NÁDIA PUCHALSKI KOZIEVITCH

DADOS METEOROLÓGICOS: UM ESTUDO DE VIABILIDADE UTILIZANDO UM SGBD EM PLATAFORMA DE BAIXO CUSTO

CURITIBA

2005

NÁDIA PUCHALSKI KOZIEVITCH

DADOS METEOROLÓGICOS: UM ESTUDO DE VIABILIDADE UTILIZANDO UM SGBD EM PLATAFORMA DE BAIXO CUSTO

Dissertação apresentada como requisito à obtenção do grau de Mestre. Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Marcos Sfair Sunye

CURITIBA

2005

Dedico esta dissertação ao meu pai Ricardo e à minha mãe Maria que me apoiaram e incentivaram durante todo o tempo em que estive desenvolvendo este trabalho.

Também dedico ao meu orientador Marcos Sunye por toda sua competência e atenção.

AGRADECIMENTOS

Inicialmente gostaria de agradecer aos amigos Alexandre Hausen e Dieter Kahl que participaram direta ou indiretamente deste estudo.

Ao professor Carlos Carvalho do Departamento de Física pelo apoio na configuração do equipamento utilizado.

Aos supervisores de GIS-LARSC da Exxonmobil Rodrigo Pierin e Bernardino Fuentes pelo apoio durante a realização do mestrado e compreensão nas atividades do cotidiano.

Ao Simepar pela cooperação com os dados e ao incentivo pela pesquisa.

A Universidade Federal do Paraná e a todo Departamento de Informática, professores e funcionários.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	-APPLICATION PROGRAMING INTERFACE
BD	-BANCO DE DADOS
BSD	-BERKLEY SYSTEM DISTRIBUTION
DBMS	-DATABASE MANAGEMENT SYSTEM
GIS	-GEOGRAPHIC INFORMATION SYSTEM
OID	-OBJECT IDENTIFICATION
SGBD	-SISTEMA GERENCIADOR DE BANCO DE DADOS
SGBDOO	-SISTEMA GERENCIADOR DE BANCO DE DADOS ORIENTADO A OBJETO
SIG	-SISTEMA DE INFORMAÇÕES GEOGRÁFICAS
SO	-SISTEMA OPERACIONAL
TPC	-TRANSACTIONAL PROTOCOL COUNCIL

RESUMO

Atualmente, os dados meteorológicos estão sendo utilizados em diversas áreas: segurança (alertas sobre chuvas, ventanias), agricultura (previsão de tempo ajuda na decisão da plantação, períodos de seca, colheita), lazer (previsão do tempo na praia, no campo) etc. Estes dados apresentam tipos, freqüência e origem diferentes e geralmente não se encontram centralizados em uma única fonte.

Neste trabalho apresentamos um estudo de viabilidade de armazenamento de dados meteorológicos utilizando um banco de dados em uma plataforma de baixo custo. Utilizamos o software de código aberto PostgreSQL e GNU/Linux.

As características do PostgreSQL e Orientação a Objeto são brevemente descritas, com o intuito de analisar como elas podem ser aproveitadas e como o sistema funciona.

Os problemas com relação a *benchmarks*, análise de desempenho com dados meteorológicos e suas características são levantados e ao final é apresentada uma implementação.

Com os resultados aferidos foi possível apontar como o PostgreSQL trabalha com tipos de dados distintos, de tamanhos distintos, centralizando a informação somente em uma única fonte: o banco de dados.

Finalmente, em face dos resultados deste estudo, são sugeridas implementações para que este SGBD interaja melhor com este tipo distinto de dado.

ABSTRACT

Meteorological data are each day even more helpful: safety (alerts about rain, windstorm), agriculture (weather forecast helps the decision of planting, dryness period, harvest time), leasure time (weather forecast at the beach), etc. These data have different frequency, types and generally are generated in different places.

We present a viability study of meteorological data using an Open Source Database. The software used was PostgreSQL and GNU/Linux.

The characteristics of PostgreSQL is mentioned, with the objective of verifying how system works and how we can take advantage of it.

The problems with benchmarking meteorological data and its characteristics is discussed and finally is presented an implementation.

Considering the results we were able to highlight how PostgreSQL works with different types of data, different sizes, centralizing the information in just one source: the database.

Finally we suggest some implementations for this DBMS interact better with meteorological data.

1. INTRODUÇÃO

O SIMEPAR (Sistema Meteorológico do Paraná) é uma instituição tecnológica do governo, que trabalha com disseminação de informação obtida com dados meteorológicos. Apesar de já utilizar um banco de dados Oracle com suas aplicações, havia o interesse de estudar a viabilidade de utilizar um sistema alternativo, que verificasse o comportamento das consultas principais em um outro ambiente. O sistema deveria interagir satisfatoriamente com a diversidade das informações e com o sistema operacional.

Há uma grande variedade de dados meteorológicos disponíveis: sensores meteorológicos e imagens transmitidas via satélite, relatórios meteorológicos de aeroportos (METAR), sensores de raios com transmissão via modem, dados de radar, dados hidrológicos medidos manualmente em usinas hidrelétricas, etc... Apesar de terem origens, transmissões e freqüência diferentes, eles têm o mesmo objetivo: monitorar e alertar sobre os fenômenos do clima, auxiliando na tomada de decisões.

Além de possuir um relacionamento complexo de dados, agregados em vários níveis (imagens de satélite, dados de sensores, dados históricos, etc...), estes dados tem como característica um grande volume de informações.

O gerenciamento, transmissão e supervisão destes dados funcionam de forma ininterrupta e caso o sistema falhe, os mesmos devem ser acessados de modo independente. Como são uma fonte de informação, a sua qualidade é verificada, através da comparação com médias e utilização de programas especializados. Estes dados também podem gerar alertas, como avisar uma refinaria quando uma tempestade está chegando.

Os dados meteorológicos podem ser utilizados na agricultura (para determinar a época ideal de colheita, previsão de geadas, granizo), energia (controle dos níveis de reservatório de usinas, informações para fontes alternativas de energia), construção civil (realização de construções mais confortáveis, observando a insolação e umidade dos locais), transporte (condições do tempo nas estradas), segurança (alertas sobre ventanias, inundações, ressacas) , ecologia e meio ambiente (acompanhamento da qualidade do ar, monitoramento de queimadas), saúde (identificação de áreas alagadas) e lazer e turismo (verificação da previsão para feriados e épocas de férias).

Apesar de apresentarem estruturas complexas, estes dados geralmente são mapeados para o modelo de dados relacional e os dados não-alfanuméricos (imagens, gráficos, textos grandes) são mantidos fora do banco de dados. Esta segregação de informações (parte no banco de dados, parte em outros servidores) dificulta a busca e prejudica a manutenção e gerenciamento dos dados. Ou seja, o banco de dados não é utilizado, de fato, para o seu objetivo: gerenciar informações.

Quando a quantidade de dados cresce, as facilidades oferecidas por um Sistema Gerenciador de Banco de Dados - por exemplo, tempo reduzido de desenvolvimento de aplicações, controle de concorrência e recuperação de dados, suporte a indexação e capacidade de consultas, se tornam importantes e necessárias. Para suportar estas aplicações, um banco de dados deve dar suporte a tipos de dados diferenciados. Os conceitos de Orientação a Objeto influenciam fortemente um banco de dados a utilizar dados diferentes e complexos. Sua característica básica é manipular grupos de informações como objetos. Isto facilita o entendimento do usuário (gerando interfaces de fácil compreensão), produz programas extensíveis e que podem ser reutilizados, além de resultar em boas interfaces. A intenção é que se os objetos forem bem escolhidos, o usuário pode formar facilmente um modelo mental do sistema.

A aplicabilidade da Orientação a Objeto já foi comprovada como um bom método para dados geográficos em OOSTEROM e BOS (1999). Por exemplo, é possível criar um objeto bacia hidrográfica, englobando seus rios, os seus vertimentos e sua vazão final.

O restante deste trabalho está dividido em 9 capítulos. O capítulo 2 apresenta uma visão geral dos objetivos e resultados esperados. O capítulo 3 apresenta uma breve descrição de trabalhos relacionados a banco de dados e suas ferramentas, orientação a objetos e *benchmarks*. O capítulo 4 apresenta conceitos do PostgreSQL e o capítulo 5 traz informações sobre *benchmarks* e os problemas para trabalhar com dados meteorológicos. O capítulo 6 apresenta a metodologia utilizada e os problemas encontrados. O capítulo 7 apresenta a implementação de um banco de dados utilizando PostgreSQL e dados verídicos. No capítulo 8 os resultados são detalhados e a viabilidade da utilização do PostgreSQL analisada. O capítulo 9 apresenta a conclusão.

2. OBJETIVOS E RESULTADOS ESPERADOS

Ao se tratar de dados meteorológicos, podemos utilizar dados oriundos de várias fontes: satélites, sites pessoais, informações disponibilizadas por órgãos públicos, etc... Além disso, estes dados podem ser de vários tipos: imagens, dados decodificados, cadeia de caracteres, etc.. onde cada um é disponibilizado em diferentes horários e formas.

Todo tipo de dado é importante para uma tomada de decisão, desde um pequeno agricultor que decide se vai plantar soja dependendo da temperatura média até uma plataforma petrolífera, que pára temporariamente porque uma tempestade com grande quantidade de raios se aproxima. Para tomar estas decisões, é possível ser necessário o histórico das informações dos últimos 30 anos quanto os dados dos últimos 5 minutos.

Atualmente, com o grande avanço tecnológico de máquinas (capacidade de processamento e grande quantidade de memória), existe a possibilidade de gerenciar todos os dados em somente uma fonte: o banco de dados. Isso facilitaria tanto para o usuário final, onde apenas uma consulta já retorna todos os tipos de dados disponíveis, quanto para a manutenção das informações, pois tudo fica centralizado em uma máquina apenas.

Da mesma maneira que o processamento das máquinas evoluiu, os métodos para desenvolver banco de dados também sofreram modificações. Antigamente os dados eram simples e proveniente de uma única fonte. Atualmente os dados são complexos e envolvem imagens (grandes, pequenas, coloridas), textos, vídeos, sons, etc... Além da mudança dos tipos de dados, é necessário criar sistemas que sejam mais fáceis de entender pelo usuário final, que sejam fáceis de reutilizar e modificar. Além disso, almeja-se criar sistemas que tenham interfaces claras, que agreguem conhecimento (contenham regras de integridade que fazem parte da semântica da aplicação) e que sejam rápidos, mesmo quando gerenciam um volume grande de dados.

Em sistemas meteorológicos que utilizam bancos de dados orientados a objeto, os dados armazenados em geral não diferem dos dados mantidos em bancos de dados relacionais: guardam somente dados alfa-numéricos, apesar de toda a estrutura complexa que dados geográficos podem apresentar. Um exemplo de organização que ocorre é guardar dados de sensores no banco de dados, porém imagens de satélite, dados de radar e

dados de previsão de vazão de usinas em arquivos, utilizando como organização a subdivisão de diretórios. Um exemplo seria o diretório 2004 com subdivisão de meses e posteriormente uma nova divisão em dias.

O objetivo deste trabalho é estudar a viabilidade de utilizar um banco de dados de plataforma de baixo custo – PostgreSQL para o armazenamento de dados meteorológicos, tirando proveito de suas características de orientação à objeto. O comportamento do sistema e suas características serão analisados.

A expectativa é que o sistema contribua para uma nova opção de SGBD para este tipo de informação, possibilitando sua centralização em uma única fonte e o gerenciamento de dados com estruturas complexas e de diferentes origens.

3. REVISÃO DE LITERATURA

A replicação é utilizada para aumentar o desempenho e disponibilidade de dados. A replicação pode ser síncrona e assíncrona, gerenciada por somente um servidor ou utilizando grupos. O objetivo é sempre diminuir concorrência por recursos e aumentar o desempenho, mantendo os dados em segurança. GRAY et al (1996) sugerem a replicação cliente-servidor, utilizando os seus benefícios. Em ZHANG et al (1996) é apresentada a clusterização, um outro método utilizado para trabalhar com banco de dados de grande porte.

Na comunidade de software livre, há 2 opções de bancos de dados que são amplamente utilizados: MYSQL (2004) e POSTGRESQL (2005). Por padrão, eles estão limitados na replicação cliente-servidor. Existem vários projetos envolvendo software livre e replicação. O ERSERVER (2004) é um sistema de replicação assíncrono, baseado em *triggers*. Há projetos que combinam software livre com pesquisa acadêmica: como o POSTGRES-R (2004). Ele é apresentado em KEMME e ALONSO (2000), e seu desempenho é avaliado em redes locais. O método de replicação utilizado é ávido *(eager replication*) ao contrário da maioria dos produtos comerciais que utilizam replicação preguiçosa. Mais recentemente, os mesmos autores apresentaram um trabalho com bancos de dados em clusters tolerantes a falhas (...) JIMENEZ-PERIS et al (2000).

Um exemplo de replicação utilizando Postgres-R é apresentado em ALMIR et al (2002), com testes de desempenho em ambientes locais e de rede. Neste trabalho, todas as réplicas no Postgres-R podem atuar como nodos centrais, e o desempenho não diminui quando servidores são adicionados.

O propósito do projeto PostgreSQL era construir um banco de dados para retificar as deficiências dos bancos de dados relacionais. O sistema precisaria manipular objetos que poderiam ser dados não tradicionais, como bitmaps, ícones, textos e polígonos) e conhecimentos, ou seja, uma coleção de regras que faziam parte da aplicação. Em STONEBRAKER e KEMNITZ (1991) há a descrição dos conceitos básicos deste sistema: a linguagem Postquel, as regras, funções, operadores, tipos abstratos de dados e outros recursos para agregar esta manipulação de objetos e conhecimento. Ainda no artigo há a

explicação de como foi implementado o PostgreSQL além de uma análise de desempenho, com *benchmarks* e comparação com o banco de dados Ingres.

