2
Conceitos e Fundamentos

Este capítulo apresenta os principais conceitos necessários para o entendimento e desenvolvimento desta tese. Como este trabalho refere-se à integração de SGBD a um ambiente de computação móvel, com o uso de agentes de software e frameworks, inicialmente são apresentados os conceitos referentes ao ambiente de dados, em seguida ao ambiente da computação móvel e, posteriormente, à engenharia de software, no que diz respeito às abordagens e tecnologias de frameworks e agentes de software. Detalhes quanto ao funcionamento do ambiente de computação móvel estão descritos no Anexo A.

2.1
O Ambiente dos Sistemas de Bancos de Dados

Diversas atividades que envolvem alguma interação com **Sistemas de Bancos de Dados (SBD)** estão presentes em nosso dia-a-dia. O depósito ou a retirada de dinheiro em um banco, reservas em hotéis ou assentos de aviões, a compra de produtos em um supermercado ou consultas a catálogos informatizados de uma biblioteca são exemplos do que se costuma denominar *"aplicações de bancos de dados tradicionais (convencionais)"* [21].

Ao longo dos últimos anos, os SBDs também têm sido utilizados em novos tipos de aplicações, tais como aplicações geográficas, de multimídia, biológicas, de apoio à decisão e processamento analítico on-line, em sistemas de tempo real e em aplicações executadas na internet ou em ambiente de computação móvel.

Bancos de dados e suas tecnologias vêm tendo um grande impacto no crescimento do uso de computadores, desempenhando um papel extremamente importante em todas as áreas da ciência em que a computação é utilizada. Dentre as diversas definições para um banco de dados podemos destacar:

- Um **banco de dados** é uma coleção de dados relacionados, em que dados são definidos como fatos conhecidos que podem ser registrados e que possuem significado implícito [21];
- Um **banco de dados** consiste em uma coleção de objetos que representam entidades do mundo real. Cada objeto do banco de dados possui um nome e um valor. O conjunto de valores de todos os objetos armazenados em um banco de dados em um dado momento no tempo é chamado de *estado do banco de dados* [7].

Definição 1: Sejam $\mathbf{obj}_p$ e $\mathbf{obj}_k$ dois objetos definidos em um sistema de computação. Dizemos que **BD** é um Banco de Dados se e somente se:

1. $\mathrm{BD} = \bigcup_{i=1}^{n} \mathrm{obj}_i$
2. $\forall\ \mathrm{i} \leq \mathrm{m},\ \mathrm{j} \leq \mathrm{n},\ \mathrm{i} \neq \mathrm{j}$: $\mathrm{obj}_i \bigcap \mathrm{obj}_j = \emptyset$

Assim, pode-se dizer que um banco de dados é uma coleção de objetos.

O *estado de um banco de dados* representa uma fotografia do mundo real, no qual somente os aspectos estáticos são refletidos. Entretanto, um banco de dados deve também refletir as mudanças do mundo real. Essas mudanças são capturadas pela noção de transição de estados do banco de dados, as quais representam as mudanças de um estado particular do banco de dados para outro estado, ou seja, uma fotografia atualizada do mundo real [7].

Um **Sistema de Gerência de Banco de Dados (SGBD)** é uma coleção de programas que possibilita que os usuários criem e mantenham um banco de dados. O SGBD é, portanto, um sistema de software de finalidade genérica, que facilita o processo de definição, construção e manipulação de bancos de dados para várias aplicações [21].

Um **Sistema de Banco de Dados (SBD)** é formado pelos bancos de dados, pelo SGBD e pelas ferramentas utilizadas para a criação e manutenção do banco de dados, bem como pelas aplicações para atualização e consulta aos dados do banco de dados.

2.2 Concorrência e Transações em Banco de Dados

Esta seção apresenta os principais conceitos sobre a funcionalidade de concorrência nos SGBDs e o conceito de transação de banco de dados.

2.2.1 Concorrência em Banco de Dados

Os modelos conceituais de dados captam as restrições que o mundo real impõe sobre suas entidades, como, por exemplo, o saldo de um produto disponível para venda tem que ser superior ou igual a zero, cada cliente deve ter apenas um endereço para correspondência, etc. Quando transformado em um modelo lógico de banco de dados, esse novo modelo deve capturar essas restrições, as quais são chamadas de *restrições de consistência*. Assim, utilizando em conjunto os conceitos de estado do banco de dados e de restrições de consistência, podemos afirmar que, se os valores dos objetos de um estado particular do banco de dados satisfazem todas as restrições de consistência, então podemos dizer que o banco de dados é consistente [7].

De uma forma geral, as operações sobre os objetos do banco de dados contidas nos programas de aplicações são as ferramentas por meio das quais as transições de estados do banco de dados são realizadas. Em ambientes com vários usuários, os programas de aplicações executam concorrentemente. Na prática, a concorrência entre programas significa que as operações de um programa podem ser executadas entre duas operações de um outro programa. Algumas execuções de intercalamentos de operações de diferentes programas podem levar a mudanças inconsistentes no banco de dados. Assim, os SGBDs devem monitorar e controlar a execução concorrente dos programas de aplicações no sentido de evitar inconsistências. Essa funcionalidade é chamada de *controle de concorrência* [7].

Para o controle da concorrência, somente as operações sobre os objetos do banco de dados são relevantes, podendo as demais operações dos programas ser desprezadas. Esse conjunto de operações sobre os objetos do banco de dados realizadas por um programa de aplicação é chamado de *transação de banco de dados*. Assim, *uma transação é uma abstração que representa a seqüencia de operações de banco de dados resultante da execução de um programa de aplicação*. A execução concorrente de um conjunto de transações é realizada pela intercalação das operações sobre os objetos do banco de dados por várias transações. Uma execução de transações concorrentes é correta se produz um estado consistente do banco de dados [7]. O objetivo principal de um modelo de transações é garantir que a execução concorrente de várias transações garanta estados consistentes do banco de dados. Maiores detalhes sobre a funcionalidade da concorrência entre as transações de banco de dados podem ser encontrados em [7], [20] e [62].

2.2.2 Transação em Banco de dados

Vários usuários podem acessar bancos de dados simultaneamente. Toda e qualquer operação solicitada a um SGBD é executada como parte de uma transação de banco de dados. *Uma transação de banco de dados é uma unidade lógica de processamento sobre o banco de dados que pode incluir uma ou mais operações de inclusão, exclusão, modificação ou obtenção* [20] e [49].

Transações são modeladas como uma seqüência finita de operações sobre os objetos do banco de dados. Existem, basicamente, dois tipos de operações que podem ser executados sobre um determinado objeto(obj): read(obj) e write(obj) (r(obj) e w(obj) ou ler (obj) e gravar (obj)). A operação r(obj) significa que a transação lê o valor de obj e w(obj) significa que a transação atualiza o valor de obj. Assim, podemos definir:

Definição 2: A invocação de uma operação sobre um objeto do banco de dados é chamada de **evento de objeto**. O tipo de um objeto define as operações e os eventos possíveis para o objeto.

Será usado o termo $\mathbf{op}_t[\mathbf{obj}]$ para denotar o evento correspondente à invocação da operação **op** sobre o objeto **obj** pela transação **t**, e $\mathbf{E\text{-}Obj}_t$ para denotar o conjunto de eventos que pode ser invocado[1] pela transação **t** (por exemplo, $\mathbf{op}_t[\mathbf{obj}] \in \mathbf{E\text{-}Obj}_t$).

Utilizando a definição formal de transações apresentada em [7], temos que:

Definição 3: Seja T uma transação. Sejam x e y objetos do banco de dados. A semântica da operação $w(x) \in op_T(x)$, representada por $T(w(x))$ é definida como:

$T(w(x)) := \int_{w(x)} (r(y_1), r(y_2), ..., r(y_n))$,
onde $(r(y_i) \in op_T(x)$, $r(y_k)$ precede $w(r)$ e $0 < k < n$.

O símbolo $\int_{w(x)}$ representa uma função sem interpretação. Essa definição significa que o resultado da operação de gravação w(x) depende dos valores que foram recuperados nas operações de leituras e que são desconhecidos.

