BRUNO PÊSSO CALDEIRA

ALTA DISPONIBILIDADE – REPLICAÇÃO DE DADOS VIA MYSQL, COM ÊNFASE EM IDENTIFICAÇÃO E RECUPERAÇÃO DE FALHAS

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação *Lato Sensu* em Administração de Redes Linux para obtenção do título de especialista em Redes Linux

Orientador
Prof. Joaquim Quinteiro Uchôa

LAVRAS
MINAS GERAIS – BRASIL
2006

BRUNO PÊSSO CALDEIRA

ALTA DISPONIBILIDADE – REPLICAÇÃO DE DADOS VIA MYSQL, COM ÊNFASE EM IDENTIFICAÇÃO E RECUPERAÇÃO DE FALHAS

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação *Lato Sensu* em Administração de Redes Linux para obtenção do título de especialista em Redes Linux

APROVADA em 28 de Setembro de 2006

_____
Prof. Sandro Pereira de Melo

_____
Prof. Heitor Augustus Xavier Costa

_____
Prof. Joaquim Quinteiro Uchôa
(Orientador)

LAVRAS

MINAS GERAIS - BRASIL

DEDICATÓRIA

Dedico este trabalho a todos que acreditaram em mim. Por causa de sua paciência, compreensão e apoio irrestrito, consegui vencer mais uma etapa desta jornada de trabalho e estudo.

Minha querida mãe Elizabeth, minha esposa Renata e principalmente meu filho Lucas. Espero ter deixado para ele um exemplo de perseverança e força de vontade.

AGRADECIMENTOS

A Deus, pela vida. Obrigado, Senhor, por tudo que nos ofereceste.

A minha família, minha mãe Elizabeth, minha esposa Renata, meu filho Lucas. Meu avô João, obrigado pelas conversas científicas que me fazem refletir sobre o mundo, seu passado e futuro. Minha avó Jane, obrigado pelo carinho e paciência de mãe.

Moisés, obrigado por acreditar em mim. Obrigado pelos votos de confiança e apoio nas horas complicadas da vida. Sua luta e trabalho me enchem de inspiração e vontade de crescer, como pessoa e como profissional.

Aos amigos de trabalho, que compartilham comigo os sucessos e frustações do dia a dia. Não podemos desistir frente aos problemas, nunca!

Obrigado ao professor Joaquim Quinteiro Uchôa, pela vontade e determinação de manter a qualidade do curso de pós-graduação em Administração de Redes Linux.

Obrigado aos profissionais e amigos que direta ou indiretamente me apoiaram na elaboração deste trabalho.

RESUMO

Sistemas Gerenciadores de Bancos de Dados (SGBD) podem executar várias funções, além de simplesmente armazenar informações. O objetivo deste trabalho é mostrar como funciona a Replicação de Dados em MySQL, um poderoso e estável SGBD de código livre, além de citar vários tipos de falhas possíveis durante uma replicação e suas recuperações.

A Replicação de Dados é uma cópia automatizada de uma origem para um ou vários destinos, e sua principal utilidade é fornecer alta diponibilidade da informação.

A partir de uma configuração e uso da Replicação de Dados em MySQL de uma empresa foi elaborado um Estudo de Caso, que ajudou a enriquecer a qualidade do presente documento.

Sumário

1 - Introdução

A replicação de dados tem sido usada principalmente para melhorar a disponibilidade e desempenho no acesso dos dados. Esta monografia tem como objetivo mostrar o funcionamento da replicação de dados MySQL, enfocando os problemas que podem ocorrer, suas causas e as soluções possíveis para a correção dos mesmos.

O MySQL é um gerenciador de banco de dados livre conhecido por sua facilidade e velocidade de operação, que pode ser instalado a partir de *download* dos arquivos no sítio http://www.mysql.com/downloads. Presume-se que o leitor tenha algum conhecimento de banco de dados MySQL (sua estrutura e tipos de tabelas) e também de redes (topologias básicas e conceitos de sockets).

A elaboração desta monografia foi baseada em consultas em bibliografia especializada no assunto, assim como pesquisas em artigos, textos e fóruns no sítio oficial do fabricante do MySQL (http://www.mysql.com) e outros. Também foi montado um cenário real

(Estudo de Caso), com vários computadores, para simulações e testes com a replicação.

Este Estudo de Caso foi fundamental para o desenvolvimento do conteúdo desta Monografia, pois proporcionou uma análise crítica dos tópicos pré-selecionados. A principal motivação para a escolha do tema foi profissional, pois o autor enfrentou problemas relacionados à disponibilidade de dados, na empresa onde trabalhava. Havia uma necessidade de um servidor exclusivo para o setor da Contabilidade da empresa, somente com os dados necessários e de forma exclusiva para os funcionários daquele setor, para que se pudesse obter maior velocidade no acesso das informações.

Em consequência da necessidade profissional, o autor viu uma oportunidade de ampliar os conhecimentos adquiridos sobre replicação de dados via MySQL. Isto possibilitou que novos recursos de alta disponibilidade e escalabilidade fossem usados na mesma empresa, em outros setores, maximizando o processamento dos vários servidores, fato este que acarretou economia na aquisição de memória e processadores mais robustos.

Para a apresentação do trabalho, este texto encontra-se organizado como se segue: no Capítulo 2 são apresentadas as características de bancos de dados relacionais e Sistemas Gerenciadores de Bancos de Dados (SGBD); no Capítulo 3 é apresentado o MySQL e os tipos de tabelas suportados por ele, além de uma seção comentando sobre possíveis inovações deste SGBD; no Capítulo 4 são abordados conceitos de replicação de dados em SGBDs; no Capítulo 5 são apresentadas as possíveis falhas de replicação de dados; no Capítulo 6 é reportada a configuração básica de uma replicação de dados em MySQL; o Capítulo 7 descreve o Estudo de Caso; o Capítulo 8 apresenta a conclusão do trabalho.

2 - Bancos de dados relacionais

Bancos de dados são estruturas de arquivos organizados de forma a cumprir um papel simples: Armazenar informações. Alguns bancos de dados têm uma característica especial que melhora muito a performance de escrita e leitura dos dados. Esta característica nada mais é que a própria estrutura do banco, que permite uma associação de uma parte do banco com outra(s) parte(s) do banco. De acordo com (Ricarte, 2002):

> Banco de dados relacional organiza seus dados em relações. Cada relação pode ser vista como uma tabela, onde cada coluna corresponde a atributos da relação e as linhas correspondem as tuplas ou elementos da relação.

A estrutura das tabelas e registros é talvez a principal característica dos bancos de dados relacionais. Esta estrutura define como os dados são organizados, para que haja uma definição de espaço de uso da memória e do disco e para que o SGBD (Sistema Gerenciador de Bancos de Dados) possa manipular os dados. Após a criação de uma tabela e de seus atributos (colunas, campos), o usuário poderá inserir os

registros, que irão obedecer as delimitações estabelecidas pela estrutura da tabela, como tipo e tamanho de cada campo.

Um importante recurso dos bancos de dados relacionais é a possibilidade de usar índices ou chaves primárias. Os índices criam uma delimitação de valores únicos para determinadas colunas ou sequências de colunas, impedindo que valores iguais sejam inseridos nas colunas, ou grupos de valores iguais, no caso de chaves para sequências de colunas.

Os índices otimizam as consultas e inserções de dados, o que justicaria a criação de vários índices, não fosse a desvantagem de que usar este recurso causa aumento considerável de uso de memória secundária (disco rígido) e também da primária (memória RAM). Isto se explica pelo fato de que na criação do índice é gerado uma sequência de dados ordenados pelo índice, nos arquivos do banco de dados.

Os bancos de dados relacionais possibilitam a junção de várias tabelas, que por fim é a justificativa para o termo “relacional”. Estas relações são estabelecidas através da comparação de colunas de diferentes tabelas, que devem obrigatoriamente possuir o mesmo tipo de informação, como por exemplo, um número inteiro (código), ou uma string. Desta forma, a relação entre as tabelas ocorre de forma coerente.

Um SGBD (Sistema Gerenciador de Bancos de Dados) é um *software* que coordena e organiza um banco de dados. As funções básicas de um SGBD são:

- prover métodos de acesso ao banco de dados;
- assegurar integridade sintática dos comandos disponíveis e integridade semântica de uma forma geral;
- estabelecer regras de segurança;
- disponibilizar várias outras facilidades, de acordo com cada modelo e versão adotados.

