UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

BANCOS DE DADOS MÓVEIS:
VISÃO GERAL, DESAFIOS E SOLUÇÕES ATUAIS

TRABALHO DE GRADUAÇÃO EM BANCOS DE DADOS

Paulo Gustavo Fell Amado

Recife
Outubro de 2002

Resumo

O desenvolvimento e a popularidade dos dispositivos com capacidade de computação e comunicação móvel criam um ambiente onde as informações deixam de ser dependentes de sua localidade. Um gerenciamento diferenciado destes dados onipresentes se faz necessário, novos desafios se apresentam e antigas questões voltam inseridas em um novo contexto. Este trabalho busca dar uma visão geral deste ambiente onde um sistema gerenciador de bancos de dados tem que levar em conta os percalços da computação móvel. Serão estudadas estas questões relacionadas ao gerenciamento de dados, abordados alguns projetos já propostos e ainda em desenvolvimento e serão apontadas linhas de pesquisa e aplicações de bancos de dados móveis. Este trabalho visa estimular a curiosidade, indicar alguns caminhos àquele que procura se aprofundar e levantar uma lista de referências sobre um tema atual e promissor, apresentando uma discussão sobre os principais aspectos envolvidos.

Agradeço de todo coração e dedico este trabalho: ao meu orientador, Fernando Fonseca, pela sua extrema paciência e tolerância comigo e por sempre me receber bem; aos professores Sanjay Madria (Universidade de Missouri-Rolla, EUA) e Arkady Zaslavsky (Universidade de Monash, Austrália), muito solícitos em apontar novas fontes de pesquisa e responder a algumas dúvidas por e-mail; a Giani Carla Ito, por se disponibilizar a ajudar e trocar informações e por enviar sua tese de mestrado em BDs móveis, enriquecendo minha pesquisa; a minha família, pelo suporte em todas as ocasiões e pelo conforto de sua companhia e a Valeschka, por me proporcionar paz e equilíbrio com seu carinho, sua atenção e seu amor.

Índice

1. Introdução

Nos últimos anos, os avanços revolucionários nas tecnologias de *hardware* e de rede possibilitaram a evolução dos computadores portáteis. Alguns desses avanços podem ser notados na comunicação sem fio e celular, na tecnologia de baterias e na constante redução das dimensões, peso e consumo de energia de vários componentes. Daí a presença de uma nova geração de máquinas portáteis que vão de computadores de bolso a poderosos laptops. E apesar de seu tamanho reduzido, estes computadores móveis geralmente possuem a capacidade de conexão sem fio, possibilitando o acesso não só a computadores fixos como a outros dispositivos móveis.

A computação móvel e onipresente traz consigo novas classes de aplicações e vários novos desafios tecnológicos. Sistemas de informação poderão ser acessados de qualquer lugar, a qualquer hora buscando consultar ou modificar dados remotamente. Neste cenário, as aplicações de bancos de dados ganharão novas funcionalidades e terão novos recursos, desde que feitas as devidas adaptações e modificações.

Este é um trabalho que tem como objetivo estudar os novos recursos oferecidos e as adaptações necessárias para a implementação de um sistema de banco de dados móvel, fazendo um levantamento das linhas de pesquisa nesta área, projetos que foram e vêm sendo desenvolvidos e estudando as características de alguns dos principais sistemas no mercado de bancos de dados.

Espera-se criar o interesse por esta tecnologia emergente e pelas questões levantadas aqui e que este trabalho seja não só um incentivo, mas um ponto de partida para quem procura se aprofundar no tema, coletando uma lista de referências, em sua maioria disponíveis na rede.

No capítulo 2, a pesquisa apresenta uma visão geral do que é computação móvel, suas arquiteturas e os problemas inerentes à tecnologia e dos dispositivos dotados de capacidade de comunicação com uma rede móvel. As características dos bancos de dados móveis serão discutidas no capítulo 3, envolvendo alguns exemplos práticos e listando suas particularidades em relação à tecnologia com a qual é muitas vezes confundida, a de bancos de dados distribuídos. Será feito um estudo detalhado dos desafios do funcionamento de um banco de dados num ambiente móvel, como as adaptações no processamento de consultas para dar suporte à mobilidade, questões de *caching* e replicação de dados e o processamento de transações. No capítulo 4, são

exemplificados projetos atuais de gerenciamento de dados móveis, como o MobiSnap, ou já amadurecidos, como o sistema de arquivos Coda, de grande contribuição para o avanço da área. Nesse mesmo capítulo, também serão apontadas algumas questões em aberto e as linhas de pesquisa mais promissoras relacionadas a bancos de dados móveis. No capítulo 5, depois de um comentário sobre participação dos produtos de bancos de dados móveis no mercado de computação, serão revisadas algumas das soluções das grandes empresas, como Oracle, Sybase, Microsoft e IBM. O capítulo 6 conclui este trabalho, fazendo um levantamento do que foi visto e discutindo o resultado final produzido, bem como sugerindo trabalhos futuros relacionados ao tema. No final do trabalho encontra-se uma lista de referências pesquisadas.

2. Computação móvel, uma visão geral

A Computação Móvel, também conhecida por Computação Nômade, Pervagante (*Pervasive*), Invisível e Ubíqua (*Ubiquitous*), é o resultado da convergência da computação com as tecnologias de comunicação sem-fio e caracteriza-se pelo uso de um computador portátil capaz de se ligar a uma rede de comunicação.

Este capítulo apresenta um breve histórico dos dispositivos e suas características, discute as restrições causadas pelas propriedades de computadores móveis e portáteis que se comunicam entre si ou com computadores numa base remota, faz ainda um estudo das arquiteturas de computação móvel, dos modelos de comunicação neste ambiente e finaliza introduzindo alguns conceitos importantes.

2.1 Dispositivos móveis

Em 1984, a empresa de tecnologia britânica Psion ao lançar o Psion Organizer I, que possuía calculadora, relógio, calendário e capacidade de programação numa linguagem similar a BASIC, inaugurou o gênero de Assistentes Digitais Portáteis (*Personal Digital Assistants*, ou também *Portable Digital Assistants*), hoje os conhecidos PDAs. Paralelamente, também em 1984, a Epson lançava o seu HX-20, o primeiro *laptop* [68].

O sucesso da Psion, logo chamou atenção de grandes fabricantes como Apple, Hewlett-Packard, Motorola, Sharp Electronics e Sony Electronics, que também acabaram por lançar seus dispositivos portáteis, tornando-os populares. Mas foi em 1996 que a Palm Inc., um ano depois de ser comprada pela US Robotics, provocou um verdadeiro impacto neste mercado ao lançar o Pilot 1000 e Pilot 5000, que introduziram o conceito de sincronização de dados com um *desktop* remoto. Ou seja, as mesmas informações do PC poderiam ser acessadas pelo computador de mão localmente ou numa rede WAN (*Wide-Area Network*) ao toque de um botão [45].

A sincronização é uma troca e atualização de dados e pode ser feita a curta ou longa distância através de vários meios, como infravermelho, satélite, radiofreqüência, *spread spectrum* e de vários protocolos de comunicação de dados, incluindo: HTTP, WSP (*Wireless Session Protocol*), OBEX (Bluetooth, IrDA), SMTP, TCP/IP e protocolos proprietários. Mais adiante, no capítulo 3 serão estudados mais alguns detalhes sobre sincronização de dados.

Desde o lançamento dos primeiros Pilots, inúmeras funcionalidades têm sido adicionadas aos PDAs modernos. Sem interferir na portabilidade, avanços na tecnologia vêm permitindo melhor visualização, maior autonomia de bateria, mais poder de processamento e uma compatibilidade cada vez maior com dados multimídia.

Handhelds e *palmtops* são principalmente diferenciados pelo tamanho, poder de processamento e pelo meio de entrada de dados. Os *handhelds* possuem geralmente pequenos teclados como os *laptops*, já os *palmtops* normalmente recebem entrada de dados através de botões e de uma caneta magnética chamada *stylus pen*. Com a *stylus*, a entrada de dados pode ser feita através de um teclado *soft* que faz parte da interface gráfica ou ainda "escrevendo" na tela, o que requer um *software* de reconhecimento de escrita (*handwriting recognition*) para realizar a conversão dos dados. Alguns PDAs também já possuem tecnologias de reconhecimento de voz.

No entanto, a disponibilidade de teclados para *palmtops* e a diminuição dos *handhelds* está fazendo com que a distinção entre eles perca o sentido, assim como a tênue fronteira entre estes dispositivos e os mais poderosos telefones celulares também está se dissipando. O Nokia 7650 [38] lançado recentemente, por exemplo, possui um display de cristal líquido colorido, câmera digital embutida, organizador de arquivos, envia mensagens com texto, voz e imagem ao mesmo tempo e tem a capacidade para troca de dados via infravermelho ou Bluetooth com PCs, além de ser um telefone celular que atua em banda dual.

Existem vários tipos de plataformas e arquiteturas para PDAs como Palm, Pocket PC, Symbian, Blackberry, Embedded Linux, etc. O futuro aponta para uma padronização de comunicação entre a maioria delas.

É importante notar que portabilidade e mobilidade são conceitos diferentes, um dispositivo pode ser portátil e não possuir capacidade de computação móvel como, por exemplo, um *laptop* num carro, ou num avião, realizando tarefas comuns (como processamento de texto e planilha eletrônica), mas sem capacidade de se conectar a uma rede sem fio (sem um cartão PCMCIA, por exemplo). Analogamente, num escritório de uma empresa que funcione num prédio sem fiação, a LAN (*Local Area Network*) será uma rede sem fio (*wireless*), mas esse não será um caso de computação móvel [66].

Ao longo deste trabalho são discutidas algumas propriedades de dispositivos dotados de capacidade de computação móvel, como *palmtops*, *handhelds*, *laptops* (*notebooks*), por exemplo. A tabela 1, a seguir, mostra alguns dos mais atuais destes computadores e algumas das suas principais características.

Dispositivo	Bateria (autonomia)	Resolução em pixels	Peso (gramas)	Processador	Preço Médio (US$)
 **Palm m105**	Ni-Cd AAA Até 2 meses	160x160 (tons de cinza)	125	Motorola Dragonball EZ 16 MHz	100
 **Palm m515**	Lítio-íon Até 1 semana	160x160	139	Motorola Dragonball VZ 33 MHz	400
 **Sony Clié NR70**	Lítio-íon 7 a 10 horas	320x480	200	Motorola Dragonball VZ 66 MHz	450
 **NEC MobilePro 790**	Lítio-íon 4:30 horas	640x240	907	NEC VR4121 168 MHz	900
 **IBM Thinkpad X21**	Lítio-íon 4 horas	1024x768	1.575	Intel Mobile Pentium 3 700 MHz	1.850
 **Dell Inspiron 8200**	Lítio-íon 2:30 horas	1400x1050 (SXVGA)	3.200	Intel Mobile Pentium 4 2 GHz	2.000

Tabela 1. Características de alguns computadores móveis atuais [7,43,46,76]

2.2 Desafios do ambiente móvel

Apesar de aparentemente só trazer benefícios e também parecer de simples implementação, a computação móvel é ainda alvo de várias pesquisas, pois existem muitos fatores que impedem a utilização máxima dos recursos desta tecnologia.

Acesso a informações de qualquer lugar, a qualquer hora com mais flexibilidade e funcionalidades do que usando telefones celulares ou *pagers*, é a maior promessa da Computação móvel, que tenta ser tão revolucionária para os computadores portáteis quanto foi o advento das redes tradicionais para os computadores comuns (*desktops*).

Os computadores portáteis aos poucos conquistam espaço no mercado de computação e embora ainda não seja notado no Brasil, na Ásia, por exemplo, esse fenômeno é mais observável. O motivo desta mudança se dar lentamente reside no fato de que a tecnologia de computação móvel enfrenta vários desafios, que vêm sendo vencidos aos poucos. Segundo Forman e Zahorjan [18], estes empecilhos têm origem em 3 essenciais propriedades da computação móvel: portabilidade, comunicação e mobilidade.

2.2.1 Portabilidade

Os computadores comuns não foram projetados para serem carregados, os montadores de tais máquinas, portanto, têm certa liberdade no sentido de espaço para cabeamento e dissipação de calor, por exemplo. Por sua vez, o projeto de um PDA deve levar em conta propriedades de um bom relógio de pulso: pequeno, leve, durável, operacional em uma variedade de condições e que requer o mínimo de consumo de energia, assegurando longa vida útil para a bateria. Algumas concessões podem ser feitas em cada uma dessas áreas para que se chegue a uma boa relação de funcionalidade-performance. Note-se que quando são citadas restrições de computabilidade nesta seção, nem sempre estarão aí incluídos os notebooks, por exemplo, que tendo menor portabilidade que os computadores de mão, têm em seus modelos mais modernos características que não deixam nada a desejar a *desktops* comuns. Por exemplo, a Intel lançou, em março de 2002, o processador para notebooks Mobile Intel Pentium 4, que chega até a 1,8GHz de clock (comparável atualmente a um *desktop* poderoso), enquanto que alguns PDAs da mais nova geração (Cassiopeia da Casio e Compaq Ipaq, por exemplo) possuem processador Intel StrongARM de 206 MHz. Porém, o maior poder computacional tem seu preço para os notebooks: o consumo de energia. Nos mais modernos (Titanium PowerBook da Apple e o Thinkpad Transnote da IBM, por exemplo), a bateria de Lítio-Íon funciona de 3 a 5

horas, enquanto que nos melhores PDAs disponíveis, a autonomia varia geralmente de 8 a 12 horas com o computador em uso, segundo testes feitos com 6 tipos de PDAs [19].

A seguir, são listadas algumas dificuldades de projeto impostas pela portabilidade de um dispositivo móvel [18]:

- Energia – Componente mais pesado de um PDA, a bateria enfrenta problemas de tamanho/peso e de necessidade de recarga. Técnicas para minimização do consumo de energia podem aumentar a portabilidade reduzindo o peso e aumentando o tempo de vida da bateria. O consumo de componentes dinâmicos é proporcional a CV^2F, onde C é a capacitância do circuito, V a voltagem e F a freqüência de clock. Esta função sugere 3 formas de economizar bateria: a capacitância pode ser reduzida por um maior nível de integração VLSI (*Very Large-Scale Integration*) e por tecnologia de multichips, a voltagem pode ser minimizada através do design dos chips e a freqüência de clock pode ser reduzida trocando-se poder computacional por menor consumo de força. O consumo pode ser diminuído também por operações inteligentes no dispositivo móvel, como desligar a tela ao perceber que o computador está ocioso. Além disso, o projeto de *software* também deve levar em conta o consumo de energia, este tipo de aplicação adaptável voltará a ser abordado na seção 2.3.1. Nas comunicações, transmitir consome mais que receber dados, desta forma, as estações base podem, por exemplo, freqüentemente enviar mensagens por difusão (*broadcast*) que normalmente teriam que ser requisitadas pelo dispositivo móvel, o qual gastaria mais energia transmitindo esta requisição;
- Riscos de perda e extravio de dados – A portabilidade aumenta o risco de danos físicos, acesso não autorizado, perda e roubo do dispositivo móvel, tornando tais computadores menos seguros e confiáveis. Para reduzir estes riscos, o usuário deve ter uma preocupação ainda maior com operações de backup e o armazenamento de dados importantes numa base remota que possa ser acessada quando necessário;
- Interface com o usuário limitada – É desnecessário lembrar que o uso de mouses é inviável nos PDAs que realizam interface com o usuário através de botões ou canetas *stylus*. Para as aplicações que necessitam da abertura de várias janelas ao mesmo tempo, estes dispositivos também são inapropriados, levando a desafios de design gráfico, muitas vezes tendo que reduzir funcionalidades;
- Capacidade de armazenamento em disco – O espaço de armazenamento num computador móvel também é comprometido pelo tamanho físico e energia. E, como drives de disco consomem mais do que chips de memória, vários computadores móveis não os possuem. Algumas estratégias interessantes para contornar o problema

do armazenamento são: compressão automática de arquivos, acesso a dados armazenados remotamente, compressão de páginas de memória virtual, compartilhamento de bibliotecas de código. Uma abordagem para reduzir o tamanho do código em programas executando num dispositivo móvel é a interpretação de linguagens de scripts, ao invés de execução de objetos de código, que são tipicamente maiores.

2.2.2 Comunicação

Redes sem fio se comunicam através de sinais de ondas de rádio, através de satélites, microondas ou pulsos de luz infravermelha, variando bastante a área de alcance. A comunicação através deste meio enfrenta mais obstáculos do que a comunicação através de fios, uma vez que o ambiente interage com os sinais, potencialmente bloqueando a transmissão ou inserindo ruídos e ecos. Desta forma, a comunicação sem fio é caracterizada por uma menor largura de banda, maior taxa de erro e desconexões indesejadas mais freqüentes. Estes fatores, por sua vez, aumentam a latência de dados na comunicação, resultado de retransmissões, processamento de protocolo de controle de erro e rápidas requisições de reconexão. O custo de implementação de uma rede sem fio é em geral maior do que aquele para uma rede comum. O usuário que se conecta a este tipo de rede deve ainda levar em consideração formas de cobrança deste serviço. Alguns acessos são cobrados por tempo de conexão, outros por mensagens enviadas (*packet-radio*) [48]. Forman e Zahorjan [18] discutem algumas dificuldades do projeto de uma boa comunicação em computação móvel, como:

- Alta taxa de desconexão – Uma falha na rede pode levar processos que estão rodando em locais diferentes a pararem completamente, potencialmente levando a sérios problemas. Para mascarar algumas falhas de rede, técnicas como operações assíncronas podem ser aplicadas. Em tais operações, o cliente envia várias requisições sem precisar de confirmações de recepção (*acknowledgements*). Outras operações similares incluem pré-carregamento (*prefetching*) dos dados, que consiste em transferir dados antes que eles sejam necessários. O sistema de arquivos Coda, que será abordado no capítulo 4 deste trabalho, é um bom exemplo de como lidar com desconexões de rede. Entretanto, nem todas as desconexões de rede podem ser mascaradas, o que poderia ser contornado, por exemplo, através de uma boa interface com o usuário que indicasse quais operações não estão disponíveis em determinado momento;

- Alta variação de largura de banda – A variação de banda (*bandwidth*) nas redes sem fio pode ser de várias ordens de magnitude, dependendo de que tipo de conexão um cliente está usando naquela hora, ou mesmo do número de clientes servidos numa mesma célula. Em números, uma LAN sem fio (*Wireless Local-Area Network*) pode transmitir numa taxa de 11 Mbps (padrão IEEE 802.11b) até no máximo 54 Mbps (padrão IEEE 802.11a), e com transmissões GSM e CDMA pode-se ter taxas variando de 9,6 Kbps a 384Kbps [16]. Uma aplicação pode lidar com essa variação através de 3 abordagens: assumindo uma alta largura de banda, assumindo baixa largura de banda ou se adaptando aos recursos disponíveis (mais complexa);
- Rede Heterogênea – Ao contrário da maioria dos computadores estacionários, computadores móveis encontram conexões heterogêneas em vários sentidos. Ao mudar de uma célula para outra, poderá também haver mudanças nos protocolos e velocidades de transmissão. Em algumas situações, um computador móvel pode solicitar várias conexões como, por exemplo, quando duas células adjacentes se sobrepõem ou quando é possível que os acessos com fio e sem fio ocorram concorrentemente. Pode também haver a necessidade de se trocar a interface de comunicação como, por exemplo, mudando da cobertura celular de uma cidade para cobertura por satélite no campo;
- Riscos de segurança – Especialmente se a transmissão se estende por uma longa área, a segurança da comunicação sem fio pode ser bem mais comprometida do que nas redes comuns. A segurança se torna ainda mais complicada quando se trata de delegar níveis de acessibilidade a um usuário móvel. A comunicação segura através de canais inseguros é alcançada através do uso de criptografia, que pode ser feita através de *software*, ou de *hardware* especializado (maior performance e maior custo). A segurança depende de uma chave criptográfica conhecida apenas pelas partes autorizadas. O WEP (*Wired Equivalent Privacy*) é o protocolo de segurança mais usado para o padrão 802.11b, mas sua chave de criptografia é fraca e o algoritmo é simples e conhecido, sendo fácil encontrar na Web ferramentas para decifrá-lo. O gerenciamento destas chaves é difícil e pode ser automatizado por um servidor de chaves criptográficas (como o Kerberos, do MIT [29]), no entanto, estes servidores também são suscetíveis a ataques. Para tornar a rede mais segura, são usados recursos como *firewalls*, sistemas de detecção de intrusos, uso de ferramentas de criptografia mais sofisticadas. É também importante a conscientização dos usuários que muitas vezes facilitam a abertura de brechas na segurança.