Em KIM e ROSSITER (1994) é apresentado um estudo do modelo objeto-relacional no contexto de banco de dados relacionais e orientados a objeto. O modelo objeto-relacional é descrito, com ênfase no PosgreSQL. O mesmo é utilizado para a implementação de um banco de dados administrativo com funções extendidas para calcular dados sumarizados. O trabalho demonstrou que aplicações robustas podem ser desenvolvidas nesta área.

Vários estudos têm sido feitos ultimamente em relação a dados geográficos. Em relação à sua modelagem, como cada sistema implementa seu próprio modelo de dados, não é possível definir um conjunto único de regras de mapeamento. A descrição de um modelo de dados de alto nível é apresentada em FILHO et al (1999) , no qual são definidos os elementos do domínio da aplicação. O projetista não precisa, em um primeiro instante, preocupar-se com as estruturas de armazenamento dos dados.

Em NAHOURAII e PETRY (1991) são descritos os conceitos, exemplos e aplicações de banco de dados orientado a Objeto. É citado SmallTalk, uma linguagem de programação modular, eficiente para trabalhar com sistemas complexos. Todos os seus códigos e dados são representados como objetos. Em OOSTEROM e BOS (1999), são analisados os benefícios do método de orientação a objeto para linguagens de programação e banco de dados. É citada a sua aplicabilidade em sistemas complexos.

A modelagem, implementação e visualização de dados geográficos, utilizando o banco de dados estensível orientado a objeto O2 é descrito em PEREZ e BATISTA (1997).

Em HARA (1997) é apresentada uma implementação com dados de geoprocessamento, utilizando SPRING (Sistema de Processamento de Informações Geo-Referenciadas), o sistema UNIX (Solaris 2.5) e o banco de dados Codebase. O trabalho têm como ênfase os dados cadastrados em um banco de dados espacial, com o objetivo de integrar dados tabulares e gráficos através de interfaces.

Medir o desempenho de um banco de dados é uma tarefa complexa. Quando não há um padrão de ferramentas utilizadas por um sistema é difícil comprovar que um sistema é mais eficiente que o outro.

O TPC (2004) surgiu como um dos principais organismos realizadores de *benchmarks*. Para atingir o objetivo de prover dados de desempenho relevantes para a

indústria, o TPC requer que o teste de *benchmark* seja implementado com sistemas, produtos, tecnologias e preços que estejam disponíveis e que possam ser implementados por um número significativo de usuários. O objetivo é gerar informação relevante ao segmento de mercado.

As medições realizadas pelo TPC utilizam SGBD's, sistemas operacionais e hardware proprietários, como, por exemplo: SGBD Oracle em máquina HP com Unix Tru64, SGBD Sybase em máquina SUN com Unix Solaris ou SGBD SqlServer em máquina Dell com Windows 2000.

Como não são apresentadas medições de SGBDs livres, verificou-se a necessidade de analisar como o banco de dados PostgreSQL trabalha, como interage com outras aplicações, como trata dados diversificados (imagens, cadeia de caracteres, etc...) e se existem projetos tratando banco de dados geográficos. Estas informações serão tratadas no próximo capítulo.

4. POSTGRESQL

Os dados meteorológicos possibilitam a emissão de alertas sobre eventos severos, possibilitando a antecipação de medidas emergenciais e a adequação a cada situação das equipes de salvamento, busca e reparos. Os mesmos eventos não respeitam feriados, fim de semana ou horário, requerendo sistemas que estejam altamente disponíveis (24 por 7).

Para auxiliar a alta disponibilidade, é utilizada a redundância de dados. Com ela, é garantida a disponibilidade imediata quando há alguma interrupção no servidor principal, utilizando a replicação de dados em tempo real para um servidor secundário ou alvo.

Estes sistemas precisam de supervisão ininterrupta (checagem de espaço em disco, checagem de dados que possam gerar alertas, verificação de sensores, etc...) e um banco de dados confiável, robusto, com baixos tempos de resposta e tolerante a falhas . Uma falha de disco em um banco de dados deste tipo durante uma tempestade pode ser fatal, pois todo o sistema pára e não serve para o seu principio fundamental: informar , alertar e prevenir.

A necessidade de modelar o mundo real mais adequadamente, facilitando a dedução lógica e a computação geométrica nos Sistemas de Informações Geográficas é avaliada em PISSINOU et al (1993). É proposto um modelo para interagir com dados geográficos, sistemas inteligentes e de multimídia, utilizando Orientação a Objeto.

Ainda em PISSINOU et al (1993) são citadas algumas perguntas fundamentais que ocorrem na integração de dados de GIS:

- Como manejar grandes quantidades de tipos diversificados de dados e evolução dinâmica de tipos?
- Como integrar imagens computacionais gráficas com estruturas de dados com atributos complexos ?
- Que método pode ser empregado para manejar diferentes visões de dados?
- Como manipular dados persistentes e não persistentes?
- Quais são os aspectos temporais e espaciais dos objetos e quais propriedades de um objeto são essenciais em um domínio geográfico?
- Qual método pode ser empregado para identificar consultas inteligentes e genéricas, e quais técnicas de processamento de consultas são apropriadas?

- Como integrar, fácil e diretamente, dados de multimídia com dados temporais e espaciais?
- Uma interface gráfica tri-dimensional é uma necessidade para aplicações SIG?
- Como manipular grandes volumes de dados e quais métodos deveriam ser usados para executar tabulações numéricas estensivas em dados?
- Que algoritmos, estruturas de dados e métodos de acesso são os melhores para implementar um modelo? Em particular, como prevenir uma explosão exponencial do espaço requerido?

Baseado nas questões citadas anteriormente, o trabalho apresenta um modelo estensível para trabalhar com dados geográficos.

Já em HARA (1990), é utilizado um protótipo de SGBD em um ambiente de desenvolvimento de software. O SGBD utilizado chama-se Demókles , cujo propósito é dar suporte a ambiente de projeto. Seu modelo estende o modelo relacional com objetos, versões e um tipo de dados para campos longos. Pode ser classificado como um SGBDOO com enfoque estrutural. O trabalho cita a dificuldade de integração de potencial semântico de um SGBDOO com a funcionalidade de linguagens de programação Orientadas a Objetos. Não havia ainda a facilidade de criar novos tipos de dados. Ou seja, era difícil ter um modelo orientado a objetos que integrasse banco de dados e linguagem de programação.

Com o passar do tempo, os bancos de dados passaram a manipular objetos, dados não tradicionais como bitmaps, ícones, textos e polígonos e conhecimento, expresso como uma coleção de regras que faziam parte da aplicação. Para retificar estas deficiências em bancos de dados relacionais existentes, foi criado o PostgreSQL. Em STONEBRAKER e KEMNITZ (1994) são descritos os conceitos básicos deste sistema: a linguagem Postquel, as regras, funções, o armazenamento de dados e como funcionam para ajudar a administração de objetos e conhecimento. O trabalho também apresenta como foi implementado o PostgreSQL .

De acordo com POSTGRESQL (2005):

O POSTGRES é um sistema gerenciador de banco de dados criado em 1986 em Berkeley na Universidade da Califórnia, como sucessor do INGRES, banco de dados

relacional, desenvolvido na mesma universidade pelo Prof. Michael Stonebraker e apoiado pelo DARPA (Denfese Advanced Research Projects Agency), pelo ARO (Army Research Office) e pelo NSF (National Science Foundation).

Em 1995 o Postgres foi renomeado para Postgres95. Foi acrescentado suporte a linguagem SQL, otimizou-se o código tornando-o 30% mais rápido e passou a ser distribuído na internet como código aberto. Em 1996 ele foi renomeado para PostgreSQL.

O Postgres organiza os dados segundo o modelo relacional, que é o mais utilizado hoje em dia. Oracle, DB2, interbase, Microsoft SQL Server, Sybase, etc são todos bancos de dados relacionais. Ele é também um modelo relacional estendido, que dá suporte aos conceitos de objetos, OIDs (Object Identifiers), objetos compostos, herança múltipla, versões, dados históricos, e uma linguagem de consulta, POSTQUEL, extensão da linguagem do INGRES, QUEL .

Na própria definição do PostgreSQL ele é citado como relacional estendido, e não Orientado a Objeto (apesar de ter características como herança e OID).

De acordo com NAHOURAII e PETRY (1991), o método de Orientação a Objeto é utilizado para analisar, projetar e desenvolver sistemas complexos. Ele oferece benefícios como decomposição consistente de uma aplicação, abstração de dados, representação estrutural de conhecimento e reutilização de código. Os valores não se restringem aos tipos tradicionais (integer, real, string, etc...), mas podem ser de tipos definidos pelos usuários (tipos de dados abstratos), como por exemplo região e bacias hidrográficas. As relações entre as entidades do mundo real podem ser mapeadas e implementadas diretamente, facilitando a navegação semântica e entendimento do usuário. O objetivo deste trabalho não é questioná-lo como relacional estendido ou OO, mas sim aproveitar todas as suas características para uma melhor utilização de seus recursos.

O PostgreSQL suporta operadores, métodos funcionais de acesso e tipos de dados definidos pelo usuário, por isso é considerado altamente estensível. Também tem suporte à linguagens internos, incluindo a linguagem nativa denominada PL/pgSql. Esta linguagem é comparada a linguagem procedural do Oracle, Pl/Sql. Outra característica do PostgreSQL é sua habilidade de usar Perl, Python, ou TCL como linguagem procedural.

O SGBD PostgreSQL também é considerado flexível para o desenvolvimento em API (*Application Programming Interface),* ou seja, o conjunto de programas ou chamadas

que permitem uma aplicação se comunicar com outro programa ou sistema. Estas interfaces incluem Object Pascal, Python, Perl, PHP, ODBC (Open Database Conectivity - interface padrão que permite uma aplicação acessar diferentes tipos de fontes de dados), Java/JDBC, Ruby, TCL, C/C++ e Pike.

Ainda citando aplicações, podemos mencionar a facilidade de trabalhar com o desenvolvimento incremental, já que a adição de novos objetos ao sistema pode ser feita sem a modificação dos códigos já existentes.

A arquitetura Cliente/Servidor utilizada no PostgreSQL é similar ao método do Apache 1.3.x para gerenciar processos (...) POSTGRESQL (2005). Há um processo principal que se ramifica para proporcionar conexões adicionais para cada cliente que queira se conectar ao banco. Sua arquitetura escala muito bem o aumento de cpu e memória.

Por ser considerado código aberto, com licença BSD (software livre), o PostgreSQL pode ser acessado, modificado e melhorado por seus próprios usuários. Isso possibilita uma evolução contínua, com alterações no código e adaptações, como a sua última versão em 64 bits, por exemplo.

O PostgreSQL também têm suporte à OID's. OID é um valor que é assignado a um objeto no banco de dados e é único durante toda a sua existência. O Sistema Gerenciador de Banco de Dados é responsável por gerar OIDs e assegurar que um OID identifique e diferencie um objeto. O OID também é utilizado quando o tamanho de um objeto é grande, ou porque é um tipo de dado estruturado ou porque é grande, como uma imagem (...) RANAKRISHVAM e GEHRKE (2003).

Há uma extensão do PostgreSQL para permitir o armazenamento de dados espaciais, chamada POSTGIS (2004). Esta extensão trabalha com objetos e funções GIS, que englobam ponto, linha, polígono, multiponto, multilinha, multipolígono e coleções geométricas.

Exemplos de representações textuais de objetos espaciais que são comportados incluem :

Point (0 0) – XY

Point (0 0 0) – XYZ

PointM (0 0 0) – XYM

Point (0 0 0 0) –XYZM

Linestring(0 0,1 1, 12)

Polygon((0 0 0,4 0 0, 4 4 0, 0 4 0, 0 0 0),(1 1 0, 2 1 0, 2 2 0, 1 2 0, 1 1 0))

Multipoint(0 0 0, 1 2 1)

MultilineString((0 0 0,1 1 0, 1 2 1),(2 3 1, 3 2 1, 5 4 1))

Multipolygon(((0 0 0,4 0 0, 4 4 0, 0 4 0, 0 0 0),(1 1 0, 2 1 0, 2 2 0, 1 2 0, 1 1 0)),((-1 –1 0, -1 –2 0, -2 –2 0, -2 –1 0, -1 –1 0)))

GeometryCollection(Point(2 3 9), Linestring((2 3 4, 3 4 5)))

Note que os exemplos citados trabalham com 2 (XY) , 3 (XYZ e XYM) e 4 (XYZM) coordenadas dimensionais.

Estes formatos são especificados no *Open GIS Well Known Format*, como as extensões XYZ, XYM, XYZM). Os objetos GIS existentes no PostGIS têm as características básicas definidas pelo Consórcio OpenGIS (OGC). A especificação OpenGIS define dois tipos padrões para expressar objetos espaciais: o formato Well-Known (WKT) e o formato binário Well-Known (WKB). Ambos os formatos incluem informações sobre o tipo do objeto e as coordenadas que o formam.

A aplicação de PostGIS é vetorial, indicada para bancos de dados espaciais. Devido a esta natureza, não o utilizamos neste trabalho, já que dados meteorológicos não somente tratam de imagens, mas de dados alfa-numéricos, cadeias de caracteres e outros tipos. Além disso, as imagens aqui tratadas são do tipo *raster*, não possibilitando a navegação por coordenadas. Caso fosse utilizado o PostGIS, as consultas para a adequação do PostgreSQL teriam que ser refeitas, adaptando-as para a sua linguagem utilizada para o tratamento dos objetos GIS. Isto não invalida o estudo para aplicações que utilizam o PostGIS pois este é uma extensão do PostgreSQL, e várias bibliotecas deste último são utilizadas como base para a implementação em objetos GIS, como por exemplo, as funções para gerar índices.

Após verificarmos como funciona o PostgreSQL e sua interação com o ambiente,

surgiu a necessidade de analisarmos metodologias para avaliar o banco de dados: quais consultas avaliar, qual o período de dados a serem utilizados, como simular um ambiente real, além de considerar aspectos característicos de banco de dados meteorológicos, como tipos de dados diferenciados e dados que dependem de uma data no tempo.