Independente do SGBD escolhido para trabalhar, o usuário precisa garantir que o fluxo de informações no banco, controlado por transações, siga determinadas regras para que cada dado depositado ou em movimento seja consistente. Segundo Oliveira (2002),

> Transações podem ser definidas como unidades de trabalho indivisíveis que contém uma série de operações das quais todas ou nenhuma deve ser executada para garantir a integridade dos dados.

De fato, um ambiente ideal onde ocorrem transações em bancos de dados deve possuir todas as propriedades ACID, acrônimo para *"Atomic,*

Consistent, Isolated, Durable", ou "Atômico, Consistente, Isolado, Durável". Estas propriedades aplicadas em conjunto garantem a integridade e consistência das informações tratadas pelas transações.

Um exemplo para a aplicabilidade das propriedades ACID seria o seguinte: A transação X é formada pelas operações A, B e C. Se uma das operações falhar, e esta falha não for prevista, a transação não é executada, mesmo que a falha seja na última operação. A idéia é de que tudo que foi programado para a transação X deve funcionar sem erro, senão nenhuma ação, de nenhuma das operações A, B e C, deve ser executada.

Existem alguns padrões de estrutura de SGBD, dentre eles pode-se destacar o SQL, definido pelo ANSI (American National Standards Institute) através do documento ISO/IEC 9075:1992. Este padrão estabelece uma semelhança entre as várias implementações de linguagens SQL disponíveis no mercado, como o SQL Server, Oracle, PostgreSQL e o MySQL, este último escolhido para o uso do Estudo de Caso desta monografia, dada a experiência do autor com o mesmo e por necessidades profissionais.

3 - MySQL

O MySQL é uma das principais distribuições de SGBD de padrão SQL disponíveis para uso comercial. É executável em vários Sistemas Operacionais, é estável (novas *releases* são disponibilizadas após meses de testes), e possui baixo TCO[1].

O MySQL AB, grupo mantenedor e desenvolvedor do MySQL, oferece excelente suporte, através do sítio http://www.mysql.com. É disponibilizado um vasto material de referência e inúmeros fóruns de discussão.

O MySQL possui uma característica interessante que é a possibilidade de escolher o tipo de tabela no momento de criação da mesma. O formato de armazenamento dos dados, bem como alguns recursos do banco de dados dependem do tipo de tabela escolhido. Segundo Suehring(2002),

1 *Total Cost of Ownership* ou Custo Total de Implementação: Total de dinheiro gasto em uma aquisição de um software, incluindo toda a estrutura física para a sua utilização, aquisição de licença, aquisição de Sistema Operacional compatível, custos com manutenção e testes, instalação e configuração.

Um tipo de tabela é um dos vários conjuntos de características que se pode aplicar a uma tabela de banco de dados no MySQL – normalmente para um propósito especializado, embora a aparência das tabelas do MySQL possa ser semelhante de um tipo para outro.

A DDL[1] usada na criação de tabelas novas no MySQL é a seguinte:

```
mysql> CREATE TABLE nome_tabela (
      campo_1  tipo_campo_1  opções,
      campo_2  tipo_campo_2  opções,
      (...)
      campo_n  tipo_campo_n  opções,
      primary key (campo_chave)
      ) TYPE=tipo_tabela;
```

O tipo MyISAM, escolhido para aplicação no Estudo de Caso, será detalhado na seção 3.1. Outros tipos de tabela também serão apresentados, com algumas características.

1 DDL: *Data Definition Language* ou Linguagem de Definição de Dados: Grupo de comandos de linguagens SQL que oferece recursos de criação e alteração de estrutura de tabelas.

3.1 - Tabelas MyISAM

Tabelas MyISAM têm um ótimo desempenho para leitura, uma vez que os seus índices são armazenados em árvores binárias balanceadas[1], garantindo acesso mais rápido às informações. Isto é muito importante durante a replicação, pela grande velocidade de leitura dos dados. Este tipo de tabela foi escolhido para o Estudo de Caso.

O MyISAM não possui controle de transações (*commit ou rollback*), o que significa que após uma alteração de dado, por exemplo, a informação é imediatamente gravada em disco, não permitindo que se desfaça o que foi feito a partir de certo ponto.

Um problema encontrado neste tipo de tabela é que elas possuem um mecanismo de *lock* por tabela. Desta forma, toda vez que há uma escrita na tabela, ela é travada pelo MySQL, o que bloqueia por um instante o acesso à ela, mesmo para processos que tentam acessar registros que estão sendo modificados. Isto gera uma fila de espera para

1 Estruturas de dados cujos nós geram um, dois ou nenhum nó, e assim sucessivamente, formando várias sub-árvores. As alturas de duas sub-árvores de cada um dos nós nunca diferem em mais de 1. O balanceamento de um nó é igual a diferença entre as suas altura esquerda e direita.

liberação dos locks, que pode diminuir o desempenho de acesso às tabelas.

Apesar do problema apresentado acima, a tabela MyISAM é bem rápida quanto a acesso, devido à sua arquitetura simplificada. Os arquivos gerados durante a criação de uma tabela MyISAM são três: Um para gravar a estrutura da tabela (extensão frm), outro para gravar os dados (extensão MYD), e outro para gravar os índices (extensão MYI).

3.2 - Tabelas HEAP

Tabelas HEAP são armazenadas na memória, o que as torna extremamente rápidas para acesso. Porém, uma queda do SGBD pode causar perda dos dados, já que o conteúdo das tabelas é "volátil".

Uma aplicação de tabelas HEAP seria para tabelas que são frequentemente consultadas, mas que não sofrem muitas alterações. Ou seja, é possível copiar dados de um disco para tabelas HEAP, residentes na memória, maximizando a velocidade de acesso.

Usar tabelas do tipo HEAP para a replicação não é uma boa idéia, visto que a replicação necessita da geração de um *log* binário. Este *log* precisa ser um arquivo disponível no mestre, até a cópia pelo(s) escravo (s), como será bem detalhado nos Capítulos 4 e 6.

3.3 - Tabelas BDB

As principais diferenças entre a tabela MyISAM e a tabela BDB (ou BerkeleyDB) é que esta última possui controle de transação através de *commit* e *rollback*, além de fornecer uma recuperação automática em caso de queda do sistema ou do SGBD.

É importante dizer que o controle transacional não é uma propriedade exclusiva do MySQL, sendo utilizado por outros SGBDs, como o Oracle. As características básicas de *commit* e *rollback* podem ser resumidas de acordo com Souza(2004):

> Um segmento de *rollback* é uma porção de um Banco de Dados que registra as ações das transações dos usuários nos dados, para que possam ser desfeitas sob certas circunstâncias.(...) O comando *commit* efetiva as alterações feitas nos dados por uma transação, tornando-as permanentes.

Tabelas BDB possuem um mecanismo de *lock* em nível de página, onde os dados de uma mesma página ficam bloqueados durante o período de *lock*. O recurso de *lock* de registros estabelece uma forma de garantir integridade na informação, já que várias instruções SQL podem conter regras que bloqueiam leitura e/ou escrita de registros.

3.4 - Tabelas InnoDB

O InnoDB é um tipo de tabela com recursos importantes, como Integridade Referencial, ferramenta de *backup on-line* (que por ser uma

aplicação comercial não será comentada aqui), *lock* de registros, níveis de isolamento, controle de transações e outros.

Integridade Referencial é um recurso importante, pois é uma forma de assegurar que uma alteração em uma tabela, seja ela remoção de registro(s) ou alteração de atributo(s), cause também um efeito em outras tabelas relacionadas a ela, para que a integridade da informação, como um todo, não seja prejudicada. Isto é feito através do uso de alguns comandos DDL, na criação da tabela, que estabelecem as regras de referências. Assim, quando remove-se uma nota fiscal de uma base de dados de um sistema contábil, por exemplo, os itens dessa nota fiscal também serão removidos, se esta regra estiver sido estabelecida durante a criação de ambas as tabelas.

O recurso de *lock* de registros, como foi citado na seção anterior, estabelece uma forma de garantir integridade na informação.

Apesar de apresentar-se mais lentas que tabelas MyISAM, tabelas InnoDB são muito usadas por causa de sua versatilidade e integridade dos dados.

3.5 - O futuro do MySQL

Novas tecnologias afetam diretamente qualquer aplicação na área de informática, principalmente aquelas que dependem de desempenho e confiabilidade – e isso inclui os bancos de dados. Qualquer novidade do MySQL é acompanhada com muita expectativa pelas comunidades de usuários e desenvolvedores, visto que o uso deste SGBD e de tantos outros cresce a cada dia, devido à cresecente necessidade de substituir o papel pela informação digital.