2.2.3 Mobilidade

A capacidade de mudar de localidade enquanto ainda conectado à rede, aumenta a volatilidade da informação. Alguns dados considerados estáticos para computadores fixos, poderão se tornar dinâmicos em dispositivos móveis. Essa volatilidade levanta questões importantes de custo-benefício como, por exemplo, armazenar um objeto volátil em *cache* faz pouco sentido. A mobilidade introduz vários problemas: o endereço de rede de um computador móvel muda dinamicamente, o caminho de comunicação aumenta quando o mesmo visita uma célula vizinha, sua localização atual afeta parâmetros de configuração, assim como respostas a consultas (como será visto na seção que trata de gerenciamento de localidade). Em [18] são discutidos alguns problemas inerentes à mobilidade de um dispositivo, listados a seguir:

- Migração de endereço – Na medida em que as pessoas se deslocam, seus computadores móveis usarão diferentes pontos de acesso à rede ou "endereços de rede". Uma vez que um sistema armazena o endereço de um *host*, este é geralmente colocado em *cache* com um longo tempo de expiração. Invalidar entradas defasadas está fora de questão, pela quantidade de processamento requerido. No protocolo IP (*Internet Protocol*), por exemplo, o IP de um *host* está intrinsecamente ligado a seu endereço de rede. Mover para uma nova localidade significa obter um novo IP. Geralmente é necessária a intervenção humana para coordenar o uso de endereços. Para serem enviadas a um computador móvel, as mensagens precisam ser enviadas a seu endereço anterior mais recente. Quatro mecanismos básicos para determinar o atual endereço de um computador são: difusão, serviços centrais, home base e ponteiros de encaminhamento (*forwarding pointers*). Na difusão, a mensagem é enviada a todas as células da rede e o computador procurado é requisitado a responder com o seu atual endereço. No método de serviço central, o endereço atual de cada computador móvel é mantido numa base de dados, toda vez que o computador muda de célula, envia uma mensagem para atualizar esta base. O home base consiste essencialmente da técnica de serviço central sendo distribuído, isto é, a localização de um dado computador é conhecida apenas por um servidor. O método de ponteiros de encaminhamento consiste em guardar uma cópia do novo endereço do computador móvel na célula anterior toda vez que ele muda de localidade, sendo possível atualizar esse endereço gradualmente para refletir novas mudanças. Estes quatro mecanismos são os blocos de construção das propostas de esquematização do IP móvel [24];

- Informação dependente de localidade – Prover mecanismos que obtenham a configuração de dados apropriada a cada localidade é um desafio importante para a computação móvel. Um computador portátil carregado por um usuário pode ser usado numa grande variedade de domínios administrativos, cada um com suas convenções. Além do problema da configuração, dispositivos móveis necessitam de acesso a mais informações dependentes de localidade para, por exemplo, servir de guias em lugares não familiares a um usuário que procura saber onde fica o posto de gasolina aberto mais próximo indo ao norte de onde ele se encontra. Este aspecto voltará a ser abordado neste trabalho quando forem tratadas as consultas sensíveis à localidade. Um grande desafio da computação móvel relacionado a este tópico é que além do custo de comunicação com elementos móveis, há o custo adicional de localizá-los. Muitas outras questões podem surgir como o acesso flexível à informação de localização do usuário sem a violação de sua privacidade;
- Migração de localidade – Mesmo que um computador móvel tenha a capacidade de achar o servidor mais próximo para um dado serviço, com o tempo, a migração poderá mudar esta condição. A distância física entre dois pontos não reflete necessariamente a distância das duas redes e o tamanho do caminho de comunicação pode crescer desproporcionalmente, levando à existência de mais passos intermediários, aumentando a latência, o risco de desconexão e consumindo capacidade da rede. Para contornar tais problemas, o serviço de conexões pode ser dinamicamente transferido para servidores que se encontrarem mais próximos à localidade da unidade móvel.

A tabela 2, na próxima página, resume os desafios discutidos acima, decorrentes das propriedades dos dispositivos móveis.

Para contornar estas restrições e dificuldades, a computação no ambiente móvel deve apresentar certas operações e funcionalidades específicas para lidar com a mobilidade e a migração dentro da rede ou entre redes, adaptar-se à variedade de recursos computacionais disponíveis, sendo as aplicações projetadas para lidar com tal adaptação e fazer com que a qualidade e às vezes inexistência de sinal ou serviço da rede sejam transparentes ao usuário, o máximo possível.

A próxima seção (2.3) estuda conceitos relacionados a algumas destas operações especiais, que facilitam a compreensão de como funciona um sistema de computação móvel.

PROPRIEDADE	CARACTERÍSTICA	DESAFIO
Portabilidade	Consumo de energia	– Peso × tempo de vida da bateria – Consumo × desempenho computacional – Operações inteligentes
	Perda e extravio de dados	– Backup em base remota
	Interface limitada	– Design gráfico
	Capacidade de Armazenamento	– Compressão automática de arquivos – Acesso a dados remotos – Interpretação de scripts
Comunicação	Desconexões freqüentes	– Operações assíncronas – *Prefetching* – Indicação de indisponibilidade de operação
	Variação de largura de banda	– Aplicações assumindo ou baixa ou alta largura de banda – Aplicações que se adaptam a baixas e altas larguras de banda automaticamente
	Rede Heterogênea	– Protocolos, velocidades de transmissão e cobertura celular potencialmente diferentes
	Segurança	– *Firewalls*, sistemas de detecção de intrusão, ferramentas de criptografia – Gerenciamento de chaves criptográficas
Mobilidade	Migração de endereço	– IP móvel
	Informação dependente de localidade	– Variedade de convenções em diferentes localidades – Consultas sensíveis à localidade – Preservação da privacidade
	Migração de localidade	– Transferência dinâmica do serviço de conexão

Tabela 2. Desafios da computação móvel, decorrentes de suas principais propriedades

2.3 Conceitos importantes

Esta seção introduz alguns conceitos usados quando se estuda o paradigma de computação móvel e que são fundamentais para seu entendimento.

2.3.1 Adaptação

À capacidade de ajuste e à possibilidade de produzir resultados apesar de condições adversas dá-se o nome de adaptação. Num sistema de computação móvel, não há mudanças neste conceito. Dispositivos neste ambiente, apesar das restrições vistas na seção anterior, devem possuir a capacidade de executar um número crescente de tipos de aplicações e se adaptar a situações onde há uma demanda por

recursos computacionais nem sempre disponíveis e outras onde os recursos são abundantes e aparentemente dispensáveis.

Estas dificuldades enfrentadas pelo computador móvel demandam uma grande confiança nos servidores fixos, no entanto, a necessidade de lidar com redes não confiáveis e de baixa performance, faz necessária a confiança também no próprio dispositivo móvel. Deve haver um balanceamento entre a necessidade e a disponibilidade de recursos computacionais para executar uma aplicação. Por exemplo, compressão de dados que serão enviados pela rede gasta poder de processamento, mas a compensação vem com a economia de largura de banda. Este balanceamento deve ser dinâmico, já que as circunstâncias em que se encontra um dispositivo móvel apresentam várias e freqüentes mudanças.

As estratégias de adaptação de um ambiente móvel variam entre dois pontos extremos, como mostra a figura 1.

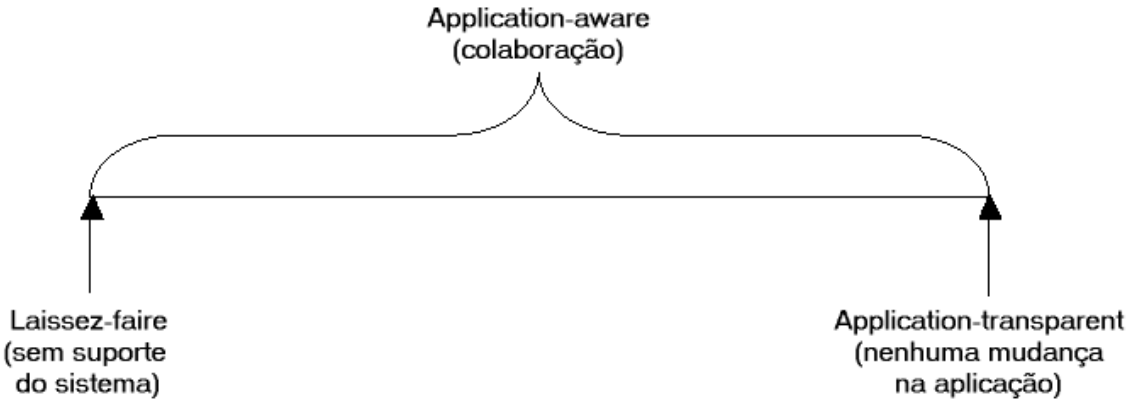

Figura 1. Variação das abordagens de adaptação

Num extremo, a adaptação é de inteira responsabilidade das aplicações individuais (*laissez-faire*). Apesar de evitar a necessidade de suporte do sistema, a este tipo de abordagem falta um gerenciador central a fim de resolver a incompatibilidade de demanda por recursos em diferentes aplicações e impor limites no uso destes recursos. As aplicações também se tornam mais difíceis de codificar.

No outro extremo (*application-transparent*), a responsabilidade da adaptação é carregada inteiramente pelo sistema. Esta abordagem é atraente porque é compatível com aplicações já existentes, que continuam a funcionar no ambiente móvel sem nenhuma modificação. O sistema fornece total gerenciamento e controle. A

desvantagem desta abordagem é que pode haver situações em que a adaptação feita pelo sistema seja inadequada e comprometa a produção.

Entre estes dois extremos, estão as possibilidades que são coletivamente referidas como adaptações cientes da aplicação (*application-aware adaptations*). Esta abordagem permite que as aplicações determinem o quanto devem se adaptar, mas preserva a habilidade do sistema de monitorar os recursos e decidir onde alocá-los [53].

As adaptações podem ser feitas em aplicações a fim de minimizar as restrições da computação móvel. Muitas formas de endereçar este problema são possíveis. Flinn e Satyanarayanan, por exemplo, propuseram um modelo de adaptação *application-aware* para minimizar o consumo de energia no cliente [17].

Outros exemplos práticos de dois tipos de adaptação são o sistema de arquivos distribuídos Coda (*application-transparent*) e a plataforma Odyssey (*application-aware*), ambos estudados no capítulo 4.

Um outro conceito relacionado à adaptação é o de escalabilidade (*scalability*). Diz-se que um sistema computacional é "escalável" se sua performance se mantém inalterada ou aumenta ao serem adicionados recursos, ou quando é capaz de reduzir ou remover o acesso a recursos escassos.

2.3.2 Operação desconectada

Devido à natureza da comunicação móvel sem fio, como foi visto, muitos fatores afetam a confiabilidade e a qualidade das conexões. No ambiente móvel e sem fio, desconexões intencionais ou não ocorrem freqüentemente. Um usuário cujo computador móvel esteja com a bateria fraca, pode desconectar da rede propositalmente, a fim de poupar energia.

Na operação desconectada, conceito introduzido e testado pela primeira vez no sistema de arquivos Coda (ver capítulo 4), o cliente continua tendo acesso aos dados durante interrupções temporárias de serviço. Isto é possível devido ao armazenamento dos dados localmente (em *cache*). Mais adiante, no capítulo 3, alguns detalhes do processo de *caching* de dados serão estudados.

Mas a modificação e a leitura dos dados armazenados localmente causam alguns problemas como necessidade de recuperação (*recoverability*) e problemas de consistência dos dados. Recuperação de dados significa a habilidade de restaurar uma base de dados depois de uma falha no sistema ou um erro de execução, fazendo com que ela volte a um estado consistente previamente conhecido. Um princípio simples de recuperação é a redundância de dados, ou seja, o armazenamento dos mesmos dados

múltiplas vezes. Os dados locais armazenados em *cache* podem estar desatualizados ou inconsistentes com os da base, mas os dispositivos móveis não devem ter conhecimento desta situação até que seja reconectado. Até que ocorra a reconexão, os dados modificados localmente não deverão ser propagados.

A transparência para a aplicação é preservada durante uma desconexão, porque o sistema tem a responsabilidade de propagar modificações e detectar possíveis conflitos de atualização, quando a conexão for retomada.

A operação desconectada pode ser útil até quando há uma boa conexão com a rede. Por exemplo, estendendo o tempo de vida da bateria ao evitar transmissões e recepções desnecessárias dos dados que estiverem armazenados localmente [53].

2.3.3 Handoff

Handoff é a capacidade de uma rede de dar suporte dinâmico à migração de terminais móveis entre suas células. É através desse processo, ilustrado na figura 2, que usuários da rede conseguem manter seus dispositivos móveis conectados ao sair de uma célula para outra vizinha. Ou seja, graças ao *handoff* é possível perder o contato com uma estação base numa célula, estabelecendo contato com outra sem, no entanto, haver desconexão, esta troca deve ocorrer quando o *host* móvel (o carro da figura) está na região onde as células se sobrepõem [27].

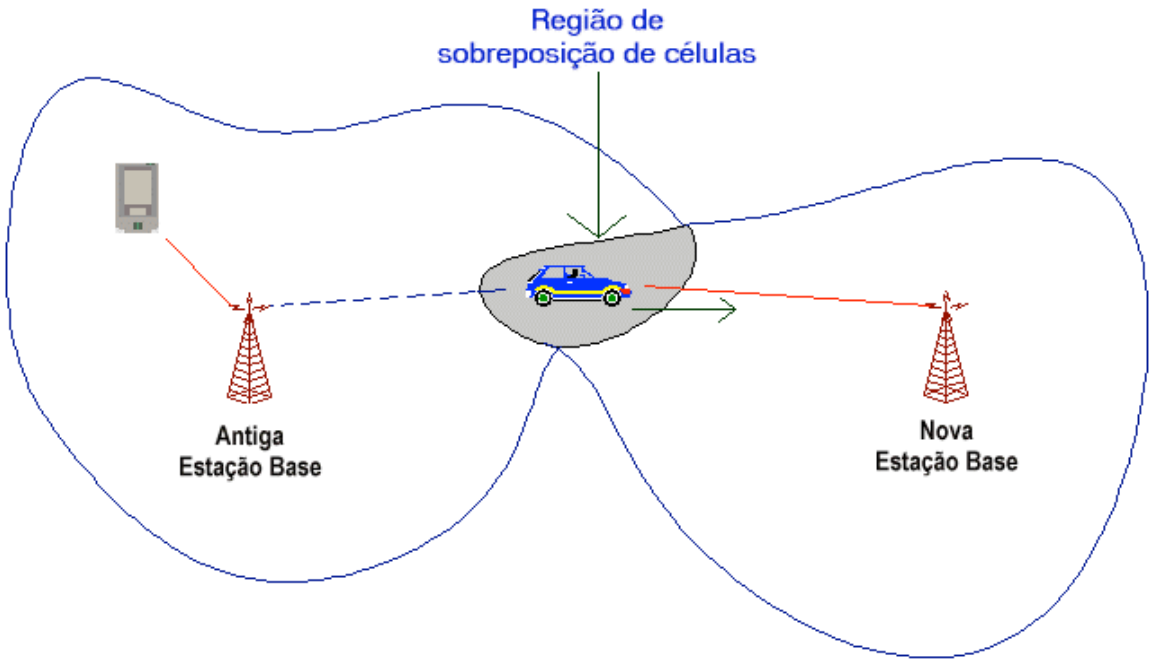

Figura 2. A unidade móvel (carro) viaja de uma célula para a outra, ainda mantendo a conexão, graças ao processo de *handoff* (adaptada de [27])

O processo de *handoff* pode ser subdividido em 3 fases: medição, decisão e execução. Na fase de medição, tanto a unidade móvel quanto a estação base realizam cálculos e medem vários parâmetros da conexão e da rede como, por exemplo, a força do sinal. Na fase de decisão, baseado nos cálculos da fase anterior, é avaliado se um *handoff* é necessário naquele momento. Finalmente, na fase de execução, o *handoff* propriamente dito, de uma célula para a outra é realizado. Destacam-se duas subfases na execução: o estabelecimento de uma nova conexão e a liberação da anterior.

Há vários critérios de classificação dos *handoffs*, Kumar [27] lista 3 deles baseados em diferentes características do processo:

Baseado na detecção do *handoff*: aqui, leva-se em conta quem realiza as fases de medição e decisão. Estas fases podem ficar a cargo da unidade móvel, das estações base ou, na estratégia mais freqüente, a medição é feita no *host* móvel e a decisão e execução realizadas pelas estações base;

Baseado na atribuição do canal: a estação base anterior e a nova podem estar ligadas por uma mesma central de comutação de conexão móvel (*Mobile Switching Center*), ou por centrais diferentes;

Baseado na transferência de conexão: o dispositivo móvel pode conectar-se com apenas uma estação base por vez ou com mais de uma. Conectando-se a apenas uma (*hard handoff*), ocorrerá uma interrupção de serviço durante a transferência. Quando o *host* móvel pode conectar-se a mais de uma estação base ao mesmo tempo, a transição ocorre suavemente (*soft handoff*), pois a desconexão com a base anterior só ocorre quando há um sinal aceitável até a nova estação base.

O *handoff* deve levar em conta alguns fatores como a renegociação de qualidade de serviço (QoS) ao chegar numa nova célula ou rede, o tráfego de sinalizações que pode ocorrer numa rede muito povoada, entre outros. Radhakrishnan [50] faz um estudo mais aprofundado deste importante processo.

2.4 Arquiteturas de computação móvel

A arquitetura geral de um sistema de computação móvel e sem fio é formada por entidades ligadas por uma rede fixa e interconectada por fios e por entidades móveis que se comunicam com esta rede.

A unidade móvel (ou *host* móvel) é um dispositivo capaz de se conectar com a rede fixa através de um link sem fio (*wireless link*). *Hosts* estacionários são conectados através de uma rede de alta velocidade interligada por fios (Fast Ethernet com taxa de 100 Mbps ou Gigabit Ethernet com 1 Gbps, por exemplo) e podem ser *hosts* fixos ou

estações base (*base stations*). O *host* fixo é um computador de propósito geral, não equipado para administrar unidades móveis, mas que pode ser configurado para isso [13]. A estação base (também chamada de estação de suporte à mobilidade, *Mobile Support Station* [22]) possui uma interface de comunicação para o acesso de dados pelas unidades móveis que se encontram em sua célula, ou seja, que estão na área de cobertura dentro da qual uma estação base consegue captar seu sinal. Quando um computador móvel sai de uma célula e entra em uma nova, está sob o controle de uma nova estação base. Como a comunicação entre eles se dá sem fio, a taxa de transferência de dados varia de poucos kilobits por segundo (Kbps) a alguns megabits por segundo (Mbps). Como foi visto no tópico 2.3.2, a variação de largura de banda é um dos desafios da comunicação em computação móvel. O esquema da arquitetura geral é mostrado na figura 3 da próxima página.

Apesar da maioria da pesquisa feita na área considerar a arquitetura geral vista acima, outro tipo de arquitetura de computação móvel vem sendo estudado, a arquitetura de rede ad-hoc móvel (*Mobile Ad-hoc Network,* ou MANET). Em redes Ad-hoc, os *hosts* móveis se comunicam entre si sem o suporte de uma infra-estrutura estática e com fios ou de uma administração centralizada. Esse tipo de rede móvel é muito usado em campos de batalha e situações de recuperação de desastres [20], por exemplo. A mobilidade dos nós em uma MANET pode levar a mudanças de topologia de rede freqüentes e imprevisíveis. Em [35], Nesargi e Prakash discutem soluções anteriores e propõem um novo protocolo de configuração de *hosts* dinâmicos e distribuídos para redes ad-hoc móveis. A figura 4, na página seguinte, mostra uma rede ad-hoc móvel, com topologia dinâmica, os nós móveis se comunicam entre si.

