBRE_DATE2, NOMBRE_DATE3, …`

Face à ce genre de situation, nous pourrions être tentés d'essayer d'écrire la requête comme une boucle imbriquée : pour chaque *n*-uplet `REQUETE, ELEMENT_COMPTE,` rechercher le `NOMBRE` pour chacune des dates demandées. La problématique de cette approche consiste dans le fait qu'elle parcourt une première fois les données pour identifier les différents couples `REQUETE,ELEMENT_COMPTE` puis qu'elle doit ensuite le refaire autant de fois qu'il y a de date.

Notre exemple deviendrait alors :

```
SELECT  REQUETE,ELEMENT_COMPTE,
        (SELECT NOMBRE
         FROM   TABLE RESULTAT AUDIT T2
         WHERE  T1.BASE AUDITE          = T2.BASE AUDITE
         AND    T1.REQUETE              = T2.REQUETE
         AND    T1.ELEMENT_COMPTE       = T2.ELEMENT_COMPTE
         AND    T2. AUDIT_EXTRACT_DATE = Date1) NOMBRE_DATE1,
        (SELECT NOMBRE
         FROM   TABLE_RESULTAT_AUDIT T2
         WHERE  T1.BASE_AUDITE          = T2.BASE_AUDITE
         AND    T1.REQUETE              = T2.REQUETE
         AND    T1.ELEMENT_COMPTE       = T2.ELEMENT_COMPTE
         AND    T2.AUDIT_EXTRACT_DATE = Date2) NOMBRE_DATE2,
         ...
FROM   (SELECT DISTINCT REQUETE,ELEMENT_COMPTE
        FROM   TABLE_RESULTAT_AUDIT
        WHERE  BASE_AUDITE = 'MaBase'
        AND    AUDIT_EXTRACT_DATE IN (Date1,Date2,Date3...)) T1;
```

Comptage pour la date 1

Comptage pour la date 2

Liste des éléments

Cependant, cette requête ne tient pas compte du fait que nous ne disposons pas forcément d'un audit pour chacune des dates demandées pour le rapport. Il faut alors chercher les données remontées lors de l'exécution de la requête précédent la date demandée.

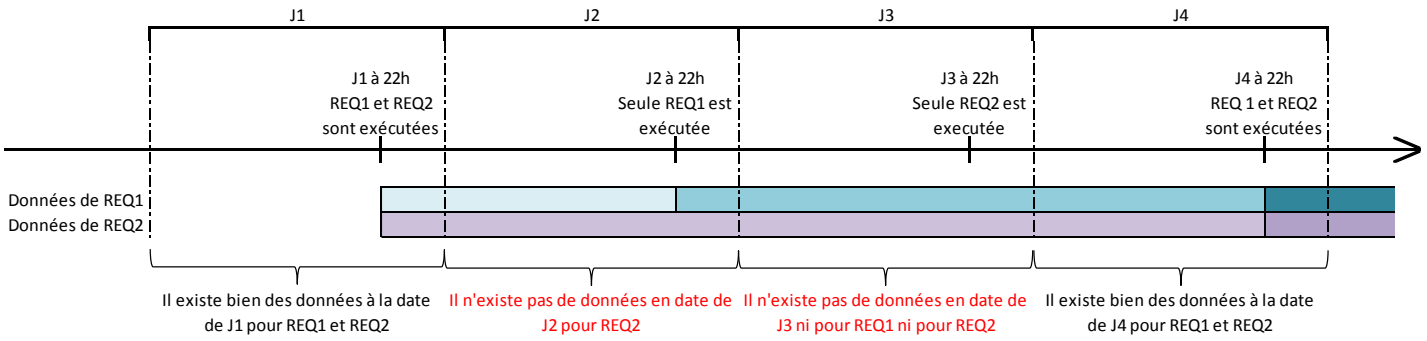

Figure 22 – Exemple d'absence d'exécution de requêtes pour certaines dates

De plus, il peut très bien y avoir plusieurs exécutions de l'audit dans la même journée, il faut donc, dans ce cas, conserver les dernières données de chaque date demandée.

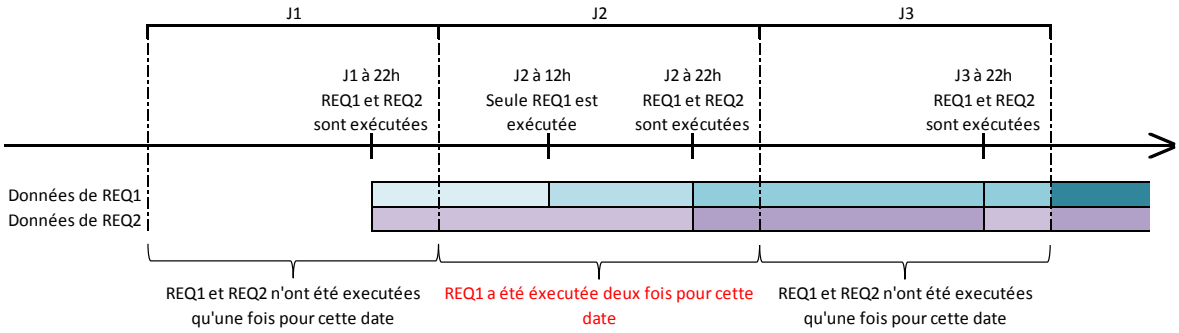

Figure 23 – Exemple de multiples exécutions de requêtes pour certaines dates

Pour tenir compte, des deux dernières remarques, il faudrait alors complexifier la requête :

```
SELECT  REQUETE,ELEMENT_COMPTE,
        (SELECT NOMBRE
         FROM   TABLE_RESULTAT_AUDIT T2
         WHERE  T1.BASE_AUDITE          = T2.BASE_AUDITE
         AND    T1.REQUETE              = T2.REQUETE
         AND    T1.ELEMENT_COMPTE       = T2.ELEMENT_COMPTE
         AND    T2.AUDIT_EXTRACT_DATE < Date1+1
         AND    NOT EXISTS (SELECT 1
                            FROM   TABLE_RESULTAT_AUDIT T3
                            WHERE  T2.BASE_AUDITE          = T3.BASE_AUDITE
                            AND    T2.REQUETE              = T3.REQUETE
                            AND    T3.AUDIT_EXTRACT_DATE < Date1+1
                            AND    T3.AUDIT_EXTRACT_DATE > T2.AUDIT_EXTRACT_DATE)
        ) NOMBRE_DATE1,
        (SELECT NOMBRE
         FROM   TABLE_RESULTAT_AUDIT T2
         WHERE  T1.BASE_AUDITE          = T2.BASE_AUDITE
         AND    T1.REQUETE              = T2.REQUETE
         AND    T1.ELEMENT_COMPTE       = T2.ELEMENT_COMPTE
         AND    T2.AUDIT_EXTRACT_DATE < Date2+1
         AND    NOT EXISTS (SELECT 1
                            FROM   TABLE_RESULTAT_AUDIT T3
                            WHERE  T2.BASE_AUDITE          = T3.BASE_AUDITE
                            AND    T2.REQUETE              = T3.REQUETE
                            AND    T3.AUDIT_EXTRACT_DATE < Date1+1
                            AND    T3.AUDIT_EXTRACT_DATE > T2.AUDIT_EXTRACT_DATE)
        ) NOMBRE_DATE2,
        ...
FROM    (SELECT DISTINCT REQUETE,ELEMENT_COMPTE
         FROM   TABLE_RESULTAT_AUDIT
         WHERE  BASE_AUDITE = 'MaBase'
         AND    AUDIT_EXTRACT_DATE IN (Date1,Date2,Date3...)) T1;
```

Comptage pour la date 1

Comptage pour la date 2

Liste des éléments

Malheureusement, même avec de bons index, le temps d'exécution de cette requête va croître, de façon exponentielle, en fonction du nombre de données présentes dans la table. En effet, pour la partie « Listes des éléments », nous devons parcourir l'ensemble des éléments de la table, nous avons donc une complexité en O(n). Par ailleurs, pour chaque partie « Comptage pour la date x », nous devons parcourir les données pour trouver celles extraites avant la date demandée et parcourir, de nouveau, les données pour vérifier que ce sont bien les données les plus proches de la date demandée. Nous avons donc une

complexité en $O(n^2)$. Etant donné que chaque partie « Comptage pour la date x » doit s'exécuter pour chaque élément de la partie « Listes des éléments » nous avons, au final, une complexité en $O(n^3)$.

Afin de pouvoir optimiser cette requête, il faut pouvoir déterminer plus facilement quelles sont les données valides pour une date demandée et éviter ainsi les clauses `NOT EXISTS`.
J'ai donc ajouté au mécanisme de collecte la définition d'un nouveau champ de gestion qui contiendra la date de fin de validité des données : `AUDIT_VALIDITY_END_DATE` (cf §IV.1.B.b.2 p34). Cette date correspond à la date après laquelle la même requête du même audit a été relancée sur la même base. Chaque donnée est maintenant bornée par la date à laquelle elle a été extraite `AUDIT_EXTRACT_DATE` et celle à laquelle elle n'est plus valide `AUDIT_VALIDITY_END_DATE`.
Avec ce nouveau champ, nous pouvons exploiter les fonctionnalités d'agrégation disponibles dans le langage SQL pour réduire la requête de la façon suivante :

```
SELECT REQUETE,ELEMENT_COMPTE,
       SUM(CASE WHEN Date1 + 1 - 1/(24*60*60)
                     BETWEEN AUDIT_EXTRACT_DATE
                     AND NVL(AUDIT_VALIDITY_END_DATE,SYSDATE+1)
                THEN NOMBRE
            ELSE 0
        END)  NOMBRE_DATE1,
       SUM(CASE WHEN Date2 + 1 - 1/(24*60*60)
                      BETWEEN AUDIT EXTRACT DATE
                      AND NVL(AUDIT_VALIDITY_END_DATE,SYSDATE+1)
                THEN NOMBRE
            ELSE 0
        END)  NOMBRE_DATE2,
       ...
FROM   TABLE RESULTAT AUDIT
WHERE  BASE_AUDITE = 'MaBase'
AND    NOT NVL(AUDIT_VALIDITY_END_DATE,SYSDATE+1) < MIN(Date1,Date2,Date3...)
AND    NOT AUDIT_EXTRACT_DATE                     > MAX(Date1,Date2,Date3...)
GROUP BY REQUETE,ELEMENT_COMPTE;
```

Comptage pour la date 1

Comptage pour la date 2

Nous parcourrons donc une seule fois les données avec une complexité en O(n).
Le plan d'exécution de la requête sera dans le pire des cas un `TABLE_ACCESS_FULL` de la table et deviendra avec le bon index un `INDEX_RANGE_SCAN`.

V Exploitation

Le déploiement des 86 délégations de la Sacem sur le nouveau système SELECT s'est effectué de la façon suivante :

- Juin 2008 : première recette de la reprise de données
- Septembre 2009 : déploiement des trois premières délégations pilotes (les 04,18 et 25 septembre)
- Juin 2010 : déploiement de trois nouvelles délégations pilotes (les 11,18 et 25 juin)
- Septembre 2010 : déploiement de 5 délégations (2 le 10, 2 le 17 et 1 le 24 septembre)
- D'octobre à début décembre 2010 : déploiement de 25 délégations (2 déploiements par semaine : un le lundi et un le mercredi avec jusqu'à 2 délégations par déploiement).
- De janvier 2011 à mai 2011 : déploiement des 50 délégations restantes sur le même rythme.

Nous avons aussi, durant l'été 2010, effectué des tests de montée en charge sur un environnement de recette afin de valider le fonctionnement correct de l'application avec les données des 86 délégations.

En 2008, avant la mise en place de DaPCol, lorsque nous avons effectué la première recette de la reprise de données d'une délégation sur le nouveau système SELECT, il nous avait fallu près d'une semaine pour avoir une vision globale du résultat de l'exécution des 21 programmes de la RDO et comprendre l'ensemble des anomalies.
En 2009, pour le déploiement des trois premières délégations, il nous fallait près d'une journée pour faire cette analyse mais sans disposer de l'historisation des résultats.
Depuis septembre 2010 avec DaPCol, les analyses sont disponibles tous les matins.
DaPCol, nous a ainsi permis, d'avoir un suivi précis du résultat du déploiement de chaque délégation et à contribuer à remplir notre engagement de 97% de données reprises en trois jours par délégation. Grâce au mécanisme des compléments d'analyse (cf. §IV.1.B.b.3

p35), nous avons pu capitaliser chaque analyse d'erreurs de façon à obtenir des rapports nous permettant d'organiser correctement le travail de l'équipe.

De 2010 à mai 2011, l'organisation de l'équipe était la suivante :

- Pour l'aspect reprise de données :
  - Une personne avait la charge de la planification des différentes étapes relatives aux déploiements de chaque délégation (synchronisation et communication avec les différents services, envoi et récupération des fichiers complémentaires)
  - Une personne avait la charge de la reprise des informations générales relatives aux clients (identification des clients dans le nouveau système, reprise des adresses, des relations et création des nouveaux comptes clients)
  - Deux personnes avaient la charge de la reprise des données contractuelles et de facturations.
- Pour l'aspect interfaçage avec le Mainframe de facturation FAP :
  - Une personne avait la charge de suivre la génération des transactions de l'eBS à destination de la FAP
  - Une personne avait la charge de suivre, l'intégration de ces transactions par le Mainframe et la remontée des informations de facturation du Mainframe à destination de l'eBS.

Outre la mise en place de ce nouvel outil, mon rôle était de suivre le bon déroulement de l'ensemble des traitements et de piloter les actions de l'équipe.

Grâce à cet outil, nous avons pu obtenir un suivi détaillé et général aussi bien par délégation que par type de traitement. Cette pluralité d'approche, nous a permis d'obtenir une grande réactivité face aux différents problèmes que nous avons connus lors de cette phase de déploiement, à savoir notamment :

- Absence de mécanisme de purge sur certaines tables qui étaient devenues trop volumineuses et qui posaient des problèmes en termes de stockage et de performance.
- Diminution des performances suite à l'intégration de nouvelles délégations. Suite à la montée en charge, un certain nombre de plan d'exécution de requête sont devenus trop coûteux, il a donc fallu faire un travail d'optimisation pour respecter les objectifs.

- Concurrence d'accès entre différents programmes exécutés en parallèle.
- Remplissage du cache de gestion de transaction (snapshot).
- Paramétrage manquant pour certains cas spécifiques (Monaco/DOM-TOM).
- Régression suite à la mise en production de certains correctifs.
- Incompatibilités de certaines évolutions avec les mécanismes de reprise de données.
- Cas de reprise non prévus dans les spécifications.
- Mise en place de nouveaux traitements suite à l'implémentation de nouvelles fonctionnalités majeures (modification de la gestion de la SPRE).

Actuellement, 6 audits sont planifiés quotidiennement sur les différentes bases. Trois audits nous permettent de faire le suivi des programmes de la reprise de données et de l'interface eBS-FAP, deux audits nous permettent de suivre l'évolution de la volumétrie des tables et des index et enfin le dernier audit nous permet d'avoir l'historique de l'ensemble des programmes exécutés sur la base et de détecter ainsi les potentiels problèmes de concurrences ou de pertes de performance.
L'exécution de ces 6 audits dure en moyenne 20 minutes par environnement auxquels il faut ajouter 15 minutes pour générer les 5 rapports que nous exploitons quotidiennement.
L'ensemble des données des audits occupe, après une année complète de collecte, 3.5 Go d'espace sur la base.

Conclusion

La Sacem, en 2003, a décidé de moderniser ses outils internes dédiés à la gestion clientèle en se basant sur un ERP. La première phase de migration vers ce nouveau système a nécessité le développement d'une vingtaine de programmes de reprise de données et la mise en place d'une interface avec le système de facturation existant, à l'aide de 7 programmes exécutés quotidiennement.

Lors de la mise en place de ces deux processus, nous nous sommes focalisés sur les programmes à créer et sur l'intégration des règles de gestion en supposant que l'ensemble du système fonctionnerait correctement. Pour la reprise de données, nous nous sommes rendu compte qu'il serait préférable d'effectuer certaines réconciliations de données manuellement plutôt que d'essayer de développer des programmes capables de gérer 100% des cas.

De plus, notre ERP n'étant pas figé, il a continué d'évoluer en fonction des besoins de l'entreprise. Dans la mesure où il est toujours particulièrement délicat de mesurer correctement l'intégralité des impacts de chaque évolution, la reprise de données et l'interface eBS-FAP ont rencontré des disfonctionnements, suite à la mise en place de ces évolutions, alors qu'ils n'étaient pas, à première vue, impactées par ces évolutions.

Dans ce contexte, l'administration de l'exécution quotidienne des programmes s'est révélée primordiale pour garantir une haute qualité de service, un management efficace de l'équipe et le respect d'un planning de déploiement ambitieux.

Pour ce faire j'ai mis en place un outil nommé DaPCol. Cet outil permet d'exécuter des requêtes d'analyse sur une base cible, d'en historiser les résultats et de générer des rapports, le tout de façon automatique ou à la demande. Grâce à cet outil, nous avons pu monitorer au quotidien le résultat de l'exécution des programmes de notre équipe et disposer ainsi d'une vision globale de la situation. Pour chaque typologie d'anomalie que nous avons rencontrée, nous avons pu intégrer nos analyses dans notre outil DaPCol et ainsi capitaliser au mieux notre travail. Grâce aux rapports générés quotidiennement, j'ai pu organiser le travail de l'équipe de façon non seulement à traiter les anomalies par ordre d'importance mais aussi à veiller à ne pas laisser une anomalie non-traitée trop longtemps. Cet outil nous a permis non seulement de respecter le planning déploiement mais aussi nos objectifs. Chaque délégation a pu travailler sur 97% de ses dossiers trois jours après le

début de la reprise de données et, deux semaines après la fin du déploiement, 100% des données des clients ont été reprises.
A l'issue du déploiement, je suis devenu responsable du back-office de notre ERP. DaPCol va maintenant être utilisé pour suivre l'ensemble des programmes du back-office exécutés quotidiennement.

Après un an d'exploitation de l'outil, plusieurs axes d'améliorations ont été identifiés :

- Optimiser l'espace utilisé par les tables des résultats d'audit et améliorer les performances. Une étude effectuée sur la possibilité de stocker les résultats des audits dans des tables architecturées plutôt que dans une seule table a montré qu'il était possible d'obtenir un gain de place de 62% à 78% et un gain d'exécution lors de la génération des rapports de 60% à 70%. Cette optimisation réduirait la taille de la base à 1.4Go et permettrait de générer l'ensemble des rapports en 6 minutes par environnement.
- Permettre la génération automatique des rapports dès la fin de l'exécution des audits.
- Améliorer le système de lancement afin d'être plus générique (pouvoir lancer n'importe quelles procédures PL/SQL).

Bibliographie

Ouvrages imprimés

Bernard Debauche, Patrick Mégard. *BPM Business Process Management : pilotage métier de l'entreprise*. Paris : Lavoisier, 2004. 212 p. Collection Management et Informatique. ISBN : 2-7462-0852-0.

Jean-Noël Gillot. La gestion des processus métiers : l'alignement des objectifs stratégiques de l'entreprise et la system d'information par les processus. Paris : Booksurge Llc 2007. 372 p. Collection NeoVision Group. ISBN: 978-2-9528-2660-0

Teodo Danciu, Lucian Chirita. *The definitive Guide to JasperReports*^{TM} . Breinigsville, PA USA : Apress, 2010. 223 p. ISBN-13 : 978-1-59059-927-3.

Giulio Toffoli. *The definitive Guide to iReports*^{TM} . Breinigsville, PA USA : Apress, 2010. 223 p. ISBN-13 : 978-1-59059-928-0.

David R. Heffelfinger. *JasperReports forJava Developpers* . Germany : Packt Publishing, 2006. 328 p. From Technologies to Solutions. ISBN : 1-904811-90-6.

Chapitre dans un ouvrage imprimé

Heidi Buelow, Manoj Das, Manas Des, Prasen Palvankar Meera Srinivasan. *Chapter 13 : Process Analytics and Business Activity Monitoring in Getting Started with Oracle BPM Suite 11gR1* . Great Britain : Packt Publishing, 2010. 513 p. Professional Expertise Distilled. ISBN : 978-1-849681-68-1.

