ENSSIB
Ecole Nationale Supérieure des Sciences de l'Information et des Bibliothèques

Université Claude Bernard-Lyon 1

DESS en INFORMATIQUE DOCUMENTAIRE

Rapport de stage

Evolution du système de gestion d'archives GAIA :
réalisation d'une maquette de méthodologie

Nathalie LEGENDRE-FISK

Sous la direction de
Mme Isabelle Rambaud

Direction des Archives et du Patrimoine de Seine-et-Marne

1996

ENSSIB
Ecole Nationale Supérieure des Sciences de l'Information et des Bibliothèques

Université Claude Bernard-Lyon 1

DESS en INFORMATIQUE DOCUMENTAIRE

Rapport de stage

Evolution du système de gestion d'archives GAIA :
réalisation d'une maquette de méthodologie

Nathalie LEGENDRE-FISK

Sous la direction de
Mme Isabelle Rambaud

Direction des Archives et du Patrimoine de Seine-et-Marne

1996

Evolution du système de gestion d'archives GAIA : réalisation d'une maquette de méthodologie.

Nathalie LEGENDRE-FISK

Résumé

Ce document rend compte de la maquette développée pour tester la méthode de modélisation d'Oracle en prévision de la réécriture du système informatisé de gestion d'archives GAIA. Les objectifs de l'étude et les outils de conception et de développement sont présentés. Les différentes phases de la modélisation sont détaillées et un bilan est dressé. En préparation des futurs développements, une base de tests bénéficiant de contraintes d'intégrité renforcées est créée.

Descripteurs :

SGBDR ; logiciel d'archives ; modélisation ; méthode ; maquette ; contrainte d'intégrité.

Abstract

This report describes a prototype developped to test the modelling method, supplied by Oracle, for the future rewriting of GAIA, the computarised archive management system. The aims of the study, the design and development tools are presented. The different steps of the modelling process are explained in detail and the model is assessed. In preparation for future development of GAIA, a test database with enforced integrity constraints is created.

Keywords :

RDBMS ; archive software; modelling process ; methodology ; model ; integrity constraint.

Remerciements

Je tiens à remercier Mme Isabelle RAMBAUD, directrice des Archives et du Patrimoine, ainsi que tout son personnel, pour l'accueil chaleureux et sympathique qui m'a été réservé.

Je tiens également à exprimer toute ma gratitude à M. Ernest SOSSAVI, directeur et chef de projet de l'Association des Utilisateurs de GAIA, pour le soin apporté au suivi de mon travail. Je le remercie en particulier pour ses conseils techniques et méthodologiques qui ont rendu ce stage très enrichissant et qui m'ont permis de mener à bien mes missions.

J'adresse aussi tous mes remerciements à :
- Mme Annie ROZENKRANTZ, documentaliste aux Archives départementales, pour sa contribution à ma connaissance de l'archivistique,
- Mme Emmanuelle BOUTEILLER-DUPAS, chargée de gestion de l'Association des Utilisateurs de GAIA, pour sa disponibilité et sa précieuse collaboration dans l'élaboration de ce rapport,
- Mme Claude CHIESA, chef de projet aux Archives départementales, pour les réponses à mes nombreuses questions sur GAIA.

Enfin, je souhaite témoigner toute ma reconnaissance à mon mari, William, pour ses encouragements, sa disponibilité et son aide.

TABLE DES MATIÈRES

1. INTRODUCTION

Le service des Archives départementales de Seine-et-Marne gère une masse importante de documents : trente huit kilomètres linéaires en 1996. Leur gestion est assurée par le logiciel d'archives, GAIA, Gestion Automatisée et Informatisée des Archives. L'application, qui a aujourd'hui neuf ans, doit être réécrite pour pouvoir être passée sous environnement graphique. Aussi le service des Archives a-t-il décidé de profiter de cette occasion pour revoir la structure générale de l'application.

Dans le cadre de cette révision, j'ai été chargée de participer à la modélisation d'un module existant, en collaboration avec un ingénieur de la société Oracle. Le suivi de cette étude devait aboutir à la constitution d'une documentation qui rende compte précisément de la démarche adoptée. J'ai donc élaboré un document qui présente la méthode et l'atelier de génie logiciel utilisés et qui explique les différentes phases de la modélisation conceptuelle. Pour aller plus loin dans la réflexion, il m'a semblé utile de compléter ce rapport par un rapide bilan de l'étude et quelques suggestions qui, je l'espère, s'avéreront utiles pour le futur de GAIA.

Enfin, il fallait profiter de l'élan de cette étude pour poser les premières bases d'un véritable environnement de développement. J'ai donc créé une base de test, dont j'ai renforcé les contraintes d'intégrité grâce aux nouvelles fonctionnalités de la version 7 d'Oracle.

2. LE SERVICE DES ARCHIVES DEPARTEMENTALES DE SEINE-ET-MARNE

Le service des Archives départementales fait partie de la Direction des Archives et du Patrimoine, qui est un des services départementaux du Conseil général de Seine-et-Marne.

2.1 Ses missions

Les Archives départementales sont responsables de :

- la collecte des documents anciens ou récents produits par les administrations, les organismes de droit public et privé, les notaires, les hôpitaux, les communes, les familles et les entreprises,
- la constitution d'une documentation de complément pour l'histoire locale composée de documents iconographiques et de presse, de revues, d'ouvrages et d'imprimés,
- la conservation des documents en luttant contre les dégradations de toutes origines,
- l'inventaire des documents sans intérêt administratif, ni historique,
- la communication des documents en salle de lecture.

Les Archives départementales sont en outre chargées du dépôt légal pour les revues, bulletins municipaux et publications diverses éditées en Seine-et-Marne. Ces documents doivent être déposés dès le premier numéro et à chaque nouvelle parution.

Enfin, les Archives départementales mettent à disposition du public, en salle de lecture, en consultation sur place, un fonds d'ouvrages et de revues spécialisés en histoire et en art. Ce sont des documents généraux de référence, des ouvrages spécialisés sur la Seine-et-Marne ou des personnages célèbres du département, des travaux universitaires consacrés à l'histoire locale ou générale et déposés par leurs auteurs.

2.2 Sa structure

Pour mener à bien ses missions, le service des Archives se décompose en deux catégories de services, des services généraux et d'autres spécialisés en fonction de l'activité des archivistes et documentalistes.

Les services généraux assurent la comptabilité générale, la gestion des recettes, la conservation et le traitement informatisé des collections.

Le service responsable du traitement informatisé des collections comprend lui-même un comité technique de l'Association des Utilisateurs de GAIA et des services chargés respectivement de la concession des droits d'usage de GAIA, des analyses et développements, des méthodes et des normes, des états des fonds et des versements contemporains.

Les services spécialisés sont un nombre de quatre. Ils ont la charge respective :

- des archives notariales et privées, de la bibliothèque et des documents iconographiques,
- des archives contemporaines et communales,
- des archives des hôpitaux, de la police et de la gendarmerie,
- de la salle de lecture et des communications de documents au public.

3. LA GESTION INFORMATISEE DES ARCHIVES : GAIA

3.1 Historique

L'informatisation des archives remonte à 1984. De 1984 à 1986, les Archives départementales ont mené des études pour déterminer les objectifs généraux, puis les fonctionnalités de l'informatisation des archives, en concertation avec l'ensemble des personnels concernés. Suite à cette étude, une enquête a mis en évidence l'absence de produit commercialisé répondant aux besoins exprimés. Ce constat a conduit le Département de Seine-et-Marne à décider le développement d'un logiciel d'archives **GAIA, Gestion Automatisée et Informatisée des Archives**, par son service informatique dès 1987.

Quelques dates :

1987 : début du développement de GAIA sous Oracle, gestionnaire de base de données relationnel qui constitue alors la seule réponse informatique capable de garantir une intégration des données de gestion et de description documentaire.
1989 : livraison du module de gestion des cotes, de leur disponibilité et de leur localisation.
1990 : livraison des procédures de gestion des lecteurs et des communications.
A partir de cette année, l'envergure du projet et les coûts induits ont été tels que les droits d'usage du logiciel ont dû être commercialisés via une société de services, Artdeal actuellement disparue. Dix départements ont acquis les droits d'usage de GAIA.
1992 : livraison des procédures de description documentaire des documents d'archives.
1994 : livraison des procédures de description documentaire des monographies et périodiques.
1995 : livraison des procédures de description des images.
1996 : livraison des procédures d'interrogation sommaire et des éditions de travail.

3.2 Originalités et fonctionnalités

L'intérêt principal de GAIA réside en l'approche globale de toutes les fonctions d'un service d'archives, depuis la gestion des stocks jusqu'à la description intellectuelle des documents. Ceci est possible grâce à l'aspect relationnel du système de gestion de base de données d'Oracle.

GAIA est basée sur des dictionnaires d'indexation communs tant aux fonds d'archives de toutes époques qu'aux publications imprimées ou aux images.
Trois procédures de description, correspondant chacune à un type de document, sont proposées :

- les documents d'archives avec plusieurs niveaux de description : la série, le fonds, le groupe d'articles cotés, l'article coté, la partie d'article (une pièce à l'intérieur d'un article),
- les publications imprimées avec trois niveaux de description qui permettent de gérer les monographies en un ou plusieurs volumes, les périodiques et l'état de collection,
- les images : avec une structure de description spécifique sur deux niveaux pour la description de l'article coté, support et objet représenté.

Plus généralement, GAIA permet de gérer :

- l'identification des documents et de leur cotation via la table des cotes qui est la table principale de tout le système,
- la description matérielle de document,
- la description intellectuelle du document grâce aux trois procédures évoquées ci-dessus,
- la gestion des communications des documents,
- les éditions telles que les rapports annuels à caractère statistique, les éditions documentaires et de gestion,
- les interrogations destinées aux archivistes.

3.3 Caractéristiques techniques

Gaia a été développée sous Unix et tourne actuellement sous AIX (Unix d'IBM). Elle est supportée par les versions 6/7 d'Oracle (versions variables selon les sites où le logiciel est installé). Elle est composée d'environ deux cents tables, de trois cents applications réalisées en SQL*Forms sous différentes versions, de 2.0 à 3.0, de 50 requêtes SQL (procédures PL/SQL et SQL/Plus) et de programmes batch écrits en langage C. En 1996, deux architectures sont en place (elles diffèrent selon les sites) : soit la base de données est centralisée sur mini-ordinateur, les terminaux sont passifs en émulation VT200 sous Windows ; soit la base est en mode client/serveur, les programmes sont alors répartis entre le serveur et les postes clients, seules les données sont stockées sur le serveur. L'accès à la base se fait par l'outil SQL*Net. Cette seconde architecture va se généraliser au dépend de la première.

3.4 Maintenance technique

L'équipe informatique des Archives dédiée au support de GAIA est limitée à deux personnes : un technicien, M. Albert GROSS, et un chef de projet, Mme Claude CHIESA. Elle a été renforcée, en janvier 1996, par l'Association de Utilisateurs de GAIA. Cette structure, elle-même constituée de deux personnes, un chef de projet, M. Ernest SOSSAVI et une chargée de gestion, Mme Emmanuelle Bouteiller-Dupas, assure la maintenance sur tous les sites GAIA et est responsable des évolutions du logiciel. Elle a notamment la charge de la réécriture de la base pour le passage en environnement graphique.

3.5 Evolutions futures

La version client/serveur 1996 de GAIA propose des procédures d'interrogation sommaire destinées uniquement à l'archiviste et des éditions de travail. D'ici à 1997, les Archives départementales et l'Association des Utilisateurs de GAIA prévoient des développements qui devraient aboutir à la livraison de nouveaux modules. Les Archives départementales sont en train de développer de nouvelles éditions et l'Association des utilisateurs de GAIA prépare une nouvelle version de GAIA sous ergonomie Windows avec les versions 7 d'Oracle et 4.5 de Forms. De 1997 à 1998, Le département de Seine-et-Marne devrait livrer de nouvelles procédures d'interrogation pour l'utilisateur final.

4. LA REALISATION DE LA MAQUETTE : OBJECTIFS ET OUTILS DE DEVELOPPEMENT

4.1 La commande de prestations à Oracle

Afin de préparer dans les meilleures conditions, les évolutions futures de GAIA, la Direction des Archives et du Patrimoine a commandé à la société Oracle des prestations d'études et de conseil en développement. Celles-ci avaient un double objectif : revoir les aspects techniques à faire évoluer pour la réalisation de la nouvelle version de GAIA sous Windows et redéfinir de nouvelles normes d'ergonomie à partir de modules existants. Ces prestations devaient s'exercer à travers trois actions :

- la revue des données aux niveaux conceptuel, logique et physique,
- la définition de nouvelles normes d'ergonomie de GAIA par le redéveloppement des quelques modules représentatifs, choisis en accord avec les Archives départementales, représentées par l'Association des Utilisateurs de GAIA, et la société Oracle, en la personne de M. TAIMA, ingénieur conseil,
- l'installation de l'environnement de développement.

Compte tenu du nombre de jours limités pour la réalisation de l'étude, de la complexité et de l'importance de GAIA en termes de tables et de programmes, il s'est avéré impossible d'envisager l'étude telle que prévue initialement. M. TAIMA et M. SOSSAVI ont donc convenu de choisir non pas quelques modules représentatifs, mais UN module afin de mener une étude exhaustive du point de vue méthodologique et surtout parfaitement documentée. Au long terme, cette maquette se devait d'être exemplaire au sens où la méthode employée pour la modélisation, serait par la suite appliquée à l'ensemble de la base en prévision de sa réécriture. A court terme, elle devait permettre à l'utilisateur d'entrevoir ce que serait GAIA en environnement graphique.

L'environnement de développement prévu se compose de trois éléments : la méthode de modélisation Oracle dite *Custom development method*, et les outils logiciels graphiques de développement *Designer 2000*, le tout dernier atelier de génie logiciel d'Oracle et à sa suite, le kit de développement *Developer 2000*.

4.2 Le domaine fonctionnel de la maquette

La gestion des entrées a été choisie comme domaine fonctionnel pour la réalisation de la maquette en raison de son rôle important dans GAIA. L'entrée est comparable à un registre chronologique où sont enregistrées toutes les nouvelles arrivées de documents aux Archives départementales de Seine-et-Marne. Les entrées représentent tous les versements de documents effectués auprès des Archives. Elles peuvent être définitives ou temporaires. Les premières concernent les documents versés définitivement pour traitement et archivage, les secondes les documents transmis temporairement pour communication, microfilmage, exposition, prêt ou reliure.

Fonctionnalités actuelles de la gestion des entrées

Le module de la gestion des entrées est accessible depuis le menu général de l'application. A l'intérieur de ce module, l'utilisateur se voit proposer six choix :

1. Enregistrer une entrée : l'utilisateur accède alors à un écran où il doit sélectionner un type d'entrée en fonction duquel il va accéder à différents écrans de description.
2. Rechercher une entrée quel que soit le type du document.
3. Entrer une prévision d'entrée temporaire, ce qui revient à saisir la demande du document effectuée auprès de l'organisme propriétaire.
4. Consulter le registre des entrées temporaires pour savoir à quel stade de la procédure on se situe.
5. Créer un motif d'entrée temporaire.
6. Créer un organisme extérieur.

Des éditions sont également possibles. Elles sont accessibles depuis le module des éditions de GAIA. Elles proposent l'édition d'un rapport annuel du registre d'entrée et des éditions par type légal de document (publications, archives et documents d'images).

Objet de l'étude

Le module de gestion des entrées étant lui-même relativement complexe, il a été décidé de se concentrer uniquement sur un de ses sous-domaines fonctionnels, la partie relative aux entrées définitives, les éditions non comprises.

4.3 La méthode de modélisation Oracle

Custom development method (ou *CDM*) est la méthode de modélisation développée et vendue par Oracle. Elle sert à mettre sous forme de schémas les énoncés de l'application en suivant des règles bien précises. *CDM* est l'héritière directe de *CASE*Method* (*Computer Aided Software Development Method*) dont elle garde les grands principes. *CDM* se compose de deux phases : l'analyse préalable (phase de stratégie dans *CASE*Method*) et l'analyse détaillée (phase d'analyse dans *CASE*Method*). Durant l'analyse préalable, les différents processus de travail de l'entreprise ou organisation sont analysés très précisément, les fonctions sont décomposées en hiérarchie, le schéma conceptuel des données est élaboré et les règles de gestion sont recensées et classées selon la classification *CDM*. La phase d'analyse détaillée ne fait que reprendre tous ces éléments, à l'exception du modèle des processus, pour les étudier beaucoup plus en détails et finalement les valider avant la phase de conception.

Contrairement à *CASE*Method*, *CDM* n'inclut pas la phase de conception. Pour mémoire, celle-ci comprend la spécification et la description des différents modules de traitement, la définition des éléments du schéma physique, les index, les vues, les tables dénormalisées, les contraintes d'intégrité et les déclencheurs, les droits d'accès et les paramétrages de stockage. Les tables et les modules sont validés.

4.3.1 Le modèle des processus

Oracle définit le processus comme une activité ou un ensemble d'activités dans lesquelles une entreprise ou une organisation s'est engagée pour créer de la valeur ajoutée pour ses clients. Le processus comporte un début et une fin, tous deux bien définis et associés à un client.

La modélisation des processus implique la création d'un diagramme qui montre les activités d'une entreprise et l'ordre dans lequel elles se produisent. La plupart des activités étant relativement complexes, un modèle se décompose souvent en un nombre de diagrammes. Chaque diagramme met l'accent sur une partie de l'activité. Un diagramme très global donne une vue d'ensemble de tout le processus au niveau le plus haut. Il aide à comprendre le fonctionnement général de l'entreprise ou de l'organisation.

Lors de l'analyse, l'analyste cherche à déterminer précisément six éléments :

1. le département, le service ou l'unité responsable d'une phase particulière du processus (le *business unit*),
2. la phase du processus (*process step*) : l'activité, la tache ou la fonction qui représente une phase particulière du processus,
3. les flux (*flows*), c'est à dire les mouvements informationnels et matériels,
4. les réservoirs (*stores*) informationnels et matériels,
5. les événements qui déclenchent des processus (*triggers*),
6. les sorties (*outputs*) qui résultent des processus.

La modélisation des processus propose donc une vue horizontale de l'activité d'une entreprise ou d'une organisation, par opposition à la hiérarchie des fonctions, expliquée ci-après dans ce chapitre. Celle-ci s'apparente plus à une vision verticale.

4.3.2 Le modèle entité-relation

Le modèle entité-relation a pour but la modélisation des données. Il permet de produire une abstraction du monde réel. Il est basé sur trois concepts :

1. **L'entité** : c'est l'objet porteur d'information, identifiable et pertinent par rapport au système d'information étudié. Cette entité est porteuse de sens : elle a des caractéristiques qui lui sont propres et elle est représentée par plusieurs occurrences. Chaque entité est identifiable : chaque occurrence de l'entité est discernable et identifiée par un code unique. Une entité peut constituer une classe d'entités, ses sous-entités représentent alors des sous-catégories. Par exemple, l'entité lecteur peut être décomposée en sous catégories : lecteur en salle, administration et autre organisme.

2. **L'attribut** : c'est une caractéristique de l'entité. Une entité a au moins un attribut. Parmi ses attributs, un est l'identifiant unique qui deviendra la clé primaire au niveau logique. Un attribut ne peut avoir qu'une valeur, celle-ci étant élémentaire et non décomposée.

3. **L'association** : c'est un lien sémantique entre deux entités, matérialisé par deux verbes et une cardinalité. Cette dernière est définie comme le nombre de fois où l'entité participe à la relation. Les principales cardinalités sont de type (1,1) ; (1,n) il s'agit dans ce cas d'un lien maître-détails ; (n,m) ce sont les relations porteuses d'informations. En plus des deux identifiants représentant la relation, cette association a souvent des caractéristiques qui lui sont propres et devient elle-même une entité.

Le schéma conceptuel des données est la représentation graphique des entités, de leurs attributs et de leurs relations construit selon un formalisme graphique précis. C'est le modèle de l'information à l'intérieur de l'entreprise, il illustre la façon dont cette information est liée.

4.3.3 Le modèle des fonctions

La modélisation des fonctions est définie comme une vue d'ensemble de toutes les fonctionnalités à faire supporter par le système d'information. A la différence du modèle des processus, la hiérarchie des fonctions ne montre pas la séquence des activités. Elle décrit ce que fait l'entreprise et ce qu'elle devrait faire pour atteindre ses objectifs. La hiérarchie des fonctions est le modèle qui formalise les traitements, il repose sur deux concepts : la fonction et la hiérarchie.

La fonction est une activité plus ou moins importante qui permet à l'entreprise de communiquer en interne et en externe, de mettre à jour et à disposition ses données. Une fonction peut être automatisée ou manuelle. Elle se matérialise par un verbe à la voix active et éventuellement un complément. Elle est caractérisée par une fréquence, c'est-à-dire le nombre de fois où elle se produit par jour, par semaine, par mois, par an, etc. Certaines actions peuvent être identiques : elles accomplissent la même action et ont la même fréquence. Ces fonctions sont dites 'communes'. Dans ce cas, l'une d'elles est déclarée comme fonction maître et les autres comme fonctions esclaves.

Les fonctions sont organisées en arborescence avec une racine et des branches, ce qui permet de les regrouper selon leur finalité. L'élaboration de la hiérarchie suit une démarche le plus souvent descendante, depuis la branche principale jusqu'aux fonctions les plus élémentaires (les feuilles de l'arbre) qui sont indécomposables.

Chaque fonction de la hiérarchie est associée à une entité. Cette association détermine quelles sont les entités utilisées par une fonction élémentaire et comment ces entités sont utilisées. Toutes les

associations fonctions/entités sont synthétisées dans une matrice dite *CRUD*, pour *create*, *retrieve*, *update* et *delete* (créer, rechercher, mettre à jour et effacer).Celle-ci permet de vérifier qu'à chaque fonction correspond une entité associée et qu'aucune fonction n'a été oubliée. De la même façon, l'utilisation des attributs par les fonctions peut être vérifiée par une matrice *CRUD* fonctions/attributs : chaque attribut doit être utilisé par au moins deux fonctions, la première en création, la seconde en recherche. Cette matrice est utile pour clarifier les différentes fonctionnalités du système d'information.

4.3.4 La modélisation des règles de gestion

Les règles de gestion sont plus ou moins modélisées lors de la création du modèle entité-relation et des fonctions. Toutefois, leur modélisation à part entière présente un certain nombre d'avantages. Les types de fonctionnalités faisant partie de la couche applicative logique sont définis explicitement, écartant d'emblée toutes les fonctionnalités liées à la présentation. La modélisation de ces règles constitue un excellent support pour les tâches de conception complexes. Elle aide à déterminer l'endroit le plus approprié (le client ou le serveur) pour l'implémentation des différents composants fonctionnels. Enfin, *Custom development method* propose une classification des règles de gestion. Cette dernière s'appuie sur les définitions suivantes : les règles de gestion sont soit une restriction appliquée à l'état des données ou à leur changement, soit une action automatique qui se produit suite à un changement d'état des données. Elles sont réparties en quatre grandes classes : les règles de données statiques (*static data rules*), les règles de traitement des données (*data operation rules*), les règles d'événement suite à un changement d'état (*change event rules*) et les règles d'autorisation (*authorization rules*). Certaines règles seront enregistrées en tant qu'éléments de description de l'entité, d'autres le seront en tant que fonctions de gestion.

Les règles de données statiques

Ces règles concernent les attributs, les tuples, les entités et les relations entre celles-ci.
Les règles sur les attributs déterminent les formats (caractère, numérique, date pour ne citer que les principaux), les domaines et les attributs obligatoires.
Les règles sur les tuples définissent les valeurs autorisées qui dépendent de la valeur des autres attributs de la même occurrence. Par exemple : une entrée doit avoir une date d'entrée.
Les règles sur les entités précisent les valeurs autorisées qui dépendent de la valeur d'attributs d'autres occurrences de la même entité. Par exemple, une entrée doit être identifiée de manière unique par un numéro d'entrée, pas plus de cinq types légaux de documents sont autorisés.
Les règles inter-entités définissent les valeurs d'attributs autorisées qui dépendent de la valeur des attributs d'une occurrence d'autres entités, ainsi que les relations entre les entités et leurs conditions (cardinalités). Par exemple, une entrée ne doit avoir qu'une et une seule localisation.

Les règles sur les traitements des données

Ces règles définissent les conditions dans lesquelles se font la création, la mise à jour et la suppression. Les règles de création restreignent la création d'une entité ou d'une relation, celles de mise à jour la modification des attributs, des entités et des relations, et celles d'effacement, la suppression d'une entité ou d'une relation.

Les règles d'événement suite au changement d'état des données

Ces règles déterminent les actions automatiques dérivées qui se produisent suite à un changement d'état des données. Elles précisent le changement de données déclenché par l'action automatique. Ceci peut se traduire par la création ou la suppression d'une entité, la création ou la mise à jour d'une valeur d'attribut, la création, le transfert ou la suppression d'une relation.

L'action automatique relève souvent de la manipulation de données, mais elle peut aussi se traduire par une autre opération telle que l'envoi d'une édition. Ainsi, l'édition des communications suite aux demandes de cotes des lecteurs.

