Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura de Coleta de Dados para Pesquisas de Campo em Ambientes Computacionais Heterogêneos

Henrique Pereira de Freitas Filho

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Coorientador
Prof. Dr. Henrique LLacer Roig

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Alba Cristina M. de Melo

Banca examinadora composta por:

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora) — CIC/UnB
Prof.ª Dr.ª Genaina Nunes Rodrigues — CIC/UnB
Prof.ª Dr.ª Lourdes Mattos Brasil — FGA/UnB

CIP — Catalogação Internacional na Publicação

Freitas Filho, Henrique Pereira de.

Arquitetura de Coleta de Dados para Pesquisas de Campo em Ambientes Computacionais Heterogêneos / Henrique Pereira de Freitas Filho. Brasília : UnB, 2014.
117 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2014.

1. Coleta de dados, 2. pesquisas de campo, 3. arquitetura,
4. ambientes computacionais heterogêneos.

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura de Coleta de Dados para Pesquisas de Campo em Ambientes Computacionais Heterogêneos

Henrique Pereira de Freitas Filho

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof.ª Dr.ª Genaina Nunes Rodrigues
CIC/UnB

Prof.ª Dr.ª Lourdes Mattos Brasil
FGA/UnB

Prof.ª Dr.ª Alba Cristina M. de Melo
Coordenadora do Programa de Pós-Graduação em Informática

Brasília, 30 de Janeiro de 2014

Dedicatória

Dedico este trabalho a minha orientadora Prof.ª Dr.ª Maristela Terto de Holanda que teve um papel importantíssimo na minha vida porque ela foi mais que uma professora para mim, foi uma amiga, foi uma conselheira, ou seja, foi uma mãe. Eu decidi me tornar professor pelo seu exemplo e dedicação. Muito obrigado professora!

Agradecimentos

A Deus, pois com ele tudo é possível. Aos meus pais, que se doaram por inteiro e muitas vezes renunciaram aos próprios sonhos para que eu pudesse realizar os meus. Aos meus irmãos e demais familiares que me acompanharam nessa caminhada. À Universidade de Brasília e aos professores que contribuíram para a minha formação. A minha querida orientadora Professora Maristela, pela ajuda inestimável, pelos esclarecimentos e pelas valiosas sugestões.

Resumo

O ambiente computacional baseado na comunicação sem fio, tornou possível o acesso à informação em qualquer lugar e a qualquer momento, o que é favorável para a coleta de dados em pesquisas de campo. A maioria das arquiteturas de coleta de dados existentes atendem uma causa específica, utilizando tecnologias específicas que são limitadas no que se refere ao tipo de dados, tipo de redes, tipo de sincronização e tipo de dispositivo. Esta pesquisa apresenta uma arquitetura de coleta de dados para pesquisas de campo que funciona em ambientes computacionais heterogêneos e que suporta dados geográficos vetoriais. A arquitetura foi implementada e validada em um estudo de caso realizado no Instituto de Geociências (IG) da Universidade de Brasília (UnB).

Palavras-chave: Coleta de dados, pesquisas de campo, arquitetura, ambientes computacionais heterogêneos.

Abstract

The computational environment based on wireless communication, made possible to information access anywhere and any time, which is favourable for collecting data in field research. Most of collecting existing data architectures meet a specific cause, using specific technologies that are limited with regard to the type of data, type of network, type of synchronization and device type. This research presents an architecture of data collection for field researches that works in heterogeneous computational environments and supports vector spatial data. The architecture was implemented and validated in a case study conducted at the Institute of Geosciences (IG) of the University of Brasilia (UnB).

Keywords: Data collection, field researches, architecture, heterogeneous computational environments.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ACID	Atomicidade, Consistência, Isolamento e Durabilidade, 16
API	*Application Programming Interface*, 44
CRUD	*Create*, *Retrieve*, *Update*, *Delete*, 34
FUNAMBOL	*Funambol Data Sync Server*, 38
GPRS	*General Packet Radio Service*, 23
GPS	*Global Positioning System*, 6
GSM	*Global System for Mobile Communications*, 23
IG	Instituto de Geociências, ii
INDE	Infraestrutura Nacional de Dados Espaciais, 8
NDG	*Nokia Data Gathering*, 23
ODBC	*Open Database Connectivity*, 23
ODK	*Open Data Kit*, 24
OMA	*Open Mobile Alliance*, 21
OMT-G	*Object Modeling Technique for Geographic Applications*, 8
PDA	*Personal Digital Assistants*, 9
SADI-AGROMET	Sistema para Aquisição e Disseminação de Informações AgroMeteorológicas, 26
SCDM	Sistema de Coleta de Dados Móveis, 4
SGBD	Sistema Gerenciador de Banco de Dados, 7
SGBD-G	Sistema Gerenciador de Banco de Dados Geográfico, 7
SGBD-OR	Sistema Gerenciador de Banco de Dados Objeto-Relacionais, 7
SGBD-R	Sistema Gerenciador de Banco de Dados Relacionais, 7
SIG	Sistema de Informação Geográfica, 2

UML *Unified Modeling Language*, 8
UnB Universidade de Brasília, ii

W3C *Word Wide Web Consortium*, 20
WAP6 *Wireless Application Protocol* 6, 20
WBXML *Wap Binary eXtensible Markup Language*, 20

XML *eXtended Markup Language*, 21

Capítulo 1

Introdução

1.1 Contextualização

O avanço tecnológico das últimas décadas, associado à tecnologia de comunicação sem fio, torna possível o acesso às informações em qualquer lugar e a qualquer momento, sendo um ambiente propício para o desenvolvimento de sistemas computacionais visando à coleta de dados em pesquisas de campo.

A coleta de dados em campo é importante em diferentes áreas do conhecimento, desde pesquisa sobre dados populacionais até as de dados geológicos. A arquitetura proposta neste trabalho deve atender a esses diferentes tipos de pesquisas. Uma questão importante das investigações sobre dados geológicos que deve ser observada é o tratamento de dados geográficos.

O ambiente de computação móvel permite metodologias de coleta de dados onde pesquisadores de campo podem coletar os dados e submetê-los para outras máquinas na rede, permitindo que a equipe de análise trabalhe paralelamente sobre os dados já coletados. Esse trabalho simultâneo entre a equipe de coleta e a equipe de análise de dados é importante em pesquisas de campo que envolve muitos lugares e possuem um tempo longo de duração, uma vez que, ao identificar um erro na pesquisa, os pesquisadores em campo podem ser informados.

Um dos desafios para a implementação de uma arquitetura de coleta de dados é garantir o estado consistente dos bancos de dados envolvidos no sistema, onde as unidades móveis de coleta de dados têm seus bancos de dados, porém as informações coletadas devem ser armazenadas também em um banco de dados central. O banco de dados central é um elemento fundamental para a análise em tempo real dos dados coletados.

A maioria das arquiteturas de coleta de dados existentes funcionam em um ambiente computacional que utiliza tecnologias específicas que são limitadas no que se refere, por exemplo, ao tipo de dados, tipo de redes, tipo de sincronização, tipo de dispositivo e atendem uma necessidade de coleta específica. A presente pesquisa propõe uma arquitetura de coleta de dados em um ambiente computacional heterogêneo onde as informações armazenadas em cada dispositivo móvel são replicadas e corretamente sincronizadas em um banco de dados central. A arquitetura proposta não só coleta dados convencionais, mas também dados geográficos vetoriais, porque a demanda por esse tipo de dado é alta nas pesquisas em geologia, geociência e sensoriamento remoto espacial.

1.2 Objetivos

O objetivo geral deste trabalho é especificar e implementar uma arquitetura de coleta de dados para pesquisas de campo em um ambiente computacional heterogêneo.

1.2.1 Objetivos Específicos

No intuito de atingir o objetivo geral, foram definidos alguns objetivos específicos:

- Definir os aspectos necessários que uma arquitetura de coleta de dados para pesquisas de campo deve ter, de forma que funcione em ambientes computacionais heterogêneos;
- Especificar e implementar uma arquitetura de coleta de dados para pesquisas de campo em ambientes computacionais heterogêneos;
- Validar a arquitetura em diferentes ambientes computacionais.

1.3 Estrutura do Trabalho

Este documento está divido nos capítulos apresentados a seguir:

- Capítulo 2 apresenta a fundamentação teórica necessária para o desenvolvimento da pesquisa e algumas arquiteturas de coleta de dados existentes. Os principais conceitos, características e desafios de um sistema de coleta de dados móveis são descritos e discutidos e alguns exemplos são mostrados. Os temas que são abordados nesse capítulo envolve, Sistemas de Informação Geográfico (SIG), computação móvel, gerenciamento de banco de dados móveis, tecnologias de sincronização e arquiteturas de coleta relacionadas com a pesquisa;
- Capítulo 3 traz o mapeamento do processo de coleta de dados em pesquisas de campo a nível de usuário e a especificação da arquitetura de coleta de dados para pesquisas de campo proposta;
- Capítulo 4 apresenta uma prova de conceito da proposta através da implementação de uma arquitetura de coleta de dados para o Instituto de Geociência da UnB e uma comparação com as arquiteturas relacionadas;
- Capítulo 5 mostra as conclusões da pesquisa obtidas através dos resultados dos testes realizados na prova de conceito e os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, a fundamentação teórica para o desenvolvimento da pesquisa e algumas arquiteturas de coleta de dados existentes são apresentadas. O capítulo é dividido nas seguintes seções: 2.1 Sistema de Coleta de Dados Móveis onde são apresentados as principais características de um sistema de coleta de dados móveis, 2.2 Sistema de Informação Geográfico que aborda os conceitos e as característica de um sistema de informação geográfico, 2.3 Computação Móvel que apresenta a topologia, as características e as restrições de um ambiente computacional móvel, 2.4 Gerenciamento de Banco de Dados Móveis que mostra as características e os aspectos que devem ser tratados no gerenciamento dos banco de dados móveis, 2.5 Tecnologias de Sincronização que fala sobre a replicação de dados, os tipos de sincronização e os protocolos de sincronização que podem ser usados em um sistema de coleta de dados móveis, 2.6 Arquiteturas Relacionadas que descreve algumas arquiteturas de coleta de dados existentes de mercado e do meio acadêmico e 2.7 Conclusão que apresenta uma síntese do capítulo.

2.1 Sistema de Coleta de Dados Móveis

Antigamente, a coleta de dados em trabalhos de campo era realizada através de questionários de papel, onde o pesquisador escrevia os dados coletados e posteriormente digitalizava esses dados para realizar o seu processamento. Hoje em dia, as coletas de dados vem sendo realizadas com dispositivos móveis (*notebooks*, *smartphones*, *tablets*) a fim de agilizar o processo de coleta e processamento dos dados.

Os questionários de papel geram um grande volume de dados nas pesquisas e esses dados precisam ser transportados fisicamente até o local onde eles serão processados. O trabalho realizado pelos pesquisadores é dobrado para o processamento dos dados (coleta realizada no papel e depois a digitalização desses dados para o seu processamento), é inviável realizar a coleta e o processamento simultaneamente e são vulneráveis no que se refere a segurança da informação.

Atualmente, ao invés do papel, muitas pesquisas tem sido realizadas de maneira eletrônica. Apesar da coleta realizada em meio eletrônico precisar de um sistema de informação, uma arquitetura de coleta de dados, mecanismos de segurança da informação e aparelhos eletrônicos para a coleta, o custo da coleta de dados utilizando meios eletrônicos é 25% mais barato para pesquisas de grandes dimensões de acordo com Thriemer [9].

Se hoje é uma tendência coletar dados utilizando meios eletrônicos, no futuro essa tendência será um padrão porque além de serem mais eficientes, tem um menor custo, logo, o custo benefício de se coletar dados utilizando meios eletrônicos é bem maior do que utilizando questionários de papel.

O Sistema de Coleta de Dados Móveis (SCDM) permite a coleta e a transmissão de dados de localizações geográficas remotas para repositórios centrais de armazenamento de dados através de rede de comunicação sem fio. Sendo assim, o SCDM é uma combinação de uma aplicação cliente, rodando em dispositivos móveis, uma infraestrutura de rede sem fio e servidores de banco de dados remotamente acessíveis [10].

A maioria desses sistemas compartilham orientações e princípios semelhantes para coletar dados remotamente. Como mostrado na Figura 2.1, o processo começa através da concepção de um formulário, que contém um conjunto de questões para a coleta dos dados relevantes. Os coletores usam esses formulários em seus dispositivos móveis para coletar dados reais em campo. Os dados dos formulários preenchidos são armazenados no dispositivo móvel até que seja possível enviá-los para o servidor central [10].

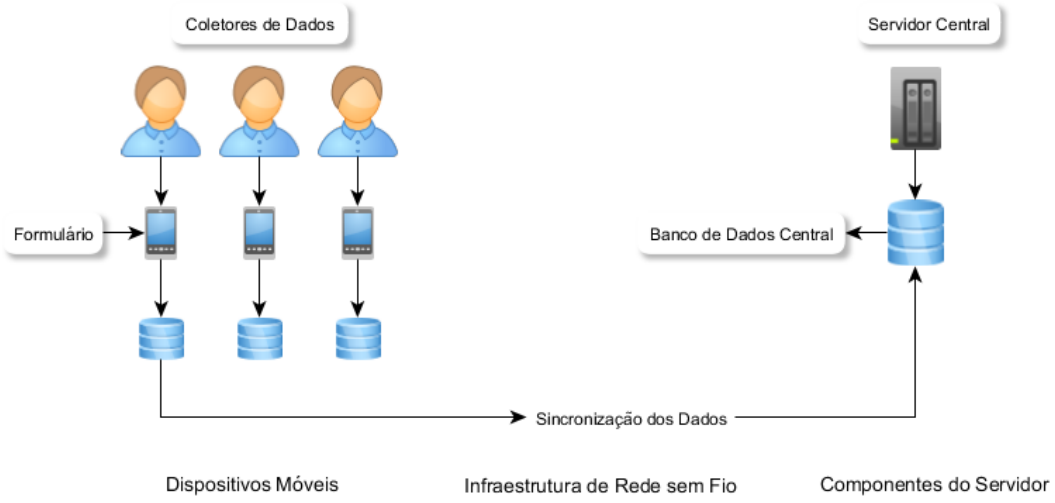


Figura 2.1: Diagrama de fluxo de dados de um sistema de coleta de dados móveis.

Esses sistemas lidam com um volume de dados que podem ser sensíveis e privados, por isso, eles devem fornecer um nível adequado de proteção aos dados em todos os momentos: quando eles estão sendo coletados e armazenados no dispositivo móvel e quando eles estão sendo transferidos e armazenados no servidor [10].

Devido às características de um sistema de coleta de dados móveis, alguns desafios devem ser levados em consideração para que o objetivo desses sistemas seja cumprido, ou seja, para que os dados coletados sejam corretamente sincronizados e armazenados em um servidor central. Os desafios são:

- Um SIG, dependendo do tipo de dado a ser coletado, deve ser desenvolvido para que ocorra a coleta de dados através de um dispositivo móvel. A Seção 2.2 trata sobre esse desafio;
- Por funcionar em um ambiente móvel, as características, as restrições e os modelos de comunicação da computação móvel devem ser levados em consideração no desenvolvimento de um SCDM. Na Seção 2.3 esse tema é abordado;

- Ao se tratar de banco de dados em um SCDM, uma questão fundamental que deve ser observada é o gerenciamento dos dados em um ambiente distribuído e heterogêneo com mobilidade. Na Seção 2.4 conceitos fundamentais desse tema é apresentado;
- Um ambiente móvel com acesso intermitente a *internet* dificulta a manutenção da consistência dos dados armazenados no dispositivo móvel e no servidor de banco de dados central, por isso, a sincronização dos dados nesses sistemas é um desafio que deve ser tratado. A Seção 2.5 lista esse desafio.

2.2 Sistema de Informação Geográfico

Dado pode ser definido como uma sequência de símbolos que são extraídos por meio de algum método científico aplicado a alguma entidade do universo [11] [12]. Números obtidos através de uma pesquisa, conjunto de letras quaisquer e sons da natureza são exemplos de dados.

Pode-se definir informação como dados apresentados de uma forma significativa e útil para os seres humanos [13]. Para gerar essa forma significativa é necessário um processamento nos dados, ou seja, interpretá-los sob uma perspectiva contextual [11] [12].

Um sistema é um conjunto de elementos ou componentes independentes que interagem para atingir um objetivo, o que caracteriza um sistema são os seus componentes e as relações entre eles para alcançar uma meta comum. Um sistema pode ser composto por componentes ou por subsistemas, que são sistemas em escala menor [14].

Segundo Bertalanffy em [14], "Um sistema de informação é uma série de elementos ou componentes inter-relacionados que coletam, manipulam, armazenam e disseminam os dados e informações local ou remotamente e podem fornecer um mecanismo de *feedback*". De acordo com a Figura 2.2, pode-se perceber que um sistema de informação tem uma entrada de dados, um processamento desses dados e uma saída, que é uma informação de um determinado contexto [15].

