MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE

SCIENTIFIQUE UNIVERSITE ABDELHAMID

IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

RAPPORT DE MINI-PROJET

Option : **Ingénierie des Systèmes d'Information**

THEME :

Contributions à la réplication et Durabilité de données dans les bases de données répartis

Etudiant(e) : « **BOUIDIDA FETHI** »

« **BENCHERIF ALI** »

Encadrant : « DJEBARA REDHA »

« BENAMEUR ABDELKADER »

Année Universitaire 2020-2021

Table des matières

DEDICACE

A toutes nos Familles ;

A tous ceux qui nous ont soutenu dans nos études ;

A tous ceux qui nous aiment ;

Nous dédions ce Mémoire.

Mr BOUIDIDA FETHI

Mr BENCHERIF ALI

REMERCIEMENTS

Nos remerciements s'adressent à toute la communauté académique de l'Université INES de Mostaganem pour l'encadrement scientifique et spirituel à notre égard. Nous tenons à remercier le docteur DJEBARA Mohamed Redha & le docteur BENAMEUR ABDELKADER, Directeurs de ce travail, pour l'intérêt qu'il a manifesté tout au long de ce projet. Nous lui témoignons nos sentiments de gratitude pour l'œuvre accomplie malgré ses multiples occupations.

Nous remercions, d'une manière particulière, la famille BOUIDIDA et BENCHERIF pour son sacrifice, amour et soutien de nos études depuis notre bas âge jusque maintenant ; qu'ils trouvent ici le fruit de leur dévouement inestimable. Sans oublier les frères, sœurs, oncles, tantes, grand-mères, fils, filles, etc.

Aux camarades étudiants, compagnons de lutte, nous disons également merci pour les moments de joie et de compréhension mutuelle dont nous avons été bénéficiaires ; Enfin que toute personne ayant marquée de son empreinte ce travail, trouve ici l'expression de notre profonde gratitude.

Mr BOUIDIDA FETHI

Mr BENCHERIF ALI

Résumé

De nos jours, les bases de données réparties et la réplication des données sont reconnues comme étant des moyens efficaces pour augmenter la disponibilité et la fiabilité des bases de données, La réplication offre aux utilisateurs de meilleures performances et une plus grande disponibilité des données. Toutefois, celle-ci introduit le problème de cohérence mutuelle des copies. La mise à jour des données doit prendre effet sur toutes les copies.

La notion de durabilité est capitale dans les bases de données. Elle assure que lorsqu'une base de données tombe en panne, les transactions qui ont réussi leur commit sont effectivement préservées sur la mémoire stable de la base de données. La durabilité est donc une technique de tolérance aux pannes. Une autre technique de tolérance aux pannes est celle de la réplication, des copies multiples assurent que, si une copie tombe en panne, les autres copies continuent de maintenir le service. Naturellement, ces deux techniques de tolérance aux pannes ont leur prix, que ce soit en termes de complicité ou de performance.

Nous avons fixé l'objectif de répondre à la question de savoir quel modèle informatique permettrait de s'assurer de la disponibilité des informations découlant des opérations en cours entre les sites distants afin de répondre, en temps réel, aux différentes requêtes que les utilisateurs peuvent exécuter à partir de n'importe quelle base.

L'application du Langage de Modélisation Unifiée (UML), des techniques algorithmiques, du prototypage, de l'expérimentation et de la simulation nous a permis d'atteindre l'objectif que nous nous étions fixé. Comme résultats, nous avons conçu un modèle informatique de la réplication des données sous forme de diagrammes UML. Nous avons également testé ces algorithmes avec SQL Serveur et PHP7 qui nous ont permis de faire le monitoring des modifications qui se font au niveau de chaque site.

Introduction Générale

Problématique

De nombreux outils de synchronisation sont déjà fournis par des serveurs de base de données tels que Microsoft SQL Server et Oracle. Ce sont des solutions performantes et intégrées mais elles ne fonctionnent qu'à l'intérieur de leur propre système. Le besoin des établissements de communiquer à l'extérieur de leur structure (avec des citoyens par exemple) amène à synchroniser des données entre bases de données de différentes organisations et qui peuvent donc être hétérogènes (SGBD).

Cet état de choses pose problème en termes de la gestion de la file d'attente et handicape un certain nombre de transactions faute de connexion au réseau ; un triste événement qui survient au quotidien.

Tous les services publics de l'administration aux citoyens sont effectuent à travers des fichiers électroniques où l'électrification et la communication via Internet ou Intranet demeurent un défi, ont du mal à assurer la haute disponibilité des informations de leurs citoyens ainsi que de leurs responsables des services, parce que ces difficultés font que le partage des données entre sites ne se fasse pas en temps réel. Cet état de chose pose également problème en termes de la satisfaction des citoyens qui ne sont pas servis au moment opportun faute de connexion au serveur des données.

Limiter les problèmes ci-haut est un moyen sûr de gagner la confiance des citoyens, car si on parvient à garantir la disponibilité des données et des services publics, on peut y parvenir. Les administrations locales (Daïra, Communes) sont, par conséquent, obligées de trouver une solution à ces différents dégâts qu'occasionnent les problèmes précités de peur d'accuser une certaine fragilité dans leurs services publiques. C'est pour toutes les raisons précitées qu'il convient de s'interroger sur le modèle informatique qui permettrait de s'assurer de la disponibilité des informations découlant des opérations en cours avec les autres bases afin de satisfaire en temps réel les demandes des citoyens.

Objectifs de l'étude

Ce travail de recherche aura pour objectif de concevoir (et tester) un Algorithme de Réplication des données d'une collectivité locale (commune) afin de permettre cette dernière à avoir en permanence les données des opérations avec les autres antennes de la commune dans l'objectif de répondre efficacement aux besoins des citoyens même lorsqu'il y a problème de connexion au réseau ou coupure du courant dans l'une des antennes.

Choix et Intérêt du Sujet

Tous les jours les établissements à succursales multiples se partagent des informations

très utiles sans lesquelles aucune évolution ne serait possible. Mais il arrive de temps à autre que ce partage devient difficile suite aux problèmes que nous pouvons citer :

✓ Il arrive des fois que la connexion au serveur, pour l'une des antennes, soit impossible

✓ Il est des cas où il y a coupure au niveau du serveur central ou au niveau de l'Antenne d'où on attend des informations

✓ Etc.

Ce faisant, les gestionnaires de ces services se retrouvent face à cette situation presque tous les jours et ne savent pas comment la résoudre. Nous avons pensé qu'orienter nos recherches dans ce sens serait non seulement utile pour ces gestionnaires, aussi constituera-t-il une documentation pour ceux qui voudront mener des travaux dans ce domaine. En outre, il sera aussi d'une grande importance dans la mesure où, il aidera l'Administration non seulement à centraliser ses opérations, mais aussi à satisfaire ses citoyens au moment voulu.

Méthodes et Techniques de Travail

Pour bien mener nos recherches, nous userons de quelques méthodes et techniques des recherches suivantes :

✓ le Langage de Modélisation Unifié (UML) pour faire l'analyse et la conception de la solution à l'aide des algorithmes que nous présenterons sous forme des diagrammes ; ✓ la définition des algorithmes que nous représenterons sous forme de diagrammes d'activité, de séquence système et d'états, mais aussi dans la Machine de Turing ; ✓ le prototypage, l'expérimentation et la simulation.

Chapitre 1

Introduction

Au cours de la dernière décennie, les recherches et les évolutions autour des technologies de réseaux, des microprocesseurs et des supports de stockage de données ont fortement contribué à l'émergence de l'informatique distribuée (Kesselman, 1998). Les performances de calcul des microprocesseurs et les capacités de stockage de données ne cessent de croître, alors que les nouvelles technologies de réseau ouvrent de nouvelles potentialités pour les communications. Ce contexte a motivé la communauté informatique à s'intéresser aux architectures distribuées à large échelle, afin d'offrir des solutions pour le stockage de données et le calcul réparti à un plus grand nombre d'applications et d'utilisateurs.

En parallèle, l'évolution des systèmes informatiques s'est caractérisée par une tendance très forte vers la décentralisation. La communication, le partage de données, la puissance de calcul et la capacité de stockage sont des besoins fortement exprimés par les nouvelles applications informatiques (Bernholdt, 2005, pp. 485-495). Des exemples d'applications couvrant différentes disciplines ont montré le besoin de disposer d'architectures plus flexibles, favorisant le calcul et la distribution de données à grande échelle (Nabrzyski, 2003). Des disciplines aussi diverses que la météorologie, la science des particules, la médecine et la biologie sont des exemples où les applications requièrent de grandes capacités de calcul et de stockage. La tendance à la distribution a entraîné une évolution de l'architecture des systèmes distribués et des outils pour la conception et la mise en œuvre d'applications distribuées.

Les bases de données réparties et la réplication des données sont reconnues aujourd'hui comme étant des moyens efficaces pour augmenter la disponibilité et la fiabilité des bases de données. La réplication offre aux utilisateurs de meilleures performances et une plus grande disponibilité des données. De nombreuses recherches ont mené à cette conclusion étant donné que la question de la haute disponibilité des données ne date pas d'aujourd'hui, et beaucoup d'auteurs se sont exprimés à propos.

Dans cette partie de chapitre, nous souhaitons passer en revue les théories, les méthodes et les résultats obtenus par d'autres Auteurs au cours de leurs investigations. Ce mémoire cherche donc à trouver une solution à la question de savoir quel modèle informatique permettrait de s'assurer de la disponibilité des informations découlant des opérations en cours entre les bases de données afin de répondre, en temps réel, aux différentes requêtes que les utilisateurs peuvent exécuter.

1. Base de Données Reparties ou Distribuées

En décembre 2007, Mathieu EXBRAYAT a désigné par base de donnée répartie (ou distribuée), une base de données logique dont les données sont distribuées sur plusieurs SGBD et visibles comme un tout. Quoi qu'Il s'est limité à expliquer les concepts liés à la notion des bases de données reparties et leurs techniques de mise en place. Ses recherches sont d'une importance capitale dans la mesure où elles aident à bien comprendre différents concepts utilisés dans la mise en place des bases de données distribuées quand bien même qu'il n'offre aucun algorithme de réplication des données capable de répondre à notre question de recherche.

Pour J. Akoka et I. Wattiau (2011), une base de données distribuée permet la création, l'accès et la manipulation des données inter reliées sur différents sites d'un réseau informatique. Ils ont listé quelques éléments à retrouver dans une architecture distribuée : • Chaque site a une capacité de traitement local autonome

• L'accessibilité, le partage, la performance et la disponibilité sont améliorés • l'optimisation globale des traitements, dans le but de préserver les avantages de la base de données centralisée.

Ils ajoutent quelques facteurs dont dépend la base de données centralisée :

- Du coût de communication
- Du coût des traitements locaux
- De la stratégie d'allocation des données et
- De la stratégie d'exécution de traitements

MATHIEU EXBRAYAT présente deux approches de la **Conception répartie** ; il s'agit de la **Conception ascendante** c'est à dire qui intègre les bases locales dans un schéma global (**Décomposition**) et la conception descendante qui part du schéma global et le scinde en schémas locaux (**Intégration**). Graphiquement, sa conception produit la figure suivante :

Figure 1 : Conception BD Répartie par Décomposition et par Intégration (Mathieu EXBRAYAT, 2007)

Ces notions sont d'une importance capitale dans la mesure où elles facilitent la compréhension des concepts de bases de données réparties étant des structures favorisant la réplication préventive des données, c'est-à-dire qui prévoit la tolérance aux pannes.

2. Réplication de Base de Données

La synchronisation des données a toujours été un problème dans l'histoire de l'informatique en général, et des SGBDR en particulier. La normalisation des modèles a apporté une solution en l'évitant au maximum, l'information ne devant se trouver qu'à un seul endroit. Pour s'assurer de la disponibilité permanente des données, souvent, il est nécessaire de répliquer de l'information. Les bonnes raisons pour le faire sont nombreuses :

- Dénormalisation pour des questions de performances
- Distributions géographiques, décentralisation
- Récupération de données d'autres environnements, centralisation
- Sécurisation, sites de secours distants

Comme l'affirme LAMBERT SONNA MONO (2001), les bases de données réparties et la réplication des données sont reconnues aujourd'hui comme moyens efficaces pour augmenter la disponibilité et la fiabilité des bases de données. De plus la réplication peut contribuer favorablement à l'amélioration des performances en utilisant les copies locales voire les copies plus proches.

