UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EDSON CARLOS DA SILVA

UM ESTUDO DOS PRINCIPAIS MODELOS DE TRANSAÇÕES EM BANCO DE DADOS MÓVEIS E UMA PROPOSTA DIFERENCIADA DO MODELO PRO-MOTION

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Murilo Silva de Camargo

Florianópolis, outubro de 2003

UM ESTUDO DOS PRINCIPAIS MODELOS DE TRANSAÇÕES EM BANCO DE DADOS MÓVEIS E UMA PROPOSTA DIFERENCIADA DO MODELO PRO-MOTION

Edson Carlos da Silva

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Concentração Sistemas de Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____
Prof. Dr. Fernando O. Gautier
(Coordenador do Curso)

_____
Prof. Dr. Murilo Silva de Camargo
(Orientador e Presidente da Banca)

Banca Examinadora

_____
Prof. Dr. Mário Antonio Ribeiro Dantas

_____
Prof. Dr. Rosvelter João Coelho da Costa

_____
Prof. Dr. Vitório Bruno Mazzola

Às minhas filhas Caroline, Daniele e Gabriele,
pela compreensão e carinho.
À minha esposa Lucélia pela paciência e dedicação.

AGRADECIMENTOS

Ao meu orientador e amigo Murilo Silva de Camargo, Dr., pela confiança dispensada, pelo apoio, pela cobrança e sua disposição para enriquecer o meu trabalho.

Aos demais professores do departamento de computação da UFSC, pela acolhida, pelos conhecimentos a mim transmitidos.

A minha família pelo incentivo e apoio nas horas mais difíceis.

Aos amigos e colegas que de certa forma me ajudaram, com materiais, artigos e outros materiais importantes para minhas pesquisas.

Por fim, agradeço a DEUS pela graça de colocar em meu caminho muitas pessoas que de alguma forma me ajudaram na realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVEATURAS

2PC	Two Phase Commit
2PL	Two Phase Locking
3PC	Three Phase Commit
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
CSI	Client Side Intercept
DAA	Data Access Agent
DLM	Data Location Mapping
ECG	Esquema Conceitual Global
ECL	Esquema Conceitual Local
EE	Esquemas Externos
EIL	Esquema Interno Local
FLM	Fragment Location Mapping
FTP	File Transfer Protocol
GT	Global Transaction
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
IPV4	Internet Protocol versão 4
IPV6	Internet Protocol versão 6
LCD	Liquid Crystal Display
LT	Local Transaction
LTM	Local Transaction Manager
MD	Mobile Database
MDSTPM	Multi Database System Transaction Process Model
MIDP	Mobile Information Device Profile
MQF	Message Queuing Facility
MTM	Mobile Transaction Manager
PDA	Personal Digital Assistants
RPC	Remote Procedure Call

SGBD	Sistema Gerenciador de Banco de Dados
SGBDD	Sistema Gerenciador de Banco de Dados Distribuído
SGBDM	Sistema Gerenciador de Banco de Dados Móveis
SGBDM	Sistema Gerenciador de Banco de Dados Móveis
SSI	Server Side Intercept
SVBD	Sistema de Vários Bancos de Dados
TCP/IP	Transport Control Protocol / Internet Protocol
TO	Timestamp Ordering
WAP	Wireless Application Protocol
WMP	Wireless Markup Language
WTP	Wireless Transaction Protocol
XML	Extensible Markup Language
XSL	XML Style Sheets

RESUMO

Este trabalho apresenta um estudo sobre as principais arquiteturas e modelos de transações móveis. Faz uma revisão sobre os principais problemas encontrados no ambiente móvel, suas características, modelos e propriedades da transação, bem com uma revisão sobre as transações convencionais dos Sistemas Gerenciadores de Banco de Dados Distribuídos – SGBDD – fixos. Estuda as principais arquiteturas de software para o ambiente móvel, as principais arquiteturas de Sistemas Gerenciadores de Banco de Dados Móveis – SGBDM. Apresenta as características e graus de uma transação móvel, os efeitos das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) na transação móvel e sua definição formal. Selecionam-se os principais modelos de transações móveis e classifica-os de acordo com as propriedades ACID e seus respectivos modelos de execução. Apresentamos um estudo mais detalhado do modelo de transação móvel PRO-MOTION, investigando os aspectos relevantes. Desta forma, modificações para o modelo de transação PRO-MOTION é proposta a fim de melhorar a eficiência na distribuição dos dados, diminuindo o consumo de recursos da unidade móvel e a latência dos dados na rede no controle e na gerência da transação.

ABSTRACT

This work presents a study about the main architectures and models of mobile transactions. It makes a revision about the problems found in mobile environment, its characteristics, models and properties of conventional transactions for fixed Distributed Data Base Management Systems – DDBMS. It shows the main software architectures for mobile environment and Mobile Data Base Management Systems – MDBMS. It also reports the features and degrees of mobile transactions and the ACID (Atomicity, Consistency, Isolation and Durability) properties effects in it. The main mobile transactions models are selected and classified according the ACID properties and its respective execution model. We take a more detailed study of PRO-MOTION model of mobile transactions, exploring its relevant aspects. Modifications are proposed in PRO-MOTION model to make a efficient management of transaction, decreasing the consume of resources and band width of network.

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

Estamos assistindo ao surgimento de uma nova era para a computação. Pessoas necessitadas de informações, empresas buscando informações em tempo real. Neste novo ambiente faz-se necessário estar permanentemente online com os sistemas de computação, independentemente do tempo e lugar. A solução para estes requisitos é a utilização de um sistema móvel, que possa comunicar-se com os demais computadores, localizados na rede fixa, transferindo dados para seus computadores pessoais, PDAs (*Personal Digital Assistants*), *laptops* ou *palmtops* em qualquer lugar e a qualquer momento.

Fatores como a evolução dos circuitos integrados, onde diminui-se o tamanho e aumenta-se o poder de processamento, aliado as melhorias das tecnologias de comunicações sem fio, permite a utilização de computadores pessoais móveis. A diminuição do tamanho e, consequentemente, do peso, tornou os PDAs comuns entre as pessoas. A medida em que a tecnologia móvel evolui, o preço tende a diminuir e o uso da computação móvel será cada vez mais acessível às pessoas.

É comum andarmos nas ruas e constatarmos o grande uso de telefones celulares. Estes aparelhos vêm desenvolvendo-se com uma velocidade espantosa. Juntamente com o desenvolvimento dos aparelhos celulares, acontece a evolução das redes celulares. A primeira geração de celulares transmitem os dados a 9,6 Kbps, sendo a grande maioria telefones analógicos. Na segunda geração de telefones celulares, incorporaram-se as tecnologias digitais, transmitindo dados a 14,4 Kbps. Atualmente, está à disposição dos usuários o telefone de terceira geração, transmitindo dados até 144Kbps. Para um futuro muito próximo os telefones de quarta geração que podem chegar a até 100Mbps.

Com a evolução dos sistemas celulares, também tem-se a evolução e expansão das redes de comunicação sem fio. Percebe-se que estas redes podem suportar os telefones celulares e também os dispositivos móveis. À medida que a infra-estrutura de

comunicação aumenta, cresce o número de usuários com dispositivos móveis acessando informações em qualquer lugar a qualquer tempo.

A computação móvel distingue-se da computação fixa pela mobilidade do usuário e dos computadores. A mobilidade destes usuários implica em uma maior complexidade para os sistemas de computação. Problemas de comunicação podem acontecer freqüentemente, caso das desconexões voluntárias. Devido ao seu tamanho reduzido, os equipamentos móveis possuem baterias de curta duração, reduzindo assim o tempo de trabalhos destes equipamentos.

Os bancos de dados usados na computação fixa não estão preparados para o problema da mobilidade. A pesquisa nesta área é intensa e já produziram várias tecnologias capazes de suportar ambos: os usuários fixos e os usuários móveis.

O presente trabalho tem por objetivo estudar os aspectos relacionados aos modelos de transações móveis, suas vantagens e restrições, elaborando uma classificação dos principais modelos.

1.2 METODOLOGIA

Para a realização deste trabalho, foram pesquisadas diversas bibliografias, tais como: livros, teses, dissertações, artigos científicos, relatórios técnicos, documentos oficiais de congressos, *workshops* e sites da Internet. Os materiais utilizados foram obtidos através de pesquisas na Internet em bibliotecas digitais como a biblioteca digital da ACM.

1.3 LIMITAÇÕES DA PESQUISA

O presente trabalho limita-se a pesquisa em modelos de transações para redes com infra-estruturas, isto é, somente para as redes onde existe iteração entre unidades móveis e unidades fixas.

1.4 ESTRUTURA DA DISSERTAÇÃO

O presente trabalho está dividido em 9 capítulos. O capítulo 2 apresenta uma visão sobre a computação móvel, sua evolução, os principais equipamentos utilizados, as características dos ambientes de computação móvel, as diversas aplicações da computação móvel, vantagens da mobilidade, os aspectos relacionados à mobilidade e por fim os aspectos de segurança.

No capítulo 3 faz-se uma revisão sobre os Sistemas Gerenciadores de Banco de Dados (SGBD) convencionais, os SGBD distribuídos e suas arquiteturas.

O capítulo 4 revisa os conceitos de transações para os SGBD convencionais. Estuda os conceitos de transação, seus modelos, suas propriedades como: Atomicidade, Consistência, Isolamento e Durabilidade, além do controle distribuído da concorrência.

No capítulo 5 abordam-se os bancos de dados móveis, as arquiteturas, os subsistemas de comunicação, o gerenciamento da informação, a gerência da localização dos dados, o processamento de consultas, a replicação de dados, o gerenciamento de transações e a recuperação de falhas. Estudam-se também os efeitos da mobilidade sobre as arquiteturas de banco de dados móveis e por fim faz a definição formal de uma transação móvel.

No capítulo 6 estudam-se os principais modelos de gerenciamento de transações móveis. Apresenta os modelos mais discutidos na comunidade científica, apresenta o modelo de execução de uma transação, suas definições formais e aspectos relevantes a seu funcionamento.

No capítulo 7 faz-se a classificação dos modelos de transações estudados. As classificações são feitas conforme o modelo de execução da transação e de acordo com cada propriedade ACID. Faz-se, também, um resumo dos principais modelos encontrados na literatura.

No capítulo 8 apresenta-se uma proposta de mudança na estrutura do modelo de gerenciamento de transação PRO-MOTION, melhorando sua eficiência no controle dos dados e das propriedades ACID.

Por fim, o capítulo 9 apresenta as conclusões finais deste trabalho, a relevância do estudo para a comunidade e as perspectivas de trabalhos futuros.

2 COMPUTAÇÃO MÓVEL

O crescimento nas áreas de comunicação celular, redes locais sem fio, serviços via satélite, sistemas operacionais e equipamentos de computação permitiram que informações e recursos fossem acessados utilizando computadores pessoais e/ou PDAs (MATEUS & LOUREIRO, 1998).

Para (TANENBAUM, 1997) “Algumas pessoas chegam a acreditar que, no futuro, só haverá dois tipos de comunicação: as comunicações por fibra e as sem fio”.

Com estes novos equipamentos surge um novo paradigma de computação distribuída, a Computação Distribuída Móvel. Para (TEWARI & GRILLO, 1995), “a computação móvel é um tipo de computação distribuída”. A computação distribuída até então era formada por várias máquinas, cada qual com seu respectivo endereço, portanto a troca de mensagens dá-se de forma transparente.

Na computação móvel os endereços das máquinas são dinâmicos, isto é, cada máquina poderá ter um endereço diferente em diferentes instantes, devido à mobilidade da máquina.

A grande parte dos sistemas distribuídos em operação é baseada no protocolo TCP/IP (*Transport Control Protocol / Internet Protocol*) (TANENBAUM, 1995). Este protocolo necessita de um endereço de origem e um endereço de destino para que aconteça a troca de mensagens na rede. O endereço não pode ser mudado durante o tempo em que as duas partes envolvidas estão trocando informações. A troca do endereço causa a perda de conexão.

Com a computação móvel surgem novos conceitos de computação distribuída, formas de endereçamento, meios físicos e lógicos de transmissão de dados, além de novas aplicações. Com estes novos requisitos de computação surgem muitos problemas como a desconexão involuntária, a necessidade de reduzir o consumo de energia (visto que os equipamentos móveis possuem pequenas baterias), a disponibilidade das informações, a independência de localização, entre outros. O desafio principal da computação móvel é garantir os mesmos serviços, com a mesma qualidade encontrada com os computadores fixos nos computadores móveis.

No início dos anos 90 mais de um milhão de *notebooks* foram vendidos no mercado mundial (HELAL ET AL, 1999). Estes equipamentos eram dotados de LCDs (*Liquid Crystal Display*) monocromáticos e eram usados por profissionais viajantes. As transferências de dados eram feitas por disquetes. Os PDAs começam a chegar ao mercado a partir de 1993. O custo era alto e a venda muito baixa. Nos dois primeiros anos os PDAs não tinham um custo razoável. Os consumidores demandavam utilidades que a primeira linha de PDAs não possuía (MARINHO, 2001).

Os equipamentos portáteis atuais possibilitam uma mobilidade maior para os usuários. Os *notebooks* e os PDAs estão cada vez menores e mais leves, podendo um *notebook* pesar apenas poucos kilogramas. Com eles é possível armazenar vários *gigabytes* e possuem poder de processamento maior que um computador *desktop*, em alguns casos (TAURION, 2002).

2.1 AMBIENTE MÓVEL

Existem dois modelos possíveis para o ambiente móvel: as redes móveis com infra-estruturas e as redes sem infra-estruturas (redes *ad hoc*). Nas redes *ad hoc*, uma estação móvel pode comunicar-se com a outra diretamente dentro do seu raio de atuação, ou pode usar uma outra como repetidora do sinal. Nas redes com infra-estruturas as unidades móveis comunicam-se através de uma estação de suporte à mobilidade, conforme figura 1.

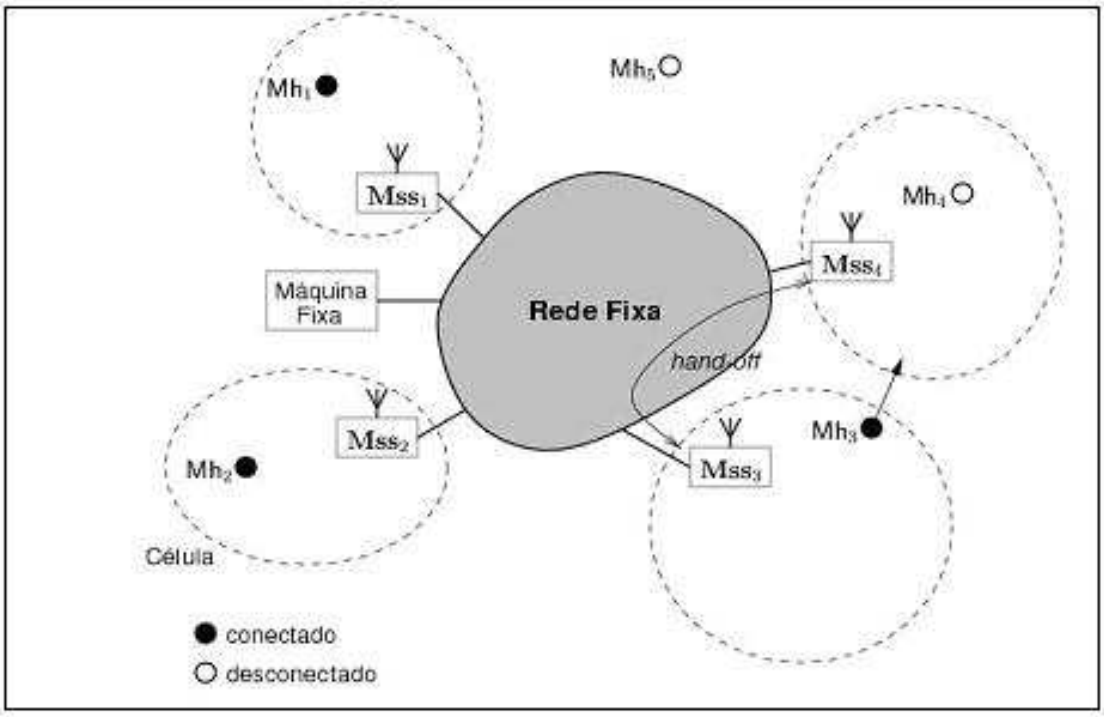
Os principais termos utilizados para o ambiente móvel são (ROCHA, 2001):

- Rede Fixa: rede formada por elementos convencionais ou fixos conectados fisicamente;
- Máquina Fixa: estação conectada fisicamente à rede. Esta estação possui localização geográfica e endereço de rede fixo;
- Estação móvel ou unidade móvel Mh (*Mobile host*): Estação que não possui uma localização geográfica e não tem endereço físico fixo;
- Estação base ou Mss (*Mobile Station Support*): Estação conectada à rede física

e responsável pela comunicação com todos os Mhs dentro de uma determinada área de abrangência chamada célula;

- Célula: área geográfica sob a responsabilidade de uma Mss;
- Canal sem fio: canal de comunicação entre um Mss e uma Mh ou ainda entre dois Mhs. Este canal não utiliza meios guiados (cabos) e sim onda de rádio para a comunicação.
- *Hand-off*: processo de transferência de responsabilidade de uma Mss para outra durante o movimento de um Mh. Este processo engloba toda a troca de mensagens para a delegação de responsabilidades entre as Mss envolvidas.

Figura 1: Modelo de sistema para computação Móvel



Fonte: (Rocha, 2001, pp. 14)

2.1.1 Tipos de Redes

2.1.1.1 Redes Sem Infra-Estrutura

Nesta topologia não existe uma infra-estrutura fixa. Todos os componentes são móveis. Caso exista alguma estação fixa, esta é tratada como móvel. Todos os componentes comunicam-se através das unidades móveis participantes. Se uma estação deseja comunicar-se com outras que não estão ao seu alcance, esta irá utilizar-se de outras estações móveis como roteador para que suas mensagens sejam encaminhadas, dispensando assim qualquer infra-estrutura fixa. Este tipo de rede, sem infra-estrutura, é chamada também de rede *ad hoc***.**

O grande desafio neste ambiente é encontrar uma solução ótima para o problema de roteamento. Como todas as unidades móveis conectam-se aos seus vizinhos e todos são repetidores dos sinais, a unidade móvel A fala diretamente com a unidade móvel C e com a unidade móvel E, conforme a figura 2. Caso seja necessário trocar informações entre as unidades móveis A e B, será necessário usar C ou E como seu retransmissor. C por sua vez necessitará de um algoritmo eficiente para encontrar B e assim por diante.

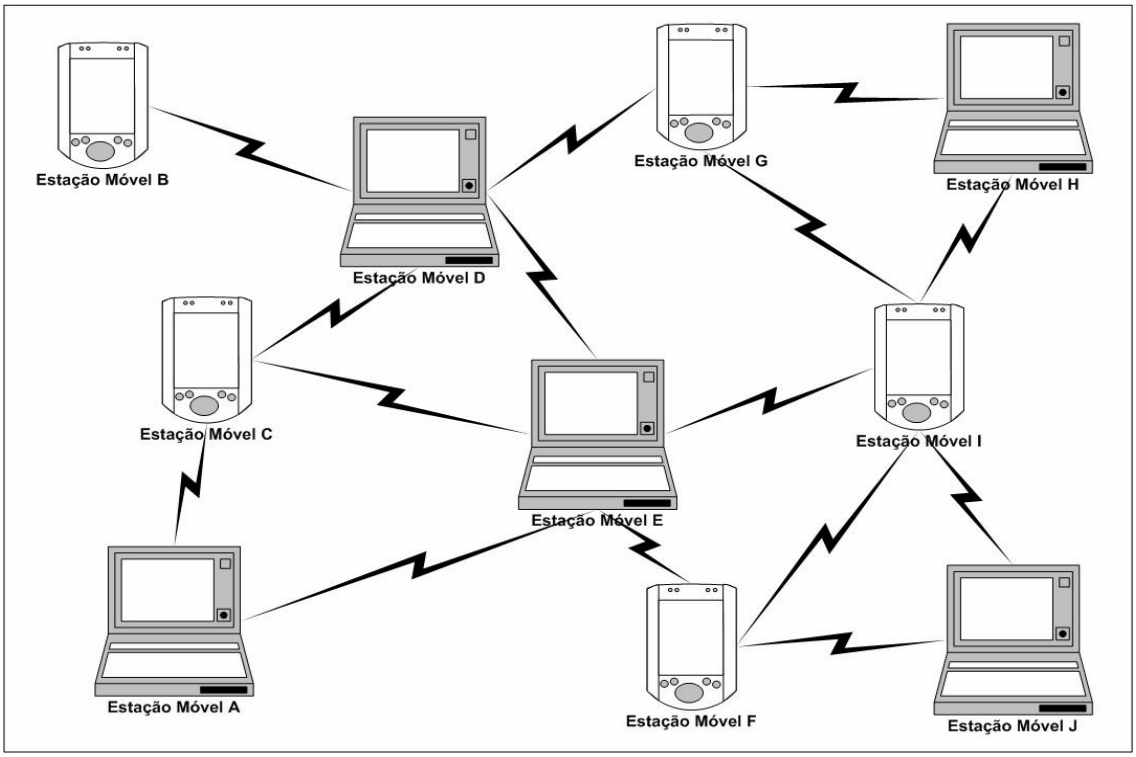
As redes *ad hoc* são indicadas para os casos onde a instalação de uma infra-estrutura é inviável ou impossível. Podem ser usadas por um exército em movimento no campo de batalha inimigo, onde uma infra-estrutura não existe, ou em casos de catástrofes onde não sem tem tempo hábil para a instalação de uma infra-estrutura.

Todas as unidades móveis participantes da rede movem-se de maneira arbitrária. Assim, torna-se difícil determinar sua topologia. Neste ambiente as taxas de erros são elevadas, desconexões são freqüentes e a largura de banda é menor que as encontradas na rede fixa. Apesar disto esta arquitetura possui inúmeras vantagens, como:

- Não dependem de infra-estrutura, entram em operação assim que as unidades móveis iniciam suas atividades;
- A conectividade é maior, desde que os elementos estejam dentro dos limites de alcance;

- A rede configura-se de maneira dinâmica, aumentando assim a tolerância à falhas;
- Existe maior mobilidade entre as unidades móveis;

Figura 2: Redes *Ad-Hoc*

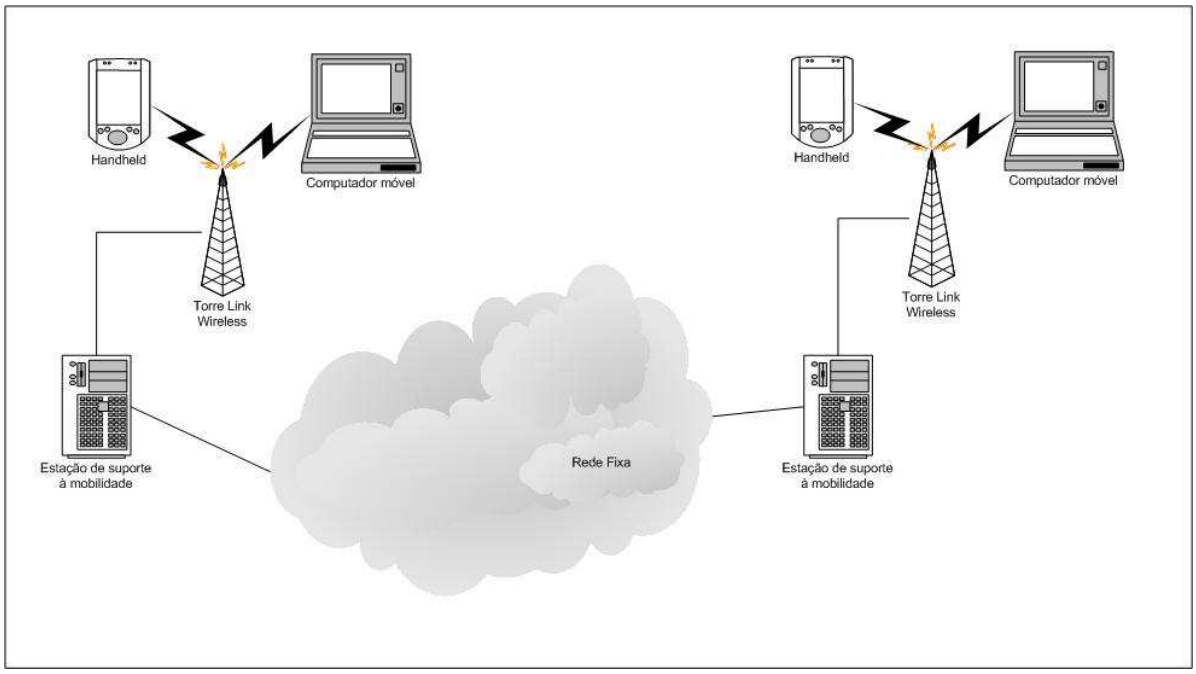


2.1.1.2 Redes com Infra-Estrutura

Nesta topologia as unidades móveis comunicam-se através de pontos de acessos instalados. Assim uma unidade móvel comunica-se com a outra através de uma unidade móvel intermediária, conforme figura 3. A rede fixa é responsável pelo suporte às unidades móveis, através de uma estação de suporte à mobilidade, executando as tarefas de localização da unidade móvel, de roteamento, redução de tráfego, adaptabilidade, etc. Estas redes podem utilizar os padrões de comunicação da família IEEE 802.11, em

conjunto com os outros padrões como o IEEE 802.3, sinais de satélite, etc.

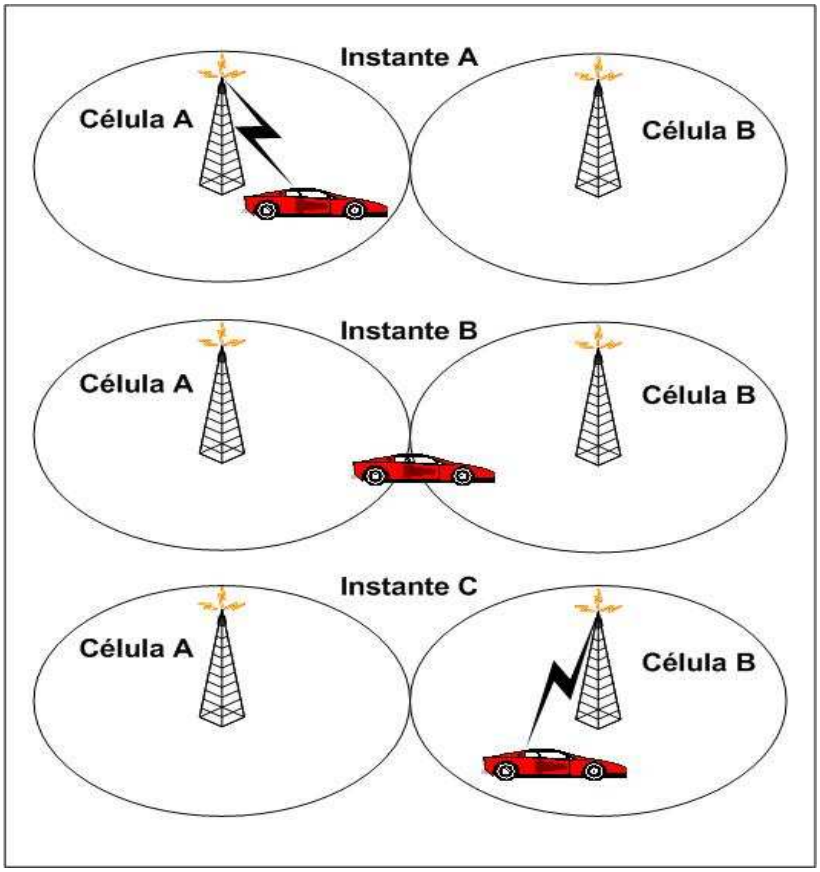
Figura 3: Redes *Wireless* com Infra-Estrutura



2.1.2 Handoff

As unidades móveis, em uma rede com infra-estrutura, captam o sinal de uma torre. Esta torre possui uma área de abrangência denominada célula. Quando a unidade móvel se movimenta de uma célula para a outra, executando aplicações, tem-se com este movimento um processo chamado *handoff*. *Handoff* é um processo entre duas estações bases adjacentes que tem por objetivo garantir uma transação enquanto o usuário se movimenta de uma célula para outra. O processo de *handoff* é totalmente transparente para o usuário, conforme ilustrado na figura 4.

Figura 4: Processo de *handoff*



Uma unidade móvel, localizada na célula A, encaminha-se para a célula B, no instante A. Existe um momento no instante B em que a unidade móvel está na fronteira entre as duas células, ocorrendo desconexão da célula A para após ocorrer a conexão com a célula B. No instante C, a unidade móvel já está conectada a célula B. No instante B ocorre o *handoff*.

2.1.3 Protocolos

O ambiente de computação móvel implica em comunicação com o uso de uma rede sem fios. Esta característica introduz características que influenciam diretamente nos protocolos de rede. Os canais sem fio possuem elevadas taxas de erros. Para permitir a mobilidade dos equipamentos os protocolos devem implementar mecanismos

capazes de tratar estes erros, além de gerenciar a localização e a qualidade da comunicação. Os dispositivos móveis possuem características limitadas e por isso devem gerenciar a adaptabilidade aos protocolos e aplicações. Discussões a respeito destes problemas podem ser encontradas em (MATEUS & LOUREIRO, 1998).

Devido a estas características, os protocolos de redes convencionais de redes fixas não são adequados para estes ambientes, por não tratarem muitos aspectos da mobilidade. Neste ambiente destacam-se 2 tecnologias: O WAP e o IP móvel.

