Sistemas de Bases de Dados Móveis

Sílvia Maria Rodrigues da Cunha

Dissertação apresentada à Universidade do Minho para obtenção do grau de Mestre em Informática, na especialidade de Sistemas Distribuídos, Comunicações por Computador e Arquitectura de Computadores, elaborada sob orientação do Professor Doutor Orlando Manuel de Oliveira Belo

2004

Aos Meus Pais.

O nascimento de algo novo depende sempre do que já existe. A árvore só surge depois de semeada, as abelhas com o seu trabalho criam o doce mel, um livro nunca existiria sem o escritor... toda a criação necessita do seu autor.

Depois de noites mal dormidas, de mil ideias diferentes a efervescerem na cabeça, de correcções e alterações, o autor, no final, pode orgulhar-se e dizer: Eis a minha obra! Eis o meu trabalho! Eis o que sou e o que penso!

(Filipa Cunha)

Agradecimentos

Ao apresentar esta dissertação quero agradecer a todos aqueles que de alguma forma contribuíram para a sua concretização, em particular:

- Ao meu orientador Professor Orlando Belo pela orientação, acompanhamento, encorajamento e crítica constantes que me permitiram levar a cabo esta dissertação.
- Aos meus pais pelo apoio incondicional durante todo o processo de realização desta dissertação.
- Ao Sérgio Couto pelo apoio e encorajamento que transmitiu durante a realização deste projecto.
- Às minhas irmãs pelo encorajamento e paciência demonstrados ao longo de todo o processo.

Resumo

Nos últimos anos, tem-se vindo a presenciar a inúmeros avanços tecnológicos, tanto ao nível das comunicações sem fios como ao nível da computação portátil. Se por um lado as comunicações sem fios são cada vez mais rápidas e seguras, por outro as estações portáteis são, também, cada vez mais fáceis de transportar, uma vez que têm evidenciado uma diminuição do seu tamanho e peso, mantendo, todavia, alguma capacidade de processamento, armazenamento e autonomia das suas baterias. A par desses avanços tecnológicos têm surgido novos sistemas computacionais, que tentam tirar partido das facilidades fornecidas pela combinação de tais tecnologias. Os *Sistemas de Base de Dados Móveis* são um desses exemplos. Nestes sistemas, tal como é sabido, existe normalmente um conjunto de estações de trabalho distribuídas, em que algumas delas possuem uma localização fixa e conhecida e outras não. Estas últimas, por se poderem deslocar durante o período de execução das suas tarefas, são designadas de estações móveis. A comunicação entre as várias estações que integram um sistema distribuído pode ser realizada através de ligações com e sem fios. No caso de uma das estações do sistema não conseguir comunicar com uma outra, por motivos de falha do sistema de comunicações ou, simplesmente, por indisponibilidade da segunda estação, esta pode, contudo, continuar a executar as suas tarefas baseadas nos dados que mantém localmente, usufruindo assim da autonomia que o SBDM na qual está integrada lhe confere em termos de dados. Mais tarde, quando estiverem novamente repostas as condições para a comunicação entre as estações, os dados processados poderão ser então validados com outras estações do sistema. A autonomia é apenas uma das muitas vantagens e facilidades que os *Sistemas de Base de Dados Móveis* colocam à disposição dos seus utilizadores. Mas, como seria de esperar, nem tudo são vantagens. A implementação e

gestão destes sistemas não é nada simples, sendo, na generalidade, bastante mais complexa do que nos *Sistemas de Bases de Dados Distribuídos*. Questões como estas serviram de estímulo para o desenvolvimento de um trabalho de estudo pormenorizado e fundamentado sobre o domínio dos *Sistemas de Bases de Dados Móveis*, no qual se abordaria toda a problemática da sua implementação, manutenção e gestão, dando-se particular atenção a questões como os seus aspectos arquitecturais e funcionais, modelos de acesso e replicação de dados, sistemas de transacções, processamento de *queries*, manutenção de consistência, protecção dos dados e implementações reais. O resultado desse trabalho de estudo está apresentado nesta dissertação de mestrado.

Palavras Chave: Sistemas de Bases de Dados Móveis, Modelos de Acesso e Replicação de Dados, Modelos para Transacções Móveis, Manutenção de Consistência e Protecção de Dados.

Abstract

In the last few years, a lot of technological advances emerged in wireless communications and mobile computing fields. Wireless communications are becoming faster and safer and portable stations are evidencing a significant decrease on their size and weight, maintaining the same processing and storage capacities, and battery autonomy. As a direct consequence of these technological advances, new computational systems appeared integrating the best characteristics that the combination of these two technologies can provide. Mobile Database Systems are a clear example of such combination. They are characterized to have a set of distributed stations, where some of them have are fixed with a known localization and others are mobile, since they can move during the execution of their tasks. The communication between these stations can be supported by wireless communications. In a Mobile Database Systems, when one station can not communicate with another, due to a system failure or simple because the second one is unavailable, they can continue to execute their tasks based on their local data replicas. This is possible only because they have a high level of autonomy in terms of data provided by the Mobile Database System where they are integrated. Latter, when communications are re-established, the data that was processed locally have to be validated with the other system's stations. Autonomy is only one of the several advantages that Mobile Database Systems can offer to their users. However, these systems have also some disadvantages. Their implementation and management tasks are not simple. In fact, in most of the cases, they are more complex than conventional Distributed Database Systems. Questions like these stimulated the developing of a detailed and supported study about the domain of Mobile Database Systems, approaching the characteristics, functionalities, and common problems that can occur in the design, implementation, and

management of such systems. System architectural and functional features, access and replication data models, transactions systems, query processing, consistency maintenance models, data protection, and real current real-world implementations were some of the topics that were also explored. This thesis presents the result of that work.

Keywords: Mobile Database Systems, Access and Replication Data Models, Mobile Transactions Models, Consistency and Data Protection.

Índice Geral

Índice de Figuras

Índice de Exemplos

Índice de Tabelas

Lista de Siglas e Acrónimos

ACM	–	Application Communication Manager
API	–	Application Programming Interface
BD	–	Base de Dados
BDC	–	Base de Dados Completa
BDL	–	Base de Dados Locais
ER	–	Escalonador de Rede
ESM	–	Estação de Suporte Móvel
FCP	–	Fixed Chohort Process
FMP	–	Fornecedor Mestre Fixo
GO	–	Gestor de Objectos
GT	–	Gestor de Transferência
ICP	–	Infra-estrutura de Chave Pública
IU	–	Interface de Utilizador
JTID	–	Identificador de uma JT
KTID	–	Identificador de Transacção Canguru
MBD	–	Multi-Base de Dados
MM	–	Manipuladores de Mensagens
MMP	–	Processo Móvel Mestre
MTM	–	Gestor de Transacções Móveis
OB	–	Object Bus

PD	–	Protocolo de Desconexão
PMF	–	Processo Mestre Fixo
PMM	–	Processo Móvel Mestre
PQ	–	Processador de *Queries*
PR	–	Protocolo de Recuperação
QC	–	*Queries* Contínuas
QRC	–	Chamas de Procedimentos Remotos em Filas
RDO	–	Objectos Dinâmicos Realocáveis
RRC	–	Request/Reply Cache
SBD	–	Sumários da Base de Dados
SBDD	–	Sistema de Base de Dados Distribuído
SBDM	–	Sistema de Base de Dados Móvel
SGBD	–	Sistema de Gestão de Base de Dados
SGBDD	–	Sistema de Gestão de Base de Dados Distribuído
SMBD	–	Sistema Multi-Base de Dados

Capítulo 1

Introdução

1.1. O Advento da Mobilidade

O mercado das tecnologias da informação e da comunicação está em permanente mutação. Diariamente é-se confrontado com novos produtos e serviços que alteram de forma radical a própria maneira de ser e estar das pessoas. Hoje, podem-se encontrar, praticamente em qualquer sítio, sistemas computacionais que ajudam a realizar as tarefas mais mundanas, disponibilizando não só avançados meios de processamento de informação como também de comunicação. Por exemplo, o pagamento de contas de telefone ou de electricidade, pedidos de reservas para transportes ou mesmo a aquisição de bilhetes para um espectáculo de variedades pode ser feito, potencialmente, a partir de qualquer sítio, desde que se tenha acesso a um dos muitos pontos existentes da rede Multibanco ou, melhor ainda, se se puder usufruir de uma ligação à rede global. Qualquer uma destas possibilidades, facilita imenso as actividades quotidianas e, na generalidade

dos casos, poupa algum tempo por dia libertando, quando possível, as pessoas para poderem realizar outras tarefas. Isto não era possível há uma dúzia de anos atrás.

Nos últimos anos, a forte emergência de soluções computacionais, combinando as últimas novidades em termos de processamento de informação e comunicações, tem acelerado a implementação de novos serviços nas mais variadas áreas de actividade. Se se observar o que se está a passar no "mundo" das comunicações móveis, verifica-se que todos os dias são lançados inúmeros serviços e dispositivos computacionais para apoio a actividades laborais ou de simples lazer. Como se sabe, hoje já é possível fazer-se muita coisa a partir de um simples telemóvel. Ainda não permitem resolver todos os problemas, mas já vão ajudando um pouco, sendo actualmente, sem dúvida alguma, um excelente meio de processamento e de comunicação de dados. O horizonte que potencialmente nos apresentam é praticamente infinito. Daqui a muito pouco tempo, aquilo que actualmente se faz nas tradicionais plataformas computacionais fixas, poderá ser feito facilmente a partir destes dispositivos. Todavia, é possível que deles apenas fique o nome, já que serão transformados, certamente, em sofisticadas plataformas computacionais móveis de muito pequena dimensão. Porém, isto não será uma grande surpresa, porque, em certa medida, já existem actualmente sistemas que combinam esses tipos de tecnologias. Os *palmtops* são um exemplo muito concreto. Não é difícil encontrar no mercado algumas boas propostas (talvez ainda a um preço um pouco distante de um simples telemóvel) que combinam as características de um computador com as funcionalidades de um telemóvel, tendo-se a possibilidade de aceder, de uma forma muito eficaz, a todos os serviços mencionados.

Se os telemóveis representam um avanço surpreendente na utilização de plataformas de "computação" móveis no quotidiano das pessoas, os *palmtops* são hoje um dos trunfos mais fortes que as empresas têm para as ajudar nos permanentes desafios lançados pelos mercados onde desenvolvem prioritariamente as suas actividades. Longe estão os tempos em que saíam, pela manhã, as "trupes" de vendedores, armadas com os seus famosos blocos de notas para visitarem os clientes e recolherem os seus pedidos. Agora, é vulgar encontrá-los munidos de sofisticados *palmtops*, incorporando funcionalidades de comunicação e de acesso a repositórios remotos de dados, nos quais anotam as mais variadíssimas coisas, através das aplicações que têm disponíveis ou interagindo directamente com os sistemas computacionais das empresas através de um acesso remoto, normalmente suportado pela Internet. Desta forma, as suas actividades são reflectidas imediatamente nos sistemas centrais, assim como os clientes são informados, em tempo real, sobre o que podem solicitar ou, eventualmente, acompanhar os seus próprios processos de encomenda. A possibilidade de acesso remoto permite, assim, rentabilizar

significativamente o tempo de saída dos vendedores, como também, melhorar o seu desempenho, qualidade de serviço e, claramente, a sua disponibilidade para o serviço.

A massificação das comunicações sem fios trouxe novos desafios, mas, acima de tudo, novas formas de exploração de serviços. Agora, as pessoas já não precisam de ir ao encontro dos serviços. Os serviços vêm ao encontro das pessoas. As vantagens são claras. Já não há desculpas que possam justificar atrasos na realização desta ou daquela tarefa, nem a impossibilidade de aceder a este ou àquele repositório de dados. Desde que tenham os meios adequados, as pessoas podem fazer praticamente tudo, a partir de qualquer local. Nem sempre é uma situação conveniente para alguns, mas não deixa de ser, realmente, muito útil para quem anda frequentemente de lado para lado e tem muito pouco tempo para fazer as coisas. Basicamente, as comunicações móveis libertaram a computação tradicional. A dependência de um lugar fixo, de uma conexão por cabo, terminou. O advento da mobilidade trouxe a "liberdade" que faltava à plena utilização e exploração dos recursos computacionais. Está-se perante a alvorada da computação ubíqua generalizada.

As tecnologias de comunicação sem fios têm apresentado uma grande evolução, tanto ao nível do aumento da largura de banda oferecida, velocidade de comunicação e segurança na transmissão de dados, como na diminuição do número tramas reenviadas por falhas de comunicação. Hoje em dia, este tipo de comunicação está também mais barata, apesar de continuar a ser um pouco mais cara do que as comunicações efectuadas através de ligações com fios [Imieslinski & Badrinath 1993]. Além disto, existem cada vez mais organizações que disponibilizam acesso a este tipo de comunicações aos seus membros. De referir, o caso da generalidade das universidades, e de algumas empresas, com grande dinâmica e implantação de soluções avançadas de processamento de informação. Para tornar ainda mais fácil o acesso ao mundo através das comunicações sem fios, os operadores de comunicações móveis já disponibilizam actualmente soluções muito interessantes. Desta forma, os utilizadores podem comunicar com outras estações, aceder a redes remotas ou a páginas *Web*, a partir do ponto em que se encontram, não necessitando, assim, de regressar à sua base de trabalho.

As plataformas computacionais portáteis têm sofrido enormes melhorias ao nível do seu tamanho, peso e capacidade das baterias. Apesar do tamanho e do peso estarem directamente relacionadas com o tempo das baterias, actualmente já se consegue uma boa relação entre estas três características. A evolução das plataformas portáteis conjugada com as comunicações sem fios e com as actuais necessidades de mobilidade e comunicação constante sentidas pelas empresas, impulsionaram fortemente o aparecimento de novos sistemas computacionais. Estes

novos sistemas agrupam a computação portátil com as tecnologias de comunicação sem fios, e designam-se, normalmente, por sistemas de computação móvel [Pitoura & Bhargava 1994].

1.2. Sistemas Móveis

Um sistema móvel possui um conjunto de estações fixas, com uma localização conhecida e estática, e um conjunto de estações móveis, que podem efectuar deslocações durante a execução das suas tarefas, não possuindo portanto uma localização exacta. A comunicação entre estes dois tipos de estações processa-se, geralmente, através do uso de comunicações sem fios. Nestes sistemas o utilizador não necessita de se encontrar sempre conectado com a restante rede, ou seja, permite-se que o utilizador se encontre desconectado por alguns períodos de tempo, operando nesta altura apenas sobre os dados que estão armazenados localmente na sua estação. As desconexões podem ser voluntárias, para reduzir os custos de comunicação e para poupar recursos, ou involuntárias, sempre que ocorra uma falha no sistema. Além disto, encontrando-se conectado ou não, com a restante rede, permite-se que o utilizador se desloque durante a realização das suas tarefas [Dunham et al. 1997] [Elmagarmid et al. 1995].

As estações móveis possuem, geralmente, recursos mais pobres do que as estações fixas, uma vez que, para aumentar a portabilidade e a mobilidade dessas estações se perdem algumas capacidades ao nível do armazenamento e processamento de dados, bem como da autonomia das suas baterias. Por outro lado, como estas estações utilizam, geralmente, comunicações sem fios, facilmente se identificam outras possíveis limitações, tais como limitada largura de banda, nível de confiança e de segurança reduzido e custos de comunicação um pouco mais elevados. As estações fixas, por sua vez, são, eventualmente, poderosas plataformas que comunicam entre si através de comunicações com fios. Como se sabe, as comunicações com fios apresentam um maior nível de eficiência e de segurança do que as comunicações sem fios. Perante este cenário, pode concluir-se que num sistema móvel existe uma grande heterogeneidade ao nível dos recursos disponíveis nas diversas estações [Noble 1998] [Lubinski 1998].

Um sistema móvel é um sistema distribuído, uma vez que possui estações distribuídas ao longo de uma rede, na qual as comunicações entre as estações são assegurados por ligações com ou sem fios. Tal como o próprio nome indica, um *Sistema de Base de Dados Móvel* (SBDM) é um caso particular de um sistema móvel, que também integra os dois tipos de estações referidos. Da

comparação dos SBDMs com os *Sistemas de Base de Dados Distribuído* (SBDD), facilmente se conclui que os primeiros são uma extensão dos segundos, pois existe um conjunto de estações fixas, com características muito idênticas à dos SBDDs. Porém, os SBDMs contêm estações móveis, com algumas características que não são, normalmente, permitidas num SBDD. Na parte fixa de um SBDM têm de existir estações que sirvam de suporte aos utilizadores móveis, podendo mesmo funcionar como servidores de alguns dados. É ainda importante destacar que num SBDM com a deslocação das estações móveis também existe deslocação de informação, ou seja, existe mobilidade de estações e de informação [Özsu & Valduriez 1999] [Pitoura & Bhargava 1995].

Este novo sistema de base de dados apresenta inúmeras vantagens e facilidades aos seus utilizadores, como mobilidade e portabilidade, e, consequentemente, pode rentabilizar e optimizar a produção das empresas que os usam. Todavia, as características das estações móveis e das comunicações sem fios fazem com que a gestão e implementação de um sistema como o referido seja mais complexo do que um tradicional SBDD, uma vez que nestes sistemas a localização dos utilizadores e da informação era conhecida e estática e as ligações usadas eram seguras e de confiança, o que nem sempre acontece com as ligações sem fios utilizadas pelas estações móveis dos SBDMs.

1.3. Motivação e Objectivos

Hoje em dia é usual ver-se nos comboios, aeroportos ou mesmo em cafés, pessoas a trabalharem com as suas plataformas portáteis, como computadores e *palmtops*. Por outro lado, as comunicações móveis têm crescido drasticamente nos últimos anos, as pessoas já não são capazes de viver sem os seus telemóveis. As empresas, por sua vez, também já têm à sua disposição uma grande diversidade de plataformas portáteis e de dispositivos de comunicação sem fios. A sua conjunção, quando bem realizada, permite-lhes aceder a um novo conjunto de serviços que lhes trazem inúmeras vantagens para as suas actividades diárias. A aplicação destas tecnologias aos sistemas de base de dados é já uma realidade. Raras são as aplicações que dispensam as bases de dados. Todas elas, de uma forma ou de outra, precisam de ser "alimentadas" com dados, provenientes de fontes de informação, ou de armazenar a informação resultante dos seus processos de trabalho.

A partir do momento em que se começou a migrar aplicações informáticas para dispositivos móveis, foi também necessário estudar a melhor forma de garantir a continuidade dos serviços de acesso a bases de dados, quer estas estivessem armazenadas nas próprias plataformas móveis ou em servidores remotos. Adicionalmente, foi necessário precaver algumas situações que exigiam a realização de tarefas de processamento de dados nas estações móveis, mesmo quando estas não estavam conectadas a um sistema servidor de dados. De forma a garantir-se a autonomia tão desejável dessas estações, era, assim, necessário assegurar também que os serviços que eram mantidos por elas, o continuariam a ser mesmo quando não existisse a possibilidade de comunicar com a máquina que geria e mantinha os dados. Foi, então, necessário desenvolver alguns mecanismos que possibilitassem às plataformas móveis a manipulação dos dados das suas aplicações em situações de desconexão e que permitissem, mais tarde, quando se restabelecessem as conexões necessárias, a reconciliação desses dados nos seus servidores. Estas e outras situações, relacionadas com a manipulação de dados em plataformas móveis, deram origem ao estudo de um novo conjunto de problemas (e soluções) que foram dando origem a uma nova área que hoje é reconhecida como SBDMs. Toda a problemática que se desenvolve em torno destes sistemas é extremamente interessante, actuando frequentemente como catalizadora de novos estudos e trabalhos de aplicação em problemas do mundo real. Os permanentes desafios lançados pelos SBDMs, assim como o grande interesse e curiosidade pela sua aplicação foram, com certeza, a principal motivação para o desenvolvimento da presente dissertação.

Os SBDMs oferecem várias vantagens aos seus utilizadores, que deixam de estar “presos” a um local. Porém, esses sistemas também podem apresentar algumas limitações, como comunicações lentas, áreas sem cobertura de comunicação e baterias, armazenamento e processamento limitado. Estas limitações estão relacionadas tanto com o uso de comunicações sem fios como também com o uso de plataformas portáteis [Imieslinski & Badrinath 1993] [Pitoura & Bhargava 1994].

O principal objectivo desta dissertação é a apresentação e discussão de soluções para algumas das questões que podem surgir aquando da implementação de um SBDM. Em particular, tratar-se-á de:

- Identificar e analisar as principais características de um SBDM, fazendo-se referência a alguns dos problemas mais comuns que normalmente ocorrem aquando da sua implementação e gestão.

- Apresentar um conjunto de arquitecturas que suportem um SBDM, definindo-se as regras de comunicação entre os diversos intervenientes, bem como as relações e funções de cada um deles.

- Discutir alguns dos modelos de replicação, tendo-se em conta que nos SBDMs existem dois tipos bem distintos de estações, e sabendo-se, ainda, que os modelos devem garantir a autonomia das estações, principalmente das móveis.

- Introduzir o conceito de transacção móvel, evidenciando as principais diferenças das transacções tradicionais. Enumerar e destacar alguns dos modelos de transacções que incluam este novo conceito de transacção e que possam ser usados em ambientes móveis.

- Enumerar os novos aspectos que os SBDMs introduzem no conceito de *querie*, assim como no seu processamento, apresentando-se alguns dos modelos de processamento que podem ser aplicados a esses sistemas.

- Expor algumas políticas de gestão que evitem a criação de estados inconsistentes ou que tentem recuperar a base de dados para um estado consistente.

- Mostrar alguns dos problemas de segurança que poderão surgir nos SBDMs, enumerando algumas das possíveis soluções.

- Apresentar a implementação de SBDMs, já criados, que usem as políticas e modelos referidos ao longo da dissertação.

1.4. Organização da Dissertação

Ao longo desta dissertação são apresentadas algumas soluções para problemas que poderão surgir durante da implementação de um sistema de base de dados móvel. Para tal, esta dissertação, além do primeiro capítulo, inclui mais nove capítulos organizados da seguinte forma:

- **Sistemas de Bases de Dados Móveis.**

  Com este capítulo pretende-se definir o ambiente computacional de um SBDM, bem como as suas principais características. Na parte final do capítulo faz-se ainda um levantamento das questões que se devem ter em conta para construir e implementar um sistema como o referido.

- **Arquitectura para Acesso a Dados em SBDMs.**

  Neste capítulo apresentam-se alguns dos modelos existentes para acesso aos dados, quer os que se encontram na parte fixa quer na parte móvel da rede. As tradicionais arquitecturas de Cliente/Servidor não podem ser directamente aplicadas a estes sistemas, pois a localização dos seus clientes e alguns servidores não é conhecida. Deste modo, apresentam-se algumas adaptações dos modelos tradicionais, bem como alguns modelos novos para acesso aos dados neste tipo de sistemas.

- **Replicação de dados.**

  Também os esquemas de replicação tradicionais não podem ser directamente aplicados aos SBDMs, pois as réplicas primárias dos novos modelos de replicação devem encontra-se sobretudo na parte fixa da rede, dado que esta é mais segura e fiável, para além de que se encontra sempre disponível, ao contrário de algumas estações móveis que se podem encontrar temporariamente indisponíveis, devido a desconexões. Aqui apresentam-se alguns dos modelos de replicação de dados possíveis para os ambientes móveis.

- **Modelos de Transacções.**

  Com a alteração dos modelos de replicação e com as novas características dos utilizadores e ambientes móveis, é de esperar que também os modelos de transacções necessitem de ser remodelados. Aqui, a noção tradicional de transacção também deixa de ser válida, pois nem sempre se podem verificar as propriedades ACID. A maioria das transacções móveis terá de ser dividida em operações que poderão ser executadas em diferentes estações, tendo portanto de partilhar os seus estados intermédios. Além disto, uma transacção pode iniciar a sua execução numa estação e terminá-la noutra. Alguns dos modelos de transacções possíveis para estes ambientes são apresentados neste capítulo.

- **Processamento de *queries.***

  Neste capítulo discute-se a adaptação dos modelos de processamento e optimização de *queries* para SBDMs. Aqui, existe um novo tipo de *query*, cujo resultado pode depender da localização do utilizador que a processou, ou mesmo da altura em que se efectuou o pedido.

- **Gestão da Consistência dos Dados.**

  Se nos sistemas distribuídos, devido à existência de várias réplicas de um item de dados já se colocava o problema da consistência, neste tipo de ambientes este problema torna-se ainda mais evidente. Pois, para além de existirem inúmeras réplicas no sistema, algumas delas podem encontrar-se temporariamente indisponíveis ou não se conhecer a sua localização, para que se possa fazer a sincronização dos dados. Além disto, algumas estações podem encontrar-se desconectadas, continuando a operar sobre as réplicas locais, sem que haja qualquer controlo da consistência. Ao longo deste capítulo apresentam-se mecanismos que permitem gerir a consistência das réplicas, bem como para detectar possíveis conflitos entre elas.

- **Protecção dos dados.**

  Aqui aborda-se a questão da protecção dos dados quer a nível da sua recuperação, quando ocorrem falhas no sistema, quer ao nível da segurança de acessos. Nenhum dos modelos de tolerância a falhas e recuperação dos sistemas distribuídos podem ser directamente aplicados aos ambientes móveis, devido às novas características destes sistemas. Também aqui os requisitos de segurança são mais difíceis de se obter, pois os utilizadores móveis, ao longo das deslocações, estão mais sujeitos a roubos e a cópias não autorizadas. Além disto, os meios de comunicação sem fios podem ser mais facilmente interceptados, sendo portanto necessário assegurar a confidencialidade quer dos dados quer dos utilizadores.

- **Sistemas Implementados.**

  Neste capítulo descrevem-se alguns dos sistemas móveis actualmente implementados ou desenhados, apresentando-se as suas principais características e modos de funcionamento. Os sistemas estudados usam na sua gestão alguns dos modelos apresentados ao longo de outros capítulos desta dissertação, nomeadamente os

relacionados com a arquitectura de acesso aos dados, replicação, transacções, entre outros.

- **Conclusões.**

  Por fim, faz-se uma apreciação global dos SBDMs, salientando-se as suas questões de gestão e funcionamento mais relevantes. Além disto, e como um possível contributo final, refere-se um conjunto de modelos que poderão eventualmente constituir uma solução ideal para a implementação de um SBDM. Faz-se ainda referência a uma possível aplicação dos SBDMs no domínio dos sistemas de ensino inteligentes.

Capítulo 2

Sistemas de Bases de Dados Móveis

Era uma vez...

Há 30 anos atrás, o Sr. António montou uma empresa têxtil. Na sua empresa, construiu o departamento de vendas, com um funcionário cuja função era atender os clientes e registar as encomendas e vendas efectuadas. Este registo era efectuado manualmente, em formulários apropriados.

Com o passar dos anos, bem como com o aumento do número de clientes, o Sr. António decidiu colocar nesse departamento um pequeno sistema computacional, no qual se passaram a registar todas as encomendas e vendas efectuadas. Posteriormente, para melhor servir os seus clientes, aumentou o número de funcionários e de computadores neste departamento.

A última solução era, aparentemente, óptima. No entanto, com o aumento do número de concorrentes da empresa e com a diminuição do número de clientes, o Sr. António achou conveniente colocar alguns dos seus funcionários a trabalhar fora da empresa. Assim, enquanto que alguns dos funcionários

ficavam na empresa a fazer o trabalho dito tradicional, outros saíam para o exterior, todos os dias, à procura de novos clientes, registando nos seus blocos de notas todas as encomendas efectuadas, sendo estas introduzidas no sistema computacional assim que os funcionários chegavam à empresa.

Contudo, a introdução dessa informação era bastante morosa e originava frequentemente alguns erros de transcrição. Outro problema estava subjacente ao facto de que não existia qualquer tipo de comunicação entre os funcionários "móveis" e os "fixos", podendo alguns deles comprometerem-se com os clientes a cumprirem determinados requisitos da encomenda (por exemplo, datas de entrega) que a empresa não conseguia satisfazer.

Face aos problemas expostos, o Sr. António decidiu distribuir por cada um dos seus funcionários "móveis" uma estação móvel munida com dispositivos de comunicação sem fios. Com esta nova remodelação a empresa passou a ter os funcionários tradicionais, a manipularem o sistema tradicional, que, por sua vez, comunicavam com as estações móveis que se encontravam no exterior. Os funcionários podiam usar esta ligação para registar novas encomendas ou, apenas, para pedir informação sobre um determinado artigo ou cliente.

Embora este sistema possuísse inúmeras vantagens em relação a todos os outros sistemas já implementados na empresa, também possuía alguns problemas, tais como: o elevado custo de comunicação, algumas falhas na comunicação quando se tenta pedir ou guardar informações, baixa autonomia das estações móveis, entre outros. No entanto, as vantagens sobrepõem-se às desvantagens.

Hoje em dia, o Sr. António mantém este mesmo sistema informático. (…)

2.1. Mobilidade de Dados

O sistema descrito na história anterior é um exemplo típico de aplicação de um SBDM. Tal, pode-se justificar pelo facto deste sistema envolver um conjunto de estações fixas que interagem com um outro conjunto de estações móveis, através de ligações com ou sem fios. Além disso, numa

situação de falta de comunicação, esse sistema permite também que as estações móveis possam realizar o seu trabalho autonomamente, não dependendo assim de uma estação fixa. Contudo, posteriormente, é possível fazer-se a sincronização dos dados através da reflexão das actualizações efectuadas durante o período de desconexão, de modo a manter a consistência de todos os dados do sistema.

Basicamente, um SBDM pode ser visto como uma extensão dos tradicionais SBDDs. Estes consistem num conjunto de múltiplas bases de dados, logicamente interrelacionadas, sobre uma rede de computadores. No entanto, um SBDD não é apenas um conjunto de ficheiros que podem ser guardados individualmente em cada nodo da rede. Pois, para além de estarem logicamente relacionados, acompanham estruturas de metadados comuns que definem a forma de aceder aos ficheiros que contêm a informação. Assim, pode-se definir um *Sistema de Gestão de uma Base de Dados Distribuído* (SGBDD) como um sistema de software que permite a gestão dos diversos ficheiros distribuídos de uma forma transparente aos seus utilizadores.

Na Figura 1 apresenta-se uma possível configuração de um ambiente computacional de um SBDD. Este é constituído por um conjunto de estações de trabalho distribuídas ao longo de uma rede. Os dados, por sua vez, encontram-se também distribuídos ao longo dos sistemas de dados contidos nessas estações de trabalho - as *Bases de Dados Locais* (BDL), sendo estes visíveis a todas as estações de uma forma transparente [Özsu & Valduriez 1999].

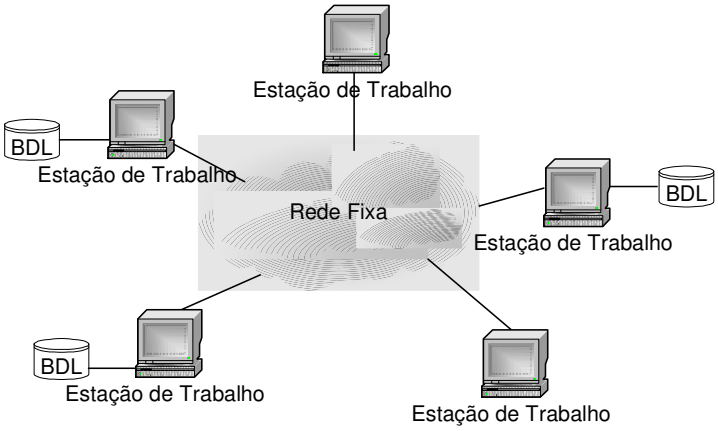

Figura 1- Ambiente de um Sistema de Base de Dados Distribuído.

Os *Sistemas Multi-Base de Dados* (SMBD) são uma extensão natural dos SBDDs, já que possuem também um conjunto de estações distribuídas ao longo de uma rede. Contudo, nos SMBDs, estas estações possuem uma maior autonomia, o que lhes providencia um controlo mais efectivo – praticamente total – das suas operações. Todavia, por uma questão de segurança, essa autonomia é limitada no sentido de não permitir que as estações integradas no sistema efectuem alterações locais nos sistemas de dados e de metadados. Por este motivo, deve existir um sistema global, estruturado em camadas, no topo dos SGBDDs locais. Uma dessas camadas deve ser a responsável por fornecer as funcionalidades completas da base de dados e por interagir com os respectivos SGBDDs locais através das interfaces de utilizador externas. O utilizador final tem assim a ilusão de existir apenas uma única base de dados. Desta forma, as características (boas e más) dos sistemas de dados locais não são visíveis aos utilizadores dos sistemas, tanto ao nível de software como de hardware. Assim, em termos gerais, um SMBD é constituído por várias bases de dados independentes, que actuam como um todo com o objectivo de fornecer um acesso uniforme a cada um dos SGBDDs locais (Figura 2) [Connolly & Begg 1999] [Segun et al. 2001]. No caso de os SBDDs ou os SMBDs permitirem que algumas das suas estações não se encontrem permanentemente conectadas à rede ou que se possam deslocar, mesmo quando se encontram a executar algum tipo de tarefa, então estamos perante um sistema típico de base de dados móvel.

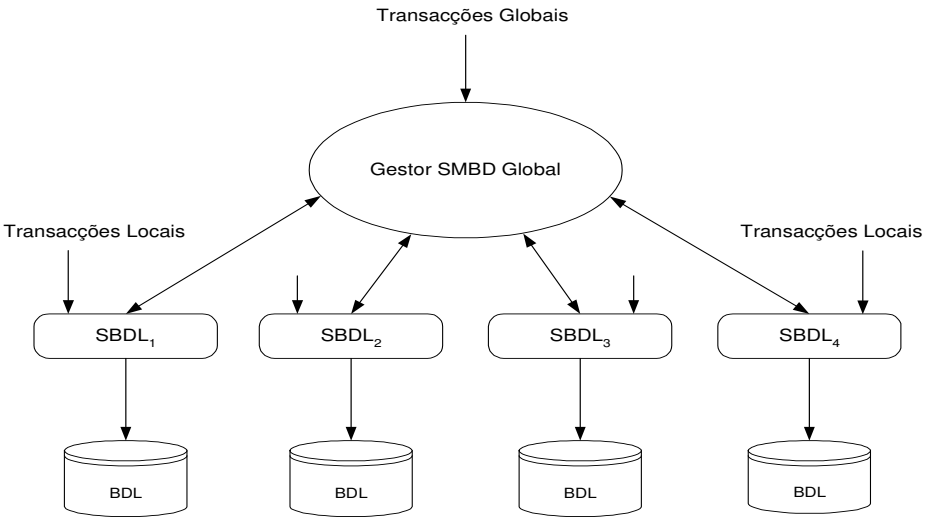

Figura 2 - Sistema Multi-Base de Dados.

Contudo, existem alguns autores que atribuem a esse novo tipo de base de dados o nome de Sistemas de Bases de Dados Nómadas, enquanto que outros os designam por Sistemas de Bases de Dados Móveis. Quando se usa o termo nómada é-se levado a fazer uma comparação com as tribos nómadas. Estas tribos deslocam-se, geralmente, para assegurar a sobrevivência dos seus membros, não possuindo desta forma um domicílio fixo [Nova 1998] [Dic] [Moderna 1987] [Lello 1988].

Se tomarmos como exemplo os casos das tribos pastorícias nómadas, que se deslocam em função dos pastos para os seus rebanhos, ou das tribos sazonais, que se deslocam conforme as estações do ano [Focus 1977] [Moderna 1987], constata-se que nestas tribos todos os seus membros se deslocam em simultâneo para um mesmo destino e com o mesmo objectivo. Quando se compara o comportamento destas tribos com o comportamento de um sistema de base de dados móvel, facilmente se verifica a existência de algumas semelhanças. Em ambos existe a necessidade de deslocação. No entanto, ao contrário das tribos, nos SBDMs cada uma das suas estações se desloca de forma independente das outras, com destinos e objectivos diferentes. Além disto, pode ocorrer a situação de que algumas das estações do sistema possuam uma localização fixa, circunstância que não acontece nas referidas tribos. Desta forma, cada estação isolada pode ser considerada como um sistema nómada, pois segue de perto a filosofia das referidas tribos. Porém, quando o sistema de base de dados integra simultaneamente estações móveis e fixas, o termo nómada pode não ser o mais adequado. Por este motivo, e dada a maior abrangência do termo móvel, optou-se pela sua utilização nesta dissertação.

São vários os autores que referem a arquitectura dos SBDMs [Dunham et al. 1997] [Elmagarmid et al. 1995] [Imieslinski & Badrinath 1994] [Pitoura & Bhargava 1994A] [Pitoura & Samaras 1998]. No entanto, todos eles convergem, sem surpresa, para um mesmo modelo (Figura 3), no qual existem dois tipos de estações, as fixas e as móveis. As estações móveis podem mover-se, alterando frequentemente a sua localização mesmo durante o processamento de uma dada tarefa. Por sua vez, as estações fixas possuem uma localização conhecida e estática. Algumas delas são designadas de *Estações de Suporte Móvel* (ESM) ou Estações Base[1]. Estas últimas estações asseguram a comunicação entre as estações móveis e a rede fixa.

[1] Estações vulgarmente designadas na terminologia inglesa por *Mobile Support Stations* ou *Base Stations*.

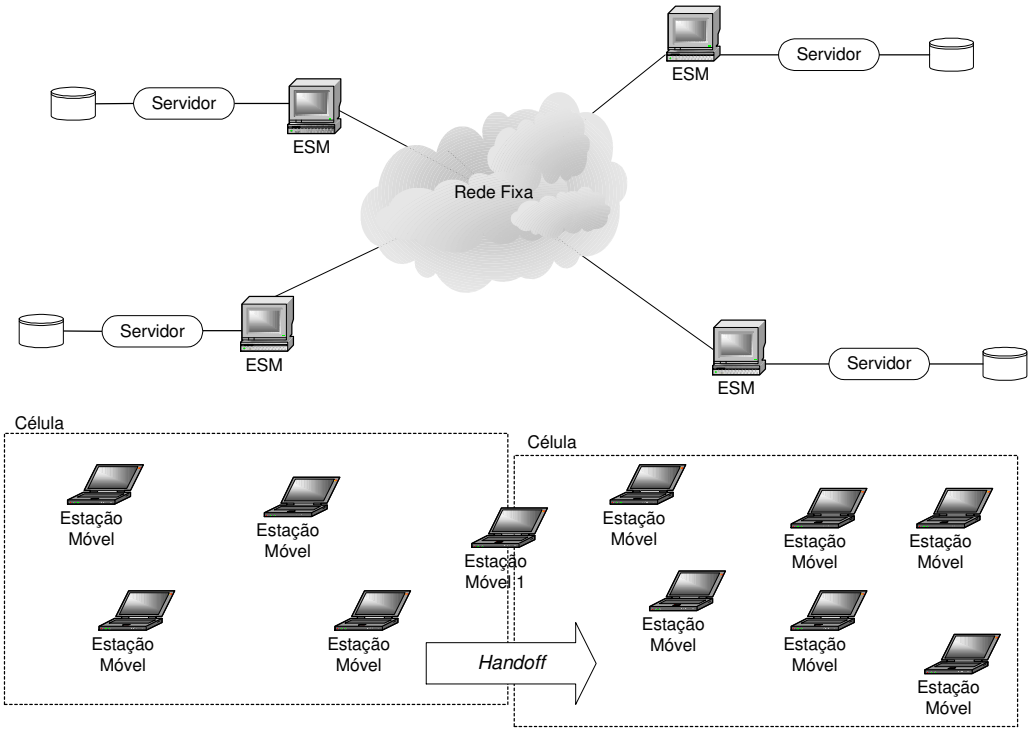

Figura 3 - Exemplo de uma Arquitectura de um Sistema de Base de Dados Móvel.

A cada ESM é atribuída uma área geográfica limitada usualmente designada por célula. Este tipo de estações têm a seu cargo a gestão de todas as estações móveis que se encontrem dentro da sua célula num determinado instante. As células são definidas de forma a constituírem conjuntos disjuntos, o que faz com que uma estação apenas se possa encontrar numa única célula em cada momento. A comunicação dentro de uma célula pode ser efectuada tanto através de uma conexão celular, como de uma conexão satélite ou de uma rede local sem fios[2]. Apesar de cada ESM ser responsável por uma única célula e de se saber que cada estação móvel comunica apenas com a ESM da célula onde se encontra, as estações que estejam localizadas em células distintas podem mesmo assim comunicar entre si através das respectivas ESMs. Assim, uma ESM actua como interface entre as estações móveis e as fixas, sendo também a entidade responsável pela troca de mensagens e dados entre essas estações [Gruenwald & Banik 2001].

Dada a grande mobilidade das estações, enquanto activas, pode acontecer que uma estação ultrapasse as fronteiras de uma célula, indo para outra (Figura 3, Estação Móvel 1). Nesta

[2] *Wireless Local Area Network*.

situação, a tarefa de encaminhamento de dados entre a rede fixa e a estação móvel deve ser transferida para a ESM da nova célula. A este processo dá-se o nome de *Handoff.*

De forma a que o utilizador não se aperceba dessa permuta de célula, é desejável que o processo de *Handoff* lhe seja transparente e que a ligação se mantenha, mesmo durante o instante da permuta. Para isso, é necessário que seja feita uma reconfiguração da topologia da rede. Além disto, quando a estação móvel se encontra a executar algumas tarefas com a cooperação da parte fixa do sistema, pode ser necessário que as tarefas em curso sejam transferidas para a ESM da nova célula.

[Madria et al. 1998] consideram que, para além das características apresentadas, tanto a ESM como a estação móvel possuem um *Processador de Queries*[3] (PQ) e *Sumários*[4] *da Base de Dados* (SBD). A ESM tem que conter também a *Base de Dados Completa* (BDC), Figura 4.

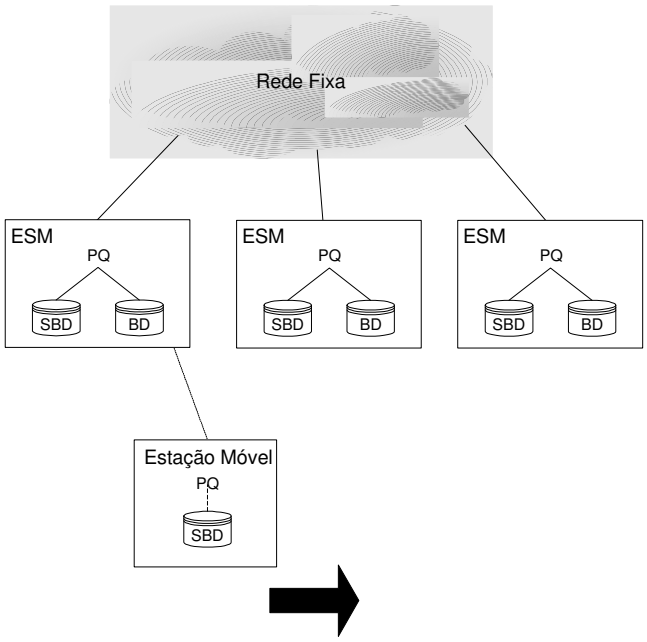

Figura 4- Arquitectura de um SBDM usando sumários da base de dados.

[Bukhres et al. 1997] apresentam um outro modelo para um SBDM, muito idêntico ao apresentado anteriormente na Figura 3. A única diferença reside no facto de que o modelo agora proposto

[3] Questões colocadas à base de dados.
[4] Extractos do conjunto de tabelas da base de dados.

define que cada estação móvel possui uma caixa de correio na qual se guardam todas as mensagens que lhe foram enviadas. Essa caixa de correio deve ter um acesso fácil e encontrar-se sempre disponível, de modo a que funcione como um repositório central. As ESMs têm de manter uma caixa de correio por cada estação móvel, podendo ser essa caixa acedida através da ESM actual. O uso das caixas de correio poderá trazer vantagens no processo de recuperação de transacções perante uma falha do sistema, já que mantêm um histórico com todas as mensagens trocadas até ao momento.

Em [Perich et al. 2001] é apresentado um modelo diferente dos anteriores. Nele é proposto um novo conceito de ambiente móvel *Ad-Hoc*, no qual cada estação móvel pode funcionar tanto como fornecedor, como consumidor de informação. Isto significa, que as estações móveis não são obrigatoriamente clientes e que nem sempre os servidores estão situados na parte fixa do sistema. As estações móveis são autónomas, dinâmicas e adaptam-se de acordo com o seu ambiente computacional, não sendo necessária a existência de suporte na rede fixa. Além disso, estas podem também cooperar com outros componentes do sistema na realização das suas tarefas. Num ambiente móvel Ad-hoc as estações móveis podem necessitar de informação, sendo mais fácil procurar essa informação numa estação vizinha do que na rede fixa. Por este motivo, as estações móveis mantêm alguma informação, mesmo quando já não necessitam dela.

[Kayan & Ulusoy 1999] consideram que a localização de uma estação móvel é equivalente a um item de dados, que se altera sempre que uma estação se desloca para uma nova célula. No entanto, a localização destas estações deve ser conhecida para que seja possível estabelecer-se a comunicação entre os vários intervenientes do sistema. Por esta razão, sempre que uma estação altera a sua localização, o seu endereço deve ser guardado na localização antiga, possibilitando assim o reencaminhamento das mensagens que lhe forem enviadas. Além disto, se tomarmos em consideração o trabalho de [Luccio et al. 2000], tem-se que os SBDMs são tipicamente sistemas assíncronos, nos quais as suas ESMs têm capacidade de difusão de mensagens (*brodcast*) de forma a facultar o seu envio para todas as estações localizadas na célula pela qual é responsável. Todavia, as ESMs podem também enviar de forma selectiva mensagens para uma dada estação.

Nos sistemas de base de dados relacionais os dados encontram-se estruturados num conjunto de tabelas. Porém, quando se pretende, por exemplo, limitar o acesso aos dados armazenados ou controlar a forma como estes são visualizados, é frequente definirem-se vistas de forma a garantir esse tipo de controlo. Uma vista é definida com base numa ou mais tabelas, seleccionando os atributos e os registos que se querem visualizar através dessa vista. As vistas não estão,

normalmente, materializadas, sendo na maioria dos casos tabelas virtuais. Isto é, não têm existência física no sistema. Apenas existem através de uma *query* que define a estrutura e o conteúdo dessa vista no momento em que é executado. A materialização dessas vistas pode ser efectuada, através do armazenamento físico dos registos que resultam da *query* definida sobre as tabelas. Desta forma, consegue-se que o acesso aos dados provenientes dessa vista seja mais rápido.

