Intergiciels et services pour la gestion de données distribuées

Claudia Lucia Roncancio

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Présentation des travaux de recherche pour obtenir le diplôme
d'HABILITATION À DIRIGER DES RECHERCHES
de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

par

Claudia Lucia Roncancio

(Arrêté ministériel du 23 Novembre 1988)

Discipline : **Informatique**

Intergiciels et services pour la gestion de données réparties

Date de soutenance : 23 juin 2004

Composition du Jury

Rapporteurs :	M Mokrane	Bouzeghoub
	M Yves	Chiaramella
	M Stefano	Spaccapietra
Examinateurs :	Mme Christine	Collet
	Mme Kathleen	Milsted
	M Jacques	Mossière

Diplôme préparé au Laboratoire Logiciels, Systèmes, Réseaux
(Institut d'Informatique et de Mathématiques Appliquées de Grenoble)

Intergiciels et services pour la gestion de données distribuées

Claudia Lucia Roncancio

.

Merci ...!

Me voila à écrire cette page si importante. Elle l'est d'autant plus que je crois que le travail d'équipe est essentiel dans la recherche. Les collègues et amis y sont pour beaucoup !

Tout d'abord je remercie l'INPG, l'ENSIMAG et le laboratoire Logiciels, Systèmes et Réseaux et ses membres. J'y travaille en tant que Maître de Conférences depuis 1995. Ils m'ont toujours offert un très bon cadre pour mes activités d'enseignant-chercheur.

Un mot à J. Mossière, Professeur à l'Institut National Polytechnique de Grenoble. Nous avons fait connaissance du temps où les bases de données de Bull étaient voisines des systèmes répartis de Bull-IMAG...Tout ce chemin de fait et comme le montre ce document un tel voisinage thématique me parait encore très bénéfique. Je suis sincèrement ravie qu'il préside ce jury d'habilitation. Merci !

Mes remerciements chaleureux à S. Spaccapietra, Professeur à l'EPFL, à M. Bouzeghoub, Professeur à l'Université de Versailles et à Y. Chiaramella, Professeur à l'Université Joseph Fourier et directeur de la fédération IMAG. Malgré leurs agendas très chargés, ils ont accepté de rapporter sur ce travail. Je leur en suis très reconnaissante.

K. Milsted, directrice du laboratoire DTL/ASR de France Télécom, me fait également l'honneur de participer au jury. Je la remercie pour cela ainsi que pour son soutien dans les coopérations entre son laboratoire et notre équipe.

Au tour de Ch. Collet, Professeur à l'Institut National Polytechnique de Grenoble et responsable de Storm, l'équipe bases de données où je mène mes recherches. Un très grand merci ! Pour son travail d'animation et d'encouragement scientifique et humain, pour moi et notre équipe ainsi que pour la confiance qu'elle m'a toujours témoignée. Tout cela a été très important pour moi.

Je remercie également B. Plateau, Professeur à l'INPG, pour son travail en tant que présidente du comité HDR INPG de notre discipline ainsi que J-L Soler (son prédécesseur) pour ses encouragements.

Ma reconnaissance à l'ensemble de l'équipe Storm. Parmi ses membres G. Vargas Solar, C. Bobineau, M. Adiba. Michel, qui m'a pris en DEA, il y a quelque temps, merci pour tous les enseignements le long de ces années. A toi, C. Labbé pour notre travail dans la joie et la bonne humeur, pour qu'il continue à être très productif.

Comment diriger des recherches sans étudiants? Ils sont nombreux à avoir travaillé, et travailler, avec moi : JL. Zechinelli, L. Rodriguez, ML. Schneider, N. Merle, N. Ibrahim, E. Benitez, L. Gurgen, L. D'Orazio, N. Henry, P. Cohen, J. Larramona, ... Je les en remercie. Mention spéciale au premier de tous S. Pafka alors que j'étais en thèse ; à mon premier docteur, S. Drapeau, pour notre voyage au fond de la duplication et à M-P. Villamil, ma première étudiante colombienne ... L'encadrer est un vrai plaisir !

A P. Serrano Alvarado pour nos longues sessions autour de la gestion de données en environnement mobile et les transactions. J'espère que toute cette mobilité nous permettra de continuer à coopérer. A T. Nedelec, L. Garcia, Q. Phoung pour nos discussions et réalisations liées à la gestion de données pour mondes virtuels répartis.

De nombreux résultats de mes recherches doivent beaucoup aux échanges avec des partenaires industriels et d'autres laboratoires. Impossible de les citer tous. Parmi eux des collègues de Xerox – L. Julliard, B. Chidlovskii –, de Bull et France Télécom – F. Dang Tran, P. Dechamboux, T. Coupaye, A. Lefebvre –, du projet IST PING, de l'Université de Zürich et Skövde, de Bell Labs, de l'AS CNRS Mobilité / Gestion de données et de l'action ACI Grid DataGraal – P. Sens et Y. Denneulin [1].

Je remercie également des collègues des équipes Sigma, Drakkar et Adele. A. Front, amie et colocatrice de bureau avec qui j'ai aussi eu le plaisir de travailler. J-P. Giraudin, M. Chabre-Peccoud, JM Favre et J. Chassin de Kergammeux pour nos divers échanges et leur soutien dans la préparation de cet HDR.

Certains sont près malgré la distance... Je pense particulièrement à R. Casallas, U. de los Andes (Bogotá). Merci pour tous nos "troques" GL/BD et pour sa superbe intervention en tant que *coach* d'habilitation qui proposait même un service de *regaños* à distance. Je n'aurais pas pu imaginer mieux ! E. Perez Cortes, U. Autónoma de Mexico, pour les discussions scientifiques et autres depuis tant d'années.

Une pensée spéciale à ma famille française autour de Titad et Timan et à ma famille colombienne... A mes parents qui m'ont inculqué très tôt le plaisir des études et du travail. Je me souviendrai toujours *"Lo único que nadie te puede quitar es lo que sabes"*.

Et pour finir, toi Yves, merci pour tout le bonheur... C'est la chose la plus précieuse et recherchée dans la vie ! A toi Katina pour être soleil et aux deux tout-petits soleils qui se préparent et qui m'accompagnent déjà aujourd'hui... Il fallait bien trois cerveaux pour une habilitation !

1. Mon collègue :)

.

Préambule

Ce manuscrit présente certains de mes travaux de recherche en matière de gestion de données, réalisés au sein de l'équipe bases de données STORM du laboratoire Logiciels, Systèmes, Réseaux de l'IMAG. Ils ont été effectués depuis 1995 (et jusqu'à 2003), année de ma prise de fonctions en tant que Maître de Conférences à l'Institut National Polytechnique (INP) de Grenoble, en poste à l'Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Gernoble(ENSIMAG).

Mes travaux concernent des aspects système — gestion de cache et duplication de données, support transactionnels, parmi d'autres. Ils contribuent d'une part à l'extension des fonctions des SGBD et d'autre part à la conception de services séparés utilisables indépendamment d'un SGBD. Mon point de vue porte sur la diversification du panorama des architectures des systèmes de gestion de données. J'ai d'abord travaillé dans l'introduction de fonctions déductives et actives dans les SGBD pour ensuite m'orienter vers des solutions plus ouvertes, des intergiciels de gestion de données distribuées sur des unités fixes et mobiles.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Évolution de la gestion de données

La gestion de données a été pendant longtemps de la responsabilité, quasi-exclusive, des Systèmes de Gestion de Bases de Données. Les SGBD se situaient au cœur des systèmes informatiques pour prendre en charge la plupart des tâches liées à la gestion de données – interrogation, persistance, cohérence, transactions, etc. Il existait ainsi un partage clair entre les aspects dits fonctionnels ou applicatifs et les aspects non fonctionnels pris en charge par le SGDB. La puissance des SGBD d'aujourd'hui, témoigne de la maturité atteinte dans le domaine de la gestion de données.

Il y a une dizaine d'années, certains domaines d'applications comme la gestion des entreprises, géraient leurs données avec des SGBD intégrant un riche ensemble de fonctions. En général il s'agissait de contextes dans lesquels, les besoins d'ouverture vers d'autres systèmes étaient faibles ou inexistants. Cependant les besoins des applications ont évolué, les domaines nécessitant une gestion de données se sont diversifiés et l'environnement matériel - ordinateurs, réseaux - s'est largement enrichi. La CAO, la géographie, le commerce électronique, la bio-informatique, le génie logiciel, les télécommunications, parmi d'autres, ont peu à peu introduit de nouveaux besoins mais aussi des éléments permettant d'aborder la gestion de données sous des angles nouveaux. Nous construisons aujourd'hui une société où les données tendent à être omniprésentes ainsi que les systèmes chargés de les gérer. Les sources de données sont très nombreuses, variées et fortement distribuées et leur gestion nécessite de systèmes aux tailles et caractéristiques diverses.

Pour répondre à ces exigences, les SGBD ont évolué selon plusieurs composantes – modèles de données, langages, fonctions, architecture. En terme de modèles de données nous notons des extensions du modèle relationnel – non normalisé – des modèles à objets complexes et des données semi-structurées. Les langages ont accompagné l'évolution des modèles de données avec des approches déclaratives et impératives et des langages à base de règles. Du point de vue de l'architecture des systèmes de gestion de données, différentes approches ont été et sont toujours proposées. Dans la suite nous donnons un panorama global de ces approches. Nous nous intéressons d'abord aux travaux centrés principalement sur les SGBD (section 1.1.1) puis aux approches introduisant la notion de service (section 1.1.2) et qui cherchent à construire des systèmes plus ouverts.

1.1.1 Systèmes de gestion de bases de données

Les travaux que nous évoquons ici, se sont déroulés dans un contexte qui considère le SGBD comme l'unité de déploiement pour les systèmes de gestion de données. Il s'agissait

ainsi de proposer des SGBD répondant au mieux aux besoins des utilisateurs, sachant que le SGBD centraliserait la gestion des données. Dans un premier temps la tendance dominante était l'intégration successive de nouvelles fonctions au sein des SGBD, puis sont apparues des approches de *génération de SGBD*, les *noyaux de SGBD extensibles* et des approches dites *boîte à outils*.

La première approche évoquée répond aux nouveaux besoins applicatifs en augmentant les fonctions des SGBD. Cet approche a été suivie, par exemple, pour créer les SGBD actifs [CHR96] et déductifs [Min88]. Nous avons ainsi conçu un SGBD de type DOOD[1], *Deductive and Object Oriented Database Systems*, intégrant des fonctions actives et déductives avec un modèle de données à objets [Ron94, PD94, RD96]. Les applications qui nécessitent à la fois la gestion de données complexes et une analyse des connaissances [FLV96] tirent pleinement profit de tels SGBD. La bio-informatique et la télévision à la demande en sont des exemples. Par ailleurs, nous constatons que les SGBD relationnel les plus largement utilisés intègrent un grand nombre de fonctions de gestion de données. Ces SGBD sont appropriés pour diverses applications, cependant pour d'autres le rapport entre le coût d'utilisation (administration, taille, coût, ressources) et le service requis n'est pas toujours avantageux.

Certains travaux sur les architectures des SGBD ont exploré des approches donnant lieu à des *générateurs de SGBD* ou de parties de SGBD. EXODUS [CDRS86] et son successeur Volcano [GM93] en sont des exemples. Ils génèrent des optimiseurs de requêtes à partir d'informations telles que les opérateurs à inclure, les règles de transformation et des règles de mise en œuvre pour les opérateurs (par exemple, pour la jointure utiliser *hash join*). Nous notons l'utilisation avec succès de Volcano par l'optimiseur de requêtes d'Open OODB [BMG93]. L'approche est cependant limitée par la complexité des interactions des différents éléments d'un SGBD. Par exemple, pour intégrer de manière satisfaisante un gestionnaire d'index généré, il faut considérer les interactions avec les modules d'optimisation de requêtes, de gestion de cache, de contrôle de la concurrence, ou encore de reprise après pannes.

Pour pallier certains problèmes liés à la génération, une autre approche se base sur un *noyau de SGBD* qui peut être étendu grâce à de nouvelles fonctions ou des mécanismes internes.[2] Cette approche a été largement utilisée, par exemple dans Starbust[HCL+90], pour permettre aux utilisateurs d'étendre l'ensemble des types gérés. Les extensions objet de systèmes relationnels peuvent être considérées comme des succès de cette approche (par exemple, IBM DB2 [IBM95], Oracle [Ora99]). L'extension des types fonctionne assez bien mais introduire de manière performante des mécanismes internes (tels qu'un type d'index spécifique) reste un problème d'ingénierie difficile.

En allant plus loin, l'approche *boîte à outils* propose des SGBD minimaux, avec un noyau de fonctions et des outils permettant de construire par-dessus des SGBD ad-hoc. Shore [CDF+94], DASDBS [PSS+87] et Open OODB [BMG93] en sont des exemples. Open OODB a proposé des modules système de haut niveau tel que le traitement de requêtes, la duplication et la distribution.

Ces travaux ont conduit à une meilleure connaissance des aspects participant à la gestion des données et constituent des racines pour une gestion de données plus adaptée et des

1. Travail réalisé dans le cadre du projet IDEA [IDE92]
2. Appelés parfois "plug-ins"

solutions plus ouvertes. Dans les approches mentionnées, l'unité minimale de déploiement reste le SGBD dans son ensemble. Elles ne permettent pas de restreindre les fonctions offertes. Néanmoins, offrir un grain assez fin pour la gestion de données est devenu essentiel pour faciliter la construction de systèmes d'information à différentes échelles. Les approches de ce type sont discutées dans ce qui suit.

1.1.2 Composants et services pour gérer les données

Les recherches les plus récentes bénéficient d'un dé-cloisonnement entre les travaux menés dans des domaines divers de l'informatique. Nous remarquons une influence mutuelle forte entre les travaux autour des bases de données, les systèmes distribués et le génie logiciel. Ce rapprochement est probablement la conséquence du besoin de développer des logiciels et des systèmes complexes où hétérogénéité, distribution et gestion de données sont la règle. Les propositions en génie logiciel telles que le principe de séparation des préoccupations (par exemple, la programmation par aspects [CE99]) et les développements à base de composants jouent un rôle essentiel dans le développement de tels systèmes. Sans vouloir discuter des modèles à composants, dans la suite, nous considérons informellement un composant comme une unité logicielle offrant un ensemble bien défini de fonctions, la construction de logiciels plus complexes se fessant par intégration de composants.

Des systèmes de gestion de données construits selon différentes architectures utilisent des composants [DG01]. C'est déjà le cas pour des extensions proposées dans certaines des approches citées dans la section précédente (par exemple, les noyaux de SGBD extensibles). Une des leçons tirée de ces expériences est qu'il est important d'identifier les composants du système et leurs interactions, à un niveau relativement fin, ceci afin de faciliter leur re-utilisation et la gestion de l'impact que peut avoir un changement (gérer l'effet *domino*).

Le besoin de développer des intergiciels [3] mettant en œuvre des systèmes répartis aux caractéristiques très variées a accru l'utilisation des approches par composants. L'importance de cette approche s'est fait également sentir en matière de gestion de données, du fait que la gestion de données doit être adaptée aux particularités propres au contexte d'utilisation et souvent à divers points des systèmes construits. De nombreux efforts sont consacrés à proposer des composants de gestion de données intégrables et réutilisables dans des systèmes ouverts. Nous remarquons ceci dans :

– les *intergiciels pour la gestion de données distribuées*

– les *services de gestion de données.*

L'objectif de ce que nous appelons *intergiciels pour la gestion de données* est de fournir une sorte d'intégration de diverses sources de données existantes. Les sources peuvent être des SGBD, mais aussi des moteurs de recherche, des sites Web ou autres entrepôts de données. Les données restent, en général, dans leur système d'origine. L'utilisation d'un intergiciel permet d'offrir certaines fonctions intégrées sur ces sources — interrogation, transactions. Il est très courant que les sources de données soient hétérogènes : modèle de données différents, langages de requêtes ayant différentes puissance d'expression, etc. Les systèmes de médiation et les méta-moteurs de recherche [Wie92, DD99, BLZS02] peuvent être considérés comme des intergiciels pour la gestion de données. Les systèmes multi-bases poursuivent des objectifs similaires mais sont destinés à l'intégration de bases gérées par des SGBD sans d'autres types de sources de données. Dans les intergiciels de gestion de

3. Traduction proposée pour *middleware*[Ber96]. Classe de logiciels qui assurent l'intermédiaire entre les applications et le transport des données par les réseaux.

données, les composants sont utilisés à différents niveaux, par exemple pour la mise en œuvre de gestionnaires de requêtes au niveau global, dans un gestionnaire de cache ou pour mettre en œuvre des adaptateurs.[4] Ces derniers prennent en charge certains aspects de l'hétérogénéité des sources.

Une autre approche architecturale est celle où les fonctions pour la gestion de données sont vues sous forme de *services*. Les applications n'interagissent pas avec un SGBD monolithique mais avec le(s) service(s) requis — persistance, duplication, transactions, etc. Chaque service propose une ou plusieurs interfaces et peut avoir diverses mises en œuvre. Un service peut être vu comme un composant complexe. L'approche par services a été préconisée par l'OMG (Object Management Group) [OMG95] autour de CORBA. L'OMG a spécifié, entre autres, les interfaces pour des services de persistance, de contrôle de la concurrence et de requêtes. Ces travaux étaient motivés principalement pour assurer l'interopérabilité de systèmes répartis.

La notion de service a également été adoptée avec une approche bases de données. Les divers aspects de la gestion de données sont séparés dans des services (persistance, duplication, requêtes), chacun d'eux étant adaptable. Une décomposition interne d'un service en composants facilite l'adaptation de la fonction rendue à un grain fin. Les services peuvent être intégrés pour construire des systèmes — *SGBD configurables* ou intergiciels — offrant exactement les fonctions requises. Les systèmes ainsi obtenus auront différentes tailles et caractéristiques. A une extrémité du spectre on peut imaginer un système très réduit assurant seulement la persistance sans autre garantie et, à l'autre extrémité, un SGBD pratiquement complet configuré de manière ad-hoc. Les projets KIDS [GSD97] et notre projet NODS [Col00] travaillent dans ce sens. La grande difficulté de cette approche est de dégager et de concevoir les composants de manière à ce qu'ils soient effectivement réutilisables et offrent un niveau adéquat d'adaptabilité.

Nous remarquons l'intérêt d'offrir hors SGBD des fonctions qui traditionnellement étaient proposées exclusivement intégrées dans ceux-ci. On tente de rendre ces fonctions facilement utilisables de manière sélective et adaptable pour mieux les intégrer dans contextes plus divers. En effet les fonctions de gestion de données présentent un intérêt pour un vaste éventail d'applications/systèmes qui ne sont pas naturellement étiquetés « bases de données » : gestionnaires de courrier électronique, jeux, plates-formes de télécommunications, etc.

Le support d'applications réparties nécessite l'utilisation de gestionnaires de données variés à différents points des systèmes d'information. Par exemple, dans les systèmes à architecture trois-tiers peuvent cohabiter un SGBD puissant dans le tiers « serveur » et certains services (par exemple, cache et évaluateur de requêtes restreintes) dans le tiers du milieu, qui capture en général une partie importante de la logique applicative.

Globalement, l'intérêt des approches évoquées précédemment est également clair pour les applications dans des environnements à ressources variables ou restreintes. C'est le cas des environnements mobiles où les systèmes répartis font intervenir des unités fixes (sur réseau câblé) et des unités mobiles telles que les portables ou les PDA (sur réseau sans fil). Dans ce contexte, les SGBD à petite empreinte (taille du code et de la mémoire requise pour fonctionner) disposant de fonctions ciblées sont indispensables. Les SGBD avec une architecture fortement modulaire (comme Poet Fast Objects j2 [bP]), ceux construits à partir de composants ou les SGBD configurables semblent particulièrement adaptés. La construction d'intergiciels pour les environnements mobiles bénéficie également des services adaptables pour la gestion de données.

4. Dits *wrappers*.

1.2 Contributions

Nos contributions reflètent d'une certaine manière l'évolution de la gestion de données que nous venons d'introduire. Nous avons tout d'abord contribué à l'extension de SGBD puis porté nos efforts sur la définition de composants et de services adaptables de gestion de données, utilisables dans des intergiciels ou des SGBD configurables. Nos recherches actuelles se poursuivent dans ce sens.

Extensions déductives et actives des SGBD Nos travaux sur des extensions de SGBD concernent d'une part des systèmes de type DOOD et d'autre part des SGBD à objets actifs.

Nous avons participé à la conception d'un SGBD offrant un modèle de données à objets complexes et intégrant dans son noyau des capacités de traitement de données importantes : un langage de programmation pour bases de données (LPBD) impératif et des langages de règles déductives et actives. Notre contribution a particulièrement porté sur l'extension du LPBD par des règles actives et des règles déductives. Ce travail a été réalisé principalement durant ma thèse [Ron94] (effectuée sous contrat avec la société Bull) et dans le cadre du projet Esprit IDEA [IDE92]. Ces travaux ne sont pas repris dans ce manuscrit.

Nous nous sommes ensuite focalisés sur l'extension des SGBD à objets avec des fonctions actives. La motivation principale était de faciliter des traitements automatiques et performants des données de manière événementielle. Nous avons proposé des modèles d'événements de haut niveau (composites, avec une durée et dynamiques) permettant de faciliter la représentation de la sémantique applicative. Nous avons également étudié en profondeur l'impact des modules de support des fonctions actives sur les performances du SGBD selon les approches utilisées pour l'intégration de telles fonctions. Nous avons par ailleurs conçu un service de visualisation d'exécutions de systèmes actifs. Ce service cherche à aider les utilisateurs dans la compréhension et la mise au point (parfois difficile) d'applications s'exécutant sur des tels systèmes. Ces travaux constituent des résultats préliminaires à des recherches plus récentes, menées par ailleurs, utilisant l'approche événementielle dans des systèmes d'information distribués.

Nos travaux sur les SGDB à objets actifs se sont inscrits dans le cadre du projet IMAG Storm dont l'objectif était le développement des serveurs d'objets actifs, temporels et multimédia. Nos contributions sur l'aspect actif font l'objet du chapitre 2.

La suite de nos recherches a été fortement motivée par le besoin fort d'une gestion distribuée des données et d'une diversification des systèmes dédiés à une telle gestion. Nous avons travaillé à la définition de services « base de données » utilisables hors SGBD en suivant une approche par composants. Nos propositions sont destinées principalement aux intergiciels de gestion de données distribuées.

Cache et intergiciels de gestion de données Nous nous intéressons ici à des intergiciels permettant l'interrogation de sources de données hétérogènes et distribuées pour le contexte WWW. Nous contribuons à l'optimisation de l'évaluation de requêtes multi-sources par la proposition d'un gestionnaire de cache sémantique. Les sources de données accédées via de tels intergiciels sont très autonomes et rendent l'utilisation d'autres types de caches peu ou pas adaptée. Notre proposition permet la réutilisation totale ou partielle de résultats en accord avec les caractéristiques des sources de données et des requêtes. Le cache est proposé comme un composant qui peut être activé ou non selon le contexte de déploiement de l'intergiciel. Nous avons également proposé une technique ad-hoc pour

pré-charger des données dans ce cadre. La proposition de cache a été expérimentée dans le prototype de méta-moteur de recherche *Knowledge Broker* et est aujourd'hui incluse dans son successeur *AskOnce* [Ask] commercialisé par Xerox. Ces travaux sont présentés dans le chapitre 3.

Le maintien de la cohérence des données cachées par rapport aux données présentes sur les sites d'origine n'a pas été approfondi dans le travail cité. Des aspects liés à la cohérence de données dupliquées et à la cohérence globale ont été abordés par la suite, dans nos travaux sur les services de duplication et de transactions. La définition de ces services suit une philosophie de séparation des préoccupations en cherchant à bien délimiter les frontières fonctionnelles pour faciliter ainsi leur réutilisation. Ces travaux s'inscrivent dans le projet NODS, *Networked Open Database Services*[5], mené par notre équipe sous la direction de Ch. Collet (laboratoire LSR-IMAG). Dans ce projet nous explorons la décomposition des SGBD en services adaptables dédiés chacun à un aspect de la gestion de données [Col02]. En plus des services de duplication et de transactions présentés dans ce manuscrit, des services de persistance [GB03], de tolérance aux fautes [Duo03], de requêtes [VC04], d'événements [VS00], de réaction [GR00] et de workflow [BVSC03a] sont issus de NODS. Ces services peuvent ainsi être utilisés dans des *intergiciels pour la gestion de données* ou pour créer des *SGBD configurables* de manière à obtenir une gestion de données aussi adaptée que possible aux besoins.

Services de duplication Nous avons mené des travaux approfondis sur la duplication de données dans le but de proposer une solution permettant de faciliter le développement de systèmes nécessitant de cet aspect. Nos travaux ont donné lieu à un canevas de services de duplication. Un service de duplication créé met en œuvre un protocole particulier pour implanter le modèle de cohérence nécessaire. Nos contributions portent sur trois axes : (1) la modélisation des services de duplication pouvant être obtenus du canevas (2) l'adaptabilité du canevas relativement au contexte non-fonctionnel et (3) l'adaptabilité des protocoles de duplication ou de parties de ceux-ci. Les services de duplication créés peuvent être utilisés dans le cadre d'intergiciels bases de données ou de SGBD configurables.

Ces travaux ont été principalement menés dans le cadre de la thèse de S. Drapeau [Dra03], réalisée sous ma direction en coopération avec P. Dechamboux (France Télécom). Nos propositions ont été mises en pratique dans le cadre du développement de plates-formes pour mondes virtuels répartis. Ceci d'une part dans le cadre du projet IST PING [PIN00], et d'autre part dans le Contrat de Recherche Externe « Services bases de données ouverts pour applications coopératives » avec France Télécom [CRGBN01, CRD+01]. Nos travaux sur les services de duplication sont présentés dans le chapitre 4.

La dernière contribution traitée dans ce manuscrit porte sur l'aspect transactionnel.

Service de transactions Nos travaux sur l'aspect transactionnel se situent clairement au niveau intergiciel de gestion de données. Ils ont été motivés par le besoin de supports transactionnels pour des exécutions sur des sources de données faiblement couplées dans des environnements variables. Nous nous intéressons à des environnements intégrant des unités fixes et des unités mobiles. La dimension « mobile » introduit des particularités importantes qui affectent la gestion des données en général et l'aspect transactionnel en particulier. Nous avons réalisé une étude détaillée sur la gestion transactionnelle de

5. http://www-lsr.imag.fr/Les.Groupes/STORM/Storm2002

données dans les environnements mobiles [SARA04]. Nous proposons un support transactionnel général souple et adaptable aux variations du contexte. Notre contribution inclut un modèle de transactions flexible (formalisé en ACTA) et un support d'exécution incluant des protocoles adaptés.