O protocolo WAP (*Wireless Access Protocol*) faz parte de uma tecnologia desenvolvida especialmente para a Internet móvel. Além do WAP, compõem esta tecnologia o protocolo de transações WTP (*Wireless Transaction Protocol*) e a linguagem básica WML (*Wireless Markup Language*). O protocolo foi criado por um consórcio de empresas chamado WAP Fórum em 1997. O WAP é o padrão mundial para apresentação, interação e entrega de informações especialmente no ambiente de telefonia celular (TAURION, 2002).

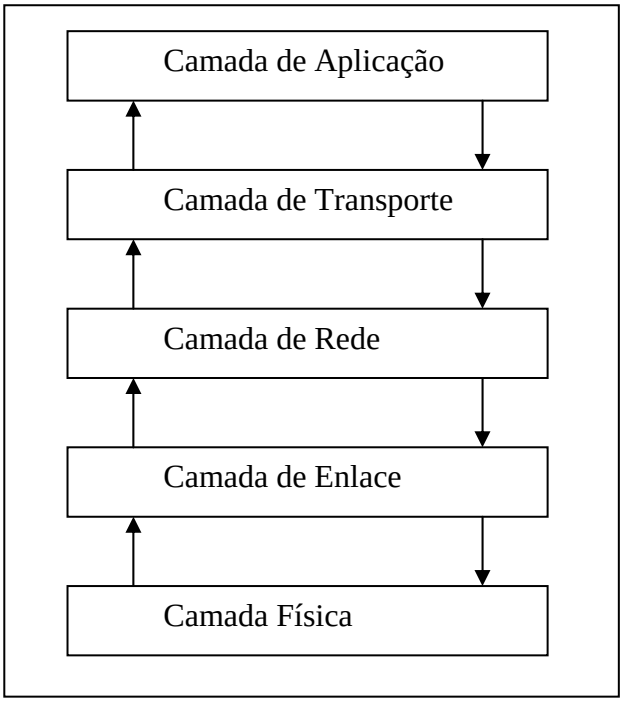
Os principais componentes são o *browser*, o *gateway* e o servidor WAP. O browser reside no aparelho móvel e tem a responsabilidade de apresentar as informações na tela, além de oferecer os serviços de navegação. O *gateway* é responsável pela conversão dos dados no formal WML para *bytecodes* WML específicos para cada aparelho celular. O Servidor WAP tem mesma finalidade de um servidor HTML (*Hyper Text Markup Language*). Pode-se ainda desenvolver um site utilizando XML (*Extensible Markup Language*) e através das folhas de estilos XSL (*XML Style Sheets*) fazer a conversão para uma página HTML ou para WML conforme a solicitação de cada *browser*.

O *Gartner* Group afirmou que a tecnologia WAP será substituída pela tecnologia Java, assim que as taxas de transferências de dados e a capacidade computacional dos celulares permitir. O WAP será utilizado apenas para os projetos que tenha retornos muitos rápidos. A Sun Microsystems recentemente anunciou o padrão Java para celulares, chamado MIDP (*Mobile Information Device Profile*). Este padrão baseia-se no J2ME, já em utilização por alguns fabricantes de telefones celulares como a Motorola, Nokia, LG e Nextel. Com a convergência dos telefones celulares com os *handhelds* o Java terá um ambiente ideal de trabalho, dando maior flexibilidade aos serviços como aconteceu com o uso do Java na Internet (TAURION, 2002).

O ambiente móvel é uma conseqüência direta da evolução da Internet (TAURION, 2002). No ambiente da Internet o protocolo predominante é o TCP/IP. Este protocolo foi definido como um modelo de cinco camadas, conforme figura 5. Devido a grande estrutura existente na Internet, este conjunto de protocolos ainda deverá ser o padrão para as comunicações entre aplicações. No entanto, existem outros protocolos que tratam o ambiente móvel, caso do WAP.

O protocolo TCP/IP não trata a mobilidade dos participantes da rede. Todo participante de rede IP tem um endereço associado a uma localização fixa na rede. Baseado neste endereço o protocolo faz o roteamento dos pacotes até o destino.

Figura 5: Arquitetura das camadas de Protocolos, baseada no modelo TCP/IP



O IP versão 4 assume que o endereço de IP identifica o ponto de conexão do nó1 à Internet. Um nó precisa ser localizado na rede através do seu número de IP para que ele receba datagramas destinados a ele, caso contrário os datagramas destinados a ele não serão enviados. Para que um nó mude seu ponto de conexão sem perder sua capacidade de se comunicar, seriam necessários novos procedimentos:

---

1 O termo nó é empregado para referir-se ao um elemento participante da rede. Este elemento pode ser um computador, um roteador ou outro equipamento de suporte.

a) O nó tem de mudar seu endereço IP toda vez que mudar seu ponto de conexão à rede;

b) um novo caminho de conexão deve ser procurado por toda a Internet, para encontrar o nó com endereço de IP especificado.

Porém, estas duas opções tornam-se impraticáveis, uma vez que para a primeira opção torna-se impossível de manter o transporte do nó e a conexão desejada quando o nó muda de posição; e a segunda exibe problemas de escalabilidade quando se trata de uma rede como a Internet.

Por estas razões, vê-se que o uso do IPv4 para o uso de comunicação móvel impõe sérios problemas de implementação. Um pacote destinado a um computador móvel torna o problema mais complexo. Como o computador pode mover-se entre as células e/ou redes distintas, seu endereço pode variar. Nesse cenário a mobilidade não mais será transparente para as aplicações (MATEUS & LOUREIRO, 1998).

Para resolver o problema o IETF (*Internet Engineering Task Force*) criou um grupo de trabalhos que propôs o IP móvel. Para a implementação deste novo protocolo e para garantir a comunicação com a Internet, algumas definições tiveram de ser estabelecidas (TANENBAUM, 1997):

- Os nós móveis devem ser capazes de se comunicar com qualquer outro nó na rede após mudança no ponto de conexão sem alterar seu endereço de IP;
- Um nó móvel deve ser capaz de se comunicar com um nó que não implementa nenhuma das funções de mobilidade. Ou seja, nenhuma mudança será feita nos *hosts* e roteadores fixos para existir a comunicação entre eles;
- O link que conecta o nó móvel à Internet será geralmente uma conexão sem fio. Esta conexão, portanto, deve ter uma banda de transmissão menor e maior taxa de erros que os links a fio;
- Nós móveis são, geralmente, movidos a bateria, portanto minimizar o consumo de energia deve ser um fator importante. Consequentemente, o número e tamanho das mensagens de administração do sistema entre o nó móvel e seu ponto de conexão à Internet devem ser diminuídas.

Para o ambiente móvel tem-se o IP móvel. O IP móvel cria três entidades funcionais, conforme figura 6:

- Nó Móvel, entidade que pode mudar seu endereço de acordo com sua

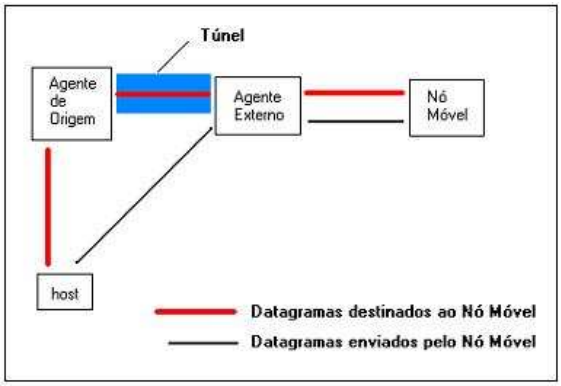
localização;

- Agente Origem, roteador que redireciona os datagramas para o nó móvel;
- Agente externo, roteador da rede visitada pelo nó móvel. Este agente fornece os serviços básicos de endereçamento e roteamento ao nó móvel.

Quando a unidade móvel está conectada em sua própria rede, isto é, através do agente origem, ele é considerado com um computador fixo. Quando ocorre uma mudança de localização, o endereço do nó móvel deve mudar de acordo com o agente externo. Após o registro do nó móvel no agente externo, faz-se o registro do novo endereço no agente de origem. Quando o pacote tem como origem o nó móvel e o destino um outro nó qualquer, o protocolo não sofre alteração alguma, pois o agente externo se encarrega do roteamento. No entanto, os pacotes em sentido contrário serão encaminhados para o agente de origem. Este por sua vez encaminhará os datagramas para o agente externo que entregará o ao nó móvel. A figura 6 ilustra o procedimento.

O IP móvel não resolveu o principal problema já encontrado com o protocolo IP tradicional: a falta de endereços. Assim surgiu o IPv6. Nesta versão o protocolo ganha mais endereços. Na versão IPv4 o campo de endereçamento tem comprimento de 32 bits. Já na versão IPv6 o endereçamento passa e ter tamanho de 128 bits. Além de aumentar o tamanho do campo de endereçamento, o IPv6 incorpora o IP móvel além de outras melhorias como a segurança com o uso de criptografia.

Figura 6: Modelo de comunicação do IP móvel



Fonte: desconhecida

2.1.4 Modelos de Aplicações

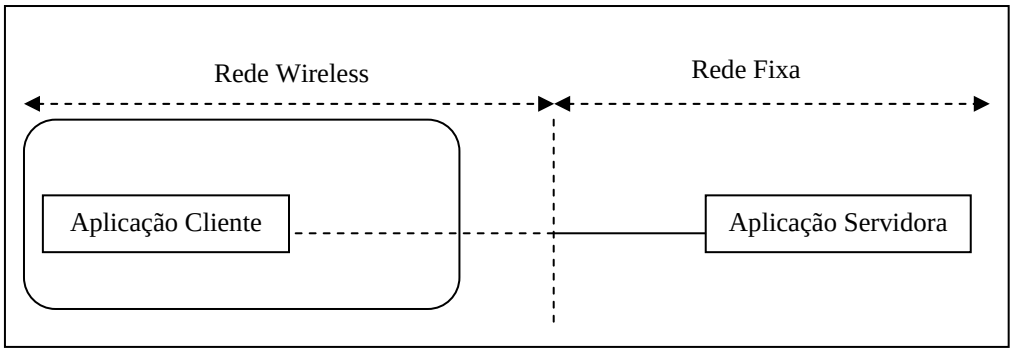
Os ambientes móveis apresentam alguns tipos de limitações. Os modelos utilizados na computação móvel devem levar em consideração estas restrições, adequando-se. Para (PITOURA & SAMARAS, 1998), os modelos de computação móvel podem ser classificados como: modelo cliente/servidor; modelo *peer-to-peer* e modelo de agentes móveis.

2.1.4.1 Modelo Cliente/Servidor

No modelo Cliente/Servidor percebe-se uma forma muito simples de computação. Uma aplicação é executada em um sistema de computação autônomo chamado de cliente. O cliente requisita um serviço de um outro sistema autônomo chamado servidor. No ambiente *wireless* (sem fio) uma estação móvel pode ser, dependendo do sistema em execução, servidor, cliente ou os dois. O mais comum é um cliente móvel requisitar algum serviço de um servidor na rede fixa. Os dados podem estar distribuídos entre diversos servidores na rede fixa e para atender a solicitação do cliente móvel deve haver uma comunicação entre todos os servidores. Em muitos casos o servidor é replicado em diferentes sites da rede fixa para aumentar a disponibilidade no caso de uma falha no site.

Neste modelo, conforme a figura 7, não existe diferença clara entre os serviços que serão executados pelo cliente ou pelo servidor. O principal parâmetro da arquitetura cliente/servidor é o mecanismo de comunicação para a troca de informações entre o cliente e o servidor, sendo a troca direta de mensagens entre o cliente o servidor um dos métodos mais utilizados. Esta técnica não é adequada para redes não estáveis, como é o caso das redes *wireless*. Neste ambiente é mais eficiente um mecanismo de filas de mensagens. Outra possibilidade é a adoção do mecanismo de RPC (*Remote Procedure Call* – Chamada Remota a Procedimentos). Neste conceito o cliente chama um procedimento que é executado no servidor (PITOURA & SAMARAS, 1998).

Figura 7: Modelo Cliente Servidor



Fonte: (PITOURA & SAMARAS, 1998, pp. 18)

Nas redes *wireless* as desconexões ocorrem com freqüência. Uma chamada RPC síncrona não é adequada, visto que existe a necessidade das estações estarem constantemente conectadas ao servidor. Já as chamadas RPC assíncronas são mais eficientes, pois pode-se colocar todas as chamadas em uma fila. Uma estensão ao modelo cliente/servidor faz-se necessário para suportar as operações em modo desconectado e as operações com conexão fraca. Filtros, compressão de dados e mecanismos de segurança também são importantes, devido às baixas taxas de transferências e ambiente de trabalho.

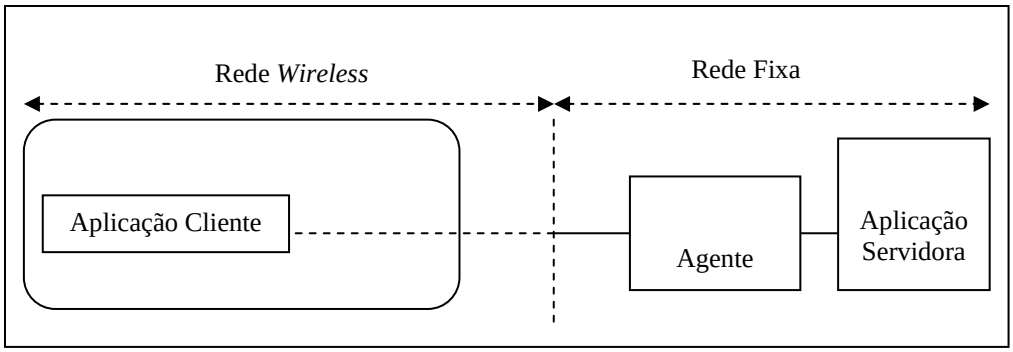
2.1.4.2 Modelo Cliente/Agente/Servidor

O modelo cliente/agente/servidor, também conhecido como modelo cliente/servidor em três camadas possui três partes distintas: o cliente, o agente e o servidor. O cliente representa o computador móvel, o agente que age em nome do cliente e o servidor que atende as requisições do cliente. Neste modelo usam-se mecanismos de mensagens e filas para a comunicação. O cliente envia uma mensagem para o agente, que por sua vez envia para o servidor. Em alguns casos os agentes ou *proxies* são simplesmente substitutos de um cliente na rede fixa. Esta abordagem alivia o impacto da largura de banda limitada e baixa confiabilidade da comunicação sem fio nas transações entre cliente e servidor (PITOURA & SAMARAS, 1998) (MATEUS &

LOUREIRO, 1998).

Nesta arquitetura o agente pode ter acesso a canais de comunicação confiáveis e de grande largura de banda. Protocolos diferentes podem ser usados em cada parte da conversação, separa-se, portanto, a comunicação entre o servidor e o cliente, conforme a figura 8. A comunicação cliente/agente e agente/servidor podem acontecer em momentos diferentes. O agente é responsável pela divisão da interação.

Figura 8: Modelo Cliente/Agente/Servidor



Fonte: (PITOURA & SAMARAS, 1998, pp. 18)

Uma decisão importante, neste modelo, é a localização do agente. Pode-se coloca-lo mais perto do cliente, ou seja, na estação base da célula onde encontra-se o cliente. Nesta abordagem consegue-se obter, com facilidade, informações sobre as condições do enlace sem fio. Assim o agente pode decidir quanto a comunicação com o cliente em um determinado instante. Nesta política o agente deve acompanhar o cliente na mudança de uma célula para outra, isto, conforme o cliente se movimenta o agente o acompanha. Para o caso de agente de serviços específicos o mais indicado é colocar o agente o mais próximo do servidor ou da maioria dos clientes (MATEUS & LOUREIRO, 1998).

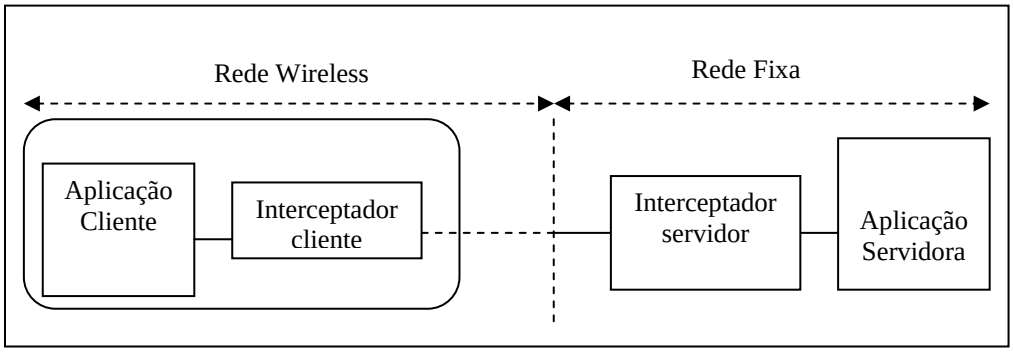
Este modelo é uma evolução do modelo cliente/servidor, porém apresenta alguns problemas. Do ponto de vista do servidor, não existem problemas de comunicação com o cliente, pois o agente se encarrega disto. Já a aplicação cliente deve ser modificada para a interação com o agente. Isto pode ser um problema quando a aplicação já existe. Outro problema é quanto à otimização da transmissão dos dados. O agente consegue otimizar a comunicação com o cliente, mas não no sentido contrário. Outro problema deve-se ao fato da desconexão. Uma vez desconectado, o cliente não pode continuar sua

operação ininterruptamente.

2.1.4.3 Modelo Cliente/Agente/Agente/Servidor

Uma solução possível para os problemas apresentados para o modelo cliente/agente/servidor é dividir o agente em duas partes: uma parte que fica no cliente móvel e outra na rede fixa. Estes dois novos elementos são chamados de interceptadores ao invés de agentes, figura 9.

Figura 9: Modelo Cliente Servidor



Fonte: (PITOURA & SAMARAS, 1998, pp. 18)

O interceptador, do lado cliente, intercepta as chamadas do cliente enquanto o interceptador, do lado servidor, intercepta as chamadas do servidor. Os interceptadores agem entre si para otimizar a comunicação. Do ponto de vista do cliente o interceptador exerce o papel de um *proxy*. De forma análoga, do ponto de vista do servidor o interceptador exerce o papel de um cliente local *proxy*.

O modelo oferece flexibilidade. Um *cache* de dados local pode ser mantido pelo cliente para satisfazer as operações em modo desconectado. Outra característica importante é que os interceptadores são transparentes para o cliente e o servidor. Assim, pode-se projetar os interceptadores com características próprias do ambiente de trabalho. Para (MATEUS & LOUREIRO, 1998), “o par de interceptadores podem ser vistos como uma camada *middleware* que otimiza e facilita a comunicação num

ambiente móvel".

O modelo de interceptadores apresenta claramente a distinção entre as responsabilidades do cliente e do servidor. Esta técnica é apresentada por *Web Express*, um sistema de otimização de pesquisas para a Internet. Neste sistema o agente do lado cliente é chamado de CSI (*Client Side Intercept*) e SSI (*Server Side Intercept*) (PITOURA & SAMARAS, 1998).

2.1.4.4 Modelo Simétrico

Em uma arquitetura simétrica (P2P, *peer-to-peer*), não existe distinção entre as responsabilidades de servidor e de cliente. Cada estação tem as funcionalidades de cliente e de servidor. No ambiente de computação móvel, os computadores tornam-se parceiros idênticos na computação distribuída. As desconexões têm efeito negativo neste ambiente.

Aplicações onde se faz necessário executar algum tipo de trabalho cooperativo entre as entidades parceiras são fortes candidatas a usarem o modelo para a par. As entidades cooperantes do ambiente móvel (clientes) não trocam informações diretamente no modelo cliente/servidor. Todas as trocas de informações são efetuadas pelo sistema servidor. Já nas aplicações simétricas, todas as informações são trocadas diretamente pelas entidades participantes, sendo indicadas onde existe uma forte conexão entre as unidades móveis, caso de redes *ad-hoc* (MATEUS & LOUREIRO, 1998).

2.1.4.5 Modelo de Agentes Móveis

Os agentes móveis são processos que migram de um computador para outro com o objetivo de executar uma tarefa específica (CHESS et al., 1995). Cada agente móvel possui instruções, dados e um estado de execução. Neste modelo o agente móvel estende o mecanismo de comunicação RPC, a mensagem enviada pelo cliente é uma chamada RPC. O agente móvel executa de forma autônoma e independente da aplicação que o invocou. O Agente, ao chegar ao seu destino, é autenticado, preparado para execução num ambiente de execução da entidade destino, e, finalmente, executado. O agente pode transferir-se para outro ambiente para cumprir o objetivo de executar sua tarefa. Pode ainda criar e disparar a execução de novos agentes ou interagir com outros. Ao finalizar sua tarefa o agente envia o resultado para a aplicação que o chamou.

Os agentes móveis são projetados para tomar decisões e resolver problemas. Suas principais características são: habilidade de interagir e cooperar com outros agentes, autonomia de execução, ser executados em diversas plataformas de hardware e software, alto grau de interoperabilidade, responder a eventos externos e ser capaz de mover-se de uma estação à outra. Um agente é composto basicamente por um *script*, responsável por expressar a tarefa do agente, e uma base de conhecimento.

O modelo computacional de agentes, suporta operações em modo desconectado. Durante uma breve conexão, o cliente móvel envia ao agente seu pedido e depois desconecta. O agente executa sua tarefa independentemente da estação móvel. Após o agente completar sua tarefa, ele irá esperar a unidade móvel reconectar para então enviar os resultados. Normalmente os agentes móveis são disparados na rede fixa. Os agentes móveis podem migrar tanto para a execução de sua tarefa como para a localização da unidade móvel. Este paradigma é totalmente oposto ao modelo cliente/servidor e ao modelo cliente/agente/servidor. No modelo cliente/agente/servidor, por exemplo, o agente deve residir em um local definido na rede fixa.

2.2 APLICAÇÕES

A computação móvel traz um novo paradigma de trabalho. Novas aplicações são desenvolvidas para atender as diversas necessidades das pessoas. A seguir apresentam-se algumas aplicações da computação móvel:

- Serviços de emergências: Os serviços de emergências podem fazer uso de aplicativos para prestar os primeiros atendimentos as pessoas acidentadas, coletando informações que posteriormente serão utilizadas em um hospital. Pode-se ainda utilizar a computação móvel em casos de desastres, como terremotos ou enchentes, coletando informações.
- Agentes de seguros: Os agentes de seguros podem elaborar uma apólice de seguro no mesmo instante da visita ao seu cliente. Pode-se ainda, fazer o termo de vistoria, imprimir o contrato. A vigência da apólice seria imediata, visto que todos os dados estariam nas centrais das companhias no mesmo instante.
- Executivos: Executivos podem usar seus equipamentos móveis para dar continuidade de seus trabalhos durante uma viagem. Receber informações sobre mercados, executar operações, comprar ou vender ações de empresas entre outros;
- Em fábricas: Nas fábricas pode-se utilizar a computação móvel para coletar informações sobre a produtividade das linhas de produções. Estas informações seriam repassadas a um computador central que poderia interferir nos ajustes de uma máquina;
- Automação de vendas: Este talvez seria o apelo mais clássico. Profissionais de vendas podem fazer uso de PDAs para automatizar as suas vendas. Os pedidos seria digitados pelos vendedores de forma mais eficiente. Podem-se consultar os estoques, verificar os prazo de entrega, consultar a ficha financeira do cliente entre outros. Os aplicativos móveis podem estar diretamente ligados ao software de gestão da empresa, tendo assim uma extensão da infra-estrutura de informática;
- Ambientes de guerra: Em campos de batalhas podem-se coletar informações

para o comando geral. Estas informações serviriam para a construção de novas estratégias.

- Transportadoras: Os funcionários de uma transportadora poderiam utilizar-se dos aplicativos em um PDAs para a emissão do conhecimento de carga no ato de uma coleta. Podem-se ainda programar as coletas das equipes de campo em tempo real. Toda chamada a central seria remetida à equipe mais próxima.

- Correio eletrônico: As unidades móveis podem enviar e receber mensagens eletrônicas. Pode-se ainda, conectar-se diretamente ao sistema de *workflow* da empresa para dar continuidade a um processo que esteja parado.
- Hospitais: Neste ambiente os equipamentos móveis podem auxiliar no tratamento dos pacientes. Os médicos e enfermeiras podem ter acesso a qualquer momento a prontuários, radiografias, exames, receitas médicas, rotinas, etc. Pode-se também fazer a solicitação de medicamentos, exames e outros com os custos diretamente debitados na conta do paciente ou e seu plano de saúde.
- Restaurantes: Nos restaurantes, lanchonetes ou bares as contas podem ser controladas pelos equipamentos móveis. Tanto pedidos, como a solicitação de contas, como o pagamento podem ser feitos diretamente pelos PDAs, podendo ainda utilizar-se de dinheiro digital.
- Companhias aéreas: Nos aeroportos os comprovantes de embarques pode ser feitos diretamente nas salas de esperas, deixando, assim, os guiches para venda de passagens.
- Construção civil: Na construção civil podem-se fazer as medições da execução da obra diretamente no canteiro de obras. O andamento da obra poderia ser controlado online. Plantas e projetos podem ser acessados diretamente em meio digital. Os mestres de obras podem informar o andamento da obras enquanto o engenheiro confere o projeto ou o departamento financeiro acompanha os custos;
- Bancos: Os bancos poderão desenvolver aplicações para seus clientes. Os clientes podem ter em suas mãos todos os detalhes de sua conta e suas

movimentações. O cliente terá uma extensão de um terminal do banco em seu equipamento móvel. Poderá ter diretamente da instituição financeiras extratos, saldos, serviços de pagamentos de contas, cotações de moedas, compra e venda de ações, fundos de investimentos entre outros.

- Pessoal: As pessoas poderão fazer compras, comprar bilhetes de cinema, teatro, passagens aéreas, compra de doces, serviços diretamente lançados em seu programa financeiro pessoal e também lançados diretamente em sua conta corrente no banco ou mesmo em sua conta telefônica.
- Marketing: Os aparelhos móveis podem ser localizados com facilidade e uma empresa poderá fazer seu marketing diretamente a elas. As campanhas podem ser feitas nos dois sentidos empresa-usuário ou usuário-empresa. Uma empresa lança uma promoção e envia para todos os equipamentos móveis de determinada região. Ou o usuário entra em um shopping e pesquisa as promoções. O usuário Pode ainda pesquisar sobre uma determinada mercadoria ou serviço em determinada região.
- Profissional móvel: Médicos, técnicos, consultores, etc. podem obter acessos as suas informações em casa ou escritório, informações sobre empresas, clientes, produtos, elaborar relatórios em tempo real, obter informações corporativas, etc;
- Outros: A computação móvel tem um potencial fantástico de aplicações. Pode-se disponibilizar vários serviços como: informações turísticas, eventos; festivais; informações sobre jogos e esportes; horóscopo; piadas; compras e resultados de loterias; programação de cinemas e teatros; informações sobre ônibus e taxis; aeroportos, etc.

Estas são apenas algumas potencialidades das aplicações em dispositivos móveis. Encontrar aplicações não será algo difícil no futuro. A convergência entre os telefones celulares, Internet e dispositivos móveis implicará em um negócio com mais de 1 bilhão de usuários. Existirão mais usuários utilizando dispositivos móveis que computadores fixos (TAURION, 2002).

2.3 SEGURANÇA

A segurança na computação móvel étema de grandes pesquisas. Os sistemas móveis espalham seus sinais nas diversas direções podendo ser captado por qualquer um, ao contrário dos meios guiados onde somente que tem acesso ao meio pode captar o sinal. Neste sentido a segurança em ambiente móvel deve ser tratada desde a camada física e de enlace até a camada de aplicações.

Na camada de física e de enlace têm-se vários protocolos que fazem uso de segurança para a comunicação. Nesta camada normalmente faz-se o uso de criptografia simétrica, isto é, os aparelhos móveis e as estações de suporte a mobilidade devem conhecer as chaves para então transmitirem seus dados criptografados.

Já na camada de aplicação podem-se utilizar técnicas de criptografia assimétrica com o uso de certificados digitais. Neste caso têm-se algumas dificuldades, pois sendo as unidades móveis, deve existir uma rede de autoridades certificadoras.

Utilizar as técnicas tradicionais de criptografia em ambientes móveis pode acarretar em alguns problemas. As unidades móveis possuem pequena capacidade processamento e pouco tempo de bateria. Desenvolver algoritmos eficientes para este ambiente faz-se necessário, servindo de motivação para muitos pesquisadores.

2.4 VANTAGENS E DESVANTAGENS DA MOBILIDADE

No ambiente móvel têm-se diversas vantagens como:

- Conforto: Torna-se possível a utilização do aparelho móvel em qualquer lugar;
- Comodidade: Podem-se realizar tarefas, a qualquer momento;
- Flexibilidade: Utilizar-se de aplicações que exijam o movimento;
- Disponibilidade: As aplicações e serviços tornam-se independente de localização;
- Portabilidade: Os equipamentos podem ser transportados dentro do bolso;

A mobilidade na computação requer um meio de comunicação que não utilize fios de cobre ou fibras óticas para a comunicação entre as entidades participantes. Esta necessidade, em conjunto com a mobilidade, implica em uma série de problemas para a computação (MATEUS & LOUREIRO, 1998). A seguir algumas desvantagens da computação móvel:

- Hardware: As unidades móveis têm limitações físicas. Problemas como tamanho da tela, tamanho do teclado, em alguns casos a falta do mouse, pouca capacidade computacional, pouca memória e capacidade reduzida dos discos são comuns;
- Ergonomia: Devido às limitações do hardware, os dispositivos móveis ainda não possuem a mesma facilidade de uso dos computadores comuns;
- Perda ou Roubo: Os computadores portáteis são menores, portanto estão mais vulneráveis a perdas e roubos;
- Desconexão: As desconexões são freqüentes devido à interferência do mio nas transmissões;
- Baterias: As baterias não possuem cargas suficientes para longos tempos em operação;

3 BANCO DE DADOS DISTRIBUÍDOS

3.1 INTRODUÇÃO

Os sistemas de bancos de dados surgiram na década de 60, como uma evolução dos sistemas de informação baseados em arquivos e podem ser vistos como uma evolução dos sistemas de arquivos tradicionais. O SGBD consiste em uma camada lógica entre a aplicação e os dados, fornecendo um mecanismo de abstração de detalhes, como relacionamentos, estruturas de armazenamento, regras de validações dos dados, entre outros (MELO et al, 1997).