Operações sobre os objetos do banco de dados podem levar a conflitos na atualização desses objetos. Assim, é necessária a definição de *operações*

[1] O sentido do termo "invocar um evento"é o mesmo de "causar a ocorrência do evento".

conflitantes. Usando a definição de conflitos de operações enunciada em [7] temos:

Definição 4: Sejam $p_i \in \mathrm{OP}(T_i)$ e $q_j \in \mathrm{OP}(T_j)$ operações das transações T_i e T_j respectivamente, em que i ≠ j. Define-se que as operações p_i e q_j *conflitam* se e somente se são executadas sobre o mesmo objeto do banco de dados e ao mesmo uma das operação é uma operação de escrita (gravação).

Por essa definição, uma operação $r_i(x)$ sempre conflita com uma operação $w_j(x)$ e $w_i(x)$ conflita com $r_j(x)$, onde i ≠ j. Por outro lado, uma operação $w_i(x)$ não conflita com $w_j(y)$ porque opera sobre objetos diferentes (x e y), e uma operação $r_i(x)$ não conflita com $r_j(x)$, uma vez que nenhuma dessas operações é uma operação de gravação.

Uma maneira usual e explícita de se especificar o início e o fim de uma transação, em uma aplicação, consiste em utilizar delimitadores do tipo `iniciar-transação` (*begin-transaction*) e `terminar-transação` (*end-transaction*). Uma aplicação pode conter diversas transações de banco de dados.

Durante sua execução, uma transação pode passar por vários estados [62]. A Figura 2.1, extraída de [20], apresenta esses estados, os quais estão descritos a seguir.

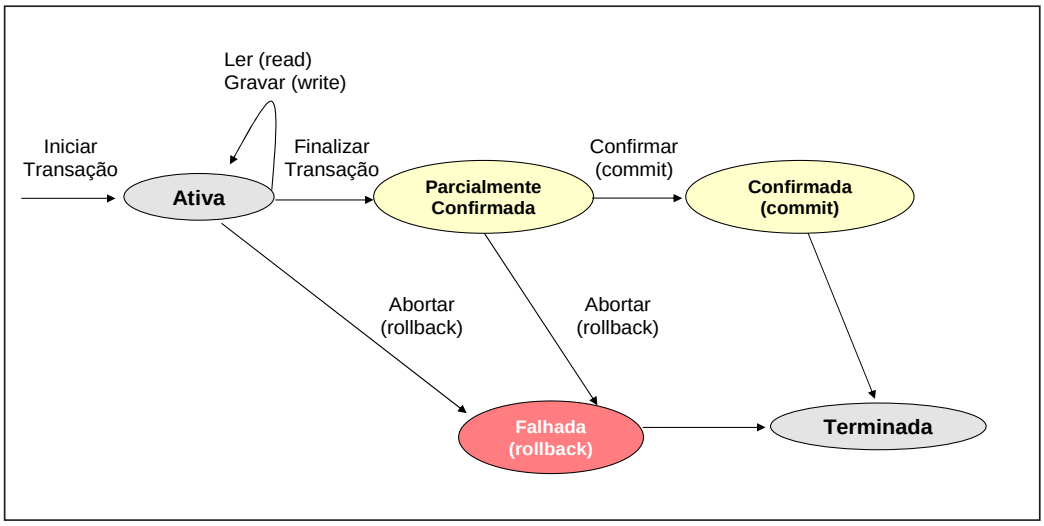

Figura 2.1: Diagrama de Transição de Estados de uma transação

– **Ativa** - após iniciar a sua execução, a transação vai para o estado ativo. Nesse estado, ela pode emitir todas as suas operação de leitura e gravação (*read* e *write*) sobre os objetos do banco de dados. Se todas as suas operações foram realizadas com sucesso, ela passa para

o estado de *parcialmente confirmada*. Caso haja alguma falha, vai para o estado de *falhada*.

- **Parcialmente Confirmada** (*committed*) - nesse ponto, a transação executará alguns protocolos de segurança para garantir que, caso haja alguma falha no sistema, o subsistema de recuperação será capaz de garantir a confirmação (*commit*) da transação no banco de dados ou será obrigado a desfazer os seus efeitos (*rollback*).
- **Falhada** (*rollback*) - indica que a transação terminou sem sucesso e que todas as suas operações ou efeito que possa ter aplicado ao banco de dados devem ser desfeitos.
- **Confirmada** (*commit*) - indica que a transação foi bem-sucedida e todas as suas operações executadas podem ser seguramente confirmadas (*commit*) no banco de dados e não serão desfeitas.

Para assegurar a integridade dos dados, o SGBD usualmente deve garantir as seguintes propriedades das transações, denominadas propriedades **ACID**:

- **Atomicidade** - ou todas as operações da *transação* são refletidas corretamente no banco de dados ou nenhuma será;
- **Consistência** - uma *transação* é preservadora de consistência se a sua execução completa levar o banco de dados de um estado consistente para outro estado também consistente, em que um estado consistente de banco de dados satisfaz às restrições especificadas no esquema, bem como a quaisquer outras restrições que devem se manter no banco de dados;
- **Isolamento** - embora diversas *transações* possam ser executadas de forma concorrente, o sistema garante que, para todo par de *transações* T_i e T_j, T_i executa como se T_j já tivesse terminado sua execução antes de T_i começar, ou como se T_j tivesse começado sua execução após T_i terminar. Assim, cada transação não toma conhecimento de outras transações concorrentes no sistema;
- **Durabilidade** - depois de a *transação* completar com sucesso, através da execução do comando *commit*, as mudanças que ela fez no banco de dados persistem, até mesmo se houver falhas no sistema.

Os sistemas de bancos de dados distribuídos surgem naturalmente como decorrência do avanço das tecnologias de sistemas de banco de dados

e redes de computadores. Um sistema de gerência de banco de dados distribuído (SGBDD) é definido como o software que permite o gerenciamento de bancos de dados distribuídos e que torna essa distribuição transparente para os usuários. Banco de dados distribuído (BDD) é uma coleção de vários bancos de dados logicamente inter-relacionados, distribuídos por uma rede de computadores. O termo sistema de banco de dados distribuído (SBDD) é empregado para se referir, em conjunto, ao banco de dados distribuído (BDD), ao SGBD distribuído e às aplicações que o acessam.

O acesso a diversos itens de dados em um **sistema distribuído** é normalmente acompanhado de transações que são obrigadas a preservar as propriedades ACID. Existem dois tipos de transações que devem ser consideradas nos sistemas distribuídos: as *transações locais* e as *transações globais*. As *transações locais* são aquelas que acessam somente o banco de dados local. Já as *transações globais* são aquelas que acessam diversos bancos de dados locais. A garantia das propriedades ACID nas *transações globais* é bem mais complicada, uma vez que diversos locais podem participar de sua execução. Maiores detalhes sobre as transações de banco de dados nos sistemas centralizados e distribuídos podem ser encontrados em [21], [62], [44], [15] e [6].

2.3 Frameworks

O reuso de software tem sido um dos principais objetivos da engenharia de software. Reutilizar software não é simples. Com o surgimento do paradigma da orientação a objetos, a tecnologia adequada para reuso de grandes componentes tornou-se disponível e resultou na definição de frameworks orientados a objetos. Os frameworks têm atraído a atenção de muitos pesquisadores e engenheiros de software e têm sido definidos para uma grande variedade de domínios. As principais vantagens de um framework são o aumento do reuso e a redução do tempo para desenvolvimento de aplicações. Frameworks são uma forma particular de representar arquiteturas, embora existam arquiteturas que não podem ser representadas como frameworks. Nesta seção são apresentados os principais conceitos relativos aos frameworks, necessários para o entendimento da proposta deste trabalho. Todas as definições apresentadas aqui foram extraídas de [65].

