MINISTERE DES ENSEIGNEMENTS SECONDAIRE, SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE POLYTECHNIQUE DE BOBO-DIOULASSO

ECOLE SUPERIEURE D'INFORMATIQUE

01 BP 1091, Tél. (226) 97 27 64
BOBO-DIOULASSO

CENTRE INTERNATIONAL DE RECHERCHE-DEVELOPPEMENT SUR L'ELEVAGE EN ZONE SUBHUMIDE

(C. I. R. D. E. S.)

01 B.P. 454 BOBO-DIOULASSO 01
Tél : (226) 97 20 53 / 97 22 87
Télécopie (226) 97 23 20

Thème

Portage d'une application mono-utilisateur et monoposte vers un environnement multi-utilisateur et distribué

(Cas particulier du « Financier »)

Mémoire de fin d'étude

Présenté et soutenu publiquement le 15 février 2001

Pour l'obtention du

Diplôme d'ingénieur de conception en informatique

Par

Joseph OUEDRAOGO

Composition du jury

Président :	Pr Théodore TAPSOBA
Rapporteur :	M. Mesmin T. DANDJINOU
Examinateurs :	M. Oscar MANSO, directeur du mémoire
	M. Ali B. KABA, maître de stage

Année académique 1999/2000

SOMMAIRE

DEDICACE

A ma mère,
A mes frères et sœurs,
A vous qui m'êtes chers.

REMERCIEMENTS

Nos remerciements vont :

- aux autorités de l'Université Polytechnique de Bobo-Dioulasso qui sont parvenues à faire du Cycle des Ingénieurs de Conception en Informatique une réalité ;
- à l'administration de l'Ecole Supérieure d'Informatique pour tout leur dévouement à faciliter la réalisation de notre formation ;
- aux enseignants des Universités européennes, canadiennes et africaines qui ont montré leur disponibilité à nous assurer une formation de qualité ;
- aux enseignants de l'Université de Ouagadougou qui ont montré leur disponibilité tout le long de notre formation.

Nous manifestons notre reconnaissance :

- à Monsieur le Directeur de l'Ecole Supérieure d'Informatique, le Professeur Théodore TAPSOBA,
- au Directeur des Etudes de l'Ecole Supérieure d'Informatique, Monsieur Mesmin T. DANDJINOU,
- au Chef du département informatique, Monsieur Ali B. KABA,

pour la persévérance et tous leurs efforts pour la réalisation des enseignements destinés à ce cycle ;

Je remercie mon directeur de mémoire Monsieur Oscar MANSO, ainsi que mon maître de stage, Monsieur Ali B. KABA pour la patience et la disponibilité avec lesquelles ils m'ont accompagné dans la réalisation de ce travail.

Je voudrais enfin manifester ma profonde gratitude à tous ceux qui ont contribué dans la discrétion ou dans l'anonymat à la finalisation de ce projet qui me tenait tant à cœur.

Qu'ils trouvent entre ces lignes l'expression de ma profonde reconnaissance !

INTRODUCTION

Dans le cadre de la formation des ingénieurs de conception en informatique, l'Ecole Supérieure d'Informatique prévoit à l'intention des étudiants en fin de cycle, un projet de six (6) mois répartis comme suit :

- trois (3) mois à l'école consacrés à la conception et la recherche bibliographique ;
- trois (3) mois en entreprise pour l'implémentation de la solution.

Il donne lieu à la réalisation d'un mémoire de fin d'étude dans lequel l'étudiant présente comment il utilise l'ensemble des connaissances acquises pour la résolution d'un problème informatique spécifique.

C'est ainsi qu'il m'a été confié le thème portant sur le « *portage d'une application mono-utilisateur et monoposte vers un environnement multi-utilisateur et distribué. (Cas particulier du "Financier")* ».

Ce document sera constitué de trois (3) parties :

- une partie consacrée à la présentation générale des aspects liés au thème. Ainsi, les concepts de monoposte, mono-utilisateur, multi-utilisateur, et distribué seront abordés ;
- une partie relative à la problématique liée à la mise en œuvre du portage ainsi que la démarche de résolution du problème. Nous présenterons quelques concepts relatifs à la méthode Merise et à l'approche UML ;
- une partie qui décrit l'implémentation avec le « Financier ». Nous y présenterons les modèles de l'existant ainsi que les résultats obtenus suite à la mise en œuvre du portage.

PREMIERE PARTIE : GENERALITES

I. DEFINITIONS DES CONCEPTS

Une application est un sous-système qui traite de problèmes :

- qui ont une même finalité,
- qui sont fortement dépendants les uns des autres,
- et qui sont indépendants ou quasiment indépendants des autres problèmes traités par le système d'information.

« Une application est informatique si son système de communication, son système de traitement et ses données résidentes sont supportés par des moyens informatiques (et téléinformatique) » [CASTELLANI 1987].

« L'environnement d'un système est l'univers auquel il appartient, univers constitué de tous les composants qui ne le constituent pas et desquels il reçoit et ou auxquels il émet des événements (et donc des messages de données qui acheminent ces événements) » [CASTELLANI 1987]. »

Une application locale est une application qui effectue ses traitements sur le poste de travail de l'utilisateur. Les données et les programmes y sont localisés.

Une application distante est une application dont les fonctionnalités sont accédées à distance par les utilisateurs en passant par un réseau de communication.

Pour mieux appréhender la problématique du portage, nous évoquerons de façon succincte les concepts d'application monoposte, mono-utilisateur, multi-utilisateur, ou distribuée.

I-1. Le concept d'application monoposte

Une application monoposte est une application dont les programmes de traitement et les données sont installés et exploités sur un seul poste de travail. Ce poste de travail qui doit être intelligent dispose d'une mémoire secondaire pour stocker les données et d'une mémoire centrale pour charger les données et les programmes à traiter. Un ou plusieurs processeurs assurent la réalisation des opérations effectuées sur les données.

La communication entre le système, les programmes et les données se fait à travers des entrées-sorties entre la mémoire centrale et les mémoires secondaires.

I-2. Le concept d'application mono-utilisateur

Une application mono-utilisateur est une application conçue pour être exploitée par un seul utilisateur.

L'interface utilisateur est figée, offrant toutes les fonctionnalités associées à l'utilisateur qui est en cours de session. Lorsque le système d'authentification des accès est mis en place, l'utilisateur a soit accès à toutes les fonctionnalités soit à aucune d'entre elles.

I-3. Le concept d'application multi-utilisateur

Une application multi-utilisateur est une application conçue pour être exploitée par plusieurs utilisateurs ayant des prérogatives différentes par rapport aux fonctionnalités qui sont offertes

par l'application. Pour cela, l'interface utilisateur se configure au démarrage en fonction des droits accordés à chaque utilisateur. Les données et les programmes peuvent être stockés sur une ou plusieurs machines.

I-4. Le concept d'application distribuée

Une application distribuée est une application dont les données et les programmes sont exploitables à partir de plusieurs postes de travail. La notion de distribution couvre deux aspects importants : la considération par rapport aux traitements et celle par rapport aux données.

Lorsque les données sont gérées au niveau d'une machine unique alors que les programmes sont répartis au niveau de chaque poste de travail, chaque traitement lancé par un utilisateur accède aux données stockées sur la machine distante en passant par le réseau.

Lorsque les données sont réparties sur plusieurs machines alors que les programmes sont installés sur une seule machine, l'application interagit alors avec plusieurs bases de données. Les autres machines peuvent exécuter les programmes via le réseau de communication mis en place.

Lorsque les données et les programmes sont répartis sur plusieurs machines, la mise en œuvre de la répartition des données permet à des groupes d'utilisateurs de disposer de certaines données en local. Cela concerne par exemple l'interconnexion de sites d'entreprise géographiquement distincts mais exploitant une application globale. Certains sites peuvent disposer de données en local auxquelles ils accèdent, tout en ayant la possibilité d'accéder aussi aux informations localisées au niveau des autres sites.

Ainsi, les traitements effectués par les utilisateurs peuvent interagir, aussi bien avec des données locales qu'avec des données distantes. Ces traitements localisés au niveau des différents sites accèdent aux données réparties. Cela nécessite la mise en place de bases de données au niveau des sites, qui peuvent être homogènes ou hétérogènes. En outre, selon les prérogatives accordées à chaque site et aux utilisateurs, les fonctionnalités de l'application ne se présentent pas de la même manière.

Le tableau ci-après *(figure 1)* propose un résumé comparatif entre, d'une part une application mono-utilisateur et monoposte, et une application multi-utilisateur et distribuée d'autre part.

Domaine spécifié	Mono-utilisateur et monoposte	Multi-utilisateur et distribué
• **Interface utilisateur**	- L'interface est figée ; - L'authentification présente la totalité ou aucune des fonctionnalités offertes par l'application.	- S'adapte aux prérogatives associées à chaque utilisateur ; - L'authentification permet de distinguer les fonctionnalités en fonction des utilisateurs.
• **Programmes**	- Les programmes sont localisés et exécutés sur un seul poste de travail.	- Les traitements peuvent être réalisés à partir de plusieurs postes de travail ; - Les programmes peuvent être répartis sur plusieurs postes de travail.
• **Données**	- Les données sont localisées sur une seule machine.	- Les données (homogènes ou hétérogènes) peuvent être localisées sur une ou plusieurs machines.
• **Système de communication**	- La communication se fait par des entrées-sorties entre la mémoire centrale et les mémoires secondaires.	- La communication se fait à travers un réseau qui permet aux différents postes de travail d'échanger.

Figure 1 : Tableau comparatif entre, d'une part une application mono-utilisateur et monoposte et, d'autre part, une application multi-utilisateur et distribuée

I-5. Le concept du portage

Porter une application d'un environnement X vers un environnement Y revient à mettre en œuvre les démarches et les techniques adéquates pour permettre à l'application initiale de remplir ses fonctions dans les contraintes du nouvel environnement. Le portage vise aussi à garantir les performances capitalisées dans l'environnement initial tout en tirant profit des avantages offerts par le nouvel environnement.

Il s'agit donc de préserver ce qui fonctionne de manière éprouvée, d'opérer les modifications nécessaires pour adapter l'application initiale au nouvel environnement d'exploitation. Il s'agit aussi d'ajouter les fonctionnalités qui s'avèrent indispensables aux conditions d'exploitation du nouvel environnement.

Il peut s'agir de supprimer des fonctionnalités qui n'ont plus d'utilité dans le nouvel environnement.

Enfin, il s'agit de décrire l'intégration de l'application dans le nouvel environnement d'exploitation.

II. LA PROBLEMATIQUE ET LE CONTEXTE DE L'ETUDE

II-1. Le contexte de l'étude

II-1-1. Présentation du CIRDES

Le Centre International de Recherche-Développement sur l'Elevage en zone Subhumide (CIRDES) a été créé le 17 mai 1994. Sont membres du CIRDES le Bénin, le Burkina Faso, la Côte d'Ivoire, le Ghana, le Mali, le Niger et le Togo. Le CIRDES est une « *institution à mandat régional, doté de la personnalité juridique internationale et de l'autonomie administrative et financière* » [KABORE & ZEMBA 1997]. Il a pour mission de favoriser une augmentation rapide de la production animale, grâce à des recherches sur les principales contraintes auxquelles l'élevage est confronté en zone subhumide et la proposition de thèmes améliorateurs.

Depuis le 21 décembre 1998, le CIRDES dispose de trois (3) unités opérationnelles de recherche rattachées à la Direction Scientifique :

- l'unité de recherche sur les bases biologiques de production intégrée (URBIO),
- l'unité de recherche sur la production animale (URPAN),
- l'unité de recherche sur l'élevage et l'environnement (UREEN).

La Direction administrative et financière comprend :

- un Service achats et stocks,
- un Service de gestion du personnel,
- un Service de la comptabilité.

II-1-2. Présentation générale du problème

Suite à l'évolution des organisations, les applications peuvent ne plus satisfaire aux règles initiales de fonctionnement. Ainsi, le Centre International de Recherche-Développement sur l'Elévage en zone Sub-humide (CIRDES) disposait dans un premier temps d'une application de gestion nommée "JP" conçue sous Dbase exploitant des fichiers multi-critères Dbase III. Grâce à la collaboration entre l'Ecole Supérieure d'Informatique et ce centre, un projet de fin de cycle de troisième année du Cycle des Ingénieurs de Travaux Informatiques a débouché sur la réalisation du « Financier » en fin 1997.

A cause de l'instabilité des groupes de projets, ce produit n'a pas été réalisé jusqu'au bout. En plus des besoins exprimés initialement, nous avons pour tâche de porter cette application dans un environnement multi-utilisateur et distribué. Ce travail demandé vise, d'une part, à achever l'implémentation commencée par les autres groupes de projets et, d'autre part, à réaliser le portage.

Le portage consistera donc à :

- prendre en compte la séparation des fonctions organisationnelles dans l'exploitation des fonctionnalités de l'application ;
- permettre à chaque utilisateur d'accéder aux fonctionnalités de l'application à partir de son poste de travail ;
- prendre en compte les nouvelles règles de gestion et d'organisation.

II-1-3. Situation actuelle du « Financier »

En plus des contraintes multi-utilisatrices évoquées précédemment, le délaissement de la réalisation de l'application depuis bientôt trois (3) années a eu pour conséquences :

- une corruption d'une partie des fichiers sources de l'application ;
- un changement de certaines règles organisationnelles qu'il faudra prendre en compte.

L'application réalisée ambitionnait les fonctionnalités principales suivantes :

- la gestion comptable qui est complètement inexploitable d'où la nécessité de reprendre partiellement l'analyse avant la programmation ;
- le suivi budgétaire qui n'a pas été complètement réalisé ;
- le suivi des conventions de financement qui est réalisé malgré l'existence de "bugs" ;
- le suivi des immobilisations qui est fonctionnel mais dont l'analyse et la conception sont à revoir ;
- les codifications comptables et budgétaires dont la réalisation est suffisament avancée ;
- le paramétrage de l'interface homme/machine qui est parfaitement implémenté.

II-2. Les objectifs et les contraintes de l'étude

II-2-1. Les objectifs de l'étude

Compte tenu des difficultés pratiques dans l'exploitation de l'application monoposte, l'objectif est de réaliser une application issue du «Financier» à même d'offrir des fonctionnalités multi-utilisatrices et des performances de traitement plus élaborées.

L'étude consistera à :

- faire des économies de temps en évitant une reprise complète de la réalisation. Cela nous amènera à adopter une démarche de préparation du portage en dégageant les modèles conceptuels de données qui n'ont pas été mis à jour lors de la réalisation. En outre, il faudra parvenir à la restauration du code source sans laquelle aucune implémentation n'est envisageable ;
- effectuer une maintenance corrective des fonctionnalités qui ont des difficultés de fonctionnement ;
- mettre en œuvre le portage en effectuant les modifications et ajouts nécessaires pour adapter le « Financier » à une exploitation dans un environnement multi-utilisateur et distribué.

Cela permettra de réduire les coûts de réalisation et d'accélérer les délais d'implémentation.

II-2-2. Les contraintes de l'étude

Les contraintes dans le cadre de la réalisation du projet sont les suivantes :

- ne pas programmer à nouveau l'application ;
- changer de version par rapport à la version de développement initiale (Delphi 3) ;
- permettre des traitements distribués ;
- proposer une solution à moindre coût.

III. Méthode de résolution

Une méthode est un ensemble de démarches raisonnées, suivies, pour parvenir à un but. Une méthode est constituée par des modèles, des langages, une démarche et des outils.

Les modèles proposent des concepts qui permettent de guider et de formaliser le raisonnement dans l'identification des aspects pertinents du système à décrire. En d'autres termes, les modèles permettent de savoir ce qui est important de décrire dans le système.

Les langages permettent de décrire les aspects pertinents d'un système dans le but de favoriser le dialogue en supprimant toutes les ambiguïtés possibles.

La démarche structure le chemin à suivre pour la construction d'un système. C'est l'ensemble des structurations qu'il faut suivre pour la résolution d'un problème.

Les outils apportent une assistance à la mise en œuvre des méthodes. On distingue les outils d'aide à la conception, à la réalisation et aux simulations.

III-1. La méthode Merise

Merise est une méthode systémique d'analyse et de conception des systèmes d'information. Elle permet de modéliser les données indépendamment des traitements

III-1-1. Cycles proposés par Merise

La démarche méthodologique de Merise propose trois (3) cycles : le cycle de vie, le cycle de décision et le cycle d'abstraction.

✓ **Le cycle de vie**

Quatre étapes caractérisent le cycle de vie d'un objet : la gestation, la naissance, la maturité et la mort.

La démarche de conception des systèmes d'information proposée par Merise comporte trois (3) grandes périodes : la conception, la réalisation, la maintenance. Ces trois grandes périodes sont illustrées dans le schéma suivant *(figure 2)*.

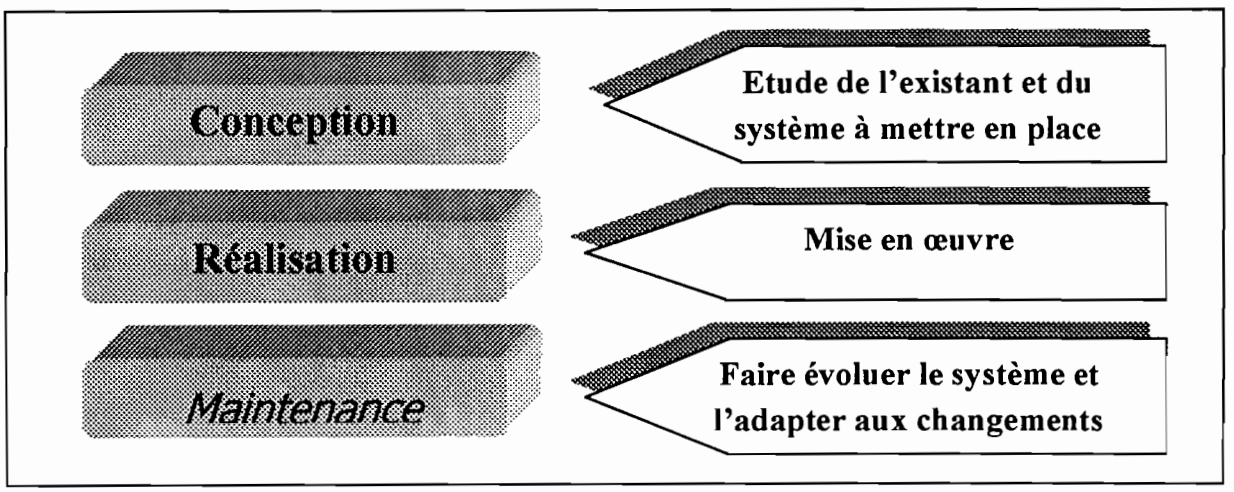


Figure 2 : Les étapes du cycle de vie proposées par Merise

✓ **Le cycle de décision**

Il permet d'organiser l'intervention des personnes de l'entreprise en fonction, de la hiérarchie des décisions à prendre à propos du contenu du système et de son mode de développement. Ainsi, Merise se soucie d'associer étroitement les utilisateurs dans les tâches d'analyse et de conception.

✓ **Le cycle d'abstraction**

Merise propose trois (3) niveaux d'abstraction : le niveau conceptuel, le niveau logique ou organisationnel et le niveau opérationnel.

Le niveau conceptuel

Il se préoccupe des éléments fondamentaux constituant l'activité à modéliser. Ces éléments reprennent les finalités, les buts et objectifs définis par les décideurs ainsi que les acteurs du système, les notions manipulées et les règles de gestion à mettre en œuvre. Cette étape donne lieu à la réalisation d'un modèle conceptuel de données (MCD). Le MCD décrit les informations mémorisées dans l'entreprise. Au niveau des traitements, la réalisation du modèle conceptuel des traitements (MCT) permet de formaliser les règles qui dictent la réaction de l'entreprise face à des sollicitations externes.

Le niveau logique ou organisationnel

Le niveau organisationnel prend en compte les ressources (homme, machine et leur combinaison). Cette organisation intègre les contraintes économiques, techniques et humaines. Ce niveau fait apparaître la partie du système à automatiser.
Avec les données, on procède à la réalisation du modèle logique de données (MLD). Elaboré à partir du modèle conceptuel de données, le MLD définit les informations mémorisées en prenant en compte les aspects techniques de gestion des données.

Avec les traitements, le modèle organisationnel des traitements (MOT) complète le modèle conceptuel de traitement en définissant les acteurs, le lieu et le moment de déroulement des différentes actions.

Le niveau physique

A ce niveau, il s'agit de faire les choix techniques en prenant en compte les besoins et les contraintes des utilisateurs, des machines, des logiciels, des outils de développement.

Compte tenu de notre approche de résolution par rapport au portage, nous éluciderons les formalismes relatifs au modèle conceptuel de données et au modèle logique de données. Il est à signaler que le modèle logique de données est obtenu à partir du modèle conceptuel de données par l'application de règles de transformation.

III-1-2. Formalismes liés au MCD, au MLD et au MPD

III-1-2-1. Formalisme lié au modèle conceptuel de données

Le modèle conceptuel de données formalise la description des données (ou informations) mémorisées par le système d'information. Il définit ainsi la vue que les acteurs ont des données qu'ils manipulent.

Un MCD représente la structure logique globale d'une base de données, indépendamment du logiciel ou de la structure de stockage des données. Un modèle conceptuel contient toujours des données qui ne sont pas encore implémentées dans la base de données physique. Il présente une représentation formelle des données nécessaires au fonctionnement d'une entreprise.

Quatre concepts essentiels sont utilisés pour la représentation du MCD : les concepts de propriété, d'entité, de relation et de cardinalité.

✓ Concept de propriété ou de propriété-type

Il permet de représenter les informations élémentaires qui peuvent être mémorisées par le système d'information.

Exemples de propriétés :

- nom d'un agent,
- prénom de l'agent,
- date d'une écriture comptable.

La valeur associée à une propriété peut changer. Chacune des valeurs que peut prendre une propriété est appelée occurrence de cette propriété. Chaque propriété ou propriété-type est caractérisée par son nom, son type, son format de présentation.

✓ Concept d'entité ou d'entité type

Une entité permet de conceptualiser les objets matériels ou immatériels qui présentent un intérêt pour le domaine d'étude. Elle définit une classe d'objets, c'est-à-dire un ensemble d'objets ayant les mêmes propriétés. Une entité doit avoir une propriété particulière dont les valeurs permettent de définir de façon unique les occurrences de l'entité : c'est l'*identifiant* qui est souligné dans la représentation. Une occurrence d'entité est une représentation dans le monde réel correspondant à des valeurs particulières de propriétés.

La représentation ci-dessous *(figure 3)* illustre une entité **LigneEcriture** et des occurrences de cette entité.

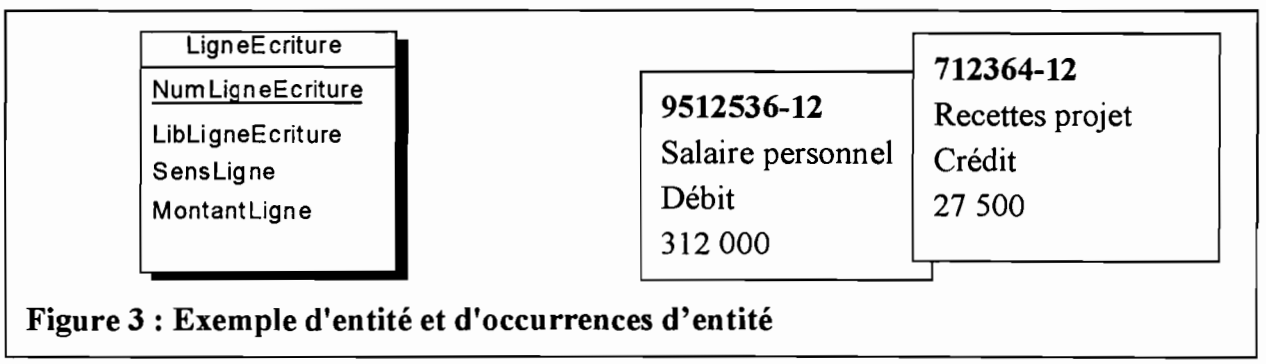


Figure 3 : Exemple d'entité et d'occurrences d'entité

✓ Concept de relation ou de relation-type

Il permet de conceptualiser les liens sémantiques qui peuvent exister entre les occurrences d'entité.

Une relation n'a pas d'existence propre. Elle est définie par rapport à des entités. Une relation peut porter des propriétés. La figure ci-dessous illustre une relation *(figure 4)*.

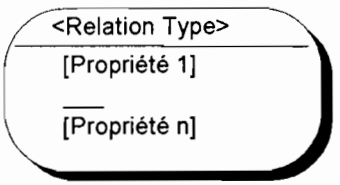


Figure 4 : Représentation d'une relation

✓ Expression des contraintes d'intégrité

Le modèle offre plusieurs formes d'expression des contraintes d'intégrité qui permettent d'expliciter les faits mémorisés qui doivent être conformes aux règles de gestion : les cardinalités, les dépendances fonctionnelles, les contraintes d'intégrité fonctionnelles et de stabilité, les contraintes d'intégrité inclusives ou exclusives. Nous nous attarderons seulement sur les cardinalités.

✓ Concepts de cardinalité

Une cardinalité indique le nombre de fois qu'une occurrence de l'entité peut participer à des occurrences de relation. Une cardinalité est composée d'une cardinalité minimale et d'une cardinalité maximale.

