République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université d'Ibn Khaldoun – Tiaret

Département Informatique

Thème

Conception et Réalisation d'une base de donnéesrépartie sous PostegreSQL(cas du CNAS)

Pour l'obtention du diplôme de Master II

Spécialité : Réseaux et Télécommunication

Rédigé par : Haddour Hanane

LourdjaneNour El Houda

Dirigé par : Mr.MostefaouiKadda

Année universitaire 2014-2015

Résumé

Une base de données répartie est une collection de données logiquement reliées et physiquement réparties sur plusieurs machines interconnectées par un réseau de communication.

L'objectif de ce travail est d'essayer de résoudre les problèmes de localisation des données et d'exécution distribuée des requêtes posées par la répartition d'une base de données, en travaillant à développer une application simulant la réalisation d'une base de données répartie.

Il s'agit de donner des définitions et des généralités sur las base de donnée réparties, en suite techniques de conception et de gestion des bases de données répartie en implémentant une application pour un système d'affiliation dans l'agence CNAS.

Mots-clés : BD répartie, SGBD réparti, PostgreSQL, Réplication, Fragmentation Horizontale et Verticale, UML.

Sommaire

Liste des figures

IntroductionGénérale:

Le monde de l'informatique évolue très rapidement, alors que son but initial, était d'offrir des services satisfaisants, du point de vue vitesse d'exécution des tâches et obtention de statistiques plus précises. Actuellement, de nouveaux besoins sont apparus, toute organisation automatisée souhaite stocker et échanger ses informations qui sont géographiquement éloignées, ce qui rend la tâche de la collecte et de traitement d'une grande quantité d'informations dispersées très délicate, de ce fait, l'amélioration des systèmes d'informations est devenue une priorité pour les gérants des entreprises.

La solution qui s'impose est de distribuer les données et les organiser dans des bases de données sur différents sites de stockage. L'ensemble de ces sites constitue un système de bases de données réparties offrant la possibilité aux utilisateurs de manipuler les différentes bases via un réseau de manière transparente, comme dans une base de données globale.

A présent, de nouveaux besoins sont apparus, toute organisation automatisée souhaite stocker et échanger ses informations qui sont géographiquement éloignées, ce qui rend la tâche de la collecte et du traitement d'une grande quantité d'informations dispersées très délicate, de ce fait, l'amélioration des systèmes d'informations est devenu une priorité pour les gérants des entreprises. Notre problématique est liée à l’absence de communication et coordination automatique en temps réel entre les différentes agences du CNAS. Plusieurs difficultés sont générées de cette rupture entre les agences qui s’opposent au bon fonctionnement de l’entreprise.

Ce mémoire comprend deux parties : la première partieprésentedes généralités sur les systèmes répartis et les bases de données réparties et les différentes techniques de conception des bases de données réparties et leurs gestions et les principes de la réplication, la fragmentation avec ses types et le système de gestion des bases de données réparties.
Dans la deuxième partie nous verrons la conception et réalisation de notre application

Chapitre1 : Généralités sur les bases de données réparties

Introduction

L'évolution des systèmes d'informations vers le traitement d'une grande masse de données éventuellement réparties, a montré les limites des méthodes classiques de gestion de données basées sur les fichiers. En effet, ces méthodes caractérisées par le dédoublement de données, l'incompatibilité des formats de fichiers et les requêtes figées,pour ne citer que celles-ci, sont devenues inadéquates pour le traitement de grossesquantités de données.

Ainsi apparurent les notions de bases de données et de systèmes de gestion de basesde données qui ont complètement révolutionné les techniques de gestion de données,apportant des concepts qui pallient aux insuffisances des méthodes basées sur les fichiers.

Depuis l'apparition des premiers SGBD1 vers 1962, d'importants résultats théoriques etpratiques ont ponctué l'histoire de la recherche en bases de données. La mise en œuvre deces résultats a permis de faciliter l'administration et la manipulation des données, etd'accroître la productivité des utilisateurs des bases de données (administrateurs,programmeurs d'applications et utilisateurs finals).

Actuellement, l'activité de recherche dans le domaine des bases de données et desSGBD est dans une dynamique extraordinaire, tant l'accroissement des besoinsd'information est important. La tendance actuelle dans le domaine des bases de données estaux SGBD répartis, parallèles et objets. Il faut aussi citer que les techniques de bases dedonnées et des SGBD se sont fait une place dans des domaines aussi nouveaux que pointusde l'informatique moderne, tel que les entrepôts de données, l'exploration informatique dedonnées et le web.

Avant de concevoir une base de données répartie, il est nécessaire de bien comprendre les étapes de conception, car différentes méthodes de conception existent et chacune d'elles nous offre une approche très différente de l'autre.

Dans le cas d'une base de données répartie, la difficulté réside dans le choix des techniques de conception, un mauvais choix pourrait conduire à la création d'un système inefficace.

La conception d'une base de données répartie peut être le résultat de deux approches totalement distinctes, soit d'une part le regroupement d'une multitude de bases de données déjà existantes où d'autre part, que cette dernière soit construite du zéro.

Dans ce chapitre nous allons définir les concepts de bases de données et desystèmes de gestion de bases de données.

I. Système réparti

1. Définition

Un système réparti est un ensemble de processeurs autonomes, reliés par un réseau de communication, qui coopèrent pour assurer la gestion des informations.

Son principe est que les données et les traitements sont répartis sur différents sites interconnectés par un réseau de communication. Ainsi, la défaillance d'un site ne peut entraîner l'indisponibilité totale du système et sa probabilité peut être négligée grâce à la tolérance aux fautes, assurée par la redondance des informations et des traitements. [2]

L'autonomie des sites est préservée par ce genre de système, en permettant à un groupe d'utilisateurs de créer et de gérer leur propre base de données tout en autorisant les accès aux autres utilisateurs via le réseau.

Un système réparti peut sensiblement améliorer les performances des traitements. En effet, avec une localisation des données et une répartition des traitements bien étudiées, la déperdition induite par les communications des données inter-sites peut être compensée par le gain (temps de réponse), issu du parallélisme dans l'exécution des traitements.

La sécurité dans un système réparti vise à garantir la confidentialité, l'intégrité de l'information et le respect des règles d'accès aux services. Les méthodes utilisées reposent d'une part sur un matériel ou un logiciel dédié, et d'autre part, sur des protocoles de sécurité utilisant la cryptographie.

2. Architecture des systèmes répartis

Les systèmes répartis recouvrent diverses architectures depuis les architectures Client / Serveur jusqu'aux architectures totalement réparties. [2]

L'architecture Client/serveur se base sur deux types de processeurs généralement distincts :

- Les serveurs qui offrent un service à des clients, par exemple un serveur base de données ou un serveur imprimante;
- Les clients qui émettent des requêtes aux serveurs pour les besoins d'une application.

Dans l'architecture Client / Serveur la plus simple, la quasi-totalité du SGBD se trouve sur le serveur, les processeurs clients ne disposant que des mécanismes de décodage et de transmission des requêtes vers ce serveur. [2]

Une architecture totalement répartie est une généralisation de l'architecture Client / Serveur : les processeurs sont autonomes dans le sens où ils peuvent disposer d'un SGBD et assurer la pleine gestion d'une base de données locale (BDL). En plus, s'ils ne disposent pas des ressources nécessaires à une application qui leur est soumise, ils déterminent la localisation des données et des traitements qui leur sont nécessaires et établissent une coopération avec les processeurs détenteurs de ces ressources. Cette architecture permet d'éviter la présence du goulot d'étranglement du serveur base de données puisque les données sont réparties voir dupliquées à travers le réseau. [2]

3. Objectifs des systèmes répartis

Au niveau des objectifs des systèmes répartis, il existe quatre points essentiels :

- **Indépendance à la localisation :**Au niveau des SGBDR, l'objectif principal est de permettre l'écriture des programmes d'application sans que l'utilisateur se soucie de la localisation physique des données. Dans ce but, les noms des données ne doivent pas dépendre de leurs localisations.

Les requêtes locales sont similaires aux requêtes exprimées en SQL. Les avantages de la transparence sont de faciliter l'écriture des requêtes pour l'utilisateur et permettre le déplacement des données sans modifier les requêtes. [3]

- **Indépendance à la fragmentation :**Dans un système réparti, une relation est constituée de différents fragments, situés sur des sites différents. La relation de base ne doit pas dépendre de la manière dont les données ont été découpées et doit pouvoir être modifiée sans altérer les programmes.
- **Indépendance aux SGBDs :**Un système réparti ne doit pas être dépendant en aucun cas des différents SGBDs, la relation globale doit être exprimée dans un langage normalisé indépendant des constructeurs.
- **Autonomie des sites :**Vise à garder une administration locale séparée et indépendante pour chaque serveur participant à la base de données répartie afin d'éviter une administration centralisée.

Toute manipulation sur un site (reprise après panne, mises à jour des logiciels) ne doit pas altérer le fonctionnement des autres sites. Bien que chaque base travaille avec les autres,

la gestion des schémas doit donc rester indépendante d'un site à l'autre et chaque base doit conserver son dictionnaire local contenant les schémas locaux. [3]

4. Inconvénients des systèmes répartis

Malgré tous les avantages que les systèmes répartis présentent, cela n'exclut pas l'apparition de certains inconvénients comme la complexité des mécanismes de décomposition de requêtes, la synchronisation des traitements et le maintien de la cohérence due à la répartition de la base de données sur plusieurs sites, ainsi le cout induit par les transmissions des données sur les réseaux locaux.

II. Principe des bases de données réparties

1. Définition d'une base de donnée répartie

Une base de données répartie BDR est une collection de bases de données localisées sur différents sites, généralement distants, mises en relations les unes avec les autres à travers un réseaud'ordinateurs, perçues pour l'utilisateur comme une base de données unique. Elle permet de rassembler des données plus ou moins hétérogènes, disséminées dans un réseau sous forme d'une base de données globale, homogène et intégrée. [4]

2. Système de gestion des bases de données réparties

2.1 Définition

Le SGBD (Système de Gestion des Bases de Données) est l'outil principal de gestion d'une base de données. Il permet d'insérer, de modifier et de rechercher efficacement des données spécifiques dans une grande masse d'informations. C'est une interface entre les utilisateurs et la mémoire de masse. Il facilite ainsi le travail des utilisateurs en leur donnant l'impression que l'information est organisée comme ils le souhaitent

- Un SGBDR repose sur un système réparti qui est constitué d'un ensemble de processeurs autonomes appelés sites (micro-ordinateurs, stations de travail, ... etc.) reliés par un réseau de communication qui leur permet d'échanger des données.
- Un SGBDR suppose en plus que les données soient stockées sur deux sites au moins. Ceux-ci, étant dotés de leur propre SGBD.

- Un SGBDR doit offrir une gestion des priorités, des verrous et de la concurrence d'accès de la même façon qu'un SGBD monolithique. Pour cela, il doit disposer de :- Dictionnaire de données réparties,

- Traitement des requêtes réparties,

- Communication de données inter sites,

- Gestion de la cohérence et de la sécurité. [5]

Le SGBDR assure la décomposition des requêtes distribuées en sous requêtes locales envoyées à chaque site. La décomposition prend en compte la localisation des données pour atteindre une base de données distante :

Pour les mises à jour, le SGBDR doit assurer la gestion des transactions réparties ainsi vérifier les règles d'intégrité multi-bases. En cas des bases de données hétérogènes, le SGBDR doit assurer la traduction des requêtes.

2.2Rôle d'un SGBD

Le logiciel de gestion d'un système de base de données (SGBD) à pour rôle :

- **D'assurer la confidentialité des données :** implémentation d'un mécanisme d'authentification par compte avec un mot de passe, attribution de rôles aux utilisateurs permettant d'ouvrir ou de réduire la surface d'exposition des données ;
- **D'assurer la cohérence des données :** vérifier les connaissances d'unicité (clés primaires) et les contraintes d'intégrité fonctionnelles (s'assurer qu'une clé étrangère référence bien une clé primaire et que la suppression d'une clé primaire ne crée pas d'enregistrement orphelins) ;
- **D'assurer la gestion des incidents :** le système doit s'assurer que l'échec d'une requête ne remet pas en cause l'intégrité des données. Il doit également pouvoir procéder aux reprises sur incident suite à une panne du serveur hébergeant la base de données.

III. Utilisation d'une base de données répartie

Au niveau de la BDR, la transparence est un principe fondamental qui apparaît dans la localisation, le partitionnement et la duplication : [4]

1. Transparence de la localisation

La transparence de la localisation des données sous-entend que ni les applications ni les utilisateurs n'ont besoin de connaître la position réelle des tables auxquelles ils accèdent. Autrement dit, ils ne doivent pas connaître la localisation physique des données.

Les utilisateurs accèdent à la BD soit directement par le schéma conceptuel soit indirectement à travers les vues externes, mais en aucun cas, ils n'ont les moyens pour accéder aux schémas locaux.

2. Transparence de partitionnement

Les utilisateurs n'ont pas à connaître les partitionnements de la base de données. Ils ne doivent pas savoir si telle information est fractionnée et ne doivent donc pas se préoccuper de la réunifier. C'est le système qui gère les partitionnements et les modifie en fonction de ses besoins. Et c'est donc lui qui doit rechercher toutes les partitions et les intégrer en une seule information logique présentée à l'utilisateur.

3. Transparence de la duplication

Enfin, le principe de transparence de la duplication est que les utilisateurs n'ont pas à savoir si plusieurs copies d'une même information sont disponibles. La conséquence directe est que lors de la modification d'une information, c'est le système qui doit se préoccuper de mettre à jour toutes les copies.

IV. La répartition des bases de données

A l'heure actuelle, de nombreuses entreprises ont des annexes partout dans le monde, et vue la complexité des problèmes auxquels elles sont confrontées d'une part et l'évolution des technologies informatiques d'autre part, a mené à penser à une nouvelle architecture qui s'adapte plus à leurs organisations.

1. Buts de la répartition des bases de données

Les objectifs de la répartition de données sont multiples :

Plus de disponibilité et de fiabilité :

Comme les bases de données réparties ont souvent des données qui sont répliquées, alors la fiabilité peut être apportée à plusieurs niveaux, la panne d'un site n'est pas importante pour l'utilisateur. En effet, celui-ci s'adresse de façon transparente à un autre site qui possède

les données requises. Par ailleurs, la fiabilité également garantie au niveau des transactions. Elles peuvent être conduites sur le même site de façon concurrente ou sur plusieurs sites en même temps.

Le système d'exploitation ou le SGBD doit garantir qu'une transaction s'accomplira de façon totalement sûre. Une transaction fait passer la base de données d'un état stable cohérent à un autre état stable cohérent, quelques soient les problèmes du réseau rencontrés ou les accès concurrents aux données. [9]

- **Meilleures performances :**Réduire le trafic sur le réseau est une première possibilité d'accroître les performances. Les gains sont particulièrement appréciables pour deux raisons principales :Une grande partie des requêtes s'effectue localement sur le site possédant les données notamment lorsque les données sont répliquées partiellement ou totalement. Le but de larépartition des données consiste est alors à les rapprocher au plus près de l'endroit oil elles sont généralement accédées.

Répartir la base sur différents sites permet de répartir l'impact sur les processeurs et sur leurs Entrées/Sorties. L'impact sur le système se trouve grandement réduit puisque les sites ne traitent qu'une partie de la base de données globale. [9]

- **Scalability :**Il est plus facile d'améliorer les performances du système de gestion de la base de données, en ajoutant des machines sur le réseau plutôt que de passer d'un grand système à un autre. Cependant, l'accroissement des performances n'est pas linéaire. Ajouter des machines sur le réseau sous-entend augmenter le trafic pour maintenir la cohérence de la base de données. [9]

2. Problèmes à surmonter

- **Coût :**La distribution des données et des traitements entraîne des coûts supplémentaires en termes de communication (trafic réseau), et en gestion des communications comme le hardware et software à installer afin de gérer les communications et la distribution.