Ces travaux, présentés dans le chapitre 5, font l'objet de la thèse de P. Serrano Alvarado [SARA04] réalisée sous ma direction et celle de M. Adiba (laboratoire LSR-IMAG).

Les travaux présentés dans ce manuscrit participent à l'évolution vers une gestion de données plus personnalisée où les utilisateurs pourraient effectivement disposer des systèmes de gestion de données à la taille requise au moment adéquat. Ceci n'est pas encore une réalité mais la diversité dans le panorama des systèmes d'information ne fait que se confirmer. Le chapitre 6 présente nos conclusions et notre point de vue sur la suite de ces recherches.

Chapitre 2

Gestion événementielle des données

Ce chapitre porte sur les fonctions actives pour la gestion de données. L'approche suivie dans nos travaux s'est basée sur l'extension des SGBD par des gestionnaires d'événements et de réactions.

2.1 Introduction

La gestion d'événements a été largement étudiée dans divers domaines tels que la programmation, les systèmes et la gestion de données. Autour des années 90-95 la communauté bases de données a consacré de nombreux efforts dans l'intégration des notions d'événement et de réaction dans les SGBD. L'objectif recherché était l'augmentation des capacités de traitement des données afin de mieux répondre aux besoins des applications utilisant le SGBD.

Un SGBD actif [CHR96] est capable de détecter des événements et de réagir, sans intervention des utilisateurs, en exécutant certaines actions. Les « réactions » d'un tel système sont habituellement exprimées sous forme de règles, dites actives[1], de la forme Événement-Condition-Action. Il s'agit d'un mécanisme général utilisable pour différents buts tels que le maintient de l'intégrité, la gestion de versions, de droits et de notifications. L'approche événementielle a été largement soulignée par les applications actives initiales et est d'un grand intérêt aujourd'hui pour des applications telles que la publication et la gestion de pages XML [FJL+01], les systèmes proposant une adaptabilité au contexte (par exemple dans des environnements mobiles [SARAL03]) ou ceux pour la capture continue de données issues de capteurs. Aussi d'une manière générale, dans des systèmes distribués pour l'intéropérabilité et la coopération de composants.

Les travaux de recherche du domaine des SGBD actifs (réalisés dans la première partie des années 90) ont porté sur les langages de définition des règles, le support d'événements, les modèles d'exécution et les architectures des systèmes actifs à proprement parler.

Nos contributions concernent différents points de ce domaine et portent plus particulièrement sur les SGBD à objets. Nous avons travaillé sur deux approches pour l'intégration des fonctions actives. La première est une approche langage où j'ai proposé le Langage de Programmation pour Bases de Données Peplom^{ad} qui intègre fortement des règles actives dans le contexte d'une BD orientée objets et déductive (DOOD) [RD96]. La deuxième approche est basée sur l'extension du noyau d'un SGBD à objets de manière à introduire un

1. Déclencheurs dans leur forme la plus simple

détecteur d'événements et à associer un module extérieur gérant les réactions [CCR96b]. Le prototype NAOS (extension active du SGBD à objets O_2) développé dans notre équipe suit cette approche [CCFR97, CCS94].

Une de nos contribution concerne la notion d'événements qui est probablement la partie des règles actives qui a suscité le plus de travaux. Les premiers SGBD actifs proposaient un support pour des événements primitifs (atomiques) issus d'opérations simples sur les données (par exemple l'insertion). L'étude de différentes applications a mis en évidence les limitations de ce modèle et nous a amené à proposer des types d'événements permettant de capturer plus de sémantique afin de refléter des situations plus riches. Nous avons proposé un ensemble d'opérateurs pour la définition d'événements complexes [CCR96a], gérant la durée et dynamiques [Ron98] ainsi que temporels [CCFR97]. Les autres contributeurs à ces travaux sont Th. Coupaye, Ch. Collet, M-C Fauvet et A. Poitou (respectivement doctorant, enseignants-chercheurs et étudiant en DEA au Laboratoire LSR-IMAG [2]).

Le vaste choix dans les modèles d'événements, leur mise en œuvre et celle des SGBD actifs en général, nous a conduit à réaliser une étude des performances des SGBD actifs. Cette étude nous a permis de confirmer/infirmer certaines hypothèses sur les performances des SGBD actifs et de dégager des conclusions servant à leurs concepteurs et utilisateurs. Ce travail a été réalisé conjointement avec A. Geppert (Université de Zürich), M. Berndsson (Université de Skövde) et D. Lieuwen (Lucent Technologies/Bell Innovations) [GBLR98],[GBLR96].

Une troisième contribution présentée ici porte sur l'aide aux utilisateurs des systèmes actifs. Les modèles d'exécution des systèmes actifs étant intrinsèquement difficiles à comprendre, le développement d'une application devient ardue lorsqu'elle concerne un nombre important d'événements et de règles actives. Des aides en phase de conception [FRG96] et de développement semblent indispensables. Pour cette dernière, nous avons conçu et implanté un service de visualisation d'exécution de règles actives. Notre outil, a été initialement pensé pour NAOS [CRBL97] puis adapté pour les systèmes actifs en général [CRB99a, CRB99b] en se basant sur une taxonomie des différents modèles d'exécution existants [Cou96].Ce travail a été réalisé avec Th. Coupaye (post-doctorant au laboratoire LSR-IMAG), C. Bruley (Doctorant du laboratoire GRAVIR-IMAG) et J. Larramona (élève ingénieur ENSIMAG).

La suite du chapitre aborde brièvement chacun des trois aspects mentionnés. La section 2.2 présente nos travaux sur les types d'événements, la section 2.3 ceux concernant l'évaluation de performances des SGBD actifs et la section 2.4 notre service de visualisation d'exécution. Nos conclusions et l'évolution des recherches dans ce domaine font l'objet de la section 2.5.

2.2 Types d'événements

Un type d'événement est une expression qui caractérise une classe de faits significatifs pour le déclenchement d'une règle. Dans un SGBD, la plupart des faits auxquels on s'intéresse correspondent à des opérations (non instantanées) sur les données, par exemple, l'exécution d'une méthode ou la mise à jour de données. Leur exécution permet la production d'un ou de plusieurs événements. Considérons la mise à jour de l'attri-

2. Affiliation au moment des travaux

but `salaire` d'entités du type `Employé` : `update(Employé.salaire, integer)`. Pour ce type d'opération nous pouvons spécifier trois types d'événements selon le grain auquel on s'intéresse :

- le type d'événement `update(Employe.salaire, integer)` permet de s'intéresser à des événements qui durent. Il caractérise les événements notés (`update(e.salaire,v), [t0, t1]`) où `e` est une instance d'Employé, `v` est la nouvelle valeur du salaire de `e` et `[t0, t1]` permet de connaître la durée de l'opération. Avec ce type d'événement, il est possible de s'intéresser aux processus (opérations atomiques, méthodes, programmes, etc) eux-mêmes et pas seulement à certains des changements d'états qu'ils réalisent (en début et fin).

- le second type d'événement `before update(Employé.salaire, integer)` caractérise les événements qui ont lieu immédiatement avant le début de la mise à jour. L'événement intéressant est « une mise à jour de salaire va commencer ».

- le troisième type d'événement, `after update(Employe.salaire,integer)`, caractérise les événements qui ont lieu immédiatement après une mise à jour. L'événement intéressant est « une mise à jour de salaire est terminée ».

Ces exemples de types d'événements montrent l'utilisation des « modificateurs » `before` et `after` qui permettent de mieux préciser l'instant d'occurrence d'un événement. La plupart des modèles d'événements proposés se basent sur des événements « instantanés » comme les deux derniers cités. Nous avons contribué à ces travaux mais aussi proposé un modèle prenant en compte des types événements qui ont une durée (voir section 2.2.1) et dynamiques (voir section 2.2.2).

En faisant abstraction de l'aspect instantané ou non, les types d'événements peuvent être divisés en deux grandes classes : ceux décrivant des faits élémentaires — événements *primitifs* — et ceux décrivant des faits composés ou issus d'autres faits élémentaires — événements *composites* [GD93, GJ91, GJS92]. Les événements primitifs sont généralement regroupés en trois catégories : les événements internes provenant du SGBD, les externes et les temporels. Un type d'événement composite est défini par une expression qui inclue des types d'événements primitifs ou composites et des opérateurs de composition dont les plus répandus sont la disjonction, la conjonction et la séquence. La section 2.2.1 introduit nos travaux sur les événements composites et la section 2.2.2 notre proposition d'événements dynamiques

2.2.1 Événements composites

Un événement composite se produit lorsque ses événements composants se sont produits dans un certain ordre ou en respectant certaines conditions décrites par la sémantique de l'opérateur de composition utilisé. Soit un type d'événement `E = op(E1, E2)` où `op` est un opérateur de composition et `E1`, `E2` deux types d'événements. La production d'un événement `e` du type `E` est fonction de la production d'événements `e1` de type `E1` et `e2` de type `E2` selon les règles sémantiques générales suivantes :

- *Disjonction :* `e` a lieu quand soit `e1`, soit `e2` a lieu.

- *Conjonction :* `e` a lieu quand les deux événements `e1` et `e2` ont eu lieu, sans considération de l'instant de production.

– *Séquence :* `e` a lieu quand `e2` survient après que `e1` se soit produit.

Si `e1` et `e2` sont eux-mêmes des événements composites, la composition de chacun de ces événements e_i n'est pas instantanée. Plusieurs cas de figures se présentent selon que les intervalles de reconnaissance des e_i se chevauchent ou non. Nous distinguons les cas suivants :

– Le dernier événement participant à la composition de `e1` survient avant le dernier événement participant à la composition de `e2`, mais aucune contrainte n'est imposée sur l'ordre d'occurrence des autres événements participant à leur composition.

– Tous les événements primitifs participant à la composition de `e1` ont lieu avant tous ceux participant à la composition de `e2`. Dans ce cas on parle de *séquence stricte*.

Ceci nous amène à la notion de **mode de production** des événements composites qui permet de lever l'ambiguïté sur les événements qui participent à un événement composite. Pour illustrer cet aspect, considérons le type d'événement

```
E = séquence(E1,E2)
```

utilisant les types d'événements primitifs `E1` et `E2`. Considérons l'historique d'événements suivant :

```
e21 e12 e13 e24 e15 e26 e27
```

où `eij` est un événement du type `Ei` produit à l'instant `tj` dans l'unité de production considérée (par exemple la transaction ou le programme). Il est alors possible de produire différents événements composites selon que l'on considère les modes de production :

– *Continu* : dans ce mode toutes les occurrences de `Ei` qui marquent le début d'un intervalle d'un type d'événement composite sont potentiellement considérées comme des événements initiateurs d'événements composites. Dans l'exemple, tous les événements `e1j` « démarrent » la détection d'un événement du type `E`. Nous avons alors la production des événements : `(e12, e24)`, `(e13, e24)`, `(e12, e26)`, `(e13, e26)`, `(e15, e26)`, `(e12, e27)`, `(e13, e27)` et `(e15, e27)`.

– *Récent* : dans ce mode seules les occurrences du type `Ei` les plus récentes sont utilisées pour produire un événement composite. Pour l'exemple d'historique cité, nous avons les événements du type `E` : `(e13, e24)`, `(e15, e26)` et `(e15, e27)`.

– *Chronologique* : dans ce mode les occurrences de `Ei` sont considérées dans leur ordre d'apparition chronologique et ne sont pas réutilisés. Pour l'exemple d'historique, nous avons les événements du type `E` : `(e12, e24)`, `(e13, e26)` et `(e15, e27)`.

– *Cumulatif* : dans ce mode lorsqu'une occurrence de `E` est reconnue, le contexte qui lui est associé inclut (*cumule*) les paramètres de toutes les occurrences des `Ei` intervenant dans la définition du type `E`. Une occurrence de `Ei` participe au contexte d'une seule occurrence de `E`. A partir de l'exemple, nous avons une occurrence de type E avec `e24` qui inclut `(e12, e13)` dans son contexte et une autre avec `e26` et qui inclut `(e15)`. Lorsque `e26` se produit, il n'y a pas d'événement instance de `E1` non consommé. `e26` ne participe donc pas à une instance de `E`.

```
Class Projet inherit Object
tuple ( nom: string,
        participants: set( Personne),
        date_debut: Instant,
        date_fin: Instant,
        durée_spec: Duration,
        periode_prog: Interval );
```

FIG. 2.1 – *Classe Projet, exemple.*

Les modes de production offrent aux programmeurs la possibilité de préciser la sémantique des types d'événements composites en fonction de l'application.

Globalement, un nombre important de travaux cherchaient à proposer des moyens d'expression d'événements permettant de faciliter l'expression de situations courantes dans les applications [CM93, GD93, GJ91, GJS92]. Les travaux se différencient par les opérateurs proposés et leur sémantique. Les travaux réalisés dans notre équipe sur NAOS ont, entre autres, inclut la définition d'événements composites [CC96]. NAOS [CCR96a, CCFR97] a été parmi les premiers à mettre en œuvre un détecteur, pour des événements utilisant la conjonction, la disjonction (exclusive ou non), la séquence (stricte ou non) et la négation. Cette dernière se réfère à la non-apparition d'un événement pendant un intervalle délimité par d'autres événements. Dans cette proposition, on raisonne sur l'instant de reconnaissance d'une instance d'événement mais on n'intègre pratiquement pas la durée d'un événement.

Par ailleurs une intégration forte d'aspects temporels associés aux données nous a permis d'introduire la notion d'événements avec durée et contraintes. Le modèle que nous proposons dans [Ron98] intègre les intervalles de manière uniforme. Le chronon, plus courte durée supportée par un système, est la durée des événements instantanés. Les événements primitifs et composites peuvent néanmoins avoir des durées plus longues. Les événements à durée peuvent être vus comme un niveau d'abstraction au-dessus des instantanés. Les opérateurs reprennent des propositions d'événement composites cités précédemment et les travaux autour des bases de données temporelles [DFS02]. Les opérateurs proposés sont *precedes, during, overlaps, starts, equal, ends, meets*. Pour les événements on distingue la période d'occurrence, la durée et l'instant de notification. Un tel modèle permet de spécifier de manière aisée des situations telles que l'occurrence d'un événement pendant un autre. Cette proposition offre aussi la possibilité d'inclure dans la définition d'un type d'événement un prédicat simple permettant de mieux filtrer les événements intéressants.

2.2.2 Types d'événements dynamiques

Les types d'événements provenant des SGBD sont issus en général des opérations effectuées sur les données (par exemple, un appel à une méthode ou une opération sur un attribut) alors que les événements temporels sont en général explicites (par exemple à une date donnée ou tous les jours à une heure précise). Dans le contexte du modèle d'événements avec durée que nous avons proposé pour une base de données à objets et temporelle nous avons introduit des types d'événements dynamiques [Ron98]. Il s'agit de types d'événements génériques qui sont instantiés dynamiquement avec les données présentes dans la base. Afin d'illustrer, considérons la classe Projet introduite par la figure 2.1. Dans cette définition `Instant`, `Duration` et `Interval` sont des classes mettant en œuvre des concepts temporels comme dans [FCS97].

Nous proposons la possibilité de définir des types d'événements comme dans la règle active suivante :

```
on Projet.datefin do { action 1 }
```

Cette règle sera déclenchée à la date limite de chaque projet. `Projet.datefin` est un type d'événement dynamique qui spécifie un ensemble d'événements temporels statiques. Cet ensemble est créé dynamiquement à partir des objets de la classe projet de la base. Si par exemple, la base a deux projets α et β avec date limite D1 et D2 respectivement, le système crée deux événements temporels correspondant à ces dates. La règle active introduite sera déclenchée à la date D1 pour le projet α et à la date D2 pour le projet β. Contrairement aux événements temporels classiques, l'objet projet fait ici partie de l'environnement véhiculé par l'événement temporel.

Notons l'aspect dynamique de cette sorte d'événement : il dépend de l'état de la base. Une mise à jour des données (par exemple l'ajout d'un projet ou la modification d'une date) peut entraîner une modification de l'ensemble des événements statiques générés. Dans ce cas la mise à jour est propagée par le système sur les types d'événements générés. L'utilisateur ne doit pas modifier la règle ni se soucier des questions de cohérence.

La proposition d'événements dynamiques (ou génériques) porte sur l'expression d'événements temporels via l'utilisation des données de la base. L'exemple ci-après utilise la date de début de projet et la durée de la phase de spécification. L'événement est le jour qui suit la fin de cette phase.

```
on Project.starting_date + Project.spec_time {same object} do ...
```

Les types d'événements dynamiques facilitent l'expression et la cohérence d'événements temporels dans de nombreuses applications. Leur mise en œuvre repose sur des règles actives tel que nous les présentons dans [Ron98].

2.3 Évaluation de performances

Les travaux sur les SGBD actifs ont donné lieu à de nombreux choix aussi bien au niveau des modèles qu'au niveau des mises en œuvre. Ces divers choix ont également amené un besoin d'évaluation de performances afin d'aider aussi bien les développeurs de SGBD actifs que les utilisateurs sur des aspects délicats tel que la détection d'événements composites.

Nous avons réalisé[3] un travail d'évaluation de performances sur des prototypes de SGBD à objets actif. Les prototypes étaient disponibles sur des plates-formes hétérogènes et se différenciaient par les fonctions offertes. Il ne s'agissait donc pas d'effectuer une comparaison des performances dans l'absolu mais de comparer et d'analyser les propositions afin de dégager les compromis entre les fonctions offertes, les performances et les choix de mise en œuvre.

Les SGBD actifs évalués sont ACOOD [Ber94], NAOS [CCS94], Ode [LGA96] et SAMOS [GGD91]. Aucun système relationnel n'a été évalué car le modèle de données joue sur les performances et une forte hétérogénéité des modèles de données aurait été difficile à intégrer. Nous avons contribué à la consolidation du banc d'essai BEAST [GGD95], *BEnchmark for Active database SysTems*, qui utilise le schéma de base de données et les bases spécifiées par le banc d'essai OO7 [CDN93]. 007 et BEAST mesurent respectivement

3. En coopération avec des chercheurs d'autres universités

les performances des aspects passifs et actifs des SGBD à objets. Nous nous sommes focalisés sur les aspects actifs sans cibler un domaine d'application particulier afin de ne pas favoriser ou défavoriser un des systèmes qui aurait été pensé en privilégiant un domaine spécifique.

D'une manière générale, un système actif va disposer d'un détecteur d'événements, d'un gestionnaire de règles et d'un module d'exécution des règles déclenchées. Dans le support d'un comportement actif on distingue une première phase pour la détection et la notification des événements ; une deuxième phase correspondant au choix des règles à exécuter tenant compte des modèles d'exécution et une troisième phase effectue le déclenchement à proprement parler. L'évaluation de performances a porté sur la détection d'événements (primitifs et composites), la sélection des règles à exécuter et leur déclenchement selon différents modes (immédiat synchrone, différé, détaché). L'indice de performance principal est le temps écoulé pour accomplir une tâche. Nous reportons dans [GBLR98] les résultats complets de l'évaluation. Les principales conclusions ont été les suivantes.

Généralement offrir plus de puissance d'expression entraîne un appauvrissement des performances. Dans le contexte des SGBD actifs augmenter la puissance d'expression peut concerner divers aspects dont le support d'événements plus riches. L'évaluation de performances réalisée a montré que le support d'un nombre important d'opérateurs pour la composition d'événements n'implique pas systématiquement la dégradation des performances. Par contre, le support de prédicats booléens dans les types d'événements tend à nuire aux performances. La mise en œuvre du détecteur d'événements implique des choix :

- d'architecture : détecteur global au système ou plusieurs détecteurs locaux aux classes ou par type d'événement.

- de technique : technique pour la reconnaissance des événements composites (par exemple arbres syntaxiques et réseaux de petri)

Concernant le choix d'architecture, il apparaît qu'un détecteur d'événements global (donc centralisé) donne des performances moins bonnes que plusieurs détecteurs dédiés soit par type d'événements soit par classe d'objets concernée. La détection d'événements spécifiques à une classe et le déclenchement de règles spécifiques se montre particulièrement performant. Ceci entraîne néanmoins une difficulté pour le support d'événements et de règles portant sur plusieurs classes. La technique de reconnaissance d'événements ne permet de dégager aucune conclusion claire. Par contre le choix du mode de production utilisé à un impact fort sur les performances. Certains modes de production peuvent entraîner une gestion lourde d'historiques d'occurrences d'événements. Le mode *récent* apparaît comme le plus performant car l'historique d'événements à maintenir est réduit. Il convient donc de le privilégier si sa sémantique convient aux applications.

Il a par ailleurs été constaté que les systèmes qui conservent systématiquement l'historique d'événements ont un sur-coût non négligeable. Il serait donc préférable d'offrir la persistance des historiques d'événements de manière sélective selon les besoins du domaine applicatif.

Notons que les aspects cités influent sur la capacité à monter en charge en terme de nombre d'événements. Par contre l'ensemble des prototypes support assez bien la montée en charge en terme de nombre de règles définies. En général, les règles sont indexées par type d'événement ou par classe et la qualité de ce choix se confirme.

Pour l'architecture globale, deux possibilités ont été étudiées pour ajouter des fonctions actives à un SGBD : l'intégration forte au sein du système et une approche par couches. Contrairement à nos attentes, les tests réalisés n'ont pas mis en évidence un net avantage

pour l'une ou l'autre des deux approches. L'approche intégrée permet néanmoins l'utilisation de techniques qui pourraient ne pas être mises en œuvre dans l'approche par couches. Cependant à technique égale les performances obtenues avec les deux approches sont du même ordre.

2.4 Visualisation d'exécutions événementielles

Malgré le large spectre applicatif, l'utilisation intensive de systèmes événementiels est confrontée à la difficulté de compréhension et de contrôle du comportement global en présence d'un grand nombre d'événements et de réactions.

Le comportement des systèmes à base d'événements a été largement étudié en s'appuyant sur le concept de modèle d'exécution. Un tel modèle spécifie comment gérer les événements, comment déclencher les réactions et comment les exécuter en prenant en compte le producteur de l'événement déclenchant et les applications concernées par la réaction [Cou96]. La grande variété de contextes d'utilisation a suscité la proposition de divers modèles d'exécution. Comprendre une exécution en présence de nombreux événements et de réactions simultanés ou en cascade est toujours difficile et cette difficulté est accrue par la diversité des modèles d'exécution existants.

Afin d'aider les développeurs dans leur tâche, nous avons adopté une approche pragmatique et proposé un service de visualisation d'exécutions d'applications événementielles [CRBL97, CRB99a]. Notre service, nommé Vizar, est général pour des systèmes à base d'événements. Il se base sur un ensemble de concepts (associés aux événements et aux réactions) communs à la plupart des systèmes à base d'événements. De plus, Vizar fonctionne avec un couplage très lâche vis-à-vis du système actif concerné.

Ce service permet d'enregistrer et de rejouer les applications pour visualiser les aspects propres à la partie événementielle. Une fois une exécution enregistrée, l'utilisateur active les vues qu'il souhaite visualiser. Vizar propose quatre vues consacrées respectivement :

1. aux définitions des événements et des réactions,
2. aux occurrences d'événements et leur composition,
3. aux ordonnancements des réactions déclenchées
4. et à l'historique des réactions exécutées.

Les vues sont synchronisées et peuvent être activées ou désactivées à tout moment. Un curseur permet à l'utilisateur de régler la vitesse de visualisation : il peut faire avancer la visualisation rapidement et la ralentir, voire la geler, sur les points qui l'intéressent le plus.

Un point central de ce travail est l'utilisation que nous avons faite des techniques de visualisation en trois dimensions. Ceci s'avère particulièrement pertinent pour la visualisation de systèmes événementiels impliquant une quantité importante de données en évolution [Bru99]. Nous avons utilisé différentes métaphores visuels, des représentations en 3D, des animations simples et des modifications dynamiques des paramètres graphiques tels que la couleur et la forme des objets qui représentent l'information graphiquement. Les choix que nous avons faits nous permettent de visualiser, de manière structurée, une grande quantité d'informations associées à un nombre arbitrairement large d'événements et de réactions. L'utilisateur peut naviguer pour se focaliser sur les événements et les réactions qui l'intéressent sans perdre de vue le contexte global de l'exécution.

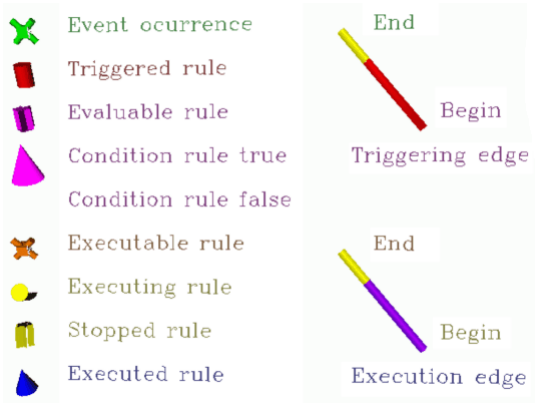

FIG. 2.2 – *Vizar: vue d'aide*

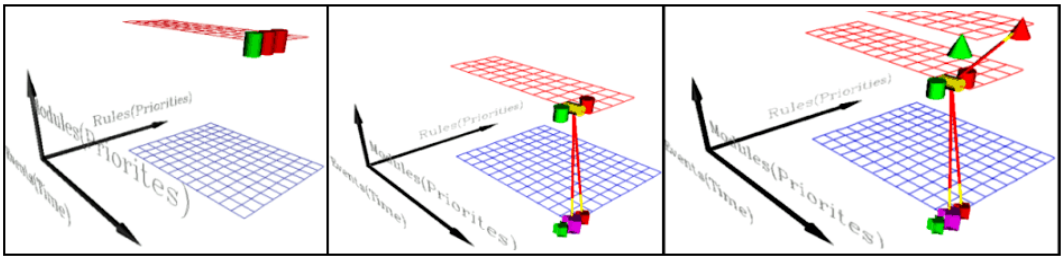

FIG. 2.3 – *Exemple de représentation graphique d'information dynamique liée aux réactions*

Un prototype de Vizar a été développé en utilisant VRML pour la partie graphique. A titre d'exemple, la figure 2.2 montre un écran d'aide de Vizar qui rappelle à l'utilisateur les conventions graphiques utilisées. La figure 2.3 reprend une séquence d'images de la vue dédiée aux ordonnancements des réactions. Les règles actives sont représentées par des objets dont l'apparence évolue selon le stade de l'exécution de la règle. Les règles sont structurées en modules représentés par des plans.

