Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Système d'Information et de Connaissances (S.I.C)*

Thème

Etude de sécurité en base de données avec une application pour le contrôle d'accès.

Réalisé par :

- *EL HADJ MIMOUNE Khadidja*
- *MERABET Meriem*

Présenté le 29 Septembre 2011 devant le jury composé de MM.

Président : -Mr Abderrahim M.A.
Encadreur : - Mr Badr Banmammar.
Examinateur: -Mlle benmansour F. -Mme Halfaoui A.
-Mme Yebedri Z. -Mr Benamar A .
-Mr Chouiti S.

Année universitaire: 2010-2011

Table de matière

Chapitre III: Modélisation et conception

Chapitre IV: Implémentation

Liste des figures

Liste des acronymes

Le monde de l'informatique est parsemé d'une multitude d'acronymes en tout genre. Voila une liste non exhaustive de ceux rencontrés les plus fréquemment dans la littérature informatique, et plus particulièrement qui sont utiliser dans nos mémoire.

Terme	***Commentaire***
API (Application Programming Interface)	Une API est une bibliothèque qui regroupe des fonctions sous forme de classes pouvant être utilisées pour développer.
DAC (Discretionnary Access Control)	Moyens de limiter l'accès aux objets basés sur l'identité des personnes ou des groupes auxquels ils appartiennent
HTML (HyperText Markup Language)	Langage à base de balises pour formater une page web affichée dans un navigateur.
ITSEC (Information Technology Security Evaluation Criteria)	Critères d'évaluation de la sécurité des systèmes d'information, définis par un comité de pays européens.
J2EE (Java 2 Entreprise Edition)	C'est une version du JDK qui contient la version standard plus un ensemble de plusieurs API permettant le développement d'applications destinées aux entreprises.
JAR (Java ARchive)	Technique qui permet d'archiver avec ou sans compression des classes java et des ressources dans un fichier unique de façon indépendante de toute plate−forme. Ce format supporte aussi la signature électronique.
JDBC (Java Data Base Connectivity)	C'est une API qui permet un accès à des bases de données tout en restant indépendante de celles ci. Un driver spécifique à la base utilisée permet d'assurer cette indépendance car le code Java reste le même.
JDK (Java Development Kit)	C'est l'environnement de développement Java. Il existe plusieurs versions majeures : 1.0, 1.1, 1.2 (aussi appelée Java 2) et 1.3. Tous les outils fournis sont à utiliser avec une ligne de commandes.
JSP (Java Server Page)	C'est une technologie comparable aux ASP de Microsoft mais utilisant Java. C'est une page HTML enrichie de tag JSP et de code Java. Une JSP est traduite en servlet pour être exécutée. Ceci permet de séparer la logique de présentation et la logique de traitement contenu dans un composant serveur tel que des servlets, des EJB ou des beans.
LMD (Langage de Manipulation des Données)	Est un langage qui permet de manipuler les données qui sont stocké se forme des tableaux.
MAC (Mandatiry Access Control)	Méthode de gestion des droits des utilisateurs pour l'usage de systèmes d'information.
ODBC (Open Database Connectivity)	Format permettant la communication entre des clients BDD fonctionnant sous windows et les SGBD.
RBAC (Role-Base Access Contol)	Modèle de contrôle d'accès à Système d'Information dans lequel chaque décision d'accès est basée sur le rôle auquel l'utilisateur est attaché.
SGBD (Système de Gestion de Base de Données)	Ensemble de programmes constituant la gestion et l'accès à plusieurs bases de données.
URL (Uniform Resource Locator)	Chaine de caractères utilisée pour adresser les Ressources dans le World Wide Web : document HTML, image, son, etc. informellement appelée une adresse Web.
XML (Extensible Markup Language)	Langage informatique de balisage générique.

Introduction générale

1. Motivations :

La sécurité des applications informatique et des bases de données en particulier est devenue une priorité pour les citoyens ainsi que pour les administrations. Le besoin de partager et d'analyser des données personnelles est multiple : pour rendre plus simples et efficaces les procédures administratives et pour personnaliser les services rendus par une grande quantité d'objets électroniques dans un environnement d'intelligence ambiante.

Dans le cadre de ce PFE, on s'intéresse à la sécurité dans le cadre des bases de données. Notre objectif est de proposer des mécanismes de contrôles d'accès afin d'améliorer la sécurité d'une application informatique.

2. Contribution :

Dans le cadre de ce PFE, nous avons réalisé une application qui fait appel à des mécanismes des sécurités dans les bases de données.

Cette application contient deux parties, la première est une application Web réalisée pour gérer les clients dans un environnement 3-tiers, nous avons proposé dans cette partie un contrôle d'accès sur le nombre de tentatives erronés d'authentification avant de désactiver un compte existant ainsi qu'un mécanisme de chiffrement afin de protéger les informations pertinentes de la base de données. La deuxième partie de notre application est une implémentation 2-tiers afin de gérer les fonctionnalités de l'Agent et de l'Administrateur, à ce stade nous avons proposé un contrôle par rôle, basé sure le modèle par rôle, nous avons également fait l'audit des opérations réaliser par le client.

3. L'organisation de notre travail : Ce travail est structuré comme suit :

Le premier chapitre est consacré à présenter des bases de données, nous avons détaillé vers les SGBD les opérations sur les données et les différents types de base de données.Le deuxième chapitre présent la sécurité dans les bases de données, ainsi que nous avant traité **les moyens de sécurité** selon les différents modèles de contrôle d'accès, et la protection d'une base de données.Le troisième chapitre présente la modélisation de notre application qui intègre des contrôles d'accès que nous avons proposés. La modélisation est basée sur le langage UML (Unified Modelisation Language). Notre application est basée sur deux types d'architecture : 2-tiers pour implémenter les rôles d'Agent et Administrateur et 3-tiers pour gérer le client avec pour chacune des propositions pour la gestion de la sécurité.

Le quatrième chapitre présente l'implémentation de notre application et la démarche de chaque fonctionnalité réalisée.

CHAPITREI:

Introduction aux bases de données

1. Introduction

L'informatique évolue vers le traitement de masses d'informations de plus en plus grandes dans des environnements répartis géographiquement où doivent cohabiter des matériels hétérogènes. Dans ce contexte, les bases de données sont utilisées de façon intensive pour de nombreux domaines d'application tels que le domaine médical, les administrations ou les associations. Les applications concernées par l'utilisation d'un SGBD (Système de gestion de base de données) possèdent des caractéristiques différentes tant au niveau du volume de données concernées qu'au niveau de la complexité de ces données et des traitements informatiques à réaliser. Néanmoins, le regroupement des données dans une base de données gérée par un système de gestion de base de données apporte de nombreux avantages dans la plupart des cas d'utilisation.

Le domaine informatique bien qu'étant jeune, a une évolution croisière. Jadis, la gestion et le traitement des données se faisaient par la méthode classique à laquelle l'on a pu dégager ces défauts suivants:

- La redondance de données ;
- La dépendance pleine entre données et traitement ;
- Le manque de normalisation au niveau de stockage de données.

Pour remédier à cette situation, il a été mis au point la notion de base de données répondant aux questions suivantes:

- L'accès aux données selon les multiples critères ;
- L'intégration des données ;
- La relation entre les données.

La notion qui remplace avantageusement celle de fichiers.

- L'ordre dans le stockage de données ;
- L'utilisation simultanée des données par différents utilisateurs.

2. Qu'est-ce qu'une base de données ?

Le concept de Base de Données (BDD) est apparu vers 1960, face au nombre croissant d'informations que les entreprises devaient gérer et partager :

- Base de données - Un ensemble organisé d'informations avec un objectif commun. Plus précisément, on appelle base de données un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche et consultions de données).

- Base de données informatisée - Une base de données informatisée est un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur, représentant des informations du monde réel et pouvant être interrogées et mises à jour par une communauté d'utilisateurs.

La gestion et l'accès à une base de données sont assurés par un ensemble de programme que constitue le système de gestion de base de données (SGBD).

Ainsi la notion de base de données est généralement couplée à celle des réseaux informatiques afin de pouvoir mettre en commun les informations d'où le nom de « base ». On parle souvent de système d'information pour désigner toute structure regroupant les moyens mis en place pour partager les données. **[1]**

3. Critères d'une base de données

Une base de données doit répondre aux trois critères suivants :

- L'exhaustivité : C'est la présence dans cette base de tous les enseignements qui ont trait aux applications en question.
- Le non redondance des données : Non répétition d'une donnée plusieurs fois.
- La structure : C'est l'adaptation du mode de stockage de données au traitement ; structuration que la base doit avoir est liée à l'évolution de la technologie. **[1]**

4. Système de Gestion de Base de données : SGBD

Ensemble des programmes et des langages de commande qui permettent de :

- Définir des "bases de données", et des relations entre les éléments de chaque base ;
- Spécifier le traitement de ces données : interrogations, mises à jour, calculs, extractions...

Le SGBD reçoit des commandes aussi bien des programmes d'application que des utilisateurs : il commande les manipulations de données, généralement par l'intermédiaire d'un SGF. **[3]**

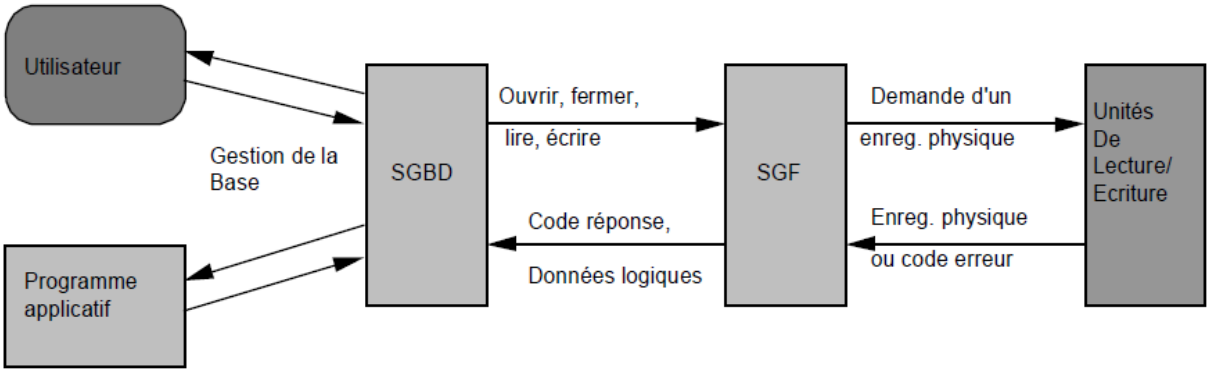


Figure I .1: Système de Gestion de Base de données « SGBD ».

Les définitions sont cependant de peu d'intérêt pour déterminer si un système est vraiment un SGBD ou s'il s'agit simplement d'un système d'information classique ou d'un système de fichiers. Il faut mieux définir le SGBD en précisant certaines des fonctions qu'il doit remplir :

- L'intégration des données afin d'éviter l'incohérence d'éventuelles données dupliquées (tout est intégré dans un seul ensemble cohérent).
- La séparation entre les moyens de stockage physique des données et la logique des applications.
- Un contrôle unique de toutes les données afin de permettre l'utilisation simultanée par plusieurs utilisateurs.
- La possibilité d'utiliser des structures de fichiers et des méthodes d'accès complexes, de façon à ce que les relations correctes entre les données puissent être exprimées et les données utilisées le plus efficacement dans un grand nombre d'applications.
- Des facilités pour le stockage, la modification, la réorganisation, l'analyse

et la consultation des données, sans que le système impose des restrictions à l'utilisateur.

- Des contrôles de sécurité afin d'empêcher l'accès illégal à certaines données.
- Des contrôles d'intégrité pour prévenir une modification indue des données (exemple: contrôle d'exactitude, de validité).
- La compatibilité avec les principaux langages de programmation, les programmes-sources existants, et les données extérieures à la base.
- Posséder une capacité de stockage élevée.
- Pouvoir répondre à des requêtes avec un niveau de performances adapté.
- Fournir des facilités pour la gestion des méta-données. **[4]**

4.1 Objectifs de l'approche SGBD

• Pour pallier aux inconvénients des méthodes classiques de gestion de fichiers, les SGBD visent quatre objectifs : intégration et corrélation, flexibilité (indépendance), disponibilité, sécurité.

• Ces objectifs exigent une distinction nette entre les données et les procédures de manipulation de ces données : aux données, on associera une fonction **d'administration des données**, aux procédures de manipulation une **fonction de programmation**.

4.1.1 Intégration et corrélation

Dans les systèmes classiques, chaque application gère ses données dans ses propres "fichiers", d'où :

❖ Un risque de redondance, et un danger d'incohérence des données.

- La même donnée peut appartenir à plusieurs applications, induisant une déperdition de stockage.
- Toute modification de cette donnée est à enregistrer plusieurs fois : si cette mise à jour multiple n'est pas effectuée correctement, les données deviennent incohérentes.
- Le coût de la mise à jour augmente du fait de la multiplication des entrées-sorties physiques.

❖ Une difficulté pour créer de nouveaux traitements

- Les nouvelles applications entraînent des duplications supplémentaires de données.
- Leur intégration avec les applicatifs en exploitation entraîne des modifications importantes. Dans l'approche SGBD, un "réservoir" commun (**intégration**) est constitué, représentant une modélisation (**corrélation**) aussi fidèle que possible de l'organisation réelle de l'entreprise :
  - ❖ Toutes les applications puisent dans ce réservoir, les données qui les concernent, évitant ainsi les duplications.
  - ❖ Mais le partage des données entre les utilisateurs pose le problème de la synchronisation des accès concurrents.

4.1.2 Flexibilité ou indépendance

- Dans les systèmes classiques, tout changement intervenant dans le stockage des données (support, méthode d'accès physique) entraîne des modifications lourdes des applications correspondantes.
- L'approche SGBD poursuit trois objectifs, pour assurer l'indépendance des données par rapport aux traitements :
  - ❖ Indépendance physique: tout changement de support, de méthode d'accès reste transparent au niveau de l'utilisateur.
  - ❖ Indépendance logique : les programmes d'application sont rendus transparents à une modification dans l'organisation logique globale, par la définition de sous-schémas couvrant les besoins spécifiques en données.
  - ❖ Indépendance vis-à-vis des stratégies d'accès : l'utilisateur n'a plus à prendre en charge l'écriture des procédures d'accès aux données. Il n'a donc pas à intégrer les modifications tendant à optimiser les chemins d'accès (ex: création d'index).

4.1.3 Disponibilité

- ❖ Le choix d'une approche SGBD ne doit pas se traduire par des temps de traitement plus longs que ceux des systèmes antérieurs.
- ❖ L'utilisateur doit ignorer l'existence d'utilisateurs concurrents.
- ❖ L'aspect "performance" est donc crucial dans la mise en œuvre d'une base de données. Un tel objectif ne peut être atteint que si la conception

d'une base de données est menée de façon rigoureuse avec un découpage fonctionnel adéquat. Les règles et contraintes inhérentes sont évoquées lors de l'apprentissage d'une méthodologie d'analyse (exemple MERISE).

4.1.4 Sécurité

La sécurité des données recouvre deux aspects :

- **L'intégrité**, ou protection contre l'accès invalide (erreurs ou pannes), et contre l'incohérence des données vis-à-vis des contraintes de l'entreprise.
- **La confidentialité**, ou protection contre l'accès non autorisé ou la modification illégale des données.

Pour ne pas trop affecter les performances, la sécurité doit également être prise en compte dès la phase de conception. **[3]**

4.2 Architecture de SGBD

4.2.1 Architecture Client-Serveur

Depuis les années 80, les SGBD sont basés sur une **architecture clients-serveur.**

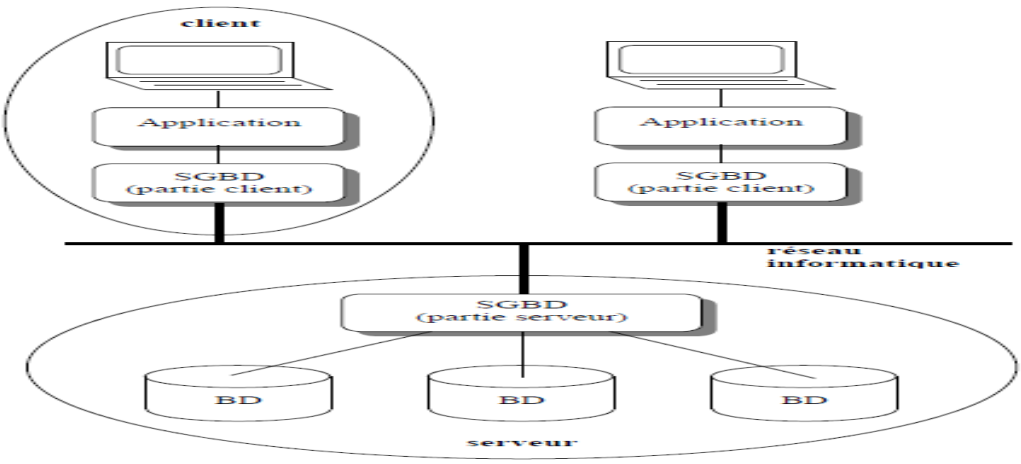


Figure I .2: Architecture Client-serveur. [2]

Serveur : On appelle logiciel serveur un programme qui **offre un service** sur le réseau. Le serveur accepte des requêtes, les traite et renvoie le résultat au demandeur. Le terme serveur s'applique à la machine sur lequel s'exécute le logiciel serveur. (Gère les données partagées et exécute le code du SGBD).

Clients : On appelle logiciel client un programme qui **utilise le service** offert par un serveur. Le client Communiquent avec le serveur envoie une requête et reçoit la réponse. Le client peut-être raccordé par une liaison temporaire.

Qu'appelle-t-on architecture client/serveur ?

C'est la description du **fonctionnement coopératif** entre le serveur et le client. Les services internet sont conçus selon cette architecture. Ainsi, chaque application est composée de logiciel serveur et logiciel client. A un logiciel serveur, peut correspondre plusieurs logiciels clients développés dans différents environnements: Unix, Mac, PC...; la seule obligation est le respect du protocole entre les deux processus communicants. Ce protocole étant décrit dans un RFC (Request For Comment). **[9]**

4.2.2 Architecture centralisée

Ce type d'architecture est appelée solution sur site central **(***Mainframe*). Historiquement, les applications sur site central ont été les premières à proposer un accès multiutilisateurs. Dans ce contexte, les utilisateurs se connectent aux applications exécutées par le serveur central à l'aide des terminaux se comportant en esclaves. C'est le serveur central qui prend en charge l'intégralité des traitements y compris l'affichage qui est simplement déporté sur des terminaux. **[1]**

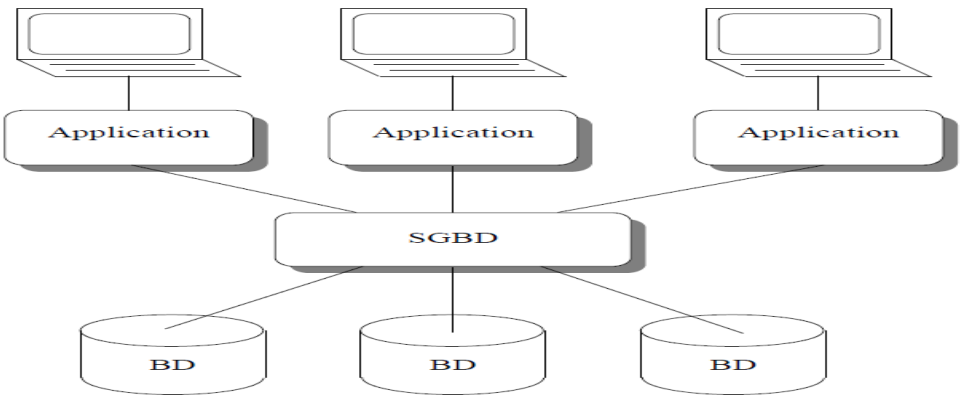


Figure I .3: Architecture centralisée. [2]

4.3 Architecture des systèmes

4.3.1 Architecture trischématique ou architecture ANSI/SPARC

Dans l'architecture trischématique on établit quatre niveaux de description du système de base de données qui sont : le *niveau interne*, le *niveau conceptuel*, *le niveau externe et le niveau physique*.

Le processus de transformation des requêtes et des résultats qui sortent d'un niveau à un autre s'appelle correspondance ou mapping.