La distribution est également coûteuse en matière du personnel utilisé car il faut les payer administrateurs de chaque site. [9]

- **Distribution du contrôle :**La distribution du contrôle crée des problèmes de synchronisation et de coordination dans l'accès aux données. Dans une base de données répartie, on ne se soucie pas de la consistance et l'intégrité d'une seule base de données, mais de plusieurs copies de la base de données.

La gestion des copies doit assurer leur cohérence mutuelle, c'est-à-dire que toutes les copies de données soient identiques. [9]

- **Sécurité :**Un des avantages évident des bases de données centralisées est sans contexte la sécurité apportée aux données, car elle peut facilement être contrôlée dans un site unique. Or, lesbases de données réparties impliquent un réseau dont la sécurité est difficile à maintenir. La sécurité est donc un problème plus complexe dans le cas des bases de données réparties que dans le cas des bases de données centralisées. [9]
- **Gestion distribuée des interblocages :**Le problème de l'inter blocage ' Deadlock ' est le même que celui rencontré dans les systèmes d'exploitation. La compétition entre les utilisateurs pour accéder à une donnée peut entraîner des interblocages. [9]
- **Bases de données hétérogènes :**Quand les bases de données sur différents sites ne sont pas homogènes en terme de modèle de données (relationnel, objet, XML, . . .), il devient nécessaire de fournir un mécanisme de translation entre les différentes bases de données, ce mécanisme de translation exige toujours une forme canonique pour faciliter la translation des données.

V. Architecture des bases de données réparties

1. Relation entre machines

Du point de vue organisationnel, nous distinguons deux types d'architectures :

1.1 Architecture Client / Serveur

Dans cette architecture applicative, les programmes sont répartis en processus clients et serveurs qui communiquent à travers des requêtes et des réponses. Sur la machine cliente, les utilisateurs disposent d'une interface.

Sur les serveurs, la gestion des bases de données est effectuée (analyse, optimisation des requêtes, et répartition). On peut distinguer deux types de clients

· **Client lourd :** l'utilisateur est obligé de se connecter explicitement à tous les serveurs dont il a besoin pour la requête qu'il veut formuler. [9]

· **Client léger :** l'utilisateur ne se connecte qu'à la base de données via un unique serveur. Le SGBDR se charge alors de gérer les différentes connexions que nécessitera la requête de l'utilisateur. Donc, il offre plus de transparence. [9]

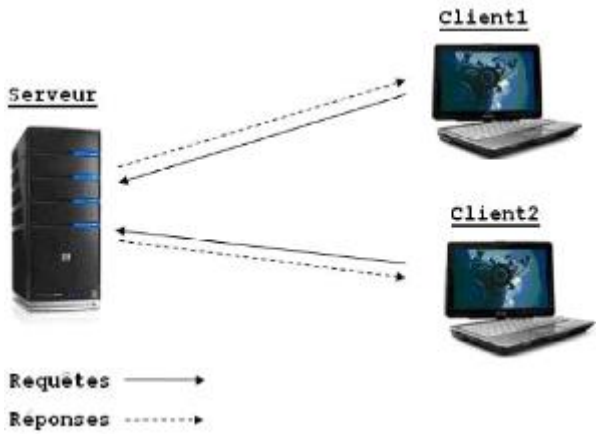


Figure I.1 : Architecture Client / Serveur. [14]

1.2 Architecture Peer To Peer

C'est un type de communication pour lequel toutes les machines ont une importance équivalente. Il n'y a pas de machine qui a une importance hiérarchique par rapport aux autres. Dite aussi, l'architecture totalement répartie.

Chacune de ces architectures possède des avantages et des inconvénients. Le Client / Serveur avec sa structure plus hiérarchique est très sensible aux problèmes de panne des serveurs, bloquant ainsi les clients. En revanche, la prise de décision des serveurs est rapide.

Pour l'architecture Peer-To-Peer, comme les machines sont strictement équivalentes, la panne d'une machine peut rarement rendre le système un peu lent. Mais cette architecture engendre énormément de communication pour toute décision. [9]

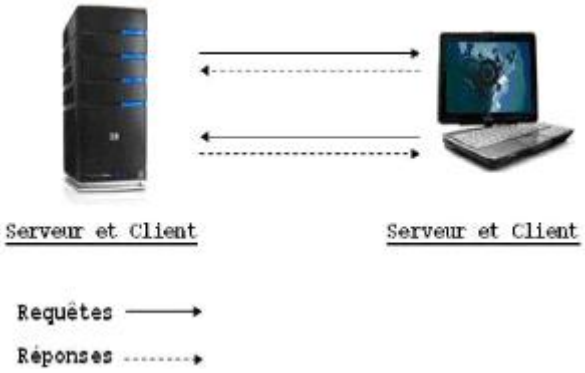


Figure I.2: Architecture Peer To Peer.[14]

1.3. Hétérogénéité

L'hétérogénéité peut apparaître à plusieurs niveaux. En effet, les incompatibilités matérielles ou logicielles au sein d'une entreprise, rendent particulièrement délicate la mise en place d'un SGBD.

L'hétérogénéité peut exister au niveau de la représentation des données, au niveau du langage de requête ou au niveau du modèle de données des différentes bases (BDs relationnelles, BDs objets). [9]

VI. Conception d'une base de données répartie

La définition du schéma de répartition est la partie la plus délicate de la phase de conception d'une BDR, car il n'existe pas de méthode miracle pour trouver la solution optimale. L'administrateur doit donc prendre des décisions dont l'objectif est de minimiser le nombre de transferts entre sites, les temps de transfert, le volume de données transférées, les temps moyens de traitement des requêtes, et le nombre de copies de fragments, ... etc.

1. Méthodes de conception

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'.

1.1 Conception ascendante

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 (LCS) existent et il faut réussir à les unifier dans un schéma conceptuel global (GCS). [1]

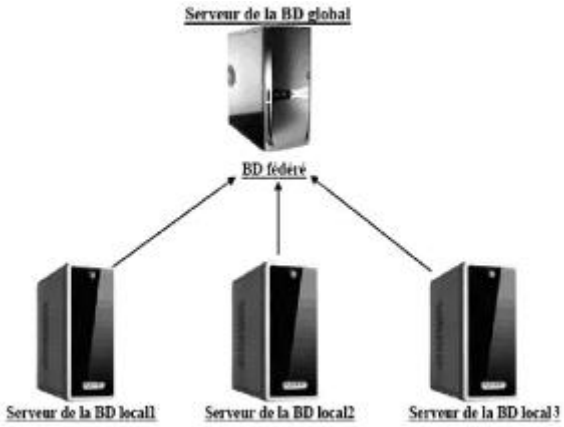


Figure I.3: Architecture de la conception ascendante.[14]

1.2 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. [9]

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

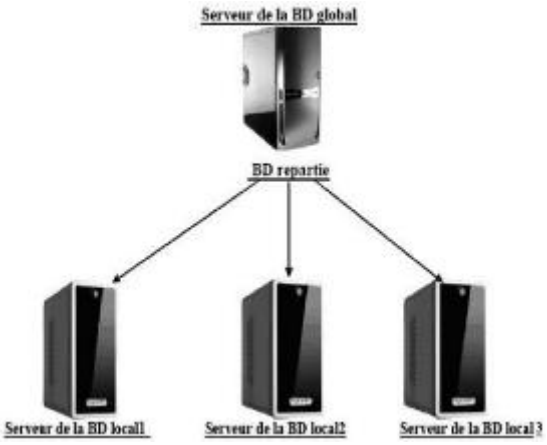


Figure I.4: Architecture de la conception descendante. [14]

2. La fragmentation

2.1 Définition

La fragmentation est le processus de décomposition d'une base de données en un ensemble de sous bases de données. Cette décomposition doit être sans perte d'information. La fragmentation peut être coûteuse s'il existe des applications qui possèdent des besoins opposés. [8]

2.2 Objectif de la fragmentation

Les applications ne travaillent que sur des sous-ensembles des relations. Une distribution complète des relations générerait soit beaucoup de trafic, soit une réplication des données avec tous les problèmes que cela occasionne : problèmes de mises à jour, problèmes de stockage. Il est donc préférable de mieux distribuer ces sous-ensembles.

L'utilisation de petits fragments permet de faire tourner plus de processus simultanément, ce qui entraîne une meilleure utilisation des capacités du réseau d'ordinateurs.

2.3 Les problèmes de la fragmentation

La fragmentation peut être coûteuse s'il existe des applications qui possèdent des besoins opposés. On est en quelque sorte dans le cas d'une exclusion mutuelle qui empêche une fragmentation correcte.

Par ailleurs, la vérification des dépendances sur différents sites peut être une opération très longue.

2.4 Types de fragmentation

La fragmentation horizontale :C'est un découpage d'une table en sous tables par utilisation de prédicats permettant de sélectionner les lignes appartenant à chaque fragment. L'opération de fragmentation est obtenue grâce à la sélection des tuples d'une table selon un ou des critères bien précis et la reconstitution de la relation initiale se fait grâce a l'union (U) des sous-relations. [10]

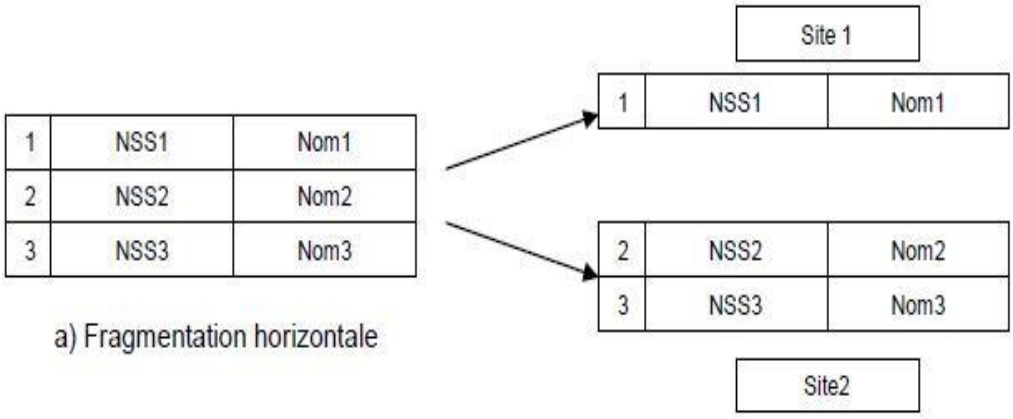


Figure I.5: Exemple de fragmentation horizontale.

La fragmentation verticale : Elle est le découpage d'une table en sous tables par projection permettant de sélectionner les colonnes composant chaque fragment. La relation initiale doit pouvoir être recomposée par la jointure des fragments. [8]

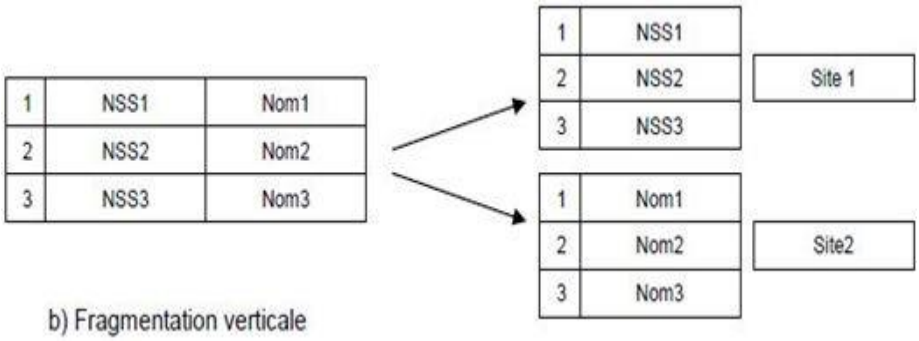


Figure I.6: Exemple de fragmentation verticale.

La fragmentation mixte: Elle résulte de l'application successive d'opérations de fragmentation horizontale et verticale sur une relation globale.

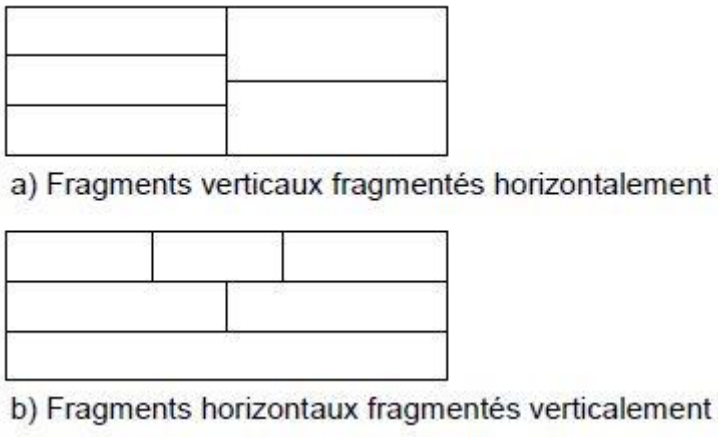


Figure I.7: Exemple de fragmentation vertical.

2.5 Les règles de la fragmentation

Un problème qui se pose pour la fragmentation est comment définir un bon degré de fragmentation. Il existe trois règles pour la fragmentation :

Complétude : pour toute donnée d'une relation globale R, il existe au moins un fragment Ri de la relation R qui possède cette donnée.

Reconstruction : pour toute relation R décomposée en un ensemble de fragments Ri, il existe une opération de reconstruction à définir en fonction de la fragmentation. Pour les fragmentations horizontales, l'opération de reconstruction est une union. Pour les fragmentations verticales c'est la jointure.

Disjonction : une donnée n'est présente que dans un seul fragment, sauf dans le cas de la fragmentation verticale pour la clé primaire qui doit être présente dans l'ensemble des fragments issus d'une relation. [9]

2.6 L'allocation des fragments

Suite à la fragmentation des données, il est nécessaire de les placer sur les différentes machines. Un schéma doit être élaboré afin de déterminer la localisation de chaque fragment et sa position dans le schéma global, c'est ce qu'on appelle l'allocation. [4]

2.7 Le schéma de répartition

Pour fragmenter les requêtes, il est nécessaire de connaître les règles de localisation des données. Lors de l'exécution d'une requête, le SGBDR doit décomposer la requête globale en sous requêtes locales en utilisant le schéma de répartition.

3. Mise à jour de BD réparties

La principale difficulté réside dans le fait qu'une mise à jour dans une relation du schéma global se traduit par plusieurs mises à jour dans différents fragments. Il faut donc identifier les fragments concernés par l'opération de mise à jour, puis décomposer en conséquence l'opération en un ensemble d'opération de mise à jour sur ces fragments.

Insertion : Retrouver le fragment horizontal concerné en utilisant les conditions qui définissent les fragments horizontaux, puis insertion du tuple dans tous les fragments verticaux correspondants.

Suppression : Rechercher le tuple dans les fragments qui sont susceptibles de contenir le tuple concerné, et supprimer les valeurs d'attribut du tuple dans tous les fragments verticaux.

Modification : Rechercher les tuples, les modifier et les déplacer vers les bons fragments si nécessaire.

VII. La réplication

La réplication consiste à copier les informations d'une base de données sur une autre. Elle peut être accompagnée d'une transformation des données sources, voir souvent d'une agrégation. Dans tous les cas, il s'agit d'une redondance d'information.

L'objectif principal de la réplication est de faciliter l'accès aux données en augmentant la disponibilité. Soit parce que les données sont copiées sur différents sites permettant de répartir les requêtes, soit parce qu'un site peut prendre la relève lorsque le serveur principal s'écroule. Une autre application tout aussi importante est l'amélioration des performances des requêtes sur les données locales, et ceci permet d'éviter les transferts de données et d'accroître la résistance aux pannes. [11]

1. Principe

Le principe de la réplication, qui met en jeu au minimum deux SGBDs, est assez simple et se déroule en trois étapes.