En 2011, AKOKA et Wattiau définissent la réplication selon son mode de mise à jour. Pour eux, une **réplication synchrone** (réplication transactionnelle, de capture instantanée) est celle qui permet une mise à jour immédiate et qui utilise un protocole de validation à deux

phases, et **une réplication asynchrone**, celle qui permet une mise à jour différée (Réplication par fusion).

2.1 Réplication des Données

Pour MOLLI et Gérald (2005), répliquer revient à dupliquer des données critiques pour la tolérance aux pannes, pour la disponibilité et pour la performance. Pour eux, il existe une différence entre répliquer et copier dans ce sens que répliquer occasionne la cohérence des données et des répliques. De ce fait la réplication est plus vaste qu'une simple copie. D'où les concepts suivants sont nécessaires pour notre sujet :

- **Les Pannes :**

Un système est en panne s'il ne sert pas les données attendues en un temps attendu. Dans

un système distribué, on peut rencontrer de problèmes tels que :

- Un site peut s'arrêter pour une raison logicielle ou matérielle ;
- Un site peut renvoyer des informations illisibles ou incompréhensibles : pannes byzantines ;
- Un lien peut être coupé ou marcher que dans un sens ;
- Un message peut être perdu ou arrivé hors délais ;
- Un message peut être altéré pendant le transport (Byzantine Communication Failure) ;
- Les erreurs de communication peuvent provoquer des partitions réseaux ;

• La Réplication et le Système Distribué

La réplication implique plusieurs sites interconnectés, donc un ensemble des sites et des liens de communication. Un site désigne, dans ce cas, un processus et un stockage des données au niveau d'un serveur local et Un lien de communication comme étant un canal de communication bidirectionnel entre deux sites. Ces derniers communiquent en utilisant des messages et un protocole de communication. La délivrance des messages n'est pas garantie dans un temps maximum.

Pour Cédric COULON (2006), la haute performance et la haute disponibilité des bases de données ont été traditionnellement gérées grâce aux systèmes de bases de données parallèles, implémentés sur des multiprocesseurs fortement couplés. Le traitement parallèle des données est alors obtenu en partitionnant et en répliquant les données à travers les nœuds du multiprocesseur afin de diviser les temps de traitement. Cette solution requiert un Système de Gestion de Base de Données (SGBD) ayant un contrôle total sur les données. Bien qu'efficace, cette solution s'avère très coûteuse en termes de logiciels et de matériels.

Les grappes sont composées d'un ensemble de serveurs (PC) interconnectés entre eux par un réseau. Ils permettent de répondre aux problématiques de haute performance et de haute disponibilité. Elles ont été utilisées avec succès pour, par exemple, les moteurs de recherches Internet utilisant des fermes de serveurs à grands volumes (e.g. Google). Les grappes peuvent également être utilisées dans un nouveau modèle économique, les Fournisseurs de Services d'Applications (ASP - Application Service Providers). Dans un contexte ASP, les applications et les bases de données des clients sont stockées chez le fournisseur et sont disponibles, typiquement depuis Internet, aussi efficacement que si elles étaient locales pour les clients.

Pour améliorer les performances, les applications et les données peuvent être répliquées sur plusieurs nœuds. Ainsi, les clients peuvent être servis par n'importe quel nœud en fonction de la charge. Cet arrangement fournit également une haute disponibilité : dans le cas de la panne d'un nœud, d'autres nœuds peuvent effectuer le même travail. Pour lui, Un autre avantage du modèle ASP concerne le déploiement. La mise à jour d'une application ou d'un SGBD ne

demande pas le déplacement d'un technicien chez tous les clients. La mise à jour des nœuds chez le fournisseur est suffisante.

2.2Algorithmes de Réplication

IMINE (2006), a conçu un éditeur collaboratif fondé sur l'approche des transformées opérationnelles qui peut être facilement déployé sur un réseau Pair-à-pair (P2P). Il a aussi proposé un modèle formel pour l'approche des transformées opérationnelles qui lui permettrait de concevoir des algorithmes corrects. Son modèle permet de spécifier et vérifier des objets synchronisés par une transformation opérationnelle et a conçu un outil qui permet de définir les opérations et l'algorithme de transformation pour un objet donné (comme l'objet texte par exemple).

3. Durabilité dans les Systèmes Répartis

Un système réparti est un système informatique dont les composants logiciels s'exécutent sur des ordinateurs(ou nœuds) interconnectés par un réseau. La notion de durabilité est capitale dans les bases de données. Elle assure que lorsqu'une base de données tombe en panne, les transactions qui ont réussi leur commit sont effectivement préservées sur la mémoire stable de la base de données. La durabilité est donc une technique de tolérance aux pannes. (Lambert SONNA MOMO, 2001)

Il ajoute : « Une autre technique de tolérance aux pannes est celle de la réplication, des copies multiples assurent que, si une copie tombe en panne, les autres copies continuent de

maintenir le service. Naturellement, ces deux techniques de tolérance aux pannes ont leur prix, que ce soit en termes de complicité ou de performance. »

4. La Tolérance aux fautes dans les systèmes répartis

La tolérance aux fautes s'inscrit dans le contexte plus large de la sûreté de fonctionnement. La sûreté de fonctionnement (***dependability***) d'un système informatique est la propriété qui permet à ses utilisateurs de placer une confiance dans le service qu'il délivre. Le service délivré par un système est son comportement tel qu'il est perçu par ses utilisateurs.

- Attributs de la sureté de fonctionnement : La sûreté de fonctionnement (dependability) d'un système peut être abordée sous des angles différents, selon les fonctions que remplit le système et selon le domaine d'applications auquel il est destiné.

1. la **disponibilité**(availability) définit le fait d'être prêt à l'utilisation.

2. la **fiabilité**(reliability) d'un système est définit comme une fonction R(t) du temps qui représente la probabilité que le système survive jusqu'au temps t. **3.** la **sûreté**(safety) respecte la non occurrence de défaillance catastrophique. **4.** la **sécurité-confidentialité**(security) prévoit la non occurrence des accès non

autorisés ou l'acquisition non autorisé des informations.

5. la **maintenabilité**(maintenability) définit l'aptitude aux réparations et aux évolutions.

- **Entraves à la sureté de fonctionnement :** Mettre en œuvre la sûreté de fonctionnement d'un système correspond à lutter contre les défaillances du système. Une défaillance (failure) survient lorsque le service délivré par le système ne correspond plus à sa spécification. La spécification du service correspond à la description du service que les utilisateurs sont à mesure d'attendre du système. Une erreur(Error) est une anomalie de l'état du système et susceptible d'entrainer une défaillance. Une erreur ne provoque pas systématiquement une défaillance. Le système peut continuer à délivrer un service correct malgré un certain nombre d'erreurs affectant son état. Une faute (fault) est la cause d'une erreur. Une erreur peut provoquer une défaillance : Une altération du service délivré par le système.

faute(fault)->erreur dans l'état du système -> défaillance dans le système.

Les défaillances pouvant affecté un système sont variées. L'élaboration d'une stratégie efficace de lutte contre les défaillances nécessite une caractérisation précise des défaillances à combattre. Les lignes suivantes sont inspirées de (A.Schiper, February 2000).

La figure 1.1 présente les classes de défaillance les plus fréquentes. Elles sont classifiées en quatre catégories, des moins graves aux plus graves. Lorsque le système omet de délivrer le service demandé, il exhibe une défaillance par omission (omission fault). Si l'omission devient permanente (le système ne répond plus), le système exhibe une défaillance par arrêt (Crash fault). Lorsque les conditions temporelles ne sont plus satisfaites (le système répond trop vite ou trop tard), le système exhibe une défaillance temporelle (Timing fault). Si le système exhibe chacun de ces comportements de manière imprévisible, il est affecté par des défaillances arbitraires (Byzantine fault).

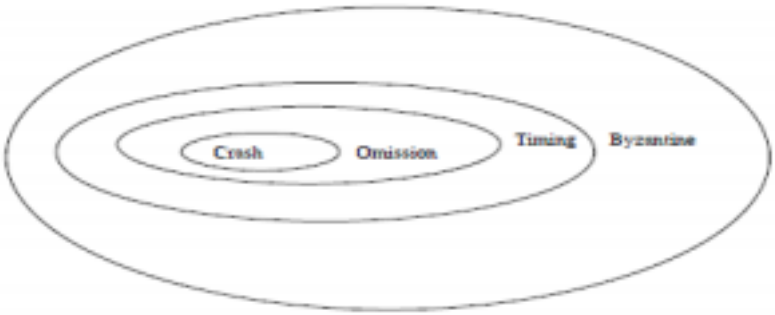


Fig.1.1. Classes de défaillance

- Techniques permettant d'atteindre la tolérance aux fautes : La tolérance aux fautes est mise en oeuvre par la combinaison de deux techniques.

1. **Le traitement de la faute** (fault treatment) qui vise à éviter qu'une faute survenue ne se reproduise.
2. **Le traitement d'erreur** (error processing) qui vise à éliminer une erreur avant qu'elle ne produise une défaillance (A.Schiper, February 2000)**.**

5. Architecture Client – Serveur ou Centralisée

L'environnement **client-serveur** désigne un mode de communication à travers un réseau entre plusieurs programmes ou logiciels : l'un, qualifié de client, envoie des requêtes ; l'autre ou les autres, qualifiés de serveurs, attendent les requêtes des clients et y répondent. Par

extension, le client désigne également l'ordinateur sur lequel est exécuté le logiciel client, et le serveur, l'ordinateur sur lequel est exécuté le logiciel serveur. (Wikimedia Foundation, 2013). La comparaison des architectures distribuées et client-serveur nous amènent à relever les avantages et Inconvénients suivants :

a. Client-serveur

• Avantages

− Toutes les données sont centralisées sur un seul serveur, ce qui simplifie les contrôles de sécurité, l'administration, la mise à jour des données et des logiciels. − Les technologies supportant l'architecture client-serveur sont plus matures que les autres.

− La complexité du traitement et la puissance de calculs sont à la charge du ou des serveurs, les utilisateurs utilisant simplement un client léger sur un ordinateur −

terminal qui peut être simplifié au maximum.

− Recherche d'information : les serveurs étant centralisés, cette architecture est − particulièrement adaptée et véloce pour retrouver et comparer de vaste quantité − d'informations (moteur de recherche sur le Web), ce qui semble être rédhibitoire − pour le P2P beaucoup plus lent, à l'image de Freenet.

• **Inconvénients**

− Si trop de clients veulent communiquer avec le serveur au même moment, ce dernier risque de ne pas supporter la charge (alors que les réseaux pair-à-pair fonctionnent mieux en ajoutant de nouveaux participants).

− Si le serveur n'est plus disponible, plus aucun des clients ne fonctionne (le réseau pair-à-pair continue à fonctionner, même si plusieurs participants quittent le réseau).

− Les coûts de mise en place et de maintenance peuvent être élevés.

− En aucun cas les clients ne peuvent communiquer entre eux, entrainant une asymétrie de l'information au profit des serveurs.

Une représentation graphique de l'Architecture Client – Serveur se présente comme suit :

Architecture Client - Serveur

Figure 2 :

b. Architecture Distribuée

• **Avantages**

- Extensibilité
- partage des données hétérogènes et réparties
- performances avec le parallélisme
- disponibilité avec la réplication

• **Inconvénients**

- administration complexe
- distribution du contrôle
- difficulté de migration

6. Disponibilité des Données dans les antennes de la commune : Les collectivités territoriales (locales), autrement appelées « des communes» constituent l'une de branches des services les plus connues par les citoyens (le public). Dans sa politique d'amélioration les services administratifs, la commune se retrouve face à une multitude des situations à gérer, entre autres : la gestion de la Relation Citoyen, la disponibilité des services mais aussi celle des données de sorte que l'information soit facilement accessible afin de répondre aux demandes d'information par les citoyens, en temps réel. Les antennes des communes ne font pas face qu'aux situations ci-haut mentionnées, mais ils se confrontent aussi à des problèmes liés au développement de la région dans son ensemble : le problème de connexion au réseau, le manque d'électricité.