Exemples :

(0,1) : une occurrence d'entité peut participer au plus une fois à une (occurrence de) relation ;
(1,1) : une occurrence d'entité participe toujours une et une seule fois à une relation ;
(1,n) : une occurrence d'entité participe au moins une fois à une relation ;
(0,n) : une occurrence d'entité participe zéro ou plusieurs fois à une relation.

La figure 5 ci-dessous montre qu'une écriture comprend deux ou plusieurs lignes d'écriture.

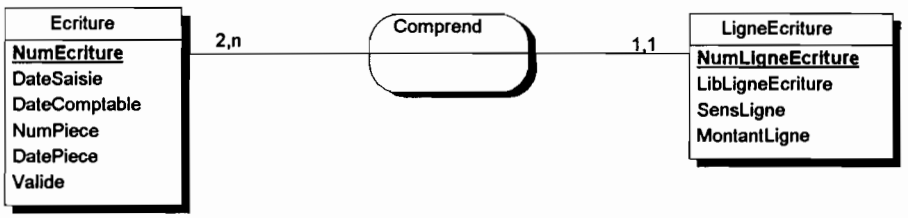


Figure 5 : Intégration des cardinalités dans la relation « Comprend »

III-1-2-2. Formalisme lié au modèle logique de données (MLD)

Le but du MLD est de définir une représentation logique du modèle conceptuel de données. En effet, cette représentation n'est pas propre à un système de gestion de base de données, mais prend en compte les principes fondamentaux de l'art. De ce fait, le MLD doit être défini en fonction de la technologie qui sera utilisée pour la gestion des données. Le passage du MCD au MLD se fait suivant une procédure algorithmique.

II-1-2-3. Règles de passage du modèle conceptuel de données au modèle logique de données

Ces règles diffèrent selon que l'on est en présence d'une entité ou d'une relation.

✓ ***Cas des entités***

Chaque entité du modèle individuel devient une table du modèle relationnel et les propriétés de l'entité deviennent les attributs de la table. L'identifiant de l'entité devient la clé primaire de la table.

✓ ***Cas des relations binaires***

1. Lorsque les cardinalités sont (*, n) – (1,1)

On duplique la clé de la table de cardinalité (*, n) dans la table de cardinalité (1,1). La clé dupliquée devient une clé externe. On fait migrer les propriétés portées par la relation dans la table dont la cardinalité est (1,1).

2. Lorsque les cardinalités sont (*, n) – (0,1)

Deux possibilités s'offrent à nous par rapport à la transformation à opérer.

La première solution consiste à dupliquer la clé de l'entité à cardinalité (*, n) dans la table à cardinalité (0,1). On fait migrer ensuite les propriétés de la relation dans la table à cardinalité (0,1).

Quant à la deuxième solution, la relation est transformée en table ayant comme clé, l'identifiant de l'entité à cardinalité (0,1) et comme attribut, l'identifiant de l'entité à cardinalité (*, n). On fait migrer ensuite les propriétés de la relation dans la table.

3. Cas des relations binaires (*, n) – (*, n) avec n ≥ 2

La relation est transformée en table ayant comme clé, la concaténation des identifiants des entités liées par leur relation. On fait migrer les propriétés de la relation dans la table.

4. Cas des relations qui impliquent plus de deux entités

On transforme la relation en table dont la clé est la concaténation des identifiants des entités qui participent à la relation. On fait migrer les propriétés de la relation dans la table créée. La *figure 6* donne un exemple de MLD généré à partir du MCD proposé au niveau de la *figure 5.*

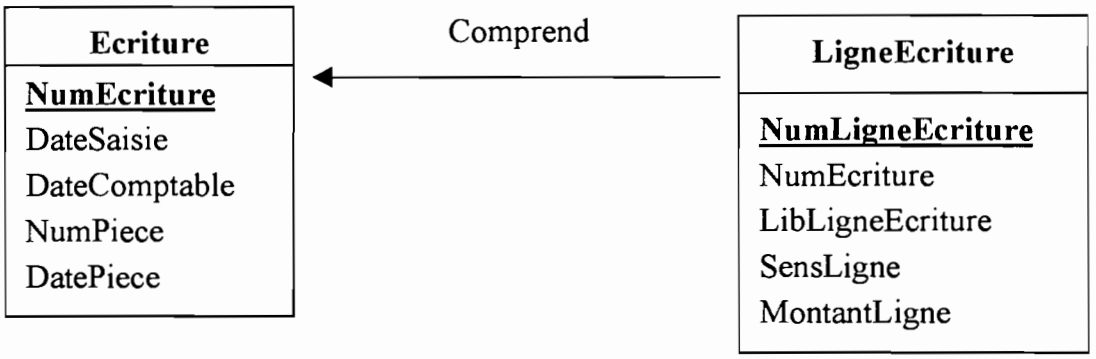


Figure 6 : Modèle logique généré à partir du modèle conceptuel de la figure 5

III-1-2-3. Le modèle physique de données

Le MPD définit la configuration physique de la base de données et permet d'en connaître les détails. Ce modèle prend en compte à la fois les structures de stockage du logiciel et celles des données. Le MPD peut être modifié afin de l'adapter à des contraintes physiques ou à des besoins spécifiques de performances.

La génération d'un MPD convertit des objets conceptuels en objets physiques *(figure 7).*

Objet d'un MCD	**Objet généré dans un MPD**
Entité	Table
Attribut d'entité	Colonne de table
Identifiant	Clé primaire ou étrangère, selon la relation de dépendance
Association	Référence

Figure 7 : Eléments de transformation du MCD au MPD

III-1-3. Quelques concepts de l'approche orientée objet

Un système orienté objet est un ensemble de classes qui coopèrent. Pour coopérer, les objets utilisent des messages qu'ils s'envoient entre eux par divers mécanismes qui dépendent de l'environnement de mise en œuvre.

Un objet

«Un objet est défini par ses données et son comportement » [SOUTOU 1999]. Un objet est un membre du système orienté objet et est défini dans son domaine par :

- une identité qui constitue le moyen d'identifier l'objet par rapport aux autres objets du système. Chaque objet doit avoir une identité ;
- un comportement qui définit la manière dont l'objet réagit aux autres messages qui lui parviennent de son environnement ;
- un état qui définit l'une des possibilités dans lesquelles un objet peut se trouver en un instant donné de sa vie.

Une classe

C'est une description abstraite d'un ensemble d'objets ayant des propriétés similaires, un comportement commun, des relations communes avec d'autres objets et des sémantiques communes.

« Une classe regroupe des objets (appelés instances de la classe) de même structure et de même comportement » [SOUTOU 1999].

Une méthode

Les opérations que l'on peut effectuer sur les objets d'une classe sont appelées méthodes.

L'abstraction

L'abstraction constitue une des bases de l'approche orientée objet. Elle peut se définir comme le processus qui consiste à identifier une entité en mettant en évidence ses caractéristiques pertinentes du point de vue de son utilisation.

L'encapsulation

C'est le mécanisme qui permet de modéliser en même temps les données et les opérations. Un objet est donc composé d'une interface publique qui contient la spécification des opérations et une partie implémentation où l'on trouve à la fois la structure de données et les codifications des opérations.

La surcharge

La surcharge permet de définir plusieurs fois un opérateur de même nom pour des classes de types d'objets différents.

Le composant

Un composant est l'élément binaire qui offre aux autres composants (clients), une interface qui leur permet de dialoguer avec le composant en question. Cette interface est un contrat qui définit la manière d'utiliser le composant.

La classification et l'instanciation

La classification est une forme d'abstraction par laquelle une collection d'objets de même nature est vue comme un objet unique appelé classe.

L'instanciation est la matérialisation d'une occurrence d'objet d'une classe donnée.

L'association

Une association représente une relation entre classes d'objets. Une association symbolise une information dont la durée n'est pas négligeable par rapport à la dynamique générale des objets instances des classes associées.

L'agrégation

Elle représente une association non symétrique dans laquelle une des extrémités joue le rôle prédominant par rapport à l'autre.

Quel que soit le nombre de classes, l'agrégation ne concerne qu'un rôle d'une association. La figure ci-dessous *(figure 8)* donne l'illustration d'une agrégation.

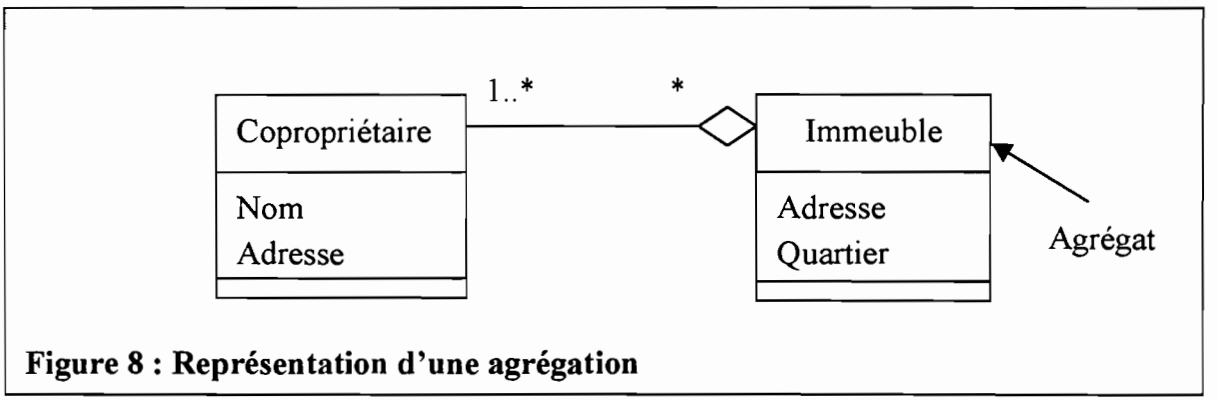


Figure 8 : Représentation d'une agrégation

L'héritage

Il permet de regrouper des attributs et des méthodes au sein d'une classe générique appelée super-classe ou sur-classe. Des sous-classes héritent ainsi de la structure et du comportement de la classe générique qu'elles enrichissent avec leurs propres attributs et méthodes *(figure 9)*.

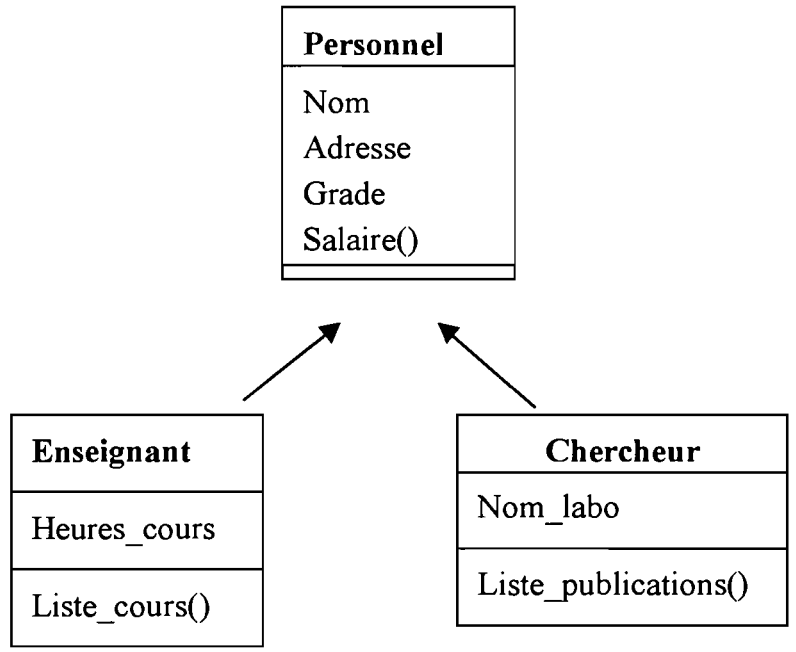


Figure 9 : Illustration d'une héritage

III-2. L'approche UML

UML (Unified Modeling Language) est un langage unifié, conçu pour modéliser « tous » les phénomènes de l'activité de l'entreprise (processus, métier, systèmes d'information, systèmes informatiques, composants logiciels), indépendamment des techniques d'implémentation (système automatisé ou non, langage de programmation) mises en œuvre par la suite. Les constituants d'UML sont, d'une part, les éléments de modélisation qui représentent toutes les propriétés du langage et d'autre part les diagrammes qui constituent l'expression visuelle et graphique. Nous présentons ici, quelques concepts que nous aurons à utiliser.

III-2-1. Les diagrammes de composants

Les diagrammes de composants permettent de décrire des éléments physiques et leurs relations dans le système en cours de modélisation. On distingue trois types de composants : la spécification, le corps et la spécification générique *(figure 10)*. La *figure 11* illustre la relation qui peut exister entre deux composants.

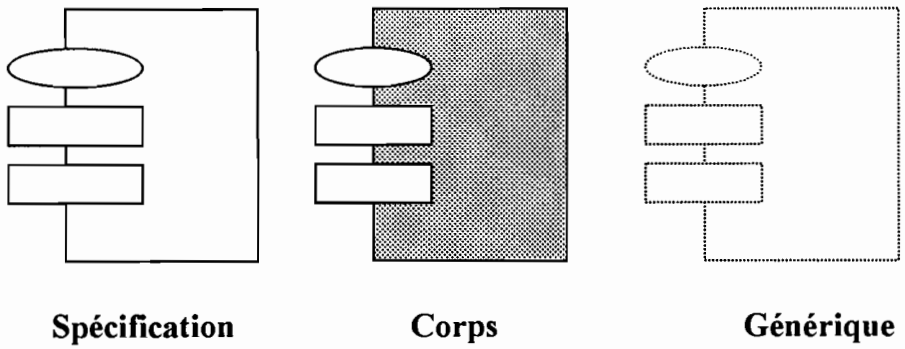


Figure 10 : Les différents types de composants

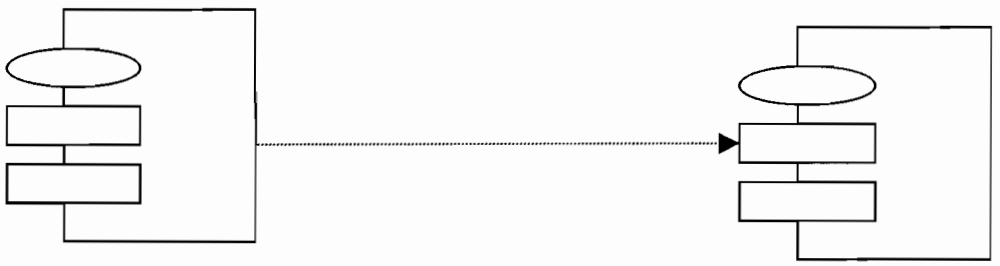


Figure 11 : Relation entre composants

III-2-2. Les sous-systèmes

Pour faciliter la réalisation des applications, les différents composants peuvent être regroupés dans des paquetages selon un critère logique. Les composants sont souvent stéréotypés en sous-systèmes pour ajouter les notions de bibliothèque de compilation et de gestion de configuration à la sémantique de partition déjà véhiculée par les paquetages.

Chaque sous-système peut contenir des composants et d'autres sous-systèmes. La décomposition en sous-systèmes n'est pas une décomposition fonctionnelle. Les fonctions du système sont exprimées du point de vue de l'utilisateur dans la vue des cas d'utilisation. La figure ci-dessous illustre une représentation d'un sous-système.

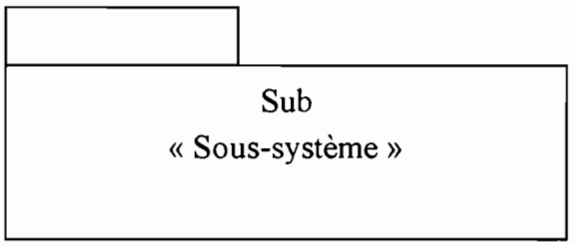


Figure 12 : Représentation d'un sous-système

III-3. La démarche utilisée

III-3-1. La méthode utilisée

La mise en œuvre du portage d'une application mono-utilisateur et monoposte vers un environnement multi-utilisateur et distribué nous amènera à utiliser trois approches de représentation, selon les niveaux du problème :

- la méthode Merise sera utilisée pour la description des modèles de données. En effet, compte tenu du fait que nous aurons à réaliser un modèle conceptuel de données à partir d'un modèle physique, les règles de transformation élucidées précédemment nous seront d'une grande utilité ;
- nous utiliserons aussi un formalisme simple de description de l'enchaînement de certaines tâches auquel nous associerons une légende.
- enfin, le langage UML sera utilisé pour décrire l'architecture des composants et des modules de l'application.

III-3-2. Les outils utilisés

Pour mettre en œuvre le portage vers un environnement multi-utilisateur et distribué du « Financier », nous utiliserons les outils suivants : Borland Delphi 4 client/serveur et AMC Designor Données.

III-3-2-1. Borland Delphi 4 Client/serveur

Nous utilisons Delphi comme environnement de développement de logiciel. En effet, la bibliothèque des composants réutilisables offerte par Delphi permet de disposer d'objets que l'on peut exploiter en passant uniquement par les méthodes et propriétés associées.

Les outils bidirectionnels intégrés à Delphi permettent une mise à jour automatique du code source lorsque des modifications sont apportées aux composants. Cela permet de préserver la cohérence du code source dans les modifications dues au portage.

Le compilateur intégré permet de produire les fichiers exécutables de l'application cible. En outre, le moteur de base de données Borland intégré permet une connectivité avec divers types de base de données ;

Les paquets intégrés permettent de partager du code entre des applications, ce qui permet de réduire la taille des exécutables et d'économiser les ressources systèmes ;

Les ensembles de données distribuées, descendant de *TClientDataSet* permettent de créer des applications indépendamment des bases de données.

III-3-2-2. AMC Designor Données*

Nous choisissons AMC* Designor Données, Version 5.1.0 de Sybase, comme outil d'aide à la conception des modèles conceptuels de données et modèles logiques de données.

En effet, AMC*Designor Données offre les avantages d'une approche de conception à deux niveaux : une possibilité de conserver les modèles tant au niveau conceptuel que physique.

Ainsi, il est possible :

- de concevoir les Modèles Conceptuels de Données (MCD) de systèmes d'information ;

- de générer le Modèle Physique de Données (MPD) correspondant, pour un système de gestion de base de données (SGBD) cible, en tenant compte des spécificités du SGBD choisi ;
- de personnaliser le MPD afin de respecter les contraintes physiques et les performances du produit ;
- de générer un script de création de base de données pour le SGBD cible ;
- d'assurer le *reverse engineering* des bases de données des applications existantes.

DEUXIEME PARTIE : PREPARATION DU PORTAGE

IV. Approche de résolution

Le portage vise à préserver les fonctionnalités de l'application existante, tout en lui donnant des fonctionnalités et un environnement d'exploitation satisfaisants pour l'entreprise et les utilisateurs. Pour cela, nous devons réorganiser les fonctionnalités selon chaque type d'utilisateur et faire une affectation des droits à même de préserver la sécurité et la confidentialité des accès aux données.

Mais une difficulté majeure à laquelle il faut trouver une solution est le code source dont nous disposons, qui ne peut pas franchir l'étape de la compilation. En effet, comme nous l'avions souligné, certains fichiers sources de l'application réalisée sont corrompus et le lancement de l'outil de compilation génère des erreurs. Faut-il reprendre la programmation depuis le point de départ ? Sachant que, sans la résolution de ce problème, aucune autre solution ne peut être envisagée pour une réalisation du portage, nous avons conçu l'architecture ci-dessous *(figures 13)* qui résume la démarche.

IV-1. L'approche conceptuelle

IV-1-1. L'approche de récupération des fichiers corrompus

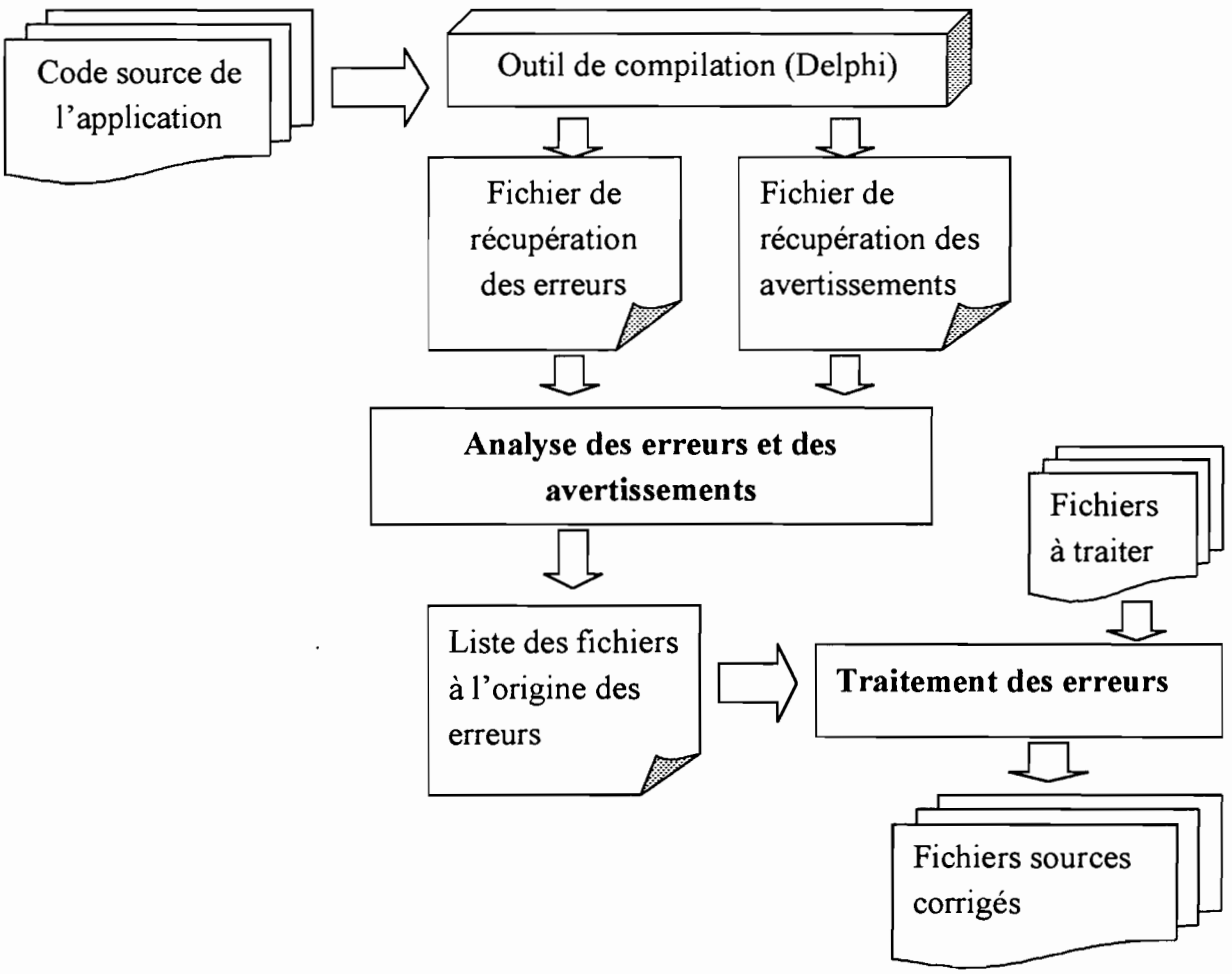


Figure 13 : Démarche de restauration du code source

IV-1-1-1. Structure d'une application Delphi

Une application Delphi est composée :

- d'un projet qui est la racine. Un projet Delphi est une collection de tous les fichiers qui, ensemble constituent une application Delphi ou une bibliothèque dynamiquement liée. Certains de ces fichiers sont générés pendant la phase de conception du projet. D'autres le sont lors de la compilation du code source. Chaque projet Delphi contient du code source Pascal Objet que Delphi compile pour former l'application finale ou la bibliothèque dynamiquement liée. Ce fichier projet est mis à jour tout le long du développement du projet. A la compilation, le compilateur produit sur le disque un fichier exécutable (.EXE) ou une DLL (Dynamic Link Library) ;
- des fichiers fiches (.DFM) qui représentent la partie visible des projets Delphi ;
- des fichiers unités (.PAS) qui correspondent au code source pascal objet rattaché à un fichier fiche. La version compilée d'un fichier unité est stockée dans un fichier binaire .DCU (Delphi Compiled Unit) ;
- des fichiers ressources (.RES) au format Windows standard qui sont utilisés par Delphi (icônes par exemple) ;
- des fichiers paquets .DPL (Delphi Package Library) qui sont des DLL spéciales, et les fichiers paquets sources .DPK (Delphi PacKage) ;
- d'autres fichiers comme les fichiers des options du projet .DOF (Delphi Option File) et le fichier des paramètres de bureau (.DSK).

IV-1-1-2. Traitement des erreurs

La compilation et l'édition de liens recouvrent toutes les opérations nécessaires pour transformer un programme écrit en langage évolué, en un module binaire exécutable dans une machine.

Après avoir soumis l'application à l'outil de compilation, on récupère les informations générées par le compilateur. Ces informations peuvent signaler soit des erreurs, soit des avertissements. L'attention prioritaire sera portée sur les erreurs qui sont souvent les causes majeures du fait que la compilation aboutisse à un échec. Le tableau ci-dessous *(figure 14)* donne une classification des erreurs et l'approche de résolution qu'il faut entreprendre.