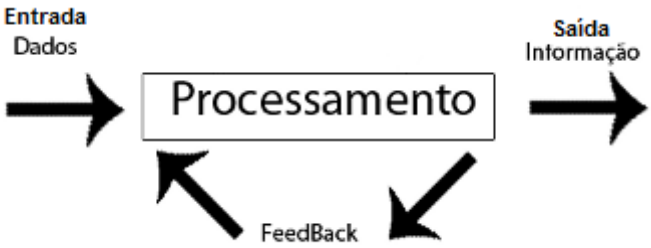


Figura 2.2: Estrutura básica de um Sistema de Informação.

Para entender o conceito de sistema de informação geográfica é importante inicialmente, apresentar os conceitos de geografia, dados geográficos e representação. Geografia é o estudo da superfície terrestre e dos elementos que a compõe [16], a partir daí pode-se inferir o conceito de dados geográficos, que são dados, sequências de símbolos, no âmbito da geografia, que além de apresentar atributos descritivos, apresentam também geometrias geográficas relacionadas. A relação entre os atributos descritivos e as geometrias se

dá por meio de construções de abstrações que descrevem o mundo real, ou seja, através da representação.

Os SIGs são sistemas que realizam o tratamento computacional de dados geográficos. Uma das principais diferenças de um SIG para um sistema de informação convencional é a sua capacidade de armazenar a geometria dos diferentes tipos de dados geográficos, juntamente com os seus atributos descritivos.

Assim, as principais características de um SIG são [1]:

- Inserir e integrar, em uma base de dados, informações descritivas, estatísticas e espaciais, por exemplo, fazer a inserção e integração de dados provenientes de cadastros urbanos e rurais, *Global Positioning System* (GPS), imagens de satélites e outras fontes de dados;
- Oferecer mecanismos para combinar as várias informações através do geoprocessamento em uma base de dados geográfica. Segundo Rodrigues em [17], geoprocessamento é um conjunto de técnicas de coleta, exibição e tratamento de informações espacializadas.

De acordo com a Figura 2.3, pode-se visualizar os componentes de um SIG, que estão divididos em 3 níveis [1]:

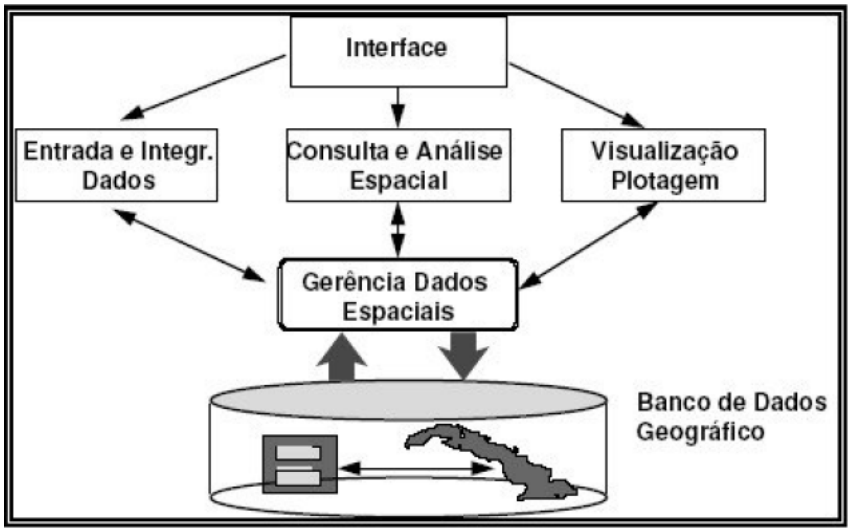


Figura 2.3: Arquitetura do Sistema de Informação Geográfico [1].

- No nível mais alto, tem-se a interface que define como o sistema é controlado e operado. Essa interface pode ser uma aplicação *desktop*, ambiente de navegação da *internet* ou até mesmo em um ambiente móvel;
- No nível intermediário, tem-se a parte do SIG responsável pelo geoprocessamento (processamento de dados espaciais). Esse nível é composto por 3 módulos: a entrada e a integração de dados, que é responsável pelo mecanismo de conversão de dados; a consulta e a análise espacial, que trata as operações topológicas, como álgebra de mapas, estatística espacial, modelagem numérica de terreno e processamento de imagens, e a visualização e a plotagem, que é responsável por oferecer suporte adequado a interpretação do usuário nos aspectos relevantes dos dados pesquisados;

- No nível mais baixo, tem-se um banco de dados geográfico que armazena todos os dados geográficos necessários para o funcionamento do SIG. O conceito de banco de dados geográfico é abordado na próxima seção.

2.2.1 Banco de Dados Geográfico

Um banco de dados é uma coleção lógica e coerente de dados que possui um significado implícito, cuja interpretação é dada por uma determinada aplicação. Um banco de dados convencional é projetado, construído e povoado por dados tradicionais para atender as aplicações convencionais [18]. Já um banco de dados geográfico armazena dados convencionais e dados geográficos a fim de atender aplicações não convencionais, como os SIGs [1].

Ao decorrer dos anos, os SIGs seguiram diferentes formas de implementação, principalmente em relação à estratégia utilizada para armazenar e recuperar dados espaciais. Mais recentemente, os SIGs passaram a utilizar cada vez mais os recursos dos Sistemas Gerenciadores de Banco de Dados (SGBDs), principalmente com o surgimento de extensões espaciais, o que facilitou o desenvolvimento do banco de dados geográfico dos SIGs.

Um SGBD é um sistema de *software* que possibilita aos usuários criar e manter um banco de dados, facilitando os processos de definição, construção, manipulação e compartilhamento de banco de dados entre vários usuários e aplicações. A construção de um banco de dados é o armazenamento dos dados em alguma mídia adequada controlada pelo SGBD. A manipulação ocorre a partir de um conjunto de funções, entre essas funções existe as pesquisas em banco de dados que recuperam dados especificados pelo usuário e a atualização do banco que reflete as mudanças dos requisitos do banco. O compartilhamento consiste no acesso concorrente ao banco de dados por múltiplos usuários e programas [18].

Os SGBDs se diferem em relação à tecnologia utilizada. Os SGBDs mais utilizados no desenvolvimento de sistemas de informação são os SGBDs Relacionais (SGBD-R) e os SGBDs Objeto-Relacionais (SGBD-OR). Nos SGBDs-R, que seguem o modelo relacional de dados, o banco de dados é organizado como uma coleção de relações, cada qual com atributos de um tipo específico (inteiros, ponto flutuante, datas, cadeias de caracteres, campos binários longos), já os SGBDs-OR estendem o modelo relacional, com um sistema de tipos de dados mais rico e estendível (oferecendo operadores que podem ser utilizados na linguagem de consulta), com possibilidade de extensão dos mecanismos de indexação sobre os novos tipos entre outras extensões [18].

Os SGBD-R foram criados para manipular grandes volumes de dados convencionais, atendendo as necessidades das aplicações, logo esses sistemas não oferecem recursos para atender as necessidades de aplicações não convencionais. Uma aplicação é classificada como não convencional neste trabalho quando manipula outros tipos de dados, além dos tradicionais, como por exemplo, os SIGs que trabalham com dados geográficos). A utilização de um SGBD-R para a manipulação de tipos de dados não convencionais pode causar problemas, como queda de desempenho, dificuldade de codificação e manutenção da aplicação. SGBD-OR, pelo fato de ser estendível, reduz os problemas ocorridos com a manipulação de dados não convencionais, mas ainda não atende totalmente as aplicações não convencionais. Para atender as aplicações não convencionais como os SIGs, surgiram

SGBD-OR com extensões geográficas ou SGBD Geográfico (SGBD-G), como por exemplo, o *PostGIS* e o *Oracle Spatial* [1].

As estruturas de dados utilizadas em banco de dados geográficos podem ser divididas em dois tipos: estruturas vetoriais e estruturas matriciais. Como nesse trabalho foi utilizado apenas as estruturas vetoriais, não serão abordadas as estruturas matriciais.

As estruturas vetoriais são utilizadas para representar as coordenadas das fronteiras de cada entidade geográfica, utilizando-se de três formas básicas: pontos, linhas e áreas (ou polígonos) [1]. Essas 3 formas básicas são definidas por suas coordenadas cartesianas, como mostra a Figura 2.4.

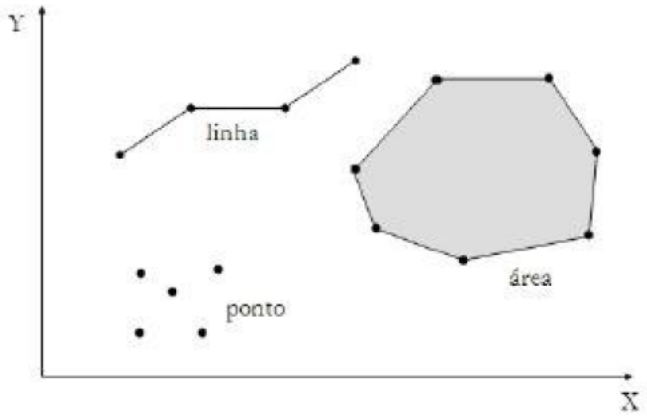


Figura 2.4: Representações vetoriais em duas dimensões [1].

O ponto, um par ordenado (x,y), pode ser utilizado para representar ocorrências ou localizações no espaço, como por exemplo, localização de crimes, interseção entre ruas e ocorrências de doenças, como é o caso do Obsevatório da Dengue, que utiliza dados do *Twitter* para identificar potênciais focos de dengue no Brasil [19]. A linha, conjunto de pontos conectados, é utilizada para guardar formas unidimensionais, como por exemplo, um grafo que indica a interconexão entre vários pontos, representando a distância entre lugares diferentes. A área ou polígono, região do plano limitada por uma ou mais linhas conectadas de maneira que o ponto inicial de uma seja o ponto final da outra, podem ser usados para representar unidades de dados geográficos espaciais individuais, como por exemplo, distritos, estados e regiões. Percebe-se que essa divisão que se obtêm através das fronteiras do polígono é dividida em duas regiões: a interior e a exterior [1].

Para modelar conceitualmente um banco de dados geográficos, o Brasil adotou através de sua Infraestrutura Nacional de Dados Espaciais (INDE) [20] a utilização do modelo de dados *Object Modeling Technique for Geographic Applications* (OMT-G). O OMT-G foi elaborado a partir das primitivas definidas do diagrama de classes da *Unified Modeling Language* (UML), adicionando-se primitivas geográficas com o objetivo de aumentar a capacidade de representação semântica do espaço. O modelo OMT-G vem com primitivas para modelar a geometria e a topologia dos dados geográficos, oferecendo suporte a estruturas topológicas, estrutura de rede, múltiplas representações de objetos e relacionamentos espaciais, oferecendo também a especificação de atributos alfanuméricos e métodos ligados a cada classe [21].

Segundo a INDE, o qual é o conjunto integrado de tecnologias, políticas, mecanismos e procedimentos de coordenação e monitoramento, padrões e acordos, necessário para facili-

tar e ordenar a geração, o armazenamento, o acesso, o compartilhamento, a disseminação e o uso dos dados geoespaciais de origem federal, estadual, distrital e municipal, o modelo conceitual a ser usado para a modelagem de sistemas que envolvam dados espaciais é o OMT-G [20].

2.3 Computação Móvel

A computação móvel permite que usuários se comuniquem com redes fixas ou móveis, independente de onde estejam localizados, através de um meio sem fio. Para tanto, é necessária a utilização de algum dispositivo móvel, tal como telefones celulares, *Personal Digital Assistants* (PDAs), *laptops*, *smartphones*, *tablets* entre outros, além de toda a infra-estrutura de comunicação sem fio. Isso é possível hoje através da tecnologia desenvolvida nas áreas de comunicação celular, redes locais sem fio e serviços via satélite [22].

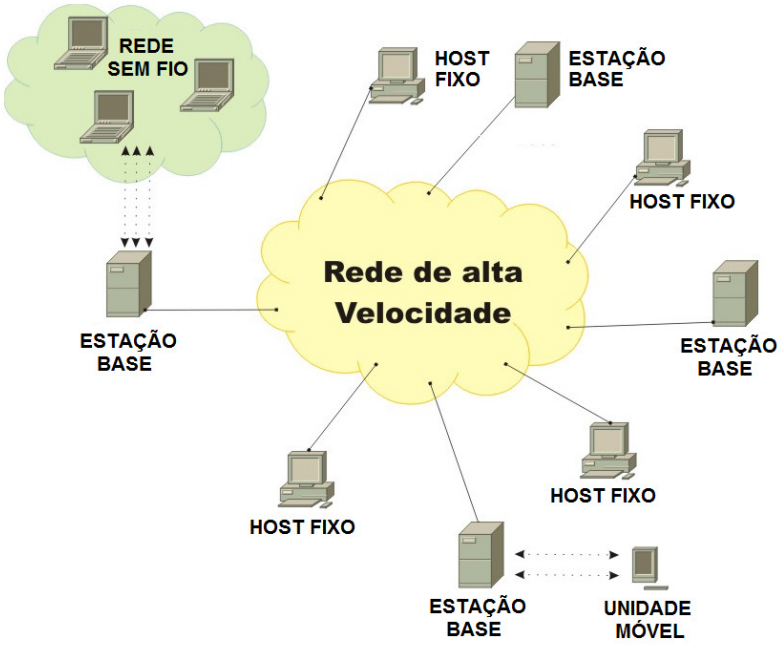


Figura 2.5: Topologia de rede baseada na computação móvel [2].

A topologia de um ambiente de computação móvel é apresentada na Figura 2.5. Como pode ser observada, essa topologia é totalmente distribuída, sendo formada por componentes estáticos e móveis. Utiliza a comunicação sem fio (representada através de linhas tracejadas) e a comunicação cabeada (representada através de linhas contínuas). Esse modelo foi adaptado de [2], [23], [24], [25] e seus componentes são:

- *Hosts* Fixos: componentes estáticos da rede de computadores;
- Estações Base: responsáveis pela comunicação entre a unidade móvel e os *hosts* fixos. São equipadas com interfaces sem fio para receber e transmitir sinais em uma determinada área geográfica (célula) e em unidades móveis;

- Unidades Móveis: dispositivos que se movem no domínio geográfico, tais como telefone celular, *notebooks* ou PDAs. Essas unidades acessam serviços de rede, independente de sua localização ou movimento.

2.3.1 Características da Computação Móvel

Em um ambiente de rede sem fio é mais difícil estabelecer uma comunicação, pois existem muitos obstáculos que podem interferir no sinal, introduzindo assim ruídos e ecos. As conexões sem fio, de maneira geral, possuem qualidade mais baixa que as conexões fixas, pois possuem baixa largura de banda, taxas de erros elevadas e desconexões mais freqüentes. Esses fatores podem aumentar a latência nos retransmissores, atraso no intervalo de retransmissão, erro no processamento do protocolo de controle e curtas desconexões [26].

As principais características da computação móvel são [27]: mobilidade, *handoff*, adaptação e desconexão. Estas características são descritas a seguir.

Mobilidade

Um dos principais objetivos da computação móvel é a mobilidade dos dispositivos móveis. Entende-se por mobilidade a capacidade que um dispositivo possui de se mover pela rede. Devido às diferenças estruturais de um sistema móvel, assim como as variações de tráfego, o ambiente computacional de operação do usuário passa a ser altamente dinâmico [28] [27].

A mobilidade introduz problemas não existentes em ambientes fixos. Problemas relativamente bem resolvidos na computação tradicional permanecem praticamente em aberto em ambientes móveis. Os principais problemas que a mobilidade apresenta vão desde a velocidade do canal, passando por interferências do ambiente e localização da unidade móvel, até a duração da bateria dessa unidade [27].

Handoff

Quando os computadores se movimentam dentro de uma célula de cobertura ou entre células vizinhas durante a execução de aplicações, tem-se um movimento que gera um processo chamado *handoff* [27].

Handoff é um movimento entre duas estações-base adjacentes que garante uma transação enquanto usuários se deslocam de uma célula para outra, mantendo a conectividade de uma unidade móvel com a rede fixa. Isto implica que uma estação-base pode detectar qualquer entrada e saída de usuários dentro e fora da sua célula. Durante o processo de *handoff*, as estações-base são responsáveis pela comunicação sem fio das unidades móveis; este processo é realizado de forma transparente ao usuário [27].

Adaptação

Todos os contextos de um usuário móvel mudam constantemente devido a sua locomoção no ambiente, para que as unidades móveis funcionem com sucesso é necessário que estas unidades sejam adaptativas. À capacidade de ajuste e à possibilidade de produzir resultados apesar de condições adversas dá-se o nome de adaptação. Os dispositivos devem possuir a capacidade de executar um número crescente de tipos de aplicações e

se adaptar às situações onde há uma demanda por recursos computacionais nem sempre disponíveis e outras onde os recursos são abundantes e aparentemente dispensáveis.