Vu les coûts que cette solution exige, les communes à revenus limités (les communes les plus petit) se retrouvent incapables de l'envisager. Pour les relever, la solution la moins coûteuse serait de mettre en place une architecture distribuée qui puisse les permettre de

répliquer les données provenant de différentes bases de données reparties, tout en minimisant les coûts.

7. Conclusion

Les résultats fouillés des recherches ci-dessus n'ont pas surement couvert le sujet en entier étant donné que :

- Les algorithmes proposés ne tiennent pas compte de la fréquence de lancement des requêtes des utilisateurs, mais non plus de la consommation de la bande passante ;
- Les modèles proposés ne prévoient pas de situation de manque d'électricité et de connexion permanente au réseau privé (intranet) qui sont des facteurs clés dans la recherche de la cohérence entre les données répliquées.

De ce fait, cette affirmation hypothétique demeure : « Si on parvient à Instaurer un système de Réplication dans le but d'avoir en permanence les informations des autres bases, alors la Conception d'un Algorithme de Réplication des données selon la fréquence de

lancement des requêtes des utilisateurs serait une solution optimale pour minimiser les coûts d'achat de la bande passante. »

Chapitre 2

La réplication dans les systèmes répartis

2.1. Introduction

Les bases de données réparties et la réplication des données sont reconnues aujourd'hui comme moyens efficaces pour augmenter la disponibilité et la fiabilité des bases de données. De plus la réplication peut contribuer favorablement à l'amélioration des performances en utilisant les copies locales voire les copies plus proches.

Toutefois, ces avantages sont contraints par un problème majeur de cohérence mutuelle des copies. La gestion des copies en termes de propagation des mises à jour est ainsi nécessaire. La charge induite peut entraîner un impact significatif sur le système. Celle-ci ne doit pas altérer de façon excessive le temps de réponse global. En d'autres termes il s'agit de garantir la cohérence mutuelle dans les délais acceptables.

De plus, on peut noter que le temps nécessaire, pour qu'une mise à jour prenne effet sur toutes les copies, peut varier selon les méthodes de gestion. Les copies peuvent ainsi présenter un décalage les unes par rapport aux autres. Ce retard, appelé temps de latence, définit la période ou une donnée peut être utilisée (lecture) alors qu'elle ne reflète pas toutes les modifications antérieures de la base de données.

L'importance de la durée moyenne de ce temps de latence varie d'une application à l'autre. Certaines applications peuvent tolérer un temps de latence de quelques minutes, alors que d'autres exigent un temps de latence d'environ quelques secondes voire millisecondes.

Dans ce chapitre nous verrons comment la réplication est utilisée dans les systèmes répartis pour mettre en œuvre la tolérance aux fautes. Il rappelle quelques résultats dans la littérature concernant la réplication.

2.2. Stratégies de réplication

Cette section présente la notion de stratégie de réplication. Ces stratégies visent à garantir une **cohérence forte** (**STRONG CONSISTENCY**) entre les copies d'un objet répliqué. Informellement ceci revient à assurer que l'état de chaque copie soit identique. La réplication passive et la réplication active sont deux stratégies de références.

2.3. Réplication active

La réplication active (active réplication ou state machine approche) se définie (Wiesmann, F.Pedonne, & A.Schipper, 1999) par la symétrie des comportements des copies d'un composant répliqué. Chaque copie joue un rôle identique à celui des autres. **Principe**

− réception des requêtes : toutes les copies reçoivent la même séquence. − traitement des requêtes : toutes les copies traitent les requêtes de manière déterministe.

− émission des réponses : toutes les copies émettent la même séquence de réponses.

La figure 2.1 illustre ce principe à l'aide d'un diagramme temporel.

Les flèches horizontales représentent l'exécution de quatre composants : Client, S1, S2, S3.

Les Si sont les copies du composant répliqués S. Elles appartiennent à un même groupe. Lorsque le client invoque S, il envoie une requête (traits en flèche continu) à tous les Si à l'aide d'un ABCAST assurant l'ordre total et la propriété d'atomicité.

Chaque Si traite la requête (petit rectangle horizontal) et renvoie une réponse au client.

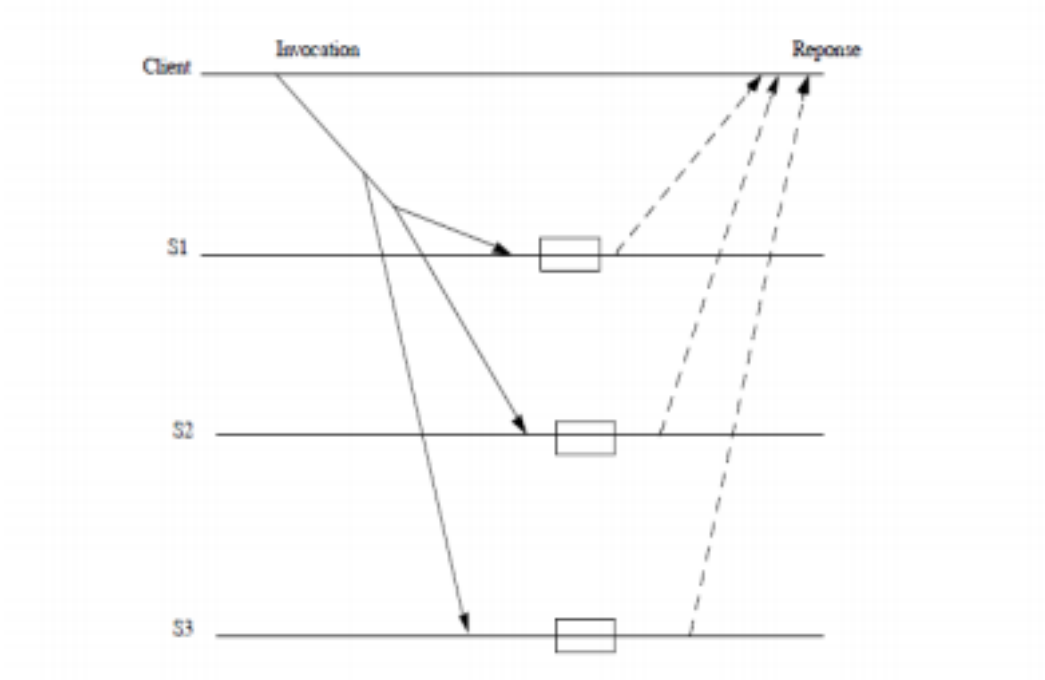


Fig.2.1. Principe de la réplication active

2.3.1. Tolérance aux fautes

La réplication active (Solidor) où chaque copie est répliquée et exécutée simultanément sur n machines distinctes. Les répliques doivent synchroniser leurs exécutions à chaque fois qu'un message est reçu ou envoyé afin d'assurer que toute réplique réalisant un même calcul reçoive les messages provenant des autres processus dans le même ordre. Plusieurs modes de défaillance d'une réplique existent, allant de la défaillance par arrêt (défaillance la plus simple) à la défaillance byzantine (défaillance la plus complexe). Dans le cas de défaillance par arrêt, une réplique défaillante ne produit plus aucun résultat. Dans le cas de défaillances byzantines, une réplique défaillante continue de produire les résultats mais ceux-ci sont erronés. La tolérance aux fautes est assurée par masquage d'erreur.

La défaillance d'une copie est masquée par le comportement des copies non défaillantes. Comme chaque copie joue un rôle identique. La défaillance de l'une d'entre elle ne perturbe pas le service fourni par le composant.

2.4. Réplication passive

La réplication **passive** distingue deux comportements d'un composant répliqué : la copie primaire (primary copy) et les copies **secondaires** (backups). La copie primaire est la seule à effectuer tous les traitements. Les copies secondaires, oisives, surveillent la copie primaire. En cas de défaillance de la copie primaire, une copie secondaire devient la nouvelle copie primaire.

Principe

La réplication passive est définie ainsi (Wiesmann, F.Pedonne, & A.Schipper, 1999)***:*** − réception des requêtes : la copie primaire est la seule à recevoir les requêtes ; − traitement des requêtes : la copie primaire est la seule à traiter les requêtes ; − émission des réponses : la copie primaire est la seule à émettre les reposes ;

La figure 2.2 illustre ce principe. Le client envoie la requête uniquement à la copie primaire S1. Celle-ci traite la requête, construit un point de reprise et l'envoie à l'aide d'un multicast fiable assurant l'ordre FIFO, aux copies secondaires S2 et S3 en précisant le changement de son état (message update). Après la mise à jour de leurs états, les copies secondaires envoie un ack à la copie primaire. La copie primaire envoie la réponse au client après réception des "ack" de toutes les copies secondaires. Le point de reprise permet de synchroniser l'état des copies

secondaires avec celui de la copie primaire puisque celle-ci est la seule qui communique avec le reste du système.

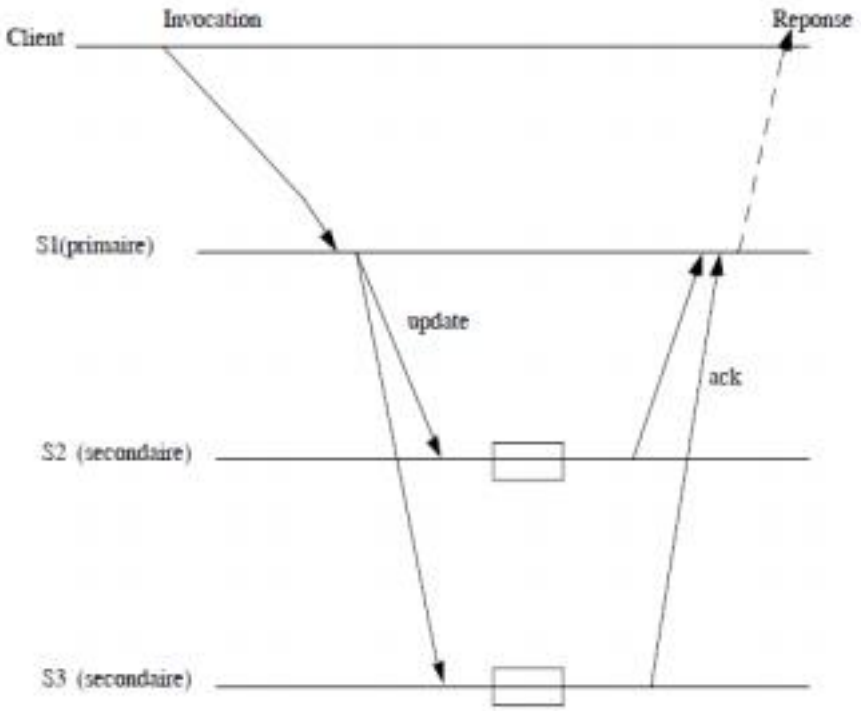


Fig.2.2. Principe de la réplication passive

2.4.1. Tolérance aux fautes

La réplication passive où un composant logiciel est répliqué en n exemplaires, mais une seule des n répliques effectué le calcul. Les n-1 autres répliques sont passives et ne prennent la relève que si la réplique active est défaillante. Pour que cette stratégie fonctionne, il est nécessaire que la réplique active transmette aux répliques passives, à intervalles réguliers son état d'exécution. Cet état d'exécution composé d'une pile, de données et de registres est stocké par chacune des répliques passives et constitue un **point de reprise**. Si la réplique active est défaillante, une des répliques passive est activée et reprend l'exécution du calcul à partir du dernier point de reprise enregistré. On dit que le processus effectué un retour arrière. Recréer un état d'exécution simule une remontée dans le temps en ramenant le calcul dans l'état qu'il occupait avant la manifestation de la défaillance.