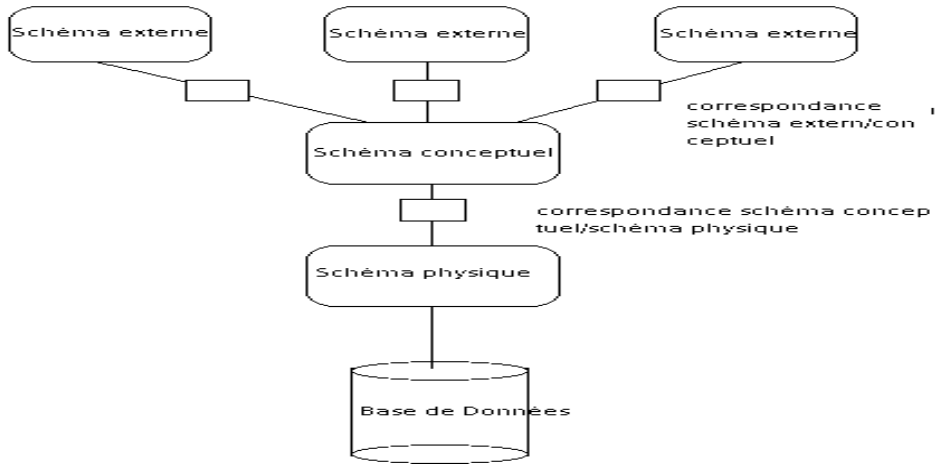


Figure I .4 : Architecture fonctionnelle d'un SGBD : ANSI-SPARC

Le niveau externe :

Le niveau externe (appelé aussi niveau vue), comprend une quantité de vues utilisateurs ; chaque utilisateur décrit une partie de la base qui convient à ses besoins. Chaque groupe d'utilisateurs s'intéresse uniquement à son propre schéma externe et les SGBD doivent transformer toute demande d'utilisateur de haut niveau en requêtes de schéma conceptuel puis en requêtes de schéma interne appliquées aux données stockées.

Le niveau conceptuel :

Dans le niveau conceptuel on décrit la structure générale de la base de données du point de vu de la communauté des utilisateurs ; c'est un schéma conceptuel qui masque les détails des structures de stockage physique des données et qui ne se soucie pas de l'implémentation physique des données ni de la façon dont chaque groupe d'utilisateurs voudra se servir de la base de données ; ce niveau se concentre sur la description des entités, du type des données, des relations existant entre les entités et des opérations des utilisateurs.

Le niveau interne :

Le niveau interne est un schéma qui décrit la structure de stockage physique de la base de données. Il s'appuie sur un système de gestion de fichiers pour définir la politique de stockage ainsi que le placement des données.

Le niveau physique :

Est donc responsable du choix de l'organisation physique des fichiers ainsi que de l'utilisation de méthodes d'accès en fonction de la requête.

4.3.2 Indépendances des données

Indépendance logique.

L'architecture définit ci-dessus permet de garantir l'indépendance des données par rapport aux programmes ; elle permet de modifier le schéma de la base de données à un niveau sans restructurer les autres. L'indépendance logique des données est la possibilité qui fait qu'on puisse modifier le niveau conceptuel sans remettre en cause les schémas externes ou les programmes d'application. L'ajout ou le retrait de nouveaux objets ne modifient pas les éléments qui n'y font pas explicitement référence.

Indépendance physique

Quand on peut changer le schéma physique et qu'on peut modifier l'organisation physique des fichiers, rajouter ou supprimer des méthodes d'accès sans remettre en cause le schéma conceptuel, alors on a une indépendance physique de la BD.

Le but de ce niveau d'indépendance est de rendre transparente la gestion physique des données aux programmes d'application. **[5]**

Cette architecture logique permet donc d'identifier la logique de structuration d'un système de gestion de bases de données. D'un point de vue fonctionnel, il est possible d'identifier plusieurs composants que l'on retrouve dans la plupart des SGBD. La figure **IV.5** illustre cette composition.

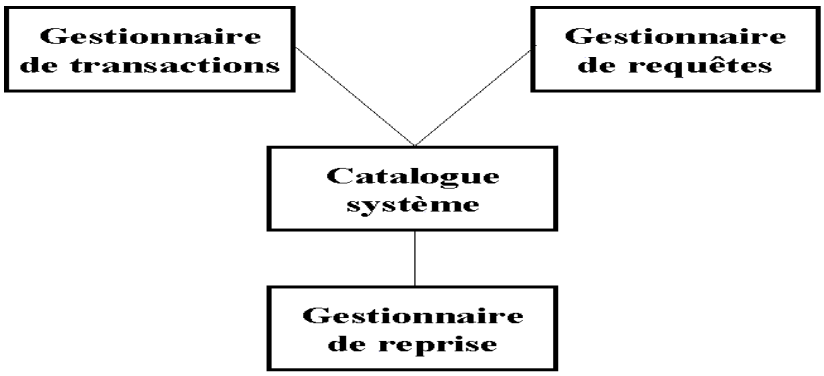


Figure I .5 : Composants d'un SGBD.

4.3.3 Le catalogue système ou dictionnaire

De données est un composant au cœur de la communication entre les autres composants. Il contient toutes les méta-données utiles au système. Ces méta-données correspondent à la description des données (type, taille, valeurs autorisées, etc.), aux autorisations d'accès, aux vues et autres éléments systèmes. Le catalogue correspond à la réalisation de l'architecture à trois niveaux décrite précédemment. Le catalogue contient la description des différents schémas (physique, conceptuel et externes) ainsi que les règles de passage d'un schéma vers l'autre.

4.3.4 Le gestionnaire de requêtes

Est responsable du traitement des commandes des utilisateurs visant à stocker, rechercher et mettre à jour des données dans la base de données. En utilisant les informations stockées dans le catalogue, il interprète ces requêtes et les traduit en des requêtes d'accès physique aux données susceptibles d'être traitées par le système de gestion de fichiers.

4.3.5 Le gestionnaire de transactions

Est responsable de traiter les transactions. Une transaction est un ensemble d'opérations d'accès et de mise à jour de données émises par un utilisateur. Ces opérations doivent être traitées comme un tout et le gestionnaire de transaction doit assurer d'une part l'indivisibilité des opérations soumises, la cohérence du système après l'exécution de l'ensemble des opérations, l'isolation des traitements par rapports aux autres traitements qui peuvent être soumis de façon concurrente et enfin la persistance des résultats une fois l'ensemble des opérations achevées.

4.3.6 Le gestionnaire de reprise

Est un élément essentiel d'un SGBD qui doit remplacer le système de gestion de fichier traditionnel afin de minimiser les effets d'une panne sur la base de données. Un tel système ne peut évidemment pas être sûr ou sécurisé à 100 %. Néanmoins, il est indispensable que le gestionnaire de reprise puisse pallier à certaines pannes qui peuvent avoir différentes causes telle qu'une division par zéro dans un programme, à un problème de disque défectueux ou à une panne d'alimentation. L'objectif essentiel du gestionnaire de reprise est de restaurer la base de données dans un état cohérent. Vue les causes très différentes de panne et les difficultés liées à cette gestion, cet élément est un élément central qui concerne environ 10 % ou plus du code d'un SGBD. **[6]**

4.4 Les opérations sur les données

Il existe 4 opérations classiques (ou *requêtes*) :

1. La *création* (ou *insertion*).

2. La *modification* (ou *mise-à-jour*).

3. La *destruction*.

4. La *recherche*.

Ces opérations correspondent à des commandes du LMD (Langage de Manipulation des Données). La plus complexe est la *recherche* en raison de la variété des critères.

Pour l'utilisateur, une bonne requête a les caractéristiques suivantes. Tout d'abord elle s'exprime facilement : l'idéal serait de pouvoir utiliser le langage naturel, mais celui-ci présente trop d'ambiguïtés. Ensuite le langage ne devrait pas demander d'expertise technique (syntaxe compliquée, structures de données, implantation particulière ...). Il est également souhaitable de ne pas attendre trop longtemps (à charge pour le SGBD de fournir des performances acceptables). Enfin, et peut-être surtout, la réponse doit être fiable.

Une bonne partie du travail sur les SGBD consiste à satisfaire ces besoins. Le résultat est ce que l'on appelle un *langage de requêtes*, et constitue à la fois un sujet majeur d'étude et une caractéristique essentielle de chaque SGBD. Le langage le plus répandu à l'heure actuelle est SQL**. [7]**

4.5 Optimisation

L'optimisation (d'une requête) s'appuie sur *l'organisation physique des données*. Les principaux types d'organisation sont les fichiers séquentiels, les index (denses. secondaires, arbres B) et le regroupement des données par hachage.

Un module particulier du SGBD, *l'optimiseur*, tient compte de cette organisation et des caractéristiques de la requête pour choisir le meilleur séquencement des opérations.

5. Types de base de données

On retrouve fréquemment les bases de données suivantes :

- Référence (qui regroupe des informations bibliographiques ou factuelles).
- Texte intégral (qui regroupe des documents à caractère textuel).
- Multimédia (qui regroupe documents sonores, visuels, etc.).

Une base de données peut se trouver sur n'importe quel support : disque dur, cédérom, sur un serveur et accessible en réseau interne ou en ligne (à distance).

5.1 SGBD réparti ou SGBD distribué

Système gérant une collection de BD logiquement reliées, réparties sur différents sites en fournissant un moyen d'accès rendant la distribution transparente.

Deux approches fondamentales sont à l'origine de la conception des bases de données réparties : la conception descendante '*Top down design*' et la conception ascendante '*Bottom up design*'.

5.1.1 Conception descendante

On commence par définir un schéma conceptuel global de la base de données répartie, puis on le distribue sur les différents sites en des schémas conceptuels locaux. La répartition se fait donc en deux étapes, en première étape la fragmentation et en deuxième étape l'allocation de ces fragments aux sites.

L'approche top down est intéressante quand on part du néant. Si les BDs existent déjà, la méthode bottom up est utilisée. **[8]**

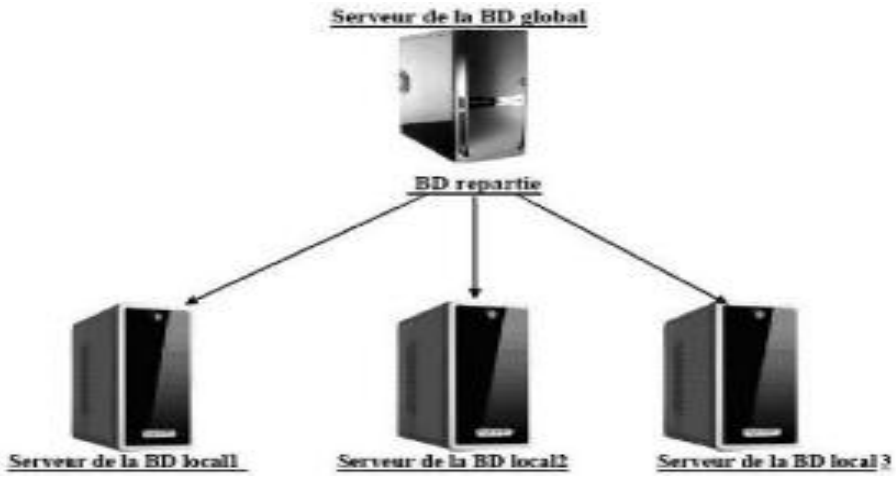


Figure I .6 : Architecture de la conception descendante.

5.1.2 Conception ascendante *(Base de données fédérée):*

Base de données fédérée (BDF) est une BD répartie hétérogène, c'est-à-dire constituée à partir de sources de données de nature variées : fichiers classiques, fichiers de textes, documents HTML, XML, BD relationnelle ou objet, etc. L'objectif est de fournir aux utilisateurs une vue intégrée de différentes données de l'entreprise soit dynamiquement sur demande, soit en la matérialisant périodiquement dans un entrepôt de données. (Plusieurs BD hétérogènes capables d'interopérer *via* une vue commune (modèle commun)).

Cette approche se base sur le fait que la répartition est déjà faite, mais il faut réussir à intégrer les différentes BDs existantes en une seule BD globale. En d'autre terme, les schémas conceptuels locaux existent et il faut réussir à les unifier dans un schéma conceptuel global. **[8]**

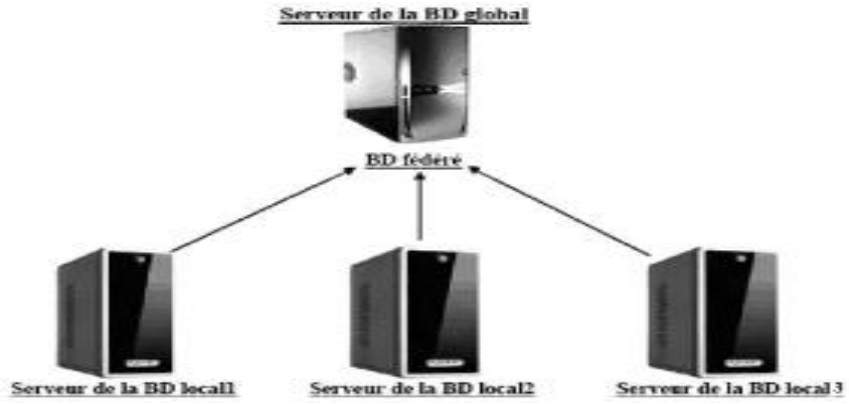


Figure I.7 : Architecture de la conception ascendante.

6. Sécurité et confidentialité d'une base de données

La base de données doit être sécurisée contre :

- Les indiscrétions : Par un mot de passe.
- Les erreurs : Des contrôles doivent être mis en place pour vérifier que des contraintes d'intégrités sont respectées.
- Les destructions : En cas d'incident (panne logicielle, panne matérielle ou panne d'électricité), des procédures de sauvegarde et reprise doivent être prévues afin de relancer le système sans avoir recommencé les saisies par la transaction. **[1]**

7. Conception d'une base de données

La conception d'un système d'information n'est pas évidente car, il faut réfléchir sur l'ensemble de l'organisation, que l'on doit mettre en place. La phase de conception nécessite des méthodes permettant de mettre en place un modèle sur lequel il faut s'appuyer. La modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon à faire ressortir les points auxquels l'on s'intéresse, Elle nécessite une analyse approfondie du monde réel ainsi que des besoins des futurs utilisateurs.

Des méthodes de conception de BDD ont donc été développées : UML, par exemple.

Une fois le schéma conceptuel de haut-niveau établi, il est traduit en termes du modèle conceptuel du SGBD choisi. **[2]**

8. Qui intervient sur une BDD ?

L'**administrateur** (une personne ou une équipe) :

Il définit le schéma conceptuel de la BDD et le fait évolué, comme il fixe les paramètres de l'organisation physique de façon à optimiser les performances, et aussi il gère les droits d'accès et les mécanismes de sécurité.

Les **programmeurs d'applications :**

Ils définissent les schémas externes et construisent les programmes qui alimentent ou exploitent la BDD en vue d'applications particulières. Ils utilisent pour cela le langage de bases de données du SGBD, éventuellement couplé avec un langage de programmation classique.

Les **utilisateurs finals** :

Ils accèdent à la BDD au travers des outils construits par les programmeurs d'application ou pour les plus avertis au travers du langage de requêtes. **[2]**

9. Principaux modèles logiques de SGBD [3]

Les bases de données sont apparues à la fin des années 60, à une époque où la nécessité d'un système de gestion de l'information souple se fait ressentir. Il existe cinq modèle de SGBD, les différenciés selon la représentation des données qu'elle contient :

9.1 Modèle hiérarchique

Le modèle hiérarchique est une forme de système de gestion de base de données qui lie des enregistrements dans une structure arborescente de façon à ce que chaque enregistrement n'ait qu'un seul processeur. Il s'agit du premier modèle de SGBD.

9.1.1 Caractéristiques générales du modèle :

- Forte dépendance entre la description de la structure des données et la manière dont celles-ci sont enregistrées sur le support physique.
- Les éléments de base du modèle sont des enregistrements logiques reliés entre eux pour constituer un arbre ordonné.
- Les entités (ou **segments**) constituent les nœuds, celui de plus haut niveau portant le nom de racine ; les branches (pointeurs logiques entre entités) constituent les **liens**. Chaque segment est une collection d'objets appelés champs (ou **Fields**).
- Chaque segment a obligatoirement un père (sauf la racine), et peut avoir plusieurs fils.

9.1.2 Avantages:

- Rigueur des structures et des chemins d'accès.
- Simplicité relative de l'implémentation.
- Adéquation parfaite du modèle à une entreprise à structure arborescente.

9.1.3 Inconvénients :

- Les accès se font uniquement depuis la racine.
- La structure interdit les liens N:M, ne permettant que le lien 1:N. La représentation d'autres relations impose de ce fait une redondance de l'information.
- Les "anomalies" que l'on constate lors des opérations de mise à jour (insertion, destruction, modification) : l'élimination d'un nœud entraîne l'élimination de tous les segments de niveau inférieur qui lui sont rattachés (risque de perdre des données uniques)
- Indépendance logique très réduite : la structure du schéma doit refléter les besoins des applications.
- Pas d'interface utilisateur simple.

9.2 Modèle réseau

Ce modèle utilise des pointeurs vers des enregistrements.

Evolution du modèle hiérarchique intégrant les résultats du travail du groupe CODASYL (comité de langage de programmation), qui avait démarré l'étude d'une extension de COBOL pour manipuler les bases de données. En 1969, il donne ses premières recommandations concernant syntaxe et sémantique du LDD et du LMD.
Même si cette vue est un peu simplificatrice, une base en réseau peut être décrite comme un certain nombre de fichiers comportant des références les uns vers les autres. Les entités sont connectées entre elles à l'aide de pointeurs logiques :

- Un enregistrement d'un ensemble de données A est associé à une série d'enregistrements (ou records) d'un autre ensemble de données B. On constitue ainsi des SET, ou COSET, structure fondamentale du modèle en réseau.
- Le lien entre les enregistrements de A et ceux de B est 1:N.
- Le COSET comporte un type d'enregistrement "propriétaire" (l'enregistrement de A est dit OWNER) et un type d'enregistrement "membre" (les enregistrements de B sont MEMBER).

9.2.1 Avantages et inconvénients du modèle

- Aucune restriction dans la conception : un type de "record" peut à la fois être propriétaire et membre de plusieurs sets.
- Représentation naturelle des liens maillés N:M.
- Pas d'anomalies pour les opérations de stockage.
- Commercialisation importante des systèmes correspondants (DMS, IDMS, TOTAL, IDS II, SOCRATE...),mais pas d'indépendance par rapport aux stratégies d'accès.
- Procéduralité importante des langages de manipulation ; l'utilisateur doit "naviguer" dans le réseau logique constitué par les enregistrements et les chaînes de pointeurs.

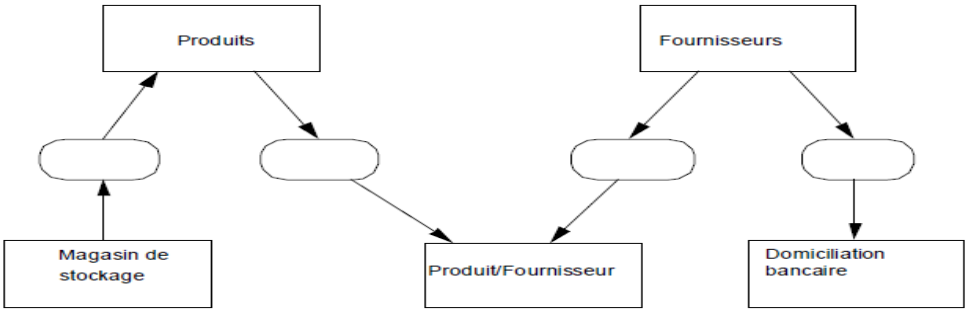


Figure I .8: Schéma représentant le sous-système d'information.

9.3 Modèle relationnel

C'est un article publié en 1969 par un mathématicien du centre de recherche IBM, **Codd**, qui définit les bases de ce modèle relationnel. Codd s'est intéressé au concept d'information et a cherché à le définir sans se préoccuper de la technique informatique, de ses exigences et de ses contraintes. Il a étudié un

modèle de représentation des données qui repose sur la notion mathématique de "relation".

9.3.1 Définitions

- Une relation est un ensemble de tuples (lignes), dont l'ordre est sans importance. Les colonnes de la table sont appelées attributs ou champs. L'ordre des colonnes est défini lors de la création de la table.
- Une clé est un ensemble ordonné d'attributs qui caractérise un tuple. Une clé primaire le caractérise de manière unique, à l'inverse d'une clé secondaire.
- On dit qu'un attribut A est un **déterminant** si sa connaissance détermine celle de l'attribut B (B dépend fonctionnellement de A).