Sites web

Sacem. Disponible sur <http://www.sacem.fr/>. (Toujours consultable le 04-mars-2012)

Piloter la performance des processus avec le BAM. In : LeJournalduNet. Disponible sur <http://www.journaldunet.com/solutions/0607/060726-panorama-bam/1.shtml>. (Toujours consultable le 04-mars-2012)

BSM pour gestion de la qualité des services métier. In : LeJournalduNet. Disponible sur <http://www.journaldunet.com/solutions/0310/031021_bsm.shtml>. (Toujours consultable le 04-mars-2012)

BAM : Business Activity Monitoring - Piloter la performance opérationnelle des processus et des activités métiers. In : guideinformatique.com. Disponible sur <http://www.guideinformatique.com/fiche-bam_business_activity_monitoring-808.htm>. (Toujours consultable le 04-mars-2012)

Gestion des Services Métier - BSM (Business Service Management). In : guideinformatique.com. Disponible sur <http://www.guideinformatique.com/fiche-gestion_des_services_metier-772.htm>. (Toujours consultable le 04-mars-2012)

ARCHIVE - BAM : Pilotage opérationnel des activités métier. In : Le CXP. Disponible sur <http://www.cxp.fr/domaine-expertise_BAM.htm>. (Toujours consultable le 04-mars-2012)

Le BAM entre dans l'entreprise par différentes portes. In : 01net. Disponible sur <http://www.01net.com/article/249532.html>. (Toujours consultable le 04-mars-2012)

Produits BusinessBridge. In : SYSTAR. Disponible sur <http://www.systar.fr/products/businessbridge/>. (Toujours consultable le 04-mars-2012)

Axway Sentinel, ou la signification d'un nom. In : AXWAY. Disponible sur <http://www.axway.fr/produits-solutions/ps-overview/axway-synchrony/sentinel>. (Toujours consultable le 04-mars-2012)

Oracle Business Activity Monitoring. In : Oracle. Disponible sur <http://www.oracle.com/technetwork/middleware/bam/overview/index.html>. (Toujours consultable le 04-mars-2012)

Introducing TIBCO BusinessEvents™ 3.0. In : TIBCO. Disponible sur <http://www.tibco.com/products/business-optimization/complex-event-processing/businessevents/default.jsp>. (Toujours consultable le 04-mars-2012)

WebMethods : Business Activity Monitoring . In : Software AG. Disponible sur <http://www.softwareag.com/corporate/products/wm/bam/default.asp#>. (Toujours consultable le 04-mars-2012)

WSO2 Business Activity Monitoring . In : WSO2. Disponible sur <http://wso2.com/products/business-activity-monitor/>. (Toujours consultable le 04-mars-2012)

Oracle Database Express Edition. In : Oracle. Disponible sur <http://www.oracle.com/technetwork/database/express-edition/overview/index.html>. (Toujours consultable le 04-mars-2012)

Welcome to apex.oracle.com. In : Oracle Disponible sur <http://apex.oracle.com/>. (Toujours consultable le 04-mars-2012)

Oracle XE - Oracle Database 10g Express Edition. In : RCI Informatique Disponible sur <http://www.rci-informatique.fr/oracle_xe/>. (Toujours consultable le 04-mars-2012)

Using XMLType. In : Oracle. Disponible sur <http://download.oracle.com/docs/cd/B10501_01/appdev.920/a96620/xdb04cre.htm>. (Toujours consultable le 04-mars-2012)

EXTRACTVALUE. In : Oracle. Disponible sur <http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/functions054.htm>. (Toujours consultable le 04-mars-2012)

DBMS_SCHEDULER. In : Oracle. Disponible sur <http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14258/d_sched.htm>. (Toujours consultable le 04-mars-2012)

Building Charts, Gantts and Maps with Oracle Application Express 4.0. In : Oracle. Disponible sur <http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/apex/r40/apexcharts/apexcharts_ll.htm>. (Toujours consultable le 04-mars-2012)

JasperForge. Disponible sur <http://jasperforge.org>. (Toujours consultable le 04-mars-2012)

Table des annexes

Annexe 1 Déclarations et procédure pour la gestion des tables dans le paquetage PKG_AUDIT_COMMUN 74
Annexe 2 Exemple de script de création de table 83
Annexe 3 Procédure pour le lancement d'audit dans le paquetage PKG_AUDIT_TRT 85
Annexe 4 Exemple de trace du mécanisme de collecte 105
Annexe 5 Définition de type de document pour le fichier de configuration XML de DaPCol-Reports 108
Annexe 6 Exemple de fichier de configuration générale 109
Annexe 7 Définition de type de document pour les fichiers des paramètres par la génération automatique et exemples 115
Annexe 8 DaPCol-Reports : Classe IHM 116
Annexe 9 DaPCol-Reports : Classe JComboBoxSQL 135
Annexe 10 DaPCol-Reports : Classe JComboBoxDependent 138

Annexe 1

Déclarations et procédure pour la gestion des tables dans le paquetage PKG_AUDIT_COMMUN

```
--------------------------------------------------------------------------------+
--
-- Type: TBL COL REC
-- Record de gestion des colonnes de la table.
--
-- Description colonne :
--    COL_NAME                    - Nom de la colonne.
--    COL TYPE                    - Type de la colonne.
--    COL COMMENT                 - Description de la colonne.
--    COL_DEFAULT                 - Valeur par défaut.
--    COL_CHECK                   - Contrainte d'intégrité sur les valeurs possibles.
--    COL_NULLABLE                - La colonne peut-elle être null?
--    COL_PK                      - La colonne est-elle clé primaire ?
--    COL FK TBL REF              - Si la colonne est clé externe, table de la clé
--    COL_FK_COL_REF              - Si la colonne est clé externe, colonne de la clé
--
-- Historique :
--   19/03/2010      SA  -  Création.
--
--------------------------------------------------------------------------------+
TYPE TBL_COL_REC IS RECORD
(
    COL_NAME                    VARCHAR2(30),
    COL TYPE                    VARCHAR2(200),
    COL COMMENT                 VARCHAR2(200),
    COL_DEFAULT                 VARCHAR2(200),
    COL_CHECK                   VARCHAR2(200),
    COL_NULLABLE                BOOLEAN DEFAULT TRUE,
    COL_PK                      BOOLEAN DEFAULT FALSE,
    COL FK TBL REF              VARCHAR2(30),
    COL_FK_COL_REF              VARCHAR2(30)
);

--------------------------------------------------------------------------------+
--
-- Type: TBL_COL_TYPE
-- Tableau de gestion des colonnes de la table.
--
-- Historique:
--   19/03/2010      SA  -  Création.
--
--------------------------------------------------------------------------------+
TYPE TBL_COL_TYPE IS TABLE OF TBL_COL_REC INDEX BY BINARY_INTEGER;
```

```
--------------------------------------------------------------------------------+
--
-- Type: TBL_COL_REC
-- Record de gestion des tables.
--
-- Description colonne:
--    TBL_OWNER                   - Propriétaire de la table.
--    TBL_NAME                    - Nom de la table.
--    TBL_DATA_TABLESPACE         - TABLESPACE pour la table.
--    TBL INDEX TABLESPACE        - TABLESPACE pour les index.
--    TBL OPTION                  - Options de création.
--    TBL_COL                     - Colonnes de la table
--
-- Historique:
--   19/03/2010      SA  -  Création.
--
--------------------------------------------------------------------------------+
TYPE TBL_REC IS RECORD
(
    TBL_OWNER                     VARCHAR2(30),
    TBL_NAME                      VARCHAR2(30),
    TBL_DATA_TABLESPACE           VARCHAR2(30),
    TBL INDEX TABLESPACE          VARCHAR2(30),
    TBL_OPTION                    VARCHAR2(200),
    TBL_COL                       TBL_COL_TYPE
);
--------------------------------------------------------------------------------+
--
-- Procedure: SP_DEFINE_TABLE
-- Création ou modification d'une table suivant la description donnée
--
-- Type: Publique.
--
-- Parameters:
--  NEW_TBL                       IN - Description de la table.
--
-- Historique:
--   19/03/2010      SA  -  Création.
--
--------------------------------------------------------------------------------+
PROCEDURE SP_DEFINE_TABLE(
    NEW_TBL                       IN            TBL_REC
)IS
  --
  --Déclaration d'une procédure interne afin de pouvoir effectuer une sortie
  -- à partir d'une chaîne contenant des saut de ligne
  PROCEDURE ISP_OUPUT_STR(STR VARCHAR2) IS
    TEMP VARCHAR2(32767) := STR;
  BEGIN
     WHILE TEMP IS NOT NULL
```

```
      --Tant que la chaîne temporaire n'est pas vide
      LOOP
        IF INSTR(TEMP,CHR(10)) >0 THEN
          --S'il existe encore un saut de ligne dans la chaîne
          --alors on écrit la partie de la chaîne jusqu'au prochain saut de ligne
          DBMS_OUTPUT.PUT_LINE(SUBSTR(TEMP,1,INSTR(TEMP,CHR(10))-1));
          --Réduction de la chaîne temporaire de la partie écrite
          TEMP := SUBSTR(TEMP,INSTR(TEMP,CHR(10))+1);
        ELSE
          --S'il n'existe plus de saut de ligne dans la chaîne
          --alors on écrit la totalité de la chaîne restante
          DBMS_OUTPUT.PUT_LINE(TEMP);
          --Vide totalement la chaîne
          TEMP:= NULL;
        END IF;
      END LOOP;
    END;
    --
  BEGIN
    --
    ISP_OUPUT_STR('------------------------------------------------------');
    --
    DECLARE
      TMP    NUMBER;
      STR    VARCHAR2(32767);
      STRPK  VARCHAR2(32767);
      N_FK   NUMBER := 0;
      STRFK  VARCHAR2(32767);
      b_key  BINARY_INTEGER;
      b_key2 BINARY_INTEGER;
    BEGIN
      --Recherche de la table
      SELECT 1
      INTO   TMP
      FROM   ALL_TABLES
      WHERE  TABLE_NAME = NEW_TBL.TBL_NAME
      AND    OWNER      = NEW_TBL.TBL_OWNER;
      ISP_OUPUT_STR('--La table existe');
      --
```

```
        --Recherche de trigger sur la table
        EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ALL_TRIGGERS WHERE  TABLE_OWNER = ''' ||  NEW_TBL.TBL_OWNER || ''' AND TABLE_NAME  = ''' ||NEW_TBL.TBL_NAME
||'''' INTO TMP;
        --
        IF TMP = 0 THEN
          ISP_OUPUT_STR('--Il n''existe pas de TRIGGER sur la table');
          --
          --Vérification du nombre d'élément présent dans la table
          EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' || NEW_TBL.TBL_OWNER || '.' || NEW_TBL.TBL_NAME INTO TMP;
          --
          IF TMP = 0 THEN
              --Si la table est vide et qu'il n'y a pas de trigger alors on peut la supprimer puis tout recréer
              ISP_OUPUT_STR('--La table est vide');
              --Suppression de la table
              STR := 'DROP   TABLE '||NEW_TBL.TBL_OWNER || '.' || NEW_TBL.TBL_NAME || ' CASCADE CONSTRAINTS';
              ISP_OUPUT_STR(STR||';');
              EXECUTE IMMEDIATE STR;
              --Lève l'anomalie NO_DATA_FOUND pour recréer la table
              RAISE NO_DATA_FOUND;
          ELSE
              --Si la table n'est pas vide alors il faut la mettre à jour
              ISP_OUPUT_STR('--La table n''est pas vide');
          END IF;
        ELSE
          --S'il existe des triggers sur la table alors il faut la mettre à jour
          ISP_OUPUT_STR('--Il existe un ou plusieurs TRIGGER sur la table');
        END IF;
        --
        ISP_OUPUT_STR('--La table  doit être mise à jour');
        --
        --Vérification des colonnes et de leur format
        --Positionnent sur la première colonne du record de description
        b_key:= NEW_TBL.TBL_COL.FIRST;
        LOOP
          EXIT WHEN b_key IS NULL;
          --
          DECLARE
            TMP_COLUMN_TYPE  VARCHAR2(200);
            TMP_NULLABLE     VARCHAR2(1);
            TMP_DEFAULT      VARCHAR2(200);
          BEGIN
            --Recherche d'une colonne du même nom
            SELECT DATA_TYPE || DECODE(DATA_TYPE,'NUMBER'  ,REPLACE(REPLACE('(' || DATA_PRECISION || ',' || DATA_SCALE|| ')',',0)',')'),'(,)','')
                                          ,'VARCHAR2','(' || DATA_LENGTH || ')'
                                          ,NULL) COLUMN_TYPE
                 ,NULLABLE
                 ,DATA_DEFAULT
            INTO   TMP_COLUMN_TYPE , TMP_NULLABLE,TMP_DEFAULT
            FROM   ALL_TAB_COLUMNS
            WHERE  OWNER       = NEW_TBL.TBL_OWNER
            AND    TABLE_NAME  = NEW_TBL.TBL_NAME
```

```
            AND    COLUMN_NAME = NEW_TBL.TBL_COL(b_key).COL_NAME;
            --
            ISP_OUPUT_STR('--La colonne ' || NEW_TBL.TBL_COL(b_key).COL_NAME || ' existe déjà');
            --
            --Préparation de la commande à exécuter pour faire la mise à jour
            STR := 'ALTER  TABLE '||NEW TBL.TBL OWNER || '.' || NEW TBL.TBL NAME || ' MODIFY ' ||  NEW TBL.TBL COL(b_key).COL_NAME || ' ';
            --La variable TMP sera utilisé pour savoir s'il faut exécuter la commande une fois celle-ci complétée
            TMP := 0;
            --
            --Comparaison des types de données antre la colonne existante et la colonne du record de description
            IF UPPER(TMP COLUMN TYPE) != UPPER(REPLACE(NEW TBL.TBL COL(b key).COL_TYPE,' '))  THEN
                STR := STR || NEW_TBL.TBL_COL(b_key).COL_TYPE || ' ';TMP := 1;
            END IF;
            --
            --Comparaison de la valeur par défaut entre la colonne existante et la colonne du record de description
            IF NVL(TRIM(TMP_DEFAULT),'NULL')='NULL' AND NEW_TBL.TBL_COL(b_key).COL_DEFAULT IS NOT NULL THEN
               STR := STR || 'DEFAULT ''' || REPLACE(NEW_TBL.TBL_COL(b_key).COL_DEFAULT,'''','''''') || ''' ';TMP := 1;
            ELSIF NVL(TRIM(TMP DEFAULT),'NULL')!='NULL' AND NEW_TBL.TBL_COL(b_key).COL_DEFAULT IS NULL THEN
                  STR := STR || 'DEFAULT NULL ';TMP := 1;
            ELSIF NEW_TBL.TBL_COL(b_key).COL_DEFAULT IS NOT NULL AND ''''||REPLACE(NEW_TBL.TBL_COL(b_key).COL_DEFAULT,'''','''''')|| '''' != TRIM(TMP_DEFAULT)
THEN
               STR := STR || 'DEFAULT ''' || REPLACE(NEW_TBL.TBL_COL(b_key).COL_DEFAULT,'''','''''') || ''' ';TMP := 1;
            END IF;
            --
            --Comparaison de la nullabilité entre la colonne existante et la colonne du record de description
            IF NEW_TBL.TBL_COL(b_key).COL_NULLABLE = FALSE AND TMP_NULLABLE = 'Y' THEN
               STR := STR || 'NOT NULL ';TMP := 1;
            ELSIF NEW TBL.TBL COL(b key).COL NULLABLE = TRUE AND TMP_NULLABLE = 'N' THEN
               STR := STR || 'NULL ';TMP := 1;
            END IF;
            --
            --Ajout des contraintes de la colonne du record de description
            IF NEW_TBL.TBL_COL(b_key).COL_CHECK IS NOT NULL THEN
               STR := STR || 'CHECK(' || NEW_TBL.TBL_COL(b_key).COL_CHECK || ') '; TMP := 1;
            END IF;
            --
            IF TMP != 0 THEN
                --Si des modifications ont été précédemment détectées alors on exécute la commande
                ISP_OUPUT_STR(STR||';');
                EXECUTE IMMEDIATE STR;
            END IF;
            --
          EXCEPTION
            WHEN NO_DATA_FOUND THEN
              --Une colonne du même nom n'a pas été trouvée
              ISP_OUPUT_STR('--La colonne ' || NEW_TBL.TBL_COL(b_key).COL_NAME || ' doit être créée');
              --
              --Préparation de la commande à exécuter pour faire ajouter la colonne
              STR := 'ALTER  TABLE '||NEW_TBL.TBL_OWNER || '.' || NEW_TBL.TBL_NAME || ' ADD ' ||  RPAD(NEW_TBL.TBL_COL(b_key).COL_NAME,40) ||
RPAD(NEW TBL.TBL COL(b_key).COL_TYPE,30);
              --
```

```
            --Ajout des éléments en fonction du record de description de la colonne
            IF NEW_TBL.TBL_COL(b_key).COL_DEFAULT  IS NOT NULL THEN STR := STR || 'DEFAULT ''' || REPLACE(NEW_TBL.TBL_COL(b_key).COL_DEFAULT,'''',''''''') ||
''' '; END IF;
            IF NEW_TBL.TBL_COL(b_key).COL_CHECK    IS NOT NULL THEN STR := STR || 'CHECK(' || NEW_TBL.TBL_COL(b_key).COL_CHECK || ') '; END IF;
            IF NEW_TBL.TBL_COL(b_key).COL_NULLABLE = FALSE     THEN STR := STR || 'NOT NULL '; END IF;
            --
            --Exécution de la commande d'ajout de la colonne
            ISP_OUPUT_STR(STR||';');
            EXECUTE IMMEDIATE STR;
          END;
          --
          --Changement de colonne du record de description
          b_key:=NEW_TBL.TBL_COL.NEXT(b_key);
        END LOOP;
        --
        --Vérification s'il ne faut pas supprimer une colonne
        FOR REC IN (SELECT COLUMN_NAME
                    FROM   ALL TAB COLUMNS
                    WHERE  OWNER      = NEW_TBL.TBL_OWNER
                    AND    TABLE_NAME = NEW_TBL.TBL_NAME) LOOP
          --Pour chaque colonne présente sur la table
          --on recherche dans le record de description s'il existe une colonne
          --du même nom
          --
          --Positionnent sur la première colonne du record de description
          b_key:= NEW_TBL.TBL_COL.FIRST;
          --La variable TMP sera utilisé pour savoir si la colonne a été trouvé
          TMP  := 0;
          LOOP
            EXIT WHEN b_key IS NULL;
            --
            --Si la colonne est présent alors TMP=1
            IF NEW_TBL.TBL_COL(b_key).COL_NAME = REC.COLUMN_NAME THEN TMP := 1;EXIT;END IF;
            --
            --Changement de colonne du record de description
            b_key:=NEW_TBL.TBL_COL.NEXT(b_key);
          END LOOP;
          --
          IF TMP = 0 THEN
             --Si la colonne n'a pas été trouvée dans le record de description
             --alors on la supprime
             ISP_OUPUT_STR('--La colonne ' || REC.COLUMN_NAME || ' doit être supprimée');
             STR := 'ALTER  TABLE '||NEW_TBL.TBL_OWNER || '.' || NEW_TBL.TBL_NAME || ' DROP COLUMN ' ||  REC.COLUMN_NAME;
             ISP_OUPUT_STR(STR||';');
             EXECUTE IMMEDIATE STR;
          END IF;
          --
        END LOOP;
        --
      EXCEPTION
        WHEN NO_DATA_FOUND THEN
```

```
--La table n'a pas été trouvée
ISP_OUPUT_STR('--La table  n''existe pas');
--
--Préparation de la commande à exécuter pour créer la table
STR := 'CREATE TABLE '||NEW_TBL.TBL_OWNER || '.' || NEW_TBL.TBL_NAME || '(' ||CHR(10);
--
--Préparation de la partie de la commande concernant la clé primaire
STRPK := 'CONSTRAINT '||NEW_TBL.TBL_NAME||'_PK PRIMARY KEY (';
--
--Parcours des colonnes pour compléter la commande
--
--Positionnent sur la première colonne du record de description
b_key:=NEW_TBL.TBL_COL.FIRST;
LOOP
  EXIT WHEN b_key IS NULL;
     --
     --Ajout des éléments en fonction du record de description de la colonne
     STR := STR || '    ' || RPAD(NEW TBL.TBL COL(b key).COL NAME,40) || RPAD(NEW_TBL.TBL_COL(b_key).COL_TYPE,30);
     IF NEW_TBL.TBL_COL(b_key).COL_DEFAULT  IS NOT NULL THEN
           STR := STR || 'DEFAULT ''' || REPLACE(NEW_TBL.TBL_COL(b_key).COL_DEFAULT,'''',''''''') || ''' '; END IF;
     IF NEW_TBL.TBL_COL(b_key).COL_CHECK    IS NOT NULL THEN STR := STR || 'CHECK(' || NEW_TBL.TBL_COL(b_key).COL_CHECK || ') '; END IF;
     IF NEW_TBL.TBL_COL(b_key).COL_NULLABLE = FALSE     THEN STR := STR || 'NOT NULL '; END IF;
     --
     --Si la colonne fait partie de la clé primaire
     --alors on complète la partie de la commande concernant la clé primaire
     IF NVL(NEW_TBL.TBL_COL(b_key).COL_PK,FALSE)  THEN STRPK := STRPK || NEW_TBL.TBL_COL(b_key).COL_NAME ||',';END IF;
     --
     IF NEW TBL.TBL COL(b key).COL FK TBL REF IS NOT NULL AND NEW_TBL.TBL_COL(b_key).COL_FK_COL_REF IS NOT NULL THEN
        --Si la colonne fait partie d'une clé étrangère
        IF INSTR(STRFK,'REFERENCES '|| NEW_TBL.TBL_COL(b_key).COL_FK_TBL_REF || '(') = 0 OR STRFK IS NULL THEN
          --Si la partie de la commande concernant les clés étrangère ne référence
          --pas la même table que celle présente dans le record de description de la colonne
          --
          ---Préparation de la partie de la commande concernant la clé étrangère
          STRFK := STRFK || CHR(10) || 'CONSTRAINT '||NEW TBL.TBL NAME||' FK' || N FK ||
                     ' FOREIGN KEY (' || NEW_TBL.TBL_COL(b_key).COL_NAME || ') REFERENCES ' || NEW_TBL.TBL_COL(b_key).COL_FK_TBL_REF ||
                     '(' || NEW_TBL.TBL_COL(b_key).COL_FK_COL_REF ||'),' ;
          --
          --Parcours des colonnes pour chercher toutes celles référençant la même table
          --
          --Positionnent sur la première colonne du record de description
          b_key2 := NEW_TBL.TBL_COL.NEXT(b_key);
          LOOP
            EXIT WHEN b_key2 IS NULL;
            --
            IF NEW TBL.TBL COL(b key).COL FK TBL REF = NEW TBL.TBL COL(b key2).COL FK TBL REF THEN
               --Si la table de la clé étrangère de cette colonne correspond à la même table de la clé étrangère de la colonne en cours
               --alors on complémente la partie de la commande concernant la clé étrangère
               STRFK := SUBSTR(STRFK,1,INSTR(STRFK,')',INSTR(STRFK,NEW_TBL.TBL_NAME||'_FK' || N_FK))-1) || ',' || NEW_TBL.TBL_COL(b_key2).COL_NAME ||
                        SUBSTR(STRFK,  INSTR(STRFK,')',INSTR(STRFK,NEW TBL.TBL NAME||' FK' || N FK)), INSTR(STRFK,')', INSTR(STRFK,
                        NEW_TBL.TBL_NAME||'_FK' || N_FK),2)-INSTR(STRFK,')',INSTR(STRFK,NEW_TBL.TBL_NAME||'_FK' || N_FK)))
```

```
                        || ',' || NEW_TBL.TBL_COL(b_key2).COL_NAME ||
                        SUBSTR(STRFK,  INSTR(STRFK,')',INSTR(STRFK,NEW_TBL.TBL_NAME||'_FK' || N_FK),2));
            END IF;
            --
            --Changement de colonne du record de description
            b key2 := NEW_TBL.TBL_COL.NEXT(b_key2);
          END LOOP;
          --
          --Augmente l'indice du nombre de clé étrangère
          N_FK := N_FK + 1;
          --
        END IF;
     END IF;
     --
     STR := STR || ',' ||CHR(10);
  --
  --Changement de colonne du record de description
  b key:=NEW_TBL.TBL_COL.NEXT(b_key);
END LOOP;
--
IF  SUBSTR(STRPK,-1) = ',' THEN
   --S'il y eu des colonnes marquées comme clé primaire
   --alors on ajoute la partie de la commande concernant la clé primaire
   --à la commande à exécuter pour créer la table
   STR := STR || SUBSTR(STRPK,1, LENGTH(STRPK)-1)|| ')';
   IF NEW_TBL.TBL_INDEX_TABLESPACE IS NOT NULL THEN
       --Ajout du tablespace des index si nécessaire
       STR := STR || ' USING INDEX TABLESPACE ' ||  NEW_TBL.TBL_INDEX_TABLESPACE;
   END IF;
   STR := STR || ',' || CHR(10);
ELSE
   STR := SUBSTR(STR,1, LENGTH(STR)-2) || CHR(10);
END IF;
--
IF STRFK IS NOT NULL THEN
    --S'il y eu des colonnes marqué comme clé étrangère
   --alors on ajoute la partie de la commande concernant la clé étrangère
   --à la commande à exécuter pour créer la table
   STR := STR || STRFK || CHR(10);
ELSE
   STR := SUBSTR(STR,1, LENGTH(STR)-2) || CHR(10);
END IF;
--
--Suppression des caractères superflus
IF  SUBSTR(STR,-1) = ',' THEN STR := SUBSTR(STR,1, LENGTH(STR)-1); END IF;
IF  SUBSTR(STR,-2) = ','|| CHR(10) THEN STR := SUBSTR(STR,1, LENGTH(STR)-2) || CHR(10); END IF;
--
--Ajout du tablespace de la table et des options
STR := STR || ') TABLESPACE ' || NEW_TBL.TBL_DATA_TABLESPACE ||CHR(10) || NEW_TBL.TBL_OPTION;
--
--Exécution de la commande
```

```
        ISP_OUPUT_STR(STR||';');
        EXECUTE IMMEDIATE STR;
        --
    END;
    --
    --Qu'il y ait eu création ou modification de la table
    --on ajoute tous les commentaires
    ISP_OUPUT_STR('--Mise en place des commentaires');
    DECLARE
      STR VARCHAR2(32767);
      b key BINARY_INTEGER;
    BEGIN
      --Parcours des colonnes
      --
      --Positionnent sur la première colonne du record de description
      b_key:=NEW_TBL.TBL_COL.FIRST;
      LOOP
        EXIT WHEN b_key IS NULL;
        --
           --Exécution de la commande d'ajout de commentaire
           STR := 'COMMENT ON COLUMN ' || NEW_TBL.TBL_OWNER || '.' || NEW_TBL.TBL_NAME || '.' || RPAD(NEW_TBL.TBL_COL(b_key).COL_NAME,30)
                  || ' IS ''' || REPLACE(NEW_TBL.TBL_COL(b_key).COL_COMMENT,'''','''''') || '''';
           ISP_OUPUT_STR(STR||';');
           EXECUTE IMMEDIATE STR;
        --
        b_key:=NEW_TBL.TBL_COL.NEXT(b_key);
      END LOOP;
    END;
    --
END;
```

