FACULDADE DE TECNOLOGIA DE GUARATINGUETÁ

MySQL x *Firebird*
Uma Análise Comparativa de Desempenho

Sylvio Villas Boas Neto

Monografia apresentada à Faculdade de Tecnologia de Guaratinguetá, para graduação no Curso Superior de Tecnologia em Informática com Ênfase em Banco de Dados

Guaratinguetá - SP
2011

FACULDADE DE TECNOLOGIA DE GUARATINGUETÁ

MySQL x Firebird
Uma Análise Comparativa de Desempenho

Sylvio Villas Boas Neto

Monografia apresentada à Faculdade de Tecnologia de Guaratinguetá, para graduação no Curso Superior de Tecnologia em Informática com Ênfase em Banco de Dados

Área de Concentração: Informática
Orientadora: Profª Esp. Cilmara Aparecida Ribeiro

Guaratinguetá - SP
2011

Neto, S V. B. ***MYSQL x FIREBIRD*** **- Uma Análise Comparativa de Desempenho**. Guaratinguetá, 2011. 66p. Monografia, Faculdade de Tecnologia de Guaratinguetá.

Dedico este trabalho ao Senhor dos Exércitos, o Deus de Israel, que através
do seu Filho Jesus tem me ajudado de muito perto.
Sem a Sua presença diária e constante em minha vida,
certamente eu já teria desfalecido.

Agradecimentos

A minha mãe, Srª **Carmen Lúcia Villas Boas**, por estar ao meu lado em minha vida.

A amiga **Lúcia Helena Ribas Machado**, pelos valiosos conselhos e incentivos.

Ao amigo **José Roberto Rozante**, por ter observado o meu potencial e acreditado em mim, quando nem mesmo eu acreditava.

Ao amigo **Marcus Jorge Bottino**, pelos inúmeros ensinamentos na área de informática.

A professora **Cilmara Aparecida Ribeiro**, pela preciosa orientação e ajuda neste trabalho.

A amiga **Demylha Arneiro**, pela sua presteza e notável paciência.

“Se te mostrares fraco no dia da angústia, é que a tua força é pequena”.
(Provérbios 24:10)

> “Gostaria de ver a ciência da computação sendo ensinada deliberadamente com uma perspectiva histórica...
> Os alunos precisam entender como chegamos à situação atual, o que foi experimentado, o que funcionou e o que não funcionou, e também como as melhorias no *hardware* permitiram o progresso. A presença desse elemento em seu treinamento faz com que as pessoas enfrentem cada problema a partir dos princípios básicos. Elas estão aptas a oferecer soluções que foram desejadas no passado. Ao invés de apoiar nos ombros de seus precursores, elas tentam ter êxito sozinhas”.
>
> Date (2004, p. 03, apud Maurice V. Wilkes)

Neto, S V. B. ***MYSQL x FIREBIRD* - Uma Análise Comparativa de Desempenho**. Guaratinguetá, 2011. 66p. Monografia, Faculdade de Tecnologia de Guaratinguetá.

Resumo

A avaliação de desempenho dos Sistemas Gerenciadores de Bancos de Dados é de grande importância no mercado de desenvolvimento de sistemas, tanto na função de orientar usuários em suas escolhas como em auxiliar os desenvolvedores dos SGBDs na identificação das falhas para as melhorias em versões futuras. Este estudo apresentou uma análise comparativa de desempenhos entre os Sistemas Gerenciadores de Banco de Dados *MySQL* (5.1) e o *Firebird* (2.5) utilizando a plataforma *Windows XP Service Pack* 3. Em cenário mono-usuário foram realizados testes de comandos DML em uma base de 1000 e 1000 000 de registros. A métrica de avaliação foi a comparação do tempo de resposta que cada SGBD obteve para os comandos DML. Os resultados apresentados através das tabelas e gráficos direcionaram o *Firebird* para aplicações com um volume mais elevado de dados e com uma maior intensidade de operações de entrada e saída. O *MySQL* foi direcionado para aplicações mais ágeis, como *websites* e formulários que não necessitam de um intenso processamento da informação no banco, e também para aplicações que não requerem bases de dados com elevado volume de registros. Pesquisas correlatas apresentaram resultados semelhantes, fato este que evidencia uma tendência de comportamento dos servidores. Uma evidenciada característica do servidor *MySQL* é sua excessiva lentidão durante a inserção de muitos registros utilizando um procedimento armazenado, fato este que deve ser observado antes de uma possibilidade de implantação deste servidor. Para esta pesquisa foi implementado um programa capaz de manipular os registros em ambos os servidores, sendo este denominado *benchmark*, valendo se de conceitos apropriados para a devida avaliação. Futuramente pretende se aprimorar o *benchmark* adicionando rotinas para avaliar outros Sistemas Gerenciadores de Banco de Dados, comandos *DDL*, manipulação de objetos, entre outros.

Palavras-chave: Avaliação de Desempenho, Sistemas Gerenciadores de Bancos de Dados, *Benchmarks*.

Neto, S. V. B. ***MYSQL x FIREBIRD* - A Comparative Performance Analysis**. Guaratinguetá, 2011. 66p. Monograph, Technology University of Guaratinguetá.

Abstract

The performance evaluation of Database Management Systems has great importance both in orient users on their choices as helping developers on fail identification to provide improvements on future versions of the software. This research presented a comparative performance analysis between the database management systems MySQL (5.1) and Firebird (2.5) using Operational System Windows XP Service Pack 3. It was executed tests in mono user environment with Data Manipulation Language commands on bases of 1000 and 1000 000 of registers. The evaluation metric was the comparison of response time that each DBMS returned to DML commands. The results presented through tables and graphics directed Firebird to applications with a larger volume of data and greater intensity of input and output operations. The MySLQ was directed for agiler applications such as websites and forms that don't require a intense information processing on database, and also to applications that don't require bases with larger volume of data. Related searches presented similar results, this fact evidences a behavior trend of the servers. A noted feature from MySQL server is a excessive slow during the insert of a lot of data using stored procedures. This fact must be observed before a possible implantation of this server. For this search, it was implemented a program that manipulated the data at both servers, that is a benchmark, using appropriate concepts for a right evaluation. In the future, the benchmark will be improved, adding routines to evaluate others Database Management Systems, to use DDL comands, objects manipulating and others.

Key-Words: Performance evaluation, Database Management Systems, *Benchmarks*.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

Durante toda a história da humanidade observa-se a necessidade de se armazenar informações de forma a poderem ser acessadas por outras pessoas, fato já notável na pré história onde os habitantes da época procuraram marcar em pedras e cavernas os registros que julgavam relevantes. Após a invenção da escrita no antigo Egito (aproximadamente em 4000 a.C.) e sua popularização no mundo, as informações puderam ser gravadas de forma mais organizada, podendo assim exprimir certa consistência, ainda que pequena, mas fundamental para a sua utilização por outros que acessavam aqueles documentos. A exemplo disso pode se citar os manuscritos religiosos e os pergaminhos relativos às Leis e costumes de diversos povos, em que informações, que inclusive são importantes até nos dias de hoje, puderam ser registradas e utilizadas.

O conceito de armazenar a informação é, sem dúvida, o fundamento principal e primordial de uma base de dados, ou, comumente chamada de banco de dados. Segundo Date (2004, p. 26) um banco de dados "é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa". Logo um banco de dados deve, além de armazenar a coleção de dados importantes, ser capaz de manter a segurança e integridade dos mesmos e também oferecer facilidade no gerenciamento dos seus registros.

Com o advento da informática, os dados puderam ser registrados e acessados com maior velocidade, e o gerenciamento do banco passou a ser feito por *softwares* chamados Sistemas Gerenciadores de Bancos de Dados. Dentre os mais conhecidos pode se citar *MySQL, Firebird, Oracle, Postgresql* entre outros.

Uma questão pertinente especialmente entre usuários de SGBDs é o desempenho apresentado pelos mesmos, levando se em consideração questões como integridade, segurança, velocidade em operações, ferramentas oferecidas, entre outros fatores.

Tendo por objetivo sanar algumas questões e contemplando o aumento contínuo de usuários de SGBDs, o presente trabalho visa expor uma análise comparativa de desempenho entre os SGBDs *Mysql 5.1 e Firebird 2.5* no Sistema Operacional *Windows XP* (*Service Pack 3*) com intuito de eliminar as informações imprecisas e apontar as características mais relevantes de cada sistema.

Como método para essa avaliação, foi implementado um *software* capaz de executar operações de manipulações de dados para duas diferentes condições de armazenagem e então medir o tempo de resposta do sistema para cada operação. O programa foi implementado utilizando a linguagem *Java SE* versão 6 na plataforma *Netbeans (6.0).*

O trabalho apontou (inclusive graficamente) o desempenho apresentado pelos SGBDs submetidos aos testes, e através deste resultado pôde se indicar a melhor aplicabilidade de cada SGBD.

Este trabalho está organizado como segue. O Capitulo 1 apresenta os conceitos de um SGBD. O Capitulo 2 apresenta o SGBD *MySQL* e suas principais características. O Capítulo 3 expõe o SGBD *Firebird* juntamente com suas funcionalidades. O Capítulo 4 descreve a avaliação de desempenho e a metodologia aplicada. Por fim, no capitulo 5 são divulgados os resultados obtidos no experimento juntamente com a discussão. A Conclusão encerra o trabalho apontando a melhor aplicabilidade para cada SGBD.

1 SISTEMAS GERENCIADORES DE BANCOS DE DADOS

Os Sistemas Gerenciadores de Bancos de Dados surgiram em meados da década de 60 tendo por finalidade organizar as informações armazenadas no computador. Entretanto essa organização de dados não era eficaz, pois os SGBDs utilizavam uma Modelagem de Dados Hierárquica que comprometia o seu uso para relacionar estruturas que existem no mundo real. Já em meados dos anos 70 surgiu a Modelagem Relacional, que atendia a necessidade de expor as relações de estruturas. Mas o que é um Modelo de Dados ?

> "Apoiando a estrutura de um banco de dados está o modelo de dados: uma coleção de ferramentas conceituais para descrever dados, relações de dados, semântica de dados e restrições de consistências". (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 5).

1.1 Modelo Hierárquico

O modelo de dados hierárquico ou HDM tem como característica conectar os registros numa estrutura de dados em árvore, ou seja, as ligações são feitas de tal modo que cada registro tenha apenas um proprietário. Este modelo foi largamente usado nos primeiros sistemas de bancos de dados, porem seu uso foi logo abandonado devido a sua restrição em representar alguns tipos de relacionamentos, como o relacionamento muitos para muitos. Atualmente alguns *data centers* com alto volume de dados ainda utilizam este modelo, pois existe uma inviabilidade de migração devido ao elevado custo.

1.2 Modelo de Redes

O Modelo de dados em Redes ou NDS é uma extensão do modelo hierárquico, entretanto este modelo permite a criação de mais de uma relação pai filho, e também uma relação entre seus elementos. Contudo este modelo apresenta grande parte dos problemas e restrições do modelo hierárquico no que diz respeito à modelagem dos dados, evidenciando assim a necessidade de um modelo que permitisse uma modelagem de dados sem a necessidade de uma reconstrução estrutural para suportar uma possível alteração.

1.3 Modelo Relacional

O modelo de dados relacional ou RDM utiliza tabelas (linha x coluna) para representar os dados e propriamente as relações entre tais dados. Uma tabela pode ter várias colunas, sendo cada coluna com um nome único. Uma linha em uma tabela representa o registro e propriamente a relação entre um conjunto de valores. De uma forma geral uma tabela é um conjunto de entidades, sendo cada linha uma entidade dessa relação. A palavra relação é definida como sendo um subconjunto de um produto cartesiano de uma lista de domínios. A diferença entre tabela e relação é que no caso das tabelas é atribuído um nome ao atributo e no caso da relação estes nomes são propriamente números. Sendo assim, é perfeitamente aceitável o termo usado por diversos autores na denominação de tabela por relação, e de linha para tupla.

> Na terminologia do modelo relacional formal, uma linha é chamada tupla, um cabeçalho de coluna é conhecido como atributo, e a tabela é chamada relação. O tipo de dado que descreve os tipos de valores que podem aparecer em cada coluna é representado pelo domínio de valores possíveis. Definimos, agora, esses termos – domínio, tupla, atributo e relação – mais precisamente. (ELMASRI; NAVATHE, 2005, p. 90.)

O modelo relacional foi criado por Edgar Frank Codd em 1970 descrito no artigo *Relational Model of Data for Large Shared Data Banks* (Modelo de Dados Relacional para Grandes Bancos de Dados Compartilhados) publicado pela revista ACM. Edgar Frank Codd ainda publicou um artigo que define 13 regras para que um SGBD seja relacional, são elas:

- 1ª Regra – Regra Fundamental: Um SGBD relacional deve gerenciar seus dados usando apenas suas capacidades relacionais.
- 2ª Regra – Regra da Informação: Toda informação deve ser representada de uma única forma, como os dados em uma tabela.
- 3ª Regra - Regra da Garantia de acesso: Todo dado pode ser acessado logicamente usando o nome da tabela, o valor da chave primária da linha e o nome da coluna.
- 4ª Regra - Tratamento Sistemático de Valores Nulos: Os valores nulos existem, de forma a representar dados não existentes de forma sistemática e independente do tipo de dados.
- 5ª Regra - Catálogo Dinâmico on-line baseado no modelo relacional: A descrição do banco de dados é representada no nível lógico como dados ordinários(isso é, em tabelas), permitindo que usuários autorizados apliquem as mesmas forma de manipular dados aplicada aos dados comuns ao consultá-las.
- 6ª Regra - Da sub-linguagem compreensiva: Um sistema relacional pode suportar várias linguagens e formas de uso, porém deve possuir ao menos uma linguagem com sintaxe bem definida e expressa por cadeia de caracteres e com habilidade de apoiar a definição de dados, a definição de visões, a manipulação de dados, as restrições de integração, a autorização e a fronteira de transações.