As estratégias de adaptação de um ambiente móvel variam entre dois pontos extremos [29]. Num extremo, a adaptação é de inteira responsabilidade das aplicações individuais. 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 contextos e impor limites no uso destes recursos. As aplicações também se tornam mais difíceis de codificar [29].

No outro extremo, a responsabilidade da adaptação é inteiramente do sistema. O sistema fornece total gerenciamento e controle [29]. Sistema nesse contexto é a infraestrutura computacional (servidores, *links* de comunicação e etc.) aos quais as aplicações individuais funcionam.

Entre estes dois extremos, estão as possibilidades que são coletivamente referidas como adaptações cientes da aplicação. 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 [29].

Desconexão

Os sistemas computadorizados dependem da rede a qual estão ligados. A falha na rede é de um interesse maior na computação móvel já que a comunicação sem fio é muito susceptível a desconexão. As operações de desconexão envolvem três estados distintos e esses estados podem se alternar a cada ocorrência de uma desconexão, como visto na Figura 2.6 [3]:

- *Hoarding*: carregamento de itens na unidade móvel. Os dados podem ser replicados ou movidos a um banco de dados local, podendo ficar inacessíveis a outros membros da rede;
- Operações Locais: período em que o dispositivo está desconectado da rede fixa podendo realizar operações apenas dos dados que possuem no banco de dados local, isso ocorre após o *hoarding*. Caso os dados necessários não estejam disponíveis, é criada uma fila para execução de operações futuras que ocorrerão na próxima conexão à rede;
- Reintegração: durante a reconexão os dados são sincronizados na unidade fixa conforme as informações provenientes dos dispositivos conectados. Uma situação de sincronização concorrente entra em funcionamento e deve ser tratada por uma semântica correta para que não ocorra inconsistência nos dados sincronizados.

Na comunicação sem fio as desconexões podem ser caracterizadas de formas diferentes. As desconexões podem ser voluntárias ou forçadas. Nas desconexões voluntárias o usuário ou dispositivo é quem desliga a conexão com o propósito de diminuir o custo da tarifa, o consumo de energia ou aumentar a largura de banda. Na desconexão forçada o usuário não pretendia desligar a conexão, mas ela acontece espontaneamente por motivo de cobertura na área, ou falta de sinal no local em que se encontra, ou falta de bateria no dispositivo móvel, dentre outros [22].

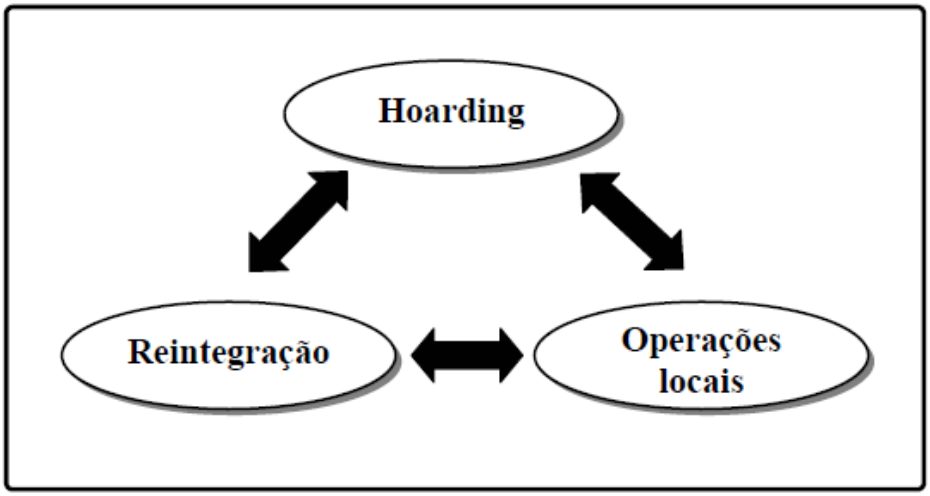


Figura 2.6: Estados de uma operação de desconexão [3].

2.3.2 Restrições da Computação Móvel

Levando em consideração as características da computação móvel descritas na seção anterior, Al-Bar [30] definiu as três principais restrições que afetam o desempenho da computação móvel, descritas a seguir:

- Restrições dos dispositivos móveis: devido à limitação de tamanho, dispositivos móveis continuarão tendo algumas limitações, como exemplo, velocidade do processador, capacidade de memória, tamanho da tela e sua resolução. A diversidade arquitetural proporciona grande variabilidade das configurações de *hardware* das unidades móveis;
- Restrições de comunicação: limitações como largura de banda, desconexão, alta latência e área de cobertura;
- Restrições de mobilidade: o usuário de dispositivos móveis está em constante movimento e, portanto, realiza diversas mudanças de localização. Devido a essas movimentações, podem existir desconexões do dispositivo móvel, havendo falhas de comunicação com a rede. A conexão móvel é altamente variável em desempenho e confiabilidade. Além disso, existe a perda temporária de comunicação quando ocorre deslocamento entre áreas mantidas por diferentes estações base e renegociação de características de acesso.

2.3.3 Modelos de Comunicação Móvel

As redes sem fio, em relação às redes cabeadas, são mais lentas e menos confiáveis, pois possuem baixa largura de banda, taxas de erros elevadas e desconexões mais freqüentes, o que demanda funcionalidade adicional aos *hosts* móveis para diminuir sua dependência dos servidores remotos. Embora ainda não haja consenso a respeito do papel das unidades móveis na computação móvel distribuída, estas considerações dão origem a modelos que proporcionam o ajuste flexível das funcionalidades das unidades móveis [3].

2.3.3.1 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 2.7. 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 [31].

Na sua forma padrão, o modelo Cliente – Servidor apresenta alguns problemas quando utilizado no ambiente móvel, pois a comunicação é realizada através da rede sem fim que é lenta e pouco 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 [31].

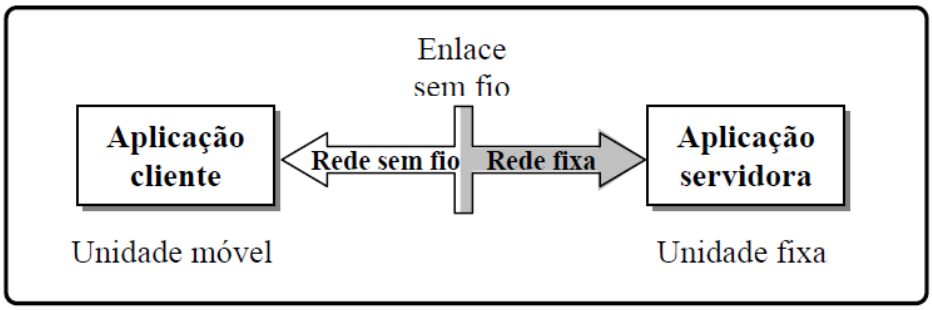


Figura 2.7: Modelo Cliente - Servidor [4].

Duas extensões do modelo cliente – servidor são discutidas a seguir: modelo cliente – agente – servidor e modelo cliente – interceptador – servidor.

2.3.3.2 Modelo Cliente - Agente - Servidor

O modelo Cliente–Agente–Servidor, apresentado na Figura 2.8, constitui uma das abordagens para o desenvolvimento de aplicações móveis que suporta a desconexão. Quando uma desconexão é realizada, tanto o servidor remoto quanto o cliente executam as atualizações a serem feitas [31].

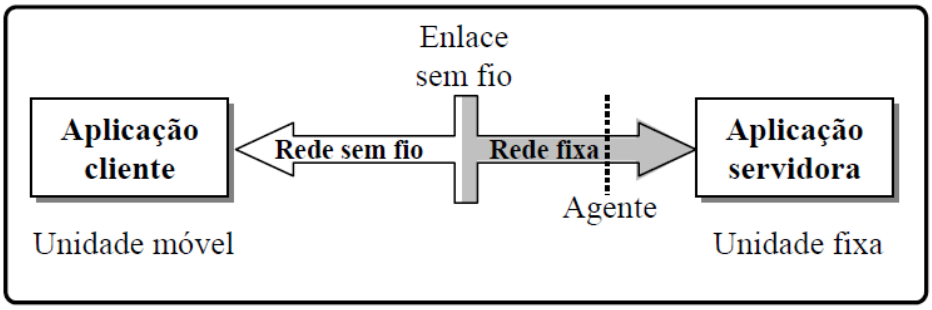


Figura 2.8: Modelo Cliente - Agente - Servidor [4].

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 [31].

2.3.3.3 Modelo Cliente - Interceptador - Servidor

Esse modelo divide o interceptador em duas partes, uma localizada no servidor na rede fixa e o outro na unidade móvel, como é representado na Figura 2.9. Esta interceptação é transparente tanto para o cliente quando para o servidor, oferecendo flexibilidade e poder de gerenciamento de desconexões [3].

Segundo Pitoura [3] o modelo cliente–interceptador–servidor é indicado quando a aplicação exige alto poder de processamento e armazenamento, além de maior autonomia de energia, porque esse modelo melhora a compressão de dados, otimiza a comunicação sem fio e o tratamento de desconexões, devido à existência dos dois interceptores.

Um dos problemas deste modelo é que cada aplicação requer processamento tanto no cliente quanto no servidor.

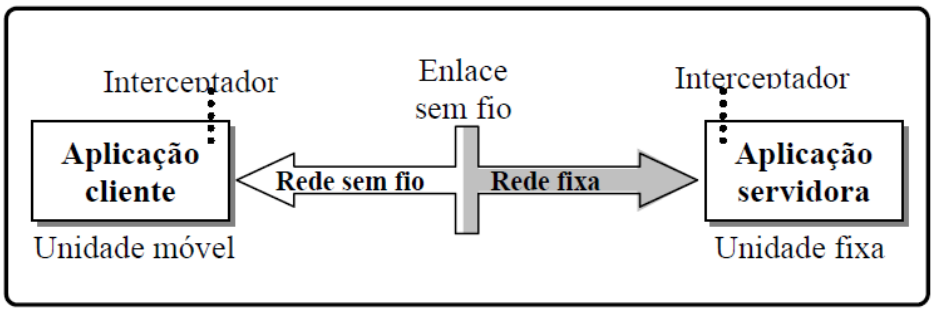


Figura 2.9: Modelo Cliente - Interceptador - Servidor [4].

2.3.3.4 Modelo *Peer - to - Peer* (P2P)

No modelo *Peer-to-Peer* 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 [3].

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 [3].

2.3.3.5 Modelo de Agentes Móveis

Este modelo trata a questão da mobilidade permitindo o trabalho desconectado. Diferentemente do modelo cliente - agente - servidor, os agentes desse modelo possuem a característica de mobilidade.

Agentes Móveis são processos enviados de um computador para realizar uma tarefa específica em outro computador. 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 [3].

Os agentes móveis migram de um computador para outro, associados com um itinerário (direção) e com uma distribuição dinâmica do seu código e do seu estado. Após a sua submissão, o agente móvel processa de forma autônoma e independente de seu emissor. Quando alcança um cliente ou servidor, se integra a um ambiente de execução de agentes. Possui a capacidade de se transportar para outro cliente ou servidor, gerando novos agentes ou interagindo com outros agentes. Quando concluído, entrega seus resultados ao cliente ou ao servidor emissor ou a outro cliente ou servidor. A seguir, são descritas algumas características do modelo de agentes móveis [3]:

- 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.

2.4 Gerenciamento de Banco de Dados Móveis

A computação móvel permite o desenvolvimento de novas aplicações em banco de dados. Para a efetiva execução dessas aplicações tornam-se necessárias mudanças no gerenciamento e nos mecanismos de garantia de consistência dos dados. Essa necessidade vem das restrições impostas pelos ambientes de comunicação sem fio: a limitação da largura de banda dos canais de comunicação sem fio, a mobilidade e as freqüentes desconexões dos dispositivos móveis, a mobilidade dos dados e o grande número de usuários.

Segundo Elmasri e Navathe [18], os bancos de dados móveis podem ser distribuídos em dois possíveis cenários. No primeiro cenário, toda a base de dados está distribuída principalmente entre os componentes ligados por fiação, possivelmente com replicação total ou parcial dos dados. Um servidor de banco de dados 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. No segundo cenário, 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 os servidores de banco de dados.

Desta forma, muitas das questões de gerenciamento de dados distribuídos também se aplicam aos bancos de dados móveis, segundo Elmasri e Navathe [18], desde que se levem as seguintes considerações adicionais e variações em conta:

- Distribuição de dados e replicação: os dados podem estar desigualmente e irregularmente distribuídos entre unidades móveis e servidores. As restrições de consistência aumentam o problema de gerenciamento de dados em *cache*. Esta deve proporcionar à unidade móvel acesso àqueles dados mais frequentemente acessados e atualizados;
- Modelos de transação: questões de tolerância a falhas e corretude das transações são agravadas no ambiente móvel. A transação móvel é executada sequencialmente através de diferentes estações base e possivelmente em múltiplos conjuntos de dados, dependendo da movimentação da unidade móvel. Não existe coordenação central na execução de uma transação, por exemplo, quando se tem a base distribuída pelos componentes com e sem fio. Para isso, as propriedades Atomicidade, Consistência, Isolamento e Durabilidade (ACID) das transações podem precisar de modificação e novos modelos de transação podem 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 redes sem fio diferentes, devem receber completamente e corretamente tais respostas;
- 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. Se uma unidade móvel suspende a conexão voluntariamente, isto não deve ser tratado como falha. Falhas de transação são mais frequentes durante operações de *handoff*. Falhas nas unidades móveis, por sua vez, causam particionamento da rede e afetam os algoritmos de roteamento;
- Serviços baseados na localização: conforme os clientes se movem, a informação dependente da localização no *cache* pode se tornar inconsistente. As técnicas de despejo de dados da *cache* para os dispositivos móveis clientes são importantes neste caso. Além disso, atualizar freqüentemente as consultas dependentes da localização, e então aplicar essas consultas para atualizar o *cache*, apresenta um problema;
- Divisão do trabalho: certas características do ambiente móvel forçam uma mudança na divisão de trabalho no processamento de consultas. Em alguns casos, o cliente precisa funcionar independentemente do servidor;
- Segurança: os dados móveis são mais vulneráveis a perdas e inconsistências , por estarem em um ambiente móvel, do que os que são deixados em uma localização fixa. As técnicas formais para o gerenciamento e autorização de acesso a dados críticos se tornam mais importantes nesse ambiente. Os dados também são mais voláteis, e as técnicas precisam ser capazes de compensar sua perda.

2.4.1 Características dos Bancos de Dados Móveis

Uma vez que esse é um desafio para a definição da arquitetura de coleta de dados proposta nessa dissertação, esta subseção visa conceituar e fazer um estudo mais detalhado de algumas características do banco de dados móveis. As principais características dos bancos de dados móveis são: *caching* e difusão de dados, transações e recuperação de falhas.

Caching e Difusão de Dados

O mecanismo de *caching* é utilizado para otimizar o tempo de resposta nas consultas da computação móvel e dar suporte quando a unidade móvel estiver desconectada. Para que o mecanismo de *caching* atinja um bom resultado é necessário que estejam definidas as estruturas de dados e as estratégias de gerenciamento do *caching* consideradas as características de cada aplicação e o padrão de acesso aos dados [32].

Quando os dados consultados 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 [27].

Segundo Silberchatz [33], 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.

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 [18].

A concorrência de transações que acessam a 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 [18].

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 ocontecem concorrentemente, por isso técnicas como serialização e agendamento de transações se fazem necessárias para evitar conflitos [33].

Uma transação móvel é uma transação distribuída, onde alguma parte da computação é executada na unidade móvel e outra parte em um *host* fixo. O uso do meio sem fio e

a mobilidade resultante dos consumidores e produtores de dados afetam o processamento das transações [34].

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 [31]. 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.

Os sistemas distribuídos utilizam o método de recuperação a falhas tradicional baseado em pontos de recuperação, conhecidos como *checkpoints*. No caso de falhas e desconexões, a aplicação usa o último *checkpoint* salvo para reiniciar sua execução [3].

2.5 Tecnologias de Sincronização

A sincronização em ambientes móveis pode ser definida como o ato de estabelecer equivalência entre duas coleções de dados, entre cliente e servidor, após a ocorrência de alteração nos registros armazenados. Depois da sincronização de dados cada elemento de dados em uma coleção é mapeado por um elemento de dados em outra, sendo que seus dados são equivalentes [35]. Para que essa sincronização ocorra é necessário a replicação de dados.

2.5.1 Replicação de Dados

A replicação é o processo no qual transações executadas em um banco de dados são propagadas para um ou mais bancos de dados de forma serial, ou seja, significa que as transações, assim como todas as operações, são replicadas na mesma ordem na qual elas foram solicitadas. Caso isto não aconteça, podem ocorrer inconsistências entre a unidade móvel e o servidor em um ambiente móvel. As operações que serão replicadas das unidades móveis podem ser armazenadas em um banco de dados local até que sejam propagadas para o servidor, no momento da sincronização, pois algumas vezes a conexão se torna impossível [36] [27].