Annexe 2

Exemple de script de création de table

```
-------------------------------------------------------------------------------+
--
-- Title: T_AUDIT.SQL
--
-- Table: T_AUDIT
--
-- Table de gestion des configurations des audits.
--   AUDITD : Tablespace des données.
--   AUDITX : Tablespace des index.
--   AUDITU : Compte de connexion à l'application.
--
-- Description colonne:
--    AUDIT_ID                    - Identifiant de l'audit.
--    NAME                        - Nom de l'audit
--    DESCRIPTION                 - Description de l'audit.
--    RESULT_TBL                  - Table des résultats
--    MODE                        - Mode de remplissage de la table :
--                                     APPEND (chaque audit ajoute ces données),
--                                     REPLACE (Supprime les données du précèdent audit sur la base demandé)
--
-- Historique:
--   19/03/2010      SA  -  Creation.
--
-------------------------------------------------------------------------------+
SET VERIFY OFF
WHENEVER SQLERROR CONTINUE;
WHENEVER OSERROR EXIT FAILURE ROLLBACK;

SET SERVEROUTPUT ON;

DECLARE
  NEW_TBL         PKG_AUDIT_COMMUN.TBL_REC;
  N_COL           NUMBER := 0;
BEGIN
  NEW_TBL.TBL_OWNER            := 'AUDITU';
  NEW_TBL.TBL_NAME             := 'T_AUDIT';
  NEW_TBL.TBL_DATA_TABLESPACE  := 'AUDITD' ;
  NEW_TBL.TBL_INDEX_TABLESPACE := 'AUDITX' ;
  NEW_TBL.TBL_OPTION           := 'PCTFREE 10 '                          || CHR(10) ||
                                  'PCTUSED 40'                           || CHR(10) ||
                                  'INITRANS 1'                           || CHR(10) ||
                                  'MAXTRANS 255'                         || CHR(10) ||
                                  'STORAGE ('                            || CHR(10) ||
```

```
                                '         INITIAL 65536'                || CHR(10) ||
                                '         MINEXTENTS 1'                 || CHR(10) ||
                                '         MAXEXTENTS 2147483645'        || CHR(10) ||
                                '         FREELISTS 1 FREELIST GROUPS 1)';
  --
  NEW_TBL.TBL_COL(N_COL).COL_NAME        := 'AUDIT_ID';
  NEW_TBL.TBL_COL(N_COL).COL_TYPE        := 'NUMBER';
  NEW_TBL.TBL_COL(N_COL).COL_COMMENT     := 'Identifiant de l''audit' ;
  NEW_TBL.TBL_COL(N_COL).COL_NULLABLE    := FALSE;
  NEW_TBL.TBL_COL(N_COL).COL_PK          := TRUE;
  N_COL := N_COL +1;
  --
  NEW_TBL.TBL_COL(N_COL).COL_NAME        := 'NAME';
  NEW_TBL.TBL_COL(N_COL).COL_TYPE        := 'VARCHAR2(30)';
  NEW_TBL.TBL_COL(N_COL).COL_COMMENT     := 'Nom de l''audit' ;
  NEW_TBL.TBL_COL(N_COL).COL_NULLABLE    := FALSE;
  N_COL := N_COL +1;
  --
  NEW_TBL.TBL_COL(N_COL).COL_NAME        := 'DESCRIPTION';
  NEW_TBL.TBL_COL(N_COL).COL_TYPE        := 'VARCHAR2(200)';
  NEW_TBL.TBL_COL(N_COL).COL_COMMENT     := 'Description de l''audit' ;
  NEW_TBL.TBL_COL(N_COL).COL_NULLABLE    := FALSE;
  N_COL := N_COL +1;
  --
  NEW_TBL.TBL_COL(N_COL).COL_NAME        := 'RESULT_TBL';
  NEW_TBL.TBL_COL(N_COL).COL_TYPE        := 'VARCHAR2(30)';
  NEW_TBL.TBL_COL(N_COL).COL_COMMENT     := 'Table de stockage des résultats' ;
  NEW_TBL.TBL_COL(N_COL).COL_NULLABLE    := FALSE;
  N_COL := N_COL +1;
  --
  NEW_TBL.TBL_COL(N_COL).COL_NAME        := 'INSERT_MODE';
  NEW_TBL.TBL_COL(N_COL).COL_TYPE        := 'VARCHAR2(10)';
  NEW_TBL.TBL_COL(N_COL).COL_COMMENT     := 'Mode de remplissage de la table :
                                             APPEND (chaque audit ajoute ces données),
                                             REPLACE (Supprime les données du précèdent audit sur la base demandé)';
  NEW_TBL.TBL_COL(N_COL).COL_NULLABLE    := FALSE;
  NEW_TBL.TBL_COL(N_COL).COL_CHECK       := 'INSERT_MODE=''APPEND'' OR INSERT_MODE=''REPLACE''' ;
  N_COL := N_COL +1;
  --
  NEW_TBL.TBL_COL(N_COL).COL_NAME        := 'COL_KEY_FOR_REPLACE';
  NEW_TBL.TBL_COL(N_COL).COL_TYPE        := 'VARCHAR2(4000)';
  NEW_TBL.TBL_COL(N_COL).COL_COMMENT     := 'Colonne Clé pour effectuer les remplacements
                                               si INSERT_MODE=REPALCE (nb : si NULL alors toutes les données seront remplacées)';
  NEW_TBL.TBL_COL(N_COL).COL_NULLABLE    := TRUE;
  N_COL := N_COL +1;
  --
  PKG_AUDIT_COMMUN.SP_DEFINE_TABLE(NEW_TBL);
END;
/
```