Os SGBD podem ser centralizados e distribuídos. Nos sistemas centralizados apenas um SGBD é responsável por todas as informações de um sistema. Já em um ambiente distribuído, vários SGBD cooperam entre si. As informações são distribuídas a fim de aumentar a disponibilidade do sistema. No ambiente móvel predomina os SGBDD – Sistemas Gerenciadores de Banco de Dados Distribuídos.

Neste capítulo destina-se ao estudo dos SGBDD, suas arquiteturas, modelos e requisitos. Faz-se uma breve revisão das arquiteturas cliente/servidor, distribuídas não--hierárquicas e de vários banco de dados.

3.2 SISTEMA DE BANCO DE DADOS DISTRIBUÍDO

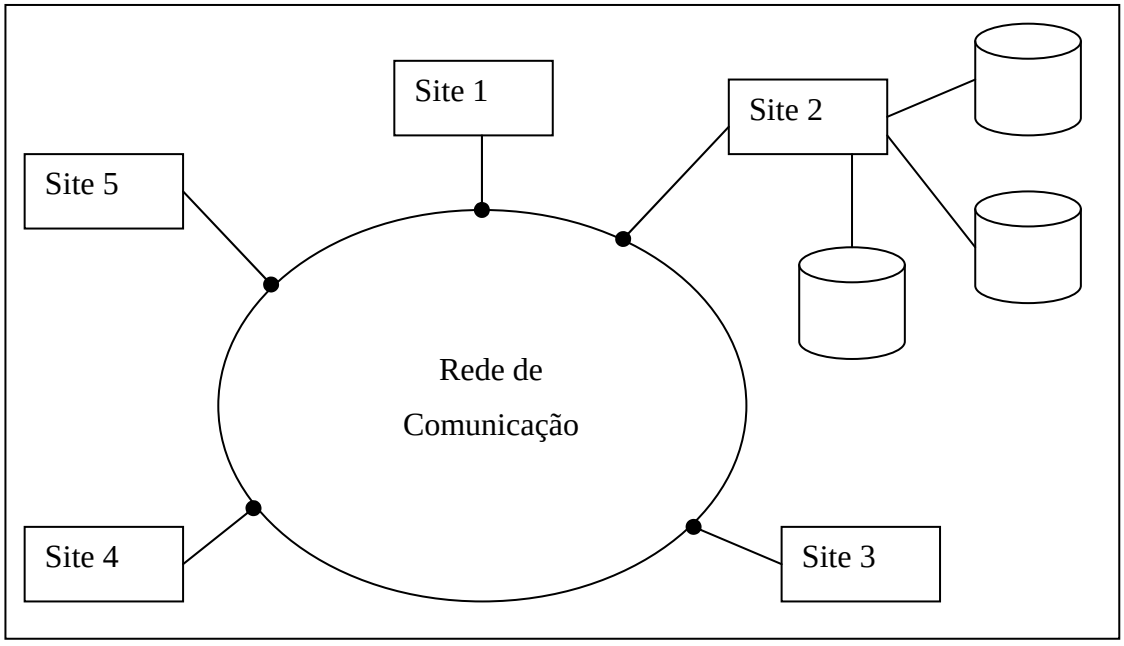
A cooperação entre dois ou mais processos está implícita nos ambientes de sistemas distribuídos. Uma aplicação distribuída é caracterizada pelo alto grau de interação entre os fragmentos da aplicação, que são distribuídos entre diversas plataformas (MELO et al, 1997). Um sistema de computação distribuída consiste em

diversos elementos autônomos de processamento em operação em uma rede de computadores que cooperam entre si para a execução de uma tarefa distribuída (ÖZSU & VALDURIEZ, 2001).

Os sistemas de bancos de dados distribuídos apresentam uma série de vantagens em relação aos sistemas de bancos de dados centralizados, como o aumento da disponibilidade e confiabilidade dos dados, aumento de desempenho, entre outros. Segundo (ÖZSU & VALDURIEZ, 2001, pp. 5), “Podemos definir um banco de dados distribuído como uma coleção de vários bancos de dados logicamente inter-relacionados, distribuídos por uma rede de computadores”, conforme ilustram as figuras 10 e 11.

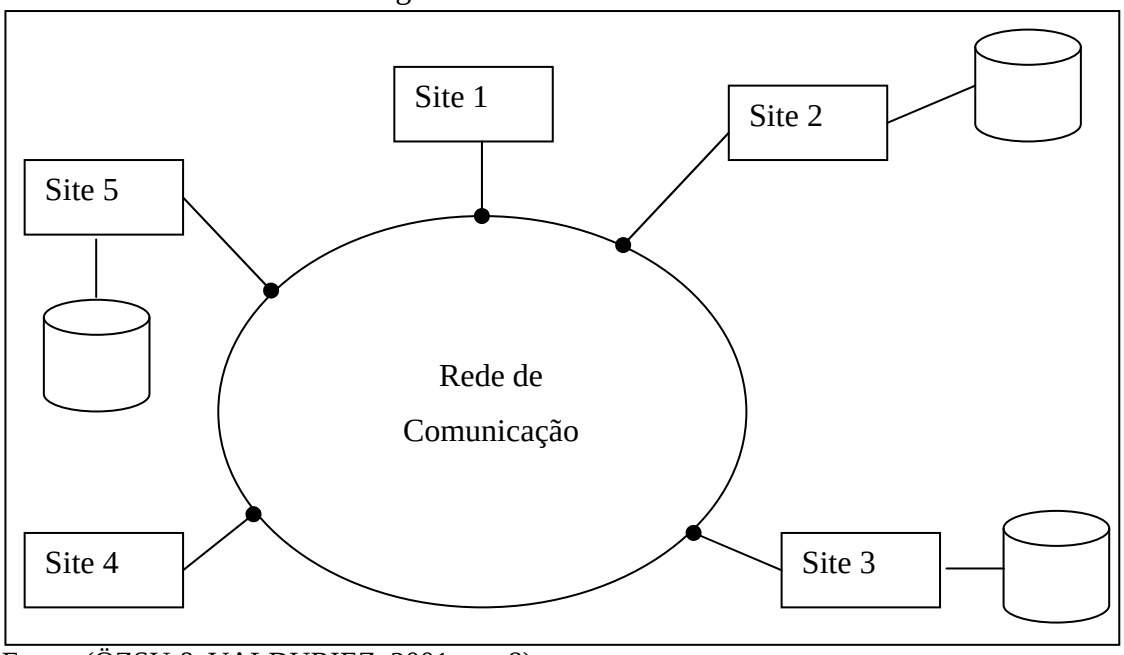
Define-se como Sistema Gerenciador de Banco de Dados Distribuídos (SGBDD), com um sistema de software que permite o gerenciamento do banco de dados distribuído, tornando-o transparente para o usuário. Um SGBDD não é simplesmente uma coleção de arquivos que podem estar armazenados individualmente em cada nó (*host*) da rede, mas sim uma estrutura entre os arquivos, sendo o acesso realizado por intermédio de uma interface comum (ÖZSU & VALDURIEZ, 2001).

Figura 10: Banco de dados central em uma rede



Fonte: (ÖZSU & VALDURIEZ, 2001, pp. 8)

Figura 11: Ambiente de SBDD



Fonte: (ÖZSU & VALDURIEZ, 2001, pp. 8)

Um SGBDD oculta os detalhes de implementação dos usuários. Ele é totalmente transparente e fornece alto nível de suporte para o desenvolvimento de aplicativos complexos. O acesso totalmente transparente significa que os usuários podem efetuar consultas complexas sem qualquer preocupação quanto à fragmentação e localização dos dados, deixando a cargo do sistema resolver estes problemas. Para obter-se esta transparência, faz-se necessários alguns requisitos de transparência (ÖZSU & VALDURIEZ, 2001):

- Independência dos Dados: A independência dos dados pode ser dividida em duas partes: a independência lógica e a física. A independência lógica refere-se à imunidade de aplicativos do usuário a mudanças na estrutura lógica do banco de dados. A independência física lida com a ocultação dos detalhes da estrutura de armazenamento em relação aos aplicativos do usuário;
- Transparência de rede: Em um ambiente de banco de dados centralizado o único recurso a ser isolado do usuário são os dados. Contudo, em um ambiente de banco de dados distribuído existe a rede de dados que necessita ser administrada, de modo a ocultar os detalhes operacionais. A rede de dados deve estar transparente para o usuário, não existindo diferenças em executar uma aplicação em um ambiente de banco de dados centralizado ou em uma

ambiente de banco de dados distribuído;

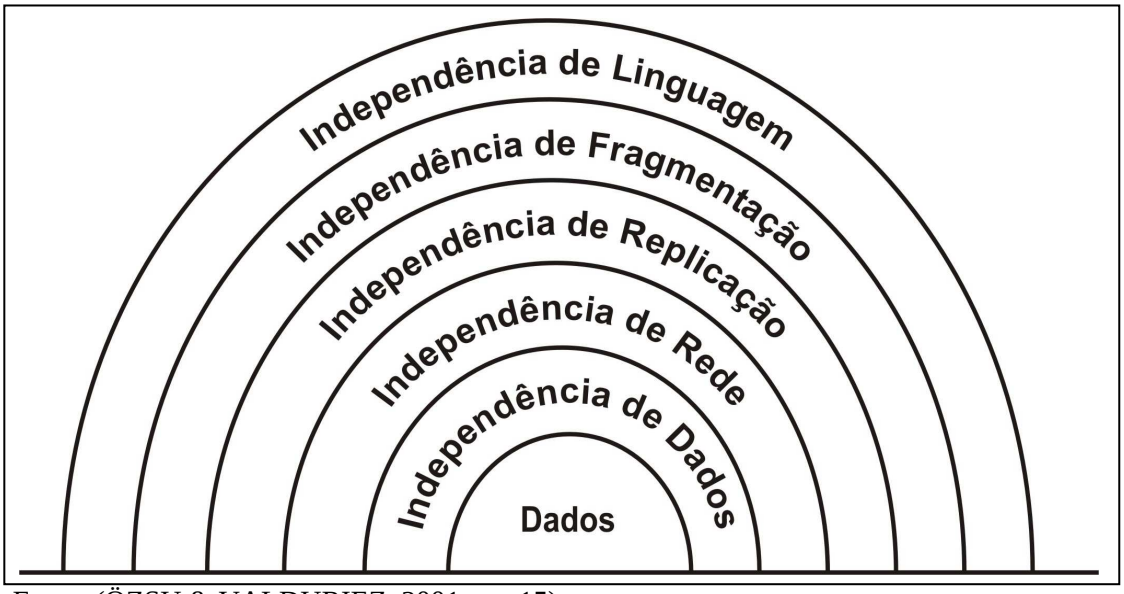
- Transparência de replicação: Para auxiliar o desempenho da rede de dados e dos bancos de dados, aumentar a confiabilidade e disponibilidade, faz-se necessário a distribuição dos dados de forma replicada pelas máquinas de uma rede. Os dados mais utilizados por um usuário podem ser alocados em uma máquina local desse usuário;
- Transparência de fragmentação: É desejável dividir cada relação de um banco de dados em fragmentos menores e tratar cada fragmento como um objeto de banco de dados separado. Além de aumentar o desempenho, confiança e disponibilidade reduz os efeitos negativos da replicação de dados. A réplica não é a relação completa, mas sim um subconjunto desta relação. As fragmentações podem ser tanto horizontais, parte dos registros, e/ou verticais, parte dos atributos;
- Transparência de linguagem: Os usuários podem ter acessos aos dados através de uma linguagem de alto nível como as linguagens de quarta geração, interfaces gráficas do usuário, entre outras.

A hierarquia dessas transparências pode ser visualizada na figura 12. Já (DATE, 2000) apresenta doze objetivos de um sistema de banco de dados distribuídos. São eles:

- Autonomia local;
- Não dependência de um *site* central;
- Operação contínua;
- Independência de localização;
- Independência de fragmentação;
- Independência de replicação;
- Processamento de consultas distribuído;
- Gerenciamento de transações distribuído;
- Independência do hardware;
- Independência do sistema operacional;
- Independência de rede;

- Independência do SGBD.

Figura 12: Camadas de tranaparência



Fonte: (ÖZSU & VALDURIEZ, 2001, pp. 15)

Os doze objetivos apresentados não são todos independentes uns dos outros, muitos menos exaustivos, nem igualmente significativos. Estes objetivos encaixam-se integralmente nos cinco níveis de transparências apresentados acima.

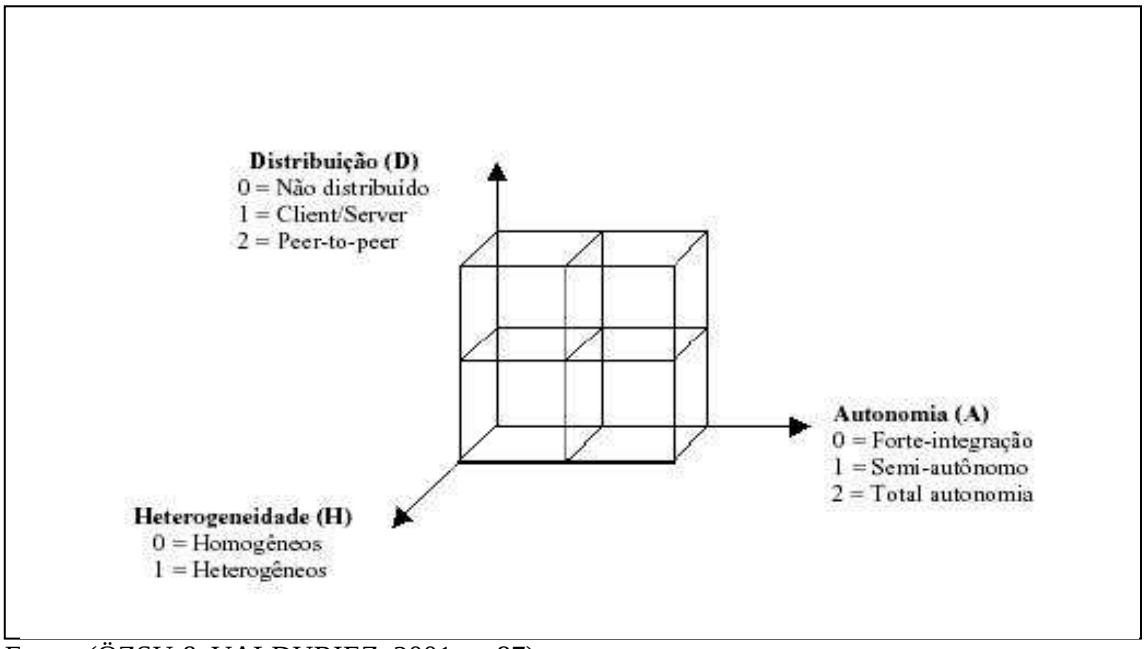
3.3 ARQUITETURAS DO SGBD DISTRIBUÍDO

Uma das principais características que distingue os sistemas de processamento distribuído dos sistemas clássicos está na descentralização do controle. Nos sistemas centralizados os usuários são mais homogêneos e possui maior acessibilidade. A congruência de informação é normalmente assegurada, visto que trata-se apenas de uma máquina. Já os sistemas distribuídos são heterogêneos, possuindo maior confiabilidade perante falhas sucessivas de componentes. Os usuários vêem um conjunto de máquinas como um único sistema de tempo compartilhado. No caso de falha de uma destas máquinas o sistema continua em operação (TANEMBAUM, 1997).

A arquitetura de um sistema define a sua estrutura. Cada componente é identificado segundo sua função e os inter-relacionamentos e interações entre estes sistemas estão bem definidas.

As possíveis alternativas de distribuição de dados são classificadas por (ÖZSU & VALDURIEZ, 2001) e consideram três dimensões: autonomia, distribuição e heterogeneidade, figura 13.

Figura 13: Alternativas de implementação de SGBDs



Fonte: (ÖZSU & VALDURIEZ, 2001, p. 87)

Autonomia pode ser entendida como a capacidade dos SGBDs operarem independentemente. A autonomia refere-se à distribuição do controle e não dos dados. Os SGBDs podem executar transações de forma independente.

Enquanto a autonomia refere-se à distribuição do controle, a dimensão da distribuição trata dos dados. Existem várias maneiras de distribuir os SGBDs. Elas podem ser resumidas em duas: a distribuição cliente/servidor e a distribuição não hierárquica ou distribuição total. Na distribuição cliente/servidor As tarefas de gerenciamento de dados ficam por conta dos servidores e os usuários fornecem um ambiente para os aplicativos. Em sistemas não hierárquicos não existem distinções entre máquinas cliente ou servidoras. Todas as máquinas possuem funções de um SGBD e

podem comunicar-se com outras máquinas para executar consultas e transações (ÖZSU & VALDURIEZ, 2001).

A heterogeneidade pode estar relacionada a diversas formas nos sistemas distribuídos, como: heterogeneidade de hardware, de protocolos, de modelo de dados, de protocolo de gerência de transação e até mesmo de gerenciadores de dados (SGBDs). Pode-se ver a heterogeneidade como uma conseqüência da autonomia dos SGBDs.

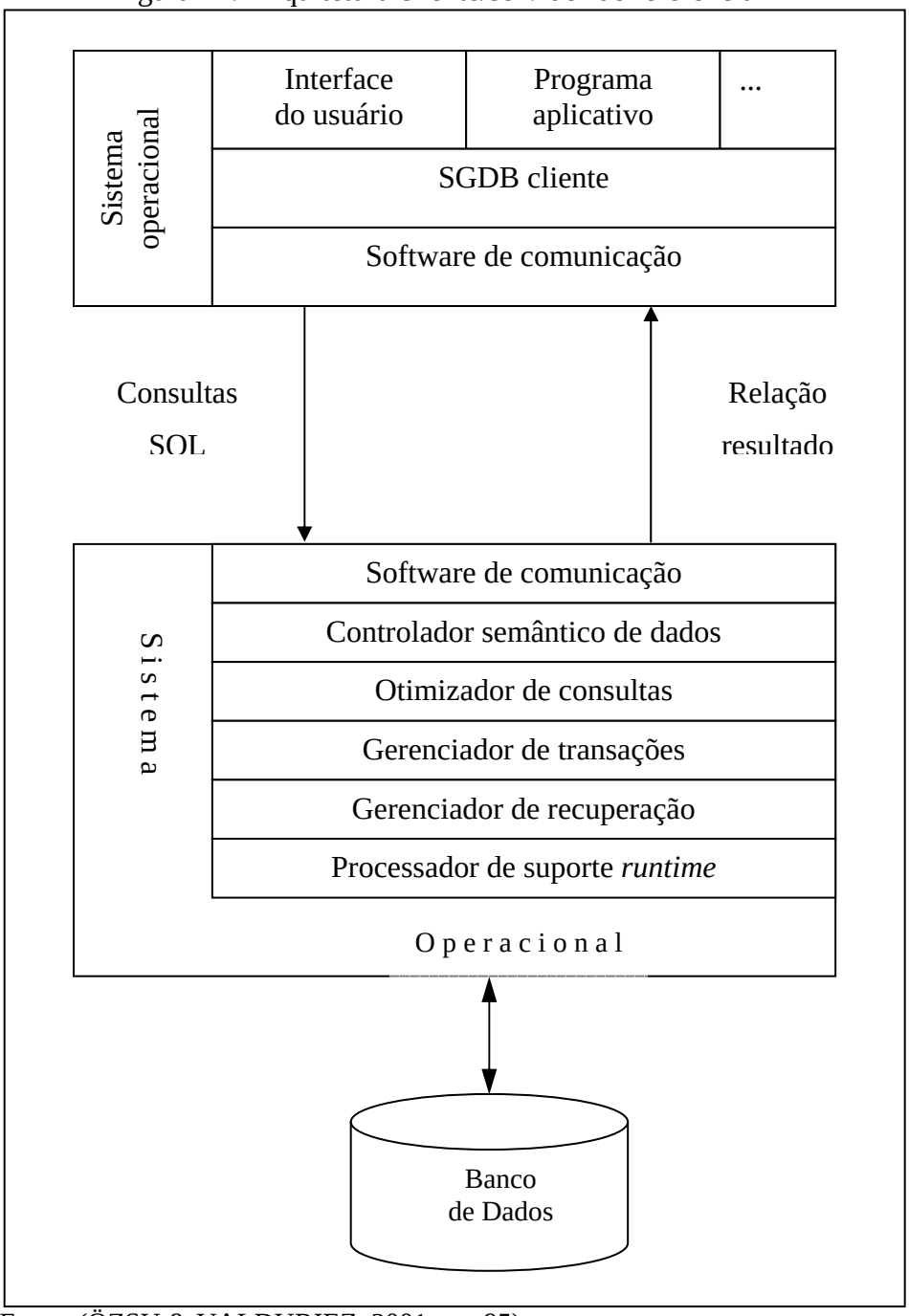
Para modelar-se e implantar sistemas distribuídos faz-se necessário uma tecnologia mais complexa que as usadas em sistemas centralizados. O SBDD (Sistema de Banco de Dados Distribuído) deve conter algumas funcionalidades adicionais das presentes nos sistemas centralizados, como a localização dos dados e a garantia da consistência destes dados.

3.3.1 Sistemas Cliente/Servidor

Os SGBDs cliente/servidor entraram no cenário da computação no início da década de 1990. Este modelo apresenta uma solução simples e elegante: distinguir a funcionalidade que precisa ser fornecida e dividir essas funcionalidades em duas classes, funções do cliente e funções do servidor. Com a divisão em dois níveis torna-se mais simples o gerenciamento (ÖZSU & VALDURIEZ, 2001).

Para (MELO et all, 1997), “As aplicações com arquitetura cliente/servidor representam um caso especial de processamento distribuído, que implica a cooperação entre dois ou mais processos”. Aplicações deste tipo são caracterizadas pelo alto grau de fragmentações da aplicação. A interação é realizada através de pedidos dos clientes e pelas respostas dos servidores, figura 14.

Figura 14: Arquitetura cliente/servidor de referência



Fonte: (ÖZSU & VALDURIEZ, 2001, pp. 95)

Nos SGBDs cliente/servidor, o servidor é responsável pela gerência dos dados. Isto significa que o gerenciamento de transações, gerenciamento de armazenamento, processamento de consultas e otimização fica por conta do servidor. O cliente, além da aplicação do usuário, possui um módulo cliente do SGBD responsável pelo gerenciamento dos dados que são alocados no cache do cliente. Em alguns casos o módulo cliente também é responsável pelo bloqueio de transações. Pode-se ainda colocar a verificação da consistência das consultas no lado cliente, porém exigiria a replicação do catálogo do sistema nas máquinas cliente (ÖZSU & VALDURIEZ, 2001).

Existem várias arquiteturas para o modelo cliente/servidor. O mais simples é a arquitetura de apenas um servidor e vários clientes. Esta arquitetura denomina-se vários clientes - servidor único. Outra arquitetura é a vários clientes - vários servidores. Esta arquitetura é mais sofisticada que a primeira. Nesta abordagem o cliente gerencia a sua própria conexão com o servidor ou cada cliente conhece apenas o seu "servidor local" (*home server*). A primeira abordagem implica em "clientes pesados", carregando as máquinas com responsabilidades adicionais. Já a segunda abordagem implica em clientes leves (ÖZSU & VALDURIEZ, 2001).

De qualquer forma, nestes ambientes os usuários possuem total transparência da estrutura, aparentando assim um sistema de banco de dados logicamente único, não importando a estrutura física.

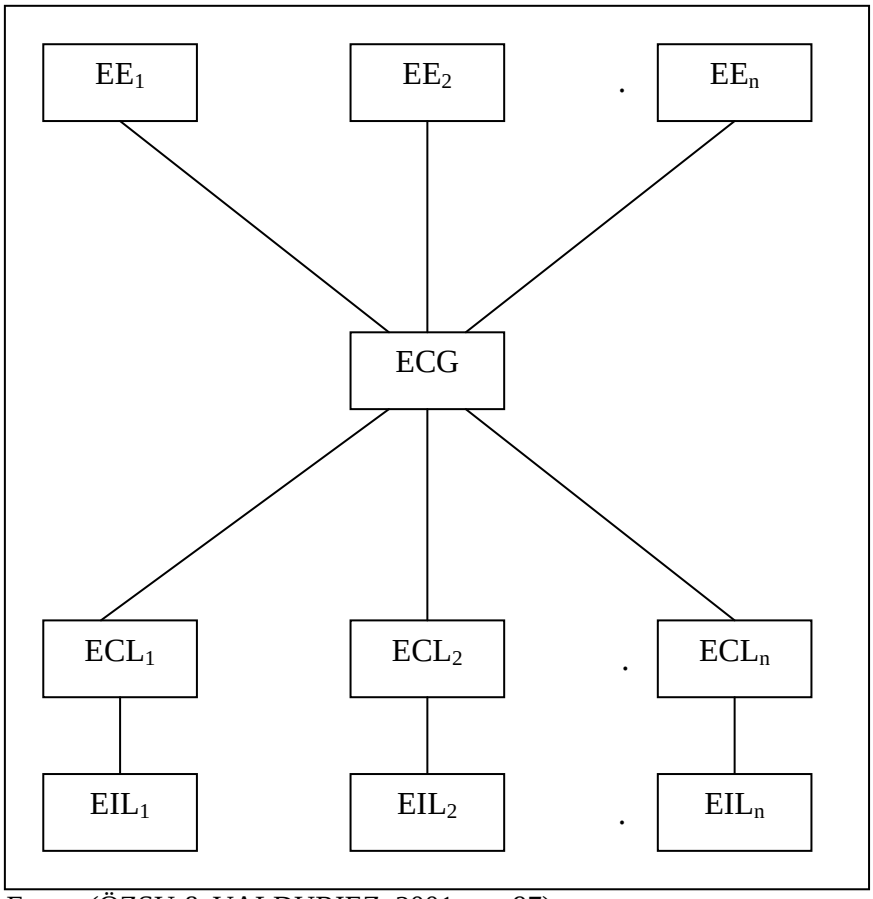
3.3.2 Sistemas distribuídos não-hierárquicos

Nos sistemas distribuídos não-hierárquicos a organização dos dados físicos pode ser diferente. Daí a necessidade da definição de um esquema interno para cada site, chamado de Esquema Interno Local (EIL). A descrição da estrutura lógica dos dados em todos os sites é chamada de Esquema Conceitual Global (ECG). O ECG é a visão empresarial dos dados.

Para lidar com a fragmentação e replicação dos dados, a organização lógica de cada site deve ser descrita. Esta descrição é o Esquema Conceitual Local (ECL). O ECG nada mais é do que a união dos diversos ECL dos sites. O acesso aos dados dá-se

através dos Esquemas Externos (EE). O EE é definido como se estivesse acima do ECG. O modelo de arquitetura, descrito na figura 15, fornece vários níveis de transparência. Este modelo é uma extensão do ANSI/SPARC. As transparência de localização e replicação são admitidas pela definição dos ECLs e ECG, além do mapeamento intermediário. A transparência de rede acontece através pela definição do ECG. O SGBDD converte suas consultas globais em grupos de consultas locais em diferentes sites (ÖZSU & VALDURIEZ, 2001).

Figura 15: Arquitetura de banco de dados distribuído de referência



Fonte: (ÖZSU & VALDURIEZ, 2001, pp. 97)

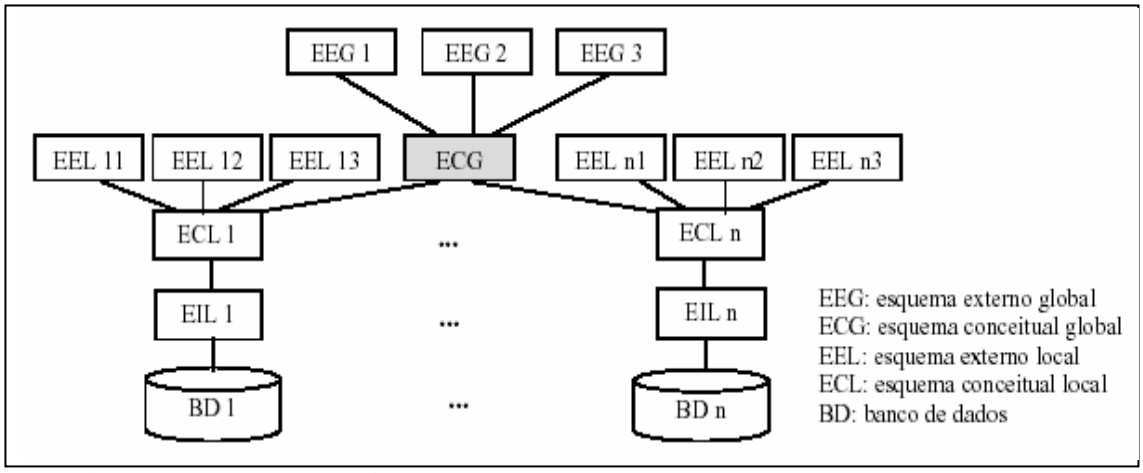
Os esquemas conceituais locais são mapeamentos do esquema global em cada site. Fazem-se as definições de visões externas de maneira global. Projetam-se estes bancos de dados de cima para baixo *(top-down*).