- 7ª Regra – Da atualização de visões: Toda visão que for teoricamente atualizável será também atualizável pelo sistema.
- 8ª Regra – Inserção, atualização e eliminação de alto nível: A capacidade de manipular a relação base ou relações derivadas como um operador único não se aplica apenas a recuperação de dados, mas também a inserção, alteração e eliminação de dados.
- 9ª Regra – Independência dos dados físicos: Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as modificações na representação de armazenagem ou métodos de acesso internos.
- 10ª Regra - Independência lógica de dados: Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as mudanças de informação que permitam teoricamente a não alteração das tabelas base.
- 11ª Regra – Independência de integridade: As relações de integridade específicas de um banco de dados relacional devem ser definidas em uma sub-linguagem de dados e armazenadas no catálogo (e não em programas).
- 12ª Regra – Independência de distribuição: A linguagem de manipulação de dados deve possibilitar que as aplicações permaneçam inalteradas estejam os dados centralizados ou distribuídos fisicamente.
- 13ª Regra – Da não-subversão: Se o sistema relacional possui uma linguagem de baixo nível (um registro por vez), não deve ser possível subverter ou ignorar as regras de integridade e restrições definidas no alto nível (muitos registros por vez).

1.3.1 Álgebra Relacional

A álgebra relacional pode ser definida como a linguagem de consulta formal ao modelo de dados relacional, ou seja, é uma forma de cálculo sobre as relações.

> "....um modelo de dados inclui um conjunto de operações para manipular o banco de dados, além dos conceitos de modelo de dados para a definição das restrições e estrutura do banco de dados. O conjunto básico de operações para o modelo relacional é a álgebra relacional." (ELMASRI; NAVATHE, 2005, p. 106.)

Basicamente as operações de Álgebra Relacional podem ser divididas em dois grupos, sendo o primeiro grupo contendo as operações da Teoria de Conjuntos da Matemática. Essas operações podem ser definidas como União, Interseção, Diferença de Conjunto e Produto Cartesiano. Note que tais operações são aplicadas porque cada relação é considerada um conjunto de tuplas definido no Modelo Relacional. O segundo grupo inclui operações especificas para bancos de dados relacionais, sendo estas: Seleção, Projeção, Junção Natural, Renomear, entre outras.

Como um Banco de Dados para ser relacional deve utilizar o Modelo Relacional, e um Modelo Relacional utiliza Álgebra Relacional em suas operações, é considerável a importância da Álgebra Relacional tanto na formalização das operações do Modelo Relacional como na implementação e otimização de consultas em Sistemas Gerenciadores de Bancos de Dados Relacionais. Inclusive muitos conceitos da Álgebra Relacional são incorporados na linguagem padrão SQL dos Sistemas Gerenciadores de Bancos de Dados Relacionais.

1.4 Modelo de Entidade / Relacionamento

O Modelo de Entidade – Relacionamento pode ser definido como um modelo abstrato que captura a estrutura organizacional dos dados independentemente de qualquer SGBD.

> "O Modelo de entidade / Relacionamento (E-R) é baseado em uma percepção de um mundo real que consiste em uma coleção de objetos básicos, chamados *entidades,* e as relações entre esses objetos. Uma entidade é uma "coisa" ou "objeto" no mundo real que é distinguível dos outros objetos. O modelo de entidade / relacionamento é muito usado no projeto de banco de dados..." (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 5).

Assim, é possível perceber que as Entidades são os objetos, os Atributos são as características destes objetos e Relacionamento é a relação entre tais objetos.
A finalidade é descrever de maneira conceitual os dados utilizados em um Sistema de Informação a fim de prover um melhor gerenciamento dos mesmos. A principal ferramenta de um MER é um DER. Um DER é basicamente uma representação gráfica de toda a estrutura lógica proveniente da análise de um MER, onde é possível identificar como os dados do mundo real serão armazenados. É possível identificar três tipos de relações utilizadas em um DER:

- Relação 1..1 (Um para Um). Indica que as tabelas possuem relação unívoca entre si, não importando qual tabela vai receber a chave estrangeira.
- Relação 1..N (Um para Muitos). É obrigatório que a chave primária da tabela do lado 1 seja uma chave estrangeira na tabela do lado N.
- Relação N..N (Muito para Muitos). Quando duas tabelas têm relação N..N é necessária a criação de uma nova tabela que vai ter sua chave primária composta pelas chaves estrangeiras da primeira e segunda tabela respectivamente.

1.5 Arquitetura

A arquitetura de um SGBD deve ser capaz de fornecer um funcionamento no mínimo satisfatório no que diz respeito aos domínios de aplicações de seus usuários. Historicamente falando, as primeiras arquiteturas utilizavam de *mainframes* para o processamento de todas as suas funções, incluindo aplicativos, programas entre outros. O processamento era centralizado e o acesso era feito remotamente, através de terminais que apenas exibiam a informação processada. Com o passar dos anos algumas arquiteturas foram implementadas, sendo:

- **Arquitetura de Plataforma Centralizada** – essa é exatamente a arquitetura que utiliza de um *mainframe* central para o processamento dos dados. Nessa arquitetura um computador com grande poder de processamento serve como hospedeiro para o SGBD, e então os usuários tem acesso apenas a visualização dos dados através de terminais remotos. É importante frisar o alto custo de implementação de uma arquitetura centralizada.
- **Arquitetura Cliente/Servidor** – essa arquitetura se tornou muito popular (e é a mais popular atualmente) devido a muitos fatores como a simplicidade de sua implementação, custo reduzido comparado à arquitetura centralizada, delegação de tarefas simplificadas as máquinas dedicadas ao cliente, além de o usuário ter a vantagem de utilizar uma interface gráfica que melhor lhe convém. Nessa arquitetura o poder de processamento do lado do cliente é explorado. Atualmente tal arquitetura foi incorporada aos SGBDs comerciais, sendo que o cliente, no caso o *front_end*, executa a tarefa do aplicativo que é fornecer uma interface ao usuário. O *back_end*, que é onde está o servidor, executa as consultas e outras operações e retorna ao usuário o resultado. A figura 1 mostra um exemplo de uma arquitetura Cliente/Servidor.

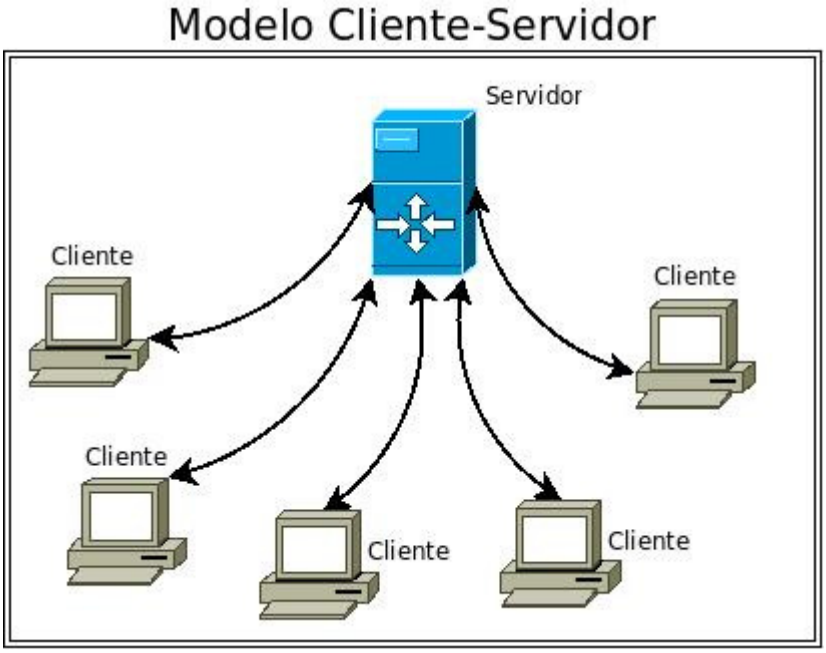

Figura 1 - Exemplo de uma arquitetura cliente/servidor
Fonte: SANCHES, A. R. , 2006

- **Arquitetura Distribuída** – essa arquitetura tem a função de “espalhar” várias bases de dados logicamente inter-relacionadas e ligadas através de rede. Num banco de dados distribuídos, os arquivos podem estar replicados ou fragmentados. No caso de serem replicados, existe uma cópia de cada um dos dados em cada nó, tornando as bases iguais. Essa transação pode ser síncrona ou assíncrona. Na fragmentação os dados encontram se divididos no sistema, ou seja, em cada nó existe uma base de dados diferente se esta visão for local, entretanto se os dados forem analisados globalmente os dados podem ser vistos de uma forma única, pois cada nó contém um catálogo que contém as informações relativas ao nó subjacente. Tal arquitetura é complexa, fato que a torna dispendiosa, uma vez que tal abordagem apresenta poucos casos de usos no mercado comparado à arquitetura cliente-servidor. A figura 2 mostra um exemplo de uma arquitetura Distribuída.

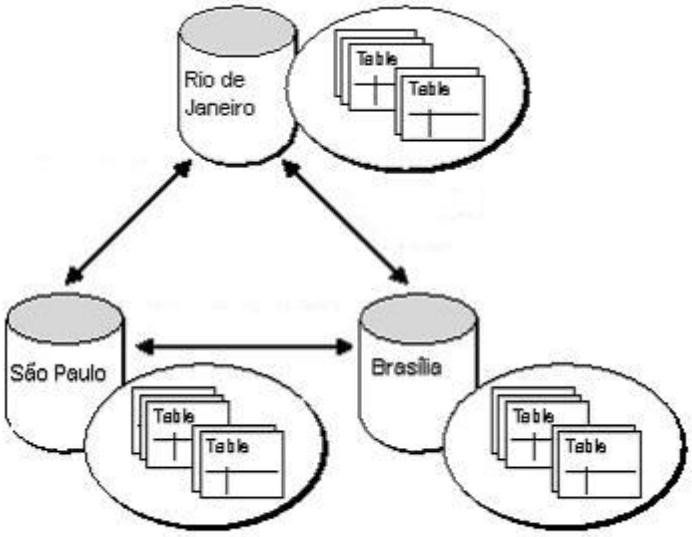

Figura 2 - Exemplo de uma arquitetura distribuída
Fonte: SANCHES, A. R. , 2006

1.5.1 Abstração de dados

Um banco de dados possui um conjunto de arquivos e programas inter relacionados que permite que o usuário possa manipular os mesmos. Porém o SGBD provê ao usuário uma visão abstrata dos dados, omitindo muitos detalhes de armazenamento entre outros. Como a abstração está dividida em níveis, o usuário consegue interagir com o sistema sem ter a necessidade de entender todo o seu funcionamento. Pode se citar três níveis de abstração:

- **Nível Físico** – este é o nível mais baixo de abstração e também o mais complexo, pois descreve como os dados são armazenados e as estruturas para tal.

- **Nível Conceitual ou Lógico** – este nível descreve a estrutura que os dados propriamente estão armazenados e também a relação entre eles. Em geral é largamente utilizado pelos *Database Analyst* durante a implementação e manutenção do SGBD.
- **Nível de Visões** – este é o nível mais alto de abstração, descrevendo apenas parte da informação. Aqui o usuário tem a visão da informação que lhe é mais interessante, não tendo a necessidade de enxergar os outros níveis.

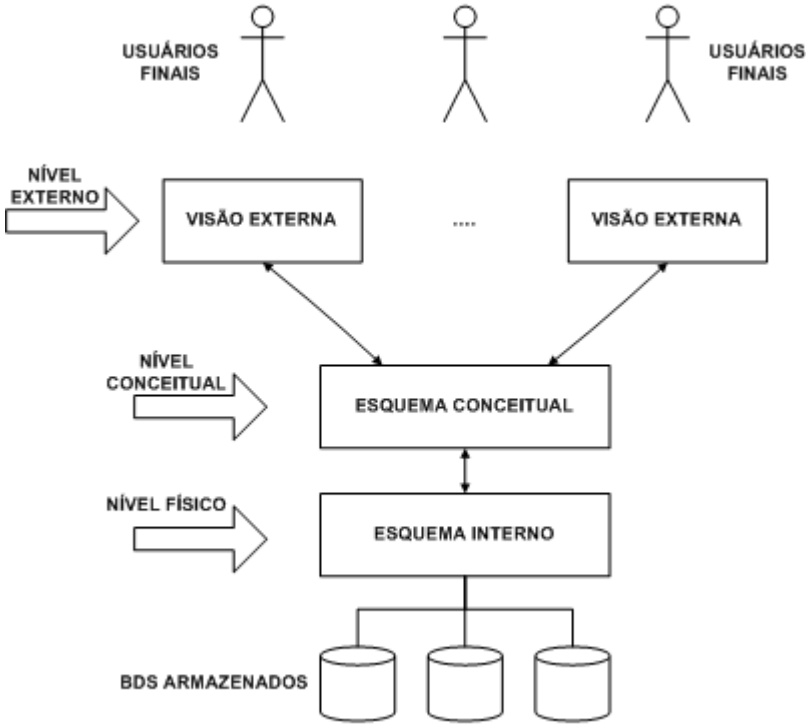

Figura 3 - Ilustração de abstração de dados
Fonte: SANCHES, A. R. , 2006

1.5.2 Estrutura do Sistema

A estrutura do SGBD é dividida em módulos que têm funções pré estabelecidas. Para o bom funcionamento de tais módulos, existe a necessidade de

que o Sistema Operacional no qual o SGBD está operando forneça os serviços mais básicos como, por exemplo, escrita/leitura. A Figura 4 fornece uma visão geral da estrutura de um SGBD.