1. La base maître reçoit un ordre de mise à jour (INSERT, UPDATE ou DELETE).

2. Les modifications faites sur les données sont détectées et stockées dans un fichier ou une file d'attente en vue de leur propagation.

3. Le processus de réplication prend en charge la propagation des modifications à faire sur une seconde base dite esclave. Il peut bien entendu y avoir plus d'une base esclave.

2. Type de réplication

2.1 Réplication asymétrique

La réplication asymétrique distingue un site maître appelé site primaire, chargé de centraliser les mises à jour. Il est le seul autorisé à mettre à jour les données, et chargé de diffuser les mises à jour aux copies dites secondaires. [8]

Le plus gros problème de la gestion asymétrique est la panne du site primaire. Dans ce cas, il faut choisir un remplaçant si l'on veut continuer les mises à jour. On aboutit alors à une

technique asymétrique mobile dans laquelle le site primaire change dynamiquement. On distingue l'asymétrie synchrone et l'asymétrie asynchrone :

Réplication asymétrique synchrone :Elle utilise un site primaire qui pousse les mises à jour en temps réel vers un ou plusieurs sites secondaires. La table répliquée est immédiatement mise à jour pour chaque modification par utilisation de trigger sur la table maître. [11]

Réplication asymétrique asynchrone : Elle pousse les mises à jour en temps différé via une file persistante. Les mises à jour seront exécutées ultérieurement, à partir d'un déclencheur externe, l'horloge par exemple. [11]

2.2 Réplication symétrique

A l'opposé de la réplication précédente, la réplication symétrique ne privilégie aucune copie c'est-à-dire chaque copie peut être mise à jour à tout instant et assure la diffusion des mises à jour aux autres copies. [8]

Cette technique pose problème de la concurrence d'accès risquant de faire diverger les copies. Une technique globale de résolution de conflits doit être mise en oeuvre. On distingue la symétrie synchrone et la symétrie asynchrone :

Réplication symétrique synchrone : Lors de la réplication symétrique synchrone, il n'y a pas de table maître. L'utilisation de trigger sur chaque table doit différencier une mise à jour client à répercuter d'une mise à jour par réplication. Cette technique nécessite l'utilisation de jeton. [11]

Réplication symétrique asynchrone :Dans ce cas, la mise à jour des tables répliquées est différée. Cette technique risque de provoquer des incohérences de données.

3. Les avantages de la réplication

Les avantages de la réplication sont assez nombreux, selon le type on trouve :

Allégement du trafic réseau en répartissant la charge sur divers sites. Par conséquent, rapidité des accès aux données.

Amélioration des performances des requêtes.

Résistance aux pannes par l'augmentation de la disponibilité des données.

4. Les inconvénients de la réplication

Bien que très avantageuse, la réplication présente tout de même un inconvénient majeur de cohérence mutuelle des données (toutes les copies d'une même BD doivent être identiques). Il peut arriver que les copies de la BD ne soit pas identiques. Pour cela des mécanismes doivent être mis en place pour assurer que les copies qui divergent à un moment donné, finissent toutes par converger vers le mêmeétat.

De plus la propagation (réplication) des mises à jour d'une copie de la BDs vers toutes les autres copies peut avoir parfois un impact négatif sur les performances du système celle-ci ne doit pas altérer de façon excessive le temps de réponse totale.

Il existe plusieurs méthodes de propagation des mises à jour. Le temps nécessaire pour qu'une mise à jour prenne effet sur toutes les répliques (copies) varie d'une méthode à l'autre.

Les copies d'une même BD peuvent ainsi présenter un décalage les unes par rapport au autre, se retard communément appelé temps de latence, définit la période ou une donnée peut être lue alors qu'elle ne reflète les mises à jour antérieures.

L'importance de la durée moyenne du temps de latence varie d'une application à l'autre. Certaines applications pourront ainsi tolérer un temps de latence de l'ordre de quelques minutes, alors que d'autres exigeront un temps de latence de l'ordre de quelques secondes voir quelque millisecondes.

VIII. Gestion des données réparties

1. Mise à jour des données distantes

1.1 Requêtes réparties en lecture

Lors de l'exécution d'une requête en lecture, la base de données répartie va décomposer la requête globale en sous requêtes locales à l'aide des métas donnés de distribution.

1.2 Requêtes réparties en écriture

La mise à jour des données sur une base de données réparties nécessite la validation préalable de chaque site avant la demande du site coordinateur. Ce protocole se nomme 'Validation à deux phases' 2PC et garantit le tout ou rien dans une base de données répartie. [6]

La première phase réalise la préparation de l'écriture des résultats des mises à jour dans la base de données et la centralisation du contrôle.

Par contre la seconde phase 'phase de validation' n'est réalisée qu'en cas de succès de la phase 1, elle intègre les résultats des mises à jour dans la base de données répartie. Le contrôle du système réparti est centralisé sous la direction d'un site appelé coordinateur. Les autres sites sont nommés des participants. [7]

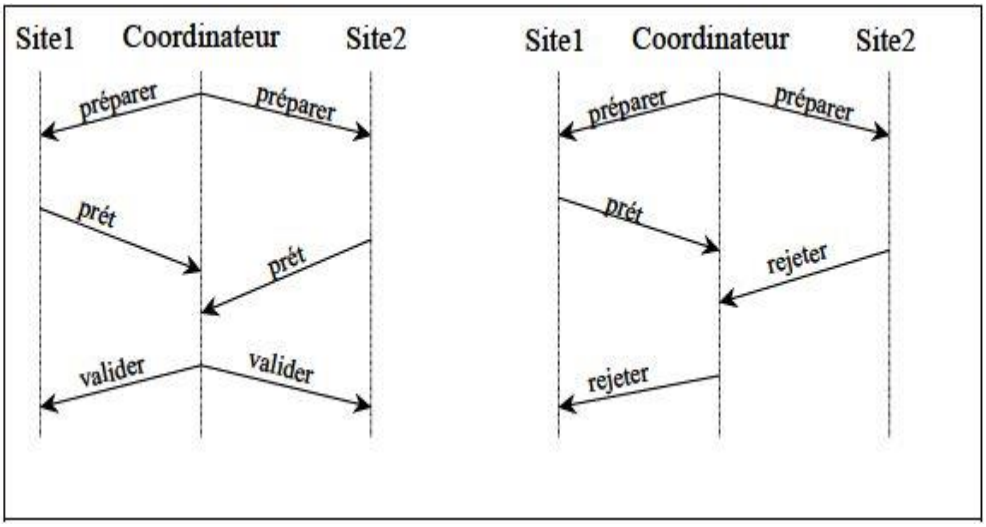


Figure I.8: Protocol de validation à deux phases - 2PC. [14]

2. Contraintes déclaratives

Il est impératif dans une base de données répartie de placer des contraintes déclaratives sur les données qui seront stockées dans le dictionnaire de données.

Dans une base de données répartie, il est nécessaire de dissocier deux types de contraintes :

2.1 Contraintes locales

Les contraintes locales sont des contraintes placées sur un seul site (schéma local). Ces contraintes sont donc stockées dans le dictionnaire de chaque site. [6]

2.2 Contraintes globales

Les contraintes globales doivent être placées sur la relation globale, il n'est pas possible de les matérialiser. Nous pouvons dire qu'il est impossible de créer des contraintes

sur des vues, mais il est plus important de comprendre qu'une contrainte globale doit être placée dans plusieurs dictionnaires. [6]

Le schéma global n'étant pas physiquement implémenté, il n'est pas possible de mettre en place ces contraintes de manière déclarative.

IX. Traitement & Optimisation de Requêtes Réparties

Les règles d'exécution et les méthodes d'optimisation de requêtes définies pour un contexte centralisé sont toujours valables, mais il faut prendre en compte d'une part la fragmentation et la répartition des données sur différents sites et d'autre part le problème du coût des communications entre sites pour transférer les données. Le problème de la fragmentation avec ou sans duplication concerne principalement les mises à jours tandis que le problème des coûts des communications concerne surtout les requêtes.

1. Requêtes sur BD réparties

Comme pour le traitement de requêtes en Bases de données centralisées, on produit l'arbre algébrique de la requête. Chaque feuille de l'arbre représente une relation, et chaque nœud représente une opération algébrique. On enrichit l'arbre avec les informations sur la répartition des données sur les différents sites, en particulier sur le site où chaque opération de la requête doit être exécutée.

La complexité d'une requête dans une base de données répartie est définie en fonction des facteurs suivants :

Entrées/ Sorties sur les disques (disks I/Os), c'est le coût d'accès aux données.

Coût CPU : c'est le coût des traitements de données pour exécuter les opérations algébriques (jointures, sélections ...).

Communication sur le réseau : c'est le temps nécessaire pour échanger un volume de données entre des sites participant à l'exécution d'une requête.

Dans une base de données centralisée, seuls les facteurs E/Ss et CPU déterminent la complexité d'une requête.

Notons que nous faisons la distinction entre le coût total et le temps de réponse global d'une requête:

Coût total : c'est la somme de tous les temps nécessaires à la réalisation d'une requête. Dans ce coût, les temps d'exécution sur les différents sites, les accès aux données et les temps de communication entre les différents sites qui entrent en jeu.

Temps de réponse global : c'est le temps d'exécution d'une requête. Comme certaines opérations peuvent être effectuées en parallèle sur plusieurs sites, le temps de réponse global est généralement inférieur au coût total.

2. Transferts de données

Le temps de transmission d'un message tient compte du temps d'accès et de la durée de la transmission (volume des données / débit de transmission). Le temps d'accès est négligeable sur un réseau local, mais peut atteindre quelques secondes pour des transmissions sur de longues distances ou via satellite. Dans ces conditions, un traitement ensembliste des données s'impose. L'unité de transfert entre sites est une relation ou un fragment, et non une occurrence.

3. Traitement de requêtes réparties

Le but est d'affecter de manière optimale un site d'exécution pour chacune des opérations algébriques de l'arbre. Pour cela, on associe à chacune des feuilles le site sur lequel la relation va être puisée. Lorsqu'une relation est dupliquée, le choix du site de départ est un élément d'optimisation. On cherche ensuite à associer à chaque nœud de l'arbre le site sur lequel l'opération algébrique associée à ce nœud sera exécutée.

Généralement, il faut faire localement tous les traitements qui peuvent y être faits.

Ainsi, lorsque tous les opérandes d'une même opération algébrique sont situées sur le même site, la solution la moins coûteuse pour exécuter cette opération est le plus souvent de l'exécuter sur ce site. Ceci est notamment toujours vrai pour les opérateurs unaires qui font diminuer le volume d'informations (sélection, projection).

Pour diminuer le volume de données transmis d'un site à un autre, il faut limiter les transferts d'information aux seules informations utiles. Pour cela il faut systématiquement rajouter des projections dans l'arbre algébrique pour abandonner les attributs inutiles. Il faut aussi noter que les parties de requêtes indépendantes peuvent être exécutées en parallèle sur des sites différents et donc baisser le temps total d'exécution.

3.1 Optimisation dynamique des requêtes

Après avoir généré un arbre de requête, la stratégie adoptée pour l'exécution est ascendante. C'est à dire que l'affectation de chaque nœud de l'arbre à un site peut être décidée en cours d'exécution en fonction des différents volumes de données intermédiaires obtenus sur

les sites. On part donc des feuilles, où l'on connaît les données (type, volume, statistiques sur la répartition des occurrences dans les domaines de valeurs...) pour remonter au niveau supérieur et prendre une décision sur le site d'affectation de l'opérateur. Et ainsi de suite pour les niveaux supérieurs jusqu'à la racine. Il faut cependant noter que cette stratégie n'est pas optimale si elle est effectuée en aveugle.

D'une manière générale, il faut tenir compte des volumes de données de chaque relation, et de leur composition. Il est en effet parfois intéressant de tenir compte du calcul effectué par un site avant que les autres sites se mettent à travailler.

3.2 Semi-jointure

Les BDR ont abouti à la définition d'un nouvel opérateur, la semi-jointure, qu'il est parfois intéressant d'utiliser. Il s'agit en fait d'une double jointure : le principe est d'effectuer deux petites jointures plutôt qu'une grosse; c'est à dire deux petites transmissions de données plutôt qu'une seule beaucoup plus volumineuse.

La semi-jointure réduit la taille des opérandes des relations. Elle permet de réduire la taille des données à transmettre.

Conclusion

Les bases de données réparties constituent un domaine important pour la gestion des informations stockées sur différents sites.

Dans ce chapitre, nous avons présenté les principes de la répartition des données. Cette répartition peut se faire selon différents scénarios choisis par le concepteur, tout en prenant en compte les restrictions et les obligations de conception.

A travers les différents points développés dans le présent chapitre, nous avons pu constater l'intérêt particulier porté aux systèmes répartis et aux différents problèmes auxquels ce type de solution a pu remédier.

Nous avons pu également, détailler et expliquer l'intérêt des bases de données réparties et les différents avantages offerts par ce type d'approche. Ce type de système est plus difficile à mettre en place et plus compliqué, et que malgré ses nombreux avantages, néanmoins des inconvénients existent, et son inconvénient majeur est la sécurité des données transmises via le réseau de communication

Chapitre 2 : La conception et la réalisation

Introduction

Le « **cycle de vie d'un logiciel** » (en anglais *software lifecycle*), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la **validation** du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la **vérification** du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.

Le cycle de vie du logiciel comprend généralement au minimum les activités suivantes:

Définition des objectifs, consistant à définir la finalité du projet et son inscription dans une stratégie globale.

Analyse des besoins et faisabilité, c'est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes.

Conception générale. Il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel.

Conception détaillée, consistant à définir précisément chaque sous-ensemble du logiciel. **Codage** (Implémentation ou programmation), soit la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.

Tests unitaires, permettant de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.

Intégration, dont l'objectif est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Elle fait l'objet de *tests* d'intégrations consignées dans un document.

Qualification (ou *recette*), c'est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.

Documentation, visant à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.

Mise en production, C'est le déploiement sur site du logiciel

Maintenance, comprenant toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

I. Méthodologies de conception :

1. Etude comparative entre MERISE et UML :

MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse et de réalisation des systèmes d'information qui est élaborée en plusieurs étapes: schéma directeur, étude préalable, étude détaillée et la réalisation.

Alors que **UML** (*UnifiedModeling Langage*), est un langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un système ie: statique, dynamique,....en s'appuyant sur la notion d'orienté objet qui est un véritable atout pour ce langage.

Merise ou UML ?

Les "méthodologues" disent qu'une méthode, pour être opérationnelle, doit avoir 3 composantes:

- ✓ une démarche (les étapes, phases et tâches de mise en œuvre),
- ✓ des formalismes (les modélisations et les techniques de transformation),
- ✓ une organisation et des moyens de mise en œuvre.

Merise s'est attachée, en son temps, à proposer un ensemble "cohérent" sur ces troiscomposantes. Certaines ont vieilli et ont du être réactualisées (la démarche), d'autre "tiennentencore la route" (les modélisations).

UML se positionne exclusivement comme un ensemble de formalismes. Il faut y associer unedémarche et une organisation pour constituer une méthode.

Merise se positionne comme une méthode de conception de SI organisationnel, plus tournéevers la compréhension et la formalisation des besoins du métier que vers la réalisation delogiciel. En sens, Merise se réclame plus de l'ingénierie du SI métier que du génie logiciel.Jamais Merise ne s'est voulu une méthode de développement de logiciel ni de programmation.

UML, de par son origine (la programmation objet) s'affirme comme un ensemble deformalismes pour la conception de logiciel à base de langage objet.