Gates (1995), disse:

> Na estrada da informação, elaborados documentos eletrônicos poderão fazer coisas que nenhum pedaço de papel pode. A poderosa tecnologia de banco de dados da estrada permitirá que eles sejam indexados e lidos por meio de exploração interativa. Será extremante barato e fácil distribuí-los. Em resumo, esses novos documentos digitais substituirão muitos dos documentos impressos em papel porque eles poderão nos ajudar de novas maneiras.

Comprovando o que Bill Gates já previa em 1995, o uso de bancos de dados digitais cresceu de forma exponencial; um exemplo recente foi o lançamento da nova versão do MySQL (5.0), disponibilizada para aplicações comerciais em 24 de Outubro de 2005 – o maior acontecimento desde o lançamento do produto, há mais de dez anos, de acordo com o próprio e-mail de divulgação do MySQL AB. Trouxe dezenas de novidades, dentre elas: *Stored Procedures, Triggers, Views* e outros. A empresa norte-americana Liveworld Inc.[1] chegou a classificar a nova versão com um componente-chave na sua infra estrutura de TI[2]. Nas três primeiras semanas após a liberação da versão 5.0 estável, um milhão de *downloads* haviam sido feitos pelo sítio oficial.

Com relação à replicação, alguns cenários poderiam ser melhorados caso fossem desenvolvidos algumas configurações especiais para atendê-los, evitando complicações e propensões a erros. O monitoramento da replicação também é uma questão delicada. Poderia existir um processo automatizado de monitoramento, de forma que o próprio MySQL tomaria as medidas adequadas de recuperação, em casos de falhas. Tudo, é claro, baseado em configurações e regras pré-

1http://www.mysql.com/news-and-events/press-release/ release_ 2005 42.html
2Tecnologia da Informação.

estabelecidas pelo usuário, e contando também, se fosse possível, com um banco de Inteligência Artificial (IA). A recuperação seria automática e segura, e desta forma o usuário seria avisado do problema somente, tomando alguma medida se fosse necessário. A idéia de usar IA é interessante pois o MySQL "aprenderia" com os erros, evitando-os na próxima possibilidade de falha.

A equipe de desenvolvimento do MySQL recebe estas e outras sugestões e procura melhorar o *software* na medida do possível, baseado em prioridades, é claro. No sítio oficial existe uma seção com as próximas metas de desenvolvimento do código, a curto e médio prazo (http://dev.mysql.com/doc/refman/4.1/pt/todo.html).

4 - Replicação de dados em SGBDs

A definição mais simples e básica que pode-se atribuir à replicação seria a de uma cópia. As regras e os resultados que definem esta cópia é que a diferencia de um *backup* comum. Copiar consiste em ler de uma origem e escrever exatamente o que foi lido em um outro local, chamado de destino. Replicar, por outro lado, pode ser entendido como copiar uma seleção de dados e disponibilizá-los para um ou vários destinos, de forma progressiva e contínua, ou seja, a medida que novas informações são adicionadas, retiradas ou modificadas na origem, assim também será feito nos destinos.

Garantir que a cópia seja idêntica tanto na origem quanto no destino é de responsabilidade da aplicação que está executando a cópia. No caso de replicação de dados, garantir a "estabilidade" da sincronia é de responsabilidade do SGBD. Obviamente, qualquer fator que impeça a correta execução de algum processo durante a replicação derruba esta "estabilidade" e pode ocasionar, no mínimo, a interrupção da replicação.

Como foi citado anteriormente, tipos de tabelas que possuem controle de transações têm uma característica importante: Elas aumentam as chances de preservar a integridade dos dados, pois conferem maior segurança durante as operações, garantindo a atomicidade de todos os processos.

4.1 - Replicação de Dados no MySQL

A idéia básica da replicação, em MySQL ou em qualquer outro banco de dados,é que um servidor mestre irá registrar todas as modificações feitas em seus dados que serão copiados por um ou vários servidores escravos. De acordo com Suehring (2002),

> A replicação se refere à cópia automatizada de alterações feitas em bancos de dados e dados entre dois ou mais servidores MySQL.

Todo o processo é controlado por um ou mais *daemons*. Uma consideração importante é que nem toda replicação precise ser necessariamente automática, apesar de que esta seja uma característica importante desta facilidade.

Entende-se por "alterações" todas as modificações que um registro sofre, inclusive inserção e deleção. No caso de tabelas novas criadas no servidor mestre, elas não serão replicadas. Neste caso, elas deverão ser criadas em todos os servidores participantes da replicação, e o MySQL deverá ser interrompido e reiniciado em todos eles para sincronização. A estrutura da tabela também deverá ser mantida durante uma replicação via MySQL, como será mostrado no Capítulo 6.

No caso de remoção de tabelas no mestre, não haverá problema para a execução da replicação, porque simplesmente não serão geradas informações daquela tabela para replicação. Já a remoção de tabelas no escravo irá causar interrupção da replicação assim que o mestre registrar modificações em algum dado daquela tabela, já que o escravo não poderá copiar as modificações. Obviamente aqui se está comentando sobre a execução da replicação, porque uma tabela removida no servidor mestre

pode gerar transtornos em termos de disponibilização coerente de dados para os vários usuário do sistema, mas isto não é o foco desta seção.

A configuração básica de uma replicação mestre-escravo é a de um servidor gerando *logs* das transações executadas que possam ser copiados por um único servidor escravo. Este tipo pode ser chamado de *Topografia de Escravo Único*, e é mais comum em aplicações de *backup* e *failover*.

Este tipo de replicação pode ser implementado com mais servidores, em uma configuração hierárquica ou de níveis, permitindo, em teoria, infinitas conexões de escravos para cada mestre. Se um escravo é copiado por outro servidor escravo, aquele pode ser chamado de mestre também. Este tipo de replicação é chamado de *Topografia de Árvore* e é a mais comum entre os usuários de MySQL. Esta topografia permite escala de queries (consultas) quase ilimitada.

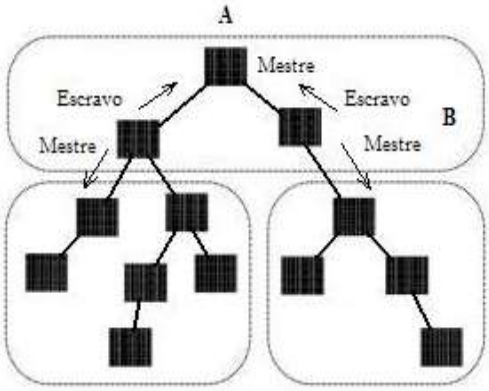

Figura 1 – Topografia de Árvore

A Figura 1 ilustra a idéia de vários servidores conectados em vários níveis hierárquicos de replicação. O ponto A indica um servidor mestre sendo replicado por dois servidores, no nível indicado por B. Os escravos, neste nível, também geram *logs* que permitem a replicação para níveis mais baixos, e assim por diante.

A *Topografia Escravo de Relay* é semelhante à Árvore, porém o mestre principal possui somente um escravo. Isto ajuda a aliviar a carga sobre este mestre, já que ele provavelmente se ocupa de inúmeras outras aplicações além da replicação. Este tipo de topografia também é útil quando existe mais de um local físico para a replicação.

O MySQL não suporta internamente replicação bidirecional (ou replicação de mão dupla) dos mesmos dados. Características do MySQL como *auto_increment*, *unique keys* e outras causam problemas durante esta topografia. A replicação bi-direcional também pode ser entendida como uma replicação com vários mestres.

A principal vantagem da replicação mestre / escravo é a simplicidade das topografias e das configurações, assim como várias funções do MySQL. Para quem usa este SGBD, não há dificuldade em montar um ambiente de replicação.

Na contra-mão, as desvantagens podem aparecer em situações que exigem um maior graus de complexidade, como topografias avançadas e seleção de linhas ou colunas (ou ambas) durante a replicação.

4.2 - Sincronia durante a Replicação de Dados em MySQL

A replicação de dados em MySQL é assíncrona, pois os dados não podem ser escritos no mestre e no(s) escravo(s) ao mesmo tempo, como será explicado a seguir.

As replicações de dados síncronas são feitas no exato instante da modificação. Quando uma cópia é gerada, as alterações são imediatamente aplicadas a todos os bancos de dados replicados. Os servidores replicados são configurados para que protocolos especializados mantenham a replicação coerente. Um destes protocolos é o *"Two Phase Locking Protocol"*, que consiste na divisão da transação em duas etapas: Na primeira, a transação pode ler dados, mas não pode gravar. Na segunda, a transação pode gravar dados, mas não pode ler. Este protocolo garante a serializabilidade da replicação. A replicação síncrona é indicada em sistemas comerciais onde as informações devem ser idênticas nos servidores, sem intervalo de tempo.