5. ANÁLISE DE BANCO DE DADOS

Medir o desempenho em sistemas é uma tarefa difícil. Principalmente tratando-se de SGBD's, que envolvem tipos de dados complexos e de tamanhos diferenciados. Existem algumas questões básicas que devem ser analisadas antes de medir o desempenho de um sistema.

A questão inicial analisada é avaliar se o *benchmark* irá utilizar dados sintéticos ou reais. Os dados sintéticos permitem um controle maior da entrada/saída dos dados quando o *benchmark* é executado, porém pode-se questionar quão válidos são os resultados, já que não trabalham com dados verídicos.

Após a escolha do tipo de dado utilizado, é necessário determinar o tamanho ideal de um banco de dados para medir seu desempenho, pois uma parcela do banco de dados real terá que ser caracterizada com uma quantidade de dados significativa. E como mesurar que os dados escolhidos são pouco ou bastante abrangentes?

Posteriormente há a questão de verificar o tipo de hardware utilizado, pois a análise de desempenho de um sistema pode ser alterada devido a este fator. E se o hardware muda, quão válida é a análise?

E finalmente, a última questão analisada: avaliar se a análise de desempenho do sistema será feita em modo mono-usuário ou multi-usuário.

Em uma primeira fase, geralmente executa-se um *benchmark* em modo mono-usuário, com dados sintéticos para obter-se medidas de performance em uma condição ótima. Em uma segunda fase é executado o *benchmark* para verificar a operação do sistema com uma carga mais realista.

A maioria dos *benchmarks* atuais utilizam dados sintéticos, pela facilidade de manipulá-los em ambientes diferentes. Em BITTON et al (1983) é apresentado um estudo sobre um conjunto de consultas que podem ser utilizadas para medir o desempenho em bancos de dados relacionais. São apresentados os resultados de *benchmarks* executados em vários bancos de dados convencionais, simulando um ambiente mono-usuário.

As limitações do resultado de um *benchmark* devido ao tamanho do banco de dados é um dos problemas apresentados neste trabalho. Também são mencionadas as dúvidas sobre quais consultas utilizar para avaliar adequadamente um banco de dados, a questão de como medir o tempo e recursos consumidos e se o *benchmark* deve ser em ambiente mono-usuário ou multi-usuário.

As consultas utilizadas por este *benchmark em* BITTON et al (1983) para avaliar o desempenho de SGBD's envolvem consultas com fatores de seleção diferentes, projeções, junção de tabelas, agregações e modificações (*append*, *delete*, *modify)*.

De acordo com o mesmo artigo, a velocidade na qual um SGBD pode processar uma operação de seleção depende de como são guardados fisicamente os arquivos, do hardware utilizado e do fator de seleção (quantas tuplas são resultado da consulta). Após a consulta, ainda há a avaliação do custo de enviar as tuplas resultantes para a tela (comparado com o custo de guarda-las em uma nova relação).

O artigo utiliza o teste em um sistema mono-usuário com o objetivo de isolar os efeitos da configuração de hardware específicas, as características de sistema operacional e os algoritmos de execução de consultas. Os dados utilizados para popular o banco são gerados aleatoriamente, com o propósito de obter valores uniformemente distribuídos. O objetivo é não criar nem um banco de dados pequeno – que não reflita aspectos de uma base real – nem um banco de dados grande demais – que não permita a flexibilidade de utilizar um *benchmark* sistemático.

Já em BORAL e DEWITT (1984) é apresentada uma metodologia para avaliar o desempenho de SGBD's e sistemas em modo multi-usuário. Os dados utilizados também são sintéticos.

Neste trabalho são citados 3 fatores principais que afetam o desempenho de um SGBD em um ambiente multi-usuário: o primeiro é o nível de multi-programação utilizado (número de consultas em qualquer fase da execução), o segundo é a seleção de consultas (escolha de um pequeno conjunto de consultas representativas) e o terceiro é o nível de compartilhamento de dados (múltiplas consultas podem estar acessando as mesmas relações concorrentemente).

Um problema que pode ocorrer com o uso de *benchmarks* proprietários é que eles podem ser "adaptados" ao sistema, gerando resultados manipulados. Com isso surge a necessidade de um *benchmark* genérico, que possa ser utilizado por qualquer sistema ou empresa.

O TPC (*Transaction Protocol Council*) é um *benchmark* para medir o desempenho de banco de dados. De acordo com TPC (2004), ele pode ser definido da seguinte forma:

O TPC é uma corporação sem fins lucrativos com o objetivo de definir benchmarks para bancos de dados e processamento de transações além de disseminar dados TPC de desempenho, objetivos e confiáveis para a indústria. ...Visando o modo computacional, uma transação pode ser definida como um conjunto de operações incluindo leitura/escrita de disco, chamadas ao sistema operacional ou alguma forma de transferência de dados de um subsistema para outro. O desempenho é medido em termos de quantidades de transações que um determinado sistema de banco de dados pode executar em uma unidade de tempo; por exemplo: transações por segundo ou transações por minuto.

O TPC permite comparar desempenho de plataformas, avaliar o custo do desempenho, avaliar o processador versus desempenho real, desempenho de software, avaliação do custo do software, comparar tecnologias.

Atualmente as grandes empresas que possuem seus produtos avaliados em *benchmarks*, sejam estes produtos software ou hardware, são membros do TPC como IBM, Microsoft, Oracle, Sybase, Sun e outras.

As principais divisões do TPC são Tpc-C, Tpc-H e Tpc-H.

O Tpc-C verifica o desempenho de usuários executando transações em um banco de dados.

O Tpc-H é um *benchmark* de suporte a decisões, que consiste de um conjunto de consultas orientadas ao negócio e modificações concorrentes de dados. Ele implementa um banco de dados fictício e possui um gerador de dados para a base de dados. Ele também tem um conjunto de consultas pré-estabelecidas para testes, simulando um ambiente real.

O Tpc-W é um *benchmark* para verificar o desempenho de servidores web orientados a transações.

As principais regras de implementação do TPC são:

- O banco de dados deve ser implementado usando um SGBD comercialmente disponível;
- No final do teste de carregamento as tabelas devem ter o número exato de linhas para o fator de escala;
- Os nomes das tabelas e colunas devem estar em conformidade com os descritos pelo TPC;
- Não é permitida a inclusão de colunas além das especificadas;
- O banco de dados deve permitir a inclusão de dados arbitrários em conformidade com o tido do dado e restrições opcionais definidas;
- O particionamento horizontal é permitido e sendo baseado num campo, que pode ser chave primária, chave estrangeira ou uma coluna única de dado;
- Alguns particionamentos requerem o uso de diretivas que especifiquem os valores dos campos de particionamento. Estas diretivas não podem ser baseadas em nenhum conhecimento dos dados nas tabelas, exceto os valores mínimo e máximo do campo de particionamento. E dentro das limitações de divisão de números inteiros, devem ser definidas partes iguais dentro do conjunto entre os valores mínimo e máximo. E finalmente, as diretivas devem permitir a inclusão de valores na coluna de particionamento que estejam fora do conjunto entre os valores mínimo e máximo.

O TPC seleciona 22 consultas de seleção de dados e 2 consultas de atualização. As principais características das consultas selecionadas pelo TPC são o seu alto grau de complexidade, uma natureza ad-hoc (consultas aleatórias), possuem parâmetros que se alteram entre as execuções e são consultas que diferem uma das outras. Além disso as consultas procuram executar vários tipos de acessos (seqüencial, aleatório) e examinam uma grande porcentagem de dados disponíveis.

Estas consultas provêem os resultados de preço e promoções, suprimentos e gerenciamento de demanda, lucro e gerenciamento de rendimentos, estudo de satisfação do cliente, estudo de participação de mercado e gerenciamento de embarque de mercadorias.

O teste de desempenho utilizado pelo TPC consiste em duas execuções que são o teste de poder (p*ower test*) e teste de produtividade *(throughput test)*.

No teste de poder é medida a execução simples de cada consulta com usuário único.

No teste de produtividade é testada a habilidade do sistema processar a maioria das consultas na menor quantidade de tempo – este teste é usado para demonstrar o desempenho no ambiente multi-usuário.

Cada etapa de execução é composta por uma consulta, um conjunto de consultas, uma seqüência de consultas, uma seqüência de atualizações e sessões.

A consulta é a execução de uma das vinte e duas consultas TPC-H descritas.

A seqüência de consultas é a execução seqüencial de cada consulta submetida por cada usuário emulado.

A seqüência de atualizações são consultas de atualização submetidas seqüencialmente por um programa em lote.

E a sessão é um processo capaz de executar o conjunto de consultas ou de atualizações.

A comunicação entre o sistema sob teste e o programa que irá submeter às consultas deverá considerar uma sessão para cada seqüência de consultas ou atualizações.

A configuração do sistema sob teste deverá ser baseada nas documentações externas de acordo com a carga de trabalho. Esta carga de trabalho é caracterizada por varreduras seqüenciais de grandes volumes de dados, agregações de grandes volumes de dados, junções de múltiplas tabelas e ordenações.

O TPC impõe que o sistema sob teste deverá ser o mesmo para os dois testes, ou seja, após o teste de poder, não poderá haver nenhuma modificação para a execução do teste de produtividade.

Algumas características e parâmetros de configuração que alteram o desempenho do sistema e que os administradores de banco de dados devem conhecer e decidir quando usar são: o conhecimento da carga de trabalho, o conhecimento do modelo lógico e físico do banco de dados, o acesso a documentação do banco de dados e sistema operacional e

nenhum conhecimento do funcionamento interno dos produtos utilizados além dos fornecidos pela documentação externa.

Em ALMEIDA (2004), é descrita a utilização dos *benchmarks* TPC-H e DBT3 utilizando PostgreSQL 7.4.2, em um estudo de viabilidade de um data warehouse em uma plataforma de baixo custo . As consultas analisadas foram: consultas de agregações, sub-consultas aninhadas, múltiplas junções e sub-consultas dentro da clausula *from*.

As operações que tiveram piores resultados foram as sub-consultas dentro de outra sub-consulta, consultas que tinham os operadores *exists* e *not exists*, agregações utilizando visões *in-line* (que são sub-consultas dentro da cláusula *from*) e seleções utilizando data.

O principal problema apresentado para estas consultas em datawarehouse foram seleções por data. Um outro problema detectado foi que o otimizador do Postgres, pois o PostgreSQL não conseguiu gerar um plano de execução adequado portanto algumas consultas que tiveram que ser reescritas . Posteriormente o sistema foi testado utilizando o *benchmark* DBT3.

O TPC utiliza um banco de dados pré-definido, com dados padronizados gerados aleatoriamente. Devido a esta natureza, não é possível utilizá-lo como padrão para medir o desempenho em bancos de dados meteorológicos, como o modelo utilizado neste trabalho. Os dados utilizados neste trabalho além de terem tipos diferentes (alfa-numéricos, cadeia de caracteres, imagens, etc...), são temporais: estão sempre relacionados a uma data no tempo.

Uma possibilidade de testar o desempenho de dados meteorológicos é a divisão dos testes de desempenho em 2 segmentos: testar somente imagens e posteriormente, dados alfa-numéricos. Porém como analisar consultas que misturam dados diferentes? Como preparar uma ferramenta que testa tipos dinâmicos e complexos (como uma bacia hidrográfica, por exemplo)?

O objetivo neste trabalho não é comparar/avaliar sistemas ou medir transações de banco de dados, mas verificar a viabilidade de utilizar o Banco de Dados PostgreSQL para dados meteorológicos, tirando o melhor proveito de suas características de orientação a objeto.

Após a verificação do PostgreSQL e metodologias e problemas com o uso de *benchmarks*, surgiu a necessidade de elaborar uma metodologia para ser utilizada no trabalho. A mesma será apresentada no próximo capítulo.

6. METODOLOGIA

Após a análise das metodologias para avaliação de banco de dados, constatou-se que o uso único do TPC não atenderia as características de banco de dados meteorológicos e que somente a utilização das consultas mais clássicas já utilizadas pelo SIMEPAR restringiriam os resultados do estudo.

O objetivo deste trabalho não é realizar uma análise de desempenho completo, pois a mesma envolveria um estudo mais detalhado da arquitetura do sistema, do banco de dados e suas funções internas, além da avaliação do *hardware*.

Com o intuito de ter um conjunto de consultas representativas para a análise de dados meteorológicos, a escolha das mesmas foi baseada nas consultas utilizadas pelo TPC, consultas clássicas já utilizadas pela instituição e consultas utilizadas por outros trabalhos avaliados (BITTON et al (1983), BORAL e DEWITT (1983) e ALMEIDA(2004)).

A viabilidade de utilizar o SGBD PostgreSQL para dados meteorológicos será analisada baseada inicialmente na habilidade para manejar grandes quantidades de dados com tipos diversificados e que sofrem evolução. Também será verificada a possibilidade de manipular tipos de dados distintos e objetos de tamanhos diferentes com tempo de resposta significativo. Estes fatores foram considerados primordiais para o sistema ser considerado "viável".

A metodologia utilizada neste trabalho está dividida nas seguintes etapas:

1. Instalação do PostgreSQL e preparação do sistema operacional;
2. Determinação de um domínio de aplicação específico;
3. Seleção de dados;
4. Elaboração das consultas teste;
5. Testes e avaliação dos resultados.

6.1. INSTALAÇÃO DO POSTGRESQL E PREPARAÇÃO DO SISTEMA OPERACIONAL

Em uma primeira etapa foi instalado o SGBD PostgreSQL na versão 7.4.2, adaptada à arquitetura Linux. Foram verificados os tipos de dados usuais (numeric, varchar, etc...) e como poderiam ser utilizados para dados meteorológicos.

Posteriormente foi verificado como seria a interação com objetos distintos (tipo OID) e quais funções o sistema apresenta para a sua manipulação (lo_import, lo_export). A interação com o banco de dados foi feita utilizando acesso remoto.

6.2. DETERMINAÇÃO DE UM DOMÍNIO DE APLICAÇÃO ESPECÍFICO

Nesta etapa foi realizada a avaliação do problema e verificação da área de aplicação. O problema era testar a viabilidade de utilizar o PostgreSQL para armazenar dados meteorológicos, tirando proveito de suas características de orientação à objeto.

