UNIVERSIDADE FEDERAL DO PAMPA

ALEX DIAS CAMARGO

INTERLIGANDO BASES DE DADOS DO SISTEMA CONTROLE DE MARCAS E SINAIS UTILIZANDO O *MYSQL CLUSTER*

Bagé
2013

ALEX DIAS CAMARGO

INTERLIGANDO BASES DE DADOS DO SISTEMA CONTROLE DE MARCAS E SINAIS UTILIZANDO O *MYSQL CLUSTER*

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Orientador: Érico Marcelo Hoff do Amaral

Coorientador: Rafael Rodrigues Bastos

Bagé
2013

ALEX DIAS CAMARGO

INTERLIGANDO BASES DE DADOS DO SISTEMA CONTROLE DE MARCAS E SINAIS UTILIZANDO O *MYSQL CLUSTER*

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Trabalho de Conclusão de Curso defendido e aprovado em: 06 de agosto de 2013.

Banca examinadora:

_____

Prof. Me. Érico Marcelo Hoff do Amaral
Orientador
UNIPAMPA

_____

Prof. Dr. Sandro da Silva Camargo
UNIPAMPA

_____

Prof. Me. Carlos Michel Betemps
UNIPAMPA

Dedico este trabalho ao Espírito Santo de Deus por ter me dado a inspiração necessária em todos os momentos. Também ao Prof. Érico Amaral da UNIPAMPA pela orientação e ao Prof. Rafael Bastos da IDEAU pela co-orientação. Da mesma forma a todos meus amigos pela ausência nesta importante fase de minha vida.

"E ainda que tivesse o dom de profecia, e conhecesse todos os mistérios e toda a ciência, e ainda que tivesse toda a fé, de maneira tal que transportasse os montes, e não tivesse amor, nada seria."

1 Coríntios 13:2

RESUMO

Percebe-se claramente uma nova geração da *web*, onde velocidade e confiabilidade são aspectos esperados para qualquer aplicação, logo, um bom Sistema Gerenciador de Banco de Dados Distribuído (SGBDD) deve trabalhar com metas e oferecer distintas vantagens ao modelo de sistema centralizado. A confiabilidade esperada em uma aplicação é atingida com um alto grau de tolerância a falhas. Para satisfazer estas demandas, novas tecnologias e arquiteturas distribuídas têm sido desenvolvidas, testadas e padronizadas. No decorrer desta pesquisa foram abordadas as estratégias de replicação síncronas e assíncronas, divididas com base na latência de propagação das alterações de um objeto para os demais. Em um âmbito de fundamental importância, as falhas em bancos de dados são tratadas desde as mais simples, onde afetam uma única transação, até falhas consideradas maiores onde são afetadas todas as transações. Ao perceber uma falha, um SGBDD confiável deve recuperar todas as tarefas terminadas ou em andamento, tentando minimizar a perda de trabalho, ou seja, a tarefa deve continuar sua execução a partir do último estado válido em que se encontrava antes da falha. Por fim, foi desenvolvido, com o objetivo de realizar testes e verificações, uma solução para interligar os bancos de dados do sistema Controle de Marcas e Sinais usando a ferramenta *MySQL Cluster*. A aplicação deve acessar os dados e replicá-los entre as bases, assegurando a integridade do conteúdo. A estrutura apresentada é baseada em *softwares* de código-fonte aberto.

Palavras-chave: Bancos de Dados Distribuídos; Sistemas Distribuídos.

ABSTRACT

We can clearly perceive a new generation on web, where speed and reliability are needed for any application, therefore, a good Distributed Database Management System (DDBMS) must to work with goals and offer different advantages over centralized system model. The desired reliability in an application is reached with a high level of falt tolerance. To satisfy these necessities, new distributed technologies and architectures has been developed, tested and standarized. During this research, we have dealt with synchronous and asynchronous replication strategies, which was divided based on changes' propagation latency of an object to the others. In a fundamental importance's scope, the failures on databases are dealt since the simplest, where it affects only one transaction, until failures considered biggest, where all transactions are affected. When it perceives a fault, a reliable DDBMS must recover all finished tasks or that are on progress, in order to try to minimize damage of work, in other words, the task must go on its execution, starting from its last valid state before the fault. A solution to connect databases of Marks and Signals Control system was developed by me, using the MySQL Cluster tool. The application it must access data and replicate them among bases, ensuring content integrity. The presented structure is based on open source-code software.

Keywords: Distributed Databases; Distributed Systems.

LISTA DE FIGURAS

LISTA DE SIGLAS

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

API - *Application Programming Interface*

BDD - Banco de Dados Distribuído

CPU - *Central Processing Unit*

FTP - *File Transfer Protocol*

GB - *Gigabyte*

GPL - *General Public License*

HD - *Hard Disk*

IP - *Internet Protocol*

LAN - *Local Area Network*

LTS - *Long Term Support*

MB - *Megabyte*

NDB - *Network Database*

NTI - Núcleo de Tecnologia da Informação

PDF - Portable Document Format

PHP - *Hypertext Preprocessor*

RAM - *Random Access Memory*

RS - Rio Grande do Sul

SGBD - Sistema Gerenciador de Banco de Dados

SGBDD - Sistema Gerenciador de Banco de Dados Distribuído

SPB - *Software* Público Brasileiro

SQL - *Structured Query Language*

VM - *Virtual Machine*

WAN - *Wide Area Network*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A partir do advento das tecnologias de comunicação, o acesso à informação permitiu que dados espalhados geograficamente pudessem ser acessados mais facilmente. A grande capacidade de trabalhar com dados, armazená-los e distribuí-los de maneira eficiente, de modo que se tenha um maior proveito, vem sendo um desafio da computação distribuída. Nestas situações a tolerância a falhas se torna importante, visto que a interrupção de componentes não comprometeria a aplicação em um todo.

Para se alcançar êxito no projeto e implementação de um banco de dados distribuído, deve ser buscado o preenchimento de alguns requisitos apresentados no decorrer deste trabalho. Desta forma, são mostrados conceitos sobre banco de dados distribuído seguidos de sua arquitetura, estratégias de replicação e modelos de atualização.

Os Bancos de Dados Distribuídos (BDDs) visam descentralizar os dados mantidos por um SGBD a fim de se obter maior desempenho e confiabilidade no sistema. Em banco de dados, uma transação é uma unidade básica de computação consistente e confiável, formada por uma sequência de operações de banco de dados executadas como uma ação atômica. Várias dessas transações devem ser executadas de forma concorrente, até mesmo quando ocorrem falhas (OZSU; VALDURIEZ, 2001). A execução de uma transação deve levar o SGBD de um estado consistente a outro estado consistente. A maior parte dos protocolos de tolerância a falhas tem seu foco em falhas de componentes físicos, principalmente falhas de nodos e da rede de comunicação, já que estas são as principais causadoras de defeitos dos componentes do sistema (JALOTE, 1994).

Pretende-se, com esta pesquisa, desenvolver uma estrutura que permita a integração das bases de dados do sistema Controle de Marcas e Sinais mantendo as características essenciais de um sistema distribuído eficiente. Com isso, através da ferramenta *MySQL Cluster*, é proporcionado uma alta disponibilidade dos bancos de dados das cidades que utilizam a aplicação, provendo um serviço tolerante a falhas e protegido contra perda de informações por falha no equipamento.

Além desta Introdução, este trabalho está organizado em mais sete capítulos. O Capítulo 2 introduz um referencial bibliográfico sobre os principais conceitos relacionados a sistemas distribuídos, e também aborda uma concepção geral sobre bancos de dados distribuídos. Já no Capítulo 3, são apresentados alguns trabalhos e ferramentas relacionados

ao proposto, com seus objetivos e funcionalidades. O Capítulo 4 relata a metodologia utilizada para o desenvolvimento deste trabalho. Nos Capítulos 5 e 6 estão especificados o ambiente de simulação utilizado, seguido dos testes realizados, bem como a análise dos resultados apresentados. Por fim, no Capítulo 7 conclui-se o trabalho evidenciando-se suas contribuições e, no Capítulo 8, futuras atividades de pesquisa que poderão serem desenvolvidas.

1.1 Problema de pesquisa

Atualmente os municípios que utilizam o Controle de Marcas e Sinais, na sua maneira convencional, trabalham com suas informações em bancos de dados centralizados e sem nenhuma integração. Procura-se, através desta pesquisa, desvendar o seguinte problema:

"Como interligar as bases de dados do Controle de Marcas e Sinais fazendo com que trabalhem como um único sistema?"

1.2 Hipótese

Este trabalho apresenta um estudo sobre Sistemas Distribuídos, bem como uma maior abordagem em SGBDDs, propondo a aplicação de seus conceitos para um caso real, onde se justifica pela oportunidade de por em prática a teoria estudada. Utilizando técnicas de replicação de dados, a solução oferece alta disponibilidade e tolerância a falhas. Também serve de apoio às forças de segurança, pois terão um acesso mais amplo ao acervo de marcas e sinais registrados pelos produtores rurais, ajudando no combate de crimes como o abigeato, um tipo de crime que envolve o furto de animais.

1.3 Objetivo geral

O objetivo geral deste trabalho é prover uma solução que tem por finalidade sincronizar bases de dados do sistema Controle de Marcas e Sinais de modo a operarem distribuidamente, fazendo uma interligação entre os municípios que utilizam a ferramenta.

1.4 Objetivos específicos

- Estudar aspectos de implementação computacional aplicados à distribuição de dados;

- Modificar o código-fonte do sistema Controle de Marcas e Sinais para atuar de maneira descentralizada;

- Configurar o banco de dados distribuído utilizando a ferramenta *MySQL Cluster*;

- Analisar a estrutura através de testes funcionais aplicados aos módulos que foram desenvolvidos no Controle de Marcas e Sinais;

- Exemplificar o uso da estrutura abordada com uma solução para interligar as cidades de Bagé/RS e Hulha Negra/RS.

2 REFERENCIAL BIBLIOGRÁFICO

Neste capítulo é realizado um estudo do estado da arte do tema, onde são descritos os conceitos básicos sobre sistemas distribuídos, bem como os principais aspectos para se conseguir uma solução ideal, mostrando os defeitos comuns neste tipo de sistema. Também apresenta uma fundamentação teórica sobre bancos de dados distribuídos expondo aspectos como vantagens e desvantagens da sua utilização, estratégias e modelos de replicação de dados.

2.1 Sistemas distribuídos

Na computação atual, é visto claramente uma nova geração da *web*, onde velocidade e confiabilidade são aspectos esperados para qualquer aplicação *mobile*, *desktop* ou *server*.

Segundo Tanenbaum e Steen (2007), "Um sistema distribuído é um conjunto de computadores independentes entre si que se apresenta a seus usuários como um sistema único e coerente." Então, formam um conjunto de processos localizados entre computadores pessoais, dispositivos móveis, estações de trabalho, servidores, etc, conectados entre si através de uma Rede Local (LAN) ou Rede Global (WAN). Seu uso tem acompanhando o crescimento das redes computacionais, sob uma demanda de melhorias em diferentes aspectos como interoperabilidade, portabilidade e flexibilidade.

Na visão de Dantas (2005), "Os sistemas distribuídos, sob o aspecto de arquitetura de máquinas para execução de aplicativos, devem ser vistos como configurações com grande poder de escala pela agregação dos computadores existentes nas redes convencionais." Para proporcionar tal agregação, vêm sendo desenvolvidas diferentes tecnologias e arquiteturas, posteriormente testadas e padronizadas.

Um sistema distribuído proporciona ao usuário a impressão de que o *software* está completamente independente e que o *hardware* são computadores independentes. Isso somente é possível graças a um conjunto de *hardware* e *software* preparados para a interação de programas, o que significa a adaptação tanto por parte do sistema operacional que é capaz de operar sobre várias máquinas, quanto por parte das máquinas capazes de se comunicar entre si (VERÍSSIMO;RODRIGUES, 2001).

Em um âmbito geral, o *hardware* desempenha a importante função de atuar como coordenador na comunicação dos sistemas distribuídos juntamente com protocolos de

comunicação entre as máquinas. Também, o *software* deve disponibilizar uma boa estrutura para que o sistema operacional possa trabalhar aproveitando o máximo do *hardware* distribuído disponível e mantendo a transparência para o usuário final.