No fundo, as vistas materializadas funcionam como *caches* dos clientes, evitando-se assim o acesso à base de dados remota para o processamento de algumas tarefas, o que faz com que se obtenha um melhor tempo de processamento. Todavia, para que estas vistas possam ser usadas nos SBDMs, devem existir mecanismos para a sua manutenção e gestão, de forma a garantir alguma consistência entre os dados armazenados na estação móvel e os armazenados na rede fixa. Tais mecanismos designam-se por detentores de vistas[5], e têm, como principais funções, fornecer as vistas materializadas e controlar os acessos entre elas e a sua origem (Figura 5) [Lauzac & Chrysanthis 1998].

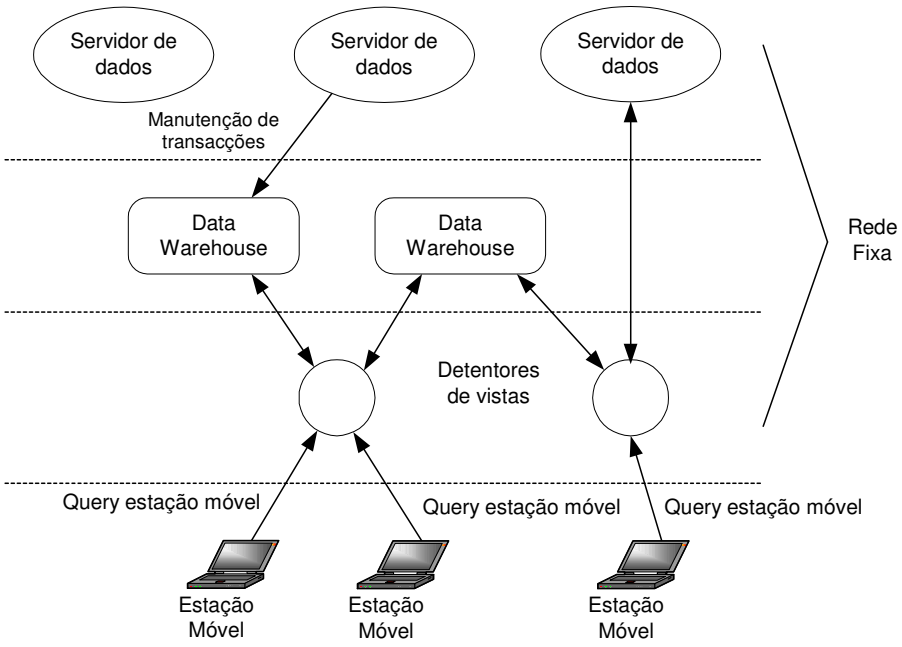

Figura 5 – Arquitectura de um SBDM usando vistas.

[5] *View Holder.*

Perante o cenário descrito ao longo desta secção, pode-se concluir que os SBDMs seguem uma filosofia idêntica à dos sistemas distribuídos tradicionais. Contudo, existem algumas pequenas diferenças. Nos SBDMs podem existir ligações com e sem fios, enquanto que nos sistemas distribuídos apenas existem ligações fixas, com fios. Por outro lado, os sistemas móveis possibilitam a deslocação de utilizadores, hardware, software e dados, o que, potencialmente, permite uma grande mobilidade de informação e recursos. Por fim, os SBDMs permitem, também, que algumas das suas estações estejam a trabalhar sem estarem conectadas ao sistema, garantindo-lhes um maior grau de autonomia. Assim, extensibilidade, portabilidade, heterogeneidade, mobilidade e comunicações sem fios, são, obviamente, características desejáveis para um SBDM.

2.2. Características do Ambiente Computacional de um SBDM

Apesar das inúmeras vantagens dos SBDMs, estes também possuem algumas limitações, pelo simples facto de terem que suportar algumas das suas características de referência. Em particular, deve referir-se, dentro de um leque mais alargado, as seguintes:

- **Comunicações sem fios.** Usada pelas estações móveis como uma das formas de acesso à rede fixa.
- **Mobilidade**. As estações móveis devem ter a capacidade e a possibilidade de alterar a sua localização.
- **Portabilidade**. As estações móveis têm de possuir um conjunto de características de modo a que sejam fáceis de transportar.

Consequentemente, nos sistemas móveis existe uma grande assimetria entre as estações móveis e as fixas. Isto, porque as móveis possuem apenas os recursos mínimos ao seu processamento, enquanto que as estações fixas têm a possibilidade de usar todos os recursos da rede fixa, que, de um modo geral, são mais ricos do que os das estações móveis. Os ambientes móveis devem ainda serem capazes de gerir todos os recursos de hardware e de interface subjacentes às redes

heterogéneas [Baggio 1999]. Nas secções seguintes faz-se a referência a todas essas características e às limitações que elas impõem dentro de um ambiente de um SBDM.

2.2.1. Comunicações Sem Fios

Em termos gerais, os SBDMs possuem várias estações, móveis e fixas, distribuídas ao longo de uma rede. As estações móveis podem comunicar com a rede fixa, através de uma conexão directa à rede, ou através do uso das comunicações sem fios, como arquitectura celular, serviços de satélite, LAN sem fios, etc. [Imieslinski & Badrinath 1993]. Um caso particular dos SBDMs surge quando não se permite que as estações móveis comuniquem com a rede fixa durante as suas deslocações, ou seja, não se recorre à utilização de ligações sem fios. Neste caso, as características do meio de comunicação são idênticas às de um sistema distribuído. Contudo, no caso de se utilizarem comunicações sem fios, convém ter em conta as suas próprias características e limitações, dado que este tipo de comunicação é mais difícil de se estabelecer e, geralmente, garante uma qualidade de serviço inferior. Tal, deve-se, em parte, ao facto dos ambientes que suportam este tipo de comunicações acolherem também muitos tipos de sinais e ruídos que perturbam o seu funcionamento. Algumas dessas características e limitações podem ser encontradas em [Alonso & Korth 1993] [Forman & Zahorjan 1994] [Katz 1995] [Bouguettaya 1996], nomeadamente:

- **Reduzida largura de banda.** A largura de banda de uma célula é normalmente um recurso limitado, já que, em cada instante, é dividida por todos os utilizadores que aí se encontram. Isto faz com que seja utilizada (ou ocupada) com muito cuidado, evitando-se desperdiçar largura de banda e optimizando-se a sua atribuição. Assim, devem ser usadas, sempre que possível, técnicas de compressão, como filtração e *buffering*, antes de qualquer transmissão de dados, de modo a reduzir a largura de banda utilizada.

- **Grande variação da largura de banda**. A disponibilidade da largura de banda oferecida a um utilizador pode sofrer grandes variações ao longo do tempo, dependendo do local onde esse se encontra. Isto deve-se ao facto de a largura de banda disponível diferir de célula para célula. Mesmo dentro de uma célula específica isso pode acontecer, já que também é influenciada pelo número de utilizadores que nela se encontram. Deste modo, um utilizador móvel, pode ver a sua largura de banda diminuída, simplesmente por terem chegado mais utilizadores a essa célula ou por se

ter movimentado para uma outra célula, com uma largura de banda inferior. A título comparativo, as estações fixas possuem, geralmente, uma maior largura de banda do que as estações móveis.

- **Desconexões frequentes**. Nos SBDMs permite-se que algumas estações se encontrem por vezes desconectadas, voluntária ou involuntariamente. As desconexões involuntárias podem ser causadas tanto por variações de sinal como por variações da largura de banda disponível, ou mesmo por falhas ocorridas nos sistemas. Quando existem desconexões deste tipo, só é possível restabelecer a comunicação quando as condições ambientais o permitirem. Por sua vez, as desconexões voluntárias só ocorrem por decisão do utilizador. Esta decisão pode ter como objectivo reduzir os custos de comunicação ou poupar a energia das estações, dado que as comunicações sem fios ainda são bastante caras e consomem muitos recursos das estações móveis. Neste caso, o utilizador restabelece a conexão quando achar mais conveniente.

- **As células suportam *broadcast***. Uma ESM deve possuir infra-estruturas que lhe permitam fazer o *broadcast* de mensagens para todas as estações que se encontrem dentro da sua célula. Desta forma, evita-se a comunicação da estação móvel para a ESM, já que esta é bastante mais cara do que a comunicação em sentido inverso (da ESM para a estação móvel). A ESM deve ter também a possibilidade de especificar apenas uma estação para receber uma determinada mensagem.

- **Riscos de segurança**. Os meios de comunicação sem fios são mais vulneráveis a perturbações de segurança do que os meios com fios, pelo simples facto de serem mais fáceis de interceptar. Assim, os meios de comunicações sem fios encontram-se mais expostos a intrusos, especialmente se o alcance da transmissão cobrir uma grande área.

Convém, ainda, realçar dois tipos de comunicação possíveis nos SBDMs: da estação móvel para a ESM, designada de *uplink*, e desta para a estação móvel, designada de *downlink*. No primeiro caso, devido aos baixos recursos das estações móveis, a comunicação é bastante mais cara, sendo portanto aconselhável, sempre que possível, a utilização de comunicações do tipo *downlink*. Este é o tipo de comunicação que é usado quando a ESM faz o *broadcast* de mensagens para estações que se encontrem na sua célula.

Algumas das características acima citadas são tratadas nos SBDDs como falhas ou excepções, como é o caso da variação da largura de banda e das desconexões. Contudo, os SBDMs devem lidar com essas características e não tratá-las como falhas, pois isso implicaria um grande número de reprocessamento de tarefas e, consequentemente, uma diminuição de trabalho útil. Todavia, e apesar dos inúmeros avanços tecnológicos que as comunicações sem fios têm apresentado, as características destes meios de comunicação ainda se reflectem, de um modo geral, num aumento do número de retransmissões, de processamento de protocolos de controlo de erros e de desconexões, bem como em atrasos nas transmissões.

2.2.2. Mobilidade

A mobilidade refere-se à capacidade que os utilizadores têm de alterar a sua localização, sem que percam a conexão ao restante sistema. A mobilidade é sem dúvida um factor fundamental nos SBDMs, uma vez que permite que alguns dos utilizadores se desloquem. Porém esta mobilidade pode aumentar a volatilidade de alguma da informação, pois os dados considerados estáticos para as estações fixas tornam-se dinâmicos para as estações móveis. Além disto, a mobilidade acarreta [Baggio 1999] [Segun et al. 2001] [Bouguettaya 1996]:

- **Migração de endereço.** Esta situação deve-se ao facto do endereço de rede de uma estação móvel se alterar dinamicamente, podendo mesmo passar para uma nova célula.

- **Informação dependente da localização.** Existe alguma informação cuja localização actual influencia os seus parâmetros de configuração, alterando também o resultado de algumas *queries* e transacções.

- **Rede dinâmica.** As redes deixam de ser estáticas, com ligações fixas e conhecidas. Nos ambientes móveis, há alterações frequentes, tanto ao nível das ligações como ao nível dos seus intervenientes.

A mobilidade pode ainda fazer com que haja um aumento da latência da rede e um aumento do risco de desconexão.

2.2.3. Portabilidade

A portabilidade, é outro dos aspectos que se deve ter em conta na construção de um SBDM e, corresponde à facilidade de transporte e à autonomia de uma estação móvel. Pode mesmo dizer-se que as estações móveis possuem uma maior portabilidade quanto menor for o seu tamanho e peso, mas com alguma capacidade de armazenamento, processamento e bateria.

Contudo, fazer com que as estações sejam cada vez mais leves e pequenas pode trazer algumas limitações, como [Pitoura & Bhargava 1994] [Rakotoniarainy 1998] [Imieslinski & Badrinath 1993]:

- **Tempo de "vida" limitado.** As baterias são o factor que mais influenciam o peso das estações móveis. Daí que reduzir o seu peso seja um aspecto importante para o aumento da portabilidade. Porém, uma grande redução do peso da bateria pode traduzir-se numa perda de portabilidade e não num aumento, pois em geral corresponde a uma diminuição do tempo de duração da bateria, o que implica recargas mais frequentes e, consequentemente, uma diminuição do uso dessas estações.

- **Interface de utilizador limitado.** A redução dos tamanhos do teclado e do monitor podem diminuir o peso e tamanho das estações. Contudo, também estes, quando reduzidos em excesso, podem comprometer a portabilidade, já que há uma redução da quantidade de informação apresentada ao utilizador, o que torna a visualização de toda a informação mais lenta, podendo causar atrasos na execução das tarefas seguintes.

- **Capacidade de armazenamento limitada.** Uma redução do tamanho físico das estações móveis pode resultar numa diminuição dos seus meios de armazenamento primários e secundários, limitando, deste modo, a capacidade de armazenamento destas estações e, consequentemente, a sua autonomia.

A portabilidade não diz respeito apenas ao tamanho e peso das estações, mas também à sua autonomia. Para tal, deve-se chegar a um compromisso entre todos estes aspectos, pois se por um lado é importante que uma estação móvel seja fácil de transportar, por outro lado é de igual forma importante que a estação possua capacidade de armazenamento, processamento e autonomia (em termos de bateria). Assim, devem ser usados outros métodos que facilitem a aquisição dos objectivos pretendidos. Por exemplo, como já foi referido, uma diminuição exagerada da bateria pode pôr em causa o tempo de processamento da estação. Por este motivo pode reduzir-se o seu tamanho até que seja possível obter um tempo de processamento razoável,

usando-se em simultâneo técnicas que minimizem o consumo de energia, como desligar os componentes individuais quando estão ociosos ou desenhar as aplicações de forma a requerer o menor número de computações e comunicações possível.

Das características acima citadas, pode-se concluir que as estações móveis são mais pequenas e leves do que as estações fixas, para que possam ser transportadas mais facilmente, o que, em conjunção com os custos e o nível de tecnologia, faz com que as estações móveis possuam menos recursos do que as fixas, incluindo memória, capacidade de disco e, até mesmo, o tamanho do monitor. Para além disto, as operações dos sistemas móveis dependem da bateria disponível na estação em que operam, podendo essa acabar durante a execução de uma transacção ou operação. Desta forma, fica mais uma vez evidenciada a grande assimetria existente entre as estações móveis e as fixas.

2.3. Modos de Operação dos Utilizadores Móveis

Dada a escassez de alguns recursos nas estações móveis, e como forma de se aumentar o seu processamento e autonomia, permite-se que esta se encontre a operar num dos quatro seguintes modos [Ahmed et al. 1996] [Pitoura & Bhargava 1994] [Pitoura & Samaras 1998]:

- **Completamente conectado.** Quando uma estação se encontra a operar neste modo significa que se encontra completamente conectada à rede fixa. Esta conexão pode ser efectuada quer através de uma ligação directa à rede fixa, quer através do uso de meios de comunicação sem fios.

- **Parcialmente conectado.** Uma estação móvel opera parcialmente conectada quando a largura de banda disponível é escassa, ou seja, quando a qualidade de comunicação é baixa, pretendendo-se, por este motivo, limitar as comunicações. Aqui, pode-se estabelecer a comunicação do tipo *downlink*, mas é impossível a do tipo *uplink*.

- **Completamente desconectado.** A estação móvel pode transitar para este modo de operação quando não se encontra no domínio de nenhuma célula, isto é, quando não consegue estabelecer a conexão com nenhuma ESM. A estação móvel pode também encontrar-se a operar neste modo por opção do utilizador. Por este motivo, este modo

de operação não deve ser considerado como uma falha e as tarefas desta estação devem continuar a ser executadas até que a comunicação se restabeleça.

- ***Doze mode*.** Quando uma estação móvel se encontra a operar neste modo tem, geralmente, como principal objectivo economizar a energia da estação. Para tal, reduz-se a velocidade do relógio do processador e virtualmente não se executa nenhuma computação.

Quando uma estação transita de um modo de operação para outro devem ser tomadas algumas medidas para que esta transição afecte o menos possível o seu processamento. Assim, é necessária a execução de alguns protocolos, nomeadamente (Figura 6) [Pitoura & Samaras 1998]:

- **Protocolo de desconexão (PD).** Este protocolo é executado antes da estação móvel se desconectar da rede fixa. Deste modo, a sua execução deve garantir a informação mínima para que a estação consiga continuar o processamento das suas tarefas durante o período de desconexão. Só se consegue executar este protocolo quando as desconexões são voluntárias, não o sendo possível quando ocorrem desconexões involuntárias.

- **Protocolo de conexão parcial.** O objectivo da execução deste protocolo é preparar a estação móvel de modo a que esta execute as suas tarefas comunicando o menos possível com o sistema fixo. Este protocolo deve ser executado quando a estação móvel passa a operar no modo de conexão parcial.

- **Protocolo de recuperação (PR).** Este protocolo é executado quando a estação móvel restabelece a comunicação com a rede fixa, depois de ter ocorrido uma desconexão, quer esta tenha sido voluntária ou involuntária.

- **Protocolo *Handoff*.** Executa-se este protocolo quando ocorre o processo de *Handoff*, dizendo qual a informação que deve passar da estação móvel para a estação fixa.

Na Figura 6 apresentam-se de forma esquematizada, todos os modos de operação em que a estação móvel se pode encontrar a operar, bem como os protocolos que devem ser executados quando há transição do modo de operação. Esta figura apresenta ainda algumas das possíveis causas da transição [Pitoura & Bhargava 1994]. Uma estação pode passar do modo completamente conectado para o *doze mode* para poupar energia, ou seja, quando tem pouca

energia (A). Por outro lado, quando restabelece os seus níveis de energia (B) pode voltar a operar no modo completamente conectado. Uma estação pode ainda transitar do modo completamente conectado para o parcialmente conectado, pelo facto de não existir largura de banda suficiente para se estabelecer a comunicação (C), devendo ser executado o PR assim que exista largura de banda suficiente para que a estação volte a operar completamente conectada. Quando uma estação se encontra completamente conectada pode transitar para o modo completamente desconectada por ter ocorrido uma falha no sistema (D), embora a estação possa transitar para este modo por opção do seu utilizador, sendo executado, neste caso, o PD. Sempre que uma estação que se encontra desconectada passa a operar completamente conectada deve ser executado o PR.

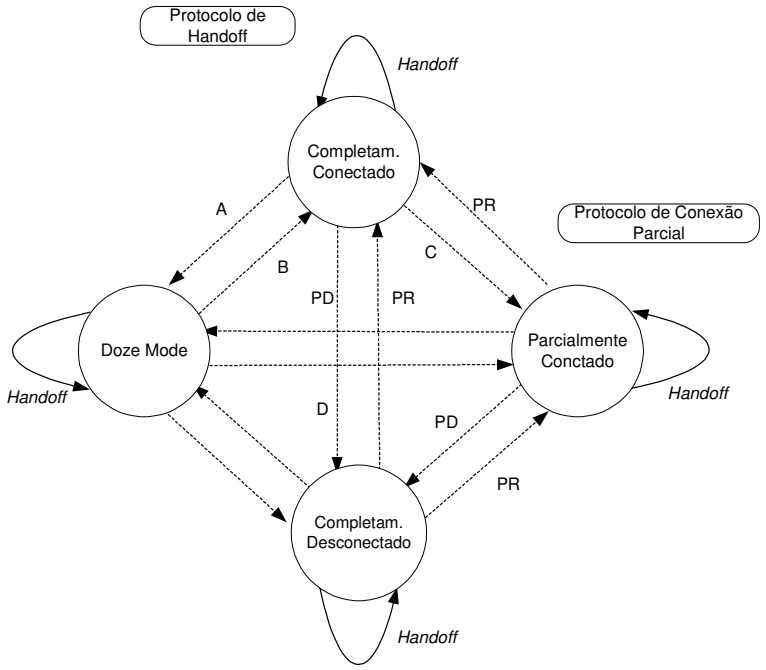

Figura 6 – Modos de Operação de uma Estação Móvel.

Deste modo, é importante fazer, mais uma vez, a distinção entre a desconexão voluntária e a desconexão provocada por uma falha na comunicação ou nos sistemas computacionais. Pois, estes dois casos devem tratados de forma diferente: no primeiro, é processado um protocolo de desconexão, que deve guardar todas as variáveis de estado, enquanto que no segundo não se faz qualquer armazenamento, devendo, portanto, ser tratado como uma falha. É muito difícil prever as

desconexões involuntárias de uma estação móvel, no entanto, a análise das diferenças de sinal poderá fazer prever uma falha na comunicação, porém, tal nem sempre é possível.

2.4. Adaptação

A heterogeneidade dos sistemas móveis e a grande variedade de recursos aí existentes fazem com que os utilizadores móveis se tenham de adaptar dinamicamente às características ambientais. Por exemplo, os recursos disponíveis nas estações móveis alteraram-se frequentemente, fazendo com que as estações exijam um acesso que consuma o menor número de recursos possível, quando estes são escassos, ou um acesso que tire partido da abundância de recursos [Noble 1998].

Existem três tipos de modelos que fornecem a adaptação das estações móveis ao ambiente computacional: adaptação transparente à aplicação[6], adaptação *deixa-fazer*[7] e adaptação baseada na aplicação[8]. A Figura 7 tenta apresentar a variação entre estas estratégias de adaptação [Satyanarayanan 1996].

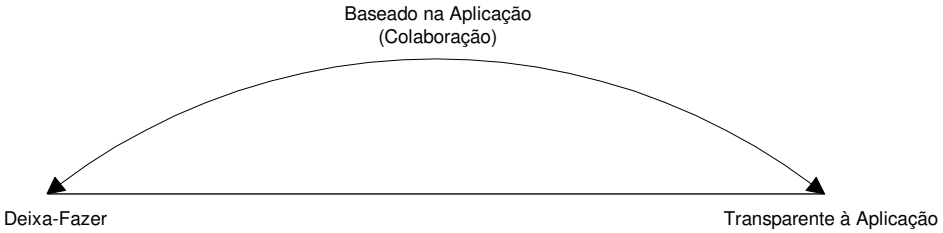

Figura 7 – Variação dos Modelos de Adaptação.

O modelo de adaptação transparente à aplicação [Noble 1998] considera que o sistema é o responsável pela adaptação às alterações no pedido de recursos. Assim, o sistema manipula dinamicamente as alterações de conectividade entre as várias estações e decide, de forma transparente, quando propaga ou invalida essas alterações. Uma desvantagem da adaptação

[6] *Application-Transparent Adaptation.*
[7] *Laissez-Faire Adaptation.*
[8] *Application-Aware Adaptation.*

transparente à aplicação é o facto de não suportar adequadamente a diversidade de aplicações, pois não permite que existam duas aplicações distintas a efectuarem pedidos de diferentes actualizações de adaptação para os mesmos dados.

Com o modelo de adaptação *deixa-fazer* [Noble 1998] as aplicações têm de efectuar a monitorização de toda a disponibilidade de recursos, realizando todas as decisões de adaptação isoladamente das outras aplicações. Algumas das vantagens associadas a este modelo são:

- Não é necessário suporte de sistema.
- As aplicações conseguem obter o comportamento de adaptação que pretendem, não sendo necessário o uso de aproximações.

Todavia, o modelo de adaptação *deixa-fazer* também possui algumas desvantagens, pois não há nenhum ponto de controlo de recursos, não existindo portanto controlo de concorrência. Além disto, algumas aplicações podem não se encontrar em locais a partir dos quais possam fazer a adequada monitorização de recursos.

Por último, o modelo de adaptação baseado nas aplicações [Noble 1998] [Noble 2000] tem por base os dois modelos descritos anteriormente. Aqui, parte-se do princípio que o sistema é a entidade que se encontra melhor posicionada para determinar quais os recursos disponíveis nas estações móveis. Por este motivo, neste modelo, delega-se ao sistema a responsabilidade da monitorização dos recursos disponíveis, sendo este também o responsável pelas decisões de alocação de recursos bem como pela optimização do seu uso por parte dos clientes. Por outro lado, as aplicações individuais são a única entidade que conhecem exactamente quais são as suas necessidades, o que faz com que, neste esquema de adaptação, as aplicações devam ser informadas de todas as alterações significativas que tenham ocorrido no sistema, para que se possam adaptar às alterações ambientais.

Deste modo, no modelo de adaptação baseado nas aplicações há uma divisão das responsabilidades de adaptação entre o sistema e as aplicações. Este modelo suporta diversidade e concorrência de aplicações, uma vez que lhes permite decidir quais as alterações na disponibilidade de recursos podem afectar a sua execução, além disto, o sistema tem o controlo da monitorização e da arbitragem de recursos.

As alterações ambientais existentes nos SBDMs podem ser tratadas através dos modelos de adaptação aqui descritos. Todavia, estes modelos devem guiar-se por dois requerimentos fundamentais dos sistemas móveis: primeiro, devem suportar uma grande variedade de aplicações, com diferentes necessidades; e, segundo, devem suportar aplicações concorrentes e competitivas. Com estes dois requerimentos consegue-se uma adaptação colaborativa entre o sistema e as aplicações.

2.5. Aspectos e Problemas dos SBDMs

Dadas as características dos SBDMs facilmente se conclui que são vários os parâmetros, como a largura de banda, a latência de transmissão ou a taxa de erros, que podem influenciar o desempenho e a execução destes sistemas [Bagrodia et al.1995]. Parâmetros como os referidos, poderão tornar mais problemática a obtenção de alguns requisitos do sistema, tais como acesso aos dados, gestão de transacções, consistência, segurança, privacidade, replicação, entre outros.

O grande número de estações fixas e móveis, normalmente existentes nos SBDMs e a conectividade intermitente de algumas dessas estações, fazem com que o ambiente operacional se altere dinamicamente. Por estes motivos, a gestão destes sistemas é mais complexa do que a dos sistemas distribuídos [Bagrodia et al.1995], nomeadamente ao nível da:

- **Granulosidade de dados.** Os diversos servidores podem apresentar vários formatos de dados.
- **Consistência dos dados.** Diferentes servidores podem apresentar diversos níveis de consistência.
- **Esquema de dados.** Os vários servidores podem encontrar-se organizados em diferentes esquemas e abstracções.
- **Limitações de largura de banda.** Os ambientes operacionais de diferentes ESMs podem suportar larguras de banda distintas.

A localização exacta dos utilizadores nem sempre é conhecida. O problema da localização pode ser dividido em duas operações: uma operação de chamada, para localizar uma estação em

movimento, e uma operação de movimento, para actualizar a informação de localização da estação. Em [Pitoura & Fudos 1998] apresenta-se um esquema para a localização de utilizadores móveis que se baseia em ponteiros de *forwarding*. Este esquema reduz os custos e o tráfego de rede, através de actualizações frequentes da localização em arquitecturas hierárquicas. Contudo, a tarefa de carregar a localização exacta de todos os utilizadores é desnecessária e irrealista, pois torna o sistema bastante pesado, podendo até conter algumas imprecisões relativamente à sua localização exacta. Deste modo, apenas se sabe que um cliente se encontra dentro da área geográfica de uma célula, sendo a ESM, dessa célula, responsável pela sua manutenção [Imieslinski & Badrinath 1993].

A deslocação das estações móveis pode acarretar problemas de gestão de dados, quando existem dados dependentes da localização. Para tal, devem existir políticas de gestão de dados que tenham em conta esta mobilidade dos dados [Baggio 1999].

As desconexões nos SBDDs eram tratadas como falhas do sistema, sendo em alguns casos, as estações faltosas excluídas do conjunto das estações participantes. Contudo, nos SBDMs isto não pode acontecer pois as desconexões são frequentes e bastante usadas. Devem-se, portanto, criar mecanismos que consigam distinguir as desconexões das falhas de sistema. Além disto, a execução das tarefas deve ser dinâmica e adaptar-se às alterações ambientais. Para além de que, pode ser necessário fazer a execução remota de algumas tarefas, isto é, as estações móveis devem ser capazes de delegar algumas tarefas à rede fixa, podendo ainda, estas estações ter de procurar recursos e serviços nas redes pelas quais vai passando [Baggio 1999].

A replicação é uma das técnicas usadas, nos SBDMs, quer para aumentar a autonomia das estações móveis, quer para reduzir a comunicação destas estações com as ESMs. A replicação é também um modo simples de fornecer transparência aos utilizadores móveis, pois deste modo quando alteram a sua localização o seu ambiente mantém-se, alterando, apenas, as variáveis de estado, que representam a sua passagem de uma ESM para outra. O uso da replicação obriga, também, a existência de uma gestão de dados local e outra global [Imieslinski & Badrinath 1993].

Um problema que se coloca aquando da replicação é saber quais os dados que serão necessários ao utilizador, ou seja, prever sobre que dados ele irá operar, para que se possam replicar estes dados enquanto a conexão se encontra activa. Depois de efectuada a replicação de dados e de ter ocorrido a desconexão, pode acontecer que ainda existam operações que necessitem de dados que se encontrem na parte fixa do sistema. Neste caso, deve existir uma fila de espera de

operações não executadas para que assim que se estabeleça a conexão estas possam ser executadas, pela mesma ordem com que o utilizador as tentou executar [Bagrodia et al.1995].

Os modelos de transacções, usados nos SBDDs, têm de ser revistos para que se consigam executar as transacções sem que se perca eficiência e confiança. Os novos modelos devem ter em conta que uma transacção pode ser iniciada numa estação e terminar noutra. Para além disto, as propriedades ACID devem ser revistas, de modo a que lidem com as características dos SBDMs, para que as transacções se executem com o menor número de suspensões e interrupções possíveis e com alguma confiança, quer nas estações móveis quer nas fixas.

As transacções atómicas são o modo normal de acesso aos dados partilhados nas bases de dados tradicionais. Contudo, as transacções móveis que acedem a dados partilhados não podem ser estruturadas usando transacções atómicas, pois estas executam em isolamento, não permitindo que a transacção se divida em várias operações, nem que se partilhe o seu estado e resultados parciais. E, nos SBDMs, as transacções móveis podem necessitar de ser organizadas como um conjunto de operações, algumas das quais a executar nas estações móveis e outras nas ESMs.

As alterações subjacentes aos ambientes móveis também vão influenciar o processamento de *queries*, podendo até limitar o seu processamento. Estas alterações fazem com que os algoritmos de optimização de *queries*, para obter os planos de processamento, se foquem principalmente nos custos das redes de comunicação sem fios, na disponibilidade da largura de banda e nos custos associados à eventual localização da informação.

Com a possibilidade de se fazer a replicação de alguns itens de dados e de se poderem executar transacções e *queries* sobre estes, mesmo durante as desconexões, é necessário que haja uma integração dos dados alterados, assim que a estação móvel se volte a conectar à rede fixa. Nesta altura, deve verificar-se se as alterações provocam inconsistências, ou seja, devem existir métodos que detectem e resolvam tais inconsistências e conflitos nos dados, restaurando a base de dados para um estado consistente.

Os SBDMs possuem ainda problemas de segurança e privacidade. Alguns destes problemas não são específicos dos sistemas de bases de dados, pois as estações móveis aparecem e desaparecem em diferentes locais. Além disto, estas estações, enquanto se movem estão mais sujeitas a roubos e a cópias não autorizadas. Numa rede onde se permite a conexão de estações visitantes, não se pode executar filtragem de pacotes, como se executava nos sistemas distribuídos, pois alguns deles podem ser legítimos. Nos SBDMs deve-se ainda ter em conta os

aspectos de segurança dos SBDDs, visto que os primeiros são uma extensão dos segundos [Alonso & Korth 1993] [Zaslavsky & Tari 1998] [Bagrodia et al.1995].

A computação móvel, por si só, acarreta também alguns problemas, pois os meios de comunicação sem fios podem ser mais facilmente interceptados, sendo portanto necessário assegurar a confidencialidade quer dos dados quer dos utilizadores [Baggio 1999]. São algumas questões como estas, associadas com a implementação e gestão dos SBDMs, que se discutirão com mais algum detalhe ao longo dos capítulos seguintes desta dissertação.

Capítulo 3

Arquitecturas para Acesso a Dados em SBDMs

Em semelhança ao que acontece nos SBDDs, também nos sistemas móveis têm de existir arquitecturas que possibilitem a comunicação, bem como a troca e acesso de informação, entre os vários membros da rede. Contudo, nos SBDMs, a arquitectura deve permitir que os utilizadores possuam alguma mobilidade e que possa existir informação dependente da localização. Desta forma, a arquitectura deve permitir que as estações se conectem de diferentes pontos, devendo permitir que os sistemas se reconfigurem dinamicamente [Liu et al. 1996].

[Bagrodia et al.1995] defendem que uma arquitectura para um SBDM deve ser transparente para o utilizador e ter a capacidade de se alterar dinamicamente de acordo com as características do ambiente computacional. Adicionalmente, deve também:

- Fornecer a inter-operação entre os vários tipos de infra-estruturas, quer sejam fixas ou móveis.

- Lidar com os imprevistos resultantes, quer dos comportamentos do utilizador quer da capacidade e da disponibilidade da rede.
- Fornecer escalabilidade, tendo em conta o espaço de endereçamento, a qualidade de serviço e o número de utilizadores intervenientes.
- Fornecer acesso integrado aos diversos serviços.
- Garantir independência entre as aplicações e a rede.
- Disponibilizar meios para que todos os elementos da rede possam cooperar.

Nas secções seguintes, apresentam-se alguns dos modelos usados nos SBDDs adaptados ou estendidos para os SBDMs, bem como alguns outros novos modelos criados especificamente para suportar as características dos sistemas móveis.

3.1. Modelo Cliente/Servidor

Existem diversos tipos de arquitecturas que seguem o modelo Cliente/Servidor. A variante mais simples deste modelo sucede quando existe apenas um servidor acedido por vários clientes. Do ponto de vista da gestão dos dados, este tipo de arquitectura não difere muito dos sistemas centralizados, já que os dados são também guardados numa única estação. Apenas existem algumas diferenças na forma como as transacções são executadas.

Contudo no modelo Cliente/Servidor também podem existir vários servidores, a serem acedidos por diversos clientes. Neste esquema existem duas soluções para o acesso aos dados: cada cliente conecta-se directamente ao servidor que possui os dados que lhe fazem falta ou, então, permite-se que cada cliente se conecte apenas a um servidor, sendo este o responsável por procurar e fornecer os dados que o cliente se encontra a pedir [Özsu & Valduriez 1999].

O modelo Cliente/Servidor usado nos SBDDs assume que a localização das estações é estática e bem conhecida. Por este motivo não podem ser directamente aplicados aos SBDMs. Em [Pitoura & Samaras 1998] apresenta-se uma possível adaptação do modelo Cliente/Servidor para ambientes móveis (Figura 8 (a)), que consiste num sistema chamado de Cliente, a pedir um

serviço a um componente da aplicação, a executar noutro sistema, chamado de Servidor. Sendo, nos SBDMs, as estações móveis são clientes a efectuar pedidos à entidade fixa. No entanto também podem existir clientes fixos.

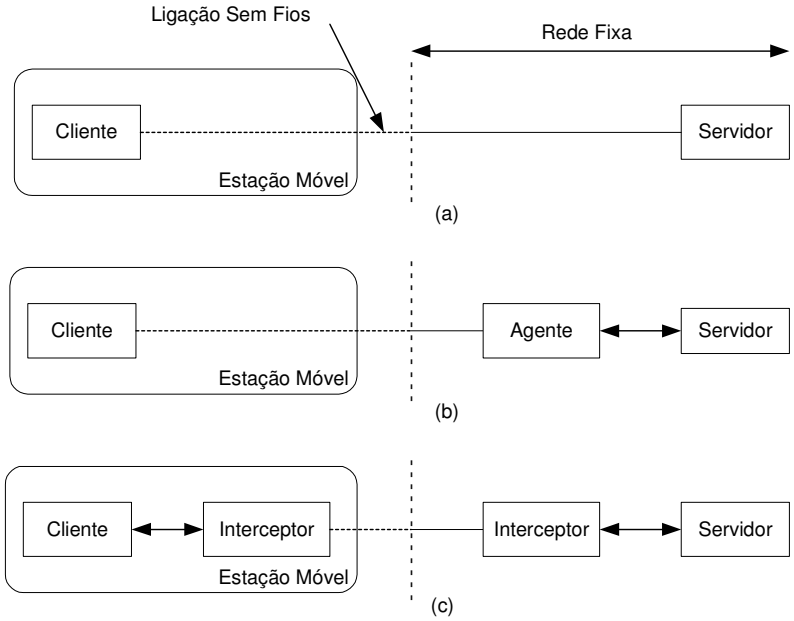

Figura 8 – Modelos baseados em Cliente/Servidor.

[Pitoura & Samaras 1998] consideram ainda que este modelo possui duas variações: Cliente/Interceptor/Servidor (Figura 8 (b)) e Cliente/Agente/Servidor (Figura 8 (c)). O modelo Cliente/Agente/Servidor usa três entidades: Cliente, Agente e Servidor. Os Agentes encontram-se sempre nos servidores e funcionam como intermediários entre os clientes e o servidor, não se permitindo que um cliente comunique directamente com um servidor. Esta arquitectura tenta aliviar o impacto de algumas das características dos SBDMs, nomeadamente a reduzida largura de banda e a pouca confiança dos meios de comunicação sem fios. O Agente tem de incluir, no mínimo, suporte para mensagens, bem como para a manutenção de filas de espera, de modo a que seja possível a comunicação entre as diversas entidades.

[Heuer & Lubinski 1996] consideram que os agentes da arquitectura Cliente/Agente/Servidor funcionam como intermediários, sendo também utilizados para filtrar ou atrasar os itens de dados

em ligações lentas. Defendem ainda que o desempenho destes agentes depende muito da largura de banda disponível.

O modelo Cliente/Agente/Servidor é mais adequado para sistemas com energia e recursos limitados, uma vez que há a transferência de algumas responsabilidades do Cliente para o Agente. Apesar das inúmeras vantagens deste modelo, ele apresenta uma grande desvantagem, em SBDMs, pois não consegue sustentar o processamento dos Clientes durante as desconexões. Assim, quando ocorre uma desconexão o Cliente móvel perde a possibilidade de continuar a executar as suas tarefas.

Ao contrário do esquema anterior, no Cliente/Interceptor/Servidor, existem dois Agentes Interceptores, uma do lado do Cliente e outro do lado do Servidor. Neste esquema a comunicação entre o Cliente e o Servidor é recorrendo ao uso desses agentes interceptores. Todavia, dadas as frequentes limitações de recursos que as estações móveis apresentam, pode acontecer que estas estações não possuam os recursos suficientes para suportarem os agentes interceptores. Outra desvantagem deste modelo é que todas as aplicações requerem processamento tanto da parte do Servidor como da parte do Cliente, devendo-se desenvolver um par de agentes para cada instância da aplicação.

3.2. Modelo Cliente/Servidor com Clientes *Hoard*

Uma outra variação do modelo Cliente/Servidor para ambientes móveis foi apresentada em [Badrinath & Phatak 1997] e [Phatak & Badrinath 1998]. Nesta abordagem, os clientes podem operar durante os períodos de desconexão, tendo permissão para acumular dados localmente nos seus discos através da criação de réplicas locais. Neste modelo os clientes são designados por Clientes *Hoard* (Figura 9).

Durante as desconexões as estações móveis podem executar tarefas sobre dados locais, reconciliando mais tarde as alterações efectuadas quando a conexão se restabelecer. Os Clientes *Hoard* não necessitam de replicar as tabelas de dados completas, podem replicar apenas fragmentos. A forma de replicação é decidida pelos próprios clientes, o que resolve alguns problemas de escala do sistema.

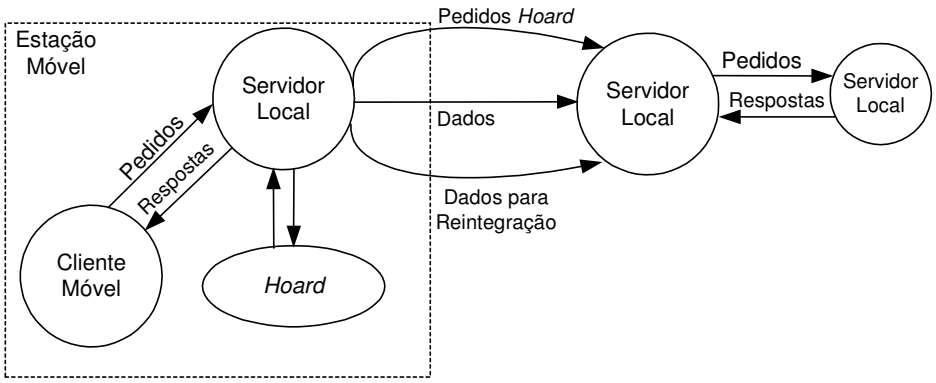

Figura 9 – Modelo Cliente/Servidor Com Clientes *Hoard.*

Nesta arquitectura, os clientes tradicionais (fixos) não podem operar sobre os seus próprios dados, confiando na conexão que possuem com o servidor para executar as suas tarefas. Durante as conexões os Clientes *Hoard* podem comportar-se como clientes tradicionais, tendo permissão para acessos completos à base de dados do servidor. Com a introdução deste tipo de clientes, os servidores da base de dados têm de responder a dois tipos de pedidos: os dos Clientes *Hoard* e o dos clientes tradicionais.

O servidor é o ponto de sincronização desta arquitectura. Todas as actualizações realizadas devem ser reflectidas no servidor. Além disto, é o servidor quem faz o controlo dos acessos aos dados efectuados pelos clientes. O servidor, ao contrário dos clientes, armazena as relações completas. Contudo a organização física dos dados nele armazenados baseia-se em fragmentos, de modo a que as operações de *hoarding* e de reintegração sejam realizadas de forma mais rápida. Os servidores devem ser projectados cuidadosamente, para que capturem correctamente a localização do acesso dos clientes *hoard*.

Do lado do servidor, todas as operações de reintegração e *hoarding* são executadas sobre os fragmentos. O servidor não necessita de guardar os dados que foram armazenados para cada um dos clientes. Neste modelo, os fragmentos físicos são as unidades de *hoarding*, de reintegração, de controlo de concorrência e de controlo de acesso. Cada cliente *Hoard* deve informar o servidor acerca dos fragmentos que pretende usar dados. Como já foi referido, um cliente *Hoard* não necessita de armazenar o fragmento completo, contudo do ponto de vista do servidor os clientes armazenam todo o fragmento. O servidor tem também a responsabilidade de resolver todos os conflitos que surjam.

Em [Badrinath & Phatak 1998] defende-se o uso de chaves *hoard* ao longo das chaves primárias e secundárias de cada relação para facilitar o acesso aos dados. Estas novas chaves devem capturar os padrões de acesso aos dados dos clientes móveis. Cada chave *hoard* particiona a relação em conjuntos disjuntos de fragmentos horizontais lógicos. Estes fragmentos constituem a granulosidade *hoard*. Os clientes *hoard* podem fazer o *hoard* ou reintegrar dentro dos limites destes fragmentos.

Existem dois mecanismos que permitem o acesso a estes dados. No primeiro, a base de dados pode manter estruturas de dados indexadas associadas às chaves *hoard*, sendo neste caso, uma chave *hoard* lógica, pois não afecta a organização física da base de dados. Os fragmentos lógicos são especificados por estruturas de dados indexadas especiais que instanciam o mapeamento entre *tuplos* e fragmentos lógicos. No segundo mecanismo, a base de dados é fisicamente organizada como um espelho dos fragmentos lógicos. Neste caso, a chave *hoard* é física, pois controla a organização física da base de dados. Aqui, os *tuplos* da base de dados são agrupados pelos fragmentos lógicos associados à chave *hoard*. Estas chaves permitem um acesso aos dados mais rápido para os clientes móveis que pretendem fazer o *hoard* de dados. Isto deve-se ao facto dos dados se encontrarem fisicamente contíguos no disco, podendo, ser acedidos eficientemente. Contudo, esta organização pode afectar o desempenho dos objectivos gerais das *queries,* porque os *tuplos* já não se encontram agrupados no índice primário. A cada fragmento físico pode-se dar autonomia. Isto é, cada fragmento pode ter o seu próprio servidor, o que faz com que sejam necessárias estratégias de indexação e políticas de controlo de concorrência.

[Badrinath & Phatak 1998] definem ainda o mapeamento dos tuplos dos fragmentos lógicos, através da especificação de um conjunto de relações de qualificação ou qualificadores. Cada uma destas relações define um fragmento lógico, tendo um funcionamento idêntico à cláusula WHERE das *queries* SQL. Deste modo, as relações de qualificação especificam agregações dos valores chave. Os qualificadores fornecem um mecanismo para uma localização precisa de acesso e actualização, controlando ainda o tamanho dos fragmentos.

3.3. Modelo Cliente/Servidor Estendido

Como já referido, o modelo Cliente/Servidor tradicional assume que a localização quer dos clientes quer dos servidores é fixa. Todavia, nos SBDMs tal não é possível. Assim, para que se possa

utilizar este modelo em sistemas móveis em [Jing et al. 1999] é proposto um modelo Cliente/Servidor estendido. Neste modelo, permite-se que, em alguns casos, os clientes executem funções de servidores e que, por sua vez, os servidores efectuem também funções de clientes. Como os clientes nos SBDMs possuem, geralmente, recursos limitados, pode ser necessário que algumas das suas tarefas, que necessitem de mais recursos, sejam executadas no servidor. Contudo, nesses ambientes as condições para comunicação nem sempre são as mais adequadas, podendo ocorrer situações nas quais as comunicações são muito fracas ou, simplesmente, não podem ser efectuadas. Nestes casos, os clientes devem poder executar alguns dos seus trabalhos, actuando como servidor das suas tarefas (Figura 10).

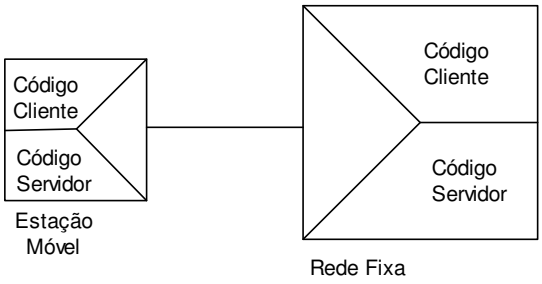

Figura 10 – Arquitectura Cliente/Servidor Estendido.

Este modelo apresenta algumas variações. Em casos extremos, possibilita todas as funcionalidades de servidor a um cliente (arquitecturas de clientes completos[9]) ou permite a implementação de arquitecturas que reduzem ao máximo o número de operações executadas nos clientes (arquitecturas de clientes pobres[10]). Entre estes dois casos, tem-se uma solução intermédia (uma arquitectura Cliente/Servidor flexível) que permite que o cliente realize algumas das suas tarefas, com algumas restrições, mas sem o objectivo de evitar o processamento dos clientes, como acontecia nas arquitecturas com clientes pobres [Jing et al. 1999].

[9] *Full Client Architecture.*
[10] *Thin Client Architecture.*

3.4. Modelo Ponto-a-Ponto

Em contraste com as arquitecturas Cliente/Servidor, numa arquitectura Ponto-a-Ponto distribuída não há distinção entre clientes e servidores, ou seja, cada componente do sistema pode funcionar tanto como cliente como servidor. Este modelo fornece independência de dados e transparência sobre a sua localização e replicação. Da perspectiva lógica dos dados, a arquitectura Cliente/Servidor e a Ponto-a-Ponto fornecem a mesma vista do sistema, pois ambos fornecem a ideia de uma única base de dados, enquanto que, na realidade, os dados se encontram fisicamente distribuídos.