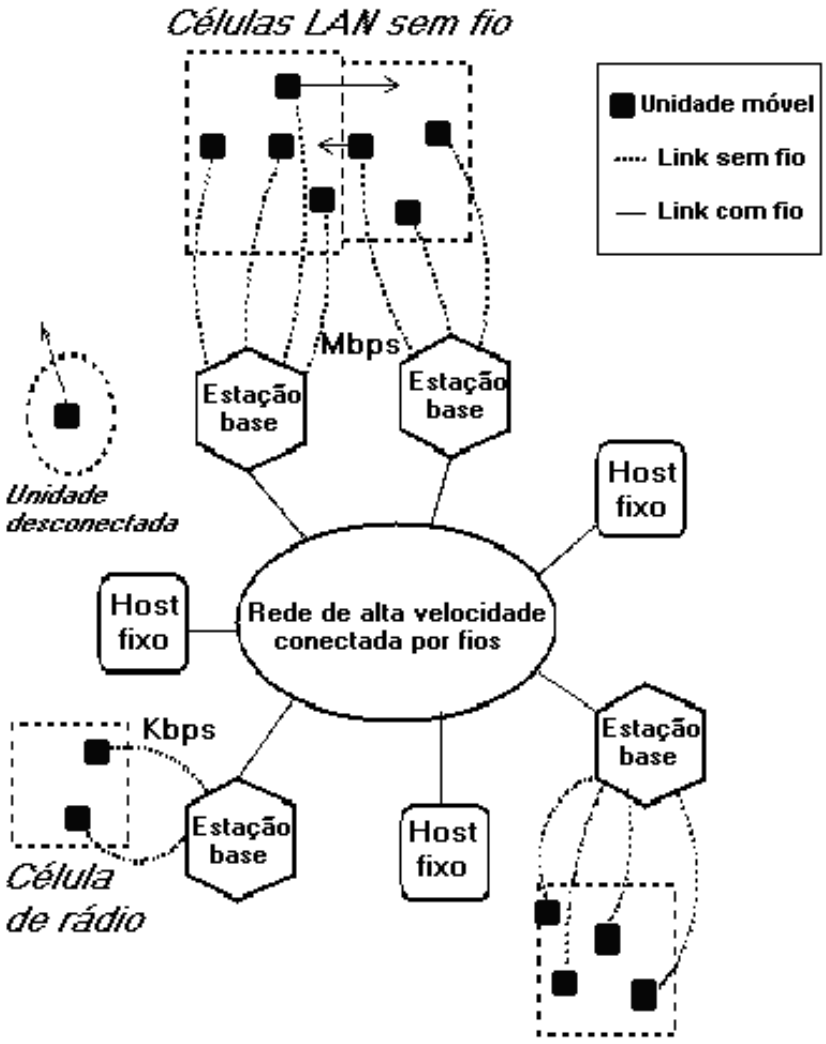

Figura 3. Arquitetura geral de um sistema de computação móvel (adaptação da figura 1 de [73])

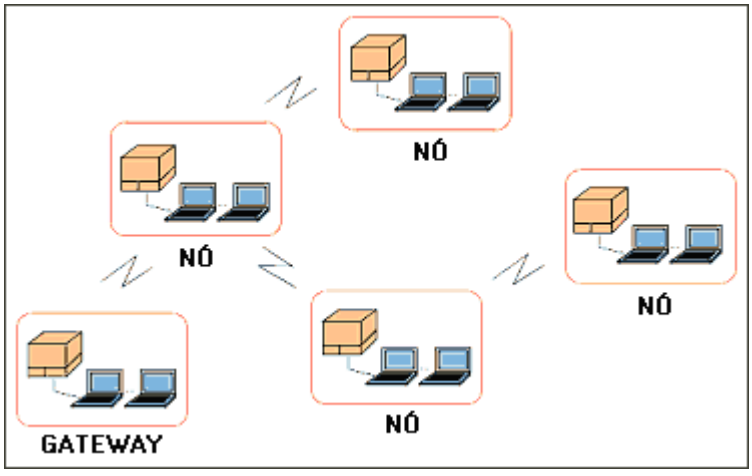

Figura 4. Rede ad-hoc móvel (MANET) (adaptação de [55])

2.5 Modelos de comunicação móvel

Quanto de funcionalidade deve ser atribuído a um *host* móvel? Como já foi visto neste trabalho, as unidades móveis se caracterizam por serem pouco confiáveis, expostas a roubos, danos e falhas de segurança e pobres em recursos.

Estas razões seriam suficientes para que estes dispositivos fossem tratados como terminais sem inteligência (*dumb terminals*) que apresentassem apenas uma interface com o usuário e deixassem toda carga de funcionalidade para as estações localizadas em rede fixa. Por outro lado, como também foi visto anteriormente, as redes sem fio são de caro acesso, lentas e pouco confiáveis, o que demanda funcionalidade adicional aos *hosts* móveis para diminuir sua dependência dos servidores remotos. Embora ainda não haja consenso a respeito do papel dos *hosts* na computação móvel distribuída, estas considerações contraditórias dão origem a modelos que proporcionam o ajuste flexível das funcionalidades dos *hosts* móveis [48].

Esta seção apresenta a discussão feita por Ito [25] a respeito desses modelos, um estudo mais aprofundado do tema é feito por Pitoura e Samaras [48].

2.5.1 Modelo Cliente – Servidor

Neste modelo, o cliente que é a unidade móvel, se comunica diretamente com o servidor localizado na rede fixa, requisitando serviços, como mostra a figura 5. Em alguns casos, a funcionalidade e os dados são distribuídos entre diversos servidores fixos, que podem ter que se comunicar entre si para atender a requisição de um cliente.

Na sua forma padrão, o modelo Cliente – Servidor apresenta alguns problemas quando utilizado no ambiente móvel, pois a comunicação é feita em baixas velocidades e não é confiável. Outro fator a ser considerado é a desconexão. Portanto, esse modelo deve ser estendido para tratar desconexões, comunicação não estável e não confiável entre clientes e servidores. Em caso de desconexão, a unidade móvel deverá emular a função do servidor para que o processamento das operações se concretize.

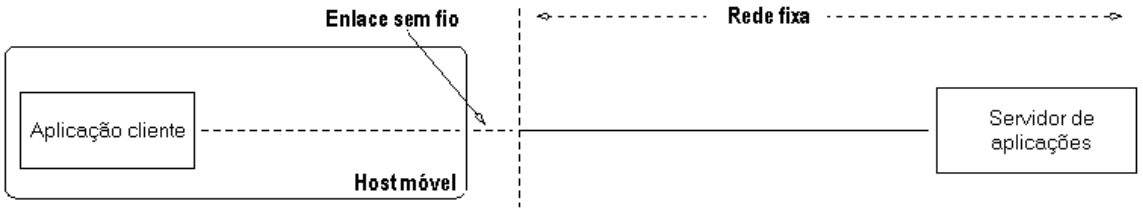

Figura 5. Modelo Cliente – Servidor (adaptada de [48])

Duas extensões do modelo Cliente–Servidor são discutidas a seguir: modelo Cliente–Agente–Servidor e modelo Cliente–Interceptador–Servidor.

2.5.2 Modelo Cliente – Agente – Servidor

O modelo Cliente–Agente–Servidor, apresentado na figura 6, constitui uma das melhores abordagens para o desenvolvimento de aplicações móveis, pois suporta a desconexão. Quando uma desconexão é realizada, tanto o servidor remoto quanto o cliente executam as atualizações a serem feitas.

Agentes foram acrescentados ao modelo Cliente–Servidor, sendo que estes podem executar tanto tarefas de comunicação como de aplicação, podendo residir na rede fixa ou na unidade móvel.

No contexto de gerenciamento de redes, o agente reúne informações sobre os dispositivos da rede e executa comandos em reposta a uma requisição do gerenciador.

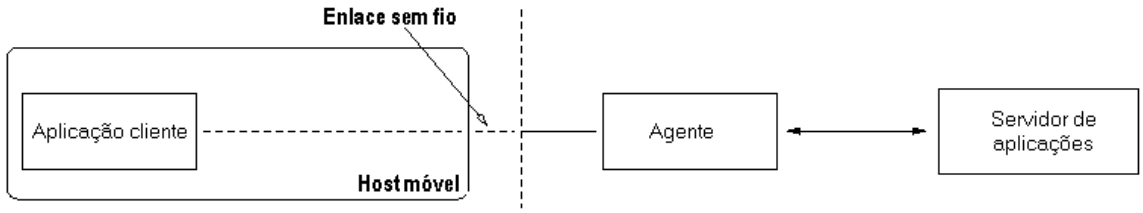

Figura 6. Modelo Cliente – Agente – Servidor (adaptada de [48])

2.5.3 Modelo Cliente – Interceptador – Servidor

Esse modelo divide o agente em duas partes, uma localizada no servidor na rede fixa e outra na unidade móvel. Esta interceptação é transparente tanto para o cliente quando para o servidor, oferecendo flexibilidade e poder de gerenciamento de desconexões.

O modelo Cliente–Interceptador–Servidor é indicado quando a aplicação exige alto poder de processamento e armazenamento, além de maior autonomia de energia. Melhora a compressão de dados, otimiza a comunicação sem fio e o tratamento de desconexões, devido à existência dos dois agentes interceptores.

Um dos problemas deste modelo, representado na figura 7 da próxima página, é que cada aplicação requer processamento tanto no cliente quanto no servidor.

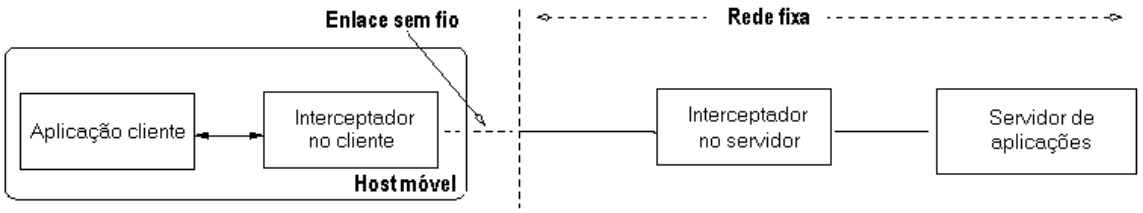

Figura 7. Modelo Cliente – Interceptador – Servidor (adaptada de [48])

2.5.4 Modelo Peer-to-Peer (P2P)

No modelo Peer-to-Peer, mostrado na figura 8, não há distinção entre cliente e servidor. Estações móveis podem assumir a função de clientes e a comunicação pode ser realizada com a estação base que estiver disponível.

Esse modelo é mais adequado quando há uma forte conexão com o servidor, fazendo com que a comunicação seja realizada com maior facilidade e se torne mais acessível do que a realizada com o servidor. Contudo, os custos de comunicação podem ser elevados, no caso das unidades móveis estarem fisicamente distantes do servidor, ou não existir um canal de comunicação entre os mesmos.

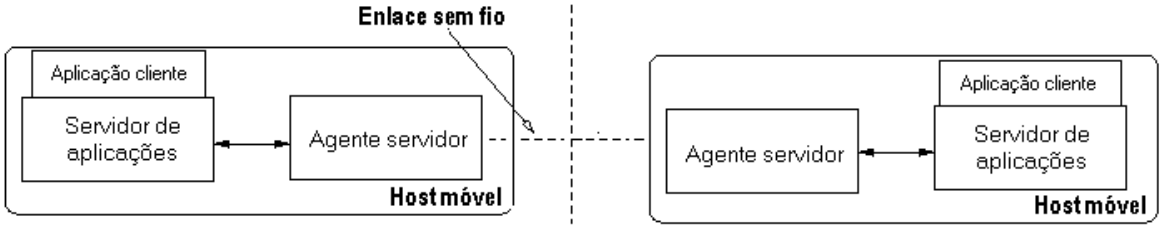

Figura 8. Modelo Peer-to-Peer (adaptada de [48])

2.5.5 Modelo de agentes móveis

Este modelo trata a questão da mobilidade permitindo o trabalho desconectado. Pode ser combinado com outros modelos, porém apresenta restrições econômicas e de segurança.

Agentes móveis são programas enviados de uma máquina (cliente) para realizar uma determinada tarefa remotamente (servidor). Possuem instruções, dados e um estado de execução. Além disso, esses elementos possuem características inerentes ao conceito de multi-agentes, que proporcionam um bom desempenho em sistemas de objetos distribuídos. Desta forma, características como cooperação, autonomia, representatividade, entre outras, foram acopladas neste modelo com a finalidade de

suprir as necessidades exigidas para o bom funcionamento de modelos que utilizam esse paradigma. A seguir, são descritas algumas características do modelo de agentes móveis [48]:

- Objetos passantes: quando um agente móvel é transferido, todo o objeto é movido, ou seja, o código, os dados, o itinerário para chegar ao servidor necessário, o estado de execução, etc;
- Assincronismo: o agente móvel possui seu próprio *thread* de execução e este não precisa ser executado de modo síncrono;
- Interação local: o agente móvel interage com outros agentes móveis ou com objetos estacionários locais. Se necessário, um agente mensageiro é despachado para facilitar a interação com agentes remotos;
- Desconexão: o agente móvel pode executar tarefas mesmo em modo desconectado. Quando se faz necessária a transferência de agentes, aguarda-se até que a conexão seja restabelecida.
- Execução paralela: múltiplos agentes podem ser enviados para diferentes servidores, com o objetivo de executar tarefas em paralelo.

Nos modelos onde há a presença de um agente entre o cliente e o servidor, muitas vezes se faz referência a uma camada imaginária onde estas entidades atuam. Esta camada é chamada de camada intermediária que, como será visto no capítulo 5, possui um papel importante nos sistemas gerenciadores de bancos de dados móveis. A arquitetura e o modelo que possuem uma camada intermediária são chamados de computação multicamada (*multi-tier*) ou em três camadas (*three-tier*).

Neste capítulo, foram introduzidos alguns conceitos da computação móvel e discutidas algumas características e obstáculos da tecnologia. Estas características deixam claro que as configurações de sistema deixam de ser estáticas, causando um impacto nos sistemas operacionais, nos protocolos e arquitetura de comunicação, na arquitetura de *hardware*, segurança, requisitos e projeto de aplicações, projeto de interface e também naquele que é um ponto fundamental deste trabalho, o gerenciamento de dados feito por um banco de dados móvel.

Todas estas dificuldades citadas anteriormente, uma vez inerentes à tecnologia móvel são também desafios de bancos de dados móveis. O próximo capítulo busca definir o conceito, exemplificar o uso e citar algumas das diferenças entre as tecnologias de bancos de dados móveis e a de bancos de dados distribuídos, bem como estudar algumas questões particulares relacionadas a esta área.

3. Introdução a Bancos de dados móveis

Um sistema de banco de dados móvel é aquele no qual o acesso à base de dados é realizado através de uma ligação sem fio [41]. Nas próximas seções, são apresentados exemplos do uso de bancos de dados móveis e, em seguida, é feito um breve paralelo entre esta tecnologia e a de bancos de dados distribuídos. Um estudo sobre as características do projeto de bancos de dados móveis finaliza o capítulo.

3.1 Exemplos de uso de bancos de dados móveis

A seguir, são apresentados alguns exemplos de aplicações de bancos de dados que existem ou poderão existir, a fim de deixar claros os recursos inovadores introduzidos pela tecnologia.

***Exemplo 1*:** um dos mais citados exemplos de aplicações de computação móvel em banco de dados é o da consulta dependente de localidade. Imagine-se uma família em viagem de férias a um estado desconhecido. As bases de dados sensíveis à localidade manterão informações a respeito de hotéis, restaurantes e outros serviços locais necessários (hospitais, por exemplo). Consultas dependentes de localização acessarão informações que responderão a perguntas como: "Qual o hotel mais próximo com piscina?", "Como chegar ao hospital mais próximo?" E essas perguntas poderão ser feitas repetidamente produzindo uma resposta diferente em cada uma das vezes, pois dependerá da localização de quem requisita a informação.

***Exemplo 2*:** vários vendedores podem fazer o uso de seus laptops para manter informações sobre seus clientes e seus pedidos. Quando o vendedor vai ao encontro do cliente, pode querer acessar os dados sobre as preferências desse cliente. Num sistema distribuído, essa consulta seria direcionada ao local em que a informação procurada encontra-se armazenada. Num ambiente de banco de dados móveis, além dessa opção, seria possível acessar uma cópia desses dados armazenados numa *cache* local.

***Exemplo 3*:** o terceiro exemplo se baseia numa cidade de tráfego intenso onde muitas vezes se passa uma hora para sair de uma localidade à outra. Numa agência de seguros de automóvel existe um funcionário que precisa avaliar os danos fisicamente de cada automóvel coberto por aquela seguradora e compor um relatório estimando os custos para os reparos necessários. Para que tal relatório seja terminado, é necessário

o acesso a informações do automóvel, a relatórios de perícia da polícia rodoviária, uso anterior do seguro por este carro, entre outras. Ao ligar seu computador móvel no início de um dia de trabalho, as informações sobre o próximo caso a ser analisado já podem ter sido pré-carregadas. O avaliador se conecta com o banco de dados da empresa e faz a requisição das informações que estão faltando. A recepção desses dados acontece numa transação enquanto o avaliador está a caminho do local (já que o exame físico ainda é necessário). Ao chegar lá, o avaliador entra os dados restantes no seu computador móvel que automaticamente atualiza os dados na rede fixa, onde outros processamentos ocorrem. A rede fixa reenvia então os dados processados para o avaliador que já estará a caminho da localidade de uma nova ocorrência. Durante esta transação, o avaliador pode querer desligar seu dispositivo móvel, a fim de economizar bateria e parar para almoçar. Quando torna a ser ligado, a unidade móvel retoma a transação de onde ela tinha parado. Num dia de trabalho, a produção do profissional terá claramente aumentado [10].

3.2 Bancos de dados móveis × distribuídos

Um sistema de computação distribuída consiste de um número de elementos computacionais, não necessariamente homogêneos, que são interconectados por uma rede de computadores e que colaboram entre si na execução de determinadas tarefas a eles atribuídas [13]. Um banco de dados distribuído é uma coleção de bases de dados espalhadas por computadores que fazem parte de uma rede num ambiente de computação distribuída. A figura abaixo ilustra a arquitetura de bancos de dados distribuídos.

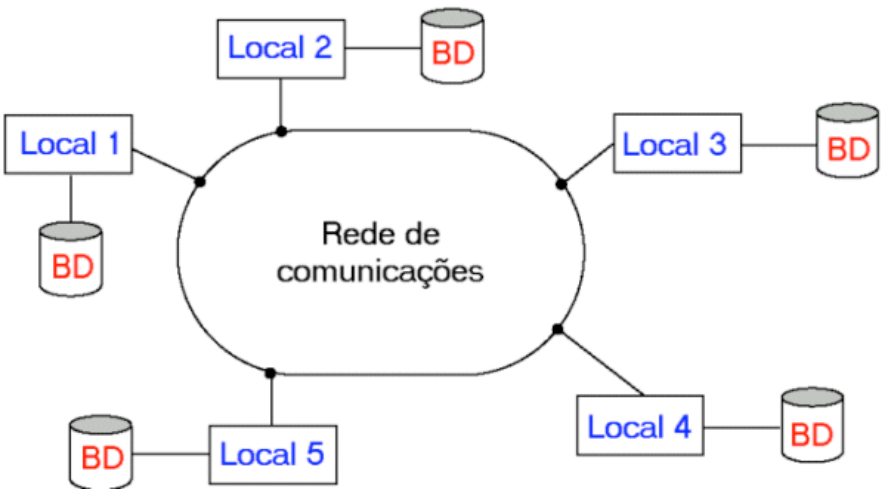

Figura 9. Arquitetura de bancos de dados distribuídos (figura adaptada de [13])

Do ponto de vista de gerenciamento de dados, a computação móvel pode ser considerada uma variação da computação distribuída. Bancos de dados móveis podem ser distribuídos em dois possíveis cenários [13]:

1. Toda a base de dados está distribuída principalmente entre os componentes ligados por fiação, possivelmente com replicação total ou parcial dos dados. Uma estação base gerencia sua própria base de dados com um SGBD com funcionalidades adicionais para localizar unidades móveis e para gerenciar consultas e transações do ambiente móvel. Badrinath e Phatak [1] propuseram uma nova arquitetura para SGBDs para clientes móveis;
2. A base de dados é distribuída pelos componentes com e sem fio. A responsabilidade do gerenciamento dos dados é compartilhada entre as unidades móveis e as estações base.

Desta forma, muitas das questões de gerenciamento de dados distribuídos também se aplicam aos bancos de dados móveis, segundo Elmasri e Navathe [13], desde que se levem as seguintes considerações adicionais e variações em conta:

- Distribuição de dados e replicação – Os dados, como visto, podem estar desigualmente e irregularmente distribuídos entre unidades móveis e estações base. As restrições de consistência aumentam o problema de gerenciamento de dados em *cache*. Esta deve proporcionar à unidade móvel acesso àqueles dados mais freqüentemente acessados e atualizados. No exemplo 2 da seção anterior, quando o vendedor acessar os dados na *cache* local, estes devem estar consistentes com os dados da estação base;
- Modelos de transação – Questões de tolerância a falhas e corretude das transações são agravadas no ambiente móvel. A transação móvel é executada seqüencialmente através de várias diferentes estações base e possivelmente em múltiplos conjuntos de dados, dependendo da movimentação da unidade móvel. A coordenação central da execução de uma transação falta, por exemplo, quando se tem a base distribuída pelos componentes com e sem fio (cenário 2 acima). Portanto, as propriedades ACID (atomicidade, consistência, isolamento e durabilidade) das transações podem precisar de modificação e novos modelos de transação devem ser definidos;
- Processamento de consultas – A informação sobre a localização dos dados é importante e afeta a análise de custo benefício do processamento de consultas. As respostas às consultas precisam ser dadas às unidades móveis que mesmo estando em trânsito e cruzando células, devem receber completamente e corretamente tais

respostas. Na seção anterior, foi citado um exemplo de uma aplicação onde são feitas consultas sensíveis à localidade (exemplo 1 da seção anterior);

- Recuperação e tolerância a falhas – O ambiente de bancos de dados móveis pode apresentar falhas locais (queda da unidade móvel), falhas de mídia, de transação e de comunicação. Uma falha local ocorre freqüentemente devido à bateria fraca no computador móvel. Se uma unidade móvel suspende a conexão voluntariamente, isto não deve ser tratado como falha (ver exemplo 3 da seção anterior). Falhas de transação são mais freqüentes durante operações de *handoff*, ou seja, quando uma unidade móvel cruza células e a responsabilidade de comunicação com esta unidade é transferida de uma estação base para a outra. Falhas nas unidades móveis, por sua vez, causam particionamento da rede e afetam os algoritmos de roteamento;
- Projeto de banco de dados móveis – O problema da determinação do nome global da unidade (seu número de IP, por exemplo), para a boa manipulação de consultas, é comprometido por causa da migração de localidade da unidade e freqüente suspensão de comunicação. O projeto de banco de dados deve considerar várias questões de gerenciamento de metadados – por exemplo, a atualização constante da localização da informação.