2.1.1 Sistema distribuído ideal

Um Sistema Distribuído é uma coleção de computadores autônomos, ligados por uma rede, com *software* projetado para produzir uma facilidade de computação integrada (COULORIS;DOLLIMORE;KINDBERG, 2005). Para o êxito no projeto e implementação de sistemas distribuídos, deve ser buscado preencher alguns requisitos, mostrados na Figura 1.

Figura 1: Sistema distribuído ideal

Fonte: (TAING, 2007)

Os requisitos de um sistema distribuído ideal ilustrados na Figura 1 apresentam as seguintes características:

- heterogeneidade: trabalhar com diferentes tipos de *hardware* e sistema operacional;
- transparência: ocultar para o usuário, na medida do possível, a distribuição do sistema;

- abertura: prover serviços com regras padronizadas, por exemplo o uso do XML;
- concorrência: manter um estado consistente com suporte a múltiplos acessos executados simultaneamente;
- segurança: proteger os recursos compartilhados e acessos;
- escalabilidade: permitir o acréscimo de novos recursos sob demanda;
- tolerância a falhas: oferecer alta disponibilidade de recursos mesmo ocorrendo uma falha.

Na visão de Tanenbaum e Steen (2007), "O fato de ser possível montar sistemas distribuídos não quer dizer necessariamente que seja uma boa idéia". Nisto vemos que, um bom sistema distribuído deve trabalhar com metas e oferecer distintas vantagens ao modelo de sistema centralizado. Neste conceito, o uso da aplicação pode ser dividido em máquinas, lugares e plataformas diferentes, comunicando-se por componentes que gerenciam as chamadas remotas.

2.1.2 Sistemas distribuídos na *web*

Para Tanenbaum e Steen (2007), "A arquitetura de sistemas distribuídos baseados na *web* não apresenta diferenças fundamentais em relação à de outros sistemas distribuídos." A *web* fornece acesso a informações distribuídas semelhante a uma arquitetura cliente/servidor.

Segundo Lima (2003), "A *web* proporciona ligações de um documento localizado em um servidor a outro, mantendo a localização real e o método de acesso invisível para o usuário. Esse modelo permite a descentralização da *web* e contribui para sua escalabilidade."

Os clientes podem acessar os servidores através de um navegador a partir de qualquer máquina conectada à *Internet* independente do tipo de documento, do sistema operacional ou do *hardware*. Uma observação importante, por exemplo, é um *web site* que pode ser visto como sendo não um único banco de dados, mas um sistema de informação composto pela integração de diversas fontes de dados juntamente com estruturas de navegação complexas (BARBOSA, 2001).

2.1.3 Defeitos em sistemas distribuídos

No entendimento de Trindade (2003), "As definições de sistema distribuído e dos diferentes tipos de defeitos previstos nesse tipo de sistema são importantes para que se possa construir um modelo de simulação de defeitos consistente." A maioria dos protocolos para tolerância a falhas tem seu foco em falhas de componentes físicos, principalmente falhas de nodos e da rede de comunicação, já que estes são os principais causadores de defeitos dos componentes do sistema (JALOTE ,1994). A classificação de defeitos em sistemas distribuídos se enquadra nas seguintes categorias: omissão, colapso, temporização, resposta e arbitrários. Tal classificação é apresentada a seguir (CRISTIAN, 1991):

- defeito de omissão: ocorre quando um componente não responde a uma solicitação;

- defeito de colapso (*crash*): ocorre quando, depois da primeira omissão em produzir um resultado, um componente omite a produção de resultados subsequentes até ser reiniciado;

- defeito de temporização: ao ocorrer este tipo de defeito, um componente responde corretamente mas a resposta está fora do intervalo de tempo especificado, podendo ser fornecida muito cedo ou muito tarde, também é chamado de defeito de desempenho;

- defeito de resposta: com esse tipo de defeito, um componente produz resultados incorretos para determinadas solicitações ou realiza uma transição de estado incorretamente;

- defeito arbitrário: faz com que um componente se comporte de maneira totalmente imprevisível.

Com exceção dos defeitos de resposta, os tipos de defeitos formam uma hierarquia. O modo de defeito mais simples é o de colapso, no qual, o componente se comporta de maneira benigna, pois simplesmente pára quando ocorre o defeito (TRINDADE, 2003).

2.2 Bancos de dados distribuídos

Banco de Dados Distribuído (BDD) pode ser definido como uma coleção de múltiplos bancos de dados logicamente inter-relacionados e distribuídos sobre uma rede de computadores. Um Sistema de Gerenciamento de Bancos de Dados Distribuídos (SGBDD) pode ser caracterizado como um sistema que permite o gerenciamento dos BDDs e efetua a

distribuição de forma transparente aos usuários (OZSU; VALDURIEZ, 2001).

Os SGBDDs nos proporcionam uma maior confiabilidade por possuírem componentes compartilhados que eliminam pontos únicos de falha. Isso significa que se houver falha em um único servidor ou em um *link* de comunicação, apenas o ponto que falhou vai ficar inativo e não todo o sistema, assim o usuário continua tendo acesso a uma parte dos dados (BORTOLINI, 2004).

As propriedades ACID, cujas iniciais formam o nome dado a elas, são as propriedades básicas que consagram o modelo de transações (CARVALHO, 2008):

- atomicidade: uma transação é uma unidade atômica de processamento que deve ser executada integralmente, ou totalmente desfeita;

- consistência: a execução de uma transação deve levar o banco de dados de um estado consistente a outro estado consistente;

- isolamento: a execução de uma transação não pode ser afetada por outras transações sendo executadas concorrentemente;

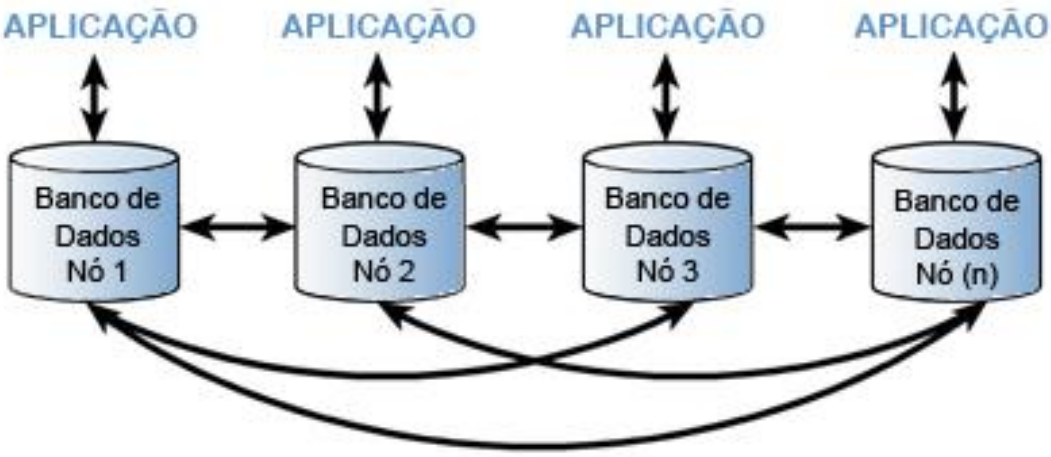
- durabilidade: os efeitos de uma transação confirmada não podem ser desfeitos.

A principal motivação do uso de bases de dados distribuídas é o que ocorre basicamente com a *Internet*; onde há compartilhamento de dados, porém, com gerenciamento e armazenamento locais. Em uma base de dados distribuída os nodos estão conectados entre si, formando um mesmo plano global. Nesse caso cada nodo possui *software* e *hardware* particulares, com memória, processador, discos e sistema operacional próprios.

Segundo Silberchatz et. al (2009), "Os BDDs diferem dos sistemas paralelos porque não compartilham memória, processadores ou discos. Mesmo quando as arquiteturas paralelas implementam a ausência de compartilhamento, o *software* gerenciador continua sendo único."

No agregado de máquinas, o administrador deve instalar o *software* em vários servidores, configurar e operar cada um deles garantindo que os dados, tais como o conteúdo de *web sites*, seja o mesmo em cada sistema. Isso pode ser um desafio e é a razão primária pela qual as tecnologias de replicação de dados estão se tornando importantes para o mercado (POLYSERVE, 2000). A Figura 2 ilustra como funciona um banco de dados distribuído com carga de trabalho compartilhada.

Figura 2: Compartilhamento da carga de trabalho



Fonte: (MCOBJECT, 2013)

As bases de dados distribuídas devem apresentar algumas características importantes, porém nem todas elas são obrigatórias (BRITO apud OZSU; VALDURIEZ, 2009):

- distribuição: pode ser considerada, horizontal, onde as linhas de uma tabela são distribuídas, ou vertical, onde as colunas de uma tabela são distribuídas;

- heterogeneidade: os *softwares* que gerenciam os bancos de dados são diversos e os esquemas variam entre eles;

- autonomia: diz respeito à independência de cada parte integrante do sistema, onde os componentes podem ter seus próprios modelos, linguagens e semântica;

- interoperabilidade: os sistemas devem ser capazes de enviar requisições e receber repostas uns dos outros.

2.2.1 Arquitetura

Na arquitetura de um SGBDD é definida sua estrutura, onde são identificados os componentes que serão utilizados e lhes são atribuídas às devidas funções.

Segundo Ozsu e Valduriez (2001), "Para projetar um modelo arquitetônico de um SGBDD deve-se levar em consideração três características: a autonomia de sistemas locais, sua distribuição e sua heterogeneidade." Os sistemas de banco de dados podem ser

classificados, de acordo com suas arquiteturas, como (SILBERCHATZ; KORTH; SUDARSHAN, 2009):

- centralizados: quando o SGBD reside em um único sistema computacional e não interage com outros sistemas, podendo ser monousuário ou multiusuário;

- cliente-servidor: quando o SGBD reside em um sistema computacional chamado servidor, e interage com outros sistemas computacionais, que são os clientes;

- paralelos: quando utiliza em paralelo diversas CPUs e discos, com objetivo de ganhar velocidade no processamento e alta escalabilidade.

- distribuídos: quando o SGBD reside em diversos sistemas computacionais, podendo estes variar de porte e capacidade. Podem ser conectados através de redes de alta ou baixa velocidade;

- móveis: quando o SGBD e o próprio banco de dados são aplicados em ambientes de computação móveis, que utilizam celulares, *palm-tops*, *tablets* e outros equipamentos análogos conectados via redes de computação sem fio.

Em arquiteturas distribuídas, pode haver uma distribuição de componentes do sistema de banco de dados, de controle em execução de tarefas ou de ambos. Portanto, dependendo dos critérios de distribuição, podemos ter arquiteturas distintas de sistemas de banco de dados distribuídos (LEITE, 2004).

2.2.2 Replicação de dados síncrona

Em uma replicação síncrona todas as cópias de dados existentes em um sistema são analisadas no instante da sincronização. As alterações feitas em alguma réplica do banco de dados são imediatamente aplicadas a todas as outras instâncias dentro de uma transação (COSTA, 2010). Tal tipo de replicação é recomendada para aplicações comerciais, tendo em vista seu alto poder de consistência. Pode ser executada pelo SGBD, ou em alguns casos, por um gerente distribuído (monitor de transações).

Quando se fala em replicação, o principal aspecto a ser tratado é a sincronia entre as réplicas. Visto que, modificações feitas nelas devem surtir efeito em todo o banco de dados distribuído. Em sistemas baseados no modelo cliente-servidor, um único servidor pode atender a vários clientes, e um aumento da carga de trabalho nele pode acarretar tempos de resposta

elevados. Em tais circunstâncias, replicar os dados ou servidores pode aumentar o desempenho. A replicação também pode melhorar a disponibilidade da informação quando os processadores deixam de funcionar ou a rede sofre uma interrupção (AMIR, 1995).

A principal característica da replicação síncrona é que a atualização acontece em tempo real, ou seja, quando um dado é atualizado em uma base de dados, o mesmo é atualizado em todas as outras bases. A grande vantagem deste tipo de replicação é a alta disponibilidade dos dados que são disponibilizados instantaneamente para todas as bases (BORTOLINI, 2004).

Para Lorêdo et. al (2004), "Na replicação síncrona se aumenta, contudo, a complexidade da identificação de conflitos entre leituras e escritas, uma vez que leituras são vistas apenas localmente." Porém, Kemme e Alonso (2000) propuseram uma solução direta para esse problema: "Abortar as transações que realizavam leituras quando houvesse uma operação de escrita conflitante."