Durante uma replicação assíncrona, como ocorre no MySQL, a cópia dos dados pode ficar fora de sincronia entre os bancos durante alguns segundos, horas ou até mesmo dias depois. Quando um dado é alterado, a modificação é registrada e fica aguardando a leitura dos outros servidores, numa segunda etapa. O registro da modificação poderá ficar

temporariamente fora de sincronia das bases dos servidores replicados por motivos diversos (dentre eles falhas que serão vistas a seguir). Quando a replicação executa a leitura dos dados pelos escravos, estes são atualizados até a ocorrência da sincronia.

4.3– Outras Alternativas de Replicação

Uma alternativa para a replicação de dados é o Rsync[1]. Este programa é baseado no rcp (*remote copy*), além de possuir propriedades de criptografia do protocolo SSH[2]. O Rsync permite uma cópia ou sincronização lógica entre diretórios de um mesmo servidor ou entre servidores remotos, através de comandos simples. A diferença para a replicação via MySQL é que esta última copia os comandos MySQL, instantaneamente, portanto trabalha com informações exclusivas das tabelas do banco de dados MySQL, enquanto o Rsync pode arquivos comuns, em lotes, já que depende de comandos para executar as cópias.

1 http://rsync.samba.org/
2 http://www.ssh.com/

O DRBD[1] é uma outra alternativa para replicação de dados, porém suas características são bem distintas da replicação via MySQL. O funcionamento do DRBD (acrônimo para *Distributed Replicated Block Device* – Dispositivo Replicado e Distribuído de Blocos) é baseado em sistemas RAID, pois é feito um espelhamento de conjuntos de blocos, via rede, se necessário. Portanto, a replicação via DRBD é física, e não lógica.

É importante observar que as soluções de replicação baseadas em Rsync ou DRBD têm por objetivo replicar arquivos ou partições inteiras, oferencendo alta disponibilidade. O autor deste trabalho necessitava apenas de replicar informações de algumas tabelas do SGBD MySQL, oferecendo balanceamento de carga, e portanto não saberia dizer se se utilizasse Rsync ou DRBD para este fim seria uma alternativa confiável. Ter o SGBD ativo no servidor escravo com estes tipos de replicação apresentados poderia gerar mais trabalho de configuração e até de execução, porque dependeria de outros programas de automação, como o Cron ou o Heartbeat.

1 http://www.drbd.org/

Outros SGBDs também possuem gerenciamento de replicação de dados, de forma semelhante ao MySQL, porém com algumas particularidades. O SQL Server[1], da Microsoft, por exemplo, oferece uma replicação chamada "Publicador-Assinante", que possui uma funcionalidade muito importante, e ainda não disponível no MySQL, que é o particionamento horizontal, vertical ou misto. Pode-se escolher linhas, colunas ou uma seleção mista, permitindo assim uma possibilidade de customização maior durante a replicação de dados.

O PostgreSQL[2] utiliza um programa chamado eRServer para gerenciar a replicação de dados. O programa oferece configurações e funcionalidades semelhantes à replicação via MySQL, inclusive quanto à recuperação de falhas, que também não é automática.

1 http://www.microsoft.com/sql/default.mspx
2 http://www.postgresql.org/

5 - Possíveis falhas de replicação

É muito importante saber a(s) causa(s) de uma falha em uma replicação, qualquer que seja seu tipo. O mais importante neste capítulo é perceber que as falhas envolvem principalmente os elementos que compõe a replicação via MySQL, e não ela em si, como programa gerenciador da replicação.

Falhas ditas "globais" são aquelas que atingem todo o sistema, como uma queda de energia, por exemplo. Date (2000) classifica as falhas globais em "falhas do sistema" (ou soft crash) e "falhas da mídia" (ou hard crash):

> Falhas do sistema afetam todas as transações em curso no momento, mas não danificam fisicamente o banco de dados. Falhas de mídia causam danos ao banco de dados ou a uma parte dele, e afetam pelo menos todas as transações que no momento estão usando essa parte.

Para tomar o melhor caminho na reparação dos erros, o administrador pode e deve estar munido de programas que possam auxiliá-lo no trabalho de gerenciamento do banco de dados. Uma

interessante ferramenta que deveria ser usada por administradores de SGBD MySQL é o *MySQL Administrator*. É um programa disponibilizado para *download* no sítio oficial, assim como seu manual, conforme MySQL AB (2006). O programa oferece inúmeras facilidades para configurar, monitorar e gerenciar o funcionamento do banco de dados, com uma interface gráfica intuitiva, além de gráficos de desempenhos e ajustes em geral. O programa oferece uma seção exclusiva para assistência e monitoramento da replicação de dados, de fácil uso e visualização, como mostra a Figura 2.

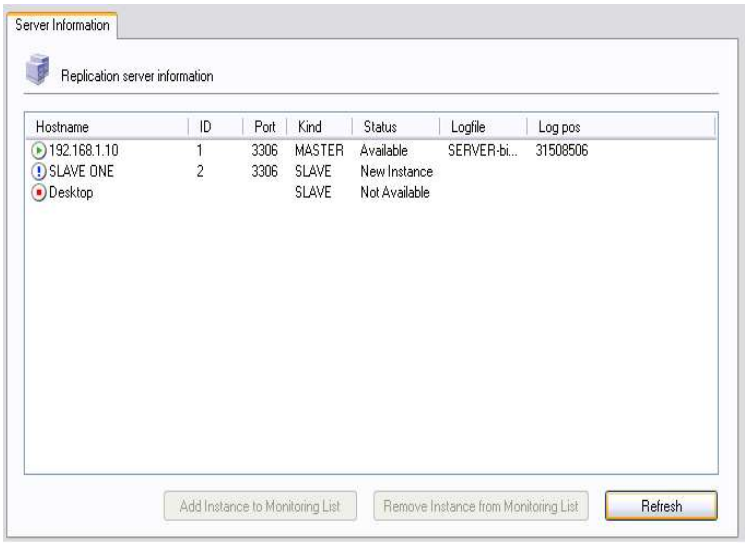

Figura 2 – Janela de monitoramento da Replicação de Dados no MySQL Administrator

O *mysqladmin* é um programa cliente embutido nas distribuições MySQL desde as primeiras versões, e serve para tarefas administrativas diversas, inclusive para controle da replicação.

Outro programa de monitoramento, em modo texto, chamado *mysqlmanager*, pode ser obtido para uso a partir da versão do MySQL 5.0.3, no sítio oficial. A partir do MySQL 5.0.4, pode-se iniciar o programa automaticamente, bastando alterar uma configuração, conforme instruções obtidas no manual de referência oficial, de acordo com MySQL AB (2006). Este programa oferece inúmeras funções de gerenciamento de processos e instâncias do MySQL.

5.1 - Falhas de rede

Uma composição estruturada da rede é primordial para o funcionamento de qualquer disponibilização de dados, como é o caso da replicação. O fluxo de dados em qualquer rede só deve ser interrompido em manutenções periódicas planejadas. Paradas inesperadas podem causar grandes prejuízos para as aplicações cujos dados trafegam na rede,

principalmente na replicação. Para o correto dimensionamento de uma rede, vários pontos são levados em consideração.

A parte física, que envolve cabos, fios, modens, *switchs* e outros equipamentos, deve ser de boa qualidade. As falhas podem acontecer por oxidação e/ou rompimento de cabos, defeitos em fontes, placas lógicas e outros componentes dos equipamentos. Este tipo de problema é freqüente em redes que não recebem monitoramento adequado, ou que são planejadas com orçamento insuficiente. Problemas como longas distâncias entre *hubs* e *switchs* também causam queda da eficiência na rede, e devem ser evitados. É inadmissível configurar uma rede com estes tipos de problemas, porque a recuperação destas falhas pode ser demorada, e geralmente exige toda uma reestruturação.

Fornecimento de energia inadequado também causa falhas na rede, principalmente por causa de tensões variáveis e interrupção. Quando uma tensão não é estável, os equipamentos podem desligar ou até mesmo queimar. Para garantir uma "energia limpa" (valor de tensão senoidal e permanente, livre de freqüências indesejadas), deve ser exigido um bom projeto elétrico para atender a rede, com aterramento, quadros de distribuição exclusivos, chaves disjuntoras e n*o breaks* para todos os

equipamentos. A estimativa de fornecimento de energia emergencial, em casos de interrupção, dependerá do orçamento disponível, mas é importante que seja suficiente para manter todos os equipamentos da rede ligados. Em grandes redes é comum o uso de energia própria, através de geradores, que minimiza este tipo de problema.