3.3 Questões de bancos de dados na computação móvel

Na seção anterior, foram citados alguns fatos que caracterizam os bancos de dados móveis. Esta seção procura conceituar e fazer um estudo mais detalhado de algumas destas características individualmente.

3.3.1 Replicação de dados e sincronização

Replicação é o processo pelo qual um arquivo ou grupo de arquivos é copiado de seu local original para outras máquinas dentro de um sistema distribuído. Para aumentar a disponibilidade de dados e a performance de aplicações sobre estes dados em unidades móveis, os dispositivos carregam dados replicados, isto é, cópias redundantes das informações contidas no banco de dados distribuído. A replicação é uma técnica estratégica importante que otimiza os recursos de *hardware* e de rede, maximizando a flexibilidade e a escalabilidade em ambientes heterogêneos.

Sincronização é o processo pelo qual os dados distribuídos são mantidos atualizados, de forma que os usuários em cada local tenham certeza de que estão vendo a versão mais recente do arquivo. Quando os dados num servidor ou numa base e suas réplicas são sincronizados, o log de transação de cada volume replicado é usado

para verificar se aqueles dados foram atualizados, inseridos ou removidos da base central. As cópias mais recentes são então replicadas para todos os outros locais que acessam esses dados [39].

Portanto, quando uma aplicação modifica dados compartilhados em uma determinada base de dados, as mudanças são propagadas para as outras bases onde se localizam as réplicas. Estas mudanças podem ser propagadas através de vários tipos de protocolos ou canais de comunicação. Após tal propagação, os dados terão sincronizado suas modificações. Pela interdependência destes conceitos, eles são muitas vezes confundidos ou tratados indistintamente, o que é compreensível, já que não há sentido em replicar dados numa base distribuída se não houver sincronização.

A base de dados que contém os dados a serem replicados é chamada de consolidada. As bases remotas são aquelas que contêm um subconjunto das informações da base consolidada (figura 10) e geralmente se localizam distantes fisicamente desta.

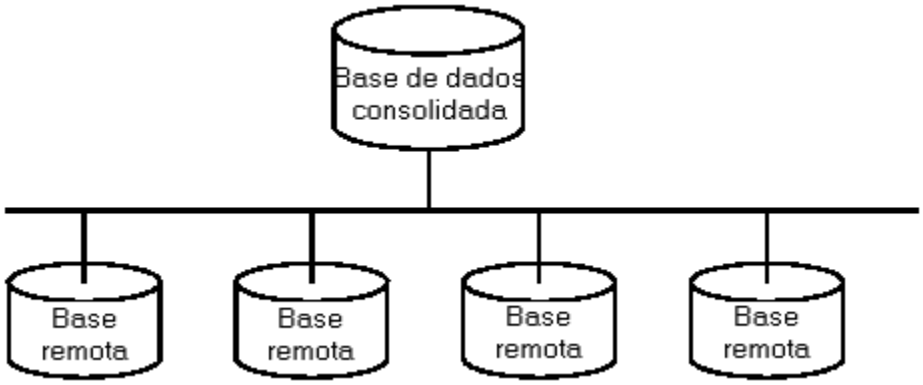

Figura 10. As bases remotas possuem réplicas dos dados da base consolidada.

Quanto aos métodos de propagação usados, a replicação de dados é dividida em: baseada em sessão (*session-based*), baseada em mensagens (*message-based*) ou baseada em conexão (*connection-based*) [64].

- Replicação baseada em sessão: neste esquema, a replicação ocorre através uma linha de comunicação direta entre a base consolidada e a base remota, dependendo da configuração inicial, os intervalos entre duas destas sessões podem ser de minutos, horas, dias ou semanas. A base remota cliente abre a comunicação com o servidor de sincronização, enviando uma lista completa das modificações feitas naquela base desde a última sincronização. O servidor, então, atualiza a base

consolidada e envia de volta as modificações relevantes. O *host* remoto incorpora o conjunto de mudanças e fecha a sessão logo após enviar uma confirmação de sucesso da operação junto com uma tabela que mapeia o contendo das informações que foram atualizadas (*mapping table*);

- Replicação baseada em mensagens: a troca de dados entre as bases pode ocorrer também através de mensagens, que normalmente são arquivos armazenados em algum diretório particular ou mensagens de correio eletrônico com formato especial. Um agente de mensagens vinculado a cada base envia mensagens, informando as mudanças recentes em sua base e recebe mensagens de um ou mais outros agentes, modificando seus dados de acordo com o conteúdo delas. Este sistema permite replicação entre bancos de dados que não estão diretamente conectados. Neste tipo de replicação, cada uma das mensagens carrega informações de controle como o seu endereço de destino para que nenhuma conexão seja necessária entre as aplicações que se comunicam remotamente;
- Replicação baseada em conexão: algumas tecnologias de replicação dependem de uma conexão contínua, ou praticamente contínua, entre as bases remotas e a base consolidada. Este tipo de replicação é caracterizado por ser rápido e confiável, porém de manutenção cara, já que necessita de muitos recursos. Este tipo de esquema é, desta forma, mais apropriado para replicações entre bases consolidadas.

Um dos desafios enfrentados por sistemas que necessitam de sincronização é a falta de padronização do protocolo. Em fevereiro de 2000, empresas como a Ericsson, IBM, Motorola e Nokia, entre outras se reuniram para começar a resolver o problema. O primeiro e importante passo foi a criação do SyncML [65], um padrão aberto para a sincronização universal de dados e informações pessoais entre vários tipos de redes, plataformas e dispositivos. O SyncML especifica 7 diferentes tipos de sincronização [42]:

1) Em dois sentidos (*Two-way sync*), o tipo mais comum de sincronização, onde cliente e servidor trocam informações sobre as modificações ocorridas em suas bases (ver os passos de uma replicação baseada em sessão, acima);

2) Lenta (*Slow sync*) é uma forma de sincronização onde todos os itens da base de dados do cliente são comparados com os da base consolidada campo a campo. Pode ser requisitada pelo cliente após uma perda do seu log de operações, por exemplo;

3) Partindo do cliente (*One-way sync from client only*): neste tipo de sincronização, apenas o cliente envia suas modificações ao servidor, aquelas ocorridas apenas na base consolidada não são enviadas ao cliente;

4) Atualização a partir do cliente (*Refresh sync from client only*): aqui, o cliente exporta todos os seus dados para o servidor, que deve então substituir os dados da base alvo por eles;

5) Partindo do servidor (*One-way sync from server only*): o cliente recebe todas as modificações ocorridas no servidor sem enviar as que ocorreram em sua base;

6) Atualização a partir do servidor (*Refresh sync from server only*): neste cenário, o servidor exporta seus dados e exige que o cliente troque os dados da base alvo por eles;

7) Sincronização alertada pelo servidor (*Server-alerted sync*): aqui, o servidor informa ao cliente que um típico específico de sincronização deve ser iniciada entre eles.

Os principais benefícios de se fazer replicação de dados num ambiente móvel estão relacionadas à disponibilidade de dados localmente, ou seja, são os dados armazenados na *cache* local (ver 3.3.2) das unidades móveis que possibilitam operações desconectadas ou com fraca conexão e aumentam o tempo de resposta das requisições de informações, já que não há a necessidade de se conectar a bases distantes. As bases fixas também têm um ganho de performance com a replicação, já que em alguns casos não é necessário requisitar seu poder de processamento.

Entre os desafios de um sistema de replicação de dados está o de assegurar que a integridade de transações seja sempre mantida em cada banco de dados (ver 3.3.4) e assegurar que haja consistência dos dados replicados.

Nem sempre os dados necessários a uma aplicação estarão disponíveis na *cache* local, por isso, as unidades móveis, geralmente apresentam a necessidade de receber dados de uma base remota. Algumas formas de envio destes dados já foram citadas anteriormente. Uma importante abordagem é a difusão de dados, onde uma estação base envia informações para todas as unidades móveis localizadas em sua célula de cobertura.

3.3.2 Caching e difusão de dados

O *caching* e a difusão são dois dos tipos de paradigmas de acesso aos dados móveis. Ambos visam aumentar a disponibilidade de dados ao usuário móvel que se encontra numa base remota.

Como já foi visto, são inúmeros os benefícios de se armazenar os dados em *cache* (em disco local), para a melhor eficiência das operações. Esta abordagem é essencial na economia da baixa de largura de banda de uma rede sem fio.

Para coordenar a classificação dos dados quanto ao seu uso na *cache*, são usadas duas regiões distintas, definidas como região quente e região fria [25]. A região quente contém o conjunto de dados que são utilizados mais freqüentemente e a fria, o conjunto de dados utilizados com menos freqüência.

O tipo de *caching* onde o cliente organiza os dados em sua *cache* de acordo com propriedades destes dados (descrição semântica) e não através de listas de páginas e tuplas é chamado de *caching* semântico e proporciona um ganho de performance nesta operação.

Quando os dados consultados por uma aplicação não se encontram na *cache*, é necessária uma solicitação de dados ao servidor. Esta solicitação pode ser total, quando o cliente precisa receber todos os dados do servidor ou parcial, quando alguns dados na base remota podem ser utilizados para responder à consulta [25].

Segundo Korth e Silberchatz [60], existem dois motivos para se usar difusão de dados. Primeiro, a unidade móvel evita o gasto de energia com a transmissão de solicitação de dados. Segundo, os dados difundidos podem ser recebidos por um grande número de unidades móveis de uma só vez, sem um custo adicional, o que é ideal para um grupo de clientes com as mesmas necessidades de atualizações.

As estratégias de envio de mensagens por difusão se classificam em *pull-based*, quando o cliente faz o papel ativo, requisitando dados ao servidor e recebendo estes dados em seguida e *push-based*, quando a iniciativa de transmissão de dados é do servidor e o cliente funciona apenas como receptor (figura 11).

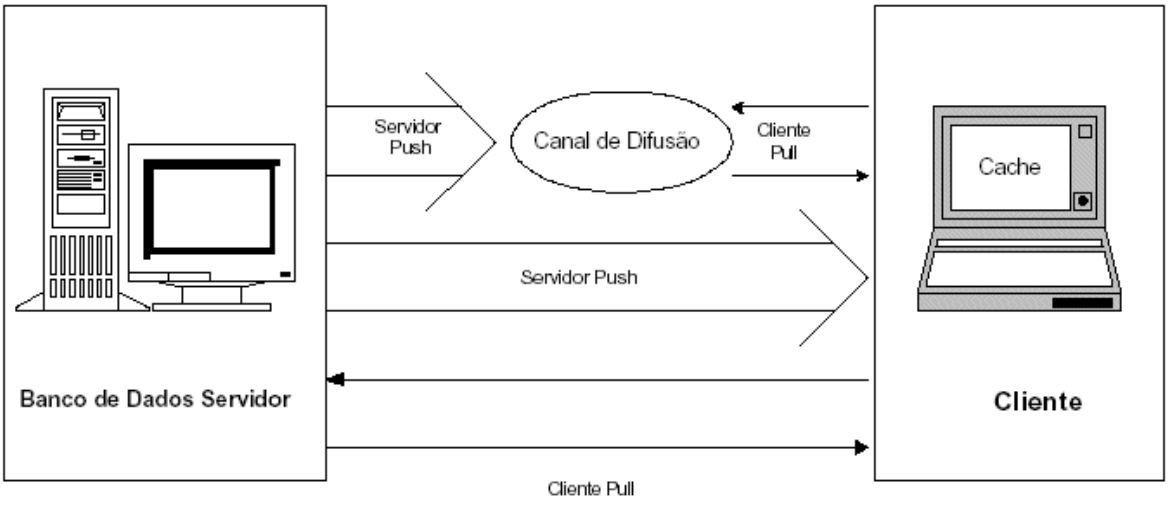

Figura 11. Estratégias de transmissão por difusão *pull-based* (iniciativa do cliente) e push-based (iniciativa do servidor) (figura retirada de [25]).

Ao usar *caching*, uma estratégia de invalidação de *cache* (*cache invalidation*) é necessária para que se certifique que os dados em um computador móvel são os mesmos dos da base consolidada, ou seja para assegurar a consistência ou coerência de *cache*. De acordo com o estado da *cache* do cliente, as estratégias de invalidação são classificadas em dois tipos: servidor com estado e sem estado [2].

Na primeira categoria, o servidor está ciente de que dados estão armazenados em *cache* na unidade móvel, ou seja, conhece o estado do cliente (*stateful server*). Uma vez que os dados são modificados na base consolidada, o servidor envia mensagens de invalidação ao cliente que possui réplicas daqueles dados. O problema aqui reside na complexidade em localizar o cliente, fazendo com que o cliente tenha que informar ao servidor quando migrar.

Na segunda categoria, o servidor nada sabe sobre o estado da *cache* do cliente (*stateless server*), na verdade, o servidor não sabe que computadores móveis estão ativos. Para manter a consistência dos dados nas bases remotas, periodicamente, o servidor envia por difusão um relatório de invalidação contendo os itens atualizados recentemente. Os clientes móveis recebem o relatório e deixam sua base consistente, se for necessário.

Wu, Yu e Chen [72] propuseram um esquema de invalidação de *cache* que checa a validade da *cache* com o servidor no caso de uma reconexão. Este esquema leva em conta um grupo de objetos também da região fria da *cache*, além dos da região quente, como outros esquemas anteriores a este. É apresentada ainda uma análise de desempenho e o esquema se mostra adaptável à energia restrita dos dispositivos móveis.

Para um estudo mais detalhado de *caching* e difusão de dados, ver Ito [25].

3.3.3 Gerenciamento de localidade

Na computação móvel, objetos como *softwares* ou usuários que usam aparelhos sem fio podem se movimentar de um local para o outro da rede. Informações sobre o atual posicionamento destes objetos devem ser mantidas em locais específicos da rede. Este gerenciamento de localidade envolve duas operações básicas: busca e atualização [49].

Uma busca é solicitada cada vez que há a necessidade de localizar um objeto, para se comunicar com usuários móveis ou executar aplicações remotamente. A atualização da localidade armazenada de um objeto é necessária quando este migra de um local para outro dentro da rede ou para uma outra rede.

As abordagens de armazenamento de informações sobre localidade variam entre dois extremos. Num extremo, a informação atualizada da exata localização da unidade móvel é mantida em cada ponto da rede que acessa suas informações. Isto reduz o tempo de uma consulta nesta unidade móvel, pois sua localização é imediata. A desvantagem desta abordagem é que muitas atualizações são necessárias. No outro extremo, nenhuma informação é armazenada na rede a respeito da localização da unidade móvel, eliminando portanto o custo de atualização desta informação. Aqui, a desvantagem é o custo da busca do objeto que tem que ser feita globalmente na rede.

Pitoura e Samaras [49] fazem uma pesquisa sobre várias abordagens ao problema de armazenar, consultar e atualizar a localidade de objetos na computação móvel. Nestes estudos deve-se levar em conta 3 fatores determinantes a respeito da informação da localização destes objetos: sua disponibilidade, precisão e atualidade (figura 12).

Em termos de disponibilidade, as possibilidades de projeto variam entre dois pontos: salvar a informação da localização da unidade móvel em todas os pontos da rede e não disponibilizar esta informação em nenhum local da rede. Há um grande número de critérios para escolher em que pontos da rede salvar a informação, por exemplo, é interessante manter os dados sobre a localização de uma unidade móvel nos locais da rede que acessem mais vezes sua base de dados.

A respeito da precisão da informação, ao invés de manter sua exata localização, os servidores podem manter um conjunto de possíveis localidades do cliente, a partir do seu histórico. Há ainda a possibilidade de se configurar uma área maior para o que se chamará localidade do cliente.

Atualidade se refere a que freqüência ocorrem atualizações a respeito da localidade do cliente. Por exemplo, para usuários altamente móveis, pode-se abrir mão de atualização da informação todas as vezes que eles mudam de localidade.

A figura 12, na página seguinte, mostra um cubo dentro do qual estão as várias possibilidades de abordagem de gerenciamento de informação de localidade.

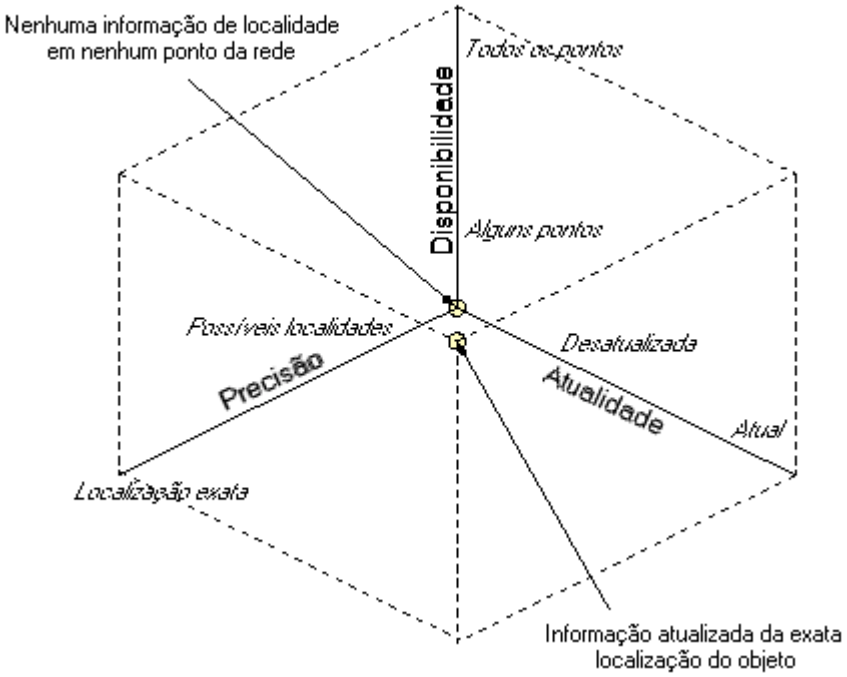

Figura 12. Cubo que ilustra a relação dos fatores determinantes para um bom gerenciamento da informação da localidade de um dispositivo móvel (figura adaptada de [49])

Dois esquemas são comumente usados para ilustrar as arquiteturas de bancos de dados distribuídos que armazenam a localidade de uma unidade móvel: em duas camadas (*Two-tier scheme*) e hierárquico.

No esquema de duas camadas, um banco de dados possui o Registro de Local de Origem (*Home Location Register*, *HLR*) que é previamente associado a cada unidade móvel e mantida na zona que é especificada para o usuário. A busca e atualização se tornam simples, pois para localizar um cliente, basta consultar seu *HLR*. Já existem melhoras neste esquema como a existência de Registros de Localidade de Visitantes (*Visitor Location Register*), que armazena cópias de informações de usuários que não pertencem àquela zona e que atualmente a estão visitando.

O esquema de localização hierárquica estende o de duas camadas, mantendo uma hierarquia da localização das bases de dados distribuídas. Nesta hierarquia, normalmente uma árvore, uma base num nível superior contém informações da localidade dos usuários das bases que estão nos níveis abaixo. Este esquema reduz os custos de comunicação, já que a maioria dos acessos acontece na mesma região geográfica. No entanto, há um número muito maior de localizações que são

atualizadas e consultadas em comparação com o esquema de duas camadas e bancos de dados localizados nos níveis mais altos da hierarquia devem gerenciar um número muito maior de mensagens e ter grande capacidade de processamento e armazenamento.

Os dados num banco de dados móvel são classificados em: dependentes de localidade e independentes de localidade. As consultas portanto, neste ambiente, podem ser dependentes da localidade ou cientes da localidade (*Location-Aware Queries*) Seydim e Dunham [59] discutem as diferenças entre estes dois tipos de consulta e propõem e apresentam uma formalização das consultas sensíveis à localidade.

Uma consulta que procure o nome e estado civil de um usuário é obviamente independente de localidade. Por outro lado, pode-se precisar consultar informações que dependam do local em que se encontra um cliente como, por exemplo, qual a distância da estação de metrô mais próxima de onde ele se encontra. Estas consultas são ditas sensíveis à localidade e dependem de um bom gerenciamento de localidade do cliente, pois precisam levar em conta fatores como trajetória e velocidade do cliente que está se movendo, tempo de processamento da consulta e execução da transação, além das restrições já discutidas anteriormente como falta de recursos do dispositivo, fraca conectividade, etc.