2.2.3 Replicação de dados assíncrona

Havendo uma replicação, é necessário que cópias de um mesmo objeto (base de dados) estejam em computadores independentes uns dos outros, preenchendo requisitos de tolerância a falhas. Apesar da replicação ser uma idéia intuitiva, rapidamente compreensível, sua implementação é difícil. Replicar um serviço em sistemas distribuídos exige que cada réplica do serviço mantenha um estado consistente (DEFAGO; SCHIPER; SERGENT, 1998).

Sempre há um grau de atraso entre a transação efetuada na base de dados e a atualização dos resultados desta transação nas réplicas. Na replicação assíncrona o processo de replicação ocorre assíncrono a transação originada, ou seja, o período de latência deve ser maior que zero (BURRETA, 1997). Na replicação assíncrona as modificações executadas nos nodos são enviadas em um segundo passo. Por se tratar de uma transação separada, pode ocorrer em minutos, horas ou dias após as modificações.

No entendimento de Lorêdo et. al (2004), "A replicação assíncrona de bancos de dados é uma estratégia adequada quando a conexão entre os servidores envolvidos não é permanente." A sincronia entre as réplicas de dados dependerá da ocorrência da comunicação entre as unidades clientes e a unidade servidora (COSTA, 2010). Segundo Kemme e Alonso (2000), "É frequente deixar a cargo da aplicação usuária a tarefa de manter a consistência.”

As propriedades ACID de uma transação não são garantidas quando se implementa

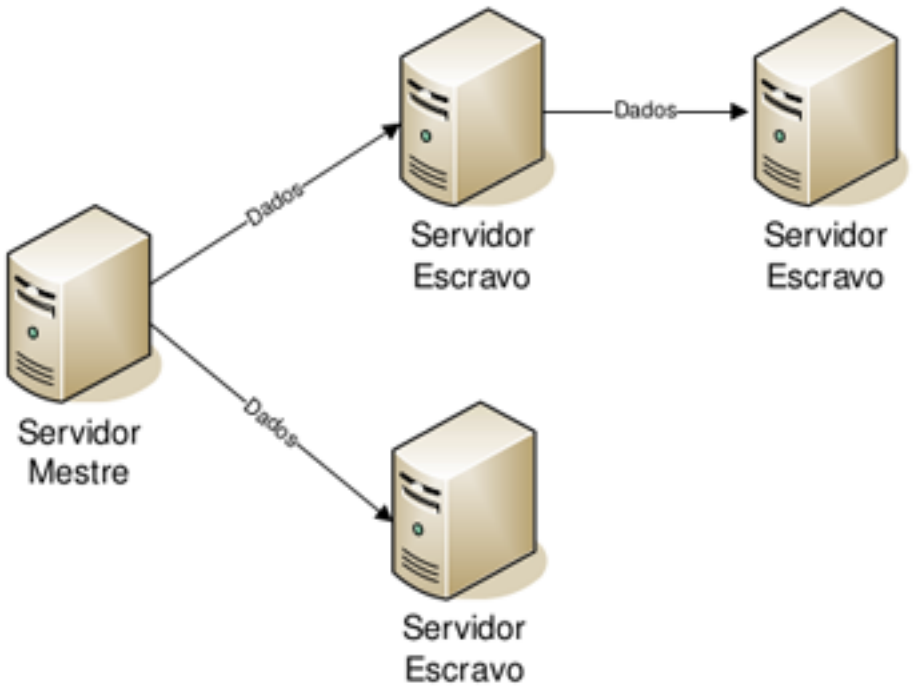
replicação assíncrona com possibilidade de atualizações em qualquer servidor (BURETTA, 1997). Na replicação assíncrona, com exceção ao modelo ponto-a-ponto, as transações de atualização ocorrem em somente um nodo, a sua replicação para outros nodos poderá ser postergada até que a comunicação entre os nodos for restabelecida (MANFREDINI, 2001).

2.2.4 Modelo de atualização mestre-escravo

No modelo mestre-escravo (*master-slave*), um dos objetos é considerado como mestre e as alterações só podem ser realizadas nesse objeto. Os demais objetos são considerados como escravos e só podem ser utilizados para leitura (MELO, 2010). Na visão de Oliveira (2007), "No modelo mestre-escravo, a replicação nos escravos é sempre idêntica à base de dados do mestre. O modelo é muito simples, de fácil implementação, porém, de uso limitado".

Pode-se dizer que o modelo mestre-escravo tem como principal característica uma unidade primária de distribuição dos dados podendo ser centralizada ou distribuída, fragmentada ou não fragmentada (BORTOLINI, 2004). A Figura 3 ilustra seu funcionamento.

Figura 3: Modelo de atualização mestre-escravo



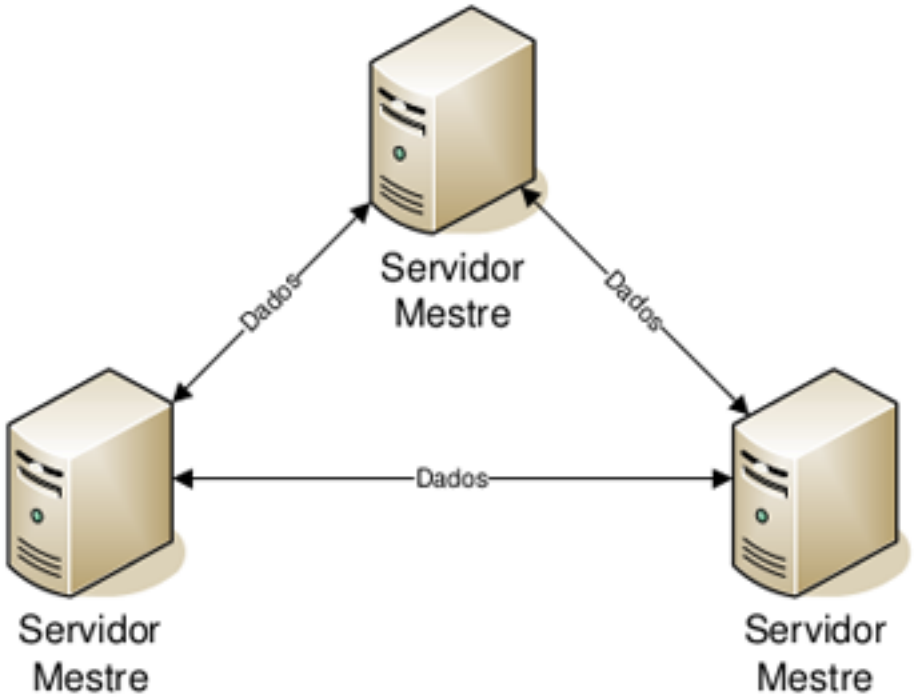
Fonte: (MELO, 2010)

2.2.5 Modelo de atualização multimestre

No modelo multimestre (*multimaster*) é proporcionado uma maior disponibilidade dos dados, em contrapartida apresenta uma complexidade maior. O objetivo a atingir nos sistemas multimestre é a convergência dos dados, ou seja, conseguir que todas as réplicas fiquem idênticas. Em sistemas replicados, o termo convergência é usado, porque em um determinado instante de tempo as réplicas podem apresentar diferenças, porém, com o passar do tempo, elas irão convergir e ficar iguais (OLIVEIRA, 2007).

Em um modelo multimestre, todos os objetos podem ser modificados e essas modificações são transmitidas para os demais, portanto o fluxo das informações pode partir de qualquer objeto participante para os demais e por isso a complexidade de controle da replicação é maior (MELO, 2010). Na Figura 4 é ilustrado seu funcionamento.

Figura 4: Modelo de atualização multimestre



Fonte: (MELO, 2010)

2.2.6 Recuperação

Recuperação de falhas em banco de dados consiste basicamente em restaurar o banco de dados para um estado consistente após a ocorrência de falhas. Quando uma falha ocorre, uma ou mais transações podem estar ativas e ainda não terem sofrido *commit* (MANFREDINI, 2001).

Para Bell e Grimson (1992), "A habilidade de assegurar a consistência de um banco de dados na presença de falhas imprevisíveis de componentes de *hardware* e de *software* é uma característica essencial de todos os sistemas de gerenciamento de banco de dados." As falhas em bancos de dados variam desde as mais simples, afetando uma única transação, até falhas maiores onde afetam todas as transações em um local determinado.

Após a ocorrência de uma falha, um sistema confiável deve recuperar todas as tarefas terminadas ou em andamento, tentando minimizar a perda de trabalho, ou seja, a tarefa deve continuar sua execução a partir do último estado válido em que se encontrava antes da falha. Para garantir esta recuperação é preciso que certas informações sobre o estado de execução da tarefa sejam recuperáveis (SILVA, 2003):

- O estado do sistema visto por todas as aplicações, isto é, os bancos de dados de cada aplicação;

- O estado local da tarefa, por exemplo, as variáveis dos programas, cursores, etc, utilizados por mais de um *step*;

- O estado global da computação, ou seja, quais *steps* já foram executados, quais não foram, etc.

Em um processo de recuperação, o mecanismo gerenciador verifica se contém em seu *log* um registro de *commit* ou *abort* da transação. Se este registro existir, significa que o gerenciador tomou uma decisão antes da falha e ele deverá implementar esta decisão (MANFREDINI, 2001). Para este procedimento funcionar de maneira esperada, é necessário que os registros de *commit* sejam gravados no *log* antes do envio da mensagem de *commit*. Estes registros são usados durante o processo onde são desfeitas as transações abortadas, e também durante o procedimento no qual o sistema é reinicializado após uma falha, restaurando-o a um estado consistente.

3 TRABALHOS CORRELATOS

A problemática envolvida na integração de bancos de dados é bastante discutida atualmente. Na elaboração deste trabalho, tornou-se possível a identificação e estudo de alguns trabalhos e ferramentas com objetivos dentro do propósito desta pesquisa, como é apresentado a seguir.

3.1 *MySQL Proxy*

O *MySQL Proxy* é uma aplicação que se comunica através da rede, utilizando o protocolo de rede *MySQL*, permite a comunicação entre um ou mais servidores *MySQL*, e um ou mais clientes do *MySQL* (MYSQL, 2013).

A ferramenta age de forma transparente, ou seja, a aplicação cliente nem o *MySQL* sabem que o *MySQL Proxy* está sendo utilizado. Esta característica facilita a utilização da ferramenta e expande o leque de possibilidades tanto para o desenvolvedor quanto para quem administra o *MySQL* (PICHILIANI, 2009).

No *MySQL Proxy* os pedidos de gravação são enviados para os bancos de dados mestres e de leitura. Estas solicitações são equilibradas entre os bancos de dados dos escravos e, se não houver um escravo ativo, todos os pedidos são direcionados para o banco de dados mestre (OLIVEIRA, 2012).

A aplicação *MySQL Proxy*, atualmente, é uma versão *Alpha* (ainda em construção e testes) sendo o seu uso inapropriado para ambientes de produção.

3.2 *PgCluster*

PgCluster é uma extensão para *PostgreSQL* que oferece replicação síncrona entre dois ou mais servidores mestres. É composto por três tipos de servidores distintos: o servidor de replicação (*Replication Server*), o balanceador de carga (*Load Balance Server*) e o servidor *PostgreSQL* em si. O balanceador de carga não é necessário apenas para realizar o balanceamento de carga entre os servidores, mas também para criar um *cluster* de alta disponibilidade (PGCLUSTER, 2005). O *PgCluster* possui duas principais funções (TONINI, 2009):

- balanceamento de carga: um módulo recepciona os comandos de consulta e transações, distribuindo da forma mais conveniente levando em consideração a carga (capacidade de

processamento comprometida) dos servidores;

- alta disponibilidade: é garantida através da replicação total dos dados, então se um servidor sair do ar, os demais podem responder em seu lugar, desde que, pelo menos um servidor de replicação continue em funcionamento.

Como a ferramenta trabalha de forma síncrona entre os servidores de banco de dados, é possível realizar consultas em bases locais. Com isso se aumenta, consideravelmente, o desempenho da aplicação que requisita as informações.