Au moment de la réalisation de ces travaux, d'autres outils d'aide au développement d'applications événementielles ont été proposés. Parmi eux nous pouvons citer des outils de simulation (VITAL [BGB95], Simbug for ADL [Beh94]), des outils qui visualisent des exécutions en directe[4] (DEAR [DJP93] for EXACT) et des outils de visualisation post-mortem (comme Vizar et [BMH99]).

L'originalité de notre travail réside dans deux points : (1) La généricité de Vizar ; excepté les outils de simulation les autres propositions sont dédiées à un système particulier. (2) L'utilisation des techniques de visualisation 3D. Celles-ci ouvrent un grand potentiel pour représenter des collections de nombreuses règles avec des déclenchements en cascade à plusieurs niveaux.

2.5 Autres travaux et conclusions

Nous avons présenté certains de nos travaux sur les SGBD actifs et plus particulièrement sur les événements. Ces travaux ont été réalisés dans l'optique d'avoir des SGBD offrant un grand nombre de fonctions pour la gestion des données. Les travaux autour des systèmes actifs ont énormément avancé et mûri donnant ainsi des bases solides pour envisager d'autres choix architecturaux.

Au sein de notre équipe plusieurs travaux ont été menés dans le but d'extraire des SGBD les fonctions actives et d'offrir des services d'événements et de réaction utilisables dans des contextes plus ouverts. Ces travaux ont fait l'objet des thèses de G. Vargas Solar [VS00] et H. Graziotin Ribeiro [GR00] réalisées sous la direction de Ch. Collet. [VS00] propose ADEES, un service d'événements permettant de spécifier et d'instancier des gestionnaires d'événements primitifs et composites. Le service repose sur deux méta-modèles : un méta-modèle de types et un méta-modèle de gestion d'événements. Les méta-modèles sont issus d'une taxonomie des modèles de définition et de gestion d'événements proposée dans la thèse. [GR00] propose ADRUS, un service de règles adaptable permettant la construction et le contrôle de gestionnaires de règles spécialisés selon les besoins des applications de fédérations de bases de données. Ce service s'appuie, entre autres, sur un méta-modèle de définition et de manipulation de règles et un méta-modèle d'exécution de règles.

Les services d'événements et de réaction proposés peuvent être utilisés conjointement ou non dans des contextes divers. Comme nous le verrons dans le chapitre 5 qui présente un service de transactions, le service d'événements peut, par exemple, être utilisé pour implanter des formes d'adaptabilité contextuelle, notamment dans des environnements mobiles. D'un point de vu de la recherche plus globale l'approche événementielle est très largement utilisée à différentes fins dont la coopération entre services et composants.

4. Approche metteur au point

Chapitre 3

Caches sémantiques

Les contributions présentées dans ce chapitre marquent le début de nos efforts pour extraire de la globalité du SGBD des fonctions de gestion de données. Nous cherchons à offrir des composants de gestion de données utilisables dans la construction d'intergiciels. Nous nous intéressons ici à des intergiciels consacrés à l'évaluation de requêtes sur des sources de données hétérogènes, autonomes et distribuées sur le World Wide Web. Nos apports portent sur la définition d'un cache de requêtes dans un tel intergiciel.

3.1 Contexte et objectif

L'utilisation de données largement distribuées et surtout leur interrogation n'a fait que croître depuis plusieurs années. Particulièrement encouragé par les grandes quantités d'information accessibles via le WWW, de nombreux travaux autour des systèmes de recherche d'information généraux ou spécifiques à un domaine d'application ont été réalisés [Chi00]. Les outils proposés n'étant pas toujours efficaces, ni complets, des systèmes de type meta-moteur de recherche ont vu le jour (voir figure 3.1). Ils permettent de chercher de l'information auprès de diverses sources, généralement hétérogènes (moteurs de recherche, SGBD ou autre). Un meta-moteur de recherche reçoit la requête utilisateur, la transmet aux sources concernées via leur adaptateur. Au retour des réponses des sources, les adaptateurs les traitent pour en extraire la partie à donner au niveau supérieur qui compose la réponse de la requête originale. Les adaptateurs prennent en charge une bonne partie de l'hétérogénéité structurelle et des langages des sources utilisées.

Les meta-moteurs de recherche partagent leurs principaux choix architecturaux avec les systèmes de médiation [Wie92, DD99, CBB+04]. Nous avons travaillé dans le contexte d'intergiciels orientés WWW mais la plupart de nos résultats présentés ci-après s'appliquent aux systèmes de médiation.

L'objectif de nos travaux était de contribuer à améliorer la qualité du service rendu aux utilisateurs des meta-moteurs de recherche. Un meta-moteur de recherche est « client » des sources d'information qui contribuent à l'évaluation des requêtes utilisateur. Dans la plupart de systèmes client-serveur, les performances sont améliorées par l'utilisation des ressources de calcul ou de stockage du client. L'utilisation de caches dans le contexte du Web est particulièrement importante vu le trafic réseaux et les performances des divers sources de données. Néanmoins, les techniques classiques de cache (niveau page ou n-uplet) ne sont pas très adaptées aux meta-moteurs de recherche à cause de l'hétérogénéité et l'autonomie des sources accédées. Nous avons ainsi proposé un service de cache sémantique permettant d'optimiser l'évaluation des requêtes posées via un meta-moteur de recherche sur des sources hétérogènes, autonomes et distribuées. Nous avons également étudié les

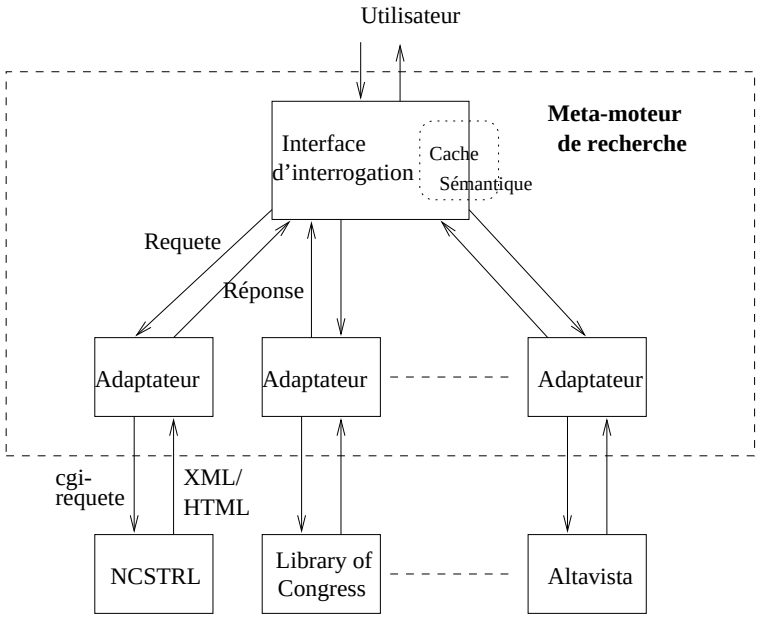

FIG. 3.1 – *Architecture d'un meta-moteur de recherche*

opportunités de faire du pré-chargement dans ce contexte et proposé une technique de pré-chargement adaptée.

Ces travaux ont été en partie réalisés dans le cadre des DEA de S. Drapeau [Dra99] et de M-L. Schneider [Sch98]. Sur ce thème nous avons mené une coopération avec B. Chidlovskii et L. Julliard, de Xerox Research Center Europe. Le cache proposé a été transféré dans le produit Ask Once [Ask] de cette société.

La suite du chapitre aborde en premier la proposition du cache sémantique (Section 3.2) puis les aspects liés au pré-chargement (Section 3.3). La Section 3.4 conclut le chapitre et évoque des travaux que nous avons réalisés sur la publication sur le web de données issues de SGBD.

3.2 Cache sémantique et requêtes multi-sources

3.2.1 Langage de requêtes et définition du cache

Considérons un meta-moteur de recherche offrant un langage permettant de sélectionner des documents en précisant des valeurs sur des attributs. Une requête est une conjonction de termes ; chaque terme est de la forme *Attribut Opérateur Valeur*. *Attribut* est un nom d'attribut spécifique à un domaine (par exemple, `Titre`, `Auteur` pour le domaine bibliographique), *Opérateur* peut être l'égalité ou l'inclusion (mots-clés *Equals* ou *Contains*) et *Valeur* est une chaîne de caractères. Par exemple, la requête

$$Q = \texttt{Titre}\ \textit{Contains}\ \texttt{``applet''}\ \texttt{AND}\ \texttt{Resumé}\ \textit{Contains}\ \texttt{``java''}$$

cherche les documents dont le titre contient le mot `applet` et le résumé le mot `java`. Une conjonction peut aussi utiliser la négation

$$Q1 = \texttt{Titre}\ \textit{Contains}\ \texttt{``applet''}\ \texttt{AND NOT}\ \texttt{Titre}\ \textit{Contains}\ \texttt{``netscape''}$$

où le titre doit contenir ``applet'' et pas ``netscape''.

Nous nous situons dans un contexte où les utilisateurs raffinent leurs requêtes successivement, par exemple, en ajoutant ou enlevant un terme. Ainsi, l'objectif est de pouvoir réutiliser les réponses déjà rapatriées afin d'améliorer le temps de réponse et diminuer le trafic et les coûts économiques en cas d'accès à des sources payantes.

Notons qu'un cache de pages, comme ceux implantés dans les SGBD ou les systèmes d'exploitation, ne serait pas utilisable dans ce contexte. La raison est que dans leur approche l'unité de transfert est la page: le client détecte les défauts de page et demande les pages explicitement au serveur. Dans notre cas, ceci n'est pas réalisable car souvent les sources d'information (serveurs) peuvent être accédées exclusivement par des requêtes/mots-clés et le meta-moteurs de recherche (client) ne connaît pas l'organisation en pages des serveurs. L'approche cache de n-uplets est utilisé dans certains caches Web ou *proxys* en se basant sur l'URL des documents WWW. Dans notre contexte, avec des requêtes booléennes, ceci est moins intéressant: d'une part l'URL ne fait pas partie de la requête utilisateur et d'autre part les pages réponse des sources (et leurs URL) sont souvent générées dynamiquement. La réutilisation des données ainsi cachées devient difficile.

Pour pallier à ces inconvénients nous proposons l'utilisation d'un cache sémantique [DFJ+96, CB98] qui gère le cache client comme un ensemble de régions sémantiques. Une région réunit des données sémantiquement reliées et couvertes, par exemple, par une requête. L'unité d'accès et de remplacement du cache est la région.

Quand une requête Q est posée à un meta-moteur muni d'un cache sémantique, elle est d'abord envoyée au service de cache. Le cache divise Q en deux parties: requête cachée et requête restante. La requête cachée ($Probe(Q)$) représente une réponse partielle à Q pouvant être obtenue du cache. La requête restante ($Rem(Q)$) concerne les données qui doivent être demandées aux sources d'information. Si la requête restante n'est pas vide, elle est évaluée de manière classique. La réponse finale est élaborée à partir des réponses des deux parties. Par exemple, supposons que le cache contient une région avec les réponses à la requête

$R =$`Titre` *Contains* ``applet'' `AND Résumé` *Contains* ``java''.

La requête posée par un utilisateur est

$Q =$`Titre` *Contains* ``applet''.

Dans ce cas,

$Probe(Q) =$ `Titre` *Contains* ``applet'' `AND Résumé` *Contains* ``java'' et,
$Rem(Q) =$`Titre` *Contains* ``applet'' `AND NOT Résumé` *Contains* ``java''.

La partie des réponses issue de $Probe(Q)$ peut être rendue à l'utilisateur immédiatement. La réponse complète est obtenue par l'union des réponses des deux parties ($Probe(Q) \cup Rem(Q)$).

3.2.2 Architecture du cache

.

Le cache est structuré comme un ensemble de régions sémantiques ou chacune regroupe les réponses obtenues pour une requête R_i. La requête est mémorisée dans le descripteur de région correspondant. La gestion d'un tel cache nécessite, entre autres, la définition d'une stratégie de fusion, division et remplacement des régions. L'utilisation des données cachées se base sur la relation existant entre la requête Q posée et celles dont la réponse

est dans le cache (R_i). Par abus de langage, dans la suite nous dirons qu'une requête R1 est contenue dans une requête R2 si la réponse à R1 est un sous-ensemble de la réponse à R2. Une « région » R du cache désigne la réponse stockée pour la requête R. Quatre cas d'utilisation du cache sont considérés (Fig. 3.2) :

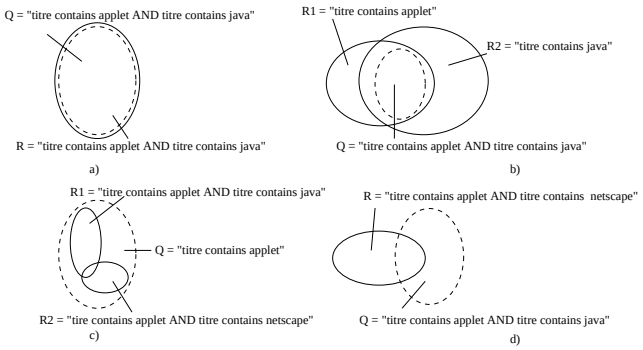

FIG. 3.2 – *Modes d'utilisation des régions du cache*

- *Équivalence*: le cache contient une région R tel que $Q = R$ (Fig. 3.2.a).
- *Inclusion de la requête* : Q est contenue dans une ou plusieurs régions $R_1, \ldots, R_m, m \geq 1$, : $Q \subset R_i$ (Fig. 3.2.b);
- *Inclusion de région*: le cache contient des régions $R_1, \ldots, R_m, m \geq 1$, incluses dans la requête posée: $R_i \subset Q$ (Fig. 3.2.c).
- *Différence d'un terme*: il s'agit d'un cas particulier d'intersection où une région R_i et Q diffèrent sur un terme $|R_i - Q| = 1$. Le terme différence est noté d. Sur la figure 3.2.d, la région $R =$ `''title contains applet AND title contains netscape''` diffère d'un terme de $Q =$ `''title contains applet AND title contains java''`. Le terme différence est $d =$ `''title contains netscape''`

Ces quatre modes d'utilisation sont exploités pour la construction de $Probe(Q)$ et $Rem(Q)$ permettant ainsi différents degrés de réutilisation des données cachées.

3.2.3 Hétérogénéité des sources de données

Réponses complètes et vérifiables L'hétérogénéité des sources peut être de différente nature: sémantique, au niveau des langages, structurelle parmi d'autres. La gestion de l'hétérogénéité est, en grande partie, prise en charge par les adaptateurs. Néanmoins, dans le contexte des sources d'information sur le WWW nous dégageons deux caractéristiques qui influencent la réutilisation des données cachées. Il s'agit de caractéristiques des réponses fournies par une source de données. Est-ce que la réponse obtenue est complète? Est-ce que la réponse obtenue peut être vérifiée?

Une réponse est dite complète si elle contient la totalité des réponses dont dispose la source. Les sources d'information peuvent fonctionner selon un modèle booléen, un modèle par *ranking* ou une combinaison des deux. Avec le modèle booléen les sources retournent les données vérifiant une requête booléenne et la réponse sera complète. Par *ranking*, la source va ordonner les réponses selon un critère de pertinence et va retourner les k meilleures. La réponse peut être complète ou non selon la sélectivité de la requête.

Le choix du modèle dépend de divers aspects tels que le domaine et la taille de la base. Pour le bon fonctionnement du cache il est nécessaire d'avoir une détection dynamique de la complétude des réponses. Ceci sera déterminé, par exemple, par les adaptateurs, et ensuite exploité par le cache.

Les réponses de certaines sources d'information sur le web ne sont pas des n-uplets stockés dans la source mais plutôt une représentation réduite à quelques éléments. Cette réponse peut par exemple contenir un URL et quelques lignes de texte et ne pas inclure toute l'information utilisée par la source pour évaluer la requête. Dans ce cas, la réponse est dite non vérifiable, sinon elle l'est (voir des exemples dans [CRS99]). En accord avec ces caractéristiques, quatre possibilités de réponse, complète ou non et vérifiable ou non, doivent être gérées par le cache. Chaque type de réponse donne lieu à une gestion particulière et leur taux de réutilisation sera varié.

Cache multi-sources Notre proposition de cache est adaptée à l'interrogation multi-sources. Dans les choix de conception d'un cache multi-sources, nous avons considéré la création d'une région sémantique réunissant des réponses issues de plusieurs sources ou la possibilité d'avoir des régions indépendantes par source. Le premier choix oblige à gérer l'hétérogénéité au sein d'une région. Le deuxième choix, que nous privilégions, donne plus de souplesse et permet une meilleure gestion de la mémoire occupée. Notons, que gérer un cache par source d'information obligerait à attribuer statiquement la mémoire allouée à chaque source. Avec un cache multi-sources, cette allocation est faite dynamiquement selon les requêtes du moment.

Gestion de l'hétérogénéité dans le cache Les problèmes posés par l'hétérogénéité au niveau du cache sont, d'une part, comment réutiliser des réponses complètes ou non, vérifiables ou non et, d'autre part, comment gérer le cache en tenant compte de ces caractéristiques. Notre proposition tient compte des caractéristiques des réponses dans la politique de remplacement et de fusion des régions. L'utilisation des régions diffère selon les caractéristiques de la réponse stockée. Des précisions sur la gestion de l'hétérogénéité sont présentées dans [CRS99]. A titre d'illustration nous introduisons ci-après l'algorithme d'utilisation d'une région avec des réponses complètes et vérifiable (cas le plus avantageux). Cet algorithme est présenté de manière synthétique dans la table 3.1.

Algorithme 1 : utilisation d'une région complète et vérifiable .
Entrée: cache sémantique et requête Q.
Sortie: réponse à Q et mise à jour du cache.
Comparer Q avec les requêtes stockées dans les descripteurs des régions du cache.

- *Équivalence :* si $Q = R$ la région contient la réponse, $Probe(Q) = R$. La valeur de remplacement de la région, $Rpc(R)$, devient la meilleure (*most-recently-used*). Cette valeur sert à la fonction de remplacement du cache.

- *Inclusion de la requête :* $Q \subset R_i, i = 1, \ldots, m$: Si Q est incluse dans une ou plusieurs régions, choisir celle de cardinalité minimale, $R = min(R_i)$. Sélectionner à partir de R, la réponse de Q: $Probe(Q) = Q \cap R$. Augmenter la valeur de remplacement $Rpc(R)$ en proportion de la partie de la région utilisée.

- *Inclusion de région* $R_i \subset Q, i = 1, \ldots, m$: Renvoyer les n-uplets des régions identifiées, en éliminant les copies, $Probe(Q) = \cup R_i$. Construire la requête restante comme: $Rem(Q) = Q - (\cup R_i)$ et l'envoyer à la source de données. A la réception de

la réponse, la rendre à l'utilisateur et fusionner les régions R_i avec $Rem(Q)$ dans une région correspondant à Q. La valeur de remplacement de la région est la meilleure.

- *Différence d'un terme* $|R_i - Q| = 1, i = 1, \ldots, m$: détecter les termes différence d_i pour les régions $R_i, i = 1, \ldots, m$. Sélectionner les n-uplets répondant à Q à partir des régions identifiées, en éliminant les copies : $Probe(Q) = Q \cap (\cup R_i)$. Construire la requête restante comme : $Rem(Q) = Q - (\cup d_i)$ et l'envoyer à la source de données. A la réception de la réponse, la rendre à l'utilisateur, créer une nouvelle région pour $Rem(Q)$ et augmenter les valeurs de remplacement $Rpc(R_i)$ en proportion des parties des régions utilisées.

L'ordre des cas indique la priorité donnée (sauf exception [CRS99]). Si aucun des cas n'est détecté, Q est envoyée à la source de données de la manière habituelle et donnera lieu à une nouvelle région dans le cache.

Cas :	Équivalence	Inclusion de requête	Inclusion de région	Différence d'un terme
Formule	$Q \equiv R$	$Q \subset R_i$, $R = min(R_i)$	$R_i \subset Q$, $i = 1, \ldots, m$	$\|R_i - Q\| = 1$, $i = 1, ..., m$
Prob(Q)	R	$Q \cap R$	$\cup R_i$	$Q \cap (\cup R_i)$
Rem(Q)	$\emptyset$	$\emptyset$	$Q - \cup R_i$	$Q - \cup d_i$
Régions cache	pas de changement	pas de changement	fusion Q avec R_i	nouvelle région $Q - \cup d_i$
Valeur de remplacement	meilleur $Rpc(R)$	modif. proportionnelle $Rpc(R)$	meilleur $Rpc(Q)$	modif. proportionnelle $Rpc(R_i)$

TAB. 3.1 – *Cache de réponses complètes et vérifiables*

3.3 Pré-chargement de requêtes et cache sémantique

Les propositions présentées précédemment ont été complétées par des travaux sur des techniques de pré-chargement [DRBG00]. Il s'agit ici d'exécuter à l'avance des requêtes qui seront vraisemblablement posées dans un futur proche et de stocker dans le cache leur résultat en vue d'améliorer le temps de réponse perçu par l'utilisateur. Le pré-chargement est pratiqué dans divers contextes mais définir une bonne stratégie de pré-chargement (quoi et quand pré-charger) reste une tâche difficile. Les stratégies peuvent être de nature déterministe ou statistique [1] : dans le premier cas l'utilisateur définit la stratégie alors que dans le deuxième, le système tente de la deviner en s'appuyant, en général, sur des patrons d'utilisations observées.

Notre proposition intègre les deux approches [Dra99]. La contribution originale porte sur la proposition d'un algorithme de génération de règles pour faire du pré-chargement statistique. Cet algorithme adapte et étend un algorithme de fouille de données ([AMS+] [BL97]) afin de générer des règles d'association sur des requêtes. Les règles générées sont de la forme $q_1 ... q_k \Rightarrow q_n$, et possèdent un taux de confiance. Elles reflètent le fait que les requêtes citées ont une relation sémantique car elles sont posées de manière rapprochée dans une session [2].

1. Aussi dite spéculative
2. L'algorithme utilise une fenêtre temporelle

L'algorithme de génération de règles opère en mode déconnecté sur un journal de requêtes posées par une communauté d'utilisateurs. Par exemple, sur un journal $\mathcal{L}$ de requêtes parmi lesquelles :

$$Qa = \texttt{Titre}\ \textit{Contains}\ \texttt{``applet''}\ \texttt{AND}\ \texttt{Titre}\ \textit{Contains}\ \texttt{``java''}$$
$$Qb = \texttt{Titre}\ \textit{Contains}\ \texttt{``java''}$$
$$Qc = \texttt{Titre}\ \textit{Contains}\ \texttt{``caching''}$$

Le résultat de l'algorithme de génération de règles est de la forme :

$$Qc \Rightarrow Qa \text{ avec confiance } 0.5,$$
$$Qc \Rightarrow Qb \text{ avec confiance } 0.5,$$
$$Qa \Rightarrow Qc \text{ avec confiance } 1.0 \text{ et}$$
$$Qb \Rightarrow Qc \text{ avec confiance } 1.0.$$

Ce résultat indique que si la requête Qa ou la requête Qb est posée, il est intéressant de pré-charger le résultat de la requête Qc , car la probabilité qu'elle soit posée à sont tour est élevée. Avec un taux de confiance plus faible, le pré-chargement de Qa ou de Qb peut être fait suite à la requête Qc.

Une variante de l'algorithme tient compte de la relation d'inclusion qui peut exister entre les requêtes. Connaissant le potentiel de réutilisation des requêtes dans le cache sémantique, l'idée est de pré-charger en priorité les requêtes dont le résultat contribue aussi à la réponse d'autres requêtes fréquemment posées. Il s'agit de requêtes pour lesquelles le cache sémantique peut appliquer les cas d'inclusion introduits dans la section 3.2.

Dans l'exemple précédent, en sachant que $Qa \subseteq Qb$, l'importance de pré-charger Qa croît, car elle servira à répondre à Qa , mais elle servira aussi comme réponse partielle à Qb.

3.4 Autres travaux et conclusions

Ce chapitre a présenté notre proposition de service de cache pour l'interrogation dans un contexte multi-sources sur le web. Dans ce même contexte nous avons mené des travaux sur l'exportation de données qui ne seront pas détaillés dans ce document. Ces travaux concernent la publication, en tant que documents Web, de données gérées par des SGBD (DEA de N. Henry [Hen98] encadré conjointement avec Ch. Collet). Dans ce cas, les documents web peuvent être considérés comme une forme de vue sur des données de la base avec toutes les difficultés de maintenance que cela entraîne. Nous avons réalisé ces travaux entre 1998 et 1999 au moment des premières propositions de langage tel que XML et XSL de la part du W3C. Nous avons proposé (DEA de L. Rodriguez [Rod99]) et mis en œuvre (Projet d'ingénieur de P.Cohen, Ecole Polytechnique [Coh99]) une solution permettant de spécifier de manière déclarative le contenu et la présentation des documents Web en question [RR99]. Le contenu est spécifié par des requêtes OQL [3] et la présentation par un langage de définition de masques permettant de définir facilement des présentations pour les réponses aux requêtes. ITTA, l'outil de génération développé, traite les requêtes et les masques et produit les documents XML (avec leur DTD) et XSL correspondant. Par rapport à l'existant, cette proposition facilite grandement l'exportation de données avec différents formats web. Le contenu (requêtes OQL) et les présentations (masques) peuvent être modifiés et réutilisés aisément. Par ailleurs, l'utilisateur n'est pas tenu de se plonger dans l'utilisation d'un langage de présentation assez complexe tel que XSL. Le

3. Standard proposé par l'ODMG [Cat97]

prototype ITTA a été expérimenté avec le SDBD Orienté Objet O2 ; une variante utilisant une interface JDBC donnant accès à un SGBD relationnel a également été développée.

Nous avons identifié des pistes de recherche intéressantes liées à la visualisation de données, telles que la présentation de grands volumes d'objets ou des présentations multimédia de données qui ne le sont pas. Plus généralement, l'exportation des données sur le web a ouvert un grand nombre d'axes de recherche comme le confirme la littérature scientifique des dernières années.

Dans la suite, nos recherches se sont focalisées sur des aspects plus « système » de la gestion de données tels que le service de cache introduit ici. Les travaux sur le cache sémantique restent d'actualité et des utilisations dans d'autres contextes ont été proposées. Par exemple, pour l'interrogation dans des environnements mobiles, dans le support de requêtes dépendantes de la localité.