Teoricamente, no modelo Ponto-a-Ponto cada estação de trabalho tem a funcionalidade completa tanto para ser um servidor como para ser um cliente. Daqui pode-se concluir, que em ambiente móveis as estações móveis podem funcionar como entidades iguais às estações fixas na realização de tarefas distribuídas. Este modelo é, assim, apenas apropriado para clientes com muitos recursos.

No modelo Ponto-a-Ponto, as desconexões acarretam ainda a desvantagem de falharem a pedidos de clientes que possam pedir os seus serviços. Deste modo, é desejável a ligação contínua da estação móvel, facto que não é praticável nestes ambientes, quer pelos custos de comunicação quer pela impossibilidade de se estabelecer uma ligação com a rede fixa, quando a estação móvel se encontrar numa área sem cobertura de rede. Assim, para tentar ultrapassar estas limitações, em ambientes que usam este tipo de arquitectura são, normalmente, usados mecanismos que iniciem a aplicação das estações automaticamente quando existem pedidos de clientes. Caso contrário, estas encontram-se desligadas.

3.5. Modelo Baseado em Agentes Móveis

[Antila et al. 2001] apresentam um modelo que se baseia na existência de processos designados de Agentes Móveis. Estes agentes são enviados por uma estação para acompanhar uma determinada tarefa. No fundo, são módulos de software que encapsulam dados e código, para cooperar na resolução de tarefas complexas, bem como, para as executar em estações remotas com a menor interacção possível do utilizador. Depois de ter sido submetido, um Agente procede autónoma e independentemente do Cliente e do Servidor. Quando o Agente Móvel chega ao

Servidor, é então enviado um agente de execução, sendo iniciada a sua parte executável. Quando concluir o cálculo, o Agente envia o resultado para o Cliente. Os Agentes Móveis também podem ser usados em conjunção com agentes localizados na rede fixa.

Algumas das vantagens deste modelo, relacionadas com o uso de agentes móveis, são:

- **Comunicação assíncrona**, uma vez que os agentes podem operar independentemente, mesmo durante as desconexões das estações móveis.
- **Comunicação autónoma**, já que os agentes podem tomar autonomamente algumas decisões a favor do utilizador como, por exemplo, voltar a tentar a execução de uma transacção que falhou.
- **Comunicação remota**, pois os agentes comunicam remotamente com os servidores para poderem usufruir dos seus recursos.

Em [Samaras et al. 2000] apresentam-se duas outras razões que reforçam a justificação para a utilização de agentes móveis, referindo que estes:

- Fornecem uma procura de serviços de informação eficiente, assíncrona e flexível.
- Suportam conectividade intermitente, redes lentas ou componentes de rede pesados.

Por estes motivos, o uso de agentes móveis nos SBDMs torna-se bastante atractivo, dado que nestes sistemas os recursos são normalmente escassos e os componentes um pouco pesados. Os agentes poderiam contribuir para libertar as estações móveis de eventuais *overheads*.

3.6. Modelo Baseado em Objectos

Em [Bönigk & Lubinski 1996] apresenta-se uma nova arquitectura para ambientes móveis que tem como objectivo principal a minimização da largura de banda necessária e poupança dos recursos dos sistemas móveis, tentando que as estações móveis tenham um trabalho mais confortável. O sistema de arquitectura básico tem de funcionar como uma plataforma flexível para a troca eficiente de todos os tipos de informação num ambiente móvel. Par tal, consideram os diferentes

componentes que caracterizam os ambientes computacionais móveis e que influenciam a sua arquitectura, distinguindo-se, assim, quatro classes de contexto: utilizadores, recursos, informação e dependências de localização e de tempo.

O principal conceito, nesta arquitectura, é o seu desenvolvimento orientado aos objectos, sendo o *Object Bus* (OB) o seu aspecto central. O OB serve como camada transparente para a comunicação móvel, sendo o responsável pela entrega de mensagens quer nos sistemas móveis quer nos fixos. Quando há troca de objectos de resposta são usados os *Manipuladores de Mensagens*[11] (MM), em vez dos processos de comunicação, que devem notificar os intervenientes acerca dos procedimentos de transferência para um certo objecto e que transferem o objecto de resposta. O MM serve também para trocar pedidos e objectos de resposta adequadamente entre processos, bem como para fornecer um acesso simples às unidades de informação. Em ambientes móveis, onde as redes são lentas e pouco credíveis é preferível o uso de mecanismos de mensagens assíncronos, em vez de síncronos.

O OB tem como tarefas principais a gestão e a transferência eficiente de todas as mensagens em todos os tipos de redes. O OB está ligado com o *Gestor de Objectos* (GO) local, no qual pode aceder à informação de que necessita para a realização do seu trabalho, (Figura 11).

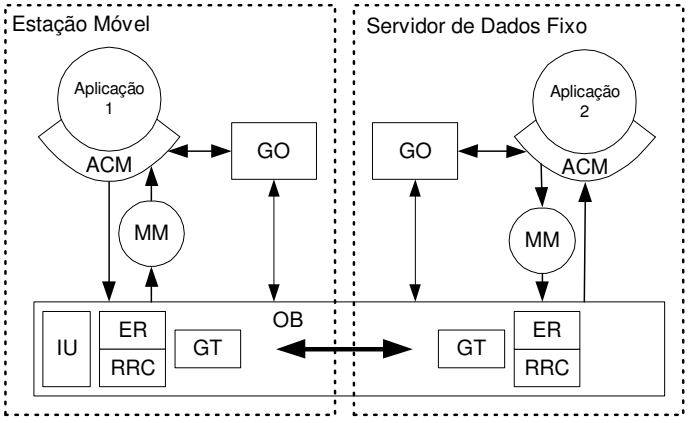

Figura 11 – Esquema de uma Arquitectura Baseada em Objectos.

O OB é constituído por:

- ***Escalonador da Rede* (ER).** Este componente, único em cada estação fixa ou móvel, é o responsável pela troca efectiva de mensagens. Para tal, gere uma lista com todos os pedidos que recebeu e de todas as respostas que foram enviadas por outro ER. É com base nesta lista que o escalonador da rede decide qual a próxima resposta ou pedido a ser enviado. Para tomar esta decisão, para além de ter em conta a prioridade, o tamanho e os parâmetros de QoS do pedido, também toma em consideração a conexão e as informações de contexto gerais.

- ***Request*/*Reply Cache* (RRC).** É aqui que se guardam os pedidos e as respostas na estação local para que se possa fazer uma entrega imediata às aplicações, quando se recebe um pedido idêntico a um já processado. Dado que as aplicações locais também comunicam entre si através do OB, então o RRC também pode ser usado para acelerar a comunicação local. Também se guarda aqui a informação adicional acerca dos contextos actuais, sendo o contexto usado para uma dada resposta guardado juntamente com a mensagem.

- ***Interface de Utilizador* (IU).** Este é um componente adicional que fornece ao utilizador, para além das funcionalidades oferecidas pelas aplicações, informação acerca do estado de todos os pedidos activos e respostas pendentes, permitindo um controlo interactivo dos parâmetros de troca.

- ***Gestor de Transferência* (GT).** É este componente que carrega as funções de baixo nível da rede, como abrir e fechar as conexões, assim como as funções de envio e de recepção. Para haver a possibilidade de avaliar os parâmetros de QoS é necessário que este GT faça a monitorização do tráfego nas diferentes conexões.

[11] *Message Handlers.*

3.7. Arquitectura a Três Níveis[12]

Os principais modelos da arquitectura a três níveis para ambientes móveis [Helal et al. 2001] têm como objectivos principais o facilitar da acessibilidade, disponibilidade e consistência dos dados em ambientes móveis. Este novo tipo de arquitectura suporta mecanismos automáticos de *hoarding* de dados provenientes de diversas fontes heterogéneas para mais do que uma estação móvel.

Nesta arquitectura, quando um cliente se pretende tornar móvel conecta-se à conta de mobilidade[13] e, de imediato, restaura-se o actual conjunto de trabalho do utilizador, sendo iniciado ainda o processo de incrementação incremental, de modo a que se seja possível a actualização dos conteúdos da estação móvel. Este processo de acumulação mantém-se activo enquanto a estação se encontrar conectada. Depois da desconexão, um dispositivo da estação móvel regista todos os itens de dados utilizados durante a desconexão, bem como todos os pedidos de dados que se não encontrem disponíveis. Quando a conexão se restabelece estes registos vão servir para fazer a actualização de dados, bem como a sincronização entre a estação móvel e a rede fixa. Deste modo, todas as actividades de acumulação de dados por parte das estações móveis e as tarefas de sincronização são feitas automaticamente. Do ponto de vista do utilizador, a única actividade exigida é que resolvam alguns conflitos durante o processo de sincronização que não foram resolvidos pelos algoritmos desta arquitectura.

O nível intermédio desta arquitectura corresponde a uma *Data Warehouse* que representa um repositório independente dos dados do utilizador, funcionando como origem dos dados. Esta hierarquia encontra-se esquematizada na Figura 12. Os três níveis, nos quais a arquitectura se baseia são:

- **Origens dos dados**. Nível que corresponde, geralmente, aos servidores da base de dados.

- **O conjunto de trabalho.** Que é o conjunto de todos os subconjuntos de dados que cada utilizador móvel possui.

- ***Caches* móveis.** São uma cópia do subconjunto de dados sobre a qual o utilizador trabalha.

[12] *Three-Tier Architecture.*
[13] *Mobile Account.*

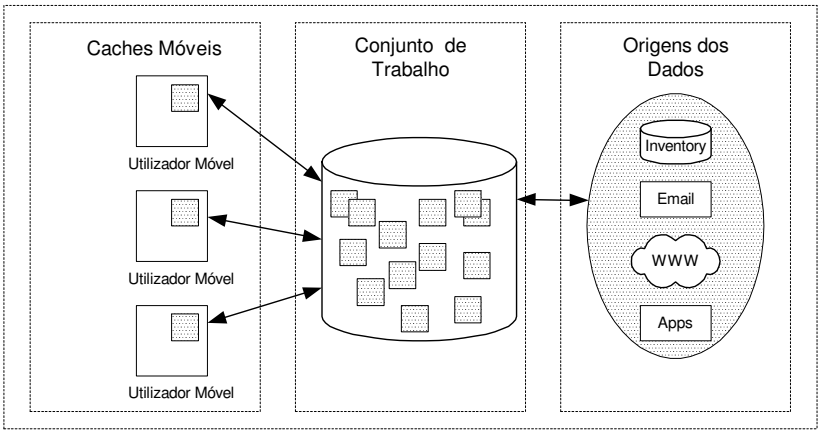

Figura 12 – Hierarquia de Dados com o Uso de Três Níveis.

Assim, como o próprio nome indica, a arquitectura aqui apresentada possui três níveis, nomeadamente: as estações fixas, as estações móveis e o nível intermédio. Este último elimina a sincronização manual, que por vezes é feita pelo utilizador, quer entre as estações móveis e a rede fixa, como também entre o utilizador e as diversas estações móveis que possui, caso o utilizador possua mais do que uma. No *hoarding* manual, o utilizador móvel deve decidir o conjunto de dados que pretende carregar na sua estação, fazendo a cópia destes ficheiros para a estação local, devendo sempre que necessário fazer a sincronização em sentido contrário, isto é, para a rede fixa. Esta camada intermédia pretende evitar esta sincronização manual fornecendo, para tal, uma sincronização automatizada quer a estação móvel se encontre conectada ou não.

Deste modo, com a introdução do nível intermédio, consegue-se fazer a separação entre as estações móveis e as origens dos dados, protegendo-os de eventuais alterações. Quando um utilizador se encontra conectado, o *Data Warehouse* acumula o seu conjunto de dados de trabalho. Por outro lado, durante as desconexões são agrupadas todas essas alterações que afectem os utilizadores que se encontrem desconectadas. Além disto, quando uma estação móvel restabelecer a conexão vai ser esta camada a responsável pela sincronização dos dados.

Capítulo 4

Replicação de Dados

Os processos de replicação permitem que alguns itens de dados sejam guardados redundantemente em algumas estações do sistema. Basicamente, estes permitem a existência de um conjunto de réplicas distribuídas por diferentes locais, sendo possível o acesso a qualquer uma dessas réplicas, desde que estas estejam disponíveis. Os SBDDs recorrem ao uso de replicação de dados para aumentar a sua capacidade, confiança, disponibilidade e desempenho, uma vez que permitem o acesso aos itens de dados mesmo quando algumas das suas réplicas não se encontram disponíveis. Para além disto, como existem diversas estações a funcionar como servidores de dados, os clientes podem escolher a réplica, à qual pretendem aceder, no local que mais lhes convier. Desta forma, de modo a reduzir os custos e a latência de comunicação, as estações podem ter acesso às réplicas que possuem localmente ou àquelas que se encontram nas estações vizinhas. Apesar das inúmeras vantagens que a replicação traz aos sistemas onde é implementada, possui também algumas desvantagens. Uma destas desvantagens está relacionada com o número de *overheads* provocado pelo grande fluxo de mensagens trocado entre as estações, de forma a manter a consistência dos conteúdos de todas as réplicas.

Nos SBDMs, a replicação de dados também é uma das técnicas mais usadas para melhorar o seu desempenho, até porque aqui os custos de comunicação são superiores aos dos sistemas distribuídos. Pode-se, deste modo, concluir que o uso da replicação de dados em ambientes móveis traz também inúmeras vantagens, podendo mesmo ser mais significativas neste caso do que nos SBDDs. De referir, uma dessas vantagens: a garantia de uma maior autonomia às estações móveis, ao nível da execução de algumas tarefas, durante os períodos de desconexão.

Um dos problemas que se coloca aquando da replicação de dados é saber quais os dados que serão necessários ao utilizador durante as desconexões, para a realização de todas ou de quase todas as suas operações. Todavia, nem sempre é possível saber quais são esses dados. Sempre que surjam operações que necessitem de dados que não se encontrem disponíveis na estação móvel, devem ser colocadas numa fila de espera para que sejam executadas assim que a estação móvel restabeleça a conexão com o restante sistema. Temos, assim, um pequeno dilema na construção de um esquema de replicação. Quantos mais dados replicados a estação móvel possuir mais operações conseguirá executar durante os períodos de desconexão. Por outro lado, as estações móveis, como já foi referido, possuem uma capacidade de armazenamento limitada, devendo por este motivo ser economizada. Além disto, com o uso da replicação de dados, as estações podem aceder a diferentes réplicas, o que faz com que possam surgir algumas inconsistências, obrigando ao uso de políticas de gestão de consistência. Quantas mais réplicas existirem mais difícil será manter a sua consistência, bem como fazer a sua reintegração no sistema sem que sucedam muitas operações interrompidas. Facilmente se conclui que, se deve encontrar um ponto intermédio nesta problemática que satisfaça as necessidades de cada utilizador, dando-lhes alguma autonomia, mas sempre tendo em conta que nunca se deve perder a qualidade e confiança do sistema.

A mobilidade, característica de algumas estações dos SBDMs, também causa algum impacto na replicação de dados, porque quando se pretende actualizar uma réplica, a estação móvel em que está armazenada pode já não se encontrar no mesmo local, obrigando desta forma à execução de um processo específico para a determinação da sua nova localização. Além disto, essa réplica pode encontrar-se quer na estação móvel, quer na ESM da célula actual, ou mesmo ainda na ESM das células anteriores [Badrinath & Imiellinski 1992].

Um sistema que use algum tipo de replicação estará correcto se assegurar a seriação de todas as suas réplicas. Nestes sistemas devem existir protocolos de controlo de réplicas, responsáveis pela sua sincronização. Em termos gerais, as técnicas de replicação para ambientes móveis devem [Heuer & Lubinski 2000]:

- Fornecer grande disponibilidade ao nível dos dados nas estações móveis, para reduzir os custos de comunicação e manter um nível de consistência aceitável.

- Usar informação contextual acerca do utilizador, dos atributos de dados e das aplicações, de modo a garantir a consistência dos dados com custos mais reduzidos.

- Usar informação de contexto para que seja possível realizar uma actualização de dados transparente a partir do servidor.

Para além disto, um esquema de replicação considerado como óptimo deve ainda possuir as seguintes características [Pitoura & Samaras 1998]:

- **Capacidade e escalabilidade**, tentando em simultâneo evitar a instabilidade.

- **Mobilidade**, para que as estações móveis, conectadas ou não, consigam ler e escrever os itens de dados mesmo quando se encontrem em movimento.

- **Seriação**[14], ou seja, fornecer um plano de execução seriável de transacções sobre uma única cópia (*single-copy serializable transaction execution*).

- **Convergência**, tentando que todas as réplicas convirjam para um mesmo valor ou então para valores aproximados.

[Heuer & Lubinski 2000] consideram ainda que as técnicas de replicação devem possuir três objectivos fundamentais: disponibilidade, consistência e economia de custos (custos mínimos). Estes autores demonstram ainda a impossibilidade de se alcançar estes três objectivos em simultâneo, pois a garantia de um destes objectivos obriga a que se tenha de relaxar pelo menos um dos outros dois. Se se considerar o caso em que a prioridade é obter dados consistentes, conclui-se que existem duas soluções possíveis. Na primeira assiste-se a uma redução do número de dados replicados, diminuindo-se desta forma a sua disponibilidade, mas, mesmo assim, conseguem-se obter os custos perto do valor pretendido. Na segunda solução consegue-se que a disponibilidade de dados se mantenha. No entanto os custos de comunicação vão aumentar significativamente. O mesmo se passa quando o objectivo principal é a disponibilidade de dados ou a redução dos custos de comunicação.

[14] *Seriability.*

Uma outra questão que se coloca, quando se fala de replicação, é a localização das diversas réplicas. [Badrinath & Imiellinski 1992] apresentam alguns dos possíveis cenários para a localização de tais réplicas em ambientes móveis:

- O servidor replica alguns dos itens de dados na estação móvel, actualizando essas réplicas sempre que se execute uma operação de escrita. De referir, que esta tarefa requer a localização das diversas réplicas de um item de dados. Ao contrário das operações de escrita, as operações de leitura podem processar-se sobre as réplicas locais.

- A réplica reside na ESM da célula onde a estação móvel se encontra, sendo a leitura da réplica efectuada a partir do seu servidor local. Aqui, tanto as operações de leitura como as de escrita se efectuam sobre as réplicas estáticas dos dados.

- A réplica pode encontrar-se numa ESM que não seja a actual. No entanto, as estações móveis devem ler e actualizar estas réplicas a partir da ESM actual.

Algumas das soluções de replicação para os sistemas distribuídos assumem que a infra-estrutura do sistema é estática e que as ligações e a localização das estações são fixas. Contudo, em ambientes móveis, se o modelo de replicação considerar que as ligações são estáticas, pode perder-se a mobilidade em vez de a ganhar, daí que o sistema de replicação deva possuir capacidades para se adaptar a tais alterações ambientais. Para tal, [Ratner et al. 1996] e [Ratner et al. 2001] defendem que um modelo de replicação eficiente para SBDMs deve possuir ou permitir:

- **Comunicação entre quaisquer estações.** Dada a grande mobilidade das estações, não se consegue prever quais as estações que se encontram numa determinada célula, num dado instante. Para além disto, é mais barato comunicar com as estações locais do que com as estações remotas. Assim, um sistema de replicação deve permitir que as estações comuniquem e sincronizem os seus dados com qualquer outra estação vizinha e não só com um conjunto específico de estações. Isto é, deve suportar a comunicação entre quaisquer duas estações.

- **Grandes factores de replicação.** É de esperar que a magnitude de replicação nos sistemas móveis seja superior à dos sistemas distribuídos, dado que nestes últimos existe normalmente uma infra-estrutura estática, conhecida, que permite uma replicação

mais distribuída. Nos ambientes móveis tal facto não acontece, o que faz prever um aumento significativo do número de réplicas. Há ainda a possibilidade de nos ambientes móveis cada utilizador possuir mais do que uma estação, sendo assim possível que o número de réplicas existentes aumente consideravelmente.

- **Controlos de replicação detalhados.** Todos os esquemas de replicação devem fornecer mecanismos de controlo de réplicas. Porém, nos SBDMs, estes mecanismos são mais complexos, já que algumas das réplicas podem não se encontrar disponíveis ou não se conhecer a sua localização no momento da sincronização. Os mecanismos de controlo de replicação devem, também, ter em conta as capacidades limitadas de armazenamento das estações móveis, para que os seus utilizadores tirem um maior partido da replicação de dados.

[Ratner et al. 1996] e [Ratner et al. 2001] defendem que as acções de antecipação de movimento[15] não são viáveis em ambientes móveis, porque se por um lado é útil que a estação móvel informe o restante sistema que se vai deslocar, sendo efectuados uma série de procedimentos para poder operar correctamente durante essa deslocação, por outro lado poderão ocorrer situações imprevistas, durante a deslocação, que dificultarão a acção do utilizador. Assim, a criação de políticas para a deslocação de uma estação restringe a sua acção, dado que não se autoriza o utilizador a alterar a sua rota pré-definida. Além disto, nem sempre se consegue prever a deslocação ou a desconexão de uma dada estação.

4.1. Protocolos de Replicação nos SBDDs

Em [Gray et al. 1996] faz-se uma caracterização dos modelos de replicação dos sistemas distribuídos, usando, essencialmente, dois parâmetros: a propagação das actualizações e localização das actualizações (Figura 13).

[15] *Pré-Motion.*

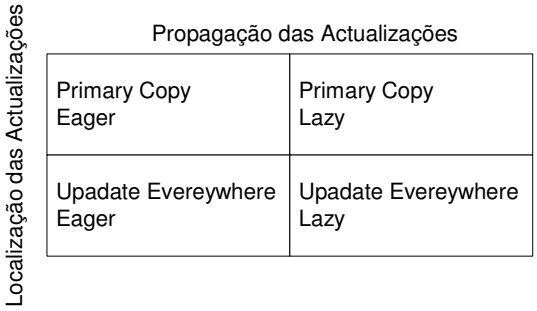

Figura 13 – Divisão dos Modelos de Replicação

Nos modelos de replicação de dados do tipo *Eager Replication*, todas as réplicas se encontram exactamente sincronizadas em todas as estações, pois a sua actualização é feita como parte integrante de uma transacção atómica. Com este tipo de replicação obtém-se uma execução seriável das operações, não causando problemas de concorrência. No entanto, o uso deste tipo de replicação aumenta o tempo de processamento das actualizações, podendo mesmo causar um aumento significativo no tempo de resposta das transacções. Isto porque aquando da actualização de uma réplica de um determinado item de dados, devem ser processadas um grande número de mensagens, de modo a que sejam actualizadas todas as réplicas desse item de dados. Pelas características dos ambientes móveis, já referidas diversas vezes ao longo desta dissertação, facilmente se conclui que este tipo de replicação não é o mais adequado para estes ambientes, uma vez que nem sempre as estações móveis se encontram disponíveis e se conhece a sua localização, não sendo, assim, possível a actualização das réplicas que aí se encontrem [Gray et al. 1996].

Em contraste, os modelos *Lazy Replication* actualizam apenas a réplica local, podendo mesmo acontecer que as diversas actualizações apenas se propaguem depois de se ter efectuado a confirmação da transacção – *commit*. Estes modelos de replicação estão mais sujeitos a inconsistências do que os do tipo *Eager Replication*, dado que nestes se realiza apenas uma actualização local, o que pode causar algumas divergências entre os valores das diversas réplicas aquando da sincronização [Gray et al. 1996]. Como, neste modelo, a actualização das réplicas é efectuada assincronamente, pode-se concluir que estes se podem aplicar aos SBDMs.

O desempenho de um sistema distribuído está directamente relacionado com a distribuição dos dados ao longo de todas as estações da rede. Isto porque, quando uma estação pretende ler ou escrever um determinado item de dados, estas operações são efectuadas, sempre que possível,

sobre as réplicas das estações vizinhas. Todavia, nem sempre existem cópias disponíveis tão próximo quanto seria desejável. Os modelos de replicação dinâmicos são desenhados tendo em conta estes aspectos. [Acharya & Zdonik 1993] [Ranganathan 2001] apresentam modelos de replicação de dados dinâmicos para sistemas distribuídos. Estes têm como principal objectivo aumentar o desempenho do sistema. Para tal, utiliza-se a alteração da localização dos dados de forma inteligente e de modo a optimizar o tráfego de mensagens na rede.

O algoritmo de replicação dinâmica apresentado em [Acharya & Zdonik 1993] é adaptativo e integrado. Este possui ainda uma natureza distribuída e reordena activamente o esquema de replicação, consoante os acessos aos itens de dados. Este algoritmo divide-se em duas fases, uma para satisfazer os pedidos de leitura e de escrita e outra para distribuir os dados ao longo da rede.

Um outro algoritmo de replicação adaptativo é o apresentado em [Wolfson et al. 1997]. Este algoritmo fornece um método de replicação de dados dinâmico capaz de alterar o esquema de replicação conforme as necessidades de leitura e de escrita das diversas estações, tentando obter um esquema de replicação óptimo.

4.2. Replicação Optimista

Os algoritmos de replicação tradicionais oferecem *single-copy serializability*, o que faz com que os utilizadores tenham a ilusão de que não existe um conjunto de réplicas, mas sim que existe apenas uma réplica que se encontra quase sempre disponível e actualizada. Esta transparência e a seriação única de todas as réplicas pode ser conseguida de diversas formas. Uma delas é através do uso de algoritmos pessimistas que proíbem o acesso a algumas das réplicas existentes.

O uso das técnicas de replicação pessimistas não acarretam grandes problemas quando aplicadas em redes locais, onde todas as estações se encontram sempre disponíveis e onde a sua localização é conhecida. Todavia, os SBDMs podem possuir redes de grandes de dimensões e estações que se podem encontrar temporariamente indisponíveis. Por este motivo, o uso de tais técnicas nos SBDMs pode causar longos períodos de espera na obtenção dos resultados e provocar algumas perdas na disponibilidade do sistema. Isto porque os algoritmos pessimistas obrigam a que todas as réplicas sejam actualizadas em simultâneo, o que pode causar algum tipo

de suspensão no processo de actualização, provocado, por exemplo, pela desconexão de uma das estações onde se encontram essas réplicas. Existem, ainda, outros factores que poderão afectar o uso dos esquemas pessimistas nos SBDMs, tais como [Saito & Saphiro 2002]:

- As redes de comunicação sem fios ainda são lentas e pouco confiáveis.
- O aumento do número de réplicas faz com que as operações de leitura sejam processadas mais rapidamente, porém o tempo de processamento das operações de escrita aumenta.
- Algumas actividades exigem a partilha de dados optimista. Por vezes é necessário que algumas tarefas sejam executadas concorrentemente.

Daí que se conclua que, os esquemas pessimistas não são os mais adequados para ambientes móveis. Várias técnicas surgiram como forma de ultrapassar ou evitar tais problemas, uma delas é a replicação optimista [Saito 2000] [Saito & Saphiro 2002]. Neste modelo permite-se que os utilizadores acedam a uma qualquer réplica, em qualquer altura. Para tal, baseia-se na presunção de que os conflitos de actualização são raros e de que todas as réplicas possuem um nível de consistência suficiente. Ao contrário dos algoritmos de replicação pessimistas, em que todas as réplicas são actualizadas simultaneamente, bloqueando-se possíveis leituras, nos algoritmos de replicação optimistas a actualização é efectuada posteriormente, quando, por exemplo, a conexão se restabelecer, sendo os eventuais conflitos resolvidos nessa altura.

Os principais objectivos das técnicas de replicação optimistas são fornecer um acesso aos dados suficientemente actualizado e minimizar as perturbações do utilizador causadas por conflitos. Para tal, estes esquemas tentam obter:

- **Uniformidade[16].** Aqui, exige-se que os conteúdos das várias réplicas convirjam para um mesmo valor. Para tal usam-se quatro mecanismos: propagação de actualização, *scheduling*, detecção e resolução de conflitos e *commitement* – estes mecanismos serão apresentados de forma mais completa no Capítulo 7.
- **Garantias da qualidade do conteúdo.** Para que se consigam obter estas garantias deve efectuar-se o controlo de quando e onde as actualizações se propagam, bem como quando os conteúdos das réplicas podem ser acedidos pelos utilizadores.

[16] Também designada de consistência eventual.

Existem sistemas que optam por não limitar este acesso aos dados, pois isso pode causar uma diminuição da disponibilidade do sistema.

- **Escalabilidade.** É fundamental que os modelos de replicação para SBDMs sejam escaláveis, uma vez que estes sistemas devem ser capazes de suportar um grande número de réplicas e de objectos.

Os algoritmos de replicação optimista apresentam dois possíveis cenários para a gestão das réplicas. Um, onde existe apenas um único mestre e, o outro, onde se permite a existência de múltiplos mestres. Os algoritmos com múltiplos mestres podem ainda ser subdivididos nos que transferem as operações[17] e nos que transferem o estado[18].

Os algoritmos com um único mestre nomeiam uma réplica mestre por cada item de dados, na qual devem ser geradas todas as suas actualizações. Aqui, a propagação das actualizações é geralmente simples, porque podem ser usados fluxos de dados num só sentido, do mestre para os escravos. Quando um mestre actualiza um item de dados coloca-lhe uma nova etiqueta temporal[19], devendo esta indicar a nova versão da réplica. As actualizações são iniciadas tanto pela réplica mestre como por cada um dos diversos escravos. Assim, por um lado permite-se que os escravos contactem frequentemente o mestre, verificando nesta altura, através das etiquetas temporais, se a sua versão coincide ou não com a versão da réplica mestre. Caso as versões não sejam coincidentes, o escravo pede à réplica mestre uma actualização da sua versão. Por outro lado, o mestre pode, sempre que actualiza um item de dados, tomar a iniciativa de enviar essas actualizações a todos os escravos. Com este tipo de algoritmos, a detecção e resolução de conflitos não constituem um problema, pois a estação que possui a cópia mestre detecta imediatamente todos os conflitos, tendo a possibilidade de atrasar ou abortar todas as actualizações concorrentes. Contudo, estes algoritmos, com um único mestre, apresentam algumas desvantagens, como o facto de possuírem um único ponto de falha, o que pode prejudicar o desempenho do sistema, já que quando ocorre uma falha na estação que possui a réplica mestre, o processamento de algumas operações de escrita pode ser suspenso. Um outro possível problema, que estes algoritmos podem gerar, é o grande congestionamento na zona da estação que possui a réplica mestre.

[17] *Operation Transfer.*
[18] *State Transfer.*
[19] *Timestamp.*

Por sua vez, nos algoritmos com múltiplos mestres, como o próprio nome indica, são nomeadas várias réplicas mestre por cada item de dados. Nestes algoritmos, os problemas de congestionamento existentes nos algoritmos com um único mestre, não são tão evidentes, porque existem diversas réplicas mestre distribuídas ao longo da rede. Por outro lado, se ocorrer uma falha numa estação que possui uma réplica mestre, não significa que se tenha de suspender o processamento de eventuais operações de escrita, visto que existem outras réplicas mestre no sistema capazes de gerir as actualizações que advenham de tais operações de escrita.

Um aspecto chave que separa os algoritmos de replicação optimistas dos pessimistas é a forma como se realizam os processos de actualização. Nos algoritmos pessimistas a actualização de todas as réplicas faz-se de uma só vez, pelo que é possível o bloqueio de algumas operações de leitura durante a aplicação dessas actualizações. Por sua vez, nos algoritmos optimistas as actualizações são propagadas em *background*, permitindo-se assim que qualquer réplica esteja, quase sempre, disponível para as operações de leitura. Deste modo, quando comparados com os algoritmos pessimistas, os algoritmos optimistas apresentam algumas vantagens importantes, nomeadamente:

- **Tolerância a falhas.** Os algoritmos optimistas foram desenhados para funcionar sobre ligações de rede lentas e de pouca confiança. Para além disso, a propagação das actualizações das réplicas é efectuada em *background* e os utilizadores podem ler ou actualizar um objecto enquanto uma das réplica existir.

- **Flexibilidade de rede.** A maioria dos algoritmos optimistas funciona sobre redes incompletas e intermitentes, permitindo que as actualizações sejam trocadas por quaisquer duas estações. Além disto, lidam ainda com as alterações da configuração da rede, que ocorrem frequentemente nos SBDMs.

- **Custo reduzido.** A implementação dos algoritmos optimistas comporta custos reduzidos, pois não necessita de grandes recursos de hardware na sua implementação e gestão.

- **Autonomia da estação.** Com estes algoritmos, as estações têm uma maior autonomia, pois existe uma menor coordenação ao longo das estações, o que lhes fornece o poder para efectuar algumas decisões locais. No entanto, estas decisões devem ser comunicadas, posteriormente, ao restante sistema.

- **Disponibilidade.** Estes algoritmos fornecem uma maior disponibilidade das réplicas, visto que não há necessidade de bloqueio destas durante o processamento das suas actualizações.

- **Escalabilidade.** Os algoritmos optimistas têm a capacidade para suportar um grande número de estações, porque a propagação das actualizações não envolve grandes esforços.

4.3. Replicação a Dois Níveis[20]

O modelo de replicação a dois níveis [Gray et al. 1996] tem por base os dois tipos de estações existentes nos SBDMs. Um dos níveis é representado pelas estações fixas e o outro pelas estações móveis. Das, já referidas, características das estações móveis e das fixas pode concluir-se que, as estações fixas possuem, geralmente, uma maior disponibilidade do que as móveis. Atendendo a este facto, este esquema de replicação a dois níveis propõe que a maioria das cópias mestre se encontre nas estações fixas, para que, assim, estas sejam mais facilmente acedidas por outras estações. Todavia, também se permite que as estações móveis contenham algumas cópias mestre, porém esta é uma solução que se tenta evitar, uma vez que durante os períodos de desconexão os valores destas cópias ficam inacessíveis.

Neste modelo de replicação, as estações móveis possuem um conjunto de cópias dos itens de dados, algumas das quais podem ser mestre. Quando as estações efectuam actualizações locais sobre cópias que não são mestre devem propor, em simultâneo, uma tentativa de actualização desses objectos a outras estações. Deste modo, cada estação móvel possui duas versões para cada item de dados: uma local e uma mestre. Os valores das versões locais devem ser enviados para a estação que possui a sua versão mestre, de modo a verificar a possibilidade de se tornarem permanentes ou não.

[20] *Two-Tier Replication.*

4.4. Protocolos de Controlo de Réplicas

Alguns dos protocolos de controlo de réplicas mais usados em SBDDs são [Faiz & Zaslavsky 1995]:

- Método de cópia primária.
- Método de *quorum consensus.*
- Método das cópias disponíveis.

No método de cópia primária, cada estação deve manter uma lista das estações com as quais comunica. A estação, que se encontra na posição mais baixa da lista de ordenação crescente, possui a réplica que se designa de cópia primária. Sempre que se actualiza um item de dados é necessário enviar-se um pedido de escrita para a estação que possui a cópia primária. Posteriormente essas alterações devem ser enviadas, assincronamente, para as restantes réplicas. Quando se efectua um pedido de leitura, este pode ser enviado para uma cópia primária. No entanto, é mais eficiente executá-lo localmente.

Existem diversos algoritmos de controlo de réplicas que são baseados no método da cópia primária, nomeadamente [Faiz & Zaslavsky 1995]:

- **Protocolo *As Soon As Possible*.** Este é executado de modo a que as operações de escrita sejam efectuadas na cópia primária, sendo para tal guardadas e acumuladas, para que, posteriormente, possam ser enviadas às restantes cópias.
- **Método *Quasi Copies*.** Neste método, o controlo de informação é efectuado, geralmente, numa única estação central, apesar de também ser possível usando várias estações. Aqui, definem-se as condições de coerência, que permitem alguma divergência entre os objectos. Estas condições podem estar relacionadas com o seu tempo, versão, ou valor. As alterações podem ser propagadas para a cópia primária através de um dos seguintes métodos: último minuto, imediatamente, mais cedo ou actualização atrasada.
- ***Differential File.*** Este é usado para guardar as alterações efectuadas na cópia primária. Aqui, o ficheiro diferencial é usado para a actualização das diversas cópias.

- ***Differential Refresh.*** Neste método são associadas etiquetas temporais com cada registo da tabela base.

- ***Copy Token.*** Neste esquema considera-se a existência de dois tipos de cópias: as lógicas e as físicas. As operações de escrita são executadas apenas sobre as lógicas e, de seguida, armazenadas até à altura da confirmação da operação – *commit*.

O método da cópia primária aqui descrito não pode ser directamente aplicado nos SBDMs. Todavia existem já algumas adaptações que funcionam sobre este tipo particular de sistemas. No método da cópia primária existe o problema de saber qual a melhor estação para se colocar a cópia primária. Se por um lado a cópia primária deve ficar na estação com maior processamento local, por outro também deve encontrar-se na estação com maior disponibilidade. Nos SBDMs têm-se as estações móveis que possuem um processamento maioritariamente local, mas que podem encontrar-se desconectadas por longos períodos de tempo. Nos SBDMs existem ainda as estações fixas que se encontram sempre, ou quase sempre, disponíveis, mas nem sempre possuem um grande processamento local. Existem várias soluções para este problema, uma delas é a de colocar a cópia primária na estação móvel e fazer um seu *backup* numa das estações fixas. Uma outra solução seria fazer exactamente o oposto, isto é, colocar a cópia primária na estação fixa e efectuar uma réplica na estação móvel. Atendendo às características de um SBDM, a segunda solução parece ser a mais viável, uma vez que é mais fácil as estações móveis conseguirem manter as suas réplicas actualizadas através das reconexões.

No método do *quorum consensus*, uma operação só pode ser executada se se obtiver a permissão de um grupo de nodos, chamado de grupo *quórum.* Ao conjunto destes grupos dá-se o nome de conjunto *quórum*. A constituição do grupo é determinada conforme os requerimentos do sistema. Como, geralmente, existem dois tipos de operações, leitura e escrita, também existem dois tipos de conjuntos *quorum*: um conjunto *quorum* de leitura e um conjunto *quorum* de escrita. Os grupos *quorum* devem satisfazer as seguintes condições [Faiz & Zaslavsky 1995]:

- Os grupos *quorum* de leitura e de escrita devem intersectar-se, de modo a assegurar que uma operação de leitura devolve o resultado da última operação de escrita e, que, quaisquer operações concorrentes são sincronizadas convenientemente.

- Os grupos *quorum* de escrita devem ter um membro em comum, assegurando que as operações de escrita não são executadas concorrentemente.

Um caso especial do método *quorum consensus* é o protocolo ROWA (*Read One, Write All*) onde o grupo de leitura consiste apenas num nodo, enquanto que o grupo de escrita engloba todos os nodos.

Também o método *quorum consensus* pode ser adaptado a ambientes móveis. Contudo, aqui existe um elevado *overhead* ligado às operações de escrita e de leitura, uma vez que o *quorum* necessita de uma ou mais réplicas móveis, tendo, assim, de se incluir os custos da sua localização. Deste modo, os algoritmos de replicação devem ser modificados de modo a que o *quorum* seja formado sem a intervenção das réplicas móveis, a menos que tal facto seja inevitável.

Por último, no método das cópias disponíveis, as actualizações são aplicadas às réplicas que se encontram nos nodos operacionais, ignorando-se os restantes nodos. A operação de leitura pode usar qualquer réplica disponível, o que pode gerar algumas inconsistências. O esquema de cópias disponíveis opera do seguinte modo: as operações de leitura podem ser direccionadas para qualquer nodo que possua o último valor dos dados, enquanto que as operações de escrita apenas se realizam quando existir, pelo menos, uma réplica para guardar as alterações. Aqui, uma transacção deve passar por dois processos de validação:

- **Validação de escrita.** A transacção deve certificar-se de que todas as réplicas, que não receberam as actualizações, continuam indisponíveis.

- **Validação de acesso.** A transacção deve certificar-se que todas as réplicas que leu ou escreveu continuam disponíveis.

O uso do método das cópias disponíveis em SBDMs implica custos de localização das réplicas móveis, já que todas as cópias, incluindo as móveis, são actualizadas no mesmo instante. Por este motivo, o método das cópias disponíveis torna-se menos adequado para os sistemas móveis.

Capítulo 5

Modelos de Transacções Móveis

Os sistemas de processamento de dados actuais são, na sua generalidade, suportados por transacções. Não faz, e nunca fez, sentido que qualquer que seja a realização de um dado conjunto de operações sobre uma base de dados possa conduzi-la para um estado de inconsistência. A utilização de sistemas transaccionais permite evitar a ocorrência de tais situações. Em sistemas mais antigos, nos quais não era possível utilizar transacções, ocorriam muitas vezes situações indesejáveis, porque esta ou aquela operação falhava na sequência do processamento que tinha sido previamente programado, enquanto que as restantes operações, integradas nessa sequência, eram realizadas com sucesso. Isto provocava frequentemente discrepâncias nos processos de actualização de dados, que, por mais simples que o fossem, levavam normalmente o sistema de dados em causa a um estado de inconsistência. Sabemos o que esta situação provoca em sistemas reais. Qualquer base de dados que atinja tal estado gera, quase de forma imediata, alguma desconfiança por parte dos utilizadores e torna a base de dados pouco credível. Sem dúvida alguma, uma situação a evitar.

Em termos simplistas, uma transacção é definida como uma unidade de computação consistente e segura. Estas unidades de computação (por vezes também designadas por unidades de trabalho)

garantem que as operações de manipulação de dados realizadas debaixo da sua alçada conduzem a base de dados de um estado consistente a um outro estado também consistente [Özsu & Valduriez 1999].

Uma transacção é essencialmente uma programa que manipula recursos contidos em bases de dados ou em ficheiros partilhados, consistindo normalmente num conjunto de operações de escrita e de leitura, terminando com uma operação de *commit*, de modo a que os seus efeitos se tornem permanentes ou então terminando com uma operação de abortar de forma a desfazer os efeitos da transacção (Figura 14) [Dunham et al. 1997].

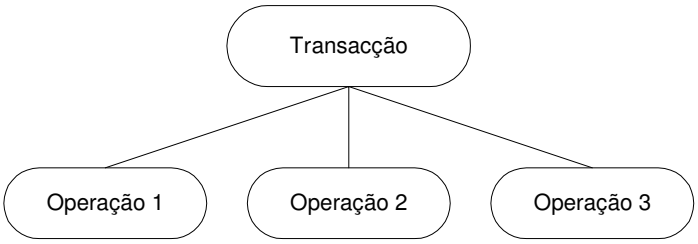

Figura 14 – Decomposição de uma Transacção.

Nos SBDDs, este processo é semelhante ao que acontece no processamento de uma *query* independentemente da sua complexidade, com a excepção de que as transacções garantam, tal como referido, que se a base de dados se encontrava num estado consistente antes da execução de uma transacção, então também se encontra depois da sua execução [Özsu & Valduriez 1999].

Ao contrário das bases de dados centralizadas, nos sistemas distribuídos existem dois tipos de transacções: locais e globais. As transacções locais podem ser definidas como aquelas que são submetidas e executadas, tal como a própria categorização indica, nos SGBDDs locais, enquanto que as transacções globais correspondem aquelas que são submetidas a todo o sistema através da sua interface global, podendo requerer a sua execução em diversas estações [Segun et al. 2001]. Nos SBDDs uma transacção pode ser executada de forma concorrente em vários processadores e sobre vários conjuntos de dados. Contudo, a sua execução é coordenada globalmente pelo sistema através dos seus mecanismos de controlo de concorrência e replicação, e de garantia de um *commit* atómico global [Madria 1998].

Num SMBD, as transacções locais e globais são executadas independentemente, não existindo uma distinção lógica entre estes dois tipos de transacções. Além disto, nestes sistemas, as transacções locais e globais geram três tipos de histórias [Segun et al. 2001]:

- **História local.** Corresponde ao conjunto de transacções locais e globais executadas numa determinada estação.

- **História de subtransacções globais.** É o subconjunto da história local de uma estação, contendo apenas as operações das transacções globais nessa estação.

- **História Global.** Corresponde à união de todas as histórias de subtransacções globais.

Nos SBDMs, as estações móveis podem encontrar-se desconectadas das restantes estações por longos períodos de tempo, podendo mesmo acontecer que a desconexão tenha ocorrido antes do final da execução da transacção. Visto que as estações móveis se deslocam, também pode acontecer que na sua deslocação passem para uma nova célula, ficando sob o controlo de uma nova ESM. Contudo, durante essa transição, a execução das suas transacções não deve ser interrompida. Assim, e dadas as características destes ambientes, as transacções vão ser ligeiramente diferentes das dos sistemas tradicionais. Os SBDMs devem então acolher a execução deste novo tipo de transacções, designadas, em termos gerais, por transacções móveis. Alguns dos aspectos a ter em conta neste novo tipo de transacções são [Madria 1998] [Segun et al. 2001]:

- Como as estações móveis se podem movimentar, mesmo durante a execução das transacções, pode acontecer que uma transacção comece a ser executada numa célula e termine noutra.

- Pode ser necessário partir uma transacção em conjuntos de operações mais pequenas. Alguns destes conjuntos serão executados nas estações fixas e outros nas estações móveis.

- Os estados das transacções, dos objectos acedidos e da localização da informação são transferidos conforme a estação se vai movendo de uma célula para outra.

- Seguindo o movimento da estação móvel, uma transacção móvel interactiva pode ser iniciada numa estação e concluída noutra. Contudo, é importante que as alterações feitas na primeira estação sejam visíveis a partir da nova localização.

- As desconexões frequentes e a mobilidade das estações podem resultar numa partilha dos estados e resultados parciais das transacções móveis, violando os princípios de atomicidade e de isolamento, normalmente existente nas transacções dos SBDDs.

- As estações móveis devem transferir o processamento das transacções para as estações fixas, assim que deixe de ser necessária a interacção com o utilizador.

- A execução das transacções móveis pode ser transferida para as estações fixas, quer por se prever uma desconexão, quer para se prevenir que a transacção seja abortada.

- As transacções móveis tendem a ter uma longa duração devido à mobilidade dos consumidores e/ou fornecedores de dados, bem como às desconexões frequentes.

- Se se estiverem a utilizar esquemas de *locking*, pode haver um aumento significativo de *deadlocks* e de transacções abortadas. Por outro lado, se se estiverem a usar esquemas optimistas, pode haver um aumento de conflitos e de recomeço de transacções. Deste modo, deve-se usar algum tipo de controlo de concorrência para minimizar o bloqueio ou o abortar da execução de transacções.

- Os modelos de transacções devem suportar e lidar com questões de concorrência, recuperação, desconexão e consistência mútua dos itens de dados replicados.