Merise est encore tout à fait valable pour:

- ✓ la modélisation des données en vue de la construction d'une base de donnéesrelationnelle, la modélisation des processus métiers d'un SI automatisé en partie par dulogiciel,

- ✓ la formalisation des besoins utilisateur dans le cadre de cahier des charges utilisateur,en vue de la conception d'un logiciel adapté.

UML est idéal pour :

Concevoir et déployer une architecture logiciel développée dans un langage objet (Java, C++, VB.net). Certes UML, dans sa volonté "unificatrice" a proposé des formalismes.

- ✓ pour modéliser les données (le modèle de classe réduit sans méthodes et stéréotypé en entités), mais avec des lacunes que ne présentait pas l'entité relation de Merise.
- ✓ pour modéliser le fonctionnement métier (le diagramme d'activité et de cas d'utilisation) qui sont des formalismes très anciens qu'avait, en son temps, amélioré Merise...

Après cette étude comparative, il est certes que nous allons adopter UML comme langage demodélisation puisquenousallonsutiliser le concept de l'orienté objet ainsi JAVA commelangage, pour développer l'application la gestion répartie d'affiliation CNAS.

2. Répartition de la base de données :

Notre projet consiste à développer un système d'information dont les données sont intégrées dans un environnement réparti dont lequel nous allons essayer de résoudre les problèmes de localisation suivants :

La redondance des données : Les informations concernant l'affiliésetrouve dans les deux bases de données (Direction de la wilaya&Agence de commune).

Difficultés de gestion : Les traitements entre la Direction de la wilaya et les agences.

Perte de temps : l'échange des informations entre les deux organisations se fait à travers des courriers électroniques, des faxes, …etc.

La répartition de notre base de données se base sur deux techniques la technique de fragmentation et la technique de la réplication, dans la première technique la question qui se pose c'est : Comment Fragmenter? Et qu'elle approche on utilise ? Donc avant de choisir la technique de fragmentation utilisée il faut déterminer les requêtes distantes les plus utilisées puis choisir la technique à partir de ces requêtes.

Requêtes réparties :

Une requête répartie est une requête s'effectuant sur une base de données répartie. Comme une requête normale, elle se base sur les relations de la base et leurs champs, en utilisant l'algèbre relationnel. Mais elle doit tenir compte en plus de certains paramètres essentiels de fragmentation, de localisation afin d'optimiser le temps de réponse global de la requête.

Fragmentation :

Comme nous l'avons dit dans le premier chapitre, la fragmentation constitue la principale phase de la conception d'une base de données répartie. Dans notre étude, la question posée c'est quelles sont les tables et les données qui seront partagées entre la Direction de la wilaya et les agences des communes ? En respectant les contraintes suivantes pour la répartition:

Chaque site doit être indépendant et avoir ses propres données.

La Direction de la wilaya peut accéder aux données des agences des communes instantanément.

Les agences des communes ont le même schéma conceptuel.

La Direction de la wilaya est représentée par le même schéma des agences des communes.

Dans notre cas nous allons appliquer la technique de répartition horizontale où Nous avons décomposé la base de données globale de la Direction de la wilaya après la saisieaffiliés vers les agences correspondants à travers une approche de conception descendante définit comme suit:

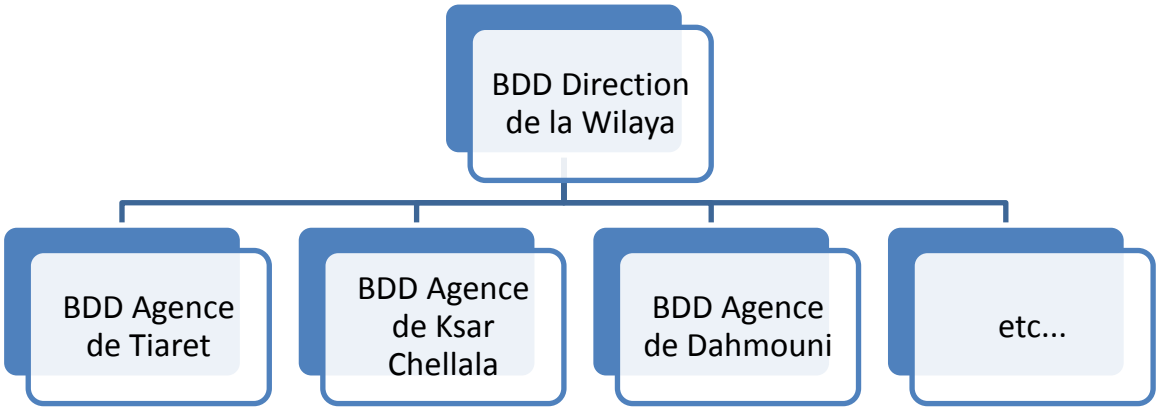


Figure II.1 : La conception descendante.

La deuxième technique consiste à mettre en œuvre une solution de réplication de données. Elle représente de nos jours un moyen efficace pour augmenter la disponibilité des données et contribue aussi à l'amélioration des performances et de la fiabilité des systèmes informatiques.

La réplication repose sur le concept de propagation des mises à jour. Une mise à jour effectuée sur l'une des copies de la base de données doit prendre effet sur toutes les autres copies.

II. Modélisation avec UML :

UML (UnifiedModelingLanguage ou « langage de modélisation unifié »)

Un modèle est une représentation simplifiée d'un problème.UML permet d'exprimer les modèles objets à travers un ensemble de diagrammes. Ces derniers sont des moyens de description des objets ainsi que des liens qui les relient. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, qu'il peut être appliqué à toutes sortes de systèmes et ne se limitant pas au domaine informatique.[16]

1. Spécification du système

Parmi les diagrammes UML largement connus on cite :

*Diagramme de contexte statique :*Le diagramme de contexte statique délimite le domaine d'étude en précisant ce qui est à la charge du système et en identifiant l'environnement extérieur au système étudié avec lequel ce dernier communique.

Diagramme de cas d'utilisation : il permet de recueillir, d'analyser et d'organiser les besoins. Avec lui débute l'étape d'analyse de notre système.

Diagramme de séquence : il permet de représenter des interactions entre objets et acteurs, selon un point de vue temporel avec une chronologie des envois de messages, c'est un type de diagramme d'interaction.

Diagramme de classe : il exprime la structure statique du système en termes de classes, ainsi qu'aux relations entre ces classes.

Dans notre travail, nous allons essayer de travailler avec deux vues (quatre diagrammes)

✓ **Vue statique :**

-Diagramme des classes : définit la structure statique.

✓ **Vue dynamique :**

- Diagramme de cas d'utilisation : décrit les besoins utilisateurs.

- Diagramme de séquence **:** scénarios et flots de messages-

- Diagramme d'activité : décrit le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchainements d'activités.

2. Diagramme de contexte

Une idée globale du système à réaliser dans la figure II.2 .L'utilisateur (administrateur) utilise l'outil d'administration (application Windows) pour connecter avec serveur CNAS et faire les différentes opérations.

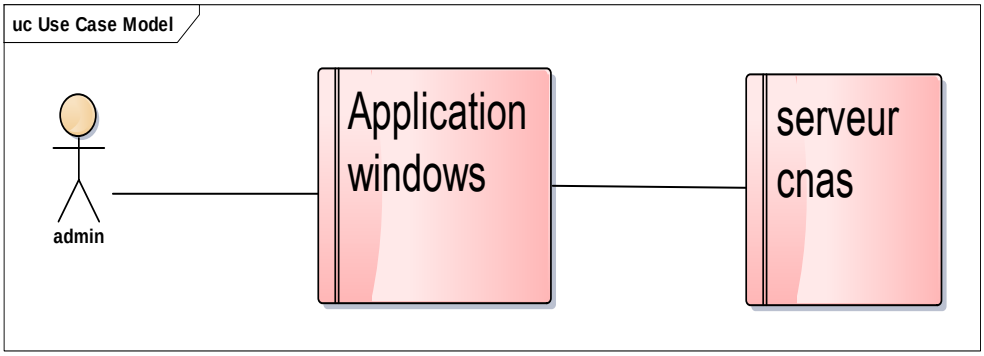


Figure II.2 : Diagramme de contexte

2.1Identification descas d'utilisation :

Les cas d'utilisation constituent une technique qui permet de déterminer les besoins des utilisateurs et de capturer les exigences fonctionnelles d'un système. En d'autres termes, ils décrivent le comportement d'un système du point de vue de ses utilisateurs. Ils décrivent les interactions entre les utilisateurs d'un système et le système lui-même. Ce diagramme permet de décrire les services les plus importants rendus par le système de CNAS.

Dans La figure II.3 nous présentons le diagramme de cas d'utilisation pour la compréhension dufonctionnement du système.[16]

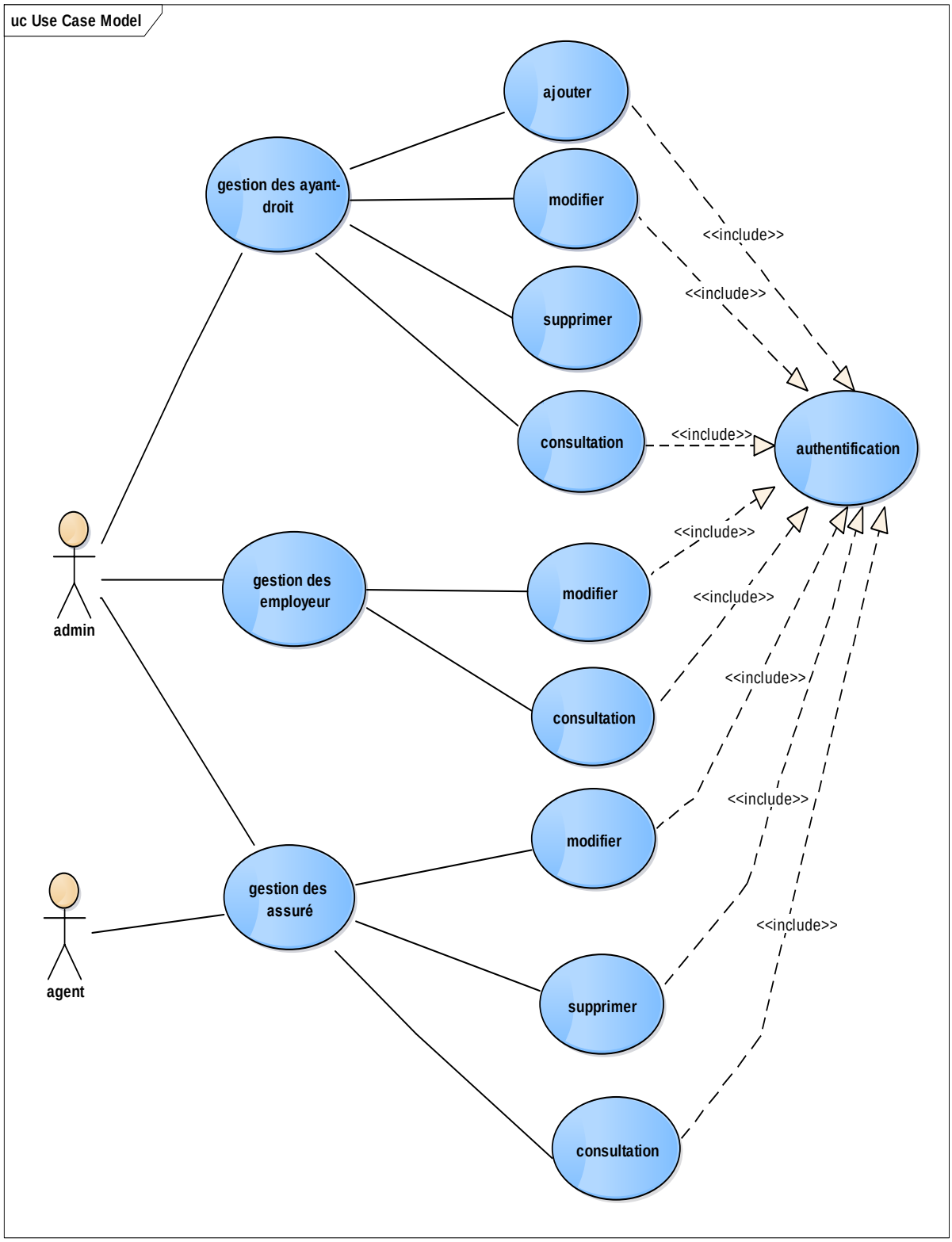


Figure II.3 : Diagramme des cas d'utilisation

La liste des cas d'utilisation déterminés avec les acteurs correspondants est présentée dans le tableau suivant :

	Cas d'utilisation	Acteur
1	Authentification	Administrateur,Agent
2	Gestion des assurés (modifier, supprimer, consulter)	Administrateur, Agent
3	Gestion des ayant-droit (ajouté, modifier, supprimer, consulter)	Administrateur
4	Gestion des employeurs (consulter)	Administrateur

2.2 Description détaillée des cas d'utilisation

Nous proposons dans ce qui suit une description textuelle des principaux cas d' utilisation.

Cas d'utilisation : «authentification »

Description sommaire :

Acteurs : Administrateur, Agent

Objectif : S'authentifier pour pouvoir accéder à ses privilèges au niveau du système.

Description des enchaînements :

Préconditions : L'utilisateur doit avoir un compte.

Postconditions : Ouverture de la session de l'utilisateur.

Scénarionominal : L'utilisateur introduit son login et son mot de passe

Exceptions : Le login ou le mot de passe (voir les deux) est erroné donc l'accès au système sera refusé ; le système demande alors à l'utilisateur d'introduire à nouveau son login et son mot de passe.

Cas d'utilisation : «Gestion des assurés»

Description sommaire :

Acteurs : Administrateur, Agent

Objectif : Gérer les assurés (ajout, modification, suppression, consultation)

Description des enchaînements :

Pré conditions : L'administrateur ou l'agent doit s'authentifier.

Post conditions : Mise à jour de la base de données.

Scénario nominal: Saisie des informations relatives au nouvelassuré, en cas de création,

Dans le cas de modification, l'administrateur ou l'agent choisit l'assuré à modifier, puis effectue les modifications souhaitées,

Dans le cas de suppression, l'administrateur ou l'agent choisit l'assuré à supprimer, alors le système affiche une fenêtre pour confirmer la suppression.

Dans le cas de consultation, l'administrateur ou l'agent choisit l'assuré à consulter, alors le système affiche une fenêtre pour confirmer la consultation.

Cas d'utilisation : «Gestion des ayant-droit»

Description sommaire :

Acteurs : Administrateur

Objectif : Gérer les ayant-droit (ajout, modification, suppression, consultation)

Description des enchaînements :

Pré conditions : L'administrateur doit s'authentifier.

Post conditions : Mise à jour de la base de données.

Scénario nominal : Saisie des informations relatives au nouvelayant-droit, en cas de création,

Dans le cas de modification, l'administrateur choisit ayant-droit à modifier, puis effectue les modifications souhaitées,

Dans le cas de suppression, l'administrateur choisit ayant-droit à supprimer, alors le système affiche une fenêtre pour confirmer la suppression.

Dans le cas de consultation, l'administrateur choisit ayant-droit à consulter, alors le système affiche une fenêtre pour confirmer la consultation.

Cas d'utilisation : «Gestion des employeurs»

Description sommaire :

Acteurs : Administrateur

Objectif : Gérer les employeurs (consultation)

Description des enchaînements :

Pré conditions : L'administrateur doit s'authentifier.

Post conditions : Consulter la base de données.

Scénario nominal :L'administrateur choisit les employeurs à consulter, alors le système affiche une fenêtre pour confirmer la consultation.