3.3 *Slony-I*

O *Slony-I* é a única solução para replicação assíncrona no *PostgreSQL* sendo mantida. Também é apontada pelo *web site* do *PostgreSQL* como a melhor solução para replicação assíncrona e vem inclusa no pacote de instalação oficial do *PostgreSQL*. As atualizações são feitas diretamente na cópia primária. Quando uma atualização é feita, um *trigger* é executado. O *trigger* salva um *log* com as alterações feitas (LIMA, 2011).

O *Slony-I* é um sistema de replicação "mestre de vários escravos". Suas principais características incluem (BROWNE, 2011):

- Pode replicar dados entre diferentes versões do *PostgreSQL*;

- Pode replicar dados entre sistemas operacionais e *hardwares* diferentes entre si;

- Permite diferentes servidores de banco de dados trabalhando como mestre.

O seu *download* pode ser feito através do *web site* <http://main.slony.info/downloads/>, atualmente se encontra na versão 2.2 e funciona a partir da versão 8.3 do *PostgreSQL*.

4 METODOLOGIA

Para o desenvolvimento deste trabalho foi de fundamental importância que conceitos e definições sobre bancos de dados distribuídos fossem claramente descritos. Com esse objetivo foi imprescindível uma revisão bibliográfica a respeito do estado da arte para os temas apresentados, dando enfoque em sistemas gerenciadores de bancos de dados distribuídos.

Além disso, buscou-se compreender o funcionamento do sistema Controle de Marcas e Sinais a fim de se projetar a distribuição de sua base de dados, onde, através de uma engenharia reversa de *software*, foi possível compreender as principais funcionalidades da aplicação. Desta forma, foram verificados os requisitos necessários para viabilizar a estrutura de banco de dados distribuído e quais os obstáculos para o seu desenvolvimento e integração com o sistema escolhido.

Neste trabalho foram realizados levantamentos para identificar quais as tecnologias seriam adequadas para a solução do problema em questão, bem como sua viabilidade. Em seguida, a partir das tecnologias escolhidas, foram desenvolvidas estratégias para a manipulação de dados dos bancos de dados distribuídos. Também foram pesquisadas metodologias e técnicas computacionais visando uma alta disponibilidade e tolerância a falhas, o que é de fato a proposta deste trabalho. Foram avaliadas a replicação dos dados e a consistência da base de dados dos nodos após a falha e seu reingresso ao banco de dados distribuído.

Como uma maneira de validar as pesquisas, foi desenvolvida a implementação da ferramenta *MySQL Cluster* aplicada ao sistema Controle de Marcas e Sinais onde é feita a verificação nos mecanismos implementados.

5 IMPLEMENTAÇÃO E TESTES

Este capítulo tem como objetivo apresentar o sistema Controle de Marcas e Sinais, além de expor passo a passo a solução proposta através do *MySQL Cluster*. Um fato que deve ser mencionado é a disponibilidade de versões do *MySQL Cluster* para, praticamente, qualquer sistema operacional, como o *Linux, FreeBSD, Windows* e *Mac OS X*. Também, o Controle de Marcas e Sinais pode funcionar em diferentes servidores de aplicações *web*, como o *Apache, Nginx* e *iPlanet*, desde que possuam o PHP e o SGBD *MySQL* instalados. Além disso, tanto o Controle de Marcas e Sinais quanto o *MySQL Cluster* são produtos regidos pela licença GPL (*General Public License*), portanto possuem código-fonte aberto.

5.1 Descrição da implementação

Podemos afirmar que, com todo o embasamento teórico apresentado nesta pesquisa, foi possível montar uma estrutura de implementação bastante próxima de uma possível utilização real. Entretanto, sem levar em consideração a latência na transmissão dos dados quando transportados entre as cidades. Porém, o foco deste trabalho é a aplicação de técnicas para se obter um nível de confiabilidade e disponibilidade do sistema Controle de Marcas e Sinais. Assim, eliminamos o uso centralizado das bases de dados da aplicação, o que é inviável pela indisponibilidade de acesso e troca de informações pelos municípios.

É importante ressaltar que as tabelas não possuirão nenhum tipo de fragmentação e apenas serão replicadas. Este protótipo fará uso da estratégia de replicação síncrona juntamente com o modelo de atualização multimestre, a fim de se manter as réplicas sempre idênticas.

5.2 Ambiente de simulação

Os experimentos distribuídos realizados neste trabalho foram desenvolvidos em um ambiente virtualizado através da ferramenta gratuita e de código-fonte aberto *Virtual Box* <https://www.virtualbox.org/wiki/Linux_Downloads>, consistindo em:

- 1 Computador hospedeiro das máquinas virtuais: Processador *Intel Core i3*, RAM 8 GB e HD 320 GB;
- 2 Servidores de aplicação (virtualizados): Processador *Intel Core i3*, RAM 256 MB e HD 8 GB;

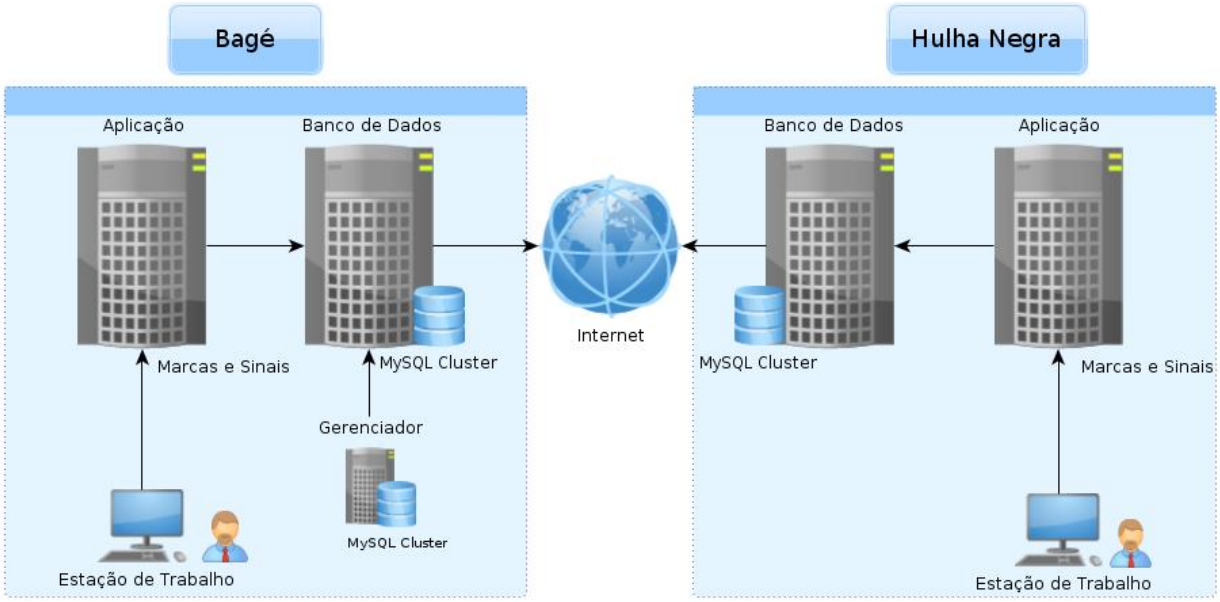
- 1 Servidor gerenciador (virtualizado): Processador *Intel Core i3*, RAM 256 MB e HD 8 GB;
- 2 Servidores de banco de dados (virtualizados): Processador *Intel Core i3*, RAM 2 GB e HD 8 GB.

Todas as máquinas possuem instalado o sistema operacional *Linux Ubuntu Server 10.04 LTS 32-bit*, exceto a máquina hospedeira que possui o *Linux Mint Desktop 14 64-Bit*. Um detalhe importante é que os servidores de banco de dados onde é executado o *MySQL Cluster* exigem um número mais elevado de memória RAM para seu funcionamento (no mínimo 2GB).

Para demonstrar a operacionalidade da solução, foi disponibilizada uma rede local com os endereços IPs 192.168.0.50/24, 192.168.0.55/24, 192.168.0.65/24, 192.168.0.75/24 e 192.168.0.85/24. Optou-se por colocar o servidor que gerencia o banco de dados distribuído na cidade de Bagé/RS por ter um maior número de habitantes e espera-se uma infraestrutura computacional mais robusta.

Com isso, podemos dimensionar o ambiente desejado para os experimentos, exemplificando uma solução para interligar os municípios de Bagé/RS e Hulha Negra/RS. A Figura 5 ilustra a solução apresentada.

Figura 5: Interligação das cidades de Bagé/RS e Hulha Negra/RS



Fonte: (DADOS PRIMÁRIOS, 2013)

5.3 Controle de Marcas e Sinais

O conceito do *Software* Público Brasileiro (SPB) é utilizado como um dos alicerces para definir a política de desenvolvimento, distribuição e uso de *software* pelo setor público do Brasil (CARDOSO; MEFFE; MARTINS, 2011).

O sistema Controle de Marcas e Sinais foi desenvolvido pelo Núcleo de Tecnologia da Informação (NTI) da Prefeitura Municipal de Bagé, e está disponível para *download*, desde junho de 2009, através do portal SPB <http://www.softwarepublico.gov.br>, juntamente com outras soluções desenvolvidas por órgãos públicos do Executivo, Legislativo e Judiciário. É importante salientar que não há custo com licença de uso, porém, todas as melhorias efetuadas no *software* são incentivadas a serem compartilhadas. A disponibilidade de *download* através do portal SPB deu uma ampla visibilidade a aplicação e, atualmente, vários municípios do Brasil já a utilizam.

Basicamente, o sistema Controle de Marcas e Sinais, como o próprio nome diz, funciona como uma catalogação digital de produtores rurais do município e suas respectivas marcas ou sinais. Após a realização do cadastro inicial, o produtor recebe um certificado de propriedade, validando o processo e evitando falsificações. Com isso, suas informações ficam disponíveis através do módulo "Consulta", geralmente contido dentro do *web site* oficial do município, tornando possível o acesso a base de dados via *Internet* e, por exemplo, servindo de apoio às forças de segurança.

O sistema trás grandes benefícios aos municípios, pois os registros ficam armazenados digitalmente, substituindo os modelos físicos que sofrem uma fácil degradação, além de apresentar uma consulta mais demorada.

5.3.1 Visão geral

O sistema Controle de Marcas e Sinais foi desenvolvido, em quase sua totalidade, na linguagem de programação PHP. Funciona via *web* (é sugerido o servidor *Apache*) e possui versões para os SGBDs *PostgreSQL* e *MySQL*. Optou-se pela segunda opção pela compatibilidade, pois a distribuição das bases de dados será realizada através do *MySQL Cluster*.

A aplicação é dividida em dois módulos: o primeiro deles tem como objetivo fornecer aos usuários do sistema as opções de cadastro e consulta dos produtores rurais e suas respectivas marcas ou sinais; o segundo módulo tem como objetivo somente permitir uma consulta pública

dos registros contidos no sistema, portanto, não será abordado neste trabalho.

Para acessar o Controle de Marcas e Sinais, basta indicar o endereço IP do servidor de aplicação através de um navegador *web*. O controle de acessos ao programa é feito através de um *login* e uma senha previamente cadastrados na base de dados. A Figura 6 mostra a tela inicial da aplicação.

Figura 6: Página inicial do sistema

Fonte: (DADOS PRIMÁRIOS, 2013)

No menu visto na Figura 9 temos todas as opções de acesso do módulo de cadastro, percebe-se que o programa apresenta uma navegação simples e objetiva. O rodapé pode ser personalizado, através do código-fonte, de acordo com os dados do município. Por ser uma aplicação bastante leve, pode ser acessada através de conexões mais lentas como, por exemplo, um acesso móvel através de um *smartphone* ou *tablet*. As Figuras 7, 8, 9 e 10 mostram as telas de cadastro do Controle de Marcas e Sinais.