3.3.3 Sistemas de Vários Bancos de Dados (SVBD)

Nos SVBD, definem-se os ECG integrando os esquemas externos de banco de dados locais e autônomos ou partes de seus esquemas conceituais locais, figura 16. Nesta arquitetura os usuários definem suas próprias visões sobre o banco de dados local, para maior autonomia. Projetam-se os Esquemas Conceituais Globais (ECG) apenas para os usuários que necessitem de acesso global. Não é necessários que o EEG e o ECG sejam definidos com o mesmo modelo de dados e a mesma linguagem. Esta definição determina se o sistema é heterogêneo ou homogêneo.

No sistema heterogêneo, existem duas alternativas de implementação: unilíngüe e multilíngue. No SVBD unilíngüe os usuários, provavelmente, irão utilizar-se de modelos de dados e linguagens diferentes para acesso ao banco de dados local e global. Já na arquitetura multilíngue, o usuário acessa o banco de dados global através de um esquema externo definido com o uso da linguagem do SGBD local do usuário. A abordagem multilíngue torna a consulta aos bancos de dados mais fáceis sob a perspectiva do usuário. Porém, elas são mais complicadas, pois deve-se trabalhar com conversões de consultas em tempo de execução (ÖZSU & VALDURIEZ, 2001).

Figura 16: Arquitetura com esquema conceitual global



Fonte: (MANICA, 2001, pp. 31)

Os bancos de dados hierárquicos e os bancos de dados em redes são modelo que atualmente ainda encontramos em produção. As tecnologias empregadas eram as melhores para seus tempos. Foram projetados para trabalharem em ambientes de grande porte como os *mainframes*. Muitos destes bancos de dados ainda encontram-se em

operação até os dias de hoje. Mais detalhes sobre estes bancos de dados pode ser encontrados em (KORTH & SILBERSCHATZ, 1995) e (DATE, 1999).

4 GERENCIAMENTO DE TRANSAÇÕES

O conceito de transação é usado em banco de dados como uma computação consistente e confiável. O banco de dados está em um estado consistente se ele obedece todas as restrições de integridades. Um banco de dados é confiável se seu mecanismo de recuperação é confiável. O gerenciamento tem por objetivo garantir que tanto o sistema quanto os dados mantenha um estado consistente e confiável, mesmo com falhas ou acesso concorrente. O objetivo deste capítulo, além de revisar os conceitos de transações, é estudar os principais modelos de transações em sistemas de banco de dados distribuídos. Usa-se o modelo de banco de dados distribuído por estar mais próximo dos bancos de dados móveis, centro de estudo deste trabalho.

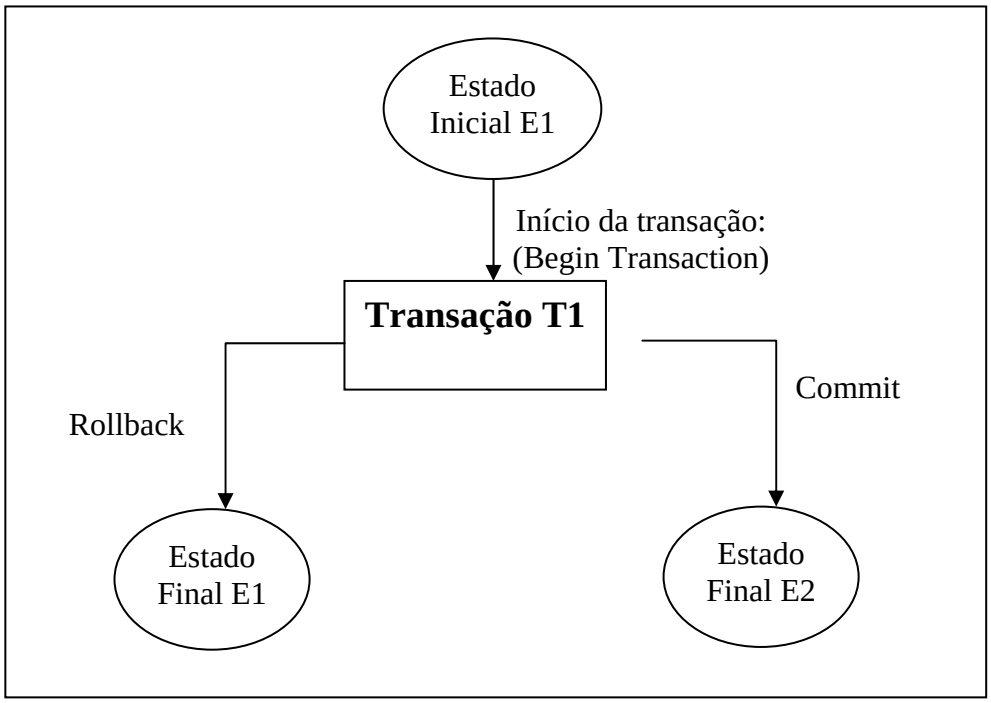
4.1 TRANSAÇÕES

Uma transação é uma unidade atômica de computação consistente e confiável. Segundo (DATE, 2000), “Uma transação é uma unidade lógica de trabalho”. Após a execução de uma transação o banco de dados modifica seu estado. A transação representa várias operações no banco de dados e seu objetivo é manter consistente o estado do banco de dados, antes e depois das atualizações realizadas.

Pode-se terminar uma transação com todas as operações realizadas ou sem nenhuma operação realizada, porém todas as transações devem terminar. A transação terminada com todas as suas operações realizadas denomina-se transação compromissada. Uma transação compromissada termina com o comando *commit*. Quando uma operação falha, todas as operações pertencentes à transação devem falhar também. Neste caso a transação tem o nome de transação abortada e termina com o comando *abort* ou *rollback*. A mudança do estado do banco de dados é conseqüência de um *commit*. Já o *rollback* não altera o estado do banco de dados, visto que nenhuma

ação foi realizada (KORTH & SILBERSCHATZ, 1995). A figura 17 exemplifica o esquema de uma transação.

Figura 17: Esquema de uma transação



O banco de dados possui um estado inicial E1 em um certo instante. Após o início de uma nova transação (*Begin Transaction*) a transação poderá ser finalizada com o comando *rollback*, voltando assim ao estado E1. Caso a finalização da transação aconteça com o comando *commit*, o banco de dados terá um novo estado E2.

Na grande maioria dos SGBDs existe a figura do gerenciador de transações. Suas principais tarefas são a recuperação do banco de dados e o controle da concorrência.

4.2 PROPRIEDADES DAS TRANSAÇÕES

As transações possuem quatro importantes propriedades (DATE, 2000):

- Atomicidade: As transações são atômicas, isto é, ou tudo ou nada. Todas as operações devem estar completas ou nenhuma operação será realizada;
- Consistência: As transações preservam a consistência do banco de dados. Antes de iniciar a transação o banco de dados deve estar em um estado consistente e permanecendo assim após a execução da transação;
- Isolamento: As transações são isoladas uma das outras. Existem várias transações ocorrendo simultaneamente no banco de dados, porém os dados que elas estão atualizando devem estar isolados um do outro, isto é, duas transações distintas não podem estar atualizando o mesmo item de dados em transações diferentes;
- Durabilidade: Depois de efetivada as transações, elas devem permanecer no banco de dados mesmo ocorrendo uma falha no sistema;

4.3 MODELOS DE TRANSAÇÕES DISTRIBUÍDAS

A literatura apresenta diversos modelos para gerenciamento de transações. Entre todos eles existem as mesmas preocupações em garantir as propriedades ACID. Em muitos modelos algumas das propriedades ACIDs são minimizadas para atender algum aspecto, removendo alguns problemas e adicionando outros.

As transações classificam-se de acordo com alguns critérios. Um desses critérios é quanto ao tempo de duração da transação. Segundo este critério as transações podem ser classificadas como on-line ou em lotes (Gray, 1987, appud). Os nomes mais usuais para estas transações são: transações de curta duração ou transação de longa duração. Transações on-line possuem tempos de duração muitos curtos, na ordem de poucos segundos. As transações de curta duração são as mais usadas nos aplicativos de transações atuais, como: transações bancárias, transações de reservas de passagens

aéreas, entre outras (ÖZSU & VALDURIEZ, 2001).

Já as transações em lotes levam mais tempo para serem executadas, necessitando de maior recursos do banco de dados, além de seu tempo de resposta estar na ordem de minutos, horas ou até mesmo em dias. Aplicativos envolvendo banco de dados para computação gráfica, aplicativos estatísticos, processamento de imagens, entre outros, são exemplos de uso das transações de longa duração (ÖZSU & VALDURIEZ, 2001).

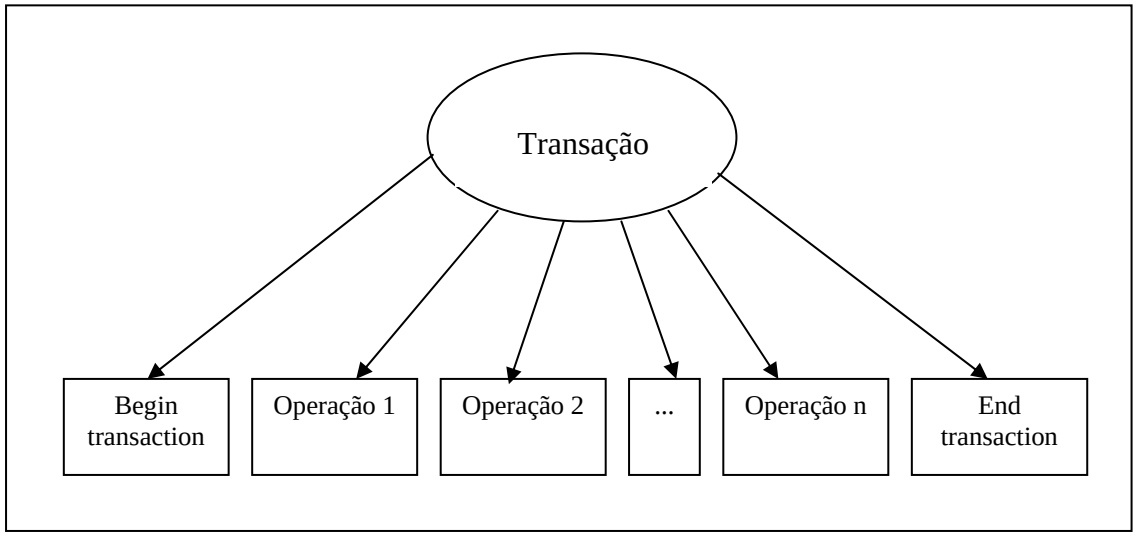
Outra classificação importante diz respeito à organização das ações de leitura e gravação. Alguns modelos consideram as leituras e gravações sem nenhuma ordem específica, isto é, misturam as tarefas de leitura e de gravação. Estes modelos são chamados de gerais. Se a transação é restrita a tal ponto de separar as tarefas de leitura e gravação, são chamadas de transação em duas etapas. Deste modo às transações são restritas a tal ponto que, necessariamente, deve ocorrer à leitura antes da gravação do dado. Existe ainda o modelo de ação das transações, onde cada par (leitura, gravação) seja executado de forma atômica (ÖZSU & VALDURIEZ, 2001).

As transações ainda podem ser classificadas de acordo com suas estruturas. São separadas em quatro grandes categorias de complexidade crescente (ÖZSU & VALDURIEZ, 2001):

- Transações planas;
- Transações aninhadas fechadas;
- Transações aninhadas abertas;
- Modelo de fluxo de trabalho (*workflow*);

As transações planas possuem um único ponto de início *(begin transaction*) e um único ponto de fim *(end transaction*), conforme figura 18. Este modelo apresenta uma única camada de controle para a aplicação. A transação não pode ser efetivada ou cancelada parcialmente, visto que possui apenas um ponto de início e um ponto de fim. Este modelo é o mais utilizado nos bancos de dados comerciais existentes.

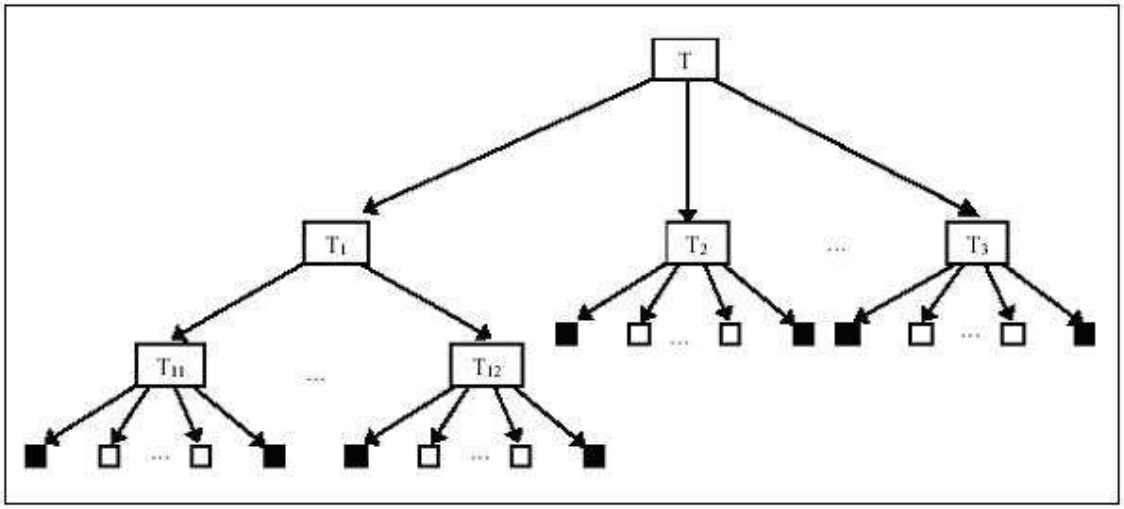
Figura 18: Modelo de transações planas



No modelo de transações aninhadas, as transações podem ser vistas como uma árvore de transações planas, ver figura 19. Uma transação raiz, chamada de transação pai, é dividida em várias outras sub-transações, chamadas de transações filhas. Por sua vez as subtransações podem ser divididas novamente em outras sub-transações e assim por diante. Apenas a transação do nível folha contém operações sobre os dados, enquanto as transações de dos nós intermediários fornecem uma espécie de controle para seus descendentes. Pode-se dizer que as transações aninhadas significam um conceito mais generalizado de transação.

Consideram-se transações aninhadas fechadas ou abertas devido a sua característica de término. As transações aninhadas fechadas consolidam-se de baixo para cima, isto é, encerra-se do nível folha para o nível raiz. Por conseguinte, uma subtransação aninhada começa depois de seu pai e termina antes dele, e a consolidação da transação é condicionada a do pai. A atomicidade fica a cargo do nível superior. Já as transações aninhadas abertas relaxa a atomicidade do nível superior de transações aninhadas fechadas. Assim, uma transação aninhada aberta permite que seus resultados parciais possam ser observados por outras transações, visto que elas são encerradas em sentido contrário das transações aninhadas fechadas.

Figura 19: Modelo de transações aninhadas



Fonte: (MANICA, 2001, pp. 75)

Sagas e as transações divididas são exemplos de transações aninhadas aberto. O modelo Sagas é definido como uma transação de longa duração. É composta por um conjunto de subtransações relativamente independentes. Associada a cada subtransação existe uma subtransação compensatória.

As transações planas modelam bem as atividades relativamente simples e curtas, sendo impróprias para transações de longa duração. A combinação de modelos de transações abertas e transações aninhadas formam outros modelos de transações mais complexos. Estes modelos não seguem nenhuma das propriedades ACID. Dá-se o nome de *workflow* para estes modelos.

Para (GEORGAKOPOULOS et al, 1995), "*workflow* é uma coleção de tarefas organizadas para a execução de algum processo de negócios". Devido ao contexto em que é empregado o termo, a definição dada se torna um tanto quanto confusa. Em (GEORGAKOPOULOS et al, 1995), são identificados três tipos de *workflows*:

- *Workflows* orientados para pessoas: envolve as pessoas na realização das tarefas. O sistema fornece o suporte para a colaboração e coordenação entre as pessoas, mas as responsabilidades pela coerência das informações ficam a cargo delas mesmas;
- *Workflows* orientados para sistemas: Consiste em tarefas com o uso da computação. Neste caso, o suporte do sistema envolve o controle da concorrência e recuperação, execução automática de tarefas, notificações, etc.

- *Workflows transactions*: Varia entre os dois tipos de *workflows* apresentados acima. Neste tipo existe a execução coordenada de várias tarefas que podem envolver pessoas e sistemas. Admitem o uso seletivo de propriedades transacionais (ACID) para tarefas individuais o para todo o *workflow*.

Modela-se um *workflow* como uma atividade que tem uma semântica de aninhamento aberto, no sentido de permitir que os resultados parciais possam ser vistos fora dos limites da atividade. As tarefas podem ser outras atividades ou transações. Porém, uma vez iniciado uma tarefa de transação fechada esta somente poderá conter outras tarefas de transações fechadas. Geram-se transações compensatórias para garantir que todas as tarefas sejam "desfeitas" caso uma das tarefas falhe. Pode-se ainda, conter tarefas de contingência. Caso alguma tarefa falhe, outra será executada em seu lugar (ÖZSU e VALDURIEZ, 2001).

4.4 CONTROLE DISTRIBUÍDO DA CONCORRÊNCIA

O controle da concorrência está diretamente relacionado com o isolamento das transações e sua consistência. O mecanismo de controle distribuído da concorrência garante que a consistência do banco de dados seja assegurada em um ambiente distribuído.

Quando vários usuários executam suas transações, em sites diferentes ou no mesmo site, que afetam objetos armazenados em sites diferentes surge o problema do controle da concorrência distribuída.

O mecanismo de controle da concorrência deve encontrar um equilíbrio entre a manutenção consistente do banco de dados e a manutenção de um nível elevado de concorrência.

Pode-se implementar o controle da concorrência de diversos modos, assim com sua classificação. Os critérios para a classificação podem estar relacionados ao modo de distribuição do banco de dados, da topologia de rede, etc. O método mais comum é a

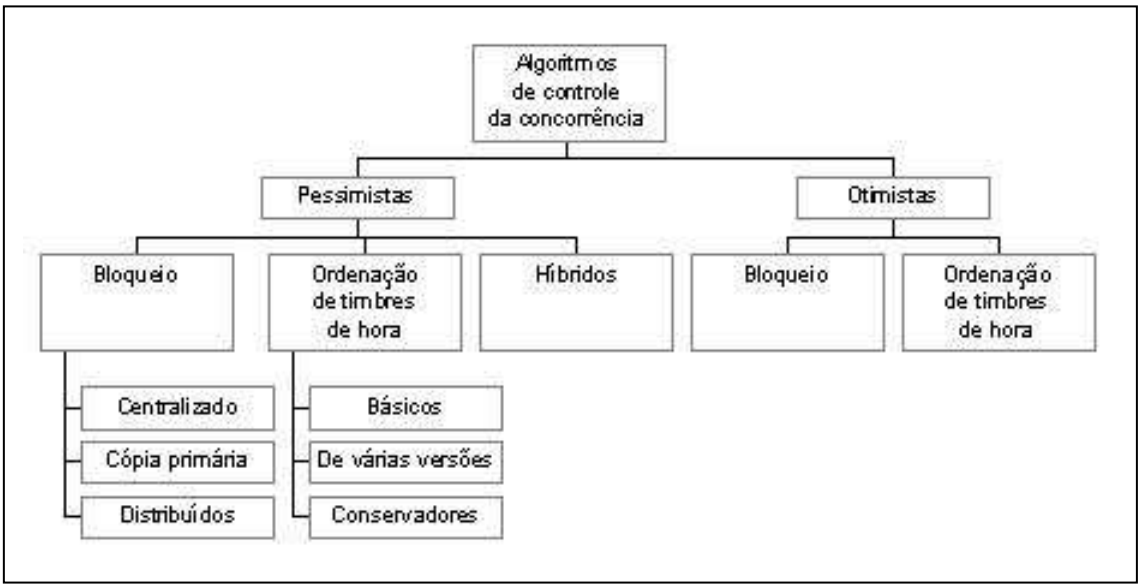
primitiva de sincronização. Para (ÖZSU e VALDURIEZ, 2001), os mecanismos de controle de concorrência podem ser agrupados em duas partes: os algoritmos pessimistas e os otimistas, conforme a figura 20.

Os algoritmos pessimistas sincronizam a execução concorrente das transações mais cedo em seu ciclo de execução. Já os algoritmos otimistas atrasam a sincronização das transações ao seu término. Os algoritmos pessimistas consistem em algoritmos de bloqueio, algoritmos baseados em ordenação de timbre de hora e algoritmos híbridos. Já os algoritmos otimistas baseiam-se em algoritmos de bloqueio e ordenação.

Na abordagem baseada em bloqueios, na visão pessimista, consegue-se a sincronização das transações aplicando bloqueios (*locks*) físicos ou lógicos. Esta classe pode ser subdividia em de acordo com locais onde estes bloqueios são executados (ÖZSU e VALDURIEZ, 2001):

- Bloqueio centralizado: um dos sites da rede é designado como site primário no qual são armazenadas as tabelas de bloqueio para o banco de dados inteiro e recebe a responsabilidade de conceder bloqueios a transações;
- Bloqueio de cópia primária: Uma das cópias de cada unidade de bloqueio é designada como cópia primária. Esta cópia deve ser bloqueada para o propósito de acessar essa unidade específica. Caso o banco de dados não seja replicado, os mecanismos de bloqueio de cópia primária distribuirão a responsabilidade pelo gerenciamento do bloqueio entre os diversos sites.
- Bloqueio descentralizado: A tarefa de gerenciamento do bloqueio é compartilhada entre todos os sites. A execução da transação envolve a participação e a coordenação de escalonadores. Cada escalonador fica responsável pelas unidades de bloqueio locais a esses sites.

Figura 20: Classificação dos algoritmos de controle da concorrência



Fonte: (ÖZSU e VALDURIEZ, 2001, pp. 326).

A classe de ordenação de timbres de hora (TO – *Timestamp Ordering*) envolve a organização da ordem de execução das transações para que elas mantenham a consistência mútua e de transações. Atribuem-se timbres de hora às transações e também aos itens de dados que serão armazenados no banco de dados. Este algoritmo ainda pode ser subdividido em três partes: TO Básico, TO de várias versões ou TO conservador.

Outros algoritmos baseados em bloqueios também fazem uso dos timbres de hora. Faz-se o uso do timbradores para melhorar e eficiência e o nível de concorrência. Dá-se o nome de algoritmos híbridos para essa classe.

5 BANCO DE DADOS MÓVEIS

Por tradição os grandes bancos de dados comerciais eram armazenados em grandes computadores centralizados. Com a evolução das máquinas e dos sistemas de bancos de dados, iniciou-se o processo de distribuição. Estes bancos de dados, normalmente, possuem os controles centralizados na rede fixa de computadores.

Com a mobilidade das pessoas e com a universalização da Internet, surgem novas necessidades para os bancos de dados. Entre todas a necessidade destaca-se a capacidade dos bancos de dados de acompanhar as pessoas a todo o momento em todos os lugares. Vários fatores influenciam para o desenvolvimento dos bancos de dados móveis, dentre eles destacam-se:

- A utilização crescente do uso de dispositivos móveis pessoais, como *notebooks* e *laptops*;
- O grande avanço dos sistemas de comunicação sem fio, a um custo relativamente baixo, como as redes de celulares.

A computação móvel tem provado ser de grande importância para os dias de hoje. A computação sem fio cria novas possibilidades, pois as máquinas não precisam mais de fios conectados à rede de computadores ou mesmo à rede elétrica para funcionarem. Assim não mais necessitam de endereço fixo na rede. Esta funcionalidade auxilia as pessoas em suas tarefas. Por outro lado aumenta a complexidade dos sistemas de computação. Neste contexto apresentam-se os bancos de dados móveis. Estes bancos de dados possuem características próprias para lidarem com estas restrições e complicações impostas pelo ambiente móvel.

A utilização de banco de dados móveis garante aos usuários facilidade de locomoção física, possibilitando o acesso aos dados a qualquer hora e em qualquer lugar. Executivos podem continuar seus trabalhos mesmo em viagens, vendedores tem posições de estoques em tempo real, outros profissionais podem continuar suas tarefas de *workflow* mesmo em casa para não perder prazos a cumprir. Estas são algumas das grandes vantagens dos bancos de dados móveis.

Para o ambiente da computação móvel, os bancos de dados necessitam autonomia

para funcionarem ligados à rede, como os computadores fixos, o fora dela. As principais funcionalidades existentes nos bancos de dados para os computadores da rede fixa devem existir nos bancos de dados móveis. Mesmo desconectado da rede, o banco de dados móvel deve permitir ao usuário consultar as informações contidas em cache ou mesmo realizar transações.

5.1 MODELOS DE SGBDS MÓVEIS

As dificuldades impostas pelo ambiente móvel, conforme visto no capítulo 2, induzem a variância da disponibilidade e dos recursos computacionais. Estas restrições afetam diretamente nas arquiteturas e nos modelos de aplicações móveis. Os modelos de computação móvel devem garantir um meio eficiente para as aplicações existentes e para as novas aplicações.

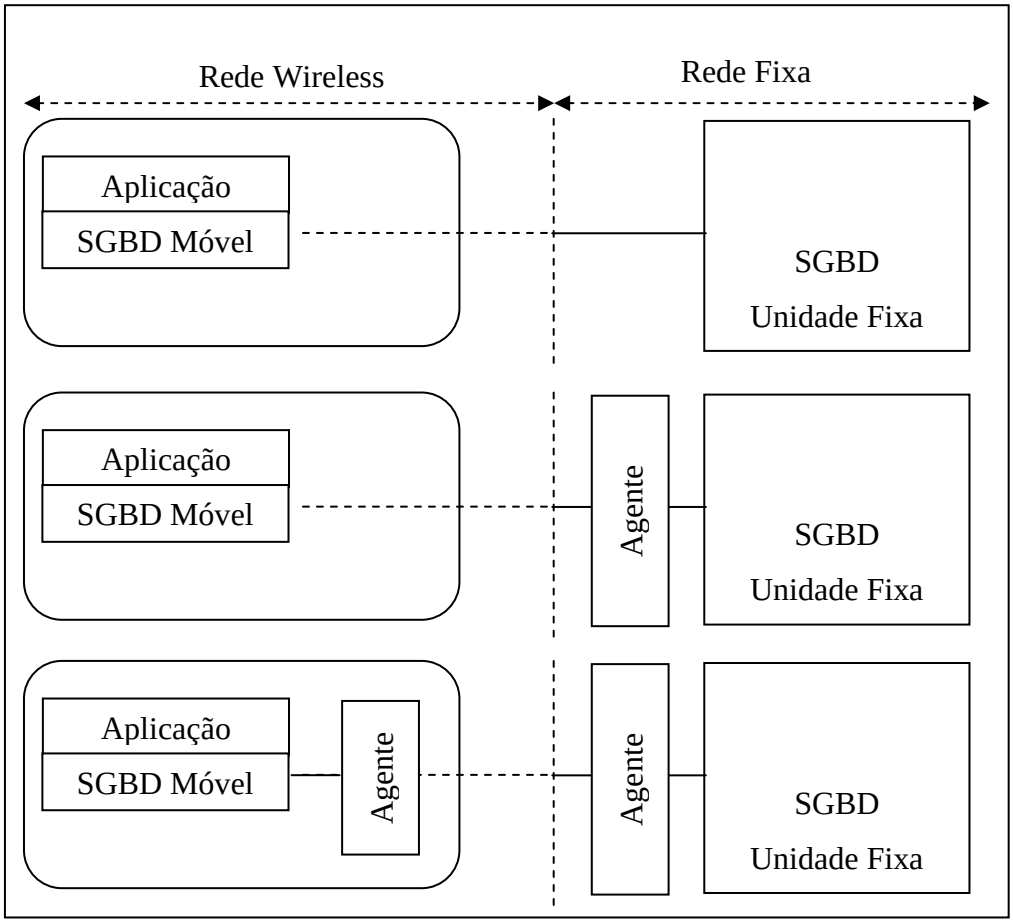
Os problemas inerentes ao ambiente móvel influem diretamente no desenvolvimento de sua arquitetura. Em (PITOURA e SAMARAS, 1998), são apresentados três modelos para computação móvel: o modelo cliente/servidor, o modelo simétrico e o modelo de agentes móveis.

5.1.1 Modelo Cliente/Servidor

No modelo tradicional cliente/servidor, o servidor é responsável pelo gerenciamento dos dados enquanto o cliente pela interface com o usuário. As mensagens são enviadas para o servidor através de algum mecanismo de comunicação, por exemplo, RPC. As mensagens são interpretadas pelo servidor que, por sua vez, retorna o resultado do processamento para o cliente. O processamento pode ser desde uma simples pesquisa na base de dados como uma transação complexa. Assim, todas as transações são realizadas no servidor, não dando autonomia para o cliente.

Para o ambiente de computação móvel, os SGBDM (Sistema Gerenciador de Banco de Dados Móvel) podem ser divididos em: o modelo cliente/servidor estendido, o modelo cliente/agente/servidor e o modelo cliente/agente/agente/servidor, como pode ser visto na figura 21

Figura 21: Modelos Cliente/Servidor para SGBDM



Fonte: (PITOURA e SAMARAS, 1998, pp. 18).

5.1.1.1 Modelo Cliente/Servidor Estendido

O modelo cliente/servidor tradicional não atende as necessidades dos bancos de dados móveis. No modelo cliente/servidor tradicional tanto as máquinas clientes como as máquinas servidoras estão conectadas à rede de computadores. Acessos e direitos são dados aos usuários em modo online. Todos os controles do banco de dados são garantidos, para cada cliente, através de uma sessão estabelecida no SGBD servidor. Uma vez a sessão finalizada, liberam-se todos os recursos alocados. O fim voluntário da sessão causa a efetivação da transação através de um *commit*, caso contrário será executando um *rollback* na transação ativa (ORACLE, 2002).