Les règles sur les autorisations

Elles sont basées sur la définition des postes de travail (*business units*) définis lors de l'élaboration du modèle des processus. Les autorisations peuvent être de trois ordres :

- effectuer certaines fonctions,
- accéder à certaines entités et certains attributs et enfin les manipuler,
- accéder à un sous-ensemble d'occurrences d'entité et les travailler.

Le prototypage

Le prototypage est une technique utile quand elle est employée au moment et à l'endroit voulus. *Custom development method* insiste sur le fait que son objectif doit être clair pour tout le monde, aussi bien l'analyste que les utilisateurs. Ce peut être la méthodologie de modélisation, la performance du système, son architecture ou ses fonctionnalités. Le prototype doit être vu comme une application éphémère. Il doit faire l'objet d'une préparation, d'une planification et d'une documentation soignées.

Custom development method est donc un outil de travail et de synthèse dans le prolongement direct de *CASE*Method,* directement supporté par *Designer 2000*. Il permet de comprendre très précisément la nature des informations à stocker dans la mémoire du système d'information et des traitements à faire supporter par ce dernier. Son application à l'analyse d'un système d'information, comme celui de GAIA, permet d'aborder la phase de conception sur des bases solides. Sa mise en oeuvre oblige à une démarche systématique et rigoureuse et à la production de documents de référence. Ces derniers s'avéreront très utiles, voir indispensables, lors de la phase de maintenance.

4.4 Les outils logiciels de conception et de développement

4.4.1 Designer 2000

De même que *Custom development method* était l'héritière de *CASE*Method*, *Designer 2000* est le successeur de l'atelier de génie logiciel *CASE*. A la différence de ce dernier, *Designer 2000* est un outil graphique et dispose d'un plus grand nombre de fonctionnalités.

Designer 2000 fonctionne sur la base d'un dictionnaire conceptuel (*repository*). Ce dictionnaire est une base de données contenant la description complète des données et des traitements du système d'information. Il constitue le référentiel d'analyse et de développement informatisé. Il présente l'avantage de pouvoir être tenu à jour très facilement. *Designer 2000* est composé de cinq modules, le modeleur de processus (*process modeller)*, le modeleur système (*system modeller)*, le concepteur système (*system designer)*, les générateurs (*generators*) et l'administration du dictionnaire conceptuel (*repository administration)*.

Le modeleur de processus

Grâce à ce module, l'analyste peut modéliser les phases essentielles d'un processus, leurs enchaînements tout en y incluant les agents responsables. Les objets ainsi modélisés sont ceux-là mêmes qui sont manipulés par les outils de modélisation des fonctions du modeleur système.

Le modeleur système

L'analyste travaille au niveau conceptuel et dispose de trois outils : le modèle entité-relation (*entity-relationship diagrammer* ou *ERD*), le modèle hiérarchique des fonctions (*function hierarchy diagrammer* ou *FHD*) et le modèle des flux (*dataflow diagrammer*).

Le modèle entité-relation permet de décrire les différentes entités et relations du système d'information. Un même schéma de base peut servir à la création d'un nombre infini de modèles de complexité plus ou moins grande.

Le modèle hiérarchique des fonctions sert à décrire les différentes fonctions d'un système d'information, tout en montrant leur décomposition hiérarchique.

Le modèle des flux de données permet de spécifier, pour chaque activité supportée par le système, quelle information est utilisée et comment elle l'est.

Le concepteur système

A travers ce module, l'analyste dispose de six outils de conception.

Le modèle de données (*datadiagrammer)* permet de créer un schéma de base de données standard, Oracle 7 ou ANSI.

Le modèle de structure des modules (*module structure diagrammer)* donne la possibilité de définir les fonctionnalités proposées par le système d'information à ses utilisateurs, la présentation adoptée et l'implémentation des flux.

Le modèle des données du module (*module data diagrammer*) permet à l'analyste de déterminer la façon dont tous les programmes sont assemblés dans un module. Par exemple, il peut définir la façon dont se fait l'accès aux tables et aux colonnes, dont les données sont utilisées, affichées, mises à jour et validées.

Le navigateur logique des modules (*module logic navigator*) permet de créer et d'éditer des *packages*[1], des procédures, des fonctions et des déclencheurs[2] PL/SQL pour implémentation par le *server generator*.

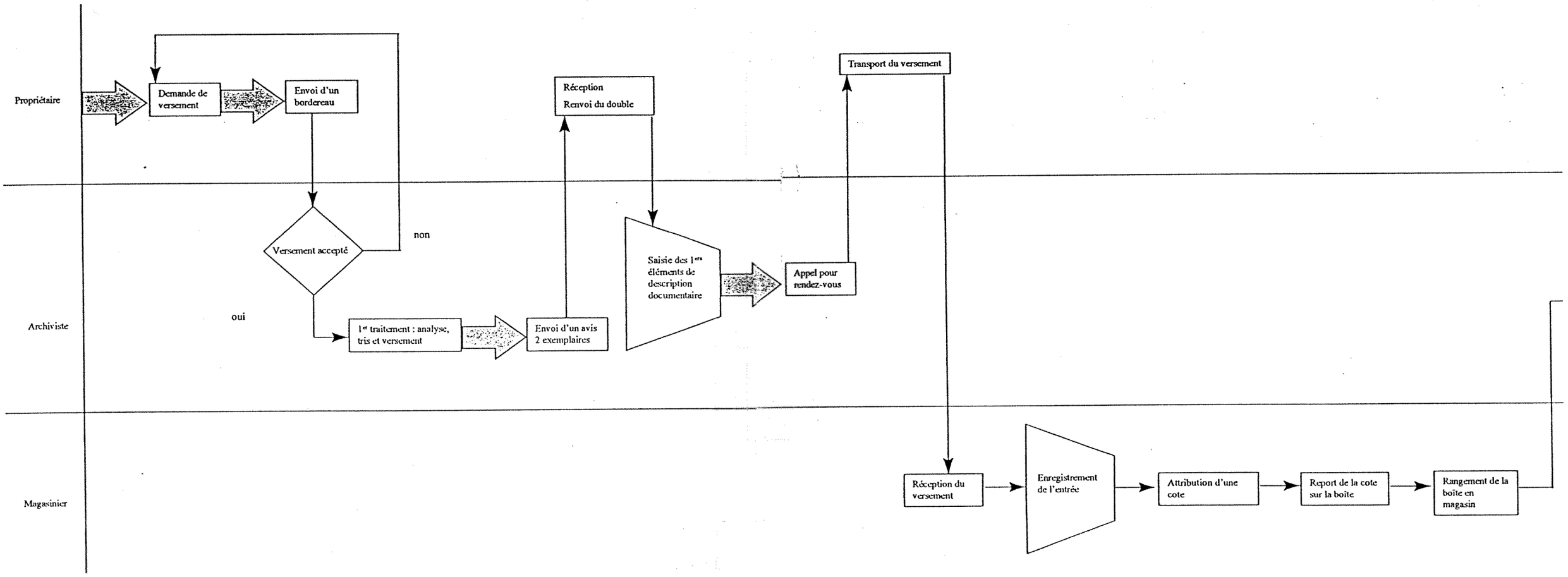
Le navigateur de préférences (*preferences navigator)* sert à créer le guide de style de l'interface graphique. Des standards de présentation et de comportement d'objets (boîtes de dialogues par exemple) peuvent y être paramétrés.

L'assistant conception base de données (*database design wizard*) génère à partir du diagramme entité-relation le modèle logique de l'application. Il déduit lui-même les tables, les colonnes, les clés, les index, les modules de programmes nécessaires pour supporter les fonctions, les flux de données et les modèles de processus déclarés dans les diagrammes de la hiérarchie des fonctions, des flux et des processus.

---

[1] Les *packages* sont des objets de la base. Ce sont des collections de fonctions, de procédures et d'autres objets stockés au sein d'une base de données. Ces derniers doivent être spécifiés lors de la création du *package*. Les *packages* sont soit privés, soit publics.

[2] Les déclencheurs (*triggers*) sont des procédures stockées dans la base et associée à un événement qui peut intervenir sur une table. Ces procédures s'exécutent lorsque qu'une commande SQL de mise à jour affecte la table associée au déclencheur. Pour plus d'informations, se reporter à la partie 9.1.2).


Propriétaire
Demande de versement
Envoi d'un bordereau
Versement accepté
non
oui
Archiviste
1er traitement : analyse, tris et versement
Envoi d'un avis 2 exemplaires
Réception
Renvoi du double
Saisie des 1ers éléments de description documentaire
Appel pour rendez-vous
Transport du versement
Magasinier
Réception du versement
Enregistrement de l'entrée
Attribution d'une cote
Report de la cote sur la boite
Rangement de la boite en magasin

Les générateurs

Trois générateurs sont disponibles. Le générateur d'écrans (*forms generator*) exploite les possibilités de *Forms 4.5*. A partir du modèle général présent dans le dictionnaire conceptuel, il construit une définition interne complète de la *form*[3] à générer. Le générateur de rapports (*reports generator*) génère des éditions, visualisables à l'écran, récupérables dans un fichier ou directement imprimables, à partir des définitions contenues dans le dictionnaire conceptuel (entités, domaines, fonctions, etc.). Enfin, le générateur de serveur, *server generator*, transforme les définitions de procédures, déclencheurs, fonctions et *packages*, créés dans le navigateur logique, *module logic navigator*, en du code implémenté PL/SQL. Il permet de gérer des objets tels que les tables, les vues, les rôles[4], etc.

L'administration du dictionnaire conceptuel

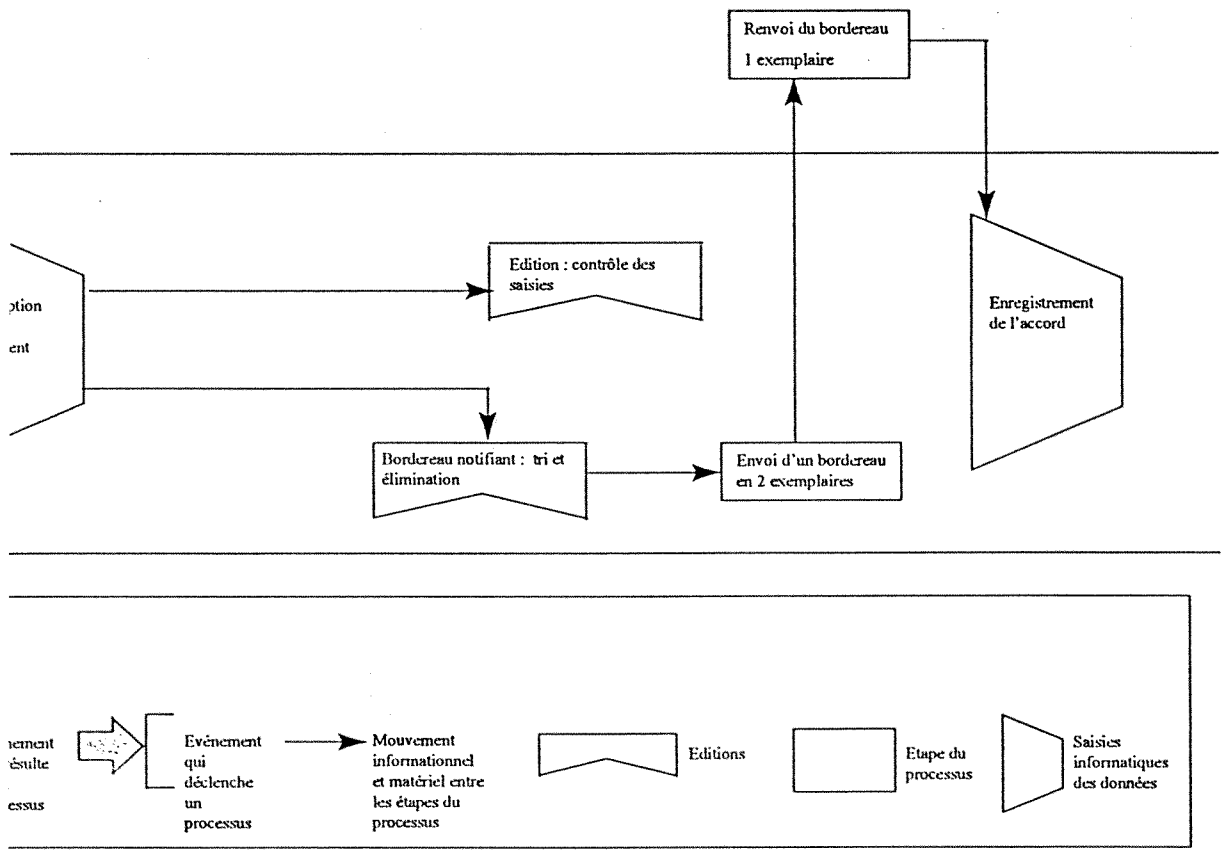
L'administration du dictionnaire conceptuel se fait via deux outils : le navigateur du dictionnaire conceptuel, *repository navigator*, et le générateur de matrices, *matrix diagrammer*.
Le premier gère les différents types d'objets, depuis les fonctions jusqu'aux préférences d'affichage définies pour les écrans. Il hiérarchise les objets ainsi que les liens qui les relient entre eux. Il permet de modifier les propriétés d'un groupe d'objets.
Le second propose une vision croisée du dictionnaire. Il permet de construire des matrices spécifiques aux besoins du système d'information (matrices fonctions/entités, fonctions/attributs).

4.4.2 *Developer 2000*

Developer 2000 est l'outil de développement utilisé dans le prolongement direct de *Designer 2000*. C'est un outil composite constitué à la base de *Forms 4.5*, auquel ont été intégrés *Reports 2.5* et *Graphics 2.5*. Il sert avant tout à concevoir, développer et mettre en oeuvre des applications client/serveur basées sur des écrans en interface à un serveur Oracle 7 ou d'autres bases de données relationnelles. Une application *Forms* peut contenir des modules issus de *Reports* et *Graphics*.

[3] La *form* est une interface utilisateur interactive qui permet de travailler directement avec les lignes des tables et les vues dans le serveur.

[4] Le rôle est un objet de la base qui définit les droits d'un utilisateur par rapport aux objets de celle-ci (les tables, les vues, les séquences, les procédures) et aux privilèges système. Ces derniers représentent toutes les opérations pouvant être effectuées sur toute la base : création d'une table, recherche sur un ensemble de tables, etc.


Renvoi du bordereau
1 exemplaire
ption
ent
Edition : contrôle des saisies
Enregistrement de l'accord
Bordereau notifiant : tri et élimination
Envoi d'un bordereau en 2 exemplaires

ement
ésulte
essus
Evénement qui déclenche un processus
Mouvement informationnel et matériel entre les étapes du processus
Editions
Etape du processus
Saisies informatiques des données

5. L'ANALYSE PREALABLE

L'analyse préalable a constitué la toute première phase de la modélisation. Elle a consisté en une analyse globale des données et des traitements à prendre en compte pour la maquette. Un certain nombre d'informations a été rassemblé. Cette collecte s'est faite à partir d'entretiens avec Ernest SOSSAVI et Claude CHIESA, à partir de quelques documents existants sur GAIA et de la partie du manuel utilisateur de GAIA consacrée à la gestion des entrées.

Un modèle des processus général a été élaboré, les entités ont été identifiées, les relations établies, les domaines définis, les attributs sélectionnés et décrits. Nous avons utilisé trois outils de *Designer 2000* : le modeleur de processus (*process modeller*), le modèle entité-relation (*entity-relationship diagrammer*) et le modèle hiérarchique des fonctions (*function hierarchy diagrammer*).

5.1 Le modèle des processus

Nous avons élaboré un modèle qui formalise l'ensemble des activités, manuelles et informatisées, effectuées lors d'un versement définitif (voir schéma ci-contre). Nous avons recensé trois unités organisationnelles : le propriétaire, l'archiviste et le magasinier.

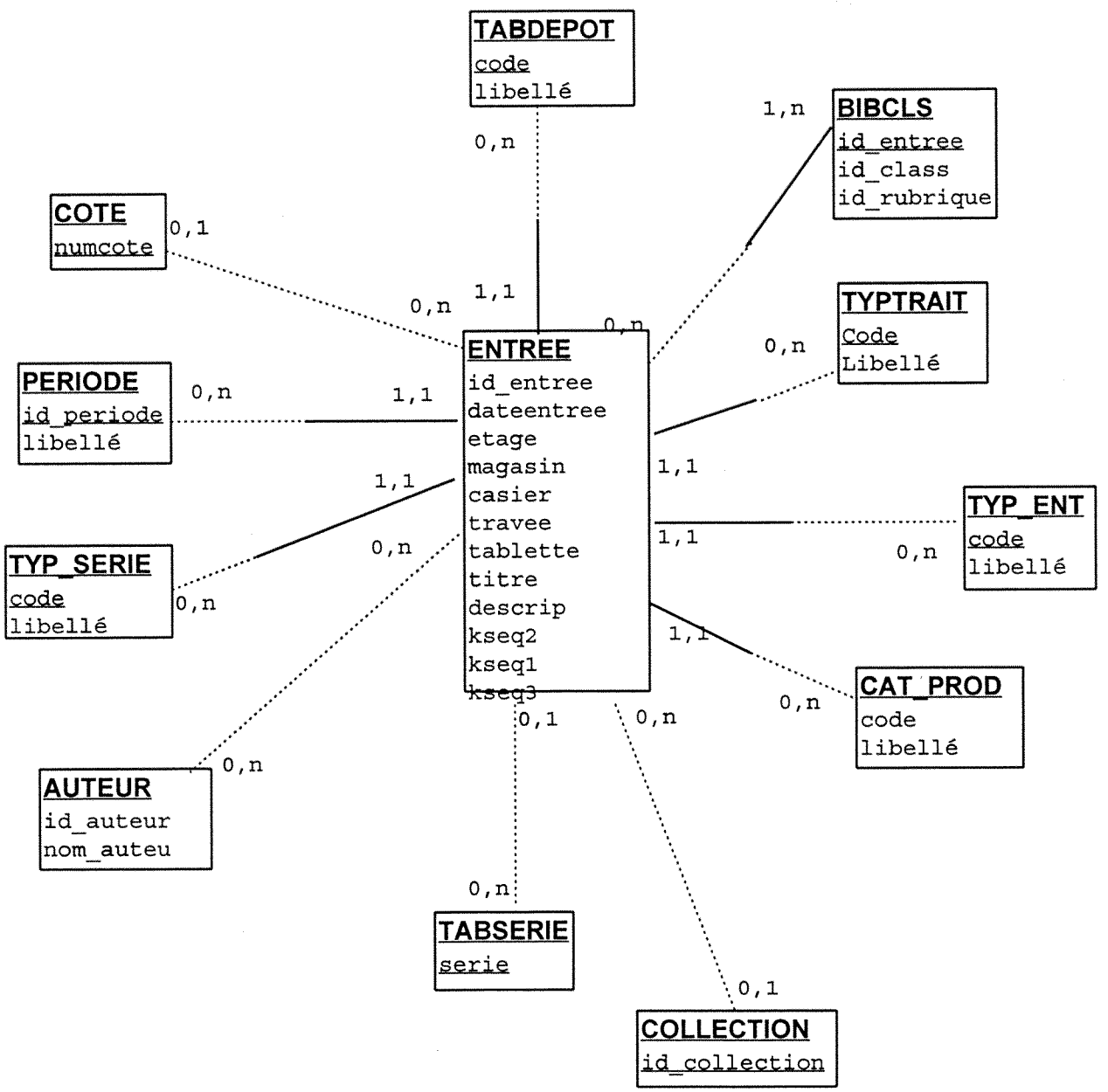
5.2 Les entités

Nous avons identifié toutes les entités impliquées dans la gestion des entrées et utilisées par toutes les fonctions de ce domaine.

Les douze entités utilisées pour la gestion des entrées sont les entités représentant le registre des entrées (`entrée`), la rubrique de classement bibliographique (`bibcls`), la catégorie des producteurs (`cat_prod`), la collection (`collection`), la cote (`cotes`), la période (`période`), le dépôt (`tabdépôt`), la série (`tabsérie`), le type d'entrée (`typ_ent`), le type de traitement (`typtrait`), le type de la série (`typsérie`). Nous avons exclu l'entité `bibcls` car elle ne joue pas un rôle fondamental dans le module des entrées.

5.3 Les relations

Le diagramme entité-relation de l'existant :



5.4 Les domaines et les listes de valeurs

Lors de l'analyse de ce schéma, nous nous avons remarqué un certain nombre de relations avec une cardinalité[1] (1,n). Ces relations (1,n), entre les entités, sont les propriétés des entités associées. Au niveau logique, elles seront représentées par des colonnes communes (code ou libellé) dont les valeurs, du point de vue du sens, sont comparables. Cet ensemble de valeurs, dans la mesure où il peut être partagé par plusieurs colonnes, répond à la définition du domaine.

Nous avons donc implémenté certaines de ces relations en domaines. Il s'agit des relations entre `entrée` et `cat_prod`, `typtrait`, `typserie`, `tabdépôt` et `période`. Les valeurs utilisées pour l'affichage sont soit celles du code, soit celles du libellé. De plus, elles font moins de quatre-vingt caractères et la question de la mise à jour ne s'est pas posée.

Nous avons exclu la transformation de `typ_ent` en domaine pour des raisons d'affichage car nous souhaitions afficher le code ET le libellé. De la même façon, la relation avec l'entité `tabsérie` n'a pas été modifiée car l'attribut `tabsérie.série` n'est utilisé que pour la validation. Aucun affichage n'est nécessaire.

5.5 Les attributs

Nous avons listé tous les attributs utilisés par toutes les fonctions dans la gestion des entrées. Nous avons dû faire des choix. Pour la seule entité `entrée`, leur nombre s'élevait à quarante six. Dans le temps qui nous était imparti, leur étude était impossible. Nous n'avons donc retenu que les attributs de type identifiants uniques, les attributs obligatoires et importants.

Nom Entité	Identifiant unique	Attributs obligatoires	Attributs importants
Entrée	id_entrée	codedoc	titre
		dateentrée	descrip
		dépôt	kseq1
		étage	ksérie
		magasin	kseq2
		casier	kseq3
		travée	
		tablette	
		période	
		codelégal	
		cat_prod	
		typ_ent	
Auteurs	id_auteur	nom_auteur	
Cotes	numcote		
Collection	id_collection		
Tabsérie	série		
Typ_ent	code	libellé	

[1]La cardinalité représente le nombre de fois où l'entité participe à la relation.

Attributs utilisés dans les listes de valeurs

Pour l'entité utilisée en liste de valeurs, typ_ent, nous avons précisé l'identifiant unique et les attributs qui seront affichés :

Attribut	Identifiant unique	Affichage
code	oui	oui
libellé		oui

Format des attributs

Au niveau conceptuel, les caractéristiques à définir concernent le format et sa longueur maximale. Tous les attributs ont été examinés, y compris ceux des futurs domaines.

Entité	Attribut	Format	Longueur
entrée	id_entrée	char	12
	codedoc	char	3
	dateentrée	date	7
	kserie	char	7
	kseq1	integer	5
	kseq2	integer	5
	kseq3	char	31
	dépôt	char	2
	étage	char	2
	magasin	char	1
	casier	char	4
	travée	char	1
	tablette	char	2
	code_légal	char	3
	titre	char	255
	id_période	integer	7
	typ_ent	char	3
	cat_prod	char	3
	descrip	char	2000
cat_prod	code	char	3
collection	id_collection	integer	7
cotes	numcote	integer	8
auteurs	id_auteur	integer	7
	nom_auteur	char	100
période	libellé	char	60
tabdépôt	code	char	2
tabsérie	serie	char	7
typ_ent	code	char	3
	libellé	char	60
typserie	libellé	char	40
typtrait	libellé	char	100

5.6 Les fonctions

Les fonctions dans la gestion des entrées ont été décomposées et organisées en hiérarchie selon les recommandations données dans l'ouvrage de M. SMIME sur *Case*[2]. Ces dernières sont brièvement rappelées dans les trois paragraphes qui suivent.

Principes de la modélisation des fonctions

Chaque fonction est codifiée de façon à être identifiée de manière unique et référencée facilement. Cette codification est séquentielle de haut en bas et de gauche à droite. La numérotation est espacée avec un pas supérieur à un, de sorte que de nouvelles fonctions puissent être insérées entre celles existantes tout en respectant la logique du moment.

L'arborescence doit être équilibrée dans la répartition des fonctions sur les branches. La fonction générale représente la synthèse de toutes ses sous-fonctions. Elle résume toute l'activité. La hiérarchie doit contenir au maximum trois niveaux en profondeur et huit niveaux en transversal.

5.6.1 La hiérarchie des fonctions

'Gérer l'entrée' constitue la tache principale, c'est la fonction racine. Elle se décompose en quatre activités principales : créer une entrée, la rechercher, la modifier et la supprimer. Ces fonctions sont dites 'de groupement' car elles sont décomposables en fonctions élémentaires, que l'on peut définir comme des tâches de base.