O dimensionamento físico deve ser adequado para suportar a exigência lógica da rede, ou seja, deve ser suficiente para garantir o tráfego de dados na rede. "Suficiência" deve ser interpretado como capacidade, e não quantidade. Também devem estar incluídos na parte lógica da rede as configurações dos equipamentos e os programas gerenciadores. Falhas de lógica de rede incluem falta de rotas, pacotes bloqueados por *firewalls*, tráfego de dados superior à capacidade dos equipamentos, programas e aplicativos gerenciadores de rede mal configurados e outros. Antes da rede ser ligada, vários testes devem ser feitos buscando garantir perda zero no tráfego de dados.

A replicação é interrompida se o tráfego de dados cessar, seja por falhas físicas ou lógicas, e a recuperação pode ser lenta, dependendo do tamanho da rede. Por isso, um bom planejamento e investimento adequado deve ser levado em consideração na construção de redes. A

Figura 3 ilustra falhas na rede. O importante foi mostrar que o fluxo de dados, representado pela seta, não atinge o seu objetivo, que é ser lido pelo servidor escravo.

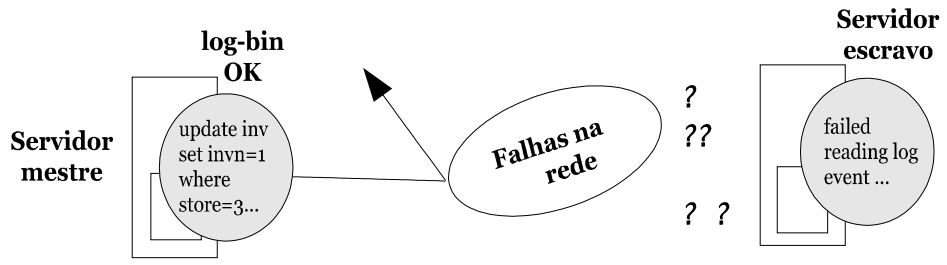

Figura 3 – Representação gráfica de falha de rede na replicação via MySQL

O administrador deve ficar atento às novas tecnologias de redes disponíveis e aos níveis de segurança que elas oferecem. Invasores podem interromper uma replicação se conseguirem acesso ao SGBD. Segundo Tanenbaum (2003),

> Como milhões de cidadãos comuns atualmente estão usando as redes para executar operações bancárias, fazer compras e arquivar sua devolução de impostos, a segurança das redes está despontando no horizonte como um problema potencial.

Mesmo sabendo que grande parte dos problemas com segurança de redes envolve seres humanos mal intencionados e não as falhas físicas e lógicas da rede, o administrador consciente sabe que deve cuidar bem dessa questão, visando resguardar as informações controladas pelo SGBD.

Existem inúmeros programas de monitoramento de redes disponíveis, inclusive de código aberto, como o Nagios[1] e o MRTG[2], que oferecem uma gama enorme de funções que podem auxiliar o administrador de redes nas suas tarefas, identificando falhas, uso de banda, avisos sobre disponibilidade da rede, e muitos outros.

5.2 - Falhas do servidor mestre

Em qualquer tipo de replicação, existe o conceito de servidor. Ele é a fonte de informações que serão compartilhadas, independente de como a distribuição dos dados será feita. Assim como acontece nas redes de um modo geral, o servidor está propício a falhas físicas e lógicas, dependendo

1 http://www.nagios.org/
2 http://www.mrtg.org/

de como a arquitetura de TI foi implementada e como ela é mantida. O administrador deve planejar adequadamente a arquitetura de TI, conforme Gordon e Gordon (2006) citaram:

> A arquitetura da tecnologia da informação descreve um plano estrutural de longo prazo, abrangendo toda a organização, que define investimentos e trata da organização da tecnologia da informação.

As falhas físicas de um servidor dizem respeito a *hardware*. Servidores comuns, do tipo CPU de gabinete, podem ser danificados por fornecimento inadequado de energia, temperaturas altas do ambiente e acúmulo de sujeira na parte interna, que podem causar defeitos nas peças que compõe o servidor, como discos rígidos, placa mãe, memória, placas e periféricos (CD-ROM, leitor/gravador de fita DAT, controladora SCSI, etc.). Servidores "montados" por pessoal inexperiente podem apresentar problemas precoces, devido à incompatibilidade entre os vários componentes. Acidentes físicos, como quedas e contato com materiais transportados próximos ao servidor podem causar sérios danos ao

equipamento. Empresas com maiores volumes de dados usam servidores montados com equipamentos maiores, em grandes salas isoladas e monitoradas (ou prédios inteiros). Seja qual for o tamanho do servidor, um defeito físico pode interromper a replicação se:

- A comunicação do servidor com a rede for interrompida por causa de defeito da placa de rede, retirada do cabo de rede, defeito no *modem* (*wireless*), etc;
- O servidor parar o processamento por causa de falhas e/ou conflitos entre os componentes internos do servidor.

A configuração do servidor deve ser feita com muita cautela. O funcionamento adequado não depende somente da instalação do Sistema Operacional e do aplicativo gerenciador da replicação, mas também de questões de segurança. Os dados fornecidos serão lidos pelos destinos certos? Os usuários têm acesso adequado aos dados que estão disponíveis? Defeitos lógicos no servidor podem interromper a replicação se:

- A configuração dos protocolos de rede do servidor forem alteradas inadequadamente, causando interrupção de comunicação com a rede;
- As configurações de *firewall* do servidor forem alteradas de forma a bloquear pacotes de dados destinados à réplica;
- A configuração da replicação for alterada de forma que dados previamente programados para cópia não fiquem mais disponíveis;
- A memória do servidor for insuficiente;
- O disco com as informações replicadas estiver cheio.

A Figura 4 ilustra o servidor mestre em falha. Neste caso, não é gerado um fluxo de dados para o servidor escravo, já que o mestre não funciona adequadamente e não pode produzir o log binário.

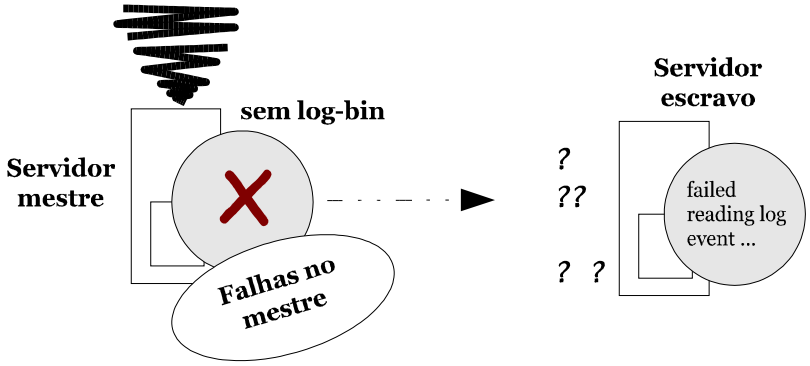

Figura 4 – Representação gráfica de falha de servidor mestre na replicação via MySQL

A conclusão é que o servidor deve ser um equipamento adequado à replicação, atendendo os requisitos mínimos indicados pelo sistema operacional e pelo distribuidor do aplicativo gerenciador da replicação. Deve ser configurado corretamente, disponibilizando os dados corretos para os destinos corretos. Deve ficar em local isolado, para evitar riscos de contato de qualquer tipo, de preferência em uma sala refrigerada.

5.3 - Falhas do servidor escravo

Analogamente ao servidor mestre, um servidor escravo geralmente é um equipamento que sofre riscos de defeitos físicos e lógicos. A principal diferença é que o escravo pode "esperar" que o mestre retorne de uma falha, e continuar lendo os dados depois da recuperação, mas o contrário não é verdade, ou seja, o mestre não pode esperar o escravo retornar de uma falha.