Nos SBDMs, uma estação móvel pode executar transacções móveis através do envio das operações da transacção para os diferentes servidores distribuídos por diferentes locais ou, então, executá-las localmente até ao momento em que haja algum tipo de conexão. Assim, as transacções móveis são executadas de forma sequencial e não concorrentemente – como nos sistemas distribuídos – ao longo de diversas estações do sistema, podendo acontecer que sejam executadas sobre diferentes conjuntos de dados, dependendo do movimento da estação. Daqui, pode-se também concluir, que o sistema não consegue coordenar, totalmente, a execução das transacções móveis, dado que a deslocação das estações móveis influenciam a sua execução [Antila et al. 2001] [Madria 1998].

Como já foi referido, um dos resultados da mobilidade é a distribuição da execução das operações da transacção por diferentes servidores. Porém, esta distribuição faz com seja necessária a transmissão de mensagens por parte dos servidores, de modo a que seja possível fazer algum tipo de coordenação sobre essas operações. Assim, as tradicionais características dos SBDDs como concorrência, atomicidade e recuperação devem ser revistas para que consigam capturar o movimento dos dados.

Um dos maiores obstáculos dos modelos de transacções móveis são as desconexões frequentes que, juntamente, com a natureza de longa duração das transacções, fazem com que possa existir um grande número de transacções a executar em simultâneo. Tais factores fazem, ainda, com que a confiança (e até mesmo a utilidade) seja um requerimento primário para o processamento de transacções móveis. Deste modo, os novos modelos de transacções devem possuir a capacidade de minimizar o número de abortos de transacções provocados pelas desconexões. Complementarmente, devem ainda permitir a construção de blocos de execução de transacção, tanto nas estações fixas como nas móveis, de forma a reduzir os custos de comunicação e a aumentar a concorrência [Antila et al. 2001].

Nos modelos de transacções dos SBDMs, também os protocolos de término devem de ser revistos, pois nestes ambientes a estação que iniciou a transacção pode ser diferente da estação que a vai terminar. Além disto, os protocolos de endereçamento que permitem fazer a realocação das transacções podem não ser capazes de garantir a entrega na estação que iniciou a transacção, isto porque a estação pode ter sido, por exemplo danificada, roubada ou, simplesmente, estar no momento fora da área de comunicação [Dunham et al. 1997].

Um aspecto fundamental dos modelos de transacções é a garantia da correcção das transacções. Para tal, nos sistemas distribuídos, usam-se as propriedades ACID:

- **Atomicidade**. Esta propriedade garante que numa transacção ou se guardam todos os resultados das suas operações ou nenhum. Estabelece ainda que as alterações numa transacção são atómicas.

- **Consistência**. Uma transacção produz resultados consistentes garantindo a integridade da base de dados.

- **Isolamento**. Apesar da execução concorrente das transacções, cada transacção deve ser executada sem afectar qualquer outra transacção. Deste modo, os resultados

intermédios de uma transacção devem ser escondidos da execução de outras transacções concorrentes.

- **Durabilidade**. Quando se conclui uma transacção, os seus efeitos devem ser tornados permanentes, podendo todas as outras transacções ter acesso a estes efeitos.

São vários os factores que pode levar uma transacção a abortar nos SBDDs. De destacar os seguintes:

- O sistema recebe um valor de entrada que pode violar os requisitos de consistência da base de dados.
- A transacção gera um estado que o sistema classifica como *deadlock* ou *time-out*.
- Ocorre uma falha no sistema fazendo com que uma transacção activa seja desfeita durante a recuperação.

Contudo, os ambientes móveis impõem novas restrições e dificuldades para manter as propriedades ACID. Estas dificuldades advêm, principalmente, do facto de se pretender que as estações móveis sejam autónomas, tendo controlo completo sobre os dados armazenados nas suas bases de dados locais. Além disto, neste tipo de ambientes, o processamento de transacções pode gerar grandes atrasos, abortos frequentes ou desnecessários, inconsistências nos dados ou esconder situações de *deadlock*.

Uma transacção pode deslocar-se quer por parte do comportamento da estação, quer por parte do acesso aos dados. Esta deslocação pode ser documentada por um mecanismo de controlo de transacções. Quando a transacção passa para uma nova célula, o controlo das transacções pode passar para a ESM da nova célula ou continuar na ESM original. Contudo se isto acontecer vai existir um grande *overhead* de mensagens entre as duas células, daí que se opte, na generalidade dos casos pela transferência do controlo da transacção para a ESM da nova célula [Dunham et al. 1997].

Outro aspecto desejável nos modelos de transacções é a sua sincronização. Em ambientes multi-utilizador é normal existirem diferentes transacções a aceder em simultâneo aos mesmos itens de dados, o que pode causar interferências e inconsistências indesejáveis. Este tipo de situações requer que existam mecanismos capazes de gerirem estas situações. Um exemplo simples de

inconsistência de dados provocado por este tipo de interferências é a perda de actualização. Assim suponhamos que duas transacções T_1 e T_2 lêem um item de dados *x*, seguidos de uma escrita de T_1 em *x*, e depois de uma escrita de T_2, então, se não houver sincronização, perde-se a actualização efectuada pela transacção T_1. Outro problema que pode surgir quando não existe sincronização é a leitura inconsistente, que ocorre quando uma transacção lê um item de dados antes de uma transacção o actualizar e lê outros itens de dados que essa mesma transacção já actualizou. Este cenário ocorre quando apenas algumas actualizações são visíveis, causando as designadas leituras sujas[21] ou *inconsistent retrieval* [Segun et al. 2001].

A solução, aparentemente, mais simples para a resolução destes problemas de interferência é a de proibir que as transacções executem simultaneamente, o que provocaria um baixo *throughput* e uma utilização pobre de recursos, especialmente no caso em que as transacções raramente acedem a dados partilhados. Outra solução, um pouco mais viável, é a de permitir que as transacções se executem concorrentemente, mas que sejam controladas por algoritmos de sincronização para que o resultado final seja equivalente à execução em série de todas as transacções, isto é, que as transacções possam ser seriadas. A seriação de transacções (*seriability*) é bastante usada nos modelos de transacções como critério de correcção para assegurar controlo de concorrência. Pode-se também fazer a seriação das operações conflituosas das transacções. Por sua vez, diz-se que duas operações são conflituosas se acedem a um mesmo item de dados e se uma dessas operações é de escrita. [Segun et al. 2001] dividem os conflitos das operações em duas categorias:

- **Directos,** quando uma ou mais operações das transacções são conflituosas.
- **Indirectos,** quando se usam valores de transacções que se encontram com conflitos.

Uma técnica pessimista para evitar os conflitos de transacções é o *locking*, no qual se assume que todas as transacções interferem umas com as outras e, como consequência, se bloqueiam as execuções de outras transacções que queiram aceder aos mesmos itens de dados. Existem ainda alguns esquemas alternativos, tais como, a ordenação de etiquetas temporais, teste de grafos de seriação e esquemas de controlo de concorrência.

Outro problema que os modelos de transacções móveis devem ter em conta é o *deadlock*, que acontece quando ocorrem conflitos de recursos. Um *deadlock* é geralmente provocado quando

[21] *Dirty Reads.*

ocorre uma espera cíclica de recursos ao longo das transacções, devida essencialmente à utilização de algum tipo de *locking* ou controlo de concorrência. Um modo optimista de lidar com os *deadlocks* é o de partir um *deadlock* quando ele ocorre, isto é, quando se detecta um ciclo, algumas das transacções activas envolvidas num *deadlock* devem ser abortadas para que possa quebrar esse *deadlock*.

Num ambiente distribuído diz-se que uma transacção pode ser guardada se e só se for guardada em todas as estações onde a subtransacção foi executada. Tradicionalmente, numa base de dados distribuída é usado um protocolo como o *two-phase commit* para assegurar o *commitment* das (sub)transacções em todas as estações. Resumidamente, este protocolo envolve, numa primeira fase, um coordenador global que envia uma mensagem prepare-para-guardar[22] para todos os SGBDs intervenientes na transacção. Por sua vez, cada participante responde com uma mensagem de READY se estiver pronto para guardar ou ABORT se ainda não estiver. Numa segunda fase, o coordenador envia uma mensagem COMMIT ou ABORT como decisão global, usando para tal as respostas obtidas na primeira fase.

Nos SBDMs, em geral, é impossível executar *commits* atómicos sem que seja violada a autonomia local. Tal sucede, porque nestes sistemas podem existir dependências entre subtransacções da mesma transacção global, sendo impossível fazer o *commit* atómico se os componentes do sistema forem autónomos e se existirem dependências funcionais cíclicas ao longo dessas subtransacções. As dependências de *commit* e aborto são os dois tipos de dependências que podem existir entre as subtransacções, podendo fazer com que o *commit* seja mais difícil nos SBDMs. As dependências que podem ocorrer entre subtransacções de uma transacção global são:

- **Dependência de *commit***. Diz-se que uma transacção T_1 tem uma dependência de *commit* de outra transacção T_2, quando não se pode fazer o *commit* de T_1 sem T_2 fazer o *commit* ou o aborto.

- **Dependência de aborto**. Uma transacção T_1 diz-se dependente de aborto de uma transacção T_2, se o aborto de T_2 implica o aborto de T_1.

O modelo de transacções móveis deve mostrar os movimentos e a localização das transacções. Dada a grande mobilidade deste tipo de transacções, deve-se ainda guardar o seu estado,

[22] *Prepare-to-Commit.*

devendo ser transparente o armazenamento do estado e o progresso de execução de uma transacção para o utilizador. Estes novos modelos não devem ser puramente ACID, pois, se por um lado, existiria um sistema consistente, por outro lado, devido aos diversos abortos, existiria apenas uma pequena fracção de trabalho útil. No entanto, o novo modelo deve garantir, de alguma forma, a consistência do sistema. No fundo um modelo que suporte transacções móveis deve encontrar-se entre um modelo ACID e um modelo sem restrições [Dunham et al. 1997].

Em conclusão, um modelo de transacções móveis deve ser capaz de [Antila et al. 2001] [Dunham et al. 1997]:

- Suportar as desconexões das estações móveis, tendo em conta que estas poderão manter-se desconectadas durante longos períodos de tempo.
- Suportar a mobilidade de uma estação ao longo da execução da transacção.
- Capturar o movimento das transacções móveis, assim como o acesso aos dados.
- Controlar a deslocação da transacção quando a estação móvel se movimenta.
- Fornecer flexibilidade em termos de atomicidade.
- Suportar transacções com grandes durações.
- Fornecer autonomia local para permitir que as transacções sejam processadas e guardadas na estação móvel quando há desconexões temporárias.
- Suportar comunicações e computações sem fios.
- Suportar e manusear a concorrência, o restabelecimento de uma desconexão e a consistência mútua dos objectos de dados replicados.

Existem já modelos de transacções móveis que tentam ultrapassar todas as limitações existentes nos ambientes móveis, bem como fornecem todos os requisitos referidos anteriormente. Cada um dos modelos tem em consideração todas as características deste tipo de transacções, todavia cada modelo dá mais ênfase a características como: a semântica das transacções, a natureza saltitante das transacções, a concorrência, objectos dinâmicos, etc. Nas secções seguintes apresentam-se alguns destes modelos.

5.1. Transacções *Open-Nested*

Em ambientes móveis pretende-se que a execução de uma transacção não seja interrompida durante os períodos de desconexão. Isto é, a parte da transacção que está executar na ESM deve continuar a executar durante a desconexão ou deslocação da estação móvel. O modelo de transacções *Open-Nested* [Chrysanthis 1993] fornece um modelo de transacções móveis que tenta superar algumas das limitações existentes nos ambientes móveis, introduzindo, para tal, o conceito de dois novos tipos de transacções: co-transacções[23] e transacções de relatório[24]. Este autor define uma transacção móvel como um conjunto de transacções (designados por componentes) relativamente independentes que podem intervir de qualquer modo com outras transacções móveis. Um componente de uma transacção pode ser também decomposto noutros componentes de transacção. Neste modelo, os componentes de transacção podem efectuar o *commit* sem ter de esperar pelo *commit* dos restantes componentes ou mesmo pelo *commit* da transacção a que pertence. Isto é, os componentes podem decidir fazer o *commit* ou o abortar da transacção unilateralmente. Contudo, se uma transacção aborta, todos os componentes de transacção que não fizeram o *commit* deverão ser também abortados.

Baseando-se nas operações de *commit* e de abortar dos componentes de transacções podem-se dividir as transacções móveis em quatro tipos distintos:

- **Transacções atómicas**. Estas encontram-se associadas a eventos significativos (começo, *commit* ou aborto), tendo o *commit* e o aborto as propriedades tradicionais. As transacções compensáveis e de compensação são transacções atómicas com uma estrutura induzida de dependências inter-transacções. Um componente compensável de uma transacção móvel é um seu componente que pode fazer o *commit* das suas operações mesmo antes da transacção fazer o *commit*. No entanto se a transacção abortar, as transacções de compensação dos componentes guardados devem fazer o *commit*. Uma transacção de compensação deve observar um estado consistente com os efeitos dos seus componentes correspondentes e assim, as transacções de compensação devem executar (e fazer o *commit*) na ordem inversa do *commitment* dos seus correspondentes componentes.

- **Transacções não-compensáveis**. São aqueles que podem fazer *commit* em qualquer altura, mas como não são compensáveis, não têm assim permissão para guardar os

[23] *Co-transactions.*

seus efeitos nos objectos quando fazem o *commit*. Por este motivo, as transacções não compensáveis devem ser estruturadas como subtransacções, que na altura do *commit* delegam todas as operações que invocaram para a transacção móvel. Aqui, delegar corresponde à capacidade de uma transacção em dar a outra transacção a responsabilidade de fazer o *commit* ou o aborto de uma transacção.

- **Transacções relatório**. Uma transacção móvel pode possuir componentes relatório que partilham os seus resultados parciais com a transacção. Um componente de relatório informa a transacção móvel de alguns dos seus resultados, em qualquer altura, mesmo durante a sua execução. O facto de um componente de relatório delegar ou não todas as operações que ainda não foram relatadas à transacção móvel, depende apenas destas estarem ou não relacionadas com uma transacção de compensação.

- **Co-transacções**. Estas são transacções de relatório que se comportam como co-rotinas, nas quais o controlo é passado de uma transacção para outra na altura da partilha dos resultados parciais. Assim, ao contrário das transacções não compensáveis, as co-transacções retêm o seu estado ao longo das execuções. Em oposição às transacções de relatório, as co-transacções não podem executar concorrentemente.

Os componentes compensáveis e não compensáveis podem ainda ser considerados como transacções vitais, onde a transacção móvel pode efectuar o *commit* apenas se os seus componentes vitais o efectuarem. Se uma transacção vital aborta, então a transacção móvel também deve abortar. Além disso, uma transacção, mesmo quando um dos seus componentes não vitais aborta, tem de esperar por todos os seus componentes não vitais para poder abortar ou fazer o *commit*.

Ao contrário do que acontece com os componentes de transacção, as transacções relatório e as co-transacções encontram-se fortemente interrelacionadas, podendo ser executadas tanto de um modo paralelo como de um modo *step-wise*, trocando potencialmente um grande número de resultados. Por esta razão, as transacções relatório e as co-transacções podem realocar a sua execução de uma estação para outra, sendo deste modo os custos de comunicação e de manutenção minimizados.

[24] *Reporting Transactions.*

Uma característica comum dos modelos de transacções relatório e co-transacções é o facto de elas possuírem delegação entre as transacções. Assumindo, que as histórias são seriáveis, a delegação afecta a ordem de seriação das transacções. Assim neste modelo um dos critérios de correcção usados é a seriação de transacções.

A falha de atomicidade é uma propriedade das transacções que assegura que todas as operações, ou nenhuma, da transacção são executadas. Nestas transacções não há necessidade de se fazer a reavaliação da definição de falha de atomicidade. A falha de atomicidade não requer a invocação de uma transacção para ser a transacção a guardar ou a abortar a operação. É de notar que os efeitos de uma operação sobre um objecto não sejam permanentes na altura da sua execução. Estes devem ser explicitamente *commited* ou abortados. Assim, a falha de atomicidade dá a possibilidade de todas as operações invocadas por uma transacção, e não delegadas por outra transacção, de serem guardadas (ou abortadas).

5.2. *Clustering*

O modelo de transacções móveis apresentado nesta secção baseia-se na criação de *clusters* [Pitoura & Bhargava 1994A] e tem como principal objectivo manter a consistência da base de dados. Neste modelo, define-se, formalmente, uma base de dados móvel como um conjunto finito de itens de dados. Assim, pode-se partir uma base de dados móvel em vários *clusters* Cl_i, em que cada Cl_i é um conjunto de itens de dados.

Para criar tais *clusters* deve-se ter em conta a localização física dos dados. Os dados localizados em nodos vizinhos ou com ligações fortes devem pertencer ao mesmo *cluster*, enquanto que dados desconectados devem pertencer a *clusters* diferentes. No entanto, a configuração de um *cluster* deve ser dinâmica, podendo os dados desconectados passar de uns *clusters* para os outros. Na criação de *clusters* deve ainda ter-se em conta a semântica dos dados, sendo, também, a maioria das características baseadas na localização dos dados. Esta localização representa o endereço da estação móvel, bem como os dados que possuem as várias réplicas e as suas diferentes versões nas diferentes estações. Os *clusters* devem ser fornecidos explicitamente, baseados nos pedidos dos utilizadores. Por último, para a criação dos *clusters*, deve ainda ter-se em conta alguma informação guardada nos ficheiros do utilizador.

O estado da base de dados (ou do *cluster*) é definido como uma função que mapeia todos os itens de dados num valor do seu domínio. Os itens de dados estão relacionados com um número de restrições de integridade, que expressam as condições que os dados devem satisfazer num determinado estado da base de dados.

No contexto deste modelo, chama-se leitura e escrita *standard* àquelas operações de leitura e escrita que têm consistência mais fraca e leitura e escrita forte àquelas que possuem uma consistência mais forte. Para maximizar o processamento local e reduzir os acessos de rede, permite-se que o utilizador interaja com os dados disponíveis localmente (num *cluster*) através do uso de operações de escrita e leitura fracas. Deste modo, e tendo-se em conta o tipo de consistência e as operações *standard* e fortes, podem-se distinguir dois tipos de transacções:

- **Transacções Fracas,** que integram operações de leitura e escrita fracas.
- **Transacções Fortes,** que integram apenas leituras e escritas fortes.

As transacções fracas permitem apenas o processamento local do utilizador, sem sobrecarregar a rede, enquanto que as transacções fortes necessitam de acessos à rede para que se garanta a consistência das suas actualizações. As transacções fracas possuem dois pontos de *commit*, um local, no *cluster* associado, e um global, implícito aquando da junção dos *clusters*. O ponto de *commit* local é expresso por uma operação de *commit* explícita. As actualizações feitas pelo *commit* de uma transacção fraca são apenas observadas por outras transacções fracas que se encontrem no mesmo cluster, sendo apenas observadas pelas transacções fortes depois de se fazer a junção dos *clusters*, altura em que as transacções locais são guardadas globalmente. Deste modo, as alterações de uma transacção fraca são consideradas permanentes apenas depois do *commit* global, podendo ser desfeitas antes deste *commit*, mesmo quando já se tiver efectuado o *commit* local. Por outro lado, o facto de existirem dois tipos de transacções faz com que cada item de dados tenha duas versões, uma forte e outra fraca.

Para processar as operações de uma transacção, o SGBD traduz as operações sobre os itens de dados em operações sobre as cópias desses itens de dados. Esta tradução é formalizada através de uma função de tradução. Uma operação de leitura fraca é traduzida numa leitura local, de uma cópia disponível, e uma escrita fraca traduz-se numa operação de escrita local sobre uma das cópias disponíveis. De referir, que estas alterações não podem ser vistas pelas transacções fortes até ao momento da junção dos *clusters*, altura em que as transacções locais são guardadas globalmente. As operações de *abort* e *commit* de transacções fracas são mapeadas em *aborts* e

commits locais. Nas transacções fortes uma leitura corresponde à leitura de dados fortes, o mesmo acontecendo para as escritas fortes. Um *abort* ou um *commit* de uma transacção forte corresponde a um *abort* ou um *commit* global.

5.3. Transacções Baseadas na Semântica

No modelo de transacções móveis baseadas em semântica [Walborn & Chrysanthis 1995] todo o processamento se baseia na semântica dos objectos, sendo esta também usada para fornecer uma maior autonomia às estações móveis durante as desconexões temporárias da restante rede.

Quase todos os tipos de modelos de processamento de transacções usam algum tipo de conhecimento semântico para fornecer uma maior disponibilidade dos dados, bem como para aumentar a concorrência e simplificar a recuperação no momento em que ocorre alguma falha. No fundo, a semântica de concorrência de um objecto depende:

- Da semântica das operações que está relacionada com os efeitos que uma operação pode causar no estado de um objecto.
- Dos valores de entrada ou saída de uma operação, referindo-se tanto à direcção do fluxo de informação de e para um objecto, como também à interpretação desses valores de entrada e saída.
- Da organização dos objectos, que está relacionada com a organização abstracta de um objecto.
- Do uso do objecto, que se relaciona com o modo como um objecto é usado, bem como o que se faz com a informação extraída do objecto.

As três primeiras características apresentadas são também usadas para definir várias formas de comutatividade que determinam, semanticamente, quando duas operações têm permissão para executar concorrentemente sem pôr em causa a propriedade de seriação, fornecendo deste modo o critério de correcção das bases de dados tradicionais. Por sua vez, a última característica apresentada, que se refere ao uso do objecto, tem sido usada para tentar definir o critério de

correcção específico das aplicações, que transcende a comutatividade e a seriação, para permitir que mais operações executem concorrente e assincronamente.

[Walborn & Chrysanthis 1995] consideram que existem dois tipos de semântica que podem influenciar o modo de processamento dos modelos de transacções baseados em semântica:

- Semântica independente da aplicação[25].
- Semântica dependente da aplicação[26].

Embora ambas tirem partido da semântica dos objectos de dados, tratam e extraem essa informação de diferentes modos. Nas secções seguintes faz-se uma descrição destes dois tipos de semântica, bem como a forma como podem influenciar o processamento de transacções móveis.

5.3.1. Semântica Independente da Aplicação

A comutatividade é uma das propriedades semânticas das operações usadas. Como se sabe já dos sistemas de base de dados tradicionais, quando duas operações comutam pode-se concluir que os seus efeitos no estado de um objecto, bem como os seus resultados, são independentes da sua ordem de execução. Deste modo, pode-se ordenar arbitrariamente as operações que comutam, não se violando o critério de seriação. Convém, ainda, referir que, existem operações que são comutativas para todos os estados do objecto e outras que apenas o são quando o objecto se encontra num determinado estado. A comutatividade é usada muitas vezes para aumentar a concorrência e para simplificar o processamento dos protocolos de recuperação. Existem, mesmo, protocolos que apenas permitem a execução concorrente das operações comutativas, como técnica de prevenção de abortos em cascata[27].

Tendo em conta a comutatividade para objectos guardados no servidor da base de dados, se todas as operações comutarem umas com as outras em todos os estados do objecto, então este pode ser guardado e manipulado localmente e de forma assíncrona na estação móvel, sem qualquer intervenção do servidor. Porém, impõe-se que a estação móvel comunique

[25] *Application Independent Semantic.*
[26] *Application-Dependent Semantic.*
[27] *Cascading Aborts.*

periodicamente ao servidor as alterações que são efectuadas localmente, de modo a que as transacções sejam guardadas, se possível, globalmente.

Todavia, na realidade prática de um Sistema de Base de Dados são poucas as operações de um objecto que comutam entre si, fazendo com que necessitem de alguma coordenação das operações conflituosas, o que implica que haja comunicação para esta coordenação. Além disto, a menos que os conflitos entre as transacções sejam raros, este modelo pode permitir que hajam alguns abortos das transacções móveis. Para que se validem mais transacções móveis, evitando-se os abortos, pode usar-se tanto a comutatividade como a dependência de série. Existem esquemas mais pessimistas, onde os objectos guardados podem ser bloqueados de modo a que sejam usados exclusivamente por uma determinada estação móvel, contudo nestes esquemas podem existir bloqueios desnecessários, pois durante as desconexões as estações móveis não podem libertar nenhum objecto.

Alguns métodos de armazenamento existentes tentam guardar os objectos de dados completos. Porém a transmissão de grandes objectos sobre redes de comunicação sem fios, com reduzida largura de banda, pode resultar em grandes congestionamentos na rede de comunicação. Para além de que, geralmente, as estações móveis têm um limitado espaço de armazenamento, o que pode implicar que, em determinadas alturas, apenas se consiga armazenar uma parte dos objectos de dados necessários ao processamento local. Por outro lado, se o tamanho do objecto for pequeno este não causa, em princípio, problemas de congestionamento de rede nem de armazenamento. Contudo pode não possuir os dados suficientes para o processamento local. Deste modo, torna-se um factor de extrema importância o tamanho dos objectos, ou seja a sua granulosidade, que convém que seja o mais fina possível, mas que contenha os objectos necessários, para que se consiga um subconjunto da base de dados capaz de sustentar o processamento local numa estação, durante as desconexões. Aqui é também útil o uso da semântica da organização dos objectos para fragmentar grandes objectos em componentes, com tamanho mais reduzido, que possam ser guardados independentemente e que, em simultâneo, sejam consistentes.

Deste modo, a comutatividade das operações pode ser usada em métodos de armazenamento, que usam o protocolo de concorrência para assegurar a coerência de armazenamento, bem como para facilitar os processos de recuperação das transacções.

5.3.2. Semântica Dependente da Aplicação

Quando os ambientes computacionais possuem uma semântica dependente de aplicação os métodos de processamento de transacções tiram partido da já referida característica da semântica do objecto, o uso do objecto, relaxando também o critério de seriação em favor de um critério específico mais fraco, mas igualmente aceitável. Ao contrário dos métodos que usam critérios de seriação, estes métodos consideram a consistência dos dados e a correcção da transacção independentemente.

A consistência dos dados captura a correcção da perspectiva dos objectos na base de dados, podendo variar de uma consistência estrita a uma consistência eventual. Os requerimentos da correcção da transacção capturam a correcção da perspectiva da sua estrutura e do comportamento das transacções. Assim, estes métodos permitem que as aplicações especifiquem:

- O grau de isolamento de diferentes transacções, isto é, o número de interacções aceitáveis entre as transacções, bem como a delegação de operações de *commit* parciais ou antecipados das operações de transacção.

- O grau de autonomia da transacção, em termos de interdependências das transacções, tais como dependência de *commit* e aborto.

Claramente, estes critérios de correcção relaxados complicam a gestão de transacções, bem como os seus processos de desenvolvimento e recuperação, não sendo suficientes as técnicas tradicionais para restaurar a base de dados para um estado consistente após a ocorrência de uma falha. Pois, são necessários alguns passos adicionais de compensação para desfazer semanticamente as operações previamente executadas e guardadas. Todavia, apesar destas desvantagens, esses critérios de correcção relaxados são bastante úteis no processamento de transacções móveis, já que aumentam a autonomia das estações móveis.

Para tentar resolver os problemas subjacentes aos protocolos usuais de controlo de divergência, neste modelo de processamento de transacções móveis usa-se a organização do objecto e a semântica da aplicação de modo a partir os objectos grandes em fragmentos mais pequenos para serem guardados independentemente e manipulados assincronamente.

5.3.3. Aspectos do modelo de transacções Baseado na Semântica

Em [Walborn & Chrysanthis 1995] defende-se que dadas as características dos ambientes móveis, o processamento baseado em semântica é o único modo viável de construir aplicações de bases de dados móveis. Para melhor suportar as operações desconectadas e, sempre que possível, manter a consistência de dados estrita, este modelo utiliza todos os tipos de informação semântica dos objectos. Este modelo de processamento de transacções móveis tenta ainda fornecer uma granulosidade de armazenamento fina, controlo de concorrência, manipulação assíncrona dos objectos guardados e o *commit* unilateral das transacções nas estações móveis.

A ideia base deste modelo é a de partir objectos grandes e complexos em fragmentos mais pequenos, que sejam do mesmo tipo do objecto original, mas que tenham a sua manipulação mais facilitada. Assim, com uma fragmentação adequada a estação móvel pode armazenar uma partição de um objecto, minimizando os seus requisitos de armazenamento. Outra ideia, subjacente a este modelo, é a de fazer destes fragmentos como a unidade de reconciliação das actualizações, ou seja, a unidade de consistência. Deste modo, o objectivo é suportar fragmentação de todos os objectos da base de dados, através da comutatividade de operações, que se baseia nas restrições de consistência e no uso do objecto.

A cópia primária de cada objecto deve residir no servidor, ou servidores da base de dados. A estação móvel é a entidade que deve especificar a granulosidade que se pretende para um determinado objecto e as restrições de utilização através das operações de fragmentação[28]. Deve ainda existir uma operação de junção[29] que permita que as transacções libertem os fragmentos, para que estes possam ser novamente incorporados na cópia primária. As operações de fragmentação e de junção podem estar encapsuladas nos próprios objectos. Os objectos de dados estendidos com estas duas operações designam-se por objectos fragmentáveis.

Quando uma estação móvel requer o acesso a um objecto, deve enviar um pedido de armazenamento ao servidor, invocando a operação de fragmentação com dois parâmetros: o critério de selecção e as condições de consistência. O critério de selecção especifica o objecto a ser armazenado, bem como o tamanho pretendido para a partição. Quando uma partição do objecto é guardada numa estação móvel é, em simultâneo, logicamente removida da cópia primária, sendo apenas acessível pelas transacções dessa estação, permanecendo a restante

[28] *Split.*
[29] *Merge.*

parte do objecto acessível no servidor. Por outro lado, as condições de consistência especificam as restrições que necessitam de ser satisfeitas pelo fragmento, de modo que se mantenha a consistência de todo o objecto.

Para que seja possível o *commit* unilateral das transacções a executar nas estações móveis, devem-se reter os efeitos das operações de transacção em cada fragmento quando estes são reunidos. Frequentemente, a ordem de recombinação dos fragmentos pode ser ditada pela estrutura do objecto original ou pelas operações executadas em cada fragmento. Contudo, existem objectos cujos fragmentos podem ser reunidos de várias formas sem destruir a sua consistência de dados. Em particular, existem objectos nos quais a sua ordenação é um artefacto da sequência das operações executadas sobre o objecto. Quando os fragmentos de um objecto podem ser rearranjados para reflectir uma sequência alternativa das operações no objecto, este diz-se reordenável[30].

5.4. Transacções a dois níveis

O modelo de transacções aqui apresentado [Gray et al. 1996] supõe a existência de um modelo de replicação a dois níveis (Secção 4.3), no qual um dos níveis é representado pelas estações móveis e outro pelas ESMs. As estações móveis guardam as cópias dos fragmentos da base de dados e executam as tarefas locais sobre elas, produzindo dados que se designam de tentativos e que, posteriormente, devem ser validados de modo a que se tornem dados definitivos. Quando as estações móveis se conectam à rede fixa podem propor transacções de actualização tentativas à estação que contém a réplica mestre. Esta, por sua vez, deve reexecutar todas as transacções que a estação móvel executou durante a desconexão. Se estas terminarem com sucesso então são tornadas definitivas. Caso contrário são rejeitadas. Porém, as transacções tentativas rejeitadas devem ser reconciliadas pela estação móvel que as gerou. Depois das trocas de informação, as estações móveis devem encontrar-se sincronizadas com as estações fixas. O sistema a dois níveis opera como um sistema *Lazy-Master*, mas com a restrição de que nenhuma transacção pode actualizar dados que tenham as cópias mestre em mais do que uma estação móvel, não sendo esta restrição verdadeiramente necessária quando as estações se encontrem conectadas.

[30] *Reorderable*.

O sistema a dois níveis opera como um sistema *lazy-master*, mas com a restrição de que nenhuma transacção pode actualizar dados que tenham as cópias mestre em mais do que uma estação móvel, não sendo esta restrição verdadeiramente necessária quando as estações se encontrem conectadas.

Nas estações móveis, os itens de dados replicados possuem duas versões: a mestre e a tentativa. A versão mestre de um item de dados corresponde ao último valor recebido da estação que possui a sua cópia mestre. As estações que contêm a cópia mestre possuem sempre a sua versão mestre, cujo valor pode coincidir ou não com o das restantes réplicas. De modo semelhante, podem-se identificar dois tipos de transacções: as base e as tentativas. As transacções base trabalham apenas sobre dados mestre, originando, obviamente, dados mestre. Por sua vez, as transacções tentativas trabalham apenas sobre os dados locais e, como o próprio nome indica, produzem versões tentativas dos dados. Sempre que executa uma transacção tentativa produz-se uma transacção base, que deve ser executada nas ESMs quando se restabelecer a conexão. Deste modo, as actualizações que foram executadas localmente podem ser executadas globalmente.

Uma transacção base gerada por uma transacção tentativa pode falhar ou produzir resultados diferentes dos obtidos pela transacção tentativa. Para isso, as transacções base possuem um critério de aceitação que testa os resultados de saída. Quando uma destas transacções falha, deve-se notificar a estação que a iniciou, indicando-se ainda os motivos da falha. Uma falha de aceitação é equivalente aos mecanismos de reconciliação dos esquemas de replicação *lazy-group*, com as diferenças de que:

- A base de dados mestre está sempre convergente – não existe *delusion* do sistema.
- A estação originária necessita apenas de contactar a ESM de forma a descobrir se a transacção tentativa é aceitável.

O critério de aceitação é específico às aplicações. O sistema de replicação não pode fazer mais do que detectar as diferenças entre os valores da transacção tentativa e os da base, o que provavelmente pode ser um teste muito pessimista. Assim, o sistema de replicação executa, simplesmente, as transacções tentativas. Se estas terminam com sucesso e satisfazem o teste de aceitação, então o sistema de replicação assume que tudo correu bem, propagando as actualizações pelas restantes réplicas. Caso contrário, o utilizador móvel deve rever e resubmeter essa transacção.

Deste modo, com o uso deste modelo de transacções, sempre que a estação móvel se conecte à ESM deve:

- Descartar as versões dos objectos tentativas, pois estas serão actualizadas pelas versões mestres.
- Enviar para a ESM as actualizações que efectuou nas réplicas mestre que possui, pois só assim é que estas podem visualizadas pelas restantes estações que pretendem aceder a estes dados.
- Enviar todas as suas transacções tentativas (e todos os seus parâmetros de entrada) para a ESM, de modo a que sejam executadas pela mesma ordem com que foram guardadas localmente.
- Aceitar a actualização de réplicas que a ESM indicar.
- Aceitar a notificação do sucesso ou falha das transacções tentativas que enviou.

Por sua vez, a ESM, que é o outro nível deste modelo, quando se encontra conectada com a estação móvel, deve:

- Enviar para outras estações, as transacções de actualização contendo os itens de dados actualizados pela estação móvel.
- Aceitar as transacções atrasadas de actualização de objectos, dos quais a estação móvel possui a réplica mestre.
- Aceitar a lista de transacções tentativas, as suas mensagens de entrada e o seu critério de aceitação.
- Reexecutar as transacções tentativas pela mesma ordem com que foram executadas na estação móvel.
- Propagar as actualizações para todas as outras réplicas (seguindo um modelo *lazy-master*) depois de ter guardado a transacção base.

As transacções tentativas devem ser desenhadas de modo a que possam comutar com outras transacções, na esperança de se aumentar a probabilidade de sucesso. As transacções tentativas devem acompanhar a seguinte regra de *scope*: podem envolver objectos de dados que possuem a versão mestre nas estações fixas e outros que possuem as versões mestre na estação móvel que originou a transacção. O objectivo é dar a ideia de que a estação móvel está conectada com as estações fixas durante a execução das transacções tentativas. Basicamente, deve dar-se a ideia de que estas transacções são executadas como verdadeiras transacções base. A regra de *scope* assegura que a transacção base apenas acede a dados mestre da estação móvel originária e das estações base.

Se a transacção base falhar o processo de aceitação deve ser abortado, sendo enviada uma mensagem de diagnóstico à estação móvel. Se o critério de aceitação requeria que as transacções base e tentativas tivessem valores de saída semelhantes, então as transacções subsequentes ao lerem esses resultados tentativos também devem falhar. No entanto, podem ser usados critérios de aceitação mais fracos. Quando todas as transacções tentativas são re-processadas como transacções base, o estado da estação móvel converge para o estado da ESM.

[Liu et al.1999] defendem que este modelo de transacções, usando a replicação a dois níveis, pode causar um grande *overhead* de reprocessamento na cópia mestre e sugerem que se acrescente, a este modelo, alguma semântica que tente evitar esse *overhead*. Para tal, usam um método de junção de histórias das transacções que substitui o reprocessamento que ocorre neste modelo. Deste modo, sempre que uma estação móvel se conecta à rede fixa faz-se a junção das histórias tentativas e das histórias base, podendo assim ser aproveitada uma grande quantidade do trabalho das transacções tentativas. De modo semelhante à classificação das transacções, diz-se que se trata de uma história tentativa quando se trata da história de execução de transacções tentativas, iniciadas e executadas nas estações móveis. Por sua vez, uma história diz-se base, quando corresponde à história de execução das transacções base, que lêem e escrevem sobre dados mestre.

Quando duas histórias entram em conflito constrói-se um grafo de precedências para se detectarem as inconsistências e para se calcular o conjunto de transacções que as causou. Estas, quando são retiradas ou remodeladas, podem resolver os conflitos, rescrevendo a história das transacções. O processo de retirar essas transacções indesejáveis pode ser bastante complexo porque podem afectar outras transacções. Em [Liu et al.1999] propõem-se alguns algoritmos de rescrita das histórias das transacções. Este modelo de transacções mantém os aspectos positivos do modelo de transacções de replicação a dois níveis, tentando evitar alguns dos problemas que

aí surgem, como o *overhead* de reprocessamento. Além disto, a propriedade de durabilidade das transacções não é violada.

5.5. O Modelo PRO-MOTION

Nesta secção apresenta-se o modelo de transacções móveis PRO-MOTION [Walborn & Chrysanthis 1997] [Mazumdar & Chrysanthis 1999]. Este modelo, para tentar lidar e mesmo ultrapassar as limitações dos ambientes móveis, fornece uma infra-estrutura flexível para o processamento de transacções num SBDM que use uma arquitectura Cliente/Servidor. O PRO-MOTION tem como principal objectivo a maximização do número de transacções executadas pelas estações móveis, durante os períodos de desconexão. Neste modelo, assume-se que o sistema possui pelo menos um servidor e que as aplicações das estações móveis operam sobre os dados geridos por esses servidores.

Num ambiente móvel ficaria muito caro se cada transacção fosse enviada directamente para o servidor, podendo mesmo ser impossível a realização de tais pedidos. Daí que, o PRO-MOTION permita que se faça a replicação de alguns itens de dados. Tais réplicas encontram-se sempre encapsuladas e são designados por *Compacto* (Figura 15).

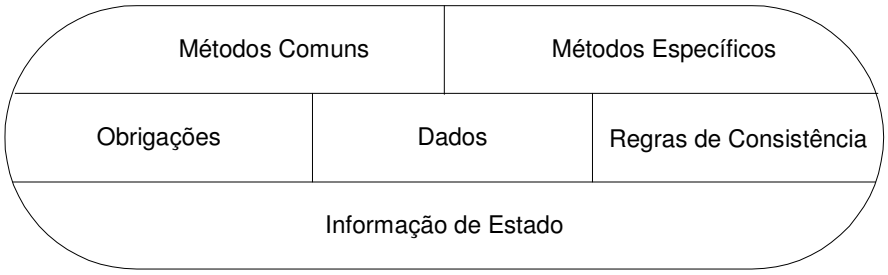

Figura 15 – Estrutura de um *Compacto.*

Um *Compacto* é constituído pelas réplicas dos dados, pelos métodos de acesso aos dados, informação sobre o estado actual do *Compacto*, regras de consistência, obrigações (por exemplo, um limite no tempo durante o qual a estação móvel tem os direitos sobre um determinado recurso) e, ainda, pelos métodos utilizados pela estação móvel para poder manipular os itens de dados.

Neste modelo de transacções existe um gestor de transacções que é o responsável pelo controlo e invocação dos *Compactos*. No servidor da base de dados existe também um gestor de *Compactos* e um gestor de mobilidade. Este último tem como funcionalidade principal auxiliar a comunicação entre o gestor de *Compactos* do servidor e o agente *Compacto* da estação móvel. Por sua vez, o agente *Compacto* é o responsável pela gestão dos *Compactos* e funciona como um gestor de transacções para a estação móvel. Pode-se assim dizer que a gestão dos *Compactos* é efectuada através de um esforço cooperativo entre o servidor e a estação móvel. Sempre que uma estação móvel necessitar de dados que não se encontrem armazenados localmente, deve enviar um pedido ao servidor da base de dados. Se os dados estiverem disponíveis então o pedido é satisfeito, devendo o gestor de *Compactos* do servidor proceder à criação de um *Compacto* com os dados pedidos e com a informação necessária para a sua correcta manipulação e interpretação. De seguida, deve enviá-lo para a estação móvel.

Os pedidos de dados podem ser adaptados de modo a que incluam alguns itens do compacto em falta ou desactualizados. Por este motivo, a transmissão dos métodos pode ser uma tarefa difícil e custosa, mas que pode ser evitada se esses métodos já se encontrarem disponíveis nas estações móveis. No fundo, um *Compacto* é, um pedido de dados, que possui também obrigações, restrições e informação de estado. Além disto, um *Compacto* representa um acordo estabelecido entre o servidor e a estação móvel, através do qual o servidor delega controlo de dados à estação móvel.

Sempre que a estação móvel recebe um *Compacto* este é guardado localmente. Cada *Compacto* tem uma interface comum que é usada pelo agente de *Compactos* para gerir todos os *Compactos* que se encontram no seu registo. Os principais métodos de gestão incluem:

- ***Inquire,*** que permite obter o estado actual do *Compacto*.
- ***Notify,*** para que o *Compacto* seja notificado sempre que é alterado o estado da estação móvel.
- ***Dispatch,*** que processa, no *Compacto*, as operações necessárias à execução de transacções da estação móvel.
- ***Commit***, que guarda as operações de uma transacção, na base de dados.
- ***Abort***, que aborta as alterações feitas por uma transacção de um *Compacto*.

Os *Compactos* devem ser actualizados periodicamente como resultado do processamento de transacção nas estações móveis. Além disto, sempre que as necessidades da estação móvel ou do servidor se alterem, os compactos devem ser renegociados para que os recursos sejam redistribuídos. Por outro lado, quando a estação móvel já não necessita de tais recursos, deve devolver os *Compactos* correspondentes ao servidor, devendo ainda removê-los do seu registo e armazém de *Compactos*.

Assim, pode concluir-se que o modelo de transacções móveis PRO-MOTION tenta fornecer um processamento de transacções para ambientes móveis. Para tal, baseia-se:

- No uso de *Compactos*, que servem como unidade de controlo e de armazenamento.
- Na exploração da semântica dos objectos sempre que possível de forma a aumentar a autonomia das estações móveis e a concorrência.
- Na introdução de um sistema de gestão de transacções, que consiste num gestor de e num agente de *Compactos* capazes de negociar e gerar os *Compactos,* e de fornecerem uma gestão local de transacções na estação móvel.

5.6. As Transacções Canguru

As Transacções Canguru representam outro modelo para transacções móveis [Dunham et al. 1997] [Dunham & Kumar 1999]. Este modelo dá especial atenção ao facto das transacções nos ambientes móveis mudarem muitas vezes de célula, e, consequentemente, possuírem saltos frequentes de uma ESM para outra. Por este motivo, designam-se estas transacções móveis por transacções Canguru.

Na definição de uma transacção Canguru deve-se ter em conta o conceito de transacção global. Esta consiste numa sequência de transacções, que pode incluir outras transacções globais (Figura 16). Neste modelo considera-se ainda o conceito de transacção *joey* (TJ), que consiste numa sequência de zero ou mais transacções usuais ou globais, seguidas de uma operação de um *commit*, um aborto ou uma troca.

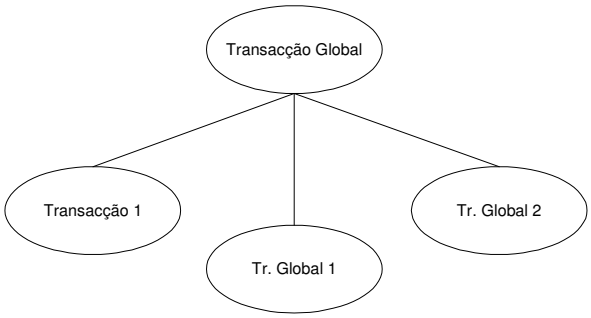

Figura 16 – Vista de uma transacção global.

O modelo de transacções *Canguru* considera a existência de uma arquitectura baseada em agentes. Assim, quando é feito um pedido de transacção, por uma estação móvel, o agente da ESM correspondente deve criar uma transacção móvel (Canguru) para realizar o pedido, associando-lhe um identificador. As transacções móveis são constituídas por um conjunto de subtransacções. Cada subtransacção representa uma unidade de execução numa ESM, que é designada por transacção *joey*. Assim, pode-se definir uma Transacção Canguru como uma sequência de uma ou mais transacções *joey* (Figura 17). A última transacção *joey* da sequência deve acabar num *commit* ou num aborto, enquanto que as restantes devem terminar com uma operação de troca. À sequência de transacções locais e globais que são executadas numa dada transacção *Canguru* dá-se o nome de *pouch*.

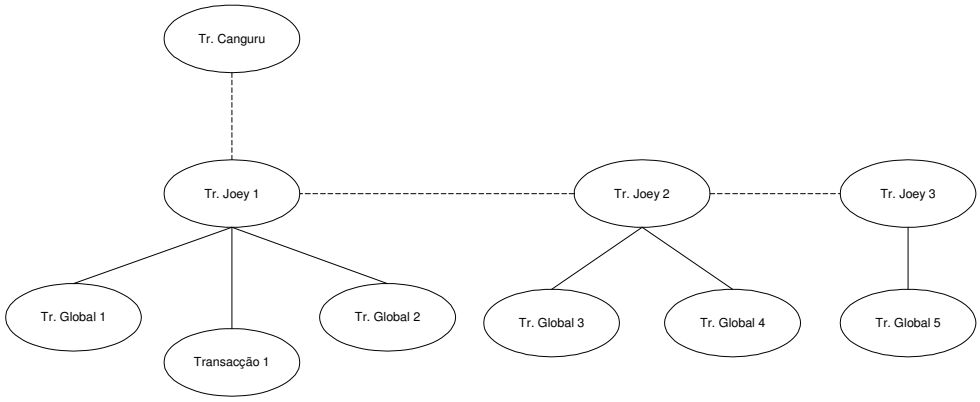

Figura 17 – Exemplo de decomposição de transacção *Canguru.*

Quando uma estação móvel salta de uma célula para a outra, o controlo da transacção Canguru deve passar para a nova célula, tendo assim um novo agente de acesso aos dados. Nesta altura,

é criada uma nova transacção *joey* que é acompanhada por uma operação de troca, enquanto que a transacção antiga é guardada independentemente da nova. Só se cria uma nova transacção *joey* quando há uma mudança de célula. Isto faz com que, a mesma transacção, pedida em diferentes alturas, possa possuir diferentes estruturas. Diz-se que duas transacções Canguru são equivalentes quando possuem o mesmo *pouch*.