La tolérance aux fautes est réalisée par détection et compensation de l'erreur. La défaillance d'une copie secondaire ne nécessite aucun traitement particulier. Par contre, la défaillance de la copie S1 implique que les copies secondaires désignent l'une d'entre elle (par exemple S2) comme la nouvelle copie primaire. Les cas suivants peuvent se présenter.

- La défaillance de la copie primaire avant l'envoie du message "update" a pour conséquence que le client n'obtienne aucune réponse à la requête. Tout le traitement de la requête effectué par la copie primaire S1 est perdu. Le client doit alors réémettre sa requête en l'adressant à la nouvelle copie primaire S2. Cette dernière doit être en mesure de construire la réponse que le client attend.
- Deux situations peuvent se présenter selon que la défaillance de S1 ait été détectée pendant l'envoie du message "update" aux copies secondaires ou avant l'envoi de la réponse au client. Ceci est le cas le plus difficile à appréhender :
  1. L'Atomicité doit être garantie : le message " update" doit être reçu par tous ou par aucune des copies secondaires.
  2. Le client doit réémettre sa requête au nouveau primaire.
- La défaillance du primaire S1 a lieu après avoir envoyé la réponse. Dans ce cas un nouveau primaire doit être désigné.

Pour la désignation du primaire et l'assurance de l'atomicité, nous renvoyons le lecteur à (A.Schiper, February 2000).

2.5. Réplication semi-active

La réplication semi-active (semi-active réplication) (M. Wiesmann, 2000) se situe à mi chemin entre la réplication active et la réplication passive. Contrairement à la réplication passive, les copies secondaires ne sont pas oisives. La copie primaire est appelée leader et les copies secondaires sont appelées suiveurs.

Principe

La réplication semi-active est défini ainsi (M. Wiesmann, 2000)

- réception des requêtes : toutes les copies reçoivent le même ensemble3 de requêtes.
- traitement des requêtes : toutes les copies traitent toutes les requêtes. La copie primaire traite une requête dès qu'elle la reçoit. Par contre, une copie secondaire doit attendre une notification de la copie primaire pour pourvoir traiter une requête ;
- émission des réponses : la copie primaire est la seule à émettre les réponses.

La figure 2.3 illustre ce principe. Le client envoie une requête à tous les Si. La copie primaire S1 envoie une notification nitify aux copies secondaires et commencent le traitement de la requête. Les copies secondaires S2 et S3 ne commencent à traiter la requête qu'après avoir reçu la notification de la copie primaire.

Sitôt le traitement terminé, S1 envoie la réponse au client.

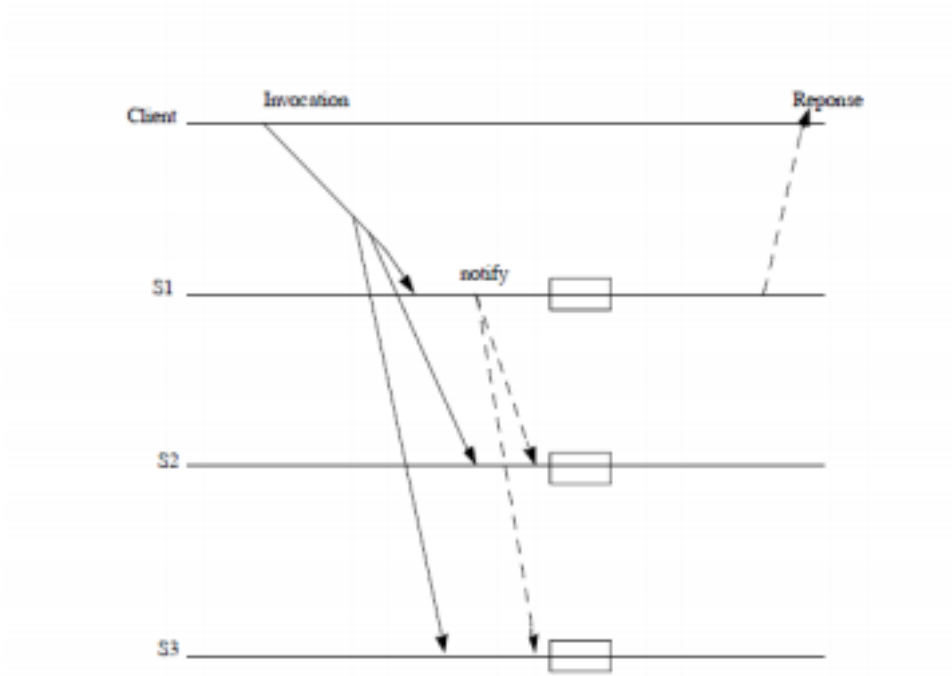


Fig.2.3. Principe de la réplication semi-active

2.5.1 Tolérance aux fautes

La tolérance aux fautes est réalisée par détection et compensation de l'erreur, comme dans le cas de la réplication passive. Cependant comme toutes les copies reçoivent la requête, le client n'a pas besoin de réémettre la requête lorsque le primaire S1 défaille. La nouvelle copie primaire S2 envoie automatiquement la réponse au client. Deux situations peuvent se présenter suivant la défaillance du primaire S1 est détectée par les copies secondaires avant ou après la notification.

Si la défaillance de la copie primaire est détectée avant la réception de la notification, la nouvelle copie primaire S2 envoie une notification concernant la première requête présente dans sa queue et la traite normalement. Si la défaillance de S1 est détectée après la réception de la notification, S2 traite la requête correspondante sans envoyer de notification et envoie la réponse au client.

2.6. Conclusion

Lors de la conception d'un système de gestion de donnes distribué, toutes ces contraintes doivent être prises en considération. Par exemple, le choix du nombre des copies d'une donnée doit prendre en compte le niveau de tolérance aux fautes souhaité, la popularité de la donnée et le niveau de cohérence requis par l'application. Lors du placement des copies, là encore on considérera la localisation des accès, la localisation des autres copies pour les aspects liés à la tolérance aux fautes (degré d'éparpillement, fautes corrélées) et les contraintes de cohérence.

C'est pourquoi nous nous proposons d'étudier la gestion de données en considérant conjointement tolérance aux fautes, performance et cohérence.

Chapitre 3

METHODES ET TECHNIQUES DE RECHERCHE

Pour répondre aux questions de notre mémoire, nous devrons avoir une marche à suivre, autrement dit un ensemble d'étapes, une méthode. Mais pour un système aussi complexe, une seule méthode ne sera pas suffisante, puisqu'il s'agit non seulement d'instaurer un système de réplication de bases de données permettant d'avoir en permanence, dans une base quelconque, les informations enregistrées sur d'autres bases, mais aussi de concevoir et tester les algorithmes en tenant compte de la fréquence de demandes des utilisateurs pour minimiser le coût de la bande passante ; raison pour laquelle nous avons combiné différentes méthodes et techniques suivantes :

3.1. MODELISATION

De manière abstraite, la réplication des bases de données, sera représentée, dans ce travail, avec le langage de modélisation **UML** (Unified Modeling Langage) étant un langage de modélisation orienté-objet utilisé dans la conception des logiciels informatiques employant des concepts tels que : objet, messages, classe, héritage, polymorphisme. Elle implémente, pour ce faire, les formalismes tels que le modèle de classe, le modèle d'objets, le modèle des états, le modèle des cas d'utilisation, le modèle d'interaction, le modèle de réalisation (**Muyisa FRED**, 2011). Aussi, avons-nous représenté la réplication des données selon les trois points de vue classiques de modélisation : fonctionnel, statique et dynamique, en insistant sur les diagrammes prépondérants.

Graphiquement, la modélisation à faire se résume dans la figure suivante : **Fig. 3.1** Trois axes de modélisation UML (Pascal Roques, 2006)

- **Du point de vue Fonctionnel**, nous avons tracé un Diagramme de cas d'Utilisation pour définir les acteurs qui interagissent avec le système de réplication, mais aussi un Diagramme d'Activité pour décrire le scénario nominal des cas d'utilisation (des actions émis par les acteurs sur le système et vis versa). Ces deux diagrammes renseignent sur le fonctionnement des acteurs et du système de réplication lui-même.
- **Du point de vue Statique**, nous avons représenté le diagramme de Classes étant le point central dans un développement orienté objet qui nous a permis de décrire la structure des entités-objets manipulés par les utilisateurs.
- Du **point de vue Dynamique**, nous avons réalisé un diagramme de communication particulier (diagramme de contexte dynamique) sous forme de diagramme de séquence système afin de répertorier tous les messages que les acteurs peuvent envoyer au système de réplication et recevoir de lui ; et un diagramme d'états pour décrire avec précision les comportements complexes. Dans le même contexte, nous avons créé une **machine de Turing** (appelée **machine à états** finis en UML) qui consiste à s'intéresser au cycle de vie d'une instance générique d'une classe particulière au fil de ses interactions avec le reste du monde, dans tous les cas possibles. Cette vue locale d'un objet (un enregistrement dans notre cas), qui décrit comment il réagit à des événements en fonction de son état courant et comment il passe dans un nouvel état (durant la réplication), sera représentée, dans ce travail comme dans beaucoup d'autres, graphiquement sous la forme d'un **diagramme d'états**.

3.2. DEFINITION DES ALGORITHMES

Il ne s'agit pas ici de programmer avec un langage ou un autre, mais bien de raisonner sur le problème afin de concevoir une solution abstraite. Ce travail de réflexion et de conception prépare le stade ultime de l'implémentation et du cycle de vie du programme concret. Un algorithme, dans sa définition simple, est une suite d'instructions qui, quand elles sont exécutées correctement aboutissent au résultat attendu.

C'est un énoncé dans un langage clair, bien défini et ordonné qui permet de résoudre un problème, le plus souvent par calcul. Cette définition est à rapprocher du fonctionnement de la machine de Turing qui avant l'apparition de l'ordinateur utilisait cette démarche pour résoudre de nombreux problèmes. L'algorithme est donc une recette pour qu'un ordinateur puisse donner un résultat donné.

Il décrit formellement ce que doit faire l'ordinateur pour arriver à un but bien précis. Ce sont les instructions qu'on doit lui donner. Ces instructions sont souvent décrites dans un

langage clair et compréhensible par l'être humain : faire ceci, faire cela si le résultat a telle valeur, et ainsi de suite.

Fig. 3.2 De la Réflexion à la Programmation

Nous avons fait recours aux techniques algorithmiques qui nous permettront d'analyser le problème en quatre phases : la Spécification ou cadrage, la Description du traitement ou des opérations en utilisant le Langage de Description Formelle des Algorithmes (LDFA), la traduction des algorithmes dans un langage de programmation et enfin la programmation et le jeu de test des algorithmes que nous avons exécuté dans une Machine de Turing.

3.3. PROTOTYPAGE

Une fois la modélisation et la définition des algorithmes finies, nous avons retracé une maquette du nouveau système : un prototype, dans le but de projeter ce à quoi va correspondre le système à la fin de notre recherche. C'est à niveau que nous essayerons de résoudre les problèmes que les utilisateurs du système auront ciblé.

Ce processus pourra nous aider à véhiculer l'expérimentation, et sa construction fournira un nouvel aperçu sur le modèle prototypé. Voilà pourquoi il nous faudra avoir un style qui traitera de la manière dont les solutions aux problèmes ou les algorithmes seront formulées. Cette phase est beaucoup plus utile étant donné que c'est la partie qui intéresse les bénéficiaires finaux du système à mettre en place.

3.4. EXPERIMENTATION

Bernard MORAND a écrit : « Un bon logiciel n'est ni celui dont on peut exhiber la preuve, ni celui qui fait ce qui est demandé, mais celui dont les utilisateurs se servent encore trois ans après sa création ».

L'expérimentation nous a permis de tester notre modèle et d'observer les effets afin de soumettre le système réalisé à un ensemble d'expériences et d'opérations destinées à l'étude et au test qui, dans ce travail de recherche, seront conduits dans un laboratoire informatique. Il est à signaler que l'expérimentation s'est fait selon la complexité des algorithmes écrits pour

dégager les résultats attendus sous la contrainte du temps d'exécution et de ressources mémoires exigées (espace).