Sem discutir novamente prováveis falhas físicas, pode-se dizer que a principal falha lógica de um escravo é "ocupar indevidamente uma vaga" destinada a um dado replicado. Como foi visto, a replicação é uma cópia organizada e estruturada de dados, cujo destino é tão semelhante à origem, que um escravo pode ser mestre de outros escravos indefinidamente. Isto significa que um mestre disponibiliza os dados para leitura de um escravo, que por sua vez lê e grava tais dados de forma que uma conseqüente disponibilização dos mesmos, a partir deste escravo e para outro escravo, seja possível. Portanto, a "vaga" descrita representa o depósito de dados que são replicados. As várias tabelas replicadas não podem ser alteradas nos escravos, a não ser pela própria replicação. Deve ser observado que os escravos podem ter seus dados alterados desde que algum usuário o faça, indevidamente, através de comandos SQL.

Outro erro que pode acontecer no escravo é que suas bases de dados podem estar diferentes do mestre. Se estiver "faltando" alguma base e/ou tabela no cliente, quando a replicação tentar escrever um dado desta base/tabela, ocorrerá a falha, já que não há onde registrar o dado, no escravo.

Uma outra falha lógica em algum servidor escravo que pode interromper a replicação neste escravo é: Se a estrutura de alguma tabela for alterada no mestre, o comando DDL será gravado no *log* e deveria ser replicado, a não ser que o escravo em questão tenha sofrido alguma outra falha qualquer e não tenha sido recuperado e ligado a tempo para ler este *log*. Logo, assim que este escravo voltar a funcionar, um comando de cópia somente dos dados das tabelas não sincroniza perfeitamente este escravo com o mestre, podendo causar novamente uma falha de replicação neste escravo. No Capítulo 6 é exibido o comando de cópia[1] correto para evitar que isso aconteça.

A Figura 5 exibe a representação gráfica para o escravo em falha. Não importa se o fluxo de dados esteja correto, se o escravo não estiver funcionando adequadamente, o log binário não pode ser lido.

1 mysqldump

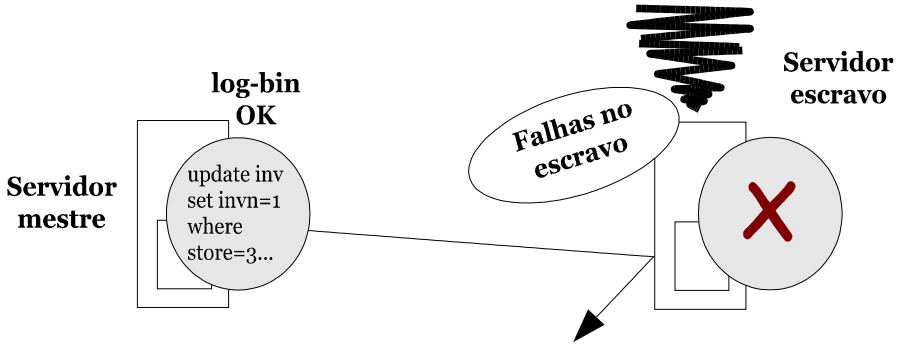

Figura 5 – Representação gráfica de falha de servidor escravo na replicação via MySQL

5.4 - Falhas da aplicação

A replicação pode ser interrompida também se um programa que gerencia os dados (inserção, alteração, remoção) estiver configurado de forma que as alterações ocorram do lado do cliente, de forma não prevista pela replicação. Ou seja, alguma rotina do programa do lado do cliente pode causar inconsistência entre o servidor e o cliente, e no momento que a leitura do *log* binário causar a escrita de um dado que já foi inserido pelo programa, ocorrerá a falha da replicação.

Geralmente, estas rotinas são funções automatizadas do programa, como por exemplo, processamento de vendas, correção de saldos de estoque, recálculos de estatísticas, etc. Portanto, as tabelas que compõe estas funções não devem ser replicadas. Se houver necessidade da replicação, então a(s) rotina(s) devem ser desabilitadas do lado do(s) cliente(s).

Pode-se citar um projeto de sistema mal elaborado como um provável gerador de falhas de aplicação. Isso porque o sistema (ou programa) pode causar inconsistências nas tabelas, seja por falta de Integridade Referencial, código mal escrito e outros. De qualquer forma, projetar o banco de dados adequadamente aumenta a confiabilidade da aplicação. De acordo com Oliveira (2002),

> O projeto do banco de dados está para o sistema da mesma forma que a estrutura do prédio está para o edifício. Se não for dada a devida atenção ao desenho do banco de dados, pode-se comprometer todo o desenvolvimento do sistema.

A Figura 6 ilustra a falha de aplicação na replicação de dados via MySQL. Com a falha, a leitura do log binário é interrompida até que seja feita a recuperação.

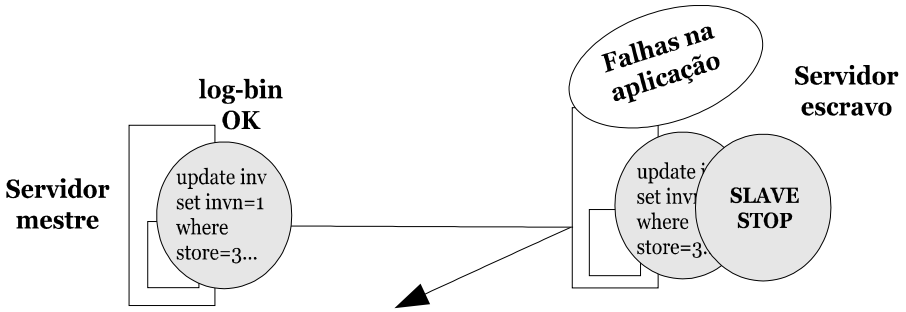

Figura 6 – Representação gráfica de falha de aplicação na replicação via MySQL

5.5 - Falhas do usuário

O próprio usuário pode ser o causador de alguma falha. Por exemplo, ele pode estar trabalhando com uma base de dados em uma máquina cliente, e pode errar uma alteração de um dado, que pode gerar a falha, caso a posição daquele dado seja modificado no servidor, em algum instante após o acidente. Neste caso, se o usuário perceber o erro, ele pode voltar o estado original do dado, antes que a replicação alcance aquela

posição. O mesmo exemplo vale para uma inclusão indevida; neste caso ele deverá remover o registro inserido no cliente, antes que a replicação falhe.

Qualquer tipo de erro de configuração também pode ser classificada como falha de usuário, principalmente se este estiver testando novas versões do SGBD, durante uma replicação em andamento. Boa preparação e conhecimento, além de paciência para testar novos recursos antes de implementá-los, são virtudes de usuários que estatisticamente causam poucas falhas à replicação.

A Figura 7 mostra a falha de usuário na replicação via MySQL. O log binário até pode ser lido pelo servidor escravo, porém não ocorrerá a replicação por causa de configurações incorretas da replicação.

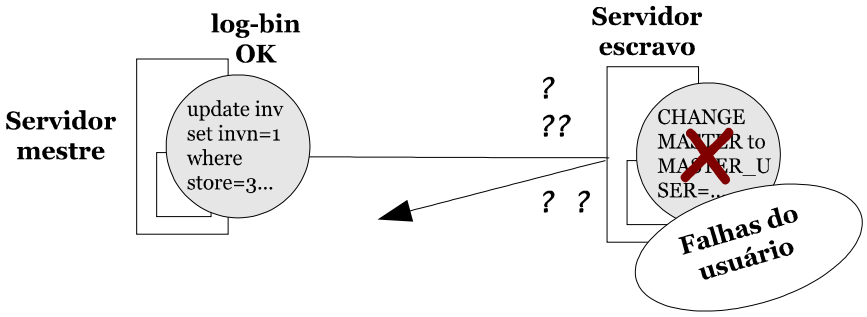

Figura 7 – Representação gráfica de falha de aplicação na replicação via MySQL

6 - Configuração de Replicação do MySQL

Neste capítulo, serão apresentadas as configurações básicas da replicação do MySQL, detalhando os requisitos para a execução da aplicação. A versão utilizada durantee os testes e preparação deste trabalho foi a 4.1, versão estável mais atual naquele momento (Janeiro a Julho de 2005) disponibilizada pelo fabricante MySQL AB. Esta versão ainda está sendo usada porque o nosso ERP[1] não foi testado pelo fornecedor para a versão mais nova.

O Sistema Operacional utilizado foi o Linux Conectiva 10, *kernel* 2.6. Não foram testados outros *kernels*, principalmente por questões de segurança (Novas configurações do *Squid* e do *Iptables* estavam em uso na época no servidor mestre). As configurações dos computadores envolvidos serão descritas no Estudo de Caso, no Capítulo 7.