3. Diagrammes de séquence

Les diagrammes de séquences permettent de représenter les interactions entre objets selon un point de vue temporel. L'accent est mis sur la chronologie des envois de messages.

Dans cette partie, nous allons présenter différents diagrammes de séquence comme suit :[16]

3.1 Diagramme de séquence (authentification):

La phase d'authentification doit précéder toute opération dans le system de CNAS voulue, à cette étape l'administrateur doit saisir le code d'identification et le password d'une manière correcte.

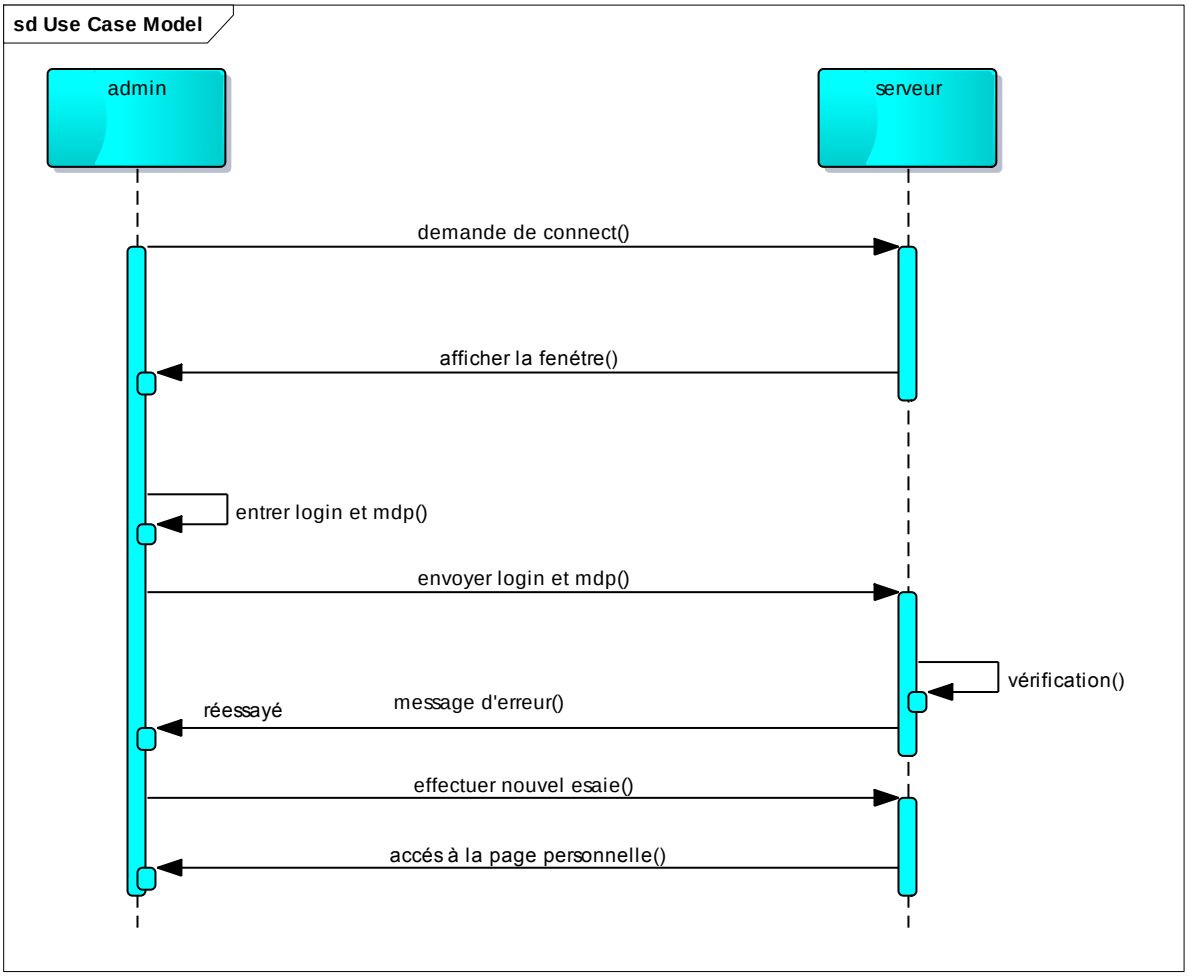


Figure II.4: Diagramme de séquence d'authentification.

3.2 Diagramme de séquence (insertion):

Cette phase est précédée par l'étape d'authentification déjà expliquée,

L'administrateur demande d'ajoute d'un nouvel assuré.

-L'outil affiche la page d'ajoutd'unnouvel assuré.

-L'administrateur saisie les informations du nouvel assuré et l'enregistré.

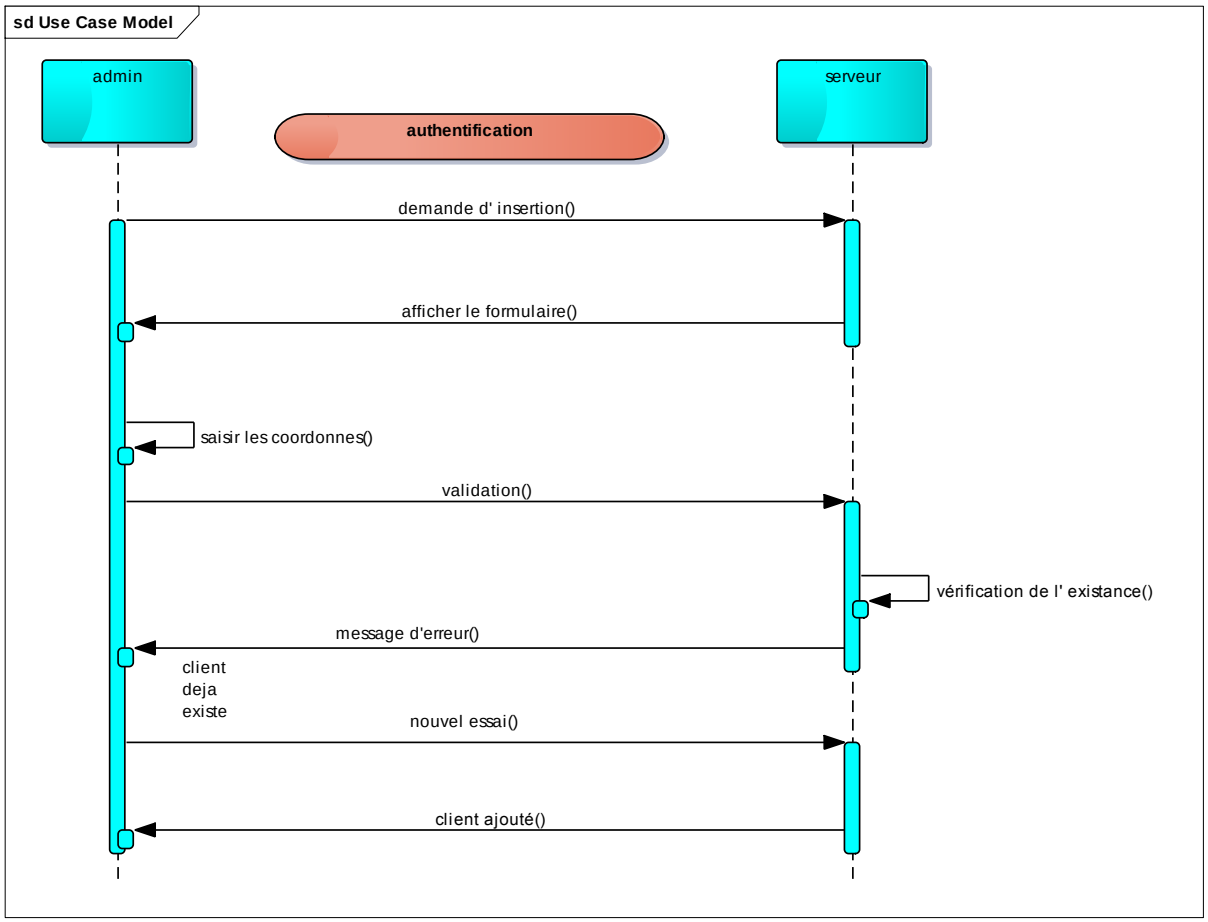


Figure II.5: Diagramme de séquence d'insertion.

3.3Diagramme de séquence (modification) :

-l'authentification

-L'administrateur demande de modifier un assuré.

-L'outil affiche la page de modification d'un assuré.

-L'administrateur saisie les informations.

-L'outil affiche le formulaire de modification.

-L'administrateur saisie les nouvelles informations et les enregistre.

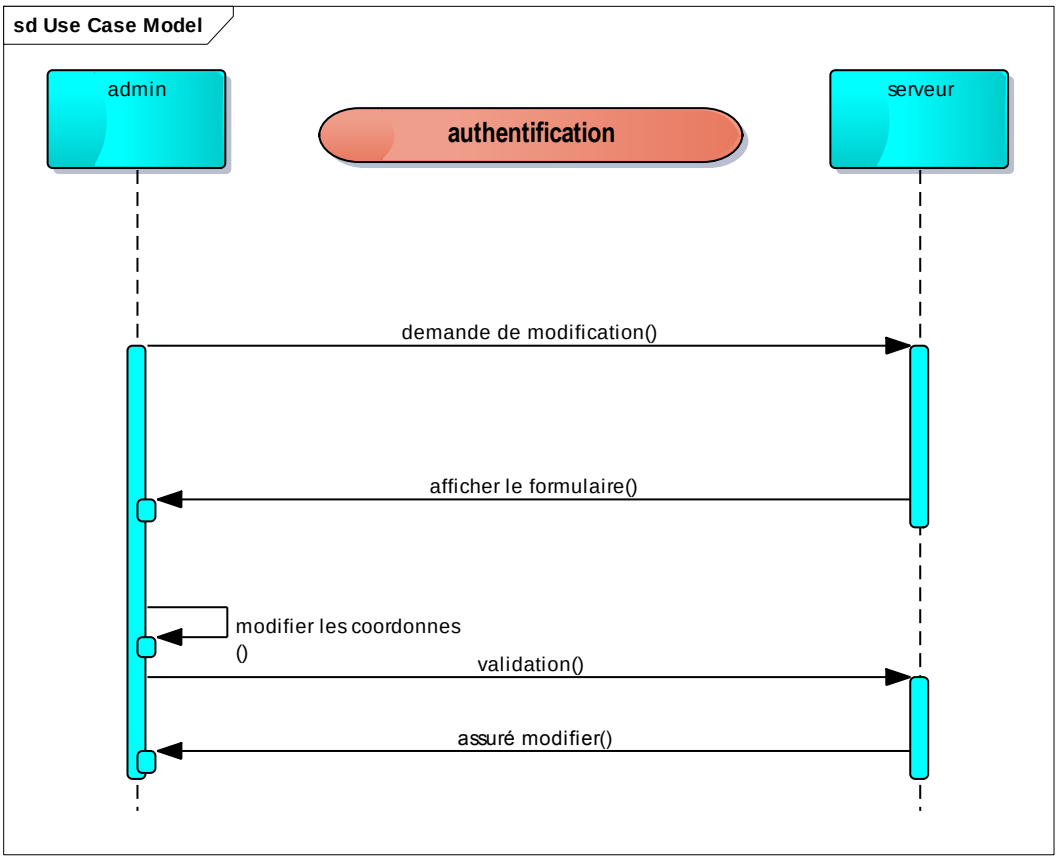


Figure II.6: Diagramme de séquence de modification.

3.4Diagramme de séquence(Suppression) :

-l'authentification

-L'administrateur demande de suppression d'un assuré.

-L'outil affiche la page de suppression d'un assuré.

-L'outil affiche un message de confirmation.

-L'administrateur confirme.

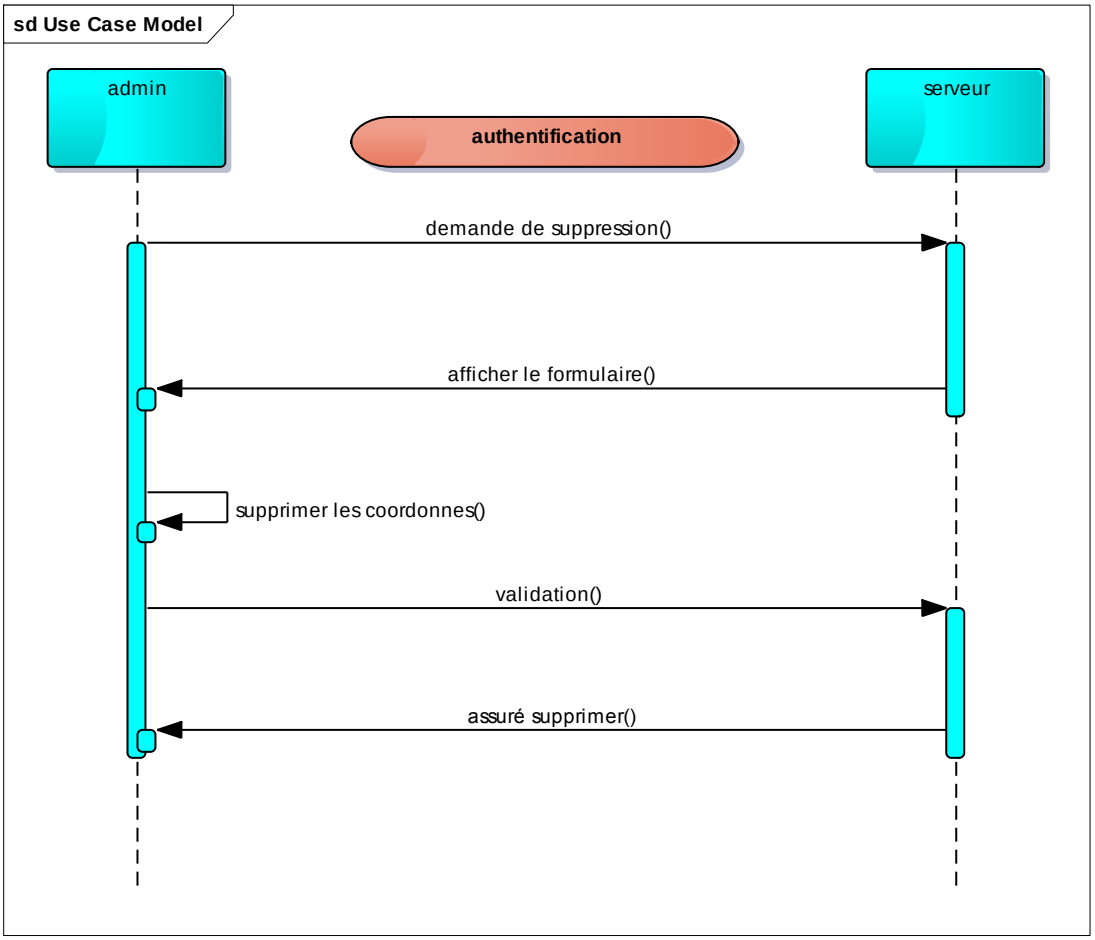


Figure II.7: Diagramme de séquence de supprimer d'un assure.