F gérer l'entrée
F1 créer l'entrée
  F10 saisir les attributs obligatoires
  F11 saisir les attributs importants
    F110 vérifier la structure de la cote
    F111 construire la clé primaire
  F12 valider l'entrée créée
F2 rechercher une entrée
  F20 saisir les critères de recherche
  F21 valider les critères de recherche
  F22 calculer le nombre de documents obtenus en résultat
  F23 afficher le nombre de documents obtenus en résultat
  F24 demander confirmation de modification de la recherche
  F25 afficher le sommaire des résultats
    F250 valider l'entrée sélectionnée
  F26 afficher l'entrée sélectionnée
F3 modifier une entrée
  F30 rechercher une entrée
  F31 modifier les attributs modifiables
  F32 valider la modification
F4 supprimer une entrée
  F40 rechercher une entrée
  F41 supprimer l'entrée
  F42 mettre à jour les enregistrements associés
  F43 valider la suppression

Nous avons identifié un certain nombre de fonctions élémentaires. Celles-ci, à la différence des fonctions de regroupement, deviennent soit des écrans, soit des rapports, soit des utilitaires dans le système applicatif. Il faut donc réfléchir au type d'utilisation des fonctions élémentaires. *Designer*

[2] SMIME H., *Concevoir et développer avec ORACLE et CASE : Oracle versions 6 et 7*. Paris : Eyrolles, 1994. 268 p.

2000 distingue deux types de fonction: *immediate*, la fonction génère alors un écran, et *overnight*, la fonction se traduit alors par une règle de gestion.

Cette première hiérarchie nous a permis de réfléchir aux fonctions semblables (elles sont soulignées). Nous nous sommes interrogés sur leur objet, la façon dont elles manipulent les informations, et sur leur structure et leur définition. Dans la mesure où, sur ces trois aspects, elles étaient identiques, nous les avons déclarées en fonctions communes. 'F2 rechercher une entrée' a été créée en tant que fonction maître, F30 et F40 en tant que fonctions esclaves. La fonction maître est la seule à pouvoir être décomposée. Seule sa description a été détaillée et mise à jour. Pour les fonctions esclaves, nous n'avons indiqué qu'une référence à la fonction maître.

F1 créer l'entrée	
F10 saisir les attributs obligatoires	*immediate*
F11 saisir les attributs importants	*immediate*
F110 vérifier la structure de la cote	*overnight*
F111 construire la clé primaire	*overnight*
F12 valider l'entrée créée	*immediate*
F2 rechercher une entrée	
F20 saisir les critères de recherche	*immediate*
F21 valider les critères de recherche	*immediate*
F22 calculer le nombre de documents obtenus en résultat	*overnight*
F23 afficher le nombre de documents obtenus en résultat	*immediate*
F24 demander confirmation de modification de la recherche	*immediate*
F25 afficher le sommaire des résultats	*immediate*
F250 valider l'entrée sélectionnée	*immediate*
F26 afficher l'entrée sélectionnée	*immediate*
F3 modifier une entrée	
F30 rechercher une entrée	*immediate*
F31 modifier les attributs modifiables	*immediate*
F32 valider la modification	*immediate*
F4 supprimer une entrée	
F40 rechercher une entrée	*immediate*
F41 supprimer l'entrée	*immediate*
F42 mettre à jour les enregistrements associés	*overnight*
F43 valider la suppression	*immediate*

Ensuite, nous avons renommé toutes ces fonctions en veillant à ce que leur nom ne dépasse pas quatre-vingt caractères. Nous avons élaboré une hiérarchie qui sépare les deux types de fonction évoqués ci-dessus. Toutes ces fonctions ont reçu un nom court GAIA qui rappelle le nom de l'application, au dépend de F peu significatif. Nous avons éclaté cette première hiérarchie en deux autres hiérarchies : GAIAE pour les fonctions aboutissant à la gestion des écrans et GAIAT pour les fonctions à implémenter en fonctions de gestion. Cette décomposition nous a permis de clarifier ces deux catégories de fonctions.

Fonctions aboutissant à la génération d'écrans :

GAIAE écrans de la gestion du registre des entrées
GAIA1 créer l'entrée
  GAIA10 saisir les attributs obligatoires
  GAIA11 saisir les attributs importants
  GAIA12 valider l'entrée créée
GAIA2 rechercher une entrée
  GAIA20 saisir les critères de recherche
  GAIA21 valider les critères de recherche
  GAIA23 afficher le nombre de documents obtenus en résultat
  GAIA24 demander confirmation de modification de la recherche
  GAIA25 afficher le sommaire des résultats
    GAIA250 valider l'entrée sélectionnée
  GAIA26 afficher l'entrée sélectionnée
GAIA3 modifier une entrée
  GAIA30 rechercher une entrée
  GAIA31 modifier les attributs modifiables
  GAIA32 valider la modification
GAIA4 supprimer une entrée
  GAIA40 rechercher une entrée
  GAIA41 supprimer l'entrée
  GAIA43 valider la suppression

Fonctions se traduisant par des fonctions de gestion :

GAIAT règles de gestion du registre des entrées
GAIAT1 traitements de gestion de la création des entrées
  GAIAT110 vérifier la structure de la cote
  GAIAT111 construire la clé primaire
GAIAT2 traitements de gestion des écrans de recherche d'une entrée
  GAIAT22 calculer le nombre de documents obtenus en résultat
  GAIAT42 mettre à jour les enregistrements associés
GAIAT3 traitements de gestion de la modification d'une entrée

Enfin, une fonction peut être définie comme élémentaire (elle équivaut alors à une transaction) ou atomique (elle est techniquement indécomposable). Seules les fonctions suivantes ont été définies comme élémentaires :

1. dans la fonction de Saisie d'une entrée : saisir les attributs obligatoires et importants, vérifier la structure de la cote,
2. dans la fonction de Recherche d'une entrée : entrer les critères de recherche, calculer le nombre de documents obtenus en résultat, afficher le nombre de documents obtenus en résultat, le sommaire des résultats et l'entrée sélectionnée,
3. dans la fonction de Modification d'une entrée : valider la modification,
4. dans la fonction de Suppression d'une entrée : supprimer l'entrée.

5.6.2 La matrice fonctions/entités

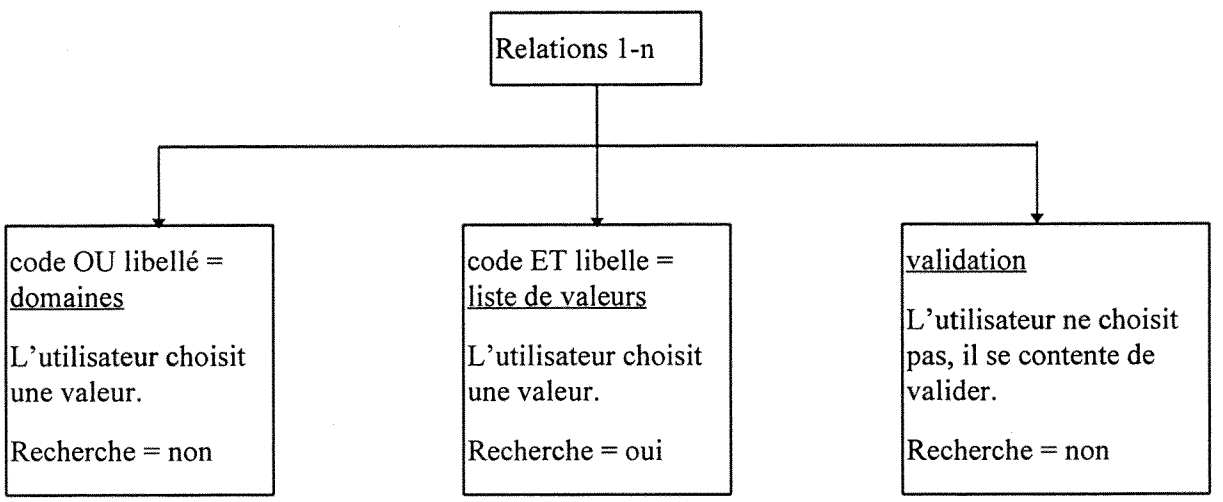
L'élaboration d'une matrice fonctions/entités avait pour nous un double objectif :

1. vérifier que chaque entité était bien utilisée au moins une fois par une fonction,

2. préciser les actions, c'est-à-dire création, recherche, mise à jour et suppression.

Nous avons détaillé le plus possible cette matrice en y faisant figurer toutes les entités utilisées avec toutes les actions. Les traitements (les fonctions élémentaires) sont mis en lignes, les entités en colonnes. Chaque fois qu'une fonction utilise une entité en création, recherche, modification ou suppression, ceci a été noté sur la matrice. Nous avons utilisé les abréviations suivantes : 'C' pour création, 'R' pour recherche, 'MJR' pour mise à jour et 'S' pour suppression.

Pour les entités de type code-libellé en relation avec l'entité principale `entrée`, ayant une cardinalité (1-n) et que nous voulions utiliser en domaine, en liste de valeurs ou pour la validation, il nous a fallu déterminer quelles entités devaient être utilisées en recherche. Le schéma ci-dessous nous a aidé dans notre analyse.



Fonctions/Entités	entrée	tabserie	typ_ent
GAIA1 créer l'entrée			
GAIA10 saisir les attributs obligatoires	C		R
GAIA11 saisir les attributs importants	C		
GAIAT110 vérifier la structure de la cote	R	R	
GAIAT111 construire la clé primaire			
GAIA12 valider l'entrée créée	C		
GAIA2 rechercher une entrée			
GAIA20 saisir les critères de recherche	R		R
GAIA21 valider les critères de recherche			
GAIAT22 calculer le nombre de documents obtenus en résultat			
GAIA23 afficher le nombre de documents obtenus en résultat			
GAIA24 demander confirmation de modification de la recherche			
GAIA25 afficher le sommaire des résultats	R		
GAIA250 valider l'entrée sélectionnée			
GAIA26 afficher l'entrée sélectionnée	R		
GAIA3 modifier une entrée			
GAIA30 rechercher une entrée			
GAIA31 modifier les attributs modifiables	R, MJR, S		
GAIA32 valider la modification			
GAIA4 supprimer une entrée			
GAIA40 rechercher une entrée			
GAIA41 supprimer l'entrée	S		
GAIAT42 mettre à jour les enregistrements associés			
GAIA43 valider la suppression			

Les actions de suppression et de mise à jour qui sont effectuées dans `cotes`, `collection` et `auteurs` suite à l'effacement d'une `entrée`, ne figurent pas dans ce tableau, car elles n'ont pas lieu d'y être. Elles sont considérées comme des propriétés de la relation.

En plus des fonctions de regroupement, certaines autres fonctions ne sont utilisées par aucune entité. Elles figurent dans cette matrice pour des raisons de compréhension, mais nous avons dû les justifier. GAIAT111 est une règle de gestion (voir la partie consacrée à cette question, paragraphe 5.7). GAIA 21, GAIA23, GAIA24, GAIA32, GAIA43 et GAIAT250 font partie des fonctionnalités gérées par les écrans *Forms 4.5*. GAIAT22 est une procédure PL/SQL.

Enfin, il est à noter que l'entité `entrée` est bien utilisée à travers les différents niveaux de hiérarchie des quatre façons possibles : création, recherche, mise à jour et suppression.

5.7 Les règles de gestion

Nous avons classé les règles de gestion selon la classification préconisée par la méthode Oracle et brièvement expliquée dans la partie consacrée à la présentation de *Custom development method*. A ce stade de l'analyse, nous avons recensé toutes les règles et nous les avons brièvement décrites.

Règles de données statiques

La première règle concerne l'entité. Elle porte sur la composition de la clé primaire. Celle-ci se décompose en trois éléments, respectivement l'année (date système), un espace, un numéro généré par la séquence[3] `seqentrée`.

La deuxième règle est une règle sur l'attribut. Elle précise les valeurs des attributs qui servent à la composition de la cote : `kseq1`, `ksérie`, `kseq2`, `kseq3`. Si `kseq1` ou `kseq2` ont une valeur différente de *null*, celle-ci doit être supérieure à zéro. `kseq3` commence par un slash si `kseq2`, de type numérique, est *null*.

La troisième règle concerne les attributs obligatoires. Ceux-ci doivent être connus avant qu'une occurrence d'entité de l'entrée associée puisse exister.

Règles de traitement des données

Trois règles d'effacement sont applicables à la suppression d'une `entrée`. En effet, cette opération entraîne :

1. la suppression de l'`entrée` référencée dans `auteur`,
2. l'initialisation à la valeur *null* de l'attribut `id_entrée` dans `collection`,
3. l'initialisation à la valeur *null* de l'attribut `id_entrée` dans `cotes`.

5.8 Conclusion

Validation

Avant de passer à l'analyse détaillée, nous avons validé cette première phase avec Ernest SOSSAVI, qui est intervenu en tant que représentant des utilisateurs de GAIA. La validation est un pré-requis à tout avancement de l'étude. Son but est de vérifier l'adéquation du modèle avec la réalité et les besoins des utilisateurs.

Saisie des données dans *Designer 2000*

Yasser TAIMA, consultant Oracle avec qui j'ai mené à cette étude, a recommandé de commencer le processus d'analyse directement de *Designer 2000* . A mon avis, il est préférable de réfléchir à l'analyse sur papier dans un premier temps. Les documents papier sont une base de discussion plus facile avec l'utilisateur (ceci est d'autant plus vrai que les rapports du logiciel sont en anglais). De plus, il faut toujours avoir à l'esprit que l'analyse préalable constitue la première ébauche de ce que sera le système. Les allers et venues entre analyse préalable et analyse détaillée sont inévitables et plus ou moins nombreux. Des changements seront donc à effectuer. La saisie des informations représente un temps de travail non négligeable. Faite trop tôt, elle ne présente donc aucun intérêt.

[3] La séquence est un objet de la base qui sert à générer des entiers uniques (pour plus d'informations, se reporter au paragraphe 7.3 de ce document).

6. L'ANALYSE DETAILLEE

L'analyse détaillée se situe dans le prolongement direct de l'analyse préalable. Pour notre étude, elle a été l'occasion de revenir sur les choix faits précédemment, de les affiner et de les valider définitivement avant de passer à la conception. Les entités, les relations, les domaines, les attributs, les fonctions et les règles de gestion ont été examinés et décrits très précisément. Nous avons utilisé trois outils de *Designer 2000* : le modeleur de processus (*process modeller*), le modèle entité-relation (*entity-relationship diagrammer*) et le modèle hiérarchique des fonctions (*function hierarchy diagrammer*).

6.1 Les entités

A ce stade, le choix des entités retenues pour la maquette a été arrêté. Nous avons décrit chaque entité en suivant le formalisme imposé par la méthode.

Chaque entité a reçu trois noms : un nom complet, au singulier et composé de trois mots maximum séparés par des espaces (et non des soulignés), un nom abrégé de trois caractères et unique au sein de l'application *Designer 2000* et un nom au pluriel qui n'est autre que le nom utilisé pour le passage au niveau logique.

Nom court	Nom complet	Nom au pluriel
aut	auteur	auteurs
col	collection	collections
cot	cote	cotes
ete	entrée	entrées
ser	série	séries
te	type entrée	typents

Nous avons complété ces informations par une description et un commentaire sur le cycle de vie de l'entité (comment, pourquoi, dans quelles circonstances elle est créée, recherchée, modifiée et supprimée).

Entité	Description	Cycle de vie
auteur	L'auteur est la personne, morale ou physique, qui a créé le document.	L'auteur est créé par rapport aux documents. Il peut donc être supprimé lorsqu'il ne référence plus aucun document.
collection	La collection peut être : - un ensemble cohérent de documents par rapport à un thème ou une orientation, - soit une collection au sens éditorial du terme.	La collection est créée par rapport aux documents. Elle peut donc être supprimée lorsqu'elle ne référence plus aucun document.
cote	La cote est l'identifiant d'une unité documentaire, qui peut correspondre à une boîte, une monographie, une liasse, un dossier ou une feuille. Elle répond à des besoins de classement : elle permet de retrouver les documents.	La cote peut être créée par rapport aux documents enregistrés dans GAIA. Elle peut donc être supprimée lorsqu'elle ne référence plus aucun document.
entrée	L'entrée est le registre chronologique des documents versés définitivement aux archives.	L'entrée est créée, modifiée et supprimée à travers les écrans de gestion du registre d'entrée.

Entité	Description	Cycle de vie
série	La série est une classification utilisée pour déterminer la période et le type des documents.	La série est créée par rapport aux documents. Elle peut donc être supprimée lorsqu'elle ne référence plus aucun document.
type entrée	Le type d'entrée représente la façon dont le document a été reçu aux Archives départementales.	Le type d'entrée est créé par rapport aux documents. Il peut donc être supprimé lorsqu'il ne référence plus aucun document.

De plus, l'entité entrée a reçu une note technique décrivant les règles relatives à quatre de ses attributs :

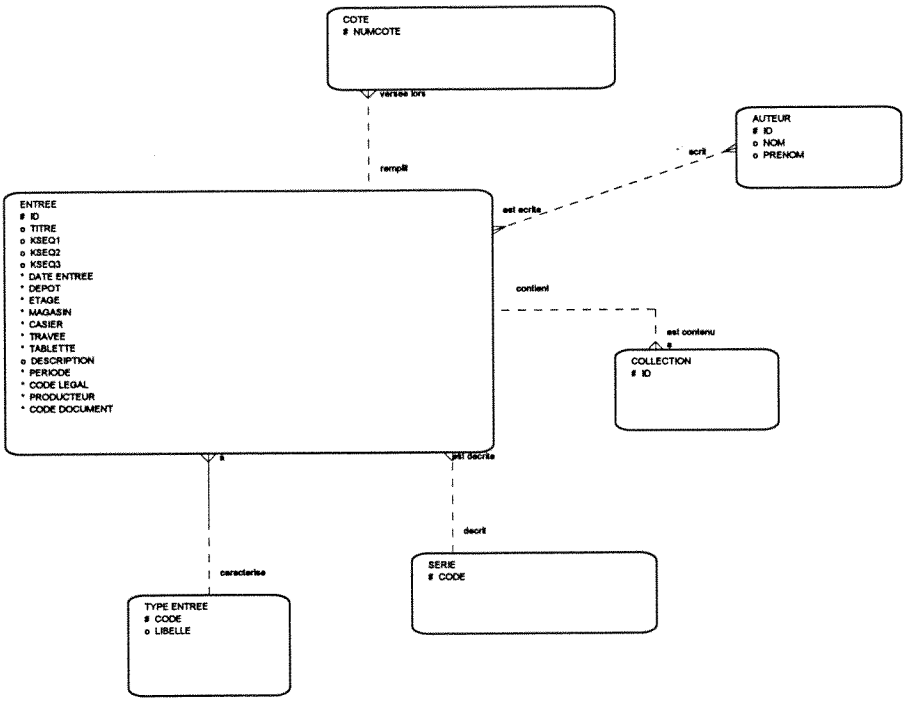
ATR001 : si kseq1 n'est pas *null*, il doit être supérieur à 0. Si kseq2 n'est pas *null*, il doit être supérieur à 0.
Révision 1 du 2/8/96
ATR002 : si kseq2 est *null*, kseq3 doit commencer par un slash et être modifiable.

ATR003 : ksérie et kseq3 ne doivent pas contenir d'espace.
ATR004 : si kseq3 est renseigné, kseq3 ne peut débuter par un chiffre.
ATR005 : si kseq3 est renseigné, il doit se terminer par un chiffre.
ATR006 : si kérie est *not null*, alors au moins un des trois autres attributs, kseq1, kseq2 ou kseq3 sera aussi *not null*.

La structure de la maquette est relativement simple, elle ne comporte ni arc exclusif, ni sous-type. Nous n'avons donc pas eu de réflexion à mener sur ce sujet en vue de leur élimination.

6.2 Les relations

Nouveau schéma validé :



Les relations sont les suivantes :

Un auteur peut écrire une ou plusieurs entrées.
Une collection peut être contenue dans une et une seule entrée.
Une cote peut être versée lors d'une et une seule entrée.
Une entrée :
- peut contenir une ou plusieurs collections,
- peut remplir une ou plusieurs cotes,
- peut être décrite par une et une seule série,
- doit avoir un et un seul type d'entrée
- peut être est écrite par un ou plusieurs auteurs.

Une série peut être décrite par une ou plusieurs entrées.
Un type d'entrée peut caractériser une ou plusieurs entrées.

Une fois les relations définies, nous les avons décrites. *Designer 2000* propose deux façons de procéder : soit une description dans laquelle sont notifiées toutes les règles de supression et de modification, soit les notes dans lesquelles sont commentées toutes les relations incorrectes, les conditions particulières qui s'appliquent aux relations.

Entrée-Auteur	**Description** : DEL001[1] : la suppression d'un `auteur` entraîne la suppression de l'`auteur` associé. **Notes** : pour permettre les recherches sur l'`auteur`, il faut parfois enregistrer le nom de l'auteur avant même que les détails de l'`entrée` soient enregistrés.
Entrée-Collection	**Description** : DEL002 : la suppression d'une `entrée` entraîne l'initialisation à la valeur *null* de la `collection` associée. **Notes** : il existe des `collections` qui n'appartiennent à aucune `entrée`. Il faut parfois créer la `collection` sans attendre la création de l'`entrée`.
Entrée-Cote	**Description** : DEL003 : la suppression d'une `entrée` entraîne l'initialisation à la valeur *null* de la `cote` associée. **Notes** : il existe des `cotes` qui n'appartiennent à aucune `entrée` : il faut pouvoir créer la `cote` sans attendre la création de l'`entrée`.
Entrée-Série	**Notes** : La `série` est le premier élément de la `cote`, mais cette relation n'est pas représentée dans ce modèle cependant valide pour la maquette. Une `série` peut exister indépendamment d'une `entrée` pour les mêmes raisons que dans le cas de la cote.
Entrée-Type entrée	**Notes** : Consultation : affichage en liste de valeurs.

6.3 Les domaines

6.3.1 Les domaines de valeurs et de formatage

Les domaines de valeurs dont la création a été décidée lors de l'analyse préalable, ont été créés et paramétrés dans *Designer 2000*. Ils ont été complétés par des domaines de formatage, c'est-à-dire un format et une longueur maximale communs à plusieurs attributs. Ceci nous a dispensé du paramétrage de la définition des attributs dont le format et la longueur étaient identiques. En rattachant un attribut à un domaine, les paramètres du format sont directement inscrits dans la définition de l'attribut.

Pour chaque domaine de valeurs, nous avons défini un nom au pluriel, un format identique à celui des attributs qui allaient appartenir au domaine, une longueur maximum pour les attributs du domaine, un type de données, une longueur de colonne et une description.

[1] La codification de cette règle de gestion et des suivantes, DEL002 et DEL003, est expliquée au paragraphe 6.6 relatif aux règles de gestion.

Nom	Format	Long. Attribut	Type de données	Long. Colonne	Description
cat producteurs	char	3	varchar2	3	Catégorie du producteur.
dépôts	char	2	varchar2	2	Dépôt où est stocké le document.
périodes	integer	7	number	7	Période de production du document versé.
type séries	char	3	varchar2	3	Code du type de la série.
type traitement	char	4	varchar2	4	Type de traitement qui détermine la description du document.
id num 4	integer	4	number	4	Identifiant sur 4 chiffres.
id num 7	integer	7	number	7	Identifiant sur 7 chiffres.
titre 255	char	255	varchar2	255	Titre sur 255 caractères.
desc 2000	text	2000	varchar2	2000	Description sur 2 000 caractères.

Pour les domains de valeurs, nous avons déterminé l'ordre d'affichage des valeurs dans la liste ainsi que leur contenu :

Nom	Séquence	Valeurs
Cat producteurs	2	ED
	4	EVE
	6	HOP
	8	NOT
	10	DAC
	12	PAD
Dépôts	2	PF
	4	AD
	6	DAC
	8	PAD
Périodes	2	23
	4	1
	6	2
	8	3
	10	4
	12	27
	14	28
	16	29
Type séries	2	A
	4	BI
	6	FI
	8	EX
Type traitements	2	A
	4	BI
	6	FI
	8	PR

6.3.2 Les valeurs statiques et dynamiques

La définition des domaines ne s'arrête pas là. Il faut aussi réfélchir à leur configuration en fonction de la fréquence de mise à jour des valeurs. Si elles sont peu sujettes mise à jour, elles sont considérées comme des valeurs statiques. Elles peuvent donc être mises en dur dans le code de l'écran. Toute mise à jour entraîne une régénération de tous les écrans concernés. L'avantage réside essentiellement dans les temps de réponse aux requêtes (dans *Designer 2000*, le paramètre *softlov* doit être égal à 'oui').

Si les valeurs sont susceptibles de changer de temps en temps, elles sont à configurer en valeurs dynamiques. Leur modification fait uniquement au niveau du serveur. Toute mise à jour entraîne une régénération de leur définition afin que cette dernière soit répercutée à tous les attributs concernés (le paramètre *softlov* doit être à égal à 'non').

Nous avons paramétré les valeurs des domaines `cat producteurs`, `dépôts`, `périodes`, `type séries` et `type traitements` en valeurs statiques, car nous n'avons envisagé aucune mise à jour dans le cadre de la maquette.

6.4 Les attributs

Les attributs définitivement retenus ont été nommés en suivant quelques principes, en particulier pour le nom. Celui-ci est au singulier, composé de trois mots maximum et limité à vingt-deux caractères. Les différents éléments d'un attribut composé sont séparés par des espaces. Le même attribut ne doit pas être utilisé pour une entité mère et l'une de ses filles, ni contenir de référence au nom de l'entité qu'il qualifie.