O subsistema de processamento de consultas deve ainda levar em conta o tipo de estratégia de gerenciamento de localidade que é usado, pois para cada um há um custo associado. Kottkamp e Zukunft [26] estudam estratégias de otimização de consultas em ambientes móveis e propondo e validando um modelo de otimização, onde são considerados: o monitoramento da localidade da informação, a escolha do local onde será executada a consulta e fatores como custos de comunicação, uso de energia, processamento e memória associados.

Há também estudos como o de Perich [47], que discutem consultas sensíveis à localidade no ambientes de redes móveis ad-hoc.

3.3.4 Transações

Uma transação é uma unidade lógica de processamento de banco de dados que inclui uma ou mais operações de acesso à base. Entre estas operações estão inserção, exclusão, modificação e consulta de dados [13].

A concorrência de transações que acessam à mesma base de dados pode levar a problemas como a perda de consistência dos dados. Portanto, transações têm que apresentar algumas propriedades desejáveis a fim de evitar estes erros.

O controle de concorrência e os métodos de recuperação de falhas devem assegurar estas que são chamadas de propriedades ACID, que são as seguintes [13]:

1. Atomicidade: uma transação deve ser uma unidade atômica de processamento sendo completamente executada ou descartada. O subsistema de recuperação de transações deve assegurar que se uma falha ocorrer durante a execução da transação, nenhuma das modificações feitas por ela persistam;

2. Consistência: a transação deve levar a base de dados de um estado consistente a outro. O estado do banco de dados é uma coleção de todos os itens de dados armazenados num dado momento. Um estado consistente é aquele que satisfaz as restrições impostas pelo esquema do banco de dados, assim como outras que forem especificadas.

3. Isolamento: assegurado pelo subsistema de controle de concorrência. A execução de uma transação não deve ser interferida por outra que ocorra concorrentemente no sistema. Há diferentes níveis de isolamento possíveis;

4. Durabilidade (persistência): depois da transação completar-se com sucesso, as mudanças operadas por ela na base de dados persistem, até mesmo se houver falhas no sistema.

O sistema de banco de dados deve controlar a execução concorrente de transações para assegurar que o estado do banco de dados permaneça consistente. Podem ocorrer alguns conflitos de operações de transações que ocorrem concorrentemente, por isso técnicas como serialização e agendamento de transações se fazem necessárias para evitar conflitos [60].

Uma transação móvel é aquela da qual pelo menos um host móvel faz parte de sua execução [56]. Dunham e Kumar discutem em [12] os efeitos do impacto que a mobilidade traz às transações de bancos de dados.

A desconexão de estações móveis possivelmente por um longo período e limitações na largura de banda exigem uma séria reavaliação do modelo de transações e das técnicas de processamento de transação. Muitos modelos e diferentes abordagens para a execução de transações móveis foram propostos. A maioria destes estudos considera uma transação móvel como parte de uma transação onde há a flexibilização na consistência e submissão de modificações (*commitments*). O gerenciamento destas transações pode acontecer apenas na unidade móvel, apenas no servidor ou pode ir de estação base em estação base, de acordo com a migração do cliente móvel.

Uma perda de conexão com a rede não deve ser considerada uma falha e se os dados e métodos necessários para que se complete uma tarefa ainda estiverem no dispositivo móvel, o processamento deve continuar apesar da desconexão. Técnicas tradicionais de serialização como monitores, agendamento (*scheduling*) e bloqueios (*locks*) não funcionam apropriadamente no ambiente desconectado, portanto novos mecanismos deve ser desenvolvidos para o gerenciamento de transações móveis [57].

Como usuários normalmente precisam executar tarefas mesmo na falta de conexão com servidor, os dispositivos móveis necessitam de algum tipo de gerenciamento de transações. Portanto, esquemas de controle de concorrência em bancos de dados distribuídos devem dar suporte à operação autônoma de dispositivos móveis desconectados. Estes esquemas devem ainda considerar o tráfego de mensagens em vista das limitações de largura de banda do ambiente móvel e o a localidade do cliente.

Estas questões discutidas foram estudadas por muitos pesquisadores e modelos foram propostos para transações móveis, como o Kangaroo [11] que executa as transações no servidor e o PRO-MOTION [6], que reparte a execução das transações entre o servidor e o cliente móvel. Como o ambiente móvel exige adaptação não há um consenso sobre qual modelo é o melhor. Serrano-Alvarado [56] e Seydim [57] publicaram estudos comparativos de vários modelos propostos.

3.3.5 Recuperação de falhas

O processo de recuperação se refere à capacidade que um sistema tem de preservar a consistência do banco de dados após falhas do sistema, de transações ou dos meios de comunicação. Esta seção transcreve o estudo feito por Ito [25] a respeito de recuperação de falhas em bancos de dados móveis.

Um sistema em um ambiente móvel está sujeito a falhas tanto de *hardware* como de *software*. Em cada um destes casos, informações que se referem ao sistema de banco de dados podem ser perdidas. Uma parte essencial do sistema de banco de dados é um esquema de recuperação responsável pela detecção de falhas e pela restauração do banco de dados para um estado consistente que existia antes da ocorrência da falha.

Para que o sistema não seja prejudicado devido às falhas em seus componentes, erros devem ser detectados o mais rápido possível, através de um diagnóstico apropriado. Para recuperar os dados, informações relevantes são armazenadas em um local fixo durante o processamento de transações.

Em sistemas distribuídos, a recuperação de falhas é baseada em pontos de recuperação, conhecidos como *checkpoints*. No caso de falhas e desconexões, a aplicação usa o último *checkpoint* salvo para reiniciar sua execução [48].

A unidade móvel deve estar sempre informada sobre em qual célula se encontra e quando o sistema entrará em modo de desconexão, pois ela é responsável por gerar um *checkpoint*, caso mude de célula ou haja uma desconexão.

Um *checkpoint* global consiste de um conjunto de *checkpoints* locais. Os estados globais são armazenados e usados pelo protocolo para recuperar falhas das aplicações. Um estado global inclui o estado de cada processo que faz parte da aplicação distribuída e possivelmente algumas mensagens. Para uma recuperação correta, o protocolo pode salvar um estado consistente global.

Neves e Fuchs [36] propõem um tipo de protocolo com características especiais para ambientes móveis. Este protocolo armazena o estado global consistente da aplicação sem troca de mensagens. Os processos de *checkpoint* podem ser armazenados no servidor ou localmente nos *hosts*. O armazenamento local não consome largura de banda e leva menos tempo para ser criado, no entanto, podem ocorrer perdas durante desconexões da unidade móvel.

Durante a execução da aplicação, o protocolo mantém o estado global em um local fixo para recuperar falhas permanentes (falhas de *hardware*) e outro em unidades móveis para recuperar falhas temporárias (falhas de *software*).

Algumas características de um esquema de *checkpoint* em um ambiente móvel, como localização, desconexão, energia, rede e falhas são analisadas a seguir.

- Localização – Para que exista a comunicação entre a rede fixa e a unidade móvel, é necessário saber a localização exata desta última. A localização de uma unidade móvel dentro de uma rede não é constante devido à sua mobilidade. É necessário que se localize um dispositivo móvel para depois determinar seu *checkpoint*. O protocolo *checkpoint* tem a função de armazenar o estado do processo em um local conhecido e de fácil acesso ou em um computador próximo da localização atual da unidade móvel. O protocolo pode ainda manter a trilha dos lugares onde o estado do processo foi salvo;
- Desconexão – Uma unidade móvel é dita desconectada quando está fora da faixa de transmissão das estações base. Durante a desconexão, todas as informações armazenadas em locais remotos não estão acessíveis às unidades móveis. Desta forma, uma unidade móvel fica impossibilitada de enviar ou receber mensagens. A criação de *checkpoints* por processos enquanto a unidade móvel está desconectada é possível. Por outro lado, os protocolos têm a função de salvar uma grande quantidade

de informações que garantam sua execução após uma falha. Estas informações incluem a ordem da recepção e os conteúdos das mensagens. Quando as informações precisarem ser salvas nas unidades móveis, poderá haver limitação de memória e de espaço para o armazenamento e devido às freqüentes desconexões das unidades móveis, o local do *checkpoint* pode não estar disponível;

- Energia – Devido às restrições oferecidas pelo tempo das baterias utilizadas pelos PDAs, o consumo das mesmas deve ser reduzido. O protocolo *checkpoint* tem a função de reduzir a quantidade de informações que são adicionadas às mensagens e evitar o envio de mensagens extras, realizando o menor número de acessos possível ao servidor;
- Rede – A diversidade das tecnologias utilizadas nas redes sem fio faz com que as características da rede não sejam constantes, tais como, custos, velocidade e latência. O protocolo *checkpoint* deverá se adaptar às características da rede onde está atuando;
- Falhas – Unidades móveis podem apresentar falhas de *software* e de *hardware*. Falhas de *software* podem ocorrer quando acontecem problemas com o sistema operacional, descarga de bateria, entre outros. São gerenciadas por *soft checkpoints*, que são armazenados localmente na unidade móvel. Falhas de *hardware* ocorrem quando, por exemplo, o computador portátil é roubado, danificado, etc. São gerenciadas por *hard checkpoints*, que são armazenados na rede fixa. O protocolo *checkpoint* deve fornecer mecanismos para tolerar os tipos de falhas que possam vir a acontecer com as unidades móveis.

3.3.6 Segurança

Os riscos de segurança de sistemas e de redes de computadores são agravados quando inseridos no ambiente móvel, muito mais propenso a ataques e falhas. Na computação móvel, a portabilidade dos dispositivos usados pode levar à perda das unidades, o que caracteriza perda de dados e de confidencialidade.

A única forma de prevenir a falta de confiabilidade é o uso de encriptação (*encryption*, fase da criptografia) e de mecanismos que assegurem identificação, autenticação e controle de acesso. Estas não são características particulares de segurança num ambiente móvel, a não ser pelo fato de que a proteção de um dispositivo móvel deve ser simplificada, devido à escassez de recursos e pouco poder de processamento destes computadores.

Como foi visto, para contornar a falta de poder computacional, as unidades móveis transferem certas tarefas para os servidores, o que impede o nível de

isolamento computacional e de comunicação muitas vezes requeridos para que se assegure um nível razoável de segurança.

A heterogeneidade do ambiente móvel é outro tipo de desafio. O usuário que freqüentemente migra para outras localidades se submete a diferentes tipos de esquema de segurança.

Em [28], Lubinski estuda questões de segurança de dados e metadados na transferência, gerenciamento e acesso a bancos de dados móveis. Em [70], Villate discute um serviço na camada intermediária proposto para segurança dos dados de usuários móveis, o *Lockers Rent Service*, que guarda os dados dos clientes na rede fixa e possui arquitetura baseada em agentes móveis, tornando os dados disponíveis mesmo em caso de desconexão.

3.4 Outras abordagens de bancos de dados móveis

Distinções e particularidades fazem com que alguns sistemas de bancos de dados móveis sejam denominados de forma diferente e estudados como casos à parte. Por exemplo, levando em conta as conexões intermitentes, onde apenas o cliente pode estabelecer comunicação com o servidor e não esta iniciativa partir do cliente, Yee [74,75] publicou estudos sobre escalabilidade e replicação em bancos de dados intermitentemente sincronizados.

Sistemas multibancos de dados (*multidatabase systems*) são tipos de bancos de dados distribuídos que não possuem um esquema global e constrói um interativamente quando é requerido pela aplicação [13]. Vlach [71] discute procedimentos de bancos de dados móveis num ambiente distribuído de multibancos de dado e questões envolvidas a respeito do MDBAS, o protótipo que é avaliado no artigo. Dirckze [9] discute uma técnica de gerenciamento de transações móveis em multibancos de dados. Para uma melhor visão das arquiteturas destes tipos de bancos de dados móveis, os trabalhos citados também são indicados.

Embora muitos conceitos estudados neste trabalho também se apliquem a estas diferentes abordagens, o foco principal do estudo foram os sistemas baseados no modelo cliente servidor e suas variações e onde pode haver comunicação partindo da unidade móvel ou da rede fixa.

Na busca de uma estrutura padrão que suportasse estas características de bancos de dados móveis, estudos em empresas e principalmente projetos de pesquisa universitários foram desenvolvidos. O próximo capítulo lista alguns destes projetos, e sugere linhas de pesquisa promissoras e questões em aberto na área.

4. Pesquisas e projetos acadêmicos relacionados

Certos projetos de gerenciamento de dados móveis e comunicações num ambiente móvel foram implementados e testados, alguns modelos foram propostos e pesquisas introduziram importantes conceitos na área. Todos contribuíram ou contribuem de alguma forma para o desenvolvimento da computação móvel e para a melhoria dos sistemas de bancos de dados neste âmbito. Nas próximas seções, três destes projetos serão abordados, o já consolidado Coda e dois importantes projetos acadêmicos em andamento que abordam o acesso a dados móveis, o SIDAM, no Brasil e o MobiSnap, em Portugal. Algumas tendências futuras em pesquisas e novas aplicações são também comentadas.

4.1 Coda

O Coda [54] é um projeto que começou a ser desenvolvido em 1987 pelo grupo de Ciência da Computação da Universidade de Carnegie-Mellon, EUA e desde então vem sendo exaustivamente estudado e citado na literatura. O Coda é um sistema de arquivos experimental, baseado no modelo de sistema de arquivos do UNIX, cujo objetivo é oferecer a clientes o acesso contínuo a dados de uma rede mesmo com falhas ou suspensões de conexão. A importância deste projeto reside no fato de nele ter sido introduzido o conceito de operação desconectada (ver 2.3.2). Mais recentemente, o mesmo grupo que lançou o Coda vem desenvolvendo o Odyssey, abordado na próxima seção.

Em cada cliente, existe um gerenciador de *cache* chamado *Venus*, que obtém dinamicamente os dados e os põe na *cache*. Para dar suporte à operação desconectada, o Venus trabalha em 3 estágios: armazenamento (*Hoarding*), emulação e reintegração [53].

Enquanto está conectado, o Venus permanece no estado de *hoarding*, assegurando-se que objetos críticos ao funcionamento de algumas aplicações estão sendo armazenados em *cache*.

Quando acontece uma desconexão, ocorre a entrada no estado de emulação, onde o gerenciador de cache fica até que a conexão com o servidor seja restaurada. Desconectado, o Venus atende requisições do sistema de arquivos através do conteúdo

da *cache* se os dados requisitados não estiverem disponíveis localmente, aplicações e usuários enxergarão este fato como uma simples falha.

A persistência das mudanças feitas durante a emulação do Venus é garantida por um log de operações, o *Change Modify Log (CML)*.

Após a reconexão, o Venus entra no estado de reintegração, onde sincroniza seus dados com os dos servidores e passa novamente ao estado de *hoarding*. A figura 13 ilustra as mudanças de estado do Venus.

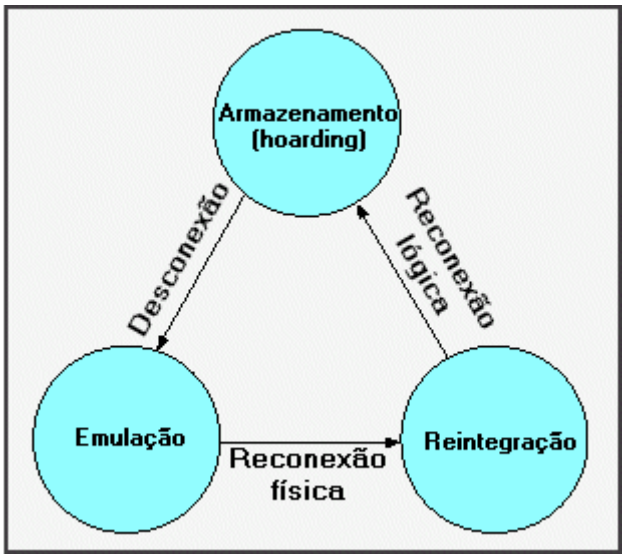

Figura 13. Estados de transição do Venus, o gerenciador de *cache* do sistema de arquivos Coda

O controle de replicação do Coda tem o objetivo de aumentar a disponibilidade, fazendo com que o controle de consistência de cache seja menos rígido (controle de replicação otimista).

O sistema de arquivos Coda tem boa portabilidade e independente de plataforma, pela simplicidade do seu kernel. Braam e Calahan [3] descrevem como o Coda foi portado para Windows 9x, depois de ter trabalhado satisfatoriamente em Linux, FreeBSD e NetBSD.

O Coda demonstrou a viabilidade da adaptação transparente a aplicações (ver 2.3.1). Todavia, há importantes situações nas quais este tipo de adaptação é inadequada, como a maioria das aplicações que envolvem dados multimídia como vídeos, mapas. O mesmo grupo de estudos do Coda propôs uma extensão do sistema operacional UNIX que suporta aplicações móveis e apresenta adaptação através da aplicação (*application-aware adaptation*), o Odyssey [53].

O Odyssey possui uma API (*Application Programming Interface*) que possibilita a negociação de recursos de aplicação genéricos como largura de banda, capacidade de *cache*, ciclos de processamento, tempo de vida de bateria ou específicos como número de consultas que um determinado grupo pode realizar.

No ano de 2000, foi iniciado o projeto chamado Aura [5], pelo mesmo grupo de pesquisas da Universidade de Carnegie Mellon. No Projeto Aura, estão sendo desenvolvidos arquiteturas, algoritmos, interfaces e técnicas de avaliação de desempenho para um sistema de computação móvel.

O Aura é composto de 4 principais elementos que executam sobre o kernel do Linux: o Coda e o Odyssey modificados e adaptados para prover acessos a arquivos por clientes móveis e monitorar recursos e adaptabilidade, respectivamente; o Spectra, um mecanismo adaptativo para receber chamadas para execução remota de aplicações e a grande novidade do Aura, o Prism, que é uma camada que está acima da camada de aplicação e se propõe a capturar e gerenciar a intenção do usuário. O Prism procura diminuir o nível de "distração" do usuário, ou melhor, a partir de técnicas de auto-ajuste e conhecimento do ambiente, tornar transparentes mudanças como variação de largura de banda e sinal.

Para mais informações e artigos relacionados aos projetos acima ver [4].

4.2 SIDAM

O SIDAM [23] (Sistemas de Informação Distribuídos para Agentes Móveis), mantido do departamento de Ciência da Computação do IME – USP, é uma cooperativa de pesquisadores que tem o objetivo de estudar os problemas relacionados ao projeto e implementação de sistemas de informação descentralizados utilizados tanto por usuários móveis quanto fixos.

Este projeto inicialmente apresenta um sistema *online* para a disseminação de informações de trânsito em uma área metropolitana, que serve como modelo de referência para a construção de uma ferramenta poderosa e flexível que seja dinamicamente adaptável ao ambiente móvel.

A arquitetura desta aplicação de referência é compreendida de três partes principais, que são:

1) Monitor: disponibiliza as ferramentas necessárias para monitorar as interações entre os objetos que compõem a aplicação distribuída, os recursos disponíveis no sistema, o gerenciamento de localidade dos dispositivos móveis, e a descoberta de novos serviços e recursos, à medida que o usuário se desloca com o seu

equipamento. Os dados coletados pelos componentes deste módulo são armazenados em *logs* ou enviados diretamente ao pacote de detecção de eventos;

2) Detector de eventos: disponibiliza as ferramentas necessárias para a avaliação dos dados coletados pela monitoração, de forma a detectar os eventos de interesse ao processo de adaptação. Disponibiliza ainda uma ferramenta de notificação destes eventos aos objetos que registraram interesses nos mesmos. O detector analisa os dados coletados identificando aqueles que demandam reconfigurações no sistema, se isto ocorrer, os eventos são enviados ao pacote de reconfiguração dinâmica;

3) Reconfigurador dinâmico: provê ferramentas através das quais pode-se definir as ações necessárias para reconfigurar a aplicação de forma a acomodar as mudanças ocorridas no ambiente. O reconfigurador dinâmico utiliza informações a respeito das dependências tanto estáticas quanto dinâmicas entre os componentes da aplicação, de forma a realizar a reconfiguração de forma segura.

A arquitetura do modelo proposto é ilustrada na figura 14 em diagrama UML.