3.5 Diagramme de séquence (Consultation) :

-l'authentification

-L'administrateurrechercher un assuré

-Afficher le résultat

-Imprimer le résultat

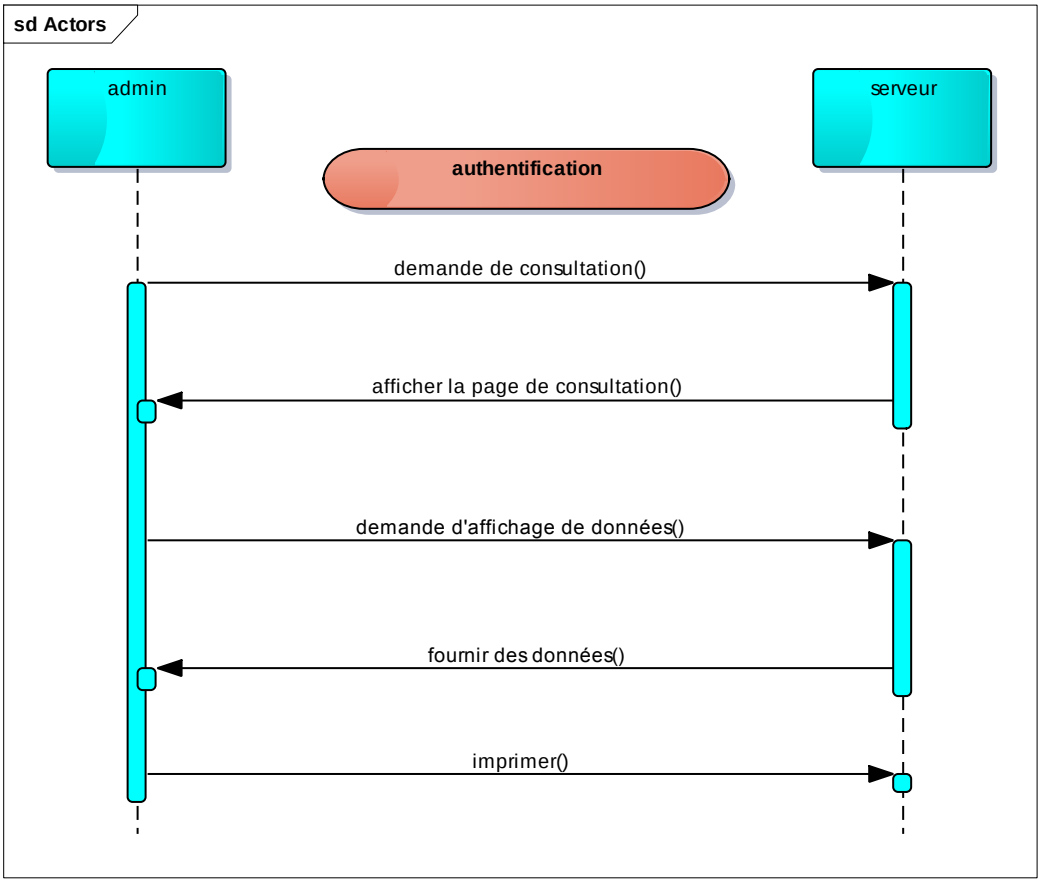


Figure II.8: Diagramme de séquence (consultation)

4. Diagramme d'activité :

Les diagrammes d'activités décrivent le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchainements d'activités.Une activité désigne une suite d'actions

Le passage d'une action vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une action et provoquent le début immédiat d'une autre (elles sont automatiques).[16]

4.1Diagramme D'activité (authentification)

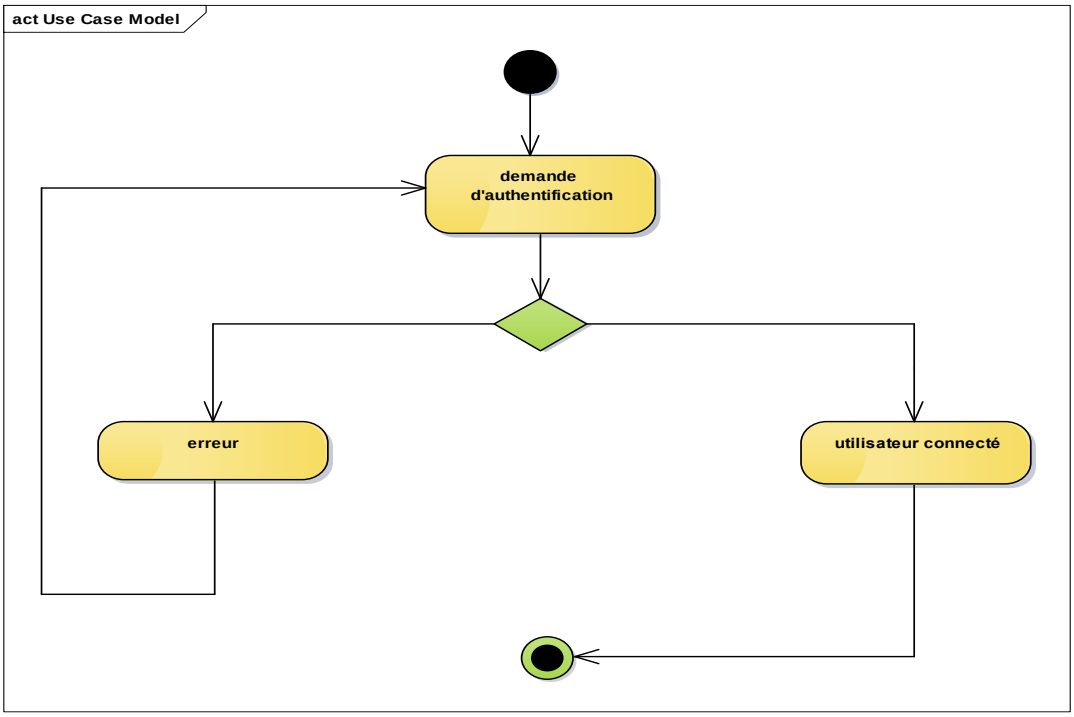


Figure II.9: Diagramme d'activité (authentification)