Com o apoio do SIMEPAR, foi possível limitar o domínio de aplicação aos dados utilizados pela instituição. O SIMEPAR já armazenava os dados em um banco de dados ORACLE (2004) e havia o interesse de verificar a viabilidade de adotar um banco de dados de código aberto. Mesmo limitando os tipos de dados, foi verificado que a variedade dos mesmos é grande. As suas características foram analisadas com o objetivo de selecionar alguns tipos para a utilização no trabalho.

Os dados de raios, por exemplo, não eram interessantes para este estudo pois utilizavam uma tabela única, com dados temporais e uma quantidade de informação extensa, não interagindo muito com outras informações. Além disso, são esporádicos: pode haver 3 dias sem ter raios, mas quando uma tempestade se inicia, podem cair vários em um segundo.

Como a intenção é simular um ambiente real para observar o sistema, a escolha dos tipos de dados era importante pois influenciaria na complexidade das consultas.

O SGBD PostgreSQL foi estudado, junto com suas funções para verificar sua interação com tipos de dados distintos. Também foi verificado o projeto PostGIS (citado anteriormente) e sua interação com o PostgreSQL.

6.3. SELEÇÃO DE DADOS

Dentre os diversos tipos de dados meteorológicos, como dados SYNOP (Surface Synoptic Observations), dados de sensores de raios, dados de radar, dados hidrológicos de usinas hidrelétricas, dados de nível transmitidos via rádio, dados enviados via email, imagens transmitidas via satélite, imagens geradas com programas, etc..., foram selecionados os dados de imagens (formato GIF) de satélite, imagens GRADS, dados de aeroportos (METAR) e dados horários de sensores meteorológicos. Os 4 tipos distintos de dados foram escolhidos com o objetivo de mostrar a diversidade de informação que este tipo de banco de dados pode ter.

Os dados meteorológicos têm importância porque são a fonte de geração de outros produtos, como médias diárias, mensais e anuais, além de serem utilizados para a geração de imagens como GRADS. Por serem transmitidos por sensores, eles estão mais sujeitos a falhas (como uma falha de transmissão ou de sensor), requerendo um controle de qualidade e supervisão específicos.

Os dados de imagens de satélite (formato GIF) são importantes porque a visualização é o meio mais rápido de compreender informações, além de sintetizar os dados facilitando a compreensão.

Os dados do METAR foram selecionados por serem de um tipo distinto: uma cadeia de caracteres sem tamanho fixo com várias combinações de código, com o objetivo de descrever a condição do tempo para a aviação.

Após a seleção dos tipos de dados, surgiu a dúvida: qual proporção de dados utilizar para resultar em uma análise significativa do sistema? A mesma dúvida já foi apresentada em outros contextos, como por exemplo em BITTON et al (1983).

Foi escolhido o período de Janeiro/2004 das cidades do Paraná devido a sua quantidade de informações. Este período foi considerado significativo para uma primeira avaliação de viabilidade do sistema.

Após a seleção dos tipos de dados, foi verificado que não seria necessária a utilização do PostGis, pois o PostgreSQL já supriria a interação com os dados. Os tipos de imagens selecionadas (tipo *raster,* formato GIF), também não precisavam das funções disponibilizadas no PostGis (mais utilizado para o tipo vetorial de imagens). A utilização de extensões do banco de dados acarretaria um custo desnecessário para o sistema.

6.4. ELABORAÇÃO DAS CONSULTAS TESTE

Na elaboração das consultas foi verificado que seria inviável a utilização do TPC, pois ele utiliza um banco de dados pré-definido, com dados gerados aleatoriamente e geralmente com tipos pré-determinados. Não seria possível reproduzir os tipos de dados meteorológicos, além de não poder utilizar a característica temporal de seus dados, onde uma informação está sempre relacionada a uma data no tempo.

Como não há um *benchmark* específico para banco de dados meteorológicos, o TPC foi utilizado como um padrão a ser seguido para avaliar as consultas realizadas em um banco de dados. O método utilizado para simular ambientes mono-usuário e multi-usuário (teste de poder e produtividade, respectivamente) também foram posteriormente utilizados para a avaliação dos resultados.

Após avaliar outros trabalhos que tratam da análise de desempenho de banco de dados (BITTON et al (1983), BORAL e DEWITT (1983) e ALMEIDA(2004)) em conjunto com consultas clássicas já utilizadas para dados meteorológicos, foram selecionadas para avaliação a inserção e seleção de dados.

A operação de inserção trata de imagens, cadeia de caracteres e dados numéricos.

Os objetivos de avaliarmos a inserção de dados são primeiramente verificar o tempo de resposta do banco de dados para manipular imagens com tamanhos diferentes. Em uma segunda etapa o objetivo é verificar quão lento fica o sistema, visto que inserções utilizam uma função (lo_import) que interage com o sistema operacional. Em uma terceira etapa temos como objetivo analisar a estabilidade do sistema com uma carga de dados extensa, além de verificar o comportamento do sistema com tipos de dados diferentes sendo inseridos simultaneamente.

A operação de seleção de dados está dividida em varredura seqüencial, junção de tabelas, agrupamento, ordenação e consultas aleatórias (ad-hoc).

Os objetivos de avaliarmos a seleção de dados são: primeiramente verificar como os objetos diferentes interagem e quanto custa esta interação. Em uma segunda etapa o objetivo é verificar o custo de consultas já realizadas pelo SIMEPAR em um banco de

dados distinto. A terceira etapa baseia-se na avaliação de consultas novas, com interação de imagens.

6.5. TESTES E AVALIAÇÃO DOS RESULTADOS

A partir das consultas selecionadas anteriormente, e baseados na avaliação do TPC (2004) , elegemos os testes de poder (execução de consultas em ordem seqüencial) e produtividade (execução de grupos de consultas em paralelo) para verificar o comportamento do sistema em um ambiente mono-usuário e multi-usuário, respectivamente.

Esta avaliação é importante para analisar qual o custo (em tempo médio) que uma consulta pode ter quando é realizada sozinha ou quando é realizada em conjunto com outras. Este fator é primordial, pois em bancos de dados meteorológicos geralmente vários processos são executados paralelamente, com dados distintos, de fontes diferentes.

Estes testes também servem como uma primeira análise do comportamento do hardware em função das consultas e processamento gerado.

Utilizando os resultados gerados, foi feita a avaliação da viabilidade do sistema, baseados nos fatores citados anteriormente como *primordiais* para o sistema ser considerado viável.

7. O PROBLEMA

O SIMEPAR é uma instituição do governo que trabalha com disseminação de dados meteorológicos.

Os dados meteorológicos são utilizados para geração de alertas como ventos severos, chuvas, queimadas, raios, geadas, granizo, etc..., além de informar a época para plantio, qualidade do ar, previsão de tempo, de secas, controle de nível de água em usinas hidrelétricas e reservatórios, etc...

Estes dados podem ser obtidos de diferentes fontes, como dados de satélites, dados SYNOP, dados de aeroportos, dados enviados via rádio, via email, dados de medição de usinas hidrelétricas, através de cooperação com outras instituições, etc... E também têm freqüência diferentes: transmitidos a cada minuto, a cada 15 minutos, a cada hora, a cada 3 horas, a cada milisegundo (dados de raios, por exemplo).

Os tipos de dados também divergem: imagens vetoriais, imagens *raster*, cadeia de caracteres, dados numéricos, etc, dependendo de sua natureza.

Além disso, há um crescimento exponencial: a cada dia chegam mais dados dos tipos já comportados, além dos novos tipos que surgem e que é necessário comportar.

Quando erros de transmissão ou decodificação ocorrem, os dados chegam ao sistema com um valor incorreto, gerando alarmes falsos e informação incorreta. Para garantir a qualidade da informação gerada, é necessário um controle de qualidade do dado, onde cada valor recebe uma nota de qualidade.

Os eventos meteorológicos não respeitam feriados, fins de semana ou horários, requerendo sistema altamente disponíveis (24 x 7). Mesmo a inserção de dados, que geralmente é feita por um processo automático, precisa ser supervisionada, pois caso um dado primordial pare de transmitir, alguém deve ser acionado para verificar prontamente o problema. Imagine uma tempestade que começa as 23:40 h de uma sexta-feira e a inserção de dados pára por causa de espaço em disco: o sistema já não pode ser utilizado para o objetivo primordial: informar, alertar e prevenir.

Tratando-se de *hardware,* são necessárias máquinas robustas, possibilitando backups e replicação. Com relação ao software, é necessário um banco de dados estável, confiável, tolerante a falhas e com supervisão ininterrupta (checagem de espaço em disco, checagem de dados que possam gerar alertas, verificação de falhas de transmissão de estações ou falhas de sensores, etc). Quanto às aplicações, são necessárias aplicações rápidas e eficientes, pois o volume de informações é extenso.

Apesar de uma estrutura complexa, somente dados numéricos geralmente são guardados no banco de dados, enquanto que outros tipos, como imagens, eram guardados em outras máquinas, utilizando estruturas de diretórios para a navegação. Este método dificulta a busca de informações para o usuário final, pois além de conhecer como funciona o banco de dados, tem que conhecer a localização física dos outros tipos de dados. Este método também dificulta a administração da informação: além de gerenciar o BD, há a necessidade de gerenciar a estrutura em que são guardados os dados restantes.

Como estes servidores tem grande capacidade de memória e alto processamento, podem ter custos exorbitantes: geralmente são de marcas proprietárias, e o preço da garantia e manutenção às vezes pode se tornar um problema.

Como a quantidade de dados cresce de forma exponencial, um servidor deste tipo pode se tornar descartável muito rapidamente.

A instituição utilizava o banco de dados ORACLE para os seus dados, porém havia o interesse de verificar a viabilidade de utilizar estas informações em um sistema alternativo. Este sistema deveria ter boas interações com o SO (através de bibliotecas, API's, etc.), ser robusto, confiável, ter boas opções para aplicações como backup, replicação, mas principalmente interagir satisfatoriamente com os dados e obter tempos de respostas viáveis.

Este trabalho foi iniciado com a revisão da literatura disponível, a verificação de como o PostgreSQL funcionava e como os *benchmarks* poderiam contribuir. Posteriormente foram definidos os fatores de viabilidade para um primeiro estudo do sistema : manipulação de dados meteorológicos com tipos e tamanhos distintos obtendo um tempo de resposta viável utilizando o banco de dados PostgreSQL. Logo foi

estabelecida uma metodologia a seguir, porém faltava a implementação, para obter uma análise inicial do sistema.

Para esta análise, foi determinada como área de abrangência os dados do mês de Janeiro/2004 das cidades do estado do Paraná (Foz do Iguaçú, Londrina, Maringá, Ponta Porã, Curitiba e Cascavel). Os dados são verídicos, obtidos com o apoio do SIMEPAR e INPE. Os dados estão divididos em imagens de satélite, imagens GRADS, dados meteorológicos horários e dados do METAR.

As imagens de satélite são imagens enviadas via satélite GOES, abrangendo parte da América do Sul. As imagens coloridas e preto e branco do satélite GOES do dia 01/01/2004 05:45 estão ilustradas nas figuras 1 e 2:

FIGURA1: IMAGEM PRETA E BRANCO DO DIA 01/01/2004 05:45

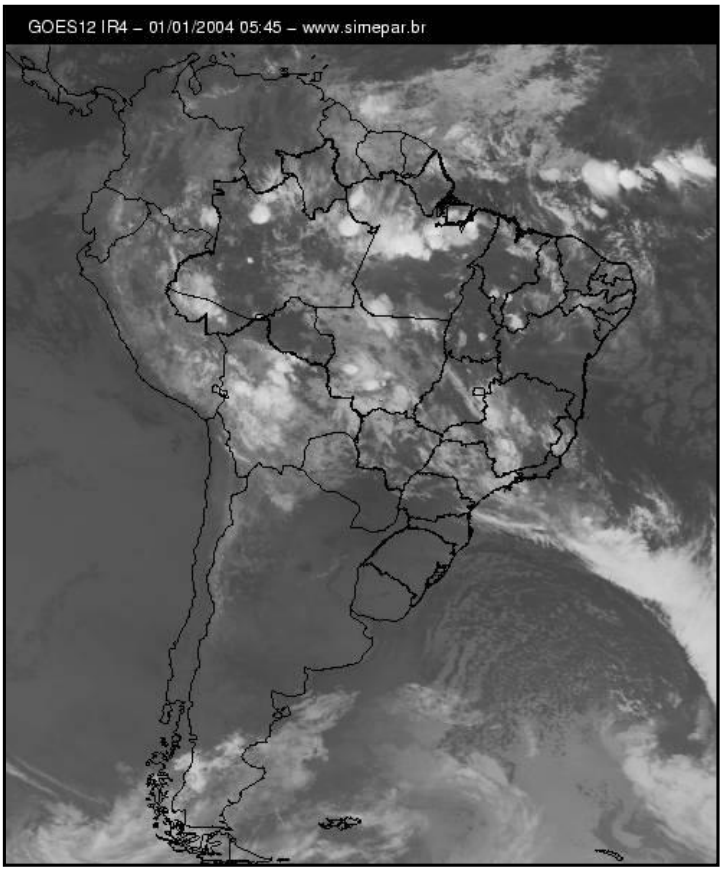
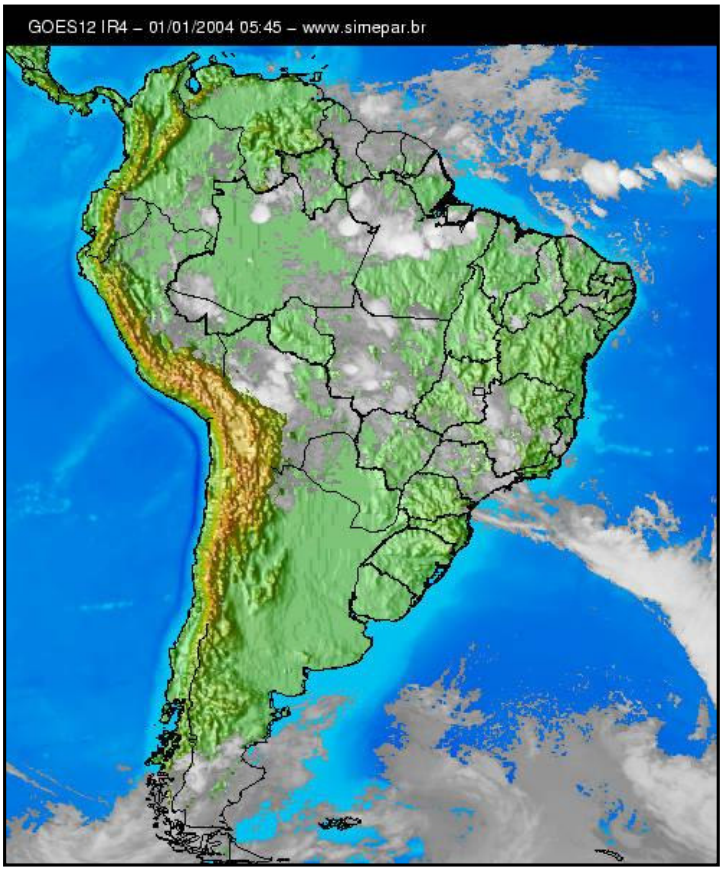