Plus généralement, la diffusion des applications et des systèmes répartis à grande échelle contribue à l'intérêt des travaux autour des caches et de la duplication de données. Nous verrons dans la suite notre contribution sur ce thème où nous travaillons tout particulièrement sur les questions de cohérence entre copies.

Chapitre 4

Services de duplication de données

Ce chapitre est consacré à nos travaux sur le support de la duplication de données. Nous suivons une approche de séparation des considérations afin d'obtenir des services adaptables et utilisables dans des *intergiciels pour la gestion de données* ou pour créer des *SGBD configurables*.

La duplication est au cœur de nombreuses approches de tolérance aux pannes ou pour l'amélioration des performances de systèmes gérant des données. Nos propositions sur les caches mentionnées dans le chapitre 3 en sont un exemple. Nous avons étudié en profondeur la cohérence d'objets dupliqués et avons proposé un canevas adaptable de services de duplication [DRD02, DRD02, DRD03]. Cette proposition fait l'objet de la thèse de Stéphane Drapeau [Dra03], réalisée sous mon encadrement, en coopération avec P. Déchamboux du département DTL/ASR de France Télécom.[1]

4.1 Contexte et objectif

Le support de la duplication a été largement étudié dans divers domaines tels que les systèmes de communication de groupes, les mémoires partagées réparties, les gestionnaires de fichiers répartis, les systèmes de gestion de bases de données réparties et les systèmes à objets répartis [GHOS96, KA00, Kin99, DR01, VM02]. Ces travaux ont donné lieu à un grand nombre de techniques et de protocoles . Malheureusement, il n'y a que très peu de rapprochement entre les projets menés dans ces différents domaines. Ainsi, de nombreuses solutions sont proposées, chacune ayant fait ses propres choix. Certaines solutions sont orientées vers une catégorie d'applications précise, d'où, finalement, leur nombre et leur variété. Ces choix particuliers conditionnent, entre autres, la cohérence entre les copies, les performances, la tolérance aux fautes, la transparence ou bien encore la difficulté de programmation. Par suite, le choix et la mise en œuvre de la duplication de données pour une application ou un système particulier est une tâche difficile.

La motivation globale de notre travail est de faciliter la gestion de données dupliquées en offrant le moyen de disposer de services de duplication aux caractéristiques variées. Un tel service prend en charge les problèmes de cohérence liés à la présence de plusieurs copies d'un même objet. Nous avons ainsi revu la fonction duplication afin d'offrir de l'adaptabilité dans sa gestion. Notre proposition se présente sous forme d'un canevas (figure 4.1) qui permet la création de services de duplication mettant en œuvre des protocoles de duplication divers. Chaque service de duplication pourra être créé avec les caractéristiques les plus adaptées au contexte d'utilisation. Un service de duplication peut être vu comme

1. Thèse sous contrat CIFRE.

un composant (complexe) intégrable dans un intergiciel offrant éventuellement d'autres fonctions de gestion de données.

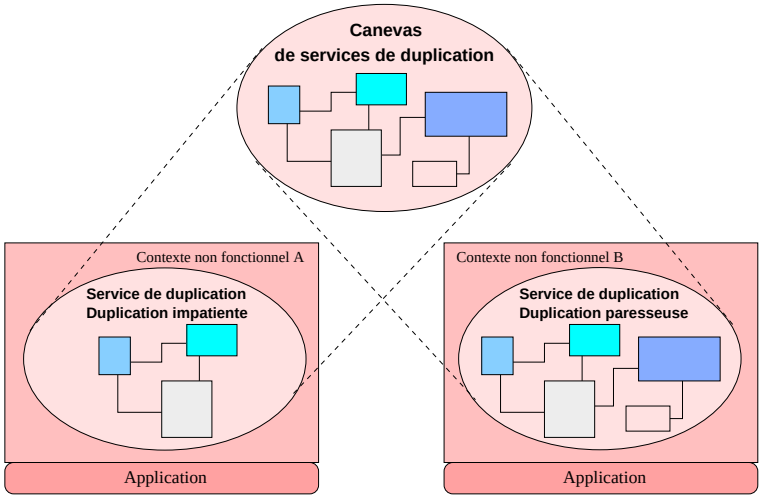

FIG. 4.1 – *Création de services de duplication à partir d'un canevas*

Une gestion adaptable de la duplication nous semble importante au vu de l'évolution actuelle des systèmes informatiques. L'utilisation répandue des systèmes répartis, l'apparition des systèmes informatiques à grande échelle (grand nombre d'unités participant et forte dispersion géographique) et des environnements mobiles (réseaux sans fil et unités mobiles tels que PDA, téléphone et portables) introduisent de nouveaux besoins en terme de duplication. Les environnements à grande échelle et mobiles présentent une grande hétérogénéité, variabilité et évolution, aussi bien au niveau des moyens d'exécution qu'au niveau des exigences à un moment donné. Les ressources offertes peuvent être extrêmement différentes selon que l'on utilise un assistant personnel, une station de travail ou un autre type de poste. De plus, les éléments constitutifs du système, les performances du réseaux et le système lui même sont soumis à d'importantes variations au cours du temps. Ceci contraste avec les environnements « classiques », où le développement et l'exécution d'applications s'effectuent en connaissant le contexte d'exécution supposé stable.

Ainsi selon le contexte et l'objectif recherché (par exemple, la tolérance aux fautes ou augmenter la disponibilité), un concepteur de systèmes doit dupliquer des données avec des contraintes variables. Un support adaptable peut lui permettre d'offrir le protocole de duplication le mieux adapté. Il peut également, toujours à l'aide de ce support, essayer et comparer divers protocoles sur ses données. Cette approche relativement récente dans le contexte des bases de données [SZ97, Ham99, CW00, DG01] redéfinit de façon assez nouvelle les principes des systèmes de gestion de bases de données.

Nous proposons un canevas qui permet d'obtenir des services de duplication implantant différents protocoles et offrant différentes garanties sur la cohérence des copies. Les contributions de ce travail portent sur trois axes : la modélisation des services de duplication que peuvent être créés, l'adaptabilité par rapport au contexte non fonctionnel et l'adaptabilité dans les protocoles de duplication ou de certaines parties de ces protocoles.

Notre contribution permet ainsi d'avoir les protocoles adaptés pour chaque type de données et de contexte. L'adaptation est faite au moment de la création du service au cours de la phase de développement du système qui va l'utiliser. Dans des recherches futures, il serait possible d'explorer une adaptabilité dynamique permettant de changer de

service de duplication pendant l'exploitation du système.

Afin d'offrir une gestion adaptable de la duplication, nous avons cherché à remplir les trois conditions suivantes.

- La première est la séparation des préoccupations, qui veut que la portée fonctionnelle du canevas soit exclusivement la duplication. Les fonctions fondamentales du canevas incluent la gestion des liens entre copies en considérant leur cycle de vie et la cohérence entre les copies d'un objet. La cohérence entre les copies d'un objet, appelé *modèle de cohérence locale* peut être mise en œuvre par des protocoles différents. Le niveau applicatif effectue le choix de modèle de cohérence et de protocole appropriés.

- La deuxième condition est la capacité à s'adapter à divers contextes non fonctionnels, par exemple persistants ou non, transactionnels ou non. Ceci nécessite l'indépendance vis-à-vis d'interactions potentielles avec d'autres aspects tels que le contrôle de la concurrence ou des techniques de point de sauvegarde.

- La troisième condition est la décomposition du support de la cohérence en plusieurs composants, chacun jouant un rôle réduit. Nous dégageons deux axes à suivre deux axes: une décomposition fonctionnelle et une décomposition structurelle. La définition de ces composants est probablement la contribution principale de ce travail. Une décomposition à un niveau fin, comme celui que nous proposons, permet aussi un réglage fin des protocoles de cohérence entre copies. Ceci peut être fait par l'assemblage des composants mis en œuvre de la manière la plus appropriée.

D'autres travaux (tels que Garf [GGM95], Core [BC98], RepliXa [KG96] et Globe [VHT99]) ont déjà expérimenté la séparation des préoccupations entre le code applicatif et le code gérant la duplication. Néanmoins, ces travaux proposent plus qu'un simple canevas de duplication car ils combinent protocoles de duplication, contrôle de la concurrence et cohérence globale. Notre proposition cherche à clairement isoler la duplication pour augmenter le potentiel de réutilisation des services dans divers contextes non-fonctionnels (deuxième condition ci-dessus).

Concernant le support des protocoles de duplication, les travaux actuels peuvent être classés en deux catégories: ceux dédiés à la tolérance aux fautes et ceux proposant une approche plus générique. Dans la première catégorie, les travaux se limitent à implanter une cohérence forte entre les copies en utilisant des techniques telles que la duplication active ou passive [LS94a], [GGM95], [OMG97], [NMMS02], [FGS98]. Dans ce cas la diversité est réduite. Les systèmes de la seconde catégorie offrent un spectre plus large de techniques de duplication et permettent le support de différents protocoles de duplication (par exemple Core [BC98] et Globe [KKvST98]). C'est également le cas de notre proposition. Cependant,contrairement aux travaux cités, nous proposons une décomposition fine des protocoles de duplication en composants permettant d'offrir aussi l'adaptabilité de parties des protocoles de duplication. Ceci facilite la prise en compte de nouveaux besoins et permet une réutilisation à un grain plus fin que le protocole entier.

La suite de ce chapitre introduit les éléments clés de nos propositions. La portée du canevas et des services sera précisée dans la section 4.2. La section 4.3 concerne l'adaptabilité des protocoles de cohérence locale. Elle introduit un protocole abstrait de cohérence locale qui est un niveau d'abstraction permettant de travailler sur la grande variété de protocoles existants. Cette section propose aussi une décomposition fonctionnelle des protocoles. La

section 4.4 évoque brièvement l'adaptabilité du canevas au contexte non fonctionnel. La section 4.5 mentionne les expérimentations de ce travail et nos conclusions.

4.2 Un canevas de services de duplication

4.2.1 Portée du canevas

Le canevas de duplication RS2.7 définit la structure générale d'un service de duplication. Il autorise différentes décisions de conception pour ensuite être instancié comme un service de duplication approprié au contexte d'utilisation. Des services de duplication divers mettant en œuvre des protocoles variés peuvent ainsi être créés.

Malgré l'existence de plusieurs propositions de support de la duplication il n'existe pas aujourd'hui un consensus sur la définition et les fonctions d'un tel support. Dans la définition du canevas nous avons considéré la séparation des préoccupations comme un élément clé pour l'adaptabilité. Ainsi, nous ne proposons pas un canevas couvrant tous les aspects liés à la duplication car ceci réduirait ses possibilités de réutilisation. RS2.7 se concentre sur deux aspects : le support du cycle de vie de groupes de copies (leur création et suppression) et les protocoles de synchronisation entre copies (appelé protocole de cohérence locale dans la suite). Notons que la gestion d'objets dupliqués n'est généralement pas réduite aux deux aspects proposés par RS2.7. Une politique de duplication nécessite, en plus, la définition des points suivants :

- le moment de duplication définissant l'instant de création ou destruction des copies,
- le degré de duplication définissant le nombre de copies qu'il doit y avoir dans le système,
- le placement des copies définissant où mettre les copies dans le système et
- la cohérence à assurer entre les copies, selon un protocole de cohérence locale.

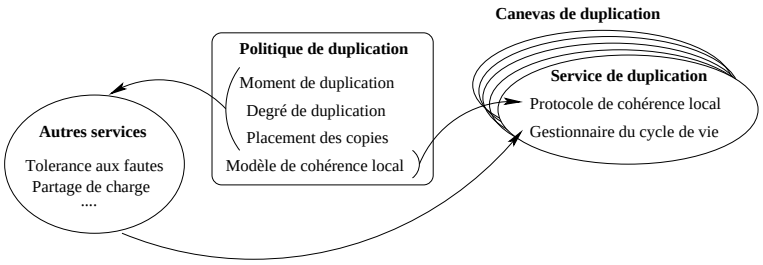

FIG. 4.2 – *Politique de duplication / canevas de duplication / service de duplication*

Le protocole de cohérence locale est à la charge d'un service obtenu à partir de RS2.7, alors que les autres points cités ne le sont pas. D'autres services ou modules utilisant la duplication prennent ces décisions et s'appuient sur le gestionnaire du cycle de vie du service de duplication pour créer et/ou détruire des copies selon leurs besoins (figure 4.2). La duplication peut être faite, par exemple, dans un but de partage de charge ou de tolérance aux fautes. Un service de duplication est dirigé par d'autres services ou par une couche de transparence qui a conscience du placement des copies, du degré de duplication et du moment de création/destruction des copies. Dans le cas extrême toutes ces décisions peuvent être prises par l'application.

4.2.2 Modèles de cohérence locale et protocoles

Les besoins des systèmes/applications en terme de cohérence des copies de ses objets dupliqués donnent lieu à l'instanciation de divers services de duplication issus du canevas proposé. Certaines applications nécessitent une cohérence forte alors que d'autres peuvent tolérer une certaine divergence entre les copies. Par exemple, une application peut adopter un protocole *maître-esclaves* simple alors qu'une autre adoptera un protocole *ROWA*(*Read One, Write All [BG84]*). Nous adoptons la notion de modèle de cohérence locale qui définit comment les utilisateurs perçoivent les différentes copies d'un objet. Ceci peut être considéré comme un contrat entre le service de duplication et ses utilisateurs. Un tel contrat inclut la définition des événements d'accès (lecture/écriture) et de synchronisation (opérations pour synchroniser des copies). Ce niveau d'abstraction introduit de la souplesse dans la gestion des copies car un même modèle de cohérence locale peut être mis en œuvre par plusieurs protocoles. Nous classons les modèles de cohérence locale (et leur protocoles) en quatre groupes (ci-après 1, 2, 3a, 3b) :

1. **Le modèle à copie unique** considère toute les copies comme équivalentes. Toute lecture sur une copie renvoie la donnée à jour et les écritures s'effectuent sur des données à jour. La synchronisation préserve l'illusion d'une copie unique mais peut ne pas être effectuée à chaque opération. Des protocoles tel que ROWA, ROWAA (*Read One, Write All Available* [GSC+83]), ceux à base de quorum [Gif79, KA88, AA90], la duplication active [Sch90, PV91], la duplication passive [PV91, BMST93] et la duplication impatiente [GHOS96] (dans les SGBD) mettent en œuvre ce modèle.

2. **Le modèle à copies divergentes** est un modèle de cohérence faible puisque les copies peuvent diverger. Néanmoins, il est possible de caractériser la divergence en donnant certaines garanties sur les opérations de lecture et écriture et leur ordre d'exécution. Par exemple, on assure qu'une séquence d'opérations sur une copie particulière est perçue dans le même ordre par les autres copies ou que la précédence entre les opérations est respectée. Les protocoles mettant en œuvre des modèles à copies divergentes sont fréquemment utilisés dans les mémoires partagées réparties car ils contribuent à des modèles de *cohérence globale* populaires [Kin99] (par exemple cohérence causale, PRAM et à l'entrée).

3. **Les modèles à copies convergentes** tolèrent la divergence des copies à condition qu'elles convergent à un moment donné. La limite de tolérance de la divergence peut être exprimée par des conditions telles qu'un délai, une période, un point dans le temps, une version, une contrainte numérique ou un événement particulier [GN95]. La synchronisation des copies est effectuée lorsque la condition précisée est violée. Les accès aux copies qui ne respectent pas les conditions ne seront pas autorisés. Les incohérences temporaires ne constituent pas un problème pour beaucoup d'applications et les tolérer permet d'améliorer les performances. Dans cette catégorie nous distinguons deux sortes de modèles :

   (a) **Modèle à copies convergentes avec lectures sur les copies divergentes :** tout objet dupliqué a un propriétaire qui possède la valeur courante. Les mises à jours (toujours effectuées sur des copies à jour) sont d'abord faites sur le propriétaire de la donnée et ensuite propagées vers les autres copies. Les lectures peuvent être faites sur n'importe quelle copie. Les protocoles maître-esclaves paresseux [GHOS96] et l'epsilon-serialisabilité [PL91, RP95] (dans les

SGBD) mettent en œuvre ce modèle. Dans un protocole de duplication paresseux maître-esclaves, les écritures sont traitées par le maître dans une transaction et les esclaves sont mis à jour de manière asynchrone dans des transactions séparées.

(b) **Modèle à copies convergentes avec écritures sur les copies divergentes :** les lectures et les écritures peuvent être effectuées sur des copies non à jour et de manière concurrente. Le mécanisme de duplication doit gérer les situations de conflit de manière à ne pas perdre de mises à jour. Des protocoles tels que *Multi AirLine Reservation* [BGM90] ou certains utilisés dans le contexte mobile [PB99a] mettent en œuvre ce modèle.

La mise en œuvre de modèles de cohérence locale nécessite différentes opérations de la part de la tolérance aux fautes et du contrôleur de concurrence. Par exemple, le modèle à copie unique nécessite que les écritures sur les copies d'un objet se fassent de manière exclusive, alors que pour les autres modèles cela n'est pas nécessaire. Les besoins et les interactions de la duplication et de la tolérance aux fautes et le contrôle de concurrence sont indiqués dans [Dra03]. Le support de ces fonctions ne fait pas partie intégrante des services de duplication. Ceci est fait dans un souci de séparation des aspects pour favoriser l'adaptabilité des services de duplication.

4.3 Adaptabilité des protocoles de cohérence locale

Notre proposition se doit de supporter différents protocoles de cohérence locale. Cette section montre comment RS2.7 offre différents protocoles de cohérence locale et comment nous obtenons l'adaptabilité dans des parties de ces protocoles. Pour obtenir cette adaptabilité, nous proposons deux factorisations des protocoles de cohérence locale qui nous permettent de regrouper les points communs des protocoles. Ceci dans le but de construire des composants réutilisables et d'introduire des points d'adaptabilité dans les protocoles. Il est possible de remplacer un composant par un autre, d'en enlever, d'en rajouter ou encore de modifier les interactions entre eux afin d'adapter les protocoles aux besoins des applications et du contexte non fonctionnel.

La première factorisation nous permet d'obtenir une décomposition structurelle des protocoles (section 4.3.1). Elle permet d'exhiber les phases communes aux protocoles de cohérence locale ainsi que l'ordonnancement de ces phases. La deuxième factorisation est fonctionnelle (section 4.3.2) et nous permet de distinguer les différentes fonctions présentes dans ces protocoles pour dégager des composants.

4.3.1 Un protocole de cohérence locale abstrait

Le grand nombre de protocoles existants rend complexe la conception d'un canevas. Pour notre proposition nous nous basons sur la constatation que les protocoles diffèrent principalement dans la manière et l'ordre dans lequel ils accomplissent différentes phases. Afin de fournir un canevas de duplication générique, nous proposons une abstraction définissant les protocoles de cohérence locale. Cette abstraction, correspondant à un protocole abstrait, consiste en cinq phases génériques : *accès*, *coordination*,*exécution*, *validation* et *réponse*. Les protocoles se différencient suivant l'approche utilisée dans chaque phase et leur ordre d'exécution des phases. Ce protocole abstrait de cohérence locale nous permet de définir un canevas de duplication dont la conception est indépendante de tout protocole.

Afin d'introduire les cinq phases, considérons un objet client qui envoie une requête à un objet dupliqué.

Phase d'accès : un objet client soumet une requête à l'objet dupliqué. La duplication est transparente si l'objet client interagit avec un objet logique et n'est pas conscient des copies (ni de leur nombre, ni de leur localisation). Par contre, si la duplication est non transparente, l'objet client envoie sa requête directement à une ou plusieurs copies.

Phase de coordination : cette phase inclut les traitements préliminaires à l'exécution de la requête. Si nécessaire, les copies se coordonnent pour synchroniser l'exécution de la requête. Cette phase peut également inclure des actions de coordination avec le service de contrôle de concurrence et/ou le service de tolérance aux fautes.

Phase d'exécution : la requête demandée est exécutée sur la(les) copie(s).

Phase de validation : les copies vérifient qu'elles sont d'accord avec les résultats de l'exécution. Par exemple, elles peuvent décider si il est nécessaire de défaire ou de refaire certaines actions.

Phase de réponse : le résultat de l'opération est renvoyé à l'objet client. Les protocoles ont l'alternative de renvoyer la réponse une fois que tous les traitements et phases sont terminés ou de la renvoyer dès qu'elle est disponible (même si certaines phases ne sont pas terminées).

La mise en œuvre d'un protocole nécessite l'ordonnancement des phases et leur implantation. Ceci peut varier d'un protocole à un autre. Certains protocoles peuvent ignorer des phases, itérer sur certaines ou encore les regrouper. Afin d'illustrer ceci, considérons quatre protocoles de cohérence locale particuliers et une requête d'écriture sur un objet dupliqué.

- Le protocole ROWA implante un modèle de cohérence locale à copie unique (section 4.2.2, modèle 1). La phase d'accès reçoit l'opération d'écriture et la transmet à la phase de coordination. Cette dernière informe le contrôleur d'accès concurrents qui s'assure qu'un seul objet client accède à l'objet dupliqué. La phase de coordination transmet ensuite la requête à toutes les copies, qui l'exécutent (i.e., phase d'exécution). Les phases de validation et de réponse sont inexistantes.

- Le protocole ROWAA implante également le modèle à copie unique. Les phases d'accès, de coordination et d'exécution sont les mêmes que pour le protocole ROWA. Cependant, afin de prendre en compte les copies défaillantes, une phase de validation est ajoutée. Lors d'une opération d'écriture le même processus que précédemment se déroule ; suite à la phase d'exécution, la phase de validation est déclenchée. Si cette phase remarque qu'une copie a été indisponible lors du processus de synchronisation, une boucle entre les phases de validation et de coordination est nécessaire. La copie défaillante est supprimée du groupe de copies gérées par le protocole de cohérence locale et la phase de coordination est ré-exécutée (suivie de la phase d'exécution et de validation). Une copie à nouveau opérationnelle est mise à jour de manière asynchrone lors d'une phase de coordination.

- Le protocole de duplication paresseux maître/esclaves implante le modèle à copies convergentes avec lecture sur les copies divergentes (section 4.2.2, modèle 3a). Pour

ce protocole trois phases sont nécessaires : la phase d'accès, la phase de coordination et la phase d'exécution. Si l'écriture est faite sur un esclave, la phase d'accès retransmet cette requête au maître. La phase d'accès du maître retransmet cette opération à la phase de coordination. La phase de coordination enregistre l'opération puis l'exécute sur le maître lors d'une phase d'exécution. De manière asynchrone, la phase de coordination du maître va se déclencher afin de construire un message de synchronisation pour les esclaves à partir des informations enregistrées. Les mises à jour sont effectuées sur les esclaves lors d'une phase d'exécution.

- Pour la duplication active [PV91] basée sur des primitives de communication, les phases d'accès et de coordination peuvent être regroupées, comme pour les phases de validation et de réponse. Les primitives de communication peuvent assurer un ordre total fiable des messages.

Le protocole de cohérence locale abstrait fait apparaître une décomposition structurelle en phases. Pour chaque phase, nous définissons un composant avec une interface générique. Ainsi, il est possible de changer certaines phases d'un protocole de cohérence locale particulier afin de l'adapter.

4.3.2 Architecture fonctionnelle

L'un des objectifs poursuivi est d'offrir l'adaptabilité dans tout ou partie des protocoles de cohérence locale. Dans ce but, nous proposons une décomposition fonctionnelle des protocoles en identifiant la phase concernée par chaque fonction. Chaque fonction sera assurée par un composant pouvant être mis en œuvre de différentes façons. Ces composants peuvent être réutilisés et remplacés pour donner lieu à des protocoles différents selon les besoins.

La figure 4.3 résume la proposition en mettant en colonne les phases où interviennent les composants fonctionnels. Par souci de clarté dans la présentation, les lignes regroupent les composants comme suit :

- D'abord les *composants communs à tous les modèles de cohérence locale*. Ils permettent de construire des protocoles simples et concernent principalement la gestion du cycle de vie des copies, la synchronisation et les interactions avec l'application utilisatrice.

- Ensuite, les *composants dépendant du modèle de cohérence locale*

- et pour finir des *composants dépendant du protocole* .

La plupart des composants peuvent être mis en œuvre de manières diverses permettant ainsi un niveau fin d'adaptabilité. La suite introduit brièvement les différents composants. Voir [Dra03] pour une présentation détaillée incluant les interfaces des composants et les interactions avec les aspects contrôle de concurrence et la tolérance aux fautes.

Composants communs aux modèles

Ces composants matérialisent les fonctions de base des protocoles de cohérence locale. Nous isolons des fonctions de ce type pour les phases d'accès, de coordination, d'exécution et de réponse. Il n'y en a pas pour la phase de validation, car elle n'intervient pas dans de nombreux protocoles.

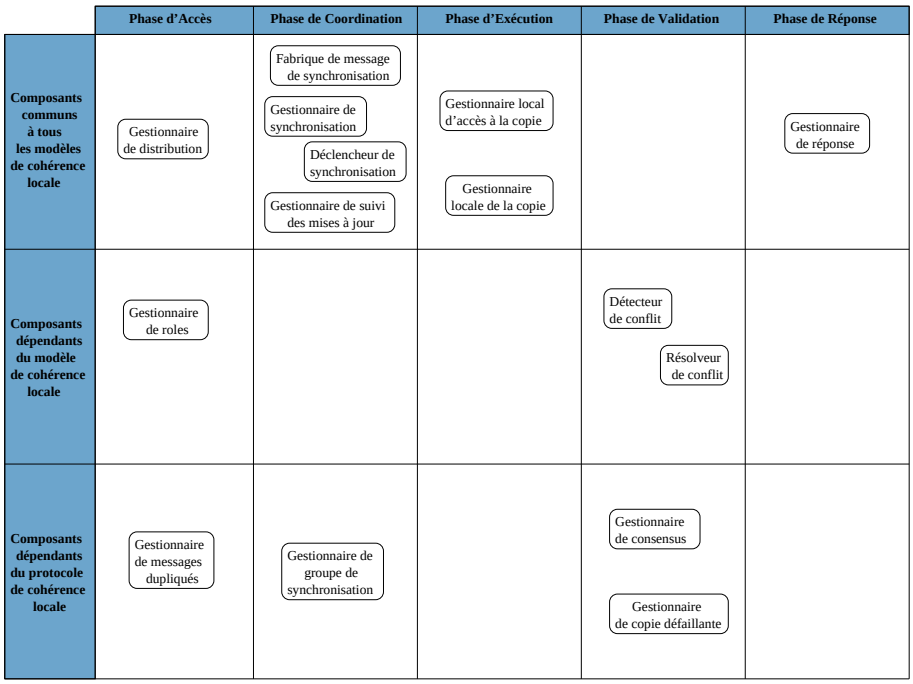

FIG. 4.3 – *Les composants de l'architecture fonctionnelle de RS2.7*

Phase d'accès : le seul composant ici est le *Gestionnaire de distribution* . Son rôle est de décider vers quelles copies (une, un groupe, toutes) retransmettre les requêtes provenant d'un objet client.