9.3.2 Caractéristiques du modèle

- Schéma de données facile à utiliser : toutes les valeurs sont des champs de tables à deux dimensions.
- Améliore l'indépendance entre les niveaux logique et physique : pas de pointeurs visibles par l'utilisateur.
- Fournit aux utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non-informaticiens (SQL, L4G) et un ensemble d'opérateurs basé sur l'algèbre relationnelle : union, intersection, différence, produit cartésien, projection, sélection, jointure, division.
- Optimise les accès aux bases de données.
- Améliore l'intégrité et la confidentialité : unicité de clé, contrainte d'intégrité référentielle.
- Prend en compte une variété d'applications, en gestion et en industriel.
- Fournir une approche méthodologique dans la construction des schémas.

Il est évident qu'une base de données relationnelle présente des avantages tels que :

- L'indépendance des utilisateurs vis à vis de la structure logique, la structure physique, la stratégie d'accès aux données.
- La puissance de représentation grâce à conception rigoureuse du schéma (approche méthodologique).

- La rapidité d'écriture du code.
- La manipulation par des non informaticiens pour des cas simples sinon la connaissance de la structure de la base de données et de la maîtrise de l'algèbre relationnelle sont nécessaires.
- L'approche non procédurale permettant un traitement uniforme de la définition, de la manipulation et du contrôle des données.
- L'évolutivité.
- La prise en compte des contraintes d'intégrité.

9.4 Modèle déductif

Un SGBD Déductif est un SGBD qui peut faire des déductions sur la base **des règles** et des **faits** enregistrés dans la base de données.

Le but est d'utiliser des méthodes semblables à celles pratiquées pour la déduction en intelligence artificielle.

Une base de données déductive (BDD) est constituée de: **BDE** et **BDI.**

Base de données extensionnelle (BDE): Ensemble des faits connus (tuples) dans la BDD relationnelle.

Base de données intentionnelle (BDI): Ensemble des faits ou règles déduits.

Les BDD déductives ne sont pas une nouvelle technologie de SGBD, mais elles rajoutent des fonctionnalités supplémentaires qui permettent de générer des données dérivées (ou déduites) à partir de données déjà existante dans la base. **[16]**

9.5 Modèle objet

Modèle objet est l'annuaire, qui est capables de stocker une multitude d'informations. Il stocke l'information dans des objets, très souvent une fiche individuelle, une machine, une ressource. à laquelle on associe des valeurs, ses attributs. **[10]**

9.6 Modèle multidimensionnel

Il permet de stocker différentes données numériques aux croisements des "n" axes correspondant aux "n" dimensions de la base.
Il est alors possible de naviguer dans cet espace, à différents niveaux d'agrégats (zooms, rotation d'axes, etc.) : ces bases de données sont appelées cubes ou hypercubes en informatique décisionnelle et sont souvent utilisés dans les métiers du contrôle de gestion. **[10]**
Les bases multidimensionnelles sont le plus souvent formées par agrégats de bases pouvant être relationnelles, en tout cas hétérogènes. **[11]**

9.7 Modèle semi-structuré

La popularité du web et l'essor de XML ont contribué à l'émergence des bases de données semi-structurées et des bases de documents. Les modèles classiques de bases de données relationnelles ou objets supportent difficilement les données semi-structurées qui sont complexes, hétérogènes, distribuées, parfois incomplètes. La force des modèles semi-structures est de ne plus imposer de structure a priori dans le schéma mais de la définir a posteriori dans les données elles-mêmes. Plus récemment, les chercheurs s'intéressent à l'extraction de connaissances à partir de données semi-structurées du web. L'objectif de bases de données semi-structurées est de faire le point sur les avancées actuelles dans le domaine des bases de données semi-structurées d'un point de vue théorique et de recenser les applications originales qui s'appuient sur ce formalisme. **[12]**

10. Conclusion :

En d'autres termes, « les bases de données assurent une gestion de données exactes, complètes et disponibles à tout moment, depuis n'importe où et dans la forme voulue ».

Les SGBD les plus évolués actuellement disponibles disposent de la plupart de ces fonctions, mais pas toutes. Il en résulte des différences significatives entre leurs caractéristiques, leur fonctionnement et leurs usages possibles.

Il convient de signaler également que les bases de données sont plus qu'une nouvelle technique de stockage et de manipulation des données. Elles impliquent une nouvelle approche de la conception et de l'utilisation des systèmes d'information et peuvent avoir des conséquences organisationnelles qui sortent largement du cadre du service informatique. Elles obligent les utilisateurs à considérer les données comme une ressource de l'entreprise qui

doit être gérée comme le sont les ressources traditionnelles (personnel, locaux, moyens de production, capitaux) pour être accessible à un grand nombre d'utilisateurs.

CHAPITRE II:

Sécurité dans les bases de données

1. Introduction

La préservation de la confidentialité est devenue une priorité pour les citoyens ainsi que pour les administrations. Le besoin d'accumuler, de partager et d'analyser des données personnelles est multiple : pour l'amélioration de la qualité des soins grâce au dossier médical électronique pour rendre plus simples et efficaces les procédures administratives, pour personnaliser les services rendus par une grande quantité d'objets éle²ctroniques dans un environnement d'intelligence ambiante ou même pour la lutte contre le terrorisme (croisement de bases de données commerciales et gouvernementales pour la recherche de suspects). Bien que le traitement de données personnelles ait généralement un but louable, il constitue une menace sans précédent aux droits élémentaires à la protection de la vie privée.

Partout dans le monde, les gouvernements adoptent des lois spécifiques pour cadrer l'utilisation de données personnelles comme le 'Federal Privacy Act' aux USA ou la directive pour la protection des données en Europe. Il est cependant difficile de traduire ces lois en moyens technologiques convaincants garantissant leur application.

Comme l'atteste le rapport 'Computer Crime and Security Survey' établi par le Computer Security Institute et le FBI, le nombre d'attaques de serveurs de bases de données est croissant malgré la mise en place de politiques de sécurité de plus en plus drastiques. Pire encore, presque la moitié de ces attaques sont conduites par des employés ayant légalement accès à tout ou partie des données. Ceci montre la vulnérabilité des techniques traditionnelles de sécurisation des serveurs bases de données.

2. Sécurité

Bien que la sécurité soit l'une des raisons de l'architecture trois couches, plusieurs challenges praticables sont enlevés lors de la construction du système tel que l'authentification des utilisateurs, le contrôle des accès et Audit les actions des utilisateurs, la protection des données entre les couches, la limitation des privilèges de l'intermédiaire, et la construction des systèmes extensibles.

2.1. Propriétés principales

La sécurité des bases de données inclus trois principales propriétés : **la confidentialité**, **l'intégrité** et **la disponibilité**.

- **Confidentialité :** L'information protégée ne doit pas être accessible aux utilisateurs ou un programme non autorisés. C'est crucial:
  - ❖ Dans des environnements critiques ou stratégiques: militaires ou commerciaux, par exemple.
  - ❖ Pour respecter le droit des individus à décider comment et dans quel but les informations les concernant peuvent être extraites, mémorisées ou transmises à d'autres individus.

- **Intégrité :** Les données ne peuvent être modifiées que par les utilisateurs habilités à le faire qu'elles soient dues à :
  - ❖ Des pannes de système,
  - ❖ Des manipulations erronées,
  - ❖ Des sabotages.

- **Disponibilité :** Il s'agit de détecter ou d'empêcher des **dénis de service**.

Il y a déni de service lorsqu'un utilisateur ne parvient pas à accéder dans un **délai raisonnable**, à une information ou à une ressource pour laquelle il a une autorisation d'accès. **[13]**

2.2. Processeur de sécurité

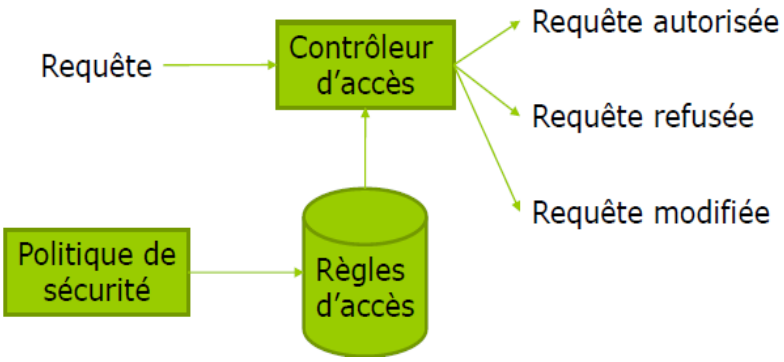


Figure II.1: Processeur de sécurité. [13]

Autorisation, interdiction et obligation :

- Les règlements les plus simples ne contiennent que des **autorisations**: ce qui n'est pas autorisé est interdit.
- Certains règlements incluent des **interdictions** à fin de spécifier des exceptions à des permissions générales. Exemple: les patients ont droit de consulter leur dossier médical sauf Jean Dupont.
- D'autres enfin, plus sophistiqués, incluent des **obligations**: difficiles à implanter dans les systèmes informatiques. **[13]**

2.3. Attaque

- Les violations de la sécurité d'une BD consistent en des lectures ou des mises à jour illicites.
- Les événements qui portent ces violations sont appelés des **attaques**.

Les attaques à une BD peuvent exploiter les failles des applications opérant sur cette BD:

- ❖ Stockage des mots de passe dans les fichiers de configuration de l'application,
- ❖ Scripts de connexion à la BD accessibles dans le code source de l'application,
- ❖ Attaques par injection SQL,
- ❖ Attaques exploitant les débordements de tampons. **[13]**

2.3.1. Types d'attaques

On distingue:

- Les attaques **non frauduleuses**:
  - ❖ Catastrophes naturelles,
  - ❖ Pannes de logiciel ou de matériel,
  - ❖ Erreurs humaines…
- Les attaques **frauduleuses**:
  - ❖ Utilisation abusive de leurs droits par les utilisateurs,
  - ❖ Agents hostiles exécutant des actions de destruction du logiciel ou du matériel, ou lisant ou mettant à jour des données protégées,
  - ❖ Ces agents peuvent être cachés dans des actions légales : **chevaux de Troie**. **[13]**

2.3.2. Qui attaque?

Dans cette partie on présente les différents attaquants :

- **Pirate externe :** il est capable de s'infiltrer sur le serveur BD et de lire ses fichiers, il peut aussi casser une clé de chiffrement avec un texte connu.
- **Pirate utilisateur :** ce type de pirate est reconnu par le SGBD et à accès à une partie des données suivant le mode de chiffrement, il a accès à certaines clés.
- **Pirate administrateur (DBA) :** employé peu scrupuleux ou pirate s'étant octroyé ces droits ; A accès à des données inaccessibles aux autres pirates (journal) et aussi peut espionner le SGBD pendant l'exécution. **[18]**

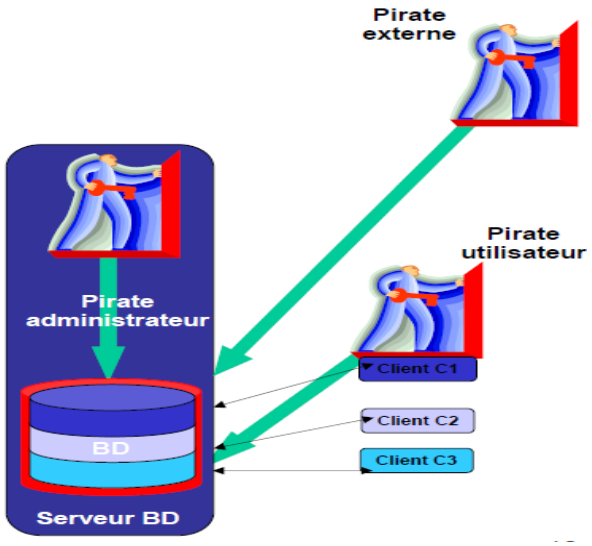


Figure II.2: Différents attaquants. [18]

2.3.3. Les risques encourus

Les risques propres à une source de données sont les suivants :

Le **vol de données** induit la perte de **confidentialité** des données stockées. La divulgation de données financières hautement confidentielles peut avoir un impact néfaste sur l'activité d'une entreprise : risque juridique, atteinte à l'image de marque, perte de confiance des partenaires industriels...

L'**altération de données** induit une perte d'**intégrité**, c'est-à-dire que les données ne sont plus dignes de confiance. En fonction de la rapidité de détection et de la qualité des sauvegardes, les conséquences peuvent en être réduites. Mais une application fonctionnant sur des données falsifiées peut voir son comportement fortement influencé : par exemple, un site de commerce électronique pourrait débiter le compte d'un autre client que celui réalisant la commande !

La **destruction de données** remet sérieusement en cause la **continuité de l'activité** de l'entreprise concernée. Privée de ses données clients, sans sauvegarde, c'est le dépôt de bilan garanti !

L'**augmentation du niveau de privilèges** d'un utilisateur d'une application est plus insidieuse que les risques précédents, car comme pour l'altération de données, il n'est remarqué qu'après un certain laps de temps durant lequel le pirate peut réaliser un grand nombre d'actions malveillantes. Il peut ainsi s'attribuer le droit d'accès à des informations confidentielles, le droit d'accès à des opérations sensibles, voire même prendre le contrôle d'une application.

Selon le SGBD utilisé, des **ressources systèmes** peuvent être attribuées à chaque utilisateur (nombre de requêtes par unité de temps...). Ces ressources peuvent être limitées par l'administrateur système afin d'éviter l'écroulement des capacités de traitement du serveur (**déni de service**) par un utilisateur malveillant. De plus, ceci permet de limiter la portée d'une attaque par altération ou vol de données en limitant le nombre d'opérations réalisables en un temps donné. La conséquence d'un tel risque peut être la paralysie du serveur (perte de **disponibilité**).

On le voit, les risques sont variés et leurs conséquences potentiellement dramatiques. Ainsi, il est nécessaire d'attribuer les droits d'accès avec parcimonie. **[21]**

2.4. Les types d'utilisateurs

Il faut identifier les utilisateurs ayant besoin d'un accès à la base de données, ils peuvent être de différents types :

L'**administrateur** est une personne physique ayant tous les droits sur le SGBD, mais pas forcément sur le contenu des bases de données : il peut réaliser des opérations de gestion des droits d'accès et des ressources systèmes mais on pourra choisir d'exclure ou non les droits d'accès en lecture et/ou écriture au contenu des bases de données. Bien que parfaitement logique d'un point de vu métier, pour la protection de données sensibles par exemple, retirer à un administrateur les droits de lecture et d'écriture sur le contenu d'une base de données n'a pas de sens d'un point de vu technique puisqu'il possède les capacités techniques de s'octroyer ses droits là. De plus, les opérations de

sauvegarde, de restauration et de maintenance après incident peuvent l'amener à devoir accéder au contenu d'une base de données.
Bref, normalement, c'est l'utilisateur qui a tous les droits sur le SGBD et les bases de données hébergées. C'est normalement une personne de confiance, compétente et prudente.

Une **application** peut être une application web, un outil de synchronisation entre sources d'informations ou tout programme accédant pour lui-même à la base de données. Ce type d'utilisateur logique n'a rien à voir avec l'utilisateur réel dénotant une personne physique ayant des besoins particuliers. Même si une application est utilisée par des personnes physiques, on pourra choisir de déléguer à l'application la gestion des droits d'accès à l'information en fonction des habilitations qu'elle décide de lui attribuer. Ainsi, une application peut être vue comme un utilisateur de base de données auquel on attribue des droits qu'elle pourra restreindre de façon transparente pour l'utilisateur final de l'application ainsi que pour le SGBD.

L'**utilisateur** est une personne physique se connectant directement à la base de données (commande *mysql* sous Linux) ou via une interface graphique (script *phpMyAdmin* sur un Intranet) ou utilisant une application qui va se connecter à la base de données sous l'identité de l'utilisateur (client lourd *MySQL Query Browser*). **[21]**

2.5. Politique de sécurité

Les ITSEC (**Information Technology Security Evaluation Criteria)** définissent la politique de sécurité comme l'ensemble des lois règles ou pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système d'information. **[13]**

3. Les moyens de sécurité

- **Protections contre les attaques**

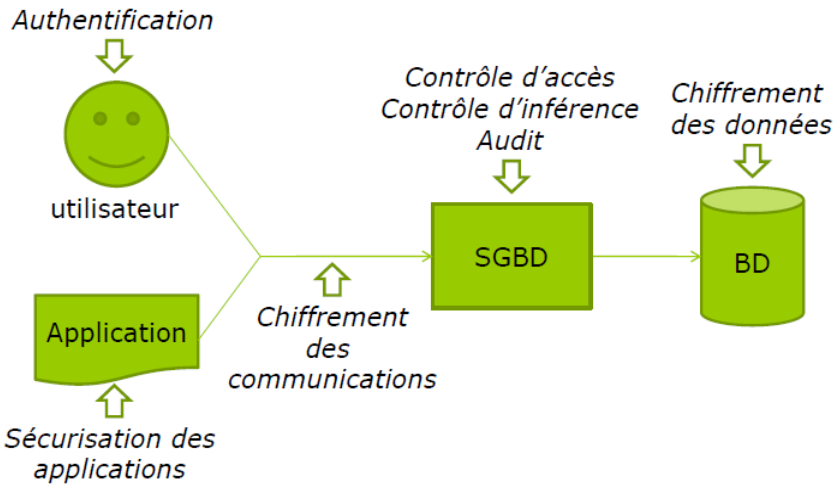


Figure II.3: Protections contre les attaques. [13]

Les SGBD fournissent différents moyens pour garantir la sécurité :

3.1 Vues

La base de données du fournisseur SQL Server comprend des vues prédéfinies qui permettent d'accéder aux données d'une fonctionnalité particulière sans accéder directement aux tables de base de données. L'accès aux vues fournies est en lecture seule. Vous ne devez pas essayer de mettre à jour les données de la base de données à partir des vues.

Importance des vues : elles permettent de définir de façon précise les portions d'une BD sur lesquelles des privilèges sont accordés. **[22]**

3.2 L'authentification des utilisateurs

- L'**authentification** a pour objectif d'assurer que l'utilisateur qui se connecte à la BD est:
  - ❖ Autorisé à se connecter,
  - ❖ Bien celui qui s'annonce.

- L'authentification repose sur :
  - ❖ La sécurité des mots de passe,
  - ❖ Des techniques d'identification biométriques. **[13]**

Dans les architectures à deux niveaux où les utilisateurs se connectent directement au serveur, la base de données fait authentifiée les utilisateurs lors de la connexion et leur associée le nécessaire pour travailler. Typiquement, dans les architectures trois tiers, l'intermédiaire est le responsable de l'authentification des utilisateurs, et de plus il traite les données envoyées par les utilisateurs avant de les transmis vers le serveur de base données. Un seul intermédiaire est commun entre les utilisateurs et la base de données (un seul point d'entrée), pour cette raison la base de donnée délègue l'authentification des utilisateurs à l'intermédiaire, dans ce niveau la réalisation d'une authentification est plus compliquée que l'architecture deux tiers. **[14]**

3.3 Le contrôle d'accès des utilisateurs

Afin de permettre l'implantation de politiques de confidentialité et d'intégrité en leur sein, les systèmes d'exploitation disposent de mécanismes de contrôle d'accès. Typiquement, ceux-ci fonctionnent sur le modèle suivant :

- Un ***sujet*** est une entité active inclut souvent les utilisateurs et les processus travaillant pour le compte des utilisateurs (qui peuvent être classés par groupes).
- Un ***objet*** est une entité passive, un conteneur d'information à protéger, sur lequel un sujet peut effectuer une action (les fichiers, Données, programmes, périphériques matériels) ;
- Une ***permission*** est une certaine action permettent aux sujets de manipuler les objets. Une permission peut être accordée ou refusée (par exemple, lecture, écriture, exécution).
- Un ***règlement de sécurité*** constitué d'un ensemble de **règles d'accès** traduisant la **politique de sécurité** du système d'information.
- Un ***processeur de sécurité*** qui vérifie que les requêtes adressées au système ne violent pas les règles d'accès et selon le cas autorise, modifie ou interdit la requête. **[13,20]**

Le contrôle d'accès est configuré par un ensemble de règles spécifiant un sujet, un objet et des droits d'accès. Un cas particulier de règles est celui spécifiant

une action d'un sujet vers un autre sujet (envoi de signal ou de message inter-processus).

Une fois que l'utilisateur est authentifié par l'intermédiaire, le système doit contrôler quelles données, applications et ressources l'utilisateur peut accéder dans le système. Les données ne doivent pas être protégées seulement contre les intrusions mais aussi les accès des utilisateurs ayant des limites qui doivent être respecté. Pour contrôler l'accès il faut d'abord renforcer la manière dont les utilisateurs font accès.