A replicação é necessária para sincronizar os dados e as operações no banco de dados. Essa replicação pode ser parcial ou total. Na replicação parcial apenas uma parte dos dados de um banco de dados são replicados para outros bancos, já na replicação total todos os dados são replicados de um banco para outro [27] [18].

Com a replicação total, se pelo menos uma das cópias estiver disponível, a confiabilidade aumenta, uma vez que o usuário não depende apenas dos dados disponibilizados

em uma única localidade. No caso de acontecer algum problema no sistema, a réplica dos dados é solicitada pelo usuário [27].

2.5.2 Tipos de Sincronização

Segundo Palazzo [37], os tipos de sincronização são classificados em *one-way* e *two-way*. Cada um destes tipos é descrito a seguir.

One-Way

A sincronização *one-way* é um processo de transferência de atualizações de dados que ocorre em sentido único. Este tipo de sincronização pode ser aplicada partindo-se tanto do cliente móvel para o servidor, quanto do servidor para o cliente móvel. O processo ocorre da seguinte forma: um elemento de rede estabelece a conexão entre a unidade móvel e o servidor, submetendo todas as atualizações efetuadas na unidade móvel, de maneira sincronizada e pré-definida, para o servidor ou vice-versa.

Quando se parte do cliente móvel, todas as alterações são submetidas ao servidor fixo não retornando nenhuma confirmação ou ação por parte do servidor fixo, cabendo ao servidor gerenciar e resolver conflitos. Partindo-se do servidor, a sincronização disponibiliza na unidade móvel os dados atualizados no banco consolidado.

Two-Way

A transmissão das modificações ocorre nos dois sentidos, tanto do cliente como do servidor. A iniciativa neste caso é por parte do cliente móvel que submete os novos dados ou os atualizados para o servidor. O servidor identifica e trata os conflitos existentes retornando ao cliente os dados corretos. No final, o cliente e o servidor possuem os mesmos dados.

2.5.3 Protocolos de Sincronização

Um protocolo de sincronização define o fluxo de trabalho para a comunicação durante uma seção de sincronização de dados quando o dispositivo móvel é conectado à rede fixa. Ele deve dar suporte a identificação de registros, comandos de protocolo comuns para sincronização de dados local e de rede, e pode apoiar a identificação e resolução de conflitos de sincronização [38].

Existem diversos produtos de sincronização de dados que utilizam protocolos de sincronização que funcionam apenas para transportes específicos implementados para alguns dispositivos. A proliferação de tecnologias de sincronização não-interoperáveis dificulta as tarefas de usuários, fabricantes de dispositivos, provedores de serviços e desenvolvedores de aplicações. A falta de um protocolo de sincronização de dados comum pode impedir o crescimento do uso de dispositivos móveis, restringindo a habilidade de usuários para ter acesso a dados e limitando a entrega de operações realizadas sobre dados móveis [38].

Aplicações que reconciliam e solucionam mudanças em dados entre duas fontes distintas são amplamente utilizadas por empresas de desenvolvimento de *softwares* para ambientes móveis. Porém, a maioria dessas aplicações consistem em soluções proprietárias múltiplas, que não satisfazem todo o espectro de funcionalidades e o suporte requerido

aos dispositivos pela maioria das organizações. Muitas organizações não estão dispostas a desenvolver múltiplos canais e a implementar várias soluções de sincronização para permitir o acesso de múltiplos dispositivos a um único banco de dados devido à complexidade e o custo de manutenção requerido [38]. Por isso, deseja-se protocolos de sincronização que suportam diversos tipos de transportes implementados para vários tipos de dispositivos de diferentes marcas.

Um protocolo necessita ser interoperável, sincronizar dados da rede com qualquer dispositivo móvel e sincronizar dados de um dispositivo móvel com qualquer rede. Isto inclui um espaço de mercado onde os dispositivos móveis têm recursos reduzidos de processamento e de memória, e são empregados servidores de rede capazes de executar uma lógica de sincronização sofisticada.

2.5.3.1 Protocolo de Sincronização Interoperável

Devido às diferenças estruturais em um sistema móvel e a imprevisibilidade de movimentação dos usuários ao longo das diferentes redes sem fio, o ambiente computacional passa a ser altamente dinâmico. Nesse contexto, a transmissão adequada de certas informações, como gráficos e figuras, exige uma capacidade que pode ser inviável para o sistema gerir, pois um ambiente computacional dinâmico requer um projeto de aplicações com capacidade de interoperabilidade ao longo de diferentes ambientes de acesso sem fio. Pontos chaves no projeto de tais aplicações são: capacidade de identificação das condições do ambiente, adaptabilidade do modo de apresentação das informações e continuidade da prestação do serviço ao longo das mudanças entre diferentes redes sem fio.

Um protocolo de sincronização comum tem que trabalhar sobre as redes usadas por dispositivos móveis. As várias redes *wireless* comumente usadas por dispositivos móveis exigem muito de um protocolo, pois compartilham características comuns como:

- Alta latência de rede: é a demora introduzida quando um pacote é momentaneamente armazenado, analisado e então remetido de um ponto a outro da rede [39]. Redes *wireless*, com alta latência requerem um protocolo de sincronização robusto.
- Largura de banda limitada: para minimizar requerimentos de largura de banda e as demandas de processo associadas, o protocolo deve permitir alternativas como técnicas de codificação binárias para os dados e os comandos de sincronização. O *Wap Binary eXtensible Markup Language* (WBXML), padrão adotado pelo *Wireless Application Protocol* 6 (WAP6) e submetido ao *Word Wide Web Consortium* (W3C) faz uso da técnica de codificação binária para ambientes de largura de banda limitada [39].
- Custo de pacotes relativamente alto: o protocolo tem que minimizar o número de iterações de *request-response* entre o dispositivo e os dados da rede. Um ótimo protocolo geraria um único par de *request-response* de mensagens. Nesse modelo, um pedido de um dispositivo móvel poderia conter todas as atualizações para seus dados locais. A mensagem de resposta proveria as atualizações, com conflitos já identificados e possivelmente resolvidos.
- Baixa confiabilidade de dados e conectividade: o protocolo tem que suportar desconexões inesperadas durante o procedimento de sincronização. Mesmo na ocorrência

de uma desconexão, deve assegurar que os dados do dispositivo e do repositório da rede permaneçam consistentes, e continuar a operação quando a conexão for restabelecida.

Com a utilização de um protocolo interoperável, um usuário pode ter acesso e manipular o mesmo conjunto de dados de dispositivos diferentes e dispositivos móveis podem suportar mais tipos de dados, como por exemplo, dados geográficos. Para que isto seja possível, o protocolo precisa das seguintes características:

- Operar efetivamente sobre redes sem fio;
- Suportar diferentes protocolos de transporte;
- Tolerar dados arbitrários de rede;
- Possibilitar o acesso a dados por diversas aplicações;
- Suportar as limitações de recurso do dispositivo móvel;
- Assistir implementações para *internet* e tecnologias de rede existentes.

2.5.3.2 O Protocolo *SyncML*

O *SyncML* é um protocolo padrão de sincronização de dados, independente de plataforma, dispositivo e rede. Ele é baseado em tecnologia *eXtended Markup Language* (XML) e mantido pela *Open Mobile Alliance* (OMA), uma aliança entre várias empresas para a criação de um protocolo comum para a sincronização de dados [40] [41].

Pode ser considerada como uma linguagem padrão para sincronizar diferentes dispositivos e aplicações sobre qualquer rede. Com o *SyncML* qualquer tipo de informação pode ser sincronizada para manter a informação consistente e atualizada.

O protocolo *SyncML* é baseado no modelo de comunicação móvel cliente - servidor e é dividido em: protocolo de representação e protocolo de sincronização. O protocolo de representação tem como foco a organização dos conteúdos dos dados da sincronização. Define o formato dos dados, faz a restauração (*refresh*), a deleção (*delete*), a substituição (*replace*), o arquivamento (*archive*) e etc. O protocolo de sincronização tem como foco a administração das operações de sincronização. Define fluxo de mensagens entre um *SyncML* cliente e um servidor durante a sessão de sincronização de dados [40] [41].

As vantagens do protocolo *SyncML* são [40] [41] :

- Por ser baseado no XML, fornece um formato padrão para a descrição de dados estruturados o que gera resultados mais significativos para consultas em diferentes plataformas;
- Possui autenticação entre um cliente *SyncML* e um servidor *SyncML*;
- Suporta uma variedade de protocolos de transporte;
- Suporta uma variedade de base de dados;
- Funciona bem em redes com pouca banda porque os dados que serão sincronizados passam por uma técnica de codificação chamada WBXML gerando um formato

binário do XML que possui um tamanho bem menor do que o original. Além disso, o protocolo possui um baixo número de interações pedido-resposta entre o cliente *SyncML* e o servidor *SyncML* para realizar a sincronização;

- É construído sob uma tecnologia de *internet* e *web* existentes;
- É um protocolo de sincronização livre.

Há sete tipos diferentes de sincronização definidas pelo *SyncML*, dependendo de como é inicializada pelo cliente ou servidor e a transferência de informação entre eles [40] [41]:

1. *Two-Way Sync*: este tipo de sincronização é o mais comum e largamente usado onde o servidor e o cliente trocam somente os dados modificados no final. Cliente é sempre esperado para começar a sincronização mandando as modificações primeiro.
2. *Slow Sync*: este tipo de sincronização pode ser considerada como uma forma de sincronização *two-way* com uma exceção de que o cliente envia o banco de dados inteiro ao invés de somente as modificações, pedindo ao servidor para preparar a análise da sincronização para os dados entre cliente e o servidor. O servidor retorna sempre as modificações requeridas pelo cliente.
3. *One-Way Sync from Client only*: somente durante este tipo de sincronização, o cliente envia todas as modificações para o servidor, mas não espera nenhuma modificação do servidor.
4. *Refresh Sync from Client only*: este é um tipo especial de *One-Way Sync from Client*, onde o cliente exporta todos os dados do banco de dados para o servidor. É esperado do servidor repor os dados existentes com o banco de dados recebido do cliente.
5. *One-Way Sync from Server only*: este tipo de sincronização envia todos os dados modificados do servidor para o cliente e não espera qualquer modificação do cliente.
6. *Refresh Sync from Server only*: este é um tipo especial de sincronização *One-Way Sync from Server*, onde o servidor exporta todos os dados do banco de dados para o cliente. É esperado do cliente repor os dados existentes com o banco de dados recebido do servidor.
7. *Server Alert Sync*: o servidor alerta o cliente para preparar uma sincronização. Ele também diz ao cliente qual dos tipos de sincronização acima será usada entre eles.

2.6 Arquiteturas Relacionadas

Esta seção apresenta algumas arquiteturas de coleta de dados para pesquisas de campo existentes. Ela é dividida em duas subseções: arquiteturas de mercado, onde mostra algumas arquiteturas comerciais existentes, e arquiteturas do meio acadêmico, onde são apresentados algumas arquiteturas de coleta desenvolvidas em pesquisas acadêmicas.

2.6.1 Arquiteturas de Mercado

Existe disponível no mercado um conjunto de *software* que pode ser utilizado para coleta de dados em pesquisas de campo inseridas em um ambiente de computação móvel. O *Nokia Data Gathering* [42] e a solução da *AuditMagic* [43] são exemplos destes *softwares*.

O *Nokia Data Gathering* (NDG) é um software *open source*, desenvolvido pelo Instituto Nokia de Tecnologia que permite a coleta de dados em dispositivos móveis e a transmissão de resultados para análises em tempo real, de acordo com o acesso a uma rede de comunicação de dados. O sistema permite a criação e entrega de questionários para telefones móveis e a integração em uma base de dados pré-existente usando uma rede de celulares comum [42]. Ele funciona apenas em *smartphones* da marca *nokia*.

O NDG permite às organizações criar questionários detalhados para distribuição em diversos aparelhos móveis por meio da rede celular. Agentes podem preencher o questionário no visor do aparelho e mandar as informações em seguida para um servidor central. O sistema também permite o uso do *geotag* (marcação geográfica) com informações de GPS, que permitem identificar em um mapa os pontos exatos onde as coletas foram realizadas [42].

A tecnologia utilizada pelo NDG garante a transmissão de dados em tempo real, por meio da conexão *General Packet Radio Service* (GPRS) das redes *Global System for Mobile Communications* (GSM). Em locais onde não há rede, os dados ainda podem ser armazenados em um cartão de memória e enviados quando o sinal se restabelecer. Ainda há a possibilidade de envio de dados para um computador via *Bluetooth*(sem fio), cabo USB ou pelo próprio cartão de memória [42].

A *AuditMatic* [43] tem um conjunto de soluções para o desenvolvimento de instrumentos de coleta de dados em campo associados às ferramentas de análise. Essas soluções funcionam da mesma forma do que o NDG, a diferença é que não possuem a restrição de marca em relação aos *smartphones* utilizados na coleta, suportam o uso de *tablets* na coleta de dados e não suportam o envio de dados para um computador via *Bluetooth*. Assim como o NDG, essas soluções sincronizam os dados atráves do *Open Database Connectivity* (ODBC) que é um padrão para acesso a sistemas gerenciadores de bancos de dados.

2.6.2 Arquiteturas do Meio Acadêmico

As arquiteturas de coleta de dados em pesquisas de campo propostas pelo meio acâdemico tem o objetivo de atender o contexto de coleta de dados ao qual a pesquisa está inserida. O objetivo é propor uma arquitetura que atenda as necessidades da coleta de dados da pesquisa em questão.

Ji [5] apresenta uma arquitetura que tem como objetivo a coleta de dados de mineração de forma georreferenciada. A figura 2.10 mostra a arquitetura implementada.

A arquitetura é dividida em três partes: sub-sistema de gerenciamento de dados do lado do servidor, sub-sistema de coleta de dados do lado do cliente e *links* de comunicação. No lado do servidor foi usado o *ArcSDE* (*ArcSDE* é um *gateway* da família *ArcGIS* para o mundo dos SGBDs, ele possibilita o armazenamento e o gerenciamento de informações geográficas diretamente no SGBD de sua escolha, e serve dados georeferenciados para os *ArcGIS desktops*) e o SGBD *Oracle* para gerenciar os dados espaciais [5].

No lado do cliente, a coleta é realizada através de um dispositivo móvel e foi adotado o SGBD *SQL Server Compact Edition* para a gestão do dados que são coletados em campo.

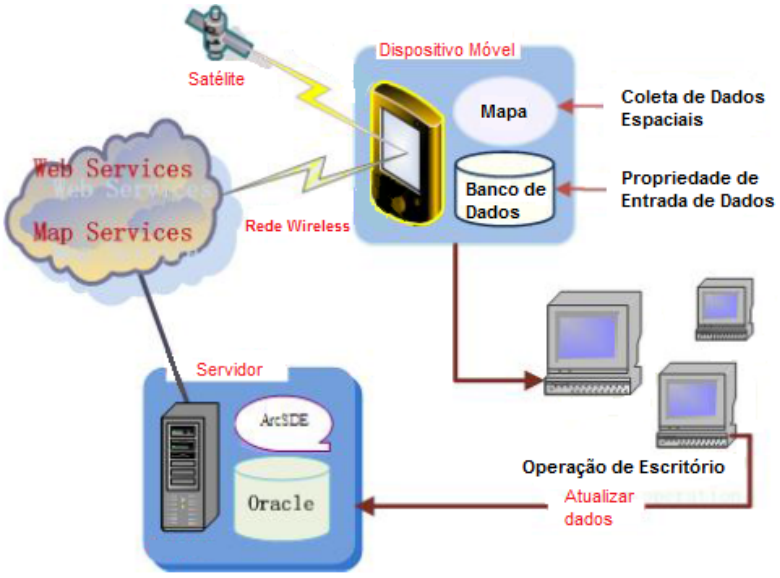


Figura 2.10: Arquitetura para coleta de dados de mineração de forma georreferenciada. Adaptado de [5].

A comunicação entre cliente e servidor é feita através de *Web Services* utilizando-se da tecnologia de redes sem fio [5].

Para a coleta de dados espaciais, o cliente solicita os mapas para o servidor (via *Web service*, mais especificamente *Wep Map Services*), coleta os dados de forma georreferenciada e atualiza as camadas dos mapas no servidor [5].

Harpinder [44] apresenta o *Open Data Kit* (ODK) um conjunto de ferramentas de código aberto que tem o objetivo de coletar e enviar dados, em pesquisas de campo, para um servidor de banco de dados central em tempo real usando um dispositivo *android*. A comunicação entre cliente e servidor também é feita através de *Web Services* utilizando a tecnologia de redes sem fio.