4.2 Diagramme d'activité (insertion):

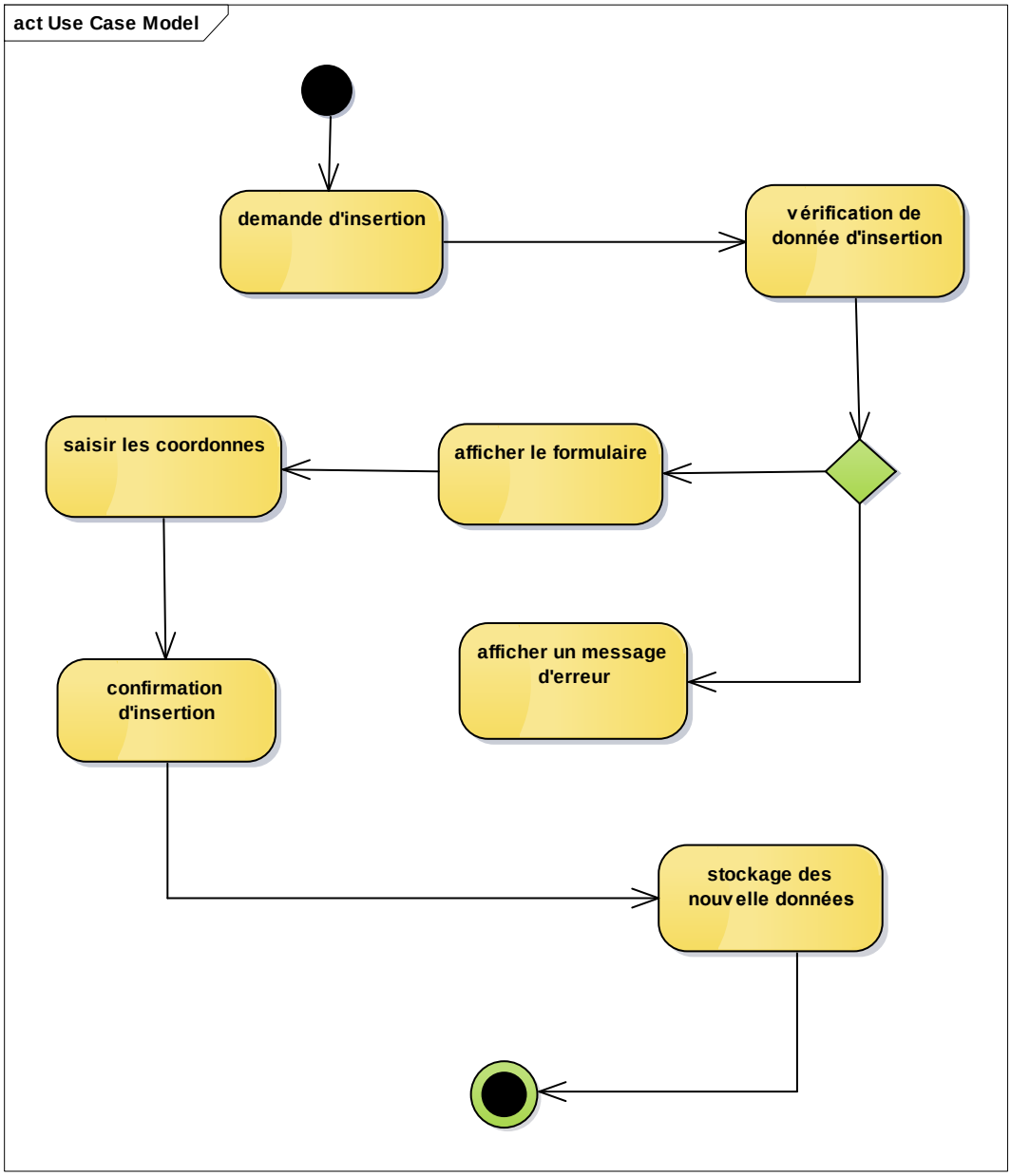


Figure II.10: Diagramme d'activité d'insertion

4.3 Diagramme d'activité (Modifier)

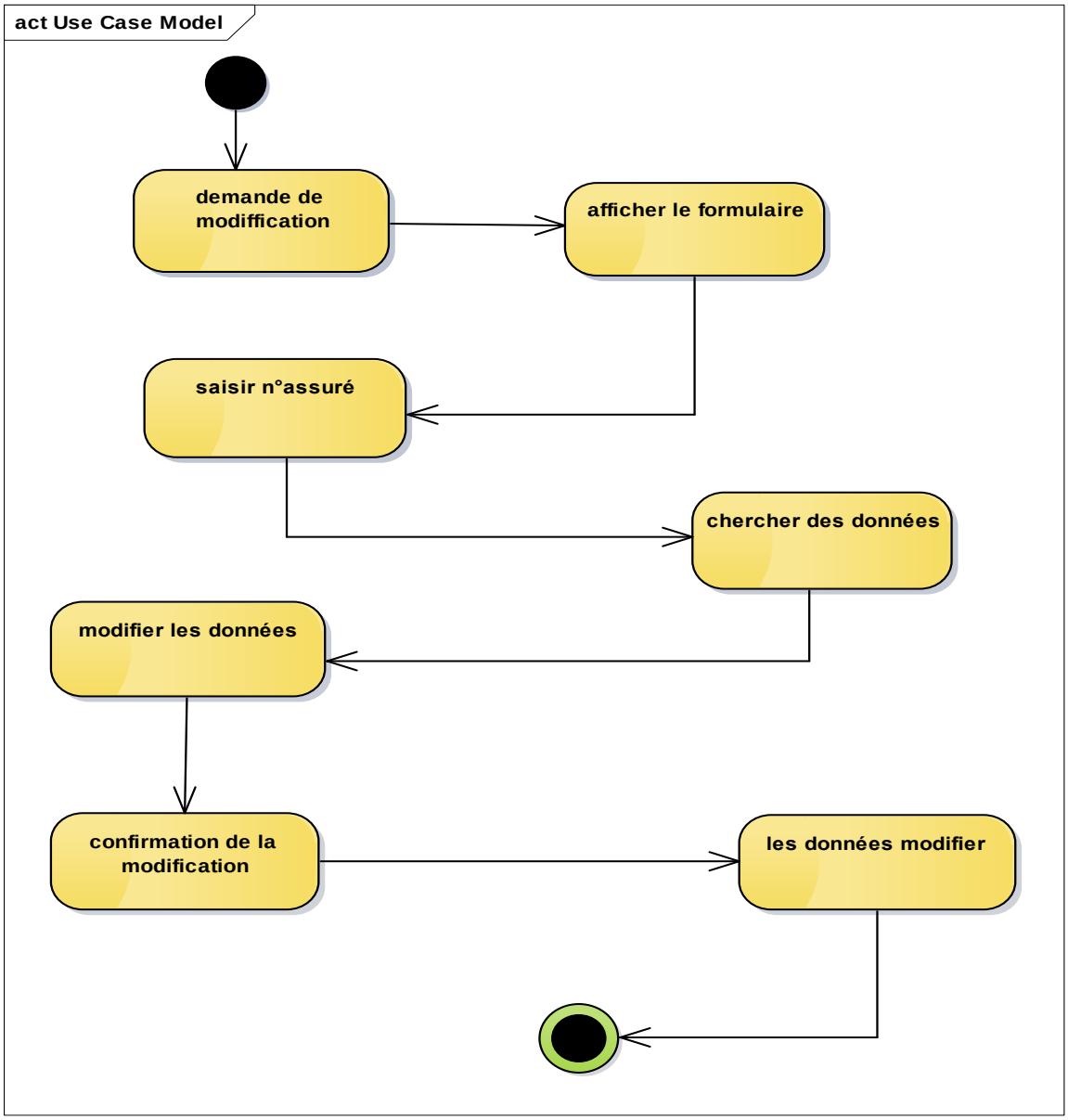


Figure II.11: Diagramme d'activité de modification

4.4 Diagramme d'activité (Suppression) :

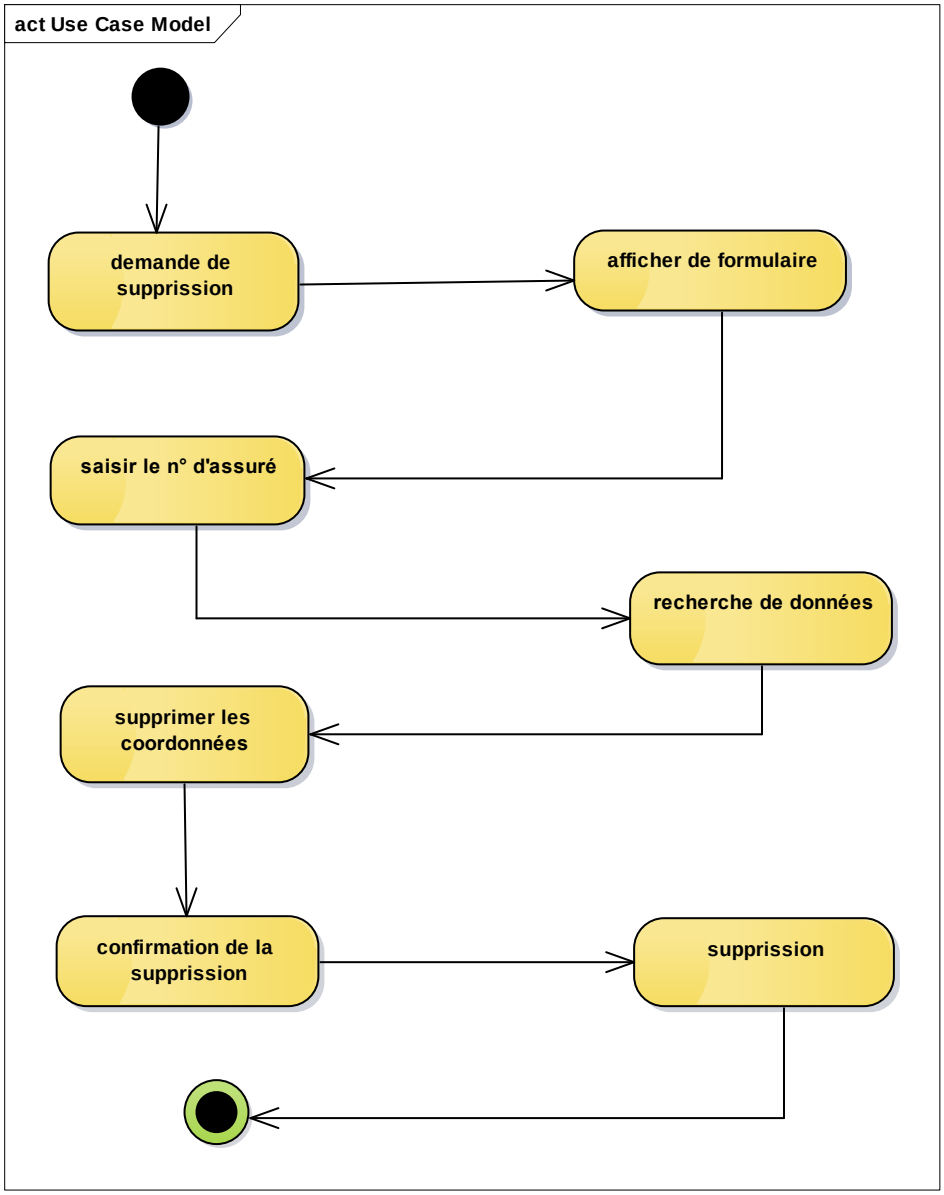


Figure II.12 : Diagramme d'activité de suppression

4.5 Diagramme d'activité (consultation) :

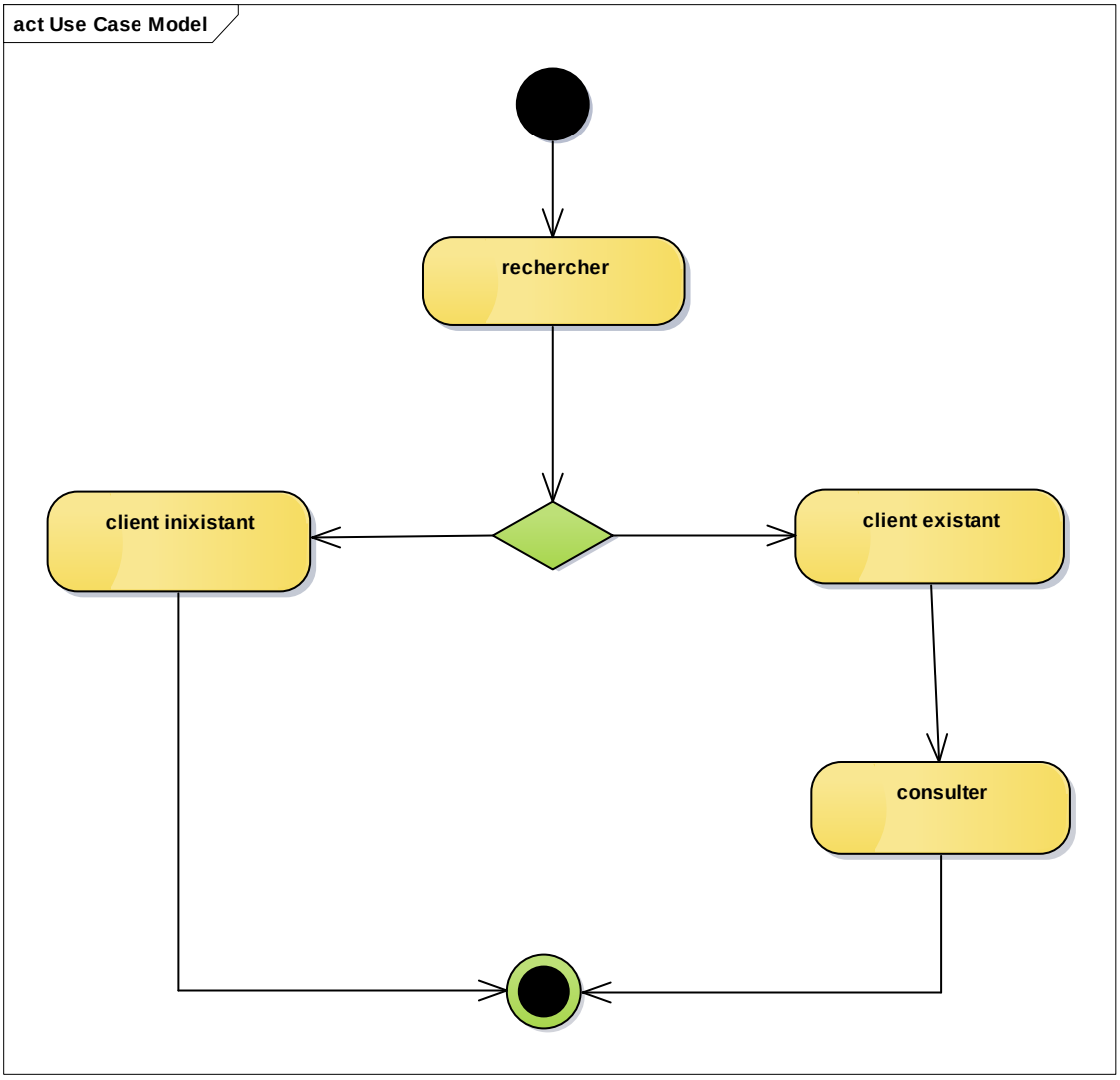


Figure II.13: Diagramme d'activité (consultation)

5. Diagramme de Classe

Les diagrammes de classes servent à comprendre la structure de classe des projets, on les utilise pour personnaliser, partager et présenter avec d'autres classes les informations relatives au projet.[16]

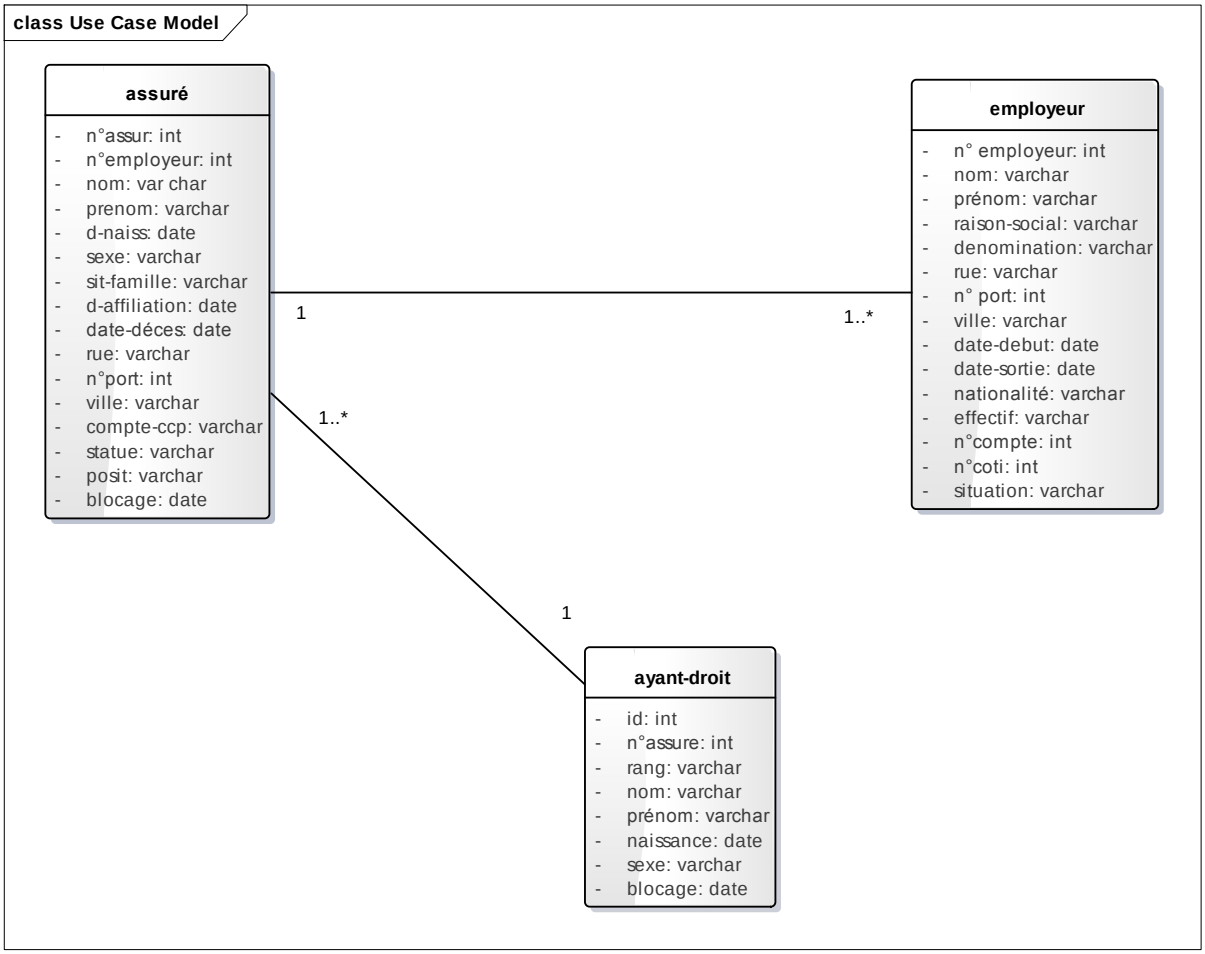


Figure II.14: Diagramme de classe.

III. Réalisation

Introduction

Après les études théoriques et préliminaires menées précédemment nous allons entamer le volet pratique de notre projet. Nous définirons les étapes de réalisation, et la mise en œuvre de notre système de gestion d'affiliation CNAS. Nous commenceront par une brève présentation des outils de développement que nous avons utilisé ; et nous terminerons par l'implémentation de notre système.

1. Outils de développement

1.1 NetBeans

Est un projet open source fondé par Sun Microsystems. L'IDE NetBeans est un environnement de développement permettant d'écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java, mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'IDE NetBeans. L'IDE NetBeans est un produit gratuit, sans aucune restriction quant à son usage.

1.2 Java

Java est un langage Objet permettant le développement d'applications complètes s'appuyant sur les structures de données classiques (tableaux, fichier) et utilisant abondamment l'allocation dynamique de mémoire pour créer des objets en mémoire. La notion de structure, ensemble de données décrivant une entité (un objet en Java), est remplacée par la notion de classe au sens de la programmation Objet. Le langage Java permet également la définition d'interfaces graphiques (GUI : Graphical User Interface) facilitant le développement d'application interactives et permettant à l'utilisateur de piloter son programme dans un ordre non imposé par le logiciel.

1.3 PostgreSQL

PostgreSQL est un Système de Gestion de Bases de Données Relationnelles Objet (SGBDRO), fonctionnant sur diverses plates formes matérielles sous différents systèmes d'exploitation.

PostgreSQL est largement considéré comme le système de bases de données non commercial le plus avancé. Il dispose de nombreuses fonctionnalités que l'on ne rencontre que dans des produits commerciaux lourds.

Ce SGBD est un projet Open Source, ce qui signifie qu'on peut obtenir son code source, l'utiliser et le modifier afin qu'il satisfasse nos besoins personnels, librement sans subir les restrictions des logiciels propriétaires.

2. Structure générale de la solution proposée :

Notre solution consiste à créer sur chaque agencede commune une base de données interfacée avec notre application de gestion de l'affiliation CNAS. Chaque serveur de l'agence partage ses données avec la Direction de la wilaya. Ainsi, ces différentes bases seront consolidées au niveau de la Direction de la wilaya pour avoir une base globale qui contiendra toutes les données.

3. Les composantes applicatives réalisées :

3.1 L'interface de la connexion :

Notre application commence par l'interface d'authentification, où l'administrateur doit saisir son nom d'utilisateur et son mot de passe pour accéder a l'interface principale.

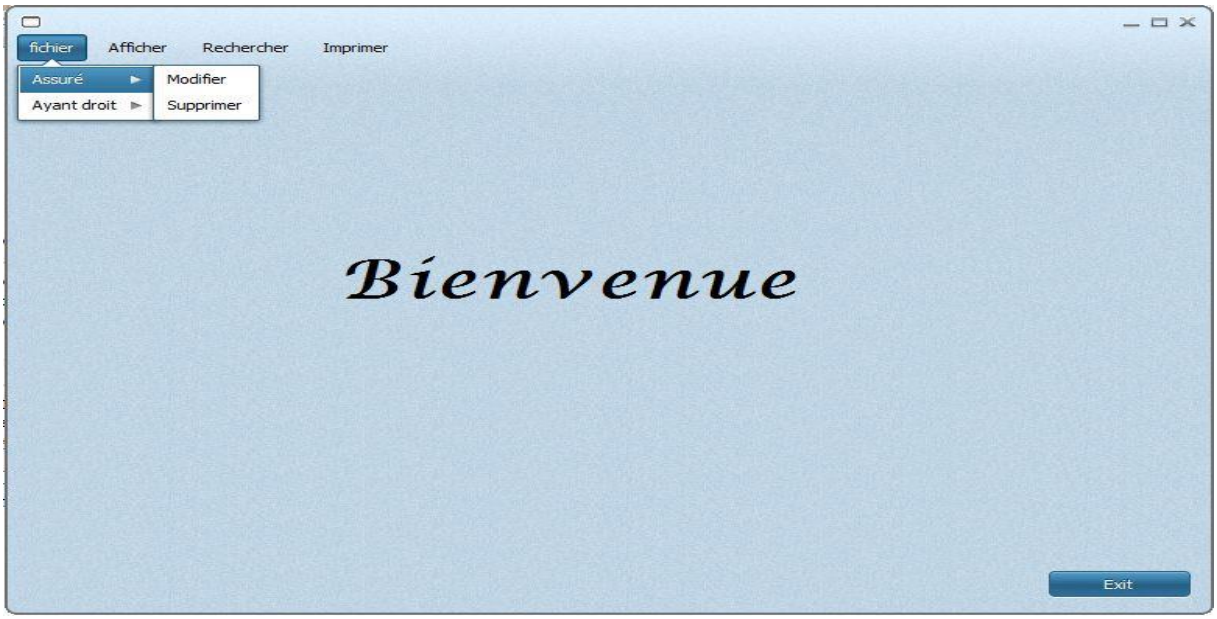
Après l'authentification l'administrateur sera orienté vers l'espace administratif suivant :

3.2 L'interface principale :

Une fois connecté, l'administrateur a le droit d'accéder a l'espace administratif, où il va choisir quel opération a traité.

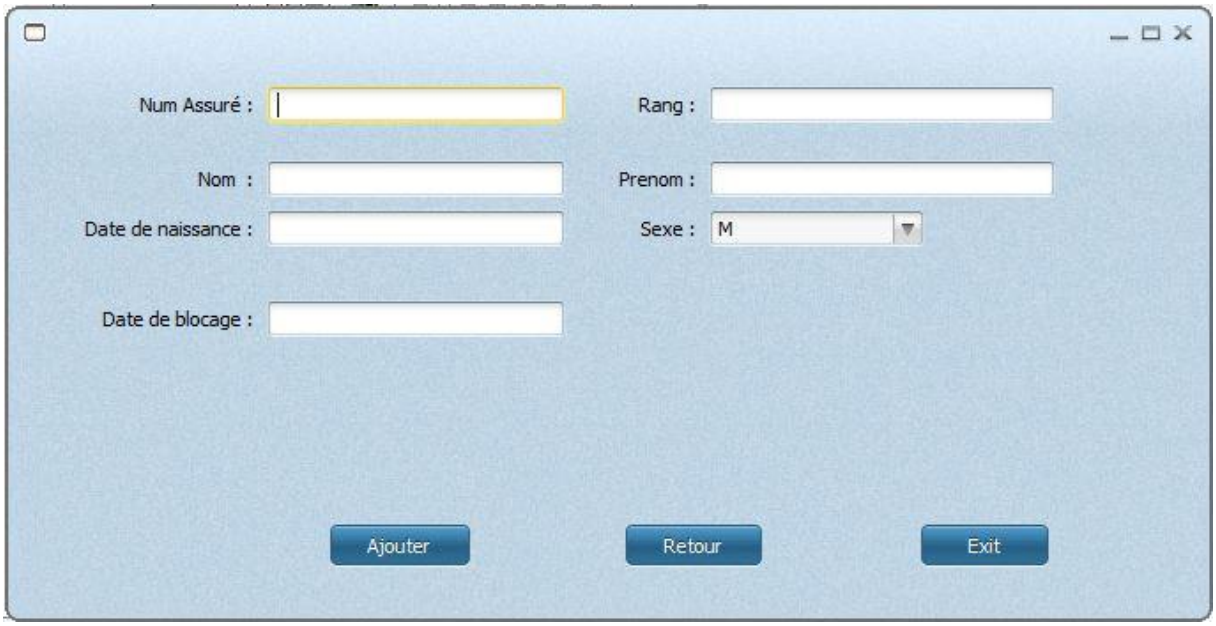
Le menu fichier permet d'exécuter les différent phases de mises à jour (insertion ; modification ; surpression) des assurées et ayants droits.