Após a aquisição e instalação do MySQL, é necessário algumas configurações para que a replicação possa funcionar. O MySQL gerencia

1 ERP: *"Enterprise Resource Planning"* ou "Planejamento de Recursos Empresariais" são sistemas de informações transacionais cuja função é armazenar, processar e organizar as informações geradas nos processos organizacionais agregando e estabelecendo relações de informação entre todas as áreas de uma companhia.

a replicação através de configurações simples, porém o administrador deve possuir bons conhecimentos a respeito do SGBD antes de poder usar esta facilidade. Não por acaso, o capítulo referente à replicação, no livro "MySQL, A Bíblia" (Suehring, 2002), está figurando na parte V, a última, chamada de "Desempenho Avançado". A esse respeito, Suehring (2002), descreve:

> A MySQL AB se esforçou bastante para tornar a configuração da replicação fácil(...). Antes de empreender as tarefas envolvidas de fato na configuração da replicação, reserve algum tempo para o planejamento e a preparação dessas tarefas.

Assim, chama-se a atenção do administrador para que ele planeje bem, em seu caso, como melhor implementar a replicação. Como existem inúmeras variáveis, é complexo apontar caminhos mais gerais, mas espera-se que o texto desta monografia contribua para isso. Para que a replicação seja habilitada, após se logar como superusuário no servidor mestre, deve-se editar o arquivo */etc/my.cnf ,* que deverá conter as seguintes linhas:

```
[mysqld]
...
server-id=1
...
log-bin
...
```

Onde *server-id* identifica o mestre e o *log-bin* habilita a gravação do log binário. Este *log* binário é um arquivo que contém as transações executadas pelo MySQL. Toda vez que se pára o MySQL, e depois o mesmo é reiniciado, um novo arquivo de *log* é gerado. Se for desejado que somente uma base de dados seja replicada, o usuário poderá adicionar a seguinte linha:

```
...
binlog-do-db=dados
...
```

Neste caso, o *log* binário irá registrar somente alterações nas tabelas da base de dados *dados*. Nenhuma outra base de dados será replicada, pois nenhum dado das tabelas de outras bases serão registrados no *log*. O contrário, ou seja, configurar para que uma ou várias bases não sejam replicadas, prevê a inserção das seguintes linhas:

```
...
binlog-ignore-db=home
binlog-ignore-db=usuario
```

```
...
...
```

Neste último exemplo de configuração, as bases de dados *home* e *usuario* não serão replicadas, e todas as outras bases de dados existentes no mestre serão.

Nos servidores escravos, também como superusuário, deve-se editar o arquivo */etc/my.cnf* , que deverá conter as linhas:

```
[mysqld]
...
server-id=2
...
```

Onde *server-id* identifica o 1º. escravo (número 2, pois o 1 é o mestre). Este *id* tem que ser numérico e é único para cada servidor que participa da replicação. Não precisa ser necessariamente sequencial, ou seja, se for desejado, pode-se usar *ids* 1, 2 e 4, em uma replicação com um mestre e dois escravos, por exemplo.

Se houver necessidade de escolher as bases de dados ou tabelas a serem replicadas, ou mesmo escolher aquelas que não serão replicadas, deve-se inserir as seguintes linhas:

```
...
replicate-do-table=dados.vendas
```

```
...
```

Neste caso, somente a tabela *vendas* da base *dados* será replicada. Assim, toda vez que o *log* binário for registrado com qualquer alteração nesta tabela, ocorrerá a replicação. Qualquer outra alteração em outras tabelas não será replicada.

É possível, portanto, particionar a replicação via MySQL até o nível de tabelas, não sendo possível chegar ao nível de colunas e linhas.

```
...
replicate-ignore-table=sqltempo
...
```

Neste caso, a base de dados *sqltempo* não será replicada em nenhuma alteração de seus dados. Todas as outras tabelas de todas as bases do servidor serão replicadas.

Se o escravo for configurado para ser mestre de outros escravos, então o seu arquivo *my.cnf* deverá conter a linha:

```
...
log-bin
...
```

Se ele não registrar o *log* binário, os outros escravos não terão o que ler. A linha acima, conforme já foi citado, habilita a gravação do *log* binário.

Além destas configurações, algumas variáveis do MySQL deverão ser alteradas para que o(s) escravo(s) possa(m) identificar a replicação. Deve-se logar como usuário administrador no MySQL do escravo e na linha de comando deve-se digitar:

```
mysql> change MASTER to MASTER_USER='mestre';
```

Onde 'mestre' é o usuário da replicação. Neste caso, este usuário precisa ser criado dentro de uma sessão do MySQL. Ele precisa ter permissões para SELECT, INSERT, UPDATE e DELETE, nas tabelas que irão ser replicadas. Estas permissões precisam também estar identificadas com o endereço do servidor.

O usuário pode ser o usuário padrão *mysql*, que já possui todos as permissões necessárias, porém é recomendado que se crie um usuário próprio para a replicação.

Continuando:

```
mysql>change MASTER to MASTER_password='123';
```

Onde '123' é a senha a ser utilizada na replicação.

```
mysql> change MASTER to MASTER_HOST='servidor';
```

Onde 'servidor' é o nome ou número IP do servidor na rede.

```
mysql> change MASTER to MASTER_LOG_FILE= 'servidor-
bin.020';
```

Onde 'servidor-bin.020' é o arquivo do mestre que contém o *log* binário. Este arquivo é criado toda vez que o MySQL é iniciado, começando com .001 e sendo incrementado automaticamente pelo próprio SGBD. Ele é gerado por padrão em */var/lib/mysql*, mas este diretório pode ser alterado em */etc/my.cnf*. Não há previsão para a rotação dos *logs* (que podem ser de tamanho considerável), porém isto pode ser feito pelo próprio SO, no caso, utilizando o *logrotate* do Linux.

Continuando:

```
mysql> change MASTER to MASTER_LOG_POS=4;
```

Onde 4 é a posição do arquivo de *log* binário onde a replicação começará a leitura. A posição 4 é igual à linha 4 do arquivo, porque as primeiras 3 linhas contém um cabeçalho de identificação.

Após as alterações, deve-se reiniciar o MySQL e cada servidor deverá ser sincronizado, ou seja, suas bases de dados deverão estar idênticas para que a replicação possa funcionar. Para sincronizar as bases de dados, é necessário usar algum comando de cópia do mestre para cada escravo replicado, de cada base de dados replicada. O comando mais seguro e eficiente é o *mysqldump,* que possui opções que otimizam a cópia e garantem a integridade dos dados. Na linha de comando do Linux, no escravo, é preciso digitar:

```
# mysqldump -h host --opt base1 | mysql base2
```

Onde:

host: IP do servidor mestre;
base1: Base de dados do mestre, que será copiada;
base2: Base de dados do escravo, que será replicada.

Parâmetro opt: O mesmo que usar de uma só vez os parâmetros quick, add-drop-table, add locks, extended-insert, lock-tables. Isto maximiza a velocidade do comando mysqldump.

Neste capítulo foram apresentadas as configurações gerais da replicação de dados via MySQL. Existem mais opções e algumas topologias de replicação alternativas, que podem ser encontradas em bilbiografia especializada, como o livro "MySQL, a Bíblia" (Suehring, 2002) ou no próprio sítio do MySQL (http://www.mysql.com).

7 - Estudo de Caso

7.1 - Descrição do Cenário

Para o estudo desta Monografia foi usado como modelo uma cenário de replicação real que foi implementado em uma empresa, com a finalidade de gerar uma base de dados adequada para o Setor Contábil desta mesma empresa. Foi reportado pelos funcionários que a visualização dos dados estava um pouco lenta. É importante observar que um único servidor, com uma configuração razoável, para tantos usuários (100 no total), realmente estava comprometido em relação à sua performance.

A replicação foi configurada há cerca de dois anos. Existe um único servidor mestre e um único escravo. A replicação utilizada foi mestre/escravo, assíncrona. As máquinas estão em salas e prédios diferentes, distantes aproximadamente 80 metros, e a rede é comum, do

tipo LAN, padrão IEEE[1] 802.1Q. Existem *nobreaks* em ambos os locais, protegendo exclusivamente cada máquina.

As bases de dados replicadas tinham na época dos testes um tamanho total de 3 *gigabytes*, e eram compostas de tabelas do tipo MyISAM. As tabelas replicadas foram 58 no total. A maior delas chegou a ter um tamanho máximo de 120 *megabytes* (somando todos os arquivos, sendo que cada tabela MyISAM é formada por três arquivos, como foi descrito na Seção 3.1). O programa que gravava as informações era um Sistema de Controle Gerencial e Comercial, proprietário, escrito em C e composto de vários módulos, que eram executados através de arquivos binários. O programa era todo em formato texto, sem telas e gráficos especiais.