FIGURA2: IMAGEM COLORIDA DO DIA 01/01/2004 05:45

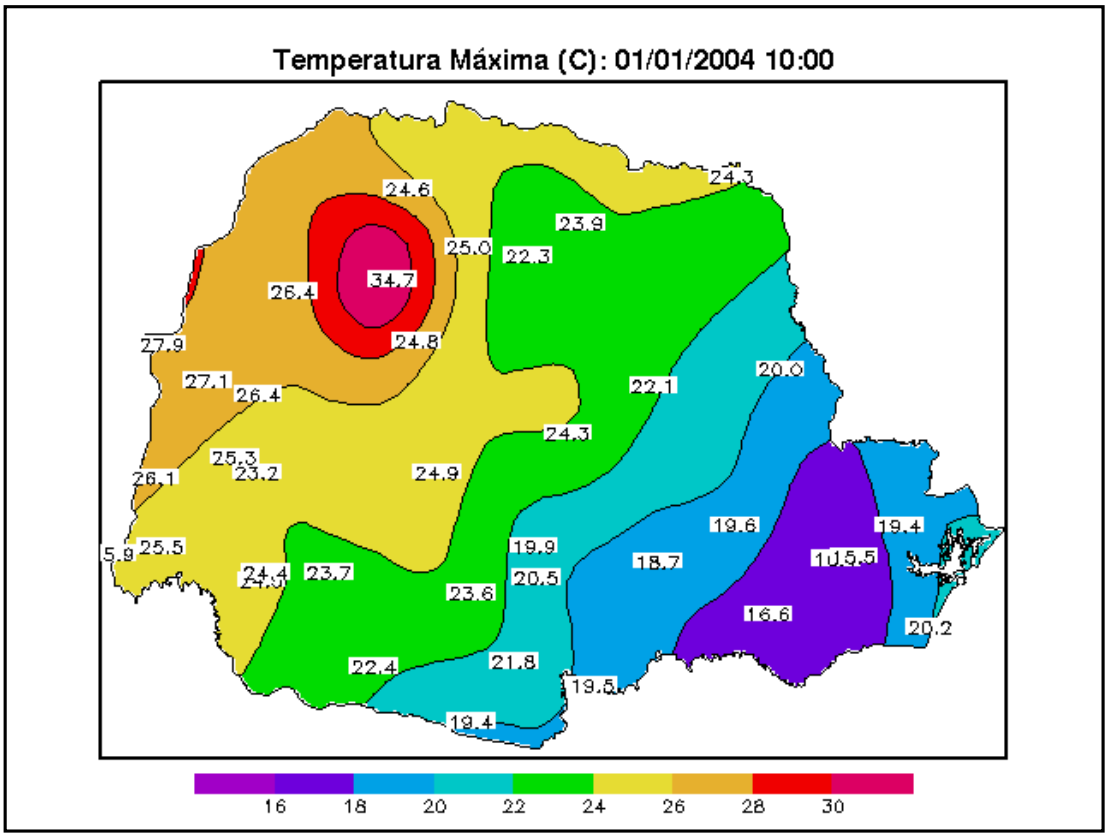


Com estas imagens é possível visualizar a movimentação de nuvens, sendo utilizadas geralmente para previsão de chuva. Neste trabalho utilizamos imagens com o formato GIF, divididas em imagens preto e branco (com aproximadamente 1.3 Mb cada uma) e imagens coloridas (com aproximadamente 70Kb cada uma). As imagens de satélite são transmitidas oito vezes ao dia, resultando em aproximadamente 248 imagens preto e branco e 248 imagens coloridas, transmitidas somente no mês de janeiro de 2004.

As imagens do GRADS são imagens geradas com o auxílio de um programa chamado GRADS (*Grid Analysis and Display System*). São dados plotados em um mapa pré-estabelecido (mapa do Paraná), utilizando dados meteorológicos horários de um sensor. Estas imagens são geradas 24 vezes ao dia para cada um dos 12 sensores: precipitação acumulada, pressão, pressão máxima, pressão mínima, radiação solar, rajada, temperatura média, temperatura máxima, temperatura mínima, temperatura de relva,

umidade relativa e velocidade do vento. São imagens no formato GIF, com aproximadamente 70Kb cada uma. Exemplo: a FIGURA3 ilustra uma imagem de um arquivo com os dados de temperatura máxima das cidades do Paraná do dia 01/01/2004 09:00:00 :

FIGURA3: IMAGEM DE TEMPERATURA MÁXIMA DO DIA 01/04/2004 10:00



Os dados meteorológicos horários são dados enviados via satélite, a partir de sensores. São pacotes de dados transmitidos 24 vezes ao dia para cada uma das estações. As estações telemétricas enviam, em horários estabelecidos, valores de todos os sensores. Estes dados são decodificados e inseridos no banco de dados. Eles têm importância especial pois a partir deles são geradas médias diárias, mensais e anuais, além de servir de fonte de informação para outros produtos, com imagens GRADS.

Com os dados meteorológicos, utilizaremos os sensores de precipitação, radiação solar, temperatura média, temperatura máxima, temperatura mínima e umidade relativa para as cidades de Foz do Iguaçú, Londrina, Maringá, Cascavel e Curitiba. Por exemplo, a estação de Foz do Iguaçú , para o dia de 01/01/2004 00:00:00 teve o valor de 22.1 para

temperatura mínima, 22.3 para temperatura média, 22.4 para temperatura máxima, 98.9 para umidade relativa, 0 para precipitação e 0 para radiação solar.

Os dados do METAR são cadeias de caracteres disponibilizadas diariamente, indicando o tempo nos principais aeroportos. Apesar de terem um padrão descrevendo a condição do tempo, tipo de nuvens, pressão atmosférica, etc., estes dados podem ter até pequenos textos descritivos, dificultando a sua decodificação automática, com distribuição da informação em campos pré-estabelecidos.

Os dados do METAR são dados gerados 24 vezes ao dia e neste trabalho utilizamos as estações de Foz do Iguaçú, Londrina, Ponta Porã e Curitiba. Para exemplificar, abaixo são mostrados os dados do Metar da estação do Aeroporto de Curitiba do dia 01/01/2004 as 10:00:00 :

SBCT 011300Z 12010KT 9999 OVC007 16/14 Q1019

Uma possível interpretação da mensagem acima:

SBCT: Aeroporto Afonso Pena

011300Z: Decodificação do dia 01 – 13hs Zulu

12010KT: vento de 120 graus com 10 nós

9999: visibilidade maior que 10 Km

OVC007: céu incoberto a 700 pés

16/14: 16 graus de temperatura, 14 graus de temperatura ponto de orvalho

Q1019: Pressão Atmosférica de 1019 milibares.

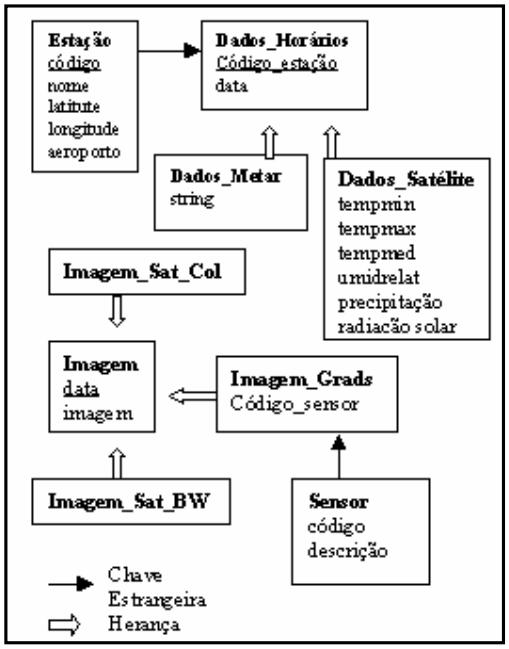
O QUADRO1 resume as características dos dados utilizados.

QUADRO1: CARACTERÍSTICAS DOS DADOS UTILIZADOS

Tipo de Dado	Freqüência	Tipo	Abrangência
Imagem de Satélite	02:45, 05:45, 08:45, 11:45, 14:45, 17:45, 20:45, 23:45 de cada dia	Imagem colorida e preta e branco	América do Sul
Imagem do GRADS	01:00, 02:00, 03:00, 04:00, 05:00, 06:00, 07:00, 08:00, 09:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00, 17:00, 18:00, 19:00, 20:00, 21:00, 22:00, 23:00, 00:00 de cada dia	Imagem colorida	Paraná
Dados Meteorológicos Horários	01:00, 02:00, 03:00, 04:00, 05:00, 06:00, 07:00, 08:00, 09:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00, 17:00, 18:00, 19:00, 20:00, 21:00, 22:00, 23:00, 00:00 de cada dia	Número coletado por sensores	Cidades do Paraná
Dados do Metar	01:00, 02:00, 03:00, 04:00, 05:00, 06:00, 07:00, 08:00, 09:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00, 17:00, 18:00, 19:00, 20:00, 21:00, 22:00, 23:00, 00:00 de cada dia	Cadeia de Caracteres	Aeroportos

Na FIGURA4**,** há uma representação do modelo de dados utilizado neste trabalho.

FIGURA4: REPRESENTAÇÃO DO MODELO DE DADOS UTILIZADO



Observe que a abrangência é distinta (dados meteorológicos abrangem cidades específicas, dados METAR abrangem aeroportos, imagens GRADS abrangem o estado do

Paraná e imagens de satélite são para a América do Sul) e horário dos dados são distintos (imagens de satélite são a cada 3 horas enquanto que os outros dados são horários).

A partir dos tipos de dados selecionados e observando suas características, foi elaborado um modelo de dados (FIGURA4). Note que a herança é utilizada na tabela de imagem (para as tabelas de imagem colorida de satélite, imagem preto e branco de satélite e imagem GRADS) e na tabela de dados horários (para dados meteorológicos e dados do METAR).

Assim, para buscar todas as imagens do banco de dados basta selecionar a tabela Imagem e para selecionar todos os dados horários basta buscar a tabela Dados Horários.

Apesar de ter um banco de dados restrito, com dados para população já selecionados, surge o problema de quais consultas avaliar. Baseado nos trabalhos avaliados e consultas clássicas já realizadas, foram selecionadas para a avaliação as consultas de seleção e inserção de dados.

As consultas de inserção estão divididas em 16 grupos: dados do METAR (grupo 1), dados de sensores meteorológicos (grupo 2), imagens de satélite preto e branco (grupo 3), imagens coloridas de satélite (grupo 4), arquivos para cada um dos sensores disponibilizados no GRADS (grupo 5 até grupo 16).

Identificadores de objetos (OIDS) são utilizados neste trabalho para guardar imagens. Como este tipo de dado é utilizado para guardar qualquer objeto (um arquivo fonte de um programa, uma imagem ou um arquivo de log), não existem muitas funções de interação, devido a sua natureza genérica. A forma de interação com estes objetos utilizada neste trabalho são os comandos de inserção no banco (lo_import) e exportação do banco (lo_export).

Exemplo de inserção de imagem de satélite preto e branco:

```
\lo_import '/home/users/inf/nadia/Mestrado/imagens/200401010245bw.gif';
insert into imagem_satbw( imgdata, imgimagem) values( '01/01/2004
02:45:00',:LASTOID);
```

Exemplo de inserção de imagem de satélite colorida:

```
\lo_import '/home/users/inf/nadia/Mestrado/imagens/200401010245cor.gif';
insert into imagem_satcor( imgdata, imgimagem) values('01/01/2004
02:45:00',:LASTOID);
```

Exemplo de inserção de imagens GRADS dos sensores de precipitação, pressão, pressão máxima, pressão mínima, radiação solar, rajada, temperatura, temperatura máxima, temperatura mínima, temperatura de relva, umidade relativa e velocidade do vento do dia 01/01/2004 00:00:00 :

```
\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/precacum2004010101.gif';
insert into imagem_grads(imgdata, imgsensor, imgimagem) values('01/01/2004
01:00:00',1,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/pressao2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',2,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/pressaomax2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',3,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/pressaomin2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',4,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/radsolar2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',5,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/rajada2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',6,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/temp2004010101.gif';
```

```
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',7,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/tempmax2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',8,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/tempmin2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',9,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/temprelva2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',10,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/umidrelat2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',11,:LASTOID);

\lo_import '/home/users/inf/nadia/Mestrado/2004/01/01/vento2004010101.gif';
insert into imagem_grads(imgdata,imgsensor,imgimagem)values('01/01/2004
01:00:00',12,:LASTOID);
```

A inserção de uma imagem é a relação de uma data da imagem e um número único que as identifica no banco de dados. As consultas utilizadas neste trabalho fazem uso do campo data para relacionar as informações com outra tabela: este é o principal campo utilizado para relacionar os dados meteorológicos. Podemos citar como exemplo a consulta 31: retorna todas as imagens dos dias que tiveram a precipitação maior que 10 milímetros. Nesta consulta é feita a junção da tabela de imagens com a tabela de dados horários, e o campo de data é utilizado para relacionar as duas tabelas.