La sécurité typique dans les systèmes trois tiers, exige que chaque utilisateur soit limité par l'exécution des applications spécifiques sur l'intermédiaire, dépendant sur l'identité de l'utilisateur et le rôle lui accorder dans l'organisation. Un utilisateur qui accède à partir d'un intermédiaire ne doit pas avoir la permission d'accéder directement à la base de données**. [14]**

3.3.1 Modèles de contrôle d'accès

Définition : une politique de contrôle d'accès est un ensemble de règles.

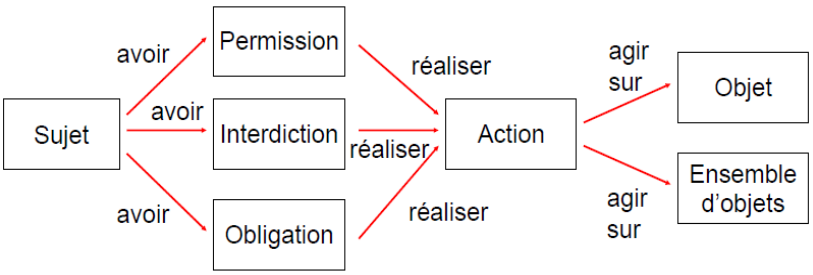


Figure II.4 : Format des règles. [18]

Les modèles de contrôle d'accès permettent de définir le cadre d'expression d'une politique de sécurité.

Cette partie on présente l'état de l'art des principaux modèles de contrôle d'accès statiques : DAC, MAC et RBAC.

Les parties suivantes présentent les différentes familles de modèles théoriques. On distingue principalement trois grandes catégories de modèles de contrôle d'accès :

❖ Modèles discrétionnaires

Contrôle d'accès discrétionnaire; L'administration des droits dans le modèle DAC repose sur la notion de propriétaire : chaque objet a un propriétaire qui décide quels sont les autres sujets qui ont accès à cet objet, étant donné que l'attribution des droits est faite par les utilisateurs et non pas par les administrateurs.

Voici les principes d'accès qui s'appliquent au modèle "discrétionnaire" :

- Un sujet dispose du plein contrôle des objets dont il est propriétaire.
- Le propriétaire est souvent le créateur des objets.
- Le propriétaire détermine les permissions et droits d'accès aux ressources sous son contrôle.

Le plus célèbre modèle discrétionnaire est le modèle **HRU** qui a été défini en 1976 par Harrison, Ruzzo et Ullman. Il s'agit d'un modèle matriciel défini à partir d'un ensemble de sujets, d'un ensemble d'objets et d'un ensemble fini de règles. Ces règles décrivent dans quelles conditions un sujet peut modifier le contenu de la matrice d'accès.
Typiquement, les droits d'accès sur un fichier sont positionnés par l'utilisateur déclaré comme propriétaire de ce fichier. **[19]**

Avantages et inconvénients

- Ils sont simples à mettre en œuvre.
- Ce sont les plus implantés: UNIX, SQL…
- Le contrôle de la propagation des droits et celui de la révocation des droits propagés posent des problèmes difficiles.
- Ils sont vulnérables aux **chevaux de Troie**.

❖ Modèles obligatoires

Contrôle d'accès obligatoire Contrairement au DAC, le MAC ou *Mandatory Access Control* délègue l'attribution des permissions à une entité tierce, typiquement un administrateur externe de la politique de sécurité. Ainsi, les utilisateurs du système ne peuvent pas intervenir dans l'attribution des

permissions d'accès, même s'ils disposent de droits d'administration dans le système d'exploitation. **[20]**

Ce module représente les mécanismes de contrôle d'accès implémentés par l'équipement auprès de l'agent de contrôle d'accès. Ce module reçoit les règles de contrôle d'accès de la part du module de configuration du contrôle d'accès (MCCA). Ces règles sont traduites par le MCA dans les syntaxes réelles des commandes nécessaires à la configuration des mécanismes implémentés. En retour de ces commandes le MCA reçoit des résultats correspondant à l'application des commandes et transmet ces résultats au module de gestion centralisée du contrôle d'accès (MGCCA). Ils sont des modèles **multi-niveaux**.

- Il s'agit de protéger le secret et l'intégrité.
- Les sujets (ou utilisateurs) et objets sont classés par niveaux (ex: Top secret, Secret, Confidentiel, Public).

1 . Modèle de Bell et LaPadula (BLP)

Le modèle de Bell et LaPadula a pour objectif la protection du secret des informations. Il est basé sur la classification des **objets** et des **sujets** par **niveaux de secret**.

L'ensemble des niveaux de secret est muni d'un ordre partiel (>).

Par exemple : Top secret (TS) > Secret (S) > Confidentiel (C) > Non classifié (NC).

Politique de sécurité : La politique de sécurité du modèle de Bell et LaPadula se résume dans les deux principes :

- **No Read Up Secrecy** : préserve de la lecture d'une information dans un objet par un sujet de niveau de secret inférieur.
- **No Write Down Secrecy** : préserve d'un transfert d'information d'un objet vers un autre objet de niveau de secret inférieur, par un utilisateur qui n'est pas de confiance. **[13]**

2. Modèle de Biba :

Le modèle de Biba (K. J. Biba, 1977) a pour objectif la protection de l'intégrité des informations. Il applique à la protection de l'intégrité une stratégie similaire à celle de la protection du secret par le modèle BLP : les sujets et les objets sont classés par **niveaux d'intégrité**.

L'ensemble des niveaux d'intégrité est muni d'un ordre partiel (>).
Par exemple : Crucial (TS) > Très important (TI) > Important (I) > Non classifié (NC)

Politique de sécurité : La politique de sécurité du modèle de Biba se résume dans les deux principes :

- **No Write Up Integrity** : préserve de l'écriture d'une information dans un objet par un sujet de niveau d'intégrité inférieur.
- **No Read Down Integrity :** préserve d'un transfert d'information d'un objet vers un autre objet de niveau d'intégrité supérieure par un utilisateur qui n'est pas de confiance.

Avantages et inconvénients

- Ils sont bien adaptés aux applications où la protection du secret et de l'intégrité est primordiale.
- Mais ils sont trop rigides et centralisés.
- La confidentialité est assurée au détriment de la disponibilité. **[13]**

❖ Modèles à base de rôles

Le modèle RBAC pour *Role-Based Access Control* a pour but de simplifier l'administration des droits d'accès des utilisateurs individuels en fournissant un niveau d'indirection supplémentaire. Plutôt que de donner directement des permissions aux utilisateurs, on définira différents rôles possibles pour l'utilisation du système d'exploitation, avec des droits d'accès associés. Ensuite chaque utilisateur a accès à une liste de rôle, suivant son activité. Un seul rôle peut être actif à un moment donné. **[20]**
On peut améliorer les modèles discrétionnaires en créant des rôles qui sont des ensembles d'autorisation. Les autorisations sont octroyées aux rôles et les rôles aux utilisateurs.

Pour pouvoir réaliser une action sur un objet un utilisateur doit ouvrir une session et activer celui de ses rôles qui contient l'autorisation de réaliser cette action.

Un rôle peut hériter des privilèges d'un autre rôle ; Sont bien adaptés aux applications de gestion. Permettent une administration souple des privilèges. **[13]**

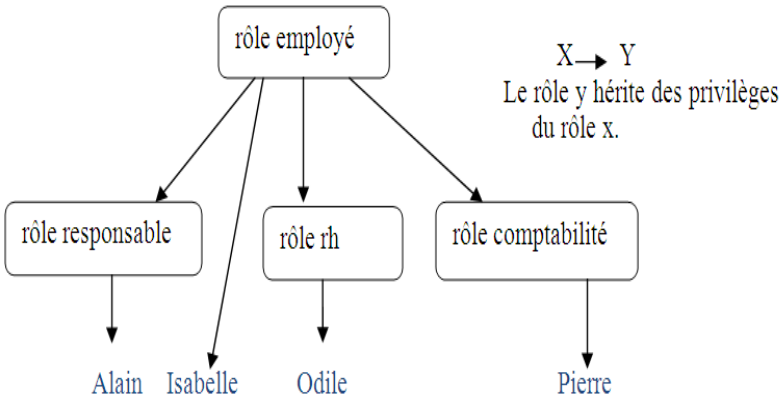


Figure II.5: Exemple d'héritage de rôle.

Les modèles de contrôle d'accès comme DAC, MAC ou RBAC ne permettent de modéliser que des politiques de sécurité qui se restreignent à des permissions statiques. Ils n'offrent pas la possibilité d'exprimer des règles contextuelles relatives aux permissions, aux interdictions, aux obligations et aux recommandations. Ce type de règle est particulièrement utile pour exprimer des politiques de sécurité dans le domaine médical. Dans l'article **[17]**, les auteurs proposent un nouveau modèle qui permet de spécifier de telles politiques de sécurité contextuelles. Ce modèle appelé Organisation Based Access Control (ORBAC) s'appuie sur un langage formel basé sur la logique du premier ordre L'approche ORBAC pour sécuriser l'interopérabilité repose sur la création de politiques de sécurité d'interopérabilité.

3.4 Protection des données de l'utilisateur

Le changement des données entre les trois tiers doits être protégé contre les révélations et les modifications non attendus, le cryptage est le mécanisme standard pour ce but. Le SSL (Secure Sockets Layer) est le protocole qui

assurer le cryptage et la communication confidentielle dans le réseau entre le client et l'intermédiaire aussi qu'entre l'intermédiaire et la base de données. **[14]**

La sécurisation des données permet de crypter ou de limiter la vision des données pour un utilisateur au niveau d'une base, d'une table, d'une colonne. La norme SQL ANSI permet déjà à travers la gestion des droits (GRANT, REVOKE) d'autoriser un utilisateur à voir ou modifier des informations sur une table, ou sur certaines colonnes. Par contre les données sont stockées en clair, non cryptées.

Dans le cas de données sensibles, il est important de proposer des solutions de cryptage des données. Pour répondre à ces problématiques, les éditeurs de SGBD proposent essentiellement deux solutions :

• La première consiste à utiliser des fonctions de cryptage directement dans le code SQL. L'emploi des fonctions de cryptage oblige à une modification du code applicatif. Les principaux algorithmes disponibles suivant les SGBD sont DES, 3DES, AES, MD5, MD4, SHA et SHA-1.

• La deuxième consiste à mettre en place un système de cryptage connu sous le nom générique de TDE (Transparent Data Encryption). Cette solution de cryptage est transparente pour les applications (Il n'y a pas besoin de modifier le code applicatif). En général, cette solution permet de protéger les fichiers de la base ainsi que les sauvegardes.

En ce qui concerne TDE (Transparent Data Encryptions), actuellement seul Oracle, SQL Server et Sybase proposent ce mode de cryptage. Sous Oracle la fonctionnalité **TDE** qui fait partie de l'option **Oracle Advanced Security,** permet d'effectuer un cryptage au niveau de la colonne ou du *tablespace*.

Pour cela, Oracle utilise une clé externe (master key) qui peut être stockée dans un Oracle wallet (protégé par mot de passe). Si un utilisateur accède à la base, les données seront décryptées automatiquement. **[16]**

3.5 Audit des accès de l'utilisateur

L'audit des accès est nécessaire pour déterminer l'utilisateur responsable de telle action dans la base de données, et comme les utilisateurs accèdent à partir d'un intermédiaire, il est difficile à un système audit de garder la trace et de corréler les activités qui peuvent être sensitives à la sécurité. **[14]**

Un outil indispensable pour assurer la sécurité d'une BD est l'**audit** basé sur un journal des différents types d'accès à la BD :

- Audit des entrées dans la base.
- Audit des utilisations de la BD en dehors des heures ouvrables.
- Audit de la manipulation du schéma.
- Audit des erreurs.
- Audit des modifications des sources des procédures stockées et des triggers.
- Audit des modifications des attributs de sécurité (login, privilèges...). **[13]**

3.6 Limitation du privilège de l'intermédiaire **[21]**

3.6.1 Principe du moindre privilège

Ce Principe stipule qu'un sujet ne doit disposer que des droits d'accès minimum pour assurer l'exécution des tâches qui lui sont assignées, pas un de plus.

Ex : ne pas donner les droits de l'administrateur à tout utilisateur d'un système (système d'exploitation, SGBD).

Puisque le mécanisme d'authentification de l'intermédiaire est moins fort que celui de la base de données et que l'intermédiaire situé en d'hors de la zone protégée par le firewall en face des intrusions interviennes de l'Internet, la base de données doit limiter les privilèges d'un intermédiaire, et de le permis d'accéder au nom des utilisateurs spécifiques. **[14]**

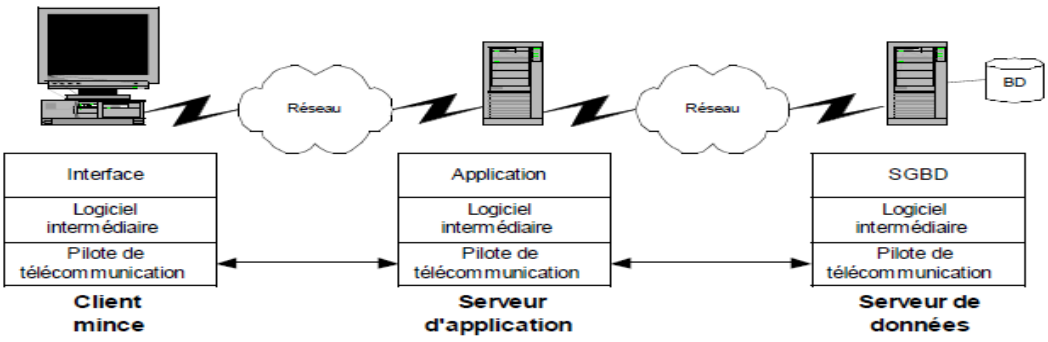


Figure II.6: Architecture à trois tiers. [14]

3.6.2 Politique de gestion des privilèges

A. Les privilèges

Il convient pour chaque compte d'accès d'identifier les privilèges minima à accorder ainsi que le niveau de granularité adéquat.
Ici ; le terme *utilisateur* désigne une application aussi bien qu'une personne physique.

A.1. Classes d'objets et granularité

Les SGBD permettent généralement de spécifier assez finement les privilèges d'un utilisateur en fonction des objets manipulé :

- Base de données.
- Table (relation).
- Colonne (attribut).

Ainsi, un utilisateur peut se voir attribuer un privilège pour toute une base de données, ou seulement pour quelques tables, ou encore sur uniquement quelques colonnes de certaines tables.

A.2. Classes de privilèges

Les privilèges s'organisent autour de plusieurs classes :

- Accès au contenu de l'information.
- Gestion du schéma de la base de données.
- Gestion des privilèges utilisateurs.
- Gestion des paramètres systèmes.

B. Règles d'attribution des privilèges

Règle fondamentale n°1 : attribution du moindre privilège.
Les utilisateurs ne doivent avoir que le minimum de droits, ceux strictement nécessaires à l'accomplissement de leurs tâches.

Règle n°2 : contrôle de la population.

Les privilèges doivent êtres synchrones avec la réalité de la population : il faut supprimer les comptes des utilisateurs quittant l'entreprise et de ceux n'étant plus affectés à telle ou telle tâche.

Règle n°3 : supervision de la délégation des tâches d'administration.

Un administrateur peut être amené à déléguer auprès d'une autre personne les tâches d'attribution des privilèges de tout ou partie de la population des utilisateurs. Un contrôle *a posteriori* doit être réalisé afin de vérifier que le résultat de cette délégation est conforme à la politique adoptée.

Règle n°4 : contrôle physique des connexions.

Il est nécessaire de restreindre les connexions à des hôtes spécifiques connus.

Règle n°5 : limitation des ressources utilisées.

Le SGBD offre souvent la possibilité de restreindre les ressources de calcul disponibles pour un utilisateur. Il est recommandé de configurer ces limitations de ressources en fonction de la charge maximale attendue pour un utilisateur.

Règle n°6 : journaliser les comportements suspects.

Certains SGBD permettent de conserver dans des **journaux de log** les requêtes non conformes aux privilèges accordés à un utilisateur. Il peut être intéressant de les surveiller afin de détecter toute anomalie dénotant des tentatives de piratage.

Règle n°7 : restrictions sur une application en fonction du public.

Une même application web peut avoir plusieurs interfaces différentes selon le contexte d'utilisation : internet / intranet.

C. Contrôle des privilèges

La principale question qui se pose lors du développement d'une application, c'est quelle stratégie adopter vis à vis des utilisateurs : contrôle de leurs droits d'accès par l'application ou par le SGBD ?

Par le SGBD. Dans le cas où toute l'information métier repose sur une base de données comportant également toutes les procédures stockées de contrôle de l'intégrité, de la logique métier et des actions utilisateurs, il est logique de déléguer au SGBD le contrôle d'accès et les habilitations. Ceci suppose que l'administrateur de bases de données réalise les opérations d'attribution des

privilèges et de synchronisation avec l'annuaire des utilisateurs du système d'information de l'entreprise. L'application ne devient alors qu'une interface graphique ergonomique d'interrogation de la base de données métier.

Par l'application. Dans le cas où l'application gère elle-même le niveau d'accréditation des utilisateurs, elle va se connecter sous sa propre identité logique à la base de données et décider des informations et des opérations que l'utilisateur peut voir, modifier et réaliser. C'est la stratégie employée par les applications dont la logique métier n'est pas intégrée directement dans la base de données et qui gèrent plusieurs sources de données.

D. Contrôle d'inférence

L'objectif du **contrôle d'inférence** est protéger une BD des attaques consistant à déduire des données non autorisées à partir de données autorisées.

Ex : Interdire l'accès à des données individuelles dans une BD statistiques à partir de requêtes agrégatives (comptage, somme, moyenne).

3.7 Base privée virtuelle (Virtual Private Database ou VPD)

A l'incrémentation des besoins des organisations et des entreprises, la base privée

virtuelle proposée par oracle pourvoi une grande sécurité d'accès contrôle. Et ceci par la création d'une politique de sécurité -au niveau de la base de données- commun entre toutes les applications, ce qui permet à chaque utilisateur d'accéder seulement à ces données, et quelque soit la manière de la quelle l'utilisateur accède à la base il ne peut pas dépasser cette politique de sécurité. **[14]**

Principe

- Associer une règle de sécurité à une table ou une vue.
- Si la VPD est activée, toute requête qui accède à cette relation ou cette vue est automatiquement modifiée en incluant une clause WHERE.

4. La protection d’une base de données

Selon la référence **[15]** la protection d’une base de données suit les étapes suivantes :

4.1 Connaître son besoin

La sécurité de la base de données commence par une réflexion sur les usages et la population d'utilisateurs accédant à celle-ci, ainsi que sur la manière dont la connexion s'effectue. Est-ce directement par les utilisateurs ou par le biais d'un applicatif (interface Web, progiciel, etc.). Il est indispensable de connaitre la méthode et la nature des accès afin de définir une politique de sécurité adaptée.

"La connexion d'un SGBD avec un progiciel, qui nécessite une méthode d'interconnexion spécifique, peut avoir pour effet d'abaisser le niveau de sécurité. Les équipes sécurité et intégration, dont les missions ne sont pas forcément en accord, doivent souvent trouver un accord".

La sécurité ce n'est cependant pas uniquement la protection contre les attaques. D'autres questions se posent sur la garantie de la traçabilité, l'intégrité, l'audibilité, la confidentialité et la sauvegarde des données. La politique va par conséquent dépendre de ce que l'on souhaite garantir en fonction des besoins identifiés. Il sera ainsi incontournable de redonder les équipements d'interconnexion si la disponibilité est critique.

4.2. Une sécurité en amont

Le déploiement d'une base de données est souvent la brique d'un projet plus global. La sécurité doit donc être pensée pour l'ensemble des éléments, surtout dans le cas d'un applicatif accédant à la base. Celle-ci peut être protégée mais si l'outil utilisé pour s'y connecter est vulnérable, il ouvrira des portes. Un SGBD ne pourra pas faire la différence entre une connexion légitime et une attaque par le biais d'un frontal Web.

4.3. Supervision

Un suivi des indicateurs de la base de données doit être assuré afin de détecter les anomalies, prévenir les interruptions de service et intervenir dans les meilleurs délais. La majorité des SGBD du marché embarquent désormais des systèmes de supervision. Charge ensuite à l'administrateur de base de données (DBA) de concevoir des filtres appropriés pour diagnostiquer toute évolution du mode de fonctionnement de la base.

4.4. Sensibiliser les DBA

L'administrateur doit être sensibilisé aux problématiques de sécurité, aux risques, à la criticité des contenus dont il a la charge et pas seulement à la performance. Un DBA peut avoir à superviser une dizaine de bases sans bénéficier de visibilité sur les données qu'elles hébergent et risquer par conséquent de ne pas avoir les bons réflexes.