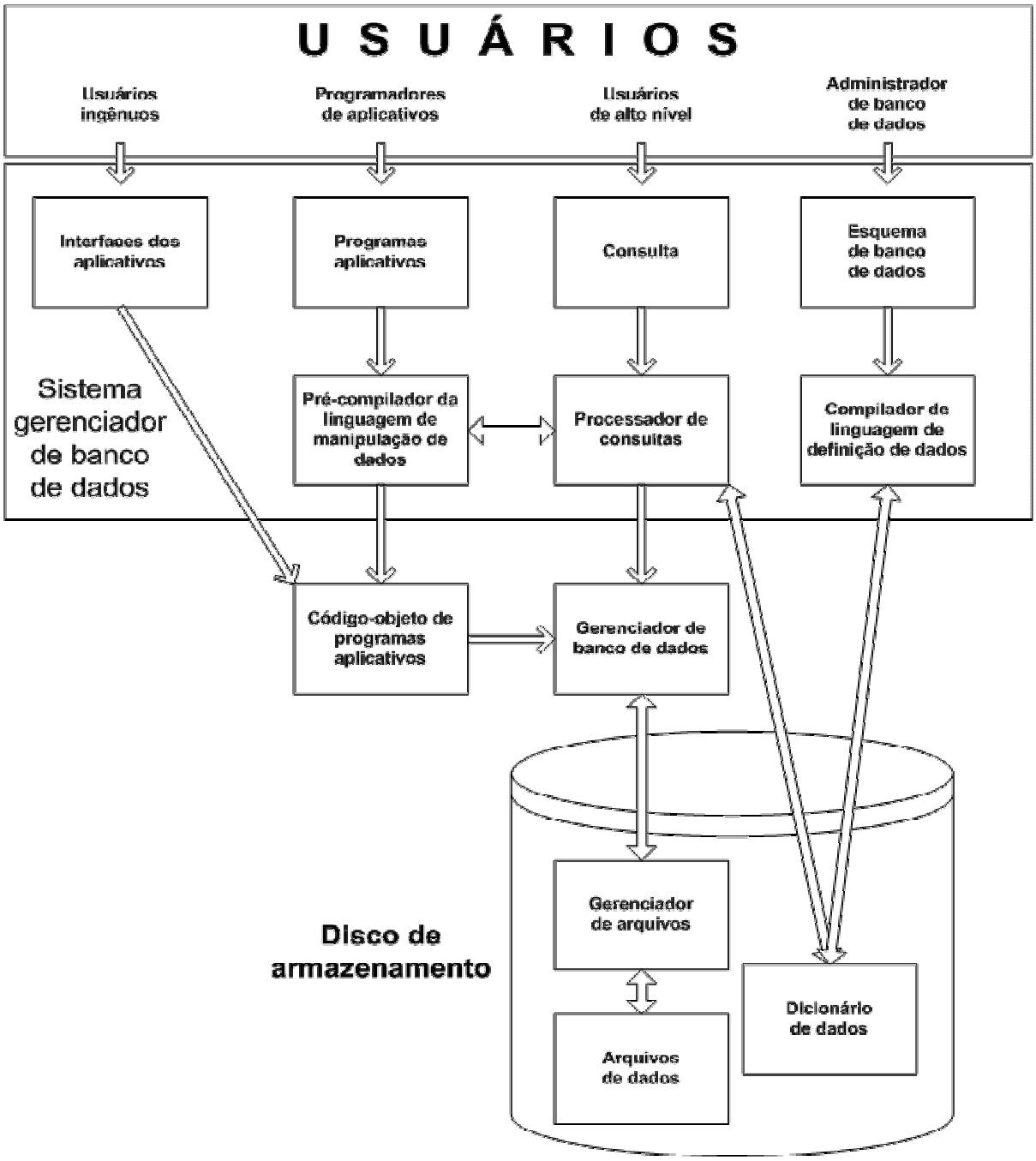

Figura 4 - Estrutura geral do SGBD
Fonte: SANCHES, A. R. , 2006

1.6 A Linguagem *SQL*

A linguagem *SQL* é sem dúvida uma das mais importantes razões para o sucesso dos bancos de dados relacionais. Desenvolvida originalmente no inicio dos anos 70 pelos laboratórios da *IBM*, a Linguagem Estruturada de Consulta tinha por objetivo demonstrar a viabilidade da implementação do Modelo Relacional proposto por Edgar Frank Codd. Seu nome inicial era *SEQUEL – Structured English QUEry Language* e foi inspirada, e é até hoje, nas características da Álgebra Relacional. Seu sucesso se dá devido à linguagem *SQL* proporcionar uma interface declarativa de alto nível, possuindo uma sintaxe amigável e de fácil entendimento comparado as linguagens formais. Outro fator que culminou no sucesso da *SQL* foi a sua padronização para os SGBDs relacionais comerciais.

> "Um esforço conjunto da ANSI (American National Standards Institute – Instituto Nacional Americano de Padrões) e da ISO (International Standards Organization – Organização Internacional de Padrões) chegou a versão-padrão da SQL (ANSI, 1986), chamada SQL-86 ou SQL1. Uma versão revisada e expandida chamada SQL2 (também conhecida como SQL-92) foi desenvolvida em seguida. A próxima versão do padrão foi originalmente chamada de SQL3, mas atualmente é conhecida como SQL-99". (ELMASRI; NAVATHE, 2005, p. 148.)

A *SQL* possui comandos para definição de dados, manipulação, atualização, consultas, alem de funcionalidades para visões e regras para permitir comandos *SQL* em linguagens de programação genérica, como *JAVA, COBOL*, ou C. Atualmente vários SGBDs utilizam a *SQL*, como por exemplo: *MySQL, Oracle, Firebird, Ingres, PostgreSQL*, etc.

1.6.1 Linguagem de Definição de Dados

A *DDL* é usada para estabelecer e descrever a estrutura de dados, ou propriamente o esquema num banco de dados. Uma vez compilados, os parâmetros de DDL são armazenados em um arquivo especial denominado dicionário de dados ou catálogo de dados. Um dicionário de dados é um arquivo que contém metadados, ou seja, dados sobre a estrutura de armazenamento. O SGBD sempre consulta este dicionário sobre cada operação feita no banco de dados. Pode se destacar alguns comandos usados em DDL:

- *Create* – o comando *create* pode ser usado para criar esquemas, tabelas (relações), domínios, visões, gatilhos, etc.
- *Drop* – o comando *drop* pode ser usado para excluir os mesmo elementos possivelmente criados com o comando *create* (tabelas, esquemas, visões, etc)
- *Alter* – o comando *alter* é usado para alterar um elemento já criado, modificando assim sua estrutura desse objeto.

Observe que os comandos DDL são executados pelo DBA que é a pessoa que gerencia todas as operações executadas no SGBD.

1.6.2 Linguagem de Manipulação de Dados

A *DML* é a linguagem que permite ao usuário fazer o acesso aos dados e manipulá-los através inclusão, remoção, alteração e consulta de registros. Um comando DML pode ser:

- Procedural - que é quando o usuário especifica qual dado é necessário obter e também especifica como obtê-lo;
- Não procedural – que é quando o usuário especifica qual dado é necessário obter, porém não especifica a forma para a obtenção do mesmo.

A SQL por padrão é não procedural, ou seja, uma operação de consulta, por exemplo, é deixada para o otimizador de consultas do banco de dados avaliar e efetuar a melhor forma de calcular tal consulta. Os comandos básicos de manipulação de dados podem ser descritos a seguir:

- *Insert* – o comando *insert* é usado para inserir um registro (formalmente uma tupla) em uma tabela (formalmente uma relação).
- *Delete* – o comando *delete* é usado para excluir um ou vários registros de uma tabela.
- *Update* – o comando *update* é usado para alterar um ou vários registros de uma tabela.
- *Select* – o comando *select* é usado para consultar um ou vários registros de uma tabela.

Existem ainda as cláusulas geralmente usadas em conjunto com os comandos DML. São elas:

- *From* – utilizada para especificar a tabela que se vai selecionar os registros
- *Where* – utilizada para especificar as condições que devem reunir os registros que serão selecionados.
- *Group by* – utilizada para separar os registros selecionados em grupos específicos.
- *Having* – utilizada para expressar a condição que deve satisfazer cada grupo.

- *Order by* – utilizada para ordenar os registros selecionados com uma ordem especifica
- *Distinct* – utilizada para selecionar dados sem repetição.

No caso dos comandos DMLs, qualquer usuário autorizado pode executar tais comandos, inclusive uma aplicação, não havendo a necessidade da intervenção de um DBA.

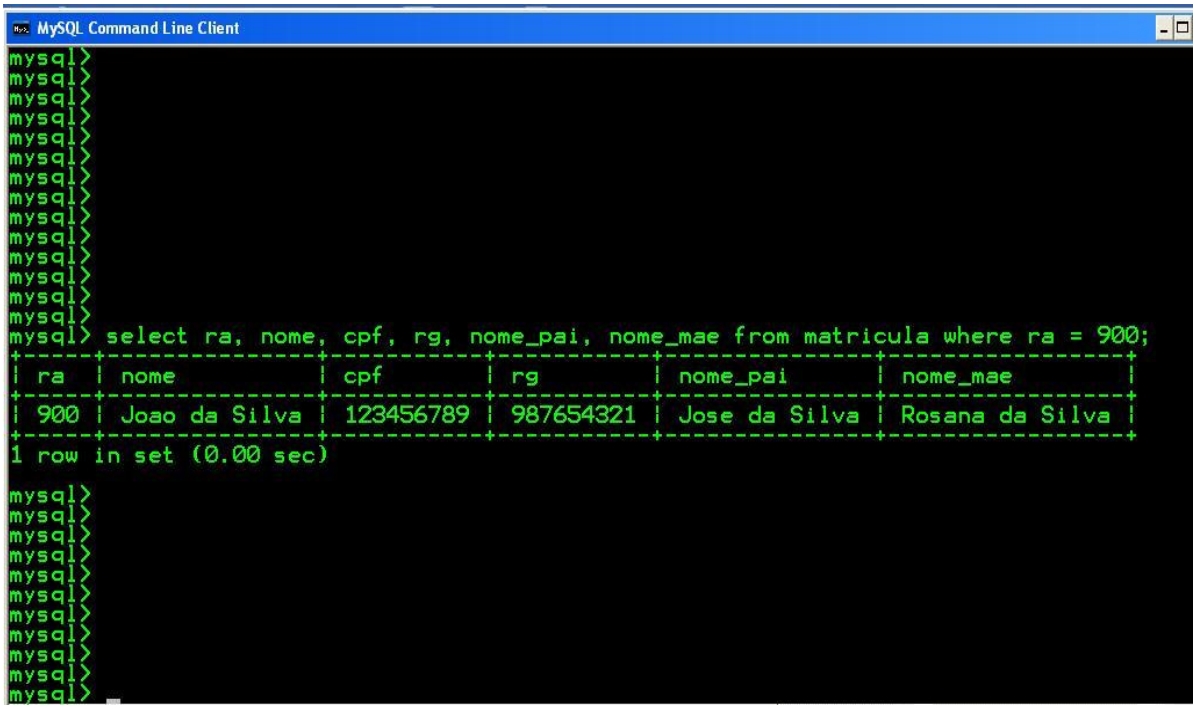

Figura 5 - Exemplo de uma consulta em *MySQL*
Fonte: Autoria própria

1.6.3 Linguagem de Controle de Dados

A *DCL* é basicamente usada para controlar o acesso aos dados de um SGBD. O DBA é o responsável por liberar ou cancelar privilégios aos usuários. Seus comandos incluem:

- *Grant* – usado para autorizar que o(s) usuário(s) possa realizar determinada tarefa.

- *Revoke* – usado para cancelar permissões concedidas a determinado usuário.

Os privilégios que podem ser permitidos ou negados incluem conexão, consulta, inserção, atualização, execução, etc.

1.6.4 Linguagem de Controle de Transação

A *TCL* é usada para gerenciar as alterações feitas durante a manipulação de dados. Através da TCL pode se confirmar uma transação na base de dados ou ainda retornar a um ponto anterior a alguma modificação. Os principais comandos da TCL são:

- *Begin work* ou *start transaction* - é usado para marcar o começo de uma transação de banco de dados que pode ser completa ou não.
- *Commit* – envia todos os dados das mudanças permanentemente
- *Rollback* – faz com que as mudanças nos dados existentes desde o último *commit* sejam descartadas.
- *Savepoint* – estabelece um ponto de restauração em uma transação que pode ser retornado pelo usuário.

2 MYSQL

O SGBD *MySQL* é atualmente um dos bancos de dados mais populares no mundo, fato este que se deve ao seu baixo custo aliado a sua praticidade e facilidade de manuseio. Dentre os usuários do *MySQL*, pode se destacar instituições renomadas como: Banco Bradesco, *Dataprev*, *HP, Nokia, Sony*, *U.S Army*, *U.S Federal Reserv Bank, Cisco System*, *NASA*, Google*, Wikipédia*, etc. A título de curiosidade o golfinho da logo marca do *MySQL* chama se *Sakila,* e o nome foi sugerido por um desenvolvedor de programas em um concurso feito para a escolha do nome.

Figura 6 - Logo marca do *MySQL*
Fonte: http://www.mysql.com/

2.1 História

O *MySQL* foi originalmente criado pela empresa *MySQL AB*, uma empresa estabelecida na Suécia pelos seus principais fundadores, David Axmark, Allan Larsson e Michael “Monty” Widenius. A parte AB do nome é um acrônimo a palavra sueca “*aktiebolag*", ou sociedade anônima.