O ODK é dividido nos seguintes módulos: ODK *Build*, módulo responsável pela construção dos formulários que serão utilizados na coleta; ODK *Collect*, aplicativo no dispositivo móvel utilizado para coletar os dados em campo, os formulários criados no ODK *Build* são carregados no aplicativo; ODK *Aggregate*, responsável por armazenar os dados coletados no servidor de banco de dados central, visualizar os dados em mapas e gráficos e exportar os dados em vários formatos, e ODK *Briefcase*, uma ferramenta responsável pela sincronização dos dados coletados para o servidor de banco de dados central utilizando *Web Services*.

Magalhães et al [6] apresentaram uma arquitetura para sistema de coleta de dados em um ambiente de computação móvel onde o acesso à *internet* é intermitente que foi validada na pesquisa do Transporte Escolar Rural cujo objetivo é avaliar a realidade do transporte escolar rural no Brasil. A arquitetura é composta de 3 fases como pode ser visto no Figura 2.11: fase de coleta, fase intermediária de replicação de dados e fase de integração final para o banco de dados central.

A fase de coleta consiste na coleta de dados pelos pesquisadores em campo através dos seus dispositivos móveis. Os módulos que compõem essa etapa são: interface, onde existe a entrada de dados no sistema através de um conjunto de formulários eletrônicos;

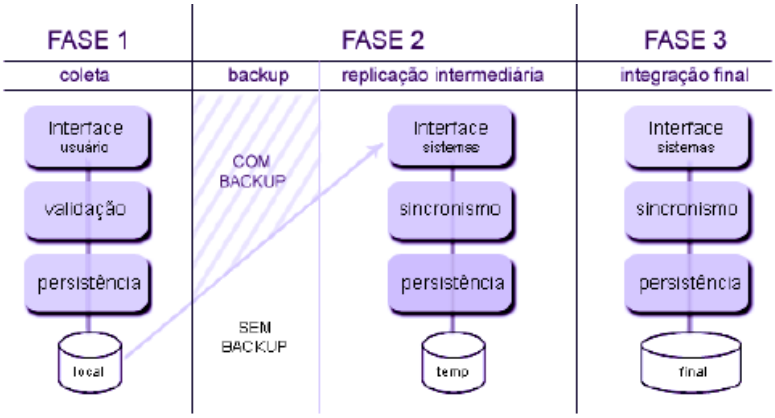


Figura 2.11: Arquitetura para coleta de dados em computação móvel com acesso intermitente à *internet* [6].

validação, onde cada formulário possui um meio de validação para que dados relevantes não sejam esquecidos e nem registrados incorretamente, e persistência, responsável pelo armazenamento das informações no banco de dados dos dispositivos móveis [6].

A fase intermediária é composta de duas partes principais: o sistema de *backup* e a replicação dos dados para o servidor intermediário. O sistema de *backup* fez-se necessário, pois a arquitetura foi desenvolvida para lugares onde o acesso à *internet* é intermitente, objetivando criar um mecanismo onde os dados armazenados no sistema possam ser copiados para um dispositivo externo, como um *Pen Drive*, um HD externo ou até mesmo um CD ou DVD [6].

A replicação dos dados para o servidor intermediário visa garantir a segurança dessas informações por meio da redundância de dados, evitando que fiquem apenas no dispositivo do pesquisador. Para o desenvolvimento do sistema de replicação foi utilizada uma arquitetura de serviços para facilitar a migração de diferentes esquemas de banco de dados. Foi implementado em Java e utiliza o *framework JPA/Hibernate* para se conectar e replicar os dados. A fase final contém a etapa de integração de todos os dados da pesquisa em um banco de dados central [6].

A arquitetura de coleta e disseminação de dados climáticos no estado do Piauí proposta pelos pesquisadores da Empresa Brasileira de Pesquisa Agropecuária e da Faculdade de Educação Tecnológica de Teresina foi modelada a partir da necessidade de se criar um repositório de dados agrometeorológicos que são coletados com o uso de sensores climáticos, instalados em estações automáticas situadas em alguns municípios do Estado do Piauí [7].

A arquitetura propõe uma estrutura de rede que disponha de: o uso de estações climáticas conectadas a um computador através de rede de telefonia, um computador com o sistema operacional *Windows* (terminal *Windows*) para realizar as chamadas às estações e efetuar a coleta dos dados e um servidor de dados para manter o serviço de acesso aos dados para usuários conectados à *internet* [7].

A Figura 2.12 apresenta a forma como os componentes arquiteturais foram relacionados. A linha pontilhada marca a fronteira do sistema *web* e distingue seu contexto dos elementos externos. Tal arquitetura comporta três partes principais: estações agrometeorológicas automáticas instaladas em diferentes regiões do Estado (item 1), *software* dos

fabricantes do equipamento instalados em terminal com o sistema operacional *Windows* (item 2) e sistema *web* para recebimento e envio dos dados para o servidor através da *internet*. Contudo, as informações poderão ser acessadas pelos usuários através de interface disponível em sítio da instituição (itens 3, 4 e 5) [7].

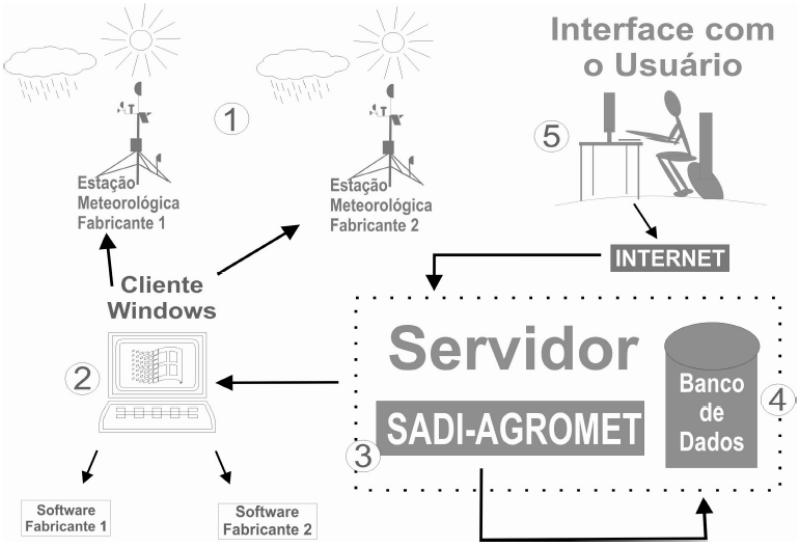


Figura 2.12: Arquitetura de coleta e disseminação de dados climáticos no estado do Piauí [7].

Os itens 3, 4 e 5 representam o trabalho específico do Sistema para Aquisição e Disseminação de Informações AgroMeteorológicas (SADI-AGROMET), que visa buscar os dados coletados pelos *softwares* dos fabricantes das estações, no terminal *Windows*. Esta possui os dados climáticos em formato com legibilidade, organização e manipulação defasadas, onde o resultado da coleta de dados no terminal gera um arquivo texto, separado por vírgula. O SADI-AGROMET lê o arquivo texto, trata seus dados e insere estas informações na base de dados hospedada em um servidor remoto [7].

2.7 Conclusão

De todo o exposto, percebe-se que a prática de coletar dados em trabalhos de campo através de questionários de papel tornou-se obsoleta, pois requeria maior esforço na coleta e no processamento dos dados. O SCDM, no entanto, requer menos esforço e o custo é consideravelmente menor, tendo em vista a coleta e o processamento de dados serem simultâneos e não necessitar manipular dados físicos.

Um sistema de coleta de dados móveis, como foi explicado, é uma combinação de uma aplicação cliente, rodando em dispositivos móveis, uma infraestrutura de redes sem fio e servidores de banco de dados remotamente acessíveis que tem o objetivo de coletar e transmitir dados de localizações geográficas remotas para localizações centrais em repositórios de armazenamento de dados. Devido às características desses sistemas, eles possuem desafios que devem ser tratados na sua implementação. Cada um desses desafios, abordados neste capítulo, devem ser levados em consideração na especificação da arquite-

tura de coleta de dados para pesquisas de campo proposta por esta pesquisa a qual será tratado no próximo capítulo.

Capítulo 3

A Arquitetura Proposta

Neste capítulo a arquitetura para coleta de dados em campo em um ambiente computacional heterogêneo é especificado. O capítulo é dividido em duas seções, a primeira apresenta o mapeamento do processo de coleta de dados em pesquisas de campo a nível de usuário e a segunda especifica a arquitetura de coleta de dados proposta.

3.1 O Mapeamento do Processo de Coleta de Dados em Pesquisas de Campo

Para um melhor entendimento da arquitetura proposta foi feito o mapeamento do processo de coleta de dados em pesquisas de campo a nível de usuário, utilizando uma notação de modelagem de processos de negócio, como mostra a Figura 3.1.

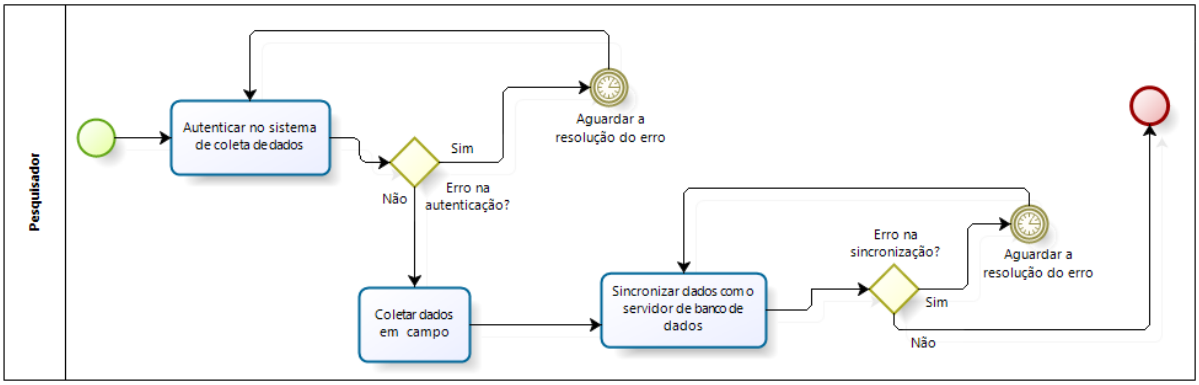


Figura 3.1: Mapeamento do processo de coleta de dados em pesquisas de campo.

O participante desse processo é o pesquisador, que é a pessoa responsável por coletar os dados.

As tarefas desse processo são descritas nas Tabelas 3.1, 3.2 e 3.3.

Inicialmente, o pesquisador faz a sua autenticação no sistema de coleta de dados. Se não ocorrer erros na autenticação, o pesquisador vai coletar os dados em campo. Se ocorrer algum erro na autenticação, o pesquisador aguarda uma solução para o erro. Quando o erro for resolvido, inicia-se novamente a tarefa de coleta de dados.

Tabela 3.1: Tarefa autenticar no sistema de coleta de dados.

Atividade	**Autenticar no sistema de coleta de dados**
Descrição	O pesquisador faz a autenticação no sistema de coleta de dados antes da coleta. Para que ocorra a autenticação é necessário conexão com a internet.
Entrada	Usuário e senha do pesquisador, ou a digital do pesquisador e etc. A entrada dessa tarefa vai depender do método de autenticação utilizado no sistema de coleta.
Saída	Acesso as funcionalidades do sistema de coleta de dados.

Tabela 3.2: Tarefa coletar dados em campo.

Atividade	**Coletar dados em campo**
Descrição	O pesquisador coleta os dados em campo utilizando o sistema de coleta de dados.
Entrada	Pesquisador portando um sistema de coleta de dados sem nenhum dado registrado em seu banco.
Saída	Pesquisador portando um sistema de coleta de dados com dados registrados em seu banco.

Tabela 3.3: Tarefa sincronizar dados com o servidor de banco de dados.

Atividade	**Sincronizar dados com o servidor de banco de dados**
Descrição	Os dados armazenados no banco de dados do dispositivo móvel utilizado na coleta são sincronizados com o servidor de banco de dados central para análise. Para que ocorra a sincronização é necessário conexão com a internet.
Entrada	Pesquisador portando um sistema de coleta de dados com dados registrados em seu banco.
Saída	Servidor de banco de dados central com os dados, que foram coletados com o sistema de coleta, em seu banco.

Após a coleta dos dados em campo, os pesquisadores sincronizam os dados coletados com o servidor de banco de dados da instituição ou das instituições aos quais eles pertencem, e assim se encerra o processo. Porém se ocorrer algum erro, o pesquisador aguarda uma solução para o problema. Quando o problema for resolvido inicia-se novamente a tarefa de sincronização de dados.

3.2 Especificando a Arquitetura Proposta

O objetivo de uma arquitetura de coleta de dados em pesquisas de campo que funciona em ambientes computacionais heterogêneos é desvincular as especificidades de cada coleta com as tecnologias utilizadas, proporcionando uma arquitetura para ser utilizada em vários contextos de coleta de dados. Considera-se nesse trabalho que para que uma arquitetura de coleta de dados funcione em ambientes computacionais diferentes é necessário que essa arquitetura leve em consideração dois aspectos: a interoperabilidade e a flexibilidade.

A interoperabilidade é necessária para que a arquitetura não seja dependente de tecnologia específica, mas que abranja a possibilidade de utilização de várias delas. A flexibilidade é necessária em dois sentidos, o primeiro para atender as diversas dimensões que uma coleta de dados em campo pode ter, ou seja, atender as coletas de dados que envolvem uma ou mais instituições. Uma determinada coleta de dados pode armazenar os dados coletados em apenas um único banco de dados central localizado em uma instituição ou em vários bancos de dados centrais localizados em diferentes instituições. O

segundo é para atender as mudanças necessárias que a arquitetura precisa passar para se adequar aos contextos de coleta de dados diferentes.

De forma abstrata, a arquitetura proposta é composta de três fases: a coleta em campo, a sincronização e o armazenamento, como pode-se ver na Figura 3.2. Cada uma dessas fases é descrito a seguir.

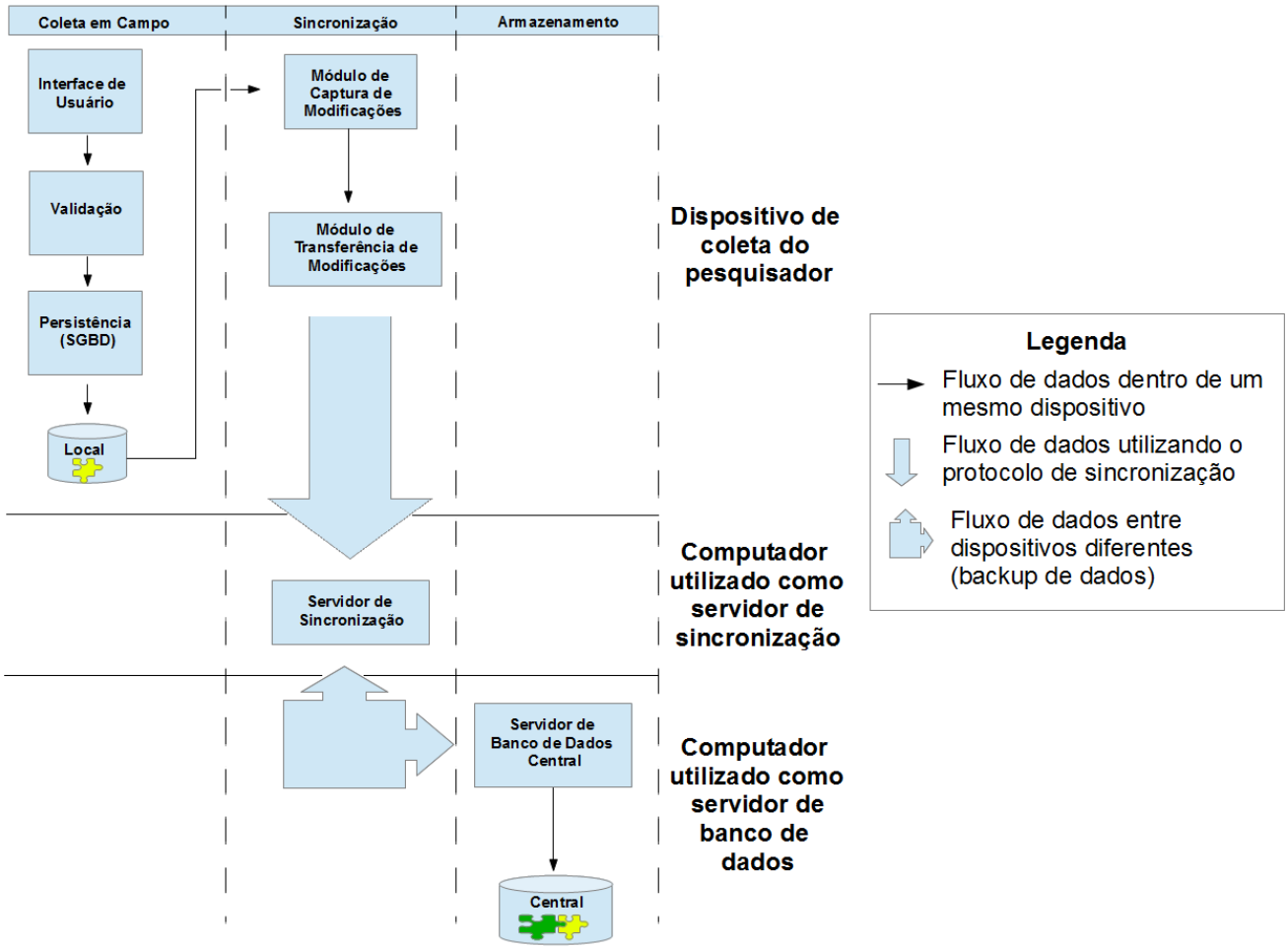


Figura 3.2: Arquitetura abstrata do sistema de coleta de dados heterogêneo.