Entité	Attribut	Domaine
entrée	cat producteur	cat producteurs
	code document	type traitements
	code légal	type séries
	dépôt	dépôts
	id	id num 4
	période	périodes
	titre	titre 255
	description	desc 2000
collection	id	id num 7

Enfin, nous avons complété la description des attributs :

Entité	Attribut	Séquence	Optionel	Format	Longueur	Identifiant unique
auteur	id	1	non	integer	7	oui
	nom		oui	char	100	non
collection	id	1	non	integer	7	oui
cote	id	1	non	integer	8	oui
entrée	id	2	non	integer	4	oui
	année	4	non	date		non
	titre	6	oui	char	255	non
	ksqe1	10	oui	number	5	non
	kseq2	12	oui	number	5	non
	kseq3	14	oui	char	31	non
	date entrée	16	non	date		non

Entité	Attribut	Séquence	Optionel	Format	Longueur	Identifiant unique
	dépôt	18	non	char	2	non
	étage	20	non	char	2	non
	magasin	22	non	char	1	non
	casier	24	non	char	4	non
	travée	26	non	char	1	non
	tablette	28	non	char	2	non
entrée (suite)	description	30	oui	text	2000	non
	période	32	non	integer	7	non
	code légal	34	non	char	3	non
	producteur	36	non	char	3	non
	code document	38	non	char	4	non
série	code	1	non	char	7	oui
type entrée	code	2	non	char	3	oui
	libellé	4	oui	char	60	non

6.5 Les fonctions

6.5.1 La hiérarchie des fonctions

Nous avons poursuivi notre réflexion sur la hiérarchie des fonctions élaborée précédemment. Nous avons procédé aux changements suivants :

Fonctions aboutissant à la génération d'écrans :

GAIAE écrans de la gestion du registre des entrées
GAIA1 créer l'entrée
GAIA10 saisir les attributs obligatoires
GAIA11 saisir les attributs importants
~~GAIA12 valider l'entrée créée~~

La fonction de validation de l'entrée disparaît, puisque la validation se fait au *commit*[2].

GAIA2 rechercher une entrée
GAIA20 saisir les critères de recherche
~~GAIA21 valider les critères de recherche~~
GAIA23 afficher le nombre de documents obtenus en résultat, demander confirmation pour modifier la recherche
~~GAIA24 demander confirmation de modification de la recherche~~
~~GAIA25 afficher le sommaire des résultats~~
~~GAIA250 valider l'entrée sélectionnée~~
GAIA26 afficher l'entrée sélectionnée

Les fonctions d'affichage du nombre de documents et de demande de confirmation de la modification ont été groupées en une seule fonction GAIA23. GAIA21 disparaît car elle n'est pas considérée comme une fonction. En raison des contraintes de temps, GAIA25 et GAIA250 ont été éliminées.

[2] Le *commit* est l'opération qui consiste à enregistrer une insertion, une modification ou une suppression dans la base de données.

GAIA3 modifier une entrée
GAIA30 rechercher une entrée
GAIA31 modifier les attributs modifiables
GAIA32 valider la modification
GAIA4 supprimer une entrée
GAIA40 rechercher une entrée
GAIA41 supprimer l'entrée
GAIA43 valider la suppression

Aucun changement n'a été apporté.

Fonctions se traduisant par des fonctions de gestion :

GAIAT règles de gestion du registre des entrées
GAIAT1 traitements de gestion de la création des entrées
GAIAT110 vérifier la structure de la cote
~~GAIAT111 construire la clé primaire~~

Nous avons supprimé la fonction de construction de la clé primaire car celle-ci était incorrecte. Cette dernière était contituée de deux éléments, la séquence `seqentrée` séparée par un espace du millésime sur quatre chiffres. Ceci aboutissait à deux valeurs et, de ce fait, au non respect de la première forme normale[3]. En effet, celle-ci exige que l'intersection d'une ligne et d'une colonne ne contienne qu'une seule et unique valeur. Aussi, avons nous préféré éclater la clé primaire en deux attributs, la séquence et l'année. La clé primaire est donc devenue la concaténation de la séquence et de l'année. L'espace a été restauré uniquement à l'affichage pour ne pas perturber l'utilisateur. Cette modification a rétabli l'entité en première forme normale.

GAIAT2 traitements de gestion des écrans de recherche d'une entrée
GAIAT22 calculer le nombre de documents obtenus en résultat
~~GAIAT42 mettre à jour les enregistrements associés~~
GAIAT3 traitements de gestion de la modification d'une entrée

Nous avons fait disparaître la mise à jour des enregistrements associés car ce n'est pas une fonction mais une règle de gestion.

Schéma définitif de la hiérarchie des fonctions :

GAIAE écrans de la gestion du registre des entrées
GAIA1 créer l'entrée
GAIA10 saisir les attributs obligatoires
GAIA11 saisir les attributs importants
GAIA2 rechercher une entrée
GAIA20 saisir les critères de recherche
GAIA23 afficher le nombre de documents obtenus en résultat,
demander confirmation de modification de la recherche
GAIA26 afficher l'entrée sélectionnée
GAIA3 modifier une entrée
GAIA30 rechercher une entrée
GAIA31 modifier les attributs modifiables
GAIA32 valider la modification

[3] La normalisation est un processus qui permet de s'assurer qu'un modèle logique de données est 'relationnel' et que ses données ne sont pas redondantes. Les formes normales font partie du cadre formel utilisé pour effecteur cette normalisation.

GAIA4 supprimer une entrée
GAIA40 rechercher une entrée
GAIA41 supprimer l'entrée
GAIA43 valider la suppression
GAIAT règles de gestion du registre des entrées
GAIAT1 traitements de gestion de la création des entrées
GAIAT110 vérifier la structure de la cote
GAIAT2 traitements de gestion des écrans de recherche d'une entrée
GAIAT22 calculer le nombre de documents obtenus en résultat
GAIAT3 traitements de gestion de la modification d'une entrée

6.5.2 La description des fonctions

La description des fonctions diffère selon qu'il s'agit d'une fonction de type *immediate,* ou de type *overnight.* Elle comprend plus ou moins d'éléments.

Les fonctions de type *immediate*

Elle peut aller jusqu'à quatre parties qui sont : le but de la fonction, sa description, des exemples d'utilisation typique et des remarques diverses. Faute de temps, nous nous sommes limités à la partie 'description'.

Fonction	Description
GAIA10	Saisir les attributs obligatoires. *Commit* (ou éventuellement *Cancel*) et revenir au menu initial.
GAIA11	Saisir les attributs importants : le `titre`, la `description` et les quatre éléments de la cote : `kseq1`, `ksérie`, `kseq2` et `kseq3`.
GAIA20	Même écran que celui de la saisie. Les attributs `période` et `description` ne sont pas interrogeables.
GAIA23	Afficher le nombre de documents obtenus en résultat ; demander confirmation.
GAIA26	Ecran semblable à GAIA10 et GAIA11, mais la mise à jour est impossible.
GAIA30	Même écran que celui de la saisie. Les attributs `période` et `description` ne sont pas interrogeables (idem GAIA20).
GAIA31	Même écran que celui de la saisie; les attributs `id`, `année`, `code légal`, `cat producteur`, `code document` ne peuvent pas être modifiés.
GAIA32	L'utilisateur décide s'il confirme ou non sa modification . Le message suivant s'affiche à l'écran : «Etes-vous sûr de vouloir enregistrer ces modifications ? » S'il répond oui, la modification est validée, s'il répond non, il revient à l'écran de modifications GAIA31.
GAIA40	Même écran que celui de la saisie. Les attributs `période` et `description` ne sont pas interrogeables (idem GAIA20).
GAIA41	Supprimer l'`entrée`.
GAIA43	L'utilisateur décide s'il confirme ou non sa suppression . Le message suivant s'affiche à l'écran : «Voulez-vous confirmer cette suppression ? ». S'il répond oui, la suppression est validée, s'il répond non, il revient à l'écran de suppression GAIA41.

La description des fonctions de type *overnight*

Elle peut préciser jusqu'à cinq éléments : le but de la fonction, les paramètres en entrée, les paramètres en sortie, le traitement et autres remarques. Nous avons reporté toutes les modifications apportées suite à l'avancement de l'analyse afin d'avoir un historique complet des mises à jour.

Fonction	Description/Notes
GAIAT110	**Description** : CRE001 : la `cote` est constituée de quatre attributs[4] (`kseq1`, `ksérie`, `kseq2` et `kseq3`). Le type de `ksérie` et `kseq3` est CHAR. Le type de `kseq1` et `kseq2` est INTEGER (5). Si `kseq2` est égal à NULL, `kseq3` doit commencer par un slash. Si `kseq1` et `kseq2` sont différents de *null*, ils doivent être supérieur à 0. PAR IN : `kseq1` - 3. PAR OUT : booléen, `kseq3`. **Notes** : Révision 1.0 du 2/8/96 : CRE001 devient ATR001 puisque c'est une règle de type *Other attribute*. Révision 1.1 du 2/8/96 : implémenter la règle dans une *check constraint* sur la table `proto entrée`, donc ce traitement disparaît. ATR001 : `kseq1` > 0 ou null ; `kseq2` > 0 ou *null*. ATR002 : si `kseq2` est *null*, `kseq3` commence par un slash. Révision 2 du 2/9/96 : ATR003 : `ksérie` et `kseq3` ne doivent pas contenir d'espace. ATR004 : si `kseq3` est renseigné, `kseq3` ne peut débuter par un chiffre. ATR005 : si `kseq3` est renseigné, il doit se terminer par un chiffre. ATR006 : si `ksérie` est *not null*, alors au moins un des trois autres attributs, `kseq1`, `kseq2` ou `kseq3` sera aussi *not null*. A implémenter en *check constraint* de la table `proto_entrées`.
GAIAT22	**Description** : Cette fonction sur `proto entrée` calcule le nombre de lignes résultant de la requête avec les critères passés en paramètre. PAR IN : critères de recherche. PAR OUT : nombre de résultats.

6.5.3 La matrice fonctions/entités

Cette matrice tient compte des modifications apportées à la hiérarchie des fonctions et évoquées au paragraphe 6.5.1. Nous avons retiré les entités transformées en domaines et les fonctions qui n'ont plus lieu d'être. Les abréviations utilisées sont les suivantes : 'C' pour la création, 'R' pour la recherche, 'MJR' pour la mise à jour et 'S' pour la suppression.

[4] La cote est l'élément d'identification à la fois physique et intellectuelle du document. Parmi les quatre éléments énoncés ci -dessus, seule `ksérie` a une signification précise puisqu'elle renvoie à la série. `kseq1`, `kseq2` et `kseq3` sont des éléments dont les notions varient en fonction de paramètres tels que la série, le type de document, etc.

Fonctions/Entités	entrée	série	type entrée
GAIA1 créer l'entrée			
GAIA10 saisir les attributs obligatoires	C		R
GAIA11 saisir les attributs importants	C		
GAIAT110 vérifier la structure de la cote	R	R	
GAIA2 rechercher une entrée			
GAIA20 saisir les critères de recherche	R		R
GAIAT22 calculer le nombre de documents obtenus en résultat			
GAIA23 afficher le nombre de documents obtenus en résultat, demander confirmation de modification de la recherche			
GAIA26 afficher l'entrée sélectionnée	R		
GAIA3 modifier une entrée			
GAIA30 rechercher une entrée			
GAIA31 modifier les attributs modifiables	R, MJR, S		R
GAIA32 valider la modification			
GAIA4 supprimer une entrée			
GAIA40 rechercher une entrée			
GAIA41 supprimer l'entrée	S		
GAIA43 valider la suppression			

6.5.4 La matrice fonctions/attributs

Dans cette matrice, nous avons fait figurer tous les attributs utilisés par au moins une fonction. Les abvréviations sont les mêmes que celles employées pour la matrice précédente.

Fonctions/ Entités	date entrée	description	kseq1	kseq2	id	kseq3	dépôt	étage	magasin	casier
GAIA1 créer l'entrée										
GAIA10 saisir les attributs obligatoires	C				C		C	C	C	C
GAIA11 saisir les attributs importants		C	C	C		C				
GAIAT110 vérifier la structure de la cote										
GAIA2 rechercher une entrée										
GAIA20 saisir les critères de recherche	R		R	R	R	R	R	R	R	R

Fonctions/ Entités	date entrée	description	kseq1	kseq2	id	kseq3	dépôt	étage	magasin	casier
GAIAT22 calculer le nombre de documents obtenus en résultat										
GAIA23 afficher le nombre de documents obtenus en résultat, demander confirmation de modification de la recherche										
GAIA26 afficher l'entrée sélectionnée	R		R	R	R	R	R	R	R	R
GAIA3 modifier une entrée										
GAIA30 rechercher une entré										
GAIA31 modifier les attributs modifiables	MJR	MJR, S	MJR, S	MJR, S	R[5]	MJR, S	MJR	MJR	MJR	MJR
GAIA32 valider la modification										
GAIA4 supprimer une entrée										
GAIA40 rechercher une entrée										
GAIA41 supprimer l'entrée										
GAIA43 valider la suppression										

[5] L'attribut `id` est utilisé en recherche car l'écran de recherche le passe en paramètre à l'écran de modification.

Fonctions/ Entités	travée	tablette	titre	période	code légal	cat produc-teur	code docu-ment	année	type entrée. code	type entrée. libellé
GAIA1 créer l'entrée										
GAIA10 saisir les attributs obligatoires	C	C		C	C	C	C		R	R
GAIA11 saisir les attributs importants			C							
GAIAT110 vérifier la structure de la cote										
GAIA2 rechercher une entrée										
GAIA20 saisir les critères de recherche	R	R	R		R	R	R			
GAIAT22 calculer le nombre de documents obtenus en résultat										
GAIA23 afficher le nombre de documents obtenus en résultat, demander confirmation de modification de la recherche										
GAIA26 afficher l'entrée sélectionnée	R	R	R	R	R	R	R	R		
GAIA3 modifier une entrée										
GAIA30 rechercher une entrée										

Fonctions/ Entités	travée	tablette	titre	période	code légal	cat produc-teur	code docu-ment	année	type entrée. code	type entrée. libellé
GAIA31 modifier les attributs modifiables	MJR	MJR	MJR, S	MJR			R		R	R
GAIA32 valider la modification										
GAIA4 supprimer une entrée										
GAIA40 rechercher une entrée										
GAIA41 supprimer l'entrée										
GAIA43 valider la suppression										

6.6 Les règles de gestion

Nous avons revu toutes les règles de gestion définies lors de l'analyse préalable. Certaines ont disparu, d'autres au contraire sont apparues.

Nous avons supprimé la règle de gestion définie sur l'identifiant unique dans l'analyse préalable puisque sa composition a changé (cf paragraphe 6.5.1). Nous avons éliminé la règle concernant la vérification de la cote de façon à l'implémenter le plus simplement et aussi le plus efficacement possible, c'est-à-dire en *check constraint*[6] au niveau logique. Nous avons également complété les informations relatives à sa structure, à savoir : `ksérie` et `kseq3` ne peuvent pas contenir de blanc, si `kseq3` est renseigné, `kseq3` ne peut commencer par un numérique, mais doit finir par un chiffre. Si `ksérie` a une valeur, alors au moins l'un des trois autres attributs `kseq1`, `kseq2` ou `kseq3` doit aussi être renseigné. Nous avons donc abouti à six règles :

ATR001 : `kseq1` > 0 ou null ; `kseq2` > 0 ou *null*.
ATR002 : si `kseq2` est *null*, `kseq3` commence par un slash.
ATR003 : `ksérie` et `kseq3` ne doivent pas contenir d'espace.
ATR004 : si `kseq3` est renseigné, `kseq3` ne peut débuter par un chiffre.
ATR005 : si `kseq3` est renseigné, il doit se terminer par un chiffre.
ATR006 : si `ksérie` est *not null*, alors au moins un des trois autres attributs, `kseq1`, `kseq2` ou `kseq3` sera aussi *not null*.

[6] Une *check constraint* est une contrainte associée à une condition. Celle-ci prévient contre toue insertion, modification ou suppresion qui irait à l'encontre des régles de gestion définies. Pour plus de précisions, se reporter à la partie 7.6 consacrée aux contraintes.

Les règles de mise à jour des entités associées lors de la suppression d'une `entrée` demeurrent. Nous les avons classées selon les spécifications de la méthode Oracle :

Signification	Classe	Sous classe	Règle	Enregistrement dans Des 2000	Accès	Nom de la règle
la suppression d'une `entrée` entraîne la suppression de l'`entrée` référencée dans `auteur`	Data operation rules	delete rules, action = cascade	relationship	relationship description	ERD	DEL001
la suppression d'une `entrée` entraîne la mise à jour de l'`entrée` référencée dans la `collection`	Data operation rules	delete rules, action = nullify	relationship	relationship description	ERD	DEL002
la `suppression` d'une `entrée` entraîne la mise à jour de l'`entrée` référencée dans `cote`	Data operation rules	delete rules, action = nullify	relationship	relationship description	ERD	DEL003

Pour le nom des règles, nous avons retenu le principe de nommage des règles de gestion enregistrées dans une fonction. Oracle préconise en effet un nom sur trois caractères, indiquant le type de règle, concaténé à une séquence de quatre chiffres. La première partie du nom est utilisée comme suit :

ENT règles concernant les autres entités
RER règles sur les relations restrictives
IER règles sur les autres relations entre entités
CRE règles sur la création
UDP règles sur la mise à jour
DEL règles sur les autres suppressions
CEV règles sur les événements survenant suite à un changement d'état.

6.7 Conclusion

L'analyse détaillée marque le fin de la modélisation conceptuelle. Au cours, de cette phase, nous n'avons pas hésité à revenir sur les choix faits lors de l'analyse préalable dès que l'un d'eux nous a semblé incorrect. Il était primordial d'apporter toutes les corrections nécessaires avant de passer à la phase de conception. En effet, ainsi que le souligne M. SMIME[7] dans son ouvrage, page 67, 'la modélisation conceptuelle des données est un processus crucial pour la réussite du projet : une mauvaise modélisation génère un faux SCD[8], qui donne une structure fausse de la base de données. Les programmes seront faux, et là il faudra remodéliser conceptuellement !'.

Avant de commencer la conception, nous avons validé avec M. Ernest SOSSAVI, représentant des utilisateurs de GAIA, les résultats de cette deuxième étude.

[7] SMIME H., *Concevoir et développer avec ORACLE et CASE : Oracle versions 6 et 7*. Paris : Eyrolles, 1994. 268 p.

[8] Le schéma conceptuel de données ou SCD est l'équivalent du modèle conceptuel de données.

7. LA CONCEPTION

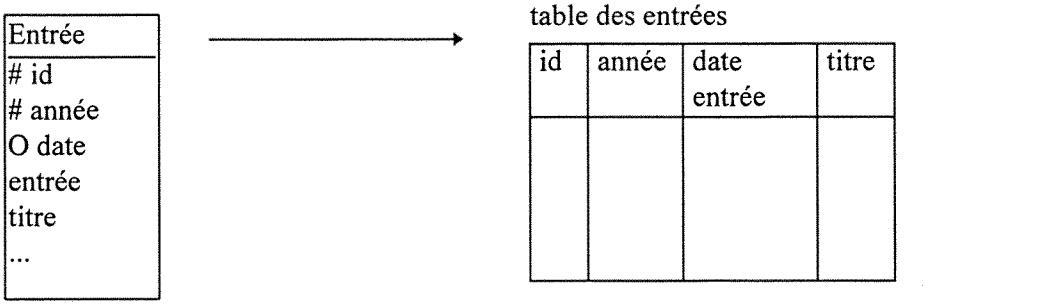
La phase de conception comprend plusieurs étapes : la conception logique de la base (*database logical design*), l'implémentation des règles de gestion et la génération des écrans. La synthèse qui suit ne rend compte que de la première étape car les suivantes sont encore en cours. Durant cette phase, nous avons créé la base de données et paramétré ses différents objets que sont les tables, les séquences, les colonnes et les contraintes.

7.1 La création de la base de données

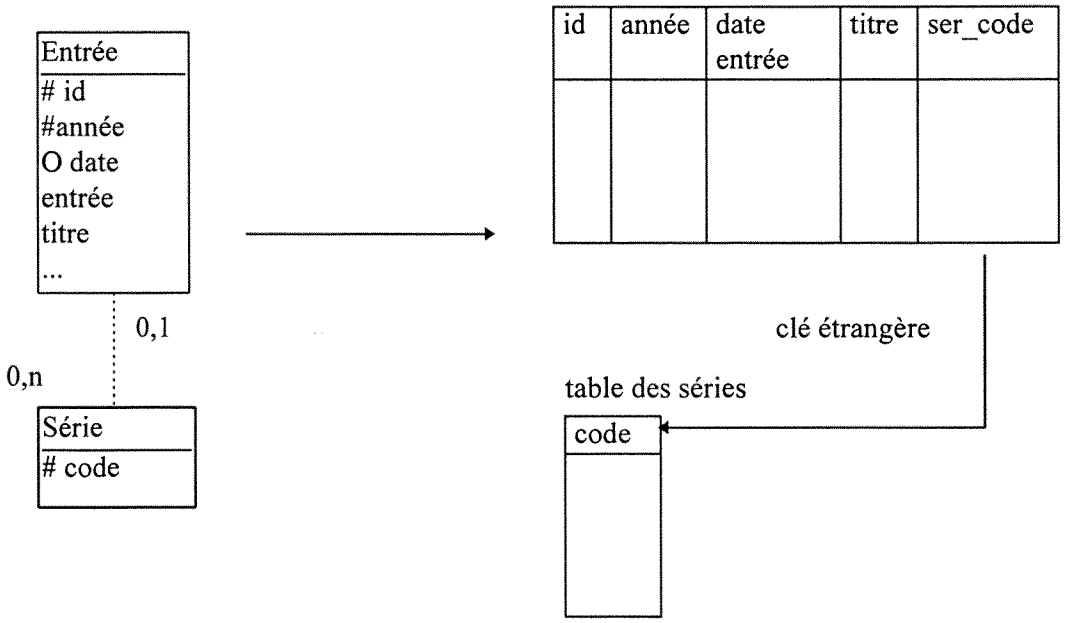
Le passage du niveau conceptuel au niveau logique se fait par la création de la base de données. Pour cela, nous avons utilisé l'assistant conception base de données de *Designer 2000*, le *Database design wizard*. Ce dernier crée et maintient les conceptions de bases de données basées sur le modèle entité-relation présent dans le dictionnaire conceptuel. Il crée les tables pour enregistrer les instances de chaque entité, les colonnes pour stocker les attributs, les contraintes pour implémenter les relations entre les entités et renforcer les identifiants uniques, les index pour supporter les clés primaires et étrangères.

Les schémas suivants permettent de bien comprendre le passage du conceptuel au logique pour les différents objets de la maquette :

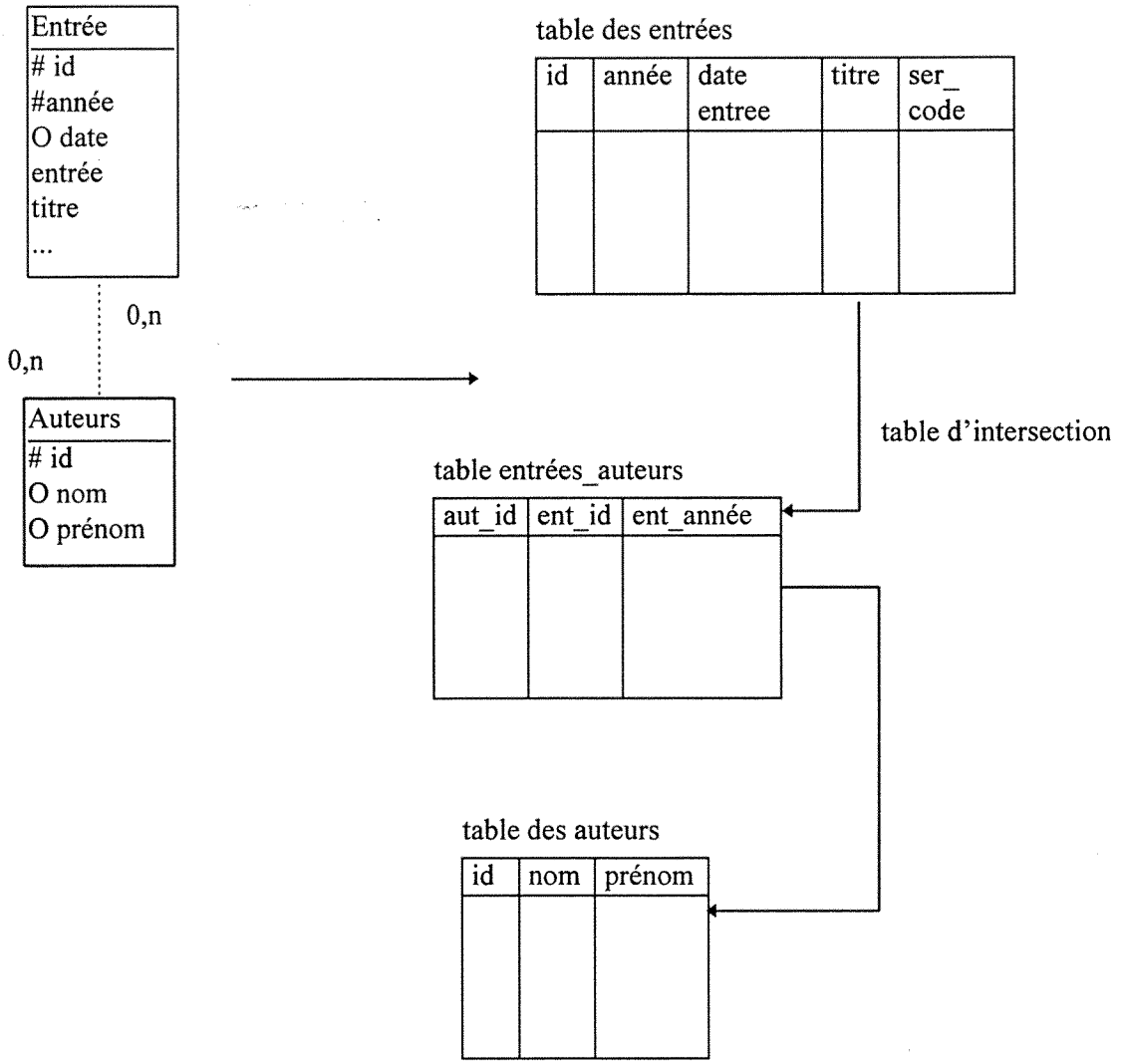
Les entités et les attributs



Les relations



signifie que l'attribut est un identifiant unique, O que l'attribut est obligatoire.