Type d'erreur	**Approche de résolution**
Table inexistante	Retro-conception de code
Liens circulaires entre modules	Séparation des modules et intégration progressive
Fichiers fiches endommagés	Conception de nouvelles fiches et restauration du code
Fichiers binaires endommagés	Reprise de la réalisation de l'unité
Erreur interne au code source	Analyse et correction

Figure 14 : Tableau résumant les cas de traitement des erreurs

IV-1-2. L'approche de réalisation du modèle conceptuel de données

Pour mettre en œuvre le portage, il faut disposer du modèle conceptuel de données initial. La question que l'on pourrait se poser est de savoir pourquoi ne pas utiliser le modèle physique de données qui est disponible ? En outre, dans la mesure où un cahier de charges utilisateurs existe, pourquoi ne pas utiliser le modèle conceptuel de données qui y est décrit ?

En effet, dans le souci d'avoir une solution conceptuelle indépendante de tout choix de système de gestion de base de données, il est préférable de ne pas modifier directement le modèle physique de données. Sinon, nous serons obligés de garder une base de données Paradox.

En outre, le modèle conceptuel issu du cahier de charges utilisateurs n'est plus en adéquation avec l'implémentation qui a été réalisée. Par conséquent, bien que le cahier de charges utilisateurs qui est disponible soit une bibliothèque de connaissances du « Financier », il n'est pas fiable d'utiliser le modèle conceptuel qui y est décrit.

Il ne reste donc plus qu'à reconstituer le modèle conceptuel de données à partir du modèle physique de données issu de la base de données Paradox.

IV-1-3. La réalisation du MCD actuel à partir du modèle physique de données

Compte tenu des difficultés énumérées plus haut, la réalisation du modèle conceptuel de données à partir du modèle physique de données se fera comme suit :

- **Etape 1 :** réalisation du modèle physique de données à partir du schéma de la base de données ;
- **Etape 2 :** réalisation des sous-modèles conceptuels de données à partir des tables recensées ;
- **Etape 3 :** unification des sous-modèles conceptuels pour former le modèle conceptuel de données actuel.

Le processus de réalisation du modèle conceptuel de données à partir du modèle physique de données est représenté à travers les schémas ci-dessous. Les étapes une (1) à trois (3) de la *figure 15* présentent la réalisation du MCD actuel.

Le MCD actuel correspond au modèle de l'implémentation initiale du « Financier ».

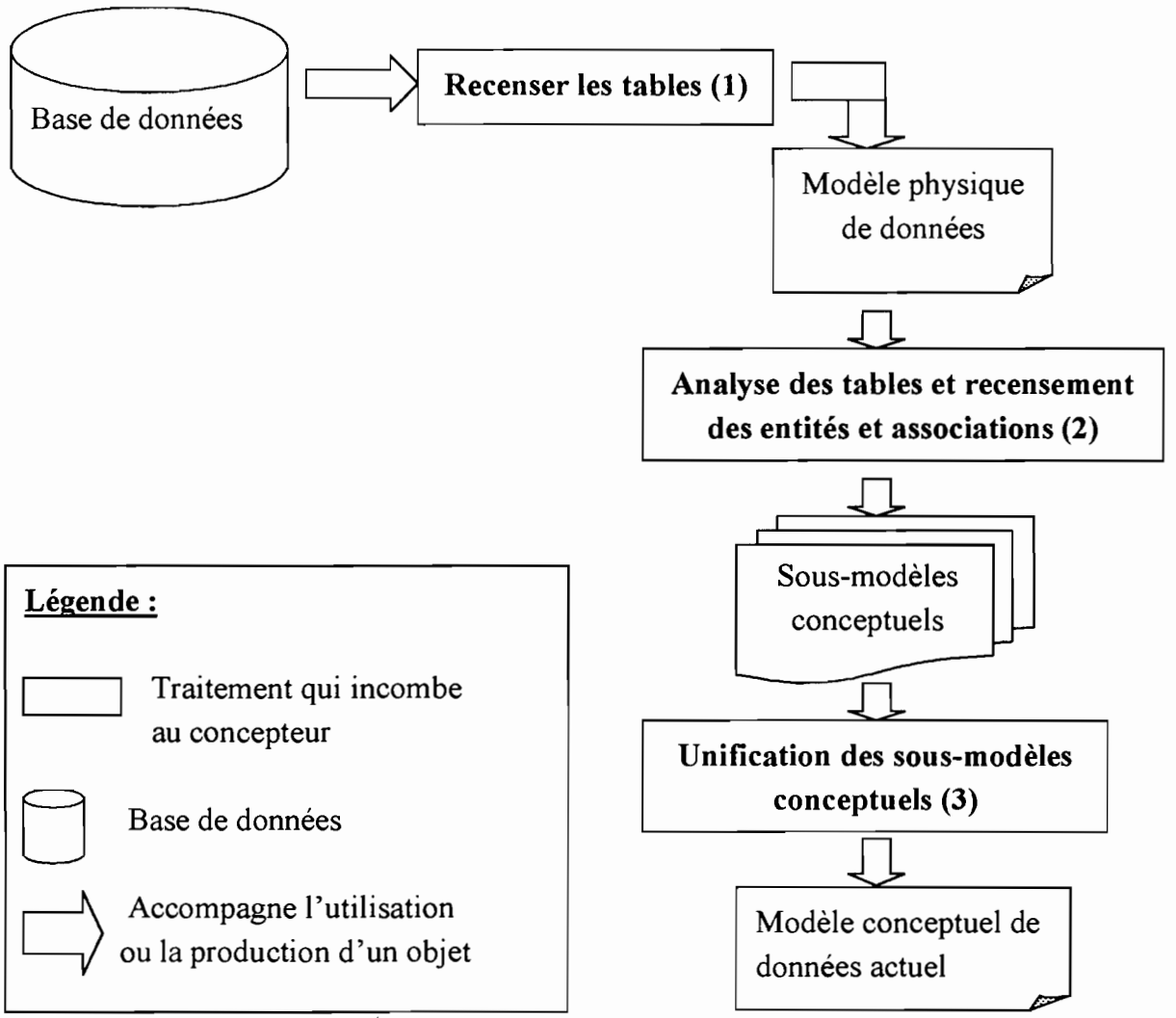


Figure 15 : Démarche de réalisation du MCD actuel à partir du MPD

IV-1-4. La réalisation du modèle conceptuel de données futur

Il est important de déterminer le modèle conceptuel de données futur avant d'envisager une mise en œuvre du portage. En effet, sans une solution conceptuelle stable, les retours-arrières seront à l'origine de perte inutile de temps. Ainsi, il s'agit d'intégrer les nouvelles informations capitales pour l'organisme, qui n'existaient pas lors de la réalisation initiale du « Financier ». En outre, compte tenu du fait que nous voulons suivre certaines données critiques, nous adjoindrons aux attributs existants de certaines tables, de nouveaux attributs. Nous minimiserons néanmoins leur taille afin de ne pas encombrer la base de données par des informations de moindre importance. La démarche de réalisation du MCD futur est résumée par la *figure 16*.

- **Etape 4 :** recensement des nouvelles propriétés à intégrer et prise en compte des règles de gestion ;
- **Etape 5 :** réalisation et vérification du modèle conceptuel de données définitif.

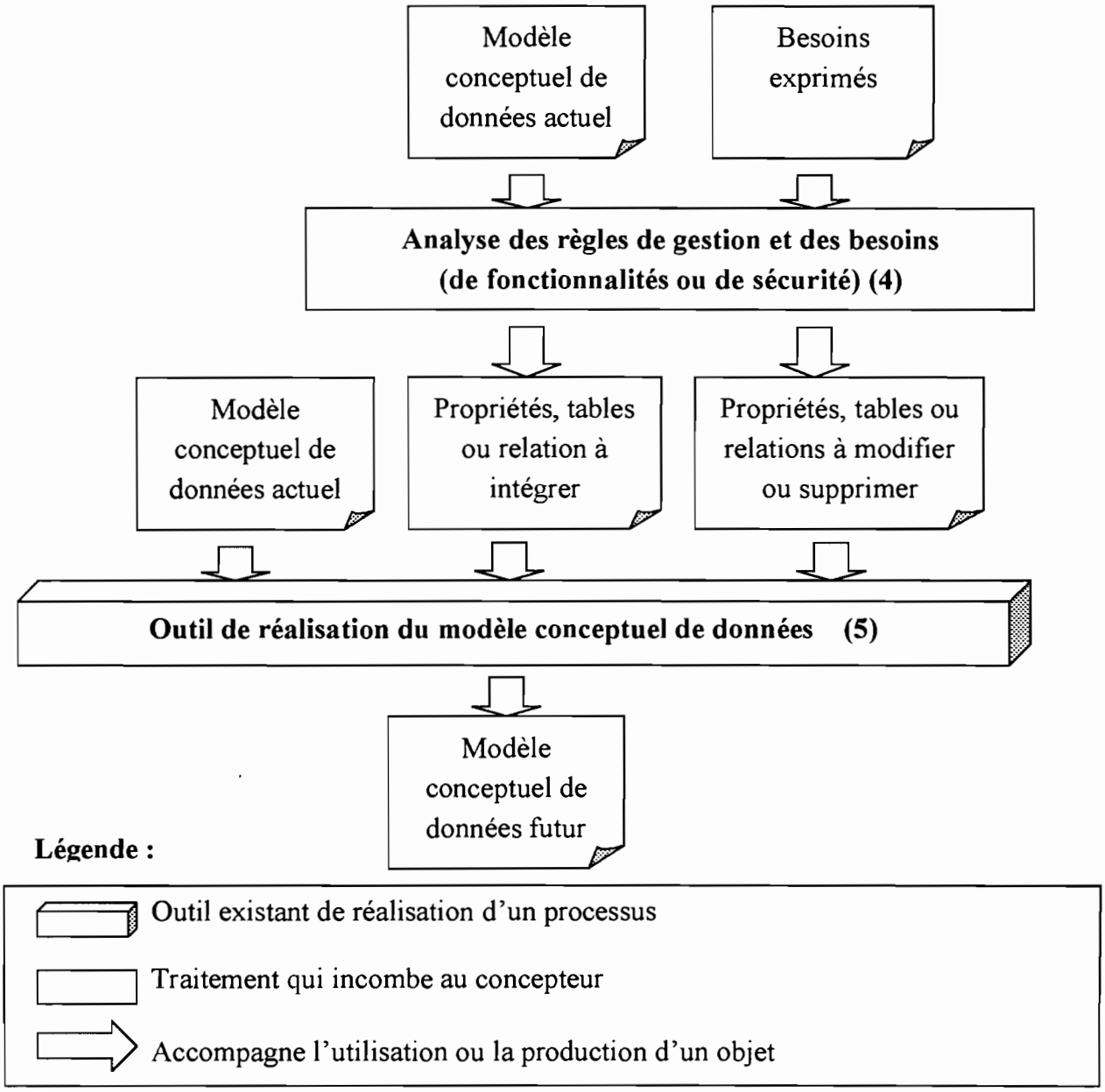


Figure 16 : Démarche de réalisation du modèle conceptuel de données futur

IV-1-5. L'approche de mise en œuvre du portage

Cette mise en œuvre intervient une fois que les deux tâches suivantes aient été exécutées :

- la remise de l'application dans un état cohérent ;
- la mise en place du modèle conceptuel de données futur de l'application.

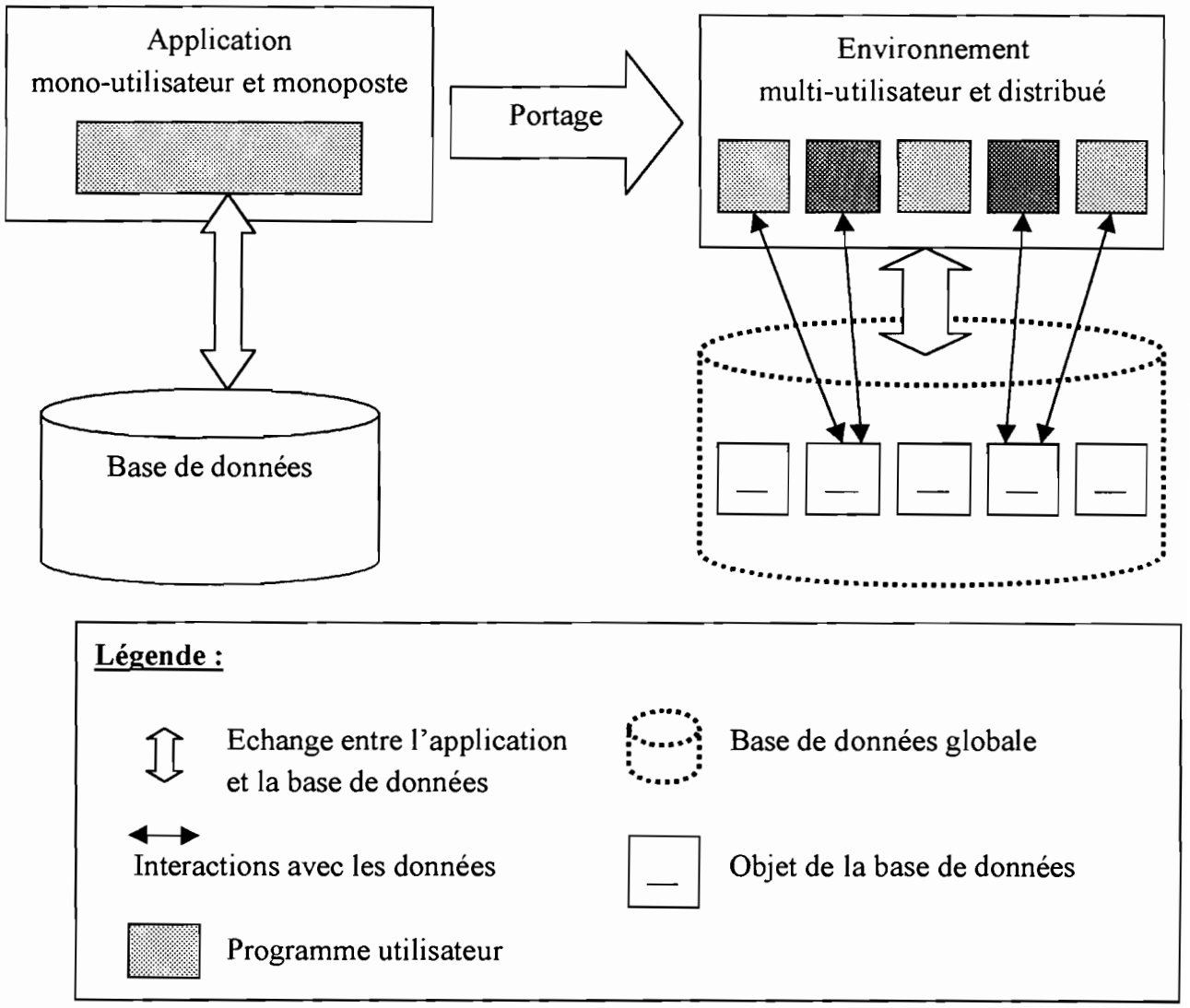


Figure 17 : Architecture globale de mise en œuvre du portage

Ce schéma ci-dessus *(figure 17)* décrit l'architecture globale de mise en œuvre du portage. L'aspect multi-utilisateur doit être pris en compte ainsi que l'hétérogénéité des droits d'accès en fonction de chaque utilisateur.

En outre, il faut adopter une stratégie par rapport à la disponibilité des données pour tous les utilisateurs.

A partir de cette approche globale, on peut proposer deux (2) scénarii de mise en œuvre du portage. Ces scénarii se basent sur l'attention que nous aurons vis-à-vis des données. Ainsi, le point commun à retenir est que tous les scénarii implémenteront les fonctionnalités multi-utilisatrices de la nouvelle application.

Les deux scénarii sont les suivants :

- Scénario 1 : base de données unique centrale avec contrôle des accès concurrents aux données ;
- Scénario 2 : répartition des données par des techniques de fragmentation.

IV-1-5-1. Base de données unique centrale

On peut aussi mettre en œuvre la distribution des traitements tout en utilisant une base de données unique centrale. Toutefois, cette gestion centralisée ne constitue pas un handicape pour l'accès simultané à la base de données. Grâce aux fonctionnalités associées à chaque type d'utilisateur, les traitements seront effectués avec toutes les performances souhaitées. Le contrôle rigoureux qu'il faut effectuer se situe au niveau des droits d'accès et du choix du système de gestion de base de données.

En effet, le contrôle des accès simultanés aux données conduit à la mise en œuvre du verrouillage sur les objets de la base de données. Mais l'application des verrous pose le problème du choix de l'unité de verrouillage. Dans une base de données relationnelle, les objets à verrouiller peuvent être des tables, des pages ou des tuples.

✓ Avantages

La simplicité de mise en œuvre constitue le principal avantage de ce scénario. En outre, comme les utilisateurs exploitent les mêmes informations, l'assurance d'avoir des données à jour permet de préserver la fiabilité de l'information. De même, la possibilité des traitements distribués permet de satisfaire les utilisateurs.

✓ Inconvénients

Au titre des points faibles de ce scénario, on peut citer le manque de transparence de la localité des données. En effet, les utilisateurs se rendent compte que leurs informations ne sont pas stockées en local. En outre, l'exploitation d'une seule base de données implique une homogénéité obligatoire pour les données. Le système mis en place devrait prendre en compte la tolérance de panne pour éviter une perte de toutes les données de l'organisme. Ainsi, l'usage de système implémentant la technologie RAID (Redundant Array of Independant Disks) permet d'apporter une tolérance de panne. Toutefois, les procédures de sauvegarde de données doivent être instaurées pour palier à ces inconvénients.

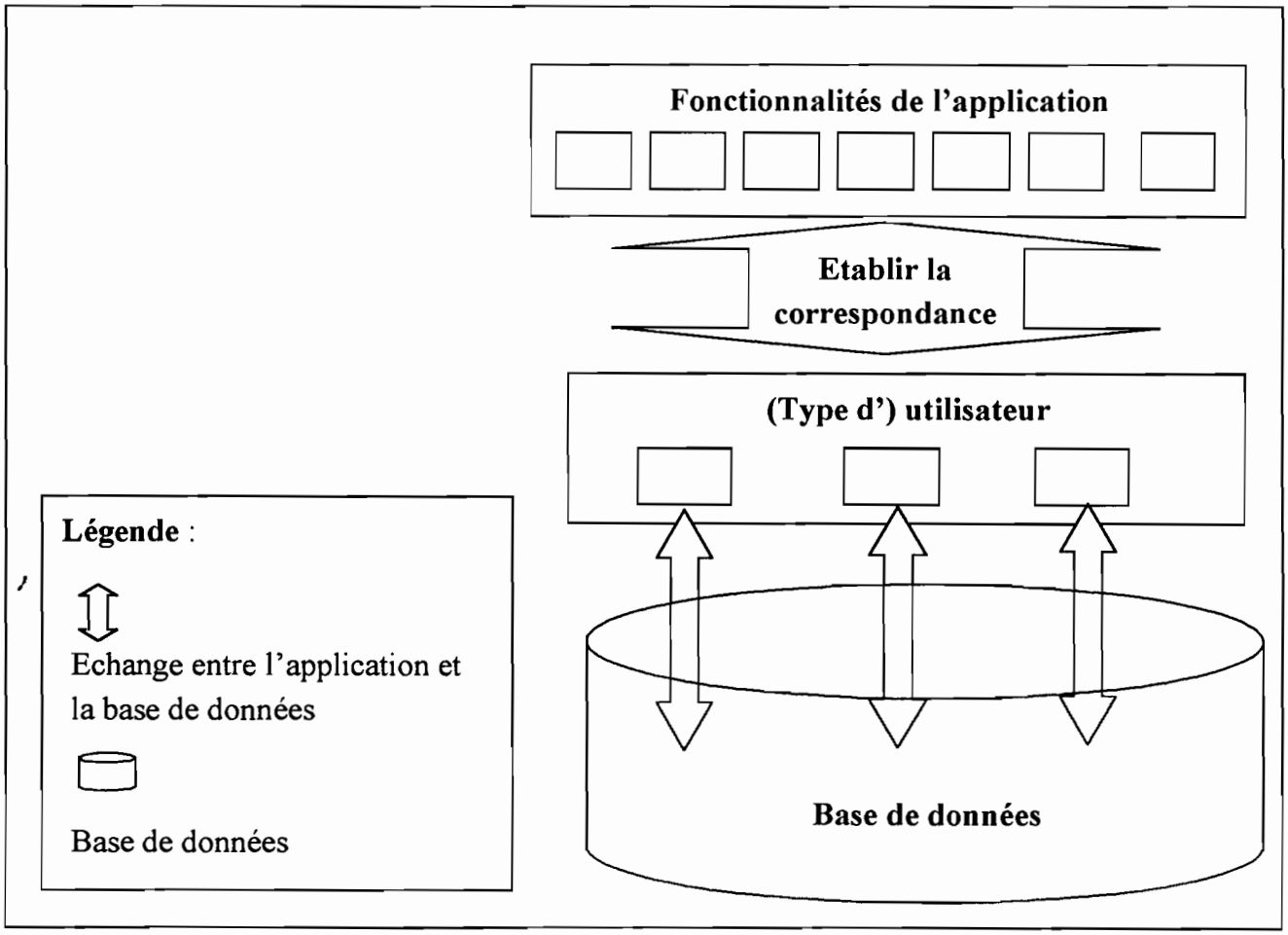


Figure 18 : Base de données unique centrale

IV-1-5-2. Répartition des données

La mise en œuvre de la répartition des données peut se faire à partir d'une technique de fragmentation. On dispose en effet de trois (3) modes de fragmentation : la fragmentation horizontale, la fragmentation verticale et la fragmentation mixte.

La fragmentation horizontale est une sélection d'un certain nombre de tuples correspondants à un critère donné. La fragmentation verticale est une projection sur un ensemble d'attributs d'une table. La table initiale se retrouve par jointure. La fragmentation mixte combine les deux modes de fragmentation précédemment énoncés.

Ces techniques permettent d'identifier des bases de données que l'on pourrait implanter par site. Sa mise en œuvre suscite cependant un certain nombre d'interrogations : quel est le critère de fragmentation à utiliser ? Quelle sera l'amélioration de performance que nous allons obtenir ?

Dans la mise en œuvre de la répartition des données, le contrôle des accès aux données peut conduire à un choix de verrouillage des objets répartis. Dans la mesure où des sites disposent de leur propre système de gestion de base de données, le problème réside dans la gestion des verrous. Autrement, qui verrouille ?

Dans le cas d'un verrouillage centralisé, un site unique est responsable des verrous au moins à un instant donné. Dans une situation de verrouillage décentralisé, chaque serveur gère les verrous de ses données. Le verrouillage centralisé n'est pas adapté à un système réparti lorsque des verrous sont appliqués sur des objets fins (pages ou tuples). C'est pourquoi les systèmes de gestion de bases de données réparties retiennent en général le verrouillage décentralisé. Chaque serveur gère donc ses propres verrous.

✓ **Avantages**

L'avantage de ce scénario est qu'il offre des possibilités d'évolutivité du système. Ainsi, le changement de site n'a pas de grandes incidences sur l'architecture du système. La transparence de la localisation est gérée de façon cohérente de telle sorte que les utilisateurs ne se rendent pas compte que toutes les données auxquelles ils accèdent ne sont pas en local.

✓ **Inconvénients**

Ce scénario illustré par la *figure 19* est en effet complexe à mettre en œuvre. Pour une application comme le « Financier » où des relations multiples lient les tables, la fragmentation sera complexe à réaliser et l'on aboutira à un déséquilibre important dans la répartition des données.

Bien que la répartition des données ne constituera pas notre choix d'implémentation, nous proposons dans le section 5 la démarche de modélisation des bases de données réparties.