> "Quando começamos, tínhamos a intenção de usar o mSQL para conectar às nossas tabelas utilizando nossas rápidas rotinas de baixo nível (ISAM). Entretanto, depois de alguns testes, chegamos a conclusão que o mSQL não era rápido e nem flexível o suficiente para nossas necessidades. Isto resultou em uma nova interface SQL para nosso banco de dados, mas com praticamente a mesma Interface API do mSQL. Esta API foi escolhida para facilitar a portabilidade para códigos de terceiros que era escrito para uso com mSQL para ser portado facilmente para uso com o MySQL". (AXMARK; LARSSON; WIDENIUS; 2010)

No dia 16 de Janeiro de 2008, a *MySQL AB*, foi adquirida pela *Sun Microsystems,* por US$ 1 bilhão, um preço exorbitante para um setor de licenças livres. No dia 20 de Abril de 2009 a *Oracle* adquiriu a *Sun Microsystems* e todos o seu produtos, inclusive o *MySQL.* Independente dessa aquisição o produto *MySQL* continua tendo uma característica muito apreciada pelo seus usuários: sua licença GNU. A licença *GNU* ou *GPL* é uma designação de licença para todo software livre, que se sustem basicamente em 4 pilares:

- A liberdade de executar o programa, para qualquer propósito (liberdade nº 0)
- A liberdade de estudar como o programa funciona e adaptá-lo para as suas necessidades (liberdade nº 1). O acesso ao código-fonte é um pré-requisito para esta liberdade.
- A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo (liberdade nº 2).
- A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie deles (liberdade nº 3). O acesso ao código-fonte é um pré-requisito para esta liberdade.

Figura 7 - Logotipo *GNU*
Fonte: http://www.gnu.org/

2.2 Características e Ferramentas

Atualmente em sua versão 5.5 o *MySQL* possui características e ferramentas relevantes, as quais são:

- **Alta Compatibilidade** – O *MySQL* é capaz de se comunicar com diversas linguagens de programação, como *Delphi, Java, C/C++, C#, Visual Basic, Python, Perl, PHP, ASP, Ruby, .NET* entre outras. Isso se deve a tecnologias como *ODBC* e *JDBC* que são padrões de conjunto de interfaces que permitem o acesso das linguagens de programação aos SGBDs.
- **Hardware** – O *MySQL* não requer grande poder de hardware para sua execução. Tanto o servidor como o cliente (caso haja um *front-end*) são capazes de serem executados, com funcionalidades mínima, utilizando até mesmo 500 MB de memória *RAM*, processador de 800 *Mhz* e 40 *GB* de disco rígido. Evidentemente que a utilização dessa configuração não é uma prática habitual e nem aconselhada.
- **Multiplataforma** – Dentre os Sistemas Operacionais suportados pelo *MySQL* pode se citar*: Windows (9x, Me, NT, 2000, XP, Vista e Seven), Solaris 2.5* e

superiores, *OpenBSD, Mac OS, Novell NetWare 6.0, Linux, HP-UX 11.x, FreeBSD 4.x, DEC Unix 4.x,* etc.

- **Suporte a Controle Transacional** – Na configuração *InnoDB* toda a estrutura transacional é suportada, inclusive com "*Lock row*" que é uma tecnologia que por segurança trava a linha que está sofrendo alteração impedindo um outro usuário de executar alguma alteração até que a primeira alteração seja confirmada através de um "*commit*".

- **Suporte a TRIGGERS** – A partir da versão 5, o *MySQL* oferece suporte a *TRIGGERS*, ou seja, gatilhos ou disparadores que são acionados sempre que há uma tentativa de modificar os dados de uma tabela que é protegida por ele. A *TRIGGER* é utilizada para manter a consistência dos dados ou para expandir as alterações de um determinado dado de uma tabela para outras. Pode se citar como exemplo, uma *TRIGGER* criada para controle de quem alterou uma determinada tabela, assim sendo, quando a alteração for efetuada, a *TRIGGER* é disparada e grava em uma tabela de histórico de alteração, o usuário e data/hora da alteração.

- **Suporte a *Views*** – que são objetos armazenados no SGBD pré definidos pelo usuário de acordo com uma consulta especifica, ou seja, a cada chamada de execução de uma *view*, ela ira apresentar os dados resultantes da consulta nela armazenada.

- **Suporte a Índices** – Exatamente como em um livro, os objetos de índices tem a função de agilizar a consulta, pois mantém a coluna que contém o índice ordenada de forma que quando é executada uma consulta, a varredura começa através do índice não perdendo tempo procurando em toda a base.

- **Suporte a *Stored Procedures*** – Procedimento armazenado ou *Stored Procedure* é uma coleção de comandos em *SQL* para uso do Banco de dados.

Tem a função de omitir tarefas repetitivas, aceitar parâmetros de entrada e retornar um valor de status (para indicar aceitação ou falha na execução). O procedimento armazenado pode reduzir o tráfego na rede, melhorar a performance, criar mecanismos de segurança, etc.

- **Suporte a Replicação** – A Replicação de dados é utilizada em bancos de dados distribuídos, como armazenamento e estratégia de *backups* entre computadores em locais distintos. A replicação sob uma rede de computadores pode ser feita inteiramente ou em partes, independente do armazenamento central de dados.

- **Suporte a *Storage Engines*** como *MyISAM, InnoDB* entre outros. A *storage engine*, traduzindo literalmente como "motor de armazenamento", pode ser definida como um conjunto de regras que o SGBD irá utilizar na manipulação das tabelas e dos dados. Basicamente, o formato *InnoDB*, que é usado na maioria dos SGBDS relacionais, oferece suporte a chaves estrangeiras, transações, travamento de linhas , etc. Entretanto este formato é um pouco mais lento se comparado ao formato *MyISAM*, que é um formato voltado para a velocidade, sem suporte a transação. Por padrão o *MySQL* utiliza o formato *MyISAM.*

- **Suporte a *multi-threads*** - usando threads diretamente no *kernel.* Isto significa que pode se usar múltiplas *CPUs* para múltiplos processamentos, caso existam.

- **Suporte a diversos tipos de campos** - como tipos inteiros de 1, 2, 3, 4 e 8 *bytes* com e sem sinal*, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM*, entre outros.

- **Suporte a multiusuários** – Oferecendo acesso direto de usuários e administradores, tratando com eficiências as concorrências entre tais.

- **Suporte a *Backup e Restore*** – Inclusive através do próprio aplicativo gráfico.
- **Segurança** – Com um sistema de privilégios e senhas que é muito flexível, seguro e que permite a verificação baseada em estações/máquinas. Teoricamente as senhas são seguras porque todo o tráfico de senhas é criptografado quando você se conecta ao servidor.

Além disso, o *MySQL* vem com funções pré definidas como *database ()* que retorna o nome do banco de dados, *current_user ()* que retorna o atual usuário entre outras. Toda a manipulação do SGBD pode ser feita através do seu *prompt* de comando, ou ainda existem muitos aplicativos gráficos para esta manipulação, como o *SQLyog*, *PHPMyAdmin* e o *MySQL Administrator,* sendo este ultimo desenvolvido pelos criadores do *MySQL.*

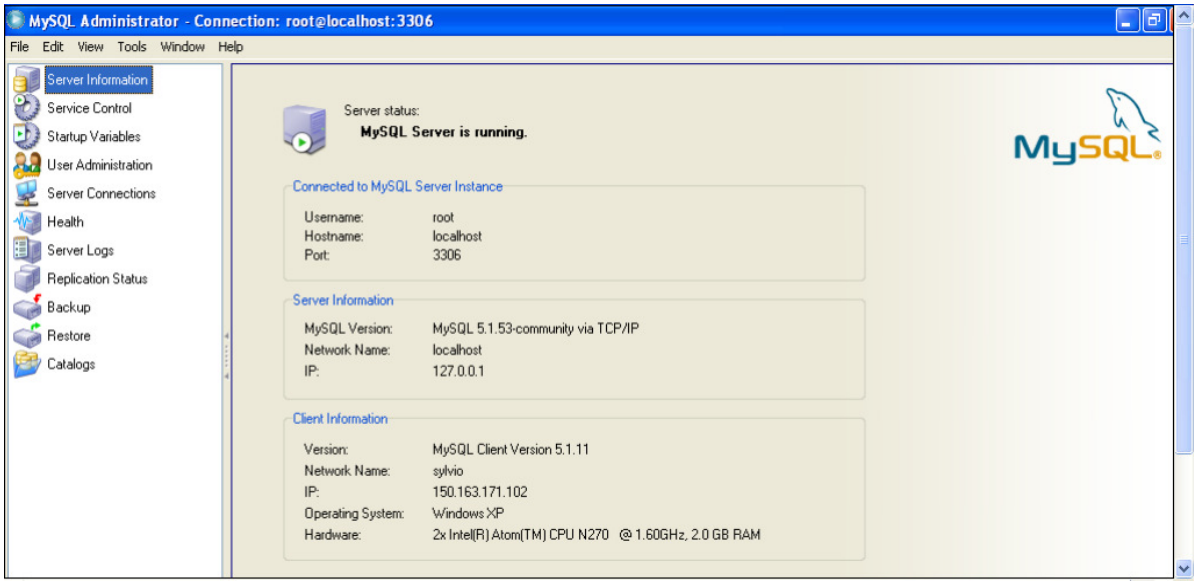

Figura 8 - *MySQL Administrator*
Fonte: Autoria própria

3 FIREBIRD

O *Firebird* (algumas vezes chamado de *FirebirdSQL*) é um SGBD em crescente popularidade em todo mundo, devido a sua estabilidade e alta *performance*. A Fundação *FirebirdSQL* coordena a manutenção e o desenvolvimento do *Firebird,* sendo que os códigos fonte são disponibilizados sob o *CVS* da *SourceForge*. Dentre os usuários do *Firebird* pode se citar instituições como CAEMA – Cia. de Águas e Saneamento do Estado do Maranhão, CEAL – Companhia Energética de Alagoas, CODECA – Cia. de Desenvolvimento de Caxias do Sul, Copervale, SBC – Sociedade Brasileira de Computação*, MultiBank S/A* entre outros.

Figura 9 - Logo marca do *Firebird*
Fonte: http://www.firebirdsql.org/

3.1 História

Tudo começou no projeto *Groton*, que teve seu início em meados dos anos 80 por uma equipe de engenheiros da *Digital Equipament Corporation*. A idéia era desenvolver um SGBDR que oferecesse benefícios inexistentes nos sistemas atuais da época. O *Groton* passou por várias alterações, até ser denominado *Interbase* 2.0 e introduzido algumas características como: *commit* automático de duas fases, replicações, acesso nativo a *driver JDBC*, tratamento de *blob´s* e sistema de eventos.

O *Firebird* surgiu em meados do ano 2000, quando a *Borland,* proprietária do banco de dados *InterBase*, resolveu abrir seu código fonte na versão 6.0. Então com essa abertura alguns programadores em associação assumiram o projeto de identificar e corrigir inúmeros defeitos da versão original, surgindo aí o *Firebird* 1.0, que se tornou um banco com características próprias, obtendo uma aceitação imediata no círculo de programadores. Note que apesar do *Firebird* ter “nascido” do *Interbase*, ele evoluiu com aspectos bem distintos, cada SGBD possui características e ferramentas próprias, inclusive com licenças diferenciadas, sendo assim o *Firebird* não deve ser considerado uma igualdade ou extensão do *InterBase*.

O *Firebird* utiliza a licença *GNU* em sua totalidade, sendo permitida toda e qualquer alteração e distribuição dos códigos fonte.

3.2 Características e Ferramentas

O *Firebird* em sua versão 2.5 possui características e ferramentas imprescindíveis a qualquer usuário. Atualmente o servidor é disponibilizado em três variações: *SuperServer, Classic e Embedded,* que são escolhidas no momento da

instalação do servidor. A versão *SuperClassic* só estará disponível a partir da versão 3.0.

- ***Classic Server*** – A versão *Classic* inicia um processo independente do servidor para cada conexão estabelecida. É indicada para máquinas com mais de um processador. A disputa entre os processos para acessar o banco é solucionada por um outro processo, que permanece sempre em execução garantindo aos clientes o travamento de recursos do banco.

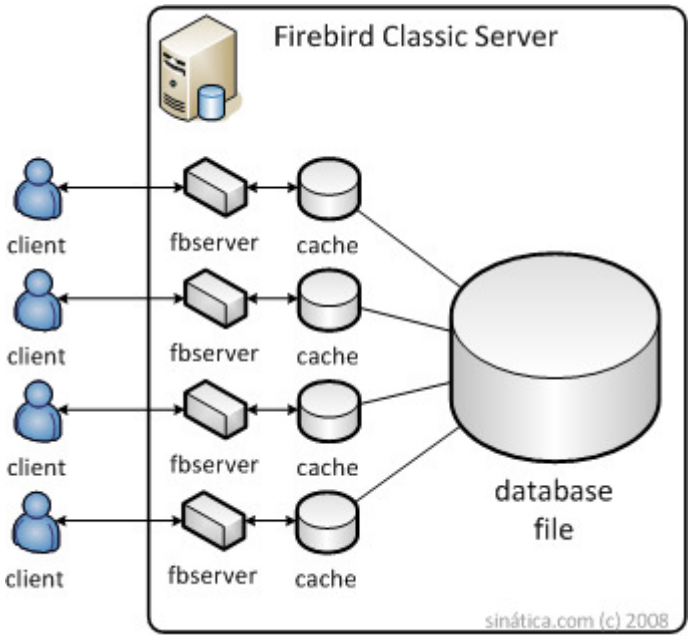

Figura 10 - *Classic Server*
Fonte: http://www.sinatica.com/

- ***Super Server*** - O *Super Server* compartilha o *cache* entre as conexões com o banco, e utiliza *threads* para gerenciar cada conexão.