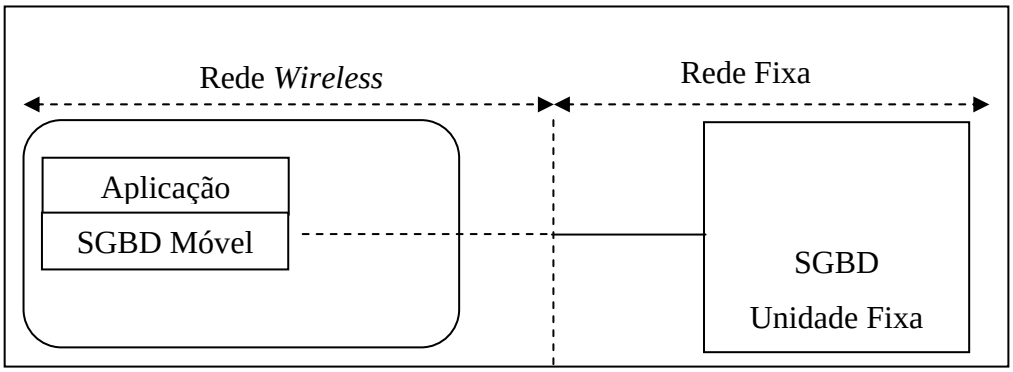
Estando no modo online, o cliente móvel se comporta como se fosse um cliente qualquer da rede fixa, visto que todas as mensagens são encaminhadas para o servidor de banco de dados da rede fixa. Assim, não é necessária nenhuma modificação para o tradicional modelo cliente/servidor.

No ambiente móvel, a possibilidade do cliente móvel desconectar-se da rede a qualquer instante é maior que na rede fixa. Aspectos como o gerenciamento de energia, interferência, a falta do sinal de comunicação ou mesmo um *handoff* pode causar uma desconexão involuntária da unidade móvel.

A mudança na estrutura cliente/servidor tradicional faz-se necessária para que o SGBDM (Sistema Gerenciador de Banco de Dados Móveis) possa continuar em atividade mesmo em modo *offline*, ver figura 22. Técnicas como filas de mensagens RPC devem ser implementadas para garantir transparência entre os SGBD e o SGBDM.

Normalmente o SGBDM trabalha em conjunto com um SGDB localizados na rede física, pois a capacidade de armazenamento da unidade móvel é muito limitada. Com a operação em modo desconectado faz-se necessárias técnicas específicas para garantir as propriedades ACID de uma transação. Problemas como a replicação de dados, gerenciamento de transações e recuperação de falhas serão discutidas mais adiante, neste capítulo.

Figura 22: Modelo Cliente/Servidor Estendido



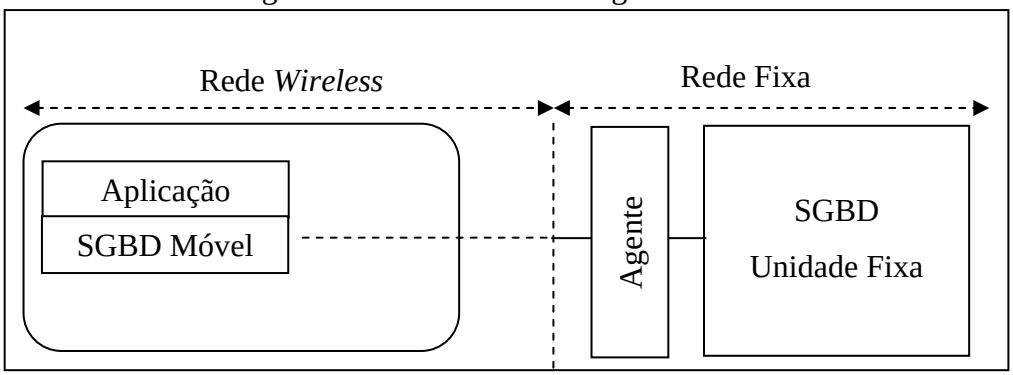
Fonte: (PITOURA e SAMARAS, 1998, pp. 18)

5.1.1.2 Modelo Cliente/Agente/Servidor

O modelo cliente/agente/servidor, conforme figura 23, também pode ser encontrado na literatura como modelo em três camadas, *three-tier* ou *proxy*. Em redes com infra-estrutura os *proxies* podem estar localizados nas estações de suporte à mobilidade (ESM ou MSS – *Mobile Station Support*). Este modelo é mais apropriado onde existe um ambiente de vários bancos de dados, ou seja, um ambiente de banco de dados heterogêneos. O Agente pode fazer a tradução do protocolo utilizado entre o SGBDM e o SGBD (PITOURA e SAMARAS, 1998).

Outra funcionalidade do agente está na otimização da largura de banda existente, efetuando a concentração e/ou compactação dos dados antes de serem transmitidos ao SGBDM. Neste modelo o SGBDM pode fazer uma solicitação ao agente e entrar no modo *doze* (dormindo) para economizar energia. O agente irá atender a solicitação do SGBDM reunindo as informações necessárias. Somente após a execução da tarefa o *proxy* enviará o resultado para o SGBDM. Técnicas específicas podem ser usadas de acordo com o tipo de trabalho a ser executado, tipo dos dados e do tipo da aplicação (PITOURA e SAMARAS, 1998).

Figura 23: Modelo Cliente/Agente/Servidor



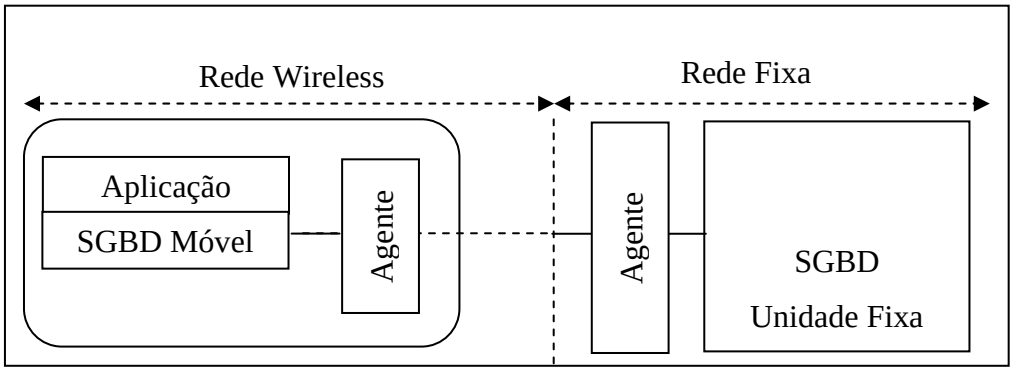
Fonte: (PITOURA e SAMARAS, 1998, pp. 18)

5.1.1.3 Modelo Cliente/Agente/Agente/Servidor

Neste modelo existe um agente localizado na unidade móvel e outro agente na rede fixa, ver figura 24. Eles estão entre os SGDBM e o SGBD. O agente, localizado na unidade móvel, intercepta as chamados do cliente e juntamente com o agente servidor executam as otimizações necessárias para a redução dos dados no canal de comunicação. O agente cliente faz o papel de um servidor *proxy* para as aplicações na unidade móvel. Eles são totalmente transparentes para ambos os lados aplicação (PITOURA e SAMARAS, 1998).

A grande vantagem deste modelo está na abstração dos SGBDM para com as técnicas de comunicação e redução dos efeitos do ambiente móvel. O modelo também apresenta escalabilidade de serviço e protocolos. Protocolos diferentes podem ser usados para assegurar a sincronização dos dados e de comunicação com o agente servidor, como HTTP *(Hyper Text Transfer Protocol)*, FTP (*File Transfer Protocol*), entre outros.

Figura 24: Modelo Cliente/Agente/Servidor



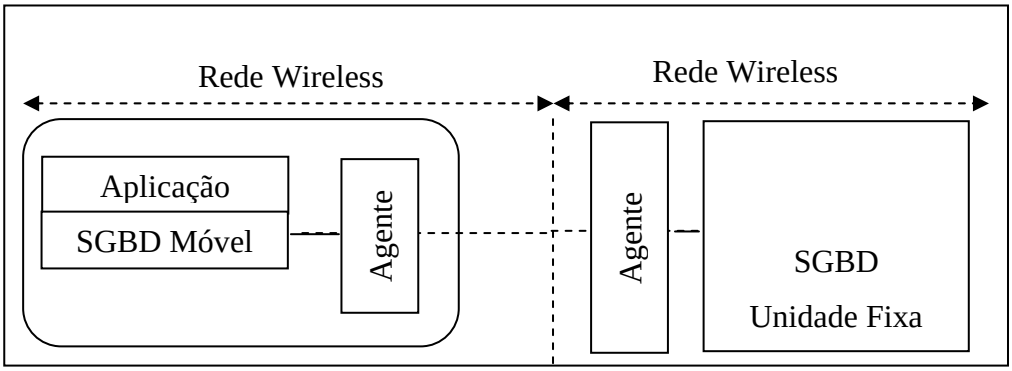
Fonte: (PITOURA e SAMARAS, 1998, pp.18)

5.1.2 Modelo Simétrico

No modelo simétrico os servidores fixos e os servidores móveis são considerados como idênticos. Os SGBDMs possuem todas as funcionalidades de cliente e de servidor. Cada *site* tem ambos os serviços: de cliente e de servidor. Um SGBDM pode comunicar-se com outros SGBDMs diretamente. Este é um caso de servidores de banco de dados totalmente distribuído. Clientes da rede fixa podem acessar as informações contidas no servidor móvel e vice-versa.

Nos modelos anteriores, o servidor localizava-se na parte fixa da rede. Assim os dispositivos móveis eram sempre clientes, operando como servidores somente no modo off-line. Neste modelo o servidor pode estar localizado na estação fixa ou móvel, tanto para o modo on-line como no modo off-line, sendo mais apropriado em ambientes onde as conexões entre as unidades móveis são mais fortes que as conexões com os servidores da rede fixa, caso das redes *ad-hoc*.

Figura 25: Modelo simétrico



Fonte: (PITOURA e SAMARAS, 1998, pp. 18)

5.1.3 Modelo de Agentes Móveis

Os agentes móveis podem ser usados em conjunto com os modelos anteriores. Oferecem outras facilidades como acompanhar as unidades móveis, redefinindo suas atividades de acordo como o ambiente e ao longo do tempo.

Este modelo tem como vantagem a economia de recursos da unidade móvel. Podem-se enviar agentes móveis pra a rede fixa com determinada tarefa. Enquanto os agentes realizam sue trabalho a unidade móvel pode entrar no modo "*doze*" ou mesmo desconectar da rede. Assim, recursos da unidade móvel são economizados.

Segundo (CHESS et all, 1995), os agentes móveis são processos (ou objetos ativos) que tem a capacidade de migrar entre computadores de uma rede durante a sua execução, carregando consigo o seu estado de execução.

Percebe-se que existem vantagens com o uso dos agentes móveis, porém aspectos sobre segurança ainda merecem atenção, entre eles pode-se citar: a garantia que somente os agentes com de origem autenticada possam executar ou a garantia que a execução do código agente não afete a máquina hospedeira.

5.2 REPLICAÇÃO DE DADOS

Para que o usuário consiga trabalhar em modo desconectado, faz-se necessário uma cópia dos dados em sua unidade móvel. A unidade móvel pode ter acesso aos dados independentemente de sua localização geográfica, ou mesmo em movimento (JING et all, 1999), e caso ocorra a desconexão da unidade móvel o sistema continua em operação. Com a desconexão o usuário não teria acesso aos dados em seu servidor corporativo, assim a replicação de dados torna-se importante para o ambiente móvel.

Outro aspecto importante quanto a replicação dos dados está no fato de que os equipamentos móveis dispõe de pouco espaço para armazenamento, portanto torna-se imprescindível que somente os dados necessários para a tarefa esteja em seu cache local.

Uma vez que a unidade móvel possui uma cópia do dado localmente, todas as leituras são feitas sem a comunicação com a rede, isto é, sem a necessidade de comunicação entre a unidade móvel e o servidor. Tão importante quanto a replicação é a sincronização destes dados. Nos ambientes de trabalhos existem diversos usuários que podem atualizar os dados no servidor, assim o sistema deve possuir algum procedimento que garanta que as cópias locais dos dados sejam iguais aos dados originais do servidor.

Segundo (BADRINATH e PHATAK, 1998), a replicação de dados pode ser dividido em três graus:

- Replicação Total: O banco de dados possui todos os seus dados replicados em todos os *sites*. Nesta abordagem a disponibilidade aumenta em detrimento do canal de comunicação e do espaço para armazenamento dos dados;
- Inexistência de Replicação: Cada fragmento está armazenado em seu respectivo *site*. Assim, todos os fragmentos sofrem uma disjunção.
- Replicação Parcial: Apenas alguns fragmentos são replicados. Apenas os fragmentos solicitados pelos usuários.

A escolha de algum grau de replicação depende do ambiente do usuário. Dados com alta taxa de atualização podem ter algumas replicas ou nenhuma.

Além dos graus de replicação, (DUNHAM e KUMAR, 1998) define três tipos de

replicação para ambiente móvel:

- Não replicação dos dados;
- Replicação temporal;
- Replicação espacial;

A replicação temporal é a replicação tradicional em ambientes distribuídos. Os dados são replicados de tempos em tempos, de acordo com seus valores. Já a replicação espacial leva em consideração a posição geográfica da unidade móvel. Os dados podem ter valores corretos em qualquer instante, ver quadro 1.

Tabela 1: Resumo dos tipos de replicações de dados

	Não Replicado	Replicação Temporal	Replicação Espacial
Copias	uma	múltiplas	múltiplas
Valores Corretos	uma	uma por tempo	Uma por localização e por tempo
Arquitetura	centralizada	distribuída	móvel
Mobilidade	não	Não	sim

Fonte: (DUNHAM e KUMAR, 1998, pp. 14)

5.3 GERENCIAMENTO DE TRANSAÇÕES MÓVEIS

Em um ambiente de computação móvel, as transações podem ser executas em diversas localidades, portanto em diversos servidores. O uso de redes sem fio acarreta em transações de longa duração. A unidade móvel pode ter autonomia para processar transações quando está no modo desconectado. Quando a unidade móvel se conecta a rede a transação deve ser efetivada globalmente.

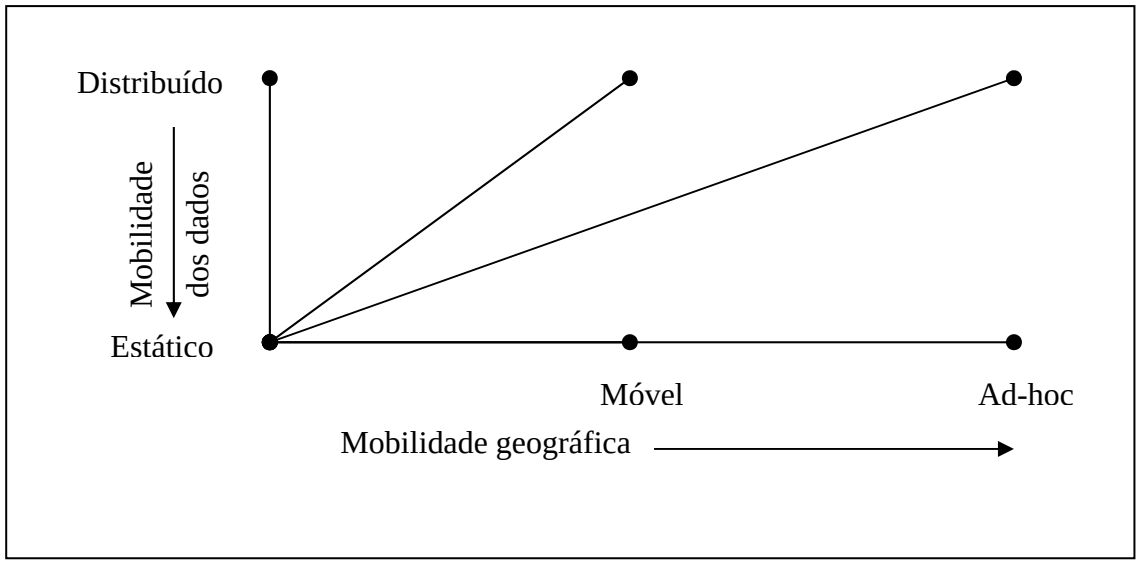
Na rede fixa, os usuários realizam transações consistentes e duráveis. Nestes ambientes as propriedades ACIDs são garantidas. A maioria das propostas de modelos de transações, apresentadas na literatura, consideram as transações móveis como sendo parte de uma transação onde existe alguma flexibilização nas propriedades ACID.

Tradicionalmente as transações são modeladas como uma seqüência de operações de leituras e escritas, normalmente iniciando com um *begin* e finalizando com um *commit* ou *abort*. No ambiente móvel as transações são mais complexas, possuem

longos tempos de vida e envolvem a possibilidade de envolver bancos de dados heterogêneos (PITOURA e BHARGAVA, 1994).

Para (DUNHAM e KUMAR, 1998), existem seis tipos, ou graus, de mobilidade. Cada um destes tipos implica em um modelo de transação, de acordo com a requisição do usuário. O mais alto grau suportado pela arquitetura é o mais alto grau permitido por qualquer transação executada nesta arquitetura, ver figura 26.

Figura 26: Graus da transação móvel



Fonte: (DUNHAM e KUMAR, 1998, pp. 9)

Os seis graus são descritos a seguir:

- (0,0): Este é o ambiente centralizado tradicional. Não é suportada nenhuma mobilidade de dados/transação ou geográfica;
- (0, 1): Este é o ambiente distribuído tradicional. A mobilidade dos dados/transação é permitida, mas a mobilidade geográfica não;
- (1,0): Nesta categoria a unidade móvel se move, porém não existe o movimento dos dados/transação. Este é o caso típico onde a transação é somente local à unidade móvel;
- (1,1): Este é o ambiente de computação móvel tradicional. A unidade móvel se move, porém não se comunica diretamente com a outra unidade móvel.
- (2,0): É o mesmo tipo de (1,0). Aqui todos os nós podem comunicar-se entre si, mas nenhuma distribuição de dados/transação é suportada;
- (2,1): Este é o ambiente *ad-hoc*. Todas as unidades movimentam-se

geograficamente e todos os dados/transação também.

As transações nos sistemas tradicionais não levam em conta a mobilidade. Alguns sistemas distribuídos tratam o aspecto tempo para suas transações, como transações de curta ou de longa duração. Neste trabalho (DUNHAM e KUMAR, 1998) incorporam o aspecto espacial para as transações móveis. Considerando este aspecto os modelos de transações sofrem mudanças, de acordo com sua classificação.

A seguir apresentam-se os teoremas para a propriedade ACID de transações móveis proposta por (DUNHAM e KUMAR, 1998).

5.3.1 Efeitos da Mobilidade na Atomicidade

A proposta da atomicidade é assegurar a consistência dos dados. Entretanto, no ambiente móvel percebe-se dois tipos de consistência. A atomicidade no nível de cada fragmento de execução faz-se necessário para assegurar a consistência espacial. Contudo, não se trata de transações atômicas. Pode algum fragmento de execução ser atômico e outros não.

Definição: Uma transação móvel, T_i, satisfaz a atomicidade espacial se cada fragmento de execução, e_{ij}, de T_i é atômica. Diz-se que T_i é espacialmente atômica se cada fragmento de execução, e_{ij}, é atômica (DUNHAM e KUMAR, 1998).

Teorema: Se todas as transações móveis satisfazem a consistência espacial então a atomicidade espacial é necessária (DUNHAM e KUMAR, 1998).

Prova: Suponha-se que uma transação móvel não satisfaz a atomicidade espacial. Então, deve existir um fragmento da transação no qual realizou uma atualização parcial do banco de dados. Portanto, a consistência espacial não é necessária (DUNHAM e KUMAR, 1998).

O inverso para este teorema não é verdadeiro. Para ser espacialmente consistente, uma transação móvel não necessariamente deverá ser atômica no nível da transação.

5.3.2 Efeitos da Mobilidade na Consistência

Em um ambiente centralizado ou distribuído, existe apenas um valor correto para cada item de dados. Utiliza-se o termo “mutuamente consistente” para indicar que os valores encontram-se corretos. Diz-se que um banco de dados replicado está em um “estado mutuamente consistente” se todas as cópias de dados possuem os mesmos valores. Já um banco de dados é dito como consistente se todas as regras de integridade são obedecidas (DUNHAM e KUMAR, 1998).

A computação móvel torna estes conceitos mais complexos. As consultas são realizadas de acordo com a localização da unidade móvel. Uma visão consistente dos dados depende da localização. Por exemplo, um viajante consultado os hotéis da cidade visitada. Trata-se o conceito original de consistência do banco de dados como consistência temporal e o conceito de consistência para o SGBDM como consistência espacial (DUNHAM e KUMAR, 1998).

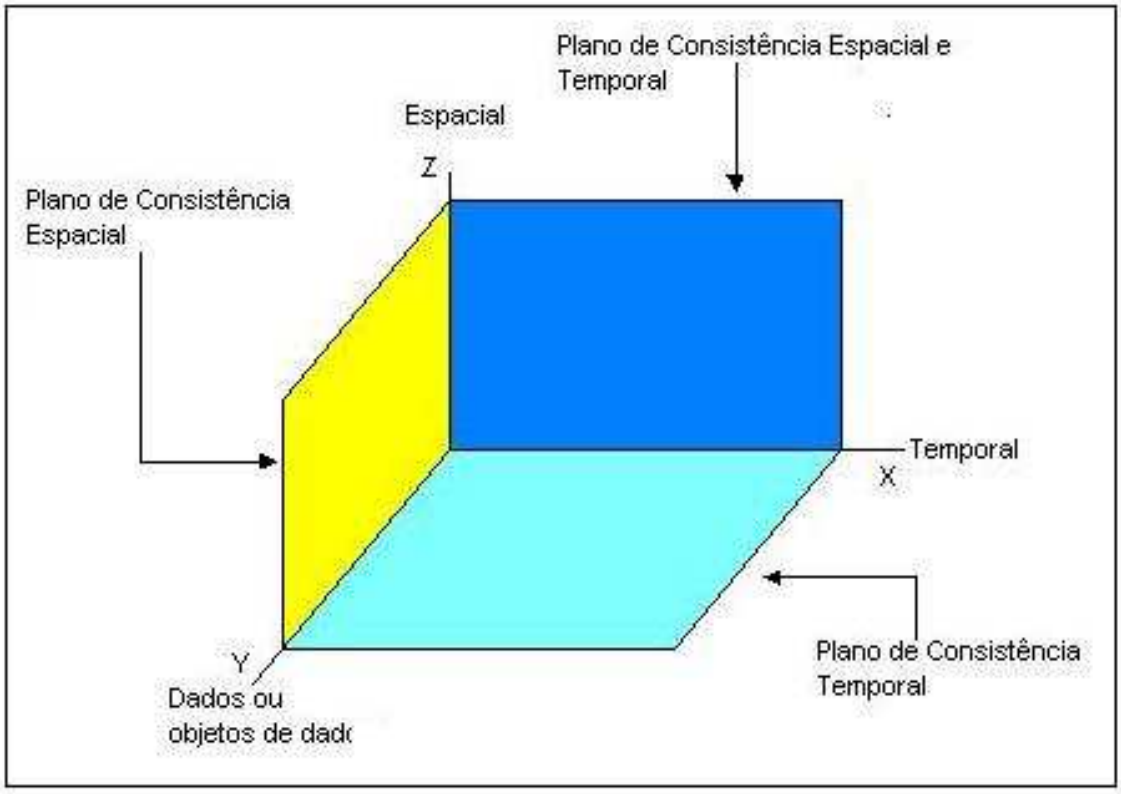
Definição: Consistência Temporal: Indica que todos os valores dos dados satisfazem um conjunto de regras d integridade. O banco de dados está em um estado consistente temporal se todas as réplicas dos dados possuem o mesmo valor.

Definição: Consistência Espacial: Indica que todos os valores para os dados de uma replicação espacial estão associados com somente uma região para os dados, satisfazendo a regra de consistência definida para a região. Este ponto indica uma transação mapeada em 1:1, conforme figura 26 acima. Uma visão mais clara sobre a consistência espacial e temporal pode ser vista na figura 27(DUNHAM e KUMAR, 1998).

Definição: (DUNHAM e KUMAR, 1998) cada fragmento de execução, e_j, de uma transação móvel, T_i, está associada a uma única localização. Dado um conjunto de $\mathcal{E}$ fragmentos de execução define-se um Mapeamento de Localização de Fragmentos (FLM – *Fragment Location Mapping*) ℓ como: $\mathcal{E} \longrightarrow \ell$

O FLM identifica a localização com respeito a como cada fragmento de execução é executado. Identifica qual replica será usada para cada dado naquele fragmento, garantindo assim a consistência dos fragmentos dentro de uma transação.

Figura 27: Graus da transação móvel



Fonte: (DUNHAM e KUMAR, 1998, pp.15)

5.3.3 Efeitos da Mobilidade no Isolamento

Isolamento das transações garante que uma transação não interfira em outra transação. O Isolamento é normalmente garantido por algum mecanismo de controle de concorrência. Como a atomicidade, o isolamento é necessário para garantir a consistência do banco de dados. Estando presente a consistência espacial, o isolamento deverá ser reavaliado. Como a atomicidade e consistência, o isolamento e muito restrito.

Definição: Uma transação móvel, T_i, satisfaz o isolamento espacial se cada fragmento de execução, e_{ij}, de T_i, é isolado para todos os fragmentos de execução de T_i, ou qualquer outra transação (DUNHAM e KUMAR, 1998).

Teorema: Se todas as transações móveis satisfazem a consistência espacial então o isolamento é necessário (DUNHAM e KUMAR, 1998).

5.3.4 Efeitos da Mobilidade na Durabilidade

Uma transação convencional faz, através de um *commit*, as atualizações necessárias no banco de dados de forma permanente. Entretanto, para garantir a consistência espacial, o isolamento espacial e a atomicidade espacial, a durabilidade também deve ser modificada.

Definição: Um fragmento de execução, e_{ij}, satisfaz um *commit* dependente de localização se o fragmento de operações terminou com uma operação de *commit* e um FLM existe. Desta forma todas as operações em e_{ij} agem sobre as réplicas de dados definidas por um DLM (*Data Location Mapping*) e sua localização identificada por um FLM. O *commit* está associado com uma única localização, ℓ . Isto indica que escreveu-se o *commit* (DUNHAM e KUMAR, 1998).

5.3.5 Definição Formal de uma Transação Móvel

Nos SGBD tradicionais, a transação é assumida como uma unidade de consistência. Já com atomicidade espacial, este é um caso de uma transação espacial. Uma transação móvel é uma unidade de consistência. Assim pode-se formalizar um fragmento de execução como:

Definição: Um fragmento de execução eij é uma ordem parcial $e_{ij} = \{\sigma_j, \leq_j\}$ onde (DUNHAM e KUMAR, 1998):

- $\sigma_j = OS_j \cup \{N_j\}$ onde $OS_j = \cup_k O_jk$, $O_jk \in$ { *read, write*}, e $N_j \in$ {*abort, commit*}. Neste ponto o *commit* e o *abort* são dependentes da localização.
- Para qualquer O_jk e O_jl onde $O_jk = R(x)$ e $O_jl = W(x)$ para um dado x, então também $O_jk \leq_j O_jl$ ou $O_jl \leq_j O_jk$.
- $\forall O_jk \in OS_j, O_jk \leq_j N_j$.

Todo fragmento é então associado a uma localização. Entretanto, se é iniciado a atualização das réplicas de dados temporais então todos os fragmentos de dados devem

ser atualizados. Assim isto não está sujeito a uma regra de localização e mostra-se como uma transação regular.

Definição: Uma transação móvel, T_i, é uma tripla <F', L', FLM'> onde F' = {e_i1, e_i2, ..., e_in}é um conjunto de fragmentos de execução, L' = {l_i1, l_i2, ...,l_in}é um conjunto de localidades, e FLM' = {flm_i1, flm_i2, ..., flm_in}é um conjunto de mapeamentos de localização de fragmentos onde $\forall_j$, $flm_ij(e_ij) = l_ij$ (DUNHAM e KUMAR, 1998).

Dado um estado do banco de dados como ambos, temporariamente e espacialmente, consistente, uma transação móvel Ti converte este estado para outro estado, temporariamente e espacialmente, consistente.

6 MODELOS DE TRANSAÇÕES MÓVEIS

Nos sistemas de banco de dados tradicionais os usuários interagem com o banco de dados por meios de transações que garantem as propriedades ACID. Estas propriedades garantem a consistência para transações cujos participantes fazem parte de uma rede fixa. Em um ambiente de computação móvel estas propriedades podem ser consideradas como sendo restritivas. Elas não suportam algumas características dos ambientes móveis, como as desconexões e o *handoff*.

A desconexão de longos períodos para uma estação móvel, além da limitação da largura de banda, requer uma revolução nos modelos e nas técnicas de gerenciamento de transações. Existem muitas propostas para modelar transações móveis com diferentes notações. Em muitas destas abordagens vêm uma transação a como um conjunto de subtransações com alguma flexibilidade na consistência e no processo de "*commit*". O gerenciamento destas transações pode ser estático em uma unidade móvel, servidor de banco de dados ou ainda se mover de estação base para estação base de acordo com a localização da estação móvel (DUNHAM et al, 1997).

As desconexões de rede não podem ser tratadas como erros ou falhas e sim como eventos inerentes ao ambiente de trabalho. A transação deve continuar até seu "*commit*" mesmo com a estação móvel desconectada, desde que os dados e os métodos necessários para completar a transação já estejam na estação base ou no servidor. As técnicas tradicionais que utilizam a serialização (ex. monitores de transações, bloqueios, etc) não funcionam corretamente no ambiente desconectado sendo necessário o desenvolvimento de novos mecanismos para o processamento de transações móveis.