Os dados de inserção compreendem os dados mostrados no QUADRO1 (verifique que dados cadastrais das estações e dados cadastrais dos sensores não estão incluídos). Foram executados os testes de poder e produtividade. Primeiramente criou-se um script

seqüencial para a carga de todo o BD. Posteriormente o script foi dividido em 16 grupos e a inserção foi feita paralelamente.

As consultas de seleção estão divididas em varredura seqüencial de dados, agrupamento de dados, junções de tabelas, ordenações e consultas *Ad-hoc,* totalizando 31 consultas.

Inicialmente as 31 consultas foram executadas em modo seqüencial, com somente um script (teste de poder), e posteriormente as mesmas consultas foram executadas em paralelo (teste de produtividade). Além de consultas já utilizadas na instituição com dados numéricos, foram acrescentadas consultas que misturavam dados numéricos com imagens.

As consultas [1-7] são acessos seqüenciais de dados (consultas que selecionam todos os dados da tabela). A consulta 4 é um exemplo, no qual são selecionados todos os dados da tabela imagem_grads:

```
select * from imagem_grads;
```

As consultas [8-17] são consultas que agrupam dados. A consulta 16, por exemplo, traz as médias do mês de Janeiro: a temperatura mínima, média, máxima, a soma da precipitação e a média da radiação solar:

```
select min(hortempmin), max(hortempmax), avg(hortempmed),  avg(horumidrelat),
      sum(horprecipitacao), avg(horradsolar)
from horaria_sat
group by to_char(hordatahora,'mm/yyyy');
```

As consultas [18-21] são consultas que fazem junção de tabelas.Na consulta 20 são verificados os valores de precipitação de todas as cidades, além de trazer todas as imagens de GRADS e satélite preto e branco (note que é necessário fazer a junção de tabelas, selecionando o sensor 1, que é o responsável pela precipitação):

```
select estnome,to_char(hordatahora,'yyyy/mm/dd hh24'), horprecipitacao,
        imagem_grads.imgimagem, imagem_satbw.imgimagem
from estacao,horaria_sat, imagem_grads,imagem_satbw
where estcodigo=horestacao
        and to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24')=
        to_char(imagem_satbw.imgdata,'yyyy/mm/dd hh24')
        and to_char(hordatahora,'yyyy/mm/dd hh24')=
        to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24') and imgsensor=1;
```

As consultas [22-23] são consultas que utilizam a clausula *order by* . A consulta 23, por exemplo, tenta ordenar a informação utilizando campos não indexados (nome da estação e valor da precipitação):

```
select estnome, hordatahora, horprecipitacao
from horaria_sat, estacao
where horestacao=estcodigo
      order by estnome, horprecipitacao;
```

As consultas [24-31] são consultas aleatórias (*ad-hoc*). A consulta 29, por exemplo, informa as datas, temperatura média, a imagem GRADS do sensor de temperatura média (sensor 7) e as mensagens do METAR, da cidade de Foz do Iguaçú (estação 25315434). Esta consulta também é um exemplo de consulta que não poderia ser utilizada antes, por causa da mistura de dados numéricos com imagem:

```
select estnome, horaria_sat.hordatahora,hortempmed,  imgimagem,horstring
from estacao,horaria_sat, horaria_metar, imagem_grads
where estcodigo=horaria_sat.horestacao and
        horaria_sat.horestacao=horaria_metar.horestacao  and
        horaria_sat.hordatahora=horaria_metar.hordatahora
        and imgdata=horaria_sat.hordatahora and  imgsensor=7
      and estcodigo=25315434;
```

Na consulta 31 primeiramente são selecionados os dias em que a precipitação é maior que 10 milímetros e logo os resultados são usados para selecionar as imagens deste dia (este tipo de consulta é uma consulta que não estaria disponível em um banco de dados da instituição, por causa da mistura de dados diferentes):

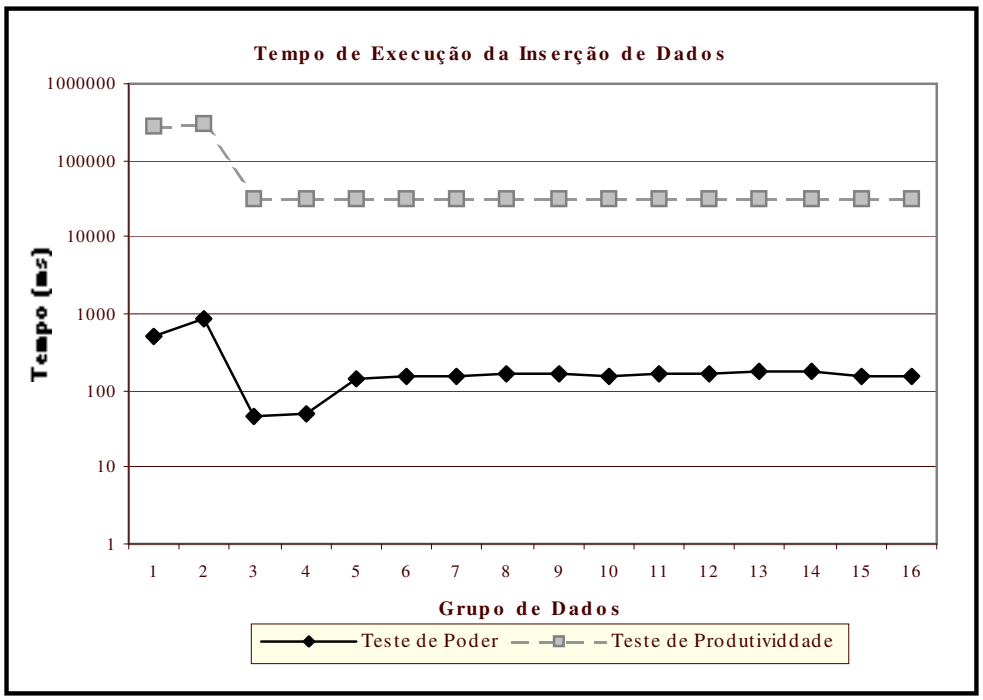
```
select imagem_satbw.imgdata, imagem_satbw.imgimagem, imagem_satcor.imgimagem
from imagem_satcor, imagem_satbw
where to_char(imagem_satbw.imgdata, 'yyyy/mm/dd') in (select distinct
        to_char(hordatahora, 'yyyy/mm/dd') from horaria_sat where horprecipitacao >10)
        and imagem_satbw.imgdata=imagem_satcor.imgdata;
```

A lista completa das consultas podem ser verificadas no anexo deste trabalho.

8. RESULTADOS

A FIGURA5 e o QUADRO2 apresentam os tempos médios de execução da operação de inserção de dados.

FIGURA5: TEMPO DE EXECUÇÃO DAS INSERÇÕES DE DADOS



QUADRO2: TEMPO DE EXECUÇÃO DAS INSERÇÕES DE DADOS

Grupo	Tipo de Dado	Teste de Poder (ms)	Teste de Produtividade (ms)	Número de registros afetados
1	Dados do Metar	523.75	288187.741	2361
2	Dados Horários de Sensores	853.761	307287.749	3720
3	Imagem de Satélite Preto e Branco	46.001	31312.25	248
4	Imagem de Satélite Colorida	47.852	31289.41	248
5	Imagem de Precipitação	144.123	30832.726	713
6	Imagem de Pressão	147.414	30748.544	713
7	Imagem de Pressão Máxima	156.841	30697.042	713
8	Imagem de Pressão Mínima	159.446	30758.092	713
9	Imagem de Radiação Solar	164.627	30626.869	713
10	Imagem de Rajada	153.522	30775.565	713
11	Imagem de Temperatura	164.543	30833.597	713
12	Imagem de Temperatura Máxima	165.783	31022.357	713
13	Imagem de Temperatura Mínima	179.485	30978.758	713
14	Imagem de Temperatura de Relva	176.662	31068.045	713
15	Imagem de Umidade Relativa	154.835	31002.027	713
16	Imagem de Velocidade do Vento	151.127	31060.342	713

O QUADRO3 apresenta o total aproximado de espaço físico ocupado por cada grupo de inserção de dados.

QUADRO3: TOTAL APROXIMADO DE ESPAÇO FÍSICO OCUPADO

Consulta	Tipo de Dado	Total físico ocupado
1	Dados do Metar	1 arquivo de 147KB
2	Dados Horários de Sensores	1 arquivo de 745KB
3	Imagem de Satélite Preto e Branco	1 arquivo de 34KB+ 322.4MB de imagens
4	Imagem de Satélite Colorida	1 arquivo de 34KB+ 17360KB de imagens
5	Imagem Precipitação	1 arquivo de 108KB+ 49910KB de imagens
6	Imagem de Pressão	1 arquivo de 108KB+ 49910KB de imagens
7	Imagem de Pressão Máxima	1 arquivo de 108KB+ 49910KB de imagens
8	Imagem de Pressão Mínima	1 arquivo de 108KB+ 49910KB de imagens
9	Imagem de Radiação Solar	1 arquivo de 108KB+ 49910KB de imagens
10	Imagem de Rajada	1 arquivo de 108KB+ 49910KB de imagens
11	Imagem de Temperatura	1 arquivo de 108KB+ 49910KB de imagens
12	Imagem de Temperatura Máxima	1 arquivo de 108KB+ 49910KB de imagens
13	Imagem de Temperatura Mínima	1 arquivo de 108KB+ 49910KB de imagens
14	Imagem de Temperatura de Relva	1 arquivo de 108KB+ 49910KB de imagens
15	Imagem de Umidade Relativa	1 arquivo de 108KB+ 49910KB de imagens
16	Imagem de Velocidade do Vento	1 arquivo de 108KB+ 49910KB de imagens
	Total aproximado=	926MB de dados

Para o processamento, foram utilizados:

Software:

- PostgreSQL 7.4.2
- Debian Linux

Hardware:

- Athlon modelo 240,
- 3.5GB RAM
- 270GB disk .

A partir do software e hardware anteriormente citados foram executadas as operações de inserção e seleção de dados.

A inserção de dados foi composta por 16 grupos de arquivos (verificar QUADRO3), comportando dados de imagens, strings METAR e sensores meteorológicos. O banco de dados resultante obteve um tamanho aproximado de 1GB.

A operação de inserção de dados foi executada 3 vezes para o teste de poder e 3 vezes para o teste de produtividade e o seu tempo médio de execução é apresentado na Figura 5.

No teste de poder os 16 grupos de dados são inseridos seqüencialmente, utilizando um *script*. No teste de produtividade os grupos de dados são inseridos paralelamente, em 16 processos simultâneos (um processo para cada grupo de inserção).

Com o comando *explain analyze* em NIEDERAUER (2001, p.26), nativo do PostgreSQL, foi gerado um *trace* da inserção de cada registro, possibilitando a avaliação do tempo médio de execução.

Note que o período de 1 mês de dados inicialmente selecionado ocupou o total de espaço físico aproximado de 926MB de informações, incluindo imagens e *scripts* de inserção (verificar QUADRO3).

Com relação a inserção de dados, verificou-se que a inserção de dados numéricos, cadeia de caracteres e imagens tiveram um tempo médio de inserção satisfatório (32.245ms) .

A segunda característica observada é que a diferença entre o tempo de inserção de imagens com tamanhos diferentes é pequena. O grupo 3, por exemplo, tem o tamanho médio de 1.3Mb para cada imagem e obteve um tempo médio de 46ms enquanto o grupo 4 tem o tamanho médio de 70Kb e obteve o tempo médio de 47ms para o teste de poder. A diferença também é pequena para o teste de produtividade: 31.312ms para o grupo 3 e 31.289ms para o grupo 4.

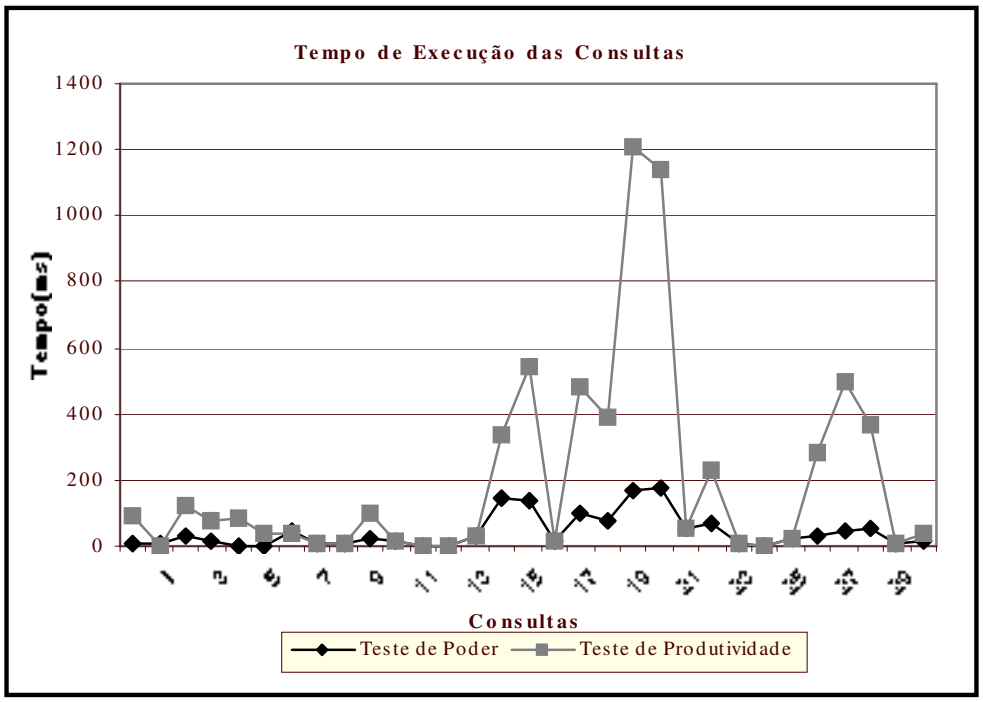
Em geral, os testes de produtividade e poder tiveram um tempo de execução satisfatório (médias gerais de 64.280ms e 211ms).