> "*Os benefícios do modelo SuperServer são grandes. Foi melhorada a performance e integridade do banco, afinal só um processo de servidor passa a acessá-lo. A existência deste único processo desencadeia vários ganhos no sistema. Como foi eliminada a criação de um processo para cada cliente*

> *conectado, junto deixa de existir o cache das páginas do banco na memória do processo, e conseqüentemente não existem dados duplicados armazenados no sistema.* (DURAES, 2007, p. 18).

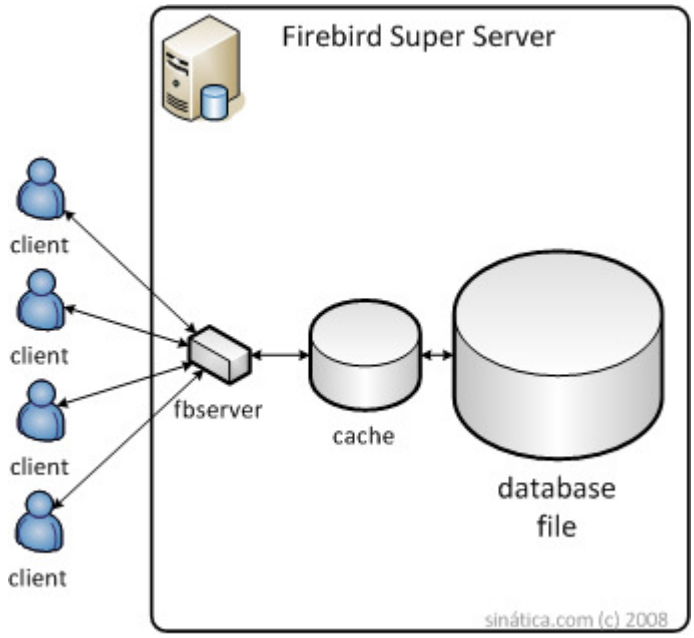

Figura 11 - *Super Server*
Fonte: http://www.sinatica.com/

- ***SuperClassic*** - O *SuperClassic* usa threads em um único processo do servidor, com *cache* independente para cada conexão. Ou seja, cada cliente tem uma *thread* dedicada e um *cache* dedicado dentro de um único processo.

> "*Criar centenas de threads é muito mais barato que criar centenas de processos e não existe perda de escalabilidade. A sincronização entre os caches pode ser feita diretamente em memória, o que reduz o custo de I/O. E outros controles que antes eram inter-processo agora são inter-thread, muito mais rápidos."* (TOSI, 2008)

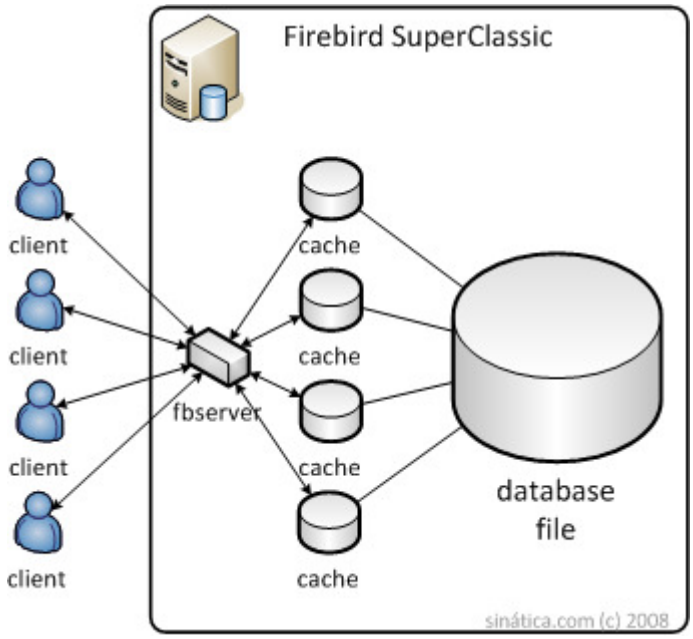

Figura 12 - *Super Classic Server*
Fonte: http://www.sinatica.com/

- ***Embedded*** - Consiste em um servidor *Firebird* completo composto por apenas alguns arquivos. É muito fácil de distribuí-lo, pois não há necessidade de instalação. Torna-se ideal para o uso em catálogos em *CDROM*, versões de avaliação de utilitários ou aplicações *standalone.*

Além dessas variações que o servidor do *Firebird* oferece, algumas características padrão são relevantes como:

- **Multiplataforma** – Dentre os Sistemas Operacionais suportados pelo *Firebird* estão: *Windows (9x, Me, NT, 2000, XP, Vista e Seven), Solaris (Sparc e Intel), Mac OS, Linux, HP-UX 11.x, FreeBSD 4.x,* etc. Além de suportar plataformas 64 *bits*, o *Firebird* receberá também uma versão *mobile* para *Android*, o sistema operacional da *Google* para dispositivos móveis.
- **Multiusuários** – Suporte completo a multiusuários, sem perda de desempenho.

- **Alta Compatibilidade** – Sendo capaz de se comunicar com linguagens de programação, como *Delphi, Java, C/C++, Visual Basic, Python, Perl, PHP, Ruby, .NET* entre outras. O *Firebird* utiliza tecnologias *como ODBC (Open Data Base Connectivity) e JDBC (Java Database Connectivity).*
- ***Hardware –*** O *Firebird* não necessita de grande poder de *hardware* para sua execução, o que o torna um SGBD de baixo custo.
- **Suporte a *TRIGGERS*** – Excepcionalmente o *Firebird* permite vários gatilhos para cada evento (*INSERT / UPDATE / DELETE*), para execução em uma seqüência especificada. Ele também suporta multi-evento, ou seja, um gatilho que pode ser executada condicionalmente ANTES ou DEPOIS de *INSERT*, *UPDATE* ou *DELETE*.
- **Suporte a *Views*** – Inclusive com a possibilidade de criação de *View* sobre *View.*
- **Suporte a Transações** – O *Firebird* oferece suporte nativo completo para *commit, rollback* e ponto de salvamento, incluindo pontos de salvamento aninhados. Além disso o *Firebrid* é totalmente compatível com ACID – Atomicidade, Consistência, Isolamento e durabilidade. Note que o *Firebird* (no caso o seu ancestral, *Interbase*) foi o primeiro banco de dados a usar a tecnologia de controle de concorrência, que foi posteriormente implementada em vários SGBDS como *Oracle, SQL Server, PostgreSQL* entre outros.
- **Suporte a Índices** – utilizando índices do tipo *Bitmap*, com organização e otimização automática.
- **Suporte a *Stored Procedures* e *UDFs*** – Nativo no *Firebird*, as *stored procedures* apresentam bom desempenho em vários tipos de comandos. As *UDFs (User Defined Functions* – Funções definidas por usuário) podem utilizar inúmeras funções externas do banco, como por exemplo uma função que envia

um e-mail ao *DBA* diretamente do servidor quando um registro importante é alterado.

- **Suporte a diversos tipos de campos –** Como *TEXT, LONG, FLOAT, DOUBLE, TIMESTAMP, BLOB, DATE, TIME, INT64*, etc
- **Suporte a Backup e restore** – Inclusive com *backup* incremental sem intervenção de um *DBA*. Também é possível realizar o *backup* sem o desligamento do banco (*online*).
- **Suporte a replicação** – Através de aplicativo externo ao banco como o *IBreplicator.*
- **Suporte a *Sub Selects* –** Sendo possível aninhar várias sub-consultas dentro de uma consulta geral.
- **Segurança** – Total controle de usuários, com manipulação de privilégios através de autenticação e autorização. Entretanto o *Firebird* não criptografa o banco de dados ou a comunicação em redes, sendo este trabalho feito por um aplicativo externo complementar chamado *Zebedee.*

Observe que como no *MySQL*, toda a manipulação do SGBD pode ser feita através de *prompt* de comando, entretanto existem aplicativos gráficos para esta manipulação, como o *IBExpert, FlameRobin, DB Workbench e Firebird Development Studio*.

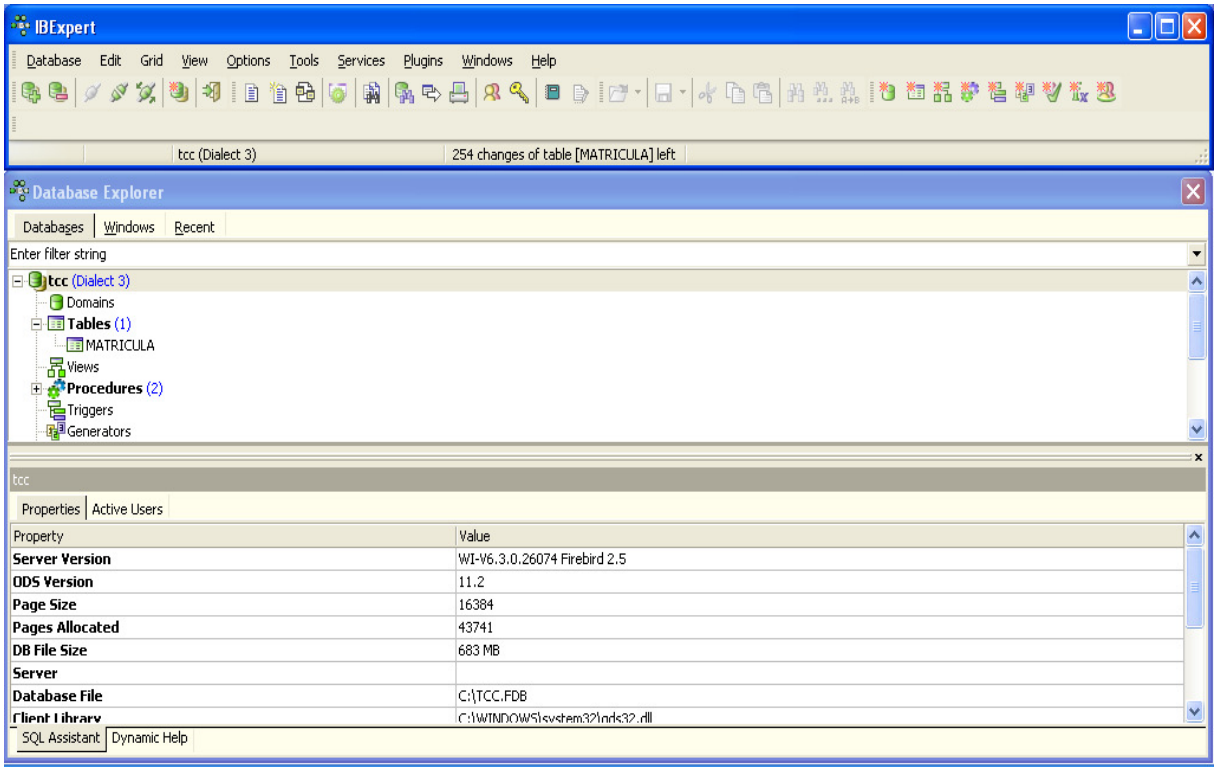

Figura 13 - *IB-Expert*
Fonte: Autoria própria

4 AVALIAÇÃO DE DESEMPENHO

A tarefa de avaliar *softwares* não é das mais triviais. É necessário que o padrão de medida usado seja único para que os *softwares* submetidos aos testes não sejam favorecidos ou penalizados. Segundo Kozievith (2005, p. 6), "*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."* Sendo assim, *benchmarks* têm sido criados para essa finalidade. Um *benchmark* é "um padrão para medida ou avaliação" (PIRES; NASCIMENTO; SALGADO, 2006 *apud Webster's II*). Ou seja, em computação, *benchmarks* são *softwares* que realizam uma série restrita e pré estabelecida de operações e retornam um resultado em um formato que descreve o comportamento do sistema.

Atualmente pode se citar vários *benchmarks* usados para comparar desempenho de SGBDs tais como: *TPCC-UVA* (HERNÁNDEZ E GONZALO 2002), *OSDB (OSDB 2001*) e o conhecido *benchmark TPC-C*, da *TPC™* (*TPC 2001*). Entretanto tais *benchmarks* apresentados não são capazes de avaliar toda sorte de SGBDs disponíveis, devido às particularidades que cada SGBD possui. Este fato culminou na necessidade de implementação de um conjunto de rotinas que pudessem satisfazer as necessidades especificas para a avaliação do desempenho dos SGBDs *MySQL* e *Firebird*. De acordo com Weiss (WEISS, 2008, *apud* Gray 1993)

> "*a realização de um benchmark de bancos de dados é importante para a especificação do domínio de atuação, que compreende o conteúdo analisado, ou seja, os itens aferidos pelo benchmark, e sua forma de avaliação através de unidade de medida que permita análise dos resultados. A escolha dos itens é diretamente relacionada ao objetivo do teste, sendo necessário*

à escolha de itens que resultem em dados relevantes para fundamentar as análises propostas pelo benchmark. Como não existe uma métrica única que possa avaliar o desempenho dos diversos sistemas e todas as suas aplicações, os domínios específicos de benchmarks são uma orientação para se aperfeiçoar os testes nesta diversidade de SGBDs."

A título de exemplo, Kozievitch em 2005 apresenta a viabilidade da utilização do SGBD *PostgreSQL* para dados meteorológicos, tendo se em vista a impossibilidade da utilização única do *benchmark* TPC, uma vez que o mesmo sozinho não seria capaz de avaliar todas as características de uma base de dados meteorológica. Ainda como exemplo de customização de *benchmarks* pode se citar o próprio OSDB, que é um *benchmark* de código aberto baseado no *AS3 AP*, disponível para alterações de seus usuários.