Hormis le test de la complexité algorithmique, il faudra expérimenter la nouvelle application dans un environnement réseau, ce qui revient à dire qu'il faudra tester la réplication entre base des données installées sur deux ou trois sites distants afin d'être sûr que les questions de la recherche avaient été partiellement ou totalement satisfaites.

3.5. SIMULATION

A défaut d'un environnement pour l'expérimentation, nous avons créé nous-mêmes des environnements des tests afin de dégager plus rapidement les résultats attendus du système. Il s'agira d'un environnement imaginaire favorable pour réplication des données entre les bases de données installées sur différents serveurs. La simulation sera faite de sorte que les serveurs s'échangent continuellement les rôles comme dans un réseau pair à pair. Un site sera donc à la fois maitre et esclave.

Chapitre 4
CONCEPTION DE LA SOLUTION

Dans ce chapitre, nous nous emploierons dans un premier temps à concevoir un système d'information simple des collectivités locales, qui est, comme signalé dans l'introduction de ce travail, le champ de notre travail de recherche.

Dans un second temps, nous un système de réplication de base de données avec UML, que nous appliquerons sur la base de données de l'institution d'Etat (Wilaya, Communes) précédemment modélisée.

4.1. Modélisation informatique d'une collectivité locale : La

transformation rapide et continue de la société algérienne a généré de nouveaux besoins de citoyens dans tous les domaines, ce qui a dû la nécessité de reconsidérer le travail de l'administration pour l'adapter à ces besoins et prendre en charge toutes les étapes de cet ambitieux développement.

Afin de concrétiser cela dans la réalité, notre recherche est prise de nombreuses mesures visant à éliminer les déséquilibres qui perturbent certains des intérêts des citoyens en raison du traitement manuel classique des dossiers administratifs et du manque de la transparence dans la gestion et la bureaucratie et le grand nombre de documents qui composent les dossiers et autres, qui sont autant de facteurs qui ont contribué aux étapes précédentes, à l'impact négatif sur la crédibilité de la relation entre l'administration et le citoyen.

Parmi ces mesures figurent la mise en place des conditions appropriées pour la modernisation de l'administration et le passage progressif de la phase de gestion classique à la phase de gestion automatisée en réformant les installations de l'administration centrale et en gérant les groupes locaux et en introduisant des technologies modernes dans les domaines de la gestion et de l'organisation et rétablissant ainsi la considération de l'utilité publique et en améliorant ses performances en permanence et en créant des modèles de travail nouveaux et modernes basés essentiellement sur l'optimisation des technologies émergentes.

Et la mise en place de bases de travail modernes soutenues par des ressources humaines qualifiées. Notre projet va créer une fenêtre électronique pour les documents juridiques pour corriger des actes d'état civil à distance via des annexes de la commune, et elle sera progressivement étendue à d'autres documents. La fenêtre électronique des documents juridiques est une solution technique qui reçoit les demandes de divers actes d'etat civil(Registre de Naissance, Registre de Mariage, Registre de décès) et les enregistre en temps réel dans une base de données centrale, en utilisant un lien direct avec la base de données de

registre de l'état civil. Son travail est basé sur la comparaison automatique de la base de données de registre de l'état civil précédemment remplis avec les images scannées du registre de l'état civil, et ainsi cette comparaison permettra de vérifier en temps réel la validité des informations liées à la demande de correction. Une expérience a été faite pour exploiter ce style de gestion moderne sur le terrain, et il a prouvé son efficacité au niveau de la commune de SAYADA de la wilaya de Mostaganem.

4.2. Modélisation Fonctionnelle

Diagramme des Cas d'Utilisation

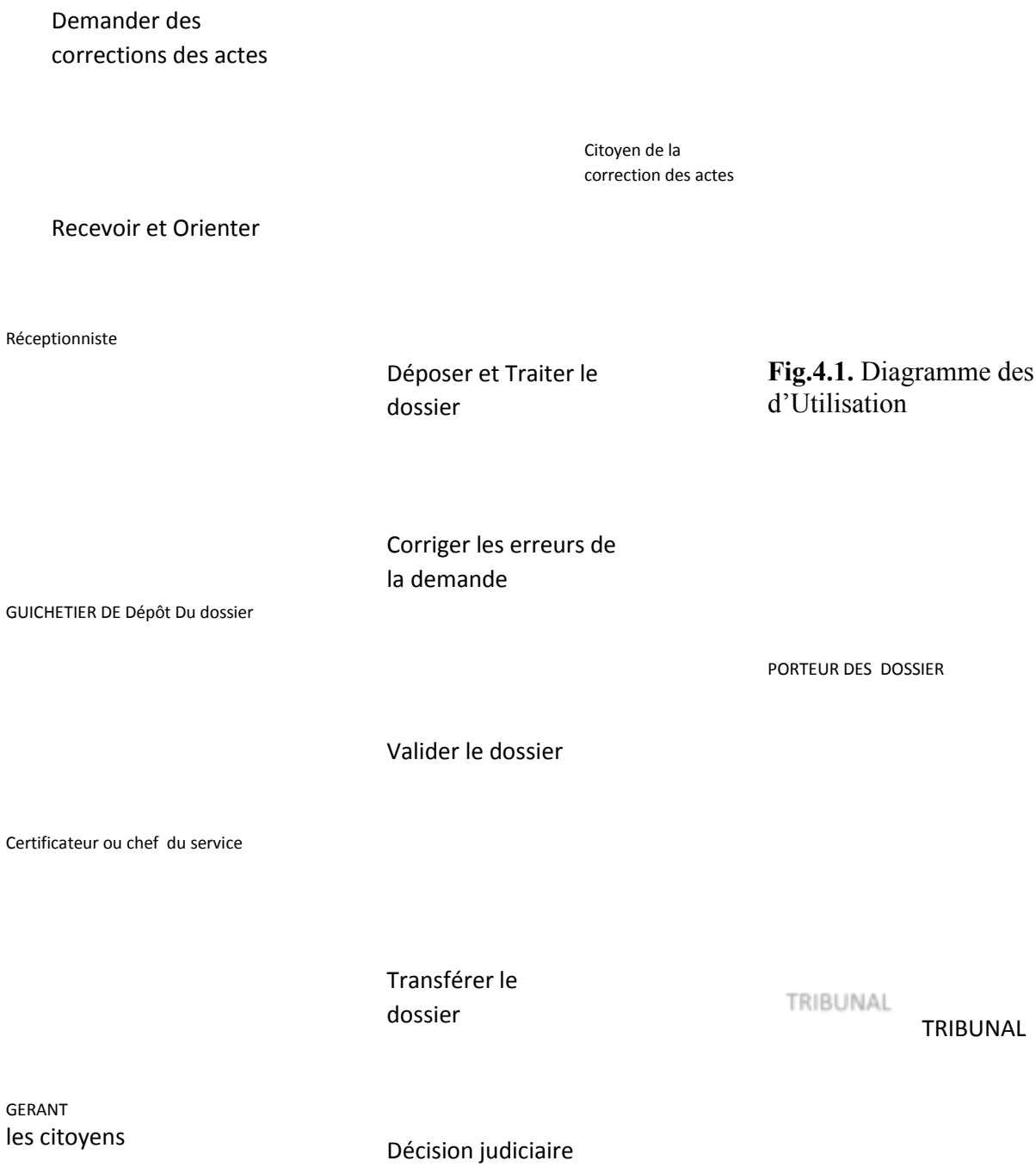


Fig.4.1. Diagramme des Cas d'Utilisation

Identification des Acteurs

- **Le Réceptionniste** : il reçoit et oriente les citoyens vers divers d'autres acteurs (guichetier de dépôt, chef du service) et même chez le gérant en cas refus du dossier par exemple.

- Le Guichetier de Dépôt : les tâches chargées de la réception du dossier sont : • Vérifier les documents qui composent le dossier administratif du

demandeur selon la situation.

• Vérifier les informations de la personne (Pieces d'identité).

• Vérifier toutes les informations avant d'enregistrer la demande.

• Imprimez le formulaire de demande et soumettez-le à la personne concernée pour vérifier les informations enregistrées .

• Confirmation de la demande et remise du récépissé de dépôt de la personne concernée

- **Le Guichetier de certification :** il vérifié les informations saisies avant d'envoyer au gérant.
- **Le Gérant** : c'est le chef des services ; il coordonne les services, il analyse les demandes de prêt et centralise les rapports émis par les services qu'il gère.

Diagrammes d'Activités : Dépôt de la demande des corrections des actes d'Etat Civil :

Début

Formuler un dossier Réception et Orientation

Demande Rectification des actes d'Etat Civil

Refusée

Annulation

Acceptée

Dépôt d'un dossier

non

ok

Validation par le gérant envoyer

Tribunal verifier Finadministratif certifier

Certification d'un dossier **Fig.4.2.** Diagramme d'Activités

4.2. Modélisation d'un système de réplication *4.2.1. Présentation de la réplication de base de données*

La réplication est une puissante fonctionnalité des SGBD (SQL Serveur, Oracle, MySQL, Access, ...) qui permet de distribuer les données et d'exécuter les procédures stockées sur plusieurs serveurs de l'entreprise. La technologie de réplication a considérablement évolué et permet maintenant de copier, déplacer les données à différents endroits et de synchroniser automatiquement les données. La réplication peut être mise en œuvre entre des bases de données résidant sur le même serveur ou sur des serveurs différents. Les serveurs peuvent être sur un réseau local (LAN), réseau global (WAN) ou sur Internet.

On distingue deux catégories de réplication :

- La réplication de serveur à serveur
- La réplication de serveur à clients

Dans le cas de la réplication de serveur à serveur, la réplication permet une meilleure intégration ou rapprochement des données entre plusieurs serveurs de base de données. L'objectif de ce type de réplication est d'effectuer un échange d'informations entre des serveurs de base de données. Les utilisateurs qui travaillent sur les bases qui participent à la réplication peuvent ainsi consulter des données de meilleure qualité.

La réplication de serveur à clients concerne principalement les utilisateurs déconnectés du réseau de l'entreprise et qui souhaitent travailler avec tout ou partie des données de l'entreprise.

4.2.2. Modélisation fonctionnelle

1. Un système de réplication optimiste est constitué d'un ensemble de sites interconnectés par un réseau.
2. Chaque site possède une copie des objets partagés par exemple des documents textuels ou des images.
3. Sur un site, une réplique peut être modifiée au moyen d'opérations.
4. Quand une réplique est modifiée sur un site, l'opération correspondante est immédiatement exécutée sur ce site, puis propagée aux autres sites pour y être réexécutée.
5. Lorsque deux répliques de deux sites différents sont modifiées en parallèle, les répliques divergent.

6. Il est donc possible d'observer au même moment une valeur sur un site et une autre valeur sur un autre site. Les algorithmes de réplication doivent assurer la convergence des répliques.
7. Le système doit être convergeant quand le système est au repos, c'est-`a-dire quand toutes les opérations ont été propagées à tous les sites.

Fig. 4.1 Représentation du Système Réparti considéré dans la Règle de Thomas (Pascal MOLLI et All, 2005)

4.2.3. Principes de la réplication

Le principe de la réplication, qui met en jeu au minimum deux Bases de données installées sur des sites distants, est assez simple et se déroule en trois temps :

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

b. Les modifications faites sur les données sont détectées et stockées (dans une table, un fichier, une queue) en vue de leur propagation.

c. Un 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.

Toutefois, il est possible de faire de la réplication dans les deux sens (de l'esclave vers le maître et inversement). On parlera dans ce cas-là de réplication bidirectionnelle ou symétrique. C'est le type de réplication que nous comptons représenter dans cette partie.