Para gerir a execução e a recuperação de uma transacção *Canguru* mantém-se uma lista duplamente ligada com todas as ESMs envolvidas na sua execução. Para finalizar uma transacção Canguru que se encontrava parcialmente completa deve-se atravessar essa lista em modo *forward*, começando na ESM que a originou. Assim, para recomeçar uma transacção interrompida, o utilizador deve ser capaz de indicar a estação onde se iniciou a transacção. Quando se pretende desfazer uma transacção *Canguru* essa lista deve ser percorrida em sentido contrário. As transacções *Canguru* têm os seguintes modos de processamento:

- **Modo de compensação[31].** Quando uma transacção se encontra neste modo, uma falha da transacção *joey* obriga a que todas as outras transacções *joey,* anteriores ou posteriores, sejam desfeitas. Deste modo, os *commits* das transacções *joey* precedentes devem ser compensados, sendo o próprio utilizador, ou sistema origem, quem deve fornecer a informação necessária para construir as transacções de compensação. O sistema não deve apenas garantir que as transacções *joey* são compensadas, como deve, também, garantir que o *commit* das transacções de compensação se efectua com sucesso.

- **Modo de troca[32].** As transacções são executadas por defeito neste modo de operação. Neste caso, quando uma transacção *joey* falha, não são pedidas, por parte da transacção Canguru, quaisquer novas transacções locais ou globais. Ficando a cargo do SGBDM decidir se deve ou não abortar as transacções *joey* que se encontram a executar. As transacções *joey* precedentes, que fizeram *commit* não necessitam de ser compensadas.

Nenhum destes modos de operação garante a seriação das transacções *Canguru*. O modo de compensação assegura a atomicidade das transacções, mas viola a propriedade de isolamento[33],

[31] *Compensating mode.*
[32] *Split mode.*
[33] *Isolation.*

já que os *locks* são obtidos e mantidos a nível local da transacção. Contudo, com este modo de operação, as transacções são seriáveis.

Neste modelo de transacções móveis, a gestão das transacções é feita com base em duas tabelas: a de estado e a de *logs*. A tabela de estados (Tabela 1) contém a informação dos estados de todas as transacções que se encontram a executar. Por sua vez, a tabela de *logs* (Tabela 2), que cada ESM deve conter, possui registos que serão necessários para eventuais processos de recuperação. A maioria destes registos está relacionada com entradas da tabela de estados.

Em conclusão, o modelo de transacções móveis baseado nas transacções *Canguru* capta tanto o comportamento dos dados como também o seu movimento, incorporando para tal a propriedade saltitante das transacções móveis. Como se sabe, o comportamento móvel é dinâmico e é conseguido neste modelo através das operações de troca. O comportamento de acesso aos dados é capturado através do uso de transacções locais e globais. Este modelo fornece, ainda, a facilidade de lidar tanto com transacções curtas como com transacções mais longas.

Tipo de Registo	**Atributos**	**Descrição**
KT	KTID	Identificador para uma transacção Canguru
	Modo	Troca ou compensação
	Contador de TJ	Número de TJs activas na transacção Canguru actual
	Estado	Activa, *commiting*, *aborting*
	FirstJTID	Apontador para o primeiro registo de estado de TJ nesta Transacção Canguru
JT	JTID	ID para a TJ
	NextJTID	Apontador para o próximo registo de estado da TJ
	PriorJTID	Apontador para o registo de estado anterior da TJ
	Estado	Activo, *Commit* ou Aborto
	STList	Lista de transacções locais e globais
	Compensável	Sim/Não
ST	STID	ID da subtransacção
	Estado	Activo, *Commit* ou Aborto
	Pedido	Pedido de transacção local ou global
	Compensável	Sim/Não
	CompTR	Transacção de Compensação

Tabela 1 – Exemplo das entradas da tabela de estado de uma transacção *Canguru.*

Tipo de Registo	Conteúdo
BTKT	KTID, Modo
CTKT	KTID, Modo
BTJT	JTID, Prior JTID
BTST	STID, Pedido, Compensação
ETJT	JTID, Próximo JTID
ETST	STID
ETKT	KTID
HOKT	KTID

Tabela 2 – Exemplo de uma tabela com registos de *log.*

5.7. Modelo *Pré-Commit*

O modelo de transacções móveis apresentado nesta secção [Madria 1998A] baseia-se no uso de operações de pré-escrita. Estas operações não actualizam o estado do objecto de dados, tornando apenas visível o valor que o objecto de dados deve possuir depois do *commit* da transacção. Ou seja, uma operação de pré-escrita dá conhecimento do valor que um dado objecto de dados possuirá depois do *commit* da operação de escrita, mas que ainda não fez o *commit* definitivo.

Uma operação de pré-leitura retorna o valor de uma operação de pré-escrita. Enquanto que uma operação de leitura retorna o valor de operação de escrita. As operações de pré-leitura dão origem a leituras fracas. Estas são diferentes das apresentadas no modelo de *Clustering*, porque as leituras fracas deste modelo não devem ser abortadas. Depois de todas as operações de leitura, pré-leitura e pré-escrita terem sido executadas, a transacção pode efectuar o *pré-commit* na estação móvel. Todas essas operações devem ser processadas antes do *pré-commit*, já que, logo a seguir a esta operação, se faz o bloqueio das operações de pré-escrita. Porém, não se bloqueia as operações de leitura devido à utilização do protocolo de bloqueio em duas fases[34].

Os valores de *pré-commit* das transacções são visíveis tanto nas estações fixas como nas móveis, antes do *commit* final, aumentando, deste modo, a disponibilidade dos dados. Uma transacção

[34] *Two-Phase Locking.*

que tenha feito o *pré-commit* e continue em execução é transferida para a ESM para não gastar recursos na estação móvel.

Quando se processa a execução seriável a ordem de execução das transacções é decidida pela ordem dos *pré-commit*. Uma transacção não pode abortar depois do *pré-commit*, evitando-se assim os possíveis abortos em cascata. Contudo, pode acontecer que as transacções sejam forçadas a abortar devido a problemas relacionados com o sistema, como por exemplo a ocorrência de uma falha, podendo neste caso a transacção que efectuou o *pré-commit* ser reiniciada na parte do sistema onde não ocorreu a falha. Para se poder recomeçar uma transacção que já efectuou o *pré-commit,* a partir do último estado consistente, devem ser guardados os *logs* de escrita e pré-escrita.

Este modelo de transacções relaxa o isolamento das propriedades ACID das transacções, pois as alterações são visíveis no final do *pré-commit* e não do *commit* final. Também a propriedade de durabilidade é realizada na altura do *pré-commit* e não aquando do *commit* definitivo.

5.8. Controlo de Concorrência Optimista

O modelo de transacções móveis apresentado nesta secção [Ahmed et al. 1996] usa controlo de concorrência optimista e baseia-se no isolamento de transacções[35], com algumas modificações para que se possam satisfazer os requisitos das transacções móveis. Aqui, as estações móveis guardam, localmente, os itens de dados a que acedem com mais frequência. Um item de dados pode conter mais ou menos informação, conforme o que se quiser representar. Isto significa que cada item de dados pode ter diferentes granulosidades, não havendo uma regra geral (e óptima) para determinar qual deve ser o valor dessa granulosidade.

Uma estação móvel, que esteja a executar uma transacção, deve ler os itens de dados da sua base de dados local. Quando um item de dados não existe localmente, a estação móvel deve fazer um pedido à ESM. Todas as operações de escrita de uma transacção são efectuadas temporariamente na base de dados local, sendo validadas globalmente quando a transacção termina. O seu *commit* global é efectuado conforme o resultado dessa validação. Neste modelo, cada transacção tem ainda associado um estado, sendo este estado alterado através de

[35] *Isolation-Only Transactions Model.*

relatórios. Na Figura 18 apresentam-se alguns desses estados, dos quais se destacam os seguintes:

- **Validação.** Significa que uma transacção está a ser validada.
- **Pendente.** Ocorre quando não se consegue estabelecer a conexão com a ESM, encontrando-se pendente o processo de validação.
- **Suspensa.** Acontece sempre que a transacção está a aguardar algum relatório para poder continuar a sua execução.
- ***Commit.*** Este estado só ocorre depois de se saber que se trata de uma transacção válida que é guardada depois definitivamente.
- **Aborto.** Quando uma transacção é considerada inválida transita para este estado para que possa ser desfeita.

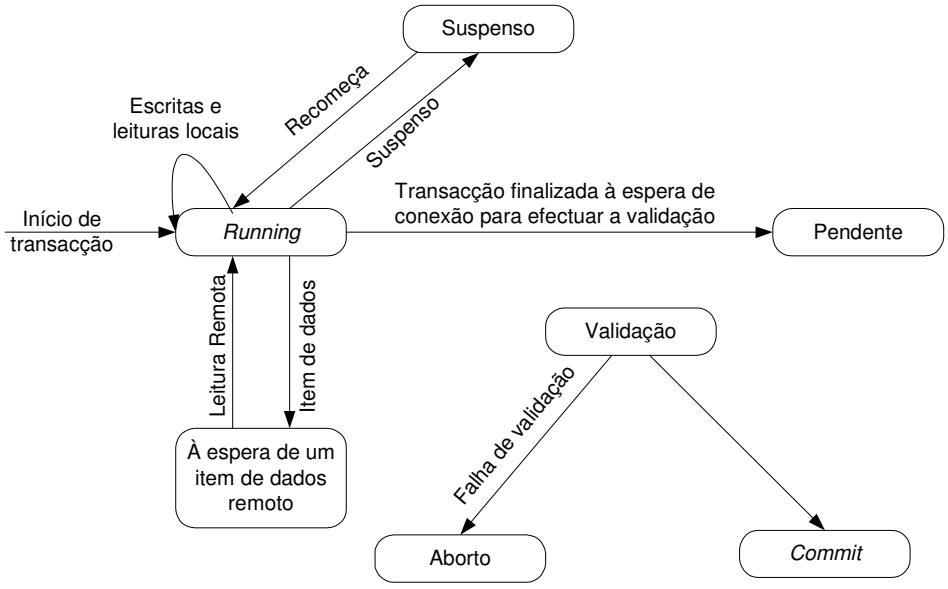

Figura 18 – Esquema com os estados de transição de uma transacção.

Neste modelo, são enviados periodicamente relatórios de invalidação para todas as estações móveis no domínio da célula, através das capacidades de *broadcast* das ESMs. Estes relatórios

identificam quais os itens de dados que foram actualizados desde o último relatório, assim como a invalidação de alguns dos itens guardados nas estações móveis. Os relatórios de invalidação não carregam valores de dados, mas sim os seus identificadores, o que faz com que não consumam muita capacidade de largura de banda.

Os relatórios de invalidação possuem inúmeras vantagens, entre as quais, o facto das estações móveis não necessitarem de estabelecer, com tanta frequência, ligações no sentido *uplink*, para invalidar os seus dados locais. Perante este cenário, pode-se concluir que é suficiente que as estações móveis estejam parcialmente conectadas para que consigam ouvir os relatórios de invalidação que a ESM envia periodicamente. Uma outra vantagem é que estes relatórios permitem uma validação local das transacções que apenas usaram operações de leitura. Isto porque, depois de executar uma transacção local de leitura, o gestor local de transacções deve esperar pelo próximo relatório de invalidação. Se nenhum dos itens de dados pertencentes ao conjunto de leitura da transacção for invalidado e se a transacção não depende de nenhuma outra transacção pendente, então a transacção pode ser guardada localmente, sem necessidade de intervenção da ESM. Com o uso destes relatórios, o modelo proposto passa a ser *kill-based* em vez de *die-based*, Isto é, as transacções conflituosas são abortadas imediatamente, podendo ser reiniciados no instante seguinte, evitando-se assim a existência de trabalho inútil. Por último, a ESM não necessita de ser um servidor repleto de estados, o que permite reduzir assim o *overhead* do sistema.

Apesar das inúmeras vantagens, os relatórios de invalidação apresentam também algumas desvantagens como o facto das estações móveis lerem relatórios sobre os quais não possuem informação necessária para o seu processamento. Contudo este trabalho pode ser limitado se se organizarem os relatórios em secções. Desta forma a estação móvel poderá ignorar as secções do relatório que não lhe interessam. Um problema que aqui se coloca é determinar a granulosidade das secções – esta é uma questão que ainda se encontra em aberto. Um outro problema poderá ser as grandes quantidades de informação que a ESM necessita de transmitir.

Para além dos relatórios de invalidação existem também os relatórios de actualização, que se apresentam como uma solução para o reinício das transacções. Tal deve-se ao facto de que, quando as estações móveis se encontram a operar em modo parcialmente conectado, pode não ser possível estabelecer a comunicação com a ESM para actualizar itens de dados que se encontram marcados como inválidos. Assim, propõe-se que a estação móvel armazene, localmente, esses relatórios antes da desconexão ou, então, tal como acontecia com os relatórios de invalidação, que a ESM transmita periodicamente esses relatórios. Os relatórios de

actualização devem conter desde o último relatório todos os itens de dados actualizados. Como refrescam constantemente os dados das estações móveis, estes podem aumentar o número de transacções guardadas localmente. Além disto, se uma transacção for abortada o utilizador continua a ter a possibilidade de a reiniciar, mesmo sem a necessidade de restabelecer a conexão com a ESM. Se uma transacção ler um item de dados marcado como inválido, então a transacção deve ser suspensa até que a estação receba um novo relatório de actualização. Embora estes relatórios sejam enviados com alguma frequência, esta é, mesmo assim, menor do que a que se verifica nos relatórios de invalidação.

Em conclusão, o modelo de transacções móveis baseado no controlo de concorrência optimista utiliza o envio periódico de relatórios de invalidação e de actualização no sentido *downlink*, o que faz com que não exista um grande consumo da largura de banda e evita a comunicação do tipo *uplink*.

5.9. Modelo Baseado na Ordem Linear

O modelo de transacções baseado na Ordem Linear [Yeo et al. 1996] tem como principal objectivo fornecer um método dinâmico para determinar a compatibilidade entre uma transacção invocada e as transacções que já se encontram em execução. Deste modo, tenta-se evitar o aborto ou recomeço global das transacções. Este modelo supõe a existência de um SMBD.

Uma transacção pode ser considerada como uma sequência finita de eventos, em que cada evento é visto como uma operação primitiva sobre um objecto de dados. Podem-se categorizar os eventos em:

- **Evento de invocação,** quando uma transacção global invoca uma operação sobre um objecto de dados.
- **Evento de resposta**, que corresponde a uma resposta de um evento de invocação.
- **Evento *Commit***, que ocorre quando se faz o *commit* de uma operação num dado objecto, numa dada transacção global.

Os autores deste modelo definem ainda história de uma transacção global como uma sequência de eventos e correspondentes respostas. Além disto, duas histórias são equivalentes se todas as transacções globais executam os mesmos eventos pela mesma ordem.

O critério de correcção usado neste modelo é a linearização[36], isto porque o processamento de transacções globais está ordenado linearmente. Se essa ordem for mantida em todas as estações envolvidas durante a execução dessas transacções globais, então a sua ordem de execução relativa também estará correcta. Neste modelo, consideram-se duas importantes propriedades intrínsecas:

- **Localidade[37].** A linearização é uma propriedade local. Se esta propriedade for mantida em cada estação então o sistema global também a vai preservar.

- ***Nonblocking.*** A seriação é uma propriedade de *blocking* e, como tal, uma transacção pode ter que ser desfeita, ou recomeçada, para que se mantenha a sua seriação.

No esquema aqui proposto, as transacções globais só são suspensas quando a apresentação das subtransacções globais necessita de ser controlada, de modo a obter a ordenação global[38]. Uma vantagem deste modelo é que não existe *overhead* associado aos mecanismos de controlo de concorrência para se manter a seriação.

A ESM deve coordenar e gerir as transacções da célula pela qual é responsável. Para tal usa três filas:

- **Fila de Entrada**, que contém todas as transacções globais que estão prontas a ser executadas.

- **Fila Activa**, que possui todas as transacções globais a executar.

- **Fila de Saída**, que contém todas as transacções globais terminadas.

Para além destas filas, o gestor de transacções globais deve ainda manter em cada estação a seguinte informação:

[36] *Linearizability*.
[37] *Locality*.
[38] *Orderability*.

- O estado das transacções globais sob o seu controlo.
- Grafos com as estações de todas as transacções globais em execução.

Para assegurar a ordem linear, uma transacção global deve obter um bilhete lógico que lhe dê permissão para iniciar a sua execução, que pode ser conseguido através do uso de exclusão mútua. Neste modelo usa-se uma técnica baseada em relógios lógicos e passagem de mensagens para configurar a ordem total das transacções globais.

5.10. Transacções em Tempo Real

Um ponto crítico da gestão de dados móveis é a necessidade de se responder em tempo real aos requisitos de acesso aos dados das aplicações suportadas. Contudo, é difícil manusear em tempo real as restrições da computação móvel impostas pelo hardware da estação portátil e pela tecnologia de comunicação sem fios. Em [Kayan & Ulusoy 1999] apresenta-se um modelo de transacções que pretende fornecer respostas em tempo real. Para tal, fazem a distinção entre transacções fixas e transacções móveis, sendo as primeiras executadas pelas estações fixas e as segundas pelas móveis. Consideram ainda que cada transacção está associada a uma restrição de tempo[39] em termos de *deadline.*

Cada transacção móvel existe no sistema sob a forma de um *Processo Móvel Mestre* (PMM) na estação móvel que gerou a transacção, de um *Processo Mestre Fixo* (PMF) na ESM correspondente e um *Fixed Cohort Processe* (FCP) nas estações onde residem os dados necessários à sua execução. Uma transacção fixa, por outro lado, encontra-se associada com um PMF na estação que a gerou e um FCP em cada uma das várias estações a que tem de aceder a dados. Cada transacção pode ter no máximo FCP por estação.

Uma transacção móvel consiste em operações de escrita, leitura e, eventualmente, em alguma interacção de utilizador, enquanto que uma transacção fixa consiste apenas em operações de leitura e de escrita. As mensagens de pedidos de dados para as operações de leitura e de escrita são enviadas para FCP das estações relevantes sob a coordenação do PMF. Para cada operação de leitura e escrita global recorre-se à utilização de um dicionário de dados global para descobrir

[39] *Timing Constraint.*

quais são as estações relevantes. Cada estação de dados tem uma cópia desse dicionário. Uma transacção móvel pode ser executada seguindo uma das seguintes estratégias:

- **Estação de execução é uma estação fixa (ESFH) – *Execution Site is a Fixed Host*).** Aqui, submete-se a transacção completa numa única mensagem de pedido à rede fixa. Nesta estratégia, o PMF tem o controlo da execução da transacção, que depois de concluir a execução da transacção deve enviar o resultado ao PMM.

- **Estação de execução é uma estação móvel (ESMH) – *Execution Site is a mobile Host*).** Neste caso, o PMM submete uma mensagem de pedido de dados para cada operação de leitura e de escrita ao PMF. O PMF manipula este pedido na rede fixa e fornece os dados ao PMM. Aqui, o processamento de cada operação é efectuado na estação móvel.

Na primeira estratégia não se usa as capacidades de processamento nem a energia da estação móvel para a execução de transacções, sendo, por este motivo, mais aconselhável para estações móveis que possuam pouca energia.

Cada transacção, móvel ou fixa, tem uma prioridade única baseada nas suas constantes temporais, que são usadas para ordenar os recursos e os pedidos de acessos aos dados das transacções. Assume-se que as transacções possuem *deadlines* firmes – as transacções que tenham perdido os seus *deadlines* são abortadas e descartadas do sistema.

O modelo aqui apresentado usa seriação para controlo de concorrência, que é conseguida através do uso de uma adaptação do esquema *Two-Phase Locking*. Para a resolução de eventuais conflitos de dados é o usado o protocolo *Priority Abort*, através do qual as transacções de baixa prioridade são abortadas quando se encontram em conflito com uma transacção de elevada prioridade. Para cada estação fixa no sistema existe um escalonador, que gere os pedidos de *lock* dos processos *cohort* a executar nessa estação. Cada FCP tem de obter um *lock* partilhado para leituras e um *lock* exclusivo para escritas.

Como já foi referido, a seriação global é conseguida através do uso de um esquema estrito do two *phase locking*, mantendo os *locks* de uma transacção até que se faça o *commit*. O *commit* atómico é conseguido através do uso de uma adaptação do protocolo *Two-Phase Commit*. Neste protocolo, designa-se o PMF como coordenador e cada processo *cohort* actua como participante.

Quando se faz o *commit* de uma transacção, as actualizações dos seus processos *cohort* devem ser armazenadas nas bases de dados locais.

Durante o período de desconexão, não é possível a comunicação entre o PMF e o PMM de uma transacção móvel. De modo a prevenir que uma transacção móvel bloqueie outra, por motivos de *deadline*, o seu PMF tem o direito de a abortar unilateralmente, mesmo quando não tenha controlo da execução da transacção. Todavia, depois da reconexão com as estações móveis deve informar o PMM acerca das suas decisões, quer de *commit* quer de aborto da transacção.

5.11. HiCoMo

[Lee & Helal 2002] apresentam um modelo de transacções móveis, designado por *High Commit Mobile Transactions* (HiCoMo), que se baseia numa noção relaxada de conflito. Este modelo tenta garantir uma elevada taxa de *commit* indiferente à ocorrência de desconexões, sendo aplicável tanto a dados derivados como a dados agregados ou estatísticos. É usado num sistema onde existem tabelas base na rede fixa e um *data warehouse* nas estações móveis.

As transacções iniciadas na rede fixa sobre tabelas base são chamadas de transacções base, enquanto que as transacções iniciadas nas estações móveis são chamadas de transacções HiCoMo, que são, basicamente, um conjunto de *queries* e actualizações sobre o *data warehouse*. As actualizações processadas no *data warehouse* são reflectidas nas tabelas base quando a conexão se restabelecer. Assim que a estação móvel se reconecta à rede fixa, as transacções HiCoMo são executadas e transformadas em actualizações nas tabelas base. Aliás, o ponto-chave deste modelo é o processamento efectuado aquando da reconexão à rede fixa, altura em que se criam as transacções base que são geradas através da combinação de tipos agregados, operações e configurações das tabelas base. É necessário um algoritmo de transformação para analisar as transacções HiCoMo e gerar as respectivas transacções base. Alternativamente, poderá ser feito também o reforço dos efeitos das transacções HiCoMo nas tabelas base, sem que se reexecute toda a transacção lógica, baseando-se apenas nas alterações que se pretendem efectuar sobre os dados das tabelas base.

Contudo, pode acontecer que não seja possível gerar actualizações nas tabelas base que reflictam capazmente os efeitos das transacções HiCoMo sobre as *data warehouses*, devendo-se assim permitir a existência de uma margem de erro que garanta o sucesso da maioria das

transacções HiCoMo. A margem de erro deve ser inicialmente desprezada para manter a consistência dos dados ao longo das tabelas base e do *data warehouse*. As transacções base geradas são aplicadas como um todo sobre as tabelas base. Se algumas destas transacções abortar deve-se tentar uma outra combinação e adoptar um modelo de transacções que suporte abortos concorrentes e individuais, como é o caso do modelo de transacções *nested*.

O critério de aceitação utilizado neste modelo de transacções HiCoMo é a convergência, ou seja, o estado das tabelas base é igual ao que é reflectido no *data warehouse*. Se as operações forem comutativas, este critério de aceitação é verificado a maioria das vezes, mas também podem falhar o critério de aceitação, quando existem interacções com outras transacções HiCoMo.

Capítulo 6

Processamento de *Queries*

O processamento de *queries* é um dos métodos mais usados na extracção de informação dos sistemas de base de dados tradicionais. Neste tipo de processamento uma interrogação – *query* –, geralmente escrita numa linguagem de alto nível, é transformada numa outra interrogação equivalente, escrita numa linguagem de mais baixo nível. Esta nova interrogação deve ser eficiente e traduzir correctamente a questão inicial, de modo a que se obtenha a informação desejada. Para além disto, devem existir métodos que realizem essa transformação de *queries* de uma forma rápida e eficiente, de modo a minimizar o consumo de recursos do sistema. Estes métodos de transformação designam-se, usualmente, por métodos de optimização de *queries*.

Nos sistemas de base de dados tradicionais estes métodos podiam ser divididos em quatro fases principais [Connolly & Begg 1999]:

- **Decomposição.** A fase de decomposição incluiu as tarefas de *parsing* e compilação de uma interrogação. O resultado desta fase é uma expressão em álgebra relacional equivalente à interrogação inicial.

- **Optimização.** Durante esta fase é criado o plano de execução da interrogação, que funciona como parâmetro da fase de geração de código.

- **Geração de código.** Aqui é gerado o código executável da interrogação.

- **Execução da *query*.** Durante esta fase executa-se o código gerado na fase anterior e obtém-se o resultado final da interrogação.

Nos sistemas distribuídos, o processamento de *queries* e a sua consequente optimização são tarefas bastante mais complexas, já que existe um maior número de parâmetros que podem afectar a execução e a eficiência de uma *query*. Como se sabe, nos sistemas distribuídos os dados encontram-se frequentemente fragmentados e replicados, dados estes que podem ser necessários ao processamento de uma dada *query*. Perante tal situação pode verificar-se um aumento do número de comunicações entre as várias estações para que seja possível realizar com sucesso o processamento da *query*. Tal como nos sistemas tradicionais, também nos sistemas distribuídos a *query* deve ser transformada numa segunda de mais baixo nível. Esta transformação deve ser correcta e eficiente, isto é, deve possuir a mesma semântica da *query* original e produzir o mesmo resultado que ela, mas de uma forma mais eficiente [Özsu & Valduriez 1999].

[Özsu & Valduriez 1999] subdividem o processo de optimização de *queries* para SBDDs em quatro fases distintas (Figura 19):

- **Decomposição da *query*.** Nesta fase processa-se a decomposição da *query* original numa outra *query* em álgebra relacional sobre as relações globais.

- **Localização dos dados.** A principal função desta fase é localizar toda a informação necessária para o correcto processamento da *query*. Para tal, é usada a informação da distribuição dos dados.

- **Optimização global.** Aqui, tenta-se encontrar a forma de execução mais eficiente para a *query*.

- **Optimização local.** Esta última fase é executada por todas as estações que possuam dados que sejam necessários à execução da *query*. Cada uma destas estações executa uma *sub-query* designada de *query* local.

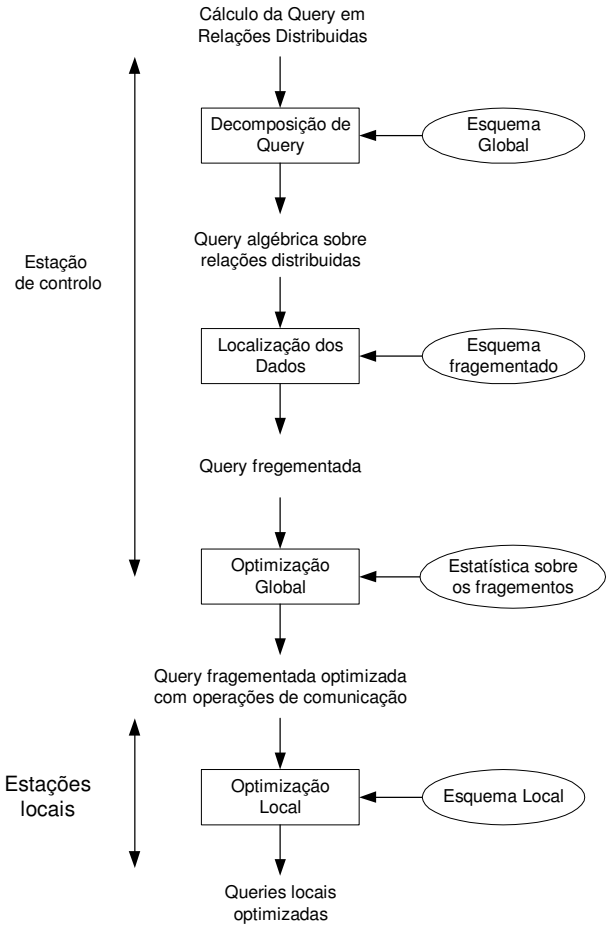

Figura 19 – Etapas da Optimização de *Queries* nos SBDDs.

A tarefa de processamento de *queries* torna-se ainda mais complexa nos SBDMs, pois, para além de existir distribuição e replicação de dados, algumas estações podem movimentar-se ou encontrarem-se temporariamente indisponíveis. Essa movimentação das estações pode influenciar o resultado de algumas *queries*. Por exemplo, se quiséssemos saber quais as farmácias de serviço disponíveis, a resposta a essa questão depende, naturalmente, do local onde o utilizador se encontra, ou ainda da hora em que a questão foi colocada.

O processamento de *queries* nos SBDDs, como já foi referido, pode ser dividido em várias fases. Uma destas fases é a optimização de *queries*, que se baseia essencialmente em funções de custos, para que, geralmente, seja escolhida a execução com os custos mais reduzidos. Nesta fase é também usual ter-se em conta os acessos a disco e a memórias. Contudo, nos SBDMs

estes critérios não são suficientes para calcular os custos associados ao processamento de uma *query*. Assim, deve ter-se em conta os aspectos particulares dos ambientes móveis como, por exemplo, os consumos de energia, uma vez que as estações móveis possuem baterias com um tempo de vida limitado.

Deste modo, a optimização do processamento de *queries* num SBDM deve considerar os seguintes aspectos [Kottkamp & Zukunft 1998]:

- **Os recursos limitados das estações móveis,** como a energia, a largura de banda reduzida, entre outros.

- **Localização das estações móveis,** uma vez que pode ser necessário o conhecimento da localização das estações que geraram a *query* ou mesmo das estações que possuem a informação necessária para a obtenção das respostas das *queries*.

Normalmente, nos SBDMs uma resposta a uma *query* do tipo "Quais as estações que se encontram numa dada região?" nunca está completamente certa, porque é frequente não se conhecer a localização exacta de algumas das estações. Assim, a resposta a uma *query* é um conjunto de pares (*e*, *p*), na qual *e* indica a estação e *p* a probabilidade dessa estação se encontrar na região pretendida. A probabilidade é, pois, utilizada como medida de certeza.

Quando se considera um SBDM *Ad-hoc*, como definido em 2.1, o problema de processamento de *queries* pode tornar-se ainda mais complexo, uma vez que nestes ambientes não se pode definir um conjunto de estações que estejam sempre acessíveis. Assim, é possível que num dado instante os dados que se procuram não se encontrem em nenhuma das estações disponíveis. Isto é, não se tem a possibilidade de se garantir que haja sempre uma réplica de dados disponível, o que pode impossibilitar, naturalmente, a geração de respostas às *queries* que forem lançadas.

Desta forma, para que se consiga obter um processamento de *queries* eficiente num SBDM devem ter-se em conta os seguintes aspectos [Imielinski & Badrinath 1993A]:

- Incorporar aquisição de dados no processamento de *queries*.

- Existência de novos tipos de *queries*:

    - Dependentes da localização do utilizador.

- Dependentes da direcção da deslocação do utilizador.
- *Queries* agregadas, como por exemplo *queries* que medem o tráfego global de uma dada área para ajudar na tarefa de alocação de recursos.

– Execução de *queries* sobre dados passageiros, uma vez que as alterações de informação são quase constantes neste tipo de ambientes, podendo mesmo acontecer que a informação se altere durante o processamento de uma determinada *query*.

– Processamento de uma grande quantidade de *queries* sobre um determinado conjunto de dados, podendo existir outros dados quase sem processamento.

6.1. Optimização de *Queries*

O processamento de algumas *queries* nos SBDMs pode, como já foi referido, requerer a localização de algumas estações. Deste modo, devem existir mecanismos que permitam a gestão da localização das estações. Como forma de solucionar o problema da localização dos utilizadores [Wolfson et al. 1999] consideram um tipo de deslocação diferente para os utilizadores. Os utilizadores têm, segundo estes autores, permissão para efectuar deslocações, mas devem para isso seguir rotas previamente definidas. Contudo, mesmo quando os utilizadores seguem essas rotas, existe alguma incerteza acerca da localização exacta do utilizador. Com as rotas predefinidas, consegue-se saber qual a região na qual o utilizador se encontra, mas não a sua localização exacta. De referir, que estes autores apresentam ainda nesse trabalho algumas funções de densidade que pretendem determinar a localização de cada uma das estações.

Assim, devem existir mecanismos que permitam a gestão da localização das estações. Podem ser usadas diversas estratégias para a localização de estações. Contudo esta procura acarreta alguns custos adicionais que, como tal, devem ser tomados em consideração aquando da fase de optimização das *queries*. Além disto, quando se faz referência a um dada estação pode-se pretender aceder: (1) à sua localização; ou (2) aos dados armazenados localmente. Todavia, só é possível aceder aos dados armazenados numa estação móvel quando se conhece concretamente a sua localização, não sendo assim possível o uso de informação incerta.

Em [Kottkamp & Zukunft 1998] é apresentado um esquema de optimização de *queries* que, para além dos custos de localização e factores internos (como a estrutura dos dados e metadados), tem em conta factores externos à base de dados que podem influenciar o desempenho do processamento do sistema. Dentro destes factores, devem-se realçar os seguintes:

- **Hardware.** Para além do tempo e da velocidade de acesso ao disco, consideram-se ainda as limitações de memória e de energia.

- **Arquitectura.** Como se trata de uma arquitectura distribuída deve ter-se em conta a replicação e a partição dos dados.

- **Comunicações sem fios.** É importante que se tenham em conta as características e limitações que este tipo de comunicações possui.

- **Mobilidade.** Neste tipo de ambientes os utilizadores possuem alguma mobilidade, podendo não se conhecer a sua localização exacta. Contudo, podem existir *queries* cujo processamento obrigue a essa localização.

- **Preferências de utilizador.** Como as estações móveis possuem à partida menos recursos do que as estações fixas, o sistema pode dar preferência ao processamento de *queries* dos utilizadores móveis, para que estes consigam obter uma performance idêntica à dos utilizadores fixos.

Uma nova tarefa que este modelo de optimização possui é a de ser capaz de encontrar a estação que deve executar a *query*. Isto porque as características dos SBDMs não permitem que se defina à priori quais as estações que vão executar as *queries*. Assim, durante o processo de optimização, é necessário que se determine qual ou quais as estações que irão executar cada uma das diferentes fases do processamento de *queries*. Dadas as características destes ambientes, esta decisão deve ser flexível e revista sempre que seja necessário.

[Kottkamp & Zukunft 1998] dividem o processamento de *queries* móveis em três fases: tradução, optimização e execução. É possível que cada uma destas fases seja executada em diferentes estações. Todavia, existem estações que não são capazes de executar algumas destas fases.

Durante a fase de optimização é determinado um plano considerado óptimo. Para tal usa-se informação do sistema que no momento está a executar essa fase. Essa informação deve ser passada às estações que executem as restantes fases. Para evitar que haja uma grande troca de

mensagens entre a estação que processa a fase de optimização e a estação que processa a fase de execução, neste modelo propõe-se que essas duas fases sejam executadas pela mesma estação.

Na Figura 20 encontram-se esquematizadas três possíveis estratégias de processamento de uma *query* num ambiente móvel. A estratégia apresentada no caso ① é a que deve ser seguida quando os recursos são escassos e a *query* deve ser enviada para o servidor. Neste caso, é o servidor que deverá efectuar quer a fase de tradução quer as fases de optimização e de execução, para que os recursos da estação móvel sejam poupados. Quando a estação possui recursos suficientes para o processamento dessas operações, podemos utilizar duas estratégias: a fase de tradução pode ser iniciada na estação móvel e se se concluir que o processamento vai consumir recursos que a estação não possui, então esta fase é abortada e a *query* é enviada para o servidor, que efectua também as fases de optimização e execução – ver situação ② da Figura 20. Por outro lado, se da análise da fase de tradução se se concluir que a estação móvel possui recursos para prosseguir o processamento da *query,* então a estação móvel realiza as restantes fases do processamento, esta estratégia corresponde ao caso ③ da referida figura. A estação móvel também pode recorrer ao uso do Caso ③ quando a estação se encontra desconectada e se pretende executar uma *query*.

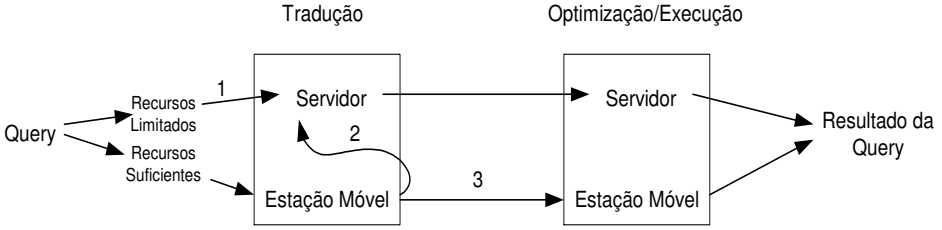

Figura 20 – Estratégias de Processamento *Queries* Possíveis.

A avaliação de custos e de recursos requeridos para a execução de *queries*, também são tidos em conta no plano de optimização de uma *query*. Os principais factores usados no esquema de optimização apresentado são o uso de CPU e o número de acessos ao disco, bem como consumos de energia, memória disponível e velocidade de CPU. Em [Kottkamp & Zukunft 1998] apresentam-se algumas fórmulas para o cálculo destes custos.

6.2. *Queries* Dependentes da Localização

Quando a localização da estação não afecta o resultado das *queries*, o sistema não necessita de saber a sua localização exacta. O seu comportamento revela-se como o descrito em [Imielinski & Badrinath 1993A]: "*Do not tell me – if I do not want to know*". Deste modo, existe sempre algum grau de ignorância acerca da localização e do estado de algumas estações. Todavia, deve estabelecer-se um limite para este nível de ignorância, de modo a que a suposta localização não se afaste demasiado da realidade. Para tal, sempre que necessário, podem ser usados algoritmos de localização.

Quando o processamento de *queries* exige o conhecimento da localização da estação, então o sistema possui um comportamento do tipo "*I do not know but can find out*" [Imielinski & Badrinath 1993A], ou seja, o sistema não sabe qual a localização da estação, mas pode proceder à sua localização. Esta localização deve ser feita com os menores custos possíveis. Além disso, as restrições de localização podem possuir vários graus de complexidade, por exemplo: (1) encontrar X; ou (2) encontrar X perto de Y; ou ainda, (3) encontrar X entre Y e Z.

Em [Imielinski & Badrinath 1993A] é também apresentada uma estratégia para tentar processar *queries* dependentes da localização. De acordo com essa estratégia, primeiramente procuram-se todos objectos que satisfaçam a restrição individual e só de seguida se efectua a sua localização. Por último, verifica-se quais os objectos que satisfazem as restrições de localização. Assim, por exemplo, se se pretender encontrar hoje, às 22 horas, todas as farmácias de serviço na região onde a estação se encontra, pelo esquema aqui proposto, primeiro tenta-se localizar todas as farmácias que estão de serviço hoje às 22 horas e só de seguida se verifica quais dessas farmácias se situam próximas da localização do utilizador. Este modelo fornece um esquema de processamento de *queries* dependentes da localização. Como vimos, primeiramente é efectuada uma procura global no sistema e só de seguida se restringe a localização. Contudo, quando o resultado da procura global, antes de se restringir a localização, é demasiado grande, este processo torna-se ineficiente, pois terá de efectuar a localização de muitos objectos.

6.3. *Queries* Contínuas

Nos ambientes móveis permite-se o processamento de *queries* contínuas, que conjugado com a existência de informação dependente da localização e do instante (em que é a *query* é executada), pode dificultar o processamento. Por exemplo, quando um utilizador se encontra em movimento e vai questionando continuamente o sistema acerca das farmácias de serviço disponíveis, é evidente que a resposta a esta *query* é influenciada não só pelo movimento da estação como também pela alteração temporal, ou seja, a resposta à *query* é uma função de tempo e de localização.

Em [Gök 1999] e [Gök & Ulusoy] é sugerido um esquema de processamento de *queries* contínuas, geralmente dependentes da localização. Neste esquema, a resposta a uma *query* consiste num conjunto de tuplos do tipo <R, I, F>, onde R corresponde à resposta da *query* entre o tempo de início I e o tempo final F. Aqui, depois das respostas serem processadas, deve-se decidir a altura em que essas respostas serão transmitidas à estação móvel. Para tal, existem cinco estratégias que permitem determinar a altura da transmissão do conjunto de respostas: transmissão imediata, transmissão atrasada, transmissão periódica, transmissão periódica adaptativa ou transmissão mista.

Na estratégia de transmissão imediata são enviados todos os tuplos, que pertençam ao conjunto de resposta da *query* contínua, assim que o processamento termine. Esta estratégia possui a vantagem de reduzir o *overhead* das mensagens de controlo, pois os todos tuplos são colocados numa única mensagem. Além disto, se a estação móvel se desconectar, depois de ter recebido um conjunto de respostas, já o possui na sua totalidade. Por outro lado, sempre que haja uma alteração no tuplo de resposta, este ser-lhe-á retransmitido.

Com a estratégia transmissão atrasada, quando se tem um tuplo de resposta <R, I, F> a transmissão efectuar-se-á na altura I. Aqui, ao contrário da estratégia anterior, existe uma mensagem de controlo para cada tuplo gerado. Tal facto, pode causar um grande *overhead* das mensagens de controlo. Além disto, se ocorrer uma desconexão depois da estação móvel ter recebido um tuplo, não significa que essa estação possua todo o conjunto de respostas. Contudo, neste esquema, a probabilidade de retransmissão é menor que no esquema anterior.

A transmissão periódica encontra-se entre as duas estratégias anteriores. Aqui, todos os tuplos de resposta <R, I, F>, para todas as unidades de tempo *w*, que satisfaçam a condição $t \leq I < t + w$, onde *t* é tempo actual, são transmitidos para a estação móvel. O valor *w* designa-se por tamanho

da janela, o qual especifica o intervalo de tempo, contendo I dos tuplos que irão ser transmitidos. Neste esquema, as mensagens de controlo são menores do que no esquema transmissão atrasada mas são maiores do que no esquema transmissão imediata. Além disto, quanto maior for *w*, menor é o *overhead* das mensagens de controlo. Tal como com o uso de transmissão atrasada, se ocorrer uma desconexão depois da estação móvel ter recebido alguns tuplos de resposta, não significa que os tenha recebido todos. Em relação à probabilidade de retransmissão, é maior do que na transmissão atrasada mas menor do que na transmissão imediata, e a probabilidade de retransmissão é tanto menor quanto menor for *w*.

De um modo idêntico à estratégia de transmissão periódica, também a transmissão periódica adaptativa recorre ao uso de *w*. Contudo, enquanto que na estratégia anterior este valor é constante, aqui faz-se uma adaptação dinâmica do seu valor. Nesta estratégia, *w* possui ainda um período de avaliação, no qual se efectua a análise da informação acerca do *overhead* das mensagens de controlo e das retransmissões, sendo ainda calculada uma razão entre o *overhead* das mensagens de controlo e o da retransmissão. Esta razão é usada como medidor de performance, para tal efectua-se a comparação da razão de *overhead* de dois períodos consecutivos.

Por último, a estratégia de transmissão mista também utiliza *w*. Na estratégia de transmissão periódica adaptativa usa-se um valor de *w* que pode ser adaptado, contudo é o mesmo para toda a base de dados, porém esta pode consistir numa mistura de objectos onde alguns se alteram frequentemente e outros não. Na estratégia de transmissão mista, permite-se que *w* se adapte de uns períodos de avaliação para outros. Para tal, a base de dados é dividida em dois conjuntos disjuntos: objectos que se alteram frequentemente, designados de quentes, e objectos que quase não se alteram, designados de frios. Quando os tuplos de respostas se referem a objectos frios é então usada a técnica de transmissão imediata, por outro lado quando se referem a objectos quentes calcula-se a razão de *overhead* de um modo idêntico à técnica de transmissão periódica adaptativa e decide-se, ou não, adaptar *w* ao próximo período de avaliação. Deste modo, para objectos frios consegue-se minimizar o *overhead* das mensagens de controlo e de retransmissão pois estas quase não são usadas. Enquanto que para os objectos quentes se efectua uma adaptação constante de *w* de modo a que se minimizem tais *overheads*.

6.4. Processamento de *Queries* em Ambientes Móveis Ad-hoc

O esquema de processamento *queries* proposto nesta secção considera a existência de um ambiente móvel Ad-Hoc, tal como definido anteriormente na Secção 2.1. Neste esquema cada estação móvel é autónoma e independente, e funciona tanto como fornecedor como consumidor de informação.

Dadas as características deste tipo de ambiente, pode concluir-se que aqui o processamento de *queries* tem de ter em consideração aspectos como:

- As origens dos dados disponíveis variam com a localização e com o tempo. Para além disto, como as entidades que integram o sistema se podem mover, o conjunto de estações vizinhas também se pode alterar. Deste modo, o resultado de uma *query* pode ser diferente de um local para outro, acontecendo mesmo por vezes que, num determinado local, o sistema não consiga fornecer uma resposta a uma *query* específica.

- A estação que pediu a *query* não pode depender de uma entidade global para fazer o encaminhamento da *query* para uma localização apropriada. A única informação garantida é aquela que a estação possui armazenada localmente.

- Uma *query* pode ser implícita ou explícita. No esquema aqui apresentado são suportadas tanto as *queries* implícitas como as explícitas. Deste modo, podem ser executadas algumas acções sem intervenção do utilizador.

- Os esquemas de transformação de *queries* não podem ser efectuados previamente, pois à priori não se conhecem as origens dos dados. Além disto, como as capacidades de algumas das entidades deste tipo de ambientes são bastante reduzidas, não se pode fazer uma transformação que consuma muitos recursos.

- Não se pode garantir cooperação entre as origens da informação, já que entre as estações podem ocorrer situações como: (1) a estação possui informação de confiança e não a quer transmitir para outras entidades; (2) a estação encontra-se disponível para fornecer informação, mas esta não é de confiança; ou (3) a estação disponibiliza informação para outras estações através de restrições de segurança.

Deste modo, para tentar ultrapassar todas estas limitações do processamento de *queries* neste tipo de ambientes, [Perich et al. 2001] propuseram um esquema que é suportado, basicamente, por instâncias de dois tipos de componentes: gestor de informação e fornecedor de informação. Assim, todas as estações móveis ou todas as entidades participantes no sistema devem possuir as seguintes características:

- Devem gerar um subconjunto conhecido do repositório de informação – que pode ser vazio. Este subconjunto serve para fornecer dados às próprias estações que o contêm, bem como a outras eventuais estações. Pode acontecer que este repositório seja inconsistente com o conhecimento de outras estações.