De acordo com o que foi exposto por Weiss em 2008, em adição aos conceitos de relevância, portabilidade, escalabilidade e simplicidade apresentados por Gray (GRAY 1993), pode se notar a viabilidade técnica e a eficácia resultante da implementação de um *benchmark* customizado para avaliar os SGBDs *MySQL* e *Firebird*. Observe que os conceitos expostos por Gray definem uma boa diretriz na configuração de um *benchmark,* sendo eles:

- Relevância: As medidas verificadas deverão descrever funcionalidade e expectativa de desempenho acerca do produto submetido ao *benchmark*. É necessário especificar com clareza o domínio da utilização da ferramenta e as operações a serem submetidas;
- Portabilidade: Na elaboração de um *benchmark,* as definições e os termos utilizados devem estar em um nível de abstração que permita transportá-lo para as implementações de diferentes ferramentas. Portanto, não deve haver privilégio na utilização de um contexto que seja particular a uma determinada

abordagem. O objetivo desta consideração é tornar o *benchmark* aplicável a uma gama maior de ferramentas a serem avaliadas;

- Escalabilidade: O padrão das medições deve atender a pequenos ou grandes sistemas, independente do nível de complexidade dos mesmos. Fatores como monoprocessamento ou multiprocessamento, *clock* e tamanhos de memórias não devem tornar as medidas inconsistentes, nem influenciar na aplicabilidade do *benchmark;*
- Simplicidade: Os critérios definidos no *benchmark* devem ser simples e de fácil compreensão. Porém, a relevância das métricas não deve ser negligenciada. Medidas muito simples podem não refletir boas características de julgamento de uma ferramenta. No entanto, se for definido um conjunto de métricas muito complexas e de difícil assimilação pela comunidade interessada nos resultados, poderá ocorrer uma perda de credibilidade do *benchmark.*

4.1 História

Os estudos realizados dentro das empresas no intuito de aumentar a produtividade e o lucro datam de meados dos anos 70. A forma de pesquisa era rudimentar, procurando se avaliar o departamento ou setor que apresentavam melhores resultados e assim sendo estabelecendo como modelo para toda a corporação.

Já nos anos 80 o termo *benchmarking* passou a ser largamente usado pelas empresas como ferramenta de qualidade, sendo este termo derivado da palavra *Benchmark*. A idéia é justamente ter um padrão, um ponto de referência, um nível acerca de qualquer processo dentro de uma empresa, para então a partir desse ponto gerar uma vantagem competitiva ou mesmo propor melhorias.

4.2 Exemplos

Atualmente existem vários *Benchmarks* utilizados para avaliar diversos tipos de sistemas, como SGBDs, servidores *WEB*, Sistemas Operacionais, *softwares* de manipulação gráfica, etc. Como já citado anteriormente, pode se destacar alguns *benchmarks* para SGBDs :

- *OSDB – Open Source Database Benchmark*, é um *benchmark* de código aberto, que suporta os SGBDS livres *MySQL* e *PostgreSQL*. Infelizmente ainda não existe uma compilação deste *benchmark* que ofereça suporte a outros SGBDS como *Ingres* ou *Firebird.*

> *"O benchmark OSDB (Open Source Database Benchmark) foi criado com o objetivo inicial de avaliar a taxa de I/O e o poder de processamento da plataforma GNU Linux/Alpha. Sua implementação é baseada no benchmark AS3AP, diferindo em alguns aspectos: quantidade de métricas retornadas e número de módulos. Enquanto a análise dos resultados gerados pelo benchmark AS3AP baseia-se em uma única métrica (tamanho máximo do banco de dados suficiente para completar o teste em menos de 12 horas), o OSDB possibilita a comparação através de outras métricas: tempo de resposta das consultas e número de linhas retornadas por segundo*." (PIRES; NASCIMENTO; SALGADO, 2006, p 2).

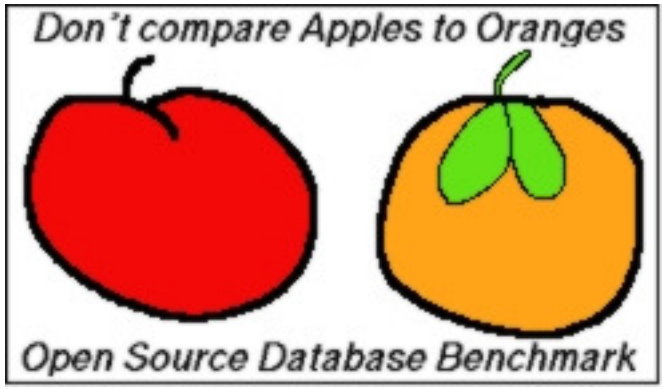

Figura 14 - Logo marca do *OSDB*
Fonte: http://osdb.sourceforge.net/

- TPC - é considerada por muitos a principal referencia no mundo tratando se de *benchmarks* computacionais. Surgido em 1988, o TPC é um consórcio sem fins lucrativos, que realiza diversas avaliações e publica os relatórios de pesquisa, sendo estes TPC-C, TPC-H, TPC-W, entre outros. Observe que o TPC-C é o relatório utilizado para a avaliação de SGBDs. Dentre os SGBDs avaliados pelo TPC pode se citar o *Oracle* e o *IBM DB-2.*

  > "*É definida uma carga de processamento de transações on-line que simula as atividades encontradas em um ambiente de aplicações complexas, caracterizado por: execução simultânea de múltiplos tipos de transações; múltiplas sessões; entrada / Saída de disco significantes; integridade da transação (propriedade ACID); distribuição não-uniforme de acessos aos dados; banco de Dados consistindo de muitas tabelas, com uma grande variedade de tamanhos, atributos e relacionamentos; concorrência no acesso aos dados. A métrica de desempenho definida é uma taxa medindo o número de transações efetivadas com sucesso por minuto (tpmC). Múltiplas transações são utilizadas para simular a atividade do negócio*." (DURAES, 2007, p 14)

4.3 Metodologia

Como já observado, os atuais *benchmarks* são bastante poderosos, entretanto ainda não são capazes de avaliar todos os SGBDs disponíveis. Com base nessa condição e seguindo os fundamentos expostos por Gray (Gray 1993), observou se a necessidade, possibilidade e viabilidade de implementação de um *benchmark* customizado para avaliar os SGBDs *MySQL* e *Firebird.*

4.3.1 Descrição da métrica utilizada.

Os SGBDs *MySQL e Firebird* tiveram como métrica de avaliação a realização de duas rotinas de testes que consistiram na execução de comandos DML e na medição do tempo de resposta do SGBD a cada comando. Ou seja, neste trabalho o quesito avaliado foi á velocidade de execução dos comandos. Em cenário mono-usuário, o experimento consistiu em realizar duas simulações utilizando uma tabela idêntica, única e padrão (número de colunas fixo) para ambos os SGBDs. Esta mesma tabela foi usada em ambas as simulações, sendo a primeira simulação com 1000 registros e a segunda com 1000 000 de registros.

As duas simulações seguiram as seguintes etapas:

- Inserção (no banco vazio) dos registros através de uma *stored procedure.*
- Consulta de **um** registro.
- Atualização de **todos** os registros.
- Exclusão de **todos** os registros.

Para cada etapa foi armazenado o tempo de execução da tarefa através da função *getTime()* da Classe *Data* da linguagem *Java.* Assim, ao iniciar a execução do comando, a função era chamada coletando o tempo atual naquele instante e alocando na variável "*TempInicial*". Após o término da execução do comando, novamente era chamada a função que coletava o tempo naquele instante, agora alocando este tempo na variável "*TempFinal".* Então obteve se o tempo total de execução do comando através da operação ("*TempFinal*" – "*TempInicial*"), convertendo o resultado da operação de milissegundos para segundos. Como a execução única do comando poderia não representar a realidade do seu tempo, então cada comando foi executado cinco vezes e então calculada a média aritmética dos tempos, obtendo assim um tempo médio de execução de cada comando.

Após as simulações, os resultados apresentando pelos SGBDs foram organizados em tabelas e posteriormente confrontados.

4.3.2 Material

Para execução das simulações foram instalados os servidores *MySQL 5.1* e *Firebird 2.5* em um *Netbook Acer Aspire One V1.06,* com processador *Intel Atom 1600 MHz , 2 GB DDR2* de memória *RAM*, 1*50GB* de disco rígido (NTFS), utilizando Sistema Operacional *Windows XP Service Pack 3.* Também foi instalado o ambiente de desenvolvimento *Netbeans 6.0.1* juntamente com a linguagem de programação *Java SE* versão 6 onde foi implementado o *software (benchmark)* que executou as simulações. Este *software* tinha por função a conexão com os SGBDs, a execução dos comandos DMLs e a medição do tempo de resposta. A tabela utilizada foi extraída de um sistema de matrícula escolar da Faculdade de Tecnologia de Guaratinguetá. A figura a seguir mostra o diagrama da tabela:

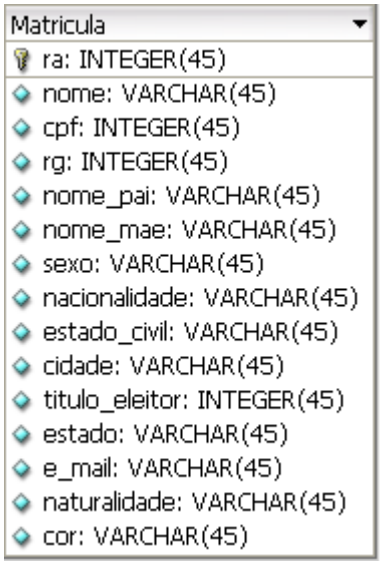

Figura 15 - Diagrama da tabela de matrícula
Fonte: Autoria própria

Observe que esta tabela foi criada uma única vez tanto no servidor *MySQL* como no servidor *Firebird.* Ela possui 15 campos, sendo os tipos dos campos descritos no próprio diagrama. Os dados usados para popular a tabela são meramente fictícios, sendo inseridos de forma repetida pelo software no momento da inserção. Como nesse caso a chave primária é o campo RA, e ela tem a propriedade de auto incremento, então os registros se tornam únicos. O tamanho dos campos foram pré definidos em 45 caracteres e totalizam 180 para o tipo *integer* e 495 para o tipo *varchar* respectivamente. Ambos servidores utilizam de 3 a 5 bytes para o armazenamento, variando de acordo com a cadeia de caracteres armazenada.

4.3.3 A linguagem Java

Em 1991, na *Sun Microsystems*, foi iniciado o *Green Project*, que pode ser chamado o berço da linguagem *Java*. Os mentores do projeto eram Patrick Naughton, Mike Sheridan, e James Gosling. O objetivo do projeto não era exatamente a criação

de uma nova linguagem de programação, mas antecipar e planejar a “próxima onda” do mundo digital. Eles acreditavam que, em algum tempo, haveria uma convergência dos computadores com os equipamentos e eletrodomésticos comumente usados pelas pessoas no seu dia-a-dia. A linguagem *Java* possui paradigmas de Orientação a Objetos e estruturação, sendo uma linguagem de programação e uma plataforma de computação. É a tecnologia que capacita muitos programas de alta qualidade, como utilitários, jogos e aplicativos corporativos, entre muitos outros, por exemplo. O *Java* é executado em milhares de computadores pessoais e em bilhões de dispositivos em todo o mundo, inclusive telefones celulares e dispositivos de televisão. Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem *Java* é compilada para um *bytecode* que é executado por uma máquina virtual.

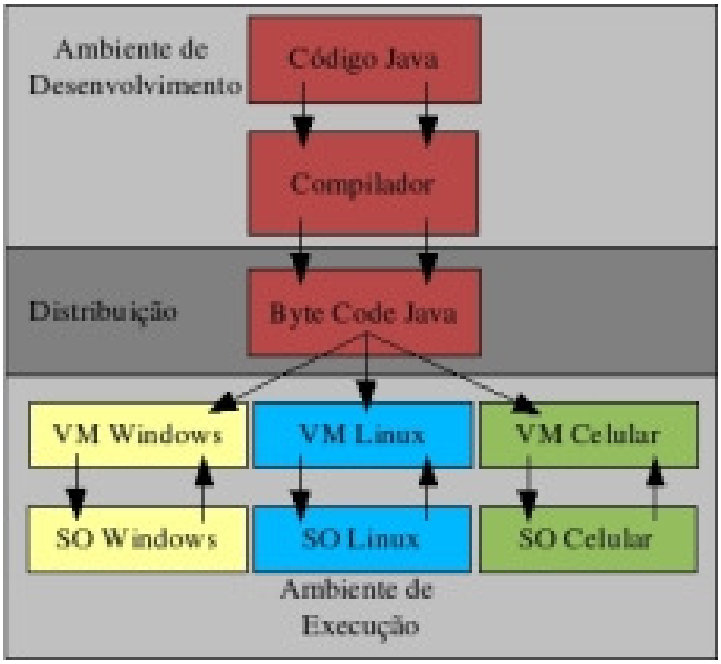

Figura 16 - Diagrama de funcionamento da tecnologia *Java.*
Fonte: http://www.java.com/

Utilizando a licença GNU em sua totalidade, a linguagem Java foi escolhida para a implementação do *software de benchmark* devido a diversos fatores, como sua heterogeneidade de plataforma (atendendo ao requisito de portabilidade estabelecido por Gray, 1993), estabilidade, velocidade de execução, licença livre, entre outros. Atualmente a *Oracle* detém as patentes e os direitos autorais sobre toda a plataforma *Java.*

Figura 17 - Logo marca *Java*
Fonte: http://www.java.com/

4.3.4 O Netbeans

O *NetBeans IDE* é um ambiente de desenvolvimento - uma ferramenta para programadores escreverem, compilarem, depurarem e implantarem programas. É escrito em *Java* - mas pode suportar qualquer linguagem de programação, *como C, C++, PHP, Groovy, Ruby,* entre outras. Tem a característica de ser multiplataforma, podendo ser executado em Sistemas *Windows, Linux, Mac OS*, etc. O *NetBeans* foi iniciado em 1996 por dois estudantes tchecos na Universidade de Charles, em Praga.