Réalisation du diagramme des cas d'utilisation

Fig. 4.2 Diagramme des Cas d'utilisation de la Réplication

Identification des acteurs

• Les Acteurs Principaux

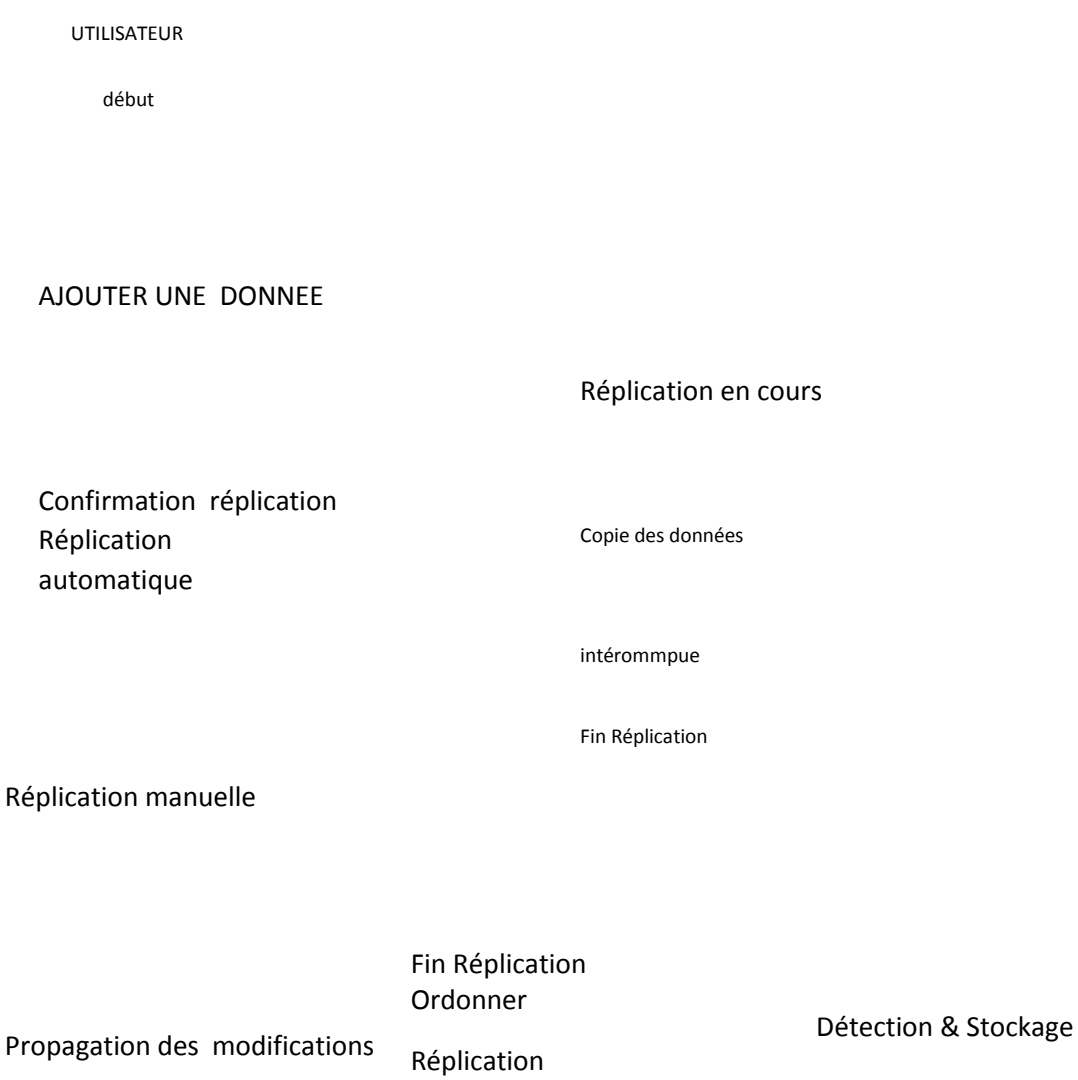
- Base de Données 1 : la base de données sur laquelle s'effectuent les modifications. On l'appelle autrement, Base Maitre. Elle est installée sur un serveur. **- Base de Données 2** : la base de données qui attend les modifications apportées sur la base de données installée sur l'autre site grâce à la réplication. C'est la Base Esclave.

- Base de Données N : représente le reste des bases de données réparties sur lesquelles la réplication sera exécutée.

• Les Acteurs Secondaires

- Utilisateur : c'est l'utilisateur de la base de données. Il déclenche le processus de la réplication manuelle. Cette dernière prend effet lorsque le processus de la réplication automatique n'a pas abouti ou quand l'heure prévue pour la réplication n'a pas encore sonné.

Diagramme d'Activités de Réplication



FIN **Fig. 4.3** Diagramme d'Activités de la Réplication

4.2.4. *Modélisation Statique :*

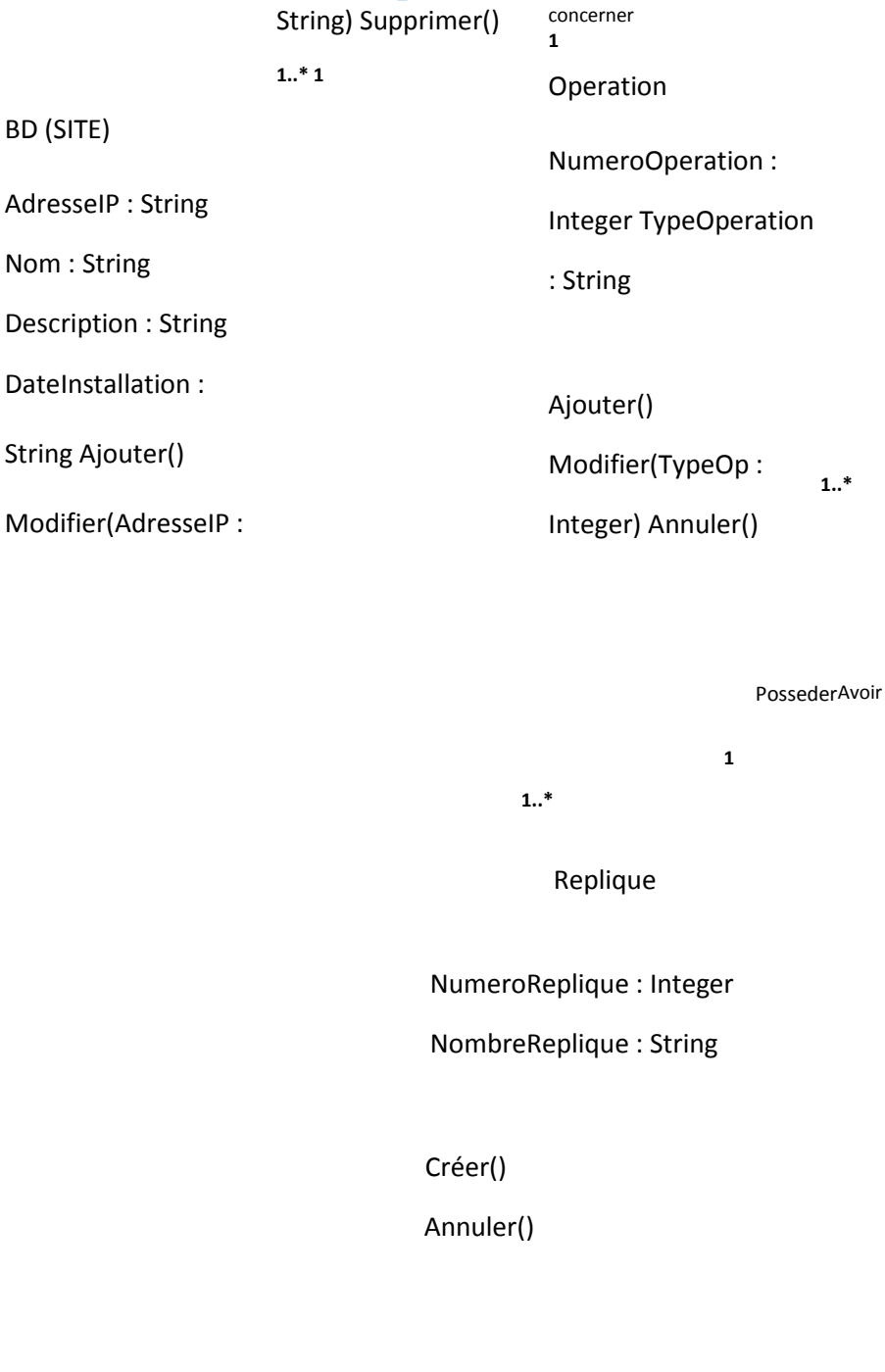


Fig. 4.5 Diagramme des Classes de la Réplication

4.2.5. Modélisation Dynamique :
Diagramme de Séquence Système

BD1 BD2 BDNSRBDR

Réplication

Automatique

Opération de

copie

REPLICATION EN COURS

Ordonner Mise à jour

DETECTION & STOCKAGE

UTILISATEUR

Ajouter Donnée

Affichage des modifications

Fig. 4.6 Diagramme de Séquence Système de la Réplication

Diagramme d'états : BD non Répliquée et BD Répliquée :

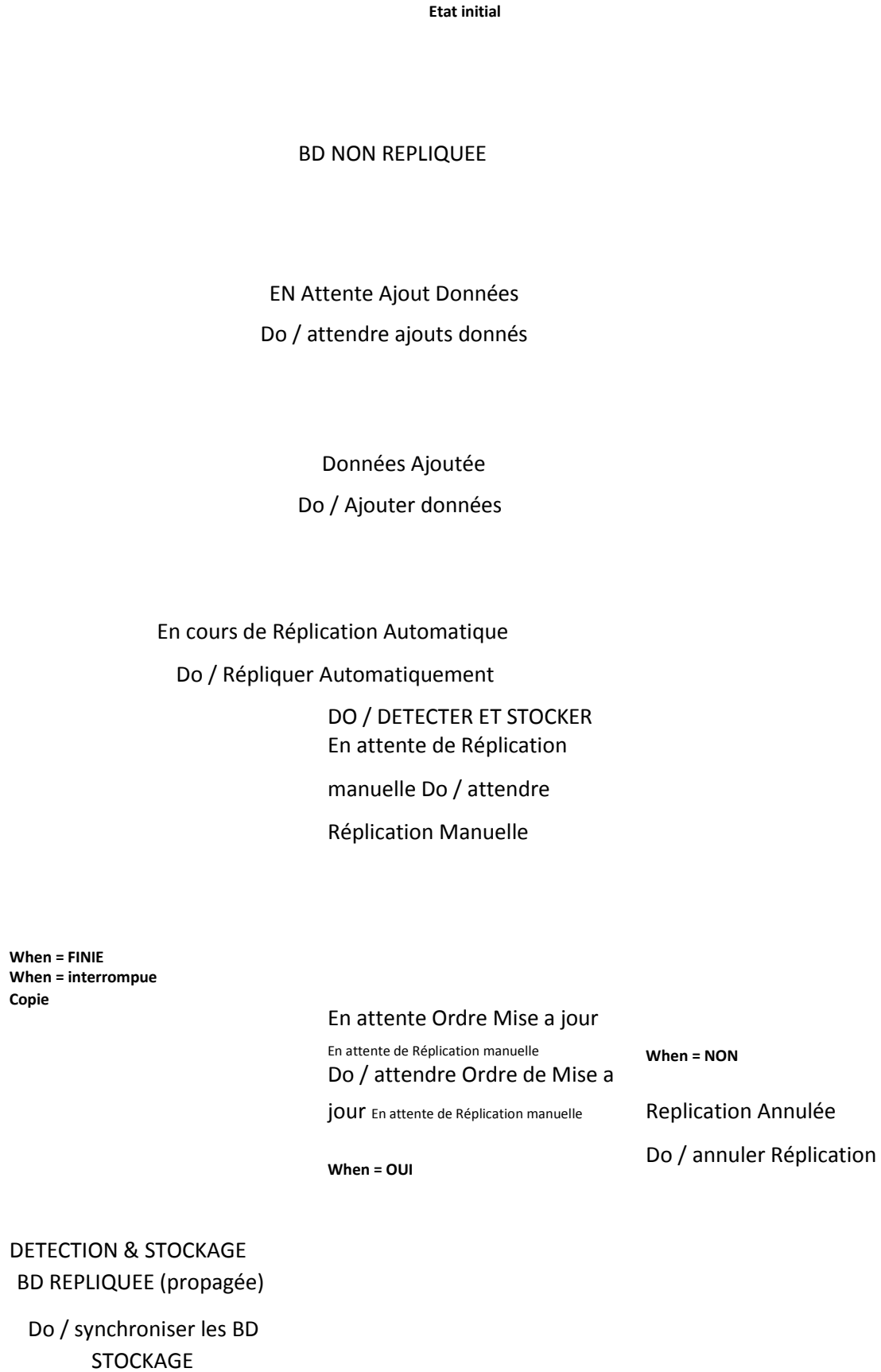


Fig. 4.7 Diagramme d'états du Système de Réplication d'une Base de données

Exécution de l'algorithme de réplication sur la Machine de Turing Dans les diagrammes d'activité, de séquence système et d'états du système de réplication, nous avons décrit les algorithmes de réplication. Ces algorithmes offrent la possibilité de répliquer manuellement les informations en cas d'interruption lors de la propagation des données. La figure suivante décrit l'exécution de ces algorithmes sur la Machine de Turing:

Fig. 4.8 Exécution des Algorithmes de réplication sur la machine de Turing

Cette figure représente la Machine de Turing qui est un ordinateur virtuel décrivant le flux de traitement des informations en précisant les différents états aux quels passent ces états lors de l'exécution des algorithmes dans le système. Les résultats des tests de nos algorithmes ont produit des automates Déterministes et Non Déterministes à la fois. Pour certains états, par exemple l'état Non Répliqué et BD répliquée, on sait déjà le prochain état par lequel on va passer, c'est le cas d'un automate déterministe. Mais pour les autres états, comme en cours de Réplication, on ne sait pas encore si on passera à l'état BD répliquée ou En cours de Réplication Manuelle, tant qu'on n'a pas quitté l'état actuel : c'est la situation d'un automate Non Déterministe.