4.5. Durcir le socle système

Une base de données repose sur une couche système. Cette dernière ne doit donc pas être négligée et faire l'objet d'un durcissement fort. Une base de données ne sera pas en mesure de se défendre contre une personne détenant des droits administrateur sur l'OS. Ce durcissement comprend l'application d'une politique de gestion des correctifs et du moindre privilège, la limitation des services (réseau et système) et applicatifs, la segmentation des droits ou encore une authentification via des mots de passe fort. Attention au paramétrage des SGBD lors de migration de versions.

4.6. Renforcer la couche BD

Tout comme le système, les correctifs de sécurité doivent être appliqués à la base de données. Pour des exigences de disponibilité, le patch management est cependant complexifié. Il faut veiller en outre au durcissement de l'installation par défaut.

4.7. Gestion des comptes

Les comptes par défaut doivent être verrouillés et les mots de passe remplacés pour respecter les normes de sécurité. La notion de politique du moindre privilège s'applique. C'est-à-dire qu'un utilisateur n'ayant par exemple besoin que de consulter les données ne doit en aucune façon disposer de droits en écriture. De même, la base doit être correctement segmentée pour qu'une habilitation ne concerne qu'un périmètre défini des données.

4.8. Méthodes d'accès

L'entrée sur la base de données doit être autorisée selon des méthodes précises. C'est à ce niveau que le filtrage sera défini. Si la connexion se fait depuis une application Web, alors seule celle-ci et le DBA seront autorisés à accéder. Ce filtrage est toutefois complexifié lors de l'intégration avec un PGI ou de connexion depuis une application en client lourd installée sur de nombreux postes.

Interviennent alors des aspects de gestions des profils et des utilisateurs, d'évolution des droits. Une cartographie des données et des habilitations doit être dressée pour définir les types de populations accédant à la base et les parties de celle-ci qu'ils sont autorisés à consulter.

4.9. Chiffrer les flux de données

Les informations envoyées en réponse à une requête ne doivent pas circuler en clair sur le réseau. Nul besoin de durcir l'accès et l'OS, s'il suffit d'écouter le trafic réseau. Les flux seront par conséquent chiffrés entre la base et les différents composants.

5. Conclusion

La sécurité des accès à une base de données est une préoccupation de tous les instants. Les privilèges doivent être restreints à l'indispensable et être actualisés régulièrement. Quant aux applications web, vulnérables par essence, elles doivent être développées de manière à réduire le risque d'accès frauduleux aux données. Une collaboration étroite entre administrateur de base de données et développeur

web permet de réduire considérablement les risques liés à la sécurité des bases de données. Les éléments de sécurité existant ne sont pas suffisants.

CHAPITREIII

Modélisation et conception

1. Introduction

De nos jours, les bases de données sont des composants incontournables de serveurs Web et d'applications en ligne, qui fournissent du contenu dynamique. Ils permettent d'organiser efficacement la sauvegarde et la lecture des données d'un programme, des données secrètes ou critiques peuvent être stockées dans les bases de données, il est donc important de les protéger efficacement.

Pour lire ou stocker des informations, vous devez vous connecter au serveur de bases de données, envoyer une requête valide, lire le résultat et refermer la connexion. De nos jours, le langage le plus courant pour ce type de communication est le langage SQL (*Structured Query Language*).

Dans le cadre de notre PFE est afin d'implémenter des mécanismes de sécurité, nous proposerons de réaliser une application bancaire qui intégrera un ensemble de contrôle d'accès. Pour faire cela, et dans le cadre de ce chapitre, nous présenteront la modélisation de notre application bancaire qui intègre des contrôles d'accès que nous avons proposés. La modélisation est basée sur le langage UML (Unified Modelisation Language). L'application qui sera présentée dans le chapitre suivant est basée sur deux types d'architecture : 2-tiers pour implémenter les rôles d'Agent et Administrateur ici, ou on a utilisé un modèle à base de rôle et 3-tiers pour gérer le client avec des propositions pour la gestion de la sécurité dans ce cas.

2. Environnement du projet - outils utilisés

Dans ce projet, nous avons utilisé le langage de programmation Java sous NetBeans 6.8, en particulier, les JSP (Java Server Pages) pour les pages Web dynamiques, et aussi EasyPHP pour administrer notre base de données.

2.1. Java sous NetBeans

Java est un langage robuste qui peut être exploité pour développer un large éventail de programmes utilisant une interface utilisateur graphique, pouvant être appliqués en réseau pour se connecter à des bases de données, et offrant d'autres fonctionnalités toutes plus sophistiquées les unes que les autres.

Le ramasse-miettes intégré à Java détecte automatiquement les objets inutilisés pour libérer la mémoire qu'ils occupent. Aussi le Java intègre la gestion des exceptions pour faciliter la mise au point des programmes (détection et localisation des bugs). Concernant la sécurité, Java protège les informations sensibles de l'utilisateur et le système d'exploitation de sa machine en empêchant l'exécution des programmes conçus de façon malintentionnée.
La bibliothèque fournie en standard avec Java couvre de nombreux domaines (gestion de collections, accès aux bases de données, interface utilisateur graphique, accès aux fichiers et au réseau, utilisation d'objets distribués, XML…, sans compter toutes les extensions qui s'intègrent sans difficulté à Java) donc la bibliothèque très riche. **[23]**

Java est enfin nativement doté d'un ensemble complet de primitives de gestion du multitâche simplifiant grandement l'écriture de programmes par exemple devant exécuter des requêtes sur une base de données. L'API JDBC (Java DataBase Connectivity), apparue dès la JDK 1.1, permet de développer des applications capables de se connecter à des serveurs de bases de données (SGBD), par le biais d'un pilote.

Nous avons utilisé un tel pilote (com.mysql.jdbc.Driver) de manière à pouvoir se connecter à un serveur mySQL.

2.2. JSP

JSP est une technologie basée sur Java qui permet aux développeurs de générer dynamiquement du code HTML, XML ou tout autre type de page Web, dans laquelle sont introduits des morceaux de code Java nommés "éléments de script".

2.3. MySQL

L'emploi de bases de données était absolument indispensable afin de sauvegarder de manière efficace l'ensemble des tables, qui à long terme peuvent constituer un bloc très volumineux de données.

Nous avons opté pour MySQL (My Structured Query Language) pour plusieurs raisons parmi les quelles la disponibilité de driver pour se connecter à une base de données à partir de Java. Mais aussi, car il offre un système optimisé de gestion de base de données, et ses commandes sont plutôt faciles d'emploi.

2.4. UML

Pour dessiner des diagrammes UML, il existe plusieurs outils disponibles en Open Source (ArgoUML, StarUML, Poséidon, etc.) ou sous forme de plug-in pour Eclipse ou NetBeans. Pour ce qui suit nous avons utilisé ArgoUML. Pour exprimer nos besoins, nous avons utilisé le formalisme UML des cas d'utilisation et des séquences.

3. Modélisation avec UML

3.1. Diagramme des Classes

Le diagramme des classes identifie la structure des classes d'un système, y compris les propriétés et les méthodes de chaque classe. **[24]**

Le diagramme des classes est le diagramme le plus largement répandu dans les spécifications d'UML car il fait abstraction des aspects temporels et **dynamiques**.

Dans ce qui suit, nous avons présenté d'une façon globale les classes qui sont liées avec elles par des relations, selon certaine condition.

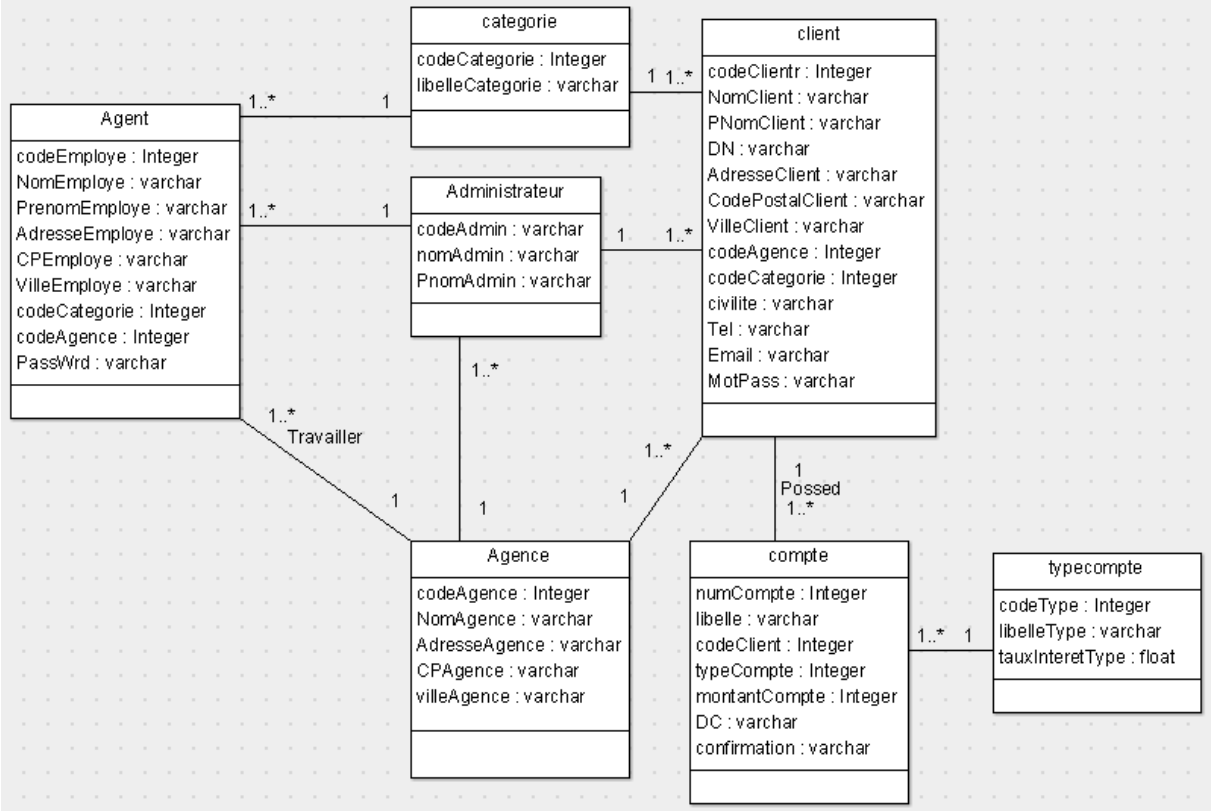


Figure III.1 : Diagramme des classes.

3.2 Diagramme des cas d'utilisation

Les diagrammes des cas d'utilisation identifient les fonctionnalités fournies par le système (cas d'utilisation), les utilisateurs qui interagissent avec le système (acteurs), et les interactions entre ces derniers. Les cas d'utilisation sont utilisés dans la phase d'analyse pour définir les besoins de "haut niveau" du système. **[24]**

Les acteurs humains qui utilisent le système sont les suivants : Agent, Administrateur et Client.

Lorsque le cas d'utilisation est lié à un acteur humain, cela signifie que cet acteur a besoin d'interagir avec le système. Il faut donc lui associer une interface graphique (IHM). Le client utilise le navigateur Web pour accéder au système informatique, alors que les employés et les administrateurs utilisent une application classique déployée sur leurs postes.

Dans le cas où l'acteur serait la base de données il n'y a pas d'interfaces graphiques. Les systèmes communiquent entre eux en échangeant des données.

Les diagrammes des cas d'utilisation se présente comme ci-dessous. Selon chaque acteur.

3.2.1. Premier acteur : Client

Le fonctionnement du premier acteur « Client » est basée sur une architecture de type 3-tiers ou le client utilise une page Web dynamique pour réaliser ses opérations. Un serveur Web qui intègre un conteneur de Servlets vient s'intercaler entre l'IHM et la base de données. Ce modèle présente l'avantage d'intégrer dans cette couche un contrôle d'accès.

La figure suivante, représente le diagramme de cas d'utilisation (Acteur client).

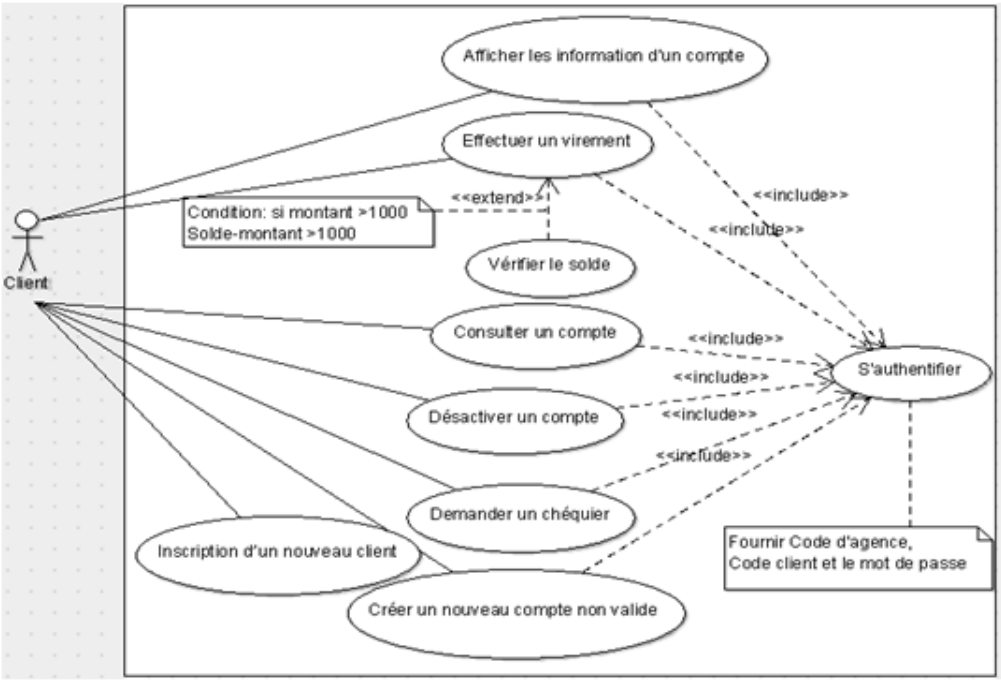


Figure III.2 : Diagramme de cas d'utilisation (Acteur client).

Dans ce qui suit, nous présenterons quelques cas d'utilisations.

- **Cas d'utilisation « Désactiver un compte »**

Le système propose au client de supprimer le compte ou de l'annuler. Dans le cas de la suppression, le système doit attendre une confirmation de l'administrateur avant de le supprimer définitivement.

- **Cas d'utilisation « Créer un nouveau client non valide »**

Permet à un client de s'inscrire dans le système s'il n'existe pas , et pour cela le système lui demande alors de remplir ses coordonnées et ses informations personnels et même les informations concernant son nouveau compte comme libellé, type et solde.

Les éléments caractérisant un client sont les suivants :
• Prénom et nom de famille.
• Date de naissance.
• Numéro de téléphone où l'on peut joindre le client ainsi que son adresse mail.
• Adresse postale.

- **Cas d'utilisation « Consulter le compte »**

Permet à un client de consulter son compte en affichant ses information comme le la date de création, solde, type…etc.

- **Cas d'utilisation « Effectuer un virement »**

Permet d'afficher au client les cordonnées du compte comme le solde, le code du compte et le libellé puis lui donne la main de choisir le compte à créditer qui est soit l'un de ses comptes ou bien un compte d'un autre client et de saisir le montant à virer et le libellé du virement après la validation, le système va vérifier les conditions du virement.

3.2.2 Deuxième acteur : Agent

Le fonctionnement de l'Agent et de l'Administrateur est basé sur une architecture de type 2-tiers vu qu'il n'y a pas d'intermédiaire entre l'IHM et la base de données dans ce cas.

L'idée ici est de faire un contrôle par rôle (Implémenter le **RBAC**) vu la différence en terme de fonctionnalité entre un Agent et un Administrateur. L'agent a pour rôle de gérer le compte des clients qu'il partage avec eux le même code catégorie et le rôle d'administrateur et de gérer la gestion des agences, la gestion des employés, la gestion des clients et la validation des nouveaux clients et des nouveaux comptes.

La figure suivante illustre le diagramme de cas d'utilisation par l'acteur **(Agent).**

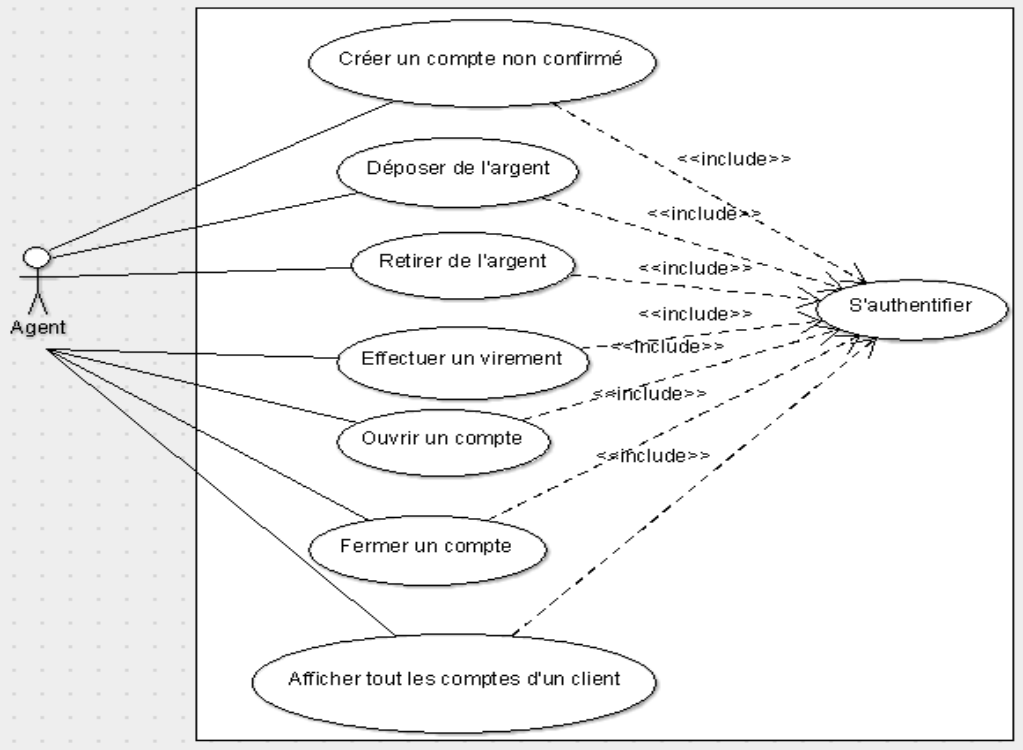


Figure III. 3 : Diagramme de cas d'utilisation (Acteur Agent).

3.2.3. Troisième acteur : Administrateur

La figure suivante illustre le diagramme de cas d'utilisation par l'acteur **(Administrateur).**

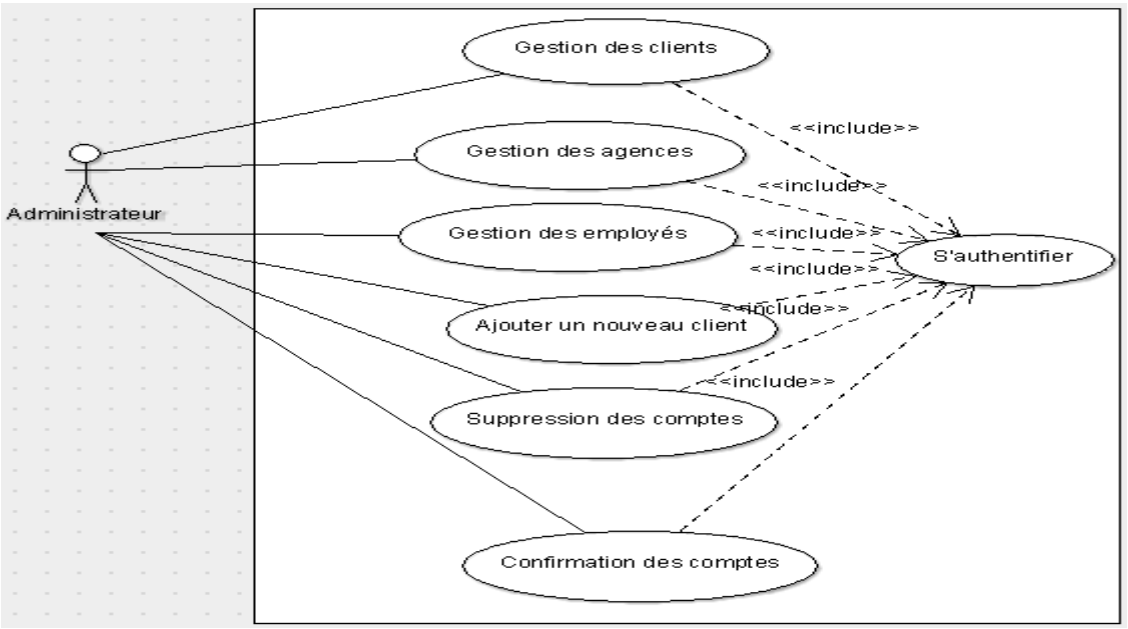


Figure III.4 : Diagramme de cas d'utilisation (Acteur Administrateur).