- Todas as entidades devem possuir um gestor de informação que funciona como um repositório local de metadados. Este deve incluir as definições de esquema para os fornecedores de dados locais, assim como factos particulares, como *queries* e respostas aos fornecedores de informação, locais ou não.

Neste esquema diz-se que uma entidade é um fornecedor de informação, quando é capaz de receber uma *query* e processar uma sua resposta, baseada na informação que possui. Por sua vez, o gestor de informação actua com outras entidades vizinhas, daí que possa ser dividido em quatro categorias:

- O gestor de informação apenas pode conter a informação requerida acerca dos fornecedores de informação locais. Todas as entidades do sistema têm de possuir, pelo menos, esta implementação, que é a mais simples de todas. Quando as estações possuem recursos limitados esta é a solução mais aconselhada.

- O gestor de informação, para além da informação acerca do fornecedor de informação, pode decidir armazenar informações adicionais que considere que serão úteis em futuras *queries*. Contudo, essa informação só se encontra disponível para a própria entidade.

- Numa outra categoria, o gestor de informação, para além de armazenar informação acerca do fornecedor de informação e das respostas de *queries* recentes, pode ainda decidir armazenar informação difundida pelas entidades vizinhas. Esta implementação do gestor de informação fornece um processamento de *queries* mais eficiente do que as

anteriores, contudo não pode ser implementado em entidades que possuam poucos recursos.

- A implementação do gestor de informação permite que este torne a sua informação disponível para as entidades vizinhas. Geralmente, faz-se esta implementação quando se prevê que a entidade se vai encontrar na mesma localização durante um longo período de tempo.

O esquema de processamento de *queries* aqui apresentado tenta disponibilizar um processamento eficiente para ambientes móveis ad-hoc. Contudo, quando as capacidades das estações são limitadas e as *queries* bastante complexas, pode ser pouco eficiente. O esquema foi desenhado tendo em conta *queries* mais simples, isto é, que possam ser executadas usando a informação local ou das entidades suas vizinhas. As *queries* complexas apenas podem ser processadas através de uma análise das relações da informação armazenada.

6.5. Outros Modelos

Na Secção 2.1 foram apresentados algumas arquitecturas para ambientes móveis, algumas das quais fornecendo diversas facilidades para o processamento de *queries*, como é o caso da arquitectura de um sistema móvel definida por [Madria et al. 1998]. Neste sistema cada estação móvel possui um sumário da base de dados sobre o qual são processadas as *queries*, o que permite a cada uma delas processar uma *query* usando os referidos sumários, retornando valores exactos ou bastante aproximados.

Nessa mesma secção foi também apresentado o sistema definido por [Bukhres et al. 1997], no qual se considera que cada estação móvel deve possuir uma caixa de correio, que servirá de "recipiente" para todas as mensagens, bem como para os resultados da *query*. Todo o processamento de *queries,* neste ambiente, é efectuado através destas caixas de correio, que por estarem sempre disponíveis, funcionam como um repositório central no qual se guardam todas as respostas a *queries* e *logs* de cada ESM.

Outra possível arquitectura para os SBDMs aí apresentada foi a defendida por [Lauzac & Chrysanthis 1998]. Neste caso, as estações móveis possuem vistas locais dos dados e o

processamento de *queries* é executado a partir das vistas existentes. Desta forma consegue-se evitar algumas comunicações remotas com as estações fixas para a obtenção de respostas às *queries*.

Capítulo 7

Gestão da Consistência dos Dados

Num SBDM a utilização de modelos de replicação de dados é indispensável para garantir alguma autonomia às estações móveis e contribuir para o aumento do seu próprio desempenho. Nestes sistemas, permite-se ainda que as estações móveis executem transacções locais ou globais, bem como *queries*, quer sobre os dados armazenados localmente, quer sobre os dados disponíveis noutras estações. Contudo, esta replicação pode originar algumas inconsistências, uma vez que os itens de dados se encontram guardados redundantemente em algumas estações do sistema. Estes problemas tornam-se ainda mais evidentes quando se trata das estações móveis, isto porque estas podem encontrar-se a operar em modo desconectado. Deste modo, devem existir mecanismos de sincronização que tentem evitar a existência de inconsistências entre os conteúdos das diversas réplicas. Contudo, esses mecanismos devem ter em conta que as réplicas das estações móveis podem não estar disponíveis na altura da sincronização.

Alguns dos modelos de replicação, gestão de transacções e processamento de *queries*, apresentados nos capítulos anteriores, já incluem algum nível de gestão da consistência dos dados. Alguns desses modelos permitem a existência de diferentes níveis de consistência, podendo variar de um nível mais restrito, quando a estação se encontra totalmente conectada, a

um nível mais fraco, quando se encontra completamente desconectado. Deste modo, é possível que diferentes graus de conexão possuam diferentes graus de consistência, ou até mesmo, que seja o próprio utilizador a definir o grau de consistência que pretende para um determinado item de dados. Existem, no entanto, outras soluções que usam esquemas mais pessimistas, onde se requer o bloqueio dos itens de dados durante todo o período de desconexão.

Neste capítulo são apresentados alguns métodos de gestão de consistência, incluindo os modelos de replicação, gestão de transacções e processamento de *queries*, referidos anteriormente, como também outras técnicas de detecção e reconciliação de conflitos para este tipo de sistemas.

7.1. Consistência versus Replicação

No capítulo 4 desta dissertação apresentaram-se alguns modelos de replicação de dados para os SBDMs, tendo sido expostas as suas técnicas de gestão e criação de réplicas. Nessa altura, fez-se ainda referência ao facto da consistência, disponibilidade e da economia de custos se encontrarem directamente relacionados e da impossibilidade de se conseguir obter todos estes três objectivos em simultâneo. Aliás, quando se escolhe como principais objectivos a disponibilidade dos dados e a economia dos custos, então a consistência de dados é, obrigatoriamente, relaxada, sendo então mais provável a ocorrência de inconsistências. Deste modo, devem existir mecanismos que detectem as situações de inconsistência e que sejam capazes também de recuperar, quando necessário, o sistema para um estado consistente. No fundo, o problema da gestão da consistência das réplicas num SBDM pode ser dividido nos seguintes aspectos:

- Distribuição das actualizações ao longo das réplicas.
- Determinação da ordem de aplicação das actualizações.
- Detecção e reconciliação de conflitos ao longo das actualizações.

Nesta secção apresenta-se o modo como o modelo de replicação optimista gere as suas réplicas para que a se mantenha consistência de dados.

Quando se usam modelos de replicação pessimistas não se levanta normalmente o problema da consistência, visto que estes modelos só possibilitam a actualização de uma réplica, quando é possível efectuar, simultaneamente, a actualização de todas as restantes réplicas. Deste modo, garante-se que todas as réplicas de um item de dados possuem o mesmo valor.

Os modelos de replicação optimistas [Saito 2000] [Saito & Saphiro 2002], ao contrário dos pessimistas, assumem que o número de conflitos resultantes da execução simultânea das operações de leitura e de escrita é baixo. Por este motivo, nestes modelos, permite-se que os utilizadores actualizem a réplica de um determinado item de dados, fazendo-se, posteriormente, a propagação e sincronização de tais alterações com as restantes réplicas.

Como já foi referido na Secção 4.2, os modelos de replicação optimistas têm como principais objectivos fornecer réplicas de dados suficientemente actualizadas e minimizar as perturbações do utilizador causadas por conflitos entre as diversas réplicas existentes. Para tal, tentam obter, entre outros aspectos, uniformidade, ou seja, tentam fazer com que todas as réplicas de um item de dados convirjam para um mesmo valor. Para que se consigam obter tais objectivos, a replicação optimista divide-se nos seguintes passos (Figura 21):

- **Propagação de actualização.** Durante este passo é efectuada a acumulação de todas as alterações executadas localmente durante os períodos de desconexão. Assim que se restabeleça a comunicação, essas alterações devem ser propagadas para as restantes estações. É desejável que esta propagação seja epidémica, ou seja, que qualquer estação que comunique com outra estação transfira tanto as alterações que foram processadas localmente como as alterações que foram recebidas de outras estações.

- **Escalonamento** (*Scheduling*)**.** Como as estações podem receber as actualizações por diferentes sequências, devem ser definidas políticas de ordenação de actualizações. Os algoritmos de *scheduling* têm dois objectivos: aplicar as actualizações rapidamente, para reduzir a latência e aumentar a concorrência, e manter uma ordem causal, para evitar conflitos e a perda de actualizações.

- **Detecção e resolução de conflitos.** Ao longo deste passo tentam-se detectar todos os conflitos existentes. Sempre que é detectado um conflito, deve proceder-se à sua resolução, substituindo as alterações conflituosas por outras que não o sejam. Algumas políticas de resolução de conflitos utilizam a norma de que “a última escrita ganha”, ou seja, é sempre escolhida a actualização mais recente. Contudo, as políticas mais

avançadas tentam usar uma união semântica das actualizações mais recentes e das mais antigas. Aqui, deve-se, ainda, ter em conta que quando se detecta um conflito alguns dos utilizadores podem não se encontrar disponíveis.

- ***Commitment.*** Neste passo, são definidos os mecanismos que fazem com que as estações concordem tanto no passo do escalonamento como nos resultados da resolução de conflitos. Deste modo, as actualizações que fizeram *commit* podem, e devem, ser aplicadas a todos os objectos.

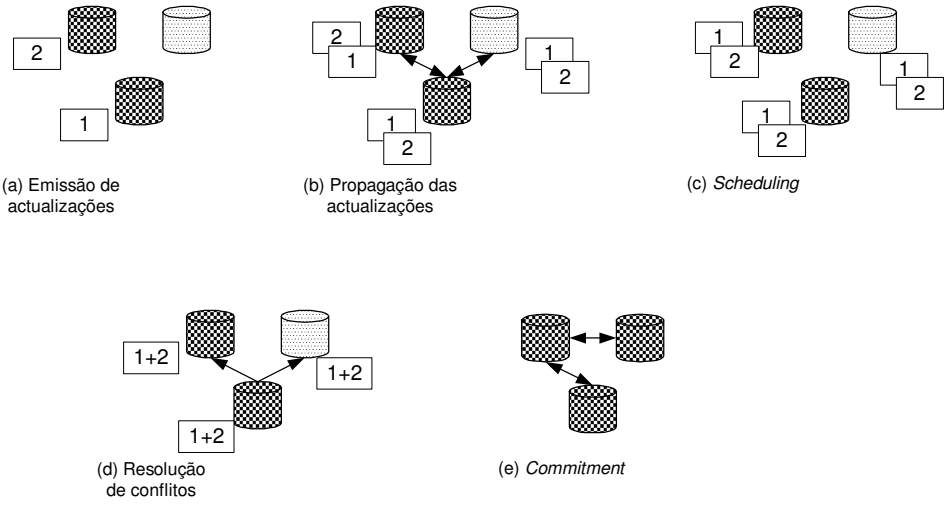

Figura 21 – Passos executados pelos algoritmos de replicação optimista.

Quando se considera um sistema no qual é usado o modelo de replicação optimista multi-mestre com transferência de estado, a gestão de consistência das réplicas processa-se da seguinte forma:

- Em alguns sistemas as estações ficam à espera das actualizações ("puxam" as actualizações), acreditando na informação periódica. Contudo, existem sistemas, nos quais as estações empurram as actualizações, isto é, sempre que uma estação altera uma réplica deve enviar as alterações às restantes réplicas.

- Duas estações que se encontrem a trocar informações devem averiguar se os conteúdos das réplicas diferem ou não, verificando também quais os itens de dados que foram actualizados de forma concorrente.

- No final do processo, as estações devem proceder ao envio do estado resolvido da réplica, para as restantes réplicas.

Por sua vez, quando se trata de um sistema onde é usado o modelo de replicação optimista com transferência de operação, a gestão de consistência das réplicas é bastante mais complexa. Pois, aqui, para se conseguir obter uniformidade as réplicas têm obrigatoriamente de concordar no conjunto e na ordem. Assim, o processo de gestão de consistência efectua-se basicamente do seguinte modo:

- As réplicas partem de um estado inicial comum.

- Cada réplica mestre mantém um *log* com as operações de actualização guardadas localmente. Além disto, cada réplica, sendo mestre ou não, mantém um *multi-log*, que possui todos os *logs* das réplicas mestre.

- Uma réplica mestre aceita uma operação de actualização do utilizador local, aplica-a localmente e, de seguida, adiciona-a ao *log* e ao *multi-log*. As réplicas recebem também operações dos mestres remotos, devendo aplicá-las e adicioná-las ao *log* e *multi-log*.

- A ordem correcta de aplicação das actualizações, chamada de escalonamento, é conhecida pelas réplicas gradualmente e, geralmente, é diferente da ordem de recepção das actualizações. Quando uma estação adiciona um pedido ao *multi-log*, ele reexecuta o escalonamento, possivelmente desfazendo e refazendo algumas das operações já aplicadas à réplica.

- Opcionalmente, quando se encontram operações conflituosas, a réplica pode resolver os conflitos.

- Finalmente, quando todas as estações concordam com o escalonamento para um dado conjunto de operações é guardado o estado e são removidos do *multi-log* os registos correspondentes a estas operações. De seguida, repete-se o algoritmo desde o início.

7.2. Consistência versus Modelos de Transacções

Nos SBDMs, como em outros sistemas de base de dados, as transacções são usadas tanto para a extracção de informação do sistema como para a actualização de alguns itens de dados. No fundo, uma transacção pode ser definida como um conjunto de operações de leitura e de escrita. Algumas transacções móveis podem ser executadas sobre dados locais, possivelmente desactualizados. Isto porque podem ser executados durante os períodos de desconexão, nos quais não é possível ler as últimas alterações efectuadas nem tornar as escritas visíveis para outras réplicas.

Alguns dos modelos de transacções móveis apresentados ao longo do Capítulo 5 incluem um certo nível de gestão dos conteúdos dos itens de dados, na tentativa de se evitarem alguns conflitos e de se resolverem outros aquando da reconciliação das várias réplicas. Nesta secção apresenta-se essa gestão de consistência para alguns dos modelos de transacções referidos.

7.2.1. *Clustering*

No modelo de transacções móveis baseado em *clusters* [Pitoura & Bhargava 1995] [Pitoura & Bhargava 1997] [Pitoura 1996], apresentado na Secção 5.2, considera-se que uma base de dados é um conjunto de itens de dados. Aí, ao agrupamento dos vários itens de dados deu-se ainda o nome de *clusters*, ficando assim a base de dados dividida num conjunto de *clusters*.

[Pitoura & Bhargava 1995], [Pitoura & Bhargava 1997] e [Pitoura 1996] consideram que o estado de um *cluster* é consistente se e só se se verificam todas as regras de integridade internas. Além disto, consideram ainda que um *cluster* é m-consistente se todos os estados do *cluster* forem consistentes e se todas as restrições de integridade *inter-cluster* também forem m-graus consistentes. A definição de m-grau de consistência para uma dada restrição de integridade depende do tipo de restrições *inter-cluster*. Deste modo, o estado de uma base de dados móvel é consistente se todas as regras de integridade *intra-cluster* são satisfeitas e se todas as regras de integridade inter-cluster possuírem inconsistências limitadas. Para os dados guardados, a relação de m-grau pode exprimir a divergência do valor de uma cópia secundária para o valor da cópia primária. Neste caso, o grau de desvio permitido deve ser limitado por uma das seguintes hipóteses:

- Um valor máximo de desvio permitido.
- Um número de transacções que podem operar sobre cópias inconsistentes.
- Um número de versões permitidas.
- Um número de item de dados que podem divergir.

Com o uso deste modelo, sempre que não haja possibilidade de operar sobre dados com consistência estrita, permite-se que os utilizadores operem sobre dados m-consistentes. Contudo, as operações executadas sobre esses dados dizem-se fracas, enquanto que as operações sobre os dados com consistência estrita se dizem fortes. De modo idêntico, também existem as transacções fortes e as transacções fracas. Perante este cenário, pode-se concluir que cada item de dados possui duas versões uma forte e uma fraca, dependendo do tipo de operação ou transacção que a criou.

Quando se juntam as operações de uma transacção fraca com as de uma forte, há uma grande possibilidade de ocorrência de conflitos. Duas operações dizem-se conflituosas sempre que se encontrem a aceder ao mesmo item de dados e uma dessas operações é uma operação de escrita. Na Tabela 3 apresentam-se os possíveis conflitos causados pela execução de duas transacções distintas sobre o mesmo item de dados. Nesta tabela, as entradas assinaladas com X indicam as situações onde é provável a ocorrência de conflitos entre operações [Pitoura & Bhargava 1995].

		Operações da Transacção 2			
		Leitura fraca	**Leitura Forte**	**Escrita Fraca**	**Escrita Forte**
Operações da Transacção 1	**Leitura Fraca**			**X**	**X**
	Leitura Forte				**X**
	Escrita Fraca	**X**	**X**	**X**	**X**
	Escrita Forte	**X**	**X**	**X**	**X**

Tabela 3 – Possíveis de Conflitos na Junção de Transacções.

Como já foi referido, num sistema que se use *clustering*, o grau de cada item de dados, dentro de um *cluster*, expressa a divergência entre o valor da versão local e o valor da versão forte. Esta

diferença de valores pode resultar tanto de *uncommits* globais de escritas fracas, como de actualizações de versões fortes que ainda não foram processadas nesse *cluster*.

Contudo, a junção de todos os *clusters* deve resultar na consistência total. Deste modo, a reconciliação de valores de cópias diferentes, do mesmo item de dados localizado em *clusters* distintos, deve ser consistente. Existem diversas soluções para este problema de reconciliação, variando desde a sintaxe pura à semântica pura. Os autores deste modelo optam pelas soluções baseadas na sintaxe pura, para tal aceitam-se tantas escritas fracas quanto possível sem que haja violação da seriação das transacções fortes.

Para resolver os conflitos nas listas de transacções *inter-clusters* pode-se fazer o *roolback* das transacções que possuem operações de escrita fracas em conflito com as transacções fortes. Porém, da anulação de transacções podem resultar *cascading aborts*, isto porque pode causar o aborto de transacções que tenham lido os valores escritos por essa transacção abortada. No entanto, apenas as transacções fracas, no mesmo *cluster*, podem necessitar de ser abortadas para resolver conflitos na lista de transacções *inter-clusters*, uma vez que apenas estas podem ler e escrever em transacções fracas. Além disto, para a maioria das aplicações, o aborto de transacções fracas não altera a semântica de outras transacções fracas que tenham lido os seus valores. Isto deve-se ao facto de que as transacções fracas apenas necessitam de valores aproximados dos itens de dados. Assim, uma transacção que leu valores produzidos por uma transacção que será abortada, só será abortada se os valores não se encontrarem dentro dos limites estabelecidos como aceitáveis.

Em conclusão, o modelo de transacções baseado em *clustering* gere a consistência dos dados permitindo a existência de diferentes níveis de consistência nos diversos *clusters*. Porém, quando se une toda a informação a consistência deve ser total. Nesta altura, é usado um esquema puramente sintáctico, aceitando-se tantas escritas fracas quantas forem possíveis sem que haja violação das restrições de integridade. Para resolver conflitos de dados entre os diversos *clusters* devem desfazer-se as transacções fracas que entram em conflito com as fortes.

7.2.2. Transacções Baseadas na Semântica

Em [Walborn & Chrysanthis 1995] apresenta-se um modelo de transacções móveis que se baseia na semântica das operações, dos objectos e das aplicações. Com o uso deste modelo, as estações móveis podem armazenar fragmentos dos objectos de dados, para que possam executar

as suas transacções localmente. Aqui, o problema de consistência coloca-se quando todos os fragmentos são reunidos de modo a que se reconstrua o objecto de dados. Para suportar consistência, os objectos devem possuir cuidadosas operações de fragmentação e de junção.

Os requerimentos de consistência dos dados variam desde a consistência estrita (como a definida pela seriação) à consistência eventual. A consistência eventual denota uma divergência espacial ou temporal da consistência estrita, podendo uma sua extensão ser expressa, tal como no modelo anterior, em termos de graus de inconsistência.

Quando uma estação invoca a fragmentação de um determinado objecto de dados, as condições de consistência fazem parte dos parâmetros da invocação. Estas condições definem as restrições que devem ser satisfeitas para que se mantenha a consistência do objecto completo, e podem incluir:

- Operações permitidas e restrições aos seus valores de entrada.
- Condições ao estado do objecto.

Deste modo, pode-se restringir, ou até mesmo proibir, a execução de operações sobre alguns fragmentos para se garantir que estes serão reunidos apropriadamente. Frequentemente, as restrições associadas aos fragmentos estão relacionadas com restrições sobre o objecto completo.

7.2.3. Transacções em Sistemas com Replicação a Dois Níveis

Como já foi apresentado na Secção 5.4 o uso do modelo de replicação a dois níveis [Gray et al. 1996] origina a existência de dois tipos de dados – tentativos e base – assim como dois tipos de transacções – tentativas e base. Os dados tentativos representam os dados provisórios e as transacções tentativas, por sua vez, são aquelas que operam sobre esses mesmos dados.

A execução de uma transacção tentativa actualiza apenas os dados locais e não produz alterações definitivas. Ao iniciar-se uma transacção tentativa origina-se uma transacção base, que será executada, posteriormente, na ESM quando ocorrer a reconexão da estação, de modo a que as actualizações locais tenham a possibilidade de se tornarem definitivas e globais.

Quando ocorre uma falha, por violação da consistência ou dos critérios de aceitação, numa transacção base enviada pela estação móvel à ESM que a está a executar, deve ser enviada uma notificação à estação móvel correspondente, indicando que a transacção falhou e quais os motivos dessa falha. No final da execução as versões tentativas dos dados são descartadas, pois a estação móvel tem a possibilidade de actualizar os seus valores locais através dos valores dos objectos mestres.

7.2.4. PRO-MOTION

O modelo de transacções móveis PRO-MOTION (Secção 5.5) [Mazumdar & Chrysanthis 1999] permite que as estações móveis efectuem pedidos de dados ao servidor. Tais dados serão entregues à estação móvel sob a forma de *Compactos*. Estes, para além dos dados pedidos, contêm informação de estado e de acesso aos dados.

Para tentar resolver os possíveis problemas de consistência, no modelo PRO-MOTION permite-se que para além dos itens de dados, também se repliquem algumas restrições. Por exemplo, se na base de dados fixa existe a restrição R então pode ser criada, a nível local, uma sua réplica R', servindo para a validação dos seus dados durante a desconexão. Deste modo, permite-se que os utilizadores executem as suas transacções com um certo nível de confiança, mesmo quando estão desconectados da rede fixa.

O PRO-MOTION baseia-se no conceito de localização, que consiste basicamente na substituição de restrições distribuídas *R* por restrições locais R_i, no nodo *i*, e no seu ajuste automático ao longo da actualização incremental. Algumas das propriedades mais significativas deste conceito são:

- Quando uma transacção local verifica a restrição R_i, num dado nodo *i*, então não há necessidade da verificação da restrição global, evitando-se assim alguns atrasos indesejáveis.

- Quando R'_i é mais restrita que R_i então a segunda pode ser substituída, localmente, pela primeira.

- A actualização incremental das condições de uma restrição pode ser feita num nodo numa altura possível, por uma determinada sequência. Não sendo necessárias

transacções distribuídas com longos protocolos de *commit*, nem mesmo controlo de concorrência distribuída.

Um aspecto importante que deve ser considerado quando se usa o PRO-MOTION, para que não resultem outras inconsistências, é que todas as réplicas de uma dada restrição devem possuir o mesmo valor. Daqui conclui-se que, não se pode fazer a actualização de uma restrição se existir alguma réplica sua, nalguma estação, que naquele momento não esteja conectado com o sistema, ou seja, só se pode alterar uma restrição quando se consegue efectuar essa alteração em todas as suas réplicas no mesmo instante.

7.2.5. Controlo de Concorrência Optimista

O modelo de transacções móveis baseado no controlo de concorrência optimista [Ahmed et al. 1996], referido em 5.8, recorre ao uso de isolamento de transacções, bem como a relatórios de invalidação e de actualização para facilitar o processamento de transacções locais. Foi ainda referido, que durante o processamento de uma transacção, esta passa por diversos estados (Figura 18). Contudo, também, aqui, o processamento de transacções pode gerar algumas inconsistências. Em [Ahmed et al. 1996] são apresentadas algumas soluções para a resolução de tais inconsistências.

Quando uma estação móvel dá início à execução de uma transacção de actualização, deve enviar o conjunto de operações de escrita e de leitura para a ESM, de modo a que seja possível efectuar uma validação remota. Neste modelo, a transacção possui apenas um ponto de *commit*, que é efectuado pelo seu coordenador. A validação da transacção é feita pela certificação de todas as operações que contém, tanto de leitura como de escrita, em todos os servidores. Se todas as operações forem válidas, então faz-se o *commit* de todas as operações de escrita, caso contrário a transacção é abortada.

Por outro lado, também para se assegurar a consistência dos itens de dados replicados, cada réplica possui ainda um identificador de versão. Este identificador contém uma etiqueta temporal, com a indicação da transacção que actualizou aquele item de dados. Por este motivo, este identificador é considerado um factor chave na fase de validação de uma transacção. Quando uma transacção manipula um item de dados desactualizado o seu processamento é abortado, sendo reiniciado logo de seguida, sem que haja qualquer tempo de espera.

Após o processo de validação, quando uma ESM verifica que uma transacção não é válida, deve então enviar um relatório de invalidação, para todas as estações da sua célula. Estes relatórios incluem todos os itens de dados que foram actualizados, desde o último relatório, bem como a invalidação de outros itens de dados guardados nas estações móveis. Contudo, os relatórios de invalidação não possuem os valores dos dados, mas sim os seus identificadores, de modo a economizar a largura de banda.

O modelo proposto pode ser facilmente adaptado, de forma a suportar consistência fraca, de modo idêntico ao que acontecia no modelo de *clustering*. Todavia, aceitar que se efectuem operações sobre dados pouco consistentes, implica que se aceite uma divergência nos itens de dados usados a partir do seu valor correcto. Assim, deve existir um critério de divergência, que se baseia no valor correcto do item de dados. Uma divergência de tempo pode ser facilmente mapeada para um valor de divergência de uma etiqueta temporal de uma operação de escrita. Esta etiqueta pode ser incluída nos relatórios de invalidação com um pequeno *overhead*. Aqui, a divergência pode ser registada de uma das seguintes formas:

- A ESM guarda o trajecto efectuado, por cada estação móvel, fazendo também o registo dos critérios de divergência.
- A estação móvel envia o critério de divergência de cada um dos seus itens de dados.

Logo que se recebe um item de dados através dos relatórios de invalidação, que invalidem uma transacção, a ESM deve ler esses dados de modo a verificar se a transacção é verdadeiramente inválida. Se o servidor mantiver o trajecto de subscrição da vista materializada, é suficiente para se enviar uma mensagem a ignorar o critério de divergência, que ainda não está a ser violado, o que diminui o *broadcast*. Se um item de dados for materializado, a estação móvel tem a opção de esperar pelo próximo relatório de actualização, que deve conter o valor do item de dados que se suspeita estar inválido.

7.2.6. HiCoMo

O modelo de transacções HiCoMo [Lee & Helal 2002], apresentado em 5.11, tem como principal objectivo maximizar, o mais possível, o número de *commits* das transacções. Este objectivo é conseguido através do relaxamento de algumas condições. Contudo, deve existir um certo

controlo sobre os dados actualizados pelas transacções, para que se evitem eventuais inconsistências.

Para manter a consistência dos dados, este modelo de transacções, tem por base o seguinte algoritmo:

- **Detecção de conflitos.** Primeiramente, devem ser detectados os conflitos entre uma transacção HiCoMo e outras transacções HiCoMo, que ainda não tenham sido transformadas em transacções base. Se se detectar um conflito aborta-se a transacção HiCoMo que está a ser considerada para a transformação. A simples estratégia de aborto deve-se ao facto de, neste modelo, se pretender fornecer durabilidade às transacções base, para além de que não existe uma forma de controlar o que se passa dentro de outras transacções base. A detecção de conflitos pode ser implementada por uma estratégia de controlo de concorrência optimista (Secção **5.8**). Com esta estratégia, uma transacção passa por três fases: execução, validação e actualização. Aqui, a fase de execução é processada quando as transacções HiCoMo executam actualizações nas tabelas agregadas na estação móvel. Os autores consideram que esta fase se inicia com a desconexão e acaba quando a estação se volta a conectar. Durante a fase de validação é verificada a existência, de eventuais, conflitos das actualizações locais com as actualizações efectuadas por outras transacções. Se as alterações da transacção HiCoMo forem mais recentes do que as das transacções base, então permite-se que as alterações sejam aplicadas às tabelas base, caso contrário são abortadas. Esta fase de validação começa quando a estação se reconecta à rede fixa. Por último, a fase de actualização consiste na transformação das transacções HiCoMo em transacções base, aplicando-as, de seguida, às tabelas base. Com o uso desta estratégia, as transacções ficam seriadas pela mesma ordem das etiquetas temporais.

- **Geração da transacção base inicial.** Se não existirem conflitos decide-se que tipo de transacção base deve ser criada. Esta decisão baseia-se em informação dada pela transacção HiCoMo, bem como por factores das funções de agregação e das tabelas. De seguida, as transacções são executadas como subtransacções de um modelo de transacções *nested* estendido.

- **Falha da subtransacção e geração da transacção base alternativa.** Algumas das subtransacções geradas podem abortar por violarem as restrições de integridade. Como estas situações são difíceis de prever antes da execução, as transacções abortadas

necessitam de ser renegociadas posteriormente. As subtransacções abortadas afectarão todos os resultados das transacções HiCoMo e precisam de ser compensadas de alguma forma. Se a diferença que as subtransacções abortadas provocam se encontrar dentro da margem de erro permitida, então a transacção pode terminar o seu processamento. Caso contrário, as actualizações das transacções abortadas devem ser redistribuídas e tentar uma nova execução. Se depois de algumas redistribuições a transacção continuar a abortar, então também se deve ter em conta a margem de erro nessa redistribuição. A transformação está completa quando as subtransacções conseguirem obter sucesso nalgum ponto, se não o conseguirem, a transacção HiCoMo deve ser abortada.

7.3. Detecção e Reconciliação de Conflitos

Em [Phatak & Badrinath 1996] e [Phatak & Badrinath 1999] é apresentado um método de resolução de conflitos para quando se faz a reconciliação de dados desconectados. A arquitectura considerada é uma extensão do modelo Cliente/Servidor. Neste método, as cópias primárias encontram-se no servidor e as transacções só são guardadas globalmente quando forem guardadas no servidor. Os clientes, por sua vez, têm permissão para replicar, localmente, um subconjunto do estado actual da base de dados. As transacções locais podem operar sobre estas réplicas e efectuar actualizações sobre elas. Quando se está ligado ao servidor é efectuado imediatamente o *commit* global, senão faz-se o *commit* local e os seus resultados ficam disponíveis, apenas, para as transacções locais. Estes resultados locais são propagados mais tarde para o servidor assim que se efectue a reconexão do cliente, sendo também nesta altura efectuado um teste de seriação global. Se a transacção não conseguir ser seriada, quer por causar conflitos quer por usar dados de outra transacção que não é seriável, então essa transacção deve ser desfeita, processando-se o seu *rollback*. Caso contrário, a transacção é guardada globalmente e as suas actualizações são efectuadas nos dados partilhados.

A reconciliação pode ser realizada quer ao nível dos itens de dados quer ao nível das transacções. No primeiro caso diz-se que é centrada nos dados, enquanto que no segundo se diz centrada nas transacções. A reconciliação centrada nos dados é a mais usada nas bases de dados comerciais.

No Exemplo 1 apresenta-se um caso de uma reconciliação de conflitos numa pequena base de dados.

Considere-se x = 1, e que esse item tem apenas uma versão (sistema de versão única). Suponhamos que o cliente executou a seguinte transacção x = x+1, e que o servidor executou outra transacção, x=2, ou seja,

Servidor

$E_0[x]=1, L_1[x]=1, E_1[x]=2$

Cliente

download $x=1, L_1[x]=1, E_1[x]=2$

onde as acções L e E representam operações de leitura e escrita, respectivamente. Quando houver reconciliação de dados, vai ser detectado um conflito, pois o cliente tem o valor de x com sendo 1, o que não é verdade pois no servidor já possui o valor 2. Usando a solução centrada nos dados para a reconciliação, têm de existir regras para negociar com os conflitos existentes. Sendo assim, o administrador, baseado nas regras de adição, deveria construir a seguinte regra:

$$x_{serv\_novo} = x_{serv\_velho} + x_{cli\_novo} - x_{cli\_downloaded}$$

ou seja,

$$x = 2 + 2 - 1$$

O x_{serv_novo} *é o valor a colocar no servidor depois da resolução do conflito,* x_{serv_velho} *é o valor actual no servidor, antes da resolução de conflitos,* x_{cli_novo} *é o valor actual no cliente e* $x_{cli\_downloaded}$ *é o valor de x que o cliente guardou.*

Assim usando a regra tem-se

$W_0[x] = 1, c_0, r_1[x]=1, W_1[x] = 2, c_1, W_c[x] = 3, T_c,$

onde T_c *é transacção criada para resolver os conflitos.*

Exemplo 1 – Reconciliação de conflitos numa pequena base de dados de versão única.

Esta solução tem a vantagem de ser barata e fácil de implementar. Além disso, o cliente não necessita de manter todas as transacções que executou, é suficiente guardar os valores actuais e os valores que carregou. Contudo, quando a semântica das transacções do cliente se altera, a regra criada pode deixar de ser válida, podendo levar a execuções erradas.

O problema torna-se quase intratável quando se usa a solução centrada nos dados, onde existem múltiplas transacções com diferentes semânticas. A situação pode ainda piorar se existirem várias transacções a acederem ao mesmo item de dados. Quando algumas actualizações da transacção são aceites e outras são rejeitadas, pode-se comprometer a atomicidade das transacções dos clientes. Uma solução, que é a seguida por muitas das bases de dados comerciais, é forçar que todas as actualizações a serem reconciliadas manualmente. Todavia, esta solução faz com que se perca capacidade da base de dados.

A reconciliação centrada nas transacções apresenta-se como uma solução para a resolução de alguns dos problemas referidos anteriormente. Com este tipo de reconciliação, as transacções dos clientes são reconciliadas como unidades, uma de cada vez. Assim, aceitam-se ou rejeitam-se transacções inteiras e não itens de dados. O cliente pode também especificar as semânticas das transacções na forma de rotinas ou funções de resolução de conflitos. No Exemplo 2 apresenta-se uma situação de resolução de conflitos usando o método centrado nas transacções.

Servidor

$E_0[x] = 1, c_0, L_1[x] = 1, E_1[x] = 2, c_1$

Cliente

$x=1, L_{1'}[x] = 1, E_{1'}[x=2*x] = 2, c_{1'}, L_{2'}[x] = 2, E_{2'}[x=x+1] = 3, c_{1'}$ *reconcilie*

Para reconciliar a primeira transacção, no servidor tinha-se

Servidor

$E_0[x] = 1, c_0, L_1[x] = 1, E_1[x] = 2, c_1, L'_{1'}[x] = 2, E'_{1'}[x = 2*x] = 4, c'_{1'}$

Depois da reconciliação da segunda transacção tinha-se

Servidor

$E_0[x] = 1, c_0, L_1[x] = 1, E_1[x] = 2, c_1, L'_{1'}[x] = 2, E'_{1'}[x = 2*x] = 4, c'_{1'}, L'_{2'}[x] = 4,$
$E'_{2'}[x = x + 1] = 5, c'_{2'}$

Exemplo 2 – Reconciliação de uma base de dados usando o método centrado nas transacções.

É de notar que este método de refazer as transacções é semântico e não sintáctico. Esta solução também não tem em conta as diferentes semânticas das transacções. Além disto, no Exemplo 2, se as semânticas forem desconhecidas, deve-se rejeitar a transacção T_1, contudo a T_2 não é rejeitada pois lê x=2 que é igual ao valor actual do servidor, como se pode verificar através do Exemplo 3.

Servidor

$$E_0[x] = 1,\ c_0,\ L_1[x] = 1,\ E_1[x] = 2,\ c_1,\ L'_2[x] = 2,\ E'_2[x = 2*x] = 3,\ c'_2,$$

Exemplo 3 – Semântica vs. Sintaxe na resolução de conflitos.

Note-se ainda que, apenas os valores de x são usados e não a semântica da transacção. Esta propriedade é útil em sistemas móveis, onde as transacções devem ter prioridade sobre ligações fracas ou intermitentes, no caso do uso de comunicações sem fios. Aqui, como as dependências já não são relevantes, pode-se enviar as mensagens por qualquer ordem.

A reconciliação centrada nas transacções é consideravelmente mais poderosa do que a centrada nos dados. No entanto, necessita de mais trabalho por parte do cliente (pois tratam-se transacções inteiras) e os algoritmos de reconciliação são mais complexos.

Os ambientes móveis são mais susceptíveis a consistências fracas. Isto deve-se ao facto de se permitir que os clientes desconectados modifiquem localmente as suas réplicas. Contudo, quando se restabelece conexão, requer-se que estas actualizações se consigam de alguma forma seriar. Os modelos centrados nos dados confiam neste facto para garantir desempenho, isto porque assumem um conjunto fixo de semânticas para transacções de clientes que são codificadas em regras de reconciliação. Esta semântica fixa pode ser usada como um caminho secundário para garantir a seriação enquanto se mantém a consistência dos dados.

Por outro lado, todos os esforços devem ser feitos para fornecer uma seriação tão forte quanto se conseguir, com um desempenho razoável. Uma dessas garantias, que trabalha com sistemas de multiversão, é o *snapshot isolation*. Esta garantia é quase tão forte como as leituras de dados guardados, mas não é suficiente para nos permitir focar um conjunto de escrita de transacções de reconciliação de clientes. *Snapshot isolation* permite ver quais as transacções lidas num *snapshot* da base de dados e quais os itens de dados onde as transacções actuais escrevem, pois, se um item de dados é partilhado por duas transacções concorrentes, apenas uma das transacções lhe pode escrever. É de notar que a seriação adicional requer que apenas uma das transacções

actuais modifique um item de dados partilhado, assim o outro pode não escrever em qualquer item de dados partilhado.

Capítulo 8

Protecção dos Dados

8.1. Tolerância a Falhas e Recuperação

Os sistemas de base de dados, tal como quaisquer outros sistemas, estão sujeitos a falhas que os podem conduzir a um estado indesejável. Quando tal acontece, o sistema deve encontrar-se munido de mecanismos que lhe permitam recuperar para um estado que é reconhecido como correcto. Estes mecanismos são designados, usualmente, por mecanismos de tolerância a falhas ou de recuperação.

Mais especificamente, uma falha no sistema é um evento a partir do qual o sistema deixa de executar as suas tarefas de acordo com as suas especificações base. As falhas podem ser provocadas por variadíssimas coisas relacionadas com questões de hardware, software ou mesmo erros humanos. É, assim, necessária a existência de técnicas que detectem essas falhas e possibilitem "trazer" o sistema para um estado correcto [Verhofstad 1978]. [Date 2000] considera

que os mecanismos de recuperação da base de dados se baseiam em técnicas que usam essencialmente sistemas de redundância, isto é, a garantia de recuperação de um determinado item de dados é dada pela existência, algures no sistema, de uma cópia desse mesmo item de dados.

Se a tolerância a falhas e a recuperação são essenciais para os SBDDs, tornam-se ainda mais importantes para os SBDMs, já que as características ambientais dos SBDMs fazem com que estes fiquem mais vulneráveis. Além disso, estes sistemas tornam o processo de tolerância a falhas e recuperação bastante mais complexo. Uma estação móvel pode ficar indisponível por vários motivos, nomeadamente [Pradhan et al. 1996]:

- Falha da estação móvel.
- Desconexão da estação móvel.
- Falha na ligação sem fios.

Adicionalmente, [Singh & Cabillic 2003] consideram também que as falhas podem ser divididas, essencialmente, em três categorias:

- **Falhas de Software.** Referindo-se a falhas do software que se encontra a executar na estação móvel.
- **Falhas Ambientais.** Falhas ocorridas em componentes físicos do sistema, como por exemplo falhas de rede.
- **Falhas Arquitectuais.** Estas referem-se às falhas causadas pelos baixos recursos das estações móveis.

Como nos SBDMs as desconexões, planeadas ou não, são bastante frequentes, o sistema não deve as deve reconhecer e tratar como falhas. Nos SBDDs, Isto não era assim tratado, dado que os esquemas de tolerância a falhas para esses sistemas tratavam a falta de comunicação entre as estações como uma falha. Além disto, os esquemas tradicionais de tolerância a falhas, baseados em pontos de verificação e *logging* de mensagens, requerem um local estável para o seu armazenamento. Nos SBDMs, a maioria das vezes, o armazenamento nas estações fixas é estável, porém o mesmo não acontece com as móveis. Um candidato lógico para este

armazenamento parece ser a ESM da célula onde a estação móvel se encontra. Mesmo assim, os esquemas tradicionais são insuficientes, pois as estações móveis podem movimentar-se de uma células para as outras, não tendo, portanto, uma ESM fixa com a qual possam comunicar. A recuperação nos SBDMs também se torna complexa pois os pontos de verificação podem encontrar-se distribuídos por diversas ESMs [Pradhan et al. 1996]. Além disto, como os SBDMs podem ser encarados como uma extensão dos SBDDs, facilmente se conclui que alguns dos problemas que se colocavam aquando da execução dos processos de recuperação nos SBDDs também se colocam nos SBDMs, de referir:

- A consistência entre as diversas cópias da base de dados distribuída.
- As falhas nos nodos de rede que contêm uma cópia da base de dados.
- Os vários problemas arquitecturais e físicos que podiam ocorrer na rede.
- O *Commit* das transacções distribuídas.
- Os *Deadlocks* distribuídos.

Nos sistemas de base de dados é também necessário ter em conta a falha de transacções. Nos sistemas distribuídos garantia-se que estas possuíam as propriedades ACID, o que facilitava a sua recuperação. Contudo, nos SBDMs não se verificam estas propriedades, dado que as transacções móveis podem ser divididas num conjunto de operações que podem ser executadas em diferentes estações, havendo ainda a possibilidade de visualização dos estados intermédios da transacção. Assim, deve-se recorrer a outros métodos de recuperação para as transacções móveis.

Um dos métodos de recuperação mais utilizados em ambientes distribuídos é o *Two-Phase Commit,* que tenta reduzir os custos do processo de recuperação depois de uma falha. Este método consiste em duas fases uma de votação (*voting*) e outra de decisão. Durante a primeira fase, o coordenador das transacções pede, a todos os participantes da transacção em execução, que se preparem para guardar, enquanto que durante a fase de decisão o coordenador apenas decide fazer o *commit* das transacções, se todas as estações intervenientes estiverem preparadas para fazer o *commit*. Assim, se algum dos participantes decidir abortar, toda a transacção também deve ser abortada [Chrysanthis et al. 1998].

O processo de recuperação do protocolo *Two-Phase Commit* depende do tipo de falha ocorrida no sistema. Quando há uma falha no coordenador, depois deste recomeçar, apenas se deve ter em

atenção todas as transacções pendentes antes da falha, reconstruindo-se a sua tabela de transacções. Quando a falha ocorre no participante, então este deve verificar se possui alguma transacção em estado de *prepare-to-commit*, e deve ainda restabelecer a comunicação com os participantes.

Em [Gore & Ghosh 2000] estão apresentados alguns dos protocolos utilizados para a execução de transacções móveis e para a sua recuperação em caso de falha:

- **Protocolo de *Timeout*.** Este protocolo é executado pela ESM e o *timeout* tem um valor acordado entre a estação móvel e a ESM, antes do início da transacção. Depois de decorrido este tempo, a ESM pode iniciar o *rollback* dessa transacção. Este protocolo deve ainda recuperar o sistema de inconsistências causadas por longos períodos de inactividade da estação móvel.

- **Protocolo de desconectado.** Ao contrário do anterior, este protocolo é executado pela estação móvel, sendo usado para redefinir as restrições de tempo. Aqui, tal como no protocolo de *Timeout*, quando as desconexões ultrapassam o período de tempo estipulado, a estação móvel pode iniciar o processo de *rollback* da transacção.

- **Protocolo de *handoff*.** A estação móvel quando executa este protocolo pede à ESM actual que guarde o estado de uma dada transacção e informa, ainda, qual o endereço da nova ESM. Por outro lado, quando a estação móvel estabelecer a conexão com a nova ESM, deve informá-la acerca do endereço da ESM anterior. Assim, a informação da estação móvel e da transacção móvel fica disponível nas duas ESMs.

- **Protocolo de Migração.** Ao contrário do que acontece no protocolo anterior, aqui a informação é passada para a ESM anterior através da nova ESM, uma vez que nem sempre é possível estabelecer a ligação entre a estação móvel e a ESM anterior. Porém, essa informação deve chegar à ESM anterior antes do *timeout*.

Todavia, a recuperação dos dados também pode ser protegida de falhas, através de outros processos de recuperação, ou seja, deve ser possível a recuperação dos próprios processos de recuperação. Facilmente se conclui que, este processo se pode tornar infinito, daí que se tenha de definir um certo critério de confiança nos dados.

8.1.1. Estratégias para Guardar o Estado

Nos sistemas tradicionais são usadas duas técnicas para o armazenamento do estado de uma transacção ou de um processo [Antila et al. 2001] [Pradhan et al. 1996]: *no logging* e *logging*. [Antila et al. 2001] defendem que estas estratégias podem ser usadas em ambientes móveis, desde que não ocorram processos de *handoff*.

Quando se utiliza a estratégia de *no logging*, o estado do sistema pode ser alterado quer pela recepção de mensagens de outras estações quer pela introdução de dados do utilizador. Os eventos que alteram o estado são designados de eventos de escrita. Nesta estratégia, os estados são guardados na ESM, sempre que ocorra um evento de escrita nos dados da estação móvel. Assim, depois de uma falha, quando a estação móvel reinicia deve enviar uma mensagem para a ESM, que lhe transmite o seu último estado guardado, podendo a estação móvel continuar a execução das suas tarefas a partir deste estado. Porém, o envio frequente do estado da estação para a ESM, sob a rede sem fios, representa um factor limitativo para este esquema.