Annexe 3

Procédure pour le lancement d'audit dans le paquetage PKG_AUDIT_TRT

```
--------------------------------------------------------------------------------+
--
-- Procedure: SP LAUNCH AUDIT
-- Lancement d'un audit.
--
-- Type: Publique.
--
-- Parameters:
--    PV DB LINK                  IN - Base de données.
--    PN_AUDIT_ID                 IN - Identifiant de l'audit.
--    PN_AUDIT_REQ_ID             IN - Identifiant de la requête (optionnel).
--    PN_AUDIT_TRACE_ID_PARENT    IN - Identifiant de la trace parent.
--    PB_FORCED_LAUNCH            IN - Forcer l'exécution des requêtes (optionel).
--    PN_NIVEAU_LOG               IN - Niveau de log (optionel).
--
-- Historique:
--   19/03/2010      SA  -  Création.
--
--------------------------------------------------------------------------------+
PROCEDURE SP LAUNCH AUDIT (
    PV DB LINK                    IN            T AUDIT TRACE.DB LINK%TYPE,
    PN_AUDIT_ID                   IN            T_AUDIT_TRACE.AUDIT_ID%TYPE              DEFAULT NULL,
    PN_AUDIT_REQ_ID               IN            T_AUDIT_TRACE.AUDIT_REQ_ID%TYPE          DEFAULT NULL,
    PN AUDIT TRACE ID PARENT      IN            T AUDIT TRACE.AUDIT TRACE ID PARENT%TYPE  DEFAULT NULL,
    PB FORCED LAUNCH              IN            BOOLEAN                                  DEFAULT FALSE,
    PN_NIVEAU_LOG                 IN            NUMBER                                   DEFAULT 0
 )IS
   PRAGMA AUTONOMOUS_TRANSACTION;
   LN_AUDIT_TRACE_ID            T_AUDIT_TRACE.AUDIT_TRACE_ID%TYPE;
   LV SEQ                       VARCHAR2(10);
   LRT AUDIT                    T AUDIT%ROWTYPE;
   LRT AUDIT REQ                T AUDIT REQ%ROWTYPE;
   LN_NB                        NUMBER := 0;
   LV_STATUT                    T_AUDIT_TRACE.STATUT%TYPE := PKG_AUDIT_COMMUN.GV_STATUT_SUC;
   v_REQ                        VARCHAR2(32767);
   --
 BEGIN
    --
    --Initialisation du contexte
    LV_SEQ := '000';
    PKG_AUDIT_COMMUN.SP_ENTER_CONTEXT('SP_LAUNCH_AUDIT','PKG_AUDIT_TRT');
    --
    --Initialisation de la trace
```

```
LV_SEQ := '010';
LN AUDIT TRACE ID:=PKG AUDIT COMMUN.FN START TRACE( 'LAUNCH'
                                                   ,PN_NIVEAU_LOG
                                                   ,PN_AUDIT_ID
                                                   ,PN AUDIT REQ_ID
                                                   ,PV DB LINK
                                                   ,PN_AUDIT_TRACE_ID_PARENT);


IF PN_AUDIT_ID IS NULL THEN
    --si l'identifiant de l'audit a exécuter n'a pas été précisé
    --alors lancer toutes les audits
    --
    --Ecriture du début du lancement
    LV_SEQ := '020';
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'-------------------------------------------------------------------------------');
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Début du lancement de tous les audits sur ' || PV_DB_LINK);
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'-------------------------------------------------------------------------------');
    --
    --Ouverture du DB Link
    LV_SEQ := '030';
    IF PKG_AUDIT_COMMUN.FB_CONNECT_TO_DBLINK(PV_DB_LINK,LN_AUDIT_TRACE_ID) THEN
        --
        --Lancement des audits fils pour chacun des audits
        LV_SEQ := '040';
        FOR REC IN (SELECT AUDIT_ID
                    FROM   T AUDIT
                    ORDER BY AUDIT ID) LOOP
           SP_LAUNCH_AUDIT (PV_DB_LINK,REC.AUDIT_ID,NULL,LN_AUDIT_TRACE_ID,PB_FORCED_LAUNCH,PN_NIVEAU_LOG);
        END LOOP;
        --
        --Lecture du nombre de données remonté par chaque audit fils et du statut global
        LV_SEQ := '050';
        SELECT SUM(NB),DECODE(MAX(DECODE(STATUT,PKG_AUDIT_COMMUN.GV_STATUT_ANO,1,0)),1,PKG_AUDIT_COMMUN.GV_STATUT_ANO,PKG_AUDIT_COMMUN.GV_STATUT_SUC)
        INTO   LN NB  ,LV STATUT
        FROM   T_AUDIT_TRACE
        WHERE  AUDIT_TRACE_ID_PARENT =LN_AUDIT_TRACE_ID;
        --
        --Fermeture du DB Link
        LV SEQ := '060';
        IF NOT PKG AUDIT COMMUN.FB CLOSE DBLINK(PV DB LINK,LN AUDIT TRACE ID) THEN
            PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_CLOSE', 'Impossible de se déconnecter de ' || PV_DB_LINK);
        END IF;
        --
        --Ecriture de la fin de l'audit
        LV SEQ := '070';
        PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL,'-------------------------------------------------------------------------------');
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Fin du lancement de tous les audits sur ' || PV_DB_LINK);
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'-------------------------------------------------------------------------------');
        --
    ELSE
```

```
        --
        --En cas d'erreur lors de l'ouverture du DB Link
        --Fin de l'audit en anomalie
        LV_SEQ := '080';
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_CONNECT_TO', 'Impossible de se connecter sur ' || PV_DB_LINK);
        LV_STATUT := PKG_AUDIT_COMMUN.GV_STATUT_ANO;
    END IF;
ELSE
  --
  --Récupération du record de l'audit à traiter
  LV_SEQ := '090';
  SELECT *
  INTO   LRT_AUDIT
  FROM   T_AUDIT
  WHERE  AUDIT_ID = PN_AUDIT_ID;
  --
  IF PN_AUDIT_REQ_ID IS NULL THEN
     --
     --Si l'identifiant de la requête à exécuter n'a pas été précisé
     --alors lancer toutes les requêtes de l'audit
     --
     --Ecriture du début de l'audit
     LV_SEQ := '100';
     PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'------------------------------------------------------------------------------');
     PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Début de l''audit : ' || LRT_AUDIT.NAME || ' (' || LRT_AUDIT.DESCRIPTION ||
                                                            ') sur ' || PV_DB_LINK);
     PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'------------------------------------------------------------------------------');
     --
     --positionne AUDIT_VALIDITY_END_DATE sur les précédents AUDIT
     LV_SEQ := '110';
     BEGIN
       v_REQ  := 'UPDATE ' || LRT_AUDIT.RESULT_TBL || CHR(10) ||
                 'SET    AUDIT_VALIDITY_END_DATE = TO_DATE(''' || TO_CHAR(SYSDATE,'DD/MM/YYYY HH24:MI:SS') ||''','DD/MM/YYYY HH24:MI:SS')'||CHR(10)||
                 'WHERE  AUDIT DB           = '''|| PV DB LINK || ''''  || CHR(10) ||
                 'AND    AUDIT VALIDITY_END_DATE IS NULL ' ;
       EXECUTE IMMEDIATE v_REQ;
       COMMIT;
       PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'AUDIT_VALIDITY_END_DATE positionné' ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
     EXCEPTION
       WHEN OTHERS THEN
         --En cas d'erreur
         --Enregistrement de l'erreur Oracle
         PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_VALID', 'Impossible de mettre à jour AUDIT_VALIDITY_END_DATE ('||
                                                                                    LV_SEQ||'):' || SQLERRM);
         --Ecriture de la requête d'ajouts des valeurs de gestion
         PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
     END;
     --
     --Ouverture du DB Link
     LV_SEQ := '120';
     IF PKG_AUDIT_COMMUN.FB_CONNECT_TO_DBLINK(PV_DB_LINK,LN_AUDIT_TRACE_ID) THEN
```

```
      --
      --Lancement des audits fils pour chacune des requêtes
      LV_SEQ := '130';
      FOR REC IN (SELECT AUDIT_REQ_ID
                  FROM   T AUDIT REQ
                  WHERE  AUDIT ID = PN AUDIT ID
                  ORDER BY AUDIT_REQ_ID) LOOP
            SP_LAUNCH_AUDIT (PV_DB_LINK,PN_AUDIT_ID,REC.AUDIT_REQ_ID,LN_AUDIT_TRACE_ID,PB_FORCED_LAUNCH,PN_NIVEAU_LOG);
      END LOOP;
      --
      --Lecture du nombre de données remonté par chaque audit fils et du statut global
      LV SEQ := '140';
      SELECT SUM(NB),DECODE(MAX(DECODE(STATUT,PKG_AUDIT_COMMUN.GV_STATUT_ANO,1,0)),1,PKG_AUDIT_COMMUN.GV_STATUT_ANO,PKG_AUDIT_COMMUN.GV_STATUT_SUC)
      INTO   LN_NB  ,LV_STATUT
      FROM   T_AUDIT_TRACE
      WHERE  AUDIT_TRACE_ID_PARENT = LN_AUDIT_TRACE_ID;
      --
      --Fermeture du DB Link
      LV_SEQ := '150';
      IF NOT PKG_AUDIT_COMMUN.FB_CLOSE_DBLINK(PV_DB_LINK,LN_AUDIT_TRACE_ID) THEN
         PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_CLOSE', 'Impossible de se déconnecter de ' || PV_DB_LINK);
      END IF;
      --
      --Ecriture de la fin de l'audit
      LV_SEQ := '160';
      PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'--------------------------------------------------------------------------------');
      PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL,'Fin de l''audit : ' || LRT_AUDIT.NAME || ' (' || LRT_AUDIT.DESCRIPTION || ') sur ' ||
                                                            PV DB LINK);
      PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'--------------------------------------------------------------------------------');
      --
   ELSE
      --
      --En cas d'erreur lors de l'ouverture du DB Link
      --Fin de l'audit en anomalie
      LV SEQ := '170';
      PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_CONNECT_TO', 'Impossible de se connecter sur ' || PV_DB_LINK);
      LV_STATUT := PKG_AUDIT_COMMUN.GV_STATUT_ANO;
   END IF;
ELSE
    --
    --Si l'identifiant de la requête a exécuter a été précisé
    --alors il faut l'exécuter
    --
    --Récupération du record de la requête
    LV_SEQ := '180';
    SELECT *
    INTO   LRT AUDIT REQ
    FROM   T_AUDIT_REQ
    WHERE  AUDIT_ID     = PN_AUDIT_ID
    AND    AUDIT_REQ_ID = PN_AUDIT_REQ_ID;
    --
```

```
            IF PN_AUDIT_TRACE_ID_PARENT IS NULL THEN
              --
              --Si ce n'est pas un audit fils
              --Ecriture du début de l'audit
              LV SEQ := '190';
              PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL,'-------------------------------------------------------------------------------');
              PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'Début de l''audit : ' || LRT_AUDIT.NAME || ' (' || LRT_AUDIT.DESCRIPTION ||
                                                                    ' : ' || LRT_AUDIT_REQ.DESCRIPTION || ') sur ' || PV_DB_LINK);
              PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'-------------------------------------------------------------------------------');
              --
              --positionne AUDIT_VALIDITY_END_DATE sur les précédents AUDIT
              LV SEQ := '200';
              BEGIN
                v_REQ  := 'UPDATE ' || LRT_AUDIT.RESULT_TBL || CHR(10) ||
                          'SET    AUDIT_VALIDITY_END_DATE = TO_DATE('''||TO_CHAR(SYSDATE,'DD/MM/YYYY HH24:MI:SS')||''','DD/MM/YYYY HH24:MI:SS')'||CHR(10)||
                          'WHERE  AUDIT_REQ_ID       = '  ||  PN_AUDIT_REQ_ID   || CHR(10) ||
                          'AND    AUDIT_DB           = '''|| PV_DB_LINK || ''''  || CHR(10) ||
                          'AND    AUDIT VALIDITY_END_DATE IS NULL ' ;
                EXECUTE IMMEDIATE v_REQ;
                COMMIT;
                PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'AUDIT_VALIDITY_END_DATE positionné' ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
              EXCEPTION
                WHEN OTHERS THEN
                  --En cas d'erreur
                  --Enregistrement de l'erreur Oracle
                  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_VALID', 'Impossible de mettre à jour AUDIT_VALIDITY_END_DATE
('||LV SEQ||'):' || SQLERRM);
                  --Ecriture de la requête
                  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
              END;


              --Ouverture du DB Link
              LV SEQ := '210';
              IF NOT PKG AUDIT COMMUN.FB CONNECT TO DBLINK(PV DB_LINK,LN_AUDIT_TRACE_ID) THEN
                --En cas d'erreur lors de l'ouverture du DB Link
                --Fin de l'audit en anomalie
                LV_SEQ := '220';
                PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_CONNECT_TO', 'Impossible de se connecter sur ' || PV_DB_LINK);
                LV STATUT := PKG_AUDIT_COMMUN.GV_STATUT_ANO;
              END IF;
            ELSE
              --Si c'est un audit fils
              --Enregistrement de la description de la requête
              LV_SEQ := '230';
              PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL,'----------------------------');
              PKG AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, LRT_AUDIT_REQ.DESCRIPTION);
            END IF;
            --
            IF LV_STATUT = PKG_AUDIT_COMMUN.GV_STATUT_SUC THEN
              --
```

```
--Si le DB Link a été crée
--
DECLARE
  v_TBL_TEMP    VARCHAR2(30);            --Nom de la table temporaire
  v_REQ         VARCHAR2(32767);         --Requête à exécuter
  b_ERR         BOOLEAN := FALSE;        --Indicateur d'erreur
  b_EXECUTE_REQ BOOLEAN := TRUE;         --Indicateur de la nécessité de lancer la requête d'audit
  v_REQ_CREATE_INDEX   VARCHAR2(32767);  --Requête pour reconstruire les indexes
  v_REQ_DROP_INDEX     VARCHAR2(32767);  --Requête pour supprimer les indexes
BEGIN
  --
  IF NOT PB_FORCED_LAUNCH THEN
    --
    --Si nous ne sommes pas en mode forcé
    --alors exécution de la requête de déclenchement pour vérifier s'il est vraiment nécessaire de faire la requête d'audit
    --
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Mode non forcé => execution de la requête de
                                                            déclenchement',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
    --
    --Initialisation de l'indicateur de lancement la requête d'audit
    b_EXECUTE_REQ := FALSE;
    DECLARE
      n_RESULT  VARCHAR2(32767);
    BEGIN
       IF TRIM(TRANSLATE(PKG_AUDIT_COMMUN.FV_GET_CLOB(LRT_AUDIT_REQ.DECLENCHEUR),CHR(9)||CHR(10)||CHR(13),'   ')) IS NOT NULL THEN
         --S'il existe une requête de déclenchement
         LV SEQ := '240';
         PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Il existe une requête de déclenchement',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
         --
         --Execution de la requête de déclenchement
         v_REQ :=  REPLACE(PKG_AUDIT_COMMUN.FV_GET_CLOB(LRT_AUDIT_REQ.DECLENCHEUR),'@DATABASE','@'||PV_DB_LINK);
         EXECUTE IMMEDIATE v_REQ INTO n_RESULT;
         --
         -- Annulation pour ne pas avoir de lock
         ROLLBACK;
         --
         IF  NVL(n_RESULT,'0') != '0' THEN
           -- Si le résultat n'est pas null ou zéro
           -- alors il faudra exécuter la requête d'audit
           LV SEQ := '250';
           b EXECUTE REQ := TRUE;
           PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Retour : '|| NVL(TO_CHAR(n_RESULT),'NULL') ||' => execution de la requête
                                                                   d''audit',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
         ELSE
           -- Si le résultat est null ou zero
           -- alors il ne faudra pas exécuter la requête d'audit
           LV SEQ := '260';
           PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Retour : '|| NVL(TO_CHAR(n_RESULT),'NULL') || ' => pas d''execution
                                                                   de la requête d''audit',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
         END IF;
       ELSE
```

```
        --
        --S'il n'existe pas une requête de déclenchement
        -- alors il faudra exécuter la requête d'audit
        LV_SEQ := '270';
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Pas requête de déclenchement => exécution de la requête
                                                                d''audit',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
        b_EXECUTE_REQ := TRUE;
      END IF;
  EXCEPTION
    WHEN OTHERS THEN
      --En cas d'erreur
      --Enregistrement de l'erreur Oracle
      PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_DECLENCHEUR', 'Erreur lors de l''exécution de la requête
                                                                                        de déclenchement ('||LV_SEQ||'):' || SQLERRM);
      --Ecriture de la requête de déclenchement
      PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
  END;
ELSE
  --Si nous somme en mode forcé
  --alors il faudra exécuter la requête d'audit
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Mode forcé => execution de la requête d''audit'
                                ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
END IF;
--
IF b_EXECUTE_REQ THEN
   LV_SEQ := '280';
  --S'il faut exécuter la requête d'audit
  --
  --Création du nom pour la table temporaire de stockage des résultats
  v_TBL_TEMP := 'T_AUDIT_TEMP_'||PV_DB_LINK|| '_' || PN_AUDIT_ID || '_' || PN_AUDIT_REQ_ID;
  --
  --Suppression de la table temporaire pour le cas où
  BEGIN
    LV SEQ := '290';
    v REQ := 'DROP TABLE ' || v_TBL_TEMP;
    EXECUTE IMMEDIATE v_REQ;
    COMMIT;
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Table ' || v_TBL_TEMP ||' supprimée',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
  EXCEPTION WHEN OTHERS THEN
    NULL;
  END;
  --
  --Création d'une table temporaire pour stocker le résultat de la requête
  BEGIN
    LV_SEQ := '300';
    v REQ := 'CREATE TABLE ' || v TBL TEMP || ' AS ' ||
              REPLACE(PKG AUDIT_COMMUN.FV_GET_CLOB(LRT_AUDIT_REQ.REQUETE),'@DATABASE','@'||PV_DB_LINK);
    EXECUTE IMMEDIATE v_REQ;
    LN_NB := SQL%ROWCOUNT;
    COMMIT;
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Table ' || v_TBL_TEMP ||' crée',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
```

```
EXCEPTION WHEN OTHERS THEN
  --En cas d'erreur
  --Annulation
  ROLLBACK;
  --Enregistrement de l'erreur Oracle
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_CREATE_TBL', 'Impossible de créer la table ' || v_TBL_TEMP ||
                                                                                  '('||LV_SEQ||'):' || SQLERRM);
  --
  LV_SEQ := '310';
  --Enregistrement de la requête de création
  PKG AUDIT COMMUN.SP ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
  --Mémorisation de l'erreur
  b_ERR := TRUE;
END;

IF NOT b_ERR THEN
  --Si la table temporaire a été crée
  --
  --Ajouts des champs de gestion à la table temporaire
  LV_SEQ := '320';
  BEGIN
    EXECUTE IMMEDIATE 'ALTER TABLE ' || v_TBL_TEMP || ' ADD AUDIT_LAUNCH_ID         NUMBER';
    PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL, 'Colonne AUDIT LAUNCH ID         ajouté à '
                                   || v_TBL_TEMP ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
    EXECUTE IMMEDIATE 'ALTER TABLE ' || v_TBL_TEMP || ' ADD AUDIT_REQ_ID            NUMBER';
    PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL, 'Colonne AUDIT REQ ID            ajouté à '
                                   || v TBL TEMP ,PKG AUDIT COMMUN.GN NIVEAU LOG DEBUG);
    EXECUTE IMMEDIATE 'ALTER TABLE ' || v_TBL_TEMP || ' ADD AUDIT_DB                VARCHAR2(30)';
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Colonne AUDIT_DB                ajouté à '
                                   || v_TBL_TEMP ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
    EXECUTE IMMEDIATE 'ALTER TABLE ' || v_TBL_TEMP || ' ADD AUDIT_EXTRACT_DATE      DATE';
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Colonne AUDIT_EXTRACT_DATE      ajouté à '
                                   || v TBL TEMP ,PKG AUDIT COMMUN.GN NIVEAU LOG DEBUG);
    EXECUTE IMMEDIATE 'ALTER TABLE ' || v TBL TEMP || ' ADD AUDIT COMPLT            CLOB';
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Colonne AUDIT_COMPLT            ajouté à '
                                   || v_TBL_TEMP ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
    EXECUTE IMMEDIATE 'ALTER TABLE ' || v_TBL_TEMP || ' ADD AUDIT_VALIDITY_END_DATE DATE';
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Colonne AUDIT_VALIDITY_END_DATE ajouté à '
                                   || v_TBL_TEMP ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
  EXCEPTION
    WHEN OTHERS THEN NULL;
  END;
  --
  --Ajouts des valeurs des champs de gestion
  LV SEQ := '330';
  BEGIN
    v_REQ  := 'UPDATE ' || v_TBL_TEMP || CHR(10) ||
              'SET    AUDIT_LAUNCH_ID    = '  ||  LN_AUDIT_TRACE_ID || CHR(10) ||
              '      ,AUDIT REQ ID       = '  ||  PN AUDIT REQ ID   || CHR(10) ||
              '      ,AUDIT_DB           = '''|| PV_DB_LINK || '''' || CHR(10) ||
```

```
            '      ,AUDIT_EXTRACT_DATE = TO_DATE(''' || TO_CHAR(SYSDATE,'DD/MM/YYYY HH24:MI:SS') ||  ''',''DD/MM/YYYY HH24:MI:SS'')';
  EXECUTE IMMEDIATE v_REQ;
  COMMIT;
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Nouvelles colonnes renseignées' ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
EXCEPTION
  WHEN OTHERS THEN
    --En cas d'erreur
    --Enregistrement de l'erreur Oracle
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_UPDATE_TBL', 'Impossible de mettre à jour
                                                                                     les nouvelles colonnes ('||LV_SEQ||'):' || SQLERRM);
    --Ecriture de la requête d'ajouts des valeurs des champs de gestion
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
END;
--
--Ajouts des informations complémentaires
LV_SEQ := '340';
BEGIN
   --
   --1) Construction des critères pour la recherche des actions complémentaires en fonction des noms de colonnes de la table temporaire

   v_REQ := '';
   FOR REC IN ( SELECT COLUMN_NAME
                FROM ALL_TAB_COLUMNS
                WHERE TABLE_NAME LIKE  v_TBL_TEMP
                ORDER BY COLUMN_NAME) LOOP
       v_REQ := v_REQ ||CHR(9)|| 'AND    (EXTRACTVALUE(CONTEXTE, ''/CONTEXTES/' ||  REC.COLUMN_NAME || ''')'
                      ||LPAD(' ',31-LENGTH(REC.COLUMN_NAME)) ||' IS NULL OR NVL2(A.' || RPAD(REC.COLUMN_NAME,31)
                      ||',''''''''''|| A.' || RPAD(REC.COLUMN_NAME,31)  ||
                        '||''''''''',''NULL'') LIKE EXTRACTVALUE(CONTEXTE, ''/CONTEXTES/' ||  REC.COLUMN_NAME || '''))' || CHR(10);
   END LOOP;
   v_REQ :=  LTRIM(V_REQ,CHR(10));

   --2) Construction de la requête qui permet de récupérer une action par ligne de la table temporaire
   LV SEQ := '350';
   v REQ := 'SELECT (' || CHR(10) ||
            '  SELECT  insertXMLbefore(ACTION,''/ACTIONS/REQ[1]'',XMLTYPE(''<ID>'' || AUDIT_COMP_ID || ''</ID>'')) '  || CHR(10) ||
            '  FROM   T_AUDIT_COMP' || CHR(10) ||
            '  WHERE  AUDIT_ID =' || PN_AUDIT_ID  || CHR(10) ||
           v_REQ || CHR(10) ||
            '  AND ROWNUM = 1'|| CHR(10) ||
            '  ) ACTION'|| CHR(10) ||
            '  ,A.ROWID'|| CHR(10) ||
            'FROM ' || v_TBL_TEMP || ' A';

   --3) Exécution de la requête dans un curseur
   DECLARE
     TYPE REF CURSOR    IS REF CURSOR;
     rc_ACTION          REF_CURSOR;
     r_TEMP_ROWID       ROWID;
     x TEMP_ACTION      XMLTYPE;
   BEGIN
```

```
--
--Ouverture du curseur
LV_SEQ := '360';
OPEN rc_ACTION FOR v_REQ;
--
LOOP
  --Récupération des données
  LV_SEQ := '370';
  FETCH rc_ACTION INTO x_TEMP_ACTION,r_TEMP_ROWID;
  --
  --S'il n'y a plus de donnée alors sortir de la boucle
  EXIT WHEN rc_ACTION%NOTFOUND ;
  --
  IF x_TEMP_ACTION IS NOT NULL THEN
    --
    --S'il existe des actions complémentaires à effectuer
    --alors il faut décoder le flux XML des actions
    --
    DECLARE
      xml_PARSER         DBMS_XMLPARSER.PARSER := DBMS_XMLPARSER.newPARSER;
      xml_DOC            DBMS_XMLDOM.DOMDOCUMENT;
      xml_NODE_LIST      DBMS_XMLDOM.DOMNODELIST;
      xml_NODE           DBMS_XMLDOM.DOMNODE;
      n len              NUMBER;
      n_ERR_ON_ACTION    NUMBER := 0;
      n_AUDIT_COMP_ID    T_AUDIT_COMP.AUDIT_COMP_ID%TYPE;
    BEGIN
      --
      -- Réalisation de l'analyse syntaxique du flux XML des actions complémentaire
      LV_SEQ := '380';
      DBMS_XMLPARSER.ParseCLOB(xml_PARSER, x_TEMP_ACTION.getclobval());
      --
      -- Récupération du document XML
      LV SEQ := '390';
      xml_DOC := DBMS_XMLPARSER.GetDocument(xml_PARSER);
      --
      -- Récupération de la liste des nœuds
      LV_SEQ := '400';
      xml_NODE_LIST   := DBMS_XMLDOM.GetElementsByTagName(xml_DOC, '*');
      --
      -- Parcours de la liste
      n_AUDIT_COMP_ID := NULL;  --> Identifiant du complément (pour gérer les erreurs)
      n_ERR_ON_ACTION := 0;     --> Indicateur d'erreur
      FOR i IN 0 .. DBMS_XMLDOM.GetLength(xml_NODE_LIST) - 1 LOOP
          --
          --Récupération du nœud i
          LV SEQ := '410';
          xml_NODE := DBMS_XMLDOM.Item(xml_NODE_LIST, i);
          --
          DECLARE
           v_NODE_NAME   VARCHAR2(32767) := DBMS_XMLDOM.getNodeName(xml_NODE);
```

```
 v_NODE_VALUE  VARCHAR2(32767) := DBMS_XMLDOM.getNodeValue(DBMS_XMLDOM.getFirstChild(xml_NODE));
BEGIN
  --
  LV_SEQ := '420';
  IF v NODE VALUE IS NOT NULL THEN
    --Si le contenue du nœud n'est pas vide
    --alors enregistrement du nom nœud et de sa valeur
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_NODE_NAME || ' => '
                                   || v_NODE_VALUE,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
  END IF;
  --
  --Effectue les actions en fonction du nom du nœud
  CASE v_NODE_NAME
    WHEN 'ACTIONS' THEN --> nœud Actions (Racine) = Aucune action
      NULL;

    WHEN 'ID' THEN      --> nœud ID = identifiant du complément en cours de traitement
      n_AUDIT_COMP_ID := v_NODE_VALUE;

    WHEN 'MSG' THEN     --> nœud MSG = récupération d'un message de description comme complément
      IF v_NODE_VALUE IS NOT NULL THEN
        -- Si la noeud n'est pas vide
        -- alors ajout du message de description à la ligne de la table temporaire en cours de traitement
        LV SEQ := '430';
        v_REQ := 'UPDATE ' || v_TBL_TEMP || CHR(10) ||
                 'SET    AUDIT_COMPLT = AUDIT_COMPLT || NVL2(AUDIT_COMPLT,CHR(10),NULL) || '''
                         || REPLACE(v NODE VALUE,'''',''''''')  || '''' || CHR(10) ||
                 'WHERE  ROWID = '''  || r_TEMP_ROWID || '''' ;
        EXECUTE IMMEDIATE v_REQ;
        COMMIT;
      END IF;

    WHEN 'REQ' THEN     --> nœud REQ = Exécution d'une requête complémentaire
      IF v NODE VALUE IS NOT NULL THEN
        -- Si le nœud n'est pas vide
        -- alors exécution de la requête
        DECLARE
          RC_RESULT         REF_CURSOR;
          v_RESULT          VARCHAR2(32767);
        BEGIN
          --
          v_REQ := v_NODE_VALUE;
          --
          --Substitution de la DATABASE
          v_REQ := REPLACE(v_REQ,'@DATABASE','@'||PV_DB_LINK);
          --
          --Parcours des champs de la table temporaire
          LV_SEQ := '440';
          FOR REC  IN (SELECT COLUMN_NAME,DATA_TYPE
                       FROM ALL TAB COLUMNS
                       WHERE TABLE_NAME LIKE  v_TBL_TEMP
```

```
                ORDER BY COLUMN_NAME) LOOP
      --
      IF V_REQ LIKE '%:'||REC.COLUMN_NAME || '%' THEN
        --Si ce champ doit être substitué dans la requête
        DECLARE
          v TEMP   VARCHAR2(4000);
        BEGIN
          --Récupération de la valeur de la ligne de la table temporaire en cours de traitement
          EXECUTE IMMEDIATE 'SELECT ' ||REC.COLUMN_NAME || CHR(10) ||
                            'FROM '   || v_TBL_TEMP || CHR(10) ||
                            'WHERE  ROWID = '''  || r_TEMP_ROWID || ''''
                      INTO v_TEMP;
          --
          IF REC.DATA_TYPE IN ('VARCHAR2','CHAR','NVARCHAR2','NCHAR') THEN
            --Si le champ est de type chaine de caractère
            --alors on ajoute des cotes
            v_TEMP := '''' || REPLACE(v_TEMP,'''','''''') || '''';
          END IF;
          --
          --Substitution de la colonne par sa valeur
          v_REQ := REPLACE(v_REQ,':'||REC.COLUMN_NAME,v_TEMP);
        END;
      END IF;
    END LOOP;

    --Ouverture de la requête sous forme d'un curseur
    LV SEQ := '450';
    OPEN RC_RESULT FOR v_REQ;
    --
    LOOP
      --
      --Récupération des données
      LV_SEQ := '460';
      FETCH RC_RESULT INTO v_RESULT;
      --
      --S'il n'y a plus de donnée alors sortir de la boucle
      EXIT WHEN RC_RESULT%NOTFOUND ;
      --
      --Ajout du résultat de la requête dans la colonne AUDIT_COMPL de la ligne
      --de la table temporaire en cours de traitement
      v REQ := 'UPDATE ' || v TBL TEMP || CHR(10) ||
               'SET    AUDIT_COMPLT = AUDIT_COMPLT || NVL2(AUDIT_COMPLT,CHR(10),NULL) || ''     ' ||
                      REPLACE(TRIM(v_RESULT),'''','''''')  || '''' || CHR(10) ||
               'WHERE  ROWID = '''  || r_TEMP_ROWID || '''' ;
      EXECUTE IMMEDIATE v_REQ;
      COMMIT;
      --
    END LOOP;
    CLOSE RC_RESULT;
END;
--
```

```
            END IF;
            --
          ELSE                --> nœud inconnu = enregistrement d'une erreur
            PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_ACTION_UNKNOW',
                                           'Type d''action inconnue : ' ||v_NODE_NAME || ' AUDIT_COMP_ID :' || n_AUDIT_COMP_ID );
        END CASE;
      EXCEPTION
        WHEN OTHERS THEN
          --En cas d'erreur
          --Enregistrement de l'erreur Oracle
          PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_EXEC_ACTION',
                                         'Erreur oracle lors de l''exécution des actions complémentaires AUDIT_COMP_ID :' ||
                                         n_AUDIT_COMP_ID || ' ('||LV_SEQ||'):' || SQLERRM);
          --Enregistrement de la requête
          PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
          --Mémorisation de l'erreur
          n_ERR_ON_ACTION := 1;
      END;
  END LOOP; --Fin de parcours de la liste des actions à effectuer

  IF n_AUDIT_COMP_ID IS NOT NULL THEN
    --Mémorisation du statut d'erreur lors du traitement de ce complément
    v_REQ := 'UPDATE T_AUDIT_COMP' || CHR(10) ||
             'SET    ERR_ON_ACTION = '||n_ERR_ON_ACTION||  CHR(10) ||
             'WHERE  AUDIT_COMP_ID = '||n_AUDIT_COMP_ID  ;
    EXECUTE IMMEDIATE v_REQ;
    COMMIT;
  END IF;

EXCEPTION
  WHEN OTHERS THEN
    --En cas d'erreur
    --Enregistrement de l'erreur Oracle
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_OTH_EXEC_ACT',
                                   'Erreur oracle lors du traitement des actions complémentaires AUDIT_COMP_ID :' ||
                                    n_AUDIT_COMP_ID || ' ('||LV_SEQ||'):' || SQLERRM);
    --Enregistrement de la requête
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
    --Mémorisation de l'erreur
    n_ERR_ON_ACTION := 1;

    IF n_AUDIT_COMP_ID IS NOT NULL THEN
      --Mémorisation du statut d'erreur lors du traitement de ce complément
      v_REQ := 'UPDATE T_AUDIT_COMP' || CHR(10) ||
               'SET    ERR_ON_ACTION = '||n_ERR_ON_ACTION||  CHR(10) ||
               'WHERE  AUDIT_COMP_ID = '||n_AUDIT_COMP_ID  ;
      EXECUTE IMMEDIATE v_REQ;
      COMMIT;
    END IF;
END;
--
```

```
        END IF;

      END LOOP; -- Fin de parcours de la table complémentaire et des actions à effectuer

      --Fermeture du curseur
      LV_SEQ := '370';
      CLOSE rc_ACTION;

    EXCEPTION
      WHEN OTHERS THEN
        --En cas d'erreur
        --Enregistrement de l'erreur Oracle
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_OTH_ACTION',
                                       'Erreur oracle lors de la recherche des actions complémentaires ('||LV_SEQ||'):' || SQLERRM);
        --Enregistrement de la requête
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
    END;

END;
--
--Transfert des données de la table temporaire à la table définitive
DECLARE
  LV_COLUMNS_LIST_TEMP VARCHAR2(32767);
  LV_COLUMNS_LIST_STCK VARCHAR2(32767);
BEGIN

  --Vérification de l'existence de la table de stockage
  LV_SEQ := '470';
  SELECT TABLE_NAME
  INTO   v_REQ
  FROM   ALL_TABLES
  WHERE  TABLE_NAME = LRT_AUDIT.RESULT_TBL;

  --Suppression des anciennes données en mode REPLACE
  LV_SEQ := '480';
  IF LRT_AUDIT.INSERT_MODE = 'REPLACE' THEN
       --
       BEGIN
         --
         v_REQ  := 'DELETE FROM ' || LRT_AUDIT.RESULT_TBL || CHR(10) ||
                   'WHERE  AUDIT_DB           = '''|| PV_DB_LINK || '''';
         --
         IF LRT_AUDIT.COL_KEY_FOR_REPLACE IS NOT NULL THEN
             v_REQ  := v_REQ   || CHR(10) ||
                       'AND (' || LRT_AUDIT.COL_KEY_FOR_REPLACE || ') IN ' || CHR(10) ||
                       '     (SELECT ' || LRT_AUDIT.COL_KEY_FOR_REPLACE ||  CHR(10) ||
                       '      FROM   ' || v_TBL_TEMP || ')';
         END IF;
         --
```

```
        EXECUTE IMMEDIATE v_REQ;
        COMMIT;
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Anciennes données supprimées' ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
      EXCEPTION
        WHEN OTHERS THEN
          PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,'LAUNCH AUDIT ERR DROP OLD DATA',
                                                    'Impossible supprimer les anciennes données ('||LV_SEQ||'):' || SQLERRM);
          PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
      END;
       --
 END IF;


 --Ajout des colonnes manquantes dans la table résultats
 LV_SEQ := '491';
 FOR REC IN (SELECT *
              FROM  ALL_TAB_COLUMNS A
              WHERE TABLE NAME LIKE v TBL_TEMP
              AND   NOT EXISTS (SELECT 1
                                FROM   ALL_TAB_COLUMNS B
                                WHERE  B.TABLE_NAME LIKE LRT_AUDIT.RESULT_TBL
                                AND    A.COLUMN_NAME =B.COLUMN_NAME)
              ORDER BY COLUMN_NAME) LOOP

     BEGIN
      IF REC.DATA_TYPE != 'VARCHAR2' THEN
        V REQ := 'ALTER TABLE ' || LRT_AUDIT.RESULT_TBL || ' ADD ' || REC.COLUMN_NAME || ' ' || REC.DATA_TYPE;
      ELSE
        V_REQ := 'ALTER TABLE ' || LRT_AUDIT.RESULT_TBL || ' ADD ' || REC.COLUMN_NAME || ' ' || REC.DATA_TYPE
                 || '('||REC.DATA_LENGTH|| ')';
      END IF;
      EXECUTE IMMEDIATE v_REQ;
      COMMIT;
      PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL, 'Colonne ' || REC.COLUMN NAME || ' ajouté à ' ||
                                     v_TBL_TEMP,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
    EXCEPTION
      WHEN OTHERS THEN
        PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_ADD_COL_TMP', 'Impossible d''ajouté la colonne ' ||
                                       REC.COLUMN_NAME || ' à ' ||  LRT_AUDIT.RESULT_TBL || '('||LV_SEQ||'):' || SQLERRM);
        PKG AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
        RETURN;
    END;
END LOOP;

--Ajout des colonnes manquantes dans la table temporaire
LV SEQ := '491';
FOR REC IN (SELECT *
              FROM  ALL_TAB_COLUMNS A
              WHERE TABLE_NAME LIKE LRT_AUDIT.RESULT_TBL
              AND   NOT EXISTS (SELECT 1 FROM ALL_TAB_COLUMNS B WHERE B.TABLE_NAME LIKE v_TBL_TEMP AND A.COLUMN_NAME =B.COLUMN_NAME)
              ORDER BY COLUMN_NAME) LOOP
```

```
                        BEGIN
                         IF REC.DATA_TYPE != 'VARCHAR2' THEN
                           V_REQ := 'ALTER TABLE ' || v_TBL_TEMP || ' ADD ' || REC.COLUMN_NAME || ' ' || REC.DATA_TYPE;
                         ELSE
                           V REQ := 'ALTER TABLE ' || v_TBL_TEMP || ' ADD ' || REC.COLUMN_NAME || ' ' || REC.DATA_TYPE || '('||REC.DATA_LENGTH|| ')';
                         END IF;
                         EXECUTE IMMEDIATE v_REQ;
                         COMMIT;
                         PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Colonne ' || REC.COLUMN_NAME || ' ajoutée à ' ||  v_TBL_TEMP
                                                        ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
                       EXCEPTION
                         WHEN OTHERS THEN
                           PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_ADD_COL_TMP', 'Impossible d''ajouté la colonne ' ||
                                                                                REC.COLUMN_NAME || ' à ' ||  LRT_AUDIT.RESULT_TBL || '('||LV_SEQ||'):' || SQLERRM);
                           PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
                       END;
                     END LOOP;

                      --Récupération des colonnes de la table temporaire
                      LV_SEQ := '492';

                      SELECT COLUMNS_LIST
                      INTO   LV COLUMNS_LIST_TEMP
                      FROM (SELECT LINE
                                  ,LTRIM(SYS_CONNECT_BY_PATH(COLUMN_NAME,',') ,',') AS COLUMNS_LIST
                            FROM ( SELECT ROWNUM AS LINE
                                          ,COLUMN NAME
                                   FROM (SELECT COLUMN_NAME
                                         FROM ALL_TAB_COLUMNS
                                         WHERE TABLE_NAME LIKE v_TBL_TEMP
                                         ORDER BY COLUMN_NAME)
                                 )
                            CONNECT BY PRIOR LINE = LINE-1
                            START WITH LINE = 1
                            ORDER BY LINE DESC
                           )
                      WHERE ROWNUM=1;
                      LV_SEQ := '500';
                      PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT_TRACE_ID,NULL, 'Colonnes de la table temporaire :  ' ||
LV_COLUMNS_LIST_TEMP,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);

                      --Récupération des colonnes de la table de stockage
                      LV_SEQ := '510';
                      SELECT COLUMNS_LIST
                      INTO   LV COLUMNS_LIST_STCK
                      FROM (SELECT LINE
                                  ,LTRIM(SYS_CONNECT_BY_PATH(COLUMN_NAME,',') ,',') AS COLUMNS_LIST
                            FROM ( SELECT ROWNUM AS LINE
                                          ,COLUMN NAME
                                   FROM (SELECT COLUMN_NAME
```

```
                                    FROM ALL_TAB_COLUMNS
                                    WHERE TABLE NAME LIKE  LRT_AUDIT.RESULT_TBL
                                    ORDER BY COLUMN_NAME)
                               )
                          CONNECT BY PRIOR LINE = LINE-1
                          START WITH LINE = 1
                          ORDER BY LINE DESC
                        )
                    WHERE ROWNUM=1;
                    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Colonnes de la table stockage   :  ' ||
SUBSTR(LV_COLUMNS_LIST_STCK,1,LENGTH(LV_COLUMNS_LIST_STCK)),PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);


                    IF  LV_COLUMNS_LIST_STCK != LV_COLUMNS_LIST_TEMP THEN
                      --Si la table temporaire ne comporte pas les mêmes colonnes que la table définitive
                      --alors il est impossible de transférer les données
                      LV_SEQ := '520';
                      PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,'LAUNCH AUDIT ERR STRUCT TBL', 'La structure de la table de temporaire
                                                     ne correspond pas à celle de table de stockage : ' || LV_COLUMNS_LIST_TEMP || '<>'
                                                     || LV_COLUMNS_LIST_STCK);
                    ELSE
                      --Si la table temporaire comporte les mêmes colonnes que la table définitive

                      --Vérification de la taille de colonne de type texte
                      FOR REC IN (SELECT ATC1.COLUMN_NAME
                                        ,ATC1.DATA_TYPE      DATA_TYPE_STCK
                                        ,ATC1.DATA LENGTH    DATA LENGTH STCK
                                        ,ATC1.DATA PRECISION DATA PRECISION STCK
                                        ,ATC1.DATA_SCALE     DATA_SCALE_STCK
                                        ,ATC2.DATA_TYPE      DATA_TYPE_TMP
                                        ,ATC2.DATA_LENGTH    DATA_LENGTH_TMP
                                        ,ATC2.DATA_PRECISION DATA_PRECISION_TMP
                                        ,ATC2.DATA_SCALE     DATA_SCALE_TMP
                                  FROM  ALL TAB COLUMNS ATC1
                                       ,ALL TAB COLUMNS ATC2
                                  WHERE ATC1.TABLE_NAME LIKE LRT_AUDIT.RESULT_TBL
                                  AND   ATC2.TABLE_NAME LIKE v_TBL_TEMP
                                  AND   ATC1.COLUMN_NAME = ATC2.COLUMN_NAME
                                  ORDER BY ATC1.COLUMN_NAME)LOOP
                         IF REC.DATA TYPE STCK IN ('CHAR','VARCHAR2') AND  NVL(REC.DATA_LENGTH_TMP,0) > NVL(REC.DATA_LENGTH_STCK,0) THEN
                            --Si les colonnes sont de de types chaine de caractère
                            --alors adaptation de la taille de la colonne de la table définitive
                            EXECUTE IMMEDIATE 'ALTER TABLE ' || LRT_AUDIT.RESULT_TBL || ' MODIFY ' || REC.COLUMN_NAME || ' ' || REC.DATA_TYPE_STCK
                                              || '(' || REC.DATA_LENGTH_TMP || ')';
                            PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Modification de la colonne '|| REC.COLUMN_NAME
                                                           || '   ' ||REC.DATA TYPE STCK || '(' || NVL(REC.DATA LENGTH STCK,0) || ')=>'
                                                           ||  REC.DATA TYPE STCK || '(' ||NVL(REC.DATA_LENGTH_TMP,0)||')'
                                                          ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
                         END IF;
                      END LOOP;
```

```
  --Construction des requêtes de création et de suppression des indexes
  LV SEQ := '521';
  v_REQ_CREATE_INDEX:=NULL;
  v_REQ_DROP_INDEX  :=NULL;
  FOR REC IN (SELECT *
              FROM   ALL INDEXES
              WHERE  TABLE_NAME=LRT_AUDIT.RESULT_TBL
              AND    INDEX_TYPE='NORMAL'
              AND    OWNER     ='AUDITU'
              ORDER BY INDEX_NAME
              ) LOOP

     v_REQ_DROP_INDEX := v_REQ_DROP_INDEX || 'DROP INDEX "' || REC.OWNER ||'"."'||REC.INDEX_NAME||'";'||CHR(10);


     SELECT v_REQ_CREATE_INDEX || DBMS_METADATA.get_ddl ('INDEX', REC.INDEX_NAME, REC.OWNER) ||';'|| CHR(10)
     INTO   v_REQ_CREATE_INDEX
     FROM DUAL;
  END LOOP;
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Requête de suppression des index:' ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, v_REQ_DROP_INDEX ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Requête de re-création des index:' ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, v_REQ_CREATE_INDEX ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);

  --Transfert des données de la table temporaire à la table de stockage avec ajout des données de gestion
  LV_SEQ := '530';
  PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,NULL, 'Début du transfert de donnée vers la table de stockage'
                                ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
  BEGIN
     v_REQ  := 'INSERT INTO ' || LRT_AUDIT.RESULT_TBL ||
                           '(' || LV_COLUMNS_LIST_STCK || ') '|| CHR(10) ||
                    'SELECT ' || LV_COLUMNS_LIST_TEMP || CHR(10) ||
                    'FROM ' || v_TBL_TEMP;
     EXECUTE IMMEDIATE v_REQ;
     COMMIT;
     PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Transfert des données vers la table '|| LRT_AUDIT.RESULT_TBL
                                    || ' effectuées' ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
   EXCEPTION
     WHEN OTHERS THEN
       PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,'LAUNCH_AUDIT_ERR_INSERT', 'Impossible transerfer les données ('||LV_SEQ
                                      ||'):' || SQLERRM);
       PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,v_REQ);
   END;
   PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Fin du transfert de donnée vers la table de stockage'
                                 ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);

END IF;

--Suppression de la table temporaire
LV SEQ := '540';
BEGIN
```

```
              v_REQ := 'DROP TABLE ' || v_TBL_TEMP || ' PURGE';
              EXECUTE IMMEDIATE v_REQ;
              COMMIT;
              PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Table ' || v_TBL_TEMP ||' supprimée',PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
            EXCEPTION WHEN OTHERS THEN
              PKG AUDIT COMMUN.SP ECRIRE MSG(LN AUDIT TRACE ID,'LAUNCH AUDIT ERR DROP TBL', 'Impossible de supprimer la table temporaire ' ||
                                                                                    v_TBL_TEMP || '('||LV_SEQ||'):' || SQLERRM);
              b_ERR := TRUE;
            END;

          EXCEPTION
            WHEN NO DATA FOUND THEN
              --Si la table définitive n'existe pas
              --alors renomage de la table temporaire
              LV_SEQ := '550';
              EXECUTE IMMEDIATE 'RENAME ' || v_TBL_TEMP || ' TO ' || LRT_AUDIT.RESULT_TBL;
              PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Table ' || v_TBL_TEMP ||' renommée ' || LRT_AUDIT.RESULT_TBL
                                             ,PKG_AUDIT_COMMUN.GN_NIVEAU_LOG_DEBUG);
          END;

      END IF; --FinSi la table temporaire a été crée

    END IF; -- FinS'il faut exécuter la requête d'audit

  END;

END IF; --FinSi le DB Link a été créé
--
IF PN_AUDIT_TRACE_ID_PARENT IS NULL THEN
  --
  --Si ce n'est pas un audit fils
  --Fermeture du DB Link
  LV_SEQ := '560';
  IF NOT PKG AUDIT COMMUN.FB CLOSE DBLINK(PV DB LINK,LN AUDIT TRACE ID) THEN
     PKG AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_CLOSE', 'Impossible de se déconnecter de ' || PV_DB_LINK);
  END IF;
  --
  --Mise à jour de la table des tailles des résultats
  LV_SEQ := '561';
  SP_COMPUTE_AUDIT_TAILLE_RES(LN_AUDIT_TRACE_ID);

  --Ecriture de la fin de l'audit
  LV_SEQ := '570';
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'-------------------------------------------------------------------------------');
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL, 'Fin de l''audit : ' || LRT_AUDIT.NAME || ' (' || LRT_AUDIT.DESCRIPTION || ' : '
                                                      || LRT AUDIT REQ.DESCRIPTION || ') sur ' || PV DB LINK);
  PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,NULL,'-------------------------------------------------------------------------------');
  --
END IF;
--
END IF; --FinSi l'identifiant de la requête a exécuter n'a pas été précisé
```

```
    --
  END IF; --FinSi l'identifiant de l'audit a exécuter n'a pas été précisé
  --
  --Finalisation de la trace
  LV SEQ := '580';
  IF LV STATUT = PKG_AUDIT_COMMUN.GV_STATUT_SUC THEN
    BEGIN
      SELECT DISTINCT PKG_AUDIT_COMMUN.GV_STATUT_ANO
      INTO   LV_STATUT
      FROM   T_AUDIT_MESSAGE TAM
      WHERE  TAM.AUDIT TRACE ID = LN AUDIT TRACE_ID
      AND    TAM.CODE_MESSAGE LIKE '%ERR%';
    EXCEPTION
      WHEN OTHERS THEN NULL;
    END;
  END IF;
  PKG_AUDIT_COMMUN.SP_END_TRACE(LN_AUDIT_TRACE_ID,LV_STATUT,LN_NB);
  --
  --Sortie du contexte
  LV_SEQ := '590';
  PKG_AUDIT_COMMUN.SP_LEAVE_CONTEXT;
  --
EXCEPTION
  WHEN OTHERS THEN
    --En cas d'erreur
    --Enregistrement de l'erreur Oracle
    PKG_AUDIT_COMMUN.SP_ECRIRE_MSG(LN_AUDIT_TRACE_ID,'LAUNCH_AUDIT_ERR_OTH', 'Erreur ORACLE ('||LV_SEQ||'):' || SQLERRM);
    --
    --Fermeture du DB Link
    IF NOT PKG_AUDIT_COMMUN.FB_CLOSE_DBLINK(PV_DB_LINK,LN_AUDIT_TRACE_ID) THEN NULL; END IF;
    --
    --Finalisation de la trace
    PKG_AUDIT_COMMUN.SP_END_TRACE(LN_AUDIT_TRACE_ID,PKG_AUDIT_COMMUN.GV_STATUT_ANO,LN_NB);
    --
    --Sortie du contexte
    PKG_AUDIT_COMMUN.SP_LEAVE_CONTEXT;
END SP_LAUNCH_AUDIT;
```