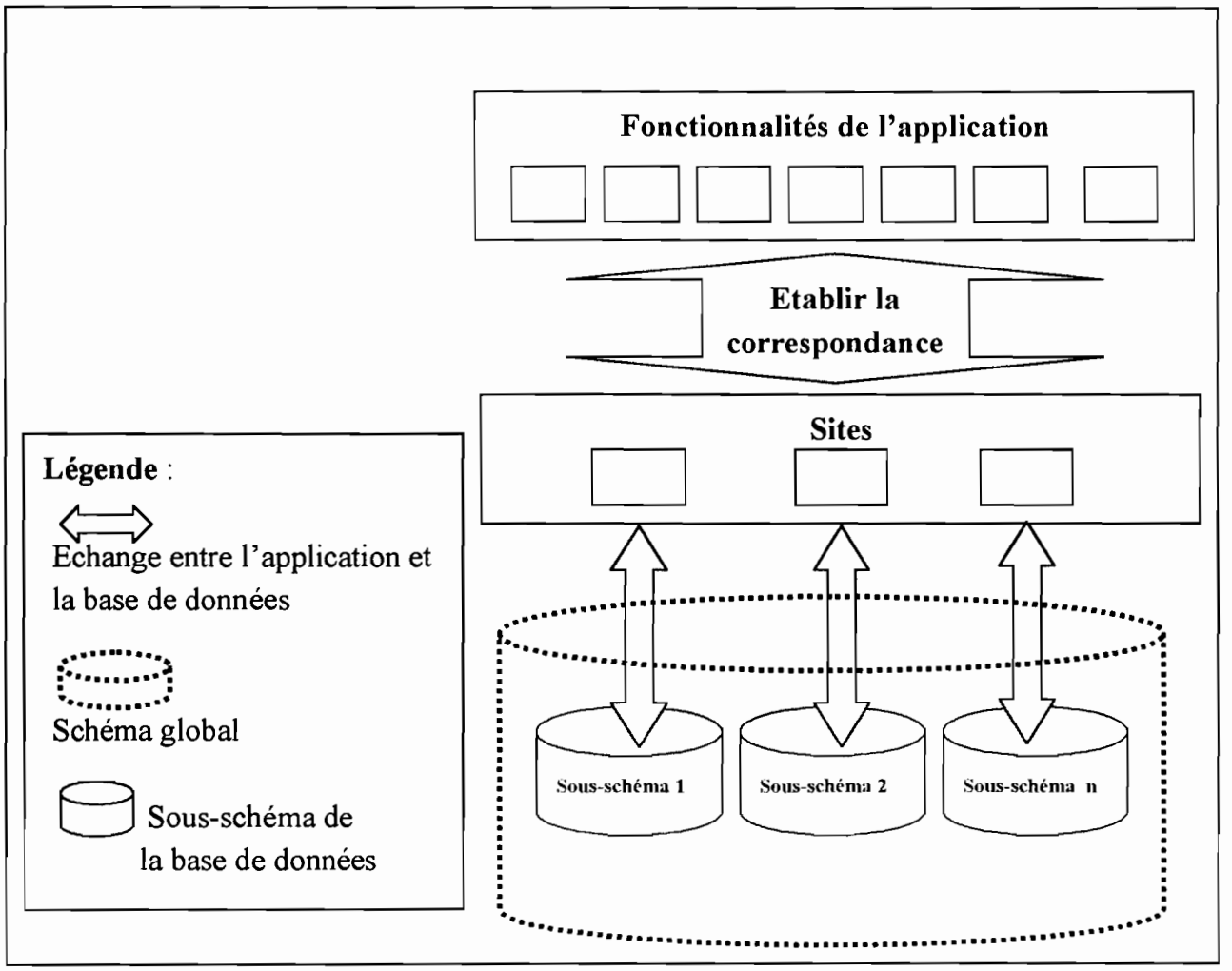


Figure 19 : Architecture distribuée

IV-2. La solution technique au portage

Dans le cadre d'une application mono-utilisateur et monoposte, un seul programme à la fois accède aux données de la base de données. Cependant, dans un environnement multi-utilisateur et distribué, plusieurs programmes accèdent simultanément à la base de données. Les traitements peuvent vouloir accéder ou modifier les mêmes informations. C'est alors que l'on se retrouve dans une situation de concurrences d'accès qu'il faut gérer pour préserver la cohérence et l'intégrité des données.

Par exemple, lorsqu'un client réserve une place d'avion, il ne faut que cette place soit réservée par deux personnes.

Dans le cadre de la mise en œuvre du portage, nous opterons pour une base de données unique. C'est pourquoi les traitements distribués qui accèdent à la base de données doivent être contrôlés pour garantir la fiabilité des données.

Notion de transaction

Une transaction est «une *unité de traitement séquentiel, exécutée pour le compte d'un usager qui, appliquée à une base de données cohérente, restitue une base de données cohérente* » [GARDARIN 1994].

Une transaction est une suite de requêtes dépendantes de la base de données qui doivent vérifier les propriétés d'atomicité, de cohérence, d'isolation et de durabilité.

Le contrôle de concurrence

L'objectif du contrôle de la concurrence d'accès est de rendre invisible aux utilisateurs, le partage simultané des données. Le contrôle s'effectue à l'aide de protocoles spéciaux permettant de synchroniser les mises à jour afin d'éviter les pertes de mises à jour et l'apparition d'incohérences.

Compte tenu des accès concurrents, notre choix se fondera sur des critères liés aux capacités :

- de gérer des concurrences d'accès tout en garantissant des performances de fonctionnement ;
- d'assurer une indépendance entre l'application et le type de base de données.

Deux (2) possibilités s'offrent à nous quant au choix des moyens de contrôle des transactions qui résulteront des accès concurrents des programmes utilisateurs : le contrôle implicite et le contrôle explicite.

IV-2-1. Les scénarii de gestion des concurrences d'accès

IV-2-1-1. Scénario 1 : Contrôle au niveau des données

Dans cette approche, le contrôle des accès concurrents revient à un système de gestion de base de données (SGBD). Il se charge de la résolution des conflits de transactions et procède ensuite à leur annulation ou à leur acceptation. Ainsi, le SGBD contient l'ensemble des contraintes liées aux données.

Les programmes utilisateurs envoient leurs instructions de mise à jour sous forme de requêtes SQL.

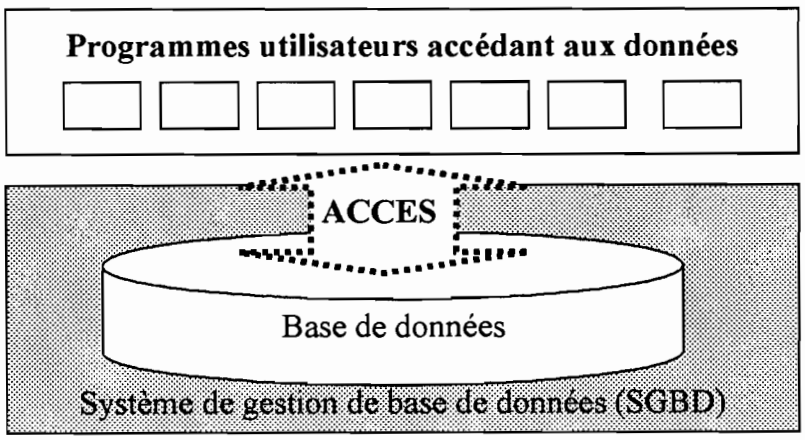


Figure 20 : Contrôle au niveau des données

IV-2-1-2. Scénario 2 : Contrôle au niveau du code applicatif

Ici, le contrôle est effectué dès l'initiation de l'opération. Ainsi, en délimitant de façon explicite les transactions, on arrive à contrôler les programmes utilisateurs qui accèdent à la base de données.

Selon le choix de système de gestion de base de données, un contrôle supplémentaire pourrait être effectué au niveau des données.

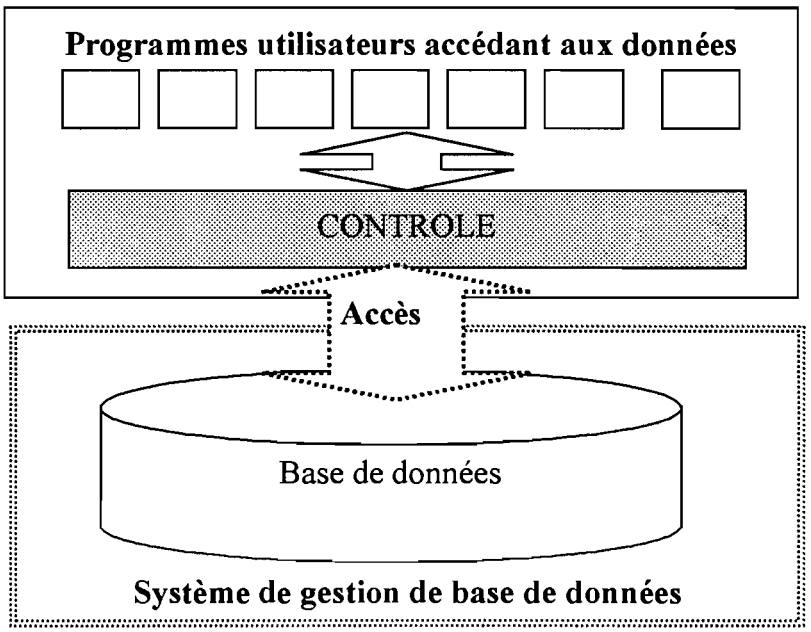


Figure 21 : Contrôle au niveau du code applicatif

IV-2-2. Le choix de composants

IV-2-2-1. Contrôle implicite des transactions

Le moteur de base de données Borland (BDE) offre par défaut un contrôle implicite des transactions au sein des applications. Dans ce cas, Delphi utilise une transaction différente pour chaque enregistrement d'un ensemble de données qui est écrit dans la base de données sous-jacente. Il valide séparément chaque opération d'écriture comme *Post* ou *AppendRecord*. Le contrôle implicite des transactions est facile à utiliser. Il garantit un minimum de conflits pour les mises à jour d'enregistrements et assure une vue cohérente de la base de données. Toutefois, comme chaque écriture d'une ligne dans la base de données s'effectue dans une transaction séparée, il peut conduire à un trafic réseau excessif et ralentir les performances de l'application.

IV-2-2-2. Contrôle explicite des transactions

En contrôlant explicitement les transactions, nous pouvons choisir les moments les plus efficaces pour démarrer, valider ou annuler nos transactions. Si l'application accède à un serveur SQL distant comme Sybase, Oracle, Microsoft SQL Server ou à des bases distantes compatibles ODBC, il est préférable d'effectuer un contrôle explicite des transactions.

Il existe deux (2) manières (mutuellement exclusives) de contrôler explicitement les transactions : utiliser soit les méthodes et propriétés du composant *TDatabase*, soit le composant *TQuery* de *SQL direct*.

✓ **Solution 1 : Utiliser les méthodes et les propriétés du composant TDatabase**

L'utilisation du composant *TDatabase* permet d'avoir une application impeccable et portable, qui ne dépend d'aucune base de données ni d'aucun serveur particulier.

- Les méthodes :
  - *startTransaction* permet de démarrer une transaction ;
  - *Commit* permet de valider une transaction ;
  - *Rollback* permet d'annuler une transaction.
- Les propriétés :
  - *InTransaction* indique si une transaction est en cours d'exécution. L'appel de la méthode *startTransaction* met *inTransaction* à ***true***.
  - *TransIsolation* permet de déterminer un niveau d'isolation *(figure 22)* des transactions pour un composant base de données. Le niveau d'isolement des transactions simultanées détermine comment une transaction interagit avec d'autres transactions simultanées lorsque ces transactions manipulent les mêmes tables.

Valeurs de la propriété *TransIsolation*	**Signification**
TiDirtyRead	Permet la lecture des modifications non validées de la base de données effectuées par des transactions simultanées. Les modifications non validées ne sont pas permanentes ; elles peuvent être annulées à tout moment. C'est le niveau le plus bas d'isolement d'une transaction par rapport aux effets d'autres transactions.
TiReadCommitted	Permet la lecture des modifications validées (permanentes) de la base de données effectuées par des transactions simultanées. C'est la valeur par défaut de la propriété *TransIsolation*.
TiRepeatableRead	Ne permet qu'une seule lecture de la base de données. La transaction ne peut connaître les modifications effectuées ultérieurement par d'autres transactions simultanées. Ce niveau garantit qu'une fois un enregistrement lu par la transaction, l'enregistrement ne change que si la transaction le change. C'est le niveau maximum d'isolement d'une transaction par rapport aux autres transactions.

Figure 22 : Niveaux d'isolement associé à une transaction avec TDatabase

✓ **Solution 2 : Utiliser un composant TQuery de SQL direct**

Avec SQL direct, on peut utiliser un composant *TQuery*, *TStoredProc* ou *TUpdateSQL* pour envoyer directement une instruction SQL de contrôle des transactions à un serveur de base de données. Le moteur de base de données Borland (BDE) ne traite pas l'instruction SQL.

Le composant TQuery permet d'accéder à des données qui se trouvent :

- dans des tables Paradox ou DBASE en utilisant SQL local (il fait partie du BDE). SQL local est un sous-ensemble de la spécification SQL-92. La majeure partie de la syntaxe DML (Data Manipulation Language) et suffisamment de syntaxes DDL (Data Definition Language) sont supportées pour permettre la manipulation de ces types de tables ;
- dans des bases de données de serveur InterBase local en utilisant le moteur InterBase ;
- dans des bases de données de serveur de base de données comme Oracle, Sybase, SQL Server, Informix, DB2 et InterBase. Pour être en mesure d'accéder à un serveur distant, le pilote SQL Link approprié et le logiciel client (fourni par le revendeur) propre au serveur de bases de données doivent être installés. Toute syntaxe SQL standard supportée par ces serveurs est autorisée.

Delphi supporte également les requêtes hétérogènes lancées sur plusieurs serveurs ou types de table (comme par exemple, une table Oracle et une table Paradox). En créant une requête hétérogène, le moteur de base de données Borland (BDE) utilise SQL local pour traiter la requête.

IV-2-3. Les choix retenus

✓ **Le Composant TDatabase**

Au niveau architectural, la solution retenue correspondait à une gestion centralisée des données.

Pour la solution technique, nous retenons le composant *TDatabase* pour le contrôle explicite des transactions. Lorsqu'une application Delphi se connecte à une base de données, la connexion est encapsulée dans un composant *TDatabase*. En fait, le composant base de données encapsule la connexion à une seule base de données dans le contexte d'une session BDE.

Le principal avantage d'utiliser les méthodes et les propriétés d'un composant base de données pour contrôler les transactions est qu'on obtient une application impeccable et portable, qui ne dépend d'aucune base de données ni d'aucun serveur particulier.

✓ **Le Composant TSession**

A chaque composant base de données, est associé un composant session par défaut. Les composants session disposent de deux propriétés qui permettent de se déplacer parmi les composants base de données associés à une session : *Databases* et *DatabaseCount*. Databases est un tableau de composants base de données actuellement actifs et associés à une session. Associée à la propriété *DataCount*, *Databases* permet de parcourir tous les composants base de données actifs afin d'effectuer une action sélective ou globale.

✓ Le Composant TDataSource

Le composant *TDataSource* agit comme canal de communication entre un ensemble de données et les contrôles visuels, orientés donnés, d'une fiche. Chaque ensemble de données doit avoir un composant source de données qui lui correspond. Un tel composant opère la liaison entre les contrôles base de données visuels des fiches et un ensemble de données. Ces derniers canalisent le flux bidirectionnel de données entre un composant ensemble de données et les contrôles visuels.

✓ Le Composant TTable

Un composant TTable permet d'accéder à chacune des lignes (enregistrements) et des colonnes (champs) d'une table sous-jacente, qu'elles proviennent de Paradox, Dbase, Access, FoxPro, ou d'une base de données compatible ODBC, ou encore d'une base de données SQL installée sur un serveur distant comme Interbase, Sybase ou SQL Server.

✓ Le Composant TField

Les descendants du composant TField représentent les colonnes d'une table associée à une base de données.

IV-2-4. L'architecture technique de la solution

Après avoir décrit les solutions conceptuelles possibles à la mise en œuvre du portage du «financier » dans un environnement distribué, nous retenons la solution caractérisée par :

- une gestion multi-utilisateur au niveau de la répartition des fonctionnalités ;
- un choix de distribution des traitements au niveau des postes de travail ;
- un choix de gestion centralisée, avec contrôle de l'accès concurrent aux données.

En outre, nous proposons un suivi des opérations critiques tout en optimisant l'évolution de la taille de la base de données.

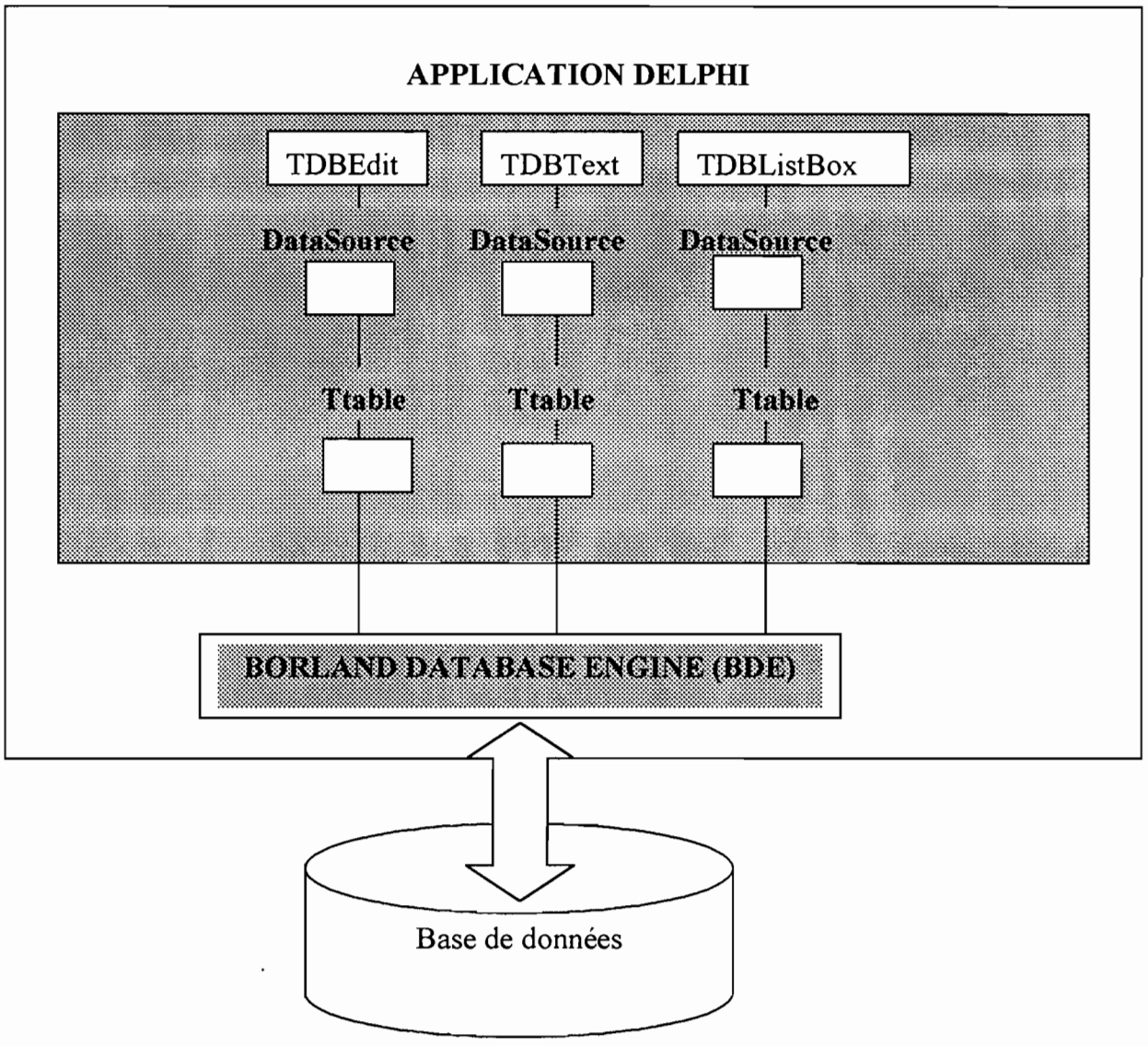


Figure 23 : Architecture technique du système

IV-3. Les résultats de la préparation du portage

IV-3-1. La restauration des fichiers de l'application

Dans la mise en œuvre de la démarche de récupération des fichiers corrompus, nous avons été confrontés à plusieurs situations résumées dans le tableau ci-dessous *(figure 24)*.

La majeure partie des cas sont relatifs à des modules entiers, dont les références à d'autres fichiers inexistants ou dégradés ne permettent pas la compilation. Ainsi, il a été indispensable de dissocier ces modules afin d'examiner et de rétablir la structuration du code source.

✓ **Liste des fichiers concernés**

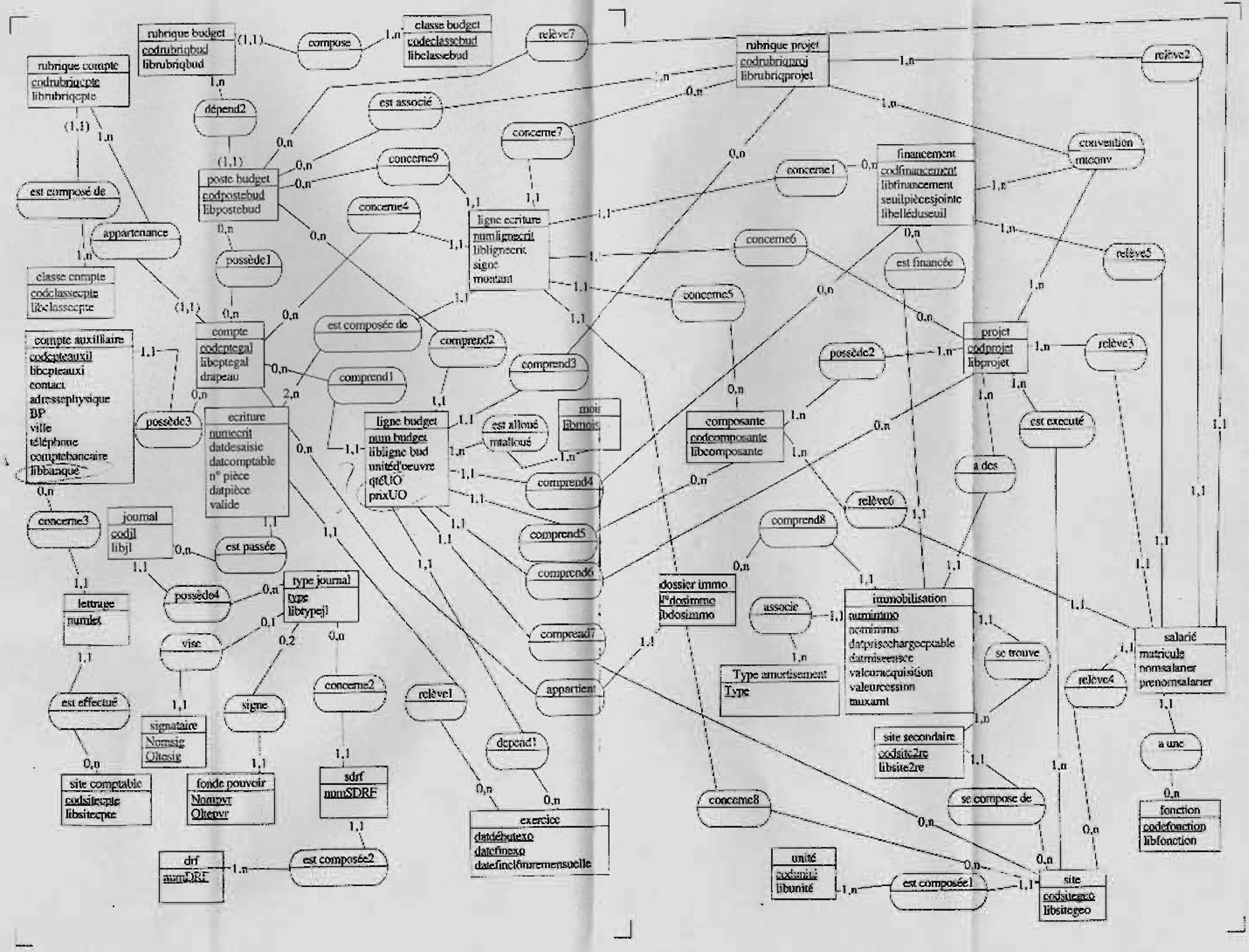
Nom du fichier	**Ext.**[1]	**Problème décelé**	**Traitement infligé**
Convention	.db	Table non retrouvée par le programme	Création de la table à partir des propriétés recensées dans le code source
SaisieJ	.pas	Déséquilibre de parenthèse	Restauration du code
Uconsultecricpteaux	.pas	Extension du fichier incorrecte	Restauration de l'extension
Uconvention	.dfm	Impossibilité d'ouvrir le fichier	Création d'une nouvelle fiche, et restauration du code Pascal Objet
Ucpteaux	.dcu	Format 32 bits non reconnu	Recherche d'un fichier de version antérieure identique puis remplacement
Uconnectdb	.dfm	Version de fichier incorrecte	Mise à jour des changements
Exercice	.db	Deux versions de tables distinctes	Mise à jour des changements

Figure 24 : Tableau résumant des situations de restauration du code source

IV-3-2. Le modèle conceptuel de données (MCD) initial

Ce modèle est tiré du cahier de charges utilisateurs réalisé en 1997. Il est présenté dans la figure ci-après *(figure 25)*.