Chapitre 5
PRESENTATION DES RESULTATS

Partant de notre affirmation hypothétique formulée comme suit : « Si on parvient à Instaurer un système de Réplication dans le but d'avoir en permanence les informations des autres bases, alors la Conception d'un Algorithme de Réplication des données selon la fréquence de lancement des requêtes des utilisateurs serait une solution optimale pour minimiser les coûts d'achat de la bande passante. », nous avons abouti à des résultats que nous comptons vous présenter dans cette dernière partie de notre travail de recherche. Pour exécuter les algorithmes que nous vous avons présentés dans le chapitre précédent sous forme de diagrammes d'activité, de séquence système et d'états, nous avons fait recours à trois méthodes : le Prototypage, la Simulation et l'Expérimentation.

5.1.PROTOTYPAGE

Le prototypage nous a été d'une grande importance puisqu'il nous a permis de connaitre d'avance à quoi ressemblerait le système à concevoir.

Le diagramme des classes modélisé à partir de règles de gestion et de traitement établis au début du chapitre précédent a dégagé le diagramme des relations suivantes :

5.1.1. PROTOTYPE D'UN DIAGRAMME

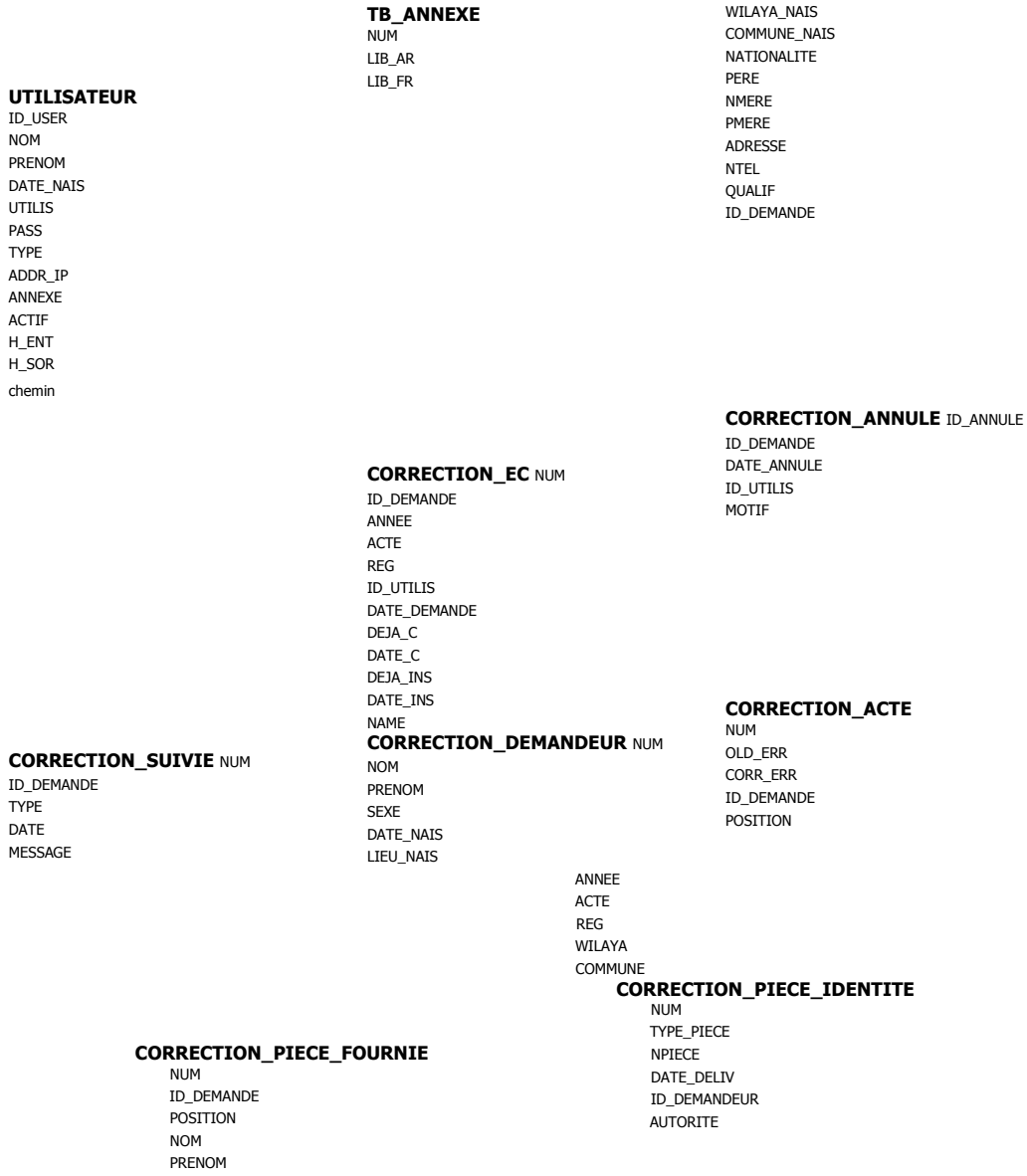


Fig. 5.1. Prototype d'un Diagramme des Relations d'une collectivités locales

Le diagramme de classes du système de réplication a généré le diagramme des relations suivant :

5.1.2. PROTOTYPE D'UN DIAGRAMME DES RELATIONS D'UN SYSTEME DE REPLICATION

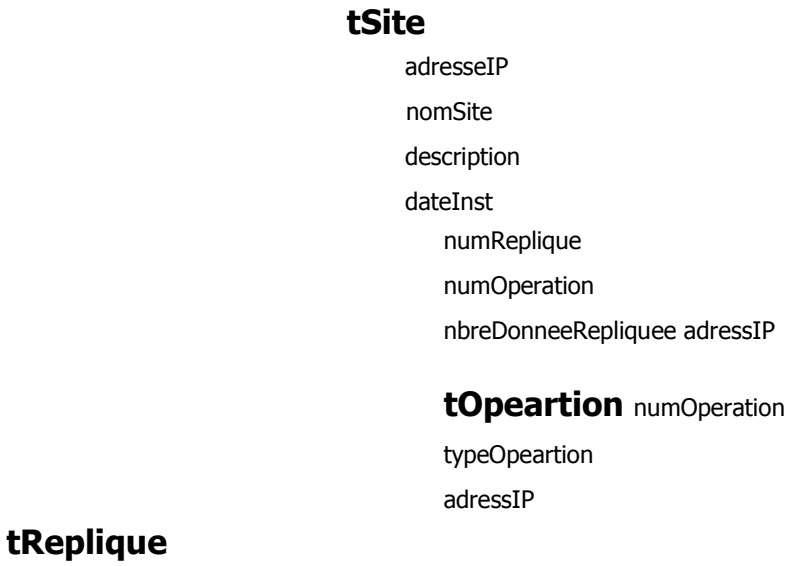


Fig.5.2 Prototype d'un Diagramme des Relations d'un Système de réplication

5.1.3. ENVIRONNEMENT DE TRAVAIL

Pour mettre en place le système de réplication de la base de données d'une institution financière, nous sommes passés par les étapes suivantes :

- Sur deux ordinateurs de 4GB de RAM et de 2 GHz de processeur, nous avons installé Windows Serveur 2008 édition Entreprise, qui est l'environnement exigé par les éditions Entreprise de SQL Serveur ;
- Pour mettre en réseau les ordinateurs, nous avons utilisé un Switch D-Link 16-Port 10/ 100 Desktop.
- Nous y avons installé SQL Serveur Management Studio Entreprise 2008 pour nous permettre la réplication de base de données ;
- Nous y avons installé Microsoft Visual Studios 2008, pour créer des interfaces pouvant faciliter la saisie des informations de tests pour la réplication des données, que nous avons connecté avec SQL Serveur ;
- pour toutes les configurations, l'environnement idéal pour la mise en place de la réplication de bases de données distribuées est décrit sur la figure suivante :

Fig.5.3 Environnement idéal pour la réplication de bases de données Réparties

SQL Serveur fournit trois types de réplication des données, la méthode : par capture instantanée, transactionnelle et par fusion. Chacun des types de réplication répond à un besoin bien précis.

5.1.4. PROCEDURES DE MISE EN PLACE DE LA REPLICATION

Réplication de Fusion

Il s'agissait, dans cette méthode, de surveiller les modifications de la base de données source et de synchroniser les valeurs entre l'éditeur et les abonnés, ces derniers pouvant tous effectuer des opérations de mise à jour sur les données distribuées. Si l'éditeur conserve la maîtrise de la publication, ce n'est pas toujours les opérations effectuées sur l'éditeur qui prennent le pas sur celles effectuées sur l'abonné. Toutes les modifications apportées à la base cible étaient reportées dans la base source.

Nous n'avons pas eu beaucoup du mal à configurer la synchronisation des données à partir de SQL Server Management Studio car il propose différents assistants graphiques pour mettre en place, surveiller et paramétrer l'environnement de réplication. Tous les éléments de configurations sont accessibles depuis le nœud **Réplication** de l'explorateur d'objets. Les algorithmes suivants décrivent les procédures de la mise en place de la réplication entre les serveurs de base de données :