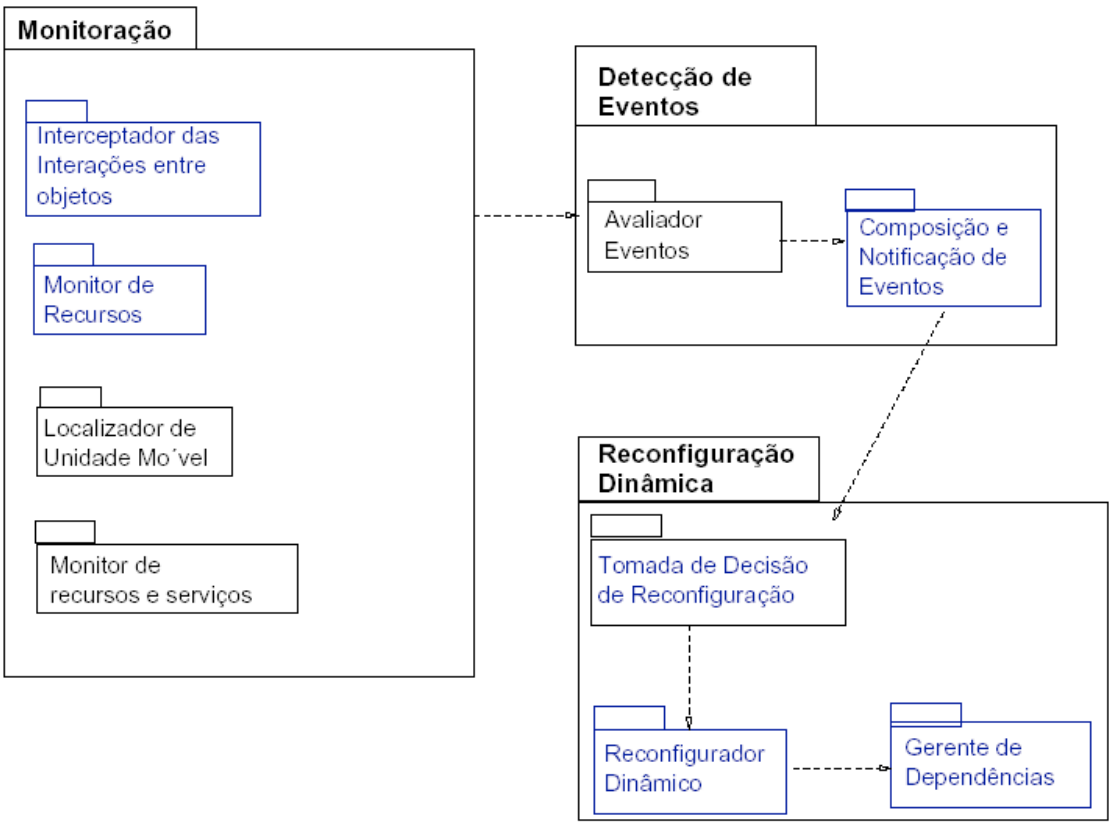

Figura 14. Arquitetura do modelo de referência para um sistema de Informação descentralizado utilizado por usuários móveis ou fixos (SIDAM)

A pesquisa do SIDAM encontra-se no estágio de implementação do modelo apresentado. Além disso, a união de pesquisadores da USP em torno do projeto rendeu o desenvolvimento de algoritmos, teorias e métodos interessantes relacionados ao tema [23].

4.3 MobiSnap

Em 1999, pesquisadores da Universidade Nova de Lisboa (UNL) e da Universidade do Minho em Portugal se uniram na iniciativa do projeto MobiSnap [31].

O MobiSnap é uma estrutura que trabalha na camada intermediária entre o cliente e o servidor, proporcionando o acesso de usuários móveis a sistemas de bancos de dados relacionais, em operações conectadas ou não. Esta plataforma busca isolar as aplicações clientes de problemas relacionados à mobilidade e desconexões.

Nesta estrutura, os algoritmos de *caching* de dados e a propagação de atualizações ao banco são baseados na noção de bancos de dados *snapshots*, compostos de tabelas *snapshots*, que incluem um subconjunto de linhas e colunas de uma dada tabela satisfazendo certas condições, as operações entre o cliente móvel e o servidor ocorrerão levando em conta apenas o subconjunto que é relevante a determinado usuário.

O modelo de transações móveis do MobiSnap apresenta um mecanismo de notificação de resultados de transações nos clientes.

Assim como em outros modelos, a submissão dos dados atualizados (*commit*) na unidade móvel só acontece quando esta modificação ocorre na base fixa. Como é computacionalmente inviável prever os efeitos de uma transação após sua submissão, para este fim o MobiSnap possui um mecanismo de reservas, que simula o que acontecerá com a base depois de uma transação. Um *script* de reservas é definido previamente pelo administrador do banco de dados que, por exemplo, pode reservar a um vendedor o direito de cobrar uma determinada quantia por um produto, mesmo que o preço deste seja atualizado. Estas reservas possuem um tempo de vida.

Neste modelo, um cliente mantém duas réplicas dos dados do servidor, chamadas de submetidas (*committed*) e experimentais (*tentative*). A réplica experimental guarda o último estado consistente do BD móvel.

A tabela 3, na próxima página, resume algumas das características do Coda, do SIDAM e do MobiSnap.

PROJETO	NATUREZA	DESENVOLVIDO POR	CARACTERÍSTICAS
Coda	Sistema de arquivos	Carnegie-Mellon University	– Venus: gerenciador de cache no servidor (estados de *hoarding*, emulação e reintegração); – Replicação otimista; – Boa portabilidade
SIDAM	Modelo de referência	IME-USP	– Controle de informações descentralizadas; – Monitor, Detector de eventos e Reconfigurador Dinâmico; – Referência para aplicações futuras
MobiSnap	Camada de acesso a SGBDR	Universidade Nova de Lisboa e Universidade do Minho	– Mecanismo de reserva; – Mecanismo de notificação de *commit*

Tabela 3. Alguns projetos que abordam o gerenciamento de dados móveis

Um outro importante projeto é o DBMate [51], que discute a respeito do nível de granularidade dos dados armazenados durante um processo de hoarding. O projeto DBMate é um integrante do grupo DATAMAN *Mobile Computing Laboratory* [52] da Rutgers University que há 10 anos apresenta pesquisas sobre diversas áreas da computação móvel. Atualmente, as funcionalidades do DBMate vêm sendo integradas a outros projetos do DATAMAN, como o Web& [52].

Mais projetos e mais informações a respeito do estudo de gerenciamento de dados no ambiente móvel em Universidades e centros de pesquisas podem ser encontradas facilmente na Web em páginas como: IEEE Distributed Systems Online – Mobile&Wireless Projects (http://dsonline.computer.org/mobile/mobile-projects.html), a página do professor Abdelsalam Helal do Departamento de Computação da Universidade da Flórida (http://www.harris.cise.ufl.edu/projects.htm), a página dos workshops ADBIS (*Advances in Database and Information Systems*), localizada em: http://synthesis.ipi.ac.ru/adbis/ e muitas outras, algumas das quais já citadas nas referências deste trabalho.

4.4 Linhas de pesquisa e questões em aberto

Algumas linhas de pesquisa se mostram promissoras na área de bancos de dados móveis e crescem à medida em que cresce o uso dos PDAs. Aplicações de camada intermediária em Java, técnicas de gerenciamento de *cache*, adaptabilidade em aplicações, entre outras.

Buscando contornar restrições do tamanho dos dispositivos móveis, alguns estudos buscam uma interface mais intuitiva e mais rica em funcionalidade. O grupo

de sistemas de bancos de dados da Universidade de Pittsburgh, por exemplo, vêm aperfeiçoando o seu projeto Query-by-Icon [69], um recurso que permite que o usuário consulte a base de dados através do uso de ícones. Ao clicar nestes ícones, o sistema compõe a consulta e a apresenta em linguagem natural.

Outra pesquisa, conduzida por Seydim e Dunham [58], visa criar um *benchmarking* (teste de desempenho de sistema) para computação móvel que inclui consultas sensíveis à localidade, comportamento da unidade móvel e normas de procedimento para adaptação de aplicações, este é o primeiro dos *benchmarks* proposto para a computação móvel que endereça dependência de localidade.

Já no que diz respeito às aplicações, Cohen [8] aborda a questão interessante de sincronização através de objetos da linguagem Java. Outros estudos principalmente relacionados à mobilidade de bancos de dados móveis orientados a objetos foram feitos recentemente ou estão em andamento.

5. O mercado de computação e SGBDs móveis

Visando ilustrar a importância da computação móvel no mercado, neste capítulo são mostrados alguns números de levantamentos feitos por importantes institutos de pesquisa e tendências do mercado internacional.

Os maiores fabricantes de *software* do mundo atentaram para o importante e crescente ramo da computação móvel e lançaram suas soluções comerciais, visando não só o usuário pessoal, mas principalmente o corporativo, que busca o melhor gerenciamento de seu negócio através da manipulação de dados e transações longe dos escritórios.

Na seção 5.2 inicia-se uma discussão das soluções na área de bancos de dados móveis lançadas nos últimos meses pela Sybase (SQL Anywhere Studio), IBM (DB2 Everyplace), Microsoft (SQL Server CE) e Oracle (Oracle 9*i* Lite).

5.1 Mercado de computação móvel

O crescimento da popularidade de dispositivos portáteis é evidente. De acordo com o EPS Statistics [15], o IDC (*International Data Corporation*) projetou que a venda de PDAs no mundo inteiro passará de 13,6 milhões em 2000 a 70,9 milhões em 2005. Segundo o ePaynews [14], o allNetDevices prevê que as aplicações de dados sem fio (mensagens, eCommerce, serviços financeiros, Internet, entretenimento, entre outras) em 2005 terão aumentado em até 42 vezes em relação a 2000.

Atualmente, novos e poderosos recursos têm ajudado os PDAs a se tornarem uma parte da infra-estrutura de uma empresa. As estatísticas mostram que os líderes do mercado de PDAs (Palm, HP, Compaq, por exemplo) são aqueles fabricantes que conseguem atrair o consumidor corporativo. De acordo com a TechWeb [67], a firma de pesquisa de mercado Cahners In-Stat constatou um aumento de 17% nas vendas de PDAs a empresas em 2001 e calcula que esse crescimento em 2002 chegará a 18%, apesar da economia vacilante deste período.

A chegada do WAP (*Wireless Application Protocol*), trouxe muitos benefícios para as aplicações móveis, porém esta tecnologia mostrou-se insuficiente para aplicações de dados intensivos ou para aquelas que requerem uma interface com o usuário mais rica e detalhada. Esta necessidade de manipular maiores quantidades de dados, combinada com a lentidão – e às vezes indisponibilidade – das redes de dados

sem fio atuais, fazem com que muitas aplicações requeiram um banco de dados relacional local no laptop ou no PDA.

As maiores empresas de bancos de dados desenvolveram suas soluções de bancos de dados móveis, que trabalham em conjunto com uma base maior e incluem suporte à sincronização para replicação de dados de e para fontes de dados corporativas (*enterprise data source*). De acordo com a PC Magazine Online [44], o Gartner Dataquest constatou que a Sybase atualmente domina o mercado com o seu SQL Anywhere Studio, com 6 milhões de usuários em cerca de 10.000 localidades representa 68 por cento do mercado de bancos de dados móveis em 2000. Entre outros SGBDs móveis estão o DB2 Everyplace da IBM, o Microsoft SQL Server 2000 Windows CE Edition e Oracle 9*i* Lite.

A seguir, será apresentada uma revisão das principais características de cada um destes produtos, com base nos textos extraídos dos artigos [32, 33, 34] por Bryan Morgan, consultor e fundador da *Wireless Developer Network*, como também de manuais do usuário, *readmes*, *datasheets* e *White Papers* dos sistemas, todos disponíveis em [40, 37, 63, 62, 21, 30].

5.2 SQL Anywhere Studio

Parte do sucesso do SQL Anywhere se deve ao fato de que a Sybase está neste mercado há mais tempo que seus competidores, desde 1996. Em sua versão mais atual, o SQL Anywhere Studio oferece operações de bancos de dados relacionais para as plataformas: Windows (9x, Me, NT, 2000 e XP), Windows CE/Pocket PC, Novell NetWare, Palm, UNIX (incluindo Sun Solaris/Sparc e Linux) e Wind River VxWorks. A sincronização do banco de dados móvel é suportada por vários sistemas gerenciadores de bancos de dados relacionais, como o Sybase Adaptive Server Anywhere, Sybase Adaptive Server Enterprise 11.5 e superior, IBM DB2 Universal Database 7.1, Microsoft SQL Server 7 e 2000, Oracle 8*i* e 9*i*. Uma versão *evaluation* que expira em 60 dias está disponível para download na *homepage* da Sybase [62].

Mais do que apenas um sistema gerenciador de banco de dados *stand-alone*, o SQL Anywhere Studio inclui desenvolvimento de *software*, ferramentas de SGBDs relacionais e de sincronização. Podem-se desenvolver aplicações em várias linguagens de programação, como será visto mais adiante. O programa inclui um sistema gerenciador de bancos de dados relacionais que é capaz de operar como um servidor de *workgroup* ou um banco de dados embutido. As opções de sincronização incluem suporte à replicação direta com o Sybase Adapter Server Enterprise baseada em

mensagens ou replicação em três camadas baseada em *streams* com uma grande variedade de fontes de dados *back-end*. As características dos componentes individuais do Studio são detalhadas a seguir.

5.2.1 SGBDs relacionais

O SQL Anywhere Studio inclui dois poderosos sistemas de bancos de dados relacionais, cada um visando uma classe diferente de usuário e de aplicação:

- O Adaptive Server Anywhere (ASA), produto principal, é um poderoso SGBD Relacional capaz de ser executar em várias plataformas, desde servidores Solaris até dispositivos Pocket PC baseados em Windows CE. O ASA oferece suporte a *stored procedures*, *triggers*, procedimentos de segurança e Java. Seu tamanho máximo é limitado apenas pela capacidade do *host*. Graças à presença de drivers ODBC (*Open Database Connectivity*) e JDBC (*Java Database Connectivity*), os bancos de dados administrados pelo ASA podem ser acessados por virtualmente qualquer programa desenvolvido nas linguagens modernas;
- A opção Ultralite é um SGBD baseado em arquivos que suporta um subconjunto razoável de SQL, ocupa pouca memória e suporta desenvolvimento em C, C++ e Java. Possui ainda suporte integrado a sincronização com o MobiLink. Esta opção é endereçada a dispositivos com recursos restritos (como Windows CE ou PalmOS) e também para aplicações que procuram minimizar a instalação baseada no cliente e questões de gerenciamento. Apenas uma aplicação por vez pode acessar a base no Ultralite.

5.2.2 Ferramentas de sincronização

O Studio oferece três opções de sincronização/replicação de dados: o MobiLink, o SQL Remote e o Replication Server. Este último é um sistema baseado em conexão para replicação entre um pequeno número de bancos de dados. Normalmente usado com uma conexão contínua, confiável e de alta velocidade (ver seção 3.3.1), esta opção é inviável para ambientes onde as máquinas remotas sejam móveis ou apresentem desconexões freqüentes. Sendo assim, serão discutidas apenas as duas outras opções.

A administração do banco de dados e os recursos de sistemas requeridos pelo Mobilink e pelo SQL Remote são mínimos, fazendo com que estas opções sejam ideais para bancos de dados móveis. Ambos podem ajustar com grande precisão a sincronização do cliente para o servidor (e vice-versa) através do desenvolvimento de

scripts de sincronização de upload/dowload em SQL. Tais scripts são, na verdade, armazenados nos servidores de bancos de dados *back-end* e são executados tanto para cada tabela replicada quanto para eventos de sincronização globais (`begin, end`).

O número de bases remotas que podem ser sincronizadas através do MobiLink ou do SQL Remote é limitado apenas pelo sistema gerenciador central. Esse número vai depender da quantidade de informação replicada, da freqüência da sincronização e do projeto da aplicação usada.

O Mobilink realiza replicação baseada em sessão, envolvendo o uso de um servidor de sincronização em camada intermediária que executa ou em Windows 2000/XP ou em Solaris. Aceita solicitação de clientes ASA ou Ultralite e roteia estas requisições de sincronização para os bancos de dados "consolidados" via conexões ODBC.

O SQL Remote realiza replicação baseada em mensagens e requer que o cliente que solicite a sincronização possua o Adaptive Server Anywhere.

Para as bases de dados corporativas, o MobiLink pode trabalhar com muitos servidores populares, incluindo Sybase Adaptive Server Enterprise, Oracle, Microsoft SQL Server e IBM DB2, além do próprio Adaptive Server Anywhere. Enquanto que a instalação do SQL Remote requer que o servidor tenha sua base gerenciada pelo ASA ou Sybase Adaptive Server Enterprise.

5.2.3 Ferramentas de administração e de desenvolvimento

Além de recursos como ferramentas de engenharia reversa, editor de consultas, depurador de *stored procedures* etc. O Studio oferece um número de outras ferramentas destinadas a fazê-lo uma solução tudo-em-um para o desenvolvimento de aplicações móveis:

- A ferramenta Sybase Infomaker pode ser acessada para gerar relatórios *on-the-fly* (ou seja, gerados por demanda e destruídos após uso, não consumindo espaço em disco);
- A poderosa ferramenta de análise e design baseada em UML, o Power Designer, pode ser usada em conjunto com projetos de desenvolvimento em Java, XML, C++ e Visual Basic;
- O Sybase Central e o Interactive SQL, ferramentas para gerenciamento de bancos de dados, permitindo ao usuário efetuar design gráfico e manuseio de bases Ultralite e ASA, bem como os processos de sincronização Mobilink e SQL Remote.

5.2.4 Avaliação

Os pontos fortes do SQL Anywhere Studio são suas múltiplas opções de acesso a dados (ASA e Ultralite), de sincronização (SQL Remote e Mobilink Server) e de suporte para desenvolvimento (via ODBC, JDBC e outros drivers nativos). Poderia se apontar como ponto fraco o fato do produto não prover suporte nativo a dispositivos RIM Blackberry e à plataforma Java 2 Micro Edition (J2ME), dois ambientes que parecem ter sido feitos para operar com a opção Ultralite. Apesar disso, o Studio é o líder de mercado principalmente devido à sua flexibilidade de estratégia de sincronização e suporte a várias linguagens de programação e plataformas.

5.3 DB2 Everyplace

O DB2 Everyplace é a solução da IBM para computação móvel e também tem versões *evaluation* disponíveis para download na *homepage* da IBM [21] nas versões Database Edition e Enterprise Edition. Os componentes do Everyplace são: DB2 Everyplace mobile database (o SGBD propriamente dito, com apenas 150 KB), o DB2 Everyplace Sync Server (que se comunica com o DB2 Everyplace Sync Client no cliente) e o DB2 Everyplace Mobile Application Builder. A figura abaixo mostra como interagem estas partes do sistema, a seguir, alguns detalhes de cada uma delas são vistos.

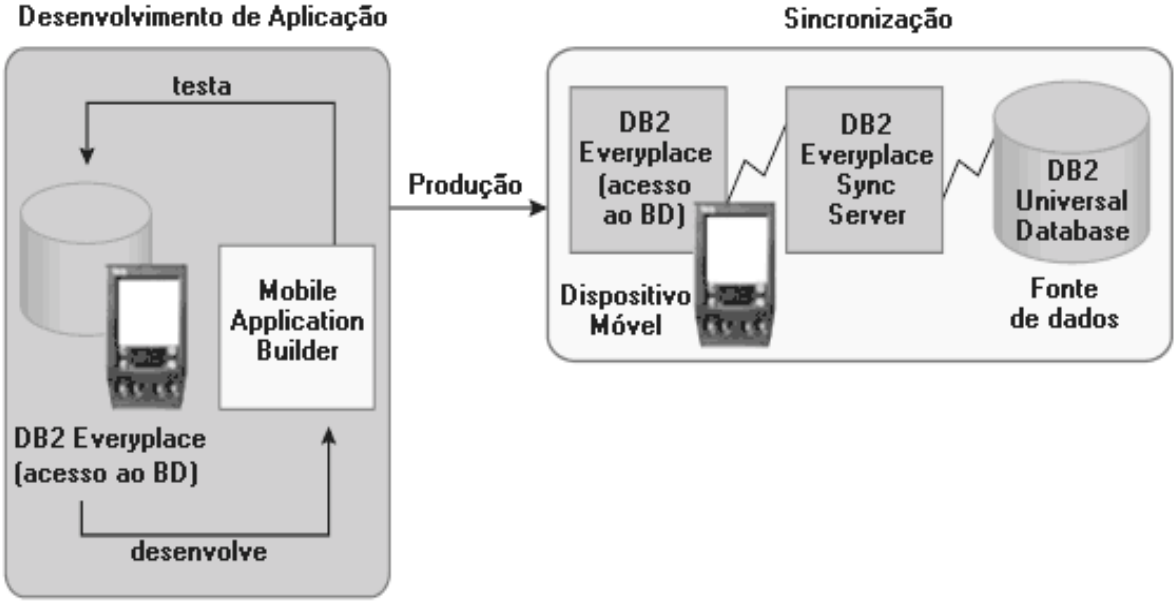

Figura 15. Interação dos componentes do DB2 Everyplace (adaptada de [37])

5.3.1 SGBD DB2 Everyplace

Especialmente planejado para aplicações no ambiente móvel e embutido, o sistema gerenciador de bancos de dados do DB2 Everyplace suporta as operações básicas de SQL como `JOIN`, `ORDER BY`, `GROUP BY`, chaves primárias e estrangeiras de múltiplas colunas, expressões, funções de agregação e restrições. Uma indexação avançada permite uma boa performance para tabelas grandes. Suporta desenvolvimento de aplicações para bancos de dados em C, C++ e Java. E é suportada pelas plataformas: Palm OS, Microsoft Windows CE/Pocket PC, Symbian OS 6, EPOC 5, QNX Neutrino e Linux para dispositivos móveis. Para sistemas de *desktop* para teste de sincronização, o DB2 Everyplace pode ser instalado em Linux, Microsoft Windows 9x, NT, 2000 e XP.