Annexe 4

Exemple de trace du mécanisme de collecte

```
SP_LAUNCH_JOB                  --------------------------------------------------------------------------------
SP_LAUNCH_JOB                  Début du lancement du job : 141
SP_LAUNCH_JOB                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Début de l'audit : RECORD_STATUS_RDO (Audit des RECORD_STATUS de la RDO SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0027 - MAJ EP,IG,MUNI
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0039 - MAJ Relation Payeur
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0040 - MAJ Relation Syndicat
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0020 - Chargement des RIBs
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0133 - Reprise des adresses de facturation
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0028 - Chargement compte-client
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0080 - Découpage des données TAO TLG
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0092 - Consolidation des contrats RLP
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0025 - Consolidation des contrats
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0029 - Injection des contrats
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0087 - Reprise des échéanciers TLG
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0023 - Consolidation des mouvements
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0024 - Injection des mouvements
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0084 - Chargement des périodes de référence SPRE
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Fin de l'audit : RECORD_STATUS_RDO (Audit des RECORD_STATUS de la RDO SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Début de l'audit : RECORD_STATUS_INTF (Audit des RECORD_STATUS de l'interface eBS-FAP SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0128 - Chargement des RIBs
SP_LAUNCH_AUDIT                    ----------------------------
```

```
SP_LAUNCH_AUDIT                    SELDR0131 - Rattachement des RIBs aux comptes-clients
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0117 - Consolidation des mouvements
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0118 - Injection des mouvements
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0108 - Génération de fichier pour l'interface eBS-FAP (Anomalie avant la création des transactions)
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    SELDR0108 - Génération de fichier pour l'interface eBS-FAP (Transactions crées)
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Mainframe - Remonté des traitements de la FAP
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Fin de l'audit : RECORD_STATUS_INTF (Audit des RECORD_STATUS de l'interface eBS-FAP SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Début de l'audit : OBJECT_SIZE (Audit des tailles des objets (table et index) impacté par la RDO SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Volume utilisé par les tables et index spécifiques à la RDO SELDR
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Volume utilisé par les tables et index du spécifiques à l'interfaçage eBS-FAP SELDR
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Volume utilisé par les tables et index partagés par la RDO et l'interfaçage eBS-FAP SELDR
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Volume utilisé par les tables et index du spécifiques du projet SELDR  (hors RDO et interfaçage eBS-FAP)
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Volume utilisé sur les tables standard impactées par la RDO SELDR
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Fin de l'audit : OBJECT_SIZE (Audit des tailles des objets (table et index) impacté par la RDO SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Début de l'audit : TABLE_COUNT (Audit du nombre de lignes des tables de la RDO SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Répartition des messages dans SELDRT_MESSAGES
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Répartition des messages dans SELDRT_MESSAGES_HIS
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Répartition des indicateurs dans SELDRT_RDO_INDICATEURS
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Répartition des indicateurs dans SELDRT_RDO_INDICATEURS_HIS
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Fin de l'audit : TABLE_COUNT (Audit du nombre de lignes des tables de la RDO SELDR) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Début de l'audit : TRANS_INTF (Suivi des transactions Interface EBS vers FAP) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Requête de suivi des transactions Interface EBS vers FAP
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Fin de l'audit : TRANS_INTF (Suivi des transactions Interface EBS vers FAP) sur SLREC
```

```
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Début de l'audit : DATE_REPRISE_RDO (Liste des délégations RDO) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Requête des délégations RDO
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Fin de l'audit : DATE_REPRISE_RDO (Liste des délégations RDO) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Début de l'audit : GTS (Liste des lancements GTS) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                    ----------------------------
SP_LAUNCH_AUDIT                    Requête des lancements GTS
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_AUDIT                  Fin de l'audit : GTS (Liste des lancements GTS) sur SLREC
SP_LAUNCH_AUDIT                  --------------------------------------------------------------------------------
SP_LAUNCH_JOB                  --------------------------------------------------------------------------------
SP_LAUNCH_JOB                  Fin du lancement du job : 141
SP_LAUNCH_JOB                  --------------------------------------------------------------------------------
```

Annexe 5

Définition de type de document pour le fichier de configuration XML de DaPCol-Reports

config.dtd

```
<!ELEMENT config (data_base, jrxml_files ) >

  <!ELEMENT data_base ( driver, connection_string, login, password ) >
    <!ELEMENT driver ( #PCDATA ) >
    <!ELEMENT connection_string ( #PCDATA ) >
    <!ELEMENT login ( #PCDATA ) >
    <!ELEMENT password ( #PCDATA ) >

  <!ELEMENT jrxml_files ( jrxml_file+ ) >
    <!ELEMENT jrxml_file ( identifier, title , file , parameters, requete, subDataset*) >
      <!ELEMENT identifier ( #PCDATA ) >
      <!ELEMENT title ( #PCDATA ) >
      <!ELEMENT file ( #PCDATA ) >
      <!ELEMENT parameters (parameter*) >
        <!ATTLIST parameter label CDATA #IMPLIED >
        <!ATTLIST parameter name NMTOKEN #REQUIRED >
        <!ATTLIST parameter visible ( NO | YES ) #REQUIRED >
        <!ELEMENT parameter ( SQL_values | dependent_values | value )* >
          <!ELEMENT SQL_values ( #PCDATA ) >
          <!ELEMENT dependent_values ( #PCDATA ) >
          <!ELEMENT value ( #PCDATA ) >

      <!ELEMENT requete ( #PCDATA ) >
      <!ELEMENT subDataset ( requete ) >
      <!ATTLIST subDataset NMTOKEN #REQUIRED >
```

Annexe 6

Exemple de fichier de configuration générale

config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE config SYSTEM "config.dtd">
<config>
  <data_base>
    <driver>oracle.jdbc.driver.OracleDriver</driver>
    <connection_string>jdbc:oracle:thin:@localhost:1521:XE</connection_string>
    <login>AUDITU</login>
    <password>AUDITU</password>
  </data_base>
  <jrxml_files>
    <!-- ================================================================================================================================================== -->
    <jrxml_file>
      <identifier>RDO_0</identifier>
      <title>Suivi de la reprise de donnée SELDR</title>
      <file>Suivi_Semaine.jrxml</file>
      <parameters>
        <parameter name="ENV" label="Base" visible="YES">
          <SQL_values>
            <![CDATA[
            SELECT DBLINK VALUE, DBLINK TEXT
            FROM   T_AUDIT_DBLINK
            ORDER BY DECODE(DBLINK,'SL2DEV',0,'SLFOR',10,'SLPRE',11,'SLREC',12,'SLDEX',13,'SLPRD',20,-1),DBLINK            ]]>
          </SQL_values>
        </parameter>
        <parameter name="DRDL" label="Délégation" visible="YES">
          <SQL_values>
            <![CDATA[
            SELECT DRDL VALUE ,LPAD(DRDL,4,0) || ' - ' || RPAD(NVL(NOM,' '),30) || '(' || TO_CHAR(DATE_REPRISE,'DD-MON-YY') || ')' TEXT,DRDL KEY
            FROM   T RES AUDIT DATE REPRISE_RDO
            WHERE  AUDIT DB = :{ENV}
            AND    DATE_REPRISE IS NOT NULL
            UNION ALL
            SELECT NULL VALUE,'(Toutes)' TEXT,-1
            FROM   DUAL
            ORDER BY KEY
            ]]>
          </SQL_values>
        </parameter>
        <parameter name="DATE_DEB1" label="Date t" visible="YES">
          <SQL_values>
            <![CDATA[
```

```
    SELECT AUDIT_EXTRACT_DATE VALUE,TO_CHAR(AUDIT_EXTRACT_DATE,'DAY DD MONTH YYYY') TEXT
    FROM   (SELECT DISTINCT TRUNC(AUDIT EXTRACT DATE) AUDIT_EXTRACT_DATE
            FROM   T_RES_AUDIT_RECORD_STATUS_RDO
            WHERE  AUDIT_DB = :{ENV})
    ORDER BY VALUE DESC
    ]]>
  </SQL_values>
</parameter>
<parameter name="DATE_DEB2" label="Date t-1" visible="YES">
  <dependent_values>
    <![CDATA[
    < :{DATE_DEB1}
    ]]>
  </dependent_values>
</parameter>
<parameter name="DATE_DEB3" label="Date t-2" visible="YES">
  <dependent_values>
    <![CDATA[
    < :{DATE_DEB2}
    ]]>
  </dependent_values>
</parameter>
<parameter name="DATE_DEB4" label="Date t-3" visible="YES">
  <dependent_values>
    <![CDATA[
    < :{DATE_DEB3}
    ]]>
  </dependent_values>
</parameter>
<parameter name="DATE_DEB5" label="Date t-4" visible="YES">
  <dependent_values>
    <![CDATA[
    < :{DATE_DEB4}
    ]]>
  </dependent_values>
</parameter>
<parameter name="DATE_DEB6" label="Date t-5" visible="YES">
  <dependent_values>
    <![CDATA[
    < :{DATE_DEB5}
    ]]>
  </dependent_values>
</parameter>
<parameter name="DATE_DEB7" label="Date t-6" visible="YES">
  <dependent_values>
    <![CDATA[
    < :{DATE_DEB6}
    ]]>
  </dependent_values>
</parameter>
<parameter name="DATE_DEB8" label="Date t-7" visible="YES">
```