Phase de coordination : cette phase introduit le *Gestionnaire de synchronisation* des copies et d'autres composants requis pour cette tâche. *Le déclencheur de synchronisation* décide du moment de la synchronisation. Il peut s'appuyer sur l'occurrence d'une opération, d'un événement externe (par exemple, synchronisation déclenchée à heure fixe) ou d'événements internes (par exemple, synchronisation toutes les n requêtes). *Le gestionnaire de suivi des mises à jour* garde trace de ce qui est fait sur les copies. Cette information sert ensuite à construire les messages liés à la synchronisation. Diverses mises en œuvre peuvent être faites (utilisation de journaux, de déclencheurs, de copies ombres, etc). La *Fabrique de messages de synchronisation* construit des messages de demande de synchronisation ou de mise à jour en récupérant les informations auprès du gestionnaire de suivi des mises à jour.

Phase d'exécution : les composants introduits ici permettent l'exécution des opérations tout en gardant la généricité de notre canevas. Afin de permettent de dupliquer tout type d'objet (objets simples ou composites) le *Gestionnaire local d'accès à la copie* offre une interface permettant les lectures/écritures sur une copie. Ce composant donne une représentation abstraite des copies. Le *Gestionnaire local de la copie* encapsule tout ce qui a trait à la gestion d'une copie (par exemple, son chargement en mémoire). Certains traitements peuvent être nécessaires avant de « activer » ou de « passiver » une copie.

Phase de réponse : son seul composant, le *Gestionnaire de réponse*, décide du résultat à retourner pour la requête, autrement dit, de la valeur reflétant l'objet dupliqué. Cette

valeur ne représente pas forcément la valeur de toutes les copies. Cela dépend du modèle de cohérence locale mis en œuvre. Ainsi, suivant les cas, ce composant peut attendre les réponses d'une copie particulière, de plusieurs copies ou de l'ensemble des copies.

Composants dépendants du modèle de cohérence locale

Phase d'accès : les modèles de cohérence locale se caractérisent par le rôle des copies. Le *gestionnaire de rôles* précise ce qu'une copie a le droit de faire suite à une requête d'un objet client. Par exemple, seules certaines copies peuvent être autorisées à exécuter des modifications.

Phase de validation : le traitement des modifications conflictuelles requiert l'introduction du *Détecteur de conflits* et du *Résolveur de conflits*. Ces composants sont utilisés par des protocoles implantant le modèle de cohérence locale à copies convergentes avec écriture sur des copies divergentes.

Composants dépendant des protocoles de cohérence locale

Phase d'accès : certains protocoles sont sensibles au problème des appels dupliqués : lorsqu'un objet dupliqué A envoie une requête à un objet dupliqué B, si A est composé de n copies, B peut recevoir n copies de la requête. Le *Gestionnaire de messages dupliqués* peut gérer cette situation de différentes façons, du coté du client (A) ou du coté du serveur (B).

Phase de coordination : le *Gestionnaire de groupe de synchronisation* décide des copies qui participent au processus de synchronisation. Pour les protocoles à base de quorum, par exemple, la coordination concerne une certaine partie des copies pour une écriture et une autre pour une lecture.

Phase de validation : pendant cette phase, certains protocoles nécessitent des actions en rapport avec la tolérance aux fautes. Le *Gestionnaire de consensus* a pour charge d'obtenir un consensus et le *Gestionnaire de copies défaillantes* détecte les copies défaillantes [2] et décide que en faire. Une copie à nouveau opérationnelle, pour certains protocoles, doit rattraper son retard afin de devenir effectivement opérationnelle.

Un service de duplication issu de RS2.7 implante un protocole particulier : les phases réalisent un certain nombre de fonctions en mettant en œuvre les composants respectifs. Certains protocoles peuvent requérir la mise en œuvre de l'ensemble de composants alors que des protocoles simples en utilisent un nombre réduit. Un composant peut être implanté de divers manières. La réutilisation de composants et la possibilité de remplacement permet l'adaptation de parties d'un protocole et l'obtention de nouveaux protocoles en se basant sur l'existant.

4.4 Adaptabilité au contexte non fonctionnel

Nous avons vu les fonctions des services de duplication issus du canevas et les éléments sur lesquels repose l'adaptabilité des protocoles de cohérence locale. Nous abordons ici, l'insertion des services de duplication dans différents contextes non fonctionnels : contextes

2. Interaction avec le service de tolérance aux fautes

transactionnels ou non, persistants ou non, etc. Ceci nous amène à considérer la gestion des objets plus globalement. Généralement, les applications s'appuient sur un modèle de cohérence globale qui est une spécification plus ou moins formelle de la manière dont la mémoire est vue par l'application. Ces modèles sont mis en œuvre par des protocoles manipulant les objets et prennent en compte la duplication, le contrôle de la concurrence et la tolérance aux fautes suivant le contexte. Nous proposons qu'en présence de duplication, un modèle de cohérence globale s'appuie sur un modèle de cohérence locale (figure 4.4).

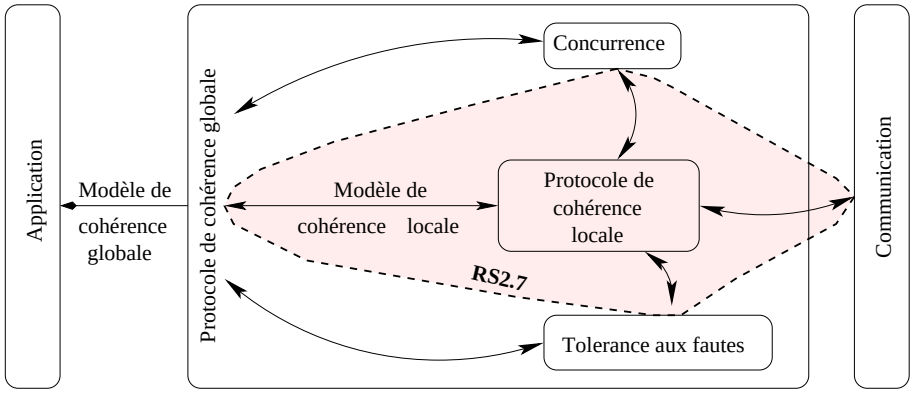

FIG. 4.4 – *Support de modèles de cohérence globale*

Les modèles de cohérence globale sont utilisés dans divers domaines. Dans les mémoires partagées réparties, ils spécifient la valeur à retourner par un événement de lecture issu de l'exécution parallèle d'un programme. De nombreux modèles de cohérence globale existent [Kin99]: séquentiel, causale, PRAM, à cohérence faible, à l'entrée et à la libération.

Les SGBD proposent aussi des modèles de cohérence globale. Un des critères de correction le plus populaire pour les bases de données dupliquées est la sérialisabilité sur une copie. Ce critère de correction assure que l'exécution sur des données dupliquées est équivalente à une exécution sur une seule copie et que l'exécution des transactions est serialisable [BCF+97]. Notons que dans ce cas le protocole de cohérence globale tient compte du contrôle des accès concurrents, la tolérance aux fautes et la duplication. Les supports offrant des transactions ACID mettent en œuvre ce protocole de cohérence globale.

Dans notre approche, nous proposons que les aspects spécifiques à la duplication soient gérés par le protocole de cohérence locale de manière à dégager le protocole de cohérence globale de cet tâche. Pour cela, il a été nécessaire de dégager les interactions entre les services de duplication et d'autres aspects portant sur les objets du système, mais pas nécessairement des objets dupliqués. [Dra03] montre comment un même modèle de cohérence locale peut participer à plusieurs modèles de cohérence globale. Ainsi, un protocole de cohérence locale mis en œuvre par RS2.7 peut être utilisé dans divers protocoles de cohérence globale.

4.5 Conclusion et perspectives

Notre travail contribue à clarifier la séparation des fonctions propres à la duplication dans le but d'améliorer l'adaptabilité et d'augmenter la réutilisabilité du code chargé de la duplication. Le canevas de duplication proposé permet de créer des services fournissant un support

– pour la gestion du cycle de vie des copies

– pour gérer la cohérence entre les copies d'un même objet.

Un service créé est utilisé par l'application ou par d'autres services pour construire des politiques de duplication diverses. Les protocoles de cohérence locale offerts par le service mettent en œuvre le modèle de cohérence locale désiré.

Un modèle de cohérence locale définit comment les utilisateurs perçoivent les différentes copies d'un objet dupliqué. Quatre modèles de cohérence locale sont distingués : le modèle à copie unique, le modèle à copies divergentes, le modèle à copies convergentes avec lecture sur les copies divergentes et le modèle à copies convergentes avec écriture sur les copies divergentes. Une grande variété de protocoles peuvent implanter ces modèles. Le modèle de cohérence locale participe à la définition du modèle de cohérence globale. Cette séparation entre modèle de cohérence locale et globale est importante car elle permet d'utiliser des services de duplication dans des contextes aussi différents que des contextes transactionnels ou non, des mémoires partagées réparties, etc.

Par ailleurs notre proposition a été conçue pour que les protocoles de cohérence locale soient adaptables. Ainsi, deux décompositions des protocoles de cohérence locale sont proposées. Le protocole de cohérence locale abstrait, première décomposition de nature structurelle, définit cinq phases : l'accès, la coordination, l'exécution, la validation et la réponse. L'adaptabilité dans tout ou partie des protocoles est également assurée par une architecture spécifiant les composants fonctionnels pour construire de tels protocoles. Ceux-ci permettent aussi une meilleure articulation entre un service de duplication et ceux assurant la tolérance aux fautes ou le contrôle de concurrence.

Nous avons travaillé sur la duplication dans le cadre des projets IST PING [EFI+01] et du Contrat de Recherche Externe (CRE) « Services bases de données ouverts pour applications coopératives » avec France Télécom. Ces deux projets ont porté sur le développement d'intergiciels pour le support de mondes virtuels distribués partagés. La gestion de la duplication d'objets est particulièrement importante dans ce contexte. Nous avons mené plusieurs expérimentations et deux prototypes (développés majoritairement par S. Drapeau) sont aujourd'hui opérationnels :

– Le prototype intégré à la plate-forme du projet PING s'est focalisée sur la séparation d'aspects entre protocoles de cohérence locale, contrôle de concurrence et protocoles de cohérence globale.

– Le deuxième prototype, réalisé dans le cadre du CRE, met en œuvre la décomposition fonctionnelle et structurelle du canevas de duplication proposé. Divers services de duplication ont été créés pour la plate-forme de monde virtuels Continuum. Il s'agit de plusieurs protocoles de nature maître-esclaves et de protocoles autorisant des mises-à-jour sur toutes les copies. Des variantes impatientes et paresseuses ont été dérivées en utilisant plusieurs moments de synchronisation (temporelle et guidé par le nombre de mises à jour).

Le potentiel de réutilisation des composants fonctionnels pour la création de protocoles à partir d'autres protocoles a bien été validé. Cependant le développement de protocoles de duplication reste une tâche d'une certaine difficulté.

Ces expériences ont aussi mis en avant l'importance de disposer d'outils de composition statique et optimisée des composants constituant les protocoles de cohérence locale. Dans le cadre du projet PING, nous avons effectué une telle optimisation manuellement. La différence de taille du code entre un protocole optimisé et l'approche avec décomposition

est de l'ordre d'un facteur 10. Cela est principalement dû au nombre important d'objets créés lors de la décomposition. Pour poursuivre ce point il serait intéressant d'utiliser le canevas pour composants Fractal[3] d'ObjectWeb [CLB+01]. Celui-ci permet de décrire des assemblages et de générer des implantations optimisées.

Notons que de telles optimisations pourraient être adoptées dans de nombreux cas car plusieurs catégories d'applications n'ont pas besoin d'une adaptation dynamique du protocole de cohérence. La propriété d'adaptabilité et la réutilisabilité des composants est surtout utile dans ce cas lors de la conception d'un protocole afin d'obtenir un service parfaitement adapté aux besoins.

L'adaptabilité dynamique des protocoles constitue en soi une perspective de recherche importante. Il nous semble particulièrement intéressant d'explorer une adaptation contextuelle des protocoles, i.e., selon l'état du contexte courant le système emploie un protocole avec des caractéristiques appropriées.

3. http://www.objectweb.org/architecture/component/index.html

Chapitre 5

Service de transactions adaptables

Ce chapitre est consacré à nos travaux sur l'aspect transactionnel. Notre proposition se situe au niveau intergiciel pour offrir un support transactionnel à des applications travaillant sur des sources de données autonomes, hétérogènes et distribuées dans un environnement variable comportant des unités mobiles et fixes.

Nous avons réalisé ce travail dans le cadre de la thèse de P. Serrano Alvarado [SA04] [1] avec la participation de M. Adiba et de C. Labbé (enseignants-chercheurs au laboratoire LSR-IMAG).

5.1 Introduction

La gestion de données est classiquement réalisée dans un cadre transactionnel qui prend en charge plusieurs aspects liés à la cohérence. La notion de transaction est une abstraction à laquelle les systèmes associent des fonctions telles que le contrôle de la concurrence des accès, la vérification de contraintes d'intégrité et l'exécution atomique d'un ensemble d'opérations. Les SGBD offrent le support de transactions centralisées sur une seule base ou distribuées sur plusieurs bases plus ou moins intégrées, hétérogènes et autonomes. La forte utilisation de données distribuées a conduit à la proposition et à l'utilisation de moniteurs transactionnels permettant la validation atomique de transactions s'exécutant sur des systèmes faiblement couplés.

Notre travail sur les services de transactions est motivé par le besoin de support transactionnel dans des environnements variables. Nous nous plaçons dans un contexte distribué intégrant des unités fixes reliées par un réseau câblé et des unités mobiles utilisant des réseaux sans fil. Notre objectif a été de fournir un support transactionnel adéquat à des applications s'exécutant dans un tel contexte tout en offrant aux utilisateurs un certain contrôle sur la qualité du service rendu. Comme nous le verrons dans la suite, les particularités des environnements mobiles, liées au réseau sans fil et aux unités mobiles, rendent les solutions transactionnelles classiques inappropriées. Notre proposition offre une adaptabilité contextuelle particulièrement motivée par l'environnement mobile mais qui est suffisamment générale pour être utilisée dans d'autres contextes aux caractéristiques variables.

Dans la suite de ce chapitre, la section 5.2 introduit la gestion de transactions dans des environnements mobiles et les grandes lignes de notre proposition. La section 5.3 se focalise sur le modèle de transactions adaptables proposé. La section 5.4 présente l'intergiciel TransMobi supportant le modèle de transactions adaptables et les protocoles adoptés

1. Thèse financée par le Conacyt - Mexique.

pour assurer la correction des exécutions. La section 5.5 conclut et présente certaines perspectives de recherche.

5.2 Supports transactionnels pour des environnements mobiles

5.2.1 Propositions actuelles

La technologie de communication 3G UMTS, la prolifération du Wi-Fi[2] et l'évolution des caractéristiques des unités mobiles[3] contribuent à la croissance de la popularité et à la diversité des applications en environnement mobile (par exemple des services dépendant de la localité [YACS03]). Cette évolution défie l'actuelle technologie de gestion de bases de données car les approches traditionnelles ne sont pas toujours directement applicables au contexte mobile [BBB+03]. En effet, les applications et les systèmes s'exécutant dans un environnement mobile doivent faire face aux caractéristiques des réseaux sans fil et des unités mobiles, qui sont souvent variables et restreintes. La communication sur les réseaux sans fil est asymétrique, le débit de transfert de données est variable, les déconnexions - volontaires ou non - peuvent être fréquentes et le coût monétaire des communications peut être un facteur non négligeable. De plus les unités mobiles (téléphone, PDA, ordinateur portable), présentent des ressources variées en termes d'autonomie (batterie), de robustesse, de capacité de stockage et de calcul.

Ces caractéristiques affectent divers aspects de la gestion de données dont les transactions. Les raisons fondamentales sont que la plupart des modèles et des protocoles considèrent que les parties sont « toujours » joignables et que les déconnexions sont une situation d'erreur (et non une situation normale). De plus ils supposent une communication symétrique entre les sites (clients/serveurs) et ne tiennent pas compte des possibles contraintes des ressources des unités mobiles. Sur un autre plan, il peut également être intéressant de tenir compte du coût monétaire induit par l'exécution d'un protocole ou plus généralement d'une transaction.

De nombreux travaux ont porté sur les supports transactionnels pour environnements mobiles. Dans [SARA04], nous présentons une analyse approfondie des propositions de recherche et introduisons certains produits. Parmi les travaux de recherche, nous remarquons *Clustering* [PB95, PB99b], *HiCoMo* [LH02], *IOT* [LS95, LS94b], *Pro-motion* [WC99, WC97], *Reporting* [Chr93], *Semantics-based* [WC95], *Prewrite* [MB01] et *Pre-serialization* [DG98, DG00]. Ces travaux revisitent différents aspects des noyaux transactionnels pour mieux intégrer les contraintes des environnements mobiles et de domaines d'application particuliers. La plupart permettent l'exécution de transactions sur une unité mobile en mode déconnecté. Seulement certains, tels que *Reporting* et *Clustering*, autorisent aussi une exécution répartie entre une unité mobile (UM) et des unités fixes (UF) en mode connecté.

Nous constatons que le support de transactions avec les propriétés ACID classiques s'avère particulièrement difficile et coûteux dans les environnements mobiles. On retrouve l'intérêt des modèles transactionnels souples avec des transactions imbriquées ouvertes, à longue durée de vie ou coopératives [Chr93]. L'intérêt des transactions à longue durée de vie vient des possibles déconnexions (non bornées) des participants. Nous remarquons également (comme dans *Clustering, HiCoMo* et *IOT*) la distinction de deux types de transactions selon si elles travaillent sur des données garanties à jour ou non. L'objectif

2. WLAN IEEE 802.11b et g
3. Par exemple leur capacité mémoire et de calcul

est de permettre l'exécution tentative de transactions sur des UM déconnectées et une exécution définitive éventuelle sur le serveur de référence (sur une UF). Dans quelques propositions comme [GHPS96], l'exécution définitive tolère un certain niveau de divergence par rapport à l'exécution tentative. Les propositions adoptant deux sortes de transactions gèrent, en général, au moins deux versions des données qui peuvent, pour certaines, être structurellement différentes. Les protocoles de contrôle de concurrence et de validation sont également revus pour tenir compte des versions de données et des transactions (par exemple dans *Clustering* et *Prewrite*). Dans ce cas, la validation d'une transaction aura une validation locale à l'unité mobile et une validation définitive lors de l'intégration sur le serveur. [4]

En ce qui concerne le contrôle de la concurrence, certaines approches intègrent des aspects sémantiques ou temporels (*Promotion* en est un bon exemple) ou adaptent des protocoles standards de manière à réduire le nombre de messages transitant sur les réseaux sans fil (par exemple, adaptation du protocole de verrouillage à deux phases [JBE95]). Le même genre de modification a été réalisé sur des protocoles de validation comme dans *UCM* [BPA00] et *TCOT* [KPDS02, Kum00].

Un autre groupe de travaux, incluant *Kangaroo* [DHB97], *MDSTPM* [YZ94] et *Moflex* [KK00], porte sur le support de transactions initiées par une unité mobile mais exécutées sur le réseau fixe par un SGBD qui assure les propriétés transactionnelles. Dans ce cas, la contribution porte sur la gestion des déplacements et des déconnexions des unités mobiles.

Parmi les solutions proposées, nous identifions le souci d'introduire des points de flexibilité concernant les modèles transactionnels utilisés. Il s'agit d'offrir de la souplesse pour que les transactions puissent être structurées et exécutées en accord avec les besoins applicatifs et les contraintes de l'environnement d'exécution : exécution de transactions en mode déconnecté, exécution distribuée et économie des ressources. Notre contribution cherche à aller plus loin dans cette flexibilité en intégrant une adaptabilité selon le contexte d'exécution. Le programmeur peut spécifier des alternatives d'exécution dont une sera choisie par le système en fonction de l'état courant du contexte d'exécution. Ces alternatives donneront différents niveaux de performance et permettront aux programmeurs d'influer sur la qualité du service. Notre proposition capitalise certains des résultats précédents et constitue un service de transactions utilisable sur des SGBD autonomes. Nous adoptons un modèle transactionnel souple permettant l'exécution de transactions réparties sur plusieurs unités mobiles ou fixes. Nous proposons également un protocole de validation approprié et un intergiciel mettant en œuvre le service à proprement parler. La suite de ce chapitre présente les grandes lignes de notre proposition. Pour une présentation détaillée il convient de se référer à [SA04, SARAL03]

5.2.2 Environnements mobiles et adaptation contextuelle

Les caractéristiques des environnements mobiles nous amènent à introduire une adaptation contextuelle en se basant sur la perception de l'environnement. Nous dégageons les dimensions de l'environnement susceptibles de varier et d'avoir un impact sur l'exécution des transactions. Les dimensions peuvent être divisées en trois groupes : celles portant sur des caractéristiques des réseaux sans fil, celles concernant l'unité mobile et les dimensions plus sémantiques.

- Les caractéristiques des réseaux sans fil qui nous intéressent sont l'état de la connexion, le débit de la bande passante et le coût monétaire de la communication. En effet, selon le réseau sans fil utilisé, le débit de la bande passante peut être très variable,

4. Notons qu'il ne s'agit pas du protocole de validation à deux phases.

les déconnexions fréquentes et la facturation significative (facturation au volume de données transféré ou au temps de connexion).

- Concernant l'unité mobile, on s'intéresse à son autonomie en batterie, sa capacité de calcul et sa disponibilité en mémoire (cache ou persistante).
- Les caractéristiques sémantiques portent sur des aspects propres à l'application concernée ou au profil utilisateur. Nous incluons ici la localité (endroit où se trouve l'unité mobile) et le temps de connexion estimé avant la prochaine déconnexion. Les déconnexions pouvant être volontaires ou non, cette dernière caractéristique pourrait aussi être considérée dans les groupes précédents.

La définition des dimensions de l'environnement est flexible afin de permettre l'introduction de dimensions « sur mesure » . Par exemple, la qualité des données requises (fraîcheur) par une application peut devenir une dimension.

Pour chaque dimension, nous considérons un ensemble d'états possibles. Par exemple l'état de la connexion peut être *connecté* ou *déconnecté*; le débit, le prix de communication, la batterie, le cache et la mémoire disponibles reflètent un certain niveau de qualité, *bon*, *moyen* ou *mauvais*. En pratique, ces états sont traduits par des intervalles dans les unités de mesure correspondantes. Pour les unités mobiles, la dimension localité concerne sa position géographique.

Les états de l'environnement sont représentés par des *descripteurs d'environnement* qui indiquent les dimensions pertinentes et leurs valeurs.

Definition 1 *Le Descripteur de l'environnement (DE) contient un ensemble de dimensions et le(s) état(s) correspondants, DE = {Dimension = état(s)}*

Comme nous le verrons dans la suite, les définitions des transactions incluront des descripteurs d'environnement spécifiant le(s) état(s) requis pour une certaine exécution. Si plusieurs sites participent à l'exécution d'une transaction, le descripteur peut décrire ou non l'état requis pour chaque site.

5.3 Modèle de transactions mobiles adaptables

Le modèle AMT, *Adaptable Mobile Transactions*, que nous proposons permet aux programmeurs de définir des alternatives transactionnelles pour une tâche applicative. Nous cherchons à offrir un cadre transactionnel souple qui associe des informations contextuelles et des alternatives d'exécution pour une action. Nous considérons des tâches ou actions impliquant des transactions sur une ou plusieurs bases de données ou sites. Le développeur d'applications peut envisager divers choix pour accomplir une même action, chacun d'eux correspondant à un certain critère de performance adapté à un état du contexte d'exécution. Une transaction de type AMT, notée T_{AMT}, permet d'introduire une ou plusieurs *alternatives d'exécution* associées chacune à un *descripteur d'environnement*. Un tel descripteur exprime l'état du contexte d'exécution requis pour le lancement de l'alternative correspondante (par exemple connexion et localité).

Les alternatives peuvent être sémantiquement équivalentes et l'exécution réussite de l'une d'entre elles représente l'exécution correcte d'une T_{AMT}. Ainsi, lorsqu'une T_{AMT} est lancée, le support examine l'état courant de l'environnement et démarre l'alternative d'exécution appropriée. Si l'état de l'environnement mobile ne permet l'exécution d'aucune alternative, l'exécution de la T_{AMT} est reportée. Dans ce cas, une alternative sera démarrée aussitôt qu'un état acceptable survient.

Le modèle AMT a été inspiré de Sagas [GMS87], DOM [BOH+92] et Flex [ELR90]. Dans ces derniers, il est possible de définir des transactions « équivalentes » qui seront exécutées en cas de panne. Nous étendons cette idée de manière à gérer les variations de l'environnement mobile[5] : nous introduisons la perception du contexte d'exécution et adoptons des propriétés adaptées aux caractéristiques des environnements mobiles. L'étude analytique que nous avons effectuée sur le modèle AMT montre qu'avec lui, le programmeur a un moyen de maîtriser certains aspects des performances de l'exécution des transactions. Ces aspects peuvent être le taux de réussite des transactions, les messages échangés ou encore le coût financier induit par l'exécution et les communications [SARAL03].

5.3.1 Définition d'une transaction T_{AMT}

Les transactions T_{AMT} définies avec le modèle AMT ont une structure arborescente. Une T_{AMT} joue un rôle de coordinateur d'un ensemble d'alternatives d'exécution. Chaque alternative coordonne un ensemble de *transactions composantes*. L'accès aux données se fait seulement par les transactions composantes, qui sont considérées comme les feuilles de l'arbre. Le niveau T_{AMT} et les alternatives sont seulement des unités de contrôle. Si on fait une analogie avec les multi-transactions, les transactions composantes sont des sous-transactions introduites par les alternatives d'une T_{AMT}.

Dans la suite, nous présentons une définition intuitive du modèle AMT. Pour sa définition formelle voir [SA04].