Definições

Na literatura da comunidade de orientação a objetos, encontra-se uma grande variedade de definições para framework, com pequenas diferenças entre elas. Três delas são bastante completas e elucidativas:

1. Bushmann et al. [11] definem framework como um (sub)sistema de software (implementado em uma linguagem de programação), parcialmente completo, criado com o objetivo de ser instanciado. Um framework define uma arquitetura para uma família de sistemas e fornece blocos básicos predefinidos para sua construção. Também define as partes que devem ser adaptadas para realizar uma funcionalidade específica. Em um ambiente orientado a objetos, um framework é composto de classes abstratas e concretas, e sua instanciação consiste em composição e herança de classes. Normalmente, as classes concretas devem ser invisíveis para o usuário do framework.

2. De acordo com Pree [54], um framework consiste em *frozen spots* e *hot spots*. Frozen spots definem a arquitetura global de um sistema de software - seus componentes básicos e os relacionamentos entre eles. Eles permanecem imutáveis em qualquer instanciação do framework. Hot spots representam aquelas partes do framework que são específicas para cada sistema de software. Hot spots são projetados para serem genéricos - eles podem ser adaptados para as necessidades da aplicação em desenvolvimento. Quando se cria um sistema de software concreto, usando um framework, seus hot spots são preenchidos de acordo com as necessidades e requisitos específicos do sistema.

3. Já Appleton [2] diz que um framework de software é uma miniarquitetura reutilizável que fornece a estrutura e comportamento genéricos para uma família de abstrações de software, com um contexto que especifica suas interações e uso dentro de um determinado domínio. Essa especificação é completada pela codificação do contexto, em que as abstrações têm "fim aberto"(open-ended) e são projetadas como plug-points específicos. Esses plug-points (tipicamente implementados usando callback ou polimorfismo) permitem que o framework seja adaptado ou estendido para atender variadas necessidades e que possa ser combinado com outros frameworks. Um framework não é uma aplicação completa: falta a necessária funcionalidade específica de uma aplicação. Em vez disso, uma aplicação pode ser construída a partir de um ou mais frameworks inserindo-se as funcionalidades ausentes nas

lacunas plug-and-play dos frameworks. Enfim, um framework fornece a infra-estrutura e os mecanismos que executam uma política para interação entre componentes abstratos com implementações abertas.

Vantagens da Utilização dos Frameworks

Os principais benefícios dos frameworks orientados a objetos decorrem da modularidade, reusabilidade, extensibilidade e inversão de controle que eles oferecem aos desenvolvedores.

Frameworks aumentam modularidade através do encapsulamento de detalhes voláteis de implementação por trás de interfaces estáveis. A modularidade dos frameworks ajuda a aumentar a qualidade do software, concentrando o impacto das mudanças de projeto e implementação, o que reduz o esforço necessário para entender e manter softwares existentes.

As interfaces estáveis fornecidas pelos frameworks aumentam a reusabilidade, definindo componentes genéricos que podem ser reutilizados para criar novas aplicações. A reusabilidade do framework alavanca o conhecimento do domínio e o esforço anterior dos desenvolvedores, a fim de evitar recriação e revalidação de soluções comuns, recorrendo aos requisitos da aplicação e aos desafios do projeto do software. Frameworks maduros permitem uma redução de mais de 90% do volume de código fonte que tem que ser escrito para desenvolver uma aplicação, quando comparado com software escrito com o suporte de uma biblioteca convencional de funções.

Um framework aumenta a extensibilidade fornecendo métodos "ganchos"(hook) explícitos, que permitem que as aplicações estendam suas interfaces estáveis. Métodos hooks desacoplam, sistematicamente, as interfaces estáveis e os comportamentos de um domínio de aplicação, em um contexto particular. A extensibilidade do framework é essencial para assegurar customização adequada de serviços e características para a nova aplicação.

A arquitetura de run-time de um framework é caracterizada por uma inversão de controle. Quando se usa um framework, usualmente apenas se implementam umas poucas funções de chamada ou se especializam umas poucas classes e, então, se invoca um único método ou procedimento. A partir desse ponto, o framework faz o resto do trabalho, invocando quaisquer chamadas a cliente ou métodos necessários, no tempo e lugar adequados.

2.4 Agentes de Software

Um agente é um sistema de computação situado em algum ambiente computacional, sistema esse que é capaz de executar ações autônomas nesse ambiente, visando atingir os objetivos do projeto para o qual foi desenvolvido. Um sistema de agentes deve poder agir sem a intervenção direta dos seres humanos ou de outros agentes, como também deve possuir todo o controle de suas próprias ações e estado interno. Um agente inteligente é um sistema computacional que é capaz de executar ações autônomas flexíveis, a fim de atingir os objetivos do projeto para o qual foi projetado e desenvolvido. Por flexível, entende-se que o sistema deve ser reativo, pró-ativo e social [32].

A principal característica de um agente é a sua habilidade de agir de forma autônoma. Isso implica que um agente recebe estímulos de seu ambiente, pode executar um conjunto de ações que alteram esse mesmo ambiente, decidindo por essas ações como base em seus próprios objetivos [24]. Os agentes de software podem ser classificados [42]:

1. Por sua mobilidade;
2. Por seus papéis (roles);
3. Por sua capacidade de reação;
4. Pelos diversos atributos que idealmente devem exibir;
5. Pelos agentes híbridos que combinam duas ou mais filosofias do agente em um único agente.

Dado o contexto deste trabalho, serão exploradas, principalmente, as duas primeiras características. Maiores detalhes podem ser encontrados em [17], [32].

Agentes Móveis

Agentes móveis são agentes que se deslocam de um ambiente computacional para outro, executam dentro de um ambiente remoto seguro para obter a informação procurada e, então, são transmitidos de volta para seu ambiente de origem. Agentes estáticos são agentes que emitem mensagens declarativas para outros agentes, freqüentemente utilizando interfaces de comunicação padrões [4].

A diferença entre os agentes móveis e os agentes estáticos não é, entretanto, demasiadamente grande. Sistemas que empregam agentes móveis também empregam agentes estáticos associados com seus ambientes computacionais.

A forma de enviar os agentes móveis para recolher as informações necessárias em um outro ambiente é análoga à forma como um agente estático emite um pedido declarativo a um outro agente, o qual contém a informação desejada. Um sistema que usa agentes móveis requer um ambiente comum para execução do agente durante toda a execução do sistema ou o mínimo possível de compatibilidade para sua execução [24].

2.5 O Ambiente da Computação Móvel

A computação móvel se transformou em uma realidade graças à convergência de duas tecnologias: o surgimento de computadores portáteis com maior poder de processamento e o desenvolvimento de redes de comunicação sem fio mais velozes e confiáveis.

O paradigma da computação móvel tem afetado conceitos e modelos tradicionais em várias áreas da ciência da computação. Por exemplo, na área das redes de comunicação de dados, as redes necessitam estar disponíveis em todos os lugares, pois devem garantir a conectividade aos seus usuários, independentemente de sua localização física. Redes de comunicação que atendam a essa propriedade são chamadas de *ad-hoc networks*. Já na área de engenharia de software, o paradigma da computação móvel introduziu a noção de código móvel, o que significa a capacidade do código de migrar, por exemplo, de um servidor para um cliente móvel, visando executar uma funcionalidade específica. Já na área de banco de dados são introduzidos o conceito e a necessidade de os clientes móveis acessarem seus bancos de dados a partir de qualquer lugar (*anywhere*) e a qualquer momento (*anytime*) [8] e [59].

Conexões sem fio podem existir em diferentes partes de um ambiente das redes de comunicação de dados. Um importante papel das redes que suportam conexões sem fio é a sua capacidade de permitir a conexão de um grande número de endereços fixos e sem fio, na sua infra-estrutura já existente. Existem muitas tecnologias de acessos sem fio para a conexão de usuários móveis ao backbone de redes com grande largura de banda. Tais tecnologias incluem: celular, rádios públicos e privados, redes locais sem fio, sistemas de paging e sistemas de satélites em órbita da Terra [49].

A computação móvel possui uma arquitetura distribuída [44] na qual diversos computadores, geralmente conhecidos como hospedeiros fixos (*fixed hosts*) e estações de base (*base stations*) ou estações de suporte à mobilidade (ESM), são interligados através de uma rede com fio de alta velocidade, exceto nas redes *ad hoc* nas quais não existem as ESM. *Hospedeiros fixos* são computadores de finalidade genérica, que não são equipados para gerenciar unidades móveis, mas podem ser configurados de forma a fazê-lo. *Estações de base* são equipadas com interfaces para as redes sem fio e podem comunicar-se com as unidades móveis para suportar o acesso a dados [21], [33].