A terceira característica verificada é que a utilização de uma função que interage com o sistema operacional (lo_import), buscando a imagem e inserindo-a como objeto, não refletiu um problema nos tempos médios de inserção.

E finalmente, o sistema demonstrou-se instável, mesmo o processo de inserção sendo executado de forma remota, com tipos de dados e imagens diferentes, sendo inseridos paralelamente.

A FIGURA6 e o QUADRO4 apresentam o tempo médio de execução das consultas.

FIGURA6: TEMPO DE EXECUÇÃO DAS CONSULTAS



QUADRO4: TEMPO DE EXECUÇÃO DAS CONSULTAS

Consulta	Descrição da Consulta	Teste de Poder (ms)	Teste de Produtividade (ms)	Número de Registros Afetados
1	Acesso Seqüencial de Dados de Sensores	5.551	90.982	2
2	Acesso Seqüencial dos Dados METAR	4.448	3.735	2
3	Acesso Seqüencial de Todos os Dados Horários	32.78	121.763	6
4	Acesso Seqüencial de Imagens GRADS	11.932	79.307	2
5	Acesso Seqüencial de Imagens de Satélite Preto e Branco	0.412	82.618	2
6	Acesso Seqüencial de Imagens de Satélite Coloridas	0.399	40.501	2
7	Acesso Seqüencial de Todas as Imagens	44.038	40.501	7
8	Agrupamento de Dados de Sensores	8.068	8.263	3
9	Agrupamento de Dados METAR	5.034	5.49	3
10	Agrupamento de Todos os Dados Horários	19.545	99.798	6
11	Agrupamento de Imagens GRADS	17.509	17.82	3
12	Agrupamento de Imagens de Satélite Preto e Branco	0.633	0.879	3
13	Agrupamento de Imagens de Satélite Coloridas	0.612	0.86	3
14	Agrupamento de Todas as Imagens	28.44	28.833	7
15	Médias Diárias de Dados de Sensores	146.141	334.695	3
16	Médias Mensais de Dados de Sensores	141.271	544.141	3
17	Agrupamento de registros para cada estação do METAR	12.367	14.096	3
18	Junção de Dados METAR com dados de Sensores	100.042	482.425	10

19	Junção de Dados Horários de Temperatura Média	74.033	393.646	13
20	Junção de Dados de Precipitação com Imagem GRADS e Satélite	169.53	1208.66	18
21	Consulta 20 com Cláusulas Alteradas	172.726	1140.998	18
22	Ordenação de Dados do Sensor de Precipitação	52.368	54.067	7
23	Ordenação de Dados do Sensor de Precipitação com campos não indexados	72.21	231.763	7
24	Seleção de Todos os Dados Horários do Dia '2004-01-13 23:00:00'	10.325	10.712	9
25	Seleção de Todas as Imagens do Dia '2004-01-13 20:00:00'	0.399	0.719	11
26	Seleção de Todos os Dados Horários da Estação 25314910 (Aeroporto de Curitiba)	20.806	21.392	9
27	Seleção de Todas as Imagens em um Período Específico	32.876	283.109	11
28	Seleção de Dados Horários e Imagens em um Período Específico	48.564	495.175	16
29	Seleção de Dados Horários e Imagem de um Sensor e Estação Específicos	53.076	364.395	16
30	Seleção de Dados Horários e Imagem de Satélite	10.432	10.981	13
31	Seleção de Imagens do dia em que a Precipitação é Maior que 10 Milímetros	13.559	41.798	15

A operação de seleção de dados foi composta por 31 consultas, divididas em varredura seqüencial de dados, agrupamento, junção de tabelas e consultas aleatórias (ad-hoc).

A operação de seleção de dados foi executada 3 vezes para o teste de poder e 3 vezes para o teste de produtividade.

No teste de poder as 31 consultas foram executadas em ordem seqüencial, utilizando um *script*. A execução do script gerou um outro arquivo com o tempo médio de execução de cada consulta.

No teste de produtividade as consultas foram executadas em 31 processos paralelos, onde cada processo gerou um arquivo com o tempo médio da consulta executada.

Com relação a seleção de dados, verificou-se que as consultas de dados numéricos, cadeias de caracteres e imagens tiveram tempos de execução satisfatórios: as consultas usuais utilizadas pela instituição não apresentaram problemas quando foram executadas no PostgreSQL .

A segunda característica observada são os resultados das consultas [23-31], que misturam tabelas e tipos de dados diferentes (principalmente imagens com outros tipos de dados). Estas consultas obtiveram um tempo de execução satisfatório: 263ms para o teste de produtividade e 40ms para o teste de poder.

Os testes de produtividade e poder tiveram, de um modo geral, um tempo de execução satisfatório para a maioria das consultas (241ms e 42ms respectivamente).

As consultas 20 e 21 (os dois piores casos de tempo médio) têm o mesmo objetivo: buscar dados de precipitação, imagens GRADS do sensor de precipitação e imagens de satélite preto e branco. As consultas estão apenas com cláusulas alteradas para testar o tempo de execução. O teste de produtividade teve um tempo de resposta considerado alto em comparação a outras consultas (verificar FIGURA6). Segue a consulta 20:

```
select estnome, to_char(hordatahora,'yyyy/mm/dd hh24'),
horprecipitacao, imagem_grads.imgimagem, imagem_satbw.imgimagem
from estacao, horaria_sat, imagem_grads, imagem_satbw
where imgsensor=1 and
estcodigo=horestacao and to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24')=
to_char(imagem_satbw.imgdata,'yyyy/mm/dd hh24') and to_char(hordatahora,
yyyy/mm/dd hh24')= to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24');
```

As consultas que apresentaram piores resultados foram junções de tabelas com imagens (consulta 20, por exemplo) e dados e agrupamentos utilizando hora (consultas 15 e 16, por exemplo).

Mas mesmo as consultas mais demoradas podem ser consideradas com um tempo baixo. Como exemplo podemos citar a consulta 21: obteve o tempo médio de 1208ms ou 1.2 segundos.

Para cada consulta foi executado um *trace*, com o objetivo de avaliar o seu custo e tempo médio de execução. Segue um exemplo do trace executado na consulta 20:

```
Merge Join  (cost=743.44..1140.59 rows=16445 width=37) (actual time=717.611..1134.944 rows=1240 loops=1)
  Merge Cond: (to_char("outer".imgdata, 'yyyy/mm/dd hh24'::text) = "inner"."?column4?")
  ->  Merge Join  (cost=217.08..237.26 rows=885 width=24) (actual time=167.574..173.552 rows=248 loops=1)
        Merge Cond: ("outer"."?column3?" = "inner"."?column3?")
        ->  Sort  (cost=15.34..15.96 rows=248 width=12) (actual time=3.626..3.707 rows=248 loops=1)
              Sort Key: to_char(imagem_satbw.imgdata, 'yyyy/mm/dd hh24'::text)
              ->  Seq Scan on imagem_satbw  (cost=0.00..5.48 rows=248 width=12) (actual time=0.116..2.527 rows=248 loops=1)
        ->  Sort  (cost=201.74..203.52 rows=713 width=12) (actual time=163.728..163.990 rows=713 loops=1)
              Sort Key: to_char(imagem_grads.imgdata, 'yyyy/mm/dd hh24'::text)
              ->  Seq Scan on imagem_grads  (cost=0.00..167.95 rows=713 width=12) (actual time=0.106..160.874 rows=713 loops=1)
                    Filter: (imgsensor = 1::numeric)
  ->  Sort  (cost=526.35..535.65 rows=3720 width=29) (actual time=549.806..551.408 rows=3720 loops=1)
        Sort Key: to_char(horaria_sat.hordatahora, 'yyyy/mm/dd hh24'::text)
        ->  Merge Join  (cost=0.00..305.74 rows=3720 width=29) (actual time=0.223..429.471 rows=3720 loops=1)
              Merge Cond: ("outer".estcodigo = "inner".horestacao)
              ->  Index Scan using pk_estacao on estacao  (cost=0.00..6.05 rows=8 width=25) (actual time=0.028..0.064 rows=8 loops=1)
              ->  Index Scan using pk_horaria_sat on horaria_sat  (cost=0.00..253.17 rows=3720 width=28) (actual time=0.030..12.679 rows=3720 loops=1)
Total runtime: 1208.666 ms
(18 rows)
```

Não obtivemos nenhum problema com o banco de dados para manipulação (inserção, update, seleção, deleção) de dados, mesmo trabalhando com imagens de diferentes tamanhos e grandes quantidades de informações (banco de dados com aproximadamente 1 Gb), sendo acessado de forma remota.

9. CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho buscamos avaliar a viabilidade de utilização de um banco de dados de código aberto - PostgreSQL, para o armazenamento de dados meteorológicos, tirando proveito de suas características de orientação a objeto.

Inicialmente foi feito um estudo do PostgreSQL , sua interação com o sistema operacional e como poderia ser utilizado com tipos de dados distintos. Posteriormente foram verificados *benchmarks* e os problemas para trabalhar com dados temporais e de tipos diferenciados – características primordiais em dados meteorológicos.

Para a implementação foi utilizado o banco de dados PostgreSQL 7.4.2, o sistema operacional Linux (Debian) e dados de imagens, strings e sensores de cidades do Paraná, referentes ao mês de Janeiro de 2004. Foram selecionadas as operações de inserção e seleção de dados para a análise de viabilidade, considerando os testes de poder e produtividade.

Como resultado deste trabalho foi verificado que o SGBD PostgreSQL administrou dados com tipos diversificados e em grande quantidade, não apresentando problemas para a sua manipulação (inserção, seleção, modificação, remoção de dados).

Com relação a inserção de dados, o SGBD apresentou um tempo de resposta viável para o teste de poder e produtividade, simulando um ambiente de mono e multi-usuário e utilizando tipos de dados distintos (dados numéricos, cadeia de caracteres e imagens). Também foi verificado que a inserção de objetos do tipo imagem com tamanhos distintos apresentou um tempo médio de resposta viável.

Com relação a seleção de dados, o SGBD PostgreSQL obteve um tempo de resposta viável para o teste de poder e produtividade, utilizando 31 consultas para simular um ambiente de mono e multi-usuário.

As consultas mais rotineiras (consultas de dados alfa-numéricos, consultas de strings, consultas de imagens e consultas que misturam tipos e tamanhos de dados diferentes) também obtiveram um tempo de execução viável para interação com tipos de dados distintos.

Como conseqüência da análise deste trabalho, verificamos que o PostgreSQL é uma opção de SGBD, possibilitando o gerenciamento de informações com estruturas complexas e tipos diferenciados, gerenciando e centralizando todos os dados em uma única fonte.

Os tipos de dados aqui tratados não requereram nenhuma função ou objeto além das que já existem no PostgreSQL e não foi necessária a utilização da extensão deste banco de dados para interagir com os dados meteorológicos.

Há alguns aspectos que podem ser melhorados, como uma versão do PostgreSQL para trabalhar com clusters. Esta característica possibilitaria o processamento paralelo, facilitando a manutenção de bancos de dados com grande quantidade de informações. Foram encontrados projetos de replicação de dados (Erserver e Postgres-R), porém ao término deste trabalho não encontramos projetos para clusters ou arquitetura de 64 bits.

O segundo aspecto que poderia ser melhorado seria a possibilidade de utilizar um *benchmark* para dados meteorológicos ou que trabalhasse com dados diversificados. A definição de um *benchmark* padrão também facilitaria a apresentação dos resultados, possibilitando a padronização e comparação com outros bancos de dados e sistemas operacionais.

Também foi verificado que existem poucas ferramentas para visualizar , interagir e analisar SGBDOO's, dificultando o uso de herança e OID, por exemplo.

Como exemplos de trabalhos futuros, podem ser citados outros tipos de análise de desempenho, avaliação de outros tipos de consultas, comparação com outros SGBD's e testes em outras arquiteturas e sistemas operacionais. Este trabalho foi baseado em Linux (Debian). A execução de testes em arquitetura de 64 bits ou computadores com multi-processamento poderia ajudar instituições que procuram uma segunda opção na área de SGBD's.

Também poderia ser citada a elaboração de métodos de testes mais completos: por causa do tempo limitado do trabalho, foram testadas somente a inserção e seleção de dados, porém consultas mais complexas poderiam ser testadas posteriormente. Uma outra possibilidade de estudo futuro seriam testes de *triggers* e regras utilizando OO interagindo com objetos de tipos diversificados. Neste trabalho, por exemplo, não foi testada é a

interação do sistema com aplicações, tirando proveito de regras de conhecimento (*triggers* e funções), utilizando orientação a objeto.

E finalmente, um teste do sistema on-line para verificar o seu comportamento: comprovando a robustez do sistema. A integração de outros tipos de dados e o gerenciamento dos recursos poderiam ser verificados. Um exemplo seria: se a *tablespace* com dados de raios pára, integrar a funcionalidade de fazer o isolamento da mesma enquanto o resto do banco de dados opera normalmente.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, E.C. Estudo de Viabilidade de um Datawarehouse em Plataforma de Baixo Custo, Curitiba, 2004. 76f. Dissertação (Mestrado em Informática) – Departamento de Informática, Universidade Federal do Paraná.

ALMIR, Y. et al. Practical Wide-Area Database Replication , CNDS – Jonh's Hopkings University , 2002.

BITTON, D.; DEWITT, J.; TURBYFILL, C. Benchmarking Database Systems A Systematic Approach. Proceedings of the 9th International Conference of Very Large Databases, Florencia, Itália, novembro 1983 .