O esquema de *logging* é um esquema pessimista usado em sistemas estáticos. Aqui, a estação móvel faz pontos de verificação periódicos do seu estado. Os eventos de escrita que ocorrem entre estes pontos de verificação são também registados, assegurando que não há perda nem de mensagens nem de dados introduzidos pelo utilizador caso ocorra uma falha. Sempre que a ESM recebe uma mensagem para uma dada estação móvel, primeiro deve guardá-la no seu registo de *logs* e só de seguida a deve enviar para a referida estação para que seja executada. Do mesmo modo, se a estação móvel receber dados através do utilizador, deve enviá-los logo para a ESM, para que estes sejam aí guardados, aguardando que a ESM confirme a sua recepção. Enquanto espera por esta mensagem, a estação pode processar os dados introduzidos pelo utilizador, contudo o resultado do processamento só pode ser enviado depois de ter ocorrido a recepção da referida mensagem.

O esquema de *logging* assegura que não há perda de mensagens, nem de dados, entre os vários pontos de verificação, devido a falhas ocorridas na estação móvel. Isto porque, se faz o *log* contínuo de mensagens e de dados até que se guarde um novo ponto de verificação na ESM, sendo nesta altura limpo o registo de *logs*. Depois de uma falha, quando a estação móvel reinicia, a ESM envia-lhe o último ponto de verificação guardado e reinicia a execução através da repetição do *log* dos eventos de escrita, obtendo-se assim o estado anterior à falha.

8.1.2. Estratégias durante o Processo de *Handoff*

As estratégias apresentadas nesta secção, pessimista, *lazy* e *trickle*, ao contrário das anteriores, fornecem meios para tolerância a falhas mesmo durante o processo de *handoff* [Pradhan et al. 1996] [Antila et al. 2001].

Quando se usa a estratégia pessimista e ocorre o processo de *handoff*, os pontos de verificação são transferidos para a ESM da nova célula. Se se estiver a usar a estratégia de *logging*, definida na secção anterior, para além dos pontos de verificação também se transfere o *log* de eventos de escrita. Depois da nova ESM receber esta informação, deve enviar uma mensagem à ESM anterior, informando-a que já recebeu toda a informação, para que essa possa apagar esses registos. A grande desvantagem do uso desta estratégia é a elevada transferência de grandes volume de informação, sempre que ocorre um *handoff*, o que pode causar longas interrupções durante este processo.

Ao contrário do que acontece com a estratégia pessimista, com a *lazy* não se efectuam quaisquer transferências aquando do processo de *handoff*. Em vez disso, é criada uma ligação com as várias ESMs pelas quais a estação móvel passou. Deste modo, quando ocorre um processo de *handoff*, a ESM da nova célula apenas necessita de guardar informação acerca da ESM anterior. Neste esquema, é necessário manter-se uma lista de ligações por cada estação móvel. Para assegurar que se limpam os pontos de verificação desnecessários, é enviada uma notificação para a última célula quando se efectua um novo ponto de verificação. Se a ESM anterior não efectuou nenhum ponto de verificação, então envia essa mensagem para a estação antecedente, e assim sucessivamente, até se encontrar a estação com o último ponto de verificação. Todas as ESMs que recebem a notificação limpam qualquer estado associado a essa estação.

O uso da estratégia *lazy* provoca um menor *overhead* quando comparado com a estratégia anterior. Contudo, o seu processo de recuperação é bastante mais complexo, pois quando ocorre uma falha e a ESM actual não possui o estado actual do processo, então têm de se obter os *logs* e os pontos de verificação das ESMs que pertencem à lista de ligações. A ESM actual, depois de encontrar a informação transfere-a para a estação móvel, que a utiliza para obter o estado que possuía antes da falha ocorrer. Para além disto, a distribuição de *logs,* em diferentes ESMs, cresce proporcionalmente ao aumento da mobilidade das estações. Convém ainda referir que, uma falha em qualquer ESM contendo o *log* torna o estado da informação inútil.

A estratégia *trickle* tenta evitar alguns dos problemas resultantes da estratégia *lazy*. Para tal, toma medidas que tentam assegurar que os *logs* e os pontos de verificação se encontram numa ESM próxima, podendo ser a ESM actual ou uma vizinha. Outro objectivo desta estratégia é que o processo de *handoff* se faça no menor período de tempo possível. Assim, devem certificar-se de que os *logs* e os pontos de verificação correspondentes à estação móvel se encontram na ESM da célula anterior. Deste modo, assumindo que a ESM vizinha se encontram sempre a um salto, garante-se que os pontos de verificação e os *logs* estão sempre, no máximo, a um salto da ESM actual. Para que tal seja possível, durante o processo de *handoff* deve ser enviada uma mensagem de controlo à ESM precedente de modo a transmitir qualquer ponto de verificação ou *log* à ESM actual.

De um modo semelhante à estratégia anterior, aqui também a ESM da célula antiga envia uma mensagem de controlo para a ESM da nova célula contendo a localização da última célula da estação móvel, antes de se efectuar o *handoff*. Neste esquema, a nova ESM apenas guarda o identificador da célula anterior da estação móvel.

Sempre que se efectuar um ponto de verificação na ESM actual, deve ser enviada uma notificação à ESM anterior para que limpe o registo associado a essa estação móvel. Do mesmo modo, durante o processo de recuperação se a ESM da célula actual não possuir informação acerca da estação móvel que falhou, deve obter a informação necessária a partir da ESM precedente, para que a estação0 possa restaurar o seu estado.

8.1.3. ARIES Adaptado a Ambientes Móveis

Um dos modelos de recuperação existentes para os SBDDs é o ARIES[40], que suporta o *rollback* parcial das transacções, *locking* com granularidade fina e recuperação de transacções, usando o método de *logging* de escrita à cabeça[41].

[Gore & Ghosh 2000] apresentam um modelo de recuperação de transacções para SBDMs que corresponde a uma adaptação do modelo ARIES. Para tal, adicionam a este modelo tabelas de

[40] *Algorithm for Recovery and Isolation Exploiting Semantics.*
[41] *Write-Ahead.*

mensagens, com estruturas de dados como *logs*, tabela de transacções, tabela de páginas sujas[42] e páginas.

O *log* mantém a história completa das transacções e corresponde a uma sequência de registos. Cada um destes registos possui alguma informação de manutenção do sistema, porém a maioria da sua informação corresponde a informação de *redo* ou *undo*. As estações móveis podem possuir mais do que uma transacção, devendo cada uma destas possuir um identificador único. Consideram ainda a existência de uma sequência numérica de *log*, que tem como objectivo distinguir dois registos de *log*.

No processo de recuperação e de rollback a sequência numérica de *log* é muito importante pois é usada como marca e indica se as transacções foram desfeitas totalmente ou não. Esta sequência também é útil para determinar qual o último ponto de verificação antes da falha, sendo repetida a história a partir desse ponto, com a ajuda dos registos de *log*, reconstruindo-se assim as tabelas de transacção e as tabelas de páginas sujas.

Neste modelo, durante o processamento normal das transacções a ESM reproduz a imagem da transacção para a estação móvel, registando ainda as restrições de tempo, que poderão ser úteis quando se utiliza o protocolo de *timeout*, as acções seguintes da transacção são comunicadas pela estação móvel à ESM. Quando ocorre uma falha, os gestores de recuperação da respectiva ESM fazem as correcções necessárias ao longo da rede fixa para que a transacção móvel seja guardada ou abortada.

O processamento da actividade de *rollback* pode ser executado de dois diferentes modos, dependendo das ESMs correspondentes possuírem ou não conhecimento da sua execução:

Todas as transacções que estão a ser executadas na estação móvel e ainda não foram comunicadas podem ser desfeitas sem quaisquer complicações.

Caso as transacções já tenham sido comunicadas à ESM, então a estação móvel tem de notificar explicitamente quais as acções que devem ser desfeitas.

Neste último caso, o gestor de recuperação, que se encontra na ESM, verifica os seus dados locais para identificar quais os *logs* que necessitam de ser desfeitos. Pode acontecer que para desfazer uma transacção seja necessária a cooperação de diversos gestores de transacções, localizados em diferentes estações. Se o *rollback* terminar na ESM actual, outras operações

[42] *Dirty Page Table.*

podem ser executas a partir do ponto onde o *rollback* terminou. Quando o processo de *rollback* termina na ESM anterior então a imagem da transacção com a parte das operações da transacção que não foram desfeitas é armazenada e as acções seguintes são executadas na ESM actual.

A falha de uma transacção pode ser causada pelo aborto de uma transacção, devendo a recuperação ser feita com o *rollback* total ou parcial das transacções. Como já foi referido, depois de uma falha o primeiro passo de um processo de recuperação deve ser o de trazer novamente o sistema para um estado consistente. O modelo ARIES efectua esta tarefa em três passos: análise, refazer e desfazer. Também o modelo apresentado em [Gore & Ghosh 2000] recorre ao uso destes três passos, porém com algumas alterações. No passo de análise são acrescentadas mensagens que são manuseadas pelo ambiente das transacções móveis e constituem pontos de análise para as transacções móveis. O passo de refazer coloca a base de dados no estado que possuía antes de ocorrer a falha. Por último, o passo de desfazer, desfaz todas as transacções que se encontravam activas quando a falha ocorreu.

8.1.4. Pontos de Verificação

Alguns dos modelos de recuperação dos sistemas de bases de dados tradicionais baseiam-se em pontos de verificação. Os protocolos baseados nestes pontos de verificação podem ser divididos em três classes: pessimistas, optimistas e consistentes. Existem alguns estudos que apresentam adaptações destes protocolos para ambientes móveis. [Chrysanthis 1997] defende que as características dos ambientes móveis que mais afectam o uso dos protocolos de recuperação baseados em pontos de verificação são:

- Maiores taxas de falhas, porque as comunicações sem fios não são tão fiáveis e seguras quanto seria desejável.
- Largura de banda limitada, devido ao uso de comunicações sem fios.
- Instabilidade do armazenamento de informação nas estações móveis, dado que estas estão mais sujeitas a estragos e perdas.

[Chrysanthis 1997] apresenta um protocolo de recuperação baseado em pontos de verificação para ambientes móveis. Para tal, dividem a execução de qualquer processo em fases que terminam com pontos de verificação.

Neste protocolo, cada estação possui um vector de dependências, que mantém a informação acerca das dependências transitivas. Aqui, um seu i-ésimo componente corresponde à última fase *k* do processo *i*, da qual a fase actual depende. Dizendo-se ainda que a fase actual depende da fase *k* do processo *i* se:

- Se recebeu uma mensagem da fase *k* ou de uma superior no processo *i*.
- Se recebeu uma mensagem da fase *m* no processo *p* que depende da fase *k* ou de uma superior no processo *i*.

[Chrysanthis 1997] apresenta uma regra para a existência de pontos de verificação, em diferentes estações, independentes e síncronas. Esta regra é desenhada de modo a que um ponto de verificação tenha conhecimento de todas as suas dependências. Este facto é assegurado porque não se permite que se criem novas dependências na fase seguinte ao primeiro envio. Esta regra divide-se em três fases: recepção, envio e estática. Nas duas primeiras fases os processos podem continuar a sua execução, enviando e recebendo qualquer tipo de mensagens, enquanto que na última fase, as mensagens que recebem e enviam não podem ser dependentes.

Uma estratégia simples para se efectuar o *rollback* é informar todas as estações que ocorreu uma falha, devendo cada uma delas fazer o seu *rollback*, para que se obtenha o estado consistente anterior. Contudo, este procedimento pode ser desnecessário, pois apenas se deve fazer o *rollback* das transacções que receberam informação do processo ou da estação que falhou desde o último ponto de verificação.

Um dos problemas associados ao uso deste tipo de protocolos em ambientes móveis encontra-se no local de armazenamento dos pontos de verificação. Este armazenamento pode influenciar todo o processo de recuperação, bem como a sua eficiência. Alguns modelos defendem que os pontos de verificação devem ser guardados nas ESMs, passando a informação necessária de umas estações para as outras, enquanto que outros modelos defendem o armazenamento nas estações móveis. O que acontece a maioria das vezes é que se armazenam os pontos de verificação globais na ESM, havendo no entanto armazenamento de alguns pontos de verificação locais nas estações móveis.

[Lin & Dow 2001] apresentam também um modelo de recuperação baseado em pontos de verificação. Este é um modelo de recuperação sem efeito de dominó que combina os protocolos de pontos de verificação coordenados e de comunicação induzida dos sistemas distribuídos. Este

protocolo também se efectua em três fases e garante a consistência dos pontos de verificação. Na primeira fase, é usado o protocolo de pontos de verificação coordenados ao longo das ESMs. Na segunda, faz-se uso do protocolo de pontos de verificação de comunicação induzida entre as ESMs e as estações móveis. Finalmente, na terceira fase, cada ESM envia um pedido de ponto de verificação a todas as estações móveis pelas quais é responsável e que não receberam nenhuma mensagem durante a segunda fase.

[Park et al. 2001], [Park et al. 2001] e [Park et al. 2002] apresentam um esquema de recuperação baseado em mensagens de *logging*, em pontos de verificação independente e recuperação de *rollback* assíncrona. Na maioria dos esquemas de *logging*, são as ESMs que gerem as mensagens de *log* e os pontos de verificação de uma estação móvel, dada a instabilidade de armazenamento destas últimas. Deste modo, a recuperação da informação da estação torna-se dispersa pelas diversas ESMs, pelas quais a referida estação passou.

Quando ocorre uma falha, a estação móvel deve pedir os pontos de verificação e as mensagens de *log* a um certo número de ESMs, o que pode gerar alguns atrasos. Como as estações móveis possuem uma taxa de falha bastante elevada é desejável que a sua recuperação se efectue de uma forma rápida e simples. Por este motivo, neste esquema defendem que as estações móveis devem fazer pontos de verificação ao longo da sua deslocação.

[Park et al. 2002] consideram ainda que a ESM pode ser ou não de confiança, assim deve existir um esquema de recuperação para cada um destes casos. Quando se possui um sistema onde a ESM é de confiança, então esta deve ser a responsável pela gestão da recuperação. Neste caso, usa-se o espaço de memória volátil para se armazenarem os pontos de verificação e as mensagens de *logging* das estações móveis. As estações móveis sempre que efectuem um novo ponto de verificação devem transferi-lo, bem como o seu identificador, para a ESM correspondente. O intervalo de tempo entre dois pontos de verificação é determinado pela estação móvel. As ESMs guardam ainda todas as mensagens de *log* dirigidas às estações que se encontrem dentro da sua célula. Por outro lado, a estação móvel deve manter a informação necessária acerca do seu trajecto, como por exemplo, identificador das ESMs das células por onde passou, pontos de verificação guardados, entre outros, todavia a ESM também deve guardar alguma dessa informação.

Como, neste caso, a ESM é de confiança, pode-se garantir que se consegue recuperar toda informação que foi recebida pela ESM antes da falha, podendo a estação móvel voltar ao estado que possuía antes de ter ocorrido a falha. Para recuperar de uma falha, a estação móvel começa

por enviar uma mensagem à ESM actual, sendo esta quem gere o processo de recuperação. A ESM envia o ponto de verificação guardado, bem como qualquer mensagem de *log* enviada depois do armazenamento desse ponto de verificação. Todas as mensagens que sejam enviadas à estação móvel durante o processo de recuperação conseguem chegar sem qualquer problema à estação. [Park et al. 2001] provam ainda que o processo de recuperação quando a ESM é de confiança e consistente.

Outra situação discutida em [Park et al. 2001] é quando a ESM não é de confiança. Assim, nesta situação os pontos de verificação e mensagens de log devem ser armazenados em locais estáveis e não voláteis, considerando ainda o logging de mensagens optimista. Sempre que uma ESM recebe um ponto de verificação deve guardá-lo num local estável, sendo também aqui guardadas todas as mensagens recebidas.

Neste caso, quando uma estação recupera de uma falha, deve enviar uma mensagem de falha para a ESM actual, devendo esta fazer o broadcast da mensagem para todas as outras ESMs. Esta mensagem deve incluir o identificador do último estado recuperável. Quando uma ESM recebe esta mensagem deve comparar com o seu vector de dependências e, de seguida, decidir se faz o rollback ou não.

Durante o processo de recuperação a estação móvel tem de localizar o ponto de verificação e a sequência de mensagens adequados. Para a obtenção desta informação, a ESM actual deve possuir informação que identifique o trajecto das estações móveis que possui naquele instante.

Em [Park et al. 2002] é apresentado um esquema de gestão de armazenamento distribuído para o logging de mensagens em ambientes móveis, usando-se para tal uma gestão baseada na região. Isto significa que, cada estação móvel possui a sua informação de recuperação na ESM mais próxima, podendo recuperar mais rapidamente. Contudo, desta forma, pode-se aumentar os custos de transferência de toda a informação para a ESM mais próxima. Todavia, se a informação continuar dispersa por várias células também pode haver um aumento dos custos do processo de recuperação. O esquema de gestão de armazenamento de informação baseado na região tenta arranjar um ponto óptimo entre estes dois aspectos.

Assim, se a estação móvel se desloca dentro de uma região, a recuperação fica a cargo do gestor dessa região. Por outro lado, se a estação móvel se desloca para outra região, então a informação de recuperação deve ser transferida para o gestor perto dessa nova região – a distância não deve exceder um valor pré-determinado. Nesta solução, alterando o tamanho das regiões e os valores a partir dos quais se deve fazer a transferência consegue-se fazer um controlo de custos.

Uma região, neste esquema, pode corresponder apenas a uma célula. No entanto, na generalidade dos casos costuma corresponder a um conjunto de células. Dentro de cada uma destas regiões deve existir um gestor da região. Sempre que a estação móvel se desloca para uma nova região deve-se transferir o ponto verificação actual e o log das mensagens para o novo gestor de região. Deste modo, em caso de troca de região, a gestão da recuperação pode ser eficiente em termos do número e do tamanho das mensagens trocadas.

Quando uma estação se desloca à volta de um pequeno número de regiões é preferível limitar as transferências de forma a reduzir o custo de transferência entre as regiões. Contudo, se a estação se deslocar para uma região mais distante e se não houver transferência de informação para a nova região, os custos de recuperação podem ser bastante mais elevados. Na tentativa de se arranjar um ponto óptimo, neste esquema faz-se uma análise para se decidir se se transfere a informação de recuperação para o novo gestor de região ou não, quando há troca de região. Esta análise é feita com base no movimento da estação móvel, possuindo esta um contador com o número de vezes que a estação mudou de ESM dentro de duas regiões distintas, sendo efectuada a transferência de informação quando este valor exceder um valor pré-definido.

8.2. Segurança

A segurança de um sistema de base de dados tenta minimizar a sua vulnerabilidade, isto é, tenta diminuir a probabilidade que uma entidade externa ao sistema interfira no seu comportamento, sem autorização ou autenticação prévia. Deste modo, a segurança é um conjunto de medidas e procedimentos definidas de forma a evitar que a informação seja destruída, alterada, ou até mesmo acedida, por entidades não autorizadas [Carneiro 2002].

Com os requisitos de segurança também se pretende que quando ocorre um acidente, dado que estes não são completamente evitáveis, se estabeleçam os requisitos mínimos para que o sistema possa continuar a trabalhar. Assim, a manutenção e assistência técnicas são duas vertentes da segurança.

Nos SBDMs as estações móveis e as ligações sem fios possuem para além dos problemas citados, problemas de disponibilidade, confidencialidade, integridade e responsabilidade. Nestes ambientes, as estações acedem à base de dados em qualquer altura a partir de qualquer lugar. Assim, existem riscos de segurança e privacidade novos.

Em [Lubinski 1998] considera-se que para se construir um modelo de segurança devem ter-se em conta os seguintes aspectos:

- Problemas de segurança de rede, como a privacidade de conteúdos e de localização.
- Mobilidade das estações e de informação.
- Desconexões.
- Modo de acesso aos dados.
- Escala das operações.

As medidas de segurança dos SBDMs devem ter em conta a distribuição e a heterogeneidade dos dados. Como estes sistemas são distribuídos e replicados também se deve ter em conta a protecção de algumas operações – gestão, acesso e transferência – para se que se protejam os dados e os metadados [Lubinski 1998].

Uma solução simples para garantir privacidade e segurança aos fornecedores de dados é a utilização de uma *Infra-estrutura de Chave Pública* (ICP), cujo objectivo é a certificação dos dados acedidos. Quando se enviam dados, se a estação receptora possuir uma ICP relacionada com a estação emissora também se pode certificar os dados recebidos. Contudo, esta tarefa de certificação nem sempre é uma tarefa fácil neste tipo de ambientes [Perich et al. 2001].

Em [Lubinski 1997], [Lubinski 2000] apresenta-se um modelo para se garantir a segurança em ambientes móveis, efectuando três principais tarefas:

- **Restrição nas transparências da base de dados,** embora o modelo continue a fornecer transparência, esta deve ser controlada para evitar que o sistema possua estados inseguros.
- **Separação horizontal e vertical dos metadados,** para evitar abusos e aplicações erradas da informação de contexto. A separação horizontal representa uma vista em camadas do sistema, evitando que hajam fluxos de dados indesejáveis entre algumas camadas do sistema, que se encontrem fora da área de controlo. A separação vertical guarda a informação do movimento do utilizador, fornecendo confidencialidade e permitindo apenas uma vista dos padrões de movimento e comportamento.

- **Adaptação da segurança,** que permite uma adaptação flexível às características ambientais, decidindo qual o mecanismo de segurança adequado.

A forma de executar a adaptação requer mais algumas funcionalidades por parte dos intervenientes no sistema. Assim, existe um componente de adaptação que mede todos os acessos e gere a segurança quando ocorrem alterações significativas. Este componente mede todos os acessos de uma estação a outra, decidindo quando é que deve activar as medidas de segurança e disparar as componentes do sistema. A adaptação resulta num acesso ou resultado de *query* ajustados, ou ainda na negação de um acesso pretendido. Deste modo, o componente de adaptação fornece garantias de segurança aos dados e à estação acedida, não permitindo que se transfiram dados para uma área desprotegida. Por outro lado, não se permite que uma estação fique com dados ou aplicações inseguras. Além disto, este componente gere as desconexões, fornecendo segurança local aos componentes durante a execução local. Neste modelo adaptam os requisitos de segurança aos modos de operação do utilizador. Na Figura 22 encontra-se esquematizado o acesso de mediação entre duas estações do sistema, através do componente de adaptação [Lubinski 2000].

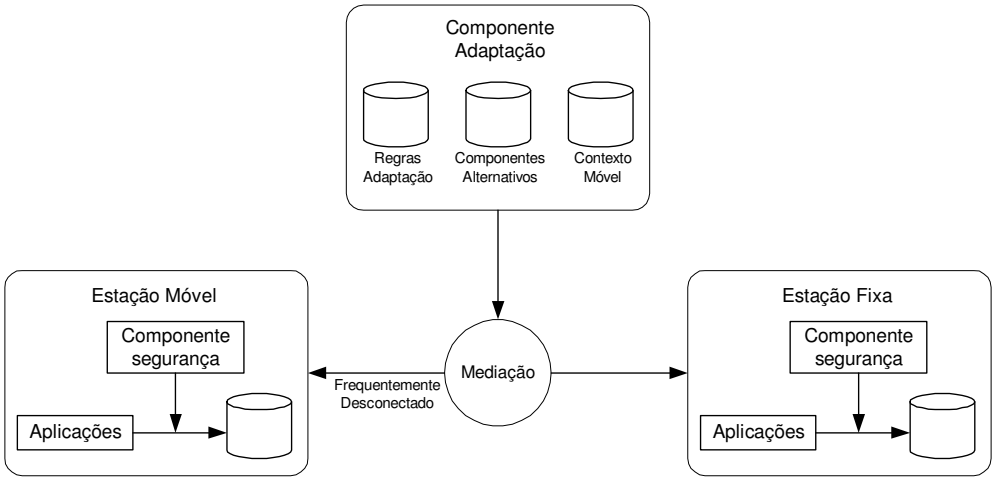

Figura 22 – Mediação de Acesso através de um Componente de Adaptação.

Este modelo fornece, pelo menos, os requisitos mínimos de segurança para ambientes móveis, permitindo a participação do utilizador, protegendo o aparecimento de metadados para evitar os

padrões de mobilidade e adaptando as medidas de segurança aos requerimentos dos ambientes móveis.

Capítulo 9

Sistemas Implementados

Ao longo desta dissertação foram apresentadas algumas, da generalidade, das características, limitações e problemas mais relevantes que estão normalmente associados com os SBDMs, bem como alguns dos possíveis esquemas para a sua gestão. Actualmente, existem já alguns sistemas implementados ou desenhados que permitem a integração de um conjunto de estações fixas e um outro de estações móveis num único ambiente, com características muito semelhantes às apresentadas pelos SBDMs. Neste capítulo, faz-se a apresentação de alguns dos sistemas que já se encontram implementados, destacando-se os seus aspectos mais relevantes.

9.1. Bayou

Em [Demers et al. 1994], [Terry et al. 1994] e [Terry et al. 1995] apresenta-se o sistema *Bayou*, cujo objectivo principal é o fornecimento de mecanismos que permitam aos clientes móveis

efectuar leituras e escritas activas sobre dados partilhados. Neste sistema, cada conjunto de dados é replicado como um todo em vários servidores.

Na Figura 23 [Jing et al. 1999] apresenta-se, esquematicamente, o sistema *Bayou*. As aplicações, a executar como clientes, interagem com os servidores através da *Application Programming Interface (*API) do *Bayou,* que é implementada por um fragmento do cliente ligado às aplicações. Esta API suporta tanto operações de leitura como de escrita. É ainda possível observar-se que na mesma estação pode existir, ao mesmo tempo, um cliente e um servidor.

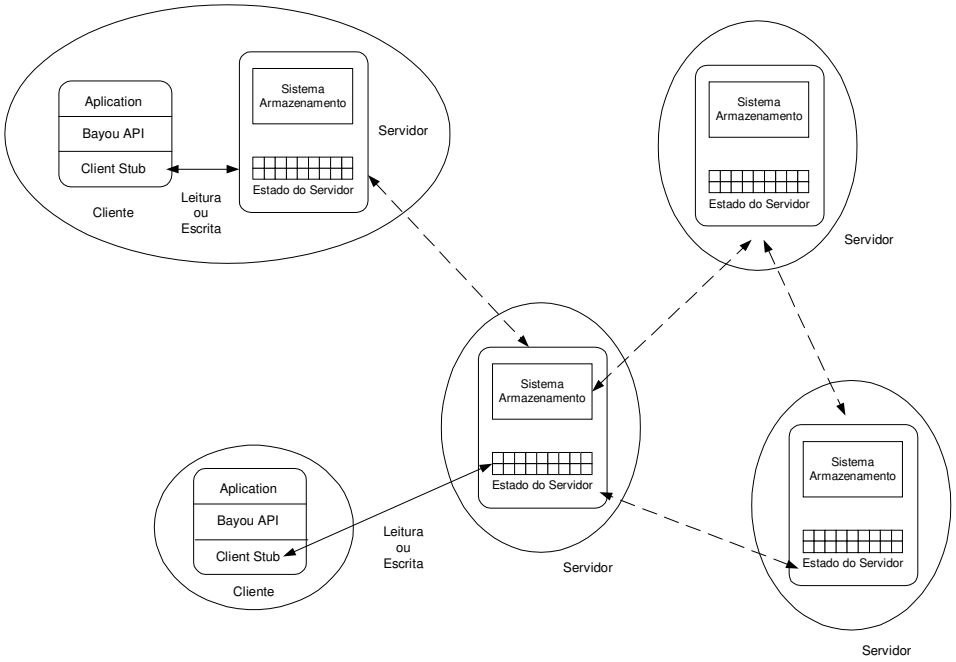

Figura 23 – Modelo do sistema *Bayou.*

O acesso a um servidor é suficiente para que um cliente consiga produzir trabalho útil, pois desta forma consegue ler os dados que o servidor possui e, ainda, pode submeter-lhe as suas operações de escrita. Quando o servidor aceita uma operação de escrita, o cliente deixa de ser o responsável por essa operação, não precisando de esperar para a submeter a outros servidores. Desta forma pode-se dizer que, o sistema *Bayou* possui um modelo de replicação com consistência fraca que incorpora acessos do tipo *lê-qualquer* e *escreve-qualquer*.

Pode-se assim concluir que, neste sistema algumas réplicas dos itens de dados suportam operações de leitura e de escrita, o que pode gerar, naturalmente, alguns conflitos pois os valores das réplicas de um item de dados, residentes em servidores distintos, podem divergir em qualquer altura, dado que recebem e executam diferentes operações de escrita. Uma característica fundamental do *Bayou* é que todos os servidores se movem para um eventual estado de consistência. Dois importantes factores que permitem que o servidor adquira consistência eventual são a garantia de que todas as operações de escrita são executadas pela mesma ordem em todos os servidores e que os processos de detecção de conflitos são determinísticos, ou seja, todos os servidores resolvem os conflitos do mesmo modo.

Quando uma operação de escrita é aceite num servidor é considerada, inicialmente, como uma tentativa. As escritas tentativas são ordenadas com etiquetas temporais, atribuídas pelos servidores que as aceitaram. Posteriormente, as escritas guardadas são ordenadas de acordo com essas etiquetas temporais. Como os servidores podem receber escritas de clientes ou mesmo de outros servidores e como os seus efeitos são processados imediatamente, pode acontecer que essas escritas não sejam processadas pela ordem correcta. Neste caso, os servidores devem ser capazes de desfazer as alterações efectuadas pelas escritas tentativas anteriores e reexecutá-las por uma ordem diferente. O número de vezes que uma operação de escrita é reexecutada depende apenas da ordem com as operações de escrita chegam. Conceptualmente, cada servidor mantém um *log* com todas as operações de escrita que recebeu, ordenadas pelas suas etiquetas temporais tentativas ou de *commit*. Os *commits* de escrita aparecem no topo do *log*. Os dados actuais do servidor são gerados pela execução de todas as operações de escrita por uma determinada ordem.

O sistema *Bayou* usa a detecção de conflitos específica de uma aplicação. Isto é, tem em conta que diferentes aplicações podem possuir diferentes significados para um conflito, o que faz com que a sua detecção nem sempre possa ser feita através da simples observação dos resultados das operações de escrita. Desta forma, o sistema de armazenamento deste modelo possibilita que as aplicações especifiquem a sua própria noção de conflito.

Quando se usa a detecção de conflitos específicos da aplicação tem de se incluir, em cada operação de escrita, uma verificação de dependência, que, basicamente, consiste numa *query* e no seu resultado pretendido. Assim, verifica-se se existe um conflito se o resultado da *query* não for o pretendido. Esta verificação de dependência é uma pré-condição da execução da actualização que está incluída na operação de escrita. Se a verificação falhar, então não se

executa a actualização e o servidor deve invocar um procedimento para resolver o conflito que foi detectado.

Os dois mecanismos de detecção de conflitos usados no sistema *Bayou* são a verificação de dependências e os procedimentos de junção. Com estes mecanismos permite-se que os clientes indiquem, para cada operação de escrita individual, como o sistema deve detectar os conflitos e a forma como esses conflitos devem ser resolvidos, baseando-se para tal na semântica da aplicação.

As dependências no *Bayou* também podem ser usadas para detectar situações nas quais existam vários utilizadores a actualizar o mesmo item de dados, sem que pelo menos um deles esteja a observar as actualizações efectuadas pelos outros utilizadores. Assim, devem verificar-se quais os itens de dados que se encontram a ser actualizados e se os seus valores não se alteraram desde a última operação de escrita. Os mecanismos de verificação também podem ser usados para detectar conflitos leitura-leitura. Além disto, cada operação de escrita pode identificar os valores esperados para cada item de dados, dos quais a actualização depende.

Sempre que é detectado um conflito o servidor *Bayou* deve executar um procedimento de junção na tentativa de resolver esse conflito. Estes procedimentos podem incluir dados embebidos e podem executar leituras arbitrárias do estado actual das réplicas do servidor. Os procedimentos de junção, em conjunto com as operações de escrita, são os responsáveis pela resolução de todos os conflitos detectados e pela verificação de dependências, devendo ainda produzir uma nova actualização a ser aplicada aos itens de dados.

Todo o processo de detecção de conflitos, executado pelos procedimentos de junção, e a aplicação das novas actualizações deve ser feita automaticamente em cada servidor como parte da execução de uma operação de escrita. O facto de se suportarem verificações de dependência separadamente faz com que se evite que o servidor execute os procedimentos de junção quando a operação de escrita não introduz qualquer conflito.

Este sistema garante ainda que os procedimentos de junção, que são executados independentemente em cada servidor, produzem actualizações consistentes, forçando-os a depender apenas dos dados actuais do servidor e de quaisquer dados fornecidos pelos próprios procedimentos de junção.

No sistema *Bayou* existem mecanismos específicos da aplicação que detectam e actualizam possíveis conflitos, definindo protocolos através dos quais se fazem actualizações de conflitos.

Para além disto, inclui ainda métodos de gestão de consistência para detecção de conflitos. Para garantir a consistência, os servidores podem desfazer algumas alterações efectuadas ou reexecutá-las por uma ordem global. Aqui, permite-se ainda que os clientes vejam os resultados de todas as escritas recebidas pelo servidor mesmo aqueles que ainda não possuem todos os conflitos resolvidos.

9.2. Coda

O sistema *Coda* [Kistler & Satyanarayanan 1992] fornece uma interface do sistema de ficheiros. As aplicações a executar nos sistemas *Coda* usam essa interface do sistema de ficheiros e podem continuar a executar nas estações móveis, sem que para tal seja necessário efectuar qualquer modificação.

Num sistema Coda os servidores encontram-se localizados na parte fixa do sistema e são, geralmente, em menor número do que os clientes. Cada cliente *Coda* possui uma estação portátil e pode comunicar com os servidores através de comunicações com grande largura de banda, podendo no entanto ligar-se à rede a partir de diferentes conexões. Os clientes podem, ainda, encontrar-se temporariamente desconectados, não existindo, nessa altura, obviamente, comunicação com o servidor. O sistema *Coda* está preparado para lidar tanto com as desconexões voluntárias como com as involuntárias. Este tipo de sistema pode não ser o mais adequado para SBDMs, pois apesar de permitir a existência de desconexões as conexões das estações móveis com as estações fixas nem sempre são efectuadas com uma grande disponibilidade de largura de banda. Aliás, muitas das vezes, apenas se consegue estabelecer a comunicação no sentido *downlink,* isto é das ESMs para as estações móveis.

Os clientes estão providos com um gestor de *caches*, designado de *Venus*, que obtém e armazena dinamicamente as unidades de *cache*. Este componente pode operar em três modos distintos: *hoarding*, emulação e reintegração – na Figura 24 encontram-se esquematizados esses modos de operação, bem como as possíveis causas de transições entre cada um deles. O *Venus* encontra-se, geralmente, a operar em modo *hoarding*, confiando nos dados do servidor, mas está sempre em alerta para possíveis desconexões. Quando ocorrem desconexões deve passar a operar no modo emulação mantendo-se assim até que a conexão se restabeleça. Nessa altura

deve tratar de realizar a reintegração das actualizações ocorridas durante o período de desconexão.

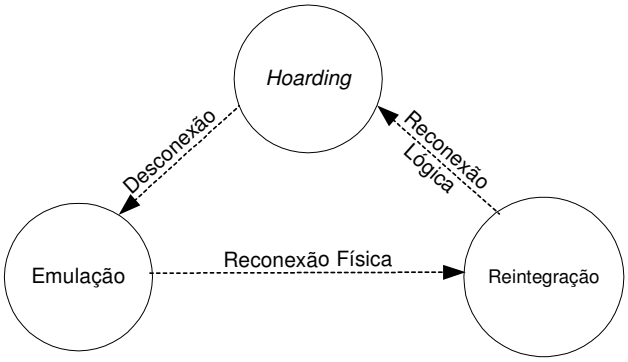

Figura 24 – Estados e Transições do *Venus.*

Dadas as diferenças de recursos existentes entre as estações fixas e as móveis, no sistema *Coda* usam-se duas classes de replicação, uma para os servidores e outras para os clientes. A primeira dessas classes de replicação possui maior qualidade e os seus valores encontram-se sempre disponíveis. Por este motivo, estas réplicas são seguras e fiáveis. A segunda classe de replicação, existente nos clientes, é inferior em todas as dimensões (consistência, qualidade e segurança). Além disto, uma réplica da segunda classe só é útil quando se efectua periodicamente a sua revalidação com a réplica primária. Por esse motivo, durante as desconexões, a qualidade das réplicas das estações móveis pode baixar. Quanto mais tempo se encontrar desconectada maior pode ser a perda de qualidade das réplicas. O sistema *Coda* usa controlo optimista de réplicas, o que melhora o seu desempenho, já que não obriga a que haja um bloqueio de actualização quando não é possível actualizar todas as réplicas em simultâneo.

No fundo, o sistema Coda simula um sistema móvel que não recorre ao uso de redes sem fios, isto é, é um sistema onde se permite a deslocação de algumas das estações, que efectuam a conexão à rede através de diferentes pontos, mas usando sempre ligações com fios. Os mecanismos que o Coda usa para garantir uma grande capacidade e disponibilidade do sistema são a replicação dos servidores e a permissão para a realização de operações durante os períodos de desconexão.

9.3. Mobisnap

O *Mobisnap* [Cunha et al. 2000] [Preguiça et al. 2000] representa um SBDM, utilizando como arquitectura de acesso aos dados a Cliente/Servidor Estendida [Jing et al. 1999], apresentada anteriormente na Secção 3.3.

Neste sistema, as cópias primárias encontram-se nas estações fixas, podendo os clientes possuir uma cópia secundária desses dados. No *Mobisnap*, todos os clientes são móveis e todos os servidores são fixos. Para limitar os prejuízos e danos causadas pela perda, danificação ou mesmo roubo da estação móvel, limitam-se os recursos necessários nos clientes. Este sistema encontra-se esquematicamente representado na Figura 25 [Cunha et al. 2000].

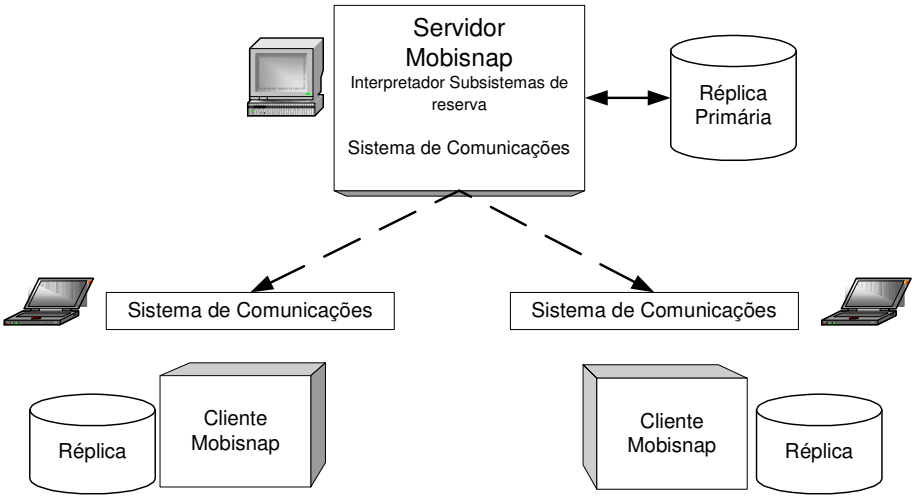

Figura 25 – Componentes do Sistema *Mobisnap.*

Quando ocorrem desconexões os clientes mantêm a sua autonomia, podendo continuar a executar as suas tarefas localmente. No entanto, devem submeter essas tarefas sob a forma de transacções móveis. Neste sistema, consideram que uma transacção móvel permite a especificação exacta de todas as condições de execução da transacção, incluindo as pré-condições, pós-condições e actualizações alternativas, permitindo ainda a especificação de mecanismos de resolução de conflitos, tal como acontecia no sistema *Bayou*. Durante esses períodos de desconexão as transacções são executadas na estação móvel, porém não fornecem um resultado definitivo, pois tal só pode acontecer quando a transacção for executada pelo

servidor. O resultado obtido pela execução no servidor pode ser diferente do resultado obtido localmente, pela estação móvel.

No Mobisnap existe ainda um mecanismo de reservas, que permite que um cliente armazene localmente alguns dos valores da base de dados central, garantindo-se assim que a transacção seja executada de uma forma mais autónoma. Desta forma, consegue-se aumentar o número de transacções executadas localmente durante os períodos de desconexão, podendo-se diminuir também o tempo de execução de uma determinada transacção.

No *Mobisnap* também se usa a ideia de cópia tentativa e de cópia guardada. Os clientes *Mobisnap* podem utilizar estes dois sistemas de cópia. A cópia guardada contém o último valor recebido do servidor, enquanto que a tentativa possui o valor alterado pelas transacções locais. Por vezes, os valores das cópias guardadas no cliente, não coincidem com os valores existentes nos servidores. Assim, quando os clientes restabelecerem a conexão com o servidor devem submeter as transacções locais, para que as cópias tentativas, se não existirem conflitos, se possam tornar em cópias guardadas. Nesta altura, o cliente pode ainda colocar mais dados nas suas reservas locais.

As transacções submetidas são executadas imediatamente pelo cliente, podendo ter um dos seguintes resultados:

- ***Reservation Commit.*** Este resultado significa que a transacção foi executada com sucesso até à altura do *commit* e que as condições da transacção foram verificadas através das reservas de dados da estação cliente.

- ***Commit* Tentativo.** Neste caso, a transacção foi executada com sucesso até à altura do *commit*, mas as condições foram verificadas através de dados tentativos.

- **Aborto Tentativo.** Quando ocorre este resultado significa que a transacção foi executada, mas que, entretanto, ocorreu uma instrução de aborto, tendo este estado sido determinado através do uso das cópias tentativas.

- **Desconhecido.** Quando se obtém este resultado significa que os dados guardados não são suficientes para determinar o resultado da transacção.

Todas as transacções processadas localmente devem ser armazenadas para que, posteriormente, possam ser propagadas para o servidor. Geralmente, as transacções que foram abortadas não são armazenadas, já que, obviamente, não necessitam de ser propagadas. As transacções que

terminam com o resultado de *Reservation Commit* devem ser propagadas para o servidor com a indicação de quais as reservas que foram usadas na sua execução. Assim que o servidor receber uma transacção deve proceder à sua execução, podendo esta ser guardada permanentemente ou abortada.

9.4. Odyssey

O sistema *Odyssey* [Noble 1998] [Noble 2000] foi desenhado para suportar uma grande variedade de aplicações de acesso a informação móvel. Estas aplicações residem em clientes móveis, mas acedem ou actualizam dados em servidores remotos. Neste sistema, assume-se que os servidores têm mais capacidade, e são mais confiáveis que os clientes, e que possuem um sistema de administração central. Os clientes não têm que ser obrigatoriamente móveis.

Num sistema replicado, o ideal é que um item de dados disponível na estação móvel possua o mesmo valor da cópia que se encontra no servidor. Porém, dados os recursos limitados das estações móveis e a possibilidade destas operarem durante os períodos de desconexão, faz com que as cópias dos itens de dados se degradem e que nem sempre possuam o mesmo valor da cópia que se encontra no servidor. Aqui, usa-se um critério de fidelidade[43] para identificar o grau com que a réplica do cliente coincide com a réplica do servidor. A fidelidade possui várias dimensões, a mais conhecida é a consistência.

Este sistema implementa uma adaptação baseada na aplicação, para tal, monitora os recursos do sistema e notifica as aplicações de alterações significativas que tenham ocorrido, devendo ainda incentivar algumas decisões de alocação de recursos. Depois de notificadas, cada uma das aplicações decide a melhor forma de se adaptar às ambientais alterações ocorridas.

No *Odyssey* os clientes funcionam como fornecedores de serviços adaptativos do sistema, pois encontram-se melhor posicionados para perceber que recursos estão disponíveis e quais as suas prioridades actuais para o uso de tais recursos. Assim, são os clientes que tomam as decisões de adaptação. Por outro lado, os servidores devem ser capazes de suportar estas decisões, através do fornecimento de dados, com diferentes qualidades. Contudo, os servidores não têm qualquer

[43] *Fidelity.*

papel activo na selecção dessa qualidade ou monitorização dos dados. A arquitectura deste sistema, que fornece adaptação baseada na aplicação, encontra-se esquematizada na Figura 26.

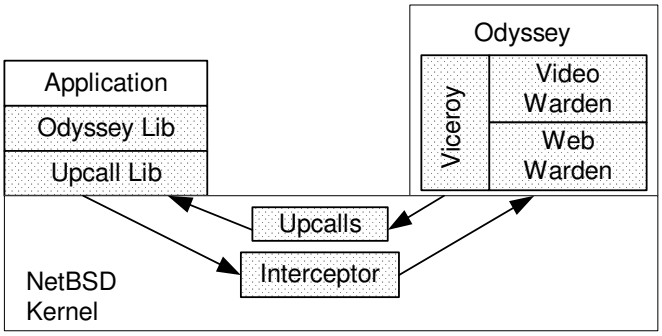

Figura 26 –Arquitectura de Cliente no Sistema Odyssey.

O *Odyssey* é implementado como um componente do *kernel* no espaço do utilizador, podendo também ser implementado directamente no *kernel* do sistema operativo ou como uma camada de *middleware*.

A gestão de acesso aos objectos de dados é feita através do sistema de ficheiros do cliente. Existe no *kernel* um módulo interceptor que redirecciona as operações de sistema sobre estes objectos para o *Odyssey*. Este módulo é constituído por dois tipos de componentes: os *viceroy* e um conjunto de *wardens*. Os *viceroy* são os responsáveis por todas as tarefas independentes de tipo nos clientes, destacando-se a monitorização da utilização de recursos e as aplicações de notificação de grandes diferenças na disponibilidade de recursos. Um *viceroy* actua como um único ponto de controlo de recursos, fornecendo o suporte para aplicações que se encontrem a executar concorrentemente. Os *wardens*, por sua vez, gerem a comunicação entre o cliente e os vários servidores. Os *wardens* são subordinados dos *viceroy*, que são os responsáveis pela gestão centralizada dos recursos.

A relação colaborativa na adaptação baseada na aplicação é realizada em duas partes. Primeiro entre o *viceroy* e os *wardens* que é centrado nos dados, sendo definidos se os níveis de fidelidade para cada tipo de dados e resultam em gestão de recursos. Segundo, entre as aplicações e o *Odyssey*, que é centrado na acção, fornecendo-se aplicações com controlo sobre a selecção dos níveis de fidelidade suportados pelos *wardens*.

A agilidade de um sistema refere-se à capacidade que este possui de se adaptar às alterações ambientais. No *Odyssey*, consegue-se esta agilidade através de um ciclo de decisões adaptativos apresentados na Figura 27.