Unidades móveis (*mobile units*) são computadores portáteis movidos a bateria, que se movimentam livremente em um domínio geográfico de mobilidade, uma área que é restringida pela limitada amplitude de banda dos canais de comunicação sem fios. Para gerenciar a mobilidade dessas unidades, o domínio de mobilidade geográfica é dividido em domínios menores chamados de *células*. A computação móvel requer que o movimento de unidades seja irrestrito dentro do domínio de mobilidade geográfica (movimento intercelular), permitindo contigüidade de acesso durante o movimento, sem afetar o processo de recuperação de dados [21].

Unidades móveis e **estações de base** se comunicam através de canais sem fio, que possuem largura de banda significativamente menor do que aquela de uma rede com fio. Um canal de conexão do tipo *downlink* é utilizado para enviar dados das estações de base para as unidades móveis, e um canal de conexão do tipo *uplink* é utilizado para enviar dados das unidades móveis para as estações de base [21]. A Figura 2.2, extraída de [18], apresenta uma arquitetura genérica para um sistema de computação móvel.

Neste trabalho, o termo equipamento fixo será usado como sinônimo para hospedeiro fixo e equipamento móvel para unidade móvel e hospedeiro móvel (mobile host).

Dentre os principais requisitos impostos pelas características da computação móvel aos sistemas de software destacam-se [22]:

1. A capacidade de localizar/endereçar elementos móveis;
2. A capacidade de perceber mudanças no ambiente de execução;
3. A capacidade de se autoconfigurar;
4. A capacidade de migrar funcionalidade;
5. O uso eficiente dos recursos no elemento móvel;

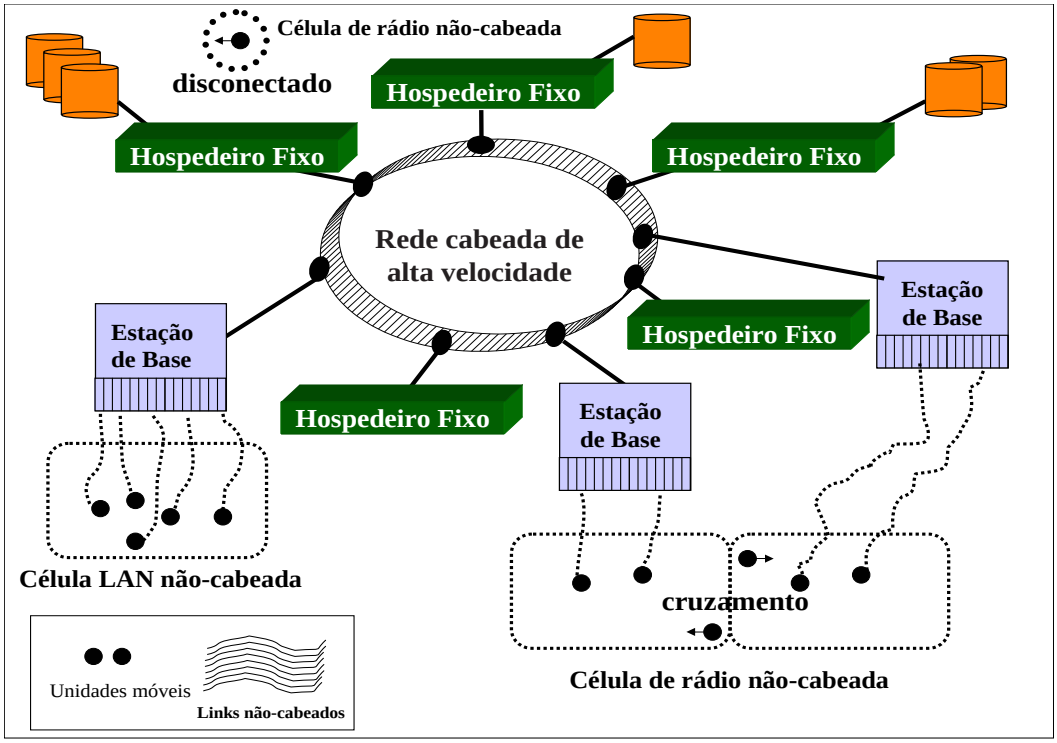

Figura 2.2: Arquitetura de um sistema de computação móvel

6. O uso eficiente dos recursos do canal de comunicação sem fio;
7. O alto grau de tolerância a falhas; e
8. Os mecanismos para autenticação e criptografia de dados.

2.5.1 Problemas e Limitações

Diversas novas características e limitações inerentes aos ambientes de computação móvel precisam ser consideradas nos sistemas de computação móvel, tais como [49], [22] e [58]:

1. **Mobilidade dos equipamentos** - A localização dos elementos móveis e, conseqüentemente, seus pontos de conexão à rede fixa se alteram com seu movimento. As principais conseqüências da mobilidade são:
   (a) A configuração dos sistemas que possuem elementos móveis não é estática:
      i. Os algoritmos tradicionais de processamento distribuído precisam ser reprojetados devido à ausência de uma topologia fixa da rede com os elementos móveis;

ii. A distribuição de carga em uma rede com elementos móveis pode mudar de maneira muito rápida, em função da possibilidade de mudança de localização desses elementos.

(b) Gerência de localização:

i. O custo para localizar um elemento móvel contribui de forma significativa para o custo de cada comunicação;

ii. Estruturas de dados, algoritmos e planos de consultas eficientes devem ser desenvolvidos para representar, gerenciar e consultar a localização dos elementos móveis, uma vez que os dados de localização podem se alterar de forma muito rápida.

(c) Heterogeneidade:

i. A conectividade de um elemento móvel pode variar muito quanto ao seu desempenho, principalmente devido à largura de banda do canal de comunicação e à latência[2] da comunicação, bem como quanto à confiabilidade dessa comunicação;

ii. O número de serviços disponíveis para um elemento móvel pode variar de uma localização para outra, em função das características da célula e da estação de base a qual ele se conecta;

iii. Os recursos disponíveis nos elementos móveis variam, como, por exemplo, a sua capacidade de memória, o tamanho de sua tela, o tempo de duração de sua bateria, etc.

2. **Interface de comunicação sem fio** - O fato de a comunicação entre um elemento móvel e uma estação de base ser através de uma rede de comunicação sem fio tem como principais conseqüências:

(a) Conectividade fraca e intermitente[3]:

i. Oferecem menores larguras de banda;

ii. Possuem uma latência maior do que as redes com fio;

iii. São menos confiáveis que as redes com fio.

(b) Conectividade variável:

i. A qualidade da conexão pode variar abruptamente em função de possíveis interferências na comunicação, da distância entre

[2]Período de inatividade entre um estímulo e a resposta por ele provocada.

[3]Apresenta interrupções ou suspensões; não-contínua.

o elemento móvel e a estação de base na qual está conectado ou devido ao compartilhamento da estação de base por vários elementos móveis;

ii. A tecnologia sem fio se modifica em função do grau da largura da banda de comunicação e da segurança que a rede fornece.

(c) Facilidade para transmissão de dados (broadcast):

i. Existe um canal com grande largura de banda para transmissão de dados da estação de base para todos os clientes móveis em sua célula;

ii. Possibilidade de disseminação de informações a grupos de clientes móveis específicos;

iii. Possibilidade de prestação de serviços específicos para certos grupos de clientes móveis.

(d) Tarifas (custos):

i. As redes sem fio são mais caras;

ii. Em algumas redes, o acesso à rede é pago em função do tempo de conexão, como, por exemplo, nas comunicações via telefones celulares;

iii. Em outras redes, o acesso à rede é pago por mensagens, como, por exemplo, nas comunicações via pacotes de rádio.