A coleta em campo ocorre através de um sistema computacional nos dispositivos móveis dos pesquisadores. Esse sistema computacional deve possuir uma interface de usuário, um método de validação dos dados e uma camada de persistência. A interface de usuário é responsável pela entrada de dados no sistema, o método de validação é responsável por validar os dados que entraram no sistema, para que dados relevantes não sejam esquecidos e nem registrados incorretamente, e por fim, a camada de persistência é responsável pelo armazenamento dos dados no banco de dados dos dispositivos móveis.

Após a coleta dos dados, ocorre a etapa de sincronização, onde os dados coletados são sincronizados com o servidor de sincronização. O objetivo dessa sincronização é gerar réplicas, através de uma replicação total, das informações coletadas para o servidor que sincroniza os dados com os servidores de banco de dados centrais envolvidos na coleta. Essa sincronização é especificada na próxima seção. Por fim, tem-se o armazenamento

dos dados (através de um *backup* de dados) nos servidores de banco de dados centrais envolvidos na coleta.

A possibilidade de se ter mais de um servidor de banco de dados central é levado em consideração nessa arquitetura, atendendo as possíveis dimensões da coleta de dados, o que possibilita o aspecto da flexibilidade à arquitetura. Quando se tem mais de um servidor de banco de dados ocorre a replicação dos dados do servidor de sincronização para esses servidores, gerando redundância de dados, o que evita que a arquitetura tenha apenas um ponto de falha.

O protocolo de sincronização utilizado na arquitetura é que vai estabelecer os aspectos da interoperabilidade e da flexibilidade. Flexibilidade no sentido de possibilidade de mudança para adequação a contextos de coleta de dados diferentes à arquitetura. Logo, o protocolo escolhido para a arquitetura deve ter como característica a interoperabilidade e deve ser um protocolo livre.

Como o maior desafio desta arquitetura, a fase de sincronização é detalhada na próxima subseção.

3.2.1 Fase de Sincronização

Na arquitetura proposta a fase de sincronização acontece após a coleta de dados e tem como objetivo garantir o estado consistente entre os bancos de dados das unidades móveis participantes da coleta e o banco de dados central.

A sincronização proposta é dividida em dois módulos como apresentado na Figura 3.3: o módulo de captura de modificações e o módulo de transferência de modificações. O módulo de captura de modificações é responsável por capturar as instruções SQL do banco de dados do dispositivo móvel na ocorrência de alguma alteração no banco. O módulo de transferência de modificações faz o papel de um cliente, em uma arquitetura cliente e servidor, sincronizando as instruções SQL capturadas no módulo de captura de modificações com o servidor de sincronização.

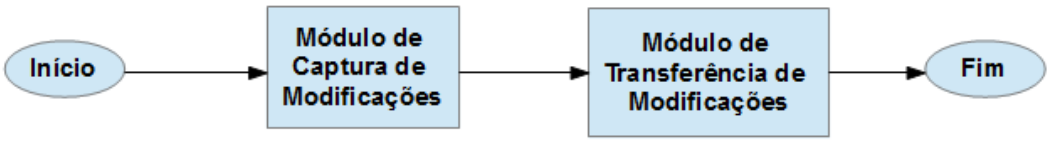


Figura 3.3: A sincronização proposta.

Módulo de Captura de Modificações

O módulo de captura de modificações é iniciado pelo pesquisador ao escolher a opção de sincronização no sistema de informação ou sistema de informação geográfico utilizado para a coleta e então começa o processo de pesquisa e captura de novas instruções SQL criadas após a última sincronização. O módulo de captura de modificações é apresentada na Figura 3.4. Esta captura ocorre através da busca de dados que possuem uma *flag* com o valor 1. Essa *flag* é que determina se o dado já foi sincronizado ou não, se ela estiver com o valor 1 o dado ainda não foi sincronizado, se ela estiver com o valor 0 o dado já foi sincronizado. Essa busca pode ocorrer em todas as tabelas do banco de dados ou em

apenas uma tabela do banco de dados que concentra todas as modificações ocorridas. A Figura mostra o diagrama de fluxo deste módulo.

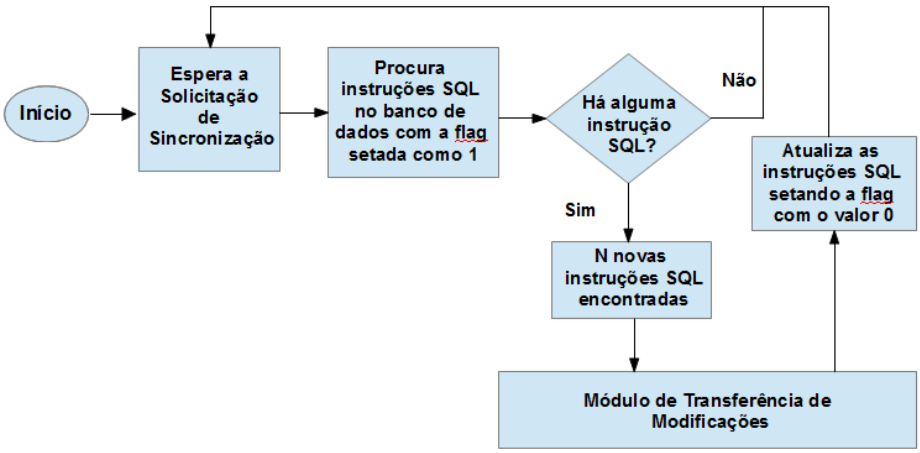


Figura 3.4: Diagrama de fluxo do módulo de captura de instruções SQL. Adaptado de [8].

Módulo de Transferência de Modificações

Após a captura das novas instruções, o módulo de transferência de modificações faz a transferência dessas instruções para o servidor de sincronização [8]. Essa transferência ocorre de acordo com o protocolo de sincronização escolhido para a arquitetura, que como já foi dito, deve ser um protocolo que tenha como característica a interoperabilidade e que seja livre. A Figura 3.5 mostra o diagrama de fluxo deste módulo.

Percebe-se no diagrama de fluxo desse módulo que se ocorrer algum erro de conexão no momento da transferência das modificações a variável n (variável presente na estrutura condicional do diagrama de fluxo) vai possuir o valor das instruções SQL que não foram transferidas. Assim, quando a conexão com a internet se reestabelecer a transferência continua normalmente de onde parou sem causar inconsistência de dados.

A Figura 3.6 mostra o diagrama de fluxo destes dois módulos juntos.

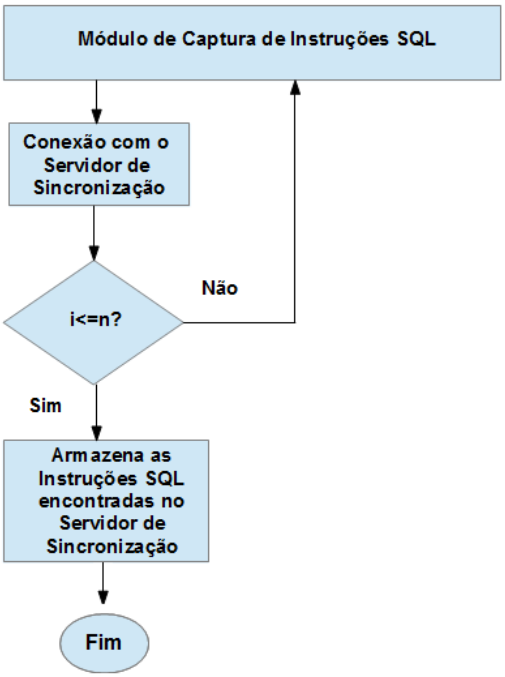


Figura 3.5: Diagrama de fluxo do módulo de transferência de modificações. Adaptado de [8].

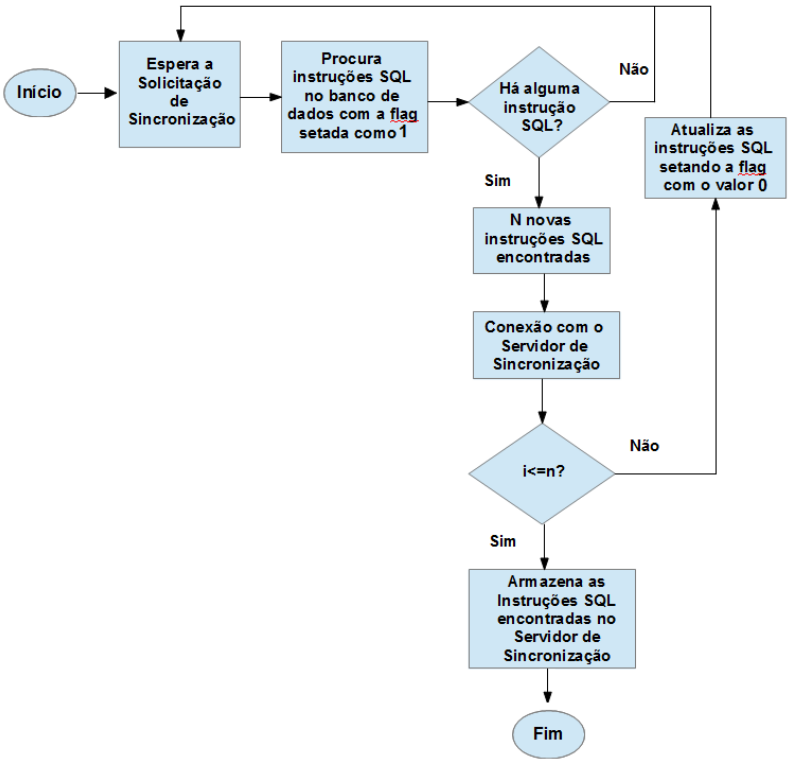


Figura 3.6: Diagrama de fluxo dos módulos de captura de instruções SQL e de transferência de modificações. Adaptado de [8].

Capítulo 4

Prova de Conceito

Neste capítulo é apresentado uma prova de conceito da arquitetura especificada no capítulo anterior e uma comparação da arquitetura proposta com as arquiteturas relacionadas. A prova de conceito foi realizada através da implementação de uma arquitetura de coleta de dados para o Instituto de Geociência da UnB.

4.1 A Arquitetura Implementada

O IG da UnB possui uma linha de pesquisa na área de sensoriamento remoto espacial cujo o objetivo é coletar dados geológicos em diversos lugares do Brasil e através da análise dos dados, gerar mapas geológicos.

Essa coleta é realizada através de cadernetas de papel, ou seja, o pesquisador coleta os dados geológicos anotando-os em uma caderneta de papel. Após a coleta, é necessário digitalizar esses dados e armazená-los em um banco de dados para uma posterior análise.

Como já foi apresentado no capítulo 2, a coleta de dados utilizando dispositivos móveis é mais eficiente e de menor custo do que a coleta utilizando questionários de papel, o que torna essa problemática um ambiente favorável para a prova de conceito da arquitetura proposta nesta pesquisa.

Essa seção apresenta a arquitetura de coleta de dados implementada no IG para a realização da prova de conceito da arquitetura proposta, ela é dividida em três subseções de acordo com as fases da arquitetura: fase de coleta de dados em campo, fase de sincronização e fase de armazenamento.

4.1.1 Fase de Coleta de Dados em Campo

A primeira fase da arquitetura, a coleta de dados em campo, foi realizada através de uma aplicação chamada *RockDroid* utilizando uma metodologia de coleta de dados baseada nas pesquisas de sensoriamento remoto espacial realizadas pelo IG da UnB. Desta forma, o propósito da aplicação desenvolvida foi facilitar a coleta, o armazenamento e a sincronização dos dados, usando, para isto, *Tablets* e *Smartphones*, recursos de computação móvel de fácil acesso aos membros da equipe de coleta.

A metodologia de coleta de dados utilizada por este trabalho consiste de duas etapas: primeiramente, os pesquisadores coletam os dados e cadastram as informações em seus dispositivos móveis; após o cadastro, os pesquisadores enviam os dados coletados para um

servidor de banco de dados central que fica na UnB (desde que haja uma conexão com a *internet* no dispositivo móvel).

A coleta dos dados é feita através do *RockDroid* instalado nos dispositivos móveis dos pesquisadores. Esse *software*, composto por uma interface com o usuário, métodos de validação dos dados inseridos e uma camada de persistência, pode ser visto, de forma abstrata, como um CRUD (acrônimo para *Create*, *Retrieve*, *Update*, *Delete*: quatro operações de manipulação de dados que podem ser realizadas em um banco de dados relacional [45]) com algumas funcionalidades de visualização de dados georreferenciadas.

A aplicação foi desenvolvida para a plataforma *Android*, versão 2.3 ou mais atual, e com a preocupação de utilizar apenas bibliotecas de componentes de código aberto para facilitar sua manutenção e disponibilização. Foi decidida a utilização da plataforma *Android*, primeiramente porque é um sistema operacional de código aberto, o que condiz com o ideal do trabalho, e depois porque é o sistema operacional presente na maioria dos *Tablets* e *Smartphones*, representando uma fatia de 59,5% do mercado no primeiro trimestre de 2013 [46]. O sistema operacional *Android* também auxilia o desenvolvedor no suporte aos aparelhos de diferentes tamanhos, formatos e especificações. Utilizando algumas boas práticas de programação é possível desenvolver um aplicativo que possa ser executado na maior parte dos dispositivos *Android*, deixando para cuidados do sistema operacional o redimensionamento dos componentes visuais.

O *RockDroid* é responsável por armazenar informações coletadas sobre os pontos geográficos, as rochas contidas neles e as amostras e estruturas de cada rocha. Ele provê formulários para o usuário preencher com os dados coletados, possui telas para apresentar informações recuperadas do banco de dados (de forma georreferenciada ou não) e exibe opções para editar e excluir registros. O *software* apresenta também alguns mecanismos de validação dos dados inseridos para garantir a integridade e a consistência do banco de dados, diminuindo assim problemas na sincronização. A parte de persistência é implementada através de um banco de dados relacional criado a partir do *SQLite*. O *SQLite* é um sistema de gerenciamento de banco de dados relacionais pequeno, comumente utilizado em sistemas embarcados e que não exige uma grande capacidade de processamento [47], ou a partir do *Spatialite*, que é o *SQlite* com extensão espacial.

A interface de usuário tem por responsabilidades exibir os formulários para que o usuário possa inserir ou atualizar informações e exibir as telas com os dados recuperados do banco. Outra forma de exibir os dados recuperados é através de um mapa, no qual o usuário pode ver os dados geograficamente distribuídos. Outro recurso do mapa é exibir o local atual do dispositivo, funcionalidade que pôde ser implementada devido aos mecanismos de localização providos pelo GPS, pelas redes de telefonia celular e pela *internet*. As Figuras 4.1, 4.2 e 4.3 mostram algumas imagens da interface de usuário do aplicativo desenvolvido – os dados não são reais, foram apenas inseridos para demonstração.

A validação dos dados antes que eles sejam inseridos no banco é imprescindível, pois garante a consistência e a integridade das informações armazenadas. Um exemplo de validação pode ser visto na Figura 4.4, na tela de inserção de ponto: os campos com um asterisco (*) são de preenchimento obrigatório. Isso significa que estes atributos no banco de dados são "*not null*". Se o usuário não preenchê-los, no momento em que ele for salvar o ponto (ou rocha, ou estrutura), o aplicativo exibirá um alerta informando uma mensagem de erro e impedirá que o registro seja salvo, evitando assim um erro de persistência.

Figura 4.1: Telas da aplicação *RockDroid* no *smartphone*.