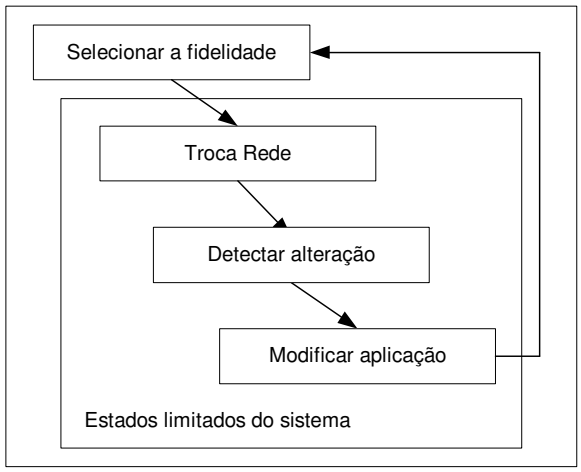

Figura 27 – Ciclo de Decisão Adaptativo.

9.5. Rover

O sistema *Rover* [Joseph et al. 1997] [Joseph et al. 1997A] fornece um ambiente que suporta tanto modelos de adaptação transparente à aplicação como modelos de adaptação baseada na aplicação para ambientes móveis. O sistema *Rover* fornece ferramentas bastante úteis para a construção de aplicações móveis.

As ferramentas *Rover* disponibilizam um sistema de objectos distribuído baseado na arquitectura Cliente/Servidor (Figura 28). Os clientes são *Rover* e encontram-se, geralmente, a executar nas estações móveis, apesar de também se puderem encontrar em execução em estações fixas. Os servidores, que podem estar replicados, correspondem principalmente às estações fixas. O *Rover* oferece uma arquitectura Cliente/Servidor e controlo de concorrência optimista.

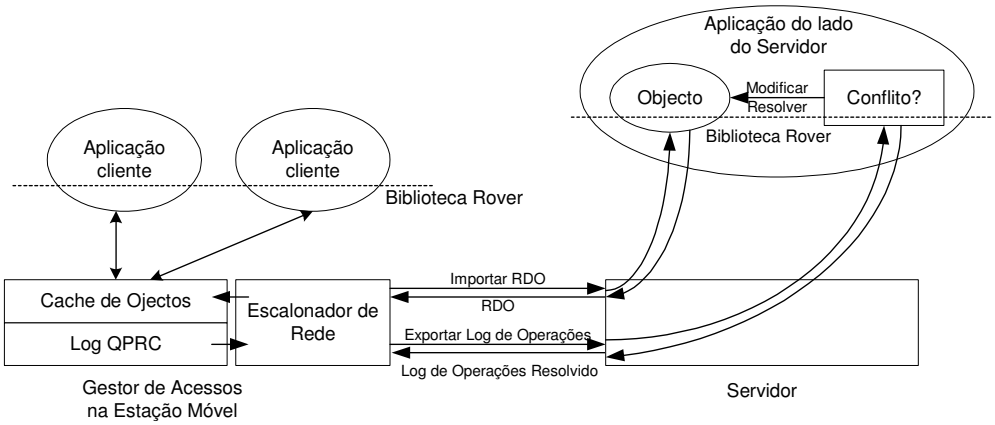

Figura 28 – Arquitectura *Rover.*

A comunicação entre clientes está limitada pelas interacções ponto-a-ponto dentro de uma estação móvel e pelas interacções entre a estação móvel e o servidor. Nestas, não existe suporte para interacções ponto-a-ponto entre estações móveis. Este sistema fornece suporte para a comunicação móvel baseada em dois importantes protocolos: *Chamadas de Procedimento Remotos em Fila* (QRPC)[44] e *Objectos Dinâmicos Realocáveis*[45] (RDO). O primeiro serve para evitar o bloqueio dos clientes, enquanto que o segundo permite que as aplicações migrem as funcionalidades entre o cliente e o servidor.

O programador das aplicações deve dividir a aplicação em duas partes, uma que irá ser executada pelo cliente e outra pelo servidor, que podem comunicar através de QRPCs, podendo de seguida cooperar com o sistema para importar RDO do servidor na *cache* local dos clientes. Para suportar esta cooperação o sistema possui quatro componentes chave no cliente e no servidor: o gestor de acessos, *cache* dos objectos (apenas no cliente), o *log* das operações e o escalonador de rede – esta estrutura encontra-se esquematizada na Figura 29.

Assim que um objecto é importado pelas *caches* dos clientes, as aplicações podem invocar os métodos fornecidos pelo RDO importado. Os métodos sem efeitos laterais são processados localmente. Todavia, os métodos que produzem efeitos laterais também podem ser processados localmente, mas neste caso devem ser gerados dados tentativos, em vez de dados definitivos. Quando se prevê uma desconexão, podem-se importar RDOs que se pensam que irão fazer falta ao processamento durante os períodos de desconexão.

[44] *Queued Remote Procedure Call.*
[45] *Relocatable Dynamic Objects.*

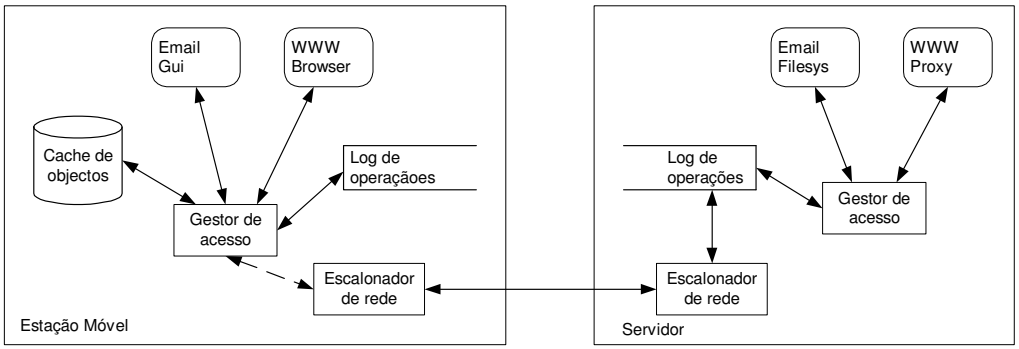

Figura 29 – Componentes da Arquitectura Rover.

O *Rover* fornece algum suporte para a implementação de cópias primárias e consistência optimista das actualizações tentativas. Cada RDO possui um servidor (*Home Server*) que mantém a sua cópia primária. Os clientes importam localmente as cópias secundárias desses RDOs, devendo posteriormente exportar os RDOs tentativos para o seu servidor. Para suportar as necessidades das diversas aplicações, este sistema também suporta outros mecanismos de manutenção de consistência para resolução de actualizações descoordenadas de um RDO – o bloqueio ao nível da aplicação[46] e algoritmos específicos de aplicações[47]. Estes mecanismos encontram-se disponíveis nas aplicações, sendo reforçadas pela infra-estrutura do *Rover* através da cooperação entre gestor de acessos e a *cache* de objectos.

Durante as desconexões, as aplicações dependem apenas das caches locais. Além disto, todos os pedidos que necessitem de RDOs, e não se encontrem disponíveis localmente, devem ser colocados numa fila de espera, para que sejam satisfeitos assim que a conexão se restabelecer. Sempre que uma aplicação alterar dados localmente esta deve marcá-los como tentativos, para que, posteriormente, se possam tornar definitivos ou não, devendo-se neste caso abortar a sua operação de actualização.

Para facilitar a reconciliação de dados na altura da reconexão, para além dos novos valores, o *Rover* regista automaticamente a invocação dos métodos como QRPCs no *log* operacional. Contudo, esta tarefa pode aumentar significativamente o tamanho do *log*. Para evitar que tal aconteça, o *Rover* envolve directamente as aplicações na manipulação e na redução desse *log*. As aplicações podem carregar procedimentos no seu gestor de acessos para que possam manipular os registos de *log*.

[46] *Application-Level Blocking.*
[47] *Application-Specific Algorithms.*

Aquando da reconexão, as modificações são actualizadas usando estratégias *lazy* através dos QRPCs para o servidor. Assim, que o QRPC chega ao servidor, este invoca o método correspondente na cópia primária. Todos os conflitos de actualização são detectados e resolvidos pelo servidor. Como o *Rover* usa controlo de concorrência específico do tipo, alguns dos conflitos podem ser evitados, os resultados da resolução de conflitos devem ser aplicados nos clientes, sobrepondo-se aos dados tentativos, que foram armazenados localmente.

Capítulo 10

Conclusões e Trabalho Futuro

10.1. Comentários Finais

Esta dissertação teve como principal objectivo apresentar algumas das possíveis estratégias para a criação de um SBDM. Nesse sentido, realizou-se um estudo detalhado, e a sua apresentação nesta dissertação, dos principais modelos implementados até ao momento e das questões mais relevantes que se levantam aquando do seu desenvolvimento e implementação. De referir: as suas arquitecturas de acesso a repositórios de dados, mecanismos de replicação de dados, modelos de transacções, processamento de *queries*, gestão de consistência e protecção dos dados. Adicionalmente, e para se tivesse uma melhor ideia da sua aplicação prática, apresentaram-se também alguns dos sistemas que se encontram actualmente implementados e que, pelas suas características, podem ser classificados com SBDMs.

Da análise de todos os modelos apresentados, bem como de todos os sistemas descritos, pode-se dissertar um pouco acerca de como obter um SBDM "ideal". Na definição deste sistema, poder-se-ia começar por pensar num modelo idêntico ao apresentado anteriormente na Figura 3 [Dunham et al. 1997] [Elmagarmid et al. 1995] [Imieslinski & Badrinath 1994] [Pitoura & Bhargava 1994A], uma vez que é o mais usado e parece ser o mais adequado para este tipo de ambiente. Para a construção desse sistema, considerar-se-ia que todas as estações fixas funcionariam como ESMs, sendo assim responsáveis por uma determinada célula. No entanto, poder-se-ia permitir que algumas estações fixas não fossem ESMs, se as condições ambientais assim o permitissem. Por exemplo, se existisse uma grande concentração de estações fixas numa determinada área, não seria necessário que todas elas funcionassem como ESMs dessa área para que se fosse possível garantir um bom nível de qualidade para o sistema em causa. Todas as ESMs deveriam ter capacidades para efectuar o *broadcast* de mensagens para as estações da sua célula. Nesse sistema, poder-se-ia também permitir que as estações móveis pudessem operar num dos quatro modos apresentados na Secção 2.3 [Ahmed et al. 1996] [Pitoura & Bhargava 1994]: completamente conectado, parcialmente conectado, *doze mode* e completamente desconectado. Estes modos garantem que as estações móveis possam continuar a seu processamento independentemente das características ambientais.

Em qualquer sistema, deseja-se que as suas arquitecturas sejam escaláveis, se adaptem de forma dinâmica às características do ambiente e forneçam mecanismos de acesso a dados eficientes e transparentes. Por estes motivos, uma boa solução para a implementação de um bom SBDM poderá ser baseada no modelo Cliente/Servidor Estendido [Jing et al. 1999] (Secção 3.3). Este modelo destaca-se, fundamentalmente, dos outros modelos pela sua simplicidade e facilidade de implementação, garantindo ainda acessos eficientes a dados sem que hajam grandes perdas de processamento durante as desconexões das estações. Basicamente, neste modelo, as estações móveis funcionariam como clientes durante as conexões com a rede fixa e como servidores, dos seus próprios dados, durante os períodos de desconexão. Além disto, dependendo das condições ambientais, qualquer estação poderia funcionar tanto como cliente como servidor dos seus próprios dados. No modelo Cliente/Servidor [Pitoura & Samaras 1998] (Secção 0) as estações móveis são sempre clientes que pedem serviços a um servidor, que não pode ser a estação móvel. Assim, se se usasse este modelo os clientes poderiam não conseguir prosseguir com a execução das suas operações durante os períodos de desconexão. O modelo Cliente/Servidor com clientes *Hoard* [Badrinath & Phatak 1997] (Secção 3.2) tem um funcionamento idêntico ao Cliente/Servidor Estendido. Porém os clientes fixos não podem operar sobre os seus próprios dados. São obrigados a utilizar sempre os serviços disponíveis. Se se pensar que nos SBDMs as

ESMs podem estar bastante distantes, poderia ser útil que estas operassem sobre os seus dados locais, o que fazia com que este último modelo não pudesse ser utilizado. De todos os modelos de acesso aos dados apresentados, o que se apresenta como a solução menos eficaz é o modelo Ponto-a-Ponto, uma vez que, conjuntamente com as características dos SBDMs, poderiam fazer com que algumas estações estivessem ociosas por longos períodos de tempo e que fosse muito difícil garantir alguma escalabilidade através do seu uso.

No referido sistema "ideal", para que as estações móveis conseguissem fornecer dados às suas próprias tarefas, durante os períodos em que se encontrassem desconectadas, seria necessário que estas possuíssem algumas réplicas de dados locais. Para tal, dever-se-ia recorrer ao uso de um modelo que permitisse a replicação de dados nessas estações. Esses modelos deveriam ser escolhidos por fornecerem meios para uma grande disponibilidade de dados nas estações, sem que para tal se tivesse de perder a sua consistência ou aumentar os custos de comunicação relacionados com a sincronização das várias réplicas. Tal como já foi referido anteriormente no Capitulo 4, é impossível a obtenção simultânea de valores óptimos nesses três factores. Contudo, actualmente, já existem modelos que conseguem uma boa combinação destes factores. Outro aspecto importante num modelo de replicação de dados é a localização da réplica primária. Pois, se por um lado, o facto das estações móveis conterem tais réplicas pode aumentar o seu processamento local, e até mesmo a sua confiança, por outro lado, isso pode diminuir o processamento de muitas outras estações, diminuindo consequentemente a eficácia do próprio sistema. Assim, para os SBDMs, os modelos mais adequados são, geralmente, aqueles que colocam tais réplicas na parte fixa do sistema, ficando assim mais disponíveis para um maior número de estações.

O modelo de replicação optimista [Saito 2000] (Secção 4.2) apresenta-se como uma boa solução de replicação para os SBDMs. Neste modelo permite-se a definição de várias réplicas primárias, o que permitir aumentar a disponibilidade das réplicas para eventuais processo de sincronização. Por este motivo, os problemas de congestionamento perto da réplica primária não são tão evidentes. Os modelos de replicação optimistas fornecem uma maior tolerância a falhas, melhor flexibilidade de rede, menores custos, maior disponibilidade de dados e melhores índices de escalabilidade. O modelo de replicação a dois níveis [Gray et al. 1996] (Secção ~) também se apresenta como uma solução viável para este tipo de sistemas, já que a maioria das réplicas primárias se encontra na parte fixa do sistema, só ficando nas estações móveis quando as características ambientais assim o exigirem. Além disto, o processo de gestão de consistência das réplicas não é muito complexo. No entanto, as vantagens e facilidades oferecidas pelo modelo de

replicação optimista são superiores às oferecidas pela replicação a dois níveis o que poderia ser um bom motivo para a sua eventual adopção por um SBDM.

Algumas das tarefas que as estações móveis podem executar sobre os seus dados são as transacções móveis. Estas, ao contrário das transacções dos SBDDs, nem sempre garantem a verificação das propriedades ACID, uma vez que tal facto poderia causar a sucessiva suspensão, ou mesmo anulação, de várias transacções. Aliás, quase sempre, as transacções móveis têm de partilhar os seus estados intermédios. Além disso, podem começar a execução de uma transacção numa estação e terminá-la noutra. Existem inúmeros modelos de transacções para os SBDMs que tentam fornecer melhores níveis de processamento com o menor número de suspensões, abortos e inconsistências possível. O modelo de transacções baseado no controlo de concorrência optimista 5.8 [Ahmed et al. 1996] apresenta-se como uma boa solução para os SBDMs. Aqui, as transacções podem ser executadas pelas estações passando para um dos estados referidos na Figura 18. Este modelo envia periodicamente relatórios de invalidação para as estações, com informação acerca dos dados que tenham sido alterados desde o último relatório de invalidação. Deste modo, evita-se a comunicação do tipo *uplink*, que como se sabe consome mais recursos e é mais cara, na validação das transacções. Para além disto, as estações possuem ainda ao seu dispor os relatórios de actualização, que também são transmitidos periodicamente, que actualizam os itens de dados de dados. Porém, existem outras soluções de modelos de transacções móveis que não incluem mecanismos de replicação de dados para SBDMs. De referir, as Transacções *Open-Nested* [Chrysanthis 1993] (Secção 5.1), *Clustering* [Pitoura & Bhargava 1994] (Secção 5.2) e PRO-MOTION [Walborn & Chrysanthis 1997] [Mazumdar & Chrysanthis 1999] (Secção 5.5).

Na Tabela 4 apresentam-se os modelos escolhidos para a implementação de um sistema supostamente “ideal”.

Arquitectura para Acesso aos Dados	Cliente Servidor Estendido
Modelo de Replicação	Replicação Optimista
Modelo de Transacções Móveis	Controlo de Concorrência Optimista

Tabela 4 – Modelos escolhidos para o sistema ideal.

Para além das transacções, as estações de um SBDM podem também executar *queries*. Como se referiu, o resultado de uma *query* pode depender da localização da estação que solicitou a sua execução. Deste modo, para além dos custos de comunicação, o processo de optimização deve ter em conta eventuais custos associados com a localização das estações [Kottkamp & Zukunft

1998]. Para que se reduzam os custos de localização das estações, só se deve efectuar a localização quando esta é imprescindível para a apresentação de um resultado correcto. Além disto, podem fazer-se algumas restrições antes de se efectuar a localização, fincando-se deste modo, com um menor domínio de procura. Durante o processamento de *queries* num SBDM pode acontecer que as suas diversas fases de processamento sejam efectuadas em diferentes estações, dependendo dos recursos que cada uma tem disponíveis [Kottkamp & Zukunft 1998] (Figura 20).

Quando se permite que as estações executem tarefas sobre as suas réplicas de dados locais, mesmo quando se encontram desconectadas, pode-se estar a contribuir para o aparecimento de situações de inconsistência de dados. Assim, deve-se fazer a sincronização de réplicas logo que seja possível. Esta sincronização, como se sabe, visa resolver eventuais conflitos entre os conteúdos das réplicas, que podem obrigar a desfazer ou a refazer algumas das tarefas já realizadas no sentido de se garantir que o sistema se mantenha consistente. Só quando todos os conflitos se encontrarem resolvidos é que as alterações sobre os dados se podem tornar definitivas. Quase todos os modelos de replicação possuem alguns níveis de gestão de consistência. No modelo de replicação optimista tenta-se obter uniformidade entre as réplicas de dados, seguindo-se os passos apresentados na Figura 21 [Saito 2000] [Saito & Saphiro 2002]. Com o modelo de transacções usando controlo de concorrência optimista [Ahmed et al. 1996] recorre ao uso de etiquetas temporais e versões de itens de dados para detectar e resolver eventuais conflitos entre os conteúdos das réplicas.

Nos SBDMs, como em quaisquer outros sistemas, podem ocorrer falhas, o que requer a existência de mecanismos de recuperação do sistema. Contudo, é essencial que os métodos de recuperação escolhidos para o sistema não tratem as desconexões das estações móveis como falhas. Para o sistema dito ideal, poderia ser escolhido o modelo baseado em pontos de verificação para ambientes móveis [Chrysanthis 1997]. Neste esquema, sempre que se termine uma fase de execução deve ser criado um ponto de verificação, guardando-se o estado do sistema nessa altura. Deste modo, quando ocorre uma falha pode-se recuperar o sistema para o estado guardado pelo último ponto de verificação.

Em suma, para a realização desta dissertação efectuou-se o desenvolvimento de um trabalho de estudo pormenorizado e fundamentado sobre o domínio dos *Sistemas de Bases de Dados Móveis*, tendo-se encontrado algumas soluções para a implementação e gestão de um SBDM, nomeadamente ao nível da arquitectura de acesso aos dados, esquemas de replicação, modelos de transacções e de gestão de consistência, entre outros.

10.2. Trabalho Futuro

De forma a completar o trabalho realizado nesta dissertação, seria sem dúvida muito interessante desenvolver um projecto para a implementação de um SBDM. Dessa forma, ter-se-ia, com certeza, uma melhor percepção dos problemas que surgem ao longo desse processo e uma melhor visão, mais real, de todos os conceitos, modelos, mecanismos e sistemas estudados nesta dissertação. Uma primeira tentativa já foi feita nesse sentido. Em [Cunha & Belo 2003] definiu-se um modelo para um sistema de ensino inteligente desenhado especialmente para ambientes móveis. Nesse projecto utilizou-se modelos e técnicas baseadas em agentes. Os agentes projectados serão capazes de emular algumas das capacidades e conhecimentos humanos em processos de aprendizagem. Através da utilização de agentes o sistema alcançará maior flexibilidade e robustez na resolução de problemas e melhores capacidades de adaptação a novas situações. Os agentes podem trabalhar como uma equipa, combinando as suas capacidades em sessões multidisciplinares, resolvendo problemas ou ajudando os alunos a resolver as tarefas que lhe foram propostas.

Com o grande avanço das tecnologias de comunicação móvel e da computação portátil, é de esperar que um sistema como o referido possa desfrutar da evolução destas tecnologias. Deste modo, um dia destes, poder-se-á presenciar algumas situações nas quais veremos, por exemplo, os alunos de uma instituição a migrarem para as suas estações portáteis a informação que necessitam para estudar um determinado assunto, desconectando-se logo de seguida para regressarem a sua casa e utilizarem as suas aplicações autonomamente. Mais tarde, se tiverem essa necessidade, voltarão a restabelecer uma conexão ao sistema anterior, através de uma ligação directa à rede fixa do através de um acesso a uma rede de comunicações em fios, para conciliar a informação que estiverem a manipular anteriormente. O uso de comunicações sem fios permitirá ainda que todos os alunos possam aceder a partir de qualquer ponto aos recursos de que necessitarem, desde que exista cobertura de rede sem fios nesses lugares. A generalização dos serviços de computação móvel será uma realidade a breve prazo, contribuindo para maior racionalização de recursos, autonomia de serviços e independência relativamente aos sítios em que estão situados as plataformas de serviços disponíveis.

Capítulo 11

Bibliografia

[Acharya & Zdonik 1993] S. Acharya and S. Zdonik, “An Efficient Scheme for Dynamic Data Replication”, Cs-93-43, Dept. of Computer Science, Brown University, September 1993. http://citeseer.nj.nec.com/acharya93efficient.html

[Ahmed et al. 1996] Khalil M. Ahmed, Mohamed A. Ismail, Nagwa M. El-Makky and Khaled M. Nagi, “A New Transaction Management Scheme for Mobile Computing Environments”, 1996 http://citeseer.nj.nec.com/477635.html

[Alonso & Korth 1993] Rafael Alonso and Henry F. Korth. “Database System Issues in Nomadic Computing”, In Proceedings of the 1993 SIGMOD Conference, Washington, May 1993. http://citeseer.nj.nec.com/alonso93database.html

[Antila et al. 2001] Thomas Antila, Jarmo Jokela and Lämsä Kenttälä, “A Survey on Data Management in Mobile Computing”, 2001

[Badrinath & Imiellinski 1992] B. R. Badrinath and T. Imielinski, "Replication and Mobility", Proc. of the 2nd Workshop on the Management of Replicated Data (WMRD-II), pp. 9-12, Monterey, CA, http://citeseer.nj.nec.com/101577.html

[Badrinath & Phatak 1997] B. R. Badrinath and Shirish Hemant Phatak, "An Architecture for Mobile Databases", Department of Computer Science, Rutgers University, http://citeseer.nj.nec.com/33669.html

[Badrinath & Phatak 1998] B. R. Badrinath and S. Phatak, "Database Server Organization for Handling Mobile Clients", Department of Computer Science, Technical Report DCS-TR-324, Rutgers University, New Jersey, http://citeseer.nj.nec.com/74745.html

[Badrinath & Phatak 1998] B. R. Badrinath and Shirish H. Phatak, "A Database Architecture for Handling Mobile Clients", 1998, http://citeseer.nj.nec.com/badrinath98database.html

[Baggio 1999] Aline Baggio, "Adaptable and Mobile-Aware Distributed Objects", Ph.D. Thesis, Universite Pierre et Marie Curie, June 1999, http://citeseer.nj.nec.com/baggio99adaptable.html

[Bagrodia et al.1995] R. Bagrodia, W. W. Chu, L. Kleinrock and G. Popek, "Vision, Issues, and Architecture for Nomadic Computing," IEEE Personal Commun. Mag., pp. 14--27, Dec. 1995, http://citeseer.nj.nec.com/bagrodia95vision.html

[Barbará 1999] Daniel Barbara, "Mobile Computing and Databases - A Survey", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, Jan/Feb, 1999, pp. 108-117, http://citeseer.nj.nec.com/barbara99mobile.html

[Bertino et al. 1997] Pegani, and G. P. Rossi, "Fault Tolerance and Recovery in Mobile Computing Systems", In Recovery Mechanisms in Database Systems (Eds. V. Kumar and M. Hsu), Prentice-Hall, 1997. http://citeseer.nj.nec.com/bertino97fault.html

[Bönigk & Lubinski 1996] Jörg Bönigk and Astrid Lubinski, "A Basic Architecture for Mobile Information Access", http://citeseer.nj.nec.com/365514.html

[Bouguettaya 1996] Athman Bouguettaya, "On the Construction of Mobile Database Management Systems", http://citeseer.nj.nec.com/100111.html

[Bukhres et al. 1997] O. Bukhres, S. Morton, P. Zhang, E. Vanderdijs, C. Crawley, J. Platt and M. Mossman, “A Proposed Mobile Architecture for Distributed Database Environment”, Technical Report, Indiana University, Purdue University, 1997. http://citeseer.nj.nec.com/bukhres97proposed.html

[Carneiro 2002] Alberto Carneiro, “Introdução à Segurança dos Sistemas de Informação”, 2002, FCA – Editora de Informática

[Chrysanthis 1993] Panos K. Chrysanthis, “Transaction Processing in Mobile Computing Environment”, In Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems, pages 77--83, October 1993. http://citeseer.nj.nec.com/chrysanthis93transaction.html

[Chrysanthis 1997] Panos Chrysanthis, “Consistent Checkpointing and Rollback Recovery for a Mobile Computing Environment”, Term Paper - CS3550, Advanced Topics in Management of Data, http://citeseer.nj.nec.com/370316.html

[Chrysanthis et al. 1998] P. K. Chrysanthis, G. Samaras and Y. Al-Houmaily, “Recovery and Performance of Atomic Commit Processing in Distributed Database Systems” (Ch. 13), http://citeseer.nj.nec.com/chrysanthis98recovery.html

[Connolly & Begg 1999] Thomas Connolly and Carolyn Begg, “Database Systems – A Practical Approach to Design, Implementation and Management”, 1999, Addison Wesley

[Cunha et al. 2000] Miguel Cunha, Nuno Preguiça, José Legatheaux Martins, Henrique João Domingos, Sérgio Marco Duarte, Francisco Moura, Carlos Barquero, “Mobisnap: Um Sistema de Base de Dados para Ambientes Móveis”, 2000

[Cunha & Belo 2003] Sílvia Cunha, Orlando Belo, “Bringing Mobility to Intelligent Tutoring Systems”, In Proceedings of the ECEL 2003, THE 2nd European Conference on eLearning", *Caledonian University*, *Glasgow, Scotland,* 6-7 November, 2003.

[Date 2000] C. J. Date, “An Introduction to Databases Systems”, 2000, Addison Wesley

[Demers et al. 1994] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M.M. Theimer, and B. Welch. “The Bayou Architecture: Support for Data Sharing among Mobile Users”. In Proceedings

Workshop on Mobile Computing Systems and Applications. IEEE, December 1994. http://citeseer.nj.nec.com/demers94bayou.html

[Dirckze & Gruenwald 2000] R. Dirckze e L. Gruenwald, "A Pre-serialization Transaction Management Technique for Mobile Multi-databases", To appear: Special Issue on Software Architecture for Mobile Applications, MONET 2000, http://citeseer.nj.nec.com/455886.html

[Dic] "Dicionário da Língua Portuguesa", Porto Editora, 6ª. Edição

[Dunham et al. 1997] Margaret H. Dunham, Abdelsalam Helal, Santosh Balakrishnan; "A Mobile Transaction Model that Captures Both the Data and Movement Behaviour", 1997, ACM/Baltzer Journal on Special Topics in Mobile Networks and Applications (MONET), 1997 http://citeseer.nj.nec.com/dunham97mobile.html

[Dunham & Kumar 1999] M.H. Dunham and V. Kumar, "Impact of Mobility on Transaction Management", In Proc. of MobiDE /Mobicom99 Workshop, Seattle, WA, pages 14--21. ACM, August 1999. http://citeseer.nj.nec.com/dunham99impact.html

[Elmagarmid et al. 1995] Ahmed Elmagarmid, Jin Jing e Omran Bukhres, "An Efficient and Reliable Reservation Algorithm for Mobile Transactions", 1995, Department of Computer Sciences, Purdue University, in Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM'95). http://citeseer.nj.nec.com/elmagarmid95efficient.html

[Faiz & Zaslavsky 1995] M. Faiz, A. Zaslavsky, "Database Replica Management Strategies in Multidatabase Systems with Mobile Hosts", Department of Computer Technology, Monash University, 1995, In Proceedings of the 6th International Hong Kong Computer Society Database Workshop, Mar. 1995. http://citeseer.nj.nec.com/faiz95database.html

[Focus 1977] "Focus, Enciclopédia Internacional", Volume III, Livraria Sá da Costa Editora, 1977

[Forman & Zahorjan 1994] George H. Forman and John Zahorjan. "The Challenges of Mobile Computing". IEEE Computer, 27(6), April 1994. http://citeseer.nj.nec.com/38782.html

[Gök 1999] Hüseyin Gökmen Gök; "Processing of Continuous Queries from Moving Objects in Mobile Computing Systems", 1999, http://citeseer.nj.nec.com/61668.html

[Gök & Ulusoy] Hüseyin Gökmen Gök and Özgür Ulusoy, “Transmission of Continuous Query results in Mobile Computing Systems”, http://citeseer.nj.nec.com/381428.html

[Gore & Ghosh 2000] M.M. Gore, R. K. Ghosh, “Recovery of Mobile Transactions”, 2000

[Gray et al. 1996] J. Gray, P. Helland, P. O'Neil, and D. Shasha. “The Dangers of Replication and a Solution”. In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Pages 173--182, June 1996. http://citeseer.nj.nec.com/gray96danger.html

[Gruenwald & Banik 2001] Le Gruenwald, Shankar M. Banik, “Power-Aware Technique to Manage Real-Time Database Transactions in Mobile Ad-Hoc Networks”, In Proceedings of the 4th International Workshop on Mobility in Databases and Distributed Systems, Munich, Germany, September 2001. http://citeseer.nj.nec.com/gruenwald01poweraware.html

[Guy et al. 1998] Richard Guy, Peter Reicher, David Ratner, Michial Gunter, Wilkie Ma, and Gerald Popek. “Rumor: Mobile Data Access Through Optimistic Peer-to-peer Replication. In Proceedings: ER'98 Workshop on Mobile Data Access”, 1998. http://citeseer.nj.nec.com/guy98rumor.html

[Heuer & Lubinski 2000] A. Heuer, A. Lubinski (2000) “Configured replication for mobile Applications”. In Proc. of the BalticDB&IS'2000. http://citeseer.nj.nec.com/heuer00configured.html

[Helal et al. 2001] Sumi Helal, Joachum Hammer, Jinsuo Zhang, Abhinav Khusharaj, “A Three-tier Architecture for Ubiquitous Data Access”, ACS/IEEE International Conference on Computer Systems and Applications, Beirut, Lebanon, June 2001, http://citeseer.nj.nec.com/helal01threetier.html

[Heuer & Lubinski 1996] A. Heuer and A. Lubinski. “Database Access in Mobile Environments” (extended version). In Preprint of the Dept. of CS, University of Rostock, CS-04-96, 1996. http://citeseer.nj.nec.com/article/heuer96database.html

[Hwang 2000] San-Yih Hwang, “On Optimistic Methods for Mobile Transactions”, Jouranl of Information Science and Engineering 16, 535-554, 2000

[Imieslinski & Badrinath 1993] Tomasz Imieslinski, B. R. Badrinath, “Data Management for Mobile Computing”, 1993

[Imielinski & Badrinath 1993A] T. Imielinski, B. Badrinath; "Querying in highly mobile distributed environments", 1993

[Imieslinski & Badrinath 1994] T. Imielinski and B. R.Badrinath, "Mobile wireless computing: Challenges in data management", Communications of the ACM,Vol. 37, No. 10, October 1994, pp. 19--28. http://citeseer.nj.nec.com/imielinski94mobile.html

[Jing et al. 1999] Jin Jing, Abdelsalam Helal, Ahmed Elmagarmid, "Client-Server Computing in Mobile Environments", 1999

[Joseph et al. 1997] A. D. Joseph, J. A. Tauber, and M. F. Kaashoek. "Mobile Computing with the Rover Toolkit" IEEE Trans. Comput., 46(3):337--352, Mar. 1997. http://citeseer.nj.nec.com/joseph97mobile.html

[Joseph et al. 1997A] A. D. Joseph and M. F. Kaashoek. "Building Reliable Mobile-Aware Applications Using Rover Toolkit", 1997, Wireless Networks 3, 5, 405-419

[Katz 1995] Randy H. Katz, "Adaptation and Mobility in Wireless Information Systems", 1994

[Kayan & Ulusoy 1999] E. Kayan and O. Ulusoy. "An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems". The Computer Journal, 42(6), 1999. http://citeseer.nj.nec.com/kayan99evaluation.html

[Kistler & Satyanarayanan 1992] James Kistler, M. Satyanarayanan, "Disconnected Operation in the Coda File System"

[Kottkamp & Zukunft 1998] H. Kottkamp and O. Zukunft "Location-Aware Query Processing in Mobile Database Systems", In Proc. of the ACM Symposium on Applied Computing, Feb. 1998. http://citeseer.nj.nec.com/kottkamp98locationaware.html

[Lauzac & Chrysanthis 1998] S. W. Lauzac and P. K. Chrysanthis, "Utilizing Versions of Views within a Mobile Environment", In Proceedings of the 9th International Conference on Computing and Information, June 1998. http://citeseer.nj.nec.com/weissmanlauzac98utilizing.html

[Lee & Helal 2002] Minsoo Lee and Sumi Helal, "HiCoMo: High Commit Mobile Transactions", Distributed and Parallel Databases, 11, 73-92, 2002

[Lello 1988] "Lello Universal", Volume 2, Lello & Irmão Editores, Porto, 1988

[Lin & Dow 2001] Cheng-Min Lin and Chyi-Ren Dow, "Efficient Checkpoint-based Failure Recovery Techniques in Mobile Computing Systems", Journal of Information Science and Engineering, N.º 17, pg. 549-573, 2001

[Liu et al. 1996] George Y. Liu, Aleksander Marlevi, Gerald Q. Magure Jr., "A Mobile Virtual-distributed System Architecture for Supporting Wireless Mobile Computing and Communications", 1996

[Liu et al.1999] Peng Liu, Paul Ammann, and Sushil Jajodia. "Incorporating Transaction Semantics to Reduce Reprocessing Overhead in Replicated Mobile Data Applications". In ICDCS'99: Proceedings 19th IEEE International Conference on Distributed Computing Systems, Austin, TX, June 1999 http://citeseer.nj.nec.com/liu99incorporating.html

[Lubinski 1997] Astrid Lubinski, "Adaptation Concepts for Mobile Database Security", http://citeseer.nj.nec.com/356715.html

[Lubinski 1998] Astrid Lubinski, "Security Issues in Mobile Database Access", 1998, In Proceedings of the IFIP WG 11.3 Twelfth Int. Conf. on Database Security. http://citeseer.nj.nec.com/lubinski98security.html

[Lubinski 2000] Astrid Lubinski "Database Security Meets Mobile Requirements", 2000, In Proceedings International Symposium on Database Technology Software Engineering, WEB and Cooperative Systems, Baden. http://citeseer.nj.nec.com/lubinski00database.html

[Luccio et al. 2000] Flaminia L. Luccio, Alberto Bartoli, Giuseppe Anastasi, "A Fault-Tolerance Support for Mobile Wireles Systems", in the proceedings of WSDAAL, Italian Workshop on Sistemi Distribuiti: Algoritmi, Architetture, e Linguaggi, Ischia (Napoli), Italy, 18-20 September 2000. http://citeseer.nj.nec.com/471357.html

[Madria 1998] Sanjay Kumar Madria, "Transactions Models for Mobile Computing", 1998

[Madria 1998A] Sanjay Kumar Madria and Bharat K. Bhargava, "On the Correctness of a Transaction Model for Mobile Computing", Database and Expert Systems Applications, pg. 573-583, 1998, http://citeseer.nj.nec.com/3380.html

[Madria et al. 1998] Sanjav Kumar Madria, Mukesh Mohania e John F. Roddick, "A Query Processing Model for Mobile Computing using Concept Hierarchies and Summary Databases", 1998, http://citeseer.nj.nec.com/30226.html

[Mazumdar & Chrysanthis 1999] Subhasish Mazumdar, Panos K. Chrysanthis, "Achieving Consistency in Mobile Databases through Localization in PRO-MOTION", Department of Computer Science, New Mexico Tech, Department of Computer Science, University of Pittsburgh, 1999, In Proc. DEXA Intl. Workshop on Mobility in Databases and Distributed Systems, pp. 82-89. http://citeseer.nj.nec.com/mazumdar99achieving.html

[Moderna 1987] "Moderna Enciclopédia Universal", Tomo XIV, Lexicoteca, Círculo de Leitores, Edição n.º 1630, 1987

[Noble 1998] Brian D. Noble, "Mobile Data Access", 1998

[Noble 2000] Brian Noble, "System Support for Mobile, Adaptative Applications", IEEE Personal Communications, February 2000

[Nova 1998] "Nova Enciclopédia Larrousse", Volume 16, Círculo de Leitores, Edição n.º 3901, 1998

[Özsu & Valduriez 1999] M. Tamer Özsu e Patrick Valduriez, "Principles of Distributed Databases", International Edition, 2ª. Edição, 1999

[Park et al. 2001] Taesoon Park, Nanyoon Woo, Heon Yeom, "A Region-Based Recovery Scheme for the Mobile Computing Systems", International Conference on Parallel and Distributed Computing and Systems, August, 2001

[Park et al. 2002] Taesoon Park, Namyoon Woo, Heon Yeom, "An Efficient Optimistic Message Logging Scheme for the Recoverable Mobile Computing Systems", IEEE Transactions on Mobile Computing 1(4): 265-2777, 2002

[Park et al. 2001] T. Park, N. Woo, H.Y. Yeom, “An Efficient Recovery Scheme for Mobile Computing Environments”, Proc. of International Conference on Parallel and distributed Systems, 2001. http://citeseer.nj.nec.com/park01efficient.html

[Perich et al. 2001] Filip Perich, Sasikanth Avancha, Anupam Joshi, Yelena Yesha, Karuna Joshi, “Query Routing and Processing in Mobile Ad-hoc Environments”, November 2001

[Phatak & Badrinath 1996] Shirish Hemant Phatak, B.R. Badrinath, “Multiversion Reconciliation for Mobile Databases”, Department of Computer Science, Rutgers University, 1996, http://citeseer.nj.nec.com/520374.html

[Phatak & Badrinath 1998] Shirish Phatak, B. R. Badrinath, “Data Partitioning for Disconnected Client Server Databases”, 1998, http://citeseer.nj.nec.com/515808.html

[Phatak & Badrinath 1999] Shirish Hemant Phatak, B. R. Badrinath, “Conflict Resolution and Reconciliation in Disconnected Databases”, Department of Computer Science, Rutgers University, 1999, In Proceedings of MDDS 1999, September 1999. http://citeseer.nj.nec.com/phatak99conflict.html

[Pitoura 1996] Evaggelia Pitoura, “A Replication Schema to Support Weak Connectivity in Mobile Information Systems”, Computer Science Department, University of Ioannina, 1996, In Proc. of 7th Intl. Conf. on Database and Expert System Applications (DEXA), September 1996. http://citeseer.nj.nec.com/pitoura96replication.html

[Pitoura & Bhargava 1994] Evaggelia Pitoura, Bharat Bhargava, “Building Information Systems for Mobile Environments”, Department of Computer Science, Purdue University, 1994, Proceedings of 3rd International Conference on Information and Knowledge Management, pp.371-378, http://citeseer.nj.nec.com/pitoura94building.html

[Pitoura & Bhargava 1994A] Evaggelia Pitoura, Bharat Bhargava, “Revising Transaction Concepts for Mobile Computing”, In Proceedings of the 1st IEEE Workshop on Mobile Computing Systems and Applications (MCSA94), pages 164--169, 1994. http://citeseer.nj.nec.com/pitoura94revising.html

[Pitoura & Bhargava 1995] Evaggelia Pitoura, Bharat Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environments", Technical Report TR-94-025, Purdue University, Dept. of Comp. Sciences, 1995, http://citeseer.nj.nec.com/pitoura95maintaining.html

[Pitoura & Bhargava 1997] Evaggelia Pitoura, Bharat Bhargava, "Data Consistency in Intermittently Connected Distributed Systems", Technical Report DCS-96-10 (Revised 7/97), Department of Computer Science, University of Ioannina, 1997. http://citeseer.nj.nec.com/article/pitoura97data.html

[Pitoura & Fudos 1998] E. Pitoura and I. Fudos, "An Efficient Hierarchical Scheme for Locating Highly Mobile Users", In Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98), November 1998. http://citeseer.nj.nec.com/article/pitoura98efficient.html

[Pitoura & Samaras 1998] Evaggelia Pitoura e George Samaras, "Data Management for Mobile Computing", Kluwer Academic Publishers, 1998

[Pradhan et al. 1996] Dhiraj Pradhan, P. Krishna, Nitin Vaidya, "Recovery in Mobile Wireless Environment: Design and Trade-off Analysis", 1996

[Preguiça et al. 2000] Nuno Preguiça, J. Legatheaux Martins, Henrique João, Sérgio Duarte, Carlos Barquero, Francisco Moura, Rui Oliveira, Orlando J. Pereira, "Mobile Transaction Management in Mobisnap", 2000, In Proc. of ADBIS-DASFAA, Setembro de 2000, http://citeseer.nj.nec.com/316163.html

[Rakotoniarainy 1998] Andry Rakotoniarainy; "Adaptable Transaction Consistency for Mobile Environments", 1998, In Proc. of 9th Intl. Conf. on Database and Expert System Applications (DEXA), pp. 440-445. http://citeseer.nj.nec.com/410658.html

[Ranganathan 2001] Kavitha Ranganathan, "Design and Evaluation of Dynamic Replication Strategies", http://citeseer.nj.nec.com/450678.html

[Ratner et al. 1996] David Ratnet, Gerald J. Popek, Peter Reiber, "The Ward Model: A Replication Architecture for Mobile Environments", Technical Report CSD-960045, University of California, Los Angeles, December 1996.

[Ratner et al. 2001] David Ratner, Peter Reiher, Gerald Popek, Geoffrey Kuenning, "Replication Requirements in Mobile Environments", in Mobile Networks and Applications, 2001

[Saito 2000] Yasushi Saito, "Optimistic Replication Algorithms", 2000

[Saito & Saphiro 2002] Yasushi Saito and Marc Saphiro, "Replication: Optimistic Approaches", 2002

[Samaras et al. 2000] George Samaras, Paraskevas Evripidou and Evaggelia Pitoura, "A Mobile-Agent Based Infrastructure for eWork and eBussiness Applications", EMMSEC 2000

[Satyanarayanan 1996] M. Satyanarayanan, "Mobile Information Access" IEEE Personal Communications, vol.3, no.1, pp. 2633, 1996. http://citeseer.nj.nec.com/satyanarayanan96mobile.html

[Segun et al. 2001] K. Segun, A. R. Hurson, V. Desai, A. Spink and L. L. Miller, "Transaction Management in a Mobile Data Access System", 2001

[Seydim 1999] Ayse Yasemin Seydim, "An Overview of Transaction Models in Mobile Environments", 1999, http://citeseer.nj.nec.com/465597.html

[Singh & Cabillic 2003] Pushpendra Singh, Gilbert Cabillic, "Fault Tolerance and Availability in Mobile Computing Environment"

[Terry et al. 1994] D. B. Terry, A. Demers, K. Petersen, M. Spreitzer, M. Theimer, B. Welch, "Session Guarentees for Weakly Consistent Replicated Data", In Proceedings of 3rd International Conference n Parallel and Distributed Information Systems

[Terry et al. 1995] D. B. Terry, A. Demers, K. Petersen, M. Spreitzer, M. Theimer, C. H. Hauser, "Managing Update Conflits in *Bayou,* a Weakly Connected Replicated Storage System", ACM SIGOPS Oper. Syst., pp. 172-182, December 1995

[Verhofstad 1978] Joost S. M. Verhofstad, "Recovery Techniques for Database Systems", Computing Surveys, Vol. 10, N.º 2, pp. 167-195, June 1978

[Walborn & Chrysanthis 1995] Walborn, G. D., Chrysanthis, P. K., "Supporting Semantics-Based Transaction Processing in Mobile Database Applications", in proceedings of 14th IEEE Symposium on Reliable Distributed Systems, pp.31-40, Sept.1995. http://citeseer.nj.nec.com/walborn95supporting.html

[Walborn & Chrysanthis 1997] Gary D. Walborn, Panos K. Chrysanthis, "PRO-MOTION: Management of Mobile Transactions", 1997, In Proc. of 11th ACM Annual Symposium on Applied Computing, Personal Technologies 1(3):171-181 http://citeseer.nj.nec.com/walborn97promotion.html

[Wiesmann et al. 2000] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. "Understanding Replication in Databases and Distributed Systems". In Proceedings of 20th International Conference on Distributed Computing Systems (ICDCS'2000), pages 264-274, Taipei, Taiwan, R.O.C., April 2000. IEEE Computer Society Los Alamitos California. http://citeseer.nj.nec.com/wiesmann00understanding.html

[Wolfson et al. 1997] O. Wolfson, S. Jajodia, and Y. Huang. "An Adaptive Data Replication Algorithm". ACM Transactions on Database Systems (TODS), Vol. 22(4), June 1997, pp. 255-314. http://citeseer.nj.nec.com/wolfson97adaptive.html

[Wolfson et al. 1999] O. Wolfson, P. Sistla, S. Chamberlain and Y. Yesha, "Updating and Querying Databases that Track Mobile Units", To appear in a special issue of the Distributed and Parallel Databases Journal, 1999, http://citeseer.nj.nec.com/91685.html

[Yeo et al. 1996] L. H. Yeo, P. Steele, J. Han, "Linear Orderability of Transactions in Mobile Environment with Heterogeneous Databases", ICCI'96 8th International Conference of, in W.W. Koczkodaj (ed), *ICCI'96 8th International Conference of Computing and Information*, Ontario Canada, 19-22 June 1996, Journal of Computing and Information, Canada, 707 - 724, 1996. http://citeseer.nj.nec.com/yeo96linear.html

[Zaslavsky & Tari 1998] Arkady Zaslavsky, Zahir Tari, "Mobile Computing: Overview and Current Status", Australian Computer Journal, 30, 1998. 7 http://citeseer.nj.nec.com/zaslavsky98mobile.html