Dans ce qui suit, nous présenterons quelques cas d'utilisations.

- **Cas d'utilisation « Gestion des clients »**

Permet à un administrateur de créer/modifier/supprimer/rechercher un client et imprimer ses informations.
Notre banque veut pouvoir créer ses clients dans le système à partir des données existantes. Elle souhaite également pouvoir les modifier, les supprimer et les rechercher.

- **Cas d'utilisation « Gestion des agences »**

Permet à un administrateur de modifier/supprimer/rechercher une agence et imprimer les informations, et afficher les clients ou les agents de chaque agence.

- **Cas d'utilisation « Gestion des employés »**

Permet à un administrateur de modifier/supprimer /rechercher des employés et imprimer ses informations.

Chaque cas d'utilisation représenté dans le diagramme précédent doit être complété par un diagramme des séquences.

3.3. Diagramme des séquences

Les diagrammes des séquences documentent les interactions à mettre en œuvre entre les classes pour réaliser un résultat, un cas d'utilisation. UML étant conçu pour la programmation orientée objet, ces communications entre les classes sont reconnues comme des messages. Le diagramme des séquences énumère des objets horizontalement, et le temps verticalement. Il modélise l'exécution des différents messages en fonction du temps. **[24]**

Dans ce qui suit, nous présenterons, les diagrammes des séquences de chaque cas d'utilisation :

3.3.1. Diagramme de séquence de cas d'utilisation « *Authentification* »

Dans ce diagramme, on propose un contrôle d'accès pour authentifier un utilisateur, notre idée est basée sur la désactivation d'un compte après 3 échecs (saisies erronés) d'authentification.

Le digramme est comme suit :

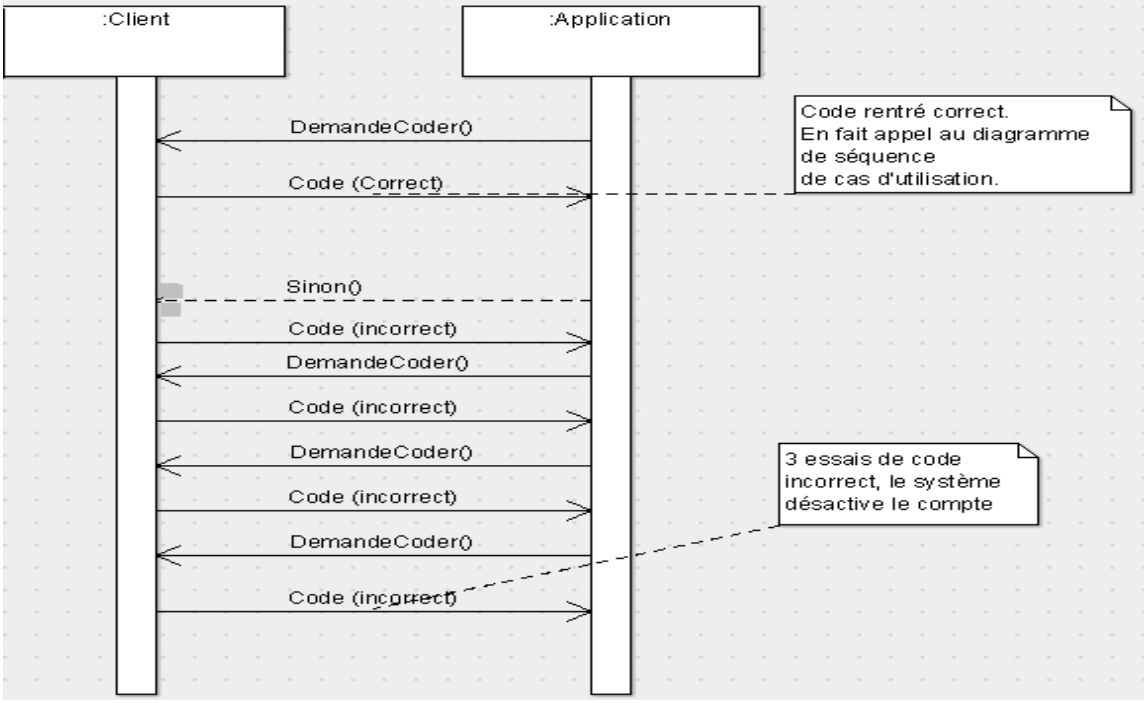


Figure III.5 : Diagramme de séquence de cas d'utilisation « *Authentification* ».

3.3.2. Diagramme de séquence de cas d'utilisation «*Effectuer un virement* »

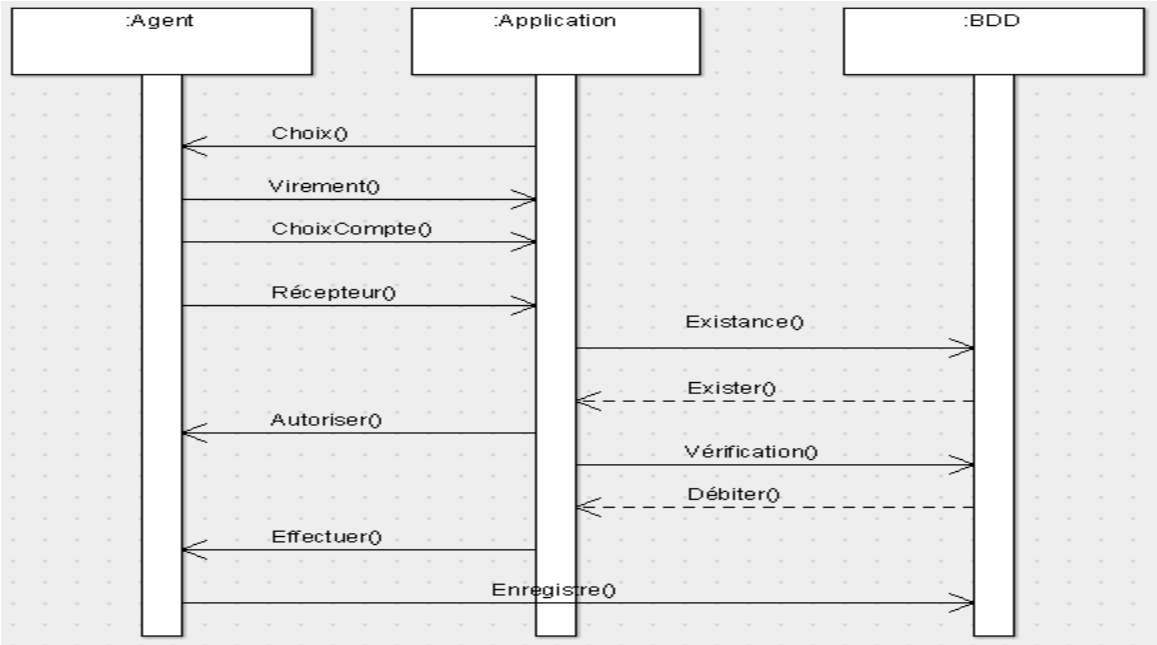


Figure III.6 : Diagramme de séquence de cas d'utilisation «*Effectuer un virement*».

3.3.3. Diagramme de séquence de cas d'utilisation « *Retirer de l'argent* »

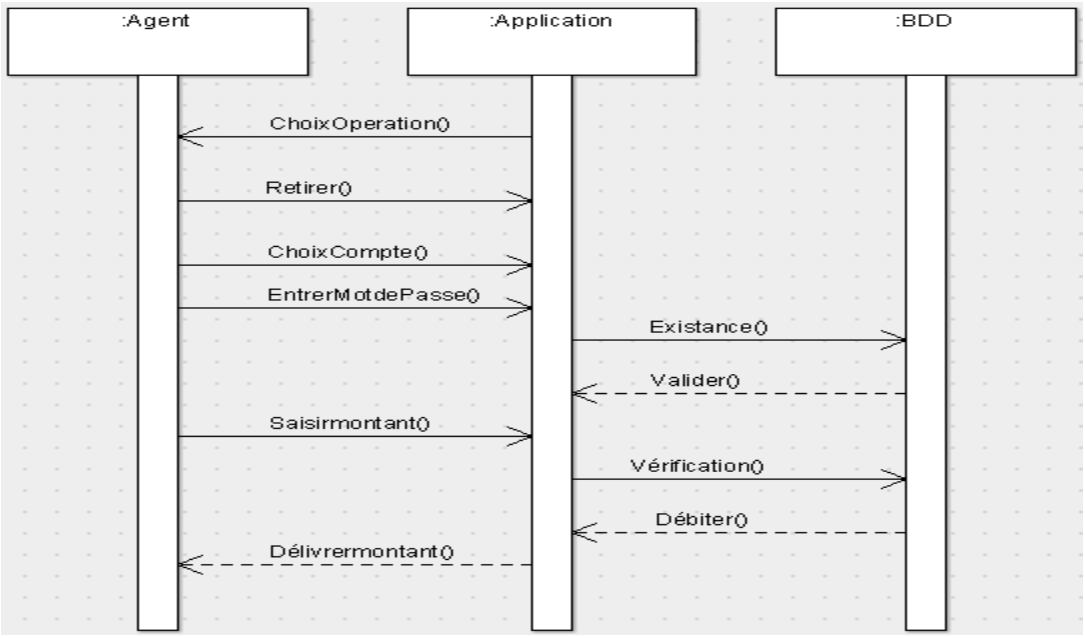


Figure III.7 : Diagramme de séquence de cas d'utilisation «Retirer de l'argent».

4. Conclusion

Dans ce chapitre, nous avons présenté l'étude de cas de l'application gestion bancaire ainsi que les acteurs qui l'utilisent. Le diagramme de cas d'utilisation nous a permis de formaliser les besoins de manière synthétique, puis d'expliciter chaque cas d'utilisation selon le diagramme de séquence. Cette application sera conçue et réalisée dans le chapitre suivant.

CHAPITREIV

Modélisation et conception

1. Introduction

Dans ce chapitre nous allons présenter l'implémentation de notre application et la démarche de chaque fonctionnalité réalisée, nous allons développer tout au long de cette étude une application pour une banque qui permet de faire entrer une autre consultation depuis internet grâce à une application Web utilisant les JSP. Nous allons également réaliser les fonctionnalités Administrateur et Agent de la banque. L'objectif est de proposer aux clients une application de gestion de leurs comptes en ligne en leurs facilitant d'effectuer leurs opérations avec toutes sécurités.

Nous avons opté pour une architecture 3-tiers afin d'implémenter un ensemble de contrôle d'accès concernant le client. Pour le rôle agent et administrateur, nous avons choisi une architecture 2-tiers. Dans ce qui suit, nous allons présenter les différentes fonctionnalités de l'application.

2. Réalisation de l'application

Notre application se compose d'une partie Web concernant l'accès client, et d'une partie classique liée à l'Agent et à l'Administrateur.

2.1. Application Web 3-tiers

Cette partie est réalisée à l'aide de JSPs. L'objectif étant de faire une gestion dynamique de l'ensemble de fonctionnalités liées aux clients. Dans ce qui suit, nous présenterons les différentes vues de l'application.

2.1.1 Page d'accueil

Toutes les pages sont construites de la même manière et utilisent les mêmes éléments, c'est-à-dire :

•Un corps qui affiche les formulaires à remplir ou les informations à afficher.

• Un menu qui se trouve sur la gauche de la page et qui contient un certain nombre d'informations.

La figure suivante représente la page d'accueil de notre application Web.

Figure IV.1 : Page d'accueil.

2.1.2. Accès client

La figure suivante représente la page utilisée pour l'authentification du client.

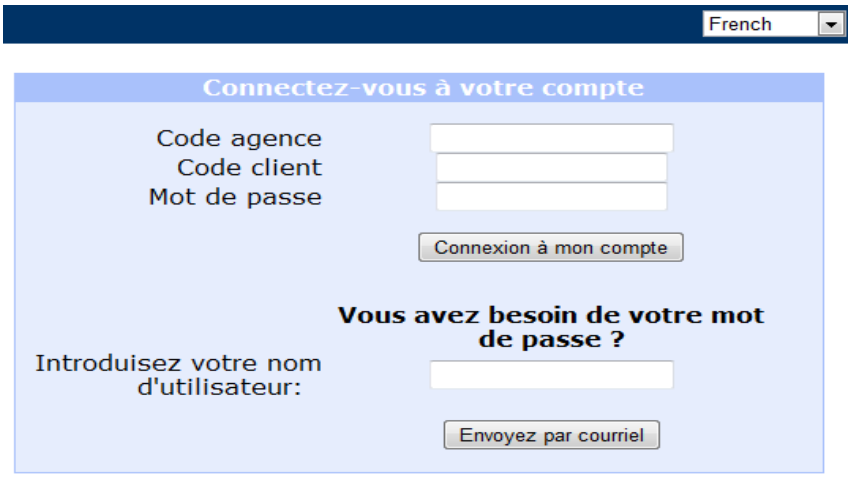


Figure IV.2 : Page d'authentification.

La page JSP vérifie si les codes correspondent aux informations d'un utilisateur existant dans la base de données. Cette page passe le contrôle à la page des opérations si le client a saisi correctement ses codes, sinon le contrôle reste dans la même page ou s'affiche un message d'erreur. Après trois essais erronés, le mot de passe sera désactivé pour des raisons de sécurité. C'est notre proposition de contrôle d'accès à ce niveau pour mieux gérer la sécurité de l'utilisateur. Maintenant et lorsque le client est connecté, l'application le redirige vers une autre JSP qui est représentée dans la figure suivante.

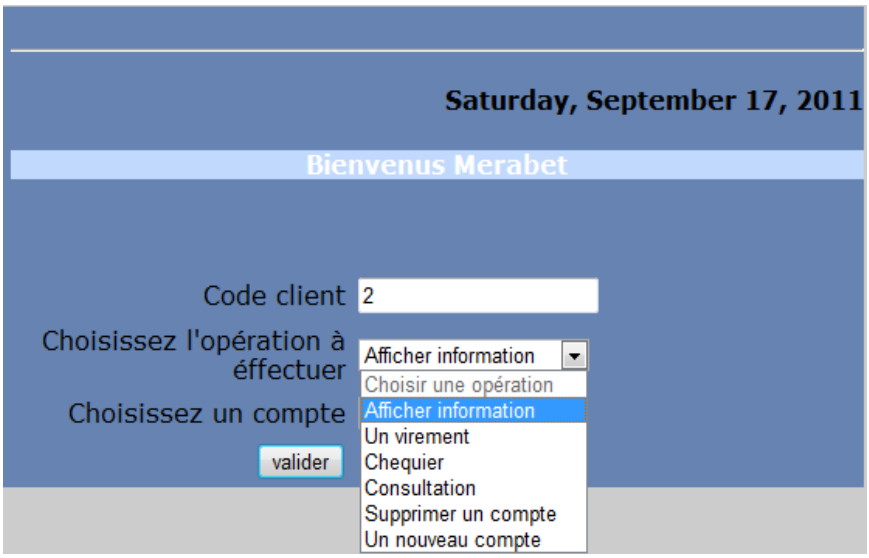


Figure IV.3 : Client authentifié.

Si le client choisit **« Afficher information »** la page suivante qui sera affichée :

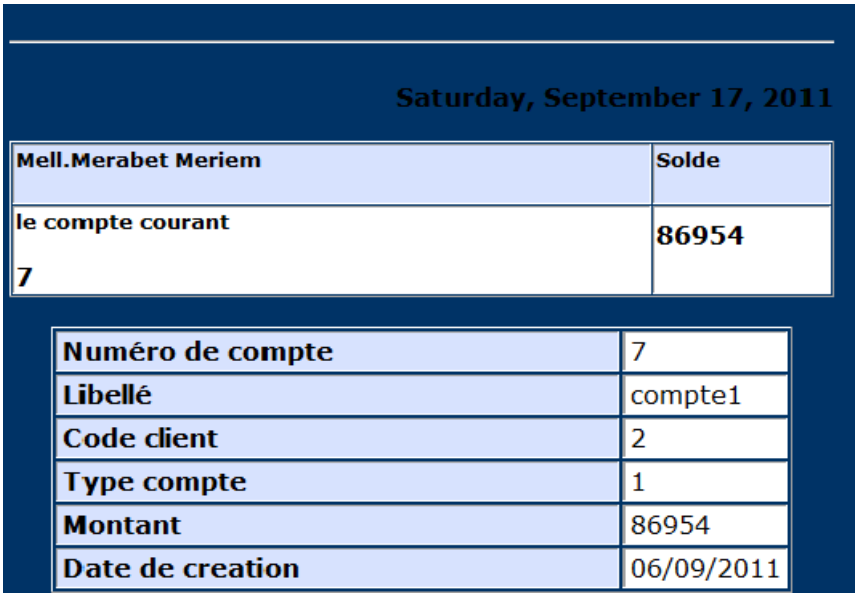
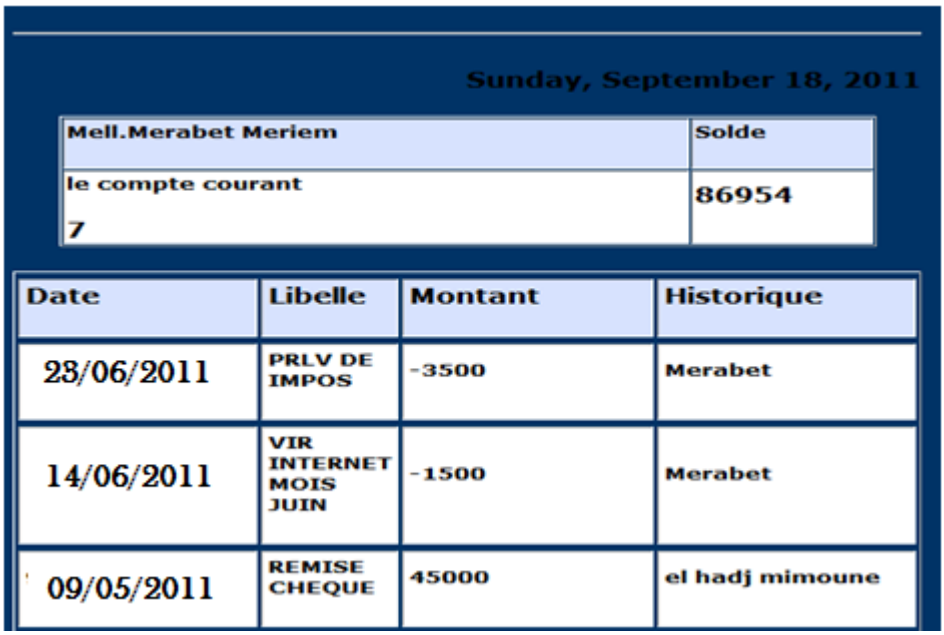


Figure IV.4 : Information d'un compte.

2.1.2.1 L'audit des accès de l'utilisateur

L'audit des accès est nécessaire pour déterminer l'utilisateur responsable de telle action dans la base de données, et comme les utilisateurs accèdent à partir d'un intermédiaire, il est difficile à un système audit de garder la trace et de corréler les activités qui peuvent être sensitives à la sécurité. Pour ce qui nous concerne, nous avons réalisé ce moyen de sécurité quand l'agent ou le client effectue une opération qui va être stocké dans la base de données et on a gardé la date ainsi que les différentes opérations nécessaires (libellé de l'opération, responsable d'opération, ..).