[1] Il s'agit de l'extension associée à ces différents fichiers.



rubrique budget
codrubriqbud
librubriqbud
compose
classe budget
codeclassebud
libclassebud
relève7
rubrique projet
codrubriqproj
librubriqprojet
relève2
rubrique compte
codrubriqcpte
librubriqcpte
dépend2
est associé
concerne7
convention
mtconv
est composé de
poste budget
codpostebud
libpostebud
concerne9
concerne1
financement
codfinancement
libfinancement
seuilpiècesjointe
libelléduseuil
appartenance
concerne4
ligne ecriture
numligneecrit
libligneecrit
signe
montant
classe compte
codclassecpte
libclassecpte
possède1
concerne6
est financée
relève5
concerne5
compte
codecptegal
libcptegal
drapeau
est composée de
comprend2
projet
codprojet
libprojet
relève3
compte auxiliaire
codcpteauxil
libcpteauxi
contact
adressephysique
BP
ville
téléphone
comptebancaire
libbanque
comprend1
comprend3
possède2
possède3
ecriture
numecrit
datdesaisie
datcomptable
n° pièce
datpièce
valide
ligne budget
num budget
libligne bud
unitéd'oeuvre
qtéUO
prixUO
est alloué
mtalloué
mois
libmois
composante
codcomposante
libcomposante
est executé
a des
comprend4
relève6
concerne3
journal
codjl
libjl
est passée
comprend5
comprend8
comprend6
lettrage
numlet
possède4
type journal
type
libtypejl
dossier immo
N°dosimmo
libdosimmo
associe
immobilisation
numimmo
nomimmo
datprisechargecptable
datmiseenrce
valeuracquisition
valeurcession
tauxamt
salarié
matricule
nomsalarier
prenomsalarier
vise
comprend7
se trouve
relève4
Type amortisement
Type
est effectué
signe
concerne2
relève1
appartient
signataire
Nomsig
Qltesig
depend1
site secondaire
codsite2re
libsite2re
a une
site comptable
codsitecpte
libsitecpte
fonde pouvoir
Nompvr
Qltepvr
sdrf
numSDRF
concerne8
se compose de
fonction
codefonction
libfonction
exercice
datdébutexo
datfinexo
datefinclôturemensuelle
drf
numDRF
est composée2
unité
codunité
libunité
est composée1
site
codsitegeo
libsitegeo

IV-3-3. Le modèle conceptuel de données (MCD) actuel

Ce modèle décrit l'implémentation réelle qui a été faite. On notera un changement aussi bien de la structure qu'au niveau de la définition des propriétés. Il a été réalisé à partir de la démarche de retro-conception de code et de reverse engineering.

Elle consiste comme décrite dans l'explication de la démarche conceptuelle, à réaliser des sous-modèles conceptuels à partir du modèle physique que nous ressortons de la base de données. Notons néanmoins que certains attributs qui ont été ajoutés au niveau des tables du modèle physique pour des besoins d'optimisation n'apparaissent pas dans ce modèle (pour respecter le formalisme du MCD).

IV-3-3-1. Retrouver les relations (1, 1) – (, n)*

Il s'agit de recenser les tables où il y a eu migration de clé. La table qui contient la clé externe correspond à une entité. Cette entité se situe dans la partie (1, 1) de la relation, l'autre entité se situant dans la partie (*, n).

On remarque *(figure 26a)* que CdeBanque se retrouve dans Agence. Cela veut dire qu'il existe une relation entre les entités Banque et Agence : (1,1) du côté de Agence et (*, n) du côté de Banque.

Figure 26a : Exemple de tables de la base de données Paradox existante

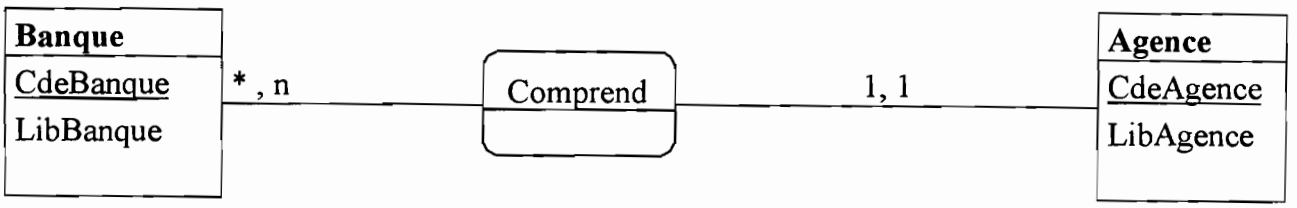


Figure 26b : MCD résultat obtenu issu de la transformation

IV-3-3-2. Retrouver les relations (, n) – (*, n)*

Il s'agit de retrouver les tables qui comportent des clés issues de tables. En effet, ces tables qui sont issues de la transformation du MCD en MLD permettent de déterminer qu'il y a eu une relation (*, n) - (*, n) entre les entités.
Ci-dessous, on note que les propriétés de la table **Poste_Rubrique** sont des clés issues des deux autres tables. Cela permet de déduire *(cf. figures 27a et 27b)* l'existence d'une relation de type (*, n) – (*, n).

Exemple : Tables Paradox initiales

Poste
CdePoste
LibPoste

Poste_Rubrique
CdePoste
CdeRubrProj1

RubrProj1
CdeRubrProj1
LibRubrProj1

Figure 27a : Exemple de tables reflétant les cas de relation (*, n) - (*, n)

Résultats obtenus

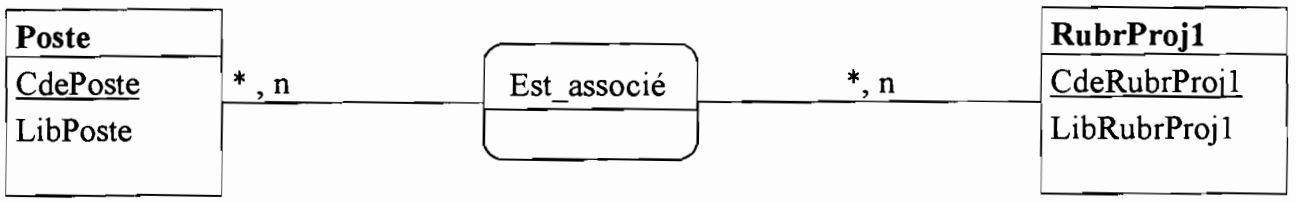


Figure 27b : Exemple de transformation pour retrouver un MCD

IV-3-3-3. Prise en compte du MCD initial

Compte tenu du fait que l'application de ces règles ne permet pas d'avoir des résultats complètement fiables, il est important de considérer les règles de gestion décrites dans le modèle conceptuel initial. Pour certains cas ambigus, il est préférable de se référer aux utilisateurs. Il s'agit notamment des cardinalités (0,1) auxquelles il faut parfois des précisions.

Dans ce processus de réalisation du MCD correspondant à la situation de réalisation du «Financier », on constate une différence par exemple au niveau de l'entité Banque où une entité Agence a été considérée. La réalisation a mis en évidence deux entités, Banque et Agence. On note aussi que l'entité Agent n'a pas été considérée dans la réalisation *(cf. figures 25 et 28)*.

En plus de ces différences, on note qu'il y a des différences sémantiques énormes entre les informations contenues dans le MCD issu du cahier de charges utilisateurs et le MCD de l'implémentation.

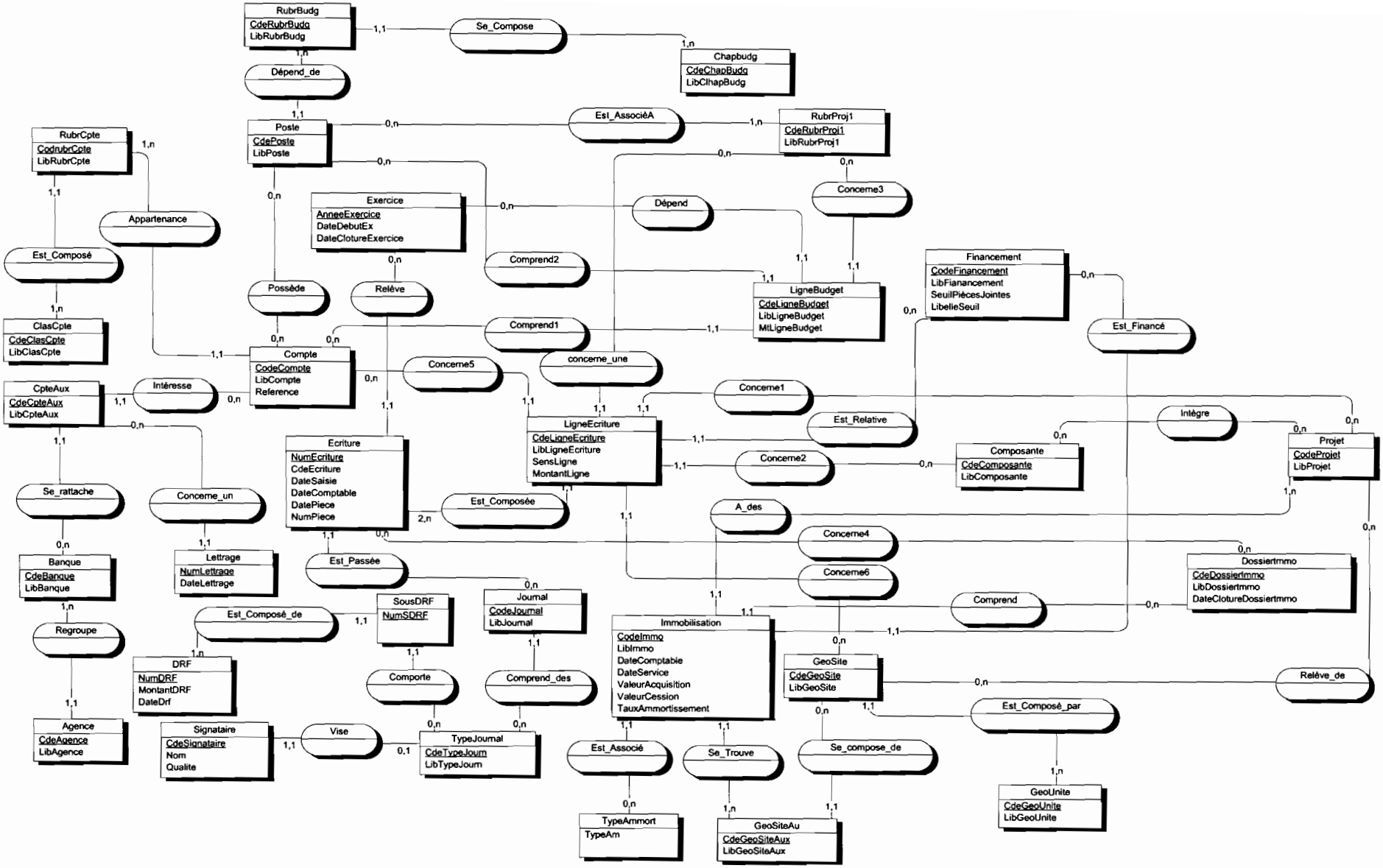


Figure 28 : Modèle conceptuel de donnée correspondant à l'implémentation initiale du « Financier »

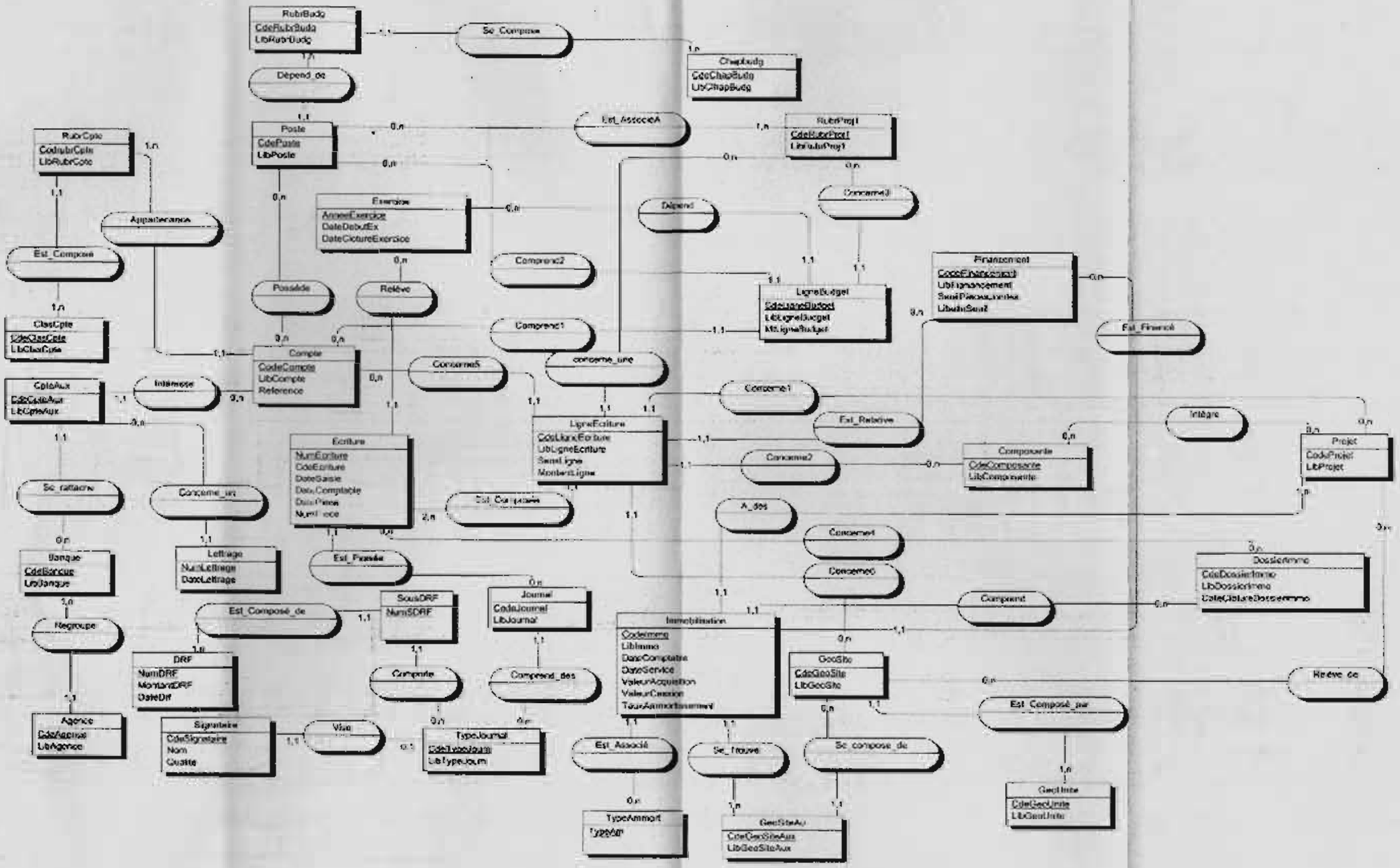


Figure 28 : Modèle conceptuel de donnée correspondant à l'implémentation initiale du « Financier »

TROISIEME PARTIE : REALISATION DU PORTAGE

V. MODELISATION D'UNE BASE DE DONNEES REPARTIES

Un site [GALACSI 1989] est un système informatique autonome, associé à un lieu géographique :

- *mémorisant, en ce lieu, des données sur un support informatique,*
- *permettant la soumission et la réception de données en des points terminaux,*
- *relié aux systèmes informatiques des autres sites par un réseau de communication à distance.*

On suppose qu'en un seul lieu géographique, il n'existe qu'un seul site et que les terminaux sont répartis par lieu géographiquement, c'est-à-dire qu'un terminal est relié directement à un et un seul site.

Un système réparti est un ensemble de types de sites. Chaque type de sites regroupe des sites effectuant le même ensemble de fonctions dans le système d'information, équipé d'un système informatique autonome et relié par un réseau de transmission.

Le portage d'une application mono-utilisateur vers un environnement multi-utilisateur fait apparaître la nécessité de gérer les concurrences d'accès aux données et la cohérence des traitements concurrents. En outre, le choix d'une répartition des données met en exergue la nécessité de disposer d'une stratégie efficace de répartition à travers les différents sites qui constituent le système réparti.

Sans cette répartition adéquate des données, le système qui exploite les données réparties perdra ses performances de traitements. Cela est résolu par la réalisation de modèle conceptuel de répartition, de modèle logique de répartition et de modèle physique de répartition.

V-1. Le modèle conceptuel de répartition

Lorsque l'on parle de portage d'une application monoposte et mono-utilisateur vers un environnement multi-utilisateur et distribué, le défit majeur réside dans la stratégie de répartition des données à travers les différents sites. Cette répartition qui se fait à partir du schéma global de la base de données vise à réaliser un système distribué garantissant une efficacité de fonctionnement.

V-1-1. Les schémas conceptuels

Le schéma conceptuel d'un système réparti est formé par le schéma conceptuel global de l'ensemble du système et de schémas conceptuels par types de sites. Ces schémas sont établis selon le modèle entité-association (ou le modèle relationnel) de données. Ces schémas ne sont pas complètement indépendants entre eux. Le schéma conceptuel global doit prendre en compte les concepts explicités dans les schémas de chaque type de sites.

Un sous-schéma B d'un schéma A est un schéma dont les éléments appartiennent au schéma A. La décomposition d'un schéma A en un ensemble de sous-schémas B1, B2, ..., Bn, est telle que chaque élément du schéma A appartient au moins à un sous-schéma Bi.

V-1-2. Les schémas conceptuels répartis par le modèle entité-association

Notion de sous-schéma

Un schéma B est dit sous-schéma d'un schéma A si :

- chaque type d'entités de B correspond à un seul type d'entité de A de même clé primaire. Les entités d'une réalisation de B correspondent à des entités de A de même valeur de clé primaire ;
- chaque type d'associations de B correspond à un type d'associations unique de A : les associations d'une réalisation de B sont des associations de A du type correspondant ;
- chaque attribut de B est un attribut de A.

Réunion de schémas

Un schéma A est une réunion des schémas B1, B2, ... Bn si :

- chacun des schémas Bi, ... Bn est un sous-schéma de A ;
- tout type d'entités de A correspond à au moins une entité d'une réalisation Sji d'un schéma Bi, les valeurs de clé primaire étant identiques ;
- tout type d'associations de A correspond à au moins une association d'une réalisation Sji d'un schéma Bi ;
- tout attribut de A est attribut d'au moins un schéma Bi.

Aucune condition n'est imposée sur les cardinalités minimum. Une cardinalité minimum 0 (resp. 1) peut correspondre, dans les sous-schémas, à des cardinalités 1 (resp. 0). Si une cardinalité maximum est n dans un schéma B, elle doit être n dans le schéma global.

Modélisation d'un système réparti

Le modèle entité-association d'un système réparti est composé du schéma global A du système et d'autant de sous-schémas B1, ... Bn que de types de sites. On complète chaque type d'entités d'un sous-schéma, par les critères de sélection dans le schéma global.

Dans un sous-schéma, un type d'entités peut correspondre à un sous-ensemble strict du type d'entités global. On a alors une sélection horizontale. Un prédicat précise le critère représentant le type d'entités dans le sous-schéma.

V-1-3. Méthode d'élaboration du modèle conceptuel de répartition

Présentation de la méthode [GALACSI 1989]

Dans le cas d'un système non réparti, le schéma conceptuel est établi par la recherche des rubriques et de leurs liaisons, puis par l'établissement d'un schéma selon le modèle entité association. La recherche des rubriques et de leurs liaisons s'appuie sur l'expression des besoins résultants de la collecte des informations. Dans un système réparti, la collecte peut se faire par type de sites, suivi d'un regroupement global.

Etape 1 : Etablissement d'un premier schéma conceptuel par type de sites

Cette première étape consiste à élaborer un schéma conceptuel pour chaque type de sites.

Etape 2 : Etablissement du schéma global

On établit le schéma global indépendamment des schémas établis par type de sites.

Etape 3 : Vérification de la cohérence et établissement des schémas définitifs

On vérifiera dans cette étape, que les schémas par type de sites sont des sous-ensembles du schéma global. Cette vérification peut faire apparaître des incohérences ou des conditions non vérifiées ; les schémas par type de sites et le schéma global sont alors ajustés afin d'obtenir cette cohérence.

Etape 4 : Validation du schéma conceptuel global comme réunion des sous-schémas

Etape 5 : Schémas conceptuels par types de sites.

Au cours de cette dernière étape, on vérifie que les corrections effectuées dans l'étape 3 assurent la cohérence du schéma global avec les sous-schémas. On vérifiera aussi qu'il n'existe pas de polysémie[2] dans les noms donnés à chacun des concepts. Il est important de faire cette étape séparément par type de site. Il faut définir des appellations identiques pour les types d'entités et les attributs de types d'associations, communs à plusieurs types, et choisir des options compatibles entre les types de sites.

V-2. Le modèle logique de répartition

V-2-1. Schéma logique

Le schéma logique de répartition est constitué d'un ensemble de schémas logiques :

- un schéma logique global,
- un schéma logique par type de sites.

Les schémas logiques par type de sites sont des sous-schémas du schéma logique global. Ainsi, le schéma logique global apparaît comme la réunion des sous-schémas logiques. « Un schéma logique est constitué de lots de données, de liaisons conceptuelles, de points d'entrées, de liaisons d'usage, de valeurs (cardinalités moyenne, minimale, maximale) d'une liaison d'usage. » [GALACSI 1989].

V-2-2. Sous-schéma logique

Si A et B sont deux schémas logiques, le schéma B est dit sous-schéma logique de A s'il vérifie les conditions suivantes :

Les lots de données

Tout lot de données LB du schéma B correspond à un lot de données LA du schéma logique B, de même clé primaire ou de même clé implicite. Les rubriques de LB sont des rubriques de LA. L'ensemble des occurrences de LB est un sous-ensemble des occurrences de LA

[2] Nom d'attribut différent désignant la même propriété

La Liaison conceptuelle

Une liaison conceptuelle du schéma B entre deux lots de données LB et LB' correspond à une liaison conceptuelle des lots LA et LA' correspondants dans le schéma global A. Les cardinalités maximales de la liaison conceptuelle du sous-schéma sont inférieures ou égales aux cardinalités maximales de la liaison conceptuelle du schéma global. Il n'y a aucune règle générale pour les autres cardinalités minimales ou moyennes.

Les points d'entrée

Les points d'entrée dans un lot de données LB du schéma B sont des points d'entrée dans le lot LA correspondant dans le schéma A.

Ils correspondent à des requêtes de type de sites qui sont des requêtes du système global.

La Liaison d'usage

Une liaison d'usage d'un lot de données LB vers un lot de données LB', du schéma B doit être une liaison d'usage entre les lots de données LA et LA', du schéma global.

La valuation

Aucune relation n'est imposée entre le nombre de parcours du schéma B et le nombre de parcours du schéma A, en dehors de la condition d'infériorité. Si un sous-schéma B représente plusieurs sites de même type, le nombre de parcours sera le nombre moyen des différents sites, alors que, dans le schéma global, on aura la totalisation des parcours de tous les sites qui peuvent être de types différents.

V-2-3. Réunion de schémas logiques

Un schéma logique A est dit réunion des sous-schémas logiques B1, ... Bn, si les conditions suivantes sont vérifiées :

Les Sous-schémas

Chacun des schémas Bi ($1 <= i <= n$) est un sous-schéma de A.

Les lots de données

A tout lot de données LA de A, il correspond au moins un sous-schéma Bi et un lot de données LBi de Bi associé à LA ayant la même clé primaire. Toute occurrence du lot de données LA appartient à une réalisation du lot de données associé LBi. Chacune des rubriques de LA appartient à au moins un lot de données associées LBi.

Les liaisons conceptuelles

A toute liaison conceptuelle entre deux lots LA et LA' de A peut être associée une liaison conceptuelle entre des lots de données LBi et LBi' d'un schéma Bi.

Les points d'entrée

A tout point d'entrée du schéma A, est associé un point d'entrée d'au moins un des sous-schémas Bi.

Les liaisons d'usage

A toute liaison d'usage du schéma A d'un lot de données LA, sur un lot de données LA', est associée une liaison d'usage entre les lots de données associés LBi et LBi' d'au moins un sous-schéma Bi.

Les liaisons d'usage

A toute liaison d'usage du schéma A d'un lot de données LA, sur un lot de données LA', est associée une liaison d'usage entre les lots de données associés LBi et LBi' d'au moins un sous-schéma Bi.

Le nombre de parcours d'une liaison d'usage de A est la somme des nombres de parcours pour cette liaison d'usage dans toutes les réalisations des sous-schémas associés.

V-2-4. Méthode d'établissement du schéma logique de répartition

La méthode d'établissement du schéma logique, consiste dans une première phase à établir une première version du schéma logique global et des schémas logiques par type de sites. Ensuite, les conditions de cohérence de ces schémas seront vérifiées, les incohérences notées. Dans un dernier temps, le schéma global et les schémas par type de sites seront modifiées afin d'éliminer ces incohérences.

Etape 1 : établissement de la première version des schémas logiques

L'établissement de la première version des schémas conceptuels se fait à travers le recensement des requêtes et l'élaboration autonome des schémas logiques.

Recensement des requêtes

Les requêtes seront recensées par type de sites. Il s'agit, pour un type de sites, de recenser les requêtes soumises à l'un des sites du type. Par rapport à la notion de requêtes sans répartition, est ajoutée la notion de site de soumission de la requête c'est-à-dire, du système informatique auquel cette requête est soumise.

Elaboration autonome des schémas logiques

Le schéma logique global ainsi que les schémas logiques locaux sont établis à partir du schéma conceptuel global et des schémas conceptuels des types de sites, des requêtes par type de sites et des requêtes du système global. Ce travail est effectué séparément pour chaque schéma. Cependant, pour un type de sites, les cardinalités moyennes des liaisons conceptuelles et les évaluations des liaisons d'usage correspondent à des moyennes portant sur les différents sites du type.

Vérification des règles de réunion de sous-schémas

A partir des schémas logiques établis, les règles de réunion de sous-schémas sont à vérifier pour le schéma global et les schémas par type de sites. On note les cas de non-vérification des règles.

Etape 3 : unification de schémas

Les incohérences notées dans l'étape précédente sont éliminées par l'ajout éventuel de lots de données dans les types de sites. Les parcours nécessaires pour établir les requêtes sont unifiées dans le schéma global et dans les schémas par type de sites. Dans cette unification, certaines règles d'usage pourront être ajoutées ou supprimées dans certains schémas. A un même type de requête du schéma global, et des schémas par type de sites, correspond le même parcours. Dans les schémas logiques par type de sites, seront aussi indiquées les critères de sélection des occurrences de lots de données. A la fin de cette étape, le schéma logique global est la réunion des schémas logiques par type de sites qui en sont les sous-schémas.