As aplicações para computação móvel podem envolver diferentes tarefas, incluindo-se as transações com longos tempos de vida e algumas tarefas de processamento de dados (ALONSO & KORTH, 1993). Alguns usuários necessitam estar apto a trabalharem efetivamente em um estado desconectado. A unidade móvel necessitará de algum grau de gerenciamento de transações. Sendo assim, esquemas de controle de concorrência para bancos de dados distribuídos móveis deveriam suportar operações autônomas durante a desconexão. Deve-se considerar também o trafego de

mensagens em relação a limitação da largura de banda além da nova localidade após o movimento da estação móvel (SEYDIM, 1999).

Para (DUNHAM et al, 1997), existem quatro requisitos básicos para uma transação móvel. São eles:

- Construir sobre os sistemas de múltiplos bancos de dados existentes e não duplicar o suporte necessário pelo sistema;
- Capturar os movimentos das transações móveis assim como o acesso aos dados. Mover o controle da transação assim como a unidade móvel se move.
- Garantir flexibilidade nos requisitos da atomicidade;
- Suportar transações com longos tempos de vida.

Já (TEWARI & GRILLO, 1995) apresentam seis requisitos para o processamento de transações em ambientes móveis:

- Habilidade de separar a computação. A estação móvel tem limitações de memória, processador, tempo de vida das baterias, sendo necessário realizar parte da computação na estação base e parte na estação móvel;
- Compartilhamento dos estados e dos resultados parciais. Como parte da transação é realizada na unidade móvel e o restante da transação é executado em parte da rede fixa, a transação deverá compartilhar seus estados e resultados parciais;
- Suportar transações com longos tempos de vida. Como a fonte dos dados e a localização da unidade móvel podem mover-se, a procura da localidade deve ser parte importante para a transação;
- Manusear falhas parciais e prover diferentes estratégias de recuperação. O ambiente de computação móvel necessita manusear falhas parciais resultantes de baixa carga da bateria, interferências, desligamento acidental e outros cenários que não conduz a uma falha completa. Consequentemente novas estratégias de recuperação deve ser projetada.
- Suportar o *Handoff*. O sistema deve ser capaz de mostrar o mesmo ambiente para a unidade móvel mesmo após um *handoff*.

Em muitos modelos algumas propriedades ACID foram relaxadas ou até mesmo retiradas para adaptar-se ao ambiente móvel. Muitas pesquisas encaminham-se para a solução destes problemas. Adiante apresentam-se alguns modelos para gerenciamento de transações móveis.

6.1 MODELO KANGAROO

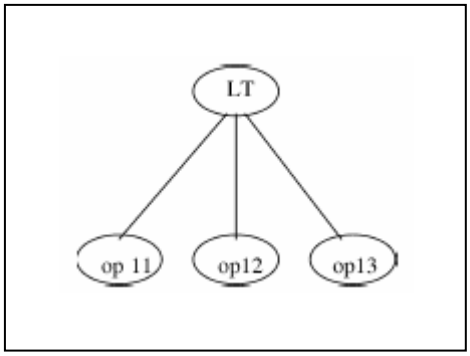
O modelo de transações chamado *Kangaroo Transactions*, proposto por (DUNHAM et al, 1997), incorpora e pressupõe que as transações, em um ambiente móvel, mudam de uma estação base para outra enquanto a unidade móvel se movimenta. O modelo captura os comportamentos dos movimentos e o comportamento dos dados. Refletem nos dados localizados nos bancos de dados na rede fixa.

O modelo assume um Agente de Acesso aos Dados (DAA – *Data Access Agent*) em cada servidor de dados e em cada estação base, usado para acessar os dados. O DAA envia as requisições de dados para o servidor que contém os dados requisitados.

Este modelo baseia-se no modelo tradicional de transações onde cada seqüência de operações é executada sob o controle do sistema gerenciador de banco de dados, conforme figura 28. As três operações (op11, op12 e op13) são executadas como parte de uma transação. Usa-se LT para identificar a transação tradicional. Para alguns sistemas gerenciadores de banco de dados, executa-se LT como uma transação local. As operações normalmente executadas são: leitura (*read*), escrita (*write*), inicio de transação (*begin transaction*), cancelamento de transação (*abort transaction*) e finalização de transação (*commit transaction*). Para a execução da primeira operação, op11, faz-se necessário o comando de *begin transaction* e após a execução da operação op13 o comando de *commit transaction* ou *abort transaction* (DUNHAM et al, 1997).

Para (DUNHAM et al, 1997), “Nossa visão para transações globais em um sistema de vários bancos de dados é um tanto quanto plena que aquelas muitas vezes assumidas”. Assumem-se dois tipos de visões de transação: Transação Global Limitada e a Transação Global.

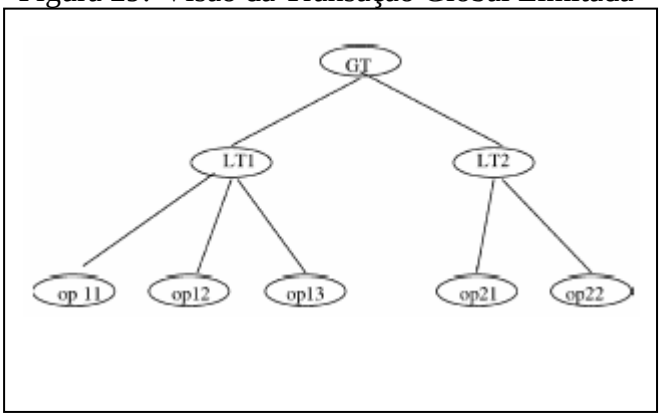
Figura 28: Estrutura base do modelo Kangaroo



Fonte: (DUNHAM et al, 1997, pp. 154)

A transação global GT compõe-se de subtransações que podem ser vistas como transações locais LT de um banco de dados, sendo cada uma destas subtransações LT uma seqüência de operações OP, figura 29.

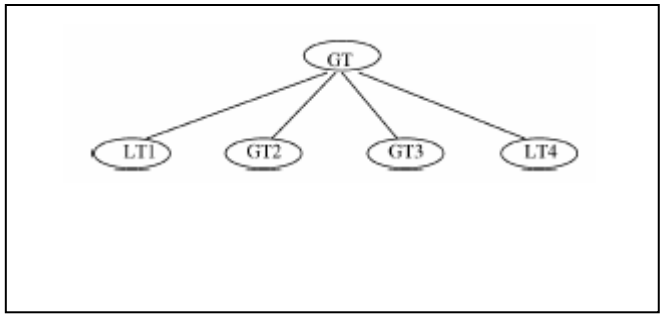
Figura 29: Visão da Transação Global Limitada



Fonte: (DUNHAM et al, 1997, pp. 154)

Têm-se várias transações globais GT para vários bancos de dados. Cada GT pode conter várias transações locais LT e/ou várias operações OP, figura 30.

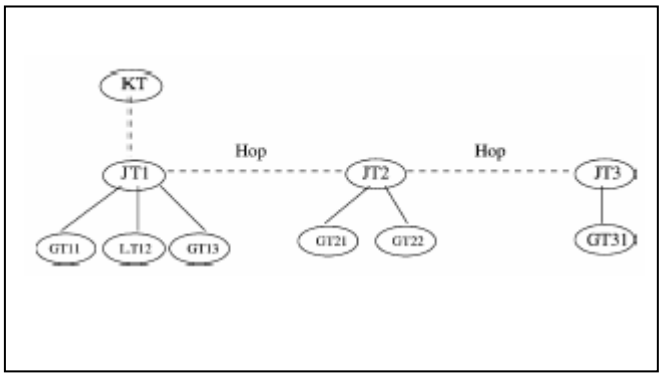
Figura 30: Visão da Transação Global



Fonte: (DUNHAM et al, 1997, pp. 154)

As GT sozinhas não capturam os “saltos” (*hop*) da estação móvel entre as várias estações base. Baseado nestes *hops* dá-se o nome de *Kangaroo Transactions*. A estrutura básica de funcionamento do modelo *Kangaroo* pode ser vista na figura 31.

Figura 31: Visão da Transação Global Limitada com saltos (*hops*)



Fonte: (DUNHAM et al, 1997, pp. 154)

Quando uma transação é requisitada pela unidade móvel o DAA, que está associado à estação base, cria uma transação móvel identificada como KT. Um identificador (ID) é criado para identificar esta transação KT. Denominam-se estes identificadores como KTID. Um KTID compõe-se do identificador da estação base concatenada a um número seqüencial (KTID = *Base Station ID* + *Sequence Number*), onde ID da estação base é único e o número seqüencial é único para àquela estação base. (DUNHAM et al, 1997).

Cada subtransação representa uma unidade de execução um uma estação base sendo chamada de *Joey Transaction* – JT. A diferença entre KT e JT é que o JT é parte de uma KT e deve ser coordenada por uma estação base. Quando a unidade móvel passa de uma estação base para outra acontece um *hop*. O Controle do KT passa para a DAA

na nova estação base que a unidade móvel está conectada. O DAA desta estação base cria uma nova JT. As JTs podem sofrer um *commit* de forma independente. A JT1 sofrer um *commit* independente da JT2 e JT3. Porém uma falha em um JT pode causar o cancelamento de toda a KT.

Para gerenciar a execução da KT e o processo de recuperação uma lista duplamente encadeada é mantida entre a estação base e os sites envolvidos. O Controle da informação sobre a JT é identificado pelo JTID.

Existem dois modos diferentes de processar uma KT: modo de compensação (*compensating mode*) e modo dividido (*split mode*). Quando uma KT é executada no modo de compensação, uma falha em qualquer JT causa o cancelamento da JT corrente e qualquer JT seguinte. As JTs já efetivadas deverão ser compensadas. Para operar neste modo o usuário ou o sistema deverá providenciar informações necessárias para criar transações compensáveis. O modo dividido é o padrão. Neste modo, quando uma JT falha uma nova global ou local transação é solicitada como parte da KT. A decisão para um *commit* ou um *abort* é deixada para o sistema gerenciador de banco de dados. As JTs já efetivadas não serão compensadas. Nenhum dos dois modos garante a serialização das KT. Embora o modo de compensação garanta a atomicidade, o isolamento poderá ser violado, visto que os bloqueios são assegurados e liberados no nível de transação local. Já no modo dividido as sub-transações JT são serializáveis.

6.1.1 Definições Formais

Segundo (DUNHAM et al, 1997), para melhor compreender o modelo Kangaroo, faz-se as seguintes definições:

- Uma transação local LT é uma seqüência de *read* (r_i), *write* (w_i), *commit* (c_i) e abort (c_i);
- Uma Transação Global GT é qualquer seqüência de transações globais G_j e transações locais L_j;

- Uma *Joey Transaction* JT é uma seqüência de zero ou mais transações globais G_k e transações locais L_k, seguidos de operações de *commit* c_k, *abort* a_k e *split* s_k;
- Uma Transação *Kangaroo* é uma seqüência de zero ou mais *Joey Transaction* J_l. A última *Joey Transaction* deve ser finalizada com um *commit* c_l ou um *abort* a_l. Todas as *Joey Transactions* anteriores a última devem ser finalizadas com um *split*;
- A seqüência de transações locais e globais pertencentes a mesma transação *Kangaroo* são chamadas de *Pouch*;
- Duas transações Kangaroo são ditas equivalentes se elas tem o mesmo *Pouch*;

6.1.2 Processamento das Transações

Segundo (DUNHAM et al, 1997), pode-se descrever o fluxo de controle das transações Kangaroo, em um MTM, como:

- Quando uma unidade móvel lança uma Transação Kangaroo, o correspondente DAA envia a transação ao MTM para gerar um identificador único (KTID) e cria um registro na tabela de status. O MTM também cria o primeiro *Joey Transaction* para ser executado localmente na célula de comunicação. No fim desta fase, um registro BTKT é escrito dentro do *log* do MTM.
- A criação de uma *Joey Transaction* é também feita pelo MTM e envolve a geração de um identificador único JTID, criando um registro na tabela de status da transação. O contador do número de *Joeys* em KT é incrementado em um. Um registro BTJT é escrito no *log*. Finalmente uma entrada JT é escrita dentro da tabela de status de transação.
- A transação Kangaroo é executada dentro do contexto de uma Transação Joey. Para cada JT, uma tradução é feita para mapear as operações KT de um sistema específico global para transações locais. Baseado no resultado da tradução, uma entrada ST é criada na tabela de transações para cada transação

nativa (local ou global). Uma entrada no registro de log do tipo BTST é escrita no MTM antes de enviar cada transação nativa para os respectivos SGBDs.

- Quando um *handoff* ocorre, o DAA é imediatamente notificado assim que ele responde pela inicialização de um protocolo *transaction-level handoff*. Quando isto acontece, o DAA inicia a execução de uma operação de *split* (separação) no *site* original. Como parte deste *split* o DAA escreve um registro do tipo HOKT no *log* indicando que um *handoff* ocorreu. Além disto o buffer é de log é escrito em um arquivo de *log* no disco. Estas ações representam uma operação de *checkpoint.* Nota-se que subtransações ainda podem ser executadas no site.
- A estação base destino inicia a outra parte da operação de *split*. Um registro do tipo CTKT é escrito após uma nova transação *Joey* ser criada como tentativa de continuar a transação KT. Uma entrada KT é registrada na tabela de status destino.
- Para garantir a atomicidade do SGBD, ele irá determinar quando fazer o *commit* ou *rollback* das subtransações. Quando o DAA é notificado, pelo SGBD, que efetuou-se o *commit* com sucesso para a subtransação, o DAA escreve um registro do tipo ETST no arquivo de *log*. Além disto se a transação está no modo *split*, a entrada ST da transação é removida da tabela de status.Se o modo da transação KT é *Compensating* (modo de compensação), esta entrada permanece para o caso da transação KT ser abortada e uma transação *Compensating* deverá ser executada. A lista ST na tabela de status é atualizada para refletir o fato que a subtransação fez um *commit*. Se não existe nenhuma subtransação ativa para este *Joey*, um registro ETJT é escrito no *log* e o status na, na tabela de status, é atualizado para *committed*. Finalmente o contador de *Joeys* ativos é decrementado de um. Se o número de JT, na tabela de status da KT, é 0 então o status para KT passa para *committed*.
- Quando o usuário móvel indica que a transação terminou, o status para KT é trocado para *committed*. Neste ponto, se o número de *Joey* é 0 então KT está *committed*.
- Um registro do tipo ETKT é escrito no *log* quando uma KT está no status de *committed*. Todas as entradas nas tabelas de status nas estações base

envolvidas nas transações são limpas.

Pode-se representar uma transação e o registro de *log* KT conforme os quadros 2 e 3 abaixo:

Tabela 2: Entradas na tabela de status das transações KT

Tipo do Registro	**Atributo**	**Descrição**
KT	**KTID**	Identificador para a transação KT
	Mode	Tipo da operação - Split ou Compensating
	Joey Count	Número de transações JT ativas para a transação KT
	Status	Indica se a transação KT está Ativa, foi realizada um Commit ou está Abortada.
	FirstJTID	Ponteiro para o primeiro registro de status para esta transação KT
JT	**JTID**	Identificador para a transação JT
	NextJTID	Ponteiro para o registro de status da próxima JT da transação KT
	PriorJTID	Ponteiro para o registro de status da JT anterior da transação KT
	Status	Indica se a transação JT está Ativa, foi realizada um Commit ou está Abortada.
	STList	Lista para transações ST locais e globais
	Compensatable	Sim ou não
ST	**STID**	Identificador para uma transação ST
	Status	Indica se a transação ST está Ativa, foi realizada um Commit ou está Abortada.
	Request	Requisição GT ou LT
	Compensatable	Sim ou não
	ComptTR	Transação Compensating

Fonte: (DUNHAM et al, 1997)

Tabela 3: Registro de Atividades

Tipo do Registro	Conteúdo
BTKT	KTID, Mode
CTKT	KTID, Mode
BTJT	JTID, PriorJTID
BTST	STID, Request, Compensating
ETJT	JTID, NextJTID
ETST	STID
ETKT	KTID
HOKT	KTID

Fonte: (DUNHAM et al, 1997)

6.1.3 Aplicação do Modelo

Este modelo de transação móvel pode ser mais indicado para as aplicações onde a necessidade da captura do movimento da unidade móvel seja mais importante. Pode-se ter como exemplo uma aplicação para captura de dados geográficos. Dados geodésicos poderia estar sendo capturados pema unidade móvel enquanto esta desloca-se. Pode ainda ser utilizado por um sistema de vigilância de veículos, onde cada unidade móvel estaria acondicionada no veículo. Uma transação *Kangaroo* seria toda a viagem. A cada nova estação de suporte a mobilidade tem-se uma transação Joey (JT). Ao final da viagem seria efetivada o *commit* global da transação.

6.2 MODELO CLUSTERING

O modelo de transações chamado *Clustering*, proposto por (PITOURA e BHARGAVA, 1995) e (PITOURA e BHARGAVA, 1999), assume um ambiente de sistemas totalmente distribuído e projetado para manter a consistência do banco de dados. Apresenta-se como um modelo para ambientes onde as desconexões é mais freqüente.

Neste modelo o banco de dados é dinamicamente dividido em grupos (*clusters*). Os dados estão semanticamente relacionados ou proximamente alocados dentro de um *cluster*. Os dados são armazenados dentro da unidade móvel para suportar as operações em modo desconectado. Quando a unidade móvel está desconectada, as operações são realizadas nos *clusters* já armazenados dentro da unidade móvel. Para cada objetos/dados existem duas cópias, uma chamada de *Strict Version* e outra chamada de *Weak Version*. A cópia *strict version* deve, necessariamente, estar globalmente consistente, já a cópia *weak version* pode tolerar algum grau de inconsistência global, mas deve ser localmente consistente. A transação móvel pode ser tanto *strict* como

weak. As transações *weak* acessam somente as cópias *weak* enquanto as transações *strict* acessam a unidade de versão *strict* (SEYDIM, 1999) (ALVARADO et al, 2001).

Os *clusters* podem estar baseados na localização física da unidade móvel. Assim, dados localizados na unidade móvel, seus vizinhos ou em um servidor fixo são considerados como pertencentes ao mesmo *cluster*. Os dados residentes nas unidades móveis desconectadas ou em servidores remotos são considerados como pertencentes a outro *cluster*.

Os *clusters* podem ser definidos tendo como base dados semânticos, utilizando alguma característica como a localização dos dados. Pode-se também, definir os cluster de acordo com regras baseadas no usuário dos dados. Informações armazenadas em um perfil de usuário podem ser usadas para definir os dados da aplicação.

6.2.1 Definições Formais

Um banco de dados móvel (MD - *Mobile Database*) é um conjunto finito de itens de dados. Um MD é particionado dentro de um conjunto finito de *clusters* Cl_i, $_i \in N$, onde Cl_i é um conjunto de itens de dados. Diz-se que um item $x \in MD$ se $x \in Cl_i$ para algum $_i \in N$ (PITOURA e BHARGAVA, 1995).

Um estado do banco de dados é definido como mapeamento de todos os itens de dados para um valor de seu domínio. Os itens de dados são relacionados por restrições chamadas de regras de integridade. Estas regras expressam os relacionamentos que o estado do banco de dados deve satisfazer.As regras existentes dentro do mesmo cluster são chamadas de *intracluster I.* Já as regras entre itens de dados em *clusters* diferentes são chamadas de interclusters (PITOURA e BHARGAVA, 1995).

Um estado de um *cluster* é consistente se todas as regras de integridade *intracluster* são asseguradas. Um estado para um MD é consistente se todos os seus *clusters* são consistentes e se todas as regras de integridade *intercluster* são *m-degree* consistentes. Este estado para o MD chamada-se *m-consistent*. A definição de *m-consistency* para uma regra de integridade depende do tipo da regra de integridade *intercluster* (PITOURA e BHARGAVA, 1995).

Uma transação, T, é uma ordem parcial (OP, <), onde OP é um conjunto de leituras *weak* e *strict*, de escritas *weak* e *strict* e operações de *commit* ou *abort* locais e o < representa a ordem de execução delas. A ordem das operações deve ser especificada e a operação de *abort* ou *commit* deve ser a última. Duas operações conflitam (operações *weak* ou *strict*) se elas acessam a mesma cópia de um item de dados e a última operação é uma operação de escrita (*weak* ou *strict*) (PITOURA e BHARGAVA, 1995).

Dois tipos de transações são suportados, *weak* e *strict*. Uma transação *weak* WT (*Weak Transaction*) é uma transação onde OP não possui qualquer operação *strict*. Assim, uma transação *strict* ST (*Strict Transaction*) é uma transação que não possui nenhuma transação *weak.*

6.2.2 Processamento das Transações

Em um banco de dados móvel MD, alguns itens de dados (*weak*) podem não satisfazer as regras de validação *intercluster*, do ponto de vista global (*strict*). Porém, durante operações em modo desconectado, estes são os únicos itens de dados que o usuário tem acesso. Para maximizar o processamento local e reduzir o acesso à rede o usuário interage com um cluster *m-consistent* utilizando operações de leitura restrita (*weak read*) e escrita restrita (*weal write*).

Dois tipos de transações são suportados pelo modelo: transações *weak* e transações *strict*. As transações *weak* consistem em operações de leitura e escrita *weak*. Já as transações *strict* consistem em operações de leitura e escrita *strict*. As transações *weak* acessam copias dos dados dentro do mesmo cluster e podem ser consideradas como transações locais. Cada transação é decomposta em subtransações de acordo com o grau de consistência requerida pela aplicação (PITOURA e BHARGAVA, 1995).

Consideram-se dois tipos de operações para cada tipo de transação. A operação *weak read* em um item de dado *x* (W_read[*x*]) lê cópias disponíveis de *x* no *cluster*. A operação *weak write* (W_write[*x*]) escreve em um cópia de dados local, não sendo permanente até que um *commit* global seja executado. Da mesma forma têm-se as operações S_read[*x*] e S_write[*x*]. Considera-se como *j* uma operação executada na

transação T*j*. A*j* e C*j* são operações de *abort* e *commit* da transação T*j* (PITOURA e BHARGAVA, 1995).

As transações *weak* possuem dois pontos de *commit*: um *commit* local e global. Um *commit* global acontece apenas quando é feita a sincronização dos dados com o banco de dados central. Denota-se como Cj[*i*] um *commit* para indicar que a transação T*j* está com um *commit* local no cluster Cl*i*. Os dados são considerados como permanentes após um *commit* global (PITOURA e BHARGAVA, 1995).

6.2.3 Aplicação do Modelo

Os modelos de transações baseados em *clusters* pode ser mais indicados para os sistemas onde a semântica dos dados em conjunto com a posição da unidade móvel seja relevante. Os Sistemas de Informações Geográficas – SIG usam estas duas informações ao mesmo tempo. Pode-se criar uma aplicação de informações turísticas sobre a cidade de forma que a unidade móvel receba seus dados em conformidade com sua posição na cidade. Além da posição a semântica dos dados irá facilitar na identificação do cluster onde as informações estão armazenadas.

Um exemplo para este sistema poderia ser dado por um usuário que estivesse a procura de um restaurante de comida francesa até 10.000 metros de sua posição. O SGBDM poderia fazer uma busca nos *clusters* de restaurantes e a possibilidade deste estar próximo seria alta, visto que o algoritmo de distribuição do cluster leva em consideração também a localidade dos dados.

6.3 MODELO MULTIDATABASE

Este modelo é proposto em (YEO e ZASLAVSKY, 1994A). O MDSTPM (*Multi Database System Transaction Process Model*) caracteriza-se por um *framework* para

computação móvel em um ambiente de vários bancos de dados cooperativos. Cada unidade móvel envia suas transações para um agente coordenador. O agente coordenador planeja e coordena a execução de acordo com o interesse de cada cliente móvel.

Devido às desconexões existentes no ambiente móvel, o MDSTPM implementa uma MQF (*Message* e *Queuing Facility*), responsável pelo gerenciamento das mensagens trocadas entre a unidade móvel e os demais SGBD.

Um MDSTPM é um integrante de um ambiente de totalmente distribuído, onde cada elemento participante é um sistema autônomo. Cada integrante é responsável pelo gerenciamento de suas transações locais. Para facilitar a execução de transações globais, implementa-se um outro nível de software que permite programar e coordenar as transações entre os SGBD participantes, conforme figura 32.

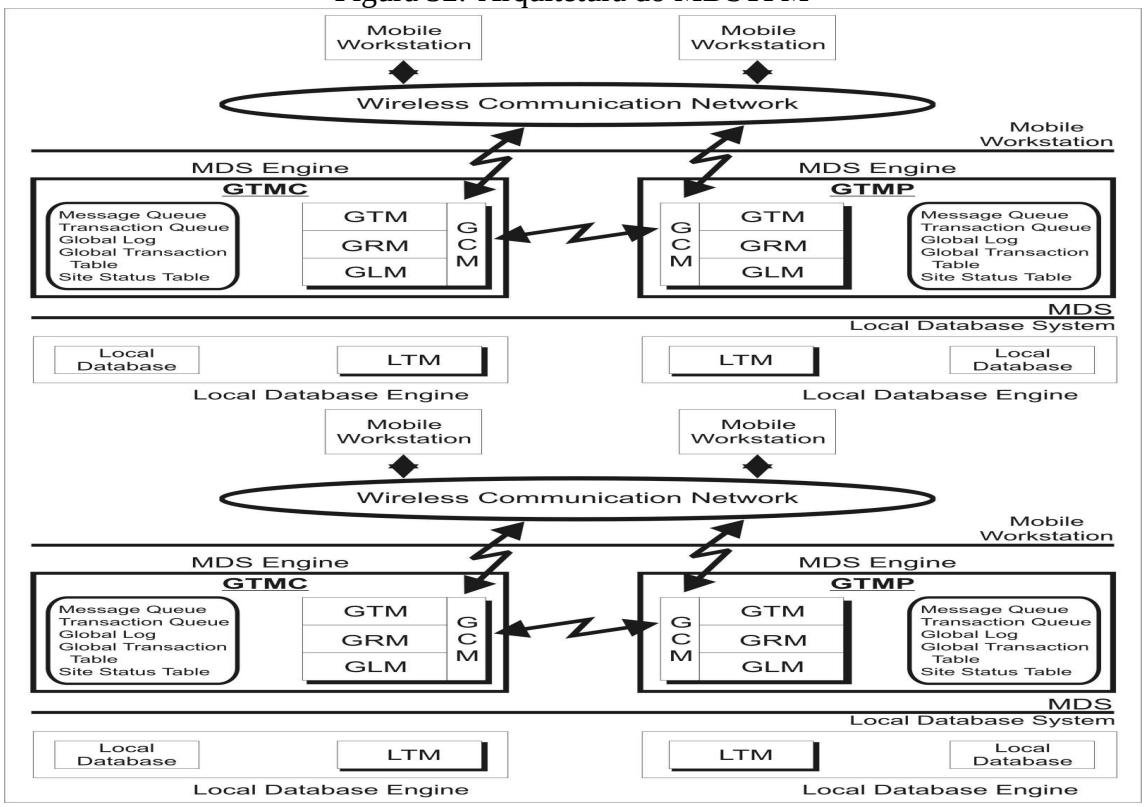
As unidades móveis podem desconectar-se da rede antes de suas transações tenham completado. O modelo MDSTPM implementa algumas estratégias que minimizam os problemas causados pela desconexão, como (SEYDIM, 1999):

- Fornece um completo *framework* de gerenciamento de transações em que os usuários e aplicativos possam acessar dados entre diversos sites transparentemente;
- Aumenta da disponibilidade dos dados e melhora a concorrência dos dados com a adoção de um controle de concorrência distribuído e um mecanismo de recuperação que preserva a autonomia local.
- Implementa o conceito de extensibilidade para suportar um sistema de vários bancos de dados em que os componentes podem cooperar com um SGDB relacional ou orientado a objetos;
- Provê um ambiente onde o processo de transação opera independentemente e transparentemente do SGBD local;
- Incorpora o conceito de computação móvel através do uso de unidades móveis dentro do modelo;

Utiliza-se o mecanismo de RPC em ambientes onde as chamadas podem ocorrer sincronamente. Pode-se fazer uma analogia entre RPC e uma chamada a uma sub-rotina onde o programa principal envia os parâmetros necessários. Como os eventos ocorrem sincronamente, o programa principal fica aguardando o retorno da chamada da sub-

rotina para continuar seu trabalho. Uma técnica alternativa é a implementação do MQF (*Message and Queuing Facility*). Uma unidade móvel envia uma mensagem de requisição, juntamente com as informações necessárias, para o pré-definido nó coordenador. As mensagens são então processadas assincronamente, habilitando a unidade móvel a se desconectar da rede, deixando o coordenador responsável pela coordenação da execução da transação global (YEO e ZASLAVSKY, 1994A).

Figura 32: Arquitetura do MDSTPM



Fonte: (YEO e ZASLAVSKY, 1994A, pp. 4).

6.3.1 Definições Formais

O modelo MDSTPM é composto pelos seguintes componentes (YEO e ZASLAVSKY, 1994A):

- Global Communication Manager (GCM): responsável pela geração e gerenciamento das filas de mensagens em um site local. Também é responsável pela comunicação, entrega e trocas de mensagens entre as

unidades participantes da rede;

- Global Transaction Manager (GTM): coordena o envio de subtransações globais os devidos sites;
- Global Transaction Manager Coordenator (GTMC): é quem coordena a transação global. O coordenador da transação global é o site em que a transação global iniciou-se;
- Global Transaction Manager Participant (GTMP): são todos os participantes da transação global;
- Global Scheduling Submanager (GSS): responsável pelo escalonamento das transações globais de das subtransações globais;
- Global Concurrency Submanager (GCS): responsável pelo controle da concorrência;
- Global Recovery Manager (GRM): coordena o *commit* e o *recovery* das transações e subtransações globais. O GRM garante que as alterações, efetuadas pelas transações global, nos diversos bancos de dados, foram escritas, no caso de um *commit*, ou abortadas em caso de uma falha;
- Global Interface Manager (GIM): coordena o envio e a resposta entre o MDSTPM e o SGBD local. Dentre suas funcionalidades destaca-se a tradução de uma requisição global em uma linguagem SQL do SGBD local.