Definition 2 *Une transaction mobile adaptable est définie par* $T_{AMT} = \langle AE_k \rangle$ *où :*

- $\langle AE_k \rangle$, $k > 0$, est une liste d'alternatives d'exécution et AE_k a priorité sur AE_{k+1}.
- $AE_k = (DE_k, PE_k)$. Une alternative d'exécution contient un plan d'exécution PE_k qui sera lancé seulement si l'environnement mobile courant offre les caractéristiques spécifiées par le descripteur d'environnement DE_k.
- $DE_k = \{Dimension = etat(s)\}$. Un descripteur d'environnement contient un ensemble de dimensions avec leur(s) état(s).
- $PE_k = \{(t_{ki}, tc_{ki}, SiteId)\}$ est un ensemble dont chaque élément introduit une transaction composante t_{ki}, sa transaction de compensation tc_{ki} (si possible) et le site $SiteId$ où l'exécution doit avoir lieu. $SiteId$ indique la base de données et l'unité (mobile ou fixe) qui l'accueille.

  Il existe une *relation de dépendance* entre les éléments de PE_k. Cette relation indique le parallélisme ($||$) ou la séquentialité ($<$) de l'exécution des transactions composantes du plan. Un plan peut donner lieu à une exécution sur plusieurs sites. Un site exécute au plus une transaction composante par alternative.

Les transactions composantes sont exécutées par les SGBD sous-jacents, sous leur responsabilité. Une transaction composante peut avoir une transaction de compensation dont le rôle est de défaire, d'un point de vue sémantique, le travail fait par la transaction qu'elle compense. Les transactions de compensation ne sont pas obligatoires : une transaction composante peut ne pas être compensable mais aussi dans certains cas, une compensation n'est pas nécessaire (par exemple, dans le cas des transactions de lecture). Cependant, comme nous le verrons par la suite, leur présence augmente la souplesse de gestion des transactions et l'autonomie des sites.

5. Ces variations ne sont pas considérées comme des pannes

5.3.2 Propriétés des transactions T_{AMT}

Pour présenter les propriétés du modèle AMT il est nécessaire de distinguer les trois niveaux : les transactions AMT, ses alternatives d'exécution et ses transactions composantes. Chaque niveau, effectue les opérations *begin, abort, commit.*

Rappelons que les transactions composantes sont exécutées par les SGBD qui doivent assurer les propriétés ACID. Dans la suite nous nous concentrons sur les propriétés des alternatives et des T_{AMT} en considérant principalement les aspects atomicité, isolation et correction des transactions. L'aspect durabilité repose sur les SGBD sous-jacents après la validation des transactions composantes. Le choix des propriétés tient compte des caractéristiques des contextes mobiles et cherche à augmenter l'autonomie des unités mobiles. Les propriétés peuvent être résumées de la manière suivante : les T_{AMT} sont semi-atomiques et reposent sur l'atomicité sémantique des alternatives. Les alternatives incluant des transactions composantes compensables relâchent l'isolation. La sérialisabilité locale à un SGBD est sa responsabilité alors que le niveau global est à la charge du support des T_{AMT}.

Atomicité sémantique des alternatives

Comme dit précédemment, les transactions composantes d'une alternative peuvent être exécutées sur une unité mobile ou sur une unité fixe. Nous cherchons à favoriser le travail autonome des unités mobiles, même en mode déconnecté, tout en limitant l'impact sur les autres participants du système. Tenant compte des limitations des ressources et des déconnexions courantes, le modèle AMT relâche l'atomicité des alternatives d'exécution. L'objectif est de limiter le blocage des ressources utilisées par les transactions composantes lorsque c'est possible. Ainsi, on distingue les transactions composantes compensables des transactions composantes non compensables. Les compensables sont autorisées à valider localement sans attendre la validation de l'alternative. Cette validation optimiste permet de libérer les ressources au plus tôt. Cependant, si l'alternative annule, les transactions de compensation correspondantes doivent être exécutées.[6]

Les transactions composantes non compensables ne peuvent pas valider de manière optimiste. Lorsqu'une telle transaction termine, les ressources utilisées seront retenues jusqu'à ce qu'une décision globale (de validation ou d'annulation) soit prise pour l'alternative. Dans ce cas, une annulation de l'alternative va simplement impliquer un *abort* local de la transaction composante.

L'utilisation de transactions de compensation comme un moyen de récupération sémantique conduit à une atomicité sémantique [GM83] au niveau des alternatives. Dans le cas où une alternative contiendrait exclusivement des transactions composantes non compensables (aucune transaction de compensation n'est nécessaire), on obtient la *failure atomicity.*

Ainsi, une AE_k contenant un ensemble de t_{ki}, est atomique sémantiquement si :

1. soit toutes les t_{ki} définies dans AE_k valident ;
2. soit toutes les t_{ki} définies dans AE_k sont soit compensées soit annulées.

Semi-atomicité des transactions T_{AMT}

La semi-atomicité est assurée si (1) la validation de la T_{AMT} implique la validation d'une seule AE_k et l'annulation ou la compensation des sous-transactions d'autres AE_l,

6. Les transactions de compensation des transactions composantes exécutées séquentiellement sont exécutées dans l'ordre inverse de validation. Celles correspondant aux transactions composantes exécutées en parallèle sont exécutées de façon concurrente.

si elles ont été démarrées, et (2) l'annulation de la T_{AMT} implique l'annulation ou la compensation de toutes les sous-transactions de l'AE_k active. Autrement dit :

1. soit toutes les t_{ki} définies dans l'AE$_k$ valident et toutes les t_{li} d'autres AE$_l$ démarrées de la même T_{AMT} sont annulées ou compensées ;
2. soit aucun résultat partiel des transactions composantes t_{ki} de la T_{AMT} ne reste permanent dans les SGBD sous-jacents.

Ordonnancement global

Les transactions composantes d'une T_{AMT} s'exécutent sur des SGBD différents. Malgré le fait que les sources des différents SGBD soient disjointes, les dépendances de précédence entre les transactions composantes d'une même alternative obligent à préserver un ordre d'exécution aussi bien à l'intérieur de l'AE_k qu'entre les AE_k concurrentes – appartenant à des T_{AMT} différentes.

Le critère utilisé pour contrôler la correction d'exécution des alternatives concurrentes est la *sérialisabilité globale*. La sérialisabilité globale des alternatives veut que les transactions composantes soient sérialisées sur chaque site de la même manière. Rappelons que sur chaque site, il existe une seule transaction composante par alternative et que des transactions locales au site (hors les T_{AMT}) peuvent y être exécutées.

La sérialisabilité des T_{AMT} est fournie à travers celle des alternatives d'exécution. Par ailleurs, nous considérons qu'il n'y a pas de contraintes d'intégrité globales (inter bases). La cohérence sémantique est préservée.

5.4 Intergiciel TransMobi

L'intergiciel TransMobi met en œuvre le modèle AMT. Cette proposition facilite l'accès transactionnel aux bases de données en offrant un certain niveau de transparence par rapport aux aspects mobiles des clients ou des sources.

TransMobi se situe entre l'application et les SGBD utilisés comme illustré par la Figure 5.1. Aucune adaptation, ni fonction particulière n'est demandées aux SGBD sous-jacents. Les interfaces standardisées (Tx de X/Open [Lim95]) sont utilisées pour interagir avec ces SGBD qui peuvent être hétérogènes et dont l'autonomie est respectée. De plus les SGBD ne doivent pas exporter des informations propres à leur gestion. Rappelons que cet intergiciel n'a pas pour objectif l'intégration de bases de données, mais le support transactionnel dans le cadre d'un couplage faible.

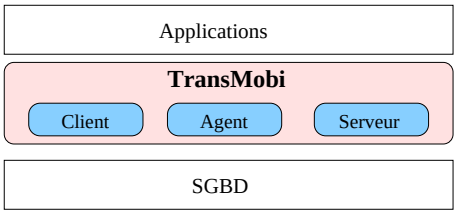

FIG. 5.1 – *Vu globale du système utilisant TransMobi*

L'existence de plusieurs SGBD légers commerciaux – par exemple FastObjects [bP], PointBase [Dat], Oracle9i Lite [Cor] – permet de considérer un environnement avec des

unités mobiles et fixes capables de gérer des données localement.

5.4.1 Architecture

TransMobi a une architecture de type *client-agent-serveur* comme le montre la Figure 5.2. Un *TMClient* interagit directement avec l'application et sera installé sur chaque site participant. Les *TMServeur* interagissent avec les SGBD sous-jacents pour invoquer l'exécution des transactions composantes ou de compensation. Ils assurent également certaines tâches de contrôle telles que la coordination de la validation globale et la gestion d'un *graphe de sérialisabilité globale*.

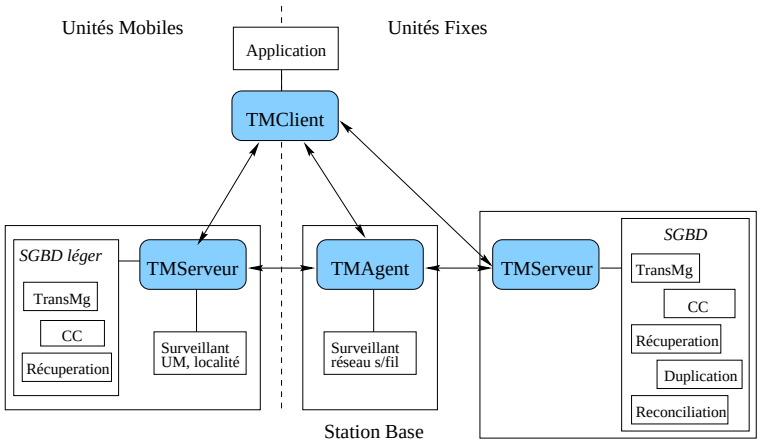

FIG. 5.2 – *Architecture client-agent-serveur de TransMobi*

Les *TMAgents* sont installés sur des UF capables de communiquer avec des UM. D'une façon générale, les *TMAgents* agissent comme des proxy pour des UM afin de réduire leur consommation de ressources et les communications sans fils. Ainsi, les *TMAgents* stockent des messages, déclenchent ou arrêtent certaines fonctionnalités en faveur des clients mobiles, sauvegardent temporairement des données, etc. Leur présence permet de diviser l'interaction entre les clients et les serveurs en deux parties : client/agent et agent/serveur. Il est possible d'utiliser des protocoles différents pour chaque partie et l'interaction peut être faite de façon indépendante. Cette architecture est très favorable aux clients mobiles à cause des déconnexions fréquentes et des ressources limitées.

5.4.2 Perception de l'environnement

Au lancement d'une T_{AMT}, TransMobi prend en charge la vérification de l'état courant de l'environnement pour effectuer le choix de l'alternative d'exécution appropriée. L'état des dimensions de l'environnement est récupéré via des événements grâce à un *service d'événements*. TransMobi s'abonne aux événements liés aux dimensions apparaissant dans les descripteurs d'événements utilisés. La notification des événements peut être en mode *pull* ou *push* selon si c'est TransMobi qui fait la demande au service d'événements ou non. Le service d'événements notifie les événements à partir de l'information fournie par des surveillants de l'environnement de divers types et niveaux (par exemple, système d'exploitation ou couche de communication). La mise en œuvre de ces détecteurs dépend de l'infrastructure. Certains événements sont faciles à générer, d'autres le sont moins. Par exemple, les systèmes d'exploitation incluent généralement des fonctions *prof* et *gprof* qui profilent

l'utilisation du CPU. Celles-ci permettent de générer l'événement concernant la capacité de calcul. Diverses approches peuvent être utilisées pour obtenir la localité d'une UM (par exemple, GPS) et l'exploiter dans l'adaptation. En effet, les positions géographiques visitées fréquemment par l'utilisateur pourraient être traduites – de longitude et latitude – à maison, travail, mairie, centre ville, etc.

Pour terminer, soulignons que l'approche événementielle facilite le déclenchement immédiat et différé des T_{AMT} et nous permet d'étendre facilement les dimensions observées.

5.4.3 Protocoles pour garantir les propriétés des T_{AMT}

Les caractéristiques des environnements mobiles ont motivé des travaux sur les protocoles qui assurent les propriétés des transactions. Un des buts est de réduire le nombre et la taille des messages, surtout ceux circulant sur les réseaux sans fil. TransMobi doit garantir les propriétés d'atomicité sémantique des T_{AMT}, leur semi-atomicité et la sérialisabilité globale. La semi-atomicité est assurée simplement par le choix d'une seule alternative. Dans la suite nous présentons les protocoles proposés pour assurer les autres deux propriétés.

Le protocole de validation CO2PC

Le protocole pour assurer l'atomicité sémantique – comme définie dans la section 5.3.2 – doit satisfaire les contraintes des environnements mobiles, mais aussi permettre la validation anticipée de transactions composantes. L'état de l'art ne fournissant pas un tel protocole, nous proposons CO2PC, une Combinaison d'une approche Optimiste et du protocole 2PC. Ce protocole cherche à augmenter l'autonomie des participants et la flexibilité dans leur travail. Ainsi CO2PC autorise les transactions composantes compensables à valider unilatéralement de manière anticipée alors que la validation des transactions composantes non-compensables est différée pour se faire de manière coordonnée en accord avec la décision globale à l'alternative qui les a lancées. CO2PC fonctionne comme suit : chaque participant envoie son vote commit/abort au coordinateur qui décide un commit global, en cas d'unanimité, ou un abort global sinon. Le moment de l'envoi du vote est différent selon si le participant exécute une transaction compensable et fait une validation optimiste ou non (voir figure 5.3). Une validation avec CO2PC permet des participants des deux types. Considérons les deux cas :

Validation optimiste Un participant faisant une validation optimiste (appelé *participant-opt*), demande l'exécution de sa transaction composante au SGBD sous-jacent. Une fois l'exécution terminée, la transaction est validée ou annulée d'une manière unilatérale. Le *participant-opt* envoie ensuite le *vote* correspondant au coordinateur CO2PC. Si la décision globale est de valider, les *participants-opt* ont terminé. Si la décision est une annulation, ils doivent démarrer la transaction de compensation correspondante.

Validation non-optimiste Un participant exécutant une transaction composante non compensable (appelé *participant-non-opt*) ne peut pas valider unilatéralement. Chaque *participant-non-opt* exécute le protocole 2PC localement. Le module TransMobi local coordonne un 2PC avec comme unique participant le SGBD sous-jacent. Conforme à 2PC, le SGBD lui envoie un vote. Celui-ci sera propagé au coordinateur du CO2PC. Si le vote est *annulation*, le participant annule t_{ki} de façon unilatérale, sinon il entre dans l'état de *préparation*. Dès que la décision globale du coordinateur CO2PC arrive au participant, elle est transmise au SGBD. Cette décision sera considérée comme la décision pour le 2PC en cours localement. Une transaction non-compensable doit être exécutée sur un SGBD

fournissant l'interface 2PC. La Figure 5.4 illustre le comportement de CO2PC lors d'une annulation globale.

CO2PC permet de libérer les ressources des transactions compensables au plus tôt. Ceci n'étant pas possible pour les transactions non compensables, nous adoptons dans ce cas 2PC qui offre l'interface *préparation*. En effet, une transaction en état *préparation* ne peut être annulée ni validée unilatéralement par le SGBD sous-jacent. Seule une opération de validation/annulation émise par un gestionnaire global peut faire sortir les transactions de cet état. L'utilisation de 2PC pour la coordination globale n'est cependant pas pertinente à cause du nombre de messages (quatre par participant) et du fait qu'il bloque la transaction entière jusqu'à la décision globale. Ceci est un inconvénient important dans les environnements mobiles à cause des déconnexions

Afin de limiter les situations de blocage, les participants comme le coordinateur CO2PC ont des temporisateurs. Malgré cela et suite à l'utilisation de 2PC (pour une partie de la validation), un participant non-opt peut être bloqué s'il a voté *commit* mais ne reçoit pas la décision globale.

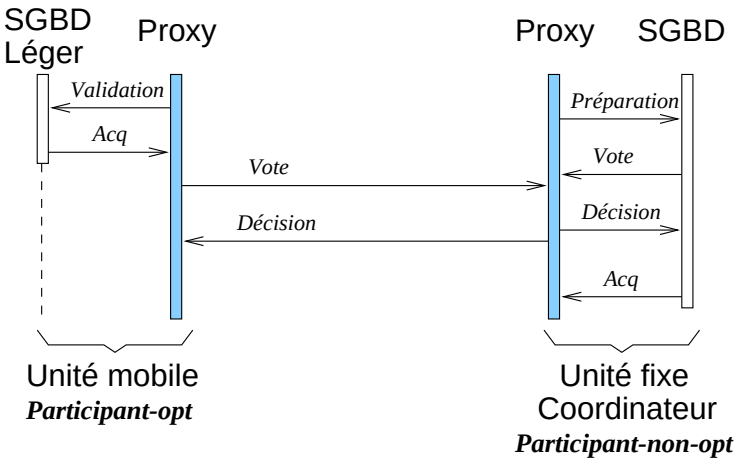

FIG. 5.3 – *Protocole CO2PC*

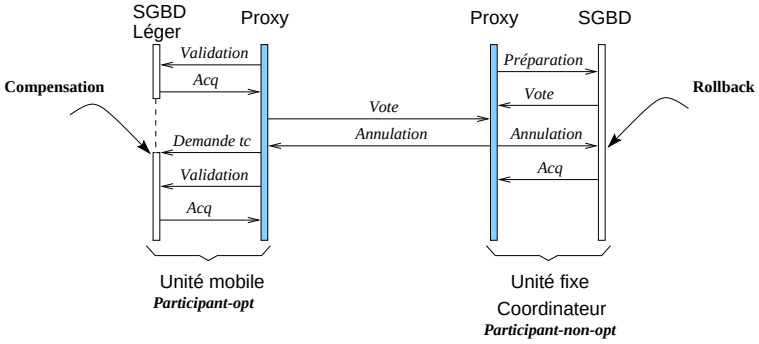

FIG. 5.4 – *Protocole CO2PC - cas d'annulation*

Reprise après pannes

Afin de fournir une récupération après pannes, CO2PC enregistre le progrès fait dans ses différentes étapes, aussi bien dans un journal chez le coordinateur que chez les participants.

Le Tableau 5.1 illustre l'information à journaliser par les participants et le coordinateur. Afin de préserver l'autonomie des SGBD sous-jacents, aucune supposition n'est faite sur les politiques de journalisation des SGBD sous-jacents et aucun partage des journaux transactionnels n'est demandé. Comme dit précédemment, l'annulation d'une alternative peut impliquer l'exécution de transactions de compensation. Une fois les transactions de compensation démarrées, elles doivent être validées. Cette caractéristique, appelé *persistance de compensation* [GMS87], est assurée par la re-soumission des transactions de compensation jusqu'à leur validation. Si la persistance de compensation est garantie, il n'y a pas besoin d'utiliser un protocole de validation pour assurer l'atomicité de l'ensemble des composantes d'une alternative.

Site	**Opération à journaliser**
Participant-opt	*Demande de Validation*
	Acquittement
Participant-non-opt	*Début2PC*
	Préparation
Tous les participants	*Vote*
	Décision
Coordinateur	*DébutCO2PC*
	Chaque *Vote*
	Décision

TAB. 5.1 – *Journalisation pendant le protocole CO2PC.*

Sérialisabilité globale

Les algorithmes de contrôle de concurrence synchronisent de transactions concurrentes, en ordonnant les opérations en conflit de telle manière qu'un ordre de sérialisabilité est maintenu. Au niveau global (AE_k), identifier les conflits n'est pas une tâche facile, en particulier lorsque les SGBD sous-jacents ne partagent pas l'information sur la gestion locale des transactions. En effet, le grand souci dans ce contexte est de gérer les *conflits indirects* sans compromettre l'autonomie des SGBD sous-jacents.

Pour fournir la sérialisabilité globale, deux conditions doivent être considérées: les SGBD sous-jacents doivent produire des ordonnancements sérialisables et l'ordre relatif des sous-transactions doit être le même que l'ordre global.

Dans notre contexte, la première condition est acquise grâce aux SGBD sous-jacents. Pour garantir la deuxième, nous adaptons une méthode optimiste basée sur l'utilisation de tickets (OTM).

L'idée principale dans OTM [GRS91, GRS93] est de convertir les conflits indirects en conflits directs. Chaque SGBD sous-jacent aura une donnée spéciale, appelée *ticket*. Chaque sous-transaction globale lit, incrémente et écrit le ticket dans le site où elle s'exécute. Au niveau global, un graphe de sérialisabilité globale (GSG) est maintenu pour refléter l'ordre induit par les tickets.

TransMobi utilise OTM au niveau des alternatives et le combine avec CO2PC. Le GSG est maintenu par un gestionnaire globale de sérialisabilité (GG) localisé sur un TMServeur.

Lorsqu'une alternative commence son processus de validation, le GG crée un nœud AE_k dans le graphe. Chaque composante T_{ki} incrémente la valeur du ticket et quand l'exécution termine, le participant envoie le *vote* ainsi que la valeur du ticket au coordinateur du CO2PC. Ce dernier envoie la valeur du ticket au GG qui insère une arête entre AE_k et d'autres AE_j récemment validées. Les arêtes reflètent l'ordre relatif des tickets [GRS93]. Si t_{ki} et t_{ji} s'exécutent sur un même site i et t_{ki} obtient un ticket plus grand que celui de t_{ji}, alors une arête de AE_j vers AE_k est insérée. Aussitôt qu'un cycle est généré, le GG averti le coordinateur qui demande l'annulation de l'alternative correspondante. Si aucun cycle n'est généré après l'insertion de toutes les arêtes, le GG donne le feu vert au coordinateur pour la validation de l'alternative.

Le coût de communication pour gérer le GSG est minimal (cf. Figure 5.5) :

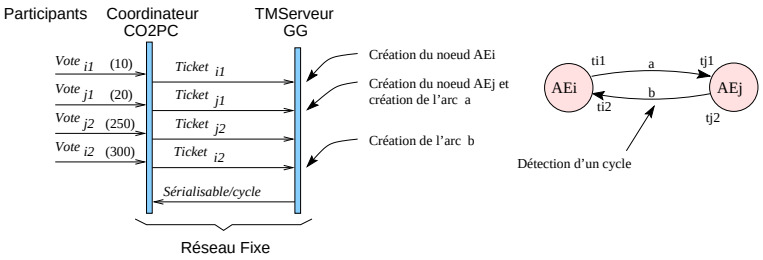

FIG. 5.5 – *Messages pour la gestion du GSG*

- Un premier message permet de créer le nœud de l'AE_k. Ce message est envoyé par le coordinateur de CO2PC au GG, lorsqu'il reçoit le premier message de *vote* d'un des participants au processus de validation.

- Chaque participant doit transmettre la valeur du ticket mis à jour. Cette valeur est ajoutée au message de *vote* envoyé au coordinateur de CO2PC. Le coordinateur envoie cette valeur au GG.

- Un dernier message de décision (validation/annulation) que le GG envoie au coordinateur de CO2PC est nécessaire.

Le fait que les participants attachent l'information nécessaire pour la gestion du GSG aux messages du CO2PC permet d'éviter de générer de nouveaux messages sur le réseau sans fil. En effet, les messages nécessaires dans le réseau fixe – entre le coordinateur de CO2PC et le GG – sont au nombre de deux pour la création de nœud et décision et un par participant (envoie de la valeur du ticket).

Tolérance aux déconnexions

Dans un système s'exécutant en environnement mobile il est important de tolérer les déconnexions. En ce qui concerne les transactions, les protocoles doivent être prévus en conséquence. L'utilisation de temporisateurs dans le protocole CO2PC permet de restreindre certaines situations de blocage. Afin de minimiser les difficultés, les TMAgents jouent le rôle de proxy pour des unités mobiles déconnectées. Notons que les déconnexions peuvent être volontaires et planifiées par souci d'économie de ressources.

Les déconnexions n'entraînent pas de dysfonctionnement du protocole de sérialisation basé sur OTM car l'ordre de sérialisabilité est défini par l'accès aux tickets. L'incrément du

ticket est une opération locale. En contrepartie, une communication avec le coordinateur de CO2PC est nécessaire. Les ordonnancements seront corrects malgré les déconnexions mais le taux de validation des transactions peut se voir affecté.

5.5 Conclusions et perspectives

Nous avons proposé un service destiné au support de transactions s'exécutant sur des sources hétérogènes, autonomes et distribuées sur des unités mobiles ou fixes. Une des particularités majeures de notre proposition est la prise en compte de l'environnement au niveau transactionnel. Nous proposons le modèle de transactions AMT qui pour une transaction globale, permet la définition de plusieurs alternatives d'exécution associées chacune à un descripteur d'état d'environnement. Le choix de l'alternative est fait au lancement de la T_{AMT} selon l'état courant de l'environnement. Le modèle AMT et ses propriétés ont été formalisé en ACTA. Ils peuvent ainsi être comparés avec d'autres modèles.

Afin d'assurer une mise en œuvre appropriée aux environnements mobiles, nous avons proposé le protocole de validation CO2PC. Ce protocole original, favorise l'autonomie des unités participantes et permet aux transactions composantes une validation anticipée ou synchronisée avec la transaction globale.

Nous avons proposé un intergiciel, nommé TransMobi, offrant le support des transactions T_{AMT}. Un prototype a été développé par P. Serrano Alvarado et N. Ibrahim (élève ingénieur ENSIMAG). L'expérimentation a été faite sur un WLAN IEEE 802.11b avec des unités mobiles de type portable Compaq Armada 1700 et PDA, IPAQ H3850 et le SGBD PointBase [Dat]. Ce dernier est écrit en Java, offre une empreinte réduite[7], l'interface JDBC, le protocole 2PC et les quatre niveaux d'isolation de l'ANSI SQL. Le prototype a permis de tester le support des divers modèles d'exécution du modèle AMT et de confirmer la faisabilité de l'approche. Les variations de l'environnement ont été simulées. Des expérimentations plus amples restent nécessaires pour compléter la validation pratique de TransMobi.

Sur le plan théorique, nous avons effectué une étude analytique du modèle AMT [SARAL03]. Nous modélisons les états des environnements mobiles et de ses variations et estimons la probabilité de déclenchement de chaque alternative d'une T_{AMT} et son coût d'exécution étant donné un état de l'environnement. Ceci permet d'analyser divers aspects du modèle AMT dont la probabilité de déclenchement d'une T_{AMT} par rapport aux dimensions de l'environnement (par exemple, bande passante faible) et l'impact des dimensions sur le coût de l'exécution. L'analyse montre comment un choix approprié d'alternatives d'exécution permet d'augmenter le taux de validation des transactions mobiles et de maîtriser leur coût d'exécution en accord avec les besoins de qualité des applications.