V-3. La répartition physique des données

Cette répartition qui se conçoit par type de sites permet de déceler les données physiques à stocker selon les types de sites. En outre, selon les urgences liées à la recherche de l'efficacité dans les traitements, il peut être envisagé des choix de duplication de données.

L'implantation physique de la base de données sur chaque site est effectuée à partir du schéma physique de répartition. Le schéma de répartition physique est élaboré à partir des schémas logiques (global et par type de sites).

V-3-1. Contexte logiciel et matériel de répartition

Par hypothèse, chaque site doit avoir :

- son propre matériel,
- un système de gestion de base de données,
- un réseau de télécommunication avec les autres sites.

Eventuellement, on peut avoir un logiciel global de gestion de données reparties. L'architecture matérielle a aussi une incidence sur la répartition. Dans certains cas, les données pour passer d'un site à l'autre peuvent transiter par un ou plusieurs sites. L'architecture influe donc sur le volume de transmission des données et par conséquent sur les coûts.

La répartition des données envisagée a pour but de préciser, dans chaque site, les données qui y sont conservées, la duplication de données sur plusieurs sites et enfin les transferts de données entre sites. Cette répartition doit être faite afin de pouvoir évaluer les requêtes et minimiser les coûts de transmission.

L'implantation physique s'effectue suivant deux (2) étapes :

- la minimisation des coûts de transmission entre sites par le choix de données implantées sur chaque site ;
- la minimisation du coût de l'implémentation sur chaque site par le choix d'une organisation physique de la base de données sur le site.

Pour mettre en œuvre la minimisation des coûts de transmission, on établit un schéma des données implantées sur chaque site, et la liste des échanges entre sites. Afin de conserver l'homogénéité entre types de sites, on prévoit de conserver le même schéma par type de sites et on établit la liste des échanges par type de sites.

V-3-2. Le modèle physique de répartition

Un modèle physique de répartition est constitué pour chaque type de sites :

- d'un schéma de répartition physique, constitué des lots de données et des liaisons d'usage locaux ;
- d'une liste des types d'échanges émis ; elle précise les destinataires, les nombres moyens d'émissions durant une période de référence et le volume des messages.
- du dictionnaire de répartition des données qui indique les implantations des diverses rubriques et les duplications éventuelles.

V-3-3. Méthode d'élaboration du schéma physique de répartition

Cette méthode décrit en quatre (4) étapes, l'élaboration du schéma physique de la répartition des données.

Première étape : La fragmentation des lots de données

Un site n'exploite jamais toutes les données du schéma global. C'est pourquoi en regroupant les sites par type de sites, on peut définir les données qui sont utilisées par chaque type de sites. A partir des données manipulées par types de sites, on peut définir les données relatives à chaque site. Cette répartition qui s'effectue par des techniques de fragmentation permet de définir les données à implanter dans un site à partir des procédés de sélection et de projection sur les lots de données du schéma global de la base de données. On distingue en effet la fragmentation horizontale, la fragmentation verticale et la fragmentation mixte ou hybride.

La classification des fragments

Après avoir déterminé les fragments par sites en utilisant les schémas conceptuels et logiques, on procède à leur classification. Cette classification vise à valoriser l'utilisation des fragments pendant une période de référence. La période de référence est celle dont le volume des opérations est le plus objectif au niveau des sites. Cela permet de regrouper les fragments ayant un taux d'utilisation très fort au sein d'un même site. Cela permet de faire une affectation provisoire des fragments par site. Après avoir attribué les fragments par sites, on établit le tableau correspondant aux droits de création et de mise à jour par site.

Ainsi, on attribue ces droits de façon cohérente pour éviter par exemple qu'un site ne crée une occurrence de données alors qu'un autre site peut la mettre à jour. Cela ne veut pas dire qu'un site ne peut pas soumettre une demande de mise à jour à un autre site. Dans ce cas, le site habilité effectue la mise à jour pour le compte du site demandeur. La classification permet d'attribuer les fragments aux sites, mais peut aussi conduire à l'attribution d'un fragment à un type de sites.

Attribution des liaisons aux types de sites

Lorsque nous décrivions de façon globale les types de distribution, nous disions qu'elle pouvait se faire avec des données centralisées ou avec des données réparties. En effet, outre les orientations organisationnelles et la structure fonctionnelle de l'application qui sont des facteurs déterminants, la disposition des liaisons est aussi un facteur déterminant dans le choix de distribution.

En effet deux cas se présentent *(figures 29a et 29b)* lorsque l'on veut faire les attributions de liaisons :

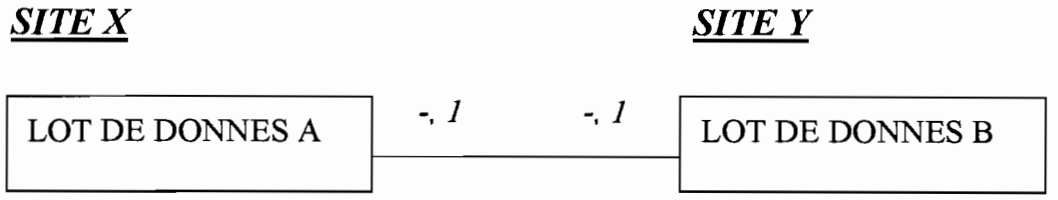


Figure 29a : Liaison conceptuelle dont les deux cardinalités sont égales à 1

Dans ce cas, on crée sur le site X, un lot B', formé uniquement de la clé de B.

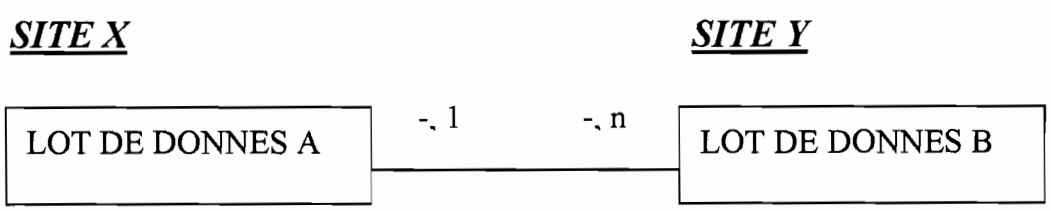


Figure 29b : Liaison conceptuelle dont une des cardinalités est égale à n

Ici, on crée sur le site X, un lot B', formé de la clé de B. Cela permet d'affecter la liaison conceptuelle sur le site X sans difficulté. Pour passer du site X au site Y, il suffit d'établir une correspondance des clés de B et B' de même valeurs.

Si on avait envisagé une duplication de la clé de A sur le site Y, on aurait noté que cela provoquerait une duplication plus volumineuse de clé.

Ce sont des contraintes d'efficacité semblables qui ont partiellement motivé le choix d'implémentation dans une base de données centralisée, pour le cas particulier du « Financier ».

Etablissement de la liste des types d'échanges

Il s'agit de recenser les requêtes et de les décomposer en sous-requêtes par sites. Une requête entraîne un double échange entre sites. Le premier est une demande, le deuxième est une réponse. Le tableau ci-dessous *(figure 30)* est illustré une répartition de fragments par site.

SITES / FRAGMENTS	S1	S2	.....	Sj	...	Sp
F1	$.a_{11}$	$.a_{12}$		$.a_{1j}$		$.a_{1p}$
...						
Fi	$.a_{i1}$	$.a_{i2}$		$.a_{ij}$		$.a_{ip}$
...						
Fn	$.a_{n1}$	$.a_{n2}$		$.a_{nj}$		$.a_{np}$

S_j : Site de rang j
F_i : Fragment de rang i
$.a_{ij}$: indique l'utilisation du fragment Fi au niveau du site Sj

Figure 30 : Tableau de classification

Pour faciliter la répartition des fragments et le choix d'éventuelle duplication, il est possible d'attribuer des valeurs qui ne sont que des qualificatifs liés à l'utilisation des fragments au niveau des sites : F (utilisation forte), M (moyenne), P (petite). Cela permet d'attribuer un fragment au site où il est le plus utilisé.

Pour ce qui concerne l'affectation des fragments aux sites, on peut utiliser un qualificatif lié au type de répartition : A (Attribué), D (Dupliqué).

Si deux sites ont le même degré d'utilisation, on attribut le fragment au site de création et de mise à jour. Cependant, si des créations et des mises à jour sont prévues au niveau de plusieurs sites, on choisit un site chargé de la création et de la mise à jour auquel les autres sites peuvent soumettre leurs demandes.

Si par contre des données sont très utilisées par plusieurs sites autres que le site d'attribution, on peut envisager leur duplication afin de minimiser le volume des échanges entre sites.

VI. IMPLEMENTATION DU PORTAGE

VI-1. La mise en œuvre du portage

VI-1-1. Le modèle conceptuel de données (MCD) futur

Ce modèle prend en compte les changements à opérer pour une meilleure identification des opérations issues des traitements concurrents. Ainsi des entités ou propriétés ont été ajoutées conformément à la description des *figures 31 et 32*.

La notation **An** signifie que le champ est de type *alphanumérique* et de longueur *n*.

Nom de la table	Champ ajouté	Type	Fonction associée
LigneEcriture	Suivi	A12	Référence toute ligne d'écriture comptable saisie ou modifiée par un utilisateur

Nom de la table	Champ	Type	Fonction associée
Utilisateur	IdUtil	A2	Indique l'identité de l'utilisateur
	NomUtil	A25	Indique le nom de l'utilisateur
	PnomUtil	A25	Indique le prénom de l'utilisateur

Nom de la table	Champ	Type	Fonction associée
Droits	IdDroits	A4	Indique le niveau de droits associé à un utilisateur
	LibDroits	A4	Libellé des droits associés à un utilisateur

Figure 31 : Tables et / ou propriétés modifiées ou ajoutées dans le cadre du portage

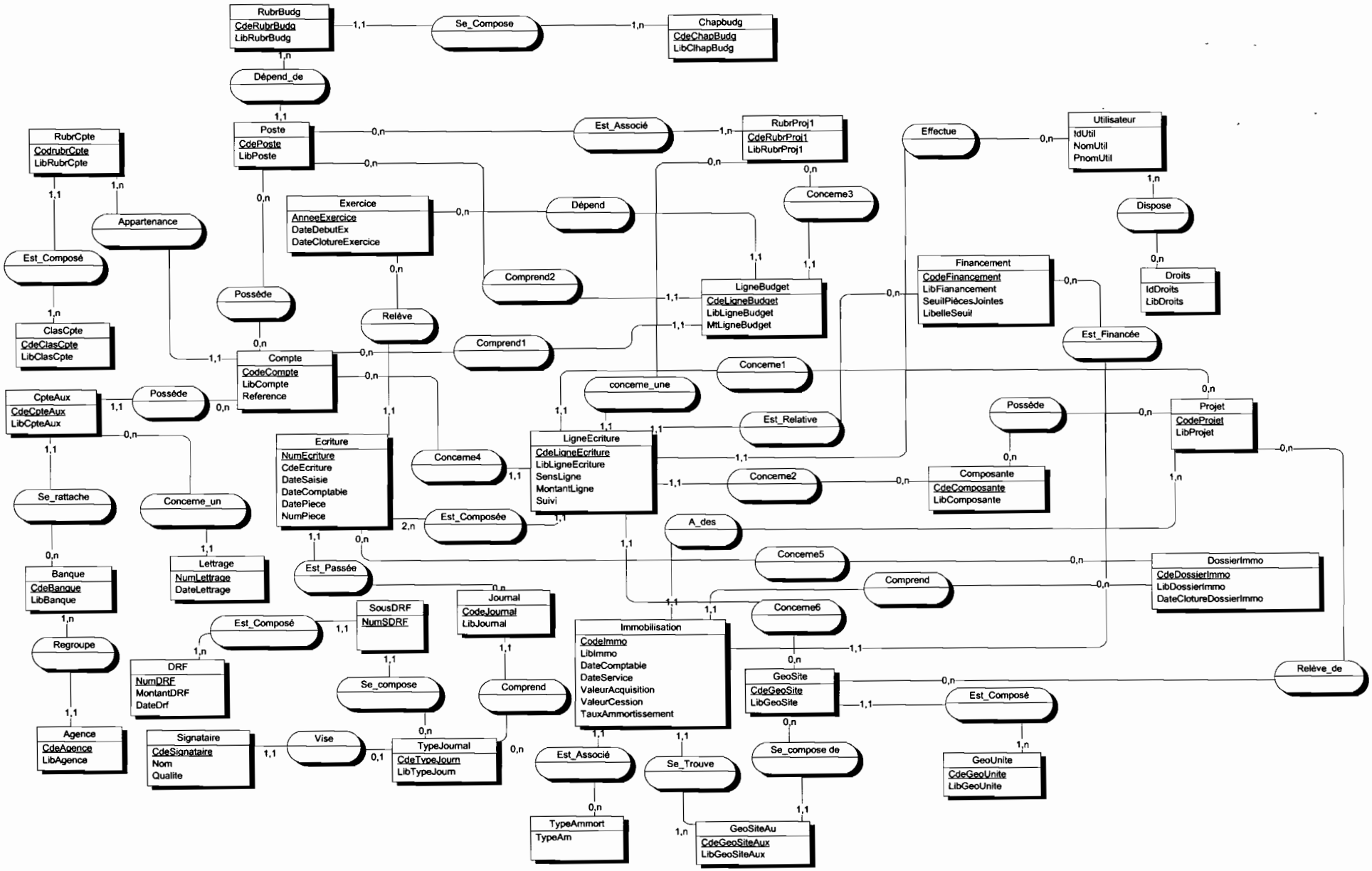


Figure 32 : Modèle conceptuel de données futur

VI-1-2. Matrice des droits d'accès au «Financier »

La matrice des droits d'accès permet de décrire les fonctionnalités multi-utilisatrices auxquelles chaque utilisateur a droit. Pour le cas du «Financier », chacun des utilisateurs a ses droits d'accès, donc un ensemble de traitements adéquats qui lui sont associés. Cette répartition fonctionnelle est présentée ci-dessous *(figure 33)*.

La légende suivante peut être proposée :

- *Interdit* signifie que l'utilisateur ne peut pas accéder à cette fonctionnalité ;
- *Consultation* signifie que l'utilisateur ne peut que consulter les informations. Il ne peut ni les créer, ni les modifier, ni les supprimer ;
- *Autorisé* signifie que l'utilisateur a tous les droits associés à cette fonctionnalité.

Utilisateurs / *Fonctionnalités*	**DAF**	**Comptable OG**	**Comptable Caisse**	**Comptable banque**	**Chef Comptable**
Codification	Interdit	Interdit	Interdit	Interdit	Autorisé
Comptabilité	Autorisé	Autorisé	Autorisé	Autorisé	Autorisé
Exercice	Autorisé	Consultation	Consultation	Consultation	Autorisé
Budget	Autorisé	Consultation	Consultation	Consultation	Autorisé
Convention	Autorisé	Consultation	Consultation	Consultation	Consultation
Immobilisation	Autorisé	Autorisé	Autorisé	Autorisé	Autorisé
Outils[3]	Autorisé	Autorisé	Autorisé	Autorisé	Autorisé

Figure 33 : Matrice des droits d'accès au « Financier»

[3] Il s'agit des outils de paramétrage de l'interface homme/machine

VI-1-3. Contrôle des transactions

Pour chaque opération de mise à jour dans la base de données, susceptible d'être réalisée par plusieurs utilisateurs à la fois, il faut délimiter la transaction correspondante. Cela se fait par l'utilisation des méthodes associées au composant *TDatabase*. Il faut néanmoins éviter de prendre des sous-ensembles trop vastes, car ils pourraient être à l'origine d'une baisse des performances.

Début contrôle de transaction

Pour chaque ressource critique faire

1. *Placer début transaction*
2. *Tester cohérence de la base de données*
   Si la cohérence est respectée alors
   Mettre à jour les données
   Sinon
   Refuser mise à jour
   Fin si
3. *Placer fin de transaction*

Fin faire

Fin contrôle de transaction

VI-1-4. Diagramme des composants de l'application

✓ **Programme principal**

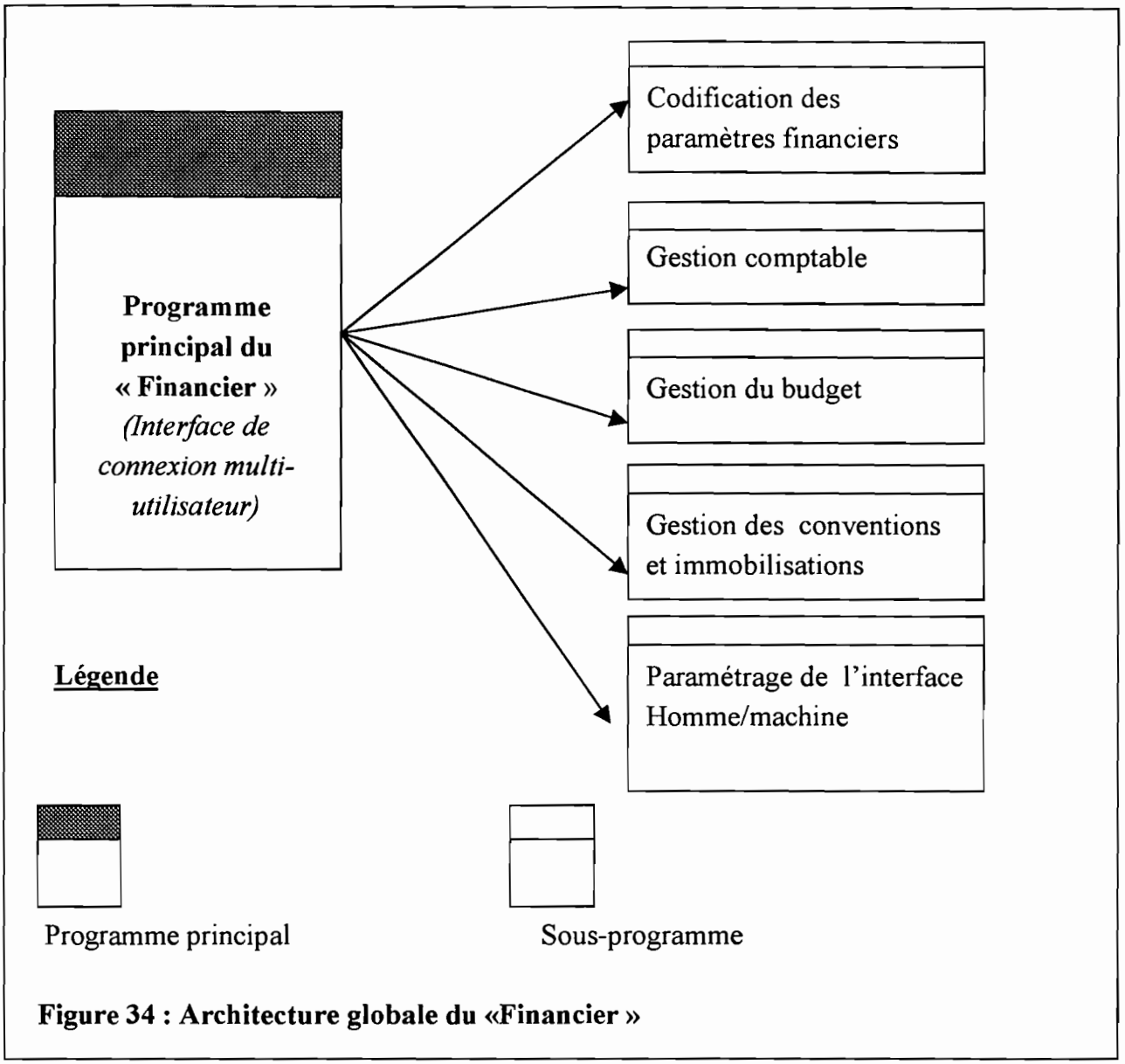
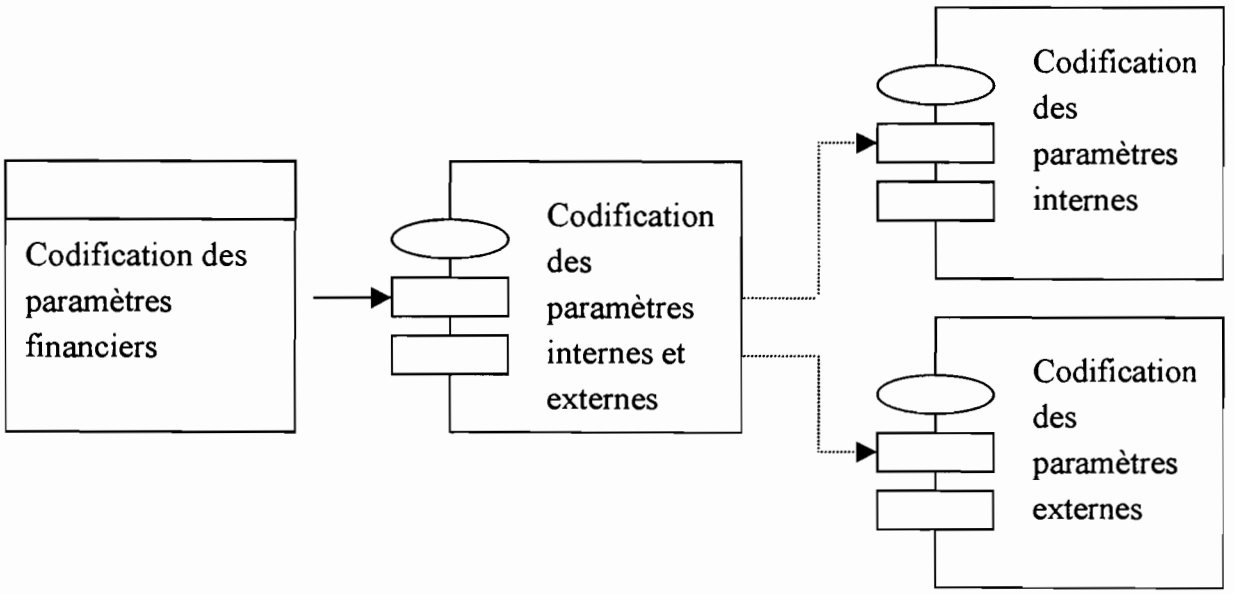


Figure 34 : Architecture globale du «Financier »

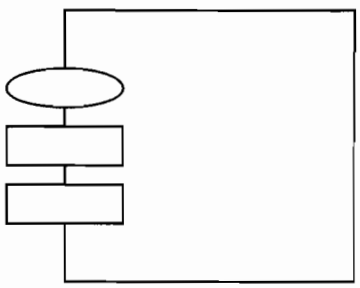
Le « Financier » *peut être découpé en cinq parties logiques dont une partie constitue le programme principal. Le programme principal comprend l'interface de connexion multi-utilisateur qui permet à l'utilisateur reconnu d'accéder aux fonctionnalités conformément à ses droits d'accès : codification des paramètres financiers, gestion comptable, gestion du budget, gestion des conventions et des immobilisations et le paramétrage de l'interface homme/machine.*

✓ **Sous-programme de codification des paramètres financiers**

Les paramètres financiers peuvent être décomposés en paramètres internes et externes. Les paramètres internes regroupent les codifications internes au CIRDES : comptes, composantes, rubriques, journaux, codes budgétaires, sites, projets, etc.
Quant aux paramètres externes, il s'agit par exemple des banques, des agences et des tiers.



Légende :



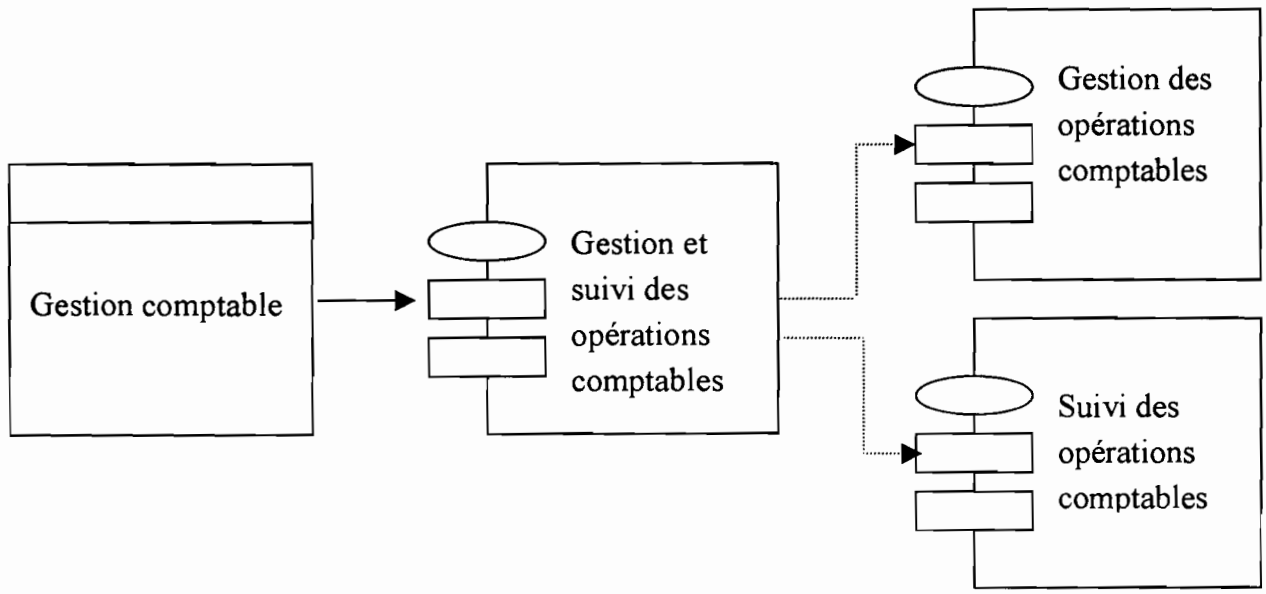
Représentation d'un composant

A ⋯⋯▶ B ***Le composant A dépend du composant B***

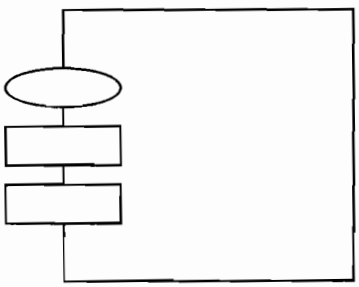
Figure 35 : Diagramme des composants de codification des paramètres financiers

✓ Sous-programme de gestion comptable

La gestion comptable comprend non seulement la saisie des opérations comptables, mais aussi leur suivi à travers les situations qui sont périodiquement éditées.
Cela permet aux décideurs de prendre les décisions y afférents mais surtout produire des informations capitales pour les partenaires.