Pour générer les tables de la maquette, nous sommes partis de l'application GAIA2, résultante du *reverse engineering*[1], qui comprend toutes les entités de GAIA. Nous avons sélectionné les entités appartenant au domaine fonctionnel des entrées (onglet *table mappings*), soit `entrée`, `cote`, `auteur`, `collection`, `série` et `type entrée`. Nous avons repris tels quels les attributs se rapportant à ces entités (onglet *column mappings*) puisque leur sélection et leur définition avaient été validées à la fin de l'analyse détaillée. Les clés primaires ont été générées sur la base des attributs définis comme identifiants uniques au niveau conceptuel.

[1] Le *reverse engineering* est la technique utilisée pour remonter, à partir d'un script de création de base de données, du niveau logique au niveau conceptuel.

Les clés étrangères pour les attributs `te_code` (code du type d'entrée) et `ser_code` (code de la série) ont été créées.

Colonne	Colonne dérivée de	Clé étrangère
`te_code`	`typents.code`	`ent_te_fk`
`ser_code`	`series.code`	`ent_ser_fk`

Nous n'avons prévu aucun index en dehors de ceux générés automatiquement pour les clés primaires, le but de la maquette n'étant pas un prototype de recherche.

Nous avons décidé de ne pas enregistrer par un *commit* le tout premier modèle logique généré en standard. Nous savions que nous voudrions l'adapter avant de l'inscrire dans le dictionnaire conceptuel. En procédant de cette façon, les modifications successives que nous avons apportées, ont été plus faciles à gérer. L'enregistrement a été effectué une fois tous les paramétrages des tables, colonnes (définition et affichage), validations et contraintes terminés.

7.2 Les tables

Le paramétrage des tables a été relativement simple et rapide. Outre le rattachement des tables à l'application GAIA2, il a consisté à définir, pour chacune d'elles, le titre à afficher à l'écran, un alias et une courte description relative à son objet. Le nom des tables a été préfixé par 'prot_' pour bien démarquer celles-ci des autres tables de l'application GAIA2 issues du *reverse engineering*.

Table	Alias	Titre d'affichage	Description
`prot_typents`	TE	Types d'entrées	Décrit les différents types d'entrée.
`prot_entrées`	ENT	Entrées	Représente une entrée de documents à archiver aux Archives départementales.
`prot_auteurs`	AUT	Auteurs	Personne qui a créé les documents.
`prot_séries`	SER	Séries	Définit la série qui permet de situer les documents dans leur contexte historique.
`prot_cotes`	COT	Cotes	Elément de description intellectuelle et matérielle qui permet de retrouver les documents.
`prot_ collections`	COL	Collections	Représente soit un ensemble de documents liés par rapport à un thème ou une orientation, soit une collection au sens éditorial du terme.
`prot_auteurs_ entrées`	AUT_ENT	Auteurs_Entrées	Tables d'intersection entre `auteurs` et `entrées`.

7.3 Les séquences

Les séquences sont des objets de la base de données. Ils servent à générer des entiers uniques en environnement multi-utilisateurs sans conflit ni risque de verrous mortels. Ils sont le plus souvent utilisés pour l'incrémentation automatique des clés primaires.

Nous avons créé deux séquences. Leur pas a été établi à un. Chaque séquence a reçu un nom composé de deux préfixes, le nom du prototype suivi du nom de la table, et de 'seq' : `proto_entrées_seq` et `proto_auteurs_seq`. Les valeurs maximales ont été définies à 9999.

Designer 2000 permet d'implémenter les séquences de deux façons : au niveau du serveur ou au niveau d'une table de contrôle. Dans le premier cas, la valeur d'une séquence augmente d'une unité chaque fois que l'on fait référence à la pseudo-colonne *nextval*. Cet événement peut se produire plusieurs fois avant la création de la table. Les valeurs peuvent donc ne pas être consécutives. Dans le

second cas, les séquences sont stockées dans une table de contrôle de code. Les valeurs sont alors consécutives. Cette solution ne peut convenir que si le nombre d'accès simultané est peu élevé. Nous avons choisi de les implémenter au niveau serveur. Le fait que les valeurs puissent ne pas être consécutives, n'avait aucune importance pour notre application. Le numéro généré par la clé primaire n'est pas un numéro d'ordre, mais un numéro d'identification.

7.4 Les colonnes

7.4.1 La définition

La définition des colonnes implique le renseignement de quelque vingt paramètres. Nous n'avons renseigné que la partie utile à notre maquette. Nous avons également rattaché les colonnes faisant l'objet d'une séquence Oracle à leur séquence.

Le nom

Chaque colonne a reçu un nom unique, de moins de quatre-vingt caractères, sans espace (ceux-ci sont remplacés par des soulignés).

Les séquences

L'ordre des colonnes dans les tables a été défini avec un pas de deux.

La longueur maximale

Celle-ci représente la longueur maximale des données que les colonnes peuvent contenir. Ce paramètre n'est pas valable pour les champs de type date.

Tableaux récapitulatifs des définitions des colonnes

table des auteurs

Colonne	Séquence	Obligatoire	Type de données	Longueur max.	Séquence Oracle
id	1	oui	number	7	proto_auteurs_seq
nom	2	oui	varchar2	100	
prénom	3	oui	varchar2	50	

table des auteurs_entrées

Colonne	Séquence	Obligatoire	Type de données	Longueur max.
aut_id	1	oui	number	7
ent_id	2	oui	number	4
ent_annee	3	oui	date	

table des collections

Colonne	Séquence	Obligatoire	Type de données	Longueur max.
id	1	oui	number	7
ent_id	4	oui	number	4
ent_annee	5	oui	date	

table des cotes

Colonne	Séquence	Obligatoire	Type de données	Longueur max.
id	1	oui	number	8
ent_id	4	oui	number	4
ent_année	5	oui	date	

table des entrées

Colonne	Séquence	Obligatoire	Domaine	Type de données	Longueur max.	Séquence Oracle
id	1	oui	id num 4	number	4	prot_entrées_seq
date	2	oui		date	2	
titre	4	non		varchar2	50	
te_code	8	oui		varchar2	3	
kseq1	10	non		number	6	
ser_doc	12	non		varchar2	7	
kseq2	14	non		number	6	
kseq3	16	non		varchar2	20	
date_entree	22	oui		date	10	
dépôt	27	oui	dépôts	varchar2	6	
étage	28	oui		varchar2	2	
magasin	29	oui		varchar2	1	
casier	30	oui		varchar2	4	
travée	31	oui		varchar2	1	
tablette	32	oui		varchar2	2	
description	34	non		long	50	
période	37	oui	périodes	number	8	
code_légal	41	oui	type séries	varchar2	7	
cat_producteur	43	oui	cat producteurs	varchar2	7	
code_document	44	oui		varchar2	4	

table des séries

Colonne	Séquence	Obligatoire	Type de données	Longueur max.
code	1	oui	varchar2	7

table des types d'entrées

Colonne	Séquence	Obligatoire	Type de données	Longueur max.
code	1	oui	varchar2	3
libelle	2	oui	varchar2	60

7.4.2 L'affichage

Nous n'avons paramétré que les colonnes de la table des entrées dans la mesure où c'est sur sa base que les écrans de la maquette allaient être développés. Les autres tables ont hérité des paramètres par défaut. La table des auteurs_entrées a été exclue de l'affichage car c'est une table de jointure.

Signification des options d'affichage

La séquence doit être comprise comme étant l'ordre d'affichage des colonnes à l'écran. Par défaut, les valeurs sont celles héritées de la séquence spécifiée lors de la définition des colonnes. La séquence à l'affichage peut toutefois être différente de celle générée par l'assistant conception base de données pour la définition des colonnes.

Le *prompt* correspond au libellé de la colonne affiché à l'écran.

Le type de données pour l'affichage correspond au type à utiliser pour la colonne dans l'application générée. Ce paramètre peut être utilisé par les générateurs pour la conversion des valeurs d'un format à un autre. Par exemple, une colonne de type numérique peut être affichée dans un format monétaire.

La longueur d'affichage est la longueur utilisée dans l'application générée. Elle peut être différente de la longueur spécifiée dans la définition de la colonne.

Le message d'aide est ce qui apparaît en bas et à gauche de l'écran, à l'attention de l'utilisateur quand il se positionne sur la colonne.

table des auteurs

Colonne	Séquence	Prompt	Type de données à afficher	Longueur d'affichage
id	1	id	integer	8
nom	2	nom	char	70
prénom	3	prénom	char	50

table des collections

Colonne	Séquence	Prompt	Type de données à afficher	Longueur d'affichage
id	1	id	integer	8
ent_id	4	id	integer	5
ent_annee	5	année	date	11

table des cotes

Colonne	Séquence	Prompt	Type de données à afficher	Longueur d'affichage
id	1	id	integer	9
ent_id	4	id	integer	5
ent_annee	5	année	date	11

table des entrées

Colonne	Séquence	Prompt	Message d'aide	Type de données à afficher	Longueur d'affichage
id	1	N°	Numéro de l'entrée.	integer	5
date	2			date	2
titre	4	Titre		char	50
te_code	8	Code du type d'entrée	Type d'entrée.	char	3
kseq1	10	Cote	Premier élément de la cote.	integer	6
ser_doc	12		Deuxième élément de la cote.	char	7
kseq2	14		Troisième élément de la cote.	integer	6

Colonne	Séquence	Prompt	Message d'aide	Type de données à afficher	Longueur d'affichage
kseq3	16		Quatrième élément de la cote.	char	20
date_ entrée	22	Date entrée		date	10
dépôt	27	Localisation	Dépôt où est stocké le document.	pop_list[2]	6
étage	28		Etage où est stocké le document.	char	2
magasin	29		Magasin où est stocké le document.	char	1
casier	30		Casier où est stocké le document.	char	4
travée	31		Travée où est stockée le document.	char	1
tablette	32		Dépôt où est stocké le document	char	2
descrip-tion	34	Description		long	50
période	37	Période		pop-list	8
code_ légal	41	Type légal	Code du type de la série.	pop-list	7
cat_ produc-teur	43	Cat producteur	Catégorie du producteur du document.	pop-list	7
code_ document	44	Type traitement	Type de traitement qui détermine la description du document.	pop-list	4

table des séries

Colonne	Séquence	Prompt	Type de données à afficher	Longueur d'affichage
code	1	code	char	8

table des types d'entrées

Colonne	Séquence	Prompt	Type de données à afficher	Longueur d'affichage
code	1	code	char	3
libelle	2	libellé	char	60

7.5 La validation

Durant cette étape, nous avons spécifié les détails de la validation pour toutes les colonnes de toutes les tables. Les informations sont utilisées pour déterminer l'accès aux tables, vues, *snapshots*[3] et colonnes, ainsi que leur utilisation. Quelques dix-neuf paramètres peuvent être renseignés. Nous n'avons défini que le message d'erreur à afficher lorsque la validation échoue pour les colonnes qui ne proposent pas de *pop-list.* Pour celles avec des *pop-lists*, la question ne s'est pas posée car l'utilisateur ne peut choisir que parmi les valeurs proposées. Pour les colonnes affichées en liste de

[2] Une *pop_list* est une liste qui se déroule lorsque l'utilisateur clique sur le bouton qui la contrôle. Elle laisse ainsi apparaître un ensemble de valeurs.

[3] Le *snapshot* (cliché) est 'une table qui contient le résultat d'une requête définie sur une ou plusieurs tables ou vues localisées sur une base de données distante. (...) Le *snapshot* est utilisé dans le contexte réparti et permet de maintenir des copies de données de la base distante sur le noeud local en lecture seulement. Un *snapshot* ne peut être mis à jour.' définition extraite de l'ouvrage *Oracle 7 : langages, architecture, administration* de messieurs A. ABDELLATIF, M. LIMAME, A. ZEROUAL. (Paris : Eyrolles, 1995. 464 p.), p. 51.

valeurs, nous avons jugé qu'un message était nécessaire. En effet, une valeur peut être saisie sans consultation préalable de la liste.

table des entrées

Colonne	Message d'erreur à la validation
te_code	Le type d'entrée est obligatoire.
kseq1	Saisir un numéro.
ser_code	Série inexistante.
kseq2	Saisir un numéro.
date_entrée	Utiliser le format JJ/MM/AAAA pour la date.
période	Période inexistante ou nulle.
code_légal	Le code légal est obligatoire.
cat_producteur	La catégorie de producteur est obligatoire.
code_document	Le code du document est obligatoire.

7.6 Les contraintes

Les règles de gestion validées lors de l'analyse détaillée ont été implémentées an niveau logique via des contraintes. De cette façon, les opérations de manipulation de données sont effectuées tout en maintenant la base dans un état cohérent.

Les contraintes ont été définies directement depuis le navigateur du dictionnaire conceptuel. Quatre types de contraintes peuvent être implémentées : les contraintes sur les clés primaires, les contraintes sur les clés uniques, les contraintes sur les clés étrangères et les contraintes associées à une ou plusieurs conditions (*check constraint*).

Les contraintes sur les clés primaires et uniques interdisent toutes deux la création de deux lignes avec la même valeur pour une même colonne. Mais les premières, à la différence des secondes, n'autorisent pas les valeurs *null*.

Les contraintes sur les clés étrangères sont définies pour préserver l'intégrité des données dans les tables qui sont reliées par des colonnes communes. Par exemple, si un utilisateur essaie de saisir une `série` dans la table `entrées`, `série` qui n'existe pas dans la table `séries`, l'insertion est interdite et donc échoue. Cette contrainte permet de s'assurer que la valeur de la colonne `ser_code` dans `entrées` est toujours valide.

Enfin, les contraintes associées à une condition sont utilisées pour vérifier que les valeurs que l'utilisateur souhaite insérer dans une colonne ou un ensemble de colonnes, sont bien celles spécifiées dans la définition de la table. La portée de ces contraintes est limitée à une seule et même table. Elles ne peuvent être utilisées pour référencer d'autres colonnes sur différentes tables.

7.6.1 La définition des contraintes

Nous avons créé des contraintes sur les clés primaires et étrangères et trois *check constraints*. Nous les avons toutes implémentées aux niveaux application ET serveur pour plus de sécurité. Il est à noter que *Designer 2000* laisse aussi la possibilité d'une implémentation, soit au niveau applicatif, soit au niveau serveur (paramètre *Validat In dropdown list*). Nous avons attribué à chaque contrainte un nom unique, composé du nom du prototype suivi d'un souligné, du nom court de la table, d'un souligné et du nom de la contrainte.

7.6.2 Les contraintes sur les clés primaires

table des auteurs

Clé primaire	Colonne	Type de données	Longueur	Valeurs nulles autorisées
proto_aut_pk	id	number	7	non

table des auteurs_entrées

Clé primaire	Colonne	Type de données	Longueur	Valeurs nulles autorisées
proto_aut_ent_pk	id	number	7	non

table des collections

Clé primaire	Colonne	Type de données	Longueur	Valeurs nulles autorisées
proto_col_pk	id	number	7	non

table des cotes

Clé primaire	Colonne	Type de données	Longueur	Valeurs nulles autorisées
proto_col_pk	id	number	8	non

table des entrées

Clé primaire	Colonne	Type de données	Longueur	Valeurs nulles autorisées
proto_ent_pk	id	number	4	non
	année	date		non

table des séries

Clé primaire	Colonne	Type de données	Longueur	Valeurs nulles autorisées
proto_ser_pk	code	varchar2	7	non

7.6.3 Les contraintes sur les clés étrangères

Les contraintes sur les clés étrangères nous ont amenés à préciser les actions qui seront déclenchées suite à l'effacement (*cascade delete*) ou la mise à jour d'une ligne (*cascade update*). Nous avons prévu trois cas de figure :

1. prévenir l'effacement ou la mise à jour d'une ligne dans la table de jointure quand une ligne associée existe dans la table principale. L'action à prendre est *restricted.*
2. effacer ou mettre à jour une ligne de la table principale si une ligne associée est supprimée ou mise à jour dans la table de jointure. L'action à prendre est *cascade.*
3. mettre à la valeur *null* la valeur de la clé étrangère dans la table principale pour laquelle une ligne de la table de jointure associée est supprimée ou mise à jour. L'action à prendre est *nullifies.*

table des auteurs_entrées

Clé étrangère	Table de jointure	Col.	Type de données	Lon-gueur	Val. nulles autorisées	Col. de jointure	Cascade delete	Cascade update
proto_aut_ent_aut_fk	prot_auteurs	aut_id	number	7	non	id	restricted	restricted
proto_aut_ent_ent_fk	prot_entrées	ent_id	number	4	non	id	cascades	restricted
		ent_année	date		non	id		

table des collections

Clé étrangère	Table de jointure	Colonne	Type de données	Longueur	Val. nulles autorisées	Col. de jointure	Cascade delete	Cascade update
proto_col_ent_fk	prot_entrées	ent_id	number	4	vrai	id	nullifies	nullifies
		ent_annee	date		vrai	annee		

table des cotes

Clé étrangère	Table de jointure	Colonne	Type de données	Longueur	Val. nulles autorisées	Col. de jointure	Cascade delete	Cascade update
proto_cot_ent_fk	prot_entrées	ent_id	number	4	vrai	id	nullifies	restricted
		ent_annee	date		vrai	annee		

table des entrées

Clé étrangère	Table de jointure	Colonne	Type de données	Longueur	Val. nulles autorisées	Col. de jointure	Cascade delete	Cascade update
proto_ent_ser_fk	prot_séries	ser_code	varchar2	7	vrai	code	restricted	restricted
proto_ent_te_fk	prot_typents	te_code	varchar2	3	non	code	restricted	restricted

7.6.4 Les contraintes associées à une condition

Pour créer ces contraintes, nous avons dû donner un nom à chacune d'elles et préciser leur signification. Nous avons également rédigé leur texte en code PL/SQL. Celui-ci a été utilisé par *Designer 2000* pour la génération des programmes chargés de garantir l'intégrité de la base.

table des entrées

Contrainte de vérification	Signication	Texte de la contraite
proto_ent_ck1	kseq1 doit être *null* ou supérieur à 0.	(kseq1 > 0) *or* (kseq1 is *null*)
proto_ent_ck2	kseq2 doit être *null* ou supérieur à 0.	(kseq2 > 0) *or* (kseq2 is *null*)
proto_ent_ck3	la cote ne doit pas contenir de blanc.	(*instr* (kseq3, ' ') <= 0) *and* (*instr* (ser_code, ' ') <= 0)
proto_ent_ck4	kseq3 ne peut pas débuter par un chiffre.	(*substr* (kseq3, 1, 1) > '9') *or* (*subtr*(kseq3, 1, 1) < '0')
proto_ent_ck5	si kseq3 est renseigné, kseq3 doit se terminer par un chiffre.	(*substr* (kseq3, *length* (kseq3), 1) >= '0') *and* (*substr* (kseq3, *length*(kseq3), 1) >= '9)
proto_ent_ck6	si ksérie n'est pas *null*, alors au moins l'un des trois attributs, kseq1, kseq2 ou kseq3 n'est pas *null*.	*complex*:gaiat11002(ser_code, kseq1, kseq2, kseq3)
proto_ent_ck7	si ksérie est *null*, alors kseq1, kseq2 et kseq3 sont aussi *null*.	*complex*:gaiat11001(ser_code, kseq1, kseq2, kseq3)

Explications sur le texte de certaines contraintes :

proto_ent_ck3

La fonction *instr* permet de rechercher dans la chaîne `kseq3`, la chaîne représentée par un espace ' '. Si cette dernière est absente, la fonction renvoie une valeur égale à 0.

proto_ent_ck4

La fonction *substr* sert à récupérer le premier caractère de `kseq3`. Ensuite, un test est effectué afin de déterminer si ce caractère est une lettre.

proto_ent_ck5

La fonction *length* retourne la longueur de `kseq3`, ce qui permet de déterminer la position du dernier caractère. Ensuite, on extrait celui-ci par la fonction *substr* et on teste pour savoir s'il s'agit ou non d'une lettre.

proto_ent_ck6 et proto_ent_ck7

Le texte de ces deux contraintes est trop long pour être enregistré comme celui des contraintes précédentes. Il a été nécessaire de le codifier dans des fonctions, gaiat11002 et gaiat11002. Ces dernières ont comme paramètres `ser_code`, `kseq1`, `kseq2` et `kseq3`. *Complex* est un mot réservé. Il signifie à *Designer 2000*, lorsque l'on demande la génération automatique du code correspondant au texte de la contrainte, de ne créer que les appels aux fonctions.

7.7 Conclusion

La conception logique de la base de données a été le premier aboutissement de l'analyse conceptuelle. Elle a représenté l'étape la plus rapide et la plus simple grâce au soin apporté aux analyses préalable et détaillée. Elle a été prolongée par l'implémentation des règles de gestion. A sa suite, trois écrans ont être développés : le premier pour la saisie d'une entrée, le deuxième pour sa recherche et le troisième pour sa modification.

8. BILAN DE L'ETUDE ET PROPOSITIONS POUR LE FUTUR

Dans les trois parties précédentes, j'ai rendu compte le plus fidèlement possible de la démarche adoptée lors de l'analyse. Je ne peux malheureusement pas présenter l'application à laquelle elle a abouti. Celle-ci n'est, à ce jour, pas encore terminée. Toutefois, j'aimerais dresser un bilan de cette étude. Il me semble intéressant de réfléchir brièvement sur ce que m'a apporté son suivi et sur l'intérêt de la méthode Oracle. Enfin, je souhaiterais proposer quelques pistes de réflexion pour le futur de GAIA par rapport aux enseignements de la maquette et à mes observations personnelles.

8.1 Le suivi de l'étude

Le suivi de cette étude a été très bénéfique sur plusieurs points : acquisitions de connaissances théoriques et pratiques sur la modélisation d'un système d'information et apprentissage d'un atelier de génie logiciel.

Ce suivi m'a permis d'apprécier tout l'intérêt de travailler à partir d'une méthode de modélisation telle que *Custom development method,* lorsqu'il s'agit de revoir la conception d'un système existant. La méthode aide à comprendre de la nature de l'information et des traitements à prendre en compte. Elle contribue à une meilleure communication entre les différentes personnes impliquées dans le projet dans la mesure où tout le monde parle de la même chose avec les mêmes mots. Enfin, elle oblige à la constitution d'une documentation de référence indispensable à la compréhension et à la maintenance du système.

J'ai fait connaissance avec l'outil informatique qui supporte la méthode : l'atelier de génie logiciel *Designer 2000.* Le suivi de l'étude m'a conduite à apprendre les concepts et manipulations de base de cet outil tout à la fois puissant et aussi parfois complexe. Toutefois, je regrette le manque de temps et l'absence de véritables compétences à disposition pour m'aider à mieux maîtriser *Designer 2000.* Celles-ci m'auraient aidée à retrouver plus facilement l'information dont j'avais besoin pour la constitution de ce rapport et peut-être à aller plus loin dans l'étude elle-même.

8.2 L'intérêt de la méthode Oracle

La méthode Oracle a été choisie pour procéder à une étude critique de la gestion des entrées telle qu'elle existe actuellement. Sa déclinaison en deux phases, analyse préalable et analyse détaillée, a conduit à une approche progressive, globale et approfondie du module.

L'analyse préalable a permis d'étudier globalement, c'est-à-dire sans entrer dans les détails, toutes les composantes du système : les informations, représentées par les entités, ainsi que les traitements à travers les différentes fonctionnalités. Elle nous a conduit à mener une réflexion qui aboutisse à une clarification et une simplification du système tout en le rendant plus performant et convivial. Nous avons réfléchi selon les cinq axes préconisés par la méthode :

1. revoir toutes les entités et nous interroger sur leur raison d'être et leur utilité, ce qui nous a conduit à exclure de la maquette l'entité `bibcls`, parce que non fondamentale dans la gestion des entrées.
2. exploiter la notion de domaine, ce qui nous a permis de faire l'économie de cinq entités.
3. ne prendre en compte que les attributs obligatoires et les plus significatifs pour aller à l'essentiel.
4. passer en revue toutes les fonctions afin de vérifier que chacune d'elles est bien utilisée. Nous avons identifié les fonctions communes pour ne pas générer des écrans identiques et écrire plusieurs fois les mêmes procédures ou fonctions inutilement.
5. recenser toutes les règles de gestion et les classer méthodiquement de façon à bien cerner toutes les contraintes à faire supporter par le système.