No modelo MDSTPM existe o GTM. Este componente trata apenas de transações globais. Para as transações locais, cada SGBD possui seu próprio LTM. Por se tratar de um *framework*, o MDSTPM necessita de um SGBD autônomo, caracterizando assim um sistema de vários bancos de dados, podendo ainda ser bancos de dados heterogêneos.

6.3.2 Processamento das Transações

Para gerenciar as transações enviadas pela unidade móvel, além do conceito de filas de mensagens, propõe-se um mecanismo simples e eficiente: o mecanismo de

estado de máquina finita. Isto é possível visto que pode-se definir claramente um conjunto de estados possíveis, e suas transições de um estado para outro, durante *life span* (tempo de vida) de uma transação global. Entende-se como *life span* as primitivas BEGIN_GLOBAL_TRANSACTION e END_GLOBAL_TRANSACTION. Portanto a chave para implementar o modelo proposto está no projeto de um instrumento que mapeie exatamente todos os estados. Assim, geralmente têm-se cinco subfilas que serão utilizadas para gerenciar as transações e subtransações globais (YEO e ZASLAVSKY, 1994A):

- Input Queue (fila de entrada): esta fila contém todas as transações/subtransações que chegaram até o nó coordenadoras, ordenadas pela hora de chegada;
- Allocate Queue (fila determinada): as transações/subtransações globais serão selecionadas para a execução baseada no algoritmo FIFO (*first-in-first-out* – primeiro a entrar/primeiro a sair). Bloqueios necessários para a execução da transação/subtransação serão feitos durante esta estágio;
- Active Queue (fila ativa): esta fila contém todas as transações/subtransações que estão ativas;
- Suspend Queue (fila suspensa): esta fila contém todas as transações/subtransações que completaram a primeira fase do protocolo 2PC (*two phase commit*);
- Output Queue (fila de saída): Nesta fila encontra-se todas as transações completadas.

Quando uma transação global é iniciada por uma unidade móvel, o GCM irá colocá-la dentro da fila *input*. Periodicamente o GSS irá procurar por uma transação global na fila *input* para execução. Uma vez selecionada, a transação global será enviada para a fila *allocate* onde serão feitos os devidos bloqueios pelo GCS. A transação então será transferida para a fila *active* onde será executada e enviada para os demais sites através do GCM. Uma vez completa a transação, tem-se a primeira fase do protocolo 2PC e esta transação será colocada na fila *suspend*. Ficará nesta fila até completar o protocolo 2PC completamente onde então será colocado na fila output (YEO e ZASLAVSKY, 1994A).

Para garantir a consistência dos dados utilizar-se um mecanismo de isolamento,

como a serialização. Um dos grandes problemas para manter a serializibidade de uma transação global está em preservar a ordem de execução pelo LTM. O método de *ticketing*, encontrado em (YEO e ZASLAVSKY, 1994B), é usado para resolver o problema. Todas as subtransações globais são obrigadas a obter um ticket primeiro, causando um conflito adicional entre eles e consequentemente preservando a ordem correta de execução (YEO e ZASLAVSKY, 1994A).

6.3.3 Aplicação do Modelo

O MDSTPM não pode ser classificado como um SGBDM. Trata-se de um *framework* para gerenciamento de transações distribuídas com alguma extensão para as transações móveis. Assim, este modelo pode ser utilizado para qualquer aplicação usuária de uma arquitetura totalmente distribuída, ou seja, tem-se um SGBD autônomo em todas as unidade participantes, fixas ou móveis. Cada transação global ou distribuída, deverá encontrar o coordenador, que efetuará todas configurações do ambiente para a execução. Este modelo é mais indicado para redes do tipo *ad-hoc*, onde todos os participantes são totalmente autônomos.

Na exploração de petróleo em automar, os navios petroleiros têm autonomia total em seu sistema, porém necessita de outras informações para o trabalho cooperativo. Cada navio necessita obter informações de seus vizinhos para determinar o momento de coleta do petróleo em cada plataforma. Assim, uma transação de extração de petróleo nas plataformas pode ser coordenada por um, porém deverá ter a participação do todos.

6.4 MODELO PRO-MOTION

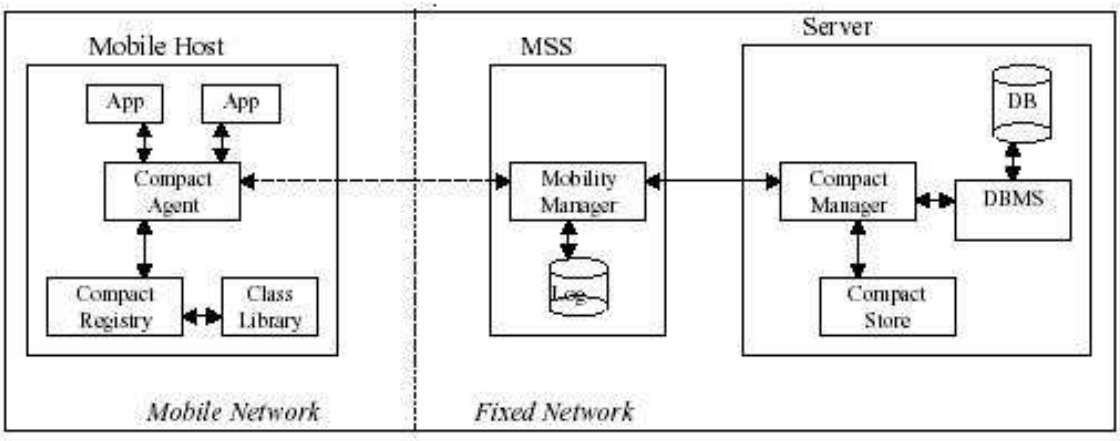
PRO-MOTION é um sistema de processamento de transações que suporta o modo desconectado em um ambiente cliente/servidor móvel. Neste modelo faz-se necessário

à migração e/ou adaptação das aplicações existentes para suportar as aplicações envolvendo uma unidade móvel e o acesso aos dados no ambiente em fio.

Este modelo utiliza o conceito de transações aninhadas abertas, onde partes dos resultados podem ser vistas por outras transações. Ele introduz os *compacts*, ver figura 33, para garantir a execução de transações locais na unidade móvel. Todas as informações necessárias para gerenciar os *compacts* estão armazenadas dentro dele. Para aumentar a autonomia e melhorar a concorrência, objetos semânticos são usados para a construção dos *compacts*. Os *compacts* são unidades básicas de cache controle (SEYDIM, 1999) (SERRANO et all, 2000).

Um *compact* é definido para satisfazer os requisitos de dados, tendo a forma de um objeto. Contém os dados, restrições, métodos de acesso aos dados e informações sobre o estado, conforme figura 35. Representa um "acordo" entre o servidor de dados da rede fixa e a unidade móvel, onde o SGBD da rede fixa delega alguns controles dos dados para a unidade móvel, permitindo as transações locais.

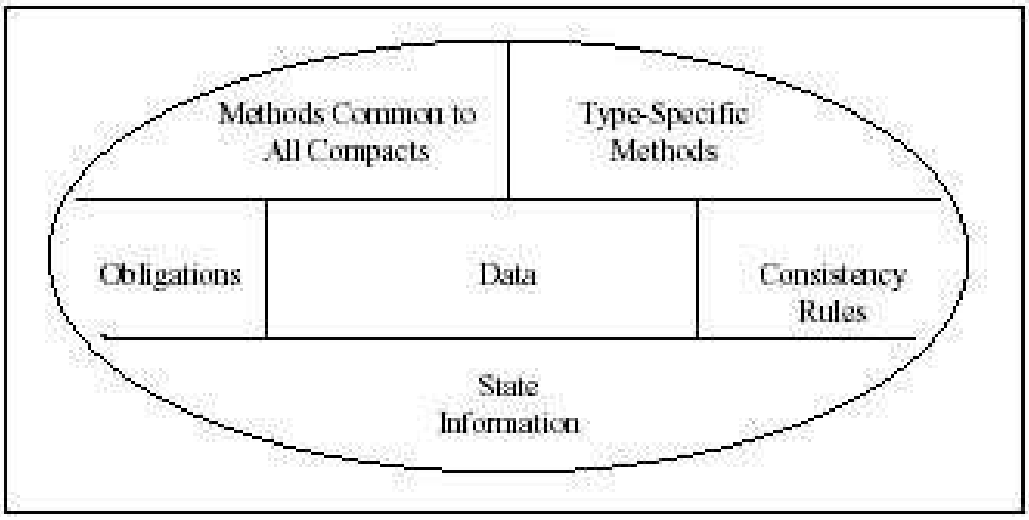
Figura 33: Arquitetura do modelo PRO-MOTION



Fonte: (WALBORN e CRYSANTHIS, 1999, pp. 4)

O gerenciamento dos *compacts* é realizado pelo gerenciador de *compacts* no SGBD, localizado na rede fixa. Os *compacts* são obtidos do SGBD através de uma demanda de dados da unidade móvel. O SGBD cria os *compacts*, através do gerenciador de *compacts* e os armazena no armazém de *compacts* para serem transmitidas às unidades móveis.

Figura 34: Compacts do modelo PRO-MOTION



Fonte: WALBORN e CRYSANTHIS, 1999, pp 4.

Todos os *compacts* possuem interfaces comuns, usadas pelo agente localizado na unidade móvel para gerenciar a lista de *compacts* locais. A implementação de uma interface comum simplifica o projeto dos *compacts* e dos agentes, garantindo as funcionalidades mínimas para uma instância específica. Entretanto, os *compacts* podem ter métodos especializados para suportar tipos especializados de dados ou um método específico para seus dados (SEYDIM, 1999).

6.4.1 Definições Formais

O modelo de transação PRO-MOTION faz uso dos seguintes componentes (WALBORN e CRYSANTHIS, 1999):

- *Compacts*: Objetos que encapsulam os dados, métodos e estados. Também representam um "acordo" entre a unidade móvel e o SGBD da rede fixa, onde o SGBD delega algumas funções sobre os dados para a unidade móvel;
- *Mobile Compact Agent*: Um agente, localizado na unidade móvel, responsável pela troca de *compacts* entre a unidade móvel e a rede fixa;
- *Mobility Manager*: Um Agente, localizado na rede fixa, responsável pelo

envio dos *compacts*, recebidos pelo *compact manager*, para o *Mobile Compact Agent*;

- *Compact manager*: Servidor de *compacts*, responsável pela encapsulamento dos *compacts* que serão armazenados no *compact store* e enviados para a unidade móvel;
- *Compact Store*: Armazém onde são armazenados todos os *compacts*, criados pelo *compact manager*;
- *Compact Registry*: Responsável por manter a lista de todos os *compacts* recebidos do *compact manager*.

O *compact agent* utiliza os métodos comuns a todos os *compacts*, listados no *compact registry*, para gerenciá-los e executar as operações necessárias para as atualizações necessárias requisitadas pelas aplicações que estão sendo executadas na unidade móvel. O conjunto básico destas operações é (WALBORN e CRYSANTHIS, 1999):

- Inquire(): recupera as informações úteis sobre o estado do de um *compact*, como o nome, tipo de dados, versão, status do cache, Id da transação e informações sobre armazenamento;
- Notify(): usado para notificar o *compact* de que uma mudança ocorreu no ambiente móvel;
- Dispatch(): usado para executar operações sobre o *compact* referente a transação que está sendo executada na unidade móvel;
- Commit(): marca uma transação como permanente no SGBD;
- Abort(): abandona as trocas feitas no *compact* pela transação;

Além das operações básicas, listadas acima, o *compact agent* também fica responsável pelo controle da concorrência, histórico de transações e recuperação.

6.4.2 Processamento das Transações

Toda a interação entre o *compact manager*, o *compact agent* e a conexão de rede, sugerem quatro atividades para o processamento de transações (WALBORN e

CRYSANTHIS, 1999):

- *Hoarding*: a unidade móvel está conectada à rede e o *compact manager* está armazenando *compacts*, preparando-se para uma eventual desconexão;
- *Connected processing*: a unidade móvel está conectada à rede e o *compact manager* está processando transações;
- *Disconnected processing*: a unidade móvel está desconectada da rede e o *compact agent* está processando as transações localmente;
- *Resynchronization*: a unidade móvel *reconectou* à rede e o *compact agent* está sincronizando as atualizações comprometidas, durante a desconexão com o SGBDF fixo.

Como utiliza transações aninhadas aberta em sua infra-estrutura, considera-se que as transações possuem um longo tempo de vida. Os recursos necessários para criar um *compac*t são obtidos através de transações usuais do SGBD localizado na rede fixa. O *compact manager* será apenas um cliente para o SGBD que estará executando uma transação com longo tempo de duração. Todos os bloqueios necessários aos dados, localizados no SGBD da rede fixa, são realizados pelo próprio SGBD fixo através de seu sistema de processamento de transação.

O *compact agent*, sendo responsável pelas atividades locais à unidade móvel, faz todo o gerenciamento da localização e das transações locais. Os bloqueios, assim como a recuperação e o histórico, necessário pelas atividades dos aplicativos locais também são gerenciados pelo *compact agent*. Um *commit* local torna visíveis os dados para outras aplicações da unidade móvel. Quando todas as aplicações residem na unidade móvel, o protocolo 2PC é utilizado para o *commit*. Uma vez estando on-line com o *compact manager*, o *compact agent* fará a sincronização dos dados e estando o *compact* em um estado globalmente válido será efetuado um *commit* global.

6.4.3 Aplicação do Modelo

O modelo PRO-MOTION pode ser aplicado em qualquer tipo de aplicação. Sua arquitetura é muito flexível e pode ser utilizada por qualquer SGBD. Porém sua

utilização em sistema com grandes números de registros pode ocasionar problemas de latência na rede, visto que o mapeamento de cada dado é feito em um único objeto.

A arquitetura proposta pelo modelo PRO-MOTION visa uma garantia da consistência dos dados. Assim, um sistema de compra e venda de ações em bolsas de valores seria poderia Ter melhores resultados com a adoção deste modelo. Os usuários poderia fazer seus lances em tempo real com seus PDAs. Todas as transações de compra e venda de ações poderiam ser registradas na unidade móvel de cada corretor. Em tempo real estas transações estaria sendo registradas pelo site central da bolsa de valores. Neste caso os corretores teriam maiores produtividade, visto que não seria mais necessário fazer os relatórios de final de pregão, estes seriam emitidos pelo próprio sistema.

6.5 OUTROS MODELOS

Neste capítulo mostraram-se os principais modelos de processamento de transações. Outros modelos também podem ser encontrados, uns mais antigos e outros ainda mais recentes. Porém, percebe-se que os modelos citados são mais discutidos e difundidos no meio científico, merecendo maior atenção. Entre os demais modelos de processamento de transações pode-se citar: o *reporting*, o *two-tier replication*, *semantic-based* e o *prewrite*.

O *reporting* considera um SGBDM como sendo um sistema especial de vários bancos de dados, onde transações na unidade móvel são consideradas como sendo um conjunto de subtransações. Este modelo também trabalha com uma unidade de suporte à mobilidade para o gerenciamento das atividades da unidade móvel (CHRYSANTHIS, 1993) (MOLINA e SALEM, 1987) (MOSS, 1981 appud) (PU et al., 1988 appud) (RAMAMRITHAN e CHRYSANTHIS, 1996) (ALVARADO et al., 2001) (SEYDIM, 1999).

O modelo *two-tier replication* considera a transação e a replicação como técnicas para o ambiente de computação móvel onde a unidade móvel às vezes pode estar conectada. Uma versão primária dos dados existe para cada cópia replicada para a unidade móvel. Dois tipos de transações são suportados: *base* e *tentative*

transactions.Uma transação base acessa diretamente a cópia principal dos dados e a *tentaive transactions* acessa a cópia replicada dos dados. Uma vez desconectado a unidade móvel executa uma *tentative transactions*. Ao conectar-se com a rede, as transações do tipo *tentative transactions* são transformadas em transações base (GRAY et all, 1996 appud) (ALVARADO et al, 2001).

Semantic-based faz uso de objetos semânticos para garantir a autonomia da unidade móvel em modo desconectado. Utiliza a fragmentação de objetos como solução para operações concorrentes e para a limitação de armazenamento da unidade móvel. Os pequenos fragmentos são do mesmo tipo e servem para aplicações onde existe a necessidade de dados semânticos. Cada fragmento localizado na unidade móvel é manipulado assincronamente e independentemente. Os fragmentos de objetos são agregações de itens de dados, conjuntos, pilhas ou filhas (CHRYSANTHIS, 1995) (ALVARADO et al., 2001) (SEYDIM, 1999).

O modelo de transação *prewrite* introduz a uma operação de pré-escrita antes da operação de escrita no SGBD. Ele utiliza uma estação de suporte à mobilidade para intermediar as requisições de dados entre a unidade móvel e o SGBD da rede fixa. Utiliza modelo de transações aninhadas abertas, onde seus resultados podem ser vistos por outras transações. A pré-escrita garante uma pré-confirmação das atualizações (*precommit*), executado antes de um *commit* da transação móvel. Atualizações permanentes são executadas posteriormente por uma operação de escrita (*write*), portanto dois tipos de dados são permitidos: os dados *prewrite* e dados *write* (MADRIA e BHARGAVA, 2001) (ALVARADO et al., 2001) (SEYDIM, 1999).

7 CLASSIFICAÇÃO DOS MODELOS DE TRANSAÇÕES MÓVEIS

Os modelos de transações estudados neste trabalho mostram algumas estratégias para garantir a execução de uma transação no ambiente móvel. Vários modelos de transações móveis flexibilizam algumas propriedades para garantir que a transação seja executada, chegando até mesmo a não atender parte delas.

Quanto ao modelo de execução, são apresentados diversas modelos e formas. Modelos de transações para SGBD fixos são estendidos para a utilização nos SGBDM. Estes modelos ainda se utilizam alguns protocolos e procedimentos padrões, caso dos protocolos 2PC e 2PL.

O presente capítulo estuda os diversos modelos, fazendo uma classificação de suas arquiteturas, modelos de execução e suporte das propriedades ACID. Apresenta ainda considerações quanto ao modelo de transação móvel e propõe mudanças no modelo PRO-MOTION para aumentar sua eficiência.

7.1 CLASSIFICAÇÃO EM RELAÇÃO AO MODELO DE EXECUÇÃO

Como já estudado, a arquitetura de implementação do SGBDM influi diretamente nos modelos de execução e gerenciamento de transações. Para uma arquitetura totalmente distribuída e simétrica, modelo ideal para redes *ad-hoc*, todos os controles da execução da transação deve ficar por conta de cada SGBDM participante. Nos modelos de computadores pessoais de hoje este tipo de modelo não é indicado, visto que possuem baixo poder de processamento, pouca memória e pouco tempo de autonomia de bateria, como visto nas limitações estudadas no capítulo 2. Já em um modelo cliente/servidor estendido, caso de muitas implementações dos SGBDM comerciais existentes, a infra-estrutura de rede é um elemento essencial para seu funcionamento.

Os modelos apresentados e estudados propõem soluções para os ambiente com uma infra-estrutura existente. Partem do princípio que as unidades móveis estarão, em algum momento, conectadas à rede fixa. Desta maneira todos os modelos podem ser classificados como integrantes da arquitetura Cliente/Servidor estendido, proposto por (PITOURA e SAMARAS, 1998).

Neste ambiente a adoção de agentes facilita a troca de mensagens e dados entre os SGBD participantes, além de garantir, com pouco esforço, a integração dos SGBD existente no mercado. Muitas soluções utilizam uma abordagem de encapsulamento dos dados em forma objetos para facilitar a distribuição dos dados e das regras de integridade.

O tipo de transações utilizado nos modelos estudados utiliza os mesmos tipos de transações dos SGBD tradicionais fixos. Uma característica relevante nestes modelos é a forma de requisição e da execução da transação. Com exceção do modelo Reporting, a iniciativa de uma nova transação parte da unidade móvel. Todos os modelos apresentam dois tipos de transações diferentes: uma para o modo desconectado e outra para o modo conectado. No modo conectado o SGBDM é mais um SGBD participante de um ambiente distribuído, utilizando-se das tradicionais técnicas dos SGBDD. Já no modo desconectado, a grande preocupação está na sincronização dos dados atualizados com suas respectivas cópias primárias no SGBD da rede fixa.

O modelo de transações móveis Kangaroo possui um modelo particular de execução de transações móveis. Suas transações são separadas em várias sub-transações. Cada sub-transação é executada em uma estação de suporte à mobilidade, assim, caso a unidade móvel se desloque para uma outra estação de suporte à mobilidade, acontece uma separação (*split*) da subtransação, iniciando uma nova sub--transação. Para cada inicio e fim de uma nova sub-transação existe registros em tabelas específicas, armazenando dados como local, data, hora de início e fim da transação, ID da transação, etc. Desta maneira o sistema pode facilmente capturar a mobilidade da execução de uma transação.

Nos demais modelos também existem um conjunto de registros de atividades das execuções, porém não tão eficiente a ponto de capturar a mobilidade da execução da transação. Dentre estes modelos, pode-se destacar o modelo PRO-MOTION devido à utilização de um agente fixo, chamado gerenciador de *compact* (*compact manager*),

responsável pelo encapsulamento dos dados em *compacts*. Este modelo garante maior consistência nos dados e nas regras de integridade, porém aumenta muito o trabalho computacional necessário para tal tarefa, visto que os *compacts* carregam consigo, além dos dados, os códigos necessários para garantir a integridade dos dados e alguns métodos de controle do sistema.

Para cada registro armazenado no SGBD da rede fixa, o gerenciador de *compacts* gera um *compact*, controlando, assim, efetivamente a concorrência dos dados e seu estado. Após a geração dos *compacts* o gerenciador de *compact*, ou agente fixo, envia os dados para o agente localizado na unidade móvel, que por sua vez será responsável pela persistência local.

O quadro 4, abaixo, apresenta um resumo dos principais modelos de execução de transações móveis.

Tabela 4: Resumo dos modelos de execução de transações móveis

Modelo	Tipo da Transação	Início	Execução na Unidade Móvel	Execução na rede móvel (modo conectado)
Clustering	Transações *Strict* e *weak*	Unidade móvel	Transações *weak* e *commit* local no modo desconectado. Participação em uma transação *strict* no modo conectado	Transações *Strict* and *commit* de uma transação *weak* (sincronismo e atualizações permanentes)
Two-tier Replication	Transações *Base* e *tentative*	Unidade móvel	Transações do tipo *tentative* no modo desconectado e participação de uma transação base no modo conectado	Transações tipo base
Promotion	Transações long-lived nested-split	Unidade móvel	O agente de *compacts* executa inteiramente a transação móvel e faz um *commit* local	O gerenciador de *compacts* fica a cargo da construção de *compacts*, executa um *commit* das transações locais (sincronismo e atualizações permanentes)
Reporting	Transações *open-nested* com atomicidade, não compensáveis e	Unidade móvel ou *host* fixo	Subtransações fazem parte de transações globais	Transações globais e subtransações

	co-*transactions*			
Semantics-based	Transações de longo tempo de duração	Unidade móvel	Transações móveis e local *commits*	Em resposta a à requisição da unidade móvel, fragmentação de objetos é feita no SGBD como também o reagrupamento
Prewrite	Transações de longo tempo de duração (*nested*, *split*)	Unidade Móvel	Transações móveis e local *commits*	Gerenciamento de bloqueios e *commit* de uma transação local
KT	Transações open-nested e split	Unidade Móvel		Coordena a execução de uma transação
MDSTPM	Multi-transações e transações locais	Unidade Móvel	Transações locais	Coordena a execução de multi-transações

7.2 CLASSIFICAÇÃO EM RELAÇÃO ÀS PROPRIEDADES ACID

O ambiente móvel é dinâmico e suas características podem variar constantemente. Devido às suas particularidades, como visto no capítulo 2, forçar as mesmas regras, para garantir a observância nas propriedades ACID, existentes nos SGBD fixos, nos SGBDM implica em uma estrutura rígida que não funcionaria neste ambiente. A garantia das propriedades ACID, tradicionalmente conhecidas, em uma transação móvel implica em uma estrutura rígida que o ambiente móvel não suporta. Assim, faz-se necessário a flexibilização de regras e implementações para que o SGBDM seja flexível e adequado ao ambiente móvel.

7.2.1 Classificação em Relação à Atomicidade

A atomicidade é uma propriedade que é garantida em todos os modelos. Utilizam-se de *commits* locais para as transações locais e *commits* globais para as transações globais. Algoritmos como o 2PC (*two phase commit*) para uma transação global também são utilizados. Além do 2PC outros algoritmos são propostos para garantir que uma transação global seja segura e atômica. Destacam-se dentro todos os modelos o modelo *Semantics-based* (baseado em semântica), visto que ele define um modelo bem particular. É proposto um modelo de distribuição de dados onde sempre existirá somente uma cópia de dados, assim na sincronização dos dados não acontecerá um conflito, garantindo a atomicidade por completo. Nos demais modelos, podem acontecer conflitos entre os dados já efetivos localmente com suas respectivas réplicas no servidor da rede fixa no momento da sincronização dos dados. Estando o banco de dados em um estado inconsistente os dados já comprometidos (*committed*) localmente serão cancelados para resolver os conflitos.

Tabela 5: Resumo das propostas em relação a Atomicidade

Proposta	Primeiro passo na Unidade móvel	Segundo passo no SGBD
Clusterig	*Commit* local no modo desconectado 2PC para o modo conectado	*Commit* envolve uma reconciliação sintática e *rollback* para solucionar os conflitos
Two-tier replication	*Commit* local para transações do tipo *tentative* e protocolo atômico para transações do tipo base	As transações do tipo tentative são re-executadas dentro dos critérios de aceitação
Promotion	*Commit* local para todas as transações locais	Um processo de sincronização verifica o *compact* envolvido na transação local. Caso exista conflito as transações locais são canceladas e rotinas de contingenciamento serão executadas
Reporting	Todas as subtransações são atômicas e pode ser comprometida (*commit*) independentemente de uma transação pai. Exceto para os caso de transações do tipo não compensáveis que podem estar associadas a sub-transações	
Semantics-based	Commit local	Reagrupamento (*merge*) das atualizações. Como os fragmentos são cópias exclusivas eles possuem consigo as condições de consistência não haverá conflitos no reagrupamento

Prewrite	*Commit* local para todas as transações locais	Atualizações locais são feitas como permanentes através de uma operação de escrita

7.2.2 Classificação em Relação à Consistência

Conforme observa-se neste trabalho, os dados são replicados do SGBD da rede fixa para o SGBDM. Várias abordagens foram propostas para esta replicação, como a divisão dos dados em clusters, a distribuição semântica dos dados e o encapsulamento dos dados em objetos. Na divisão dos dados em clusters ou na distribuição semântica os dados são replicados, porém as regras de integridade não estão associadas aos dados. O controle da integridade implica em trocas de mensagens com o SGBD da rede fixa, aumentado o tráfego de informações no canal de rede, além de obrigar a unidade móvel estar conectada à rede fixa. Assim, uma vez desconectado o SGBDM não pode garantir que uma transação está integra. As regras de integridade serão verificadas assim que os dados forem sincronizados com os dados existentes no SGBD da rede fixa.

Já na abordagem de distribuição de dados encapsulada em objetos, as regras de integridade podem ser implementadas como métodos dos objetos, garantindo assim que a integridade dos dados será mantida mesmo a unidade móvel estando desconectada. O modelo de transação PRO-MOTION implementa estas características, deixando por conta do gerenciador de *compacts* a tarefa do encapsulamentos dos dados e das regras de integridade em objetos para sua posterior distribuição. Esta técnica aumenta a eficiência das regras para garantir a consistência dos dados, porém apresenta uma sobrecarga de processamento para o encapsulamento destas regras como métodos de objeto (*compact*), que estará sendo disponibilizado para o SGBDM.

O quadro 6 apresenta um resumo das técnicas usadas para garantir a integridade dos dados.

Tabela 6: Resumo das propostas em relação a Consistência

Proposta	Conceito envolvido	Uso de informações semânticas
Clusterig	Duas versões de dados: *strict* (uma cópia serializável) e *weak* (graus de inconsistência, dados atualizados no modo desconectado)	Definição de uma função *h* e graus de inconsistência
Two-tier replication	Duas versões de dados: Mestre (uma cópia serializável) e *tentative* (dados atualizados no modo desconectado)	Critérios de aceitação
Promotion	Os *compacts* possuem tipos específicos de dados, regras de consistência e obrigações	Construção de *compacts* e rotinas para contingências
Reporting	Técnicas de Multi-transações	Delegação e transações compensáveis
Semantics-based	Fragmentação de objetos (condições de consistência e operações de separação/junção)	Fragmentação
Prewrite	A serialização é baseada em uma ordem de *commit* local para uma transação móvel	definição de dados variantes (*prewrite* e *write*)

7.2.3 Classificação em Relação ao Isolamento

Sendo o ambiente móvel inconstante e a perda de conexão é um fato, o controle tradicional de concorrência poderia tornar o sistema inviável. Muitos modelos, conforme resumo apresentado no quadro 7, simplesmente não tratam esta propriedade, podendo deixar o SGBD em um estado inconsistente. Outros modelos implementam o controle da concorrência nos agentes localizados na rede fixa, caso do modelo PRO-MOTION ou utilizam protocolos conhecidos como o 2PL. No modelo PRO-MOTION o gerenciador de *compacts* determina o critério correto para controle da concorrência de cada *compact*. O modelo apresenta uma escala de dez níveis de isolamento, como previsto no padrão ANSI SQL variando do grau zero até o grau nove (GRAY et al, 1995). O Grau zero não garante isolamento algum, já o grau nove representa uma execução serializada da transação.