Para uso no cliente, uma das funcionalidades do SGBD Everyplace é o Query-By-Example, uma interface gráfica para executar consultas e visualizar dados de uma tabela, permitindo rolar por entre as várias colunas e registros. Através do Query-By-Example, pode-se apagar, atualizar ou inserir registros, mas estas modificações só serão concretizadas após a sincronização dos dados com o Sync Server.

5.3.2 DB2 Everyplace Sync Server

Programa cliente/servidor da IBM que permite sincronização bidirecional e unidirecional (apenas para inserção) de arquivos e dados. A unidade móvel cliente deve ter instalado o DB2 Everyplace Sync Client que se comunica com o Server que por sua vez, se comunica com o banco de dados da empresa. O Sync Server é compatível com IBM AIX, Solaris, Linux e Windows NT e 2000, suporta consultas remotas e *stored procedures* e sincronização com fontes de dados DB2 Universal Database ou qualquer outra que possua interface JDBC.

O Sync Server é gerenciado na empresa através de uma ferramenta gráfica chamada Mobile Devices Administration Center, que facilita a replicação para um grupo de usuários com necessidades similares de sincronização de dados. Os usuários que receberão conjuntamente os dados de sincronização devem fazer uma assinatura num grupo de usuários, que podem ser feitas na ferramenta DataPropagator ou através do próprio JDBC.

O processo de sincronização pode ocorrer de duas formas: usuários móveis submetem modificações em sua cópia local dos dados da fonte; usuários recebem modificações que foram feitas aos dados que estão no servidor da empresa desde a última sincronização.

Quando os dados são modificados nos usuários móveis, o cliente aciona uma solicitação de atualização dos dados através do Sync Client, a solicitação é então autenticada e posta numa fila no sistema da camada intermediária. Importante notar que o usuário apenas pode modificar o subconjunto de dados do qual é assinante. Na camada intermediária, os dados são colocados numa tabela temporária, onde possíveis conflitos serão resolvidos. No servidor, o DataPropagator captura os dados e aplica as mudanças à tabela no servidor.

Para dados modificados no servidor, o procedimento é inverso, o DataPropagator que executa continuamente no sistema fonte captura as modificações feitas à tabela e as aplica a uma tabela temporária no sistema da camada intermediária, os dados das modificações são colocados numa fila e capturados pelo Sync Client rodando no cliente móvel, que aplica as mudanças na base de dados local.

5.3.3 Mobile Application Builder

O Mobile Application Builder é uma rápida ferramenta de desenvolvimento de aplicações para dispositivos móveis. Algumas aplicações podem ser criadas sem que seja necessário escrever uma só linha de código. Alguns dos muitos recursos desta ferramenta são:

- Criação de formulários diretamente da definição das tabelas;
- Edição de formulários através de uma paleta de controles;
- Modificar os parâmetros de sincronização de uma aplicação;
- Suporte a vários tipos de scripts;
- Aplicações exemplo desenvolvidas usando o Mobile Application Builder;
- Integração com as ferramentas do Palm OS e GNU para compilar, testar e depurar aplicações.

5.3.4 Avaliação

O DB2 Everyplace é um produto compacto e robusto que possui como seus maiores trunfos: o Query-By-Example, por proporcionar uma interface que além de amigável, executa operações no banco de dados; o Mobile Application Builder que torna fácil a construção de novas aplicações por usuários não avançados e o Mobile Devices Administration Center, que torna transparente a administração de bancos de dados ao usuário final. Como pontos fracos do DB2 Everyplace, podem ser apontados a falta de suporte nativo à linguagem Visual Basic e o menor número de opções de plataforma em relação ao SQL Anywhere da Sybase, discutido na seção anterior.

5.4 SQL Server CE

A Microsoft foi a última das grandes empresas a lançar sua solução para um banco de dados móvel. O SQL Server 2000 Windows CE Edition, mais freqüentemente chamado de SQL Server CE é uma versão móvel (operando com 1 a 3 Mb de memória) do banco de dados corporativo gerenciado pelo SQL Server da Microsoft. Pode possuir um *host* de capacidades avançadas, incluindo suporte à gramática de SQL completa do SQL Server, uma grande variedade de tipos de dados, transações aninhadas, encriptação de 128 bits e replicação intercalada com o SQL Server 2000. Os dados podem ser acessados através das APIs ADOCE (*ActiveX Data Objects for CE*) e OLE DB/CE (discutidas na próxima subsessão). Obtém-se um ganho de performance nas buscas indexadas via ADOCE. O SGBD SQL Server CE ocupa entre 800KB e 1.3MB de espaço e bancos de dados de até 2GB são suportados.

A figura abaixo mostra a relação dos ambientes de desenvolvimento, cliente e servidor usando SQL Server CE. A seguir, são discutidas algumas particularidades destes ambientes (figura adaptada do manual do usuário do SQL Server CE, disponível em [30]).

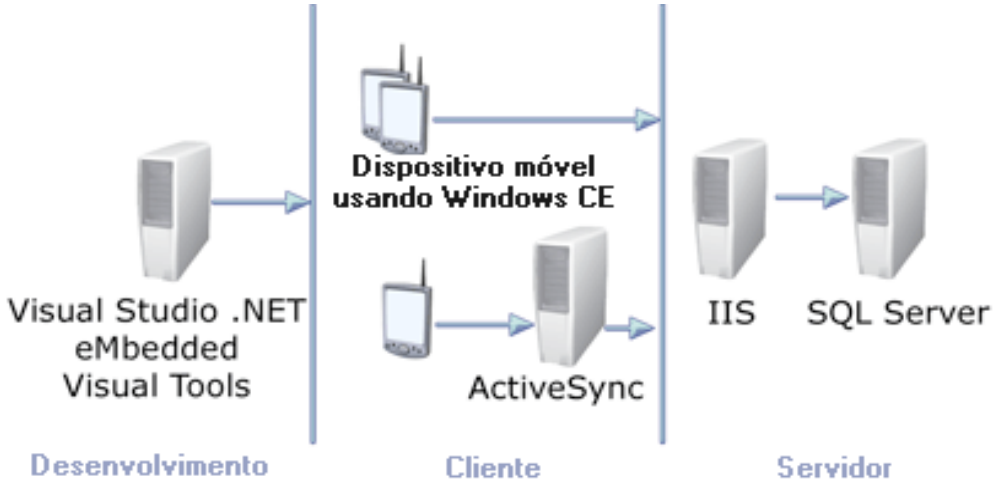

Figura 16. Aplicações desenvolvidas para SQL Server CE no cliente que, por sua vez, acessa a base do servidor para sincronização de dados

5.4.1 Ambiente de desenvolvimento

Dois ambientes de desenvolvimento são suportados: o Visual Studio .NET e o eMbedded Visual Tools. Ambos fornecem um conjunto de APIs e um subconjunto da sintaxe de SQL feitos especificamente para trabalhar com o SQL Server CE.

O SQL Server CE é implementado como um conjunto de DLLs que suportam as seguintes tecnologias de acesso a dados:

- ADOCE e ADOXCE;
- ADO.NET;
- OLE DB para SQL Server CE

ADOCE é a API padrão da Microsoft para acessar e manipular registros de dados no sistema operacional Windows CE. ADOCE apresenta um subconjunto da biblioteca Win32 ADO (*ActiveX Data Objects*) da qual inclui os objetos `Connection`, `Error`, `Recordset`, `Field` e `Fields`.

ADOCE será do interesse de qualquer um que pense em construir uma aplicação para Windows CE, já que ela é não só a API do SQL Server CE, mas também de todo o próprio sistema de arquivos nativo. Usando ADOCE, os bancos de dados podem ser consultados (`SELECT`), editados (`DELETE`/`INSERT`) e modificado (usando comandos DML como `CREATE`/`DROP`/`ALTER TABLE` e `CREATE`/`DROP INDEX`).

A ADOXCE é uma extensão da ADOCE com objetos adicionais de DDL (*Data Defition Language*) e segurança.

ADO.NET é um aperfeiçoamento do ADO e é parte integrante do .NET Compact Framework (um subconjunto da tecnologia .NET suportado pelo Windows CE). O ADO.NET atende a uma variedade de necessidades de desenvolvimento, que incluem criação de aplicações no banco de dados do cliente e objetos de negócio da camada intermediária usados por aplicações, ferramentas, linguagens e browsers.

OLE DB/CE é uma tecnologia de interface de dados de baixo-nível e proporciona maior capacidade de granularidade que o ADOCE e o ADOXCE e torna o acesso e manipulação dos dados mais rápida. Acessando as bases de dados do SQL Server CE via OLE DB/CE, as aplicações terão suporte às interfaces Data Source, Session, Command e Rowset, também um subconjunto do OLE DB.

5.4.2 Sincronização de dados

O SQL Server CE proporciona dois modos de sincronizar um banco de dados baseado em CE com um SQL Server corporativo. O primeiro método usa objetos Remote Data Access (RDA) para realizar uma sincronização *push/pull* entre o dispositivo móvel e o servidor, através do comando `FILTER` de SQL, permitindo que o desenvolvedor customize o fluxo de dados entre as bases. A segunda opção, conhecida como *merge replication* (replicação intercalada) usa o modelo *publish/subscribe*

(publicar/inscrever) para sincronizar dados publicados para uma variedade de dispositivos previamente inscritos. Ambos RDA e replicação intercalada fazem uso do IIS (*Internet Information Server*) e são transportados via HTTP.

No Visual Studio .NET, a aplicação deve chamar os métodos `Synchronize` e `Dispose` na instância do objeto a ser replicado (*Replication object*). Para sincronização de aplicações no eMbedded Visual Tools, os métodos `Initialize`, `Run` e `Terminate` em sucessão devem ser chamados numa instância do *Replication object*.

Quando o dispositivo móvel baseado em Windows CE não possui capacidade de conectividade, o Microsoft ActiveSync pode ser usado para se conectar com o ambiente servidor. Para desenvolver e depurar aplicações, o SQL Server CE requer a instalação do ActiveSync.

A ilustração abaixo mostra o esquema de uma aplicação que usa sincronização de dados com um servidor SQL Server (figura adaptada de SQL Server CE FAQ [30]).

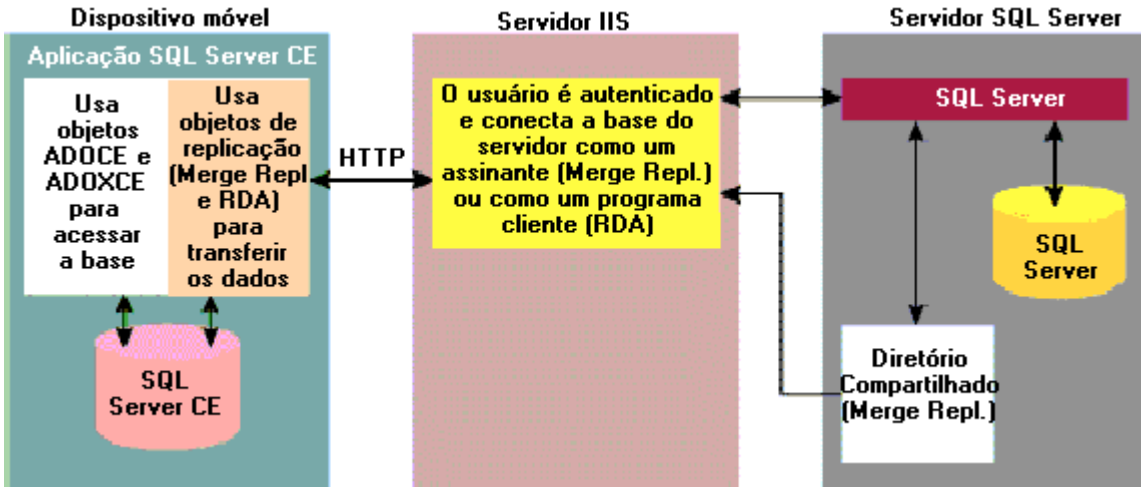

Figura 17. Aplicação realizando sincronização no SQL Server CE

5.4.3 Avaliação

O SQL Server CE pode ser um SGBD muito poderoso nas mãos de um desenvolvedor de aplicações para Windows CE, pois dá suporte a operações de bancos de dados e opções de replicação avançados. Para quem acredita que o Windows CE suplantará o PalmOS e será a plataforma líder de mercado móvel no futuro, o SQL Server CE será um sistema capaz de virtualmente qualquer tarefa. No entanto, vale lembrar que a sincronização com outras fontes de dados corporativos (Oracle, DB2, etc.) não é suportada. Da mesma forma que não são suportadas outras plataformas importantes como PalmOS ou J2ME.

5.5 Oracle9*i* Lite

O Oracle9*i* Lite é o produto de banco de dados móvel da Oracle e não deve ser confundido com o Oracle9*i*AS Wireless que é um servidor de aplicações destinado a habilitar e distribuir conteúdo a uma grande quantidade de dispositivos. Como será visto a seguir, o Oracle9*i* Lite é na verdade uma solução completa para aplicações que requerem o uso de um SGBD relacional no cliente móvel. Inclui suporte a clientes de bancos de dados Win32, Windows CE, PalmOS e EPOC, integração com o mecanismo Advanced Queueing (AQ) da Oracle e *software* de sincronização de dados e aplicações (para bases corporativas gerenciadas por Oracle). Vários recursos foram retirados das versões anteriores e simplificados num único produto com as mesmas funcionalidades, o Oracle9*i* Mobile Server. Por exemplo, todas as funções exercidas pelo Web-to-Go Server, Consolidator Server, Synchronization Server e Web-to-Go Web Server no Oracle 8*i* Lite são executadas pelo Mobile Server no 9*i* Lite.

5.5.1 SGBD relacional Oracle9i Lite

Muito poderoso, este sistema gerenciador de bancos de dados chega a questionar seu próprio nome "Lite" (leve). Oferece 100% de suporte a desenvolvimento em Java, através de drivers JDBC nativos e o suporte nativo a SQLJ [61] embutido e a *stored procedures* em Java, bem como a biblioteca Java Access Class (JAC) que provê uma interface orientada a objeto ao banco de dados. Programas feitos em qualquer ferramenta de desenvolvimento que ofereça suporte a ODBC (Visual Basic, C++, Delphi, etc.) também são suportados. Para aqueles que querem fazer uso das capacidades do SGBD objeto-relacional Oracle9*i* Server, a Oracle inclui suporte para o OKAPI (*Object Kernel API*), que pode ser chamada através de qualquer aplicação C/C++.

O subconjunto de SQL do Oracle9*i* Lite perde pouco para a funcionalidade completa do PL/SQL. O Lite também inclui uma ferramenta que é o equivalente a uma versão móvel do SQLPlus, o Mobile SQL que acessa a base de dados via interfaces ODBC e OKAPI, suportando consultas relacionais e orientadas a objeto. Como foi dito antes, entre os SGBDs móveis, o Lite é o que oferece maior suporte a Java. Para construir uma stored procedure ou um gatilho (*trigger*) em Java, carregam-se as classes de Java através do comando `loadjava` e então uma chamada à classe é feita usando `CREATE FUNCTION` ou `CREATE PROCEDURE`, ambos comandos SQL. E as *stored procedures* em Java podem ser chamadas como qualquer outra, através da sintaxe padrão de SQL. O driver JDBC do Oracle9*i* Lite suporta totalmente as especificações do

JDBC 1.22 e alguns dos recursos do JDBC 2.0 como, por exemplo, o suporte aos tipos de dados BLOB/CLOB.

5.5.2 Ferramentas de sincronização

A sincronização é realizada através das ferramentas Mobile Server e Message Generator and Processor (MGP) que rodam no servidor. O Mobile Server faz uso do modelo publicar/inscrever (*publish/subscribe*), no qual uma ou mais "publicações" de bancos de dados podem ser definidas, consistindo de uma consulta usada para extrair dados da base do servidor. As "inscrições" ou "assinaturas" dos clientes são realizadas no Mobile Server e especificam que publicações (acompanhadas de parâmetros de filtragem) eles gostariam de sincronizar. Lógica de aplicação pode ser inserida no processo através da união de *stored procedures* de PL/SQL nos itens publicados. Todo esse trabalho de publicar/inscrever é gerenciado numa camada intermediária através do Mobile Server. O Mobile Server pode ser acessado pelo cliente móvel através do uso da classe de Java `ResourceManager`. Uma API na linguagem C, MobileSync, proporciona suporte à conexão de sincronização direta do cliente para o servidor de banco de dados.

5.5.3 Ferramentas de administração e de desenvolvimento

Como foi descrito acima, o Mobile Server gerencia a sincronização de dados para e vindos do dispositivo móvel. Além disso, ele pode ser usado para a construção de aplicações para tais dispositivos. Aplicações desenvolvidas no Mobile Server têm seus dados atualizados sempre que ocorre uma sincronização. Este tipo de desenvolvimento é baseado na arquitetura Web-to-Go da Oracle, que essencialmente envolve uma aplicação web num servlet usando o Lite como o banco de dados cliente. O Mobile Client para Web-To-Go está disponível para dispositivos baseados em Win32. Clientes Mobile Sync estão disponíveis para clientes Win32, Windows CE, PalmOS e EPOC. Desenvolvedores de Windows CE também podem fazer uso da API com o ADOCE baseada em objetos para Oracle9*i* Lite.

Além do Mobile Server, o outro componente principal do Oracle 9*i* Lite é o Mobile Development Kit, que dá ao desenvolvedor 4 opções de modelos de aplicação: *Native* (usam ODBC para acessar a base), *Java* (usam JDBC), *Web-to-Go* (apresenta um subconjunto das operações do Web-to-Go do servidor, é ideal para operações desconectadas) e *Branch Office* (que é uma versão multi-usuário da Web-to-Go, suportando até 32 diferentes usuários).

A figura 15 mostra o esquema de funcionamento de uma das ferramentas do Mobile Development Kit mais comumente usadas, o Web-to-Go, ideal para aplicações que rodam num *browser*.

A qualquer momento, o usuário pode escolher se desconectar e então, dados específicos deste cliente serão automaticamente trazidos ao dispositivo móvel, permitindo que aplicações continuem a ter acesso aos dados. Ao reconectar, ocorre a reintegração dos dados para assegurar a consistência.

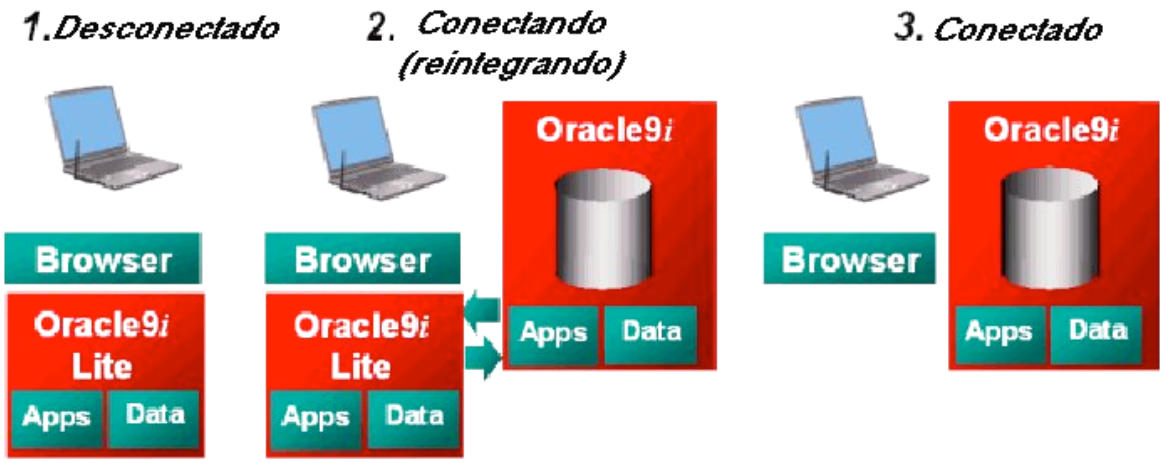

Figura 18. O Web-to-Go permite operação desconectada para aplicações que executam num *browser*, como um servlet (adaptada do *White Paper* de [40])

5.5.4 Avaliação

O Oracle Lite é um produto poderoso feito para o uso numa variedade de plataformas de computação móvel. Ao contrário do Sybase SQL Anywhere, por exemplo, o Lite assume o uso de um SGBD Oracle no *back end*, que o possibilita de fazer uso de recursos específicos (como o Advanced Queueing e o 9*i*AS), deixando de fora desenvolvedores de organizações que usam outros tipos de SGBDs. Outras questões que podem ser levantadas ao avaliar este produto incluem o espaço ocupado na tela do dispositivo móvel (grande quando usado o Web-To-Go em laptops com Win32) ou questões de largura de banda sem fio, já que a sincronização via HTTP não é tão eficiente quanto a feita com TCP/IP ou UDP/IP puros. Porém, estas restrições não comprometem esse que é um produto que preenche os requisitos de qualquer aplicação empresarial e de larga escala.