Légende :



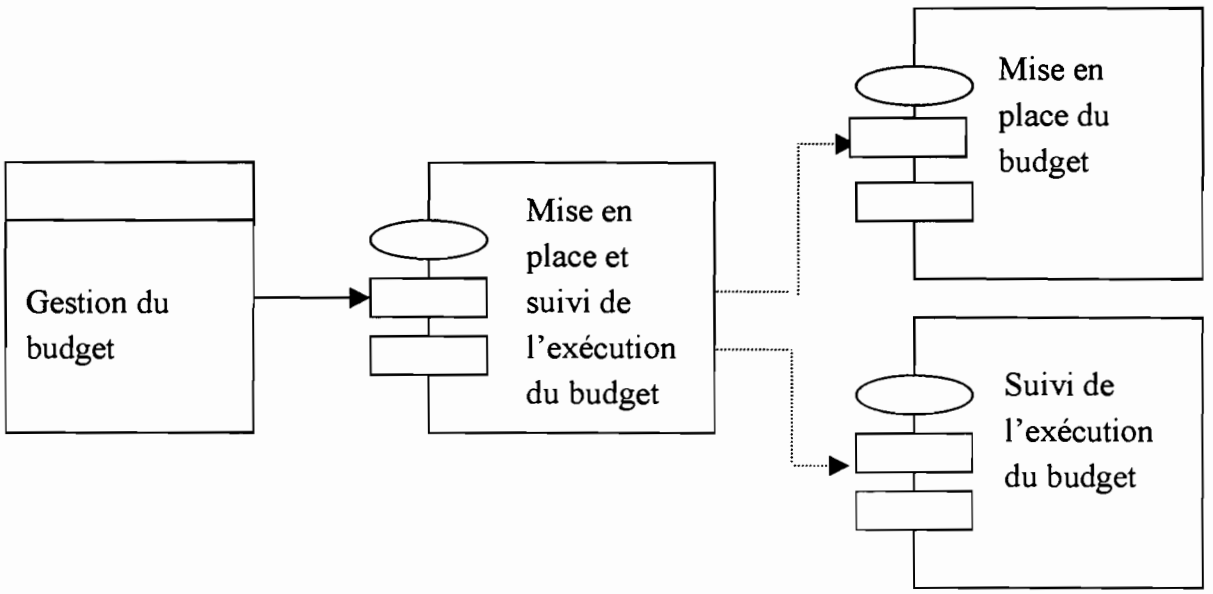
Représentation d'un composant

A ⋯⋯▶ B ***Le composant A dépend du composant B***

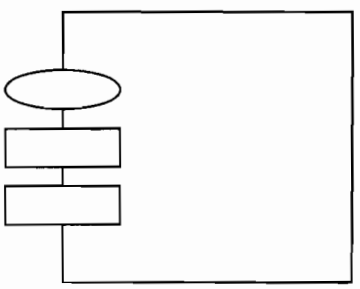
Figure 36 : Diagramme des composants de gestion comptable

✓ ***Sous-programme de gestion du budget***

Les deux phases essentielles dans la gestion du budget sont : la mise en place du budget et le suivi de l'exécution du budget. La mise en place du budget concerne la saisie des prévisions en matière de recettes et de dépenses, adoptées par l'organisme. Le suivi de l'exécution consiste à contrôler l'évolution des différentes situations financières produites afin de les aligner sur les objectifs définis par le budget adopté.



Légende :



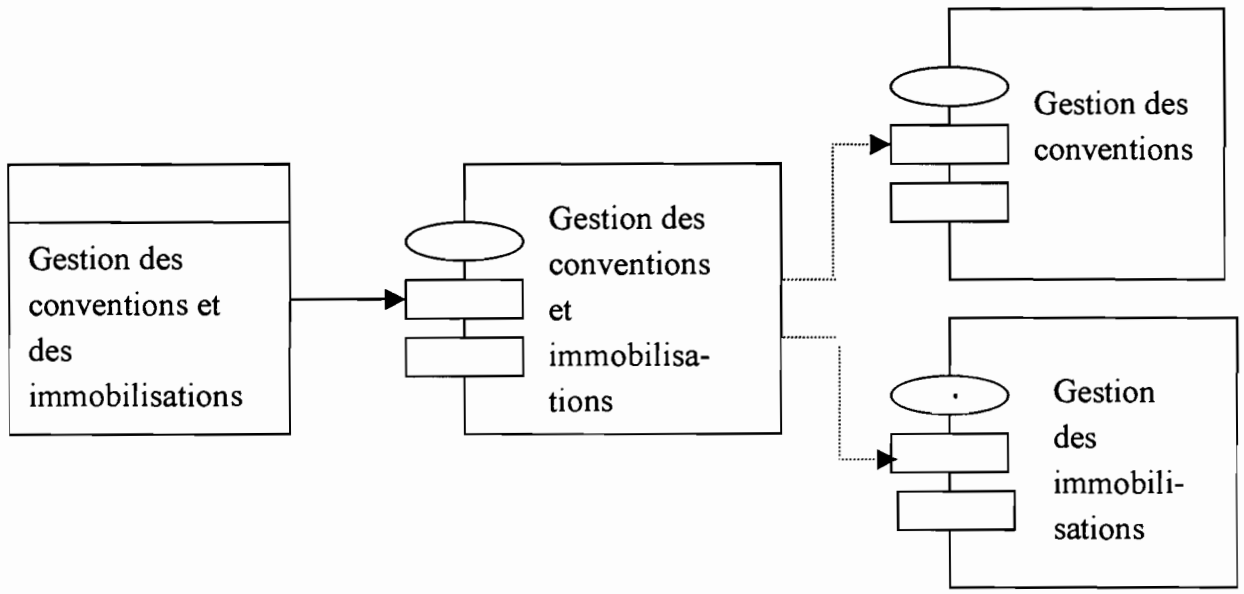
Représentation d'un composant

A ⋯▶ B ***Le composant A dépend du composant B***

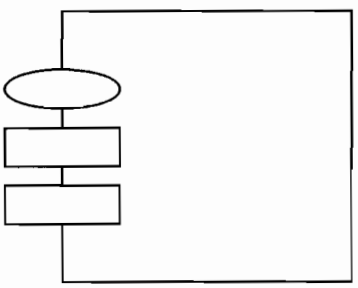
Figure 37 : Diagramme des composants de gestion du budget

✓ **Sous-programme de gestion des conventions de financement et des immobilisations**

La gestion des conventions et des immobilisations, intégrée dans les fonctionnalités du « Financier » permet de suivre les conventions de financement signées avec les partenaires. Quant aux immobilisations, le suivi automatique des différents taux choisis (constant ou régressif) permet d'automatiser les traitements.



Légende :



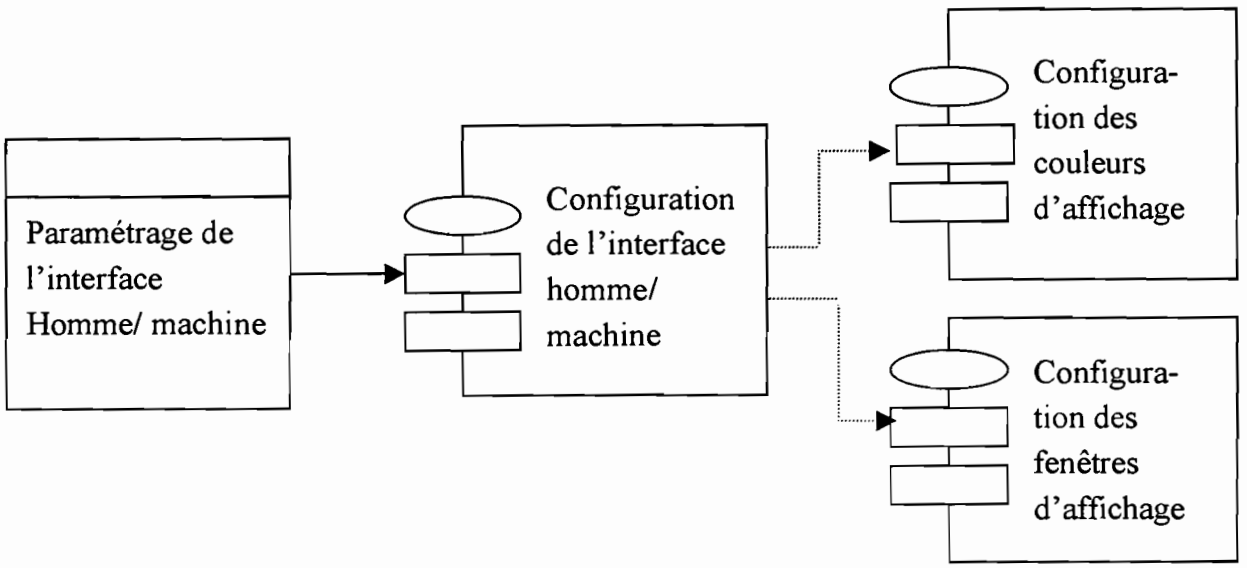
Représentation d'un composant

A ⟶ B ***Le composant A dépend du composant B***

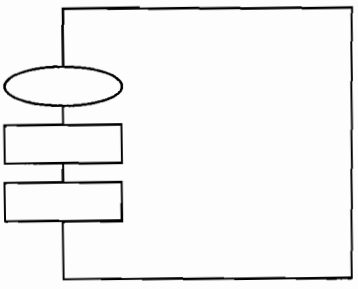
Figure 38 : Diagramme des composants de gestion des conventions et des immobilisations

✓ Sous-programme de paramétrage de l'interface Homme/machine (utilisateur)

Le paramétrage de l'interface homme/machine n'est pas indispensable au fonctionnement du « Financier » mais il est essentiel pour les utilisateurs. En effet, chaque utilisateur connecté, a la possibilité de choisir ses couleurs (de fond, police des caractères, fenêtres, etc) selon son goût. Cela permet au programmeur de ne pas avoir à modifier le code applicatif simplement pour satisfaire un utilisateur trop exigeant en matière de couleur.



Légende :



Représentation d'un composant

A ⟶ B ***Le composant A dépend du composant B***

Figure 39 : Diagramme des composants du paramétrage de l'interface utilisateur

VI-1-5. Quelques unités de traitement du «Financier 2.0 »

VI-1-5-1. Authentification des accès

Même si les transactions sont bien gérées, un manque d'authentification rigide pourrait être à l'origine d'opérations malveillantes sur les données de l'organisme. C'est pourquoi, au démarrage, une interface de connexion doit permettre de s'assurer de l'identité de la personne qui effectue l'accès.

Au démarrage du «Financier », une interface de connexion permet à chaque utilisateur d'entrer son compte utilisateur et son mot de passe. Si ces informations sont valides, le menu des fonctionnalités de l'application lui est présenté avec une configuration adaptée à son profil. Ainsi, conformément à la matrice des droits d'accès que nous avons présentée, le DAF par exemple n'a pas le même menu que le chef comptable.

Figure 40 : Interface introductive avant l'affichage de l'interface de connexion

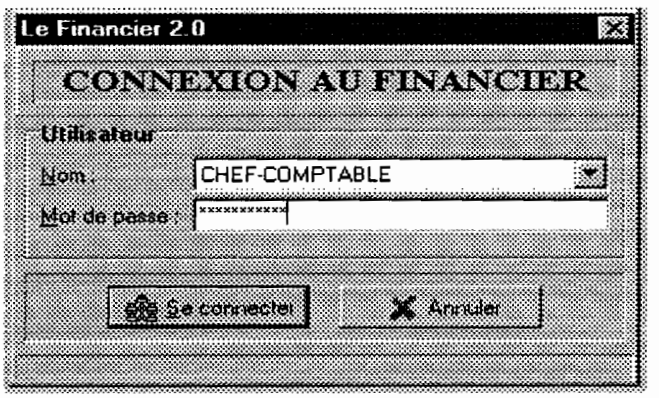


Figure 41 : Interface de connexion multi-utilisateur

Ainsi, on peut décrire l'authentification des accès comme suit :

Authentifier l'accès
Lire le compte et le mot de passe de l'utilisateur (Compte, MotDePasse) Vérifier l'existence du compte (Compte_Utilisateur, Compte) Vérifier la validité du mot de passe (MotDePasse_Compte, MotDePasse) Traiter (Compte, MotDePasse)

Figure 42 : sous-système décrivant les traitements associés à l'authentification des accès

Début Authentifier l'accès

1. *Etablir une connexion avec la base de données*
2. *Présenter l'interface utilisateur pour la connexion*
3. *Lire compte et mot de passe saisis*
4. *Vérifier la validité du compte et du mot de passe*

```
Si Non _Valide (compte) alors
    Refuser la connexion
Sinon
    Vérifier (MotDePasse)
    Si Non_Valide (MotDePasse) alors
            Refuser la connexion
    Sinon
            Accepter la connexion
    FinSi
FinSi
```

5. *Restaurer les paramètres par défaut du menu*
6. *Fermer la connexion à la base de données*

Fin Authentifier l'accès

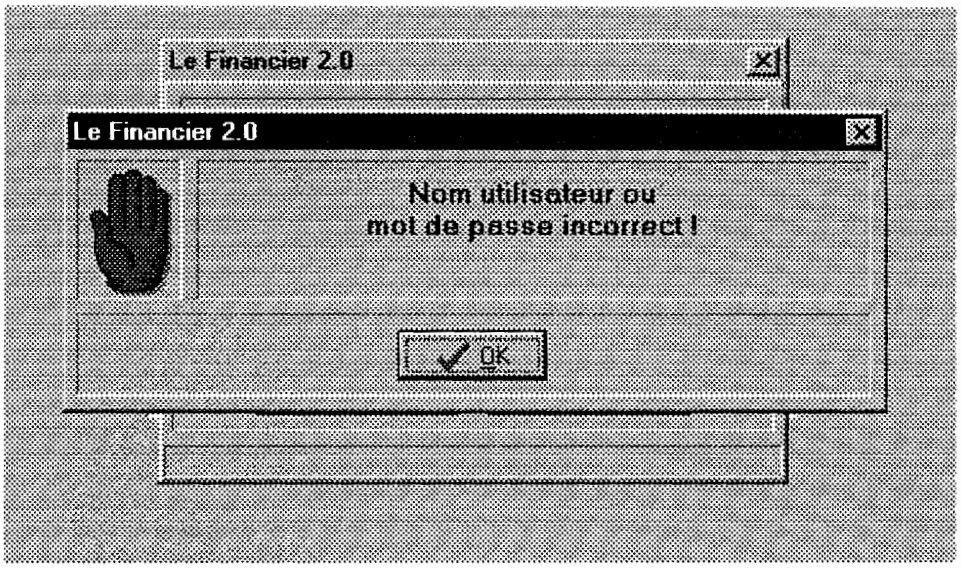


Figure 43 : Exemple de connexion refusée suite à des informations de connexion incorrectes

VI-1-5-2. Configuration de l'application par rapport à l'utilisateur

Pour éviter que les utilisateurs aient accès aux mêmes fonctionnalités, on associe à chaque utilisateur, un niveau de droit donné qui permet d'adapter le menu à son profil. Ainsi, on considère que toutes les fonctionnalités sont inhibées au départ. En fonction des droits octroyés à l'utilisateur, les fonctionnalités sont activées. En fin de compte, on a un menu correspondant au profil de l'utilisateur.

Configurer l'application
Lire le compte et le mot de passe de l'utilisateur (Compte, MotDePasse) Lire les droits associés à l'utilisateur (Compte, Paramètres) TransmettreParamètres (Menu, Paramètres) TraiterConfiguration (Menu, Paramètres)

Figure 44 : sous-système décrivant la configuration de l'application selon les utilisateurs

Début Configurer l'application

1. *Se connecter à l'application*
2. *Inhiber toutes les fonctionnalités*
3. *Lire droits associés (Droits)*
   Pour chaque Élément(Droits) faire
      Transmettre Paramètre (Paramètres, Menu)
      Activer_Élément_Menu (Paramètres)
   Fin Faire
4. *Activer_Menu (Droits)*

Fin Configurer l'application

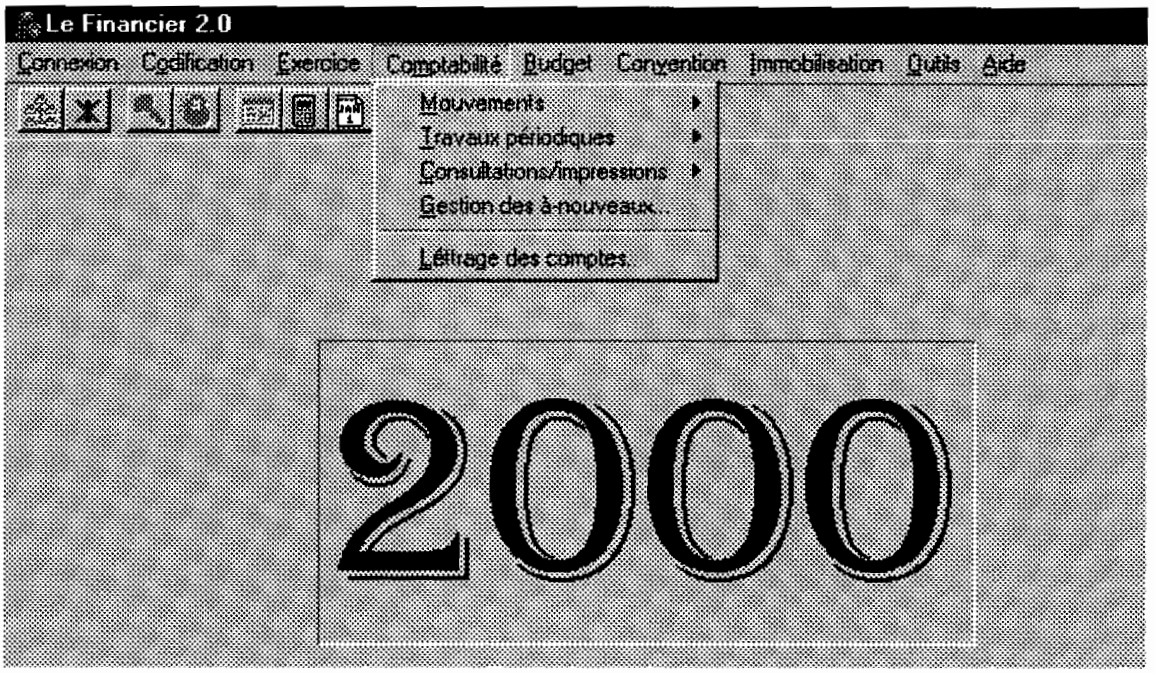


Figure 45 : Connexion avec comme identité « chef comptable »

On remarque à travers ce menu (correspondant) que les fonctionnalités accordées au chef comptable ne sont pas les mêmes que celles accordées au DAF *(figure 46)*.

Etant simplement un exemple pour illustrer l'aspect changeant de l'interface utilisateur, une application multi-utilisateur doit offrir les fonctionnalités adaptées à chaque utilisateur selon son profil et ses attributions dans l'organisme concerné.

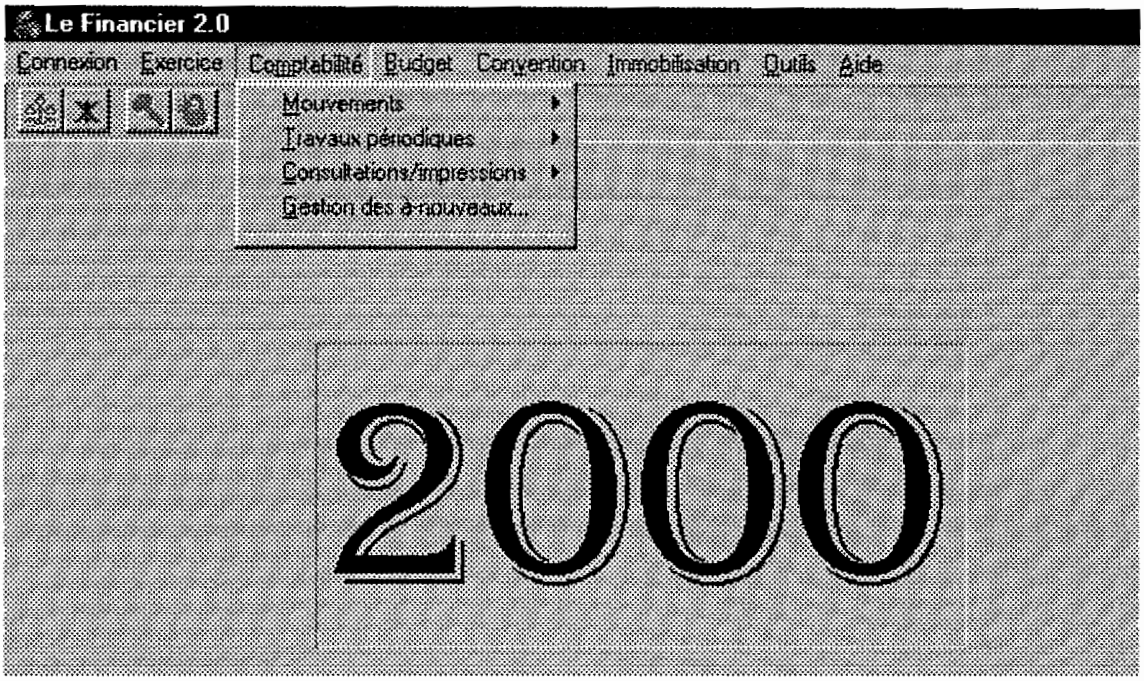


Figure 46 : Connexion avec comme identité « DAF »

On remarque que ce menu (correspondant) aux fonctionnalités accordées au DAF ne présente pas de possibilités de codifications de paramètres comme celles du chef comptable.