Figura 7: Cadastro de localidade

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 8: Cadastro de produtor

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 9: Cadastro de marca

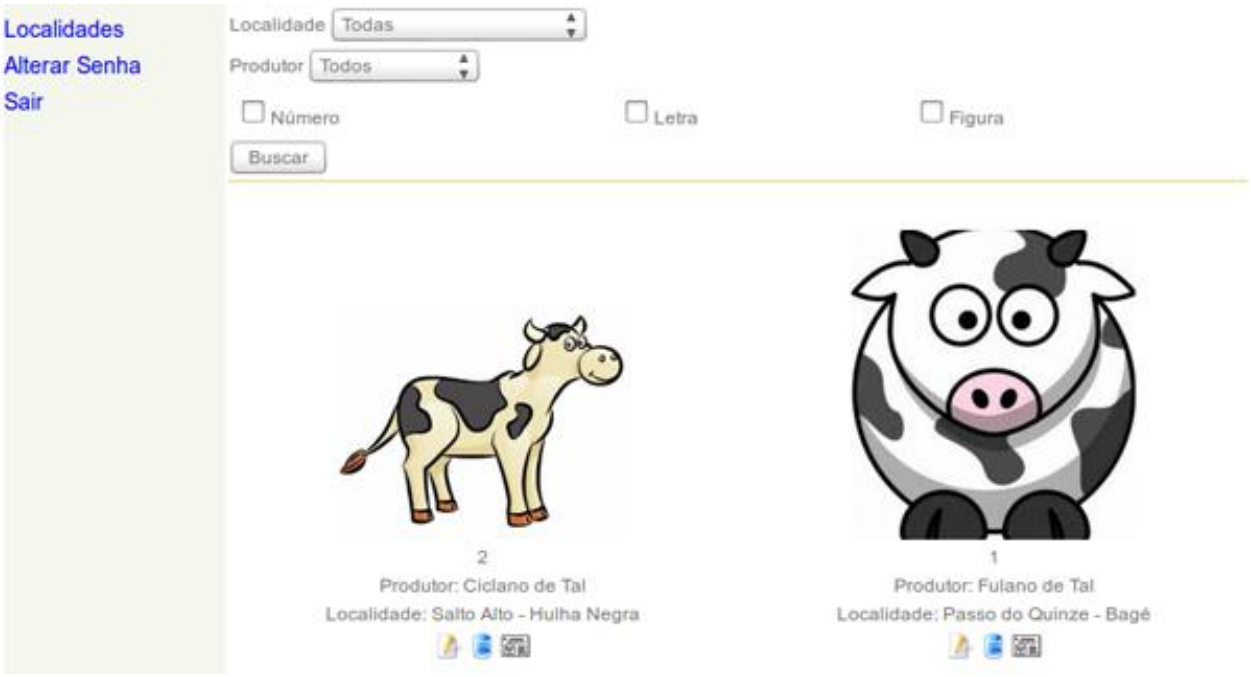
Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 10: Cadastro de sinal

Fonte: (DADOS PRIMÁRIOS, 2013)

Nesta etapa, é exibida a seção de pesquisa de registros do Controle de Marcas e Sinais. Um ponto interessante desse módulo é a opção de pesquisa por caracterização de conteúdo (Número, Letra ou Figura). Com a estrutura proposta já podemos ver as informações de ambos os municípios, como é mostrado na Figura 11.

Figura 11: Pesquisa de registros



Fonte: (DADOS PRIMÁRIOS, 2013)

Outra funcionalidade interessante é a validação do processo de cadastro juntamente a prefeitura municipal. O sistema emite um certificado no formato PDF para que os produtores rurais possam comprovar seus registros. Um detalhe do documento é apresentado na Figura 12.

Figura 12: Certificado emitido pelo sistema

Fonte: (DADOS PRIMÁRIOS, 2013)

5.3.2 Modificações no código-fonte

Após o *download* do sistema Controle de Marcas e Sinais através de sua comunidade <http://www.softwarepublico.gov.br/ver-comunidade?community_id=11791260> no portal SPB, para que funcionasse apropriadamente de maneira distribuída, foram necessárias algumas modificações no código-fonte da aplicação. É importante destacar que, por ser um *software* livre e de código-fonte aberto, as modificações são legais.

Como na estrutura proposta a aplicação funcionará intermunicipalmente, para começar, foi modificado o arquivo de conexão com o SGBD. Então, no caso de o servidor local ficar inoperante, a aplicação assume automaticamente o servidor de outro município. A Figura 13 exibe o arquivo de conexão referente às cidades exemplificadas de Bagé/RS e Hulha Negra/RS.

Figura 13: Arquivo de conexão ao SGBD

```
10  //Acessa o servidor de Bagé/RS
11  $dbServer = 'mysql:dbname=cms_v2;host=192.168.0.55;charset=utf8';
12  $dbLog = 'root';
13  $dbPass = 'bage';
14  $db = new DataBase($dbServer, $dbLog, $dbPass);
15
16  //Acessa o servidor de Hulha Negra/RS
17  if(!$db){
18  $dbServer2 = 'mysql:dbname=cms_v2;host=192.168.0.75;charset=utf8';
19  $dbLog2 = 'root';
20  $dbPass2 = 'hulhanegra';
21  $db = new DataBase($dbServer2, $dbLog2, $dbPass2);
22  }
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Uma considerável modificação realizada foi a inclusão do campo "Cidade" no sistema. A Figura 14 mostra o trecho incluído no código-fonte para a busca de cidades cadastradas no banco de dados.

Figura 14: Busca de cidades

```
294         while ( $linha = $db->fetchArray($sql) ) {
295             $idmarca = $linha['idmarca'];
296             $localidade = $linha['localidade'];
297             $produtor = $linha['nome'];
298             $numero = $linha['numero'];
299             $caminho = $linha['caminho'];
300
301             $idcidade = $linha['idcidade'];
302             $sql2 = "select cidade from cms_cidades where idcidade = $idcidade";
303             $sql2 = $db->query($sql2);
304             $linha2 = $db->fetchArray($sql2);
305             $cidade = utf8_decode($linha2['cidade']);
306
```

Fonte: (DADOS PRIMÁRIOS, 2013)

O mesmo foi feito no cadastro de produtores rurais, tendo em vista que eles estão ligados a uma localidade específica. A alteração é mostrada na Figura 15.

Figura 15: Adição do campo "Cidade" nas localidades dos produtores

```
<td><label>Localidade </label>
    <select name="localidade" size="1">
        <?php
    $sql = "select idlocalidade, localidade, idcidade from cms_localidades order by localidade";
    $sql = $db->query($sql);

        while ($linha = $db->fetchArray($sql) ) {
                $idlocalidade = $linha['idlocalidade'];
                $localidade = utf8_decode($linha['localidade']);
                $idcidade = $linha['idcidade'];
                $sql2 = "select cidade from cms_cidades where idcidade = $idcidade";
                $sql2 = $db->query($sql2);
                $linha2 = $db->fetchArray($sql2);
                $cidade = utf8_decode($linha2['cidade']);


            if ($local == $localidade)
            echo "<option value=" . $idlocalidade . " selected>".$localidade." - ".$cidade."</option>";
        else
        echo "<option value=" . $idlocalidade . ">".$localidade." - ".$cidade."</option>";
    }
    ?>
</select></td>
</tr>
```

Fonte: (DADOS PRIMÁRIOS, 2013)

A solução proposta neste trabalho também se preocupou com a disponibilidade das imagens em todos os servidores de aplicação, para isto, foi incluído ao sistema uma classe para redimensionamento em PHP juntamente com um script que utiliza protocolo de transferência FTP (*File Transfer Protocol*). A Figura 16 mostra o trecho do código-fonte com as alterações feitas.

Figura 16: Redimensionamento e transferência de imagens

```
//redimensiona a imagem e salva no servidor
$img = WideImage::load($dir.$foto);
$img = $img->resize(520, 520, 'inside')->crop('center', 'center', 500, 500);
$img->saveToFile($dir.$foto);
$caminho =  $dir.$foto;
$caminho2 = $dir2.$foto;

//copia a imagem para o servidor da outra cidade
$fconn = ftp_connect($fdados["fhost"]);
ftp_login($fconn, $fdados["fusuario"], $fdados["fsenha"]);
ftp_put($fconn, $caminho2, $caminho, FTP_BINARY);
ftp_close($fconn);
```

Fonte: (DADOS PRIMÁRIOS, 2013)

As alterações supra apresentadas fizeram com que o sistema Controle de Marcas e Sinais passasse a mostrar os registros de ambos os municípios, o que não acontecia em sua versão

oficial disponibilizada para *download*. A replicação e recuperação das bases de dados foi implementada diretamente na ferramenta *MySQL Cluster*.

5.4 *MySQL Cluster*

O *MySQL* é um SGBD que utiliza a linguagem SQL (*Structured Query Language*). Surgiu na Suécia, criado por David Axmark, Allan Larsson e Michael Widenius, cuja primeira versão foi lançada em 1998. A estrutura proposta nesta pesquisa trabalha com o *MySQL Cluster*, que é uma tecnologia que permite a distribuição de SGBDs *MySQL*, mantida atualmente pela *Oracle*.

A ferramenta utiliza uma arquitetura *shared-nothing*, na qual cada nodo é independente e auto-suficiente. A arquitetura *shared-nothing* permite que o sistema trabalhe com um mínimo de requisitos específicos de *hardware* e *software* (MYSQL, 2013). A ideia principal é prover alta disponibilidade e um alto desempenho.

Um dos pontos fortes do *MySQL Cluster* é que ele pode ser executado em diferentes tipos de *hardware* e não tem requisitos incomuns a este respeito, além de uma grande quantidade de memória RAM disponível, devido ao fato de carregar completamente as tabelas para a memória principal *(in-memory database*). Os sistemas operacionais de *host* não necessitam de quaisquer módulos, serviços ou aplicativos incomuns para suportar a ferramenta (MYSQL, 2013).

O *MySQL Cluster* se baseia no mecanismo de armazenamento *NDBCLUSTER* (também chamado NDB - *Network Database*), bastando somente indicar o "*ENGINE = NDBCLUSTER*" na criação ou alteração das tabelas, que automaticamente será feita a replicação das mesmas, como mostra a Figura 17.

Figura 17: Mecanismo de armazenamento *NDBCLUSTER*