Tabela 7: Resumo das propostas em relação ao Isolamento

Proposta	Visibilidade	Protocolo de controle de concorrência
Clusterig	Os resultados de transações locais são visíveis localmente na unidade móvel	2PL, 4 tabelas de conflitos e um novo tipo de bloqueio são propostos
Two-tier replication	Os resultados de transações locais são visíveis localmente na unidade móvel	Mecanisnos de bloqueio
Promotion	Os resultados de transações locais são visíveis localmente na unidade móvel	2PL
Reporting	A sub-transações atômicas, *reporting* e *co-transactions* são visíveis antes de um *commit* da transação global	
Semantics-based	Os resultados de transações locais são visíveis localmente na unidade móvel	2PL para controlar o acesso aos fragmentos locais
Prewrite	Os resultados de transações locais são visíveis para todas as unidades	2PL extendido, uma tabela de conflito e um novo tipo de bloqueio é proposto

7.2.4 Classificação em Relação à Durabilidade

Outro fator importante no gerenciamento de transações é a durabilidade de uma transação. A transação deve ser durável garantindo assim que os dados permaneçam no SGBD. Para os dados locais, todos os modelos apresentam um *commit* local, garantindo a durabilidade. Já para as transações globais, a durabilidade somente poderá ser garantida após sincronização entre o SGBDM da unidade móvel e o SGBD da rede fixa. Na sincronização dos dados podem ocorrer conflitos entre os dados modificados e as regras existentes no SGBD fixo, podendo ocorrer um *rollback* das transações já comprometidas localmente.

O quadro 8 apresenta um resumo dos principais modelos e sua garantia da durabilidade.

Tabela 8: Resumo das propostas em relação a Durabilidade

Proposta	Garantia	Desvantagem
Clusterig	Sim, após o *commit* (re-sincronização)	Transações locais podem ser desfeitas durante o processo de reconciliação dos conflitos
Two-tier replication	Sim, após o *commit* (re-execução)	Transações locais podem ser desfeitas durante o processo de reconciliação dos conflitos
Promotion	Sim, após o *commit* (re-sincronização)	Transações locais podem ser desfeitas durante o processo de reconciliação dos conflitos, porém muitas mensagens são necessárias entre a unidade móvel e o gerenciador de *compacts*
Reporting	Sim, se a transação pai faz o *commit* então as sub-transações são duráveis	Transações locais podem ser desfeitas durante o processo de reconciliação dos conflitos
Semantics-based	Sim, após um *commit* local	Redução dos fragmentos disponíveis no SGBD
Prewrite	Sim, após um *commit* local	Muitas mensagens e trocas entre a unidade móvel e a estação base

7.2.5 Resumo da Classificação

O quadro 9 a seguir apresenta um resumo dos principais modelos de transações com relação às propriedades ACID.

Tabela 9: Resumo da Classificação em relação às propriedades ACID

Proposta	Atomicidade	Consistência	Isolamento	Durabilidade	Executa em
Kangaroo	Sim	Não	Não	Não	RF
Clustering	Não	Não	Não	Não	UM ou RF
MDSTPM	Não	Não	Não	Não	UM ou RF
Semantics-based	Sim	Sim	Sim	Sim	UM ou RF
Promotion	Sim	Sim	Sim	Sim	UM ou RF
Toggle	Sim	Sim	Sim	Sim	UM ou RF
Prewrite	Sim	Sim	Sim	Sim	UM ou RF
Reporting	Sim	Sim	Sim	Sim	UM ou RF
Two-tuer replication	Sim	Sim	Sim	Sim	Um ou RF

8 PROPOSTA SUGERIDA AO MODELO PRO-MOTION

8.1 INTRODUÇÃO

Dentre todos os modelos estudados o modelo PRO-MOTION apresenta melhores resultados para garantir que uma transação seja atômica, consistente, isolada e durável. Porém esta garantia implica maior uso de processamento e recursos de máquina e do link de transmissão de dados.

PRO-MOTION utiliza uma arquitetura cliente/servidor estendida, onde tem-se um SGBD na rede fixa, um agente fixo (gerenciador de *compacts*), um agente localizado na unidade móvel (*compact agent*) e um SGBDM. Desta maneira este modelo, como todos os demais estudados, operaram em uma rede com infra-estrutura, onde a unidade móvel é participante de uma rede de computadores ligados à rede fixa. Porém, ao contrário do modelo Kangaroo, onde para se iniciar uma transação é necessário estar conectado à rede fixa, o modelo PRO-MOTION pode iniciar uma transação mesmo estando desconectado da rede fixa, bastando apenas ter os dados em seu *cache* local, ou mesmo incluindo novos dados.

Na rede fixa o aumento de uso de recursos não é problema, visto que as máquinas existentes possuem grande capacidade de processamento, memória e armazenamento, além dos *links* de rede possuem altas taxas de vazão. De outro lado temos pequenos dispositivos móveis que possuem pouca memória, baixo poder de processamento e utiliza-se de um link com baixa taxa de vazão.

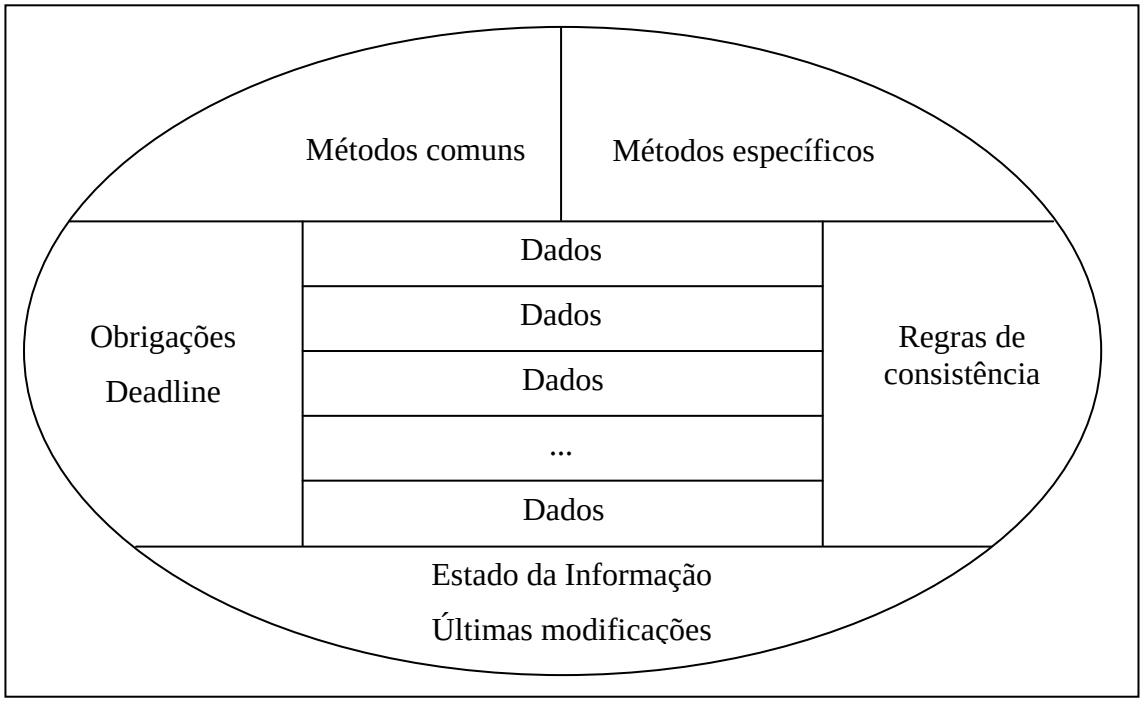
O agrupamento dos dados em *compacts* (objetos), conforme visto no capítulo anterior, gerenciados por um agente (gerenciador de *compacts*) resolve muito problemas com relação às regras de consistência e garantia de um estado consistente do banco de dados, porém, representa um aumento das mensagens e dos tamanhos dos pacotes envolvidos na transmissão entre o SGDB fixo e o SGBDM, além de necessitar mais recursos para armazenamento na unidade móvel.

8.2 PROPOSTA

Uma solução possível visando minimizar o uso da banda de rede, de processamento e de memória seria a modificação da estrutura do *compact* (objeto), agregando os dados, pertencentes ao mesmo grupo, em um único *compact*. Assim, podem-se compartilhar os códigos específicos dos *compacts* de mesmo grupo em uma única estrutura, diminuindo o tamanho dos *compacts* e consequentemente o tamanho dos enviados para a rede e o uso de recursos da unidade móvel.

Tanto os métodos comuns como os métodos específicos para cada grupo de dados, o grupo de dados, regras de consistência, obrigações (*deadline*) e informações de estado seriam encapsulados normalmente no *compact*. Desta maneira pode-se fazer uma nova representação do *compact* conforme a figura 35.

Figura 35: Proposta para o novo modelo de *compact*



8.3 EFICIÊNCIA

Neste novo modelo a eficiência da transação, como um todo, poderia ser aumentada. A distribuição dos dados iria requer menores recursos de rede e da unidade móvel. O número de mensagens na rede poder ser menor, pois com menos *compacts* distribuídos o processo de sincronização dos *compacts* será menor. Já na unidade móvel será necessária uma quantidade menor de memória, pois um número menor de *compacts* será usado para transportar os dados do SGBD fixo até a unidade móvel. Cada *compact* será maior em seu tamanho, mas no escopo geral serão necessário menos *compacts* para a mesma quantidade de dados. Para calcular a quantidade provável de bytes que serão transmitidos pela rede, basta quantificarmos o tamanho de cada *compact* e multiplicarmos pelo número de *compacts* necessário para os dados selecionados pelo usuário. Desta maneira pode-se usar a seguinte fórmula:

TTB = (TTD + TME + TMC + TOI) * NC
Onde:
TTB = Tamanho Total em Bytes;
TTD = Tamanho Total dos Dados;
TME = Tamanho dos Métodos Específicos;
TMC = Tamanho dos Métodos Comuns;
TOI = Tamanho de outras Informações;
NC = Número de *Compacts*

Para exemplificar a utilização da fórmula acima, toma-se como exemplo um vendedor que necessite dos dados relativos aos seus clientes em uma determinada região e em um determinado momento. Convencionando que o tamanho de cada informação de vendas do cliente possui cerca de 500 bytes, o tamanho médio para os métodos específicos de cada *compact* seja 5Kbytes (5120 bytes) , os métodos comuns de cada *compact* algo em torno de 5Kbytes (5120 bytes), outras informações como o estado, regras de consistência e obrigações com tamanho de 1Kbyte (1024 bytes)e que o

usuário deseja recuperar as 20 últimas vendas para seu cliente, assim tem-se:

Tabela 10: Total de Bytes utilizando o modelo de *compact* original

TTD	500 bytes
TME	5120 bytes
TMC	5120 bytes
TOI	1024 bytes
NC	20 *compacts*
TTB	(500 + 5120 + 5120 + 1024) * 20 = **235.280** bytes

No modelo original, o tamanho teórico2 total de bytes transmitidos para a unidade móvel ficaria em torno de 235.280 bytes, ou seja, 229,76 *Kbytes*. Já utilizando o novo modelo proposto tem-se:

Tabela 11: Total de Bytes utilizando o novo modelo de *compact*

TTD	500 * 20 = 10000 bytes
TME	5120 bytes
TMC	0 bytes
TOI	1024 bytes
NC	01 *compact*
TTB	(10000 + 5120 + 1024) * 1 = **16.144** bytes

Neste novo modelo, o tamanho teórico total de bytes transmitidos para a unidade móvel ficaria em torno de 16.144 bytes, ou seja, 15,76 Kbytes. No cenário apresentado acima, o modelo proposto diminuiria em 219.136 Bytes, ou 214 Kbytes.

Este novo modelo de *compact* poderia aumentar a eficiência na distribuição dos dados em 93,20%, visto que o novo modelo de *compact* teria apenas 6,80% to tamanho do *compact* original, consequentemente uma economia significativa dos recursos utilizados pelo SGBDM para o gerenciamento dos *compacts*. Porém, exigiria um aumento de processamento no gerenciador de *compacts* para garantir o isolamento da transação, visto que para cada *compact* ter-se-ia vários registros de um mesmo grupo. Este aumento de utilização de recurso poderá não apresentar problemas, visto que o gerenciador de *compacts* está localizado na rede fixa, portanto poderão ser utilizados equipamentos com maiores capacidades, isto é, sem as limitações existentes nos

2 Teórico, pois ainda são necessárias as informações adicionais de cada camada de protocolo que será utilizado na transmissão da informação.

equipamentos móveis.

Com os *compacts* transportando maiores quantidades de dados, a eficiência da rede será maior, visto que os fragmentos de pacotes serão menores. Porém, sistemas onde a quantidade de dados tenha números altos a vantagem é nítida. Já em sistemas com baixos volumes de dados esta eficiência não seria assim tão vantajosa.

9 CONCLUSÕES E TRABALHOS FUTUROS

No ambiente móvel a arquitetura de software adotada influencia diretamente na complexidade do SGBDM. A arquitetura de agentes móveis não é recomendada para um SGBDM, pois a execução de uma transação ficaria a cargo um uma unidade fixa na rede e somente o resultado da transação seria enviado à unidade móvel. Assim, a transação realmente seria iniciada após a conexão da unidade móvel à rede fixa. Já a arquitetura simétrica, implicaria em maior consumo de processamento, largura de banda, memória e espaço local de armazenamento. Os dados realmente seriam móveis e as unidades móveis totalmente autônomas, obtendo um ambiente totalmente distribuído e móvel. A arquitetura Cliente/Agente/Agente/Servidor, modelo estendido do tradicional Cliente/Servidor, apresenta como uma arquitetura ideal, pois utiliza os SGBD fixos já existentes. Implementa um agente móvel e um agente fixo, ambos responsáveis pelas trocas de informações, gerenciamento da transação distribuída e o encapsulamento dos dados segundo um protocolo específico para o ambiente em operação.

Na classificação dos modelos de transações móveis perceberam-se algumas diferenças quanto a forma de execução de cada modelo. No modelo Kangaroo o foco do modelo de execução está na mobilidade da transação e não na garantia de uma transação móvel segura. Com exceção da atomicidade, todas as demais propriedades ACID foram deixadas de lado a favor de um modelo simples. Por outro lado o modelo de transações móveis PRO-MOTION apresenta regras rígidas quanto às propriedades ACID, garantindo todas, e um modelo de execução simples que pode ser implementado em uma linguagem orientada a objetos, como o Java, para qualquer SGBD.

O novo modelo de *compact* apresentado poderia tornar mais eficiente o modelo de execução da transação móvel no modelo PRO-MOTION. Influi diretamente na eficiência da distribuição dos dados, economizando os recursos ainda escassos da unidade móvel.

9.1 RELEVÂNCIA DO TRABALHO

O presente trabalho contribui significativamente para todos os interessado em SGBDM. Dentre muitas contribuições cita-se:

Revisão bibliográfica: Apresenta-se uma revisão bibliográfica dos principais pesquisadores da área de Sistema de Banco de Dados Móveis. Faz-se um resumo de suas principais obras e artigo, servindo como referência para consultas futuras;

Classificação dos modelos de execução: Classifica o modelo de transações móveis segundo o modelo de execução da transação móvel. Esta classificação mostra uma visão resumida dos principais modelos de transações móveis e sua foram de execução, servindo como ponto de partida para estudos mais aprofundados sobre as arquiteturas dos SGBDM.

Classificação dos modelos em relação a ACID: Nesta classificação pode-se perceber claramente como cada modelo implementa os controles necessários para uma transação confiável e íntegra. Observam-se quais os modelos são mais apropriados para cada tipo de ambiente de computação e por fim dá uma visão geral dos principais modelos e suas garantias com relação às propriedades ACID.

Proposta ao modelo PRO-MOTION: Este trabalho apresenta uma melhoria significativa com relação a eficiência da distribuição e do controle dos dados do ambiente móvel.

9.2 PERSPECTIVAS DE PESQUISAS FUTURAS

Este trabalho apresenta estudos teóricos sobre os modelos de execução e de transações para os SGBDM. Dar continuidade a este trabalho é fundamental para conseguir modelos de transações mais eficientes e confiáveis.

A implementação dos modelos citados neste trabalho em uma linguagem de computação para comprovar sua eficiência e complexidade é de suma importância para atestar a viabilidade dos modelos apresentados.

Implementar um simulador de transações móveis para que se possa analisar uma arquitetura específica e quantificar o consumo de recursos da unidade móvel, poderiam economizar muito tempo para a validação de uma proposta de transação móvel.

Estudo comparativo entre os protocolos de comprometimento global 2PC e o 3PC, analisando suas viabilidade e de utilização no ambiente de computação móvel poderia ajudar a apresentar modelo que sejam mais eficientes.

O isolamento de uma transação, juntamente com a durabilidade são partes fundamentais para um ambiente colaborativo, onde vários usuários podem apropriar-se de um mesmo dado ao mesmo tempo e suas alterações devem ser duráveis. O protocolo 2PL apresenta inconvenientes devido ao fato de que as transações possam ter um longo tempo de duração. Por outro lado a serialização de uma transação não é uma tarefa fácil. Assim, um estudo mais aprofundado neste campo faz-se necessário.

10 BIBLIOGRAFIA

ALONSO, Rafael; KORTH Henry F. **Database System Issues in Nomadic Computing**. Washington: Proceedings of the 1993 SIGMOD Conference, Maio, 1993.

ALONSO, Rafael; Korth Henry F., **Database System Issues in Nomadic Computing**, Princeton: Matsushita Information Technology Laboratory, 1995.

ALVARADO, Patrícia S.; RONCANCIO Claudia; ADIBA Michel. **Móbile Transaction Supports for DBMS: An Overview**. *Rapport de Recherche RR 1039-I-LSR 16*, Laboratoire LSR - IMAG, Grenoble, Julho, 2001.

ANATEL. Obtido online em www.anatel.gov.br, 2002

BADRINATH, B. R.; PHATAK, Shirish H.. **An Architecture for Mobile Databases**. Relatório Técnico DCS-TR-351, 1998, p. 1-8.

BRICABRAC. Obtido online em http://www.bricabrac.com.br/fset_telefone.htm - Copyright © 1996/2001 - Bric-A-Brac - O Túnel do Tempo Virtual – 11/06/02

CHESS, D.; GROSOF, B.; HARRISON, D. Levine; PARRIS, C; TSUDIK, G. **Itinerant Agents for Mobile Computing.** IEEE Personal Communications, 2(5), Outubro 1995.

CHRYSANTHIS, Panos K. **Supporting Semantics based Transaction Processing in Mobile Database Applications**. 14th IEEE Symposium on Reliable Distributed Systems: setembro, 1995.

CHRYSANTHIS, Panos K. **Transaction Processing in a Mobile Computing Environment**. In Workshop on Advances in Parallel and Distributed Systems. IEEE: outubro, 1993, pp 77-82.

CHRYSANTHIS, Panos K.; PITOURA, Evaggelia. **Mobile and Wireless Database** Access for Pervasive Computing**.** 16th International Conference on Data Engineering**,**

CHU, Shao Yong. **Banco de dados: Organização, Sistemas e Administração**. São Paulo: Atlas, 1983.

COMER, E. Douglas. **Interligação em Rede com TCP/IP - Volume I**. Rio de Janeiro. Editora Campus - 1998.

Coulouris G.; Dollimore, J.; Kindberg, T. **Distributed Systems: Concepts and Design**. Addison Wesley, 3ª edição, 2001.

CUNHA, Miguel; PREGUIÇA, Nuno; MARTINS José L.; DOMINGOS, Henrique J.; DUARTE, Sérgio M.; MOURA, Francisco; BAQUERO, Carlos. **Mobisnap: Um sistema de Base de Dados para Ambientes Móveis**. 4ª Conferência sobre Redes de Computadores, Portugal, 2001.

DATE, C. J. **Introdução a Sistemas de Banco de Dados**. Rio de Janeiro: Campus, 1990.

DATE, C.J. **Introdução a Sistemas de bancos de Dados**. Rio de janeiro: Campus, 2000.

DB2 EVERYPLACE. http://www.ibm.com/software/data/db2/everyplace/, Outubro, 2002.

DORMAN, A. **Wireless Comunication - O Guia Essencial de Comunicação Sem Fio**. Rio de Janeiro. Editora Campus - 2001.

Dunham , Margaret H.; Kumar , Vijay. **Location Dependent Data and its Management in Mobile Databases.** *Proceedings of the Ninth International Workshop on Database and Expert Systems Applications,* August, 1998, p. 414-419.

DUNHAM, Margaret H; Helal A. **Mobile Computing and Databases: Anything New?** SIGMOD Record, v. 24, n. 4, dezembro, 1995, p. 5-9.

DUNHAM, Margareth H.; HELAL, Abdelsalam; BALAKRISHNAM, Santosh. **A Mobile Transaction Model that Captures Both the Data and Movement Behavior**. Monet: ACM-Baltzer Journal on Mobile Networks and Applications, vol. no. 2, Outubro, 1997,pp. 149-161.

ENDLER, Markus; Silva, Francisco J.S. **Requisitos e Arquiteturas de Software para Computação Móvel**. São Paulo: Departamento de Ciência da Computação, Universidade de São Paulo, Maio de 2001.

GEORGAKOPOULOS, D.; HORNICK, M.; SHETH A. **An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure**. Distributed and Parallel Databases, 1995, p. 119-153

GRAY, J. N.; HELLAND P.; O'NEIL, P. SHASHA, D. **The Dangers of Replication and Solution**. In Conference an management of Data. Canadá: junho, 1996, pp 173-182.

Gray, J. **Why Do Computers Stop and What Can Be Done About It**. Tutorial Notes, CIPS (Canadian Information Processing Society) Edmonton'87 Conference: Edmonton, Canada, novembro de 1987.

GRAY, J.; BERENSON, H.; BERNSTEIN, P**. A Critique of ANSI SQL Isolation Levels**. SIGMOD (ACM Special Interest Group on Management of Data), 2(24):1-10, Maio, 1995.

HELAL, A., HASKELL B., CARTER J. L., BRICE R., WOELK D., RUSINKIEWICZ M. **Any Time, Anywhere Computing.** Kluwer Academic Publishers, 1999.

HERNADEZ, Michel J. **Aprenda a Projetar seu Próprio Banco de Dados**. São Paulo: Makron Books, 2000.

INFORMIX SE: http://www-4.ibm.com/software/data/informix/se/, Outubro, 2002.

ITO, Giani Carla. **Banco de dados Móveis: Uma Análise de Soluções Propostas para Gerenciamento de Dados**, Florianópolis: UFSC, 2001.

JING, Jin; HELAL, Abdelsalam; ELMAGARMID, Ahmed. **Client-Server Computing in Mobile Environments**. ACM Computing Surveys, Vol. 31, Nº 2, June 1999.

KAYAN, Ersan; ULUSOY, Özgür. **An Evaluation of Real-Time Transaction management Issues in Mobile Database Systems**. The Computer Journal, Vol. 42, No. 6, 1999.

KOHNO, R. MEIDAN, R. E Milstein, l.b., **Spread Spectrum Access Method for Wireless Communication**, IEEE Commun. Magazzine, vol. 33, pag. 58-67, janeiro de 1995.

KORTH, Henry F.; SIBERSCHATZ, Abrahm. **Sistema de Banco de Dados**. São Paulo: Makron Books, 1995.

KROENKE, David M. **Banco de Dados – Fundamentos, Projeto e Implementação**. Rio de janeiro: LTC, 1999.

LUCENT TECHNOLOGIES: http://www.lucent.com, Janeiro, 2003.

LUCENT: **Manual da Placa PCMCIA - Orinico Gold** - LUCENT Technologies, 2002.

MADRIA, Sanjay K. **Transaction Models for Mobile Computing**. Singapore: Nanyang Technological University, 1998.

MADRIA, Sanjay K.; BHARGAVA, Bharat. **A Transaction Model for Improving Data Availability in Móbile Computing.** Distributed and Parallel Databases, 2001.

MADRIA, Sanjay K.; BHARGAVA, Bharat. **On The Correctness of a Transaction Model for Mobile Computing**. Purdue University: Department of Computer Sciences, summer, 1997.

MANICA, Eloise. **Banco de Dados Distribuídos e Heterogêneos: Arquiteturas, Tecnologias e Tendências**. Florianópolis: Instituto de Informática e Estatística da UFSC, 2001.

MARINHO, José Edemar Palubiack, **Uma Análise Sobre Replicação Assíncrona de Dados em Ambientes Móveis.** Florianópolis: Instituto de Informática e Estatística da UFSC, 2002.

MATEUS, Geraldo R; LOUREIRO Antônio A. Ferreia, **Introdução à Computação Móvel.** DCC/IM, COPPE/Sistemas, NCE/UFRJ, 11ª Escola de Computação,1998.

MELO, Rubens N.; SILVA, Sidney Dias; TANAKA, Asterio K. **Banco de Dados em Aplicações Cliente – Servidor**. Rio e Janeiro: Infobook, 1997.

PREGUIÇA, Nuno M.; BAQUERO, Carlos; MOURA, Francisco; MARTINS, José Legatheaux; OLIVEIRA, R.; DOMINGOS, Henrique João L.; PEREIRA, J. O.; DUARTE, Sérgio. **Mobile Transaction Management in Mobisnap**. ADBIS-DASFAA 2000: 379-386

MOLINA, H. Garcia; SALEM, K. **SAGAS**. ACM SIGMOD International Conference on Management of Data. ACM: Maio, 1987 pp 249-259.

MOSS, J. E. B. **Nested Transaction: An approach to Reliable Computing**. MIT: Phd Thesis, Abril 1981.

ORACLE 9i LITE: http://otn.oracle.com, Maio, 2003.

ORACLE. **Manual do Banco de Dados Oracle versão 8.1.7**. Oracle Corporation, 2002.

ÖZSU, M. Tamer. Valduriez, Patrick. **Princípios de Sistemas de Banco de Dados Distribuídos**. Rio de janeiro: Campus, 2001.

PITOURA, Evaggelia; BHARGAVA, Bharat. **Building Information Systema for Mobile Environments**. ACM: novembro, 1994.

_____; _____. **Data Consistency in Intermittently Connected Dsitributd Systems.** In Transactions on Knowledge and Data Engineering, novembro, 1999.

_____; _____. **Maintaining Consistency of data in Mobile Distributed Environment**. In 15th Int. Conference on Distributed Computer Systems, Vancouver, Canada, Maio, 1995.

_____; _____. **Revising Transaction Concepts for Mobile Computing**. Proceeding of the IEEE Workshop on Mobile Systems and Applications, Santa Cruz, CA, Dezembro, 1994.

PITOURA, Evaggelia; SAMARAS, George. **Data Management for Mobile Computing**, Kluwer Academic Publishers, 1998.

PU, C.; KAISER, G. HUTCHINSON, N. **Split Transactions for Open-Ended Activities**. In Proceedings of the Fourteenth International Conference on Very large

Databases. ACM: setembro, 1988, pp 26-37.

RAMAMRITHAN, K.; CHRYSANTHIS, Panos K**. Advances in Concorrency Control and Transaction Processing**. IEEE: 1996.

ROCHA, Ricardo C. Antunes. **Uma Arquitetura para Simulação Flexível de Protocolos para Computação Móvel**. São Paulo: Instituto de Matemática e Estatística da Universidade de São Paulo, Maio de 2001.

SEYDIM, Ays Yasemin. **An Overview of Transaction Models in Mobile Enviroments**. Dallas: Departament od Computer Science and Engeneering Southen Methodist University, 1999.

SILBERSCHATZ, Abraham; KORTH, Henry F; SUDARSHAM, S. **Sistema de Banco de Dados**. São Paulo: Pearson Education do Brasil, 1999.

SOUZA, Lindeberg Barros, **Redes de Computadores Dados, Voz e Imagem**, Editora Érica, 1999.

STALLINGS, William, **Local & Metropolitan Area Networks**, Prentice-Hall, 1997.

TANENBAUM, A. S. **Redes de Computadores**. Rio de Janeiro. Editora Campus – 1997.

TANENBAUM, Andrew S. "**Distributed Operating System**". Prentice Hall. Englewood Cliffs. New Jersey, 1995.

TAURION, Cezar. **Internet Móvel – Tecnologias, Aplicações e Modelos**, Rio de Janeiro: Campus, 2002

TEWARI, R.; Grillo, P.. **Data Management for Mobile Computers in Internet**, Proceedings of the 23rd ACM Computer Science Conference, Março, 1995, p. 246-252.

UFRGS – **Grupo de Redes. Transmissão de dados sem Fio**. http://penta.ugrgs.br/redes.94-2/lisianeh/wireless.html. UFRGS, Porto Alegre, 1996.

VARSHNEY, Upkar; Vetter Ron. **Emerging Mobile and Wireless Networks**. Communications of The ACM, vol. 43, nº 6, junho de 2000.

WALBORN, Gary D.; CRYSANTHIS, Panos K. **PRO-MOTION: Management of Mobile Transactions**. In 11th ACM Annual Symposium on Applied Computing. San Jose, Ca: Março, 1997.

WALBORN, Gary D.; CRYSANTHIS, Panos K. **Transaction Processing in PRO-MOTION**. In 14th ACM Annual Symposium on Applied Computing. San Antonio, Tx: Fevereiro, 1999.

YEO, G.D.; ZASLAVSKY, A. **Layered Approach to Transaction Management in Multidatabase Systems**. Proceedings of the 5th International Hong Kong Computer Society Database Workshop: Next Generation Database Systems, 1994, 179-189.

YEO, G.D.; ZASLAVSKY, A. **Submission of Transaction from Mobile Workstations in a Cooperative Multidatabase Processing Environment**. Preceeding of the 14th International Conference on Distributed Computing Systems, Poznan, Poland: June 1994, pp 372-379.