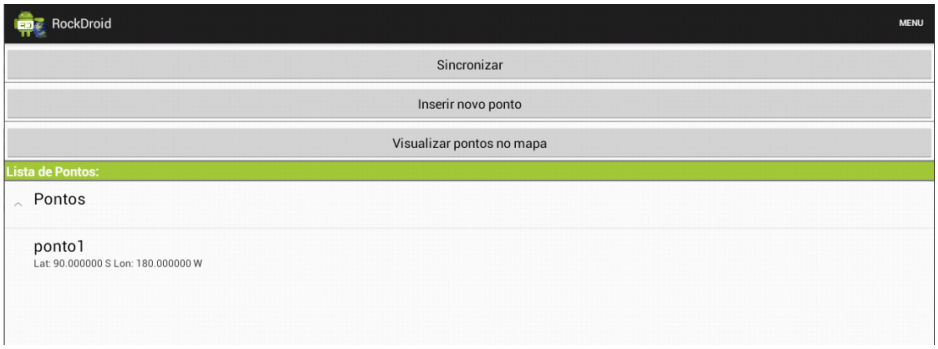


Figura 4.2: Tela da aplicação *RockDroid* no *tablet*.

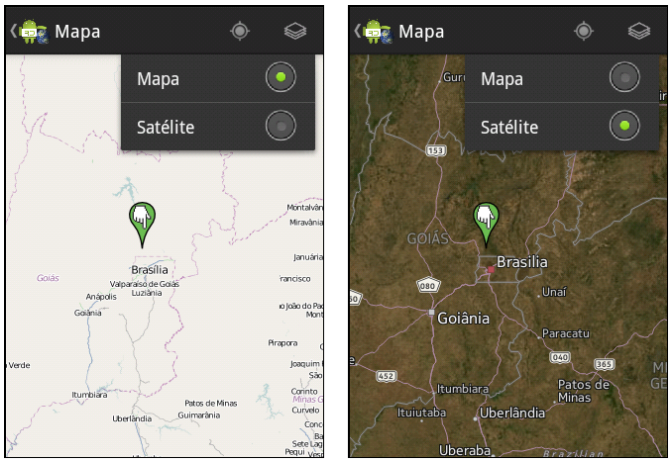


Figura 4.3: Dados exibidos em um mapa na aplicação *RockDroid*.

Existem outras formas de validação dos dados, mas estas não estão totalmente relacionadas à integridade do banco de dados. São validações relativas à limitação da faixa de valores que um campo pode assumir. Por exemplo, o valor do campo Latitude deve ser maior que 0^o e menor que 90^o. Essas limitações ajudam a evitar que dados teoricamente incorretos sejam inseridos no banco (Figura 4.4).

A parte de persistência do *software* móvel foi implementada com o auxílio do *SQLite*,

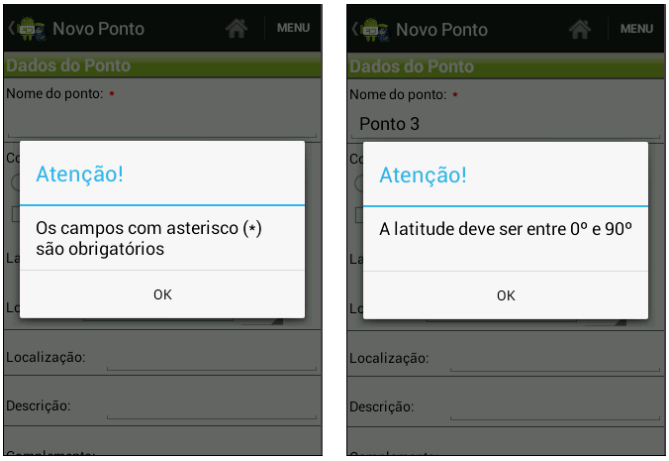


Figura 4.4: Validação na aplicação *RockDroid*.

um conjunto de bibliotecas que implementa um sistema de gerenciamento de banco de dados com as mesmas funcionalidades da maioria dos SGBDs, mas com algumas vantagens: é pequeno e simples, não exige configurações, não possui dependências externas, entre outras [47]. O sistema operacional *Android* fornece uma integração com o *SQLite* [48].

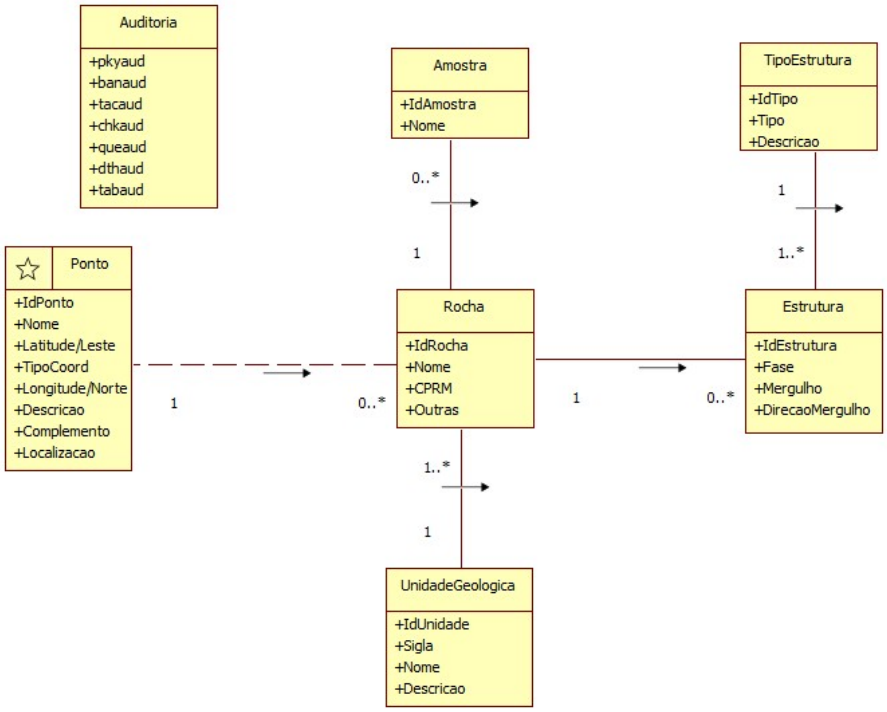


Figura 4.5: Modelo conceitual do banco de dados da aplicação *RockDroid*.

O *script* para criar o banco de dados foi embutido na aplicação. Dessa forma, sempre que for instalado em um dispositivo móvel, o aplicativo criará um banco de dados local. Se o aplicativo for desinstalado, o banco será apagado também. É importante salientar que

o banco de dados só será criado se ainda não houver um no momento em que o aplicativo for iniciado. Caso já haja um banco de dados para o *RockDroid*, ele será mantido intacto. A Figura 4.5 mostra o modelo conceitual do banco de dados do *RockDroid*.

4.1.2 Fase de Sincronização

Já na fase de sincronização da arquitetura, o módulo de captura de modificações foi desenvolvido com base na criação de *triggers* (regras que especificam ações disparadas automaticamente por meio de certos eventos [18]) que são disparadas quando ocorre uma operação de inserção, atualização ou deleção no banco de dados. Essas *triggers* têm como objetivo armazenar todas as modificações ocorridas no banco de dados em uma só tabela, facilitando a busca dessas modificações. Todas as modificações são armazenadas na tabela auditoria que tem as seguintes colunas como pode ser observado na Figura 4.6: data e hora da operação (*timestamp* da operação), o banco de dados onde ocorreu a ação, a tabela onde ocorreu a ação, a *query* executada durante a ação, e uma *flag* que indica se o registro já foi sincronizado ou não, como foi explicado na seção anterior. Inicialmente essa *flag* é configurada como 1 para todas as modificações porque nenhuma delas foram sincronizadas com o servidor de sincronização ainda.

RecNo	pkyaud	dthaud	banaud	tabaud	tacaud	queaud	chkaud
Click here to define a filter							
1	98	2013-06-25 14:13:11	BdRockDroid	TabelaUnidadeGeologica	INSERT	insert into TabelaUnidadeGeologica (_id, Sigla, Nome, Descricao) values (36, 'MGM', 'Magma', 'null');	1
2	99	2013-06-25 14:13:16	BdRockDroid	TabelaUnidadeGeologica	UPDATE	update TabelaUnidadeGeologica set _id=36, Sigla='GRT', Nome='Granito', Descricao='null' where _id=36 and Sigla='MGM' and Nome='Magma' and Descricao='null';	1

Figura 4.6: Tabela auditoria.

O protocolo *SyncML* foi escolhido para ser utilizado na arquitetura proposta porque esse protocolo atende os pré-requisitos exigidos, ou seja, promove a interoperabilidade da sincronização de dados e é um protocolo livre.

Na sincronização proposta, foi utilizada a sincronização denominada uma via do cliente para o servidor (*One-Way Sync from Client only*), onde as modificações no banco de dados dos dispositivos móveis dos pesquisadores (clientes *SyncML*) são sincronizados com o servidor de sincronização (servidor *SyncML*) como mostra a Figura 4.7.

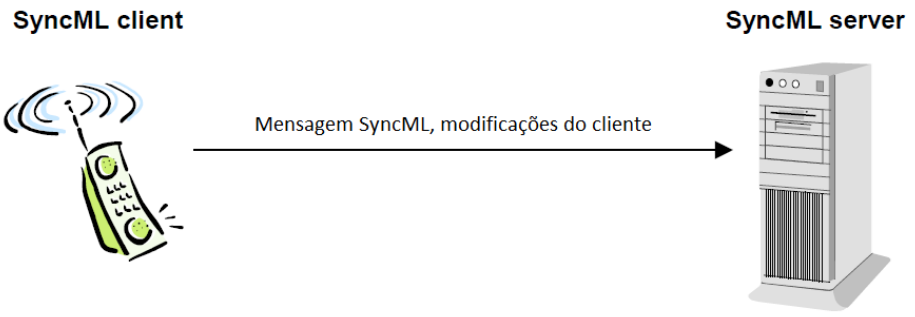


Figura 4.7: Sincronização uma via do cliente para o servidor.

O módulo de transferência de modificações foi desenvolvido com a linguagem JavaSE 1.7 utilizando-se da biblioteca *sync4j* do servidor *Funambol Data Sync Server* (FUNAM-

BOL) . O FUNAMBOL é um servidor de sincronização que implementa o protocolo *SyncML* e foi utilizado para implementar o servidor de sincronização da arquitetura [49]. Esse módulo nada mais é do que um cliente *SyncML* que consulta a tabela de auditoria, capturando as modificações no banco, e transfere as modificações ocorridas para o servidor de sincronização. Após a sincronização, as *flags* das modificações sincronizadas são configuradas como zero.

4.1.3 Fase de Armazenamento

A última fase, a fase de armazenamento, foi desenvolvida através da replicação (*backup*) dos dados do servidor de sincronização (FUNAMBOL) para o servidor de banco de dados, que foi desenvolvido com base no SGBD *PostgreSQL* com a extensão espacial *PostGIS*. Foram utilizadas plataformas diferentes nos servidores para enfatizar a heterogeneidade da arquitetura. No servidor de sincronização foi utilizado o *Linux Ubunto Server 12* e no servidor de banco de dados foi utilizado o *Windows Server 2008 R2*.

Devido essa diferença de plataforma, a implementação do *backup* dos dados foi dificultada. Foram criados *scripts* para a realização do *backup* e da restauração dos dados (executando as funções *pgdump* e *pgrestore*). Os *scripts* foram agendados para serem executados a cada 1 minuto, no *Ubuntu* esse agendamento ocorreu através da utilização do *crontable* e no *Windows* o agendamento ocorreu através do agendador de tarefas.

4.1.4 Testes e Resultados

Foram realizados três testes em um ambiente computacional heterogêneo, ou seja, em um ambiente computacional que utiliza tecnologias diferentes. Os três testes realizados foram:

- Verificar se a sincronização ocorre corretamente e em um tempo aceitável;
- Verificar se ao ocorrer uma desconexão da *internet* no meio de uma sincronização, a mesma continua de onde parou ou sincroniza novamente após o reestabelecimento da conexão;
- Verificar se a sincronização de vários dispositivos móveis ao mesmo tempo ocorre corretamente (foram utilizados cinco dispositivos móveis para o teste).

A Figura 4.8 mostra o ambiente computacional em que a arquitetura proposta foi testada. Como pode ser observado, o pesquisador coleta os dados com um dispositivo móvel (*smartphone* ou *tablet*) com o sistema operacional *Android* e o sistema de coleta *RockDroid*. Após a coleta, os dados que estão armazenados localmente, nos SGBDs *SQLite* ou *Spatialite*, são sincronizados com o servidor de sincronização que utiliza o sistema operacional *Ubuntu Server* e o Servidor FUNAMBOL que implementa o protocolo *SyncML* e usa o SGBD *PostGis* para armazenar os dados. Após a sincronização, ocorre um *backup* automático dos dados armazenados no servidor de sincronização para o servidor de banco de dados central que utiliza o sistema operacional *Windows Server 2008 R2* e armazena os dados no SGBD *PostGis*.

Fase de coleta de dados em campo:

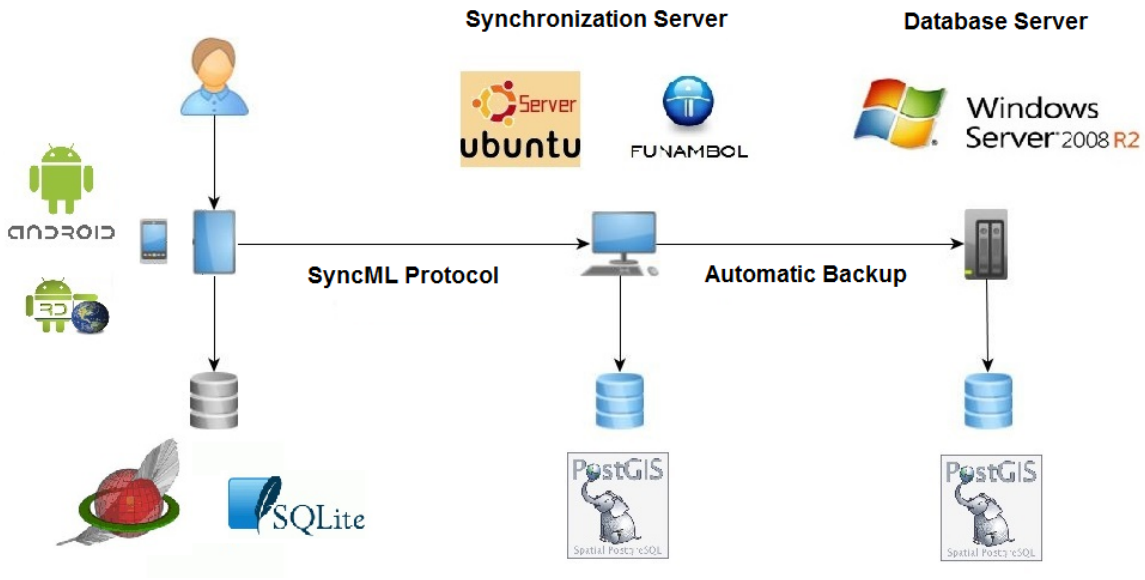


Figura 4.8: O ambiente computacional em que a arquitetura proposta foi testada.

- Dispositivo utilizado na coleta: *tablet* e *smartphone*;
- Sistema operacional: *Android* versão 2.3 ou mais atual;
- Sistema computacional: Sistema de informação geográfico(*RockDroid*);
- Tipo de dado suportado: dados convencionais e geográficos;
- Sistema gerenciador de banco de dados: *SQlite* e *SpatiaLite*.

Fase de sincronização:

- Computador utilizado como servidor de sincronização: Processador *Intel Pentium Dual Core* CPU E2140 de 1,60GHz cada, memória RAM de 3,2GB e memória secundária de 500GB;
- Sistema operacional: *linux Ubuntu Server 12*;
- Sistema gerenciador de banco de dados: *PostGIS*;
- Tipos de dados suportados: dados convencionais e dados geográficos;
- Protocolo de sincronização: *SyncML* (FUNAMBOL).

Fase de armazenamento:

- Computador utilizado como servidor de banco de dados: Processador *Intel Core* i7 de 3,40GHz, memória RAM de 16GB e memória secundária de 1TB;
- Sistema operacional: *Windows Server 8 R2*;
- Sistema gerenciador de banco de dados: *PostGIS*;

- Tipos de dados armazenados no servidor de banco de dados: dados convencionais e dados geográficos.

A sincronização foi avaliada em redes sem fio 3G e *wi-fi* e em diferentes estados da rede: internet com volume de tráfego de dados de 600 *megabits* por segundo, de 100 *megabits* por segundo e de 10 *megabits* por segundo. Durante estes testes, em todos os casos, tanto a coleta como a sincronização foram bem sucedidos e em tempos aceitáveis (de 1 a 12 segundos), o tempo de sincronização variou de acordo com o estado da rede como mostra a Tabela 3.4.

Tabela 4.1: Tempo de sincronização em diferentes estados da rede.