```
CREATE TABLE `City` (
  `ID` int(11) NOT NULL auto_increment,
  `Name` char(35) NOT NULL default '',
  `CountryCode` char(3) NOT NULL default '',
  `District` char(20) NOT NULL default '',
  `Population` int(11) NOT NULL default '0',
  PRIMARY KEY  (`ID`)
) ENGINE=NDBCLUSTER DEFAULT CHARSET=latin1;
```

Fonte: (MYSQL, 2013)

Basicamente, o *MySQL Cluster* é dividido da seguinte maneira (MYSQL, 2013):

- nodo de gerenciamento: o papel deste tipo de nodo é gerir os outros nodos dentro do *MySQL Cluster*, também realiza funções como fornecimento de dados de configuração, inicialização e interrupção dos nodos;

- nodo de dados: armazena os dados e compartilha automaticamente as tabelas de maneira transparente;

- nodo SQL: é um servidor *MySQL* (*mysqld*) que acessa os dados distribuídos.

5.4.1 Componentes Principais

Os componentes que fazem parte do *MySQL Cluster* são, essencialmente, os processos *ndbd*, *ndb_mgmd*, *ndb_mgm* e o *mysqld*. Entendendo a operacionalidade de cada um deles, foi possível projetar e configurar a ferramenta para funcionar juntamente com o Controle de Marcas e Sinais. As funcionalidades destes principais processos são descritas a seguir (MYSQL, 2013):

- *ndb_mgmd*: é o processo que lê o arquivo de configuração do *MySQL Cluster* e distribui essa informação para todos os nodos que o solicitem, também mantém um registro das atividades;

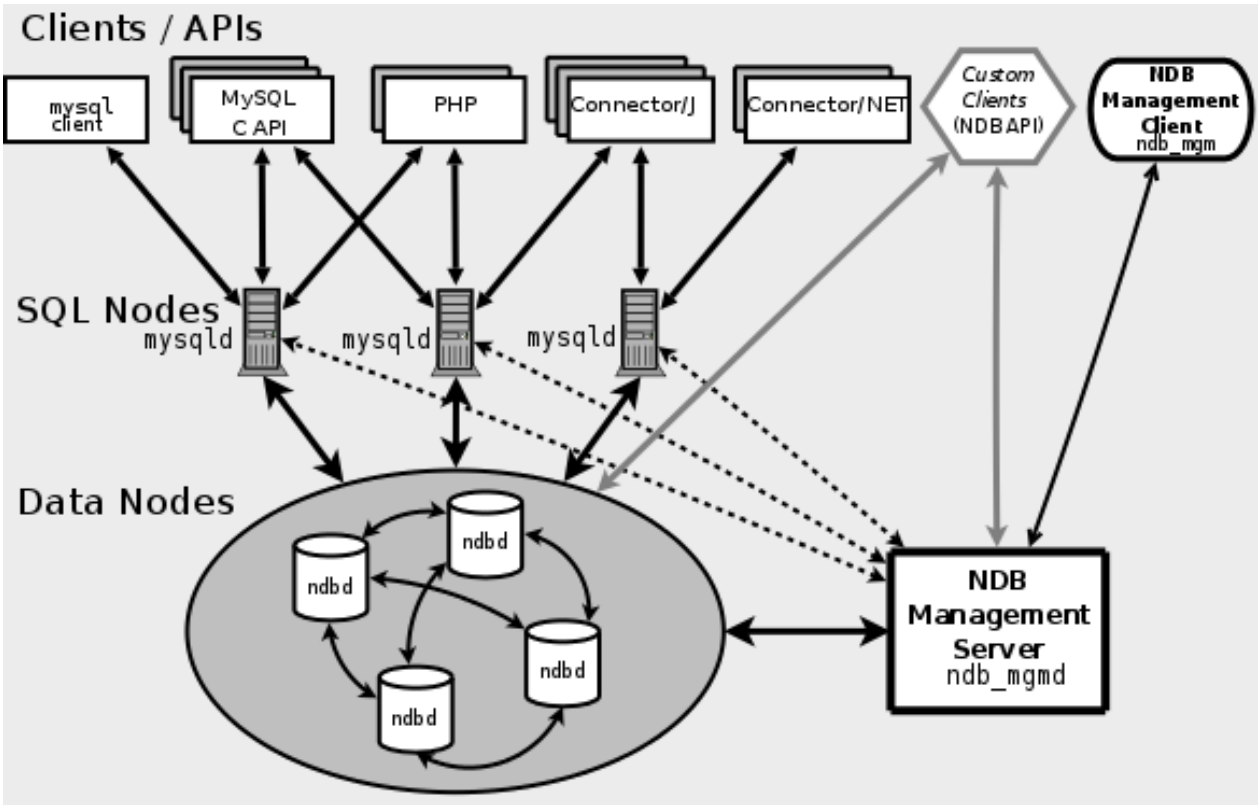
- *ndb_mgm*: é o processo cliente do gerenciador de distribuição. Inicializa os *backups* e executa funções administrativas;

- *ndbd*: é o processo utilizado para lidar com os dados das tabelas, usa o mecanismo de armazenamento NDB *Cluster*;

- *mysqld*: também conhecido como *MySQL Server*. Gerencia o acesso ao diretório do *MySQL* que contém as tabelas do banco de dados.

O fato do *MySQL Cluster* ser um programa de código-fonte aberto, facilita o estudo e engenharia reversa da ferramenta. O seu uso pode ser aliado a programas desenvolvidos em diferentes linguagens de programação devido a sua facilidade de conexão através de APIs específicas. A Figura 18 exemplifica os componentes da ferramenta.

Figura 18: Componentes principais do *MySQL Cluster*



Fonte: (MYSQL, 2013)

De maneira hierárquica, vemos primeiramente o NDB *Management Node* (NDB *Management Server*), que é o servidor utilizado para a configuração e monitoramento do *MySQL Cluster* juntamente com seu cliente NDB *Node* (NDB *Management Client*). Também são mostrados os SQL *Nodes*, que são os SGBDs *MySQL* que recebem as requisições das aplicações (*web sites*, sistemas, etc). A inicicialização do *MySQL Cluster* é feita de maneira bastante simples, pois cada nodo é inicializado separadamente.

Empregando as regras necessárias para a distribuição dos dados, a ferramenta vai ser completamente independente, podendo operar com suas transações normalmente com as tabelas replicadas, garantindo assim a confiabilidade da aplicação.

5.4.2 Instalação

A partir da estrutura especificada foi implementada a solução com o *MySQL Cluster* para validar a funcionalidade e a viabilidade da mesma. Toda a configuração é feita através de arquivos do tipo texto no qual contém as informações necessárias para se especificar as máquinas que farão parte do banco de dados distribuído. Foi atribuído o endereço IP 192.168.0.50 para o

servidor "gerenciadorbg", IP 192.168.0.55 para o servidor "bancodedadosbg" e IP 192.168.0.75 para o servidor "bancodedadoshl".

Inicialmente, foi realizado o *download* através do site oficial do *MySQL Cluster* <http://dev.mysql.com/downloads/cluster/>. Optou-se pela versão 7.2.10 por ser a mais atualizada durante o desenvolvimento deste trabalho. Na próxima etapa é feita a extração o arquivo baixado. A Figura 19 exibe o procedimento realizado no servidor "gerenciadorbg".

Figura 19: Extração do instalador

```
bage@gerenciadorbg:~$ ls
mysql-cluster-gpl-7.2.10-linux2.6-i686.tar.gz
bage@gerenciadorbg:~$ sudo tar -C /usr/local -xzvf mysql-cluster-gpl-7.2.10-linux2.6-i686.tar.gz
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Neste momento, dentro do diretório supra criado, é configurado o principal arquivo do *MySQL Cluster*, o "config.ini". Nele é indicado os endereços IP das máquinas participantes do banco de dados distribuído, como indica a Figura 20.

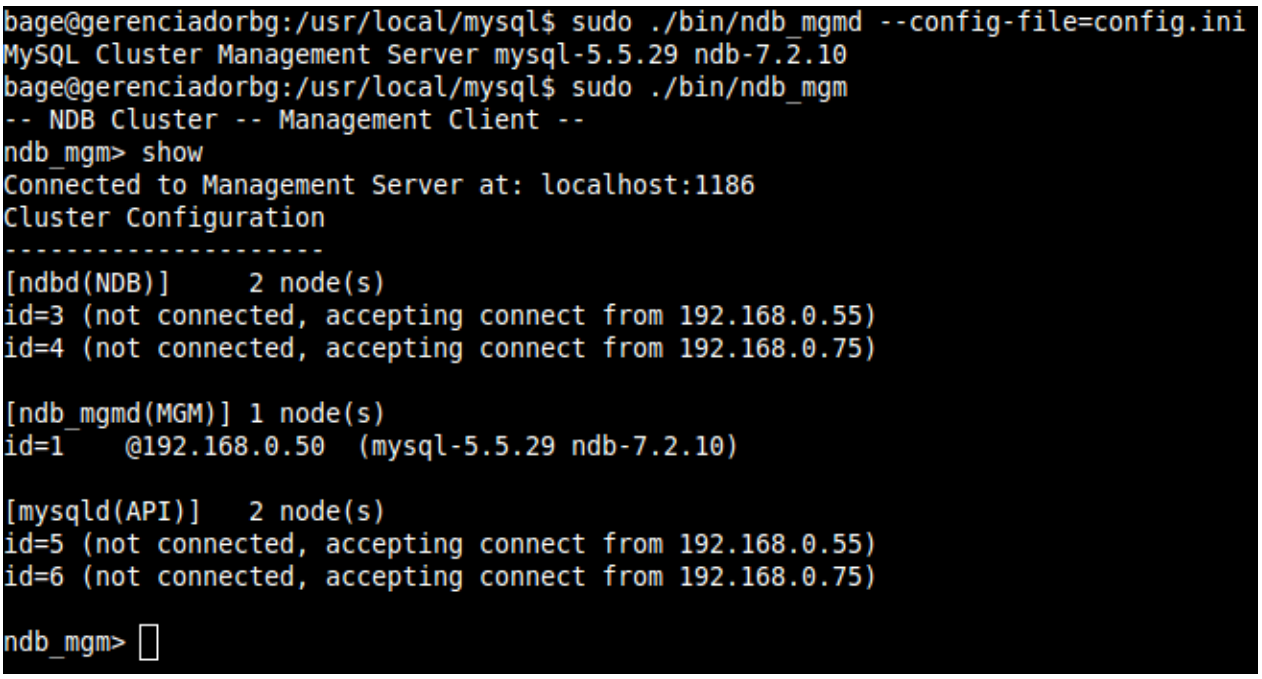
Figura 20: Arquivo "config.ini"

```
bage@gerenciadorbg:/usr/local/mysql$ cat config.ini
# config.ini content
[ndb_mgmd]
NodeId=1
HostName=192.168.0.50 #IP gerenciador
DataDir=/var/lib/mysql-cluster

[ndbd default]
DataDir=/var/lib/mysql-cluster
NoOfReplicas=2
MaxNoOfConcurrentOperations=32000
MaxNoOfAttributes = 10000
MaxNoOfOrderedIndexes=512
DataMemory=1G
IndexMemory=500M

[ndbd]
NodeId=3
HostName=192.168.0.55 #IP cliente gerenciador (Bage)

[ndbd]
NodeId=4
HostName=192.168.0.75 #IP cliente gerenciador (Hulha Negra)

[mysqld]
HostName=192.168.0.55 #IP SGBD (Bage)

[mysqld]
HostName=192.168.0.75 #IP SGBD (Hulha Negra)
bage@gerenciadorbg:/usr/local/mysql$
```

Fonte: (DADOS PRIMÁRIOS, 2013)

A partir de agora, executamos o serviço de gerenciamento "ndb_mgmd" já carregando o arquivo de configuração "config.ini". Então, é exibido o console de administração do *MySQL Cluster*, mostrado na Figura 21. É importante ressaltar que o serviço está indicando a aceitação de conexão dos endereços IP configurados, porém eles ainda não foram conectados.

Figura 21: Console de administração



Fonte: (DADOS PRIMÁRIOS, 2013)

Feitas as devidas configurações no servidor de gerenciamento do *MySQL Cluster* "gerenciadorbg", então é configurado os demais serviços de armazenamento de dados. Os procedimentos seguintes foram realizados nos servidores "bancodedadosbg" e "bancodedadoshl".

Outro arquivo de fundamental importância do *MySQL Cluster* é o "my.cnf", exibido na Figura 22. Nele é indicado a porta em que a ferramenta irá trabalhar, seguido do endereço IP do console de gerenciamento.

Figura 22: Arquivo "my.cnf"

```
root@bancodedadosbg:/etc/mysql# cat my.cnf
# my.cnf content
[client]
port=3306
socket=/tmp/mysql.sock

[mysqld]
port=3306
socket=/tmp/mysql.sock
ndbcluster
ndb-connectstring=192.168.0.50 #IP gerenciador

[mysql_cluster]
ndb-connectstring=192.168.0.50 #IP gerenciador
root@bancodedadosbg:/etc/mysql#
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Nesta etapa, através do serviço "ndbd", é possível testar a conexão do banco de dados distribuído, como mostra a Figura 23.

Figura 23: Teste de conexão

```
root@bancodedadosbg:/usr/local/mysql# ./bin/ndbd
2013-07-21 15:06:14 [ndbd] INFO     -- Angel connected to '192.168.0.50:1186'
2013-07-21 15:06:14 [ndbd] INFO     -- Angel allocated nodeid: 3
root@bancodedadosbg:/usr/local/mysql#
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Então, o console de administração da ferramenta já mostra o seu serviço cliente como registrado, conforme exibe a Figura 24.

Figura 24: Exibição do cliente do gerenciador conectado

```
root@gerenciadorbg:/usr/local/mysql# ./bin/ndb_mgmd --config-file=config.ini
MySQL Cluster Management Server mysql-5.5.29 ndb-7.2.10
root@gerenciadorbg:/usr/local/mysql# ./bin/ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3    @192.168.0.55  (mysql-5.5.29 ndb-7.2.10, starting, Nodegroup: 0, Master)
id=4    @192.168.0.75  (mysql-5.5.29 ndb-7.2.10, starting, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.50  (mysql-5.5.29 ndb-7.2.10)

[mysqld(API)]   2 node(s)
id=5 (not connected, accepting connect from 192.168.0.55)
id=6 (not connected, accepting connect from 192.168.0.75)

ndb_mgm>
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Para finalizar, basta ativar o SGBD *MySQL* incluído na ferramenta *MySQL Cluster*. A Figura 25 exibe a confirmação da instalação do serviço, com isso, o console administrativo também mostra a API "*mysqld*" conectada com sucesso, como ilustra a Figura 26.

Figura 25: Ativação do SGBD *MySQL*

```
root@bancodedadosbg:/usr/local/mysql# ./scripts/mysql_install_db  --user=mysql
Installing MySQL system tables...
OK
Filling help tables...
OK
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 26: API *"mysqld"* conectada

```
ndb_mgm> show
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3    @192.168.0.55  (mysql-5.5.29 ndb-7.2.10, Nodegroup: 0)
id=4    @192.168.0.75  (mysql-5.5.29 ndb-7.2.10, Nodegroup: 0, Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.50  (mysql-5.5.29 ndb-7.2.10)

[mysqld(API)]   2 node(s)
id=5    @192.168.0.55  (mysql-5.5.29 ndb-7.2.10)
id=6    @192.168.0.75  (mysql-5.5.29 ndb-7.2.10)

ndb_mgm>
```

Fonte: (DADOS PRIMÁRIOS, 2013)

5.5 Testes realizados

Definidos os cenários que implementam a estrutura, a metodologia para a condução do experimento e as configurações necessárias nas ferramentas, passou-se então aos testes propriamente ditos.