Donc lorsque le client choisit **« Consultation d'un compte »** la figure suivante sera affichée:



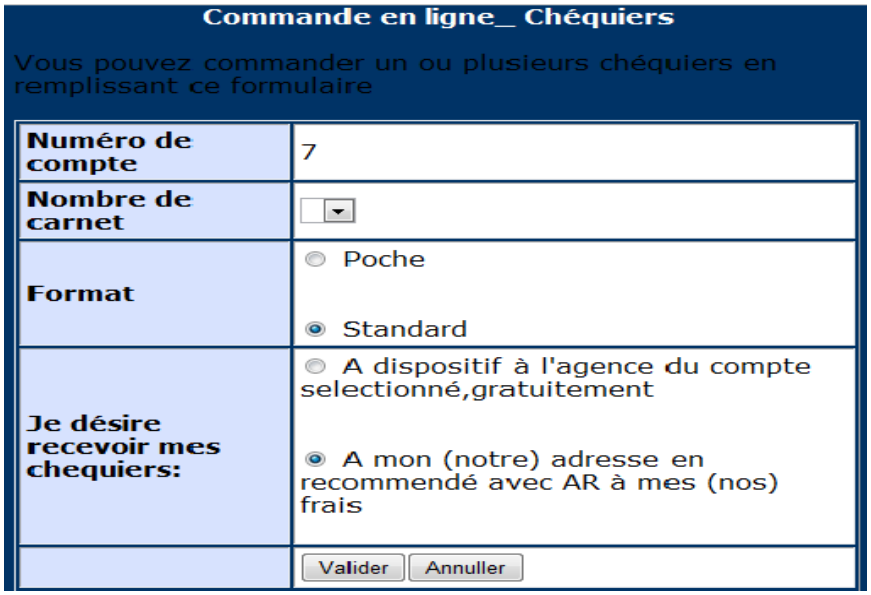
Sunday, September 18, 2011

Mell.Merabet Meriem	Solde
le compte courant 7	86954

Date	Libelle	Montant	Historique
23/06/2011	PRLV DE IMPOS	-3500	Merabet
14/06/2011	VIR INTERNET MOIS JUIN	-1500	Merabet
09/05/2011	REMISE CHEQUE	45000	el hadj mimoune

Figure IV.5 : Page d'historique.

Le client demande un chéquier par ce choix **« Demande chéquier »**, la figure suivante illustre cette opération :



Commande en ligne_ Chéquiers

Vous pouvez commander un ou plusieurs chéquiers en remplissant ce formulaire

Numéro de compte	7
Nombre de carnet	
Format	Poche Standard
Je désire recevoir mes chequiers:	A dispositif à l'agence du compte selectionné,gratuitement A mon (notre) adresse en recommendé avec AR à mes (nos) frais
	Valider Annuller

Figure IV.6 : Commande en ligne un chéquier.

Parmi les choix du client c'est choisir **« Un virement »** et le résultat sera:

Figure IV.7 : Effectuer un virement.

Pour les clients qui ne possèdent pas de compte, ils peuvent formuler leurs demandes via internet et à travers un formulaire bien spécifique. Le nouveau client est ajouté dans la base de données mais il reste toujours en attente jusqu'à la validation de l'administrateur.

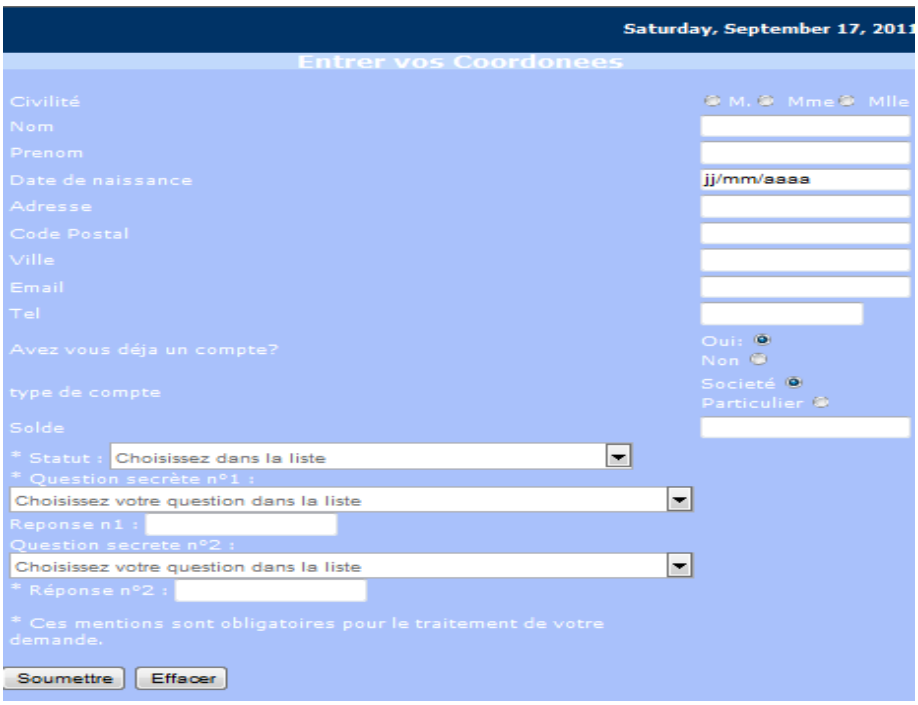


Figure IV.8 : Nouvelle inscription.

Lorsque le client demande de supprimer un compte, il va recevoir un message pour la confirmation de suppression et après la désactivation de ce compte, le message d'information suivant sera affiché :

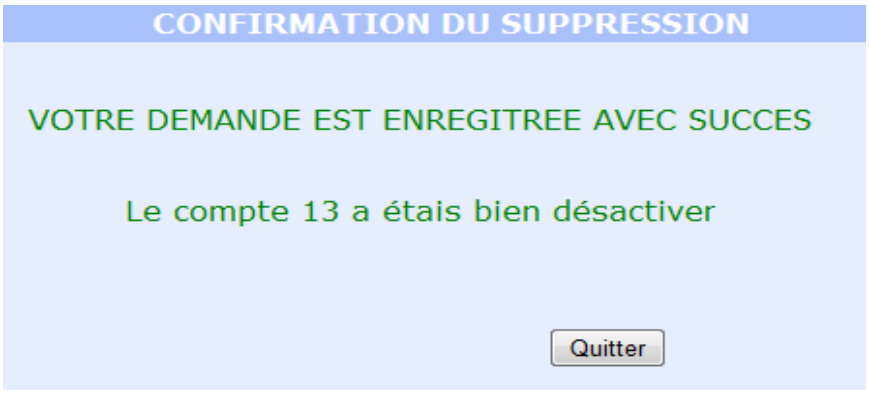


Figure IV.9 : Confirmation de suppression.

Le client peut cliquer sur le choix **trouver une agence**, une JSP affiche la liste des wilayas possédant des agences existantes dans la base de données. Puis une agence peut être sélectionnée parmi celles de cette wilaya.

Le choix de l'agence « Kiffane » de la wilaya de Tlemcen correspond à l'affichage suivant :

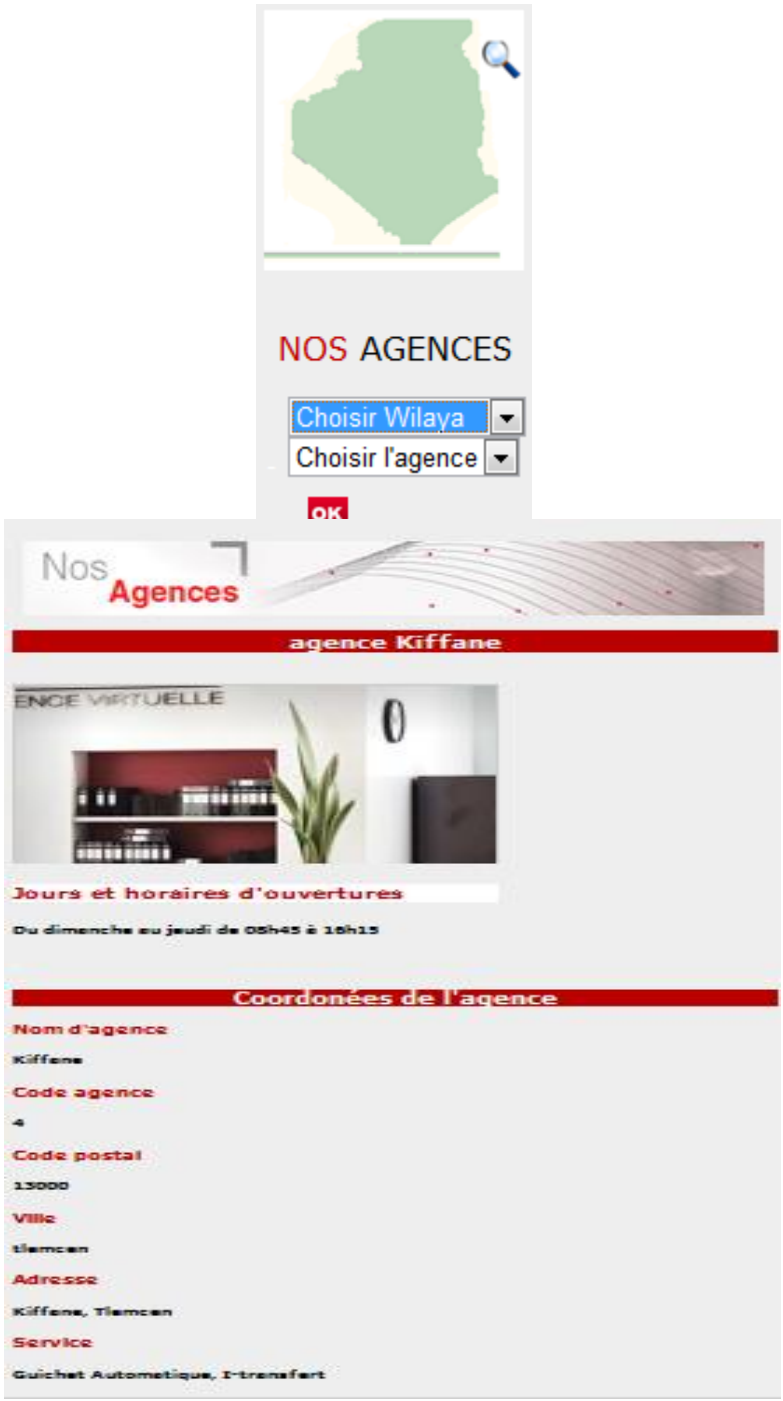


Figure IV.10 : Information sur une agence.

Dans ce qui précède nous avons présente notre première proposition de contrôle d'accès, le chiffrement qui sera détaillé dans la section suivante est transparente pour le client de notre application car c'est l'administrateur qui va crypter les mots de passes.

2.2. Application classique 2-tiers

Dans ce qui suit, nous présenterons notre deuxième partie de l'application qui concerne les fonctionnalités liées à l'Agent et à l'Administrateur. Mais avant tous s'intéressant à cette notion de rôle.

2.2.1 Rôle

Un rôle est le nom d'un groupe, tel que les gestionnaires. Après avoir établi des rôles bien spécifiques, on assigne l'administrateur et l'agent de la banque à des rôles bien défini. Puis, on accorde des autorisations à ces rôles. Chaque utilisateur appartenant à ce rôle hérite des autorisations qu'on a assignées. Les rôles sont par conséquent une façon efficace de gérer des autorisations pour les groupes d'utilisateurs. Le principal objectif des rôles consiste à faciliter la gestion des règles d'accès pour les groupes, tel que les administrateurs ou les agents. Une fois les rôles définis, nous pouvons créer les règles d'accès de notre application.

2.2.2 Accès de l'agent

Un agent dans une banque s'occupe de la relation avec le client. En fonction du profil du client, il va lui proposer des services adaptés à ses besoins et suivre l'évolution de ses opérations bancaires. Il suivra le client dans toutes ses démarches. L'agent a pour rôle de gérer le compte en banque des clients qui partagent avec lui le même code catégorie. Nous avons choisi cette clé afin d'implémenter le modèle par rôle.

La page d'identification est la plus simple du formulaire : c'est un formulaire de saisie avec trois champs, l'un pour choisir le code agence existant dans la BDD, l'autre pour la saisie du nom et le dernier pour la saisie du mot de passe.

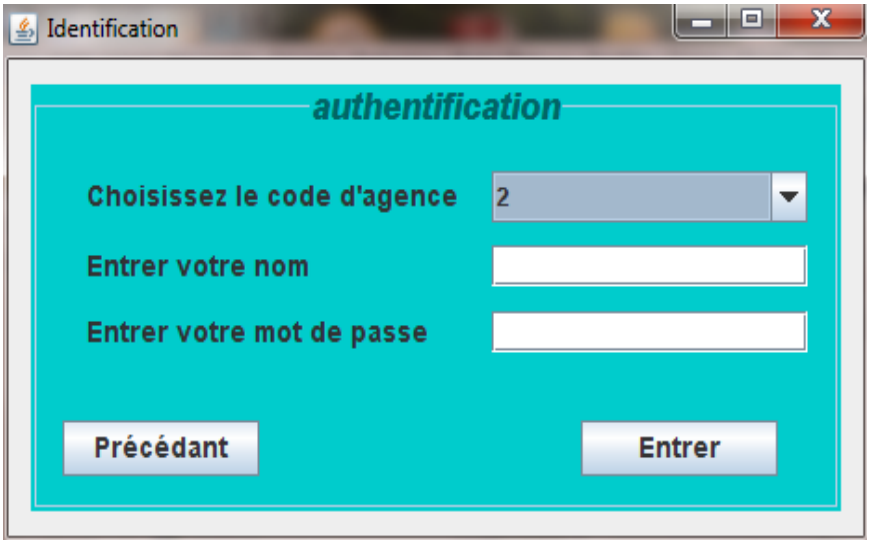


Figure IV.11 : Page d'authentification de l'agent.

Une fois l'agent connecté, la liste des clients qui partagent avec lui le même code catégorie sera affichée.

L'agent choisi le client, puis il va choisir les opérations, qu'il veut effectuer sur les comptes de ce dernier ; la figure suivante affiche la liste des clients.

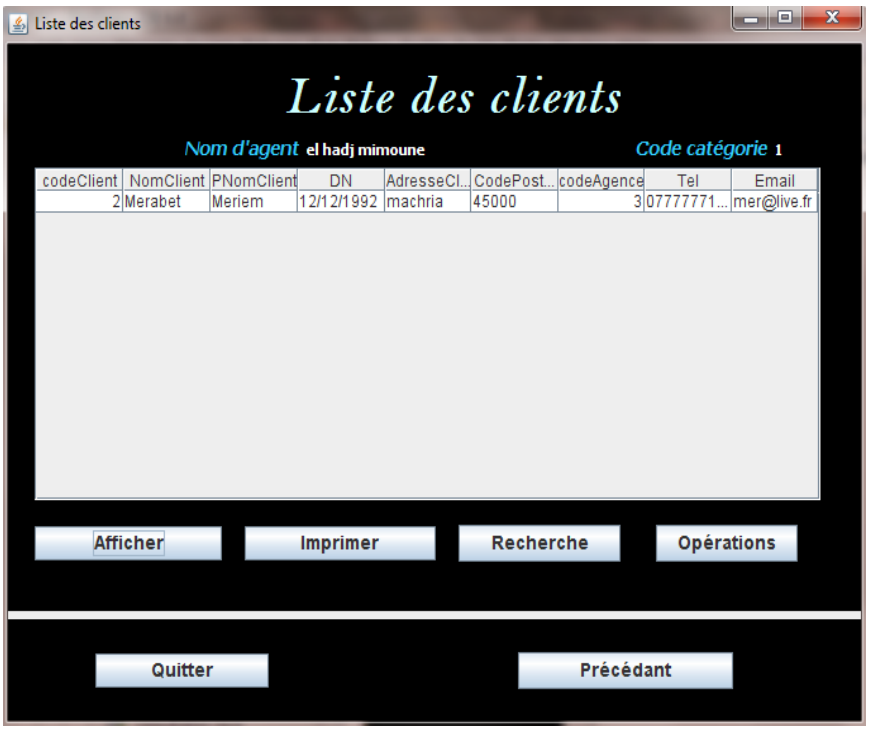


Figure IV.12 : Liste des clients selon la catégorie.

En cliquant sur « **Opérations** » après le choix d'un client l'interface suivante est affichée.

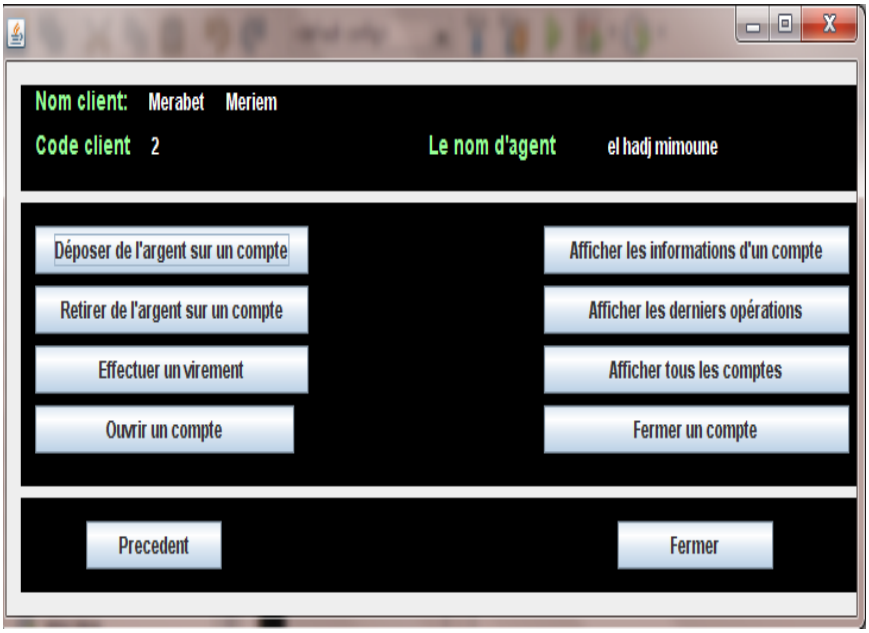


Figure IV.13 : Choisir une opération.

Lorsque l'agent choisit l'opération « **Retirer de l'argent sur un compte »** l'interface suivante sera affichée :

Figure IV.14 : Opération Retirer.

2.2.3 Accès administrateur

L'interface suivante représente l'authentification de l'administrateur.

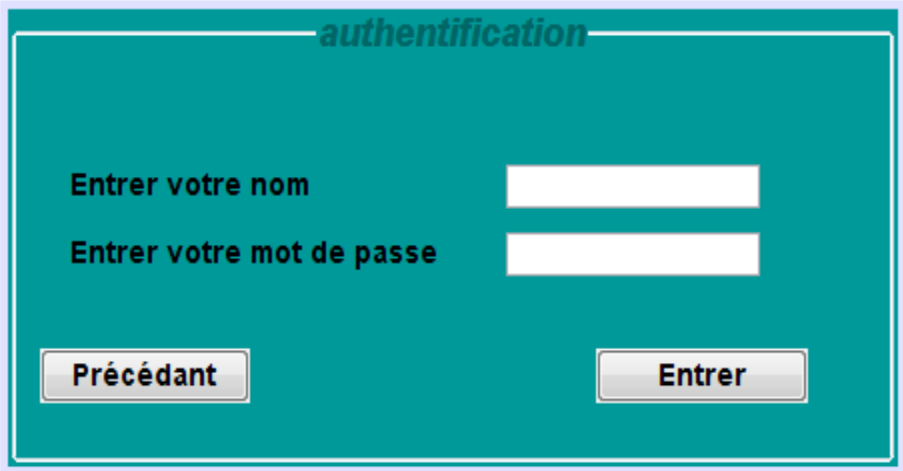


Figure IV.15 : Page d'authentification d'administrateur.

Après l'identification de l'administrateur, la figure suivante permet d'afficher les choix possibles pour ce dernier.

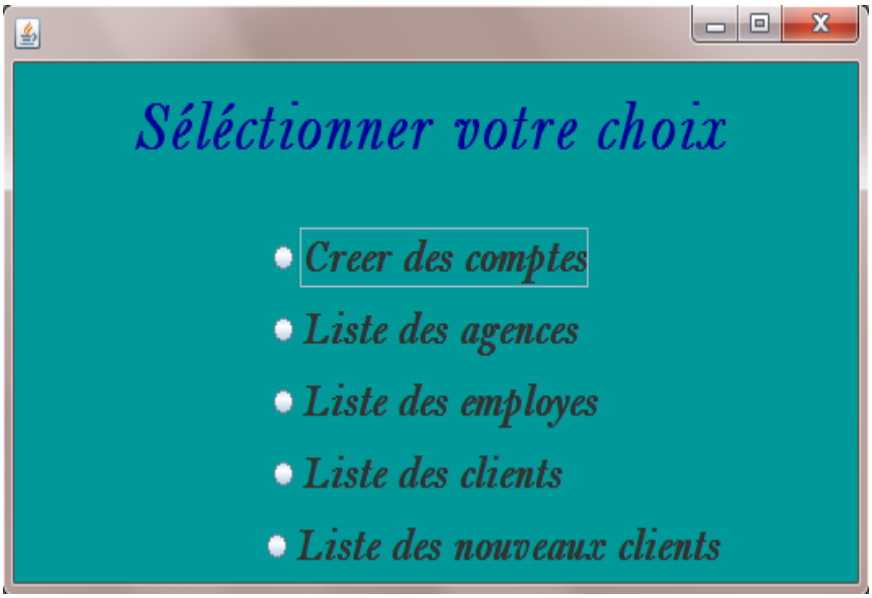


Figure IV. 16: Choix administrateur.