3. **Limitações Computacionais** - Por serem portáteis, os elementos móveis precisam ser pequenos e leves, o que os torna inferiores, em suas capacidades de processamento, aos computadores convencionais. Assim:

(a) Elementos móveis são pobres em recursos quando comparados com os elementos estáticos:

i. São equipados, em sua maioria, com pouca memória do tipo RAM (Random Access Memory);

ii. Seus processadores são mais lentos;

iii. Possuem pouca memória não-volátil (podem não possuir disco rígido);

iv. A interface do usuário é mais limitada;

v. O monitor é menor ou tem menor capacidade em seu buffer;

vi. Os dispositivos de entrada de dados também são menores e limitados (por exemplo, teclados de telefones celulares).

(b) Elementos móveis contam com pouca capacidade em suas baterias:

i. Dependem da energia fornecida por baterias;

ii. Normalmente, as baterias possuem capacidade limitada;

iii. Em alguns lugares, dificuldade para recarga da bateria.

(c) Elementos móveis são poucos robustos:

i. Fáceis de serem danificados;

ii. Facilmente perdidos ou roubados.

2.5.2 Gerência de Dados em um Ambiente Móvel

Um número cada vez maior de computadores necessita consultar bancos de dados através de canais de comunicação sem fio. Esses *clientes móveis*, dependendo do contexto, irão freqüentemente desconectar-se de suas redes de comunicação de dados por longos períodos de tempo, desconexão essa causada, entre outros motivos, pela pouca capacidade das baterias de seus equipamentos. Além disso, irão se locomover entre pontos distintos de suas redes, se conectando a diferentes servidores de dados em diferentes momentos [5].

O uso de SGBDs em ambiente de computação móvel pode ser considerado uma extensão dos sistemas distribuídos. O trabalho apresentado em [44] classifica os SGBDs distribuídos com base nas características de *autonomia, distribuição e heterogeneidade* do sistema. A Figura 2.3 apresenta uma extensão dessa classificação, considerando os SGBDs em ambiente de computação móvel, através de um novo ponto no eixo de *distribuição* [18]. A lógica para essa extensão está no fato de que os sistemas de computação móveis devem ter uma rede fixa (Figura 2.2) que é um sistema distribuído.

Considerando os sistemas móveis um caso especial dos sistemas distribuídos, percebe-se que muitos problemas são iguais, porém existem outros consideravelmente diferentes. Os sistemas distribuídos têm como principal objetivo a *transparência de distribuição*, enquanto os sistemas móveis buscam o *conhecimento da localização*, ou seja, saber onde um cliente móvel está localizado geograficamente (espacialmente) para poder manter uma comunicação constante, sem interrupções, com ele.

Dentre as principais características dos SGBDs quando presentes em um ambiente de computação móvel destacam-se:

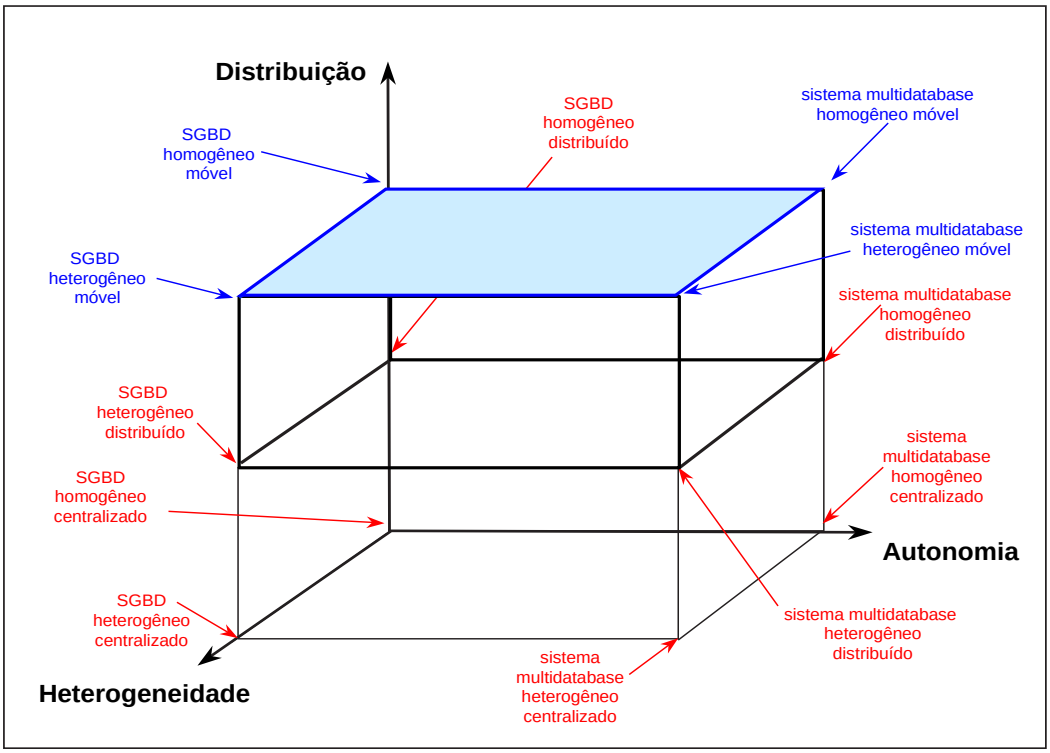

Figura 2.3: Classificação dos SGBDs

- Vários bancos de dados interligados por uma rede de comunicação sem fio;
- Acesso de um computador móvel (*equipamento móvel*) a um banco de dados, residindo em um computador fixo (*equipamento fixo*) ou em um computador móvel;
- O computador móvel pode desempenhar o papel de *cliente* ou de *servidor de banco de dados*;
- Os bancos de dados são autônomos, distribuídos e, provavelmente, heterogêneos.

Quanto à gerência e administração dos dados, diversos fatores podem influenciar o funcionamento e o desempenho dos bancos de dados no ambiente de computação móvel. Entre esses fatores, destacam-se:

- *Velocidade dos links sem fio* - pode ocasionar demora nas consultas;
- *Escalabilidade* - o crescimento dos bancos de dados tem impacto nas consultas e limita a quantidade de dados nos bancos de dados residentes em clientes móveis;
- *Mobilidade* - pode causar desconectividade, necessitando de novos modelos de transações para evitar o cancelamento de transações;

- *Localização dos equipamentos móveis* - implica a necessidade de administrar a localização dos equipamentos móveis, para interação, durante as consultas;
- *Limite da capacidade das baterias* - exigem trabalho em modo desconectado da rede e podem causar desconexões, necessitando de novos modelos de transações;
- *Desconectividade* (voluntária ou involuntária) - geram a necessidade de novos modelos de recuperação (*recovery*) do banco de dados, bem como de transações;
- *Replicação/Caching* - limitadas pelo pouco poder de memória dos equipamentos móveis;
- *Handoff*[4] - aumenta a necessidade de controle e administração da localização dos equipamentos móveis.

Para facilitar o entendimento, é importante analisar o comportamento de algumas características específicas de SGBDs no ambiente de computação móvel. A Tabela 2.1, extraída de [18], apresenta essas características.

Em um ambiente distribuído, o SGBD necessita se recuperar de eventuais falhas em sites, nos meios de comunicação, de transações e de comunicação. Esses mesmos problemas persistem na computação móvel porém, a freqüência pode ser muito maior. Esses problemas complicam a recuperação devido ao fator de mobilidade nas transações.

Transações distribuídas são executadas concorrentemente em múltiplos conjuntos de dados e processadores. As execuções das transações distribuídas são totalmente coordenadas pelo sistema, incluindo o controle de concorrência, a gerência de replicação e a atomicidade das transações. Transações móveis, por outro lado, são executadas seqüencialmente através de múltiplas estações de base, em vários conjuntos de dados, dependendo do movimento da unidade móvel. Assim, a execução da transação móvel na estação de base pode ser considerada uma transação distribuída. A execução da transação é, então, não totalmente coordenada pelo SGBD. O movimento da unidade móvel, por uma grande extensão espacial, controla a execução.

[4]Migrações dos clientes móveis entre as estações de suporte à mobilidade.