```
        <dependent_values>
          <![CDATA[
          < :{DATE_DEB7}
          ]]>
        </dependent_values>
      </parameter>
      <parameter name="TITRE" visible="NO">
        <value>Suivi de la reprise de donnée SELDR</value>
      </parameter>
      <parameter name="TITRE_GRAPHS" visible="NO">
        <value>Évolution des anomalies de la reprise de donnée SELDR</value>
      </parameter>
      <parameter name="TITRE_GRAPH_1" visible="NO">
        <value>Reprise des données interlocuteurs</value>
      </parameter>
      <parameter name="TITRE_GRAPH_2" visible="NO">
        <value>Création des comptes clients</value>
      </parameter>
      <parameter name="TITRE_GRAPH_3" visible="NO">
        <value>Reprise des contrats</value>
      </parameter>
      <parameter name="TITRE_GRAPH_4" visible="NO">
        <value>Reprise des échéances</value>
      </parameter>
    </parameters>
    <requete>
      <![CDATA[
SELECT TRT, SRC, RECORD STATUS, CODE ANO
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB1})+1-1/(24*60*60) BETWEEN  AUDIT_EXTRACT_DATE AND NVL(AUDIT_VALIDITY_END_DATE,AUDIT_EXTRACT_DATE+1) THEN NB ELSE 0 END) J1
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB2})+1-1/(24*60*60) BETWEEN  AUDIT_EXTRACT_DATE AND NVL(AUDIT_VALIDITY_END_DATE,AUDIT_EXTRACT_DATE+1) THEN NB ELSE 0 END) J2
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB3})+1-1/(24*60*60) BETWEEN  AUDIT_EXTRACT_DATE AND NVL(AUDIT_VALIDITY_END_DATE,AUDIT_EXTRACT_DATE+1) THEN NB ELSE 0 END) J3
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB4})+1-1/(24*60*60) BETWEEN  AUDIT_EXTRACT_DATE AND NVL(AUDIT_VALIDITY_END_DATE,AUDIT_EXTRACT_DATE+1) THEN NB ELSE 0 END) J4
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB5})+1-1/(24*60*60) BETWEEN  AUDIT_EXTRACT_DATE AND NVL(AUDIT_VALIDITY_END_DATE,AUDIT_EXTRACT_DATE+1) THEN NB ELSE 0 END) J5
     ,SUM(CASE WHEN TRUNC($P{DATE DEB6})+1-1/(24*60*60) BETWEEN  AUDIT EXTRACT DATE AND NVL(AUDIT VALIDITY END DATE,AUDIT EXTRACT DATE+1) THEN NB ELSE 0 END) J6
     ,SUM(CASE WHEN TRUNC($P{DATE DEB7})+1-1/(24*60*60) BETWEEN  AUDIT EXTRACT DATE AND NVL(AUDIT VALIDITY END DATE,AUDIT EXTRACT DATE+1) THEN NB ELSE 0 END) J7
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB8})+1-1/(24*60*60) BETWEEN  AUDIT_EXTRACT_DATE AND NVL(AUDIT_VALIDITY_END_DATE,AUDIT_EXTRACT_DATE+1) THEN NB ELSE 0 END) J8
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB1}) = TRUNC(AUDIT_EXTRACT_DATE)                                    THEN  1
               WHEN TRUNC($P{DATE_DEB1}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB1}) > B.MAX_EXTRACT_DATE   THEN -1
          ELSE  0 END) V1
     ,SUM(CASE WHEN TRUNC($P{DATE DEB2}) = TRUNC(AUDIT EXTRACT DATE)                                    THEN  1
               WHEN TRUNC($P{DATE_DEB2}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB2}) > B.MAX_EXTRACT_DATE   THEN -1
          ELSE  0 END) V2
     ,SUM(CASE WHEN TRUNC($P{DATE_DEB3}) = TRUNC(AUDIT_EXTRACT_DATE)                                    THEN  1
               WHEN TRUNC($P{DATE_DEB3}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB3}) > B.MAX_EXTRACT_DATE   THEN -1
          ELSE  0 END) V3
     ,SUM(CASE WHEN TRUNC($P{DATE DEB4}) = TRUNC(AUDIT EXTRACT DATE)                                    THEN  1
               WHEN TRUNC($P{DATE_DEB4}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB4}) > B.MAX_EXTRACT_DATE   THEN -1
          ELSE  0 END) V4

      ,SUM(CASE WHEN TRUNC($P{DATE DEB5}) = TRUNC(AUDIT EXTRACT DATE)                                     THEN  1
                WHEN TRUNC($P{DATE_DEB5}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB5}) > B.MAX_EXTRACT_DATE   THEN -1
```

```
           ELSE  0 END) V5
      ,SUM(CASE WHEN TRUNC($P{DATE DEB6}) = TRUNC(AUDIT EXTRACT DATE)                                    THEN  1
                WHEN TRUNC($P{DATE_DEB6}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB6}) > B.MAX_EXTRACT_DATE   THEN -1
           ELSE  0 END) V6
      ,SUM(CASE WHEN TRUNC($P{DATE DEB7}) = TRUNC(AUDIT EXTRACT DATE)                                    THEN  1
                WHEN TRUNC($P{DATE_DEB7}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB7}) > B.MAX_EXTRACT_DATE   THEN -1
           ELSE  0 END) V7
      ,SUM(CASE WHEN TRUNC($P{DATE_DEB8}) = TRUNC(AUDIT_EXTRACT_DATE)                                    THEN  1
                WHEN TRUNC($P{DATE_DEB8}) < B.MIN_EXTRACT_DATE OR TRUNC($P{DATE_DEB8}) > B.MAX_EXTRACT_DATE   THEN -1
           ELSE 0 END) V8
FROM   T RES AUDIT RECORD STATUS RDO A
      ,(SELECT MIN(AUDIT EXTRACT DATE) MIN EXTRACT DATE,
               MAX(AUDIT_EXTRACT_DATE) MAX_EXTRACT_DATE
        FROM   T_RES_AUDIT_RECORD_STATUS_RDO
        WHERE  AUDIT_DB = $P{ENV}
        AND    NO_DRDL  = NVL($P{DRDL},NO_DRDL)) B
WHERE  AUDIT_DB = $P{ENV}
AND    NO DRDL  = NVL($P{DRDL},NO DRDL)
AND   (   (TRUNC(AUDIT_EXTRACT_DATE) >=  TRUNC($P{DATE_DEB8})-1 AND    TRUNC(AUDIT_EXTRACT_DATE) <=  TRUNC($P{DATE_DEB1}))
       OR (TRUNC(AUDIT_VALIDITY_END_DATE) >=  TRUNC($P{DATE_DEB8})-1 AND    TRUNC(AUDIT_VALIDITY_END_DATE) <=  TRUNC($P{DATE_DEB1})))
GROUP BY TRT, SRC, RECORD_STATUS, CODE_ANO
ORDER BY DECODE (TRT,'SELDR0027','05'
                    ,'SELDR0039','10'
                    ,'SELDR0040','15'
                    ,'SELDR0020','20'
                    ,'SELDR0133','25'
                    ,'SELDR0028','30'
                    ,'SELDR0080','35'
                    ,'SELDR0025','40'
                    ,'SELDR0092','42'
                    ,'SELDR0029','45'
                    ,'SELDR0087','50'
                    ,'SELDR0023','55'
                    ,'SELDR0024','60'
                    ,'SELDR0084','65'
                                ,'70')
        ,DECODE(RECORD_STATUS,'SUC',0,'WRN',1,'ANO',2,NULL ,3,'NT' ,4,'DCH',5,'HP' ,6,'BLQ',7,8)
        ,REPLACE(CODE_ANO,'_',' '),REPLACE(SRC,'_',' ')
      ]]>
     </requete>
```

```
<subDataset name="GRAPH_1_DATASET"><requete>
    <![CDATA[
      SELECT AUDIT_EXTRACT_DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
      FROM   T_RES_AUDIT_RECORD_STATUS_RDO
      WHERE  AUDIT DB = $P{ENV}
      AND    NO DRDL = NVL($P{DRDL},NO DRDL)
      AND    TRUNC(AUDIT_EXTRACT_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
      AND    TRUNC(AUDIT_EXTRACT_DATE)      >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
      AND TRT IN ('SELDR0027','SELDR0039','SELDR0040')
      GROUP BY AUDIT_EXTRACT_DATE,TRT
      UNION ALL
      SELECT AUDIT VALIDITY END DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
      FROM   T_RES_AUDIT_RECORD_STATUS_RDO
      WHERE  AUDIT_DB = $P{ENV}
      AND    NO_DRDL = NVL($P{DRDL},NO_DRDL)
      AND    TRUNC(AUDIT_VALIDITY_END_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
      AND    TRUNC(AUDIT_VALIDITY_END_DATE) >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
      AND  AUDIT VALIDITY END DATE IS NOT NULL
      AND TRT IN ('SELDR0027','SELDR0039','SELDR0040')
      GROUP BY AUDIT_VALIDITY_END_DATE,TRT
      ORDER BY TRT, AUDIT_DATE
    ]]>
</requete></subDataset>
<subDataset name="GRAPH_2_DATASET"><requete>
    <![CDATA[
      SELECT AUDIT_EXTRACT_DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
      FROM   T RES AUDIT RECORD_STATUS_RDO
      WHERE  AUDIT DB = $P{ENV}
      AND    NO_DRDL = NVL($P{DRDL},NO_DRDL)
      AND    TRUNC(AUDIT_EXTRACT_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
      AND    TRUNC(AUDIT_EXTRACT_DATE) >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
      AND TRT IN ('SELDR0020','SELDR0133','SELDR0028')
      GROUP BY AUDIT_EXTRACT_DATE,TRT
      UNION ALL
      SELECT AUDIT VALIDITY END DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
      FROM   T_RES_AUDIT_RECORD_STATUS_RDO
      WHERE  AUDIT_DB = $P{ENV}
      AND    NO_DRDL = NVL($P{DRDL},NO_DRDL)
      AND    TRUNC(AUDIT_VALIDITY_END_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
      AND    TRUNC(AUDIT VALIDITY END DATE) >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
      AND  AUDIT VALIDITY END DATE IS NOT NULL
      AND TRT IN ('SELDR0020','SELDR0133','SELDR0028')
      GROUP BY AUDIT_VALIDITY_END_DATE,TRT
      ORDER BY TRT, AUDIT_DATE
    ]]>
</requete></subDataset>
```

```
    <subDataset name="GRAPH_3_DATASET"><requete>
        <![CDATA[
          SELECT AUDIT_EXTRACT_DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
          FROM   T_RES_AUDIT_RECORD_STATUS_RDO
          WHERE  AUDIT DB = $P{ENV}
          AND    NO DRDL = NVL($P{DRDL},NO DRDL)
          AND    TRUNC(AUDIT_EXTRACT_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
          AND    TRUNC(AUDIT_EXTRACT_DATE) >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
          AND TRT IN ('SELDR0117','SELDR0118')
          GROUP BY AUDIT_EXTRACT_DATE,TRT
          UNION ALL
          SELECT AUDIT VALIDITY END DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
          FROM   T_RES_AUDIT_RECORD_STATUS_RDO
          WHERE  AUDIT_DB = $P{ENV}
          AND    NO_DRDL = NVL($P{DRDL},NO_DRDL)
          AND    TRUNC(AUDIT_VALIDITY_END_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
          AND    TRUNC(AUDIT_VALIDITY_END_DATE) >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
          AND  AUDIT VALIDITY END DATE IS NOT NULL
          AND TRT IN ('SELDR0025','SELDR0092','SELDR0029','SELDR0087')
          GROUP BY AUDIT_VALIDITY_END_DATE,TRT
          ORDER BY TRT, AUDIT_DATE
        ]]>
    </requete></subDataset>
    <subDataset name="GRAPH_4_DATASET"><requete>
        <![CDATA[
          SELECT AUDIT_EXTRACT_DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
          FROM   T RES AUDIT RECORD_STATUS_RDO
          WHERE  AUDIT DB = $P{ENV}
          AND    NO_DRDL = NVL($P{DRDL},NO_DRDL)
          AND    TRUNC(AUDIT_EXTRACT_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
          AND    TRUNC(AUDIT_EXTRACT_DATE) >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
          AND TRT IN ('SELDR0023','SELDR0024','SELDR0084')
          GROUP BY AUDIT_EXTRACT_DATE,TRT
          UNION ALL
          SELECT AUDIT VALIDITY END DATE AUDIT_DATE,TRT,SUM(DECODE(RECORD_STATUS,'ANO',NB,0)) NB
          FROM   T_RES_AUDIT_RECORD_STATUS_RDO
          WHERE  AUDIT_DB = $P{ENV}
          AND    NO_DRDL = NVL($P{DRDL},NO_DRDL)
          AND    TRUNC(AUDIT_VALIDITY_END_DATE) <= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+8
          AND    TRUNC(AUDIT VALIDITY END DATE) >= TRUNC($P{DATE_DEB1})-To_NUMBER(To_CHAR($P{DATE_DEB1},'D'))+1-7*8
          AND  AUDIT VALIDITY END DATE IS NOT NULL
          AND TRT IN ('SELDR0023','SELDR0024','SELDR0084')
          GROUP BY AUDIT_VALIDITY_END_DATE,TRT
          ORDER BY TRT, AUDIT_DATE
        ]]>
    </requete></subDataset>
  </jrxml_file>
 </jrxml_files>
</config>
```

Annexe 7

Définition de type de document pour les fichiers des paramètres par la génération automatique et exemples

auto_launch.dtd

```
<!ELEMENT auto_launch (identifier, parameters,output_file) >
  <!ELEMENT identifier ( #PCDATA ) >
  <!ELEMENT parameters (parameter*) >
    <!ATTLIST parameter name NMTOKEN #REQUIRED >
    <!ELEMENT parameter ( #PCDATA ) >
    <!ELEMENT output_file ( #PCDATA ) >
```

Exemples :

RDO_0_SLPRD.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE auto_launch SYSTEM "auto_launch.dtd">
<auto_launch>
  <identifier>RDO_0</identifier>
  <parameters>
        <parameter name="ENV">SLPRD</parameter>
  </parameters>
  <output_file>Bilan RDO_SLPRD.pdf</output_file>
</auto_launch>
```

INTF_0_SLPRD.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE auto_launch SYSTEM "auto_launch.dtd">
<auto_launch>
  <identifier>INTF_0</identifier>
  <parameters>
        <parameter name="ENV">SLPRD</parameter>
  </parameters>
  <output_file>Bilan INTF_SLPRD.pdf</output_file>
</auto_launch>
```

Annexe 8

DaPCol-Reports : Classe IHM

IHM.java

```
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.*;

import javax.swing.*;
import java.io.*;
import java.math.BigDecimal;
import java.sql.*;

import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.design.*;
import net.sf.jasperreports.engine.xml.*;

import org.jdom.*;
import org.jdom.input.*;



public class IHM implements ItemListener,ActionListener,WindowListener{

  //Variable locale référençant le fichier XML da configuration (initialisé dans le constructeur)
  private org.jdom.Document DOC_XMLConfig,DOC_XMLAuto;

  //Variable locale référençant la connexion à la base d'audit
  private Connection Audit_DB_Conn = null;

  //Variable locale référençant la fenêtre principale de l'IHM
  private JFrame F_MainFrame = new JFrame();

  //Variable locale référençant la liste déroulante des rapports possible
  private JComboBox CB_Title = new JComboBox();

  //Variable locale référençant le panneau qui contiendra la liste des rapports
  JPanel P_Title = new JPanel();
  //Variable locale référençant le panneau qui contiendra tous les paramètres à définir par l'utilisateur
  private JPanel P_Parameters = new JPanel();
  //Variable locale référençant le panneau qui contiendra les boutons
  private JPanel P_Buttons = new JPanel();
```

```
private boolean IsAuto = false;

private int F_MainFrame_Orig_Height = 0;

//-----------------------------------------------------------------------------+
//
// Constructeur: report_launcher_ihm
// Créer une nouvelle IHM à partir d'un fichier XML de configuration.
//
// Type: Publique.
//
// Parameters:
//    doc_XMLConfig               IN - Fichier XML de configuration
//
// Exception: Aucune.
//
// Algorithme:
//>
//
// Historique:
//   01/09/2010      SA  -  Création.
//
//-----------------------------------------------------------------------------+
public IHM(org.jdom.Document doc_XMLConfig,org.jdom.Document doc_XMLAuto) {

  Context.SP_ENTER("report_launcher_ihm");

  //Sauvegarde du fichier XML de configuration dans une variable locale
  DOC_XMLConfig = doc_XMLConfig;
  DOC_XMLAuto   = doc_XMLAuto;

  //1-Lecture des paramètres de configuration
  String Driver_ClassName = DOC_XMLConfig.getRootElement().getChild("data_base").getChildText("driver");
  Trace.SP_ECRIRE("Driver_ClassName  = " + Driver_ClassName,Trace.GN_NIVEAU_LOG_DEBUG);

  String Connection_String = DOC_XMLConfig.getRootElement().getChild("data_base").getChildText("connection_string") ;
  Trace.SP_ECRIRE("Connection_String = " + Connection_String,Trace.GN_NIVEAU_LOG_DEBUG);

  String Login = DOC_XMLConfig.getRootElement().getChild("data_base").getChildText("login");
  Trace.SP_ECRIRE("Login             = " + Login,Trace.GN_NIVEAU_LOG_DEBUG);

  String PassWord = DOC_XMLConfig.getRootElement().getChild("data_base").getChildText("password");
  Trace.SP_ECRIRE("PassWord          = " + PassWord,Trace.GN_NIVEAU_LOG_DEBUG);

  try {
      //2-Initialisation de la class du driver JDBC Oracle
    Class.forName (Driver_ClassName);
    Trace.SP_ECRIRE("Driver OK",Trace.GN_NIVEAU_LOG_DEBUG);
      try {
```

```
//3-Ouverture de la connexion vers la base d'audit
Audit_DB_Conn = DriverManager.getConnection (Connection_String,Login,PassWord);
  Trace.SP_ECRIRE("Data Base Connection OK",Trace.GN_NIVEAU_LOG_DEBUG);

  //4-Définition des paramètres de la fenêtre
  Trace.SP_ECRIRE("Title");
  F_MainFrame.setTitle("Lanceur de rapport");
  F_MainFrame.addWindowListener(this);
  F_MainFrame.setName("F_MainFrame");
  F_MainFrame.setVisible(false);
  F_MainFrame_Orig_Height = F_MainFrame.getSize().height ;



  //5-Création d'un premier panneau pour la liste des rapports
  P_Title.setLayout(new FlowLayout());
  P_Title.setName("P_Title");
  F_MainFrame.add(P_Title,BorderLayout.NORTH);

  //5a-Ajout d'un label dans le premier panneau
  P_Title.add(new JLabel("Rapport à editer :"));

  //5b-Configuration de la liste déroulante des rapports possible
  //5b1 - Ajout d'un listener pour gérer les changements
  CB_Title.setName("TITLE");
  CB_Title.addItemListener(this);

  //5b2-Ajout de la liste déroulante dans le premier panneau
  P_Title.add(CB_Title);

  //6-Création d'un second panneau pour les paramètres à saisir par les utilisateurs
  //  Ce panneau sera remplis lors des changements de la précédente liste déroulantes
  Trace.SP_ECRIRE("Parameter");
  P_Parameters.setLayout(new GridLayout(0,2));
  P_Parameters.setBorder(new javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.LOWERED));
  P_Parameters.setName("P_Parameters");
  F_MainFrame.add(P_Parameters,BorderLayout.CENTER);

  //7-Création d'un dernier panneau pour les boutons
  Trace.SP_ECRIRE("Bouton");
  P_Buttons.setLayout(new FlowLayout());
  P_Buttons.setName("P_Buttons");
  F_MainFrame.add(P_Buttons,BorderLayout.SOUTH);

  //7a-Ajout d'un boutton Valider
  JButton B_Valider = new JButton("Editer");
  B_Valider.setActionCommand("GENERATE");
  B_Valider.addActionListener(this);
  P_Buttons.add(B_Valider);

  //7b-Ajout d'un bouton Quitter
```

```
            JButton B_Quitter = new JButton("Quitter");
            B_Quitter.setActionCommand("QUIT");
            B_Quitter.addActionListener(this);
            P_Buttons.add(B_Quitter);

            //8- Création d'un Iterator sur les nœuds jrxml_file pour remplir la liste déroulante
            String s_identifier_auto = "";
            String s_filename_auto = "";
            if (DOC_XMLAuto != null) {
              s_identifier_auto = DOC_XMLAuto.getRootElement().getChildText("identifier");
              Trace.SP_ECRIRE("Identifier AUTO : " + s_identifier_auto,Trace.GN_NIVEAU_LOG_DEBUG);
              s_filename_auto = DOC_XMLAuto.getRootElement().getChildText("output_file");
              Trace.SP_ECRIRE("Output File Name AUTO : " + s_filename_auto,Trace.GN_NIVEAU_LOG_DEBUG);
              IsAuto=true;
            }

            Iterator i = DOC_XMLConfig.getRootElement().getChild("jrxml_files").getChildren("jrxml_file").iterator();
              while(i.hasNext()) {
              Element elt = (Element) i.next();
              String s_Title = elt.getChild("title").getText();
              String s_identifier = elt.getChild("identifier").getText();
              Trace.SP_ECRIRE("Ajout du Titre : " + s_identifier+ "|"+ s_Title,Trace.GN_NIVEAU_LOG_DEBUG);
              if (s_identifier.equals(s_identifier_auto) || CB_Title.getItemCount() ==0 ) {IsAuto=false;}
              CB_Title.addItem(s_Title);
              if (s_identifier.equals(s_identifier_auto)) {
                IsAuto=false;
                CB_Title.setSelectedItem(s_Title);

              }

            }
            IsAuto=false;

            if(!s_filename_auto.equals("")){
              Generate(s_filename_auto);
              Closing();
              Context.SP_LEAVE();
            }
            F_MainFrame.setVisible(true);

F_MainFrame.setSize(600,F_MainFrame_Orig_Height+P_Title.getHeight()+P_Buttons.getHeight()+P_Parameters.getComponentCount()/2*(CB_Title.getHeight()+1));


            DOC_XMLAuto=null;