BORAL, H.; DEWITT, D. A Methodology for Database System Performance Evaluation, Proceedings of the 1984 SIGMOD Conference, June, 1984

ERSERVER. Projeto de Replicação do PostgreSQL Disponível em: http://gborg.postgresql.org/project/erserver

FILHO, J.L.; COSTA, A.C.; IOCHPE, C. Projeto de Banco de Dados Geográficos: mapeando esquemas GeoFrame para o SIG Spring, 1999 Disponível em: http://www.geoinfo.info/geoinfo1999/papers/Jugurta.pdf

GORDON, B.; ORAL, S.; SU, G.; GEORGE, A. Performance Analysis of HP AlphaServer ES80 vs. SAN-based Clusters, Proceedings of 22nd IEEE International Performance, Computing, and Communications Conference (IPCCC), Phoenix, AZ, April 9-11, 2003 Disponível em: http://www.hcs.ufl.edu/pubs/IPCCC2003.pdf

GRADS. Grid Analysis and Display System Disponível em: http://grads.iges.org/grads/grads.html

GRAY, J.; HELLAND, P.; O'NEIL, P.; SASHA, D. The Dangers of Replication and a Solution Proceedings of the ACM SIGMOD'96, Montreal, Canadá, p.173–182 ,1996.

HARA, C.S. Utilização de um Banco de Dados Orientado a Objetos em um Ambiente de Desenvolvimento de Software, Dissertação (Mestrado em Informática) – DCC, Unicamp, 1990

HARA, L.T. Técnicas de Apresentação de dados em Geoprocessamento, 1997 Disponível em http://www.dpi.inpe.br/dpi/teses/lauro/

INPE. Instituto Nacional de Pesquisas Espaciais Disponível em: http://www.inpe.br

JIMENEZ-PERIS, R.; PATINO-MARTINEZ, M.; KEMME,B.; ALONSO,G. Improving the Scalability of Fault-Tolerant Database Clusters, Proc. Of 22nd IEEE Int. Conf. On Distributed Computing Systems Tech. Repport,2000. Disponível em http://citeseer.ist.psu.edu/506566.html

KEMME, B.; ALONSO, G.; Don't Be lazy, be consistent: Postgres-R, a new way to implement Database Replication, Proc. of the 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt, September 2000 Disponível em: http://www.cs.mcgill.ca/~kemme/papers/vldb00.html

KIM, M.J.; ROSSITER, B.N. Evaluation of the Object-Relational DBMS PostgreSQL .I. Administrative Data, Newcastle University. October 1994.

METAR. Dados Meteorológicos de aeroportos Disponível em: http://weather.noaa.gov/weather/metar.shtml Acesso em: 10/jan/2004

MYSQL. Banco de dados Mysql Documentação online disponível em: http://www.mysql.com Acesso em: 10/jun/2004

NAHOURAII, E.; PETRY, F. Object Oriented Databases, IEEE Computer Society , 1991

NIEDERAUER, J.; PostgreSQL- Guia de Consulta Rápida, São Paulo: Novatec Editora LTDA. 2001

OOSTEROM, P.V.; BOS, J.V.D. An Object-Oriented Approach to the Design of Geographic Information Systems, Em Computer & Graphics, número 4, volume 13, 409-418, 1989. 1999

ORACLE. Banco de dados Oracle Disponível em: http://www.oracle.com Acesso em: 10/jun/2004

PEREZ, C.R.; BATISTA, D.C.F. BDGEO: Modelagem, Implementação e Visualização de dados Geográficos, Proceedings of GIS Brasil 97, 1997
Disponível em http://www.di.ufpe.br/~crp/bdgeo.html

PISSINOU, N.; MAKKI, K.; PARK, E.K. Towards the Design and Development of a New Architecture for Geographic Information Systems, Second International Conference on Information and Knowledge Management , Washington, DC, USA, 1993

POSTGIS. Manual Online do PostGIS Disponível em: http://www.postgis.org Acesso em: 01/fev/2004

POSTGRESQL. Banco de dados PostgreSQL. Documentação disponível em: http://www.postgresql.org/files/documentation/ Acesso em: 05/mai/2005

POSTGRES-R. Projeto de Replicação de dados do PostgreSQL. Disponível em: http://gborg.postgresql.org/project/pgreplication Acesso em: 10/jun/2004

RANAKRISHVAM, R.; GEHRKE, J. Database Management Systems, Editora McGrow-Hill, 2003.

SIMEPAR. Sistema Meteorológico do Paraná Disponível em: http://www.simepar.br

STACEY, D. Replication: DB2, Oracle or Sybase?, SIGMOD record, Vol. 24, No. 4. pp 95-101, 1995

STONEBRAKER, M.; KEMNITZ, G.; The Postgres Next Generation DBMS, 1991

TPC. Transaction Protocol Council - TPC Policies, Version 5.5. Disponível em: http://www.tpc.org/ Acesso em: 01/fev/2004

ZHANG, T.; KAMAKRISHNAN, T.Z.; LIVNY, M. Birch: An Efficient Data Clustering Method for Very Large Databases, in ACM Sigmod Intl. Conf. In Management of Data, SIGMOD, Montreal, Quebec, Canada, 1996, pp 103-114.

ANEXOS

1. SCRIPT DE CRIAÇÃO DE BANCO DE DADOS

```
 create database geo;
\c geo;
drop table horaria_metar;
drop table horaria_sat;
drop table horaria;
drop table estacao;
create table estacao(
        estcodigo numeric(8),
        estnome varchar(30),
        estlatitude numeric(4,2),
        estlongitude numeric(4,2),
        estaeroporto varchar(4),
        constraint "pk_estacao" primary key (estcodigo)
);
create table horaria(
        horestacao numeric(8),
        hordatahora timestamp,
        constraint "pk_horaria" primary key(horestacao,hordatahora),
        constraint "fk_horaria_estacao" foreign key (horestacao) references
                estacao(estcodigo)
);
create table horaria_sat(
        hortempmin numeric(4,2),
        hortempmed numeric(4,2),
        hortempmax numeric(4,2),
        horumidrelat numeric(5,2),
```

```
        horprecipitacao numeric(5,2),
        horradsolar numeric(5),
        constraint "pk_horaria_sat" primary key(horestacao,hordatahora),
        constraint "fk_horaria_sat_estacao" foreign key (horestacao) references
                estacao(estcodigo)
) inherits (horaria);
create table horaria_metar(
     horstring varchar(200),
        constraint "pk_horaria_metar" primary key(horestacao,hordatahora),
        constraint "fk_horaria_metar_estacao" foreign key (horestacao) references
                estacao(estcodigo)
)inherits (horaria);
create table imagem(
    imgdata timestamp,
    imgimagem OID,
    constraint "pk_imagem" primary key(imgdata)
);
create table imagem_satbw(
    constraint "pk_imagem_satbw" primary key(imgdata)
)inherits (imagem);
create table imagem_satcor(
    constraint "pk_imagem_satcor" primary key(imgdata)
)inherits (imagem);
create table sensor(
    snscodigo numeric(2),
    snsdescricao varchar(30),
    constraint "pk_sensor" primary key (snscodigo)
);
create table imagem_grads(
    imgsensor numeric(2),
```

```
    constraint "pk_imagem_grads" primary key(imgdata,imgsensor),
    constraint "fk_imagem_grads_sensor"  foreign key (imgsensor) references sensor(snscodigo)

)inherits (imagem);

insert into estacao(estcodigo,estnome,estlatitude,estlongitude,  estaeroporto)values(25315434,
    'FOZ DO IGUACU',-25.52,-54.58,'SBFI');
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values(23195107,'LONDRINA'    ,-23.33,-51.13,'SBLO');
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values(23255157,'MARINGA'   ,-23.42,-51.95,'SBMG') ;
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values( 25314831;'PARANAGUA'   ;-25.52;-48.52;'SBPG');
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values( 22335542,'PONTA PORA'  ,-22.55,-55.7,'SBPP');
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values( 25154909,'CURITIBA'   ,-25.26,-49.16,'SBBI');
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values( 24575325,'CASCAVEL'   ,-24.95,-53.43,'SBCA');
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values( 22264317,'PARANAGUA',-22.45,-43.28,'SBPI');
insert into estacao(estcodigo,estnome,estlatitude,estlongitude,
    estaeroporto)values( 25314910,'CURITIBA AEROPO.'  ,-25.52,-49.17,'SBCT');

insert into sensor(snscodigo,snsdescricao)values(1,'Precipitacao Acumulada');
insert into sensor(snscodigo,snsdescricao)values(2 ,'Pressao Atmosferica Media');
insert into sensor(snscodigo,snsdescricao)values(3,'Pressao Atmosferica Maxima');
insert into sensor(snscodigo,snsdescricao)values(4,'Pressao Atmosferica Minima');
insert into sensor(snscodigo,snsdescricao)values(5,'Radiacao Solar');
insert into sensor(snscodigo,snsdescricao)values(6,'Rajada');
```

```
insert into sensor(snscodigo,snsdescricao)values(7,'Temperatura Media');

insert into sensor(snscodigo,snsdescricao)values(8,'Temperatura Maxima');

insert into sensor(snscodigo,snsdescricao)values(9,'Temperatura Minima');

insert into sensor(snscodigo,snsdescricao)values(10,'Temperatura de Relva');

insert into sensor(snscodigo,snsdescricao)values(11,'Umidade Relativa');

insert into sensor(snscodigo,snsdescricao)values(12,'Velocidade do Vento');
```

2. RELAÇÃO DAS CONSULTAS EXECUTADAS

1) explain analyze select * from horaria_sat;

2) explain analyze select * from horaria_metar;

3)explain analyze select * from horaria;

4)explain analyze select * from imagem_grads;

5)explain analyze select * from imagem_satbw;

6)explain analyze select * from imagem_satcor;

7)explain analyze select * from imagem;

8)explain analyze select count(*) from horaria_sat;

9)explain analyze select count(*) from horaria_metar;

10)explain analyze select count(*) from horaria;

11)explain analyze select count(*) from imagem_grads;

12)explain analyze select count(*) from imagem_satbw;

13)explain analyze select count(*) from imagem_satcor;

14)explain analyze select count(*) from imagem;

15)explain analyze select min(hortempmin), max(hortempmax), avg(hortempmed),

avg(horumidrelat),sum(horprecipitacao),avg(horradsolar)

from horaria_sat

group by to_char(hordatahora,'dd/mm/yyyy');

16)explain analyze select min(hortempmin), max(hortempmax), avg(hortempmed), avg(horumidrelat), sum(horprecipitacao), avg(horradsolar)

from horaria_sat

group by to_char(hordatahora,'mm/yyyy');

17)explain analyze select horestacao, count(*) from horaria_metar group by horestacao;

18)explain analyze select estnome,horaria_sat.hordatahora,hortempmed,horstring

from horaria_sat, horaria_metar, estacao

where horaria_sat.horestacao=horaria_metar.horestacao and horaria_sat.horestacao=estcodigo and horaria_sat.hordatahora=horaria_metar.hordatahora;

19) explain analyze select estnome, hordatahora, hortempmed, imgimagem

from estacao,horaria_sat, imagem_grads

where horestacao=estcodigo and hordatahora=imgdata and imgsensor=7;

20)explain analyze select estnome,to_char(hordatahora,'yyyy/mm/dd hh24'), horprecipitacao,

imagem_grads.imgimagem, imagem_satbw.imgimagem from estacao,horaria_sat, imagem_grads,imagem_satbw where estcodigo=horestacao and to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24')=

to_char(imagem_satbw.imgdata,'yyyy/mm/dd hh24') and to_char(hordatahora,'yyyy/mm/dd hh24')=

to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24') and imgsensor=1;

21) explain analyze select estnome, to_char(hordatahora,'yyyy/mm/dd hh24'),

horprecipitacao, imagem_grads.imgimagem, imagem_satbw.imgimagem

from estacao,horaria_sat, imagem_grads, imagem_satbw

```
where imgsensor=1 and estcodigo=horestacao and to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24')=
to_char(imagem_satbw.imgdata,'yyyy/mm/dd hh24') and to_char(hordatahora,
yyyy/mm/dd hh24')= to_char(imagem_grads.imgdata,'yyyy/mm/dd hh24');
```

22)
```
explain analyze select estnome, hordatahora, horprecipitacao
from horaria_sat, estacao
where horestacao=estcodigo
order by estnome, hordatahora;
```

23)
```
explain analyze select estnome, hordatahora, horprecipitacao
 from horaria_sat, estacao
where horestacao=estcodigo
order by estnome, horprecipitacao;
```

24)
```
explain analyze select * from horaria where hordatahora='2004-01-13 23:00:00';
```

25)
```
explain analyze select * from imagem where imgdata='2004-01-13 20:00:00';
```

26)
```
explain analyze select * from horaria where horestacao=25314910;
```

27)
```
explain analyze select * from imagem where imgdata between '2004-01-01 00:00:00'
and '2004-01-12 04:00:00';
```

28)
```
explain analyze select * from horaria,imagem
where to_char(hordatahora, 'yyyy/mm/dd')= to_char(imgdata,' yyyy/mm/dd')
and  hordatahora between '2004-01-10' and '2004-01-13';
```

29)
```
explain analyze select estnome, horaria_sat.hordatahora,hortempmed,  imgimagem,horstring
 from estacao,horaria_sat, horaria_metar, imagem_grads
```

```
where estcodigo=horaria_sat.horestacao and   horaria_sat.horestacao=horaria_metar.horestacao  and
horaria_sat.hordatahora=horaria_metar.hordatahora
and imgdata=horaria_sat.hordatahora and  imgsensor=7   and estcodigo=25315434;
```

30)

```
explain analyze select estnome, hordatahora, horprecipitacao,  imagem_satbw.imgimagem
 from estacao,horaria_sat,imagem_satbw
where estcodigo=horestacao  and to_char(hordatahora, 'yyyy/mm/dd)=to_char(imgdata,'yyyy/mm/dd');
```

31)

```
explain analyze select imagem_satbw.imgdata, imagem_satbw.imgimagem,
imagem_satcor.imgimagem
from imagem_satcor, imagem_satbw
where to_char(imagem_satbw.imgdata, 'yyyy/mm/dd') in (select distinct
to_char(hordatahora, 'yyyy/mm/dd') from horaria_sat where horprecipitacao >10)
and imagem_satbw.imgdata=imagem_satcor.imgdata;
```