Principais diferenças para os BDs móveis	
Característica do SGBD	*Tratamento no Ambiente Móvel*
Aplicações	- Pode ser dependente de localização do equipamento móvel - Necessita adaptar-se às mudanças no contexto do sistema
Transações	- Novos modelos considerando a mobilidade
Recuperação	- Divisão (particionamento) freqüente da rede - Desligamento voluntário do equipamento móvel não é uma falha do sistema - A mobilidade pode causar inúmeros acessos (logins) ao sistema - Técnicas para recuperação de desconexões durante o handoff
Replicação	- Diferentes restrições de consistência - Novas técnicas para os equipamentos móveis atualizarem suas memórias cache durante as freqüentes desconexões
Processamento de consulta	- Dependente da localização do equipamento móvel - Diferentes fatores de custos - Resposta das consultas retornam para diferentes localizações - Necessidade de adaptações das técnicas
Resolução de nomes (endereços)	- Estratégia de nome global devido à desconectividade e mobilidade

Tabela 2.1: Características específicas da gerência de dados móveis

2.5.3 O Modelo Cliente/Servidor e Suas Extensões para a Computação Móvel

O projeto de modelos apropriados para aplicações que envolvem elementos sem fio é uma questão fundamental no desenvolvimento do software para a computação móvel. A seguir, são apresentados um breve resumo sobre o modelo cliente/servidor, bem como algumas extensões desse modelo para a computação móvel.

O Modelo Cliente/Servidor

O modelo *cliente-servidor*, também conhecido por arquitetura cliente/servidor, supõe uma estrutura básica que consiste em muitos microcomputadores (PCs) e estações de trabalho e de um número menor de equipamentos de maior porte, conectados através de redes de comunicação. Um **cliente** nessa arquitetura é, geralmente, um equipamento que possui ca-

pacidades de interface com usuário e processamento local. Quando uma aplicação do cliente solicita acesso a funcionalidades adicionais, como, por exemplo, acesso a um banco de dados, normalmente o cliente se conecta a um **servidor**, que fornece a funcionalidade necessária. Um **servidor** é um equipamento que pode fornecer serviços para as estações clientes, como impressão, armazenamento de dados ou acesso ao banco de dados [20].

A idéia da arquitetura *cliente-servidor* é dividir os serviços que precisam ser fornecidos aos usuários do SGBD em duas camadas, a camada *servidor* e a camada *cliente*, ambos com funcionalidades distintas. Essa arquitetura facilita o gerenciamento da complexidade dos SGBDs atuais [44]. A arquitetura cliente-servidor está incorporada aos SGBDs. Nos clientes estão os programas de aplicação e a interface do usuário, enquanto as funcionalidades de processamento de consultas e transações permanecem no lado do servidor. Quando é necessário acesso ao banco de dados, a aplicação solicita uma conexão com o SGBD, que está no lado do servidor, e, uma vez que a conexão é estabelecida, os programas no cliente podem se comunicar com o SGBD. Os resultados das consultas são enviados de volta para os programas no cliente, que podem processar ou exibir os resultados [20].

Conforme mostra a Figura 2.4, extraída de [44], a camada servidor em geral faz a maior parte do trabalho de gerenciamento de dados, enquanto a camada cliente cuida da interface do usuário e da aplicação, além de administrar uma memória local para solicitação e armazenamento do resultado das consultas.

Diferentes tipos de arquiteturas cliente/servidor podem ser implementados. A arquitetura mais simples é aquela em que vários clientes acessam um único servidor, denominada *múltiplos clientes-único servidor*. A arquitetura mais complexa é aquela em que o ambiente possui *múltiplos clientes-múltiplos servidores* [44] e [40].

O Modelo Cliente/Servidor para Ambiente de Computação Móvel

Na computação móvel, o ***equipamento móvel*** atua como um ***cliente***, requisitando serviços dos servidores localizados na rede fixa. Nesse caso, os dados e as funcionalidades estão distribuídos através de vários ***servidores*** em diferentes ***equipamentos fixos***, que podem comunicar-se entre si para atender às solicitações dos clientes. Em muitos casos, o servidor é replicado em diferentes sites, na rede fixa, para aumentar a disponibilidade nos casos de falhas nos sites, ou na rede de comunicação. A Figura 2.5,

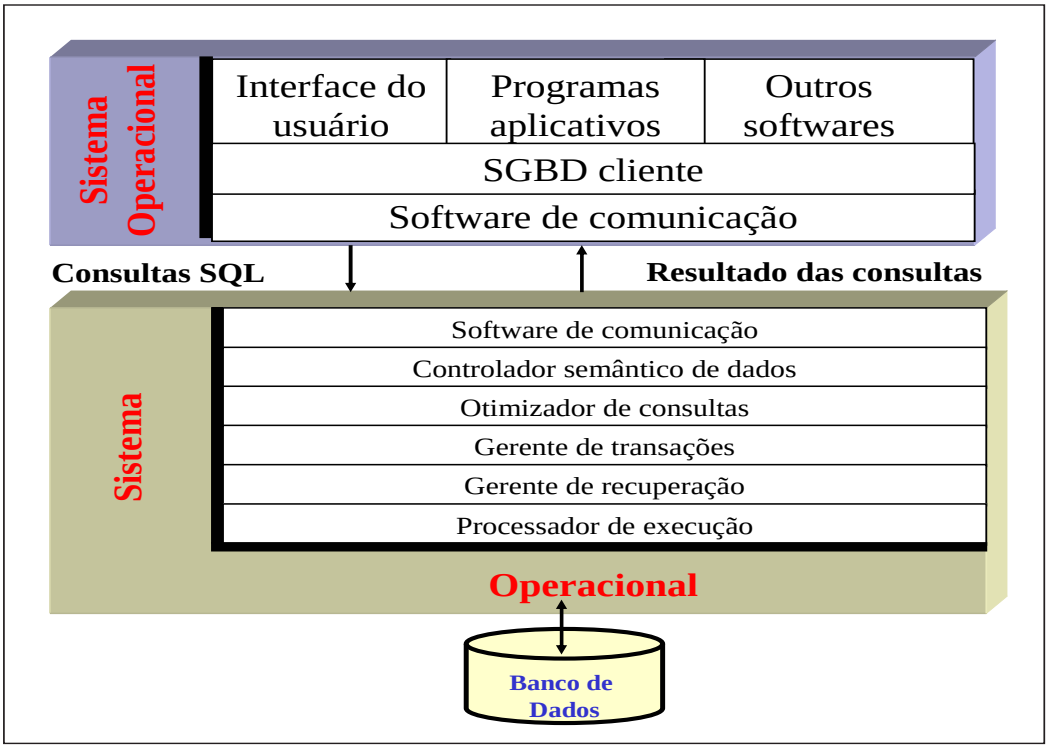

Figura 2.4: Componentes do modelo cliente/servidor para SGBDs

extraída de [51], apresenta um ambiente de computação móvel em uma arquitetura cliente/servidor tradicional.

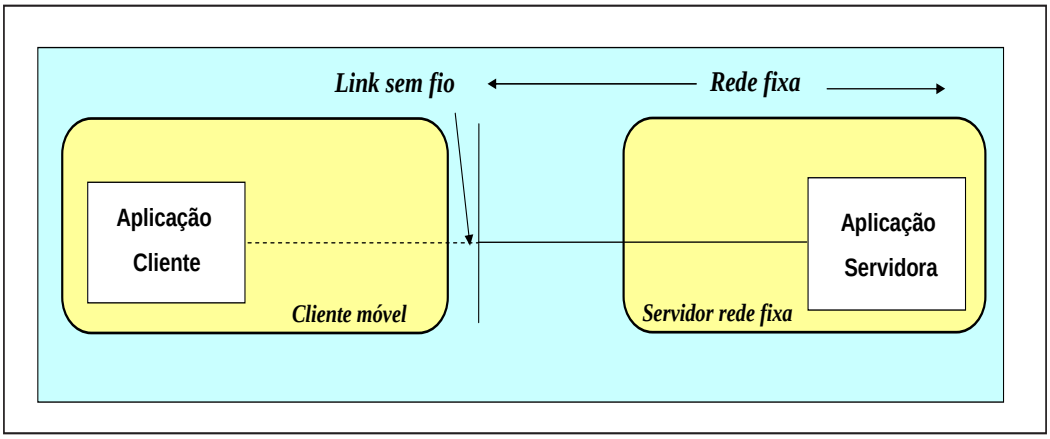

Figura 2.5: Ambiente de computação móvel em uma arquitetura cliente/servidor convencional

A divisão das funcionalidades entre os *clientes móveis* e os *servidores fixos*, que estão nas redes fixas, não é muito precisa. Em muitos casos, os papéis parecem sobrepostos, como durante as desconexões, em que o cliente móvel, para continuar operando, necessita emular as funcionalidades de um servidor.