          } catch (Exception e) {
            Trace.SP_ECRIRE("Erreur lors de la connexion à la base d'audit : " + Connection_String + " / " + Login + " / " + PassWord + "\n" +
e.toString(),"ERROR","ERR_CONN_AUDIT_DB");
            }
      } catch (ClassNotFoundException e) {
```

```
        Trace.SP_ECRIRE("Erreur lors du chargement de la class : " + Driver_ClassName + "\n" + e.toString(),"ERROR","ERR_INIT_CLASS");
        }

      Context.SP_LEAVE();
  }


    //--------------------------------------------------------------------------+
    //
    // Évènement: itemStateChanged
    // Evènement changement de sélection : se déclenche lors du changement de "Rapport à éditer".
    //
    // Type: Publique.
    //
    // Parameters:
    //    ItemEvent                  IN - Descripteur de l'élément qui a été sélectionné ou désélectionné
    //
    // Exception: Aucune.
    //
    // Algorithme:
    //>
    //
    // Historique:
    //   01/09/2010      SA  -  Création.
    //
    //--------------------------------------------------------------------------+
  public void itemStateChanged(ItemEvent ie) {
    if (!IsAuto){
      IsAuto=true;

      Context.SP_ENTER("report_launcher_ihm.itemStateChanged");

      //1-Filtrage sur les actions de sélection uniquement
      if (ie.getStateChange() == ItemEvent.SELECTED && (ie.getSource() instanceof JComboBox || ie.getSource() instanceof JComboBoxSQL || ie.getSource()
instanceof JComboBoxDependent)) {
        long startTime = System.nanoTime();

        Trace.SP_ECRIRE("************************************************************");
        if (((JComboBox)ie.getSource()).getName().indexOf('\n') >0 ) {
          Trace.SP_ECRIRE("** "+ ((JComboBox)ie.getSource()).getName().substring(0,((JComboBox)ie.getSource()).getName().indexOf('\n')) + "=> Selection de :" +
ie.getItem().toString());
        }else{
          Trace.SP_ECRIRE("** "+ ((JComboBox)ie.getSource()).getName() + "=> Selection de :" + ie.getItem().toString());
        }
        Trace.SP_ECRIRE("************************************************************");


        F_MainFrame.setCursor(new Cursor(Cursor.WAIT_CURSOR));

        if (ie.getSource()==CB_Title) {
```

```
//2-Suppression de tous les éléments précédemment ajouter au panneau de configuration
P_Parameters.removeAll();

//3-Recherche du noeud jrxml_file qui a pour titre l'élément sélectionné
Iterator i = DOC_XMLConfig.getRootElement().getChild("jrxml_files").getChildren("jrxml_file").iterator();
while(i.hasNext())
{
  //3a-Définition du noeud jrxml_file courant
  Element jrxml_file_node = (Element)i.next();

  //3b-Si le noeud courant à pour titre l'élément sélectionné alors on va remplir le panneau de de paramètres
  if (jrxml_file_node.getChild("title").getText() == ie.getItem().toString()) {

    //4-Parcours de chacun des paramètres définis
    int paratemeters_count =0;
    Trace.SP_ECRIRE("Liste des paramètres : ");
    Iterator j = jrxml_file_node.getChild("parameters").getChildren("parameter").iterator();
    while (j.hasNext())
    {
      //4a-Définition du noeud « parameter » courant
      Element parameter_node = (Element)j.next();
      Trace.SP_ECRIRE("\tname\t\t: " + parameter_node.getAttributeValue("name").toString());

      //4b-Vérification si il faut afficher le paramètre
      boolean parameter_visible = true;
      try {
        Trace.SP_ECRIRE("\tvisible\t\t: " + parameter_node.getAttributeValue("visible").toString().toUpperCase());
        if (parameter_node.getAttributeValue("visible").toString().toUpperCase().equals("NO")) {
          parameter_visible = false;
        }
      }catch (java.lang.NullPointerException e) {}

      if (parameter_visible==true){
        paratemeters_count++;

        //5-Ajout d'une étiquette pour le nouveau paramètre
        JLabel La_Parameter = new JLabel();
        La_Parameter.setName("LABEL_" + parameter_node.getAttributeValue("name").toString());
        try{
          La_Parameter.setText(parameter_node.getAttributeValue("label").toString() + " :");
        }
        catch (java.lang.NullPointerException e) {
          La_Parameter.setText(parameter_node.getAttributeValue("name").toString() + " :");
        }
        P_Parameters.add(La_Parameter,BorderLayout.EAST);

        //6 - Recherche du « parameter » dans le fichier de génération auto
        String s_value_auto="";
        try{
          Iterator i_auto = DOC_XMLAuto.getRootElement().getChild("parameters").getChildren("parameter").iterator();
```

```
                    while(i_auto.hasNext())
                    {
                      Element parameter_node_auto = (Element)i_auto.next();
                      if
(parameter_node.getAttributeValue("name").toUpperCase().toString().equals(parameter_node_auto.getAttributeValue("name").toUpperCase().toString())){
                        s_value_auto=parameter_node_auto.getValue();
                        Trace.SP_ECRIRE("\tvalue auto\t: '" + s_value_auto + "'");
                      }
                    }
                  }
                  catch(java.lang.NullPointerException e) {}

                  //6a-Ajout d'une zone de texte si le noeud possède un fils value
                  try {
                    String s_value = parameter_node.getChild("value").getText();
                    Trace.SP_ECRIRE("\ts_value\t\t: " + s_value);
                    //Si le paramètre est défini dans le fichier de génération auto alors on utilise sa valeur
                    if (!s_value_auto.equals("")){s_value=s_value_auto;}
                    //6a1-Création s'un zone de texte à partir de la valeur
                    JTextField TB_Parameter = new JTextField(s_value);
                    //6a2-Définition du nom pour retrouver les valeurs de se paramètre
                    TB_Parameter.setName(parameter_node.getAttributeValue("name").toString());

                    //6a3-ajout de la zone de texte au panneau des paramètres
                    P_Parameters.add(TB_Parameter,BorderLayout.CENTER);
                  }
                  catch (java.lang.NullPointerException e) {}
                  catch (Exception e                    ) {Trace.SP_ECRIRE("\ts_value erreur : " + e.toString());};

                  //6b-Ajout d'une liste déroulante si le noeud possède un fils « SQL_values »
                  try {
                    //6b1-Lecture du noeud « SQL_values »
                    String s_sql_values = parameter_node.getChild("SQL_values").getText().trim();
                    Trace.SP_ECRIRE("\ts_sql_values : " + s_sql_values,Trace.GN_NIVEAU_LOG_DEBUG);
                    while (s_sql_values.indexOf("\n ")>=0 || s_sql_values.indexOf("\n\t")>=0){
                      s_sql_values = s_sql_values.replaceAll("\n ", "\n");
                      s_sql_values = s_sql_values.replaceAll("\n\t", "\n");
                    }

                    //6b2-Création d'une liste déroulante
                    JComboBoxSQL CB_ListeSQL  = new JComboBoxSQL();

                    //6b3-Définition du nom pour retrouver les valeurs de se paramètre
                    CB_ListeSQL.setName(parameter_node.getAttributeValue("name").toString());

                    //6b4-Ajout de la liste déroulante au panneau des paramètres
                      P_Parameters.add(CB_ListeSQL,BorderLayout.CENTER);

                      //6b4-Ajout de la requête SQL
                      CB_ListeSQL.setConnection(Audit_DB_Conn);
                    CB_ListeSQL.setSQL(s_sql_values);
```

```
      //6b5-Définition du listener
      CB_ListeSQL.addItemListener(this);

      //Si le paramètre est défini dans le fichier de génération auto alors on utilise sa valeur
      if (!s_value_auto.equals("")){
        CB_ListeSQL.setSelectedId(s_value_auto);
      }

    }
    catch (java.lang.NullPointerException e) {Trace.SP_ECRIRE("\ts_sql_values erreur : " + e.toString());}
    catch (Exception e                     ) {Trace.SP_ECRIRE("\ts_sql_values erreur : " + e.toString());};

    //6c-Ajout d'une liste déroulante dépendantes si le noeud possède un fils « dependent_values »
    try {
      //6c1-Lecture du noeud « dependent_values »
      String s_dependent_values = parameter_node.getChild("dependent_values").getText().trim();
      Trace.SP_ECRIRE("\ts_dependent_values : " + s_dependent_values,Trace.GN_NIVEAU_LOG_DEBUG);
      while (s_dependent_values.indexOf("\n ")>=0 || s_dependent_values.indexOf("\n\t")>=0){
        s_dependent_values = s_dependent_values.replaceAll("\n ", "\n");
        s_dependent_values = s_dependent_values.replaceAll("\n\t", "\n");
      }

      //6c2-Création d'une liste déroulante
      JComboBoxDependent CB_ListeDependent  = new JComboBoxDependent();

      //6c3-Définition du nom pour retrouver les valeurs de ses paramètres
      CB_ListeDependent.setName(parameter_node.getAttributeValue("name").toString());

      //6c4-Ajout de la liste déroulante au panneau des paramètres
        P_Parameters.add(CB_ListeDependent,BorderLayout.CENTER);

        //6c4-Ajout de la requête SQL
        CB_ListeDependent.setFormula(s_dependent_values);

      //6c5-Définition du listener
        CB_ListeDependent.addItemListener(this);

        //Si le paramètre est défini dans le fichier de génération auto alors on utilise sa valeur
      if (!s_value_auto.equals("")){
        CB_ListeDependent.setSelectedId(s_value_auto);
      }

    }
    catch (java.lang.NullPointerException e) {Trace.SP_ECRIRE("\ts_dependent_values erreur : " + e.toString());}
    catch (Exception e                     ) {Trace.SP_ECRIRE("\ts_dependent_values erreur : " + e.toString());};

  }

}
//7-Redessine le panneau des paramètres
```

```
            F_MainFrame.setSize(600,F_MainFrame_Orig_Height+P_Title.getHeight()+P_Buttons.getHeight()+paratemeters_count*(CB_Title.getHeight()+1));
            Trace.SP_ECRIRE("Redimensionnement de la fenêtre : HEIGHT=" + F_MainFrame.getHeight() + "  WIDTH=" + F_MainFrame.getWidth());
            P_Parameters.doLayout();
          }

          }



        }else{
          //Parcours des paramètres pour trouver la position de l'élément qui vient d'être modifié
          for (int j=0;j<P_Parameters.getComponentCount();j++){
            if (P_Parameters.getComponents()[j]==ie.getSource()){
              //Parcours des paramètres suivants afin de faire les substitutions dans la requête
              for (int k=j+1;k<P_Parameters.getComponentCount();k++){
                if (P_Parameters.getComponents()[k] instanceof JComboBoxSQL){
                  Trace.SP_ECRIRE("A recalculer = " + P_Parameters.getComponents()[k].getName());
                  ((JComboBoxSQL)P_Parameters.getComponents()[k]).UpdateSQLItems();
                }else if(P_Parameters.getComponents()[k] instanceof JComboBoxDependent){
                  Trace.SP_ECRIRE("A recalculer = " + P_Parameters.getComponents()[k].getName());
                  ((JComboBoxDependent)P_Parameters.getComponents()[k]).UpdateItems();
                }
              }
            }
          }
        }


        F_MainFrame.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));

        Trace.SP_ECRIRE("************************************************************");
        if (((JComboBox)ie.getSource()).getName().indexOf('\n') >0 ) {
          Trace.SP_ECRIRE("** Fin "+ ((JComboBox)ie.getSource()).getName().substring(0,((JComboBox)ie.getSource()).getName().indexOf('\n')) + " => Selection de
:" + ie.getItem().toString());
        }else{
          Trace.SP_ECRIRE("** Fin "+ ((JComboBox)ie.getSource()).getName() + " => Selection de :" + ie.getItem().toString());
        }
        Trace.SP_ECRIRE("************************************************************");

        if (Trace.GN_NIVEAU_LOG == Trace.GN_NIVEAU_LOG_DEBUG && DOC_XMLAuto == null){
          double elapsedTime = (System.nanoTime() - startTime)/1000000L;
          String s_duree;
          if (elapsedTime<1000){s_duree = elapsedTime + "ms";}
          else {
            elapsedTime = elapsedTime/1000;
            s_duree = (new java.text.DecimalFormat("0.000")).format(elapsedTime % 60)  + "s";
          }
          javax.swing.JOptionPane.showMessageDialog(null,Context.SP_CURRENT_NAME() + " durée :" +
s_duree,Context.SP_CURRENT_NAME(),JOptionPane.INFORMATION_MESSAGE);
        }
```

```
    }

    Context.SP_LEAVE();
    IsAuto=false;
  }
  }


  //-------------------------------------------------------------------------+
  //
  // Évènement: actionPerformed
  // Evènement bouton cliquer : se déclenche lors du clic d'un de bouton de la fenêtre ("Editer" ou "Quitter").
  //
  // Type: Publique.
  //
  // Parameters:
  //    e                          IN - Descripteur de l'élément qui a été cliqué
  //
  // Exception: Aucune.
  //
  // Algorithme:
  //>
  //
  // Historique:
  //   01/09/2010      SA  -  Création.
  //
  //-------------------------------------------------------------------------+
public void actionPerformed(ActionEvent e) {
  Context.SP_ENTER("report_launcher_ihm.actionPerformed");

    Trace.SP_ECRIRE("Action : " + e.getActionCommand().toString() );

    if (e.getActionCommand().toString()=="QUIT") {Closing();};
    if (e.getActionCommand().toString()=="GENERATE") {Generate();};

    Context.SP_LEAVE();
}

  //-------------------------------------------------------------------------+
  //
  // Évènement: windowActivated
  // Evènement fenêtre activé : se déclenche lors de l'activation de la fenêtre.
  //
  // Type: Publique.
  //
  // Parameters:
  //    e                          IN - Descripteur de la fenêtre qui a été activé
  //
  // Exception: Aucune.
  //
  // Algorithme:
```

```
    //>
    //
    // Historique:
    //   01/09/2010      SA  -  Creation.
    //
    //-------------------------------------------------------------------------+
  @Override
  public void windowActivated(WindowEvent e) {}

    //-------------------------------------------------------------------------+
    //
    // Évènement: windowClosed
    // Evènement fenêtre fermé : se déclenche une fois la fenêtre fermé.
    //
    // Type: Publique.
    //
    // Parameters:
    //    e                          IN - Descripteur de la fenêtre qui a été férmé
    //
    // Exception: Aucune.
    //
    // Algorithme:
    //>
    //
    // Historique:
    //   01/09/2010      SA  -  Creation.
    //
    //-------------------------------------------------------------------------+
  @Override
  public void windowClosed(WindowEvent e) {}

    //-------------------------------------------------------------------------+
    //
    // Évènement: windowClosing
    // Évenement demande de fermeture de la fenêtre : se déclenche a la demande de fermeture de la fenêtre.
    //
    // Type: Publique.
    //
    // Parameters:
    //    e                          IN - Descripteur de la fenêtre qui doit être férmé
    //
    // Exception: Aucune.
    //
    // Algorithme:
    //>
    //
    // Historique:
    //   01/09/2010      SA  -  Creation.
    //
    //-------------------------------------------------------------------------+
  @Override
```

```
  public void windowClosing(WindowEvent e) {
    Context.SP_ENTER("report_launcher_ihm.windowClosing");

    Closing();

    Context.SP_LEAVE();
  }

    //-------------------------------------------------------------------------------+
    //
    // Évènement: windowDeactivated
    // Évenement désactivation de la fenêtre : se déclenche à la désactivation de de la fenêtre.
    //
    // Type: Publique.
    //
    // Parameters:
    //    e                          IN - Descripteur de la fenêtre qui est désactivé
    //
    // Exception: Aucune.
    //
    // Algorithme:
    //>
    //
    // Historique:
    //   01/09/2010      SA  -  Creation.
    //
    //-------------------------------------------------------------------------------+
  @Override
  public void windowDeactivated(WindowEvent e) {}

    //-------------------------------------------------------------------------------+
    //
    // Évènement: windowDeiconified
    // Évenement dé-icônification de la fenêtre : se déclenche lorsque la fenêtre passe de la forme d'icône à la forme normale.
    //
    // Type: Publique.
    //
    // Parameters:
    //    e                          IN - Descripteur de la fenêtre qui est désactivé
    //
    // Exception: Aucune.
    //
    // Algorithme:
    //>
    //
    // Historique:
    //   01/09/2010      SA  -  Creation.
    //
    //-------------------------------------------------------------------------------+
  @Override
  public void windowDeiconified(WindowEvent e) {}
```

```
  //-------------------------------------------------------------------------+
  //
  // Évènement: windowIconified
  // Évenement icônification de la fenêtre : se déclenche lorsque la fenêtre passe de la forme normale à la forme d'icône.
  //
  // Type: Publique.
  //
  // Parameters:
  //    e                          IN - Descripteur de la fenêtre qui est désactivé
  //
  // Exception: Aucune.
  //
  // Algorithme:
  //>
  //
  // Historique:
  //   01/09/2010      SA  -  Creation.
  //
  //-------------------------------------------------------------------------+
@Override
public void windowIconified(WindowEvent e) {}

  //-------------------------------------------------------------------------+
  //
  // Évènement: windowOpened
  // Évenement ouverture de la fenêtre : se déclenche lors de l'ouverture de la fenêtre.
  //
  // Type: Publique.
  //
  // Parameters:
  //    e                          IN - Descripteur de la fenêtre qui est désactivé
  //
  // Exception: Aucune.
  //
  // Algorithme:
  //>
  //
  // Historique:
  //   01/09/2010      SA  -  Creation.
  //
  //-------------------------------------------------------------------------+
public void windowOpened(WindowEvent e) {}

  //-------------------------------------------------------------------------+
  //
  // procédure : Generate
  // Génération d'un nouveau rapport (se déclenche lors du clic du bouton "Editer").
  //
  // Type: privé.
  //
```

```
  // Parameters:
  //    aucun
  //
  // Exception: Aucune.
  //
  // Algorithme:
  //>
  //
  // Historique:
  //   01/09/2010      SA  -  Création.
  //
  //-------------------------------------------------------------------------------+
private void Generate()  {
  Generate("");
}

private void Generate(String FileName)  {

  Context.SP_ENTER("report_launcher_ihm.Generate");

  Trace.SP_ECRIRE("************************************************************");
  Trace.SP_ECRIRE("** Rapport à editer : " + CB_Title.getSelectedItem());
  Trace.SP_ECRIRE("************************************************************");

  long startTime = System.nanoTime();



  //01 - Création d'un Iterator sur les noeuds jrxml_file pour trouver le rapport à éditer
  Iterator i = DOC_XMLConfig.getRootElement().getChild("jrxml_files").getChildren("jrxml_file").iterator();
  while(i.hasNext())
  {
    //     Définition du noeud jrxml_file courant
    Element courant = (Element)i.next();

    //     Comparaison du noeud « title » à l'élément sélectionné dans la liste box "Rapport à éditer"
    if (courant.getChild("title").getText() == CB_Title.getSelectedItem()) {

      try {
              //02 - Chargement du rapport
        //     Lecture du nom du fichier JRXML
        String JRXML_filename = (new File (".")).getCanonicalPath().toString() + File.separator +"JRXML"+ File.separator + courant.getChild("file").getText();
        Trace.SP_ECRIRE("Fichier à utiliser : " + JRXML_filename);

        //     Ouverture du fichier JRXML
        org.jdom.Document JRXML_document = (new SAXBuilder()).build(new File(JRXML_filename));

        //     Chargement du fichier JRXML dans un designer Jasper
        JasperDesign My_JasperDesign = JRXmlLoader.load(JRXML_filename);

        //03 - Définition du DATASET principal
```

```
        //     Récupération du DATASET principal
              JRDesignDataset My_MainDataset = My_JasperDesign.getMainDesignDataset();

              //     Création d'un nouveau DATASET
              JRDesignQuery My_Query = new JRDesignQuery();
              My_Query.setLanguage("SQL");
              Trace.SP_ECRIRE("Main DATASET: " + courant.getChild("requete").getText());
              My_Query.setText(courant.getChild("requete").getText());
              My_MainDataset.setQuery(My_Query);

              //     Modification du DATASET principal
              My_JasperDesign.setMainDataset(My_MainDataset);

              //04 - Définition des DATASET secondaires
              try{

                //     Création d'un Iterator sur les noeuds subDataset du fichier XML de configuration
                Iterator k = courant.getChildren("subDataset").iterator();
                while(k.hasNext())
                {
                          //     Définition du noeud subDataset courant
                  Element subDataset_courant = (Element)k.next();

                  Trace.SP_ECRIRE("sub-DATASET " + subDataset_courant.getAttributeValue("name") + " : " + subDataset_courant.getChild("requete").getText());

              //     Lecture de la requête à utiliser
                  My_Query = new JRDesignQuery();
                    My_Query.setLanguage("SQL");
                    My_Query.setText(subDataset_courant.getChild("requete").getText());

                    //     Parcours des DATASET secondaires du designer Jasper
                    boolean find = false;
                    for (int SubDataset_Indice = 0 ; SubDataset_Indice<My_JasperDesign.getDatasets().length;SubDataset_Indice++){
                      if (My_JasperDesign.getDatasets()[SubDataset_Indice].getName().equals(subDataset_courant.getAttributeValue("name"))) {
                        //     Modification du DATASET secondaire
                        find = true;
                        ((JRDesignDataset) My_JasperDesign.getDatasets()[SubDataset_Indice]).setQuery(My_Query);
                      }
                    }
                    //     Si le DATASET secondaire n'a pas été trouvé alors on affiche une erreur
                    if (find==false) {
                      Trace.SP_ECRIRE("Impossible de définir le DATASET secondaire : " +
subDataset_courant.getAttributeValue("name"),"ERROR","ERR_SUB_DATASET_NOT_FOUND");
                    }

            }
              } catch (Exception e) {
                Trace.SP_ECRIRE("Erreur lors de la définition des DATASET secondaires" + "\n" + e.toString(),"ERROR","ERR_DEF_SUB_DATASETS");
              }
```

```
  //05 - Définition des paramètres
  Map parameters = new HashMap();

  try{

    //     Création d'un Iterator sur les noeuds subDataset du fichier XML de configuration
    Iterator k = courant.getChild("parameters").getChildren("parameter").iterator();
    while(k.hasNext())
    {
      String Parameter_name=null,Parameter_value=null,Parameter_class=null;

      //     Définition du noeud parameter courant
Element parameter_node = (Element)k.next();
Parameter_name = parameter_node.getAttributeValue("name").toString();
Trace.SP_ECRIRE("\tname\t\t: " + Parameter_name);