L'analyse détaillée a contribué à affiner, à corriger et à valider les choix faits lors de l'étape précédente. Nous avons systématiquement revu tout ce qui avait été défini : les entités, les domaines,

les attributs, les relations, les fonctions et les règles de gestions. Ce réexamen, plus approfondi, nous a permis d'éliminer sept fonctions et donc de diminuer d'autant le travail induit ultérieurement par celles-ci. Nous avons également simplifié la définition de certains attributs par l'utilisation des domaines de formatage et, de ce fait, amélioré leur gestion.

Si tout ce travail de définition de l'application a pris un certain temps, dix sept jours au total, la génération de l'application n'a pris que quelques minutes. Le code informatique nécessaire à la création des tables, des contraintes, des séquences et des déclarations des utilisateurs est automatiquement généré dans des scripts par *Designer 2000*. Cette génération automatique a représenté un gain de temps considérable.

8.3 Les enseignements à tirer de la maquette

8.3.1 Créer un modèle des processus

Avant même de commencer les phases d'analyse proprement dite, il serait indispensable de procéder à une modélisation précise de tous les processus faisant partie de la gestion des archives. Celle-ci devrait faire se rencontrer impérativement l'informaticien et l'archiviste ainsi que les autres professions concernées. Cette phase a été quelque peu négligée lors de la réalisation de la maquette. Cette négligence a été source d'incompréhensions, d'hésitations et d'erreurs aussi. Les phases d'analyses préalable et détaillée devraient être tout aussi approfondies et menées en étroite concertation avec les utilisateurs. Le temps ainsi passé, même s'il peut sembler parfois inutilement long, sera regagné en phase de conception.

8.3.2 Eliminer les entités et attributs superflus

Il serait souhaitable d'éliminer les entités et les attributs qui ont servi, mais qui ne servent plus. Certaines entités ont des attributs qui font double emploi, notamment au niveau des identifiants. Les entités `type entrée` et `catégorie producteur` ont chacune trois attributs : un identifiant, un code et un libellé. Or l'identifiant n'a de l'identifiant que le nom puisque c'est en fait le code qui joue ce rôle. A l'occasion de la création de la base de développement, dont il est question au chapitre suivant, les identifiants '`id`' ont été supprimés de la définition des tables `typ_ent` et `cat_prod`. Il faut poursuivre dans cette voie de façon à 'nettoyer' et simplifier toutes les entités concernées.

8.3.3 Limiter la redondance de l'information

Il faudrait également repenser certaines entités de manière à réduire le nombre de leurs attributs. Certaines d'entre elles sont très chargées, elles vont jusqu'à près de cinquante attributs : quarante six pour `entrée`, cinquante pour `répertoire`, cinquante cinq pour `archiv` et soixante quatre pour `cote`. La redondance est aussi un point à revoir. Certains attributs se retrouvent plusieurs fois dans différentes entités avec le même signification. La cote, qui est constituée des quatre attributs `kseq1`, `ksérie`, `kseq2` et `kseq3`, est un bon exemple. Ces derniers sont présents dans cinq entités : `collection`, `cote`, `entrée`, `fonds` et `répertoire`. `kseq1` et `ksérie` font aussi partie de `lot`. Tous ces attributs sont intégralement repris dans une quinzaine de tables de travail. Il faudrait voir s'il n'y a pas moyen de réorganiser autrement ces informations.

8.3.4 Définir des standards pour nommer les différents objets de la base

Il faudrait également penser à définir des règles de nommage des entités et des attributs. L'absence de convention suivie tout au long du développement de GAIA apparaît clairement. Quelques tables ont été préfixées par 'tab', comme `tabrenv`, `tabrepro`, `tabserie`, `tabsujet`, `tabtr`, `tabtri`, `tabtype`, `tab_interro` et `tab_rep`. Ce préfixe n'apporte rien à la compréhension et à la clarté du système. Certaines tables dites 'de travail', qui servent à la génération d'éditions ou de certains

traitements de mise à jour, ont aussi été préfixées de quatre façons différentes : 'ed', 'tep', 'w' et 'rep'. La distinction des tables de travail est très utile, mais elle devrait être faite de manière homogène. Les noms de certaines tables sont au singulier, d'autres au pluriel. Toutes les tables devraient recevoir un nom au pluriel comme le préconise la méthode Oracle, le nom au singulier étant réservé pour les entités au niveau conceptuel. De la même façon, les noms des clés étrangères devraient se rapprocher le plus possible des tables vers lesquelles elles renvoient : il est difficile de faire le rapprochement entre la colonne `entrée.code_légal` et le nom de la table vers laquelle elle renvoie `typsérie`, de même pour la colonne `entrée.codedoc` et la table `typtrait`.

8.3.5 Revoir les relations dépourvues de sens

Dès l'analyse préalable, certaines relations sont apparues incorrectes du point de vue du sens. Les entités `auteur` et `collection` sont en relation avec `entrée`. Cette relation répond à des besoins de recherche exprimés par les utilisateurs. Cependant, `auteur` et `collection` sont des notions qui se rapportent au document lui-même. Elles recouvrent des notions de description documentaire, alors que l'`entrée` elle-même est un élément de gestion. Cette relation devrait être modifiée. La recherche sur l'`auteur` ou la `collection` par rapport à une `entrée` devrait pouvoir se faire dans le module de description documentaire. Le numéro d'entrée y serait interrogeable au même titre que n'importe quel autre élément de description des documents.

8.3.6 Améliorer la présentation des écrans

Au niveau conception des interfaces, il serait souhaitable de définir des standards d'affichage et d'alléger certains écrans trop chargés qui frisent l'illisibilité tant les informations sont nombreuses et présentées de façon compacte. La solution pourrait être la réorganisation de cette information sur deux pages d'écran au lieu d'une. Les champs obligatoires et le plus souvent renseignés pourraient être présentés sur la première page d'écran, les champs moins pertinents sur la seconde. Cette nouvelle répartition permettrait alors l'affichage de champs tels que les titres ou les descriptions sur un minimum de trois lignes au lieu d'une actuellement.

8.3.7 Orienter l'aide vers l'utilisateur

L'aide en ligne à l'utilisateur serait également à revoir quantitativement et qualitativement. Il faudrait proposer une aide contextuelle aux niveaux des tables et des colonnes. Il faudrait également bannir tout mot de vocabulaire informatique tel que 'champ', 'enregistrement' et les affichages de messages tels que 'Touche <menu bloc>', '<Liste ><Remplace>' qui sont inintelligibles pour l'utilisateur novice. De même, lorsqu'un utilisateur commet une erreur dans la saisie d'une information telle que la cote par exemple, lui renvoyer un message tel que 'Structure de la cote incorrecte' ne sert pas à grand-chose sinon à lui dire qu'il est en faute. Il faudrait mieux un message qui lui indique comment corriger son erreur et qui pourrait être 'La cote ne doit pas contenir de blanc.'.

8.4 Propositions pour la future base

8.4.1 Développer le module de recherche de la gestion des entrées

La recherche a été l'élément sacrifié de la maquette. Comme le montre le diagramme de la hiérarchie des fonctions réalisé lors de l'analyse préalable, la recherche d'une entrée de documents se fait en quatre étapes :

1. saisie des critères de recherche,
2. affichage du sommaire des réponses, qui comprend, en plus des références signalétiques des documents, un compteur indiquant le nombre de documents effectivement visualisés,

3. sélection d'une entrée,
4. visualisation d'une entrée.

Lors de la modélisation prochaine du système, ce module pourrait être réétudié de façon à proposer un élargissement des possibilités de recherche et une amélioration de la convivialité. Le schéma pourrait être amélioré de la façon suivante.

Une fois la première recherche effectuée, le nombre de documents obtenus en résultats pourrait être affiché. En fonction de ce nombre, l'utilisateur pourrait choisir entre la restriction ou l'élargissement de sa requête. De plus, si le nombre de documents est égal à zéro, le système pourrait lui renvoyer un message contextuel tel que : 'Aucune entrée ne répond à ce type d'entrée.' s'il a interrogé sur le type d'entrée. Ce message serait plus significatif que 'L'interrogation n'a ramené aucun enregistrement.' où le terme 'enregistrement' appartient plus au langage de l'informaticien qu'à celui de l'utilisateur.

L'affichage du sommaire des résultats devrait être gardé et amélioré, même s'il a été éliminé pour la maquette. Cependant, l'utilisateur devrait choisir, en 'cochant', les références qui l'intéressent. Du point de vue informatique, ceci implique la création d'une nouvelle requête. Cette sélection pourrait être redirigée soit vers l'écran, soit vers un fichier de sortie récupérable sous traitement de texte, soit directement vers l'imprimante. Pour ces deux dernières redirections, deux formats pourraient être proposés : une édition abrégée correspondant aux références du sommaire, une édition complète comprenant toutes les informations relatives à l'entrée telles qu'elles figurent sur l'écran de saisie.

8.4.2 Repenser la documentation utilisateur

La question de la documentation utilisateur n'a pas été abordée car elle n'a pas été incluse dans la mission Oracle. En principe, elle fait partie de la phase de réalisation et de mise en oeuvre d'une application. Elle comprend la rédaction des manuels d'utilisation des programmes et des procédures automatisées ou manuelles. Ces manuels sont importants. Ils servent à l'utilisateur quand il se retrouve seul face au programme. Ils contribuent à réduire le nombre d'appels adressés à l'assistance et liés à l'utilisation pure du système. Ils doivent permettre de répondre à des questions telles que : 'comment inscrire un versement de documents au registre d'entrée ?, 'comment rechercher une entrée versée début septembre ?'.

Lors de l'analyse, le recours au manuel utilisateur de GAIA devait être un moyen de recueillir une partie des informations nécessaires à la modélisation et de comprendre la signification du registre d'entrée en général ainsi que le fonctionnement de sa gestion informatisée. Or, le manuel tel qu'il est conçu actuellement, n'a permis de recueillir que très peu d'informations utiles à notre travail et à notre compréhension du module. Si sa décomposition en deux parties, théorique et pratique, convient tout à fait, leur contenu ne répond malheureusement pas à ce que l'on pourrait en attendre.

La partie théorique reprend plus ou moins la structure du menu général des entrées de GAIA. En fait, il serait souhaitable qu'elle explique la gestion des entrées indépendamment de la gestion informatisée. Elle pourrait par exemple situer le registre des entrées dans la gestion des archives telle que pratiquée dans la réalité et présenter les différentes notions et actions qui lui sont associées. Ceci permettrait de bien comprendre l'ensemble du processus.

La partie pratique quant à elle, devrait se limiter à l'utilisation du logiciel. Elle devrait s'appuyer le plus possible sur des copies d'écrans et expliquer de bout en bout la création d'une entrée, comme cela est plus ou moins fait pour les entrées temporaires. Les explications devraient être présentées comme les instructions à suivre pour une recette de cuisine et à partir d'un exemple réel. Ainsi guidé, l'utilisateur qui ne connaît pas GAIA, pourrait alors apprendre seul les manipulations de base du logiciel. Pour la création d'une entrée, les instructions pourraient expliquer :
1. comment accéder à la gestion des entrées depuis le menu général de GAIA,
2. comment remplir l'écran destiné à préciser le type de l'entrée : utilité d'un tel écran, les conséquences des choix faits à ce niveau pour la suite de la description documentaire,

3. comment renseigner l'écran de saisie : les éléments obligatoires et importants, ceux qui sont plus accessoires.

Pour la recherche, les explications pourraient détailler :

1. l'accès au menu de recherche d'une entrée depuis le menu général de GAIA,
2. la grille de saisie des critères de recherche, l'opérateur de recherche utilisé lorsque plusieurs critères sont entrés,
3. l'affichage du sommaire des résultats : la présentation des différents éléments, la sélection d'une référence et sa visualisation au format complet.

9. UN ENVIRONNEMENT DE DEVELOPPEMENT POUR GAIA

La réalisation de la maquette représente une étape importante de la réflexion menée sur GAIA. Elle marque le début d'une véritable politique de gestion des développements, jusqu'alors inexistante. Dans un tout premier temps, elle s'est traduite par la mise en place d'un environnement de développement : création d'une base de test et constitution d'une documentation de référence sur l'application.
Avant août 1996, il n'existait qu'une seule base de production. Celle-ci servait aussi à tester tous les nouveaux modules et modifications de traitement et de structure. Cette situation présentait quelques risques, entre autres, celui de perturber le travail des utilisateurs si des problèmes survenaient. La création d'une base de développement, qui prépare la phase de réécriture de GAIA, s'imposait. Son premier usage a été de tester les nouvelles possibilités de la version 7 d'Oracle. J'ai donc été chargée de mettre à jour le script de création de la base de développement GAIA2 en renforçant les contraintes d'intégrité et en remplaçant certains déclencheurs *Forms* par des déclencheurs de base de données. Une fois le script de création mis à jour, la base de développement a été générée.
Enfin, j'ai travaillé sur le seul référentiel valide qui existe sur GAIA de façon en faire en une petite base de données que l'on puisse facilement mettre à jour et interroger.

9.1 Le renforcement des contraintes d'intégrité

9.1.1 Les contraintes déclaratives

La version 6 d'Oracle offre la possibilité de déclarer des contraintes d'intégrité, mais celles-ci restent limitées. Les contraintes déclaratives sont définies à la création des tables. Elles se limitent au type de données, à l'autorisation ou non de valeurs nulles (*not null*) et à l'unicité des lignes (contrainte *unique*). Ces contraintes ne sont pas prises en compte par le noyau d'Oracle, elles sont utilisables par *SQL*Forms*. Les contraintes plus élaborées sont appliquées par les programmes de l'application.

La version 7 d'Oracle offre des améliorations notables. Les contraintes déclaratives sont supportées par le noyau du logiciel. Elles sont définies une seule fois et sont valables quel que soit le programme qui accède à la base de données. Leur champ a été élargi. Ce peut être des contraintes d'entité telles que *default*, *not null*, *unique* et *check*, des contraintes d'intégrité référentielle d'entité : clé primaire et clé étrangère; des contraintes d'action d'entité : action de suppression en cascade, restriction de mise à jour et de suppression.

Les contraintes que j'ai ajoutées dans le script de création de la base sont des contraintes d'entité (*not null*) et surtout des contraintes d'intégrité référentielle : clés primaires et étrangères. *Primary key* définit une ou plusieurs colonnes comme la clé primaire, chaque ligne est ainsi identifiée de manière unique. *Foreign key* indique qu'un champ simple ou composé est en relation avec une clé primaire ou unique définie précédemment.

Exemple extrait du script de création de la base :

```
create table "GAIA"."ACTION"("ID_ACTION"number(7) not null,
            "ID_ATTRIB" number(7),
            "ACTION" varchar2(100) not null,
            "ID_PERIODE" number(7),
            "CODE_LEGAL" varchar2(3),
            "REGL" varchar2(1),
            "AGR" varchar2(1),
            constraint action_id_action_pk primary key (ID_ACTION),
            constraint action_id_attrib_fk foreign key (ID_ATTRIB)
              references "GAIA"."ATTRIB"(ID_ATTRIB),
            constraint action_id_periode_fk foreign key (ID_PERIODE)
              references "GAIA"."PERIODE"(ID_PERIODE),
            constraint action_code_legal_fk (foreign key (CODE_LEGAL)
              references "GAIA"."TYPSERIE" (CODE))
pctused 85 pctfree 10 initrans 1 maxtrans 255 tablespace"TS_GAIA3DATA"
```

9.1.2 Les contraintes procédurales

Les contraintes déclaratives ne suffisent pas à assurer l'intégrité de la base. Elles doivent être complétées par des *triggers*, ou déclencheurs. Pour rappel, ces derniers sont des traitements associés à un événement tel que l'insertion, la modification ou la suppression (voir également la note en bas de page de la partie 4.4.1 consacrée à la présentation de *Designer 2000*).

Dans la version 6 d'Oracle, la mise en place de ces *triggers* se fait via *SQL*Forms*. Ceux-ci sont alors définis au niveau de l'application (niveau le plus haut), du champ et/ou d'une touche particulière du clavier. Les restrictions affectant le niveau de définition conduit à réécrire inévitablement plusieurs fois le même code dans le cas d'applications relativement proches.

La version 7 d'Oracle apporte de nouvelles possibilités : les déclencheurs peuvent être définis au niveau de la base de données sous la forme de procédures et de fonctions. Ce sont les *'database triggers'*. Ceux-ci contribuent à améliorer la modularité, la factorisation du code, l'optimisation des performances et des ressources de la machine.

Les traitements les plus fréquents et les plus utilisés par plusieurs applications sont implémentés une seule fois sous forme de procédures et de fonctions. La consommation est optimisée en raison de l'unicité et de la partageabilité du code. Enfin, le code stocké est déjà compilé. Dans un contexte client-serveur comme celui de GAIA, il est plus performant d'appeler une procédure stockée dans la base de données exécutant n ordres SQL que d'effectuer n appels au serveur, chaque appel exécutant un ordre SQL. De plus, les ordres sont exécutés au niveau du serveur, ce qui évite les allers-retours entre le client et le serveur.

Les *triggers* mis en place l'ont tous été au niveau de la touche de suppression. Ce sont des déclencheurs '*key-delrec*'. Dans la version 6 d'Oracle, ces derniers avaient été définis au niveau de *SQL*Forms* et présentaient donc tous les inconvénients mentionnés ci-dessus, plus celui du manque d'homogénéité au niveau de l'écriture du code.

Les quelque quarante *triggers Forms* étudiés déclenchent quatre catégories d'action :

- l'affichage de deux types de messages. Le premier invite l'utilisateur à positionner correctement le curseur pour procéder à l'effacement. Le second l'avertit de l'interdiction de procéder à une telle opération.
- des contrôles sur le type d'utilisateur voulant effacer l'enregistrement,
- des tests sur l'existence ou non d'enregistrements dépendants,
- des mises à jour ou des suppressions.

Les *triggers* gérant les messages ont été exclus car ils se rapportent toujours à une application en particulier. Tous les autres ont fait l'objet d'une réécriture. Leur analyse a permis de mettre en évidence trois critères conditionnant la suppression d'un enregistrement :

- l'utilisateur est ou n'est pas DBA,
- il existe ou non des liens vers d'autres enregistrements.

Suivant la nature des tables, tables principales ou tables de jointure, la mise à jour se traduit respectivement soit par une modification de certaines colonnes à la valeur *null*, soit par une suppression de l'enregistrement.

Ont donc été créées :

- une fonction pour tester le type de l'utilisateur :

```
create or replace function UtilisateurEstDBA
    return boolean is

  Administrateur   char(3) := 'DBA';
  Utilisateur      varchar2(10);
  DBATrouve        boolean;

  begin
    DBATrouve := false;

    select distinct(utitype) into Utilisateur from utilisateurs
    where uticuse = user;
    if Utilisateur = Administrateur then
      DBATrouve := true;
    end if;
    return DBATrouve;
end UtilisateurEstDBA;
/
```

- des procédures de contrôles sur la présence de liens entre les tables. Exemple de tests sur la table `acteur` :

```
create or replace function LienAvecActeurExiste (
    UnActeur IN number ) return boolean is

  LienTrouve       boolean;
  LiensCpte        number(8);

  begin
    LienTrouve := false;
    select count(*) into LiensCpte from archiv
    where id_acteur = UnActeur;
    if LiensCpte > 0 then
      LienTrouve := true;
    end if;

  if not LienTrouve then
    select count(*) into LiensCpte from repertoire
      where id_acteur = UnActeur;
      if LiensCpte > 0 then
        LienTrouve := true;
      end if;
    end if;
```

```
    return LienTrouve;
end LienAvecActeurExiste;
/
```

- des procédures de mise à jour des enregistrements liés. Exemple sur la table `attribution` :

```
create or replace procedure MettreAJourLiensAttribution (
    UneAttribution IN number ) is

  begin
    update action set id_attrib = null where id_attrib = UneAttribution;
    if sql%notfound then
      null;
    end if;
    update archiv set id_attrib = null where id_attrib = UneAttribution;
    if sql%notfound then
      null;
    end if;
    delete from attass where ( id_attrib = UneAttribution ) or
                             ( id_attass = UneAttribution );
     if sql%notfound then
       null;
     end if;
    delete from hisatt where id_attrib = UneAttribution ;
    if sql%notfound then
      null;
    end if;
    update repertoire set id_attrib = null where id_attrib =
                                                 UneAttribution;
    if sql%notfound then
      null;
    end if;
    update tabtr set id_attrib = null where id_attrib = UneAttribution;
    if sql%notfound then
      null;
    end if;
end MettreAJourLiensAttribution;
/
```

- des procédures de suppression des enregistrements liés. Exemple sur la table `auteurs` :

```
create or replace procedure SupprimerLiensAuteurs (
    UnAuteur number) is

  begin
    delete from aut_coll where id_auteur = UnAuteur;
    if sql%notfound then
      null;
    end if;
    delete from bibl_auteur where id_auteur = UnAuteur;
    if sql%notfound then
      null;
    end if;
    delete from aut_doccomp where id_auteur = UnAuteur;
    if sql%notfound then
      null;
    end if;
```

```
    delete from entree_auteur where id_auteur = UnAuteur;
    if sql%notfound then
        null;
      end if;

end SupprimerLiensAuteurs;
/
```

Ces procédures et fonctions sont appelées depuis les *database triggers*. Ceux-ci ont une structure bien précise. Pour définir un *trigger*, il faut lui donner un nom, lui indiquer le moment où il sera déclenché (avant/après), l'événement (l'insertion, la modification, la suppression), le type (transaction, ligne), la restriction (elle est optionnelle , *where* <condition>), le code du block PL/SQL qui peut contenir des ordres SQL, des appels à des procédures et fonctions. Enfin, si des erreurs se produisent lors de l'exécution du programme, une procédure peut être appelée : *raise-application-error*. Cette procédure a deux paramètres : un numéro d'erreur et un message. Elle fait sortir l'utilisateur du programme.

Les déclencheurs définis dans GAIA2 sont déclenchés soit avant, soit après l'effacement. Ils portent tous sur la ligne. Aucune restriction n'a été apportée.

Les déclencheurs avant suppression font appel aux fonctions de tests sur l'utilisateur et sur la présence de liens entre les tables. Ils préviennent toute suppression dans les cas suivants :

- l'utilisateur n'est pas le DBA et il existe des liens avec une ou plusieurs tables :

```
create or replace trigger controler_liens_acteur
  before delete
  on acteur
  for each row

  begin
  if ( not UtilisateurEstDBA )and
       ( LienAvecActeurExiste ( :old.id_acteur ) ) then
      raise_application_error (-20011, 'Suppression impossible : il
      existe des liens.');
    end if;

end;
/
```

- il existe des liens avec une ou plusieurs tables :

```
create or replace trigger controler_liens_comm_lecteur
  before delete
  on lecteurs
  for each row

  begin

  if LienAvecCommLecteurExiste ( :old.numlect ) then
    raise_application_error (-20011, 'Suppression impossible : il existe
    des liens');
  end if;

end;
/
```

- l'utilisateur n'est pas le DBA ou l'utilisateur est le DBA, mais il reste des enregistrements liés :

```
create or replace trigger controler_liens_service
before delete
on service
for each row

  begin

    if not UtilisateurEstDBA then
      raise_application_error (-20012, 'Suppression interdite, veuillez
      consulter votre DBA');
    else if LienAvecServiceExiste ( :old.id_service ) then
      raise_application_error (-20011, 'Suppression impossible : il
      existe des liens.');
    end if;

end;
/
```

Les déclencheurs après suppression font tous appel aux procédures de mise à jour ou d'effacement (ou dans quelques cas, les deux en même temps) et conditionnent ces opérations au fait que l'utilisateur est DBA. S'il n'y a aucun contrôle sur l'utilisateur depuis le déclencheur, ce cnotrôle se trouve en général dans la procédure de mise à jour/suppression des fichiers dépendants.

```
create or replace trigger mettre_a_jour_liens_acteur
after delete
on acteur
for each row

  begin

    if UtilisateurEstDBA then
       MettreAJourLiensActeur ( :old.id_acteur );
    end if;

end;
/


create or replace trigger mettre_a_jour_liens_entree
after delete
on entree
for each row

  begin

    MettreAJourLiensEntree ( :old.id_entree );

end;
/
```

Enfin, lorsque le déclencheur lancé avant l'effacement empêche toute mise à jour parce qu'il reste des liens, que l'utilisateur soit ou non habilité à effectuer ce type d'opérations, aucun déclencheur "*after delete*" n'a été créé.