3.3 Le menu fichier



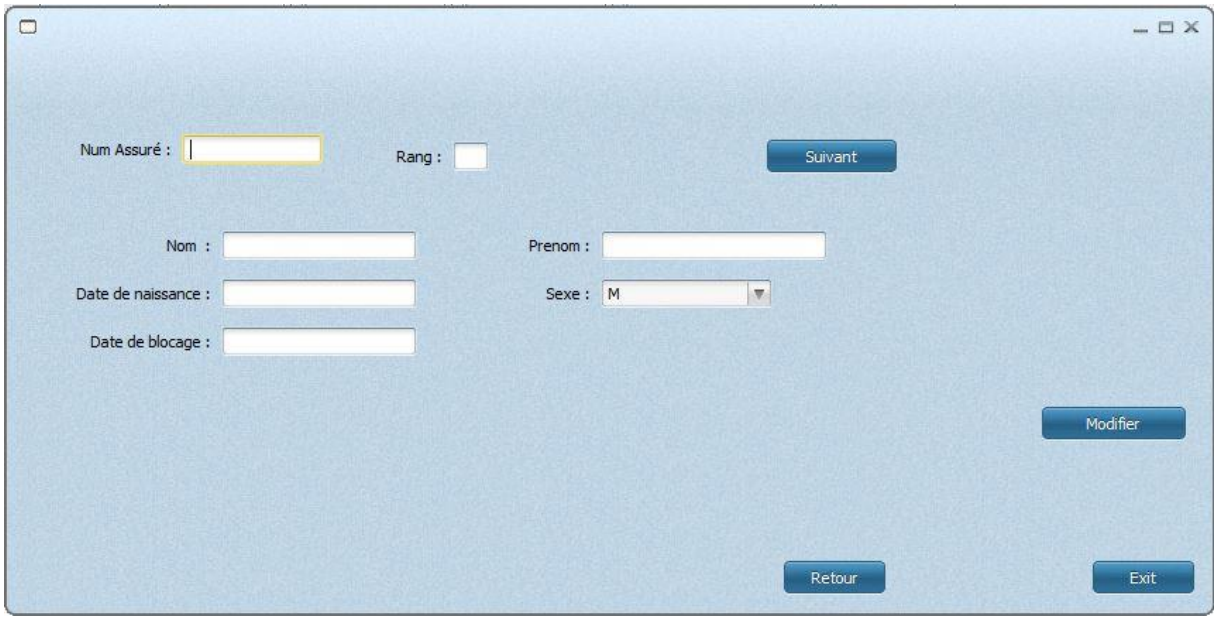
3.3.1 L'interface d'insertion

La phase d'insertion contient des champs à remplir selon les attributs des table stocké dans la base de donné.



3.3.2 L'interface de modification :

Dans la phase de modification, l'administrateur doit saisir l'identifiant de la personne voulu à modifier,

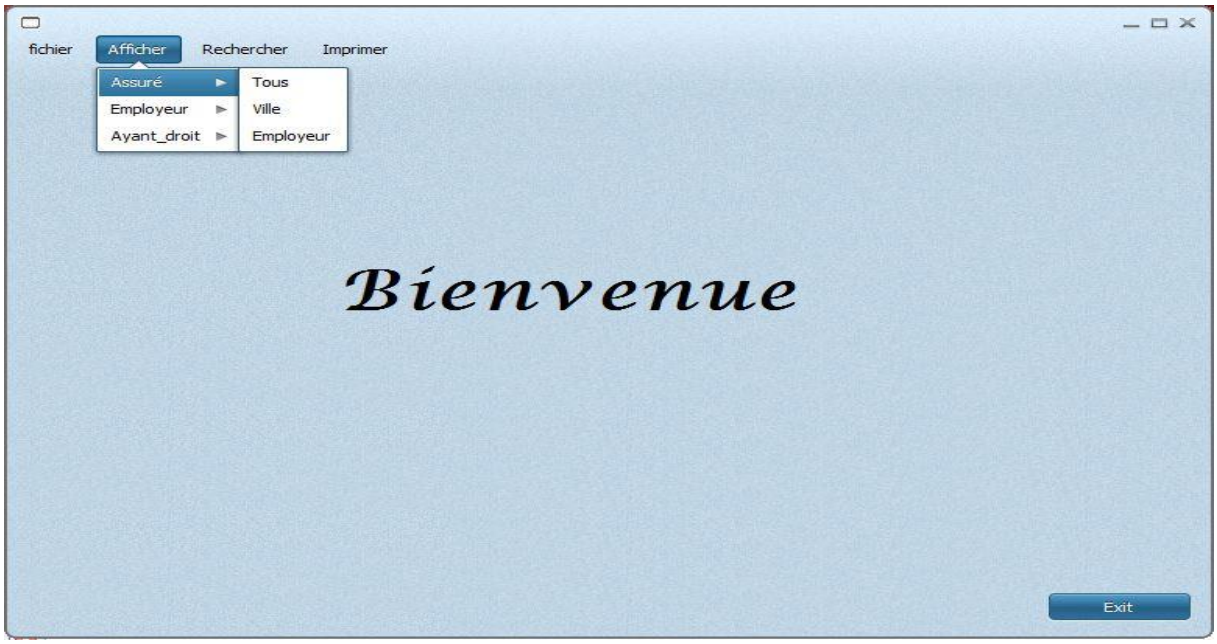


Après la saisi de l'identifiant, tous les champs de cette personne seront visible pour les modifié.

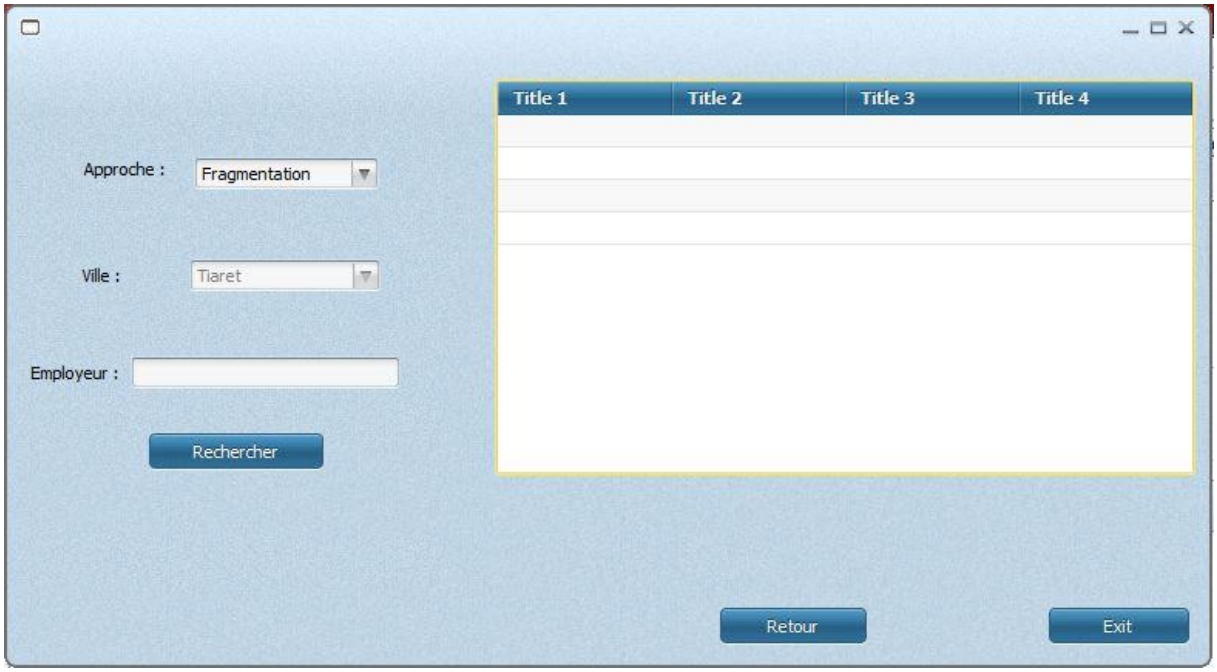
3.3.3 L'interface de suppression :

Similaire à la modification, la validation de l'identifiant permet de supprimer la personne voulu (assuré, employeur, ayant droit) après une confirmation de la suppression.

3.4 Le menu d'affichage

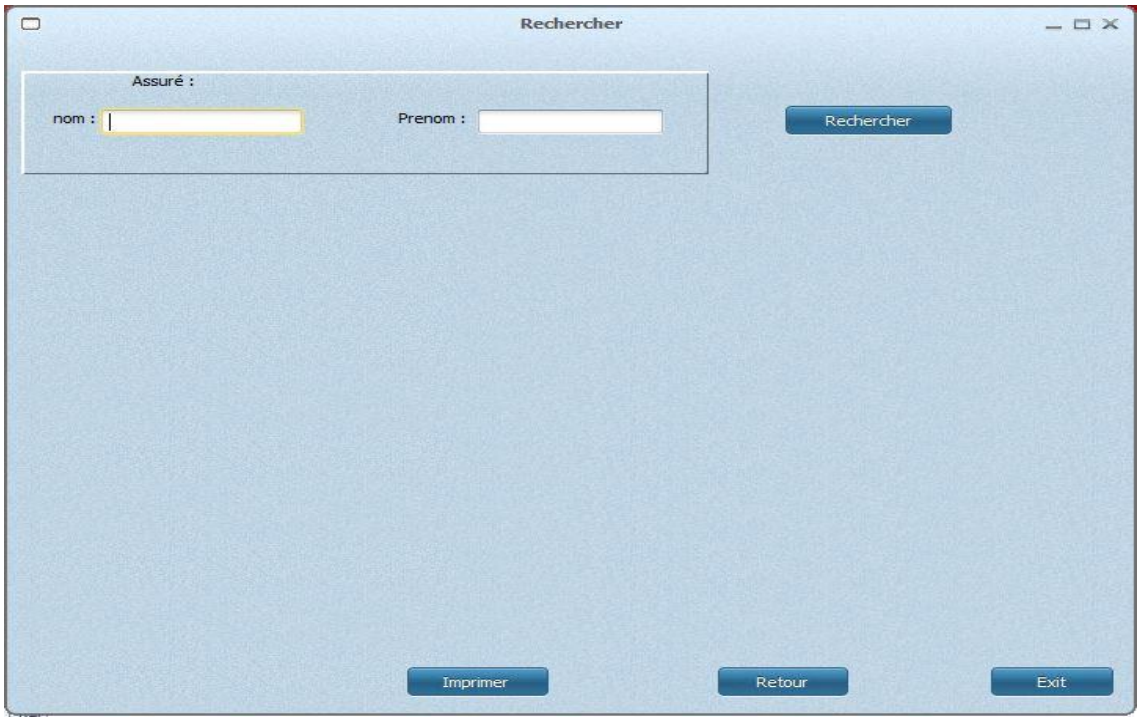


Le menu afficher permet de visualiser les enregistrements (assuré, employeur, ayant droit), selon un tel choix.



Le menu recherche permet de rechercher les ayants droit et l'employeur qui convient a l'assuré voulu afficher.

3.5 Le menu de recherche



Conclusion

Dans cette dernière partie, nous avons présenté les différentes parties nécessaires à la réalisation de notre système de gestion d'affiliation, à savoir la création de la base de données, présentation des différents outils utilisés pour réaliser notre solution, illustration des différentes interfaces de notre application.

Conclusion générale

Une base de données répartie (BDR) est une base de données dont les différentes parties sont stockées sur des sites géographiquement distants, reliés par un réseau. La réunion de ces parties forme la base de données répartie. L'objectif que nous avons visé lors ce projet est la conception et la réalisation d'un système gestion des affiliés CNAS en appliquant deux techniques de répartition de données entre la Direction de la wilaya et ses agences.

Notre mémoire est articulé autours de deux partie ; la première est théorique, il s'agit de l'état de l'art sur les bases de données réparties.

La seconde partie de notre mémoire est la partie pratique, dont laquelle nousavons commencé par la création des bases de données qui modélisentlesystème de gestion des affiliés CNAS et nous avons construit les interfaces graphiques permettant de réagir avec cette base grâce à NetBeans.

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 la modélisation avec le langage UML, l'environnement de développement java à savoir l'IDE Netbeans, sous lequel, le développement n'a pas été une tâche facile, avec notamment la présence d'un SGBD puissant comme PostgreSQL.

Bibliographie

[1] Karkowiak S, ‘Introduction aux systèmes et applications réparties’. Université Joseph FOURRIER de Grenoble, 2005, PP.86.

[2] Chrisment C, Gujolle G etZurfluh G. ‘Base de données réparties. Techniques de l'ingénieur’, Décembre 1993, pp.15.

[3] Ferrara P. ‘Base de données répartie’. HEG Haute Ecole de Recherche Neuchâtel Suisse, Avril 2004, pp.105.

[4] Spaccapietra S. ‘Base de données réparties’. Ecole Polytechnique Fédérale de Lausanne, Mars 1998, pp.206.

[5] Cornuéjol A. ‘Bases de données’. Université Paris Sud, 1999, pp.68.

[6] Zeylan E. ‘Base de données distribuées’. Haute Ecole de Gestion ARC Neuchâtel Suisse, Novembre 2003, pp.50.

[7] Baudet C et Meylan E. Bases de données réparties’, Haute Ecole de Gestion ARC Neuchâtel Suisse, Octobre 2007, pp.85.

[8] Fleury P et Guignard J. ‘Réplication d'une base de données’. Ecole Nationale des Arts et Métiers de Paris, Edition 2000, pp.34.

[9] DesfontainV. ‘Introduction aux bases de données réparties’. Université de Technologie de Compiègne, Septembre2000, pp.60.

[10] Moussa R. ‘Système de gestion de bases de données réparties et mécanisme de répartition sous ORACLE’. Ecole Supérieure de Technologie et d'Informatique à Carthage, Novembre 2005, pp.17.

[11] Meylan E. ‘Réplication de données. Bachelor of science en informatique de gestion’, Haute Ecole de Gestion ARC Neuchâtel Suisse, Février 2005, pp.97.

[12] Deleglise D. ‘Manipulation des vues matéralisées’, Juin 2008, pp.101.

[13] Woscichowski P. ‘La réplication sous ORACLE’, Ecole Supérieure d'Informatique, Promotion SUPINFO 2005, pp.48.

[14] Madi A et Mokrani F. ‘Conception et réalisation d’une BDr sous oracle’, Université A. Mira de Bédjaia. Mémoire Master recherche en informatique, Promotion 2009, pp.38. 14. 22.

[15] Madani N,‘ParallelQuerySystem’,Ecole Nationale Superieure d’informatique ESI Alger. Promotion 2010, pp.64.

[16] Abdat.N et Mahdaoui.L ,'UML outil du génie logiciel',septembre 2007,pp.28.32.48.67.