//     Vérification si le paramètre a été affiché
boolean parameter_visible = true;
try {
  Trace.SP_ECRIRE("\tvisible\t\t: " + parameter_node.getAttributeValue("visible").toString().toUpperCase());
  if (parameter_node.getAttributeValue("visible").toString().toUpperCase().equals("NO")) {
    parameter_visible = false;
  }
}catch (java.lang.NullPointerException e) {}

try {
  if (parameter_visible==true){
    //     Recherche du paramètre dans le panneau des paramètres
    for (int j=0;j<P_Parameters.getComponentCount();j++){
      if (P_Parameters.getComponents()[j].getName() == Parameter_name) {
        //     Récupération de la valeur choisis pas l'utilisateur
          if (P_Parameters.getComponents()[j] instanceof JTextField) {
                  Parameter_value = ((JTextField) P_Parameters.getComponents()[j]).getText();

                }
                if (P_Parameters.getComponents()[j] instanceof JComboBox) {
                  Object L_Value = ((Item)((JComboBox) P_Parameters.getComponents()[j]).getSelectedItem()).getId();
                  if (L_Value instanceof java.sql.Date){
                    Parameter_value = (new SimpleDateFormat("dd/MM/yyyy HH:mm:ss", Locale.FRENCH)).format((java.sql.Date)L_Value);
                  }else {
                    Parameter_value = L_Value.toString();
                  }
                }
                Trace.SP_ECRIRE("\tvalue_class\t: " + Parameter_value.getClass().getName());
                Trace.SP_ECRIRE("\tvalue\t\t: " + Parameter_value);
      }
    }
  }else{
      //     Utilisation de la valeur présente dans le fichier de configuration
    Parameter_value =  parameter_node.getChild("value").getText();
```

```
    };
    Trace.SP_ECRIRE("\tvalue\t\t: " + Parameter_value);
  }catch (java.lang.NullPointerException e) {
    Trace.SP_ECRIRE("\tvalue_class\t: java.lang.String");
    Trace.SP_ECRIRE("\tvalue\t\t: NULL");
  }

    //    Recherche de la class du paramètre à transmettre
    //    Création d'un Iterator sur les noeuds subDataset du fichier XML de configuration
        Iterator l = JRXML_document.getRootElement().getChildren().iterator();
        while(l.hasNext())
        {
          Element JRXML_courant = (Element)l.next();
          if (JRXML_courant.getName().equals("parameter")){
        if (JRXML_courant.getAttributeValue("name").equals(Parameter_name)) {
          Parameter_class=  JRXML_courant.getAttributeValue("class");
        }
      }
        }
        Trace.SP_ECRIRE("\tclass\t\t: " + Parameter_class);

        //    Conversion de la valeur en fonction de la class
        try {
          if  (Parameter_class.equals("java.math.BigDecimal")){
            parameters.put(Parameter_name, new BigDecimal(Parameter_value));
          }else if  (Parameter_class.equals("java.util.Date")){
            try{
              parameters.put(Parameter_name, (new SimpleDateFormat("dd/MM/yyyy HH:mm:ss")).parse(Parameter_value));
            } catch (Exception e) {
              Trace.SP_ECRIRE("Erreur lors de la conversion en date de " + Parameter_value + "/n" + e.toString(),"ERROR","ERR_CONV_DATE");
              }
          }
          else {
            parameters.put(Parameter_name, Parameter_value);
          }
        }catch (java.lang.NullPointerException e) {
          parameters.put(Parameter_name, null);
        }
      }
}catch (Exception e) {
      Trace.SP_ECRIRE("Erreur lors de la definition des paramètres" + "\n" + e.toString(),"ERROR","ERR_DEF_PARAMETERS");
    }


     //06 - Compilation du rapport
Trace.SP_ECRIRE("compileReport");
    JasperReport jasperReport = JasperCompileManager.compileReport(My_JasperDesign);


    //07 - Définitions des paramètres et de la base de données
    Trace.SP_ECRIRE("fillReport");
```

```
            JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, parameters, Audit_DB_Conn);

            //08 - Création du rapport au format PDF
            if (FileName.equals("")){
              String tmpdir = System.getProperty("java.io.tmpdir");
              Trace.SP_ECRIRE("Répertoire temporaire : " + tmpdir);
              FileName = "RL_"+UUID.randomUUID().toString()+".pdf";
              Trace.SP_ECRIRE("Fichier temporaire : " + FileName);
              FileName = tmpdir+FileName;

              Trace.SP_ECRIRE("exportReportToPdfFile");
              JasperExportManager.exportReportToPdfFile(jasperPrint, FileName);

              //09 - Ouverture du document
              try {
                Trace.SP_ECRIRE("Ouverture du fichier");
                Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + FileName);
              } catch (Exception e){
                Trace.SP_ECRIRE("Erreur lors de l'ouverture du fichier" + "\n" + e.toString(),"ERROR","ERR_OPEN_REPORT");
              }

            }else{
              Trace.SP_ECRIRE("Fichier AUTO : " + FileName);
              Trace.SP_ECRIRE("exportReportToPdfFile");
              JasperExportManager.exportReportToPdfFile(jasperPrint, FileName);

            }



        } catch (JRRuntimeException e) {
          Trace.SP_ECRIRE("Erreur Jaspre lors de la génération du rapport" + "\n" + e.toString(),"ERROR","ERR_JASPER");
        } catch (JRException e) {
          Trace.SP_ECRIRE("Erreur Jaspre lors de la génération du rapport" + "\n" + e.toString(),"ERROR","ERR_JASPER");
        } catch (IOException e) {
          Trace.SP_ECRIRE("Erreur I/O lors de la génération du rapport" + "\n" + e.toString(),"ERROR","ERR_IO");
        } catch (JDOMException e) {
          Trace.SP_ECRIRE("Erreur JDOM lors de la génération du rapport" + "\n" + e.toString(),"ERROR","ERR_JDOM");
        }
        }
    }

    Trace.SP_ECRIRE("************************************************************");
    Trace.SP_ECRIRE("** Fin de l'édition du Rapport  : " + CB_Title.getSelectedItem());
    Trace.SP_ECRIRE("************************************************************");

    if (Trace.GN_NIVEAU_LOG == Trace.GN_NIVEAU_LOG_DEBUG && DOC_XMLAuto == null){
      double elapsedTime = (System.nanoTime() - startTime)/1000000L;
      String s_duree;
      if (elapsedTime<1000){s_duree = elapsedTime + "ms";}
      else {
```

```
        elapsedTime = elapsedTime/1000;
        s_duree = (new java.text.DecimalFormat("0.000")).format(elapsedTime % 60)  + "s";
       }
       javax.swing.JOptionPane.showMessageDialog(null,Context.SP_CURRENT_NAME() + " durée :" +
s_duree,Context.SP_CURRENT_NAME(),JOptionPane.INFORMATION_MESSAGE);
    }

    Context.SP_LEAVE();
  };


  private void Closing() {
    try {
      Audit_DB_Conn.close();
      } catch (Exception ex) {
            ex.printStackTrace();
        }

    System.exit(0);
  }

}
```

Annexe 9

DaPCol-Reports : Classe JComboBoxSQL

JComboBoxSQL.java

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import javax.swing.ComboBoxModel;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class JComboBoxSQL extends javax.swing.JComboBox {
  private static final long serialVersionUID = 1L;

  public JComboBoxSQL(){super();}

  public JComboBoxSQL(ComboBoxModel aModel){super(aModel);}

  public JComboBoxSQL(Object[] items){super(items);}

  String C_SQL;
  Connection C_DB_Con;

  public void setSQL (String SQL) {
    C_SQL = SQL;
    this.UpdateSQLItems();
  }

  public String getSQL (){
    return C_SQL;
  }

  public void setConnection (Connection DB_Con) {
    C_DB_Con = DB_Con;
  }

  public Connection getConnection (){
    return C_DB_Con;
  }

  public void setSelectedId(String ID){
```

```
  for(int i=0;i<this.getItemCount();i++){
    if (ID.equals(((Item)this.getItemAt(i)).getId().toString())){
      this.setSelectedIndex(i);
    }
  }
}

public void setSelectedDescription(String Description){
  for(int i=0;i<this.getItemCount();i++){
    if (Description.equals(((Item)this.getItemAt(i)).getDescription().toString())){
      this.setSelectedIndex(i);
    }
  }
}


//-------------------------------------------------------------------------+
//
// Procedure: UpdateSQLItems
// Procédure permettant de recalculer la liste des éléments en fonction de la requête SQL.
//
// Type: Publique.
//
// Parameters: Aucun.
//
// Exception: Aucune.
//
// Algorithme:
//>
//
// Historique:
//   01/09/2010      SA  -  Création.
//
//-------------------------------------------------------------------------+
public void UpdateSQLItems(){
  Context.SP_ENTER("JComboBoxSQL.UpdateSQLItems");
  String s_sql_values = this.getSQL();
  Trace.SP_ECRIRE("s_sql_values="+s_sql_values);
  //Parcours des autres paramètres afin de faire les substitutions dans la requête
  for (int k=0;k<this.getParent().getComponentCount();k++){
    try{
      if ( !(this.getParent().getComponents()[k] instanceof JLabel)) {
        String s_parameter_name = this.getParent().getComponents()[k].getName();
        Trace.SP_ECRIRE("\t\tParameter\t\t: " + s_parameter_name,Trace.GN_NIVEAU_LOG_DEBUG);

        //     Récupération de la valeur choisis pas l'utilisateur
        Object s_parameter_value=null;
        Trace.SP_ECRIRE("\t\tComponents_class\t: " + this.getParent().getComponents()[k].getClass().getSimpleName().toString(),Trace.GN_NIVEAU_LOG_DEBUG);
        if (this.getParent().getComponents()[k] instanceof JTextField) {
          s_parameter_value = ((JTextField) this.getParent().getComponents()[k]).getText();
              }else if (this.getParent().getComponents()[k] instanceof JComboBox) {
```

```
                s_parameter_value = ((Item)((JComboBox) this.getParent().getComponents()[k]).getSelectedItem()).getId();
              }
              try {
                Trace.SP_ECRIRE("\t\tValue\t\t\t: " + s_parameter_value.toString(),Trace.GN_NIVEAU_LOG_DETAIL);
                Trace.SP_ECRIRE("\t\tValue_class\t\t: " + s_parameter_value.getClass().getName().toString(),Trace.GN_NIVEAU_LOG_DETAIL);
                if(s_parameter_value instanceof String){
                  s_sql_values=s_sql_values.replace(":{"+s_parameter_name+"}", "'"+s_parameter_value.toString()+"'");
                }else if(s_parameter_value instanceof Number){
                  s_sql_values=s_sql_values.replace(":{"+s_parameter_name+"}", "'"+s_parameter_value.toString()+"'");
                }else if(s_parameter_value instanceof Date){
                  s_sql_values=s_sql_values.replace(":{"+s_parameter_name+"}", "TO_DATE('"+(new SimpleDateFormat("dd/MM/yyyy HH:mm:ss",
Locale.FRENCH)).format(s_parameter_value)+"','DD/MM/YYYY HH24:MI:SS')");
                }
                else{
                  s_sql_values=s_sql_values.replace(":{"+s_parameter_name+"}", s_parameter_value.toString());
                }

              }catch (java.lang.NullPointerException e) {
                Trace.SP_ECRIRE("\t\tvalue\t\t\t: NULL",Trace.GN_NIVEAU_LOG_DETAIL);
                Trace.SP_ECRIRE("\t\tValue_class\t\t: NULL",Trace.GN_NIVEAU_LOG_DETAIL);
                s_sql_values=s_sql_values.replace(":{"+s_parameter_name+"}", "NULL");
              }
        }
      }catch (java.lang.NullPointerException e) {Trace.SP_ECRIRE("\tValue Error\t: "+e.toString());}
    }

    Trace.SP_ECRIRE("\ts_sql_values\t: ",Trace.GN_NIVEAU_LOG_DETAIL);
    Trace.SP_ECRIRE("\t\t\t  "+s_sql_values.trim().replaceAll("\n", "\n\t\t\t  "),Trace.GN_NIVEAU_LOG_DETAIL);

    try {
      //6b2-Exécutions de la requête sur la base d'audit
      ResultSet rset =  C_DB_Con.createStatement().executeQuery(s_sql_values);

      //6b3-Initialisation d'un compteur
      int cpt = 0 ;
      this.removeAllItems();
      while (rset.next()) {

            //6b4-Ajout des champs VALUE et TEXT au vecteur
            this.addItem( new Item(rset.getObject("VALUE"), rset.getObject("TEXT") ) );
            cpt++;
        }
      //6b5-Fermeture de la requête
      rset.getStatement().close();
    }catch(SQLException e){Trace.SP_ECRIRE("\tSQL Error\t: "+e.getMessage());}

    Context.SP_LEAVE();
  }
}
```

Annexe 10

DaPCol-Reports : Classe JComboBoxDependent

```
import java.util.Date;
import javax.swing.ComboBoxModel;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class JComboBoxDependent extends javax.swing.JComboBox {
  private static final long serialVersionUID = 1L;

  public JComboBoxDependent(){super();}

  public JComboBoxDependent(ComboBoxModel aModel){super(aModel);}

  public JComboBoxDependent(Object[] items){super(items);}

  String C_Formula;

  public void setFormula (String Formula) {
    C_Formula = Formula;
    this.UpdateItems();
  }

  public String getFormula(){
    return C_Formula;
  }

  public void setSelectedId(String ID){
    for(int i=0;i<this.getItemCount();i++){
      if (ID.equals(((Item)this.getItemAt(i)).getId().toString())){
        this.setSelectedIndex(i);
      }
    }
  }

  public void setSelectedDescription(String Description){
    for(int i=0;i<this.getItemCount();i++){
      if (Description.equals(((Item)this.getItemAt(i)).getDescription().toString())){
        this.setSelectedIndex(i);
      }
    }
  }
```

```
  //-------------------------------------------------------------------------------+
  //
  // Procedure: UpdateItems
  // Procédure permettant de recalculer la liste des éléments en fonction de la Formule.
  //
  // Type: Publique.
  //
  // Parameters: Aucun.
  //
  // Exception: Aucune.
  //
  // Algorithme:
  //>
  //
  // Historique:
  //   01/09/2010      SA  -  Creation.
  //
  //-------------------------------------------------------------------------------+
public void UpdateItems(){
  Context.SP_ENTER("JComboBoxDependent.UpdateItems");
  String s_Formula = this.getFormula();
  Trace.SP_ECRIRE("s_Formula="+s_Formula);
  //Parcours des autres paramètres pour trouver celui de la Formule
  for (int k=0;k<this.getParent().getComponentCount();k++){
    try{
      if ( !this.getParent().getComponents()[k].getClass().equals((new JLabel()).getClass())) {
        String s_parameter_name = this.getParent().getComponents()[k].getName();

        if (s_Formula.indexOf(":{"+s_parameter_name+"}")>0) {
          Trace.SP_ECRIRE("\t\tParameter\t\t: " + s_parameter_name);//,Trace.GN_NIVEAU_LOG_DEBUG);

          //Recupération de la valeur choisis par l'utilisateur
          Object s_parameter_value=null;
          Trace.SP_ECRIRE("\t\tComponents_class\t: " + this.getParent().getComponents()[k].getClass().getSimpleName().toString());
          if (this.getParent().getComponents()[k] instanceof JTextField) {
            s_parameter_value = ((JTextField) this.getParent().getComponents()[k]).getText();
                }else if (this.getParent().getComponents()[k] instanceof JComboBox) {
                  s_parameter_value = ((Item)((JComboBox) this.getParent().getComponents()[k]).getSelectedItem()).getId();
                }

                Trace.SP_ECRIRE("\t\tValue_class\t\t: " + s_parameter_value.getClass().getName().toString(),Trace.GN_NIVEAU_LOG_DETAIL);
                Trace.SP_ECRIRE("\t\tValue\t\t\t: " + s_parameter_value.toString(),Trace.GN_NIVEAU_LOG_DETAIL);

                //6b1-Création d'un vecteur pour stocker les résultats de la requête
                if (s_Formula.trim().indexOf("!=")==0){
                  Trace.SP_ECRIRE ("\t\t\tTest : !=",Trace.GN_NIVEAU_LOG_DEBUG );
            }else if (s_Formula.trim().indexOf("<=")==0) {
              Trace.SP_ECRIRE ("\t\t\tTest : <=",Trace.GN_NIVEAU_LOG_DEBUG );
            }else if (s_Formula.trim().indexOf(">=")==0) {
              Trace.SP_ECRIRE ("\t\t\tTest : >=",Trace.GN_NIVEAU_LOG_DEBUG );
            }else if (s_Formula.trim().indexOf("<")==0) {
```

```
  Trace.SP_ECRIRE ("\t\t\tTest : <",Trace.GN_NIVEAU_LOG_DEBUG );
}else if (s_Formula.trim().indexOf(">")==0) {
  Trace.SP_ECRIRE ("\t\t\tTest : >",Trace.GN_NIVEAU_LOG_DEBUG );
}else{
  Trace.SP_ECRIRE ("\t\t\tTest : Undefined",Trace.GN_NIVEAU_LOG_DEBUG );;
};



      //6b3-Initialisation d'un compteur

      this.removeAllItems();

      for(int cpt=0;cpt<((JComboBox) this.getParent().getComponents()[k]).getItemCount();cpt++){
        try {
          Item elt = ((Item)((JComboBox) this.getParent().getComponents()[k]).getItemAt(cpt));
          if (s_Formula.trim().indexOf("!=")==0){
            if (s_parameter_value instanceof String ){
              if (((String)elt.getId()).compareTo((String)s_parameter_value)!=0){this.addItem(elt);};
            }else if (s_parameter_value instanceof Number ){
              if(((Number)elt.getId()).doubleValue()!=((Number)s_parameter_value).doubleValue()){this.addItem(elt);};
            }else if (s_parameter_value instanceof Date ){
              if(((Date)elt.getId()).compareTo((Date)s_parameter_value)!=0){this.addItem(elt);};
            };
          }else if (s_Formula.trim().indexOf("<=")==0) {
            if (s_parameter_value instanceof String ){
              if (((String)elt.getId()).compareTo((String)s_parameter_value)<=0){this.addItem(elt);};
            }else if (s_parameter_value instanceof Number ){
              if(((Number)elt.getId()).doubleValue()<=((Number)s_parameter_value).doubleValue()){this.addItem(elt);};
            }else if (s_parameter_value instanceof Date ){
              if(((Date)elt.getId()).compareTo((Date)s_parameter_value)<=0){this.addItem(elt);};
            };
          }else if (s_Formula.trim().indexOf(">=")==0) {
            if (s_parameter_value instanceof String ){
              if (((String)elt.getId()).compareTo((String)s_parameter_value)>=0){this.addItem(elt);};
            }else if (s_parameter_value instanceof Number ){
              if(((Number)elt.getId()).doubleValue()>=((Number)s_parameter_value).doubleValue()){this.addItem(elt);};
            }else if (s_parameter_value instanceof Date ){
              if(((Date)elt.getId()).compareTo((Date)s_parameter_value)>=0){this.addItem(elt);};
            };
          }else if (s_Formula.trim().indexOf("<")==0) {
            if (s_parameter_value instanceof String ){
              if (((String)elt.getId()).compareTo((String)s_parameter_value)<0){this.addItem(elt);};
            }else if (s_parameter_value instanceof Number ){
              if(((Number)elt.getId()).doubleValue()<((Number)s_parameter_value).doubleValue()){this.addItem(elt);};
            }else if (s_parameter_value instanceof Date ){
              if(((Date)elt.getId()).compareTo((Date)s_parameter_value)<0){this.addItem(elt);};
            };
          }else if (s_Formula.trim().indexOf(">")==0) {
            if (s_parameter_value instanceof String ){
              if (((String)elt.getId()).compareTo((String)s_parameter_value)>0){this.addItem(elt);};
```

```
                        }else if (s_parameter_value instanceof Number ){
                          if(((Number)elt.getId()).doubleValue()>((Number)s_parameter_value).doubleValue()){this.addItem(elt);};
                        }else if (s_parameter_value instanceof Date ){
                          if(((Date)elt.getId()).compareTo((Date)s_parameter_value)>0){this.addItem(elt);};
                        };
                      };


                    }catch(Exception e){Trace.SP_ECRIRE("\tError compare\t: "+e.getMessage());}
                  }
                break;
          }
        }
      }catch (java.lang.NullPointerException e) {Trace.SP_ECRIRE("\tValue Error\t: "+e.toString());}
    }
    Context.SP_LEAVE();
  }
}
```

Liste des figures

Figure 3 – Architecture BAM 25
Figure 4 – WSO2 BAM in Action 26
Figure 5 – Architecture DaPCol 28
Figure 6 – Schéma conceptuel du mécanisme de collecte 30
Figure 7 – Schéma des tables 33
Figure 8 – Liste des audits 44
Figure 9 – Modification d'un audit 45
Figure 10 – Modification d'une requête d'audit 46
Figure 11 – Modification d'un complément d'audit 47
Figure 12 – Liste des planifications 48
Figure 13 – Modification d'une planification 49
Figure 14 – Modification des actions d'une planification 49
Figure 15 – Liste des DB Links 50
Figure 16 – Suivi des exécutions 51
Figure 17 – Exemple de traces 52
Figure 18 – Tailles des résultats des audits 53
Figure 19 – Architecture DaPCol - Reports 55
Figure 20 – Modèle de rapport sur 8 dates 60
Figure 21 – Modèle de rapport sur 3 dates avec les compléments pour la date la plus récente 61
Figure 22 – Exemple d'absence d'exécution de requêtes pour certaines dates 62
Figure 23 – Exemple de multiples exécutions de requêtes pour certaines dates 63

Mise en place d'un suivi dynamique de base de données : DaPCol

Mémoire d'Ingénieur C.N.A.M., Paris 2012

RESUME

La mise en production d'un projet basé sur un progiciel de gestion intégré touchant près de 400 personnes a mis en évidence la nécessité de disposer d'un nouvel d'outil de gestion. Basé sur Oracle Express et JasperReports, cet outil, entre le technique et le fonctionnel, permet de suivre le bon fonctionnement des programmes des différents processus métiers de l'entreprise. Grâce à cet outil, nous obtenons une vision globale du déroulement de l'ensemble de nos programmes. Il nous permet donc de fiabiliser nos processus, de garantir leur niveau de service et d'optimiser au mieux le travail de nos équipes.

Mots clés : Suivi d'activité, reprise de données, Oracle Express, JasperReports.

SUMMARY

The deployment of a project based on ERP software affecting nearly 400 people brought to light the necessity of new management software. Based on Oracle Express and JasperReports, this software, between technical and functional, tracks the results of programs of the different business processes of the company. With this software, we obtain an overall view of the smooth progress of all our programs. It allows us to enhance reliability of our processes, to ensure service levels and optimize the best work of our teams.

Keywords: Business Activity Management, Oracle Express, JasperReports.