Notre modélisation peut servir de base à des outils d'aide à la conception des transactions. La conception des T_{AMT} pouvant être complexe, la mise en place de tels outils parait importante. L'administrateur du système qui connaîtra les caractéristiques de(s) l'environnement(s) pourra ainsi assister les développeurs des applications dans la définition des alternatives d'exécution.

Certaines perspectives de recherche concernant les protocoles de validation et de sérialisabilité proposés se dégagent. Nous pensons particulièrement à l'introduction d'un point d'adap-

7. Entre 45 KB et 1 MB.

tabilité dans le choix des protocoles utilisés. Dans notre proposition actuelle, nous avons retenu les protocoles CO2PC et une adaptation d'OTM pour favoriser l'autonomie des unités participantes tout en économisant leurs ressources. Néanmoins, une étude plus approfondie permettrait d'identifier les cas de figure où ces protocoles sont les plus pertinents et éventuellement ceux où d'autres choix seraient plus appropriés.

D'une manière plus globale, nous considérons que le domaine d'utilisation de notre proposition peut être élargi. Jusqu'à présent, nous avons donné priorité aux caractéristiques des environnements mobiles pour nos choix. Néanmoins il s'agit d'une proposition générale permettant le support de transactions sur plusieurs SGBD faiblement couplés dans des environnements aux caractéristiques variables. La prise en compte de l'état de l'environnement au sein des transactions introduit un potentiel d'adaptation qui peut être exploité dans divers contextes (par exemple réseaux Ad-Hoc, systèmes P2P). Par ailleurs, l'approche événementielle pour la surveillance de l'environnement permet d'étendre la notion d'état d'environnement avec de nouvelles dimensions liées aux applications, aux profils des utilisateurs ou à des contextes différents.

Chapitre 6

Conclusions et perspectives de recherche

6.1 Conclusions

Le rôle et la place des données et plus généralement de l'information dans la société ont fortement évolué ces dernières années. Conjointement, les systèmes pour gérer ces données ont suivi une évolution très importante. Les travaux présentés dans ce manuscrit en sont un reflet. Nous avons d'abord travaillé à la construction de SGBD puissants, avec une vision qui place le SGBD au cœur des systèmes d'information. Puis nous nous sommes tournés vers des recherches qui facilitent la construction de solutions informatiques comportant une gestion des données répartie en plusieurs points des systèmes. Le panorama de la gestion de données est aujourd'hui très varié et hétérogène, tant par la taille des systèmes que par les environnements où ils opèrent. On trouve à la fois des SGBD intégrant un grand nombre de fonctions ou au contraire des systèmes restreints et très légers, sans oublier les nombreuses variantes d'intergiciels de gestion de données.

SGBD actifs Nos travaux sur les SGBD actifs ont abouti à des propositions solides sur les aspects événementiels. Ils restent d'actualité pour une utilisation soit au sein des SGBD, soit dans des systèmes à architecture ouverte ou à base de services. Nos recherches ont également permis de mettre en avant l'impact des choix architecturaux, des types d'événements et des modèles d'exécution de règles actives sur les performances du système. Une dernière contribution à cette thématique est un service de visualisation d'exécutions permettant de rejouer des applications utilisant des règles actives. Ce service permet une meilleure compréhension de ce type d'applications dont le comportement peut être assez complexe quand plusieurs événements-réactions interviennent. La conception de cet outil de visualisation n'est pas liée à un système actif spécifique.

Nos recherches se sont poursuivies dans l'optique de détacher du SGBD comme seule unité de déploiement pour les systèmes de gestion de données. Une motivation forte est venue de la généralisation de l'intérêt suscité par les systèmes répartis, avec en particulier le succès du WWW ainsi que de la diversification croissante des domaines d'application nécessitant une gestion de données.

Nous avons cherché à isoler les différents aspects de la gestion de données (par exemple la duplication et la persistance) de manière à offrir des solutions flexibles dédiées à chaque aspect. Le but est de proposer des services qui peuvent ensuite participer à la construction

de systèmes répondant à des besoins particuliers. Traiter les aspects séparément permet de limiter la complexité et de travailler ainsi l'aspect concerné à un niveau fin.

Nos contributions portent principalement sur la gestion de cache, la duplication de données et l'aspect transactionnel. Le contexte d'utilisation privilégié de ces services est celui des intergiciels pour la gestion de données distribuées.

Cache sémantique Notre proposition de cache sémantique est destinée à optimiser l'évaluation de requêtes dans un méta-moteur de recherche travaillant sur des sources de données hétérogènes du WWW. Cette proposition est néanmoins intégrable dans d'autres contextes. L'expérimentation a montré que l'utilisation d'un tel cache de requêtes est surtout intéressante en présence de sources de données fournissant des réponses vérifiables, de préférence complètes. Par ailleurs, les caractéristiques du langage de requêtes et le choix du descripteur des requêtes dans le cache apparaissent comme des aspects déterminants pour les performances du cache.

Services de duplication de données Une des contributions importantes de notre travail d'isolation de services non-fonctionnels pour la gestion de données porte sur l'aspect duplication. Nous proposons un canevas de services de duplication, résultant d'un travail approfondi sur les protocoles de duplication et la distinction de la notion de modèle de cohérence pour les copies d'un objet (dite cohérence locale). Nous distinguons et formalisons quatre types de modèles de cohérence locale. Notre proposition de services de duplication repose sur une décomposition structurelle et une décomposition fonctionnelle des protocoles. Ces décompositions permettent une adaptabilité des protocoles ou de parties de ceux-ci et favorisent la réutilisation, à grain fin, des composants des protocoles. Nos expérimentations ont permis de confirmer l'intérêt de l'approche mais montrent néanmoins que la mise en place d'une solution globale reste un problème difficile.

Service de transactions adaptable Notre proposition a la particularité d'intégrer une conscience de l'environnement pour adapter en conséquence le modèle d'exécution des transactions. Ceci est intéressant pour l'exécution dans des environnements répartis variables comme ceux qui incluent des unités mobiles. Une attention particulière a été donnée aux contraintes de ces environnements. Nous avons défini, formalisé et validé un modèle de transactions adaptables, nommé AMT. Notre étude analytique montre comment le modèle AMT permet aux utilisateurs de maîtriser les coûts induits par l'exécution de transactions et d'augmenter leur taux de réussite. Ce modèle est mis en œuvre par l'intergiciel TransMobi qui implante les protocoles appropriés. Le protocole de validation de transactions CO2PC en fait partie. Ce protocole original permet de libérer au plus tôt les ressources des transactions compensables participant à une transaction globale de type AMT et tolère les déconnexions des participants. Ces points sont essentiels pour les applications faisant intervenir des unités mobiles.

Nos recherches s'insèrent dans un travail plus ambitieux mené au sein de notre équipe. Des services complémentaires à ceux développés ont été proposés dans ce cadre. Parmi eux le service de persistance Perseus [GB03] et de tolérance aux pannes Alanca [Duo03]. Nous avons mené des expériences d'utilisation des services par la construction de gestionnaires de données. Deux de ces gestionnaires sont destinés au support de mondes virtuels distribués persistants. Il s'agit d'applications non transactionnelles qui ne sont pas considérées traditionnellement comme des applications « bases de données ». Nous avons réalisé ce travail dans le cadre du projet IST PING [CTD+01] et du Contrat de Recherche Externe

« Services bases de données ouverts pour applications coopératives » avec France Télécom [RND02]. Ont contribué à ce travail Ch. Collet et quatre doctorants de notre équipe — T. Nedelec, L. Garcia, S. Drapeau et Q. Duong. Les gestionnaires développés utilisent le service de duplication et celui de persistance qui inclut indexation, cache, stockage et journalisation. L'adaptabilité de chacun des aspects a permis de construire des intergiciels aux caractéristiques cherchées. En ce qui concerne la duplication, plusieurs protocoles maître-esclaves et protocoles acceptant des mises-à-jour indifférenciées sur toutes les copies ont été mis en œuvre. Des variantes impatientes et paresseuses ont été dérivées en utilisant plusieurs événements de synchronisation (événements temporels et guidés par le nombre de mises à jour). Le potentiel de réutilisation des composants fonctionnels pour la création de protocoles à partir d'autres protocoles de duplication a pu être validé.

A propos de la persistance, nous avons expérimenté différentes approches pour le gestionnaire de stockage. D'une part, une gestion faite exclusivement par Perseus sur des fichiers et d'autre part, en utilisant l'intergiciel JORM, Java Object Repository Mapping, proposé en logiciel libre dans le cadre d'ObjectWeb [Jor]. JORM cherche à standardiser la médiation entre les objets java en mémoire et leur image en support de stockage. Un des avantages de son utilisation est de pouvoir changer de support de stockage selon les besoins et, à terme, d'utiliser plusieurs supports de stockage simultanément. Il est intéressant de noter que le développement de ces solutions a impliqué la résolution de problèmes déjà apparus lors des travaux sur les SGBD et LPBD à objets, par exemple, la gestion de références entre objets et l'adressage. Néanmoins, les solutions qui étaient les plus favorables pour les SGBD à objets ne s'avèrent pas forcément les plus performantes pour les intergiciels développés en Java.

L'expérience de ces travaux montre que l'approche par services est viable et prometteuse mais que divers points restent néanmoins à approfondir. Parmi eux la composition des services qui gagnerait à être facilitée (dans la ligne de [BVSC03b]), la vérification des propriétés de cette composition et l'optimisation des systèmes construits.

6.2 Perspectives

Nos perspectives de recherche se situent donc dans le prolongement de nos travaux sur les services et les intergiciels. Elles sont motivées par le besoin confirmé de gestion de données dans des contextes très variés. Nous nous intéressons plus particulièrement à la construction d'intergiciels de gestion de données pour des environnements variables et à grande échelle (en terme de nombre de sites participants et de volume de données). Nous donnons priorité aux aspects transactionnels, duplication, gestion de cache et interrogation. Dans la suite nous introduisons ces aspects dans le contexte qui les motive en premier lieu. Néanmoins, ces aspects sont significatifs aussi bien pour les environnements variables que pour les systèmes à grande échelle. Nos propositions des services doivent, d'une certaine manière, être orthogonales au contexte d'utilisation. Elles se doivent d'être générales et adaptables pour être utilisées dans la construction d'intergiciels avec des caractéristiques diverses (voir figures 6.1, 6.2 et 6.3). Par exemple nos propositions pour la gestion des caches doivent permettre la création de services de cache pour des systèmes de type grille, pair-à-pair ou dans des environnements mobiles.

Gestion de données dans des environnements variables Nous appelons environnements variables ceux dont le fonctionnement normal admet des variations significatives

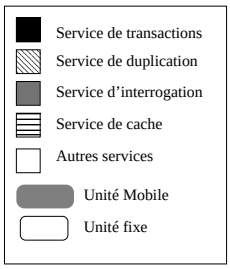

FIG. 6.1 – *Légende*

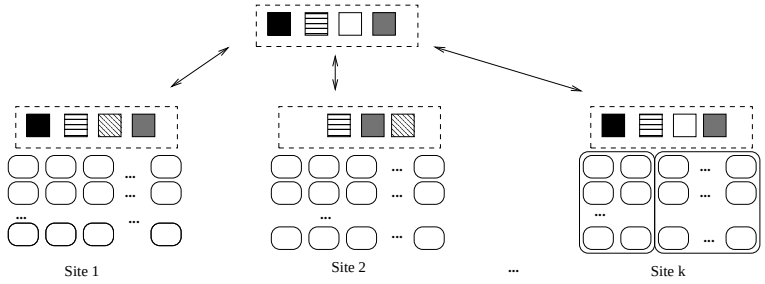

FIG. 6.2 – *Utilisation des services pour des grilles*

des ressources disponibles. Nous considérons principalement des ressources telles que les unités participant au système, leur capacité et leur connectivité. Ces variations ont un impact sur la nature de la gestion de données qui peut être offerte et sur la manière de la mettre en œuvre. Les solutions éprouvées pour des contextes répartis stables ne sont pas toujours adaptées.

Nous avons évoqué dans ce document des environnements variables faisant intervenir des unités mobiles. Nos travaux futurs concernent des services de gestion de données pour ces types d'environnements mais aussi, plus largement, des systèmes dits « pair à pair », P2P. Des tels environnements sont dynamiques mais très souvent amènent des restrictions notables. Une caractéristique importante, qui à notre avis sera à privilégier dans ces travaux, est la prise en compte des caractéristiques de l'environnement et une certaine capacité d'adaptation dynamique du service, en vue d'offrir la qualité la plus appropriée.

Environnements mobiles : L'augmentation importante du nombre d'unités mobiles et de la qualité des infrastructures pour connexions sans fil, motive le développement d'applications pour de tels contextes. Ces applications peuvent concerner des unités mobiles et fixes ou des communautés de mobiles composées de manière spontanée. Dans ce dernier cas, un réseaux ad-hoc permet les échanges entre un nombre, a priori restreint d'unités mobiles aux caractéristiques hétérogènes. La coopération entre les participants se met en place dynamiquement pour partager des données/ressources ou exécuter des tâches. Parmi les exemples d'applications nous pouvons citer des services d'information régionaux spécialisés, tels ceux pour des zones sinistrées par un désastre, des applications de trafic routier et de guidage ou le commerce électronique dans des contextes ouverts comme les foires.

Les intergiciels déployés dans de tels environnements nécessitent divers aspects, dont l'interrogation, la duplication de données et les caches. Nous abordons ici l'aspect transactionnel.

- `Aspect transactionnel` : nos travaux sur le support transactionnel offrent des transactions flexibles comportant un certain niveau d'adaptabilité pour des exécutions réparties sur des unités mobiles et fixes. Néanmoins deux points doivent être étudiés pour offrir une solution plus souple. Le premier est l'adaptation dynamique du modèle d'exécution des transactions. En présence d'un environnement présentant des variations significatives fréquentes, un changement du modèle d'exécution permettrait d'augmenter le taux de réussite des transactions. Pour offrir une telle adaptation, il est nécessaire d'offrir une stratégie de réutilisation des résultats déjà obtenus par la transaction, tout en garantissant des propriétés claires.

  Un deuxième point qui mérite un travail plus étendu est le choix des protocoles à mettre en œuvre dans le support transactionnel [BLRr04b, BLRr04a]. Il s'agit

d'étudier, selon les caractéristiques des transactions et de l'environnement d'exécution, comment changer ou adapter les protocoles — voire alléger certaines propriétés des transactions. Ceci permettrait de ne pas induire de coûts inutiles et de mieux cibler les caractéristiques du service.

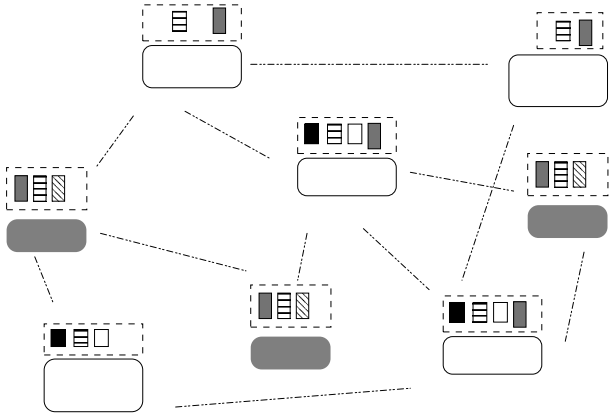

FIG. 6.3 – *Utilisation des services pour des systèmes P2P*

Les Systèmes pair à pair à grande échelle permettent la coopération lâche entre un nombre élevé et variable d'unités géographiquement distribuées et autonomes. Leur caractère variable est dû au fait que les unités peuvent se connecter et se déconnecter du système librement. Les systèmes P2P structurés [1] offrent une gestion performante de l'ensemble variable de participants reposant sur une organisation complètement distribuée [DZDS03]. Certaines applications sont proposées sur ces systèmes, parmi elles le partage en lecture d'objets (par exemple de documents) entre les participants. Cependant à l'heure actuelle, peu de fonctionnalités de haut niveau pour gérer les données sont offertes. La variabilité de l'ensemble des participants, leur faible intégration et l'échelle en nombre de participants et d'objets dans le système rendent les solutions classiques inappropriées.

- `Interrogation :` dans les environnements variables, mobiles ou pair à pair, les possibilités d'interrogation proposées sont encore restreintes. Pour les enrichir, il est nécessaire de disposer de capacités de découverte des données [Gur03] disponibles à un instant précis et de proposer des stratégies d'évaluation de requêtes adaptables [VC04]. La puissance d'expression des langages de requêtes doit être étudiée et le support des méta-données (ou catalogues) doit être repensé. Ceci est particulièrement vrai pour les systèmes P2P structurés, à cause de leur caractère complètement distribué et des autres caractéristiques déjà énoncées. Certaines de nos recherches s'orientent dans cette voie (thèse de M-P. Villamil [VRL04]).

Gestion de données dans les systèmes répartis à grande échelle Les systèmes P2P mentionnés précédemment constituent l'un des types de systèmes répartis à grande échelle d'aujourd'hui. Les *grilles* en constituent un deuxième type. Ce type de système est principalement motivé par le changement de taille des applications scientifiques — physique, bio-informatique, sciences de la terre. Ces applications, toujours plus précises, prennent une telle ampleur que le calcul et le stockage de données ne peuvent plus être pris en charge par un seul site.

1. Systèmes qui utilisent une table de hachage distribuée pour la structuration.

Une grille est un système qui permet le partage de ressources à grande échelle dans le but d'offrir des services nouveaux [FK99]. Les ressources sont fournies par des super-calculateurs ou des grappes d'ordinateurs localisés sur plusieurs sites. La grille peut être vue comme une grappe de grappes qui fonctionne avec un contrôle distribué. Les grilles de calcul et les grilles de données permettent de fédérer les capacités de calcul et de stockage de données respectivement. D'un point de vue « données », l'unité de transfert et de travail est en général le fichier. Des systèmes performants pour la gestion et le transfert de fichiers sur grille sont proposés aujourd'hui [LLC+03]. Les SGBD sont très peu utilisés dans ce contexte. Les raisons, techniques et parfois culturelles, sont liées à la rigueur imposée par leur utilisation, leur complexité ou dans certains cas par les limitations de capacité en présence de volumes de données très importants.

Nous pensons qu'il est intéressant d'explorer comment l'approche par services peut contribuer à la construction d'intergiciels pour les grilles. Nous pensons particulièrement à examiner l'utilisation des services de duplication de données et de transactions, à gérer des fonctions d'interrogation et un service de cache général. [2]

- `Service de caches` : la construction d'intergiciels de gestion de données distribuées dans des contextes variables ou non, à grande échelle ou non, fait apparaître le besoin de caches de données. Cependant, les besoins en terme de structuration et de gestion des caches changent selon les applications et le contexte de déploiement. Une solution générale permettant la création de services de caches parait intéressante. Pour une utilisation large, la gestion des caches devra être adaptable et permettre un fonctionnement coopératif ou hiérarchique.

D'un point de vue plus général, nous identifions un axe de recherche connexe très prospectif. Il s'agit de la conception d'un système de gestion de données hybride entre les systèmes de gestion de fichiers et les SGBD. L'objectif est d'offrir une solution plus légère que les SGBD mais offrant des fonctions de plus haut niveau que les systèmes de gestion de fichiers. Étant donné qu'une partie importante des applications actuelles sur grille travaillent directement sur des fichiers, un tel système vise à faciliter la gestion de leur données. Nous cherchons à voir comment les services cités (ou éventuellement de nouveaux) peuvent enrichir les systèmes de gestion de fichiers dans ce but. Ces travaux peuvent également être intéressants pour des systèmes à grande échelle avec architecture pair-à-pair. Par ailleurs, la convergence entre ce type de systèmes et les grilles est envisageable à long terme.

Pour conclure, revenons à la vision déjà évoquée: aujourd'hui, les données sont omniprésentes dans notre société et les systèmes pour les utiliser et les gérer tendent à le devenir également. Le choix des systèmes de gestion de données nécessaires devient très large et varié. Il inclut à la fois des SGBD à empreinte réduite et fonctions restreintes pour fonctionner sur des cartes à puces et des intergiciels pour gestion de données sur des grilles de dizaines de milliers de processeurs à travers plusieurs pays. Beaucoup de résultats sont déjà disponibles mais le support et la cohabitation d'un tel éventail de systèmes laisse pressentir de belles années de recherche en gestion de données.

2. Nous pensons à une solution plus générale que celle présentée dans ce manuscrit.

Bibliographie

[AA90] D. Agrawal and A. El Abbadi. The tree quorum protocol: an efficient approach for managing replicated data. In *Proc. VLDB Int'l. Conf.*, pages 243–254, Brisbane, Australie, Août 1990.

[AMS+] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *[FPSSU96]*, pages 307–328.

[Ask] Askonce. http://www.askonce.com.

[BBB+03] G. Bernard, J. BenOthman, L. Bouganim, G. Canals, B. Defude, J. Ferrié, S. Gancarski, R. Guerraoui, P. Molli, P. Pucheral, C. Roncancio, P. Serrano-ALvarado, and P. Valduriez. Mobilité et bases de données. *Revue Tecniques et Sciences de l'Informatique (TSI)*, 22(3 et 4), 2003.

[BC98] G. Brun-Cottan. *Cohérence de Données Répliquées Partagées par un Groupe de Processus Coopérant à Distance.* PhD thesis, Université Pierre et Marie Curie, Septembre 1998.

[BCF+97] J. Besancenot, M. Cart, J. Ferrié, R. Guerraoui, P. Pucheral, and B. Traverson. *Les systèmes transactionnels: concepts, normes et produits.* Hermès, 1997.

[Beh94] H. Behrends. Simulation-based debugging of active databases. In *4th Int. Workshop on Research Issues in Data Engineering: Active Database Systems(RIDE 94)*, Houston (TX), Etats-Unis, Septembre 1994.

[Ber94] M. Berndtsson. Reactive Object-Oriented Databases and CIM. In *Proc. 5^{th} International Conf. on Database and Expert System Applications,*, pages 769–778, Athene, Grece, Septembre 1994.

[Ber96] P. Bernstein. "Middleware: A Model for Distributed Services." . *Communications of the ACM*, 39(2), 1996.

[BG84] P.A. Bernstein and N. Goodman. An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases. In *ACM TODS*, volume 9, pages 596–615. Décembre 1984.

[BGB95] E. Benazet, H. Guehl, and M. Bouzeghoub. Vital: a visual tool for analysis of rules behavior in active databases. In *2nd Workshop on Rules in Database Systems (RIDS 95-LNCS 985)*, Athene, Grece, 1995.

[BGM90] D. Barbara and H. Garcia-Molina. The case for controlled inconsistency in replicated data. In *Proceedings of the Workshop on Management of Replicated Data*, Houston, TX, Etats-Unis, Novembre 1990.

[BL97] Michael J. A. Berry and Gordon Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Support.* John Wiley and Sons, 1997.

[BLRr04a] C. Bobineau, C. Labbe, C. Roncancio, and P. Serrano-Alva rado. Comparing transaction commit protocols for mobile environments. In *IEEE International Workshop on Mobility in Databases and Distributed Systems*, Saragoza, Espagne, Septembre 2004.

[BLRr04b] C. Bobineau, C. Labbe, C. Roncancio, and P. Serrano-Alva rado. Performances de protocoles transactionnels en environnement mobile. In *Journées Bases de Données Avancées*, Montpellier, Octobre 2004.

[BLZS02] M. Bouzeghoub, B. Farias Lóscio, Z.Kedad, and A. Soukane. Heterogeneous data source integration and evolution. In *Database and Expert Systems Applications - Conference DEXA*, Aix-en-Provence, France, Septembre 2002.

[BMG93] J. A. Blakeley, W. J. McKenna, and G. Graefe. Experiences building the open oodb query optimizer. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., Mai 26-28, 1993*, pages 287–296. ACM Press, 1993.

[BMH99] M. Berndtsson, J. Mellin, and U. Högberg. Visualization of the composite event detection process. In *Proc. of the Int. Workshop on User Interfaces to Data Intensive Systems (UIDIS)*, Edinburgh, Ecosse, Septembre 1999.

[BMST93] N. Budhijara, K. Marzullo, F.B. Schneider, and S. Toueg. The primary-backup approach. In *Distributed Systems*, chapter 8, pages 199–216. Addison-Wesley, 1993.

[BOH+92] A. Buchmann, M. T. Ozsu, M. Hornick, D. Georgakopoulos, and F. A. Manola. *Database Transaction Models for Advanced Applications*, chapter 5, A Transaction Model for Active Distributed Object Systems. Morgan Kaufmann Publisher, 1992.

[bP] FastObjects by Poet. FastObjects j2 http://www.fastobjects.com/.

[BPA00] C. Bobineau, P. Pucheral, and M. Abdallah. A Unilateral Commit Protocol for Mobile and Disconnected Computing. In *Conf. Parallel and Distributed Computing Systems (PDCS)*, Las Vegas, Etats-Unis, Août 2000.

[Bru99] C. Bruley. *Analyse des représentations graphiques de l'information - extension aux représentations tridimensionnelles.* PhD thesis, Université Joseph Fourier, Grenoble, France, 1999.

[BVSC03a] K. Belhajjame, G. Vargas-Solar, and C. Collet. Defining and coordinating open-services using workflows. In *Eleventh International Conference on Cooperative Information Systems (COOPiS03)*, Catania, Italie, Novembre 2003.

[BVSC03b] K. Belhajjame, G. Vargas-Solar, and C. Collet. Defining and coordinating open-services using workflows. In *In Proceedings of the Eleventh International Conference on Cooperative Information Systems (COOPiS03)*, Catania, Italie, Novembre 2003. Springer Verlag.

[Cat97] R. G. G. Cattell(ed). *The Object Database Standard: ODMG 2.0.* Morgan Kaufmann Publishers, 1997.

[CB98] B. Chidlovskii and U. Borghoff. Signature file methods for semantic query caching. *2nd European Conf. on Digital Libraries*, (LNCS 1513), 1998.

[CBB+04] Ch. Collet, K. Belhajjame, G. Bruno, F. Jouanot Ch. Bobineau, G. Vargas-Solar, T. Vu, G. Bernot, D. Laurent, F. Tahi, B. Finance, Z. Kedad, and X. Xue. Towards a mediation system framework for transparent access to largely distributed sources. In *Proceedings of International Conference on Semantics of a Networked World*, June 2004.

[CC96] C. Collet and T. Coupaye. Primitive and composite events in naos. In *12ièmes Journées Bases de Données Avancées (BDA'96)*, Cassis, France, Septembre 1996.

[CCFR97] C. Collet, T. Coupaye, L. Fayolle, and C. Roncancio. Naos prototype - version 2.2. In *Exhibit program, 13th International Conference on Data Engineering*, Birmingham - Royaume-Uni, Avril 1997.