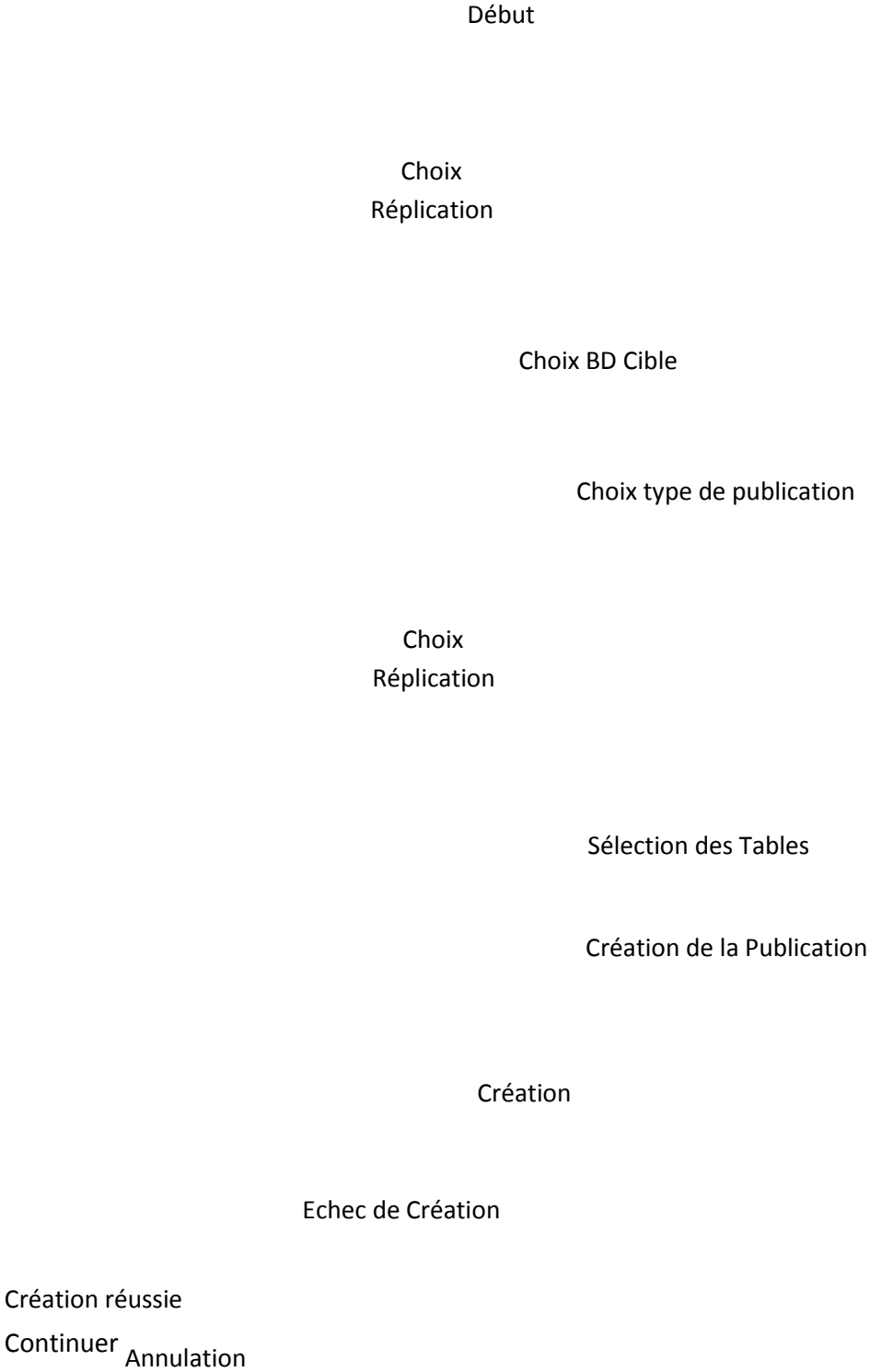


Fig.5.4 Algorithme de sélection du Noeud réplication pour commencer les configurations de mise en place

Réplication Transactionnelle

Après nos tests par la méthode transactionnelle, nous avons constaté que cette méthode est facile à mettre en place parce qu'elle ne nécessite pas de passer à des configurations au niveau de l'agent de publication et de l'agent de souscription ; c'est-à-dire que une fois qu'une opération est effectuée sur la base maitre, les changements s'appliquent en temps réel sur les bases esclaves.

5.1.5. INTERFACE HOMME – MACHINE

L'application servira de monitoring de la réplication. Nous avons souhaité mettre en place le prototype de ce qui serait une vraie application intégrant des fonctionnalités avancées des bases de données comme la Réplication des données, le Back Up et le Compactage. Pour nos tests, nous n'avons travaillé qu'avec quelques tables de la base de données que nous avons nommé GE_COMxxxx.

- Pour des raisons de confidentialité, nous avons pensé que mettre en place un système d'authentification par poste de travail serait une solution optimale. L'interface suivant est donc utile pour filtrer la connexion des utilisateurs au serveur étant donné que la base de données est distribuée, car sinon tous les utilisateurs auront accès à tout.

Fig.5.5 Filtrage de connexions des utilisateurs selon leur poste de travail

❖ SIMULATION

Nous avons fait recours à cette méthode parce que nous avons effectué tous les tests dans un laboratoire informatique. Dans notre laboratoire, nous sommes partis d'un énoncé pour parvenir à simuler un environnement favorable à notre étude :

➢ ENONCE

« Soit une Commne de sayada qui cherche comment satisfaire, en temps réel, les demandes administratives des citoyens pour les fidéliser. Elle ne souhaite également pas engager beaucoup de ressources humains et matérielles pour y arriver. Quelle solution serait envisageable dans ce cas ? »

➢ Données de Simulation

Soient A, B, N ; les communes de la wilaya de Mostaganem. Dans la commune A, on enregistre des demandes des corrections judiciaires qu'on souhaite partager en permanence avec tous les autres sites dans le territoire de la wilaya dans un environnement instable quant à la connexion réseau et aux perturbations du courant. Considérant les normes de confidentialité, Comment procéder pour que B et les N puissent avoir ces informations en permanence sur leurs sites ?

➢ Solution

Pour répondre à cette question, nous avons fait recours à notre affirmation hypothétique selon laquelle si on parvenait à Instaurer un système de Réplication dans le but d'avoir en permanence les informations des autres bases, alors la Conception d'un Algorithme de Réplication des données selon la fréquence de lancement des requêtes des utilisateurs serait une solution optimale pour minimiser les coûts d'achat de la bande passante.

Pour des raisons de sécurité, nous avons conçu une interface d'authentification (Figure 5.5) pour que les informations confidentielles des autres bases ne soient accessibles que par les ayant droits. A partir des algorithmes que nous avons proposés, nous avons conçu une interface qui permet de saisir les informations dans la base de données et de les répliquer avec les autres sites :

Fig.5.6 Interface de saisie des données de la demande de correction judicaire

Fig.5.7 Interface de Réplication des Données à propager

➢ **Conclusion**

La politique de haute disponibilité des données passant par la réplication des données dans une architecture distribuée semble être une solution moins coûteuse, mais efficace. Si cette institution financière souhaite pallier à ce problème de disponibilité d'informations, elle peut recourir à la mise en place d'un système de réplication qui prend en compte les difficultés liées à l'environnement dans lequel elle évolue, par exemple ce problème majeur de manque d'énergie électrique et des perturbations réseaux.

5.1.6. PRESENTATION ET DISCUSSION DES RESULTATS ➢

Résultats Obtenus

Tout au début de ce travail de recherche, nous nous sommes fixés l'objectif de concevoir et tester les algorithmes de réplication des données selon la fréquence de demandes des clients d'une institution financière. Voici les résultats que nous avons obtenus :

- Nous avons proposé un algorithme de réplication de données, que nous avons présenté sous forme de diagramme d'état, qui offre la possibilité de répliquer manuellement les données lorsqu'il y a coupure du courant ou problème de connexion entre les bases de données qui se partagent les informations ;
- Nous avons conçu avec UML des modèles de traitement des données par le Système de Réplication de bases de données, que nous avons décrit sous forme de diagramme d'activité et de séquence système à partir des quels nous sommes-nous dégagé les états par lesquels transite la base de données pendant la réplication ;
- Après exécution de ces instructions sur la Machine de Turing, nous avons constaté que ces premières ont produit des Automates Déterministes et des automates non déterministes ;
- Le fait qu'il y a eu des automates non déterministes, nous pouvons prouver les cas possibles d'interruption de la communication entre les sites distants étant donné qu'on ne sait pas avec exactitude si la réplication prendra fin sans qu'il y ait des perturbations.
- Quant aux méthodes de réplication auxquelles nous avons recourues pour les tests, nous avons constaté que la méthode transactionnelle est la meilleure parce qu'elle ne demande pas de faire une mise à jour sur la base cible pour que les modifications soient propagées, mais lorsqu'il s'agit de la méthode par fusion et de capture instantanée, on est obligé de redémarrer l'agent de réplication. Mais le

désavantage de la méthode transactionnelle, c'est qu'elle n'est pas programmable à partir de PHP raison pour laquelle nous avons programmé la méthode par Fusion (Merge Replication).

Conclusion générale

Nous avons vu que les systèmes de gestion de données distribués à grande échelle doivent offrir un stockage fiable, performant et cohérent. La réplication de données est au cœur de ces problématiques.

La réplication est une technique permettant de mettre en œuvre la tolérance aux fautes dans un système réparti. Répliquer une base de données BD consiste à faire en sorte qu'il existe plusieurs copies BDI de cette base dans le système, Chaque copie BDi est localisée sur un nœud distinct, Si une copie BD1 tombe en panne, il existe une probabilité non nulle qu'une autre copie BD2 soit toujours opérationnelle.

Lors de la conception d'un mécanisme de réplication, les aspects "performances" sont importants. En effet, la réplication permet d'améliorer la localité des accès en plaçant des copies des données proches des utilisateurs finaux, et la répartition de la charge en adaptant le nombre de copies d'une donnée à sa popularité.

La réplication de données est une technique clé pour permettre aux systèmes de gestion de données distribués à grande échelle d'offrir un stockage fiable et performant. Comme il gère un nombre suffisant de copies de chaque donnée, le système peut améliorer la pérennité.

De plus, la présence de copies bien placées réduit les temps d'accès. Cependant, cette même existence de plusieurs copies pose des problèmes de cohérence en cas de modification, notre projet est porté sur ces trois aspects liés à la réplication de données, la pérennité des données, la performance des accès et la gestion de la cohérence.

Le principal en jeu de cette recherche, formulée 'Réplication de bases de données d'une collectivité locale', était de trouver solution à la question de savoir quel modèle informatique permettrait de s'assurer de la disponibilité des informations découlant des opérations en cours avec les autres bases afin de répondre, en temps réel, aux demandes utilisateurs.

La solution que nous avons envisagée pour cette question, est la mise en place d'une architecture distribuée favorisant la propagation, en temps réel, des informations sur les opérations enregistrées au niveau de chaque agence. C'est ainsi que nous nous sommes fixé l'objectif de concevoir (et tester) un Algorithme de Réplication des données d'une collectivité locales afin de permettre cette dernière à avoir en permanence les données des opérations avec les autres bases dans l'objectif de répondre efficacement aux besoins du citoyen même lorsqu'il y a problème de connexion au réseau ou coupure du courant dans l'une des annexes.

La modélisation avec UML, la définition des algorithmes, l'expérimentation et la simulation nous ont conduits aux résultats suivants :

- Nous avons proposé un algorithme de réplication de données, que nous avons présenté sous forme de diagramme d'état dans le troisième chapitre, qui offre la possibilité de répliquer manuellement les données lorsqu'il y a coupure du courant ou problème de connexion entre les bases de données qui se partagent les informations ;
- Nous avons conçu avec UML des modèles de traitement des données par le Système de Réplication de bases de données, que nous avons décrit sous forme de diagramme d'activité et de séquence système à partir des quels nous avons dégagé les états par lesquels transite la base de données pendant la réplication ;
- Un prototype d'une application de gestion des collectivités locaux pour surveiller (Monitoring) les modifications des autres bases a été mis en place.

Enfin, signalons que nous ne prétendons en aucun cas avoir parfait le modèle informatique de la réplication des données, quand bien même c'était notre idéal. Ainsi donc, tout en endossant la responsabilité de tout genre de faille que peut renfermer ce dernier, nous encourageons toutes les autres tentatives de résolution des problèmes de haute disponibilité des données dans les entreprises à succursales multiples. Cela étant, nous restons ouverts à toutes les remarques et suggestions tant à son fond qu'à sa forme.

Bibliographie

• A.Schiper, P. a. (February 2000). *"Fault tolerance in Distributed System".*

• Bernholdt, D. (2005). *"The earth system grid : Supporting the next generation of climate modeling research".* Proceedings of the IEEE.

• Kesselman, I. F. (1998). *"The Grid : Blueprint for a New Computing Infrastructure".* San Francisco, USA: Morgan Kaufmann.

• M. Wiesmann, F. A. (2000). *Database replication technique: a three parametre classification .* Nuenberg, Germany: IEEE Computer Society.

• Nabrzyski, J. (2003). *"Grid Resource Management".* Kluwer Publishing. •

Solidor. (s.d.). *Tolerence aux fautes.* 1999.

• Wiesmann, M., F.Pedonne, & A.Schipper. (1999). *A Systematic Classification of Replited Database Protocole based on Atomic Broadcast.* Madeira Island Portugal.