O estudo incluiu alguns dos problemas e métodos relacionados na bibliografia, porém somente alguns foram implementados, em função da grandeza de soluções. Nos testes realizados não se preocupou com a velocidade de processamento e tempo de reposta. Porém, focou-se nas funcionalidades da solução, visto que os experimentos ocorreram em ambientes emulados, com a utilização de VMs.

5.5.1 Operações básicas (*select, insert, update* e *delete*)

Foram submetidas ao sistema uma série de instruções de consulta, inserção, atualização e remoção para verificar se as operações foram executadas corretamente e se o sistema se comporta de fato, de acordo com a configuração estabelecida. A aplicação mostrou-se bastante leve e possui uma estrutura muito simples. A Figura 27 apresenta algumas consultas executadas na base de dados do Controle de Marcas e Sinais.

Figura 27: Consultas na base de dados

```
mysql> select * from cms_sinais;
+---------+--------+------------+--------------+------------------+---------------+-------+
| idsinal | numero | idprodutor | idlocalidade | caminho          | data_cadastro | ativo |
+---------+--------+------------+--------------+------------------+---------------+-------+
|       6 |      1 |          3 |            1 | sinal/sinal.png  | 2013-07-07    |     1 |
|       7 |      2 |          5 |            2 | sinal/sinal2.jpg | 2013-07-07    |     1 |
+---------+--------+------------+--------------+------------------+---------------+-------+
2 rows in set (0,00 sec)

mysql> select * from cms_logacessos;
+----+---------------+---------+------------+----------+
| id | ip            | usuario | data       | hora     |
+----+---------------+---------+------------+----------+
| 34 | 192.168.0.100 | 123     | 2013-07-07 | 16:02:00 |
| 33 | 192.168.0.100 | 123     | 2013-07-06 | 15:40:00 |
| 32 | 192.168.0.100 | 123     | 2013-07-05 | 14:38:00 |
+----+---------------+---------+------------+----------+
3 rows in set (0,00 sec)

mysql> select * from cms_usuarios;
+----+-------+----------------------------------+-------+------------+
| id | login | senha                            | nome  | tentativas |
+----+-------+----------------------------------+-------+------------+
|  1 |   123 | 202cb962ac59075b964b07152d234b70 | teste |          0 |
+----+-------+----------------------------------+-------+------------+
1 row in set (0,00 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Através da linguagem de programação PHP, utilizada pela aplicação, é possível a inserção de registros no banco de dados. Foi inserida uma imagem de nome "sinal_3.jpg". Tal operação é mostrada nas Figuras 28 e 29.

Figura 28: Inserção de registro via *web*

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 29: Exibição da inserção realizada

```
mysql> select * from cms_sinais;
+---------+--------+------------+--------------+-------------------+---------------+-------+
| idsinal | numero | idprodutor | idlocalidade | caminho           | data_cadastro | ativo |
+---------+--------+------------+--------------+-------------------+---------------+-------+
|       6 |      1 |          3 |            1 | sinal/sinal.png   | 2013-07-07    |     1 |
|       7 |      2 |          5 |            2 | sinal/sinal2.jpg  | 2013-07-07    |     1 |
|       9 |      3 |          3 |            1 | sinal/sinal_3.jpg | 2013-07-13    |     1 |
+---------+--------+------------+--------------+-------------------+---------------+-------+
3 rows in set (0,00 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

O sistema permite a alteração dos dados cadastrados diretamente em sua interface *web* e, por funcionar via servidor, qualquer alteração feita no banco de dados reflete automaticamente em toda a aplicação. A Figura 30 exibe a opção de alteração no nome de uma localidade no Controle de Marcas e Sinais. Neste caso foi alterado o nome de uma localidade de "Passo do Quinze" para "Passo do 15", a Figura 31 apresenta o registro alterado.

Figura 30: Alteração de registro

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 31: Exibição da alteração realizada

```
mysql> select * from cms_localidades;
+--------------+-------------+----------+-------+
| idlocalidade | localidade  | idcidade | ativo |
+--------------+-------------+----------+-------+
|            1 | Passo do 15 |        1 |     1 |
|            2 | Salto Alto  |        2 |     1 |
+--------------+-------------+----------+-------+
2 rows in set (0,00 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Por questões de segurança, foi substituída a exclusão definitiva de registro por uma indicação de "ativo" no banco de dados. A Figura 32 mostra a exclusão de um sinal cadastrado no sistema, seguido da Figura 33 que ilustra como ficou a tabela após a operação.

Figura 32: Exclusão de registro

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 33: Tabela "cms_sinais"

```
mysql> select * from cms_sinais;
+---------+--------+------------+-------------+------------------+--------------+-------+
| idsinal | numero | idprodutor | idlocalidade | caminho          | data_cadastro | ativo |
+---------+--------+------------+-------------+------------------+--------------+-------+
|       6 |      1 |          3 |           1 | sinal/sinal.png  | 2013-07-07    |     1 |
|       7 |      2 |          5 |           2 | sinal/sinal2.jpg | 2013-07-07    |     1 |
|       9 |      3 |          3 |           1 | sinal/sinal_3.jpg | 2013-07-13    |     0 |
+---------+--------+------------+-------------+------------------+--------------+-------+
3 rows in set (0,00 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

5.5.2 Replicação entre as bases de dados

Na solução deste trabalho, por se tratar de uma estratégia de replicação síncrona, as bases de dados permanecem sempre atualizadas, exceto os casos onde há interrupção de comunicação no instante da operação. A ferramenta *MySQL Cluster* replica os registros incluídos, alterados ou excluídos de um determinado banco de dados.

Inicialmente, foi feita a alteração do "*engine*" das tabelas do banco de dados do Controle de Marcas e Sinais, com isso a ferramenta *MySQL Cluster* as reconhece e efetua a replicação dos dados. A Figura 34 mostra o comando SQL utilizado para alterar a tabela "cms_sinais", o mesmo procedimento foi executado nas demais tabelas.

Figura 34: Alteração do "*engine*" da tabela "cms_sinais"

```
mysql> ALTER TABLE cms_sinais ENGINE = NDBCLUSTER;
Query OK, 3 rows affected (0.95 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Fonte: (DADOS PRIMÁRIOS, 2013)

A primeira situação a ser exemplificada é a sincronia dos dados ocorrida sem nenhum conflito e nenhuma falha de comunicação entre os nodo de origem e nodo de destino. Ao inserir um novo registro na tabela "cms_localidades" no servidor de banco de dados com IP 192.168.0.55, o mesmo é sincronizado com o servidor de banco de dados de IP 192.168.0.75. As Figuras 35 e 36 exibem a operação realizada. Caso a conexão entre dois determinados nodos não se estabeleça no momento em que deveria ocorrer a replicação dos dados, o *MySQL Cluster* executa a replicação em uma próxima conexão com o nodo de destino.

Figura 35: Inserção de registro no servidor de IP 192.168.0.55

```
mysql> insert into cms_cidades (idcidade, cidade, ativo) values (3, 'Teste Replicação', 1);
Query OK, 1 row affected (0.00 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 36: Registro replicado no servidor de IP 192.168.0.75

```
mysql> SELECT * FROM cms_cidades;
+----------+-------------+-------+
| idcidade | cidade      | ativo |
+----------+-------------+-------+
|        1 | Bagé        |     1 |
|        2 | Hulha Negra |     1 |
+----------+-------------+-------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM cms_cidades;
+----------+--------------------+-------+
| idcidade | cidade             | ativo |
+----------+--------------------+-------+
|        1 | Bagé               |     1 |
|        2 | Hulha Negra        |     1 |
|        3 | Teste Replicação   |     1 |
+----------+--------------------+-------+
3 rows in set (0.00 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

5.5.3 Recuperação após uma falha de nodo

Outra situação para se demonstrar é a sincronização dos dados após alguma falha de comunicação. Neste caso as bases de dados somente ficarão sincronizadas depois que for possível estabelecer a conexão do nodo danificado com algum outro nodo da rede. Consequentemente, a base de dados danificada permanecerá com os dados desatualizados perante os outros nodos.

Neste cenário, um dos servidores de banco de dados foi desligado. A ferramenta indica que houve uma falha de comunicação em um determinado ciclo de sincronização e, depois de restabelecida a comunicação entre os nodos de origem e destino, é efetuada a sincronia dos dados sem haver conflito na replicação. A Figura 37 mostra o gerenciador do MySQL *Cluster* indicando

que um dos servidores está fora de operação. Durante esse período foi inserido um registro em um dos nodos de dados, como ilustra a Figura 38, logo, se percebe na Figura 39, que o servidor foi recuperado, finalizando com a Figura 40 exibindo a atualização automática.

Figura 37: Indicação de interoperabilidade

```
2013-07-20 21:36:17 [MgmtSrvr] ALERT    -- Node 4: Node 6 Disconnected
2013-07-20 21:36:21 [MgmtSrvr] INFO     -- Node 4: Communication to Node 6 opened
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 38: Registro inserido em um nodo de dados

```
mysql> insert into cms_cidades (idcidade, cidade, ativo) values (4, 'Teste Recuperação', 1);
Query OK, 1 row affected (0,01 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 39: Indicação de recuperação de um nodo

```
2013-07-20 21:37:12 [MgmtSrvr] INFO     -- Nodeid 6 allocated for API at 192.168.0.75
2013-07-20 21:37:12 [MgmtSrvr] INFO     -- Node 6: mysqld --server-id=0
2013-07-20 21:37:12 [MgmtSrvr] INFO     -- Node 4: Node 6 Connected
```

Fonte: (DADOS PRIMÁRIOS, 2013)

Figura 40: Atualização após recuperação