[CCR96a] C. Collet, T. Coupaye, and C. Roncancio. NAOS 2.1: dealing with composite events. In *Demonstration and poster, EDBT96 Conference*, Avignon, France, Mars 1996.

[CCR96b] C. Collet, T. Coupaye, and C. Roncancio. NAOS, native active object oriented database system. In *Journée O_2 dans le cadre du Congrès INFORSID'96*, Bordeaux, France, Juin 1996.

[CCS94] C. Collet, T. Coupaye, and T. Svensen. NAOS: Efficient and Modular Reactive Capabilities in an Object-Oriented Database System. In *Proc. VLDB Int'l. Conf.*, pages 132–143, Santiago, Chile, Septembre 1994.

[CDF+94] M. J. Carey, D. J. DeWitt, M. J. Franklin, N. E. Hall, M. L. McAuliffe, J. F. Naughton, D. T. Schuh, M. H. Solomon, C. K. Tan, O. G. Tsatalos, S. J. White, and M. J. Zwilling. Shoring up persistent applications. pages 383–394, 1994.

[CDN93] M.J. Carey, D.J. DeWitt, and J.F. Naughton. The OO7 Benchmark. In *Proc. ACM-SIGMOD Int'l Conf. on Management of Data*, pages 12–21, Washington, DC, Mai 1993.

[CDRS86] M. J. Carey, D. J. DeWitt, J. E. Richardson, and E. J. Shekita. Object and file management in the exodus extensible database system. In W. W. Chu, G. Gardarin, S. Ohsuga, and Y. Kambayashi, editors, *Proc. VLDB Int'l. Conf.*, pages 91–100. Morgan Kaufmann, Août 1986.

[CE99] K. Czarnecki and U. Eisenecker. Aspect-oriented decomposition and composition. In *Generative Programming: Methods, Techniques, and Applications.* Addison-Wesley, 1999.

[Chi00] Y. Chiaramella. Information retrieval and structured documents. In *Information Retrieval: Third European Summer-School, ESSIR, LNCS 1980/2001*, 2000.

[Chr93] P. K. Chrysanthis. Transaction Processing in a Mobile Computing Environment. In *IEEE Workshop on Advances in Parallel and Distributed Systems (APADS)*, Princeton, Etats-Unis, Octobre 1993.

[CHR96] C. Collet, P. Habraken, and C. Roncancio. Les règles dans les SGBD. *Ingénierie des Systèmes d'Information (ISI)*, 4(3), 1996.

[CLB+01] T. Coupaye, R. Lenglet, M. Beauvois, E. Bruneton, and P. Déchamboux. Composants et Composition dans l'Architecture de Systèmes Répartis. In *Journées Composants: Flexibilité du système au langage, ASF (ACM SIGOPS France)*, Besancon, France, Octobre 2001.

[CM93] S. Chakravarthy and D. Mishra. Snoop: An expressive event specification language for active databases. Technical report, Université de Florida, Gainsville, Etats-Unis, Mars 1993.

[Coh99] P. Cohen. Projection de données sur le Web . Projet d'ingénieur ecole polytechnique, Laboratoire LSR – IMAG, Juin 1999.

[Col00] C. Collet. The NODS project: Networked Open Database Services. In *Proc. ECOOP*, 2000.

[Col02] C. Collet. Architectures ouvertes pour systèmes d'information réparties : vers des services bases de données. In *Actes des 2èmes Assises nationales du GdR I3*, Décembre 2002.

[Cor] Oracle Corporation. Oracle9i Lite: The Internet Platform For Mobile Computing http://otn.oracle.com/products/lite/.

[Cou96] T. Coupaye. *Un modèle d'exécution paramétrique pour systèmes de bases de données actifs.* PhD thesis, Université Joseph Fourier, Grenoble, France, Novembre 1996.

[CRB99a] T. Coupaye, C. Roncancio, and C. Bruley. A visualization service for event-based systems. In *BDA'99,15èmes Journées Bases de Données Avancées,*, Bordeaux, France, Octobre 1999.

[CRB99b] T. Coupaye, C.L. Roncancio, and C. Bruley. Vizar, a visualisation service for event-based systems. Rapport de recherche RR 1019-I-LSR-9, LSR - IMAG, Juin 1999.

[CRBL97] T. Coupaye, C.L. Roncancio, C. Bruley, and J. Larramona. 3d visualization of rule processing in active databases. In *ACM WS on New Paradigms in Information Visualization and Manipulation, NPIVM'97*, Las Vegas, Etats-Unis, Novembre 1997.

[CRD+01] Ch. Collet, C.L. Roncancio, PQ. Duong, L. Garcia-Banuelos, and T. Nedelec. Specification of the object management support architecture (simple objects space). Technical report, Contrat France Télécom R&D cre no 00.1b.999, Juin 2001.

[CRGBN01] Ch. Collet, C.L. Roncancio, L. Garcia-Banuelos, and T. Nedelec. Lot 1 : Different approaches for persistency management: State of the art. Technical Report 100 pages, Contrat France Télécom R&D cre no 00.1b.999,, Avril 2001.

[CRS99] B. Chidlovskii, C. Roncancio, and M-L. Schneider. Semantic cache mechanism for heterogeneous web querying. *Computer Networks*, 31:1347–1360, 1999.

[CTD+01] Ch. Collet, F. Dang Tran, S. Drapeau, L. Garcia Banuelos, A. Gérodolle, T. Nedelec, L. Parmentier, and C.L. Roncancio. Object and event management: First specification. Ping-imag-07-d22-r1-1, projet ping ist-1999-11488, Juin 2001.

[CW00] S. Chaudhuri and G. Weikum. Rethinking Database System Architecture: Towards a Self-tuning RISC-style Database System. In *Proceedings of 26th International Conference on Very Large Data Bases*, Cairo, Egypt, 2000.

[Dat] PointBase Java Databases. PointBase Micro http://www.pointbase.com.

[DD99] R. Domenig and K. Dittrich. An Overview and Classification of Mediated Query Systems. *SIGMOD Record*, 28(3), 1999.

[DFJ+96] S. Dar, M.J. Franklin, B. Jonsson, D. Srivastava, and M. Tan. Semantic data caching and replacement. *Proc. VLDB Int'l. Conf.*, pages 330–341, 1996.

[DFS02] M. Dumas, M.-C. Fauvet, and P.-C. Scholl. *Chapitre "Modèles temporels" dans Bases de Données et Internet.* Hermès Science publications, 2002.

[DG98] R. A. Dirckze and L. Gruenwald. A Toggle Transaction Management Technique for Mobile Multidatabases. In *Int. Conf. on Information and Knowledge Management (CIKM)*, Bethesda, Maryland, Etats-Unis, Novembre 1998.

[DG00] R. A. Dirckze and L. Gruenwald. A Pre-Serialization Transaction Management Technique for Mobile Multidatabases. *Mobile Networks and Applications (MONET)*, 5(4), 2000.

[DG01] K. Dittrich and A. Geppert. *(eds) Component Database Systems.* Morgan Kaufmann Publishers, 2001.

[DHB97] M. H. Dunham, A. Helal, and S. Balakrishnan. A Mobile Transaction Model that Captures Both the Data and the Movement Behavior. *ACM/Baltzer Journal on special topics in mobile networks and applications*, 2(2), 1997.

[DJP93] O. Diaz, A. Jaime, and N. Paton. Dear: a debugger for active rules in an object-oriented context. In *1st Workshop on Rules in Database Systems (RIDS'93)*, Edinburgh, Ecosse, Septembre 1993.

[DR01] S. Drapeau and C. Roncancio. Concepts et Techniques de Duplication. Technical report, Laboratoire LSR IMAG, 2001.

[Dra99] S. Drapeau. Préchargement lors de l'interrogation dans les systèmes à grande échelle. Rapport de dea d'informatique : Systèmes et communications, LSR – IMAG, Juin 1999.

[Dra03] S. Drapeau. *RS2.7: un Canevas Adaptable de Services de Duplication.* PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, Juin 2003.

[DRBG00] S. Drapeau, C.L. Roncancio, and E. Benitez-Guerrero. Generating association rules for prefetching. In *IEEE WS Knowledge Discovery and Data Mining in the WWW*, Taipei, Taiwan, Avril 2000.

[DRD02] S. Drapeau, C. Roncancio, and P. Déchamboux. Overview of an adaptable replication framework. In *Poster in International Symposium on Distributed Objects and Applications (DOA'02)*. Extended abstract published as a technical report of the Université de California, Irvine, Octobre 2002.

[DRD03] S. Drapeau, C. Roncancio, and P. Déchamboux. Rs2.7: un canevas adaptable de duplication. *Revue Tecniques et Sciences de l'Informatique (TSI)*, 22(10):1297–1324, 2003.

[Duo03] P-Q. Duong. *Tolérance aux fautes adaptable pour les systèmes à composants.* PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, Décembre 2003.

[DZDS03] F. Dabek, B. Zhao, P. Druschel, and I. Stoica. Towards a common API for structured peer-to-peer overlays . *IPTPS*, 2003.

[EFI+01] ENST, France Télécom R&D, IMAG, Univ. Readings, and SICS. Object and Event Management: First Specification. Technical Report PING IST-1999-11488, Delivrable 2.2, Juin 2001.

[ELR90] A. K. Elmagarmid, Y. Leu, and M. Rusinkiewics. A Multidatabase Transaction Model for INTERBASE. In *Proc. VLDB Int'l. Conf.*, Brisbane, Australie, Août 1990.

[FCS97] M-C. Fauvet, J.-F. Canavaggio, and P.-C. Scholl. Modeling Histories in Object DBMS. In *7th International Conference on Databases and Expert Systems Applications*, Toulouse, France, Septembre 1997.

[FGS98] P. Felber, R. Guerraoui, and A. Schiper. The implementation of a corba object group service. *Theory and Practice of Object Systems*, 4(2):93–105, 1998.

[FJL+01] F. Fabret, H. Jacobsen, F. Llirbat, K. A. Ross J. Pereira, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe. In *SIGMOD Conference*, Mai 2001.

[FK99] I. Foster and C. Kesselman. *(eds) The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers, 1999.

[FLV96] O. Friesen, A. Lefebvre, and L. Vieille. Validity: Applications of a dood system. In P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, Proceedings*, volume 1057 of *LNCS*, pages 131–134. Springer, 1996.

[FPSSU96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[FRG96] A. Front, C. Roncancio, and J.-P. Giraudin. Behavioral situations and active databases systems. In ACM SIGART & SIGIR, editor, *WS on Databases: Active and Real-Time (Concepts meet Practice)*, pages 59–62, Rockville, Maryland, Etats-Unis, Novembre 1996.

[GB03] L Garcia-Banuelos. *PERSEUS: Un canevas logiciel pour la construction des gestionnaires d'objets persistants*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 2003.

[GBLR96] A. Geppert, M. Berndtsson, D. Lieuwen, and C. Roncancio. Performance evaluation of object-oriented active database management systems using the BEAST benchmark. Tec. report 96.07, Department of Computer Science, Université de Zurich, Suisse, Octobre 1996.

[GBLR98] A. Geppert, M. Berndtsson, D. Lieuwen, and C. Roncancio. Performance Evaluation of Object-Oriented Active DBMS using the BEAST Benchmark. *TAPOS Intl.Journal*, 4(8), 1998.

[GD93] S. Gatziu and K.R. Dittrich. Events in an active object-oriented database. In N.W. Paton and M.H. Williams, editors, *Proc. 1^{st} Int'l Workshop on Rules in Database Systems*, pages 23–39, Edinburgh, Ecosse, 1993.

[GGD91] S. Gatziu, A. Geppert, and K. Dittrich. Integrating Active Concepts into an Object-Oriented Database System. In *Proc. 3^{rd} Int'l Workshop on Database Programming Languages*, Nafplion, Grece, Août 1991.

[GGD95] A. Geppert, S. Gatziu, and K. Dittrich. A Designer's Benchmark for Active Database Management Systems: OO7 Meets the Beast. In *Proc. 2^{nd} Int'l Workshop on Rules in Database Systems*, pages 309–323, Athene, Grece, Septembre 1995.

[GGM95] B. Garbinato, R. Guerraoui, and K. R. Mazouni. Implementation of the GARF Replicated Object Plateform. *Distributed Systems Engineering Journal*, 2:14–27, 1995.

[GHOS96] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The Danger of Replication and a Solution. In *ACM SIGMOD International Conference on Management of Data*, Montreal, Juin 1996.

[GHPS96] J. Gray, P. Helland, P.O'Neil, and D. Shasha. The Dangers of Replication and a Solution. In *ACM SIGMOD Conference*, Montreal, Canada, Juin 1996.

[Gif79] D. Gifford. Weighted voting for replicated data. In *Proceeding of the 7th Symposium on Operating Systems Principles*, pages 150–159, Pacific Grove, California, Etats-Unis, 1979.

[GJ91] N.H. Gehani and H.V. Jagadish. ODE as an Active Database: Constraints and Triggers. In R. Camps G.M. Lohman, A. Sernadas, editor, *17th Int'l Conf. on Very Large Data Bases, Barcelona*, pages 327–336. Morgan Kaufmann, 1991.

[GJS92] N.H. Gehani, H.V. Jagadish, and O. Shmueli. Composite Event Specification in Active Databases: Model & Implementation. In *Proc. 18th Int'l Conf. on Very Large Data Bases*, pages 327–338, Barcelona, Espagne, Août 1992.

[GM83] H. Garcia-Molina. Using Semantic Knowledge for Transaction Processing in a Distributed Database. *ACM Transactions on Database Systems (TODS)*, 8(2), 1983.

[GM93] Goetz Graefe and William J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*, pages 209–218, 1993.

[GMS87] H. Garcia-Molina and K. Salem. Sagas. In *ACM SIGMOD Conference*, San Francisco, Etats-Unis, Mai 1987.

[GN95] M. Gallersdörder and M. Nicola. Improving Performance in Replicated Databases Through Relaxed Coherency. In *Proc. VLDB Int'l. Conf.*, Suisse, 1995.

[GR00] H. Grazziotin-Ribeiro. *Un service de règles actives pour fédérations de bases de données*. Thèse de doctorat, Université Joseph Fourier, Grenoble, France, juillet 2000.

[GRS91] D. Georgakopoulos, M. Rusinkiewicz, and A.P. Sheth. On Serializability of Multidatabase Transactions Through Forced Local Conflicts. In *Int. Conf. on Data Engineering (ICDE)*, Kobe, Japon, April 1991.

[GRS93] D. Georgakopoulos, M. Rusinkiewicz, and A.P. Sheth. Using Tickets to Enforce the Serializability of Multidatabase Transactions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 6(1), 1993.

[GSC+83] N. Goodman, D. Skeen, A. Chan, U. Dayal, S. Fox, and D. Ries. A Recovery Algorithm for a Distributed Database System. In *Proceedings of the 2nd ACM SIGACT-SIGMOD*, pages 8–15, Atlanta, GA, Mars 1983.

[GSD97] A. Geppert, S. Scherrer, and K. Dittrich. Kids: A construction approach for database management systems based on reuse. Technical report, Université de Zurich, 1997.

[Gur03] L. Gurgen. Découverte de données dans les réseaux mobiles. Rapport de dea d'informatique : Systèmes et communications, LSR – IMAG, Juin 2003.

[Ham99] J. Hamilton. Networked Data Management Design Points. In *Proceedings of 25th International Conference on Very Large Data Bases*, Edinburgh, Ecosse, Septembre 1999.

[HCL+90] L. M. Haas, W. Chang, G. M. Lohman, J. McPherson, P. F. Wilms, G. Lapis, B. Lindsayand H. Pirahesh, M. Carey, and E. Shekita. Starburst mid-flight: As the dust clears. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), 1990.

[Hen98] N. Henry. Quark, Editeur de requjtes-risultats didii aux Bases de Donnies du Web. Rapport de dea d'informatique : Systèmes et communications, Laboratoire LSR – IMAG, Juin 1998.

[IBM95] IBM. Db2 relational extenders. Technical report, IBM Corp., 1995.

[IDE92] IDEA. Idea : Intelligent database environment for advanced applications. Technical Report Esprit 3: EP6333, 1992.

[JBE95] J. Jing, O. Bukhres, and A. K. Elmagarmid. Distributed Lock Management for Mobile Transactions. In *Int. Conf. on Distributed Computing Systems (ICDCS)*, Vancouver, Canada, Mai 1995.

[Jor] Jorm. http://jorm.objectweb.org/ .

[KA88] A. Kumar and A.Segev. Optimizing voting-type algorithms for replicated data. In *Advances in Database Technology EDBT'88*, LNCS 303, pages 428–442. 1988.

[KA00] B Kemme and G. Alonso. Don't be lazy, be consistent: Postgres-R, a new way to implement Database Replication. In *Proc. VLDB Int'l. Conf*, Cairo, Egypt, 2000.

[KG96] J. Kleinöder and M. Golm. Transparent and Adaptable Object Replication Using a Reflective Java. Technical Report TR-I4-96-07, Universität Erlangen-Nürnberg, Septembre 1996.

[Kin99] A. Kindler. A Classification of Consistency Models. Technical Report B99-14, Humboldt-Universitat zu Berlin, Institut für Informatik, D-10099 Berlin, Allemagne, Octobre 1999.

[KK00] K. Ku and Y. Kim. Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems. In *IEEE Workshop on Research Issues in Data Engineering*, San Diego, Etats-Unis, Février 2000.

[KKvST98] Anne-Marie Kermarrec, Ihor Kuz, Maarten van Steen, and Andrew S. Tanenbaum. A framework for consistent, replicated web objects. In *International Conference on Distributed Computing Systems*, pages 276–291, 1998.

[KPDS02] V. Kumar, N. Prabhu, M. H. Dunham, and A. Y. Seydim. TCOT- A Timeout-Based Mobile Transaction Commitment Protocol. *IEEE Transactions on Computers*, 51(10), 2002.

[Kum00] V. Kumar. A Timeout-based Mobile Transaction Commitment Protocol. In *ADBIS-DASFAA Symp. on Advances in Databases and Information Systems*, volume 1884 of *LNCS*, Prague, Republique Tcheque, Septembre 2000.

[LGA96] D.F. Lieuwen, N. Gehani, and R. Arlein. The Ode Active Database: Trigger Semantics and Implementation. In *Proc. 12^{th} Int'l Conf. on Data Engineering, New Orleans*, pages 412–420, March 1996.

[LH02] M. Lee and S. Helal. HiCoMo: High Commit Mobile Transactions. *Kluwer Academic Publishers Distributed and Parallel Database (DAPD)*, 11(1), 2002.

[Lim95] XOpen Company Limited. Distributed Transaction Processing: The Tx [Transaction Demarcation] Specification . Technical report, 1995.

[LLC+03] P. Lombard, A. Lebre, C.Guinet, O. Valentin, and Y. Denneulin. Distributed filesystems for clusters and grids. In *5th International Conference on parallel processing and applied mathematics*, Czestochowa,Pologne, Septembre 2003.

[LS94a] M. Little and S. Shrivastava. Object Replication in Arjuna. Broadcast project deliverable report, Dept. of Computing Science, Université de Newcastle, Royaume-Uni, Novembre 1994.

[LS94b] Q. Lu and M. Satynarayanan. Isolation-Only Transactions for Mobile Computing. *ACM Operating Systems Review*, 28(2), 1994.

[LS95] Q. Lu and M. Satynarayanan. Improving Data Consistency in Mobile Computing Using Isolation-Only Transactions. In *IEEE HotOS Topics Workshop*, Orcas Island, Etats-Unis, Mai 1995.

[MB01] S. K. Madria and B. Bhargava. A Transaction Model for Improving Data Availability in Mobile Computing. *Kluwer Academic Publishers Distributed and Parallel Databases (DAPD)*, 10(2), 2001.

[Min88] Jack Minker. *(Ed.) Foundations of Deductive Databases and Logic Programming.* Morgan Kaufmann Publishers, 1988.

[NMMS02] P. Narasimhan, L.E. Moser, and P.M. Melliar-Smith. Strong Replica Consistency for Fault-Tolerant CORBA Applications. *Journal of Computer System Science and Engineering*, 2002.

[OMG95] OMG. Corba services : Common object services specification. Object Management Group, Framingham, MA, Etats-Unis, 2.0 edition, 1995.

[OMG97] OMG. *CORBA services: Common Object Service Specification.* Object Management Group, Framingham, MA, USA, 1997.

[Ora99] Oracle. All your data: The oracle extensibility architecture. Technical report, Oracle, 1999.

[PB95] E. Pitoura and B. Bhargava. Maintaining Consistency of Data in Mobile Distributed Environment. In *Int. Conf. on Distributed Computer Systems (ICDCS)*, Vancouver Canada, Mai 1995.

[PB99a] E. Pitoura and B. Bhargava. Data Consistency in Intermittently Connected Distributed Systems. In *Transactions on Knowledge and Data Engineering*, volume 11, Novembre 1999.

[PB99b] E. Pitoura and B. Bhargava. Data Consistency in Intermittently Connected Distributed Systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 11(6), 1999.

[PD94] C. Roncancio P. Dechamboux. Peplomd: an object oriented database programming language extended with deductive capabilities. In *Database and Expert Systems Applications Conference, DEXA*, pages 2–14, 1994.

[PIN00] PING. Project presentation. Technical Report D6.1, Projet européen PING IST-1999-11488, 2000.

[PL91] C. Pu and A. Leff. Replica control in distributed systems: An asynchronous approach. In *In Proc. of the SIGMOD Conf.*, 1991.

[PSS+87] H. Paul, H. Schek, M. Scholl, G. Weikum, and U. Deppisch. Architecture and implementation of the darmstadt database kernel system. In U. Dayal and I. L. Traiger, editors, *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, California, Mai 27-29, 1987*, pages 196–207. ACM Press, 1987.

[PV91] D. Powell and P. Verissimo. *Delta4: A Generic Architecture for Dependable Computing*, chapter 6, pages 89–123. Distributed Fault-Tolerance. Springer-Verlag, 1991.

[RD96] C. Roncancio and P. Dechamboux. Active and deductive rules in an object oriented database programming language. In *Second International Baltic Workshop on Databases and Information Systems*, Estonia, Juin 1996.

[RND02] C.L. Roncancio, T. Nedelec, and PQ. Duong. Spécification du gestionnaire d'objets complexes. Technical report, Report 34 pages, Contrat France Télécom R&D cre no 00.1b.999, 2002.

[Rod99] L.M. Rodriguez. Exportation et visualisation de données sur le web. Rapport de dea d'informatique : Systèmes et communications, Laboratoire LSR – IMAG, Juin 1999.

[Ron94] C. Roncancio. *Règles actives et déductives dans les Bases de Données à objets.* PhD thesis, Université Joseph Fourier, Grenoble, France, Décembre 1994.

[Ron98] C. Roncancio. Toward duration-based, constrained and dynamic event types. volume 1553. Springer Verlag, 1998.

[RP95] K. Ramamritham and C. Pu. A formal characterization of epsilon serialisability. In *IEEE Transactions on Knowledge and Data Engineering*, volume 7, pages 997–1007, Décembre 1995.

[RR99] L. Rodríguez and C.L. Roncancio. Flexible data publishing on the www. In *Second WS on Adaptive Systems and User Modeling on the WWW (Position paper)*, Toronto, Canada, Mai 1999.

[SA04] P. Serrano-Alvarado. *Transactions adaptables pour les environnements mobiles.* PhD thesis, Université J. Fourier, Grenoble, France, 2004.

[SARA04] P. Serrano-Alvarado, C. L. Roncancio, and M. Adiba. A survey of mobile transactions. *Int'l Journal on Distributed and Parallel Databases (DAPD)*, à paraître 2004.

[SARAL03] P. Serrano-Alvarado, C. L. Roncancio, M. Adiba, and Labbé. Adaptable mobile transactions and environment awareness. In *BDA'03, 19èmes Journées Bases de Données Avancées*, Lyon, France, Octobre 2003.

[Sch90] F.B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. In *ACM Computing Surveys*, volume 22, pages 299–319. Décembre 1990.

[Sch98] M-L. Schneider. Mécanismes de caches sémantiques appliqués à la recherche d'information distribuée sur le Web. Rapport de dea d'informatique : Systèmes et communications, Laboratoire LSR – IMAG, Juin 1998.

[SZ97] A. Silberschatz and S. Zdonik. Database Systems - Breaking out the Box. In *SIGMOD Record*, 26(3), Septembre 1997.

[VC04] T. Vu and C. Collet. QBF: a Query Broker Framework for Adaptable Query Evaluation . In *6th International Conference On Flexible Query Answering Systems, FQAS04*, Lyon, France, Juin 2004.

[VHT99] M. Van Steen, P. Homburg, and A. Tanenbaum. Globe: A Wide-Area Distributed System. *IEEE Concurrency*, pages 70–78, January-March 1999.

[VM02] D. Hagimont V. Marangozova. An architectural approach to replication configuration. In *OPODIS*, 2002.

[VRL04] M.P. Villamil, C. Roncancio, and C. Labbe. PinS: Peer to Peer Interrogation and Indexing System. In *IEEE Intermntional Symposium IDEAS*, Juillet 2004.

[VS00] G. Vargas-Solar. *Service d'événements flexible pour l'intégration d'applications bases de données réparties.* PhD thesis, Université Joseph Fourier, Grenoble, France, décembre 2000.

[WC95] G. D. Walborn and Panos K. Chrysanthis. Supporting Semantics-Based Transaction Processing in Mobile Database Applications. In *Symp. of Reliable Distributed Systems (SRDS)*, Bad Neuenahr, Allemagne, Septembre 1995.

[WC97] G. D. Walborn and P. K. Chrysanthis. PRO-MOTION: Management of Mobile Transactions. In *ACM Symp. on Applied Computing*, San Jose, Etats-Unis, March 1997.

[WC99] G. D. Walborn and P. K. Chrysanthis. Transaction Processing in PRO-MOTION. In *ACM Symp. on Applied Computing*, San Antonio, Etats-Unis, Février 1999.

[Wie92] G. Wiederhold. Mediator in the Achitecture of Future Information systems. *The IEEE Computer Magazine*, 25(3):38–49, Mars 1992.

[YACS03] S. Yu, M. Aufaure, N. Cullot, and Stefano Spaccapietra. A collaborative framework for location-based services. In *Proc. CAiSE 2003*, Klagenfurt/Velden, Austria, June 2003.

[YZ94] L. H. Yeo and A. Zaslavsky. Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment. In *Int. Conf. on Distributed Computing Systems (ICDCS)*, Poznan, Pologne, Juin 1994.