Tráfego de dados (Mb/s)	**Tempo de sincronização (s)**
10	8-12
100	4-8
600	1-3

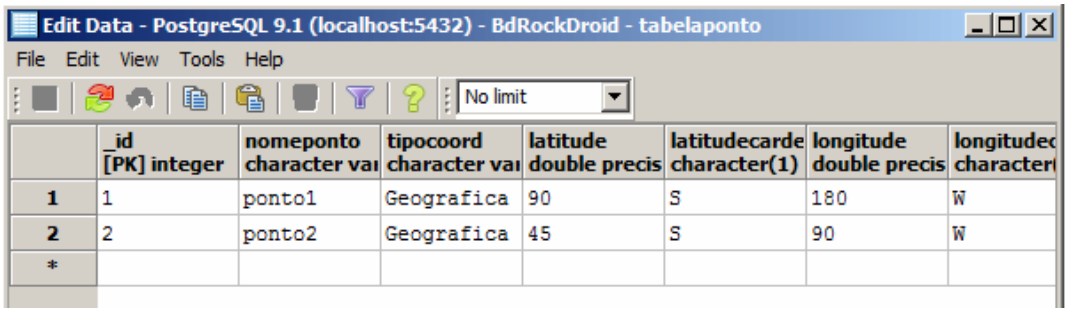
Foram realizados testes de desconexão da *internet* no meio de uma sincronização e após o reestabelecimento da conexão, a sincronização reiniciou novamente. Tanto o módulo de captura de modificações como o módulo de transferência de modificações seguiram o diagrama apresentado na Seção 3.2.1, logo, possuem tolerância a falhas. Esse teste de desconexão foi realizado da seguinte forma: inicialmente foi ativado a *internet* 3G do *smartphone* e acionado a funcionalidade de sincronização; quando a sincronização estava ocorrendo, a *internet* 3G foi desativada, gerando erro na sincronização; o servidor de banco de dados foi verificado e nenhum dado havia sido sincronizado, então foi repetido o processo de sincronização sem a desativação da *internet* 3G e a sincronização ocorreu com sucesso e os dados foram armazenados no servidor de banco de dados central.

O último teste foi a sincronização de cinco dispositivos móveis ao mesmo tempo. Esse teste tinha o objetivo de simular a realidade da coleta, porque normalmente todos os pesquisadores, após a coleta, sincronizam os seus dados simultaneamente para o servidor. Foram sincronizados cinco dispositivos móveis ao mesmo tempo e não ocorreu nenhum erro na sincronização. As Figuras 4.9 e 4.10 mostram dois dispositivos sendo sincronizados ao mesmo tempo e os dados sendo armazenados no servidor de banco de dados central corretamente.

4.2 Comparando a Arquitetura Proposta com as Arquiteturas Relacionadas

As arquiteturas de coleta de dados de mercado atendem a diversos contextos de pesquisas de campo, porém, estão vinculados a tecnologias específicas, o que é uma limitação em relação ao uso. O NDG [42] utiliza uma arquitetura de coleta que só funciona para telefones celulares *nokia*, o que fere o aspecto da interoperabilidade. Já a solução da *AuditMagic* [43] utiliza uma arquitetura de coleta fechada onde não é possível efetuar alterações para as necessidades de cada pesquisa, por não ser *open source*, ferindo o aspecto da flexibilidade.

Figura 4.9: Dois dispositivos sendo sincronizados ao mesmo tempo.



	_id [PK] integer	nomeponto character var	tipocoord character var	latitude double precis	latitudecarde character(1)	longitude double precis	longitudec characteri
1	1	ponto1	Geografica	90	S	180	W
2	2	ponto2	Geografica	45	S	90	W
*							

Figura 4.10: Dados armazenados no servidor de banco de dados central.

A maioria das arquiteturas de coleta de dados em pesquisas de campo propostas pelo meio acâdemico atendem apenas um contexto de coleta de dados, o contexto ao qual a pesquisa está inserida. O objetivo é propor uma arquitetura que atenda as necessidades da coleta de dados da pesquisa em questão. Exemplo disso é a arquitetura de coleta e disseminação de dados climáticos no estado do Piauí [7]. A arquitetura não suporta dados geográficos e utiliza tecnologias específicas que atende apenas ao contexto da coleta em questão não importando se a arquitetura pode ser utilizada em outros contextos de coleta ou não, o que fere o aspecto da interoperabilidade.

A arquitetura para sistema de coleta de dados em um ambiente de computação móvel onde o acesso à *internet* é intermitente [6] é uma arquitetura que pode ser usada em diferentes contextos de coleta de dados. Ela possui o aspecto da flexibilidade, porém, não possui o aspecto da interoperabilidade porque não suporta dados geográficos.

Já as arquiteturas propostas pela academia que não ferem os aspectos da interoperabilidade e da flexibilidade, como a arquitetura para coleta de dados de mineração de forma georreferenciada [5] e o ODK [44], usam tecnologias que podem prejudicar o desempenho da sincronização dos dados coletados em ambientes com poucos recursos computacionais.

A arquitetura proposta por esta pesquisa além de atender aos aspectos da interoperabilidade e da flexibilidade, possui um tempo de sincronização aceitável em contexto de coleta com poucos recursos computacionais, devido as características do protocolo

SyncML utilizado na sincronização dos dados.

Capítulo 5

Conclusão

A coleta de dados em trabalhos de campo através de questionários de papel, tornou-se obsoleta, pois requeria maior esforço na coleta e no processamento dos dados. O SCDM, no entanto, requer menos esforço e o custo é consideravelmente menor, tendo em vista a coleta e o processamento de dados serem simultâneos e não necessitar manipular dados físicos.

Devido as características dos SCDMs, eles possuem desafios que devem ser tratados na sua implementação. Um desses desafios é a sincronização de dados que deve utilizar um protocolo padrão para que a arquitetura de coleta seja interoperável e funcione em diferentes contextos de coleta. Porém, as arquiteturas de coleta de dados existentes atendem uma causa específica, utilizando tecnologias específicas que são limitadas no que se refere a tipo de dados, tipo de redes, tipo de sincronização, tipo de dispositivo e etc. Para cada contexto de coleta de dados diferente é proposto uma arquitetura de coleta de dados diferente limitada ao orçamento, ao ambiente computacional e a tecnologia disponível na empresa ou organização.

A arquitetura proposta por este trabalho tem como objetivo a coleta de dados para pesquisas de campo em ambientes computacionais heterogêneos onde as informações armazenadas em cada dispositivo móvel são replicadas e corretamente sincronizadas em um banco de dados central. Para chegar a esse objetivo era necessário que a arquitetura obedecesse a dois aspectos: a interoperabilidade e a flexibilidade. Essa arquitetura obedece o aspecto da interoperabilidade, através da utilização do protocolo de sincronização *SyncML*, e o aspecto da flexibilidade, através da replicação total de dados, da existência de um servidor intermediário de sincronização e por ser o *SyncML* um protocolo livre. Além disso, a arquitetura utiliza alguns mecanismos de segurança como tolerância a falhas, replicação total de dados e autenticação.

O estudo de caso realizado foi submetido a três diferentes testes relacionados com o tempo e a consistência dos dados na sincronização entre o dispositivo móvel e o servidor de banco de dados central. Em todos os testes realizados a arquitetura obteve um resultado positivo.

Com uma arquitetura desse tipo, não é mais necessário planejar uma sempre que for fazer uma coleta de dados em campo, basta usar a arquitetura proposta, que além de funcionar em ambientes heterogêneos, é de baixo custo.

A arquitetura proposta ainda será testada no campo. Esse teste será realizado com o apoio do IG da UnB e consiste na coleta de dados geológicos em diversas cidades do

Brasil.

Como trabalhos futuros tem-se o desenvolvimento de um *kit* de coleta de dados que é composto por um *framework* e uma *Application Programming Interface* (API). Um *framework* para o desenvolvimento ágil de sistemas de coleta de dados e uma API na linguagem *java* para facilitar a implementação da sincronização de dados utilizando tecnologias interoperáveis. Com esse *kit* de coleta de dados, o desenvolvimento das arquiteturas de coleta de dados se torna fácil, favorecendo as pesquisas de campo.

Referências

[1] M. Casanova, G. R. Queiroz, C. Davis, L. Vinhas, and G. Ribeiro, *Bancos de Dados Geográficos*. MundoGEO, 2005. vi, 6, 7, 8

[2] A. Tanenbaum, *Redes de computadores, Quarta Edição*. Editora Campus, 2003. vi, 9

[3] E. Pitoura and G. Samaras, "Data management for mobile computing," *Kluwer Academic Publishers*, 1998. vi, 11, 12, 14, 15, 18

[4] D. P. da Cunha, "Um estudo das estratégias de replicação e reconciliação de banco de dados móveis em um ambiente wireless," *Dissertação de Mestrado em Ciência da Computação da Universidade Federal de Santa Catarina*, (Santa Catarina - Brasil, 2003). vi, 13, 14

[5] M. Ji, Y. Sun, F. Jin, T. Jiang, J. Wang, and X. Yao, "Research and development of field data collecting synchronously system of mining area," *IEEE Internacional Geoscience and Remote Sensing Symposium*, (Hawaii - USA, 2010). vi, 23, 24, 42

[6] J. Magalhães, M. Holanda, and R. Chaim, "Arquitetura para coleta de dados em computação móvel com acesso intermitente à internet," *6ª Conferência Ibérica de Sistemas e Tecnologias de Informação*, (Chaves - Portugal, 2011). vi, 24, 25, 42

[7] A. J. S. Silva, A. S. de Andrade Júnior, and F. R. Marin, "Arquitetura para plataforma de coleta e disseminação de dados climáticos no estado do piauí," *Revista de Tecnologia de Fortaleza*, vol. 29, (Ceará - Brasil, 2008). vi, 25, 26, 42

[8] M. I. Hossain and M. M. Ali, "Sql query based data synchronization in heterogeneous database environment," *International Conference on Computer Communication and Informatics*, (Coimbatore - India, 2012). vi, 32, 33

[9] K. Thriemer, B. Ley, S. M. Ame, M. K. Puri, R. Hashim, N. Y. Chang, L. A. Salim, R. L. Ochiai, T. F. Wierzba, J. D. Clemens, L. V. Seidlein, J. L. Deen, S. M. Ali, and M. Ali, "Replacing paper data collection forms with electronic data entry in the field: findings from a study of community-acquired bloodstream infections in pemba, zanzibar," *BMC Research Notes*, 2012. 3

[10] S. Gejibo, F. Mancini, K. A. Mughal, R. A. B. Valvik, and J. Klungsøyr, "Secure data storage for mobile data collection systems," *IEEE HealthCom conference*, (Beijing - China, 2012). 4

[11] P. L. Côrtes, *Administração de Sistemas de Informação*. Saraiva, 2008. 5

[12] V. W. Setzer, "Dado, informação, conhecimento e competência," *DataGramaZero - Revista de Ciência da Informação*, 1999. 5

[13] K. C. Laudon and J. P. Laudon, *Sistemas de Informação gerenciais*. São Paulo, Brasil: Pearson, 2007. 5

[14] L. V. Bertalanffy, *Teoria Geral dos Sistemas*. Vozes, 1975. 5

[15] K. C. Laudon and J. P. Laudon, *Sistemas de Informação com Internet*. Rio de Janeiro, Brasil: LTC, 1999. 5

[16] A. C. R. Moraes, *Geografia: Pequena História Crítica*. São Paulo, Brasil: Anna Blume, 2003. 5

[17] M. Rodrigues, *Introdução ao Geoprocessamento*. São Paulo, Brasil: Sagres, 1990. 6

[18] R. Elmasri and S. B. Navathe, *Sistemas de Banco de Dados*. São Paulo, Brasil: Pearson, 2005. 7, 15, 16, 17, 18, 38

[19] O. da Dengue. Disponível em: `http://observatorio.inweb.org.br/dengue/conteudo/inicial`. 8

[20] C. C. de Planejamento da Infraestrutura Nacional de Dados Espaciais, *Plano de Ação para Implantação da INDE - Infraestrutura Nacional de Dados Espaciais*. Ministério do Planejamento, Orçamento e Gestão, 2010. 8, 9

[21] K. A. V. Borges, C. A. D. JR, and A. H. F. Laender, "Omt-g: An object-oriented data model for geographic applications," *Journal Geoinformatica, Kluwer Academic Publishers*, (Massachusetts - USA, 2001). 8

[22] G. R. Mateus and A. A. Loureiro, "Introdução à computação móvel," *11ª Escola de Computação, COPPE/Sistemas*, (Rio de Janeiro - Brasil, 1998). 9, 11

[23] D. Barbara, "Mobile computing and databases - a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, 1999. 9

[24] T. Imielinsky and B. Badrinath, *Wireless computing*, vol. 37. Communications of the ACM, 1994. 9

[25] M. Dunham and A. Helal, "Mobile computing and databases: Anything new?," *ACM SIGMOD Record*, vol. 24, (California - USA, 1995). 9

[26] G. H. Forman and J. Zahorjan, "The challenges of mobile computing," *IEEE Computer Volume 27*, 1994. 10

[27] G. C. Ito, M. Ferreira, and N. Sant'Ana, "Computação móvel: Aspectos de gerenciamento de dados," *INPE - Instituto Nacional de Pesquisas espaciais*, 2003. 10, 17, 18, 19

[28] F. J. S. Silva and M. Endler, "Requisitos e arquiteturas de software para computação móvel," *I Workshop SIDAM - Sistemas de Informação Distribuída de Agentes Móveis*, 2000. 10

[29] M. Satyanarayanan, "Fundamental challenges in mobile computing," *ACM Symposium on Principles of Distributed Computing*, (Filadélfia - EUA, 1996). 11

[30] A. Al-Bar and I. Wakeman, "A survey of adaptive applications in mobile computing," *Internacional Conference on Distributed Computing Systems Workshops*, (Arizona - USA, 2001). 12

[31] G. C. Ito, "Bancos de dados móveis: uma análise de soluções propostas para gerenciamento de dados," *Tese de mestrado em Ciência da Computação da Universidade Federal de Santa Catarina*, (Santa Catarina - Brasil, 2001). 13, 14, 18

[32] E. J. Ferreira and M. Finger, "Controle de concorrência e distribuição de dados: a teoria clássica, suas limitações e extensões modernas," *Coleção de textos especialmente preparada para a Escola de Computação*, (São Paulo - Brasil, 2000). 17

[33] A. Silberchatz, H. F. Korth, and S. Sudarshan, *Sistema de Banco de Dados*. 1999. 17

[34] S. C. Cortês and S. Lifschitz, "Banco de dados para um ambiente de computação móvel," *XXIII Congresso da Sociedade Brasileira de Computação – SBC*, (São Paulo - Brasil, 2003). 18

[35] B. R. Badrinath and S. H. Phatak, "Bounded locking for optimistic concurrency control," *Technical report DCS-TR-380*, (New Jersey - USA, 1995). 18

[36] M. Rennhackkamp, "Mobile databases tetherless computing liberates end users but complicates the enterprise," *DBMS online*, 1997. 18

[37] S. Palazzo, A. Puliofito, and M. Scarpa, "Design and evaluation of a replicated database for mobile systems," *Wireless Networks Volume 6, p.131-133*, 2000. 19

[38] Y. Breitbart, R. Komondoor, R. Rastogi, and et al., "Update propagation protocols for replicated databases," *ACM SIGMOD International Conference on Management of Data*, (Pennsylvania - USA, 1999). 19, 20

[39] M. Dantas, *Tecnologias de Redes de Comunicação e Computadores*. Rio de Janeiro, Brasil: Axcel Books, 2002. 20

[40] Ericsson, IBM, Lotus, M. C. I. C. Ltd., Motorola, Nokia, Openwave, Palm, Psion, S. Software, Symbian, and outros, *Building an Industry-Wide Mobile Data Synchronization Protocol*. SyncML White Paper, 2000. 21, 22

[41] Ericsson, IBM, Lotus, M. C. I. C. Ltd., Motorola, Nokia, Openwave, Palm, Psion, S. Software, Symbian, and outros, *SyncML Sync Protocol*. SyncML White Paper, 2002. 21, 22

[42] N. D. Gathering. Disponível em: http://www.nokia.com/corporateresponsibility/society/nokia-datagathering/english. 23, 41

[43] AuditMatic. Disponível em: http://www.auditmatic.com. 23, 41

[44] H. Singh, "Mobile data collection using an android device," *International Journal on Computer Science and Technology (IJCST)*, (Punjab - India, 2013). 24, 42

[45] L. C. Tanaka, F. M. Camargo, and R. A. Gotardo, "Sistema gerenciador de banco de dados: Sgbd exist xml," *Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica 2.1*, 2012. 35

[46] Canalys. Disponível em: http://www.canalys.com/newsroom/smart-mobile-device-shipments-exceed-300-million-q1-2013. 35

[47] A. SQLite. Disponível em: http://www.sqlite.org/about.html. 35, 37

[48] R. Lecheta, *Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. São Paulo, Brasil: NOVATEC, 2009. 37

[49] A. L. B. Alonso, C. Oliveira, L. Fedalto, F. V. Boas, T. L. G. Assis, and C. S. Hara, "Uma experiência de sincronização de bases de dados relacionais utilizando syncml," *VI Escola Regional de Banco de Dados*, (Santa Catarina - Brasil, 2010). 39