9.2 La création de la base de test

Le script de création mis à jour et enrichi des contraintes d'intégrité a servi de base à la création de deux nouveaux scripts. Le premier comprend uniquement la définition des différentes tables et des contraintes *not null* et *primary key*. Extrait du script de création :

```
set term on
set feed on
set echo on
spool gentab.lis
create table "ABONNE"("ID_ABONNE"number(7) not null,
            "ABONNE"varchar2(40) not null,
            "TYPE"varchar2(2) not null)
/
create table "ACTEUR"("ID_ACTEUR"number(7) not null,
            "LIBELLE"varchar2(100) not null,
            constraint acteur_id_acteur_pk    primary key (ID_ACTEUR))
/
create table "ACTION"("ID_ACTION"number(7) not null,
            "ID_ATTRIB"number(7),
            "ACTION"varchar2(100) not null,
            "ID_PERIODE"number(7),
            "CODE_LEGAL"varchar2(4),
            "REGL"varchar2(1),
            "AGR"varchar2(1),
            constraint action_id_action_pk  primary key (ID_ACTION))
/
create table "ARCHIV"("ID_ARCH"number(7) not null,
            "NUMCOTE"number(8) not null,
            "NOTES"varchar2(255),
            "ID_TYPO"number(7),
            "ID_CLASSIF"number(7),
            "ANALYSE"long,
            "METRAGE"number(7, 2),
            "INTCLASS"varchar2(60),
            "CODE_TYPART"varchar2(3),
            "CODE_SUPP"varchar2(3),
            "CODE_ECRI"varchar2(3),
            "CODE_ETAT"varchar2(3),
            "CODE_TRAV"varchar2(3),
            "CODE_FORMAT"varchar2(3),
            "ID_TUTELLE"number(7),
            "ID_PROD"number(7),
            "ID_SERVICE"number(7),
            "CODECOMP"varchar2(1),
            "INTCLASS2"varchar2(60),
            "ID_ACTEUR"number(7),
            "ID_DICTEXTE"number(7),
            "ID_ATTRIB"number(7),
            "ID_ACTION"number(7),
            "NOEDIT"varchar2(15),
            "DIM"varchar2(15),
            "ECHELLE"varchar2(I5),
            "COLLECTION"varchar2(60),
            "ID_LIEUEDIT"number(7),
            "CODAT_DEB"number(3),
            "JM_DEB"varchar2(15),
```

```
            "ANNEE_DEB"varchar2(5),
            "DELTA_DEB"number(3),
            "DATE_DEB"number(8),
            "CODAT_FIN"number(3),
            "JM_FIN"varchar2(15),
            "ANNEE_FIN"varchar2(5),
            "DELTA_FIN"number(3),
            "DATE_FIN"number(8),
            "CODE_ILLUST"varchar2(3),
            "ID_FONDS"number(8),
            "ID_COLLECTION"number(8),
            "RAMEAU"varchar2(250),
            "NB_PAGES"varchar2(30),
            "CARTES"varchar2(15),
            "IND_BIB"varchar2(15),
            "BIBLIO"varchar2(15),
            "ID_EDITE"number(7),
            "EDIT"varchar2(40),
            "DTCRE" date,
            "DTMOD" date,
            "DTTXT" date,
            "DTINT"date,
            "FLTXT"varchar2(1),
            "FLINT"varchar2(1),
            "FLSUP"varchar2(1))
/(...)
```

Le second script modifie la définition des tables en ajoutant les contraintes *foreign key*. Extrait du script de modification :

```
set term on
set feed on
set echo on
spool genfk.lis
alter table "ACTION" add (
            constraint action_id_attrib_fk  foreign key (ID_ATTRIB)
              references "ATTRIB"(ID_ATTRIB),
            constraint action_id_periode_fk  foreign key (ID_PERIODE)
              references "PERIODE"(ID_PERIODE),
            constraint action_code_legal_fk foreign key (CODE_LEGAL)
              references "TYPESERIE" (CODE))
/
alter table "ARCHIV" add (
            constraint archiv_id_arch_fk foreign key (ID_ARCH)
              references "COTES"(NUMCOTE),
            constraint archiv_id_typo_fk foreign key (ID_TYPO)
              references "TYPO"(ID_TYPO),
            constraint archiv_id_classif_fk foreign key (ID_CLASSIF)
              references "CLASSIF"(ID_CLASSIF),
            constraint archiv_id_prod_fk foreign key (ID_PROD)
              references "PRODUCTEUR"(ID_PROD),
            constraint archiv_id_service_fk foreign key (ID_SERVICE)
              references "SERVICE"(ID_SERVICE),
            constraint archiv_id_acteur_fk foreign key (ID_ACTEUR)
              references "ACTEUR" (ID_ACTEUR),
            constraint archiv_id_dictexte_fk foreign key (ID_DICTEXTE)
              references "DICTEXTE" (ID_DICTEXTE),
            constraint archiv_id_attrib_fk foreign key (ID_ATTRIB)
```

```
                references "ATTRIB" (ID_ATTRIB),
              constraint archiv_id_action_fk foreign key (ID_ACTION)
                references "ACTION" (ID_ACTION),
                 constraint archiv_id_lieuedit_fk foreign key (ID_LIEUEDIT)
                references "LIEUEDIT" (ID_LIEUEDIT),
              constraint archiv_id_fonds_fk foreign key (ID_FONDS)
                references "FONDS" (ID_FONDS),
              constraint archiv_id_collection_fk foreign key(ID_COLLECTION)
                references "COLLECTION" (ID_COLLECTION),
              constraint archiv_id_edite_fk foreign key (ID_EDITE)
                references "EDITE" (ID_EDITE))
/
alter table "ARTICLE" add (
              constraint article_numcote_fk foreign key (NUMCOTE)
                references "COTES" (NUMCOTE),
              constraint article_id_prod_fk foreign key (ID_PROD)
                references "PRODUCTEUR" (ID_PROD),
              constraint article_id_typo_fk foreign key (ID_TYPO)
                references "TYPO" (ID_TYPO))
/(...)
```

Ces deux scripts ont été lancés l'un après l'autre. Il était en effet impossible de procéder avec le seul script de départ, car inévitablement certaines clés étrangères auraient été référencées alors même que les tables concernées n'existaient pas encore.

Problèmes rencontrés

Les deux scripts ont dû être lancés plusieurs fois car de nombreuses erreurs ont perturbé le bon déroulement des opérations. Les erreurs de frappe tout d'abord ont rendu incohérentes certaines parties du code et ont empêché la création de quelques tables. Les erreurs plus complexes ont concerné le programme de mise à jour des tables introduisant les clés étrangères. Certaines d'entre elles renvoyaient vers des colonnes qui n'étaient ni clé primaire, ni clé unique, d'autres étaient mises en relation avec des colonnes de formats différents. Ainsi, une colonne de type caractère renvoyait sur une colonne de type numérique, et vice et versa. Ou encore les liens portaient sur des colonnes de même format, mais de longueur différente. A chaque passage du script, de nouvelles erreurs sont apparues révélant les incohérences du programme de création.

Corrections apportées

Les deux catégories d'erreurs identifiées ont entraîné de nombreuses modifications dans le script de création :

- introduction de nouvelles clés primaires comme par exemple `service.id_service`, `tutelle.id_tutelle`,
- effacement de quelques colonnes redondantes ayant anciennement joué le rôle de clé primaire comme `typ_ent.id_typ_ent`, `cat_prod.id_typ_prod`. Pour ces deux tables, les colonnes '`code`' jouaient en fait le rôle de la clé primaire. Elles ont donc été transformées en clé primaire.
- modification du format de la colonne *varchar2* au lieu de *number* ou le contraire,
- allongement de la longueur de la colonne.

Toutes ces modifications ont obligé systématiquement à une suppression de toutes les tables concernées. La commande `drop table <nom_table> cascade constraint` m'a permis de supprimer les tables, '`cascade constraint`' permettant de passer outre les contraintes de clés étrangères. Les mises à jour ont été effectuées en modifiant le script de création original de sorte que ce dernier corresponde exactement à la structure de la base de test.

Les changements effectués dans le script de modification des tables ont essentiellement consisté à remplacer les références aux colonnes non clés primaires ou uniques par des références à des colonnes clés primaires ou uniques.

9.3 Le début d'un référentiel sur GAIA

Le fichier lsttable.xls est, en 1996, le seul document à jour que possède le service informatique des Archives sur la structure de GAIA. Les autres documents papier, s'ils ont le mérite d'exister, commencent à dater et sont malheureusement inexploitables pour la maintenance de l'application. Les modifications apportées à la structure de la base et aux programmes n'ont pas été enregistrées systématiquement. Cette absence de référentiel commence à poser certains problèmes, notamment lorsqu'il s'agit d'apporter des corrections à l'application.

Aussi un premier effort a-t-il été fait pour essayer de combler cette lacune. Toutes les tables de GAIA ont été recensées et répertoriées dans ce fichier Excel avec quelques éléments de description : le nom de la table, sa définition, le nom de toutes les colonnes, leur signification, leur type, leur longueur et les contraintes *not null*. Cette liste des tables m'a été très utile pour faire connaissance avec GAIA. Cependant sa présentation rendait très difficile tout ajout de nouvelles informations. J'ai donc transformé la structure de cette liste de façon à la gérer comme une petite base de données. J'ai utilisé le module de gestion de données d'Excel qui offre des facilités d'affichage, d'ajout, de recherche et de suppression de données, très pratique dans le cas des listes longues comme celle des tables de GAIA (quelque mille six cent soixante et onze lignes !).

Extrait du référentiel sur GAIA :

TABLE	CONTENU	CHAMP	TYPE	LGR	NOT NULL	CLE ETRANGERE	DOMAINE DE DEFINITION
TABLE	CONTENU	CHAMP	TYPE	LGR	NOT NULL	CLE ETRANGERE	DOMAINE DE DEFINITION
ABONNE	Type d'abonnement	ID_ABONNE	NUMBER	7	NOT NULL		
ABONNE		ABONNE	VARCHAR2	40	NOT NULL		
ABONNE		TYPE	VARCHAR2	2	NOT NULL		
ACTEUR	Acteurs	ID_ACTEUR	NUMBER	7	NOT NULL		
ACTEUR		LIBELLE	VARCHAR2	100	NOT NULL		
ACTION	Actions administratives	ID_ACTION	NUMBER	7	NOT NULL		
ACTION		ID_ATTRIB	NUMBER	7		ATTRIB. id_attrib	ATTRIB

Les propriétés des tables ont été disposées en colonnes. J'ai ainsi pu rajouter d'autres informations telles que les clés primaires, les clés étrangères et les domaines très facilement grâce au masque de saisie proposé par défaut.

L'autre intérêt offert par la gestion des données dans Excel concerne les possibilités de recherche. L'utilitaire de recherche, le filtre élaboré personnel, permet l'exécution de requêtes avec les opérateurs de comparaison 'ET' et 'OU'. Jusqu'à deux critères de recherche peuvent être posés. J'ai ainsi pu répondre très facilement à des demandes d'édition ponctuelles sur certaines parties de la structure de GAIA.

Pour la liste de toutes les colonnes de la table des entrées, j'ai demandé l'affichage de toutes les lignes dont la table était égale à 'ENTREE'. La liste des tables, sans les tables de jointures, a été éditée en procédant à l'extraction de toutes les lignes dont le contenu était différent de 'LIENS*' ET 'ASSOCIATIONS*'. Cette dernière extraction a toutefois montré les limites du système. La recherche se faisant uniquement sur le texte de description du contenu, toutes les tables de jointure dont le contenu ne comprenait pas les termes de recherche ont été retenues. Il serait donc utile d'établir des normes de description.

Cette 'métabase' pourrait être améliorée par l'écriture de macro-commandes. Celles-ci permettraient des éditions avec mise en page. On peut toutefois se demander si cela se justifie. *Designer 2000*, avec son dictionnaire conceptuel, véritable dossier informatisé sur l'application, devrait prochainement prendre le relais et fournir un référentiel exhaustif.

10. CONCLUSION

Le compte rendu de l'analyse conceptuelle du module de la gestion des entrées est publié dans ce rapport. Il sera soumis à l'approbation des utilisateurs de GAIA qui devront juger de l'utilité de la démarche pour les développements futurs. J'espère que son contenu est suffisamment clair et suffit à prouver l'intérêt qu'il y a de revoir l'ensemble de l'application avec les outils de conception expérimentés.

Custom Development Method propose une démarche d'analyse rigoureuse et organisée. De ce fait, elle est l'outil approprié pour revoir un système qui, depuis 1987, a considérablement grandi. De plus, les développements dont il a été l'objet, ont parfois été effectués dans l'urgence. Ils ont abouti à certaines incohérences et rendent sa maintenance quelquefois difficile. Si l'application de la méthode à la gestion des entrées a révélé certaines déficiences, elle a surtout fait entrevoir des solutions pour y remédier. La remise en cause de GAIA selon la méthode Oracle ne peut qu'être bénéfique.

Designer 2000 est un logiciel très puissant. On peut regretter ses exigences au niveau configuration matérielle, sa sensibilité à l'environnement extérieur et la complexité de certains modules, mais on ne peut nier tout ce qu'il peut apporter à l'évolution de GAIA :

- un référentiel à jour et complet, comprenant tout à la fois les niveaux conceptuel et logique, qui facilitera la maintenance corrective et évolutive de l'application.
- des facilités de conception et de développement : un même modèle de données et de traitements peut servir à la création de plusieurs applications, des scripts peuvent être générés automatiquement à partir des définitions présentes dans le dictionnaire conceptuel, de standards interface/homme machine peuvent être facilement implémentées grâce aux modèles fournis par défaut.

Les apports de ces outils seront d'autant plus grands s'ils sont soutenus par une véritable politique de gestion des développements. Le personnel doit se former. L'effort pour développer la base de test doit être maintenu pour y explorer les nouvelles fonctionnalités de la version 7 d'Oracle et en tirer tous les bénéfices.

Annexes

Annexe A Bibliographie

SMIME H., *Concevoir et développer avec ORACLE et CASE* : Oracle versions 6 et 7. Paris : Eyrolles, 1994. 268 p.

ABDELLATIF A., LIMAME M., ZEROUAL A., *Oracle 7: langages, architecture, administration.* Paris : Eyrolles, 1995. 464 p.

STURNER G., *ORACLE 7: A User's and Developer's guide.* Londres : International Thomson Publishing Company, 1995. 424 p.

ACKER B, *The promise of Oracle Developper/2000.* Data based advisor, 14, 3, march 1996, p. 36-40.

SLOTNICK D, *Field report: Oracle tools update.* Database programming & design, 9, 4, april 1996, p. 71.

RICCIARDI S, *Database design: redundancy and normalization.* PC magazine, 25, january 1994, p. 285-287.

ORACLE, *Custom development: requirements modelling using Designer/2000.* Standards and guidelines, vol. 1, release 2.00. Redwood shores : Oracle corporation, 1996. 230 p.

ORALCE, *Oracle designer/2000: product overview, release 1.* Redwood Shores : Oracle corporation, 1995. 68 p.

ORALCE, *Oracle designer/2000: a guide to systems modelling, release 1.* Redwood Shores : Oracle corporation, 1995. 214 p.

ORALCE, *Oracle designer/2000: tutorial, release 1.2.* Redwood Shores : Oracle corporation, 1995. 151 p.

Annexe B Extrait du manuel utilisateur de GAIA, partie théorique sur la gestion des entrées

1.3 Enregistrement d'une nouvelle entrée / versement

Introduction

Il existe un registre d'entrée spécifique à chaque type de documents : archives, images.

1.3.1 TYPE LEGAL ET TYPE DE TRAITEMENT

cf. analyse détaillée en annexe.

L'agrémentation des séries au sein de la table des séries est un préalable obligatoire. Elle est effectuée par l'administrateur du système. Elle comporte la création des caractères spécifiques à la série (ex. : T), ainsi que l'indication du type légal du document et de la période historique concernée pour les documents d'archives.

Cette sélection permet de spécialiser les procédures ainsi que les dictionnaires d'indexation.

L'ANOMALIE

Toutefois, certaines séries ne se prêtent pas à une classification aussi simple, soit qu'elles comportent différents types de documents (ex. : série FI), soit qu'elles accueillent des documents d'archives de toutes époques (ex. : série E).

La série est alors déclarée en "anomalie" pour l'un ou l'autre de ces deux critères, à charge d'en spécifier ultérieurement le type légal et éventuellement la datation, soit dès le registre d'entrée (facultatif), soit lors de la description des documents (obligatoire).

Gaia propose trois types de traitement qui sont : archives, images, et publications.

Les différentes procédures de description des documents sont donc indépendantes du type légal des documents.

Gaia considère seulement a priori qu'un document sera décrit selon le type de traitement qui correspondant au type légal de la série. Toutefois, l'utilisateur peut choisir un type de traitement.

Si un document appartient à une série agrémentée "Archives", il est donc possible de choisir ensuite un "type de traitement" différent à l'écran d'accueil (menu initial : "description des documents"). Ce choix permet d'accéder aux écrans de description des images par exemple, ce qui n'interférera pas sur le type légal.

Par ailleurs, la description d'une cote d'archives en procédure "image" peut ensuite être poursuivie en procédure archives. Toutefois, la cote doit obligatoirement être rattachée à un seul fonds ou (collection).

1.3.2 CATEGORIE DE PRODUCTEUR ET TYPE D'ENTREE

La description des entrées doit comporter obligatoirement , outre le type légal des documents et leur période éventuelle, les informations suivantes :
- type d'entrée
- catégorie de producteur

Ces informations sont utilisées notamment pour l'édition du rapport annuel et des registres d'entrée par catégorie.

LA CATEGORIE DE PRODUCTEUR :

La catégorie de producteur EVE est utilisable pour les seules archives privées.

Les archives publiques, entrées par voie extraordinaire doivent être indexées, selon leur statut légal (versement, dépôt), par les catégories de producteurs suivantes : PAD (producteur administratif), NOT (notaire), HOP (hôpitaux), DAC (communes) puis par leur type d'entrée effectif (achat, don, restitution...).

A priori, en l'état actuel, la catégorie PAD reçoit toutes les archives publiques hors notaires, hôpitaux et communes.

Dès lors que cette distinction, archives privées/archives publiques/type d'entrée, est clairement effectuée sur une année civile complète, la table des types d'entrée peut être "apurée" des valeurs "en double" (ex. dépôt d'archives publiques, dépôt d'archives privées) qui n'étaient destinées qu'à gérer la transition.

ENCHAINEMENT DES PROCEDURES VERS LA DESCRIPTION DES ARTICLES COTES : documents d'archives.

L'intégration du registre d'entrée avec les procédures de description évite à l'utilisateur de saisir certaines informations une première fois au registre d'entrée et une seconde fois aux écrans de description des documents.

Si vous souhaitez poursuivre la description des articles après qu'ils aient été décrits au registre d'entrée, vous pouvez le faire à partir du même registre, soit immédiatement, soit que vous repreniez la saisie ultérieurement, après cotation par exemple.

Le passage aux écrans de description des documents vous permet de "rapatrier" les informations déjà créées au registre d'entrée et de les valider pour la description des articles, sans avoir à saisir à nouveau ces informations.

1 - sommaire des entrées (facultatif)
2 - registre d'entrée
3 - écran "crecote3", traitement par cote,
4 - ou "creflot3", traitement par groupe de cotes.

A ces deux écrans, la fonction <AIDE> permet de rapatrier les informations saisies au registre d'entrée. Vous pouvez supprimer ou modifier une de ces informations puis compléter votre saisie (code comm...) avant de valider.

Toutes les informations de la table des cotes y compris la cote elle-même peuvent être créées à ces deux écrans.

Si ces dernières existent déjà dans la table des cotes, vous pouvez les interroger cote par cote à l'écran CRECOTE3, et les mettre à jour. En revanche, l'écran "creflot 3" ne permet pas l'interrogation. Il n'est donc pas possible de venir y modifier un groupe de cotes déjà créé.

Après validation, vous pouvez passer aux écrans de description documentaire :
- de "créflot 3" vers "repert 1" (niveau 2)
- de "crecote 3" vers "archives 1" (niveau 3)

1.3.3 LE "SOMMAIRE" DU REGISTRE D'ENTREE

- Le sommaire est valable pour tous les types de documents. Le choix du type peut être indiqué à l'écran.

Il permet de sélectionner une ou plusieurs entrées puis de les visualiser et d'accéder aux écrans de saisie du registre d'entrée.

1.3.4 LES "ENTREES TEMPORAIRES"

Les "entrées temporaires" concernent les documents qui n'appartiennent pas au service mais qui y entrent pour communication, microfilmage, exposition... Ces procédures complètent donc les programmes de communication pour la gestion des communications de documents en provenance d'autres services d'archives.

Ces "entrées" sont considérées comme temporaires, puisque les documents doivent être restitués, à terme, à leurs "propriétaires".

Ces documents n'étant, par définition, pas cotés au sein du service, la série EXT et la cote "1OO EXT" (munie du code de comm. "incomplet") doivent être créées de manière à autoriser ensuite la gestion de l'entrée, puis de la communication éventuelle en salle de lecture, sur "cote interne".

La création d'une série spécifique permet de distinguer ensuite plus facilement les opérations effectuées sur ce type de documents (statistiques...).

1.3.5 LES PREVISIONS D'ENTREE

- le bulletinage des périodiques

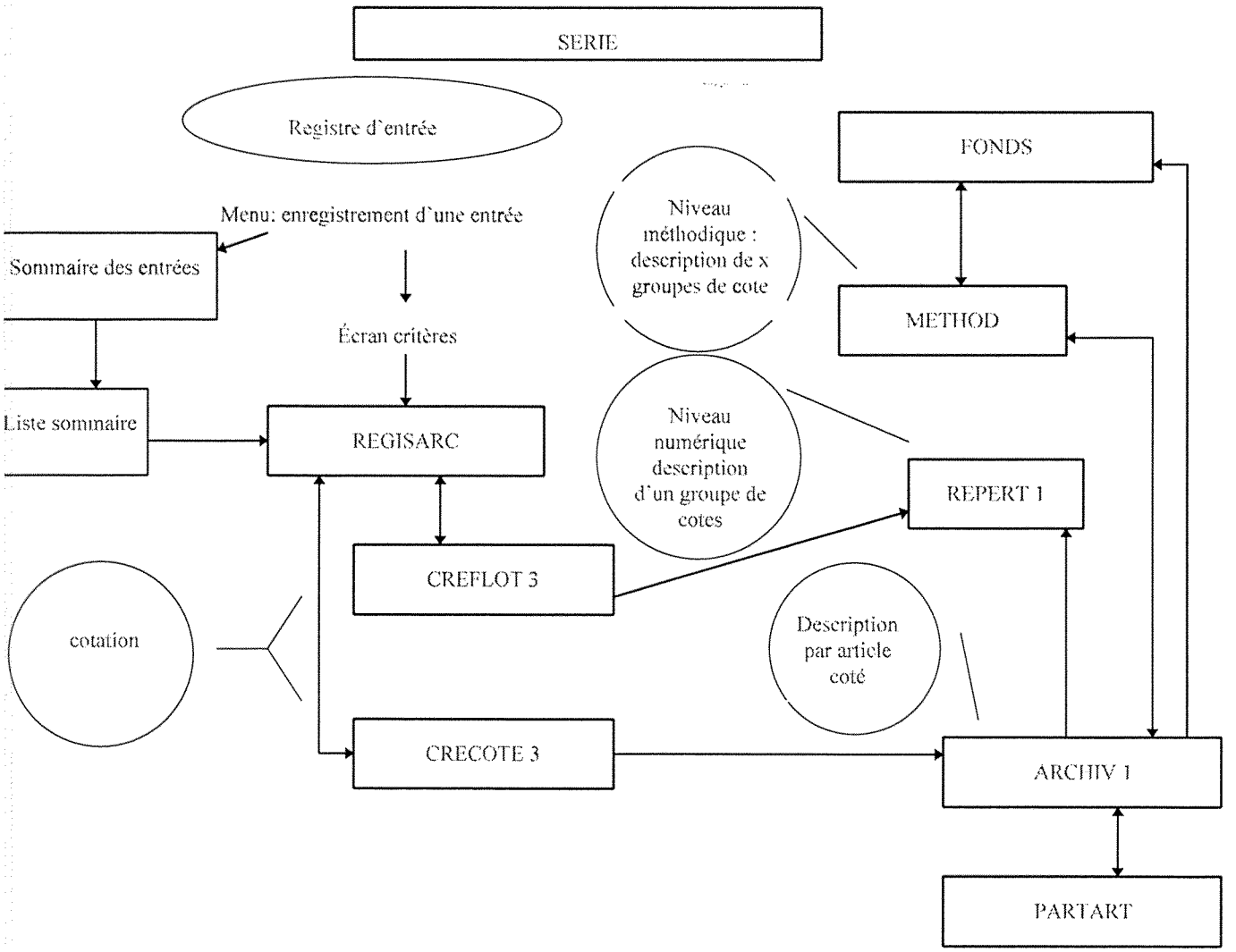
La collation des périodiques est effectuée, selon les cas, au niveau "publi-collection" pour les collections non cotées à l'article, et au niveau "bibli-article coté" pour les collections cotées à l'article. ("état de collection")

* A l'étude :

- la gestion des commandes d'ouvrages (cf. catégorie de producteur ED)
- la préparation des versements d'archives
- les relances automatiques pour le bulletinage.

1.3.6 LE REGISTRE D'ENTREE

ENTREE ET DESCRIPTION DOCUMENTAIRE, COTATION, ENCHAINEMENT D'ECRANS



- description de l'entrée au registre d'entrée
- réservation éventuelle d'une cotation pour les entrées en "séries continues".