Na página seguinte, é apresentada uma tabela que resume os componentes e os requisitos de cada um dos sistemas abordados neste capítulo.

PRODUTO (versão atual)	SISTEMAS OPERACIONAIS	ESPAÇO EM DISCO	FERRAMENTAS DE SINCRONIZAÇÃO	TECNOLOGIAS DE DESENVOLVIMENTO E GERENCIAMENTO
Sybase SQL Anywhere Studio 8.0.1	– Windows 9x/Me/NT/2000/XP; – Windows CE/Pocket PC; – NetWare; – UNIX/Solaris/Linux; – VxWorks; – Palm OS	– 3Mb-8Mb (ASA); – 50Kb (Ultralite)	– MobiLink; – SQL Remote	– Sybase Infomaker; – Power Designer; – Sybase Central; – Interactive SQL
IBM DB2 Everyplace 8.1	– Windows 9x/Me/2000/NT/XP; – AIX; – Linux; – Palm OS; – Windows CE; – Symbian 6	– 175 Mb (servidor); – 150Kb (cliente)	– DB2 Everyplace Sync Server; – DB2 Everyplace Sync Client	– Mobile Application Builder; – Mobile Devices Administration Center; – DataPropagator
Microsoft SQL Server 2000 Windows CE 2.0	– Windows NT/2000/XP; – Windows CE	– 800Kb-3Mb; – 30Mb-45Mb (desenvolvimento)	– Remote Data Access (RDA); – Replicação Intercalada (*Merge Replication*)	– Microsoft Visual Studio.NET; – Microsoft eMbedded Visual Tools; – ADOCE, ADOXCE, OLE DB/CE, ADO.NET
Oracle 9i Lite 5.0.1	– Windows 9x/NT/2000; – Windows CE; – Palm OS; – EPOC 5	– 200Mb (servidor); – N/D (cliente)	– Mobile Server; – Message Generator and Processor (MGP)	– Mobile Development Kit

Tabela 4. Resumo das características dos sistemas apresentados.
Obs.: N/D = a informação não estava disponível na documentação pesquisada

5.6 Conclusões

Uma empresa que vise implementar sua organização em PDAs e fazer uso de um sistema gerenciador de banco de dados móvel não estará tão propensa a problemas de ajuste (*tuning*) e administração dos seus dados quanto estaria se estivesse usando SGBDs corporativos, embora técnicas sensatas de design lógico e físico como, por exemplo, modelagem e normalização, uma vez que os bancos de dados ainda seriam relacionais.

O maior impacto desta mudança estaria no planejamento e gerenciamento da sincronização de dados. Surgirão questões como: quando a sincronização deverá ser agendada? Como serão afetadas as aplicações de alta produtividade envolvidas em

sincronizações? Como garantir que um cliente móvel irá sincronizar seus dados confiavelmente e na hora prevista? Para resolver tais questões é necessário que se tenha um bom conhecimento dos novos sistemas gerenciadores de dados móveis que serão implantados na empresa para que se possa tirar o maior proveito de suas funcionalidades. Os DBAs devem estar preparados para esse novo tipo de desafio, pois a computação móvel pertence ao futuro e o futuro já começou.

6. Considerações finais e trabalhos futuros

No início deste trabalho foi apresentada uma visão geral da computação móvel, através de seus dispositivos, dos desafios e restrições do ambiente de comunicação sem fio e portátil e estudando modelos e arquiteturas aplicados neste paradigma.

O objetivo da primeira parte da pesquisa, portanto foi o de situar a tecnologia de bancos de dados em um ambiente onde a computação apresenta particularidades que afetam as aplicações, quase sempre mudando a concepção que se tinha delas. Questões clássicas como *caching* de dados num sistema distribuído, gerenciamento de transações, recuperação de falhas, devem ser repensadas e reestruturadas para assegurar o bom funcionamento de um sistema de banco de dados móvel. E novas questões surgem como gerenciamento de localidade e sincronização num meio de ligação sem fio.

Por fim, o trabalho discutiu algumas abordagens que prometem para o futuro uma boa quantidade de pesquisa e finalizou através de estudos de caso tanto do meio acadêmico quanto do comercial.

O tema, fascinante e rico em detalhes, possui uma evidente abrangência, já que se trata do ponto de convergência entre as áreas de bancos de dados, comunicação móvel e redes de computadores. Pelo excesso de diferentes abordagens ligadas de uma forma ou de outra ao assunto, é enorme a quantidade de material disponível para uma pesquisa aprofundada. Ao mesmo tempo, poucas publicações tratam de bancos de dados móveis dando uma visão geral da tecnologia e, na bibliografia básica consolidada, o assunto é tratado de forma muito breve e superficial.

A maior dificuldade deste trabalho foi, portanto, estruturar um relatório que seguisse uma só linha, em vista da tamanha quantidade de material recolhido. Pela dificuldade em alcançar esta estruturação ideal dos tópicos, alguns assuntos podem ter sido supervalorizados enquanto outros negligenciados.

Melhoras nesta monografia, produto final da pesquisa do trabalho de graduação, podem ser produzidas através de uma reestruturação, como por exemplo, uma visão mais breve dos desafios de computação móvel e mais aprofundada dos tópicos de segurança e a divisão em duas seções daquela que abordou o *caching* de dados e difusão. Além da inclusão de um maior número de projetos de pesquisas

levados a cabo pelas universidades de computação mais importantes, que se constituem em interessantes casos de estudo.

Em futuros trabalhos nesta área, um aprofundamento maior em apenas uma das questões de bancos de dados móveis é recomendado, principalmente a que diz respeito às consultas sensíveis à localidade e as transações, que não ainda não possuem um modelo universalmente aceito e vêm sendo alvo de várias pesquisas e estudos.

Interessante também seria abordar bancos de dados móveis não convencionais, ou seja, inseridos em redes ad-hoc, ou no modelo peer-to-peer, ou sincronizados intermitentemente, por exemplo. A pesquisa mostrou que o número de estudos endereçados a estes paradigmas é crescente, já que muitos dos artigos mais novos endereçavam-se a estas abordagens.

Um trabalho prático também é muito viável, uma sugestão seria testar as várias tecnologias de replicação nos diferentes sistemas vistos e, possivelmente, fazer um estudo comparativo do desempenho do Oracle 9*i* Lite e do SQL Server CE, por exemplo. Vários tipos de aplicações para bancos de dados móveis usando Java poderiam ser propostos e implementados, o espectro aqui é enorme, uma vez que a cada dia, os dispositivos portáteis ganham poder computacional e se tornam de propósito geral.

Enfim, conhecer todos os aspectos da computação móvel é tão difícil quanto não se interessar de forma irreversível após um primeiro contato. E, especialmente na área de bancos de dados móveis, o trabalho de pesquisa anseia por prosperar e amadurecer, este trabalho buscou principalmente criar o interesse necessário que proporcione este crescimento.

Referências

[1] Badrinath, B. R.; Phatak, Shirish H. "*Database Architecture for Handling Mobile Clients*", abril de 1998. Disponível em: http://www.cs.berkeley.edu/~adj/cs294-1.s98/papers/BadriDB.ps

[2] Barbar'a, Daniel; Imielinski, T. "*Sleepers and Workaholics: Caching Strategies in Mobile Environments*", agosto de 1994. Disponível em: http://citeseer.nj.nec.com/86405.html

[3] Braam, P.J., Callahan, M.J., Satyanarayanan, M., Schneider, M. "*Porting the Coda File System to Windows*", junho 1999. Disponível em: http://almond.srv.cs.cmu.edu/afs/cs.cmu.edu/project/coda/Web/docdir/freenix99.pdf

[4] Carnegie Mellon University – School of Computer Science. Mobile Information Access - Coda and Odyssey. Acessado em agosto de 2002. http://www-2.cs.cmu.edu/afs/cs/project/coda/Web/coda.html

[5] Carnegie Mellon University – Project Aura. http://www-2.cs.cmu.edu/~aura/. Acessado em setembro de 2002.

[6] Chrysanthis, Panos K.; Walborn, G.D. "*Transaction Processing in PRO-MOTION*", fevereiro de 1999. Disponível em: http://doi.acm.org/10.1145/298151.298393

[7] CNET.com. http://www.cnet.com. Acessado em setembro de 2002.

[8] Cohen, Norman H. "*A Java Framework for Mobile Data Synchronization*", 2000. Disponível em: http://www.research.ibm.com/sync-msg/CoopIS2000.pdf

[9] Dirckze, R.A.; Gruenwald, L. "*A pre-serialization transaction management technique for mobile multidatabases*", dezembro de 2000. Disponível em: http://www.cs.ou.edu/~database/documents/dg00.pdf

[10] Dunham, Margaret; Helal, A. "*Mobile Computing and Databases: Anythign New?*", dezembro de 1995. Disponível em: http://www.seas.smu.edu/~mhd/pubs/95/record.ps

[11] Dunham, M. H.; Helal, A.; Balakrishnan, S. "*A Mobile Transaction Model That Captures Both the Data and Movement Behavior*", 1997. Disponível em: http://www.seas.smu.edu/~mhd/pubs/97/monet.ps

[12] Dunham, Margaret H.; Kumar, Vijay. "*Impact of Mobility on Transaction Management*", agosto de 1999. Disponível em: http://www.seas.smu.edu/~mhd/pubs/99/mobide1.ps

[13] Elmasri, Ramez; Navathe, Shamkant B. "*Fundamentals of Database Systems*". Addison-Wesley, 2000.

[14] ePaynews. ePayment Resource Center – Mobile Commerce Statistics. http://www.epaynews.com/statistics/mcommstats.html. Acessado em setembro de 2002.

[15] EPS Statistics. New Media Statistical Data – Mobile. http://www.epsltd.com/IndustryInfo/Statistics/Mobile.htm. Acessado em setembro de 2002.

[16] Fitzpatrick, Paul. "*Introduction to Mobile and Wireless Communications Systems*". University of Melbourne, 2002. Disponível em: http://www.ee.mu.oz.au/courses/431-633/documents/1_Introduction.pdf

[17] Flinn J., Satyanarayanan, M. "*Energy-aware adaptation for mobile applications*", dezembro de 1999, disponível em: http://almond.srv.cs.cmu.edu/afs/cs.cmu.edu/project/coda/Web/docdir/s17.pdf

[18] Forman, George H.; Zahorjan, John. "*The challenges of Mobile Computing*", abril de 1994. Disponível em: http://www.cs.ucsb.edu/~cs290i_mc/papers/Forman_Challenges.pdf

[19] Gruber, Patrick; Mungioli, Marcus; Völkel, Frank. "*The New Generation - Multitalented PDAs - Comparison of 6 PDAs*", 4 de abril de 2002. Disponível em: http://www4.tomshardware.com/mobile/02q2/020404/index.html

[20] Gruenwald; Banik, Shankar M. "*Power Aware Management of Mobile Real-Time Database Transactions in Ad-hoc Networks*", Outubro de 2000. Disponível em: http://www.cs.ou.edu/~database/documents/gb00.pdf

[21] IBM Corp. DB2 Everyplace. http://www.ibm.com/software/data/db2/everyplace/. Acessado em agosto de 2002.

[22] Imielinski, Tomasz; Badrinath, B.R. "*Mobile Wireless Computing: Solutions and Challenges in Data Management*", 1993. Disponível em: http://citeseer.nj.nec.com/imielinski93mobile.html

[23] Instituto de Matemática e Estatística (IME) – Universidade de São Paulo (USP) – Sistemas de Informação Distribuídos para Agentes Móveis (SIDAM). http://www.ime.usp.br/~sidam. Acessado em agosto de 2002.

[24] Internet Engineering Task Force (IETF). IP Routing for Wireless/Mobile Hosts (mobileip). http://www.ietf.org/html.charters/mobileip-charter.html, acessado em agosto de 2002.

[25] Ito, Giani C. "*Bancos de dados móveis: uma análise de soluções propostas para gerenciamento de dados*", tese de mestrado em Ciência da Computação. Universidade Federal de Santa Catarina, abril de 2001.

[26] Kottkamp, H.-E.; Zukunft, O. "*Location-aware query processing in mobile database systems*", 1998. Disponível em: http://doi.acm.org/10.1145/330560.330850

[27] Kumar, Vijay. "*Mobile and Wireless Database Systems*", tutorial da X Conference on Management of Data (COMAD 2000), dezembro de 2000. Disponível em: http://www.cse.iitb.ernet.in:8000/proxy/db/~dbms/comad2000/tutorials/

[28] Lubinski, Astrid. "*Security Issues in Mobile Database Access*", novembro de 1998. Disponível em: http://wwwdb.informatik.uni-rostock.de/~lubinski/artikel/11_3_98.ps

[29] Massachussets Institute of Technology. "*Kerberos: The Network Authentication Protocol*". http://web.mit.edu/kerberos/www/. Acessado em agosto de 2002.

[30] Microsoft Corporation. SQL Server CE Home. http://www.microsoft.com/sql/CE/default.asp. Acessado em agosto de 2002.

[31] MobiSnap Project Team. http://www-asc.di.fct.unl.pt/MobiSnap/. Acessado em agosto de 2002.

[32] Morgan, Bryan. "*Mobile Database Review: Microsoft Databases for Windows CE*", 5 de abril de 2002. Artigo da InformIT: http://www.informit.com

[33] Morgan, Bryan. "*Mobile Database Review: Oracle 9i Lite*", 22 de fevereiro de 2002. Artigo da InformIT: http://www.informit.com

[34] Morgan, Bryan. "*Mobile Database Review: Sybase SQL Anywhere Studio 8.0*", 15 de fevereiro de 2002. Artigo da InformIT: http://www.informit.com

[35] Nesargi, S.; Prakash, R. "*Configuration of Hosts in a Mobile Ad-Hoc Network*", 2002. Disponível em: http://www.utdallas.edu/~ravip/papers/infocom2002.pdf

[36] Neves, N.; Fuchs, W. K. "*Adaptive Recovery for mobile environments*", outubro de 1996. Disponível em: http://composer.ecn.purdue.edu/~fuchs/fuchs/haseNN96.ps

[37] Noether, Angela. DB2 Everyplace White Paper, 2001. Disponível em: http://www.ibm.com/software/data/db2/everyplace/extendingenterprise.pdf

[38] Nokia. Nokia 7650. http://www.nokia.com/phones/7650/. Acessado em setembro de 2002.

[39] Novel Inc.,Novell AppNotes. *"How does Replication Work?*" , junho de 1997. Disponível em: http://developer.novell.com/research/appnotes/1997/june/02/03.htm

[40] Oracle Corporation. Oracle 9*i* Lite: The Internet Platform for Mobile Computing. http://otn.oracle.com/products/lite/content.html. Acessado em agosto de 2002.

[41] Özsu, M. Tamer; Valduriez, Patrick. "*Principles of Distributed Database Systems*". Prentice-Hall, Inc, 1999.

[42] Pabla, C. "*A beginner's look at the SyncML protocol and procedures*", abril de 2002. Disponível em: http://www.ibm.com/developerworks/library/wi-syncml2/?dwzone=wireless

[43] Palm Inc. Palm Products. http://www.palm.com/products/. Acessado em setembro de 2002.

[44] PC Magazine Online. "*On the Road to Mobile Databases*", 26 de março de 2002. Disponível em: http://www.pcmag.com/article/0,2997,a=23371,00.asp

[45] PCTechGuide – The PC Technology Guide. http://www.pctechguide.com/25mobile.htm. Acessado em agosto de 2002.

[46] PDA Buyer's Guide. http://www.pdabuyersguide.com. Acessado em setembro de 2002.

[47] Perich, Filip et.alii. "*Query Routing and Processing in Mobile Ad-hoc Environments*", novembro de 2001. Disponível em: http://research.ebiquity.org/re/papers/mogatu_trcs0118.pdf

[48] Pitoura, Evaggelia; Samaras, George. "*Data Management for Mobile Computing*", Kluwer Academic Publishers, 1998.

[49] Pitoura, Evaggelia; Samaras, George. "*Locating Objects in Mobile Computing*", 2001. Disponível em: http://www.cs.uoi.gr/~pitoura/distribution/location-survey.ps

[50] Radhakrishnan, S. "*Handoff management – An overview*", setembro de 1998. Disponível em: http://www.ittc.ukans.edu/~rsarav/801/handoff.html

[51] Rutgers University – DBMate Project. http://www.cs.berkeley.edu/~adj/cs294-1.s98/dbmate/. Acessado em agosto de 2002.

[52] Rutgers University – DATAMAN Mobile Computing Laboratory. http://athos.rutgers.edu/dataman/. Acessado em agosto de 2002.

[53] Satyanarayanan, Mahadev. "*Accessing information on demand at any location – Mobile Information Access*", outubro de 1995. Disponível em: http://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/ieeepcs95.pdf

[54] Satyanarayanan, M. et.alii. "Coda: a highly available File System for a Distributed Workstation Environment", abril de 1990. Disponível em: www.cs.cmu.edu/afs/cs/project/ coda/Web/docdir/wwos2.pdf

[55] Sensors Online. Mobile Ad-hoc Networking, janeiro de 2001. Disponível em: http://www.sensorsmag.com/articles/0101/18/main.shtml

[56] Serrano-Alvarado, P.; Roncancio, C.; Adiba, M. "*Mobile Transactions Supports for DBMS: An Overview*", maio de 2001. Disponível em: http://www-lsr.imag.fr/Les.Publications/paperBDA.ps.gz

[57] Seydim, A. Y. "*An Overview of Transaction Models in Mobile Environments*", novembro de 1999. Disponível em: http://engr.smu.edu/~yasemin/mobile_trans.pdf

[58] Seydim, A. Y.; Dunham, Margaret H. "*A Location Dependent Benchmark with Mobility Behaviour*", julho de 2002. Disponível em: http://engr.smu.edu/~yasemin/ideas.pdf

[59] Seydim, A. Y.; Dunham, Margaret H.; Kumar, Vijay. "*Location dependent query processing*", maio de 2001. Disponível em: http://engr.smu.edu/~yasemin/mobide.pdf

[60] Silberchatz, Abraham; Korth, Henry F.; Sudarshan, S. "*Sistema de Banco de Dados*". Makron Books, 1999.

[61] SQLJ.ORG. http://www.sqlj.org/. Acessado em agosto de 2002.

[62] Sybase Inc. SQL Anywhere Studio. http://www.sybase.com/products/anywhere/. Acessado em agosto de 2002.

[63] Sybase Inc. Sybase Datasheet, 2001. Disponível em: http://www.sybase.com/detail/1,6904,1016232,00.html

[64] Sybase Inc. Product Manuals - SQL Anywhere Studio Documentation, 2002. Disponível em: http://sybooks.sybase.com/sas.html

[65] SyncML – Data Synchronization and device management. http://www.syncml.org/ Acessado em setembro de 2002.

[66] Tanenbaum, A. S. "*Computer Networks (3rd Edition)*". Prentice-Hall Inc., 1996.

[67] TechWeb. *"PDAs get a grip on the Enterprise"*, 22 de abril de 2002. Disponível em: http://www.techweb.com/tech/mobile/20020422_mobile

[68] Tedeschi, Enrico. "*Historic Home Computers*". http://www.etedeschi.ndirect.co.uk. Acessado em agosto de 2002.

[69] University of Pittsburgh - Database System Groups - Mobile Query By Icon. http://www.cs.pitt.edu/admt/Projects/mobile_computing.html, acessado em setembro de 2002.

[70] Villate, Yolanda; Illaramendi, A.; Pitoura, E. "*Data Lockers: Mobile-Agent Based Middleware for the Security and Availability of Roaming Users Data*", setembro de 2000. Disponível em: http://www.cs.uoi.gr/~pitoura/distribution/villateLockersCoopis2000.ps

[71] Vlach, Richard. "*Mobile Database Procedures in MDBAS*", 2001. Disponível em: http://aglaja.ms.mff.cuni.cz/~vlach/papers/mdds01.ps

[72] Wu, K.-L.; Yu, P.S.; Chen, M.-S. "*Energy-Efficient mobile cache Invalidation*", 1998. Disponível em: http://www.ee.ntu.edu.tw/~mschen/paperps/mobile.pdf

[73] Yao, Jenq-Foung; Dunham, Margaret H. "*Caching Management of Mobile DBMS*", 1999. Disponível em: http://www.seas.smu.edu/~mhd/pubs/99/icae.doc.

[74] Yee, W.G.; S.B. Navathe, S.B.; Datta, A.; Mitra, S. "*Grouping design to improve server scalability in intermittently synchronized databases*", fevereiro de 1998. Disponível em: http://www.cc.gatech.edu/computing/Database/papers/icde98.ps

[75] Yee, W.G.; Donahoo, M.J.; Omiecinski, E.; Navathe, S.B. *"Scaling Replica Maintenance in Intermittently Synchronized Databases*", novembro de 2001. Disponível em: http://cs.baylor.edu/~donahoo/papers/YDO+01.ps

[76] ZDNET Reviews. http://www.zdnet.com/reviews. Acessado em setembro de 2002.