Primeiramente o nome do projeto era *Xelfi*, em alusão ao *Delphi,* pois a pretensão deste projeto era ter funcionalidades semelhantes aos *IDEs* então populares do *Delphi* que eram mais atrativas por serem ferramentas visuais e mais fáceis de usar, porém com o intuito de ser totalmente desenvolvido em *Java.*

Dentre as características mais relevantes do *Netbeans,* pode se citar: editor integrado, visualizador de classes integrado, *plugins* para *UML,* suporte a banco de dados, *data view*, etc. O NetBeans IDE é um produto totalmente gratuito e sem restrições de uso. A escolha deste ambiente para o desenvolvimento se justifica devido a sua grande popularidade e facilidade de implementação para a linguagem *Java.* Atualmente a *Oracle* é a proprietária do produto *Netbeans.*

Figura 18 - Logo marca *NetBeans*
Fonte: http://www.netbeans.org/

4.3.5 O *Software* Implementado

O *software* de *benchmark* implementado neste trabalho seguiu os conceitos estabelecido por Gray(1993), sendo relevância, portabilidade, escalabilidade e simplicidade. A relevância, pois mede o desempenho de pico e a relação custo / desempenho ao realizar operações típicas. A portabilidade, pois é fácil de ser

transportado para outros sistemas. Isso se deve a tecnologia *JAVA* que permite sua execução em qualquer sistema. A escalabilidade, pois é capaz de executar em sistemas de pequeno e grande porte. Ainda o *software* esta preparado para manipular uma porção crescente de trabalho, estando preparado para ser redimensionado (caso necessário). E a simplicidade, pois é de fácil entendimento e de fácil compreensão. O *sofware* seguiu a estruturação utilizando Classes e Orientação a Objetos, sendo as 3 classes:

- **Conexão *MySQL*** – esta classe possui as rotinas utilizadas para conexão com o banco *MySQL,* as rotinas de atualização, exclusão e carga de dados.
- **Conexão *Firebird*** – semelhantemente, esta classe possui as rotinas utilizadas para a conexão com o banco *Firebird,* bem como as rotinas de atualização, exclusão e carga de dados.
- **Telas** – esta classe possui a tela sendo esta a parte gráfica do software que é totalmente orientada aos objetos (painéis, botões, caixas de texto, etc). Esta classe também possui a rotina de execução principal. Tal rotina é utilizada para chamar as conexões com os bancos, de acordo com o banco escolhido para a simulação. Esta rotina também calcula o tempo de execução dos comandos DML, bem como sua exibição ao usuário. A figura 19 mostra um exemplo da tela principal de execução do *software*.

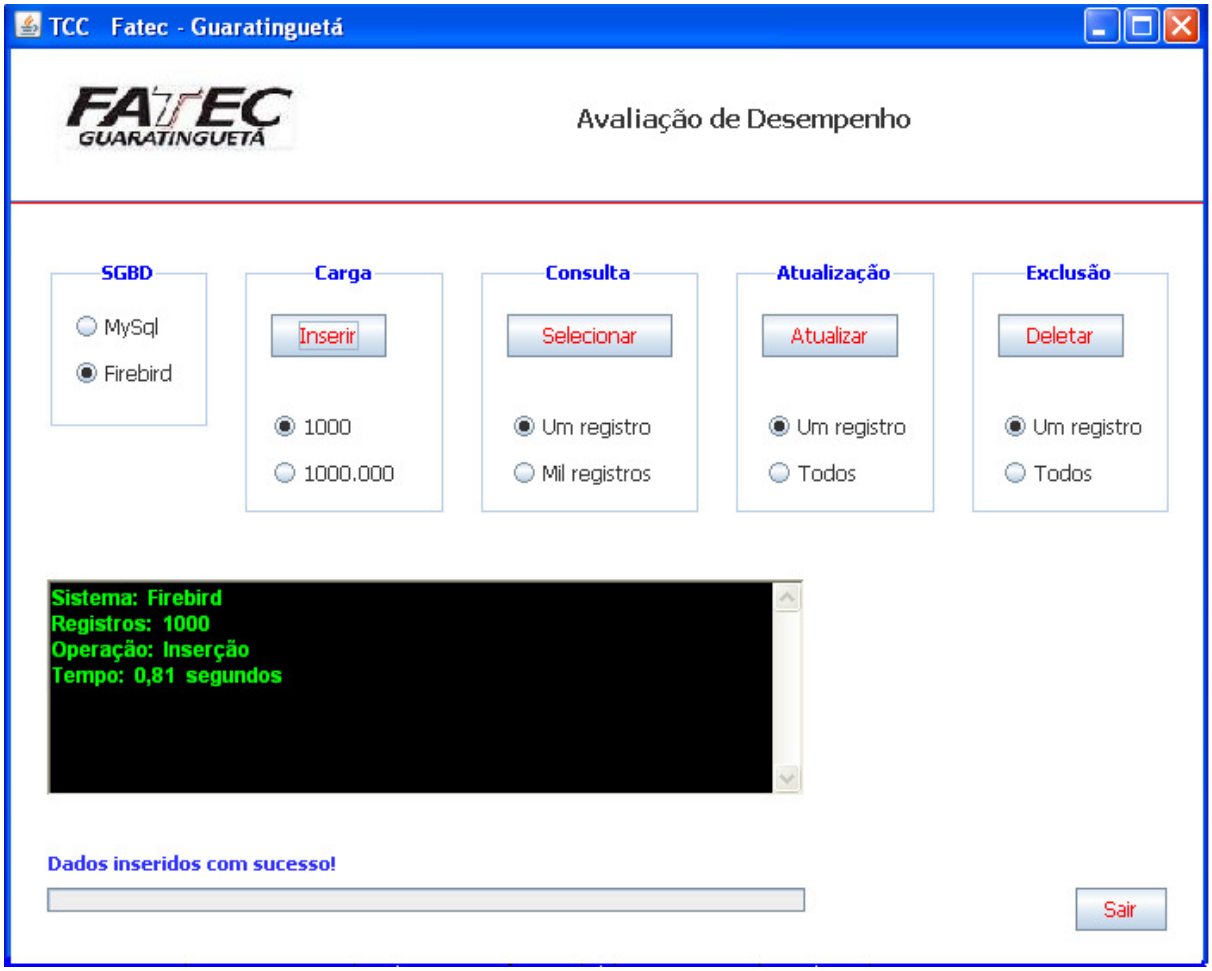

Figura 19 - Exemplo de execução do software de avaliação de desempenho.
Fonte: Autoria própria

Apesar de ainda não ser uma ferramenta completa, necessitando de complementos para se tornar comercial ou disponível ao público, este *benchmark* atende perfeitamente a proposta deste trabalho, que é medir o desempenho dos SGBDs *Firebird e MySQL* em comandos DML, para posterior comparação dos resultados. Futuramente, sugere se uma agregação de novas técnicas, métricas e rotinas neste *benchmark,* como por exemplo a execução em modo multiusuário, a comparação para comandos DDL, a inclusão de outros SGBDs, a manipulação de índices e de outros objetos de Bancos de Dados, entre outras rotinas. Sendo assim, no momento que este software de *benchmark* estiver com todas as funcionalidades

citadas, a inclusão de uma documentação de software se fará necessária, que deverá lhe ser adicionada no momento que houver um trabalho de publicação do *benchmark*, que não é o escopo desta pesquisa.

5 RESULTADOS E DISCUSSÃO

Este capítulo descreve os resultados obtidos através das simulações. O primeiro SGBD submetido aos testes foi o *Firebird,* e posteriormente o *MySQL.* A tabela 1 mostra os resultados obtidos das simulações para o SGBD *Firebird*

Tabela 01 - Resultados do SGBD *Firebird*
Fonte: Autoria própria

Firebird	**Base de dados (Registros)**	
Comando	**1000**	**1000 000**
Inserção	0,86 seg	1,64 min
Seleção	0,23 seg	0,20 seg
Atualização	0,24 seg	3,15 min
Exclusão	0,20 seg	1,05 min

Note que o comando para seleção tem desempenho muito semelhante tanto na base de 1000 como na base de 1000 000 de registros. Isso demonstra que o *Firebird* manipula com a mesma eficiência tanto bases pequenas como bases grandes de dados. A inserção através da *stored procedure* apresentou um tempo satisfatório em ambas as bases, apontando que o *Firebird* tem uma boa capacidade no tratamento de *stored procedures*. A atualização apresentou o tempo mais critico na base de 1000000 dados devido a necessidade de reescrita de muitos registros. O tempo de exclusão também teve um tempo satisfatório mesmo na função de deletar uma grande quantidade de dados (1000000).

A tabela 2 mostra os resultados obtidos das simulações para o SGBD *MySQL:*

Tabela 02 - Resultados do SGBD *MySQL*
Fonte: Autoria própria

MySQL	**Base de dados (Registros)**	
Comando	**1000**	**1000 000**
Inserção	32,58 seg	6,77 hor
Seleção	0,11 seg	0,50 seg
Atualização	0,49 seg	6,95 min
Exclusão	0,17 seg	1,06 min

Observe que o comando de inserção utilizando a *stored procedure* é penoso para o *MySQL,* chegando a ter um elevado tempo para a inserção de 1000 000 de registros. O comando de seleção apresenta um ótimo desempenho em ambas bases de dados, evidenciando a eficiência do SGBD. O comando de atualização apresentou um tempo favorável em 1000 registros, entretanto teve um alto tempo na base maior. O comando de exclusão obteve bom desempenho em ambas as bases.

A Tabela 3 mostra o confronto entre os SGBDs.

Tabela 03 - Confronto entre os SGBDs *MySQL e Firebird*
Fonte: Autoria própria

Base	**Comandos**	***MySQL***	***Firebird***
1000	**Inserção**	32,58 seg	**0,86 seg**
	Seleção	**0,11 seg**	0,23 seg
	Atualização	0,49 seg	**0,24 seg**
	Exclusão	**0,17 seg**	0,20 seg
1000 000	**Inserção**	6,77 hor	**1,64 min**
	Seleção	0,50 seg	**0,20 seg**
	Atualização	6,95 min	**3,15 min**
	Exclusão	1,06 min	**1,05 min**

O gráfico a seguir mostra o confronto na base de 1000 registros:

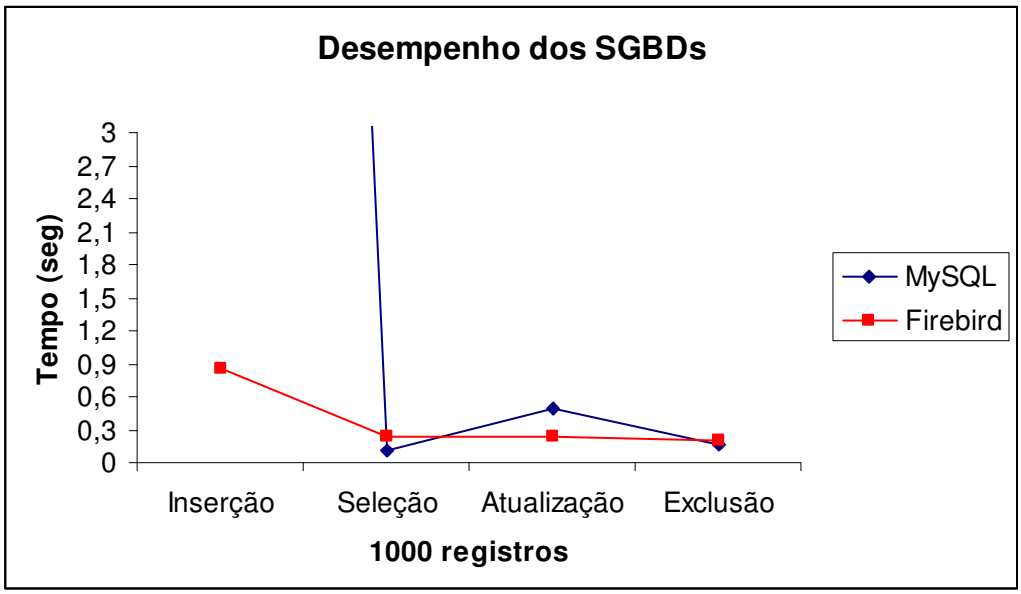

Figura 20 - Gráfico de desempenho dos SGBDs (1000 registros)
Fonte: Autoria própria

A figura a seguir mostra o gráfico do confronto na base de 1000 000 de registros:

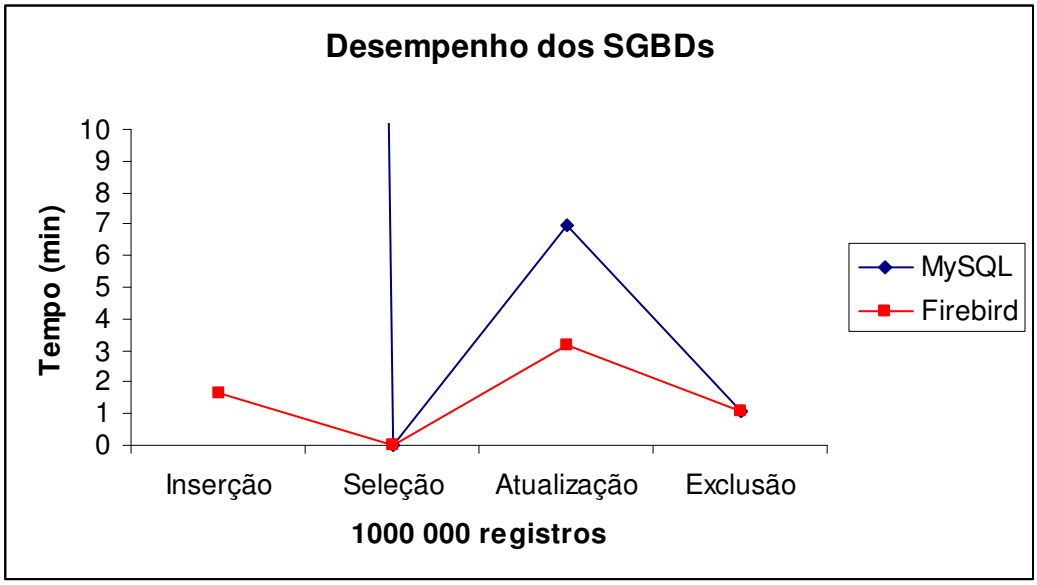

Figura 21 - Gráfico de desempenho dos SGBDs (1000 000 registros)
Fonte: Autoria própria

Tanto na tabela 3 como nas figuras 20 e 21 pode se observar que em comandos que requerem uma escrita em disco, como inserção e atualização, o *Firebird* apresentou um melhor desempenho em relação ao *MySQL* em ambas as simulações. Ainda, o *Firebird* possui um melhor tratamento com *stored procedures* e processamento de funções que envolvem laços de repetição e cálculos numéricos, sendo essa característica mais evidenciada em bases de grande volume de dados. No comando de seleção na base de 1000 registros, o *MySQL* apresentou melhor resultado, porém obteve um pior resultado na base de 1000 000. Nota se que o desempenho de consulta do *MySQL* é diretamente proporcional ao volume de dados de sua base, fato este que não ocorre com o *Firebird.* Ambos SGBDs apresentaram um desempenho muito próximo no quesito de exclusão, tendo o *MySQL* um melhor tempo na base de 1000 registros. Aqui é observado o fato de que o *MySQL* possui um melhor desempenho em comandos de leitura em disco, como seleção e exclusão, em bases de pequeno volume de dados.

Apenas a título de **analogia**, Duraes (2007) executou em sua pesquisa uma *Stored Procedure* no servidor *Firebird* para inserção de 100 000 registros que levou **1,49 minutos** (a *stored procedure* executada nesse projeto foi para inserção de 1000 000 de registros no *Firebird* e levou **1,64 minutos**). Ainda, no trabalho de Matioski (Matioski et all, 2008) foi executado uma exclusão no servidor *MySQL* de 10 000 registros que levou **48,48 segundos** (a exclusão de 1000 000 de registros no *MySQL* deste trabalho levou **1,06 minutos**). Também no mesmo trabalho de Matioski foi executada uma inserção no *MySQL* de 1000 registros que levou **31,51 segundos** (neste trabalho a inserção de 1000 registros no *MySQL* utilizou uma *stored procedure* e levou **32,58 segundos**). Evidentemente que estas pesquisas correlatas foram executadas em cenários diferentes (*hardware,* metodologia, versão, etc) **e não devem ser comparadas com esta**, é notável somente a proximidade dos resultados apresentados.

CONCLUSÃO

Uma análise comparativa de desempenho de banco de dados proporciona uma segurança na escolha do SGBD mais apropriado para os desafios que as corporações enfrentam diariamente. Assim pode se direcionar o *Firebird* para aplicações que envolvem um volume mais elevado de dados, uma constante escrita de registros em disco e aplicações que envolvem processamento e cálculo de muitas operações de entrada e saída no SGBD, como por exemplo, sistemas de caixa financeiro e sistemas de estoque. O *MySQL* pode ser direcionado para aplicações que necessitam somente de agilidade como páginas de internet, formulários de cadastro, que não requerem muitas operações de processamento da informação, sendo assim mais comum apenas para consultas e simples inserções sem o uso de *stored procedure*. O *MySQL* é favorável a essas aplicações ágeis onde a base de dados não apresenta um elevado volume e a mesma não sofre constantes e intensas alterações. Apesar de possuírem tecnologias sofisticadas em suas estruturas, podendo se trabalhar com grandes bases de dados, ambos os SGBDs ainda são classificados de médio porte, sendo aconselhável, portanto, direcionar seu uso a aplicações de mesmo porte.

Todo trabalho de avaliação e análise de desempenho neste ramo é imprescindível na função de expor as características dos sistemas, não somente para divulgação do conhecimento experimentado, mas também nas tarefas de incentivo de melhorias, orientação de usuários e eliminação de informações imprecisas.

Pesquisas como a apresentada neste projeto fornecem um auxilio aos Analistas de Bancos de Dados na escolha do SGBD mais adequado para as soluções empresariais. Naturalmente que muitos outros fatores devem ser relevados durante esta escolha, como a observação prévia de uma modelagem estruturada de dados e a finalidade que a aplicação se destina.

Como já citado para trabalhos futuros, propõe se uma continuidade desta pesquisa na evolução do *benchmark,* incluindo um cenário multiusuário, avaliação de outros SGBDs, comparação de desempenho de objetos de bancos de dados, viabilidade do SGBD para uso de dados meteorológicos, entre outros.

REFERENCIAS BIBLIOGRAFICAS

AXMARK, D.; LARSSON, A.; WIDENIUS, M. **Site de documentação sobre o MySQL**. <http://dev.mysql.com/>, último acesso em 28/10/2010.

CODD, E. F. **Relational Model of Data for Large Shared Data Banks.** (1970) ACM Magazine. <http://portal.acm.org/citation.cfm?id=362685>, ultimo acesso em 10/04/2011.

DATE, C. J**. Introdução a Sistemas de Banco de Dados**, Ed. Campus, 2004

DURAES, S. H. D. **Análise de Desempenho entre os Bancos de Dados PostgreSQL e Firebird na plataforma Windows 2000**. Projeto de Conclusão de Curso. Universidade Estadual de Montes Claros, 2007.

ELMASRI, R; NAVATHE, S. B. **Sistemas de Banco de Dados**, Ed. Pearson Education, 2005

FIREBIRD, About Firebird. **Site da Firebird/Interbase**. <http://www.firebirdsql.org/>, último acesso em 28/10/2010.

GNU, o que é. **Site sobre licenças livres**. <http://www.gnu.org/>, ultimo acesso em 26/04/2011.

GRAY, J. **Database and Transaction Processing Performance Handbook**, 2. ed. San Francisco: Morgan Kauffman Publishers, 1993.

HERNÁNDEZ, P. E GONZALO, J. (2002). **Implementación en C del benchmark de transacciones distribuidas TPC-C**, Bs.C Thesis. Escuela Universitaria Politécnica de Valladolid, Universidad de Valladolid, Spain.

JAVA, sobre Java. **Site da Oracle sobre o Java**. <http://www.java.com/>, ultimo acesso em 26/04/2011.

KOZIEVITH, N. P. **Dados meteorológicos: um estudo de viabilidade utilizando um SGBD em plataforma de baixo custo**, Curitiba, 2005. 70 p. Dissertação (Mestrado em Informática) – Departamento de Informática, Universidade Federal do Paraná.

MySQL, why MySQL, **Site da Oracle sobre o MySQL**. <http://www.mysql.com/>, último acesso em 28/10/2010.

NETBEANS, o que é. **Site da Oracle sobre o Netbeans**. <http://www.netbeans.org/>, ultimo acesso em 26/04/2011.

OSDB. **The Open Source Database Benchmark**. (2001). http://osdb.sourceforge.net/, último acesso em 30/10/2010.

PIRES, C.S.; NASCIMENTO, R.O.; SALGADO, A.C. **Comparativo de Desempenho entre os Bancos de Dados de Código Aberto**. Universidade Federal de Pernambuco (UFPE) 2006.

SANCHES, A. R. **Fundamentos de armazenamento e manipulação de dados** (2006) disponível em <http://www.ime.usp.br/~andrers/aulas/bd2005-1/aula4.html>. Último acesso em 10/04/2011.

SILBERSCHATZ, A.; KORTH, H. F; SUDARSHAN, S.. **Sistema de Banco de Dados**, Ed. Campus, 2006.

SOUZA, M. O.; MATIOSKI, M. E.; NEVES, L. A. P. **Análise de Desempenho dos Bancos MySQL, Postgresql e Firebird: Um estudo de caso.** GESTÃO - Revista Científica de Administração, v. 11, n. 11, jul/dez 2008.

TOSI, D. **Firebird – SuperServer, ClassicServer ou SuperClassic?** <http://www.sinatica.com/blog/br/index.php/artigos/firebird-superserver-classicserver-ou-superclassic> , ultimo acesso em 18/04/2011.

TPC. **Transaction Processing Performance Council**. (2001). <http://www.tpc.org/>, último acesso em 30/10/2010.

WEISS, E. (2008). **Desempenho de Bancos de Dados para Pequenas Arquiteturas.** Universidade Estadual de Londrina - Paraná

BIBLIOGRAFIA CONSULTADA

AZEVEDO, B. **Um exemplo de implementação de banco de dados distribuídos**. (2009) <http://www.webartigos.com>. Último acesso em 10/04/2011

CANTU, H. C. **Conheça o Firebird em 2 Minutos** <http://www.firebirdnews.org/docs/fb2min_ptbr.html > ultimo acesso em 25/05/2011

SILVA, O. K. E. **Um Estudo sobre Sistemas de Bancos de Dados Cliente/Servidor**. Faculdade Paraibana de Processamento de Dados. 2001

VERGINIO, S.J; RIOS, S. R; SILVA, R. T. S. **SimpleAdmin – Interface Gráfica para administração em Banco de Dados Firebird**. Projeto de Conclusão de Curso, Centro Universitário do Sul de Minas. 2006

.

APÊNDICE A – *STORED PROCEDURES* DO MYSQL

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `insercao` $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `insercao`()
BEGIN

declare i int default 1;
declare tt int default 1000;
while (i<=tt) do

insert into matricula (ra, nome, cpf, rg, nome_pai, nome_mae,
   sexo, nacionalidade, estado_civil, cidade, titulo_eleitor,
   estado, e_mail, naturalidade, cor ) values (i,'Joao da Silva',
   123456789, 987654321, 'Jose da Silva', 'Rosana da Silva',
   'Masculino', 'Brasileiro', 'Solteiro', 'Guaratingueta',
   567891234, 'SP', 'joao@joao.com.br', 'Guaratingueta', 'Branco');

set i=i+1;
end while;
END $$
DELIMITER ;


DELIMITER $$
DROP PROCEDURE IF EXISTS `insercao_bruta` $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `insercao_bruta`()
BEGIN

declare i int default 1;
declare tt int default 1000000;
while (i<=tt) do

insert into matricula (ra, nome, cpf, rg, nome_pai, nome_mae,
   sexo, nacionalidade, estado_civil, cidade, titulo_eleitor,
   estado, e_mail, naturalidade, cor ) values (i,'Joao da Silva',
   123456789, 987654321, 'Jose da Silva', 'Rosana da Silva',
   'Masculino', 'Brasileiro', 'Solteiro', 'Guaratingueta',
   567891234, 'SP', 'joao@joao.com.br', 'Guaratingueta', 'Branco');

set i=i+1;
end while;
```

APÊNDICE B – *STORED PROCEDURES* DO FIREBIRD

```
CREATE OR ALTER PROCEDURE INSERCAO
as
declare variable i numeric(15,2);
declare variable a numeric(15,2);
begin
   a=1000;
   i=1;
   WHILE (:i <= :a) DO BEGIN

   insert into matricula (ra, nome, cpf, rg, nome_pai, nome_mae, sexo,
   nacionalidade, estado_civil, cidade, titulo_eleitor, estado, e_mail,
   naturalidade, cor ) values (:i,'Joao da Silva',123456789, 987654321,
   'Jose da Silva', 'Rosana da Silva', 'Masculino', 'Brasileiro',
   'Solteiro',  'Guaratingueta',  567891234,  'SP',  'joao@joao.com.br',  'Guaratingueta',
'Branco');

   i = i + 1;
   END
End



CREATE OR ALTER PROCEDURE INSERCAO_BRUTA
as
declare variable i numeric(15,2);
declare variable a numeric(15,2);
begin
   i=1;
   a=1000000;
   WHILE (:i <= :a) DO BEGIN

   insert into matricula (ra, nome, cpf, rg, nome_pai, nome_mae, sexo,
   nacionalidade, estado_civil, cidade, titulo_eleitor, estado, e_mail,
   naturalidade, cor ) values (:i,'Joao da Silva',123456789, 987654321,
   'Jose da Silva', 'Rosana da Silva', 'Masculino', 'Brasileiro',
   'Solteiro', 'Guaratingueta', 567891234, 'SP', 'joao@joao.com.br',
   'Guaratingueta', 'Branco');


   i = i + 1;
   END
end
```

GLOSSÁRIO

ACM – Association for Computing Machinery

DBA – Database Analyst

DBMS – Database Management System

DCL – Data Control Language

DDL – Data Definition Language.

DER – Diagrama Entidade Relacionamento

DML – Data Manipulation Language.

HDM – Hierarquical Data Model

IDE – Integrated Development Environment

MER – Modelo Entidade Relacionamento

NDS – Network Data Model

OSDB – Open Source Database Benchmark

RDM – Relational Data Model

SGBD – Sistema Gerenciador de Banco de Dados

SGBDR – Sistema Gerenciador de Banco de Dados Relacional

SQL – Structured Query Language

TCL – Transaction Control Language

UML – Unified Modeling Language