Le registre d'entrée est exploité pour les éditions relatives au rapport annuel d'activité.

Il permet de préparer l'état des versements contemporains

GAIA REGISTRE D'ENTREE DOCUMENTS D'ARCHIVES REGISARC
N° : 93 1274 type légal : DOCUMENTS D'ARCHIVES période: époque contemporaine Cat.Producteur : PAD : Producteur : CDIF centre des impôts fonciers de Melun : service rattaché Id Fonds : 618 fonds des services chargés du cadastre propriétaire Date entrée : 03-FEB-1992 Type entrée: VER localisation : AD-6-C-123-A-7
Attribution automatique d'une cotation : entrer vos critères et demander <EDIT> sur la zone d'attribution : ___3__ P___ _____ _____
nbre art. : 618 métrage de l'entrée
commentaire : cons.définitive/élim.partielle/élim.totale
Sommaire : description
période : Cod
edate : 3 Jour/mois déb. année déb.1824 delta Codedate : 3 Jour/mois fin année fin.1845 Cotation C +(menu bloc) Accès documentaire

NUMERO D'ENTREE

numéro d'entrée attribué automatiquement par GAIA, il est structuré sur 10 caractères numériques dont les quatre premiers correspondent à l'année (1992...).

TYPE LEGAL

zone renseignée, à cet écran, automatiquement par GAIA à partir de l'écran précédent.

PERIODE

cette zone est complétée automatiquement à partir de l'écran précédent.

CATEGORIE DE PRODUCTEUR

- PAD : producteur administratif
- DAC : dépôt d'archives communales
- HOP : dépôt d'archives hospitalières
- NOT : versement de minutes notariales
- EVE : archives privées
- ED : éditeur distributeur (achat sur abonnement, fiche de gestion du bulletinage)

SIGLE DU PRODUCTEUR

la zone sigle est essentiellement conçue pour les archives contemporaines.

PRODUCTEUR

- la saisie du producteur provoque l'affichage automatique du fonds.

SERVICES RATTACHES

FONDS

Le fonds est sélectionné si l'on ne souhaite pas indiquer de producteur, sinon il sera automatiquement par la validation du producteur (cf. ci-dessus).

PROPRIETAIRE

S'il s'agit d'un service versant "accidentel", l'indiquer en "sommaire" et non dans la zone "propriétaire".

DATE D'ENTREE

GAIA propose la date du jour que vous pouvez modifier le cas échéant.

TYPE D'ENTREE

versement, dépôt, achat, abonnement, don, legs, dation...

Certains types d'entrée ont été rajoutés provisoirement de manière à effectuer une distinction systématique entre archives privées et publiques. (cf. ci-dessus paragraphe B "catégorie de producteur").

LOCALISATION

Il s'agit souvent d'une localisation temporaire en bureau ou en salle de tri. La localisation définitive pourra être indiquée lors de la création de la cote.

ATTRIBUTION AUTOMATIQUE D'UNE COTATION

L'attribution automatique d'un numéro de versement (cote partielle) est utilisée pour les séries, sous-séries ou fonds où les entrées sont ordonnées selon un ordre numérique croissant.

Cette procédure n'aboutit pas à une création de cote. Elle provoque la réservation d'une cote ou cote partielle, (fonds, n° de versement) de manière à ne pas attribuer le même N° au versement suivant.

NOMBRE D'ARTICLES

Il est destinée au seul registre d'entrée et à l'établissement du rapport annuel, le cas échéant.

METRAGE

Métrage de l'entrée. Il est destiné également au rapport annuel. Cette zone est spécifique aux documents d'archives.

COMMENTAIRES

indiquer le sort final des documents:

- cons.définitive
- élim.partielle
- élim.totale

SOMMAIRE (de l'entrée ou du versement)

Utilisation à déterminer

«post it» analyse temporaire en cas d'urgence.

DESCRIPTION

cette zone est destinée à recevoir l'analyse normalisée du versement qui sera éditée ensuite pour constituer l'état des versements contemporains. La description d'un versement est effectuée soit globalement, soit par unité documentaire. Dans ce cas, il est saisi une analyse par «unité documentaire».
cf. circulaire AD 94.8

la structure de l'analyse, qu'elle soit unique ou multiple, est la suivante:

intitulé global (ou objet principal) . --- objet, sous-objet (nom commun ou nom propre) ; attribution, action : typologie du document (type de classement, dates)
groupes de cotes concernées. délai de communicabilité

exemple: recensement général agricole (classement alphabétique par commune, 1970, 1979, 1980)
3160 W 1 à 106. librement communicable

Compte tenu des attributions du service «statistique agricole», l'analyse est réduite à :
typologie du document (type de classement, date)

exemple: structures agricoles, terres labourables : questionnaires et fiches d'enquête (1968-1984)
3160 W 107 à 132. librement communicable

objet, autre objet : typologie des documents (dates)

CODE DATE

ces zones permettent la saisie de tous les types de format de dates.

Vous pouvez choisir un "code date début" différent du "code date fin", pour indiquer par exemple une date début avec un "delta" et une date fin complète.

1.3.6.1 Cotation des articles et indication des critères de conservation (DUA...) et de communication. aux écrans CREFLOT et CRECOTE

Les informations saisies lors de l'entrée sont récupérées à cet écran.

Cette procédure peut être effectuée par cote ou par groupe de cotes.

```
GAIA      *** TRAITEMENT D'UNE COTE ***   CRECOTE3

COTE      ____3        P_        36____        /1_____________________________

No entrée :                                   Sommaire entrée :
1992_01716                                    plans parcellaires
TYPE LEGAL                DOCUMENTS D'ARCHIVES_____________
PERIODE :                 ANOMALIE_____________________________
LOCALISA AD0X0X0 DATE DEBUT__ __ ____ DATE DE FIN __ __1837
COMMUNICABIL.cod._ _____délai com.___date de com.__ _____
OBSERV. cod._ _____date obs.__ __ ____Reproduc._ _ _____
ENTREE date _1 12 1992                        Réf réglem. _____
ELIM.délai___tri_date élimin.__ __ ____Accord_ le__ __ ____
TRANSF délai __date transf.__ __ ____ D.U.A ___ critère _
Producteur : PAD _CDIF_centre des impôts fonciers de Melun__
Fonds : _618____ fonds des services chargés du cadastre
Service:  Propriétaire : CDIF centre des impôts fonciers de Melun
```

CODE DE COMMUNICABILITÉ

* le code A: « communications administratives » interdit le communication des documents par les programmes de communications publiques (salle de lecture). Il est utilisé pour gérer « en masse » des articles dont on veut réserver la communication aux administrations. Généralement, on l'utilise pour indexer l'arriéré des versements pour lesquels, à défaut d'indexation du producteur et du propriétaire, le code S « réservée au service propriétaire » ne peut être utilisé.

= Le document sera communicable à tous les propriétaires, sans contrôle.

* le code S: « réservée au service propriétaire »
= le document ne sera communicable qu'au seul service propriétaire.

code L: ne l'utiliser que s'il n'y a pas de délai de communicabilité de manière à indiquer que la communicabilité a été traitée.

DÉLAI DE COMMUNICABILITÉ

= délai réglementaire de communication au public . Cette information est différente des codes de communicabilité ci-dessus qui constituent des procédures dérogatoires au délai légal dans le cadre des communications administratives

RÉFÉRENCE RÉGLEMENTAIRE

indiquer soit:

- le numéro informatique du tableau de traitement. Dans ce cas, la saisie des zones DUA et CRITERE est inutile.

- à défaut de tableau de traitement, la référence de la circulaire pour tris et éliminations.

DÉLAI D'ÉLIMINATION

indiquer le délai après lequel des éliminations doivent être effectuées.

- soit élimination totale de l'article coté
- soit, s'il s'agit d'éliminations partielles (tri), indiquez-le en zone de tri .

remarque :
- ne pas utiliser cette zone dans le cas de délai d'élimination s'appuyant sur d'autres critères que la date-fin du document. (ex: 21 ans à compter de la date de naissance.). En effet le calcul automatique de la date d'élimination serait faux.

Dans ce cas, indiquez directement la date d'élimination en zone de « date d'élimination » sans mettre de délai. A défaut, complétez la DUA en indiquant une * dans la zone critère.
Ces informations doivent être explicitées dans les tableaux de traitement.

TRI

cette zone doit être utilisée en cas d'élimination partielle d'un article. Dans ce cas, la procédure automatique d'indexation du pilon ne s'effectuera pas sur l'article.
code 1: tri à effectuer
code 2: tri effectué.

Lors de procédure automatique d'élimination, les cotes indexées du code 1 «tri à effectuer» seront signalées.

(DATE D'ÉLIMIN(ATION)

elle est calculée automatiquement à partir de la date-fin de l'article + délai d'élimination + 1

Par convention, deux dates factices sont utilisées pour :

- signaler les documents en conservation perpétuelle par la saisie de l'année 9999. de la sorte, ils ne sont pas confondus avec les documents dont l'élimination n'a pas été traitée.

- signaler, en saisissant l'année 1111., les traitements «incohérents» effectués sur les bordereaux qui devront être réexaminés.

Annexe C Extrait du manuel d'utilisation de GAIA, partie pratique : la gestion des entrées

I - L'ENTRÉE DES DOCUMENTS

* menu général : "gestion des entrées"

```
        Gestion des entrees

1_    Enregistrement d'une entree_____

2_    Sommaire des entrees_____
3_    Previsions d'entrees temporaires_____
4_    Sommaire documents temporaires_____
5_    Motifs d'entrees temporaires_____
6_    Organismes exterieurs_____

__    _____

            TAPEZ VOTRE CHOIX :
```

* enregistrement d'une entrée : accès à l'écran de sélection de critères puis aux différents écrans permettant de créer une entrée.

* sommaire des entrées : interrogation des entrées quelque soit le type de document.

Les "entrées temporaires" concernent les documents qui n'appartiennent pas au service mais qui y entrent pour communication, microfilmage, exposition...

* prévision d'entrée temporaire : saisie de la demande du document auprès de l'organisme propriétaire.

* sommaire documents temporaires : permet de visualiser le "registre" de ce type d'entrée pour savoir où en est la procédure.

* motifs d'entrées temporaire : table permettant de créer les différents motifs dont il a été fait état plus haut.

* organismes extérieurs. Table des organismes, identique à celle qui est utilisée pour les communications à des organismes extérieurs.

A. Table des séries (voir TDOC)

B. Catégorie de producteur et type d'entrée (voir TDOC)

C. Enchaînement des procédures (voir TDOC)

D. LE "SOMMAIRE" DU REGISTRE D'ENTRÉE

- Le sommaire est valable pour tous les types de documents. Le choix du type peut être indiqué à l'écran.

Il permet de sélectionner une ou plusieurs entrées puis de la visualiser et d'accéder aux écrans de saisie du registre d'entrée.

Cet écran est programmé pour l'interrogation, il est donc inutile de passer en mode "query"

```
GAIA                    SOMMAIRE DES ENTREES                    SOENTREE

   N° entrée :

   Date entrée :

   Type légal :                Type d'entrée :

   Cat. prod. :                 Producteur :

   Cotation :

   Localisation :

   Titre :
```

- saisissez directement vos critères de sélection

- fonction <AIDE>

```
GAIA                    SOMMAIRE DES ENTREES                    SOENTREE

N° entrée        code          Date          code          localisation
                                trait                entrée          légal
1991-00076                FI                  29-oct-91                FI
cotation :
1991-00076                BI                  21-nov-91                FI

1991-00026              BI                  20-sept-91

1991-00023              BI                  15-oct-60

1991-00022              BI                  31-oct-90

              APPUYER SUR <AIDE> POUR LE REGISTRE D'ENTREE
```

- positionnez le curseur sur l'entrée que vous recherchez, fonction <ENREGISTREMENT SUIVANT>, et effectuez la fonction <AIDE> pour accéder au détail de l'entrée recherchée.

2 - LES "ENTREES TEMPORAIRES"

A l'écran de sélection des critères ("critère", cf. ci-dessous, saisissez "O" pour accéder à ces programmes, puis "ext" dans la zone série.

```
GAIA        CRITERES DE SELECTION DES DOCUMENTS        CRITERE

     Entrée    temporaire                  <O/N>      :        N

     Indiquez     une     série                <Liste>     :     P

     Type     légal     du     Doc.     <Liste>     :     Archives

     Datation (période) <Liste> : Archives modernes

     Type     de     traitement          <Liste>       :     Images

     TAPER  SUR  <AIDE>  POUR  ACCEDER  AU  REGISTRE  D'ENTREE
```

A. Demande d'entrée temporaire

- au menu sélectionnez : "prévision d'entrée temporaire".

La table des motifs comporte actuellement les valeurs suivantes : communication, microfilmage, exposition, prêt, reliure-restauration. Elle peut être mise à jour et complétée.

```
GAIA              DEMANDE ENTREE TEMPORAIRE                TEMP

Date demande : 11/11/1992 Accuse de reception : nbre art :
Categorie : EXT                  Proprietaire       :    AD   45
N° du lecteur : 410              Nom        :              DELIVRE
                    Motif : COMMUNICATION
     Date de retour                  Accuse   de   reception   :
```

```
COTE  INTERNE/DESCRIP.  DISPO   DATE ENTREE N° ENTRE TYPE LEGAL
2 MI 165 . 175          NON
2 MI 188 . 204          NON
```

A cet écran doivent être enregistrées les demandes préalablement à l'arrivée du document. Lorsque le document arrivera, l'entrée sera saisie à l'écran "entrée temporaire" où l'on accède au menu par "enregistrement d'une entrée".

B. "enregistrement d'une entrée temporaire"

```
GAIA                        ENTREE TEMPORAIRE              REGISTEM

Date demande :11/11/1992    Accuse de reception : nbre art.    :
Categorie   :  EXT                              Proprietaire : AD  45
 N° du lecteur : 22          DURANT :
  Motif : COMMUNICATION
  Date de retour :                          Accusé de réception   :

                    ENREGISTREMENT D'UNE ENTREE ?

    Date entrée :              Type légal :

COTE  INTERNE/DESCRIPTION   DISPO   N° ENTREE        TYPE LEGAL
556 a                        OUI    1992-00192 DOCUMENTS D'ARCH
556 b                        OUI    1992-00193 DOCUMENTS FIGURE
556 c                   NON
556 d                        OUI    1992-00192 DOCUMENTS D'ARCH
```

CATÉGORIE : EXT

PROPRIÉTAIRE :

- nom de l'organisme prêteur.

N° DE LECTEUR : pour lequel a été demandée l'entrée, s'il y a lieu.

COTE INTERNE/DESCRIPTION :

cette zone permet d'indiquer soit la cote du document si elle est connue, soit sa description sommaire.

ENREGISTREMENT D'UNE ENTRÉE :

cette zone permet d'indiquer si un document qui a été demandé, est disponible ou non. La mention "NON" s'affiche automatiquement dès la demande de document. Elle doit être mise à jour manuellement ("oui"), dès lors que le document est arrivé et que vous avez demandé l'enregistrement de l'entrée (même écran).

C. Interrogation des entrées temporaires.

menu "sommaire documents temporaires".

```
GAIA        SOMMAIRE DOCUMENT TEMPORAIRE          SOMMAIR

INTERROGATION SUIVANT LA COTE INTERNE/DESCR.


N° lecteur      Propriétaire          Date demande
    783         AN CANADA             15/06/1992
    410         AD 45                 11/11/1992
    783         AN CANADA             29/09/1992
                AD CHARENTE           10/10/1992
   2796         AD NIEVRE             12/10/1992
   1690         AD NORD               10/10/1992
   1145         AD NORD               10/09/1992
      3         MARTIGNY(Mr. J-P.DE)  26/10/1992

     POUR ACCEDER AUX COTES  <BLOC SUIVANT>
```

Vous pouvez choisir d'interroger selon la cote interne, c'est-à-dire la cote d'origine du document ou sa description.

Vous pouvez également passez aux zones suivantes <BLOC SUIVANT> pour interroger sur d'autres critères.

```
GAIA        SOMMAIRE DOCUMENT TEMPORAIRE          SOMMAIR

INTERROGATION SUIVANT LA COTE INTERNE/DESCR.


N° lecteur          Propriétaire          Date demande

COTE   INTERNE/DESCRIPTION   DISPO   N°   ENTREE   DATE   ENTREE

RG 24 E7 bob cl12437 histor de  OUI  1992  01500  05/11/1992
RG 24 E7 bob cl12436            OUI  1992  01500  05/11/1992


          POUR ACCEDER AUX COTES <BLOC SUIVANT>
```

<AIDE> = ACCES A LA DEMANDE/ENTREE TEMPORAIRE

Annexe D Sauvegardes de la maquette

Les travaux effectués chaque jour dans *Designer 2000* ont été sauvegardés quotidiennement par un export. Ce dernier est une copie de toute l'application. Un répertoire a été créé spécialement à cet effet sur l'ordinateur de travail :

c:\database\aix_gaia2\expdes2k à l'intérieur duquel des sous-répertoires correspondant aux différents jours de la semaine ont été créés. Ainsi, le fichier export du 29 août a été copié dans le répertoire GAIA29_8.

Manipulations à effectuer :

1. dans le R*epository object navigator*, dérouler le menu *Application* et choisir l'option *Archive*.
2. sélectionner l'application à archiver et donner un nom aux *roll-back segments*[1] qui peut être le même pour tous les exports (ex : DES2_RS).
3. exporter l'application en donnant un nom au fichier d'export, dont l'extension est .dmp, et en indiquant le chemin d'accès du répertoire d'accueil.

[1] Les *roll back segments* sont des objets de la base de données. Ils enregistrent les actions qui devraient être annulées lors de circonstances particulières, par exemple, lors d'annulations de transaction (*transaction rollback*).

Annexe E Remarques relatives à la modélisation d'un projet

La méthode d'analyse à mettre en oeuvre pour une maquette et un projet sont sensiblement les mêmes. Ce qui a été fait pour la gestion de entrées, est aussi valable à l'échelle d'un projet à quelques différences près. Si l'on veut aller plus loin dans l'analyse, on peut se reporter au plan de travail type pour l'élaboration d'un projet, à l'annexe suivante. Cependant, quelques points importants méritent d'être soulignés.

La matrice fonctions/entités

Si la matrice effectuée en phase d'analyse préalable pour une maquette doit être détaillée, la matrice à construire dans le cadre d'un projet peut être un peu plus sommaire. Les entités utilisées pour la vérification sur les clés primaires et pour la recherche (listes de valeurs et validation) ne sont pas indispensables.

Par contre, il faut faire attention à bien identifier les fonctions communes. Il faut se baser sur les informations à fournir lors la description des fonctions : type de la fonction (*immediate/overnight*), fréquence et unité de temps. Si ces informations sont identiques pour plusieurs fonctions, alors celles-ci sont communes. Dans *Designer 2000*, le remplissage de ces champs ne génère aucun code, il est purement documentaire et à destination de l'analyste. Il est important de noter que, si *Designer 2000* trouve deux fonctions de type *overnight* avec des noms différents, mais avec les mêmes usages, il génère sans avertissement deux écrans identiques sauf pour le titre.

Le modèle des processus

Il faut construire un premier schéma global du système et diviser ce dernier en sous-systèmes. Les entités primaires doivent figurer au centre de chaque diagramme, les entités secondaires à la périphérie.

Les entités

Il faut penser à estimer le volume et le taux d'accroissement des données pour chaque entité. Il est préférable de procéder à des évaluations manuelles, plutôt que d'utiliser *Designer 2000*. La mise en oeuvre de cette évaluation y est longue et complexe.

Il faut essayer de limiter au maximum le nombre des arcs exclusifs et des sous-types.

Les relations

Il faut définir les relations transférables pour le côté many et décrire le transfert. Une relation est dite 'transférable' si elle autorise des changements au niveau des lignes. Par exemple : dans la relation 'est employé de', entre les entités 'employé' et 'société', on admettra qu'un employé puisse changer de service, et donc, que la relation puisse être transférée sur une autre ligne de l'entité 'société'.

Le prototypage

Le prototype doit avoir un objectif bien déterminé. Il peut porter sur la performance du système, sur ses fonctionnalités, sur les écrans ou sur la méthodologie employée pour la modélisation. Si la méthodologie de modélisation et les fonctionnalités sont des aspects très importants, il ne faut cependant pas négliger les prototypes de performance et d'écrans. Le premier permet d'identifier, qui du serveur, qui du client, est le plus gourmand en ressources. Il est à utiliser pour se rapprocher au

mieux des temps de réponse nécessaires à une application performante. Le second passe par la définition de standards en terme d'interface homme/machine. Sa mise en oeuvre est facilitée par les modèles fournis par défaut par *Designer 2000*.

Le reverse engineering

Il est préférable de ne pas faire du *reverse engineering* à partir d'écrans existants. Ceci est de toute façon exclu si les écrans sont en *Forms 3*. Le code relatif aux traitements, déclencheurs et procédures de sorties utilisateurs ne seront probablement remontés que partiellement. Par ailleurs, la migration d'écrans *Forms 3* vers *Forms 4.5* n'est pas automatique. Elle constitue à elle seule un projet. Il faut compter cinq à six écrans par jour.

Annexe F Les phases de réalisation d'un projet

Ce plan de travail a été proposé à titre indicatif par la société Oracle.

La définition

Décrire l'existant :

- créer les modèle des processus et de données
- décrire l'organisation interne
- décrire les interfaces
- décrire l'architecture technique

obtenir la volumétrie
obtenir les documents de référence
créer le modèle des processus fonctionnels
créer le modèle des fonctions
créer un premier modèle fonctionnel de données
décrire les besoins :

- interfaces système
- conversion des données
- besoins opérationnels

préparer l'architecture technique initiale, la volumétrie initiale et un glossaire
rédiger le rapport de phase

L'analyse

obtenir les informations fonctionnelles
décrire l'architecture technique révisée et les fonctions du système existant
décrire le modèle de données du système existant et les procédures d'exploitation utilisées
créer le modèle fonctionnel de données détaillé ainsi que les modèles de données, de fonctions et des processus système
définir les besoins détaillés d'exploitation du système
développer les définitions de la plate-forme logicielle et matérielle et de l'architecture distribuée
développer les standards interface homme-machine, la volumétrie révisée, la stratégie de restauration après incident et les questions de sécurité du système
déterminer la stratégie de conversion des données existante et le plan initial de celle-ci
établir les besoins de formation et son planning
définir la stratégie de mise en production du système
rédiger le rapport de phase

La conception

Dans *Designer 2000* :

- définir les standards de conception
- créer les modèles logiques de données et de traitements
- créer la structure des menus
- renseigner la description fonctionnelle des modules[1] et leur description technique
- créer les index

créer les autorisations sur les objets de la base
déterminer les questions sur les performances
développer le plan de tests unitaires et tests d'intégration
définir la stratégie de mise en production
concevoir les modules de conversion des données
rédiger le rapport de phase.

N.B. : Cette phase sera plus ou moins longue selon les niveaux de complexité du projet et des écrans. Pour rappel, une fonction de complexité niveau deux est un maître-détail avec un affichage simple et contient au plus trois facteurs de complexité. Par facteur de complexité, il faut entendre : un affichage différent de la structure de la base, avec des graphiques, des options de recherche ou de tri, des validations ou traitements complexes (procédures ou fonctions), des options de menus spéciales avec des particularités de navigation, des validations d'accès sécurisé, des vues et des tables temporaires, des besoins particuliers de performance, un affichage dynamique impliquant des développements dans *Forms*.

La réalisation

Dans *Designer 2000* :

- définir les standards de développement
- créer l'environnement de développement
- créer le modèle physique des données
- générer les modules primaires

développer le module de conversion des données
effectuer les tests unitaires
tester le module de conversion
développer le plan de test du système et les jeux d'essai le concernant
développer un plan d'installation en fonction de l'architecture distribuée
créer l'environnement de test en tenant compte de l'architecture distribuée
effectuer les tests système
développer le plan des tests d'intégration, leurs jeux d'essai et les tester
créer les scripts de la base de production

La documentation

spécifier les besoins de documentation et ses standards
produire les modèles et prototypes de documents
créer l'environnement de documentation

[1] Un module est une fonction implémentée (écrans, rapports, etc.)

produire les documents de référence
produire les manuels d'utilisation
produire les documents techniques particuliers
créer les documents de formation
produire le guide d'exploitation initial
compléter les documents de référence, le guide d'utilisation et le référentiel technique
créer une base de données de formation
compléter les documents de formation
compléter le guide d'exploitation du système
produire l'aide en ligne, les documents du cours de formation ainsi que la documentation du maintenance client

La transition

préparer l'installation de la formation, des tests de recette, l'environnement de maintenance client et de production
convertir les données de production
former les utilisateurs, les administrateurs et l'équipe de recette
effectuer les tests de recette
arrêter l'ancien système

La production

spécifier les métriques de performance
mesurer les performances du système
faire le monitorat et répondre aux problèmes
auditer le système
se mettre d'accord sur les exceptions de performance
analyser les problèmes et les exceptions de performance
préparer le plan de mise à jour de l'application
produire les scripts et les procédures de mise à jour de l'application
implémenter les modifications de l'application
tester les modifications de l'application et effectuer sa mise à jour
recueillir les demandes d'amélioration
planifier les mises à jour de l'application.