Um importante componente da arquitetura cliente/servidor é o tipo de mecanismo de comunicação que é usado para a troca de informações entre o cliente e o servidor. Uma possibilidade é a troca direta de mensagens entre o cliente e o servidor. Essa abordagem não é adequada para redes lentas e pouco confiáveis, como é o caso da computação sem fio, e novos mecanismos de troca de informações estão sendo estudados.

Extensões do modelo cliente/servidor tradicional, como enfileirar mensagens RPC (Remote Procedure Call), são necessárias para dar apoio a operações desconectadas e à fraca conectividade. Otimizações adicionais como compressão e filtros de dados também são importantes. Por esses motivos, o modelo cliente/servidor tradicional deve ser estendido para prover componentes para a implementação de otimizações adequadas, com o mínimo possível de mudanças nos clientes e nos servidores [49].

O Modelo Cliente/Agente-Servidor/Servidor

O modelo cliente/agente-servidor/servidor, considerado um modelo em três camadas, realiza a comunicação entre o cliente móvel e o servidor, através da troca de mensagens do cliente com o agente[5] e desse com o servidor. Genericamente, podemos dizer que o agente assume o papel de substituto (*surrogate*) do cliente móvel na rede fixa. Essa arquitetura alivia o impacto da limitação da largura de banda e da pouca segurança do link sem fio, através da presença do cliente móvel na rede fixa, via o agente. Os agentes, tipicamente, dividem a interação entre os clientes móveis e os servidores fixos em duas partes, uma entre o cliente e o agente e a outra entre o agente e o servidor fixo. Assim, diferentes protocolos podem ser usados para interação em cada parte, e essas partes podem executar suas funcionalidades independentemente. A Figura 2.6, extraída de [51], apresenta essa arquitetura.

Esse modelo é mais apropriado para clientes móveis com recursos limitados. Assim, diversas responsabilidades dos clientes móveis migram para os agentes. Por exemplo, uma consulta com grande quantidade de dados como resposta pode ser gerenciada pelo agente e somente ao seu final ter seus resultados transmitidos para o cliente móvel. A localização do agente na rede fixa possibilita o acesso à rede com grande largura de banda e grande poder computacional, podendo usar esses recursos em favor dos

[5] A palavra agente, neste contexto, não tem o mesmo significado da palavra agente, discutida na seção Agentes de Software. Neste contexto, agente significa simplesmente um módulo de software.

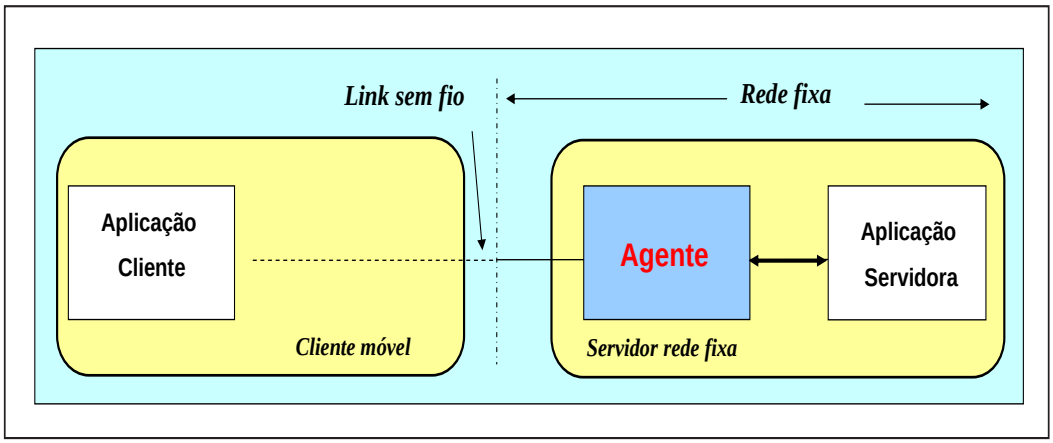

Figura 2.6: Modelo em três camadas cliente/*agente-servidor*/servidor

clientes móveis. Por exemplo, a compressão dos dados pode ser executada no agente, antes da transmissão dos dados para os clientes móveis.

Para lidar com a *desconexão*, os clientes móveis podem submeter suas solicitações aos agentes e esperar a devolução dos resultados quando a conexão for restabelecida. Durante a desconexão, várias solicitações dos clientes móveis podem ser processadas e enfileiradas para serem transmitidas aos clientes móveis, logo após a nova conexão.

Essa estratégia também pode ser usada pelos clientes móveis para preservar o tempo útil de suas baterias. Os clientes móveis submetem suas solicitações, entram em modo de *cochilo (doze)* e aguardam a espera das respostas dos agentes. A fraca conectividade também pode ser efetivamente manuseada pelos agentes de várias formas, empregando inúmeras técnicas para minimizar o tamanho dos dados a serem transmitidos para os clientes móveis, dependendo do tipo de dados e das aplicações. O agente também pode manipular os dados antes da sua transmissão, alterando a sua ordem de transmissão em função da prioridade dos mesmos para os clientes móveis.

Os agentes podem ser usados de várias formas, desempenhando diversos papéis nessa arquitetura. A exata posição dos agentes, na rede fixa, depende do papel que venham a desempenhar. Colocá-lo na extremidade da rede fixa, como, por exemplo, nas *estações de base*, traz algumas vantagens quando o agente atua como substituto do cliente móvel em sua área de cobertura. Assim, fica mais fácil reunir informações das características do link sem fio, utilizar um protocolo especial entre o cliente móvel e o agente e personalizar informações sobre o cliente móvel que podem ser disponibilizadas localmente.

O modelo *cliente/agente-servidor/servidor* oferece um grande número de vantagens, porém falha no suporte da operação dos clientes móveis

durante o período de desconexão. Quando acontece uma desconexão, o cliente móvel não continua a operar ininterruptamente. O agente somente consegue otimizar a transmissão de dados do agente para o cliente móvel, mas não no caminho inverso [49].

O Modelo Cliente/Agente-Cliente/Servidor

O modelo cliente/agente-cliente/servidor apresenta uma extensão do modelo cliente/servidor também em três camadas, porém com a inclusão do *agente de software* junto ao equipamento móvel, ou seja, no cliente móvel. A Figura 2.7, extraída de [51], apresenta essa extensão. Genericamente, podemos dizer que o agente assume o papel de ampliar as funcionalidades dos clientes móveis, geralmente pobres em recursos computacionais.

Dentre as muitas atividades específicas destacam-se:

- Administrar a memória *cache* no cliente móvel;
- Disponibilizar memória progressivamente para o cliente móvel durante o pouco tráfego da rede (*prefetching*[6]);
- Copiar parte do banco de dados para a memória do cliente móvel (*hoarding*[7]), caracterizando um caso especial de replicação;
- Otimizar a comunicação entre o cliente móvel e sua estação de base.