VI-1-5-3. Mise en place et suivi du budget

Mise en place du budget
Créer un exercice budgétaire (Exercice) Mettre à jour les codes de lignes budgétaires (Exercice, CodeBudget) Mettre à jour les montants affectés (LignesBudget)

Figure 47 : Sous-système décrivant la mise en place du budget

Début Mise en place du budget

1. *Créer nouveau exercice budgétaire (exercice, CodeBudget)*
2. *Pour chaque ligne budgétaire (LignesBudget) faire*
   Etablir prévisions (LignesBudget)
   Etablir les financements associés (LignesBudget, Financement)
   Mettre à jour (LignesBudget)
   Fin pour

Fin Mise en place du budget

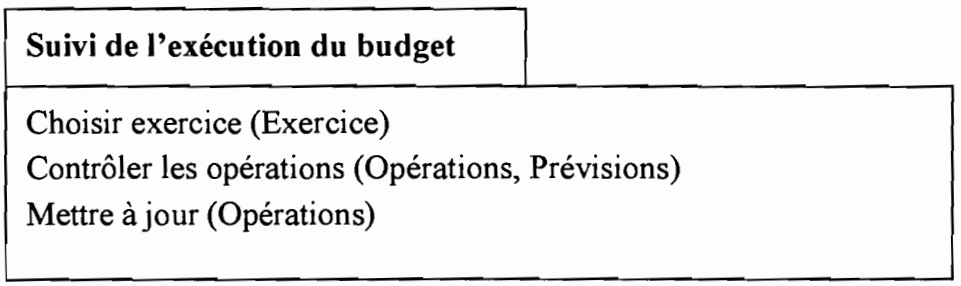


Figure 48 : sous-système décrivant le suivi de l'exécution du budget

Début Suivi de l'exécution du budget

1. *Choisir un exercice budgétaire (Exercice)*
2. *Pour chaque opération financière (Opération) faire*
   Si nature = « recette » alors
   Accepter l'opération (LignesBudget, Montant(Opération))
   Valider l'opération (LignesBudget, Montant(Opération))
   Sinon
   { nature = « dépense » }
   Vérifier disponibilité (LignesBudget)
   Si disponibilité (LignesBudget) < Montant (Opération) alors
   Refuser l'opération (Opération)
   Sinon
   Accepter l'opération (LignesBudget, Montant(Opération))
   Valider l'opération (LignesBudget, Montant(Opération))
   Fin si
   Fin si
   Fin pour

Fin Suivi de l'exécution du budget

VI-1-5-4. Gestion et suivi de la comptabilité

Gestion de la comptabilité
Choisir journal (Journal) Choisir compte (Comptes) Imputer opération (Débit, Crédit) Valider opération (LignesEcriture)

Figure 49 : sous-système décrivant la gestion de la comptabilité

Début gestion de la comptabilité

Pour chaque opération comptable faire

1. *Choisir journal*
2. *Créer écriture comptable (Ecriture)*
   Créer Lignes d'écriture (Débit)
   Créer Lignes d'écriture (Crédit)
3. *Si Total (Débit) = Total (Crédit) alors*
   Accepter (Opération)
   Sinon
   Refuser (Opération)
   Fin si

Fin pour

Fin gestion de la comptabilité

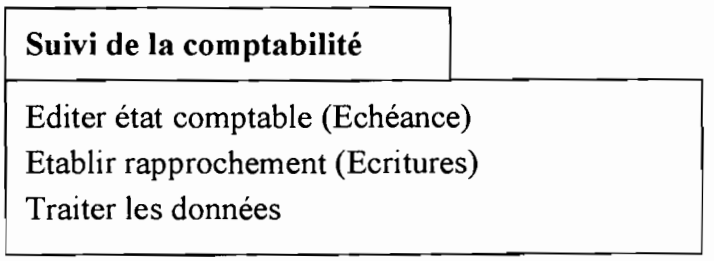


Figure 50 : sous-système décrivant le suivi de la comptabilité

Début suivi de la comptabilité

Pour chaque échéance faire

1. *Editer (Echéance, état comptable)*
2. *Contrôler (Opérations)*

Si Conforme (Opérations) alors

Valider (Opérations)

Sinon

Corriger (Opérations)

Fin si

Fin Pour

Fin suivi de la comptabilité

VI-1-6. La sécurité des opérations sur les données

Pour renforcer la sécurité des opérations, il est important de suivre les opérations qui ont été effectuées sur la base de données.

Une solution pourrait être de conserver un fichier contenant les traces de connexions des utilisateurs.

✓ Fichier des traces de connexion

Un fichier des traces de connexion est un fichier créé pour conserver les informations rattachées aux connexions des utilisateurs.

- o l'identité de l'utilisateur qui a effectué la connexion,
- o l'instant auquel la connexion a eu lieu,
- o et l'instant où la déconnexion est survenue.

Cette solution n'est pourtant pas efficace car elle ne permet pas de savoir les opérations effectuées sur les données.

En effet, le simple fait qu'un utilisateur se soit connecté à un moment donné ne suffit pas pour qu'on puisse le tenir responsable de certaines opérations douteuses. Dans la mesure où un utilisateur peut être connecté sans pour autant effectuer des opérations de mise à jour, sa période de connexion pourrait être utilisée à tort pour repérer des tâches malveillantes.

La *figure 51* décrit la configuration d'un fichier des traces de connexions.

Identité	**Nom**	**Connecté à**	**Déconnecté à**
PZ	PARE Azara	20/11/2000 : 14 : 55	20/11/2000 : 16 : 12
KS	KABORE Issa	20/11/2000 : 15 : 28	20/11/2000 : 17 : 05
OA	OUATTARA Abdou	20/11/2000 : 15 : 48	20/11/2000 : 17 : 00

Figure 51 : configuration d'un fichier des traces de connexion

✓ Suivi des données critiques

Pour que les informations soient fiables, il est important que les opérations sensibles comme les lignes d'écriture comptables soient suivies de manière spécifique. Pour cela, à chaque tuple correspondant à une information critique, on adjoint une information qui décrit l'identité de l'auteur de l'opération.

La mise en œuvre du portage consistera donc à ajouter dans les instructions de mise à jour, l'insertion d'un champ *Suivi* que nous avions défini dans le modèle conceptuel de données futur. La configuration réelle de ce champ est laissée à l'initiative du programmeur.

Voici un format général :

IdUtil [+ DateEnCours][+ HeureEnCours][+MinuteEnCours][+SecondeEnCours][+Rq]

- **IdUtil** permet d'indiquer l'utilisateur auteur de l'opération ;
- **DateEnCours** comprend le jour, le mois et l'année de réalisation de l'opération ;
- **HeureEncours** indique l'heure à laquelle l'opération a été effectuée sur la donnée ;
- **MinuteEncours** indique la minute à laquelle l'opération a été réalisée ;
- **SecondeEnCours** indique la seconde a laquelle l'opération a été réalisée ;
- **Rq** permet d'indiquer le type d'opération qui a été réalisée (modification, création, etc.).

Afin de ne pas conserver des informations qui peuvent grossir rapidement la base de données, nous choisissons *IdUtil+DateEnCours+ HeureEnCours+MinuteEnCours*
En nombre de caractères : XX+dd+mm+[aa]aa+hh+mm soit 14 caractères au minimum.

Cela permet d'éviter un encombrement inutile tout en fournissant des éléments de contrôle fiables pour la sécurité des données comptables. Cette solution n'exclue pourtant pas la conservation d'un fichier gardant la trace des connexions.

NumLigneEcriture	LibLigneEcriture	SensLigne	MontantLigne	Suivi
2000-95124	Charge personnel	Débit	120.500 F	ZA : 17112000:10 :45
2000-95125	Virement banque	Crédit	110.000 F	ZA : 17112000:10 :50
2000-95126	Divers précomptes	Crédit	5.500 F	ZA : 17112000:10 :55
2000-95127	Précompte IUTS	Crédit	5.000 F	ZA : 17112000:10 :58

Figure 52 : Illustration du suivi des lignes d'écriture comptable

Un problème subsiste néanmoins, puisque ce suivi ne concerne que les données ajoutées ou modifiées. Comment suivre alors les données qui ont fait l'objet de suppression ? Deux possibilités peuvent être exploitées :

- insérer une balise sur le tuple « supprimé » sans permettre une suppression physique des données. Ce tuple demeure dans la base de données mais n'est plus comptabilisé dans les calculs ;
- créer un fichier où l'on inscrit les tuples supprimés avant leur suppression physique de la base de données. On indique en même temps dans ce fichier, l'identité de l'utilisateur qui a été à l'origine de l'opération.

Ces différents outils de contrôle que l'on met en place ne doivent pas être visibles par tous les utilisateurs.

VI-2. Le déploiement du «Financier »

VI-2-1. Déploiement basé sur le moteur de base de données Borland

Les applications Delphi simples, constituées d'un seul fichier exécutable, s'installent facilement sur un ordinateur cible. Il suffit de copier le fichier sur l'ordinateur. Mais les applications plus complexes comme le «Financier », composées de plusieurs fichiers exigent une procédure d'installation plus sophistiquée. De telles applications nécessitent un programme d'installation spécifique.

Les boîtes à outils d'installation automatisent le processus de création d'un programme d'installation, le plus souvent sans avoir besoin d'écrire une seule ligne de code. Les programmes d'installation créés par les boîtes à outils d'installation effectuent diverses tâches liées à l'installation des applications Delphi, y compris la copie de l'exécutable et des fichiers nécessaires sur l'ordinateur cible, la création des entrées de registre Windows et l'installation du moteur de bases de données Borland pour les applications de base de données.

InstallShield Express est une boîte à outils d'installation fournie avec Delphi. Cette boîte à outils est spécialement adaptée à l'utilisation de Delphi et du moteur de bases de données Borland.

VI-2-2. Déploiement basé sur l'architecture multi-niveaux

Pour un déploiement basé sur une architecture multi-niveaux, certains fichiers sont à installer sur les postes client alors que d'autres sont destinés au serveur. Les fichiers décrits concernent les fichiers systèmes. La base de données étant implicitement installée sur le serveur.

✓ **Liste des fichiers à installer au niveau du poste client**

NBASE.IDL	ODEN40.DLL	RPMFEN40.DLL	OLENTEXP.EXE
ODECTN40.DLL	RPMARN40.DLL	RPMUTN40.DLL	OLECFG.EXE
ODEDIN40.DLL	RPMAWN40.DLL	OLERAN40.DLL	SETLOG.EXE
ODEEGN40.DLL	RPMCBN40.DLL	OLEAAN40.DLL	W32PTHD.DLL
ODELTN40.DLL	RPMCPN40.DLL	OLEWAN40.CAB	
ODEMSG.DLL	RPMEGN40.DLL	OBJX.EXE	

✓ **Liste des fichiers à installer au niveau du serveur**

UNINSTALL.EXE	OBJFACT.ICO	W32PTHD.DLL	NBASE.IDL
LICENSE.TXT	ODEBKN40.DLL	RPMARN40.DLL	OBJX.EXE
README.TXT	ODECTN40.DLL	RPMAWN40.DLL	OLECFG.EXE
OLENTER.HLP	RPMEGN40.DLL	RPMCBN40.DLL	OLEWAN40.CAB
OLENTER.CNT	ODEDIN40.DLL	RPMCPN40.DLL	OLENTEXP.EXE
FILELIST.TXT	ODEEGN40.DLL	BROKER.EXE	OLENTEXP.HLP

SETLOG.TXT	ODELTN40.DLL	RPMFEN40.DLL	OLENTEXP.CNT
SETLOG.EXE	LIBAVEMI.DLL	RPMUTN40.DLL	BRKCP.EXE
OBJPING.EXE	OLEAAN40.DLL	RPMFE.CAT	BROKER.ICO
OBJFACT.EXE	OLERAN40.DLL	EXPERR.CAT	

VI-3. La critique de la démarche

VI-3-1. Bilan de la réalisation

Pour mettre en œuvre le portage vers un environnement multi-utilisateur et distribué, nous avons été confrontés dans un premier temps à une corruption partielle du code source. Afin de ne pas reprendre la conception et la réalisation, ce qui n'était pas l'objectif, nous avons adopté une démarche permettant de restaurer le code applicatif.

En effet, la réalisation de l'application existante a déjà mobilisé des ressources importantes. Le temps déjà consacré à la réalisation s'évalue à plus de (6) mois, sans compter qu'au moins deux personnes constituaient le groupe de projet.

Si nous ne parvenions pas à restaurer le code source, ce serait une perte des fruits de tous ces efforts qui ont été mobilisés. Ainsi, le fait d'avoir pu restaurer le code source de l'application constitue une économie de temps. Cette restauration nous a aussi permis d'optimiser le code applicatif en éliminant les procédures inutiles.

Avec le temps de réalisation insuffisant qui nous a été accordé, nous avons pu néanmoins :

- mettre à jour les différents modèles conceptuels de données ;
- achever la correction des codifications ;
- faire des tests de fonctionnement dans un environnement multi-utilisateur à travers un réseau ;
- achever la maintenance de certaines fonctionnalités relatives aux immobilisations et conventions ;
- implémenter en partie le suivi des opérations critiques.

Quant à la réalisation au niveau des opérations comptables, il est nécessaire que les utilisateurs soient disponibles afin que l'on puisse identifier des jeux de tests éprouvés à employer pour déceler les dysfonctionnements.

VI-3-2. Avantages de la démarche du portage

La démarche qui a consisté à retrouver le schéma conceptuel de données à partir du schéma physique de la base de données a permis de réaliser un modèle conceptuel de données à jour par rapport à l'application existante. En outre, elle nous a permis de retrouver des tables inutiles qui ne faisaient que grossir la taille de la base de données. Leur retrait a donc permis d'optimiser la taille de la base de données.

Le modèle conceptuel de données ainsi que les composants d'accès aux données qui ont été proposés permettent, d'effectuer un choix indépendant d'une implantation particulière de base de données ce qui assure une grande indépendance par rapport aux technologies existantes.

VI-3-3. Limites de la démarche

La démarche de reverse engineering, basée sur la réalisation du modèle conceptuel à partir du schéma physique de la base de données et du code applicatif occasionne des temps de traitements importants.

AMC*Designor Données permet de retrouver automatiquement le modèle conceptuel de données à partir soit de scripts de création de la base de données soit du schéma d'une base de données évoluée. Il n'y avait donc pas de possibilité de réaliser automatiquement le modèle conceptuel de données à partir des tables Paradox dont nous disposions.

Cela a eu pour conséquence, l'élaboration à l'inverse du modèle conceptuel de données à partir des tables Paradox.

CONCLUSION

A l'issu des six (6) mois consacrés à l'étude et à l'implémentation d'une solution, par rapport au portage d'une application mono-utilisateur et monoposte vers un environnement multi-utilisateur et distribué, nous avons présenté les concepts de monoposte, mono-utilisateur, multi-utilisateur, et distribué.

Pour notre cas où le «Financier » reste la préoccupation majeure, nous avons été confrontés à une dégradation de certains fichiers sources de l'application. Cela nous a amené à mettre en œuvre une démarche de réhabilitation de ce code source sans lequel aucune tentative de portage ne saurait être envisagée. En passant par des démarches progressives d'identification et de réparation des fichiers, nous sommes parvenu à remettre l'application dans son état avant l'altération des fichiers sources.

Pour la mise en œuvre du portage, une démarche exploitant en partie la retro-conception de code a été exploitée pour reconstruire le modèle conceptuel actuel à partir du modèle physique de l'application. En prenant en compte l'environnement multi-utilisateur, une proposition de modèle conceptuel de données futur a été faite avec pour objectif la sécurité et l'optimisation de la taille de la base de données.

Ainsi, l'architecture proposée, basée sur une distribution des traitements avec une base de données unique, dont la gestion des transactions revient au composant *TDatabase* a fait l'objet d'une étude comparée. Des modules fonctionnels ainsi que des algorithmes associés aux unités de traitements ont été proposés pour mettre en exergue l'aspect multi-utilisateur du «Financier ».

Enfin, la solution proposée vise prioritairement une résolution des besoins opérationnels du Centre International de Recherche-Développement sur l'Elevage en zone Subhumide (CIRDES) tout en proposant un environnement d'exploitation satisfaisant.

GLOSSAIRE

MOT	DESCRIPTION
Atomicité	Une transaction doit effectuer toutes ses mises à jour ou ne rien faire du tout. En cas d'échec, une transaction doit annuler toutes les modifications qu'elle a engagées.
Base de données	Collection de données structurées indépendamment d'une application particulière, cohérente de redondances minimales et accessible par plusieurs utilisateurs.
Base de données réparties	Ensemble de bases de données gérées par des sites différents et apparaissant à l'utilisateur comme une base unique.
Budget	Le budget est une prévision de recettes et de dépenses pour une année civile.
Cohérence	La transaction doit faire passer la base de données d'un état cohérent à un autre état cohérent. En cas d'échec de la transaction, l'état cohérent initial doit être restauré.
Compte	Les comptes proposés par le Plan Comptable Général sont répartis en neuf classes numérotées de 1 à 9 : - classe 1 : comptes de capitaux, - classe 2 : comptes d'immobilisation, - classe 3 : comptes de stock et d'en cours, - classe 4 : comptes de tiers, - classe 5 : comptes financiers, - classe 6 : comptes de charges, - classe 7 : comptes de produits, - classe 8 : comptes spéciaux, - classe 9 : comptes analytiques d'exploitation.
DAF	Directeur Administratif et Financier. Il contrôle et supervise l'exécution des opérations comptables et budgétaires. Ainsi, il est chargé de viser tous les documents financiers officiels.
DLL	Une DLL (Dynamic Link Library) ou bibliothèque liée dynamiquement est un module exécutable contenant du code ou des ressources destinées à d'autres applications ou à d'autres DLL. Les DLL permettent de partager du code et des ressources entre plusieurs applications.

Dépense	Une dépense est une charge constatée à l'endroit d'un organisme.
Durabilité	Dès qu'une transaction valide ses modifications, le système doit garantir que ces modifications seront conservées en cas de panne.
Ecriture comptable	Une écriture comptable est une opération qui met en jeu plusieurs comptes dont la partie débit doit avoir un total égal à la partie crédit.
Fragment	Sous-table obtenue par sélection de lignes ou de colonnes à partir d'une table globale, localisée sur un site unique.
Fragmentation horizontale	Découpage d'une table en sous-tables par utilisation de prédicats permettant de sélectionner les lignes appartenant à chaque fragment.
Fragmentation verticale	Découpage d'une table en sous-tables par projections permettant de sélectionner les colonnes composant chaque fragment.
Immobilisation	C'est un actif de l'entreprise qui peut subir au cours du temps des pertes de valeur qui résultent de leur usage, de changements technologiques (obsolescence) ou de toutes autres causes.
Incohérence	L'incohérence se produit lorsque des données liées par une contrainte d'intégrité sont mises à jour par deux transactions dans des ordres différents de sorte à violer la contrainte. Exemple : {T1 : A=A+1 ; T2 : B=B*2 ; T2 : A=A*2 ; T1 : B=B+1 } Si au départ A=2 et B=2 avec la contraintes d'avoir toujours A=B, à la fin on aura : A= 6 et B=5.
Intégration de schémas	Ensemble des aspects d'un système informatique perceptible par un utilisateur et que celui-ci utilise pour communiquer avec le système.
Interface homme/machine	Procédé qui consiste à rapprocher des schémas sémantiquement différents afin de constituer un schéma unique équivalent.
Isolation	Les résultats d'une transaction ne doivent être visibles aux autres transactions qu'une fois la transaction validée, ceci afin d'éviter les interférences avec les autres transactions.
Journal	Le journal ou livre journal est un document imposé par la loi où sont enregistrées, dans l'ordre chronologique, les différentes opérations de l'entreprise décrites dans les pièces justificatives.

ODBC ODBC (Open DataBase Conectivity) est un middleware qui permet l'accès à des bases de données. Il exploite le verrouillage deux phases.

Perte de mise à jour Une perte de mise à jour survient lorsqu'une transaction T1 exécute une mise à jour calculée à partir d'une valeur périmée de données, c'est à dire une valeur modifiée par une autre transaction T2 depuis la lecture par la transaction T1. La mise à jour de T2 est donc écrasée par celle de T1.

RAID Le réseau redondant de disques indépendants est une matrice de disques dans laquelle une partie de la capacité physique est utilisée pour y stocker de l'information redondante concernant les données d'utilisateurs. Cette information redondante permet la régénération des données d'utilisateurs perdues lorsqu'une unité ou un chemin de données à l'intérieur d'une matrice est défaillant.

Recette Une recette est une ressource constatée au bénéfice d'un organisme.

Réplication Technique de répartition consistant à gérer des copies sur différents sites, les copies pouvant être différentes à un instant donné, mais devant converger vers une même valeur si on arrête la production de transactions de mises à jour.

Retro-conception de code Cette technique vise à garantir que le code et le modèle de conception soient synchronisés à tout instant. Par conséquent, la retro-conception doit être incrémentale pour permettre la mise à jour d'un modèle existant.

Schéma global Schéma constitué sur un site client par intégration globale des schémas importés décrivant la base de données répartie vue depuis ce site.

Schéma local Schéma décrivant les données d'une base de données locale gérée par un système de gestion de base de données local.

SQL SQL (Structured Query Language) est un langage de requêtes structuré permettant de réaliser des opérations sur des données d'une base de données.

Verrouillage deux phases Technique de contrôle des accès concurrents consistant à verrouiller les objets au fur et à mesure des accès par une transaction et à relâcher les verrous seulement après l'obtention de tous les verrous.

BIBLIOGRAPHIE

✓ *DOCUMENTATION*

1. [BORLAND 1997] Borland, **Delphi 3 - Guide de l'utilisateur**, Borland, 1997.

2. [BORLAND 1997] Borland, **Delphi 3 - Guide du développeur**, Borland 1997.

3. [BORLAND 1997] Borland, **Delphi 3 - Guide Pascal Objet**, Borland 1997.

4. [CASTELLANI 1987] Xavier CASTELLANI, **Méthode générale d'analyse des applications informatiques**, Edition MASSON, 1987.

5. [COLASSE 1991] Bernard COLASSE, **Comptabilité générale**, ECONOMICA, 1991.

6. [CORNAFION 1981] CORNAFION, **Systèmes Informatiques Répartis – Concepts et Techniques**, Edition DUNOD, 1981.

7. [DATE 1985] C. G. DATE, **An Introduction to Database Systems**, Addison-Wesley Publishing Company, 1985.

8. [GALACSI 1986] GALACSI, **Conception de base de données**, Edition Dunod, 1986.

9. [GARDARIN & VALDURIEZ 1989] G. GARDARIN, P. VALDURIEZ, **SGBD relationnels**, Edition Eyrolles, 1989..

10. [GARDARIN 1994] Georges GARDARIN, **Bases de données. Les systèmes et leurs langages**, Edition Eyrolles, 1994.

11. [GARDARIN 1996] Georges GARDARIN, Olivier GARDARIN, **Le Client-Serveur**, Edition Eyrolles, 1996.

12. [KABORE & ZEMBA 1997] S. KABORE, B. ZEMBA, **Cahier de charges utilisateurs**, ESI, 1997.

13. [KETTANI & MIGNET& PARE & ROSENTHAL-SABROUX 1996] N. KETTANI, D. MIGNET, P. PARE, C. ROSENTHAL-SABROUX, **De Merise à UML**, Edition Eyrolles, 1996.

14. [LEFEBRE 1994] Alain LEFEBRE, **L'Architecture Client-Serveur**, Edition Armand Colin, 1994.

15. [MATHERON 1992] Jean Patrick MATHERON, **Comprendre Merise**, Edition Eyrolles, 1992.

16. [READ & CHACE & ROWE 2000] T. READ., C. CHACE, S. ROWE **The Internet Start-up Bible**, Random House, 2000.

17. [SOUTOU 1999] Christian SOUTOU, **Objet relationnel sous Oracle 8 Modélisation avec UML**, Edition Eyrolles, 1999.

18. [THERON 1988] Paul THERON, **Guide pratique du génie logiciel**, Edition Eyrolles, 1988.

19. [ULLMAN 1982] J. D. ULLMAN, **Principles of Database Systems**, Computer Science Press, 1981.

✓ *RESSOURCES INTERNET*

a. http://www.inf.ethz.ch/departement/IS/iks/
M. Wisemann, F. Pedone, A. Chiper
Operating Systems Laboratory, Swiss Federal Institute of Technology (EPFL).

b. http://lsewww.epfl.ch/Documents/html/
B. Kemme, G. Alonso, Institute of Information Systems
Swiss Federal Institute of Technology (ETHZ).

Y sont décrits, les protocoles de réplication, des études comparatives de réplication de bases de données ainsi que l'étude de la tolérance aux erreurs dans le cas des bases de données réparties. On y trouve des algorithmes de gestion adaptés aux systèmes distribués.

c. http://www.ellipse.ch/cours/
Ce site présente les outils de programmation de Delphi. Des exemples de programmes écrits avec Delphi ainsi que des orientations relatives à la création d'applications y sont présentés.

d. http://www.info.univ-angers.fr/pub/fm/
Ce site offre une présentation de Francette MONET sur les bases de données relationnelles et surtout les techniques de gestion de bases de données relationnelles implémentées dans Oracle 7. On y trouve une présentation de SQL*Plus ainsi que l'emploi du langage SQL.
On y trouve aussi une présentation de PL/SQL (Langage du SGBD Oracle) et de Oracle 8, appuyée par des études de cas concrets.

e. http://www.univ-pau.fr/~bruel/CNAM/
On trouve au niveau de ce site, un exposé de Jean-Michel BRUEL sur les architectures client/serveur ainsi que les évolutions constatées depuis les années 1970.

f. http://www.oracle.com
Ce site décrit en détail les caractéristiques des produits Oracle ainsi que les outils de création, de configuration et d'administration d'un serveur de base de données Oracle.

g. http://www.liv.ac.ut/middleware/htm/
Ce site décrit l'architecture des médiateurs standards. Ainsi, on y trouve une description détaillée de ODBC (Open DataBase Connectivity).

h. http://www.cs.purdue.edu/people/helal/
Donne une présentation (en anglais) des techniques de réplication de base de données dans les systèmes distribués. Cette présentation est inspirée du livre **Replication Techniques in Distributed Systems** de Abdelsalam A. HEDAL de l'université de Boston, de Abdelsalam A. Helal (MCC) et de Bharat B. Bhargava de l'université de Purdue.

i. http://www.microsoft.com/sql/
Ce site de Microsoft donne des informations générales sur SQL Server.

j. http://www.afpalim.tm.fr/VB2/MODULE/
Ce site présente les outils de SQL Server pour la gestion des bases de données. Des études de cas sur la création et la gestion de base de données y sont présentées de façon détaillée.

k. http://cic.sop.cstb.fr/ILC/people/va/probleme/
Ce site présente un exposé sur les architectures distribuées, l'interopérabilité entre les systèmes hétérogènes. Il décrit l'évolution du client/serveur et donne une présentation succincte de COM/DCOM/ActiveX ainsi que CORBA.

ERRATA

Lors de la finalisation du document, certaines erreurs s'y sont glissées :

1. Page 8 : I. DEFINITIONS DES CONCEPTS

Deuxième paragraphe : lire supportés par des moyens informatiques (*et téléinformatiques)* au lieu de téléinformatique.

2. Page 45 : veuillez ignorer les observations manuelles faites sur la page extraite du cahier des charges utilisateurs (1997)