```
mysql> select * from cms_cidades order by idcidade;
+----------+---------------------+-------+
| idcidade | cidade              | ativo |
+----------+---------------------+-------+
|        1 | Bagé                |     1 |
|        2 | Hulha Negra         |     1 |
|        3 | Teste Replicação    |     1 |
|        4 | Teste Recuperação   |     1 |
+----------+---------------------+-------+
4 rows in set (0.00 sec)
```

Fonte: (DADOS PRIMÁRIOS, 2013)

6 RESULTADOS E DISCUSSÕES

Com o ambiente de simulação elaborado para o experimento foi possível identificar, na prática, os conceitos previamente estudados neste trabalho. Procurou-se, à medida do possível, documentar os procedimentos realizados de maneira simples e objetiva. O sistema Controle de Marcas e Sinais mostrou-se bastante descomplicado em sua configuração. Basicamente, funciona como qualquer outro sistema *web* e com as devidas alterações foi possível migrar, facilmente, algumas de suas funcionalidades para trabalhar intermunicipalmente, tendo em vista que a aplicação foi projetada para trabalhar de maneira centralizada em cada município.

Um dos requisitos importantes impostos ao sistema foi a replicação dos dados e recuperação após uma falha. Neste sentido o *MySQL Cluster* ostentou robustez. A estratégia de replicação síncrona, adotada por padrão na ferramenta, funcionou da maneira esperada, mantendo as réplicas dos bancos de dados idênticas. Com isto, a estrutura caracterizou-se pela confiabilidade das informações. A replicação de dados se mostrou uma ótima alternativa como solução para a problemática exposta neste trabalho, tendo em vista que possibilita a integração das informações geradas pelo Controle de Marcas e Sinais, viabilizando assim o seu uso mais efetivo e completo. Este requisito apresentou uma maior relevância pelo fato de o processo de replicação e recuperação dos dados serem executados totalmente de forma automática, sem a intervenção do usuário.

O *MySQL Cluster* também apresentou uma notável simplicidade na instalação e configuração, tendo em vista o amplo e didático material oferecido no site oficial do SGBD *MySQL*. Porém, em contrapartida, foi comprovado o alto consumo de memória RAM exigido pela ferramenta. Outro ponto a ser salientado é a precisão do serviço de gerenciamento "*ndb_mgm*" do *MySQL Cluster*, onde através dele é exibido o status das bases de dados distribuídas, alertando para possíveis anormalidades como o desligamento de algum nodo.

Um ponto relevante nesta solução, é que o acréscimo de mais cidades ao sistema Controle de Marcas e Sinais com bases de dados distribuídas se mostrou possível. No experimento exposto neste trabalho foi exemplificada a interligação das cidades de Bagé/RS e Hulha Negra/RS, porém nada impede que sejam incluídas outras cidades à estrutura. Então, o sistema poderá ser implantando em regiões específicas do país, porém com futuras expansões.

7 CONCLUSÃO

A larga utilização da *Internet* aliada ao uso cada vez maior das redes de computadores tem estimulado os estudos relativos ao desenvolvimento de bancos de dados distribuídos. Esses *softwares*, quando comparados aos modelos centralizados, apresentam uma eficiência maior, isso se deve às suas características diferenciadas, como por exemplo, tolerância à falhas, alta disponibilidade, compartilhamento de recursos, distribuição e escalabilidade.

É fato que algumas dificuldades referentes ao processo de replicação de base de dados são realidade. Mas, por outro lado, os benefícios que o processo de replicação pode nos proporcionar são muito relevantes, justificando os esforços na procura de soluções para esse tipo de problema.

A principal contribuição deste trabalho é a proposta de uma estrutura que realiza a integração de SGBDs *MySQL* com a ferramenta *MySQL Cluster*, formando uma estrutura de banco de dados distribuído. Com isso, tornou-se possível a interligação das bases de dados do sistema Controle de Marcas e Sinais. O projeto de sistema proposto nesta pesquisa teve como elemento direcionador o foco na facilidade e qualidade da implementação.

O estudo teórico revelou uma grande diversidade de propostas, conceitos e técnicas utilizadas inerentes ao tema proposto. Através da utilização da replicação de dados, o *MySQL Cluster* busca aumentar a disponibilidade e confiabilidade das aplicações, bem como prover um considerável aumento de desempenho, tendo em vista que os sistemas podem trabalhar com réplicas locais, não precisando requisitar bases remotas. Assim, as cidades podem manter suas estruturas atuais de *hardware*, bastando somente configurar o *MySQL Cluster* em seus servidores de banco de dados e fazer as alterações necessárias no código-fonte do Controle de Marcas e Sinais.

A ferramenta implementada também permitiu a verificação das premissas consideradas para o trabalho proposto, uma vez que os testes realizados mostraram a real possibilidade de se implementar um banco de dados distribuído ao sistema Controle de Marcas e Sinais.

8 TRABALHOS FUTUROS

Como proposta de trabalhos futuros é sugerida a evolução desta pesquisa no sentido de melhorar a eficiência com relação ao tempo de reposta das transações, garantindo um maior desempenho da estratégia de replicação síncrona utilizada pelo *MySQL Cluster*. Também, desenvolver uma solução para lidar com as chaves primárias e registros duplicados, para o caso de interligação das bases de dados que já se encontram em produção. Além disso, é sugerido um estudo para a elaboração de uma outra abordagem onde os servidores replicados não enviem as mensagens a todos os demais servidores de banco de dados instantaneamente, adotando assim uma estratégia de replicação assíncrona. Também, como oportunidades para a realização de trabalhos futuros, podem ser desenvolvidos estudos teóricos e práticos relacionados tanto ao desenvolvimento das tecnologias de distribuição de banco de dados quanto à sua aplicação em casos reais. Como exemplos podemos citar:

- Escrita de artigos científicos relacionados a área de Banco de Dados Distribuído;

- Colaboração com a comunidade do portal *Software* Público Brasileiro no que diz respeito ao sistema Controle de Marcas e Sinais;

- Acrescentar a estrutura uma ferramenta de sincronização automática das figuras contidas no servidor de aplicação do Controle de Marcas e Sinais;

- Agregar um servidor de gerenciamento a cada município, aumentado-se a disponibilidade da estrutura;

- Elaborar formalmente uma proposta para interligar, inicialmente, as cidades de Bagé/RS e Hulha Negra/RS;

- Implementar o uso do *MySQL Cluster* em outras aplicações.

Por esse trabalho se tratar de uma utilidade pública, a demanda pelo desenvolvimento de mecanismos mais eficientes serve de motivação a realização de novas pesquisas. A sua utilização real se torna bastante possível, tendo em vista que várias cidades já utilizam o Controle de Marcas e Sinais. Sendo assim, a integração das bases de dados trará grandes benefícios aos municípios utilizadores da aplicação.

REFERÊNCIAS

AMIR, Y. **Replication Using Group Communication Over a Partitioned Network**. Tese (Doutorado). The Hebrew University of Jerusalem, 1995.

BARBOSA, A. C. P. **Middleware para Integração de Dados Heterogêneos Baseado em Composição de Frameworks**. Tese (Doutorado). Pontifícia Universidade Católica. Rio de Janeiro, 2001.

BELL, D.; GRIMSON, J. **Distributed Database Systems**. Addison-Wesley Publishing Company,1992.

BORTOLINI, C. A. **Um Protótipo de Banco de Dados Distribuídos (Caso Expresso São Miguel)**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Universidade Comunitária Regional de Chapecó - Centro Tecnológico, 2004.

BRITO, M. S. **Arquitetura de um Sistema para Integração de Bancos de Dados com Suporte a Replicação Utilizando Tecnologia de Grades Computacionais**. Dissertação (Mestrado). Escola Politécnica da Universidade de São Paulo, 2009.

BROWNE, C. **Slony-I Documentation**. 2011. Disponível em: <http://slony.info/>. Acesso em: Mai/2013.

BURETTA, M. **Data replication: tools and techniques for managing distributed information**. John Wiley & Sons, 1997.

CARDOSO, J. L; MEFFE, C.; MARTINS, S. P. **O Software Público Brasileiro**. Revista Linux Magazine, nº 6, 2011.

CARVALHO, F. S. **Integração entre Sistema Multi-Agentes e Sistemas de Banco de Dados Distribuídos**. Dissertação (Mestrado). Escola Politécnica da Universidade de São Paulo, 2010.

COSTA, A. S. **Um Protocolo Distribuído para Controle de Consistência em Bancos de Dados Replicados para Ambiente de Computação Móvel**. Dissertação (Mestrado). Universidade de Fortaleza, 2010.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Distributed Systems – Concepts and Design**, Pearson - Prentice Hall, 4ª edição, 2005.

CRISTIAN, F. **Understanding Fault Tolerant Distributed Systems**. Comunications of the ACM, 1991.

DANTAS, M. **Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais**. Axcel Books do Brasil, 1ª edição, 2005.

DEFAGO, X.; SCHIPER, A.; SERGENT, N. **Semi-Passive Replication**. In: Symposium on Reliable Distributed Systems. West Lafayette: IEEE, 1998.

JALOTE, P. **Fault Tolerance in Distributed Systems**. Englewood Cliffs: Prentice Hall, 1994.

KEMME B.; ALONSO G. **Don't be lazy, be consistent: Postgres-R, a new way to implement Database Replication**, VLDB, Cairo, Egypt, 2000.

LEITE, E. A. **Uma Estratégia para Otimização do Processamento de Consultas na Arquitetura AMDB Usando Regras (Evento-condição-ação)**. Dissertação (Mestrado). Universidade de Fortaleza, 2004.

LIMA, C. C. **ORPIS: um Modelo de Consistência de Conteúdo Replicado em Servidores Web Distribuídos**. Dissertação (Mestrado). Universidade Federal do Rio Grande do Sul. Porto alegre, 2003.

LIMA, M. S. **Aplicação de Estratégias de Replicação de Bases de Dados em Sistemas Gerenciadores de Banco de Dados**. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação). Universidade Federal de Sergipe, 2011.

LORÊDO, H. Q.; FERREIRA L. N.; ASSIS G. T. **Replicação Assíncrona entre Bancos de Dados Heterogêneos: uma abordagem prática**. IV Congresso Brasileiro de Computação. 2004. Disponível em:
<http://www.ufrgs.br/niee/eventos/CBCOMP/2004/pdf/Banco_Dados/t170100218_3.pdf>.
Acesso em: Mai/2013.

MANFREDINI, R. A. **Condução de Experimentos de Injeção de Falhas em Banco de Dados Distribuídos**. Dissertação (Mestrado). Universidade Federal do Rio Grande do Sul, 2001.

MCOBJECT LLC. **eXtremeDB Cluster: McObject's distributed database system for real-time application**s. 2013. Disponível em: < http://www.mcobject.com/distributed-database > Acesso em: Abr/2013.

MELO, P. C. B. **Desenvolvimento de um Sistema de Replicação de Dados entre Bancos de Dados Relacionais**. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação). Universidade Federal do Rio Grande do Norte, 2010.

MENASCE, D. A. **Web Performance: the most obvious challenge for electronic commerce success**. In: Anais do Seminario Integrado de Software e Hardware, SEMISH. Belo Horizonte, 1998.

MYSQL. MySQL **5.0 Reference Manual: MySQL Proxy - Revisão: 35232**. Disponível em: <http://dev.mysql.com/doc/refman/5.0/en/>. Acesso em: Mai/2013.

OLIVEIRA, C. **An Architecture for Experimentation with Multi-tier Internet Applications in Virtualized Server Environments**. In: Anais do II Workshop de Sistemas Distribuídos Autonômicos, WOSIDA. Ouro Preto, 2012.

OLIVEIRA, V. F. **Especificação e Implementação de um Modelo Assíncrono para Replicação, Propagação e Conciliação de Bases de Dados Distribuídas**. Dissertação (Mestrado). Universidade Federal do Rio Grande do Norte, 2007.

OZSU, M.; VALDURIEZ, P. **Princípios de Sistemas de Banco de Dados Distribuídos**. Pearson - Prentice Hall, 2ª edição, 2001.

PGCLUSTER. PGCluster**: The multi-master and synchronous replication system for PostgreSQL**. 2005. Disponível em: <http://pgcluster.projects.postgresql.org/>. Acesso em: Mai/2013.

PICHILIANI, M. **Balanceamento de carga no MySQL – Parte 1**. 2009. Disponível em: <http://imasters.com.br/artigo/11161/mysql/balanceamento-de-carga-no-mysql-parte-1/>. Acesso em: Mai/2013.

POLYSERVE. **Data Replication for High Availability Web Server Clusters**. 2000. Disponível em: <http://users.encs.concordia.ca/~bcdesai/grads/steluta/references/datarep_highavail.pdf>. Acesso em: Abr/2013.

SILBERCHATZ, A; KORTH, H. F.; SUDARSHAN, S. **Sistemas de Banco de Dados**. Pearson Makron Books, 1999.

SILVA, D. B. **Interface para um Sistema Gerenciador de Transações Longas de Banco de Dados**. Dissertação (Mestrado). Universidade Federal do Rio Grande do Sul, 2003.

TAING, N; **Challenges in Distributed Systems**. 2011. Disponível em: <http://www.lycog.com/distributed-systems/challenges-distributed-systems>. Acesso em: Abr/2013.

TANENBAUM, ANDREW S.; STEEN, MAARTEN VAN. **Sistemas Distribuídos, Princípios e Paradigmas**, Pearson – Prentice Hall, 2ª edição, 2007.

TONINI, G. **pgGrid: Uma implementação de fragmentação de dados para o pgCluster**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Universidade Federal de Santa Catarina, 2009.

TRINDADE, R. M. **Uso do Network Simulator-NS para Simulação de Sistemas Distribuídos em Cenários com Defeitos**. Dissertação (Mestrado). Universidade Federal do Rio Grande do Sul. Porto Alegre, 2003.

VERÍSSIMO, P; RODRIGUES, L. **Distributed Systems forSystems Architect**, Kluwer, 2001.