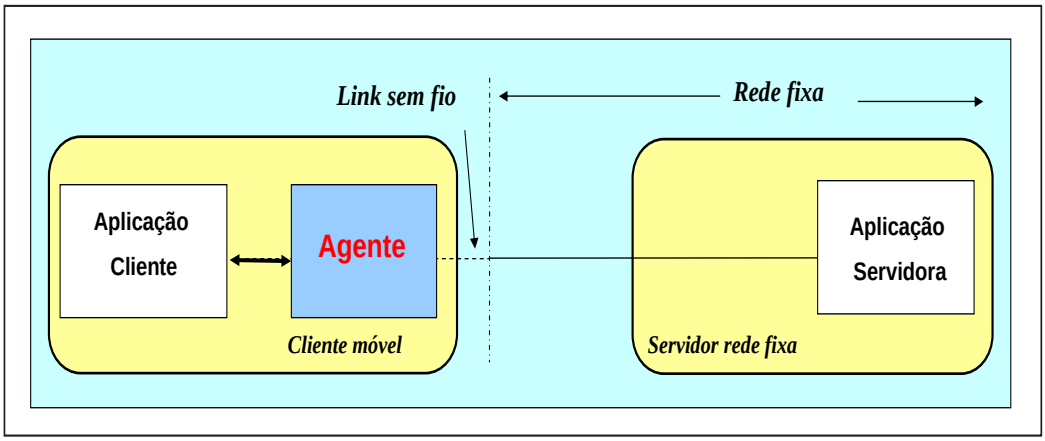

Figura 2.7: O modelo em três camadas cliente/*agente-cliente*/servidor

[6] *Prefetching* é um processo progressivo de transferir dados para a memória cache do equipamento móvel, o mais cedo possível, durante os períodos de baixo tráfico na rede.

[7] *Hoarding* é um processo de carga antecipada dos dados, para garantir a sobrevivência do equipamento móvel durante a desconectividade que está por vir.

O Modelo Cliente/Agente-Agente/Servidor

Para sanar as deficiências dos dois modelos anteriores, nos quais os agentes de software residem e executam em apenas um dos lados da arquitetura, ou seja, no cliente móvel ou no servidor da rede fixa, os autores de [57] propuseram o modelo cliente/agente-agente/servidor, também chamado de *client/intercept/server*, no qual o agente, situado no cliente móvel, executará em conjunto com o agente situado no servidor. O agente situado no cliente móvel detecta (intercepta) as solicitações do cliente e, juntamente com o agente situado no servidor, executa otimizações para redução da quantidade de dados transmitidos na rede sem fio, melhorando a segurança na transferência dos dados e sustentando a não-interrupção da computação móvel. A Figura 2.8, extraída de [51], apresenta esse modelo.

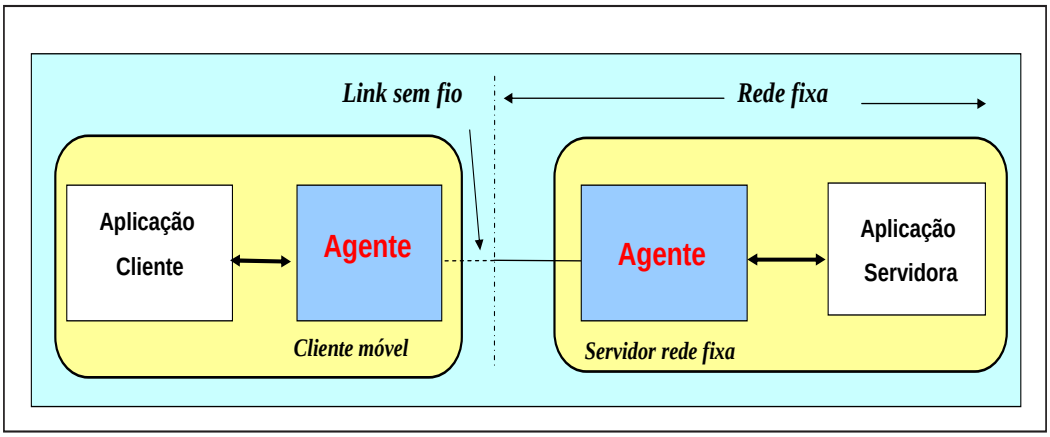

Figura 2.8: O modelo em três camadas cliente/*agente-agente*/servidor

Esse modelo é transparente tanto para o cliente como para o servidor. Conseqüentemente, o par de agentes pode ser empregado em várias aplicações clientes. O protocolo de comunicação entre os dois agentes pode proporcionar uma alta eficiência na redução (compressão) dos dados e na otimização de protocolos, sem limitar as funcionalidades ou interoperabilidade do cliente. A cooperação entre os dois agentes permite uma maior otimização nos links sem fio, beneficiando diferentes aplicações. Além disso, otimizações específicas para algumas aplicações podem ser efetivamente realizadas pelos agentes.

Esse modelo oferece flexibilidade no manuseio da desconexão. Por exemplo, a memória *cache* dos clientes móveis pode ser mantida pelo seu agente. Essa memória pode ser usada para atender às necessidades e solicitações dos clientes durante o período de desconexão. Na falta de memória cache, o agente pode enfileirar solicitações para serem enviadas

ao servidor por ocasião da reconexão. Similarmente, o agente situado no servidor pode acumular e transferir requisições do cliente quando de uma nova reconexão. A fraca conectividade também pode ser manuseada de várias formas, como, por exemplo, por uma relocação computacional do agente do cliente ao agente do servidor ou vice-versa. O retardo de gravação e disponibilidade de memória progressivamente para o cliente móvel (*prefetch*) também reduz a comunicação durante o período de fraca conectividade ou nos momentos de inatividade da rede.

Esse modelo é apropriado para clientes móveis com suficiente poder computacional e que possuam memória secundária. O ponto fraco desse modelo é a necessidade de desenvolvimento de agentes tanto para o servidor quanto para o cliente. Porém, não é necessário o desenvolvimento do par de agentes para cada nova aplicação. Construindo-se funcionalidades e otimizações suficientemente genéricas, somente será necessário o desenvolvimento de pares de agentes para aplicações de tipos diferentes como de acesso a banco de dados, aplicações Web, etc.

O modelo cliente/*agente-agente*/servidor provê uma clara distinção entre os agentes do cliente e do servidor, bem como divide a responsabilidade da execução das tarefas necessárias ao ambiente de computação móvel entre os mesmos. Um exemplo da aplicação desse modelo é apresentado em [67].

2.5.4 O Modelo de Agentes Móveis

Dois fatores principais motivam a utilização dos agentes móveis na computação móvel. Em primeiro lugar, os agentes móveis oferecem um método assíncrono eficiente para procurar informações ou serviços, de forma rápida, em uma rede. Clientes móveis são lançados na rede e procuram em toda parte pela informação ou serviço desejado. Em segundo lugar, os agentes móveis suportam a conectividade intermitente, as redes lentas e os equipamentos com poucos recursos. Esse segundo fator torna a utilização dos agentes móveis muito atraente para a computação móvel.

Durante um *período de desconexão*, o cliente móvel envia um agente móvel para a rede fixa, e esse agente age como um substituto da aplicação na rede fixa, permitindo interação durante a desconexão. De forma oposta, o agente móvel é carregado no cliente móvel, oriundo da rede fixa, quando acontece uma nova conexão. Quando em *fraca conectividade*, melhora o tráfego de comunicação através do link sem fio, reduzindo a submissão de um grande número de mensagens pelos agentes fixos e a obtenção de

seus resultados. Por esse motivo, os agentes móveis deslocam a carga de computação dos clientes móveis, pobres em recursos, para a rede fixa [49].

Um dos principais obstáculos para a utilização dos agentes móveis em aplicações comerciais é a questão da segurança.

2.5.5 Resumo do Capítulo

Este capítulo apresentou os principais conceitos necessários ao entendimento desta tese. Inicialmente, foram apresentados os conceitos relativos ao ambiente de dados, abordados no modelo de computação proposto. Em seguida, foram enunciados os conceitos de framework, destacando suas definições e vantagens, e de agentes de software, com destaque para os agentes móveis, visto que a arquitetura proposta neste trabalho é um framework composto de agentes de software, inclusive do tipo móvel. Logo a seguir, discutiram-se os conceitos do ambiente da computação móvel, destacando-se a gerência de dados, seus problemas e limitações e as arquiteturas, já que o objetivo desta tese é a integração de SGBD a um ambiente de computação móvel. Maiores detalhes sobre o funcionamento do ambiente de computação móvel podem ser encontrados no Anexo A.

No próximo capítulo, serão apresentados diversos modelos de transações para o ambiente da computação móvel, destacando como as propriedades de *Atomicidade, Consistência, Isolamento e Durabilidade (ACID)* são tratadas por cada um deles.