La figure suivante montre les agences disponibles pour cet administrateur.

Figure IV.17 : Liste des agences.

La figure suivante montre la liste des clients pour cet administrateur.

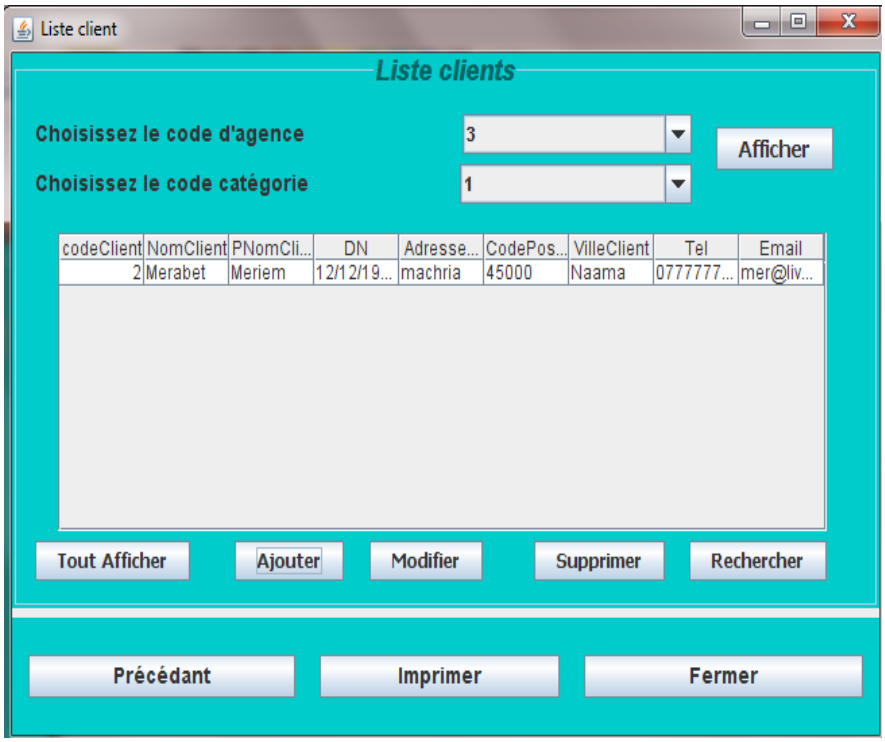


Figure IV.18 : Liste des clients.

Figure IV.19 : Liste des nouveaux comptes en attente de validation.

Pour le moment, nous avons réalisé un contrôle par rôle pour ce qui concerne l'Agent et l'Administrateur. Nous avons également, réalisé un contrôle d'accès pour le client qui consiste à désactiver un compte existant après 3 échecs d'authentification. Dans ce qui suit, nous proposons d'utiliser un algorithme de cryptage afin de coder les mots de passes des utilisateurs stockés dans la base de donnés afin de mieux protéger les informations des clients en cas de piratage de la base de données. Nous avons choisi le code de César pour réaliser cette opération.

2.2.3.1 Définition de code César

En cryptographie, le chiffrement par décalage, aussi connu comme le chiffre de César, est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes (ce qui explique le nom « chiffre de César »). Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une permutation

circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue la clé du chiffrement qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message. Dans le cas de l'alphabet latin, le chiffre de César n'a que 26 clés possibles (y compris la clé nulle, qui ne modifie pas le texte). **[25]**

2.2.3.2 Cryptage de mot de passe de notre application

Nous avons utilisé le chiffrement de césar pour coder les mots de passe des utilisateurs. Cette tache est réalisée par l'administrateur.

Selon **la figure IV.8** qui est représente l'inscription d'un nouveau client, la validation d'un client est en attente du la validation par l'administrateur.

C'est l'administrateur qui doit fournir les mots de passe aux clients. Mais avant cette tache l'administrateur doit chiffrer chaque mot de passe en utilisant le code César.

La figure suivante indique la validation d'un client par un administrateur, il fournit les mots de passe dans le champ « Mot de passe » et le confirmer dans le champ « Confirmation de mot de passe », dans ce cas il a entré la chaîne de caractère « jamila » qui est cryptée en « RIUQTI » dans la case « mot de passe » de la base de données du client avec le code 11 qui est affiché dans la figure IV.21 ci-dessous.

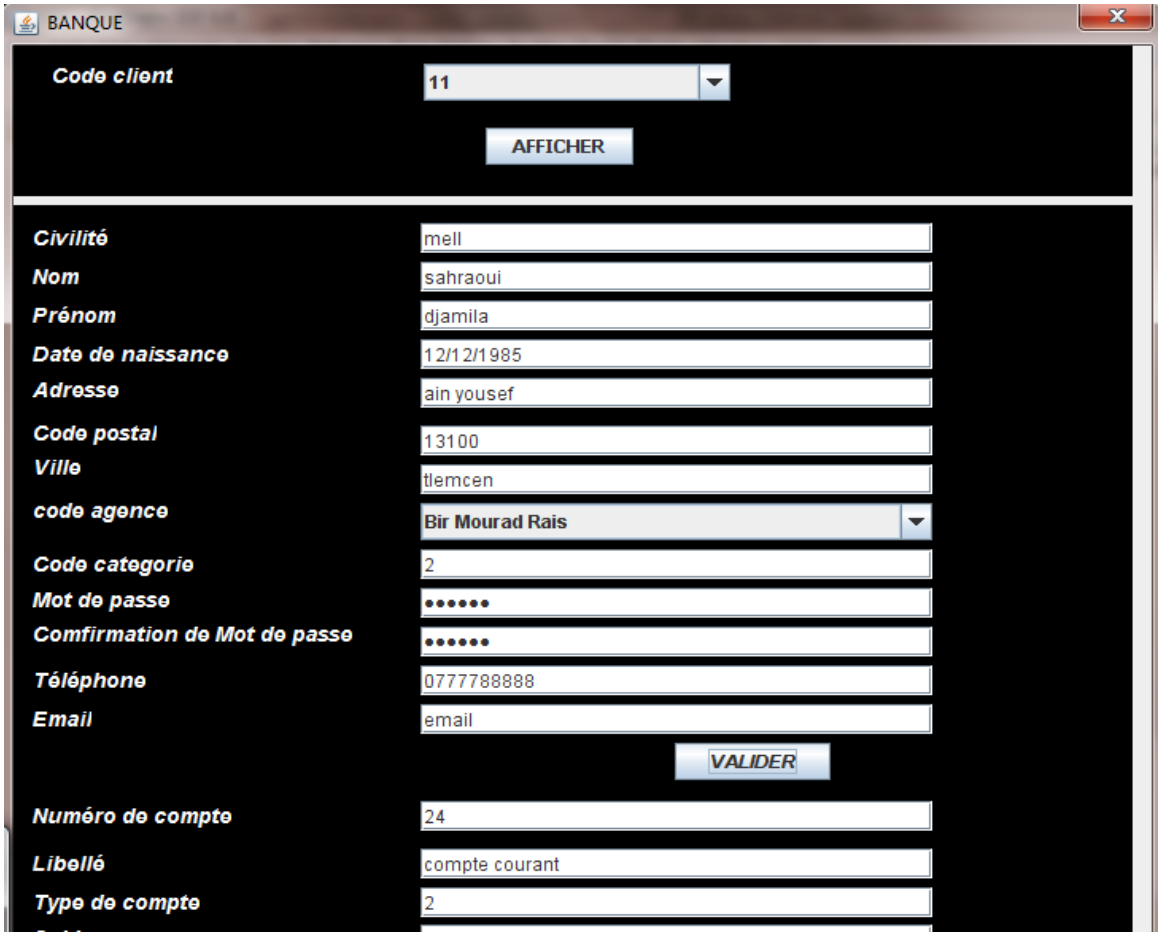


Figure IV.20 : Validation d'un client par un administrateur.

codeClient	NomClient	PNomClient	MotPass
2	Merabet	Meriem	OGT
11	sahraoui	djamila	RIUQTI

Figure IV.21 : Ajout d'un client avec le chiffrement de mot de passe.

3. Conclusion

Ce chapitre a été consacré à la présentation de notre application, cette dernière contient deux partie, la première est une application Web réalisée pour gérer les clients dans un environnement 3-tiers, nous avons proposé dans cette partie un contrôle d'accès sur le nombre de tentatives erronés d'authentification avant de désactiver un compte existant ainsi qu'un mécanisme de chiffrement afin de protéger les informations pertinentes de la base de données. Cette dernière

opération est totalement transparente par le client car elle est réalisée par l'administrateur.

La deuxième partie de notre application est une implémentation 2-tiers afin de gérer les fonctionnalités de l'Agent et de l'Administrateur, à ce stade nous avons proposé un contrôle par rôle, basé sure le modèle par rôle, nous avons également fait l'audit des opérations réaliser par le client.

Conclusion générale

Le travail réalisé dans le cadre de ce PFE nous a ramené à étudier plusieurs domaines dans le cadre de la sécurité informatique et notamment en base de données.
Notre mémoire est articulé autours de deux parti ; la première est théorique, il s'agit de l'état de l'art et elle contient deux chapitres, le premier traite les bases de données en général, le scond parle d'un domaine spécifique dans le cadre des bases de données qui est la sécurité.

La seconde partie de notre mémoire est la partie pratique, dans laquelle nous avons implémenté une application WEB dans un environnement 3-tiers. Nous avons proposé dans ce contexte un contrôle d'accès pour protéger les informations des utilisateurs mais également une méthode de chiffrement basée sur le code de César pour empêcher toute divulgation d'informations liées aux utilisateurs en cas d'attaques visant leurs données.

Nous avons également implémenté une application classique dans un environnement 2-tiers afin de faire un contrôle par rôle concernant l'Agent (employé de la banque) et l'Administrateur.

Pour les deux types d'applications, une conception a été nécessaire pour mieux contourner toute les fonctionnalités de chaque cas d'utilisation, dans ce cadre nous avons utilisé UML.

Au cours de l'élaboration de notre application, nous avons acquis plusieurs connaissances qui s'avèrent bénéfiques dans le cadre de notre formation et qui sont un complément essentiel pour la consolidation de plusieurs données théoriques acquises tout au long de notre formation académique. Nous avons appris en particulier le développement Web dans des environnements n-tiers en utilisant la technologie JSP, avec notamment la présence d'un SGBD puissant comme MySQL.

Nous espérons que ce modeste travail profitera largement à notre université ainsi qu'aux futurs étudiants et ouvrira des perspectives sur des applications regroupant d'autres services.

Nous pensons, qu'il serait intéressant d'implémenter un autre algorithme de chiffrement (RSA ou MD5), et également ajouter d'autres fonctionnalités comme le clavier visuel.

Références bibliographiques

[1] http://www.memoireonline.com/07/10/3701/m_conception-et-realisation-dune-base-de-donnees-pour-la-gestion-de-facturation--loffice-con.html, 04 mai 2011.

[2] Jacques le Maitre, « les bases de données relationnelles et leurs systèmes de gestions », université de Toulon et du Var.

[3] **http://www.telechargercours.com/cours-informatique/acces-aux-bases-de-donnees-odbc-et-jdbc** manipulations de données.

[4] http://div_info.110mb.com Généralités SGBD : Systèmes de Gestion de Base de Données.

[5] http://coursdb.mickey-east.fr/Architecture.aspx

[6] http://*cmsoft.net/utinfo/cours/1/SGBDR/SGBD.doc*, 05 Avril 2011.

[7] Philippe Rigaux, « Cours de bases de données », 13 juin 2001

[8] http://www.memoireonline.com/02/11/4278/m_Conception-et-realisation-dune-base-de-donnees-repartie-sous-oracle--cas-de-lhebergement-d2.html

[9] **http://www-lsr.imag.fr/users/Cyril.Labbe/M2P/Conception_M2P.pdf**

[10] http://www.adproxima.fr

[11] http://www.journaledunet.com, 05 mai 2011.

[12] **http://www.guideinformatique.com/index**

[13] Jacques Le Maitre, « Sécurité des bases de données », Université du Sud Toulon-Var.

[14] http://www.memoireonline.com/07/08/1225/m_mise-en-place-systeme-information-oracle-architecture-trois-tiers10.html , 22 Avril 2011.

[15] http://www.journaldunet.com/solutions/securite/analyses/07/0917-9-etapes-securiser-sgbd.shtml, 23 mai 2011.

[16] Odile PAPINI, **« cours-SSI-P-II-cours-3S »** , université de la méditerranée

[17] Jaccques Le Maitre, **bd-sécurité**, Université de sud Toulon-Var

[18] Nicolase Anciaux, « sécurité et bases de données », source de transparents : Luc bouganim,philipe pucheral, Fridiric, C uppens.

[19] Céline COMA, « Interopérabilité et cohérence de politiques de sécurité pour les systèmes auto-organisants », THÈSE pour obtenir le grade de Docteur de TELECOM Bretagne, soutenue le 22 avril 2009.

[20] Mathieu_Blanc, « Sécurité des systèmes d'exploitation répartis : architecture décentralisée de méta-politique pour l'administration du contrôle d'accès obligatoire.», Thèse A L'UNIVERSITÉ D'ORLÉANS, Soutenue le : 19 décembre 2006 version 161 Mar 2010.

[21] http://cyberzoide.developpez.com/securite/privileges-base-de-donnees/

[22] msdn.microsoft.com/fr-fr/library/ms164596.aspx, 23 mai 2011.

[23] Emmanuel PUYBARET, « Les Cahiers du Programmeur Java1.4 et 5.0 3e édition.», EYROLLES. Page 8.

[24] M. Grimaldi, « Modélisation UML Diagrammes Structurels », Toulon var, février 2007.

[25] Théophile GURLIAT - Benjamin ROULET, « Cryptage d'un texte », Projet R&T, 2006.

Annexes

1. Code de césar

```
{     String mescod="";

      for (int i
=0;i<=mes.length()-1;i++)

 {      int
d=(int)mes.charAt(i)+dec;

         if (d>90) d =d-26;

         mescod
=mescod+(char)(d);
}
         return(mescod);  }
```

```
{
Stringmes="""+getjPasswordField1ActionPerformed().getText()+""";
 String newcode=mes.substring(1, mes.length()-1);
         int dec=2;

         dec=dec%26;

         String mescod= codage (newcode, dec);

         System.out.println(mescod);

// mescod : c'est le mot de passe codé (c'est mescod qu'on doit
insérer dans la BDD)
 st.setString(1,mescod) ;}
```

2. Identification

La page **identif.jsp** pour l'identification d'un client lorsque le client accéder a l'application.

```
<% Class.forName("com.mysql.jdbc.Driver");

try{                    Connection                    con                    =
DriverManager.getConnection("jdbc:mysql://localhost/gestionbancaire", "root", "");
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select codeClient from client");
while(rs.next()){ int cc=rs.getInt("codeClient");
if(B.equals(String.valueOf(cc))){
/*int i=0;
while (i<3)
i++;
if(i>=3){      //    PreparedStatement    st    =    con.prepareStatement("Update    client    set
MotPass='"+"desactiver"+"'where codeClient='"+B+"'");
 // st.executeUpdate();
   }
else{
*/
Statement stm=con.createStatement();
ResultSet      r=stm.executeQuery("select      codeAgence      from      client      where
codeClient='"+B+"'");
while(r.next()){
int ca=r.getInt("codeAgence");
```

```
if(A.equals(String.valueOf(ca))){

Statement stmt1=con.createStatement();

ResultSet rs1=stmt1.executeQuery("select MotPass from client where
codeClient='"+B+"'");
while(rs1.next()){

String mop=rs1.getString("MotPass");

{  int dec=2;
dec=dec%26;
String mesdec= codage (mop, 26-dec);

        // pour récupérer le mot de passe initial il suffit d'appeler la fonction codage
        mais avec le decalage qui //est égal à 26-dec
out.println(mesdec);
if(C.equals(mesdec)){
%> <jsp:forward page="connecter.jsp"/>
<% } else  { %>
<jsp:forward page="identif_1.jsp"/>
<% } } } } } } } catch (SQLException e) { ; } %>
```

3. Virement

Cette partie décrive pour effectuer un virement depuis le compte client vers le compte bénéficiaire selon certain condition.

```
BaseDeDonnee basee = new BaseDeDonnee();
      try {
         PreparedStatement  stmt1 = (PreparedStatement)
basee.connection.prepareStatement("SELECT  montantCompte FROM compte WHERE
numCompte=? ");
          stmt1.setInt(1,Integer.valueOf(jComboBox1.getSelectedItem().toString() ));
        ResultSet rs1 = stmt1.executeQuery();
   int solde1=0;
 while (rs1.next()) {
            solde1=rs1.getInt("montantCompte");  }
   int montant1=solde1-Integer.valueOf(jTextField6.getText());
   if((montant1>=1000)&& (Integer.valueOf(jTextField6.getText())>1000 )){
       PreparedStatement stmt3 = basee.connection.prepareStatement("UPDATE compte
SET montantCompte =?  WHERE numCompte=?");
                    stmt3.setInt(1, montant1);
                    stmt3.setString(2, jComboBox1.getSelectedItem().toString());
                     stmt3.executeUpdate();
       PreparedStatement  stmt = (PreparedStatement)
basee.connection.prepareStatement("SELECT  montantCompte FROM compte WHERE
numCompte=? ");
          stmt.setInt(1,Integer.valueOf(jTextField5.getText()) );
        ResultSet rs = stmt.executeQuery();
   int solde=0;
 while (rs.next()) {
            solde=rs.getInt("montantCompte");   }
   int montant=solde+Integer.valueOf(jTextField6.getText());
```

```
                    PreparedStatement stmt2 =
basee.connection.prepareStatement("UPDATE compte SET montantCompte =? WHERE
numCompte=?");
                    stmt2.setInt(1, montant);
                    stmt2.setInt(2, Integer.valueOf(getjTextField5().getText()));
                    stmt2.executeUpdate();
                      java.util.Date date = new java.util.Date();

      java.text.SimpleDateFormat sdf = new java.text.SimpleDateFormat("yyyy/MM/dd");
String s=sdf.format(date);    }
 else            JOptionPane.showMessageDialog(this, "montant incorrecte", "Erreur ",
JOptionPane.ERROR_MESSAGE);

      } catch (SQLException ex) {}
```

Résumé :

Dans le cadre de PFE, nous avons réalisé une application qui fait appel à des mécanismes de sécurité dans le cadre des bases de données. Notre application contient deux parties : la première est une application Web réalisée pour gérer les clients dans un environnement 3-tiers, nous avons proposé dans cette partie un contrôle d'accès sur le nombre de tentatives erronés d'authentification avant de désactiver un compte existant ainsi qu'un mécanisme de chiffrement afin de protéger les informations pertinentes de la base de données. La deuxième partie de notre application est une implémentation 2-tiers afin de gérer les fonctionnalités de l'Agent et de l'Administrateur, à ce stade nous avons proposé un contrôle par rôle, basé sure le modèle par rôle, nous avons également fait l'audit des opérations réaliser par le client.

Abstract:

This work is a part of research regarding database security mechanisms. Our application has two parts: the first one is a three-tier web application, we have proposed in this part an access control on the number of incorrect authentication attempts before deactivating an existing account and also a data encryption mechanism. The second one is a two-tier application used to manage administrator and bank employee. At this stage we have proposed a role-based access control; we have also performed audits mechanism.

ملخص:

في إطار المشروع قدمنا تطبيق يستخدم آليات الأمن داخل قاعدة البيانات. المشروع يتكون من جزأين: الأول هو تطبيق ويب لإدارة العملاء أجريت في 3-الطبقة , اقترحنا في هذا الجزء التحكم في الوصول على عدد من محاولات تعطيل مصادقة غير صحيحة قبل حساب موجود وآلية التشفير لحماية المعلومات ذات الصلة من قاعدة البيانات.

الجزء الثاني من طلبنا هو تنفيذ 2-الطبقة لإدارة وظائف بنكي و المدير, في هذه المرحلة لدينا دور الرقابة المقترحة, ودور قاعدة آمنة نموذج, ونحن أيضا في جعل مراجعة العمليات التي يقوم بها العميل