A versão do MySQL, no mestre, era o 4.0.16, e no escravo, o 4.0.10. Esta diferença foi mantida por um tempo, pois não foi dada a devida importância a ela (não gerou nenhum problema), mas depois de

1 IEEE: *Institute of Electrical and Electronics Engineers*. Fundado em 1884, o IEEE é uma organização composta de engenheiros, cientistas e estudantes. O IEEE é bem conhecida por desenvolver padrões para a indústria eletrônica e de computadores. Em particular, o padrão IEEE 802 para LANs são os mais bem sucedidos.

algum tempo o escravo foi atualizado e ambos os servidores estavam com a versão 4.0.16.

A Contabilidade da empresa necessitava consultar informações de algumas bases somente, motivo principal da adoção da replicação. Contudo, a replicação passou a ser útil também no balanceamento de carga dos servidores mestre e escravo, a medida que o número de usuários foi crescendo. É importante observar que as informações eram geradas por todos os outros setores da empresa, como Vendas, Financeiro, Crediário, Departamente de Compras, Departamento Pessoal, Diretoria, etc.

O servidor escravo (Contabilidade) tinha uma função de exportação de dados, que eram importados por uma outra máquina, com SO Windows®. O programa principal da Contabilidade era executado nesta outra máquina, que importava os dados do servidor escravo. Portanto, qualquer falha na replicação causava falhas no fechamento da Contabilidade. Antes da replicação, todos os dados eram digitados nesta máquina com SO Windows®, o que causava um enorme retrabalho.

7.2 - Aplicação dos Conceitos ao Estudo de Caso

No início da implementação da replicação ocorreram muitos erros, de vários tipos. A dificuldade inicial foi sobre o entendimento do processo e suas configurações, e principalmente nas tentativas de recuperar as falhas. Como não exitiu um ambiente de testes, qualquer falha despendia muito tempo para a correção, já que a sincronia só seria possível no final do dia, quando nenhum usuário estava logado no sistema.

As falhas da rede já eram monitoradas há mais tempo, devido ao uso normal do sistema da empresa, e portanto não interferiram na replicação.

Uma falha do servidor detectada no começo do estudo foi falta de espaço no disco rígido, que causou interrompimento da replicação porque o *log* binário não poderia ser acrescido de mais informações. Este erro foi corrigido com a remoção de *logs* binários antigos, que já haviam sido copiados em fitas DAT para *backup*. O diretório de gravação dos *logs* pode ser alterado, se for necessário. O mesmo pode ser feito para o(s) diretório(s) onde ficam gravadas os arquivos das tabelas replicadas. No

Estudo de Caso, todo o diretório que continha as tabelas replicadas chegou a um tamanho aproximado de 1,3 *gigabytes,* e a média de tamanho de cada *log* era de aproximandamene 30 *megabytes,* porque todo dia, após o *backup* automático dos dados, via cron, no Linux, o serviço mysql era parado e depois reiniciado, gerando um novo arquivo de *log* binário.

Outras falhas comuns no servidor eram problemas de memória, que causavam *kernel panic* no sistema operacional. Após um período de testes, foi resolvido que a melhor solução era adquirir uma nova máquina, com um processador mais potente e mais memória RAM.

O servidor escravo apresentou falhas de *hardware* com muito mais frequência, por ser uma máquina mais antiga e por estar num ambiente pouco protegido, sem isolamento adequado de poeira e temperatura. Toda falha no escravo era recuperada no final do dia, conforme citado anteriormente, o que causava transtornos constantes na Contabilidade.

O servidor escravo também apresentou problemas com as bases de dados configuradas na replicação. Algumas bases foram configuradas para serem replicadas, porém elas só existiam no mestre, por questões de configuração do cenário. Isto causou interrupção na replicação, já que o

escravo não tinha como escrever os dados lidos daquelas bases. Esta falha foi corrigida com a reconfiguração do cenário, adequando as bases corretas para a replicação.

As falhas mais complicadas de se identificar e corrigir eram da aplicação. O programa, como foi dito, apresentava muitos módulos e muitas tabelas, algumas com mais de cem campos. Algumas rotinas do programa eram automatizadas de forma a atender os processos e retornar com as soluções desejadas. O problema aparecia sempre que estas rotinas escreviam dados em tabelas replicadas no escravo, o que causava a interrupção da replicação. O erro identificado aparecia quando o usuário, logado no MySQL, digitava:

```
mysql> show slave status \G;
```

Este comando pode mostrar qual linha do *log* binário o MySQL tentou escrever no escravo, sem sucesso. Mas, dependendo do tipo de falha, pode mostrar simplesmente que não existe comunicação entre o escravo e o mestre. No primeiro caso, o erro é reportado com a chave primária do registro cuja escrita no escravo falhou. Fica relativamente

fácil para o usuário localizar a tabela e o registro, no escravo, para poder tomar uma decisão: Remover o registro no escravo, que ocupa a posição que deveria ser ocupada pelo registro da replicação, ou recomeçar a replicação, sincronizando as máquinas novamente.

Durante o Estudo de Caso, foi bem mais comum recomeçar a replicação a partir de uma nova sincronização. Foi considerado mais seguro pelo fato de que uma remoção indevida no escravo poderia não resolver o problema. Porém, uma ou duas vezes foi usado o artifício da "limpeza" do registro indevido, quando foi possível fazê-lo.

7.3 - Avaliação Crítica do Estudo de Caso

A medida que as novas configurações e sincronizações eram feitas, com ajuda de bibliografia, os usuários participantes foram identificando mais problemas e eliminando-os, um a um, até que a replicação chegou em um nível satisfatório, de poucos erros ou falhas. Hoje, por exemplo, a replicação não apresenta falhas (a última reportada foi em Novembro de 2005).

O Estudo de Caso permitiu uma conclusão óbvia: a implementação de uma replicação deve ser planejada com muito cuidado. Desde a parte de estrutura de máquinas, equipamentos e rede, até a execução da replicação, tudo deve ser planejado e organizado, para não haver surpresas desagradáveis, como perda de informações e até mesmo problemas mais graves com o MySQL, como a indisponibilidade permanente do serviço. Isto pode ocorrer se alguma configuração indevida for executada, necessitando de reinstalação do MySQL.

Tabelas com índices danificados e registros com informações inadequadas também podem advir de mal uso da replicação via MySQL, que prejudica todo o sistema, não só a replicação. Este ponto foi identicado e analisado durante o estudo, e foi considerado grave pelas consequências negativas.

Outro ponto que não foi bem esclarecido a princípio foi a versão do MySQL em cada máquina. Se forem diferentes, podem causar problemas, por causa de opções diferentes de configurações disponíveis em cada versão. Foi feito um *update* do MySQL em cada servidor, para a mesma versão, no caso a mais estável na época, a 4.0.16.

Portanto, a replicação é um processo seguro e eficiente, já que todos os problemas identificados foram de origem externa do MySQL. Em resumo, a equipe envolvida não obteve dados para reportar um possível *bug* na replicação.

8 - Conclusão

Depois das análises do Estudo de Caso, e após as leituras dos fóruns do sítio oficial do MySQL e de bibliografia especializada, pode-se concluir que as falhas identificadas foram de origem externa ao funcionamento da replicação em si.

A conclusão é aceitável pois o objetivo deste trabalho era identificar e recuperar as possíveis falhas na replicação via MySQL, através de um Estudo de Caso. Atualmente a empresa em que foi executada a replicação está funcionando adequadamente, sem problemas que não possam ser resolvidos pela equipe de TI.

A motivação para a confecção deste documento estendeu, na verdade, para um resultado maior e muito bem aproveitado pelo autor, tanto na empresa onde trabalha quanto em cursos e palestras que o mesmo ministra. Os resultados excederam as expectativas, enfim.

Portanto, o objetivo foi cumprido, tanto da parte do CPD da empresa, pelo bom funcionamento da replicação, quanto deste trabalho, visto que foram relatados os erros idenficados e seu tratamento.

Referências Bibliográficas

Bosnic, I. Integridade Referencial no MySQL. 30 de Janeiro de 2005. [on-line]. URL: http://www.softscience.com.br/modules.php?name=News&file=print&sid=25
Arquivo capturado em 10 de Setembro de 2006.

Date, C.J. (2000). *Introdução a sistemas de bancos de dados* / C.J. Date ; tradução Vandenberg D. de Souza. Rio de Janeiro: Campus.

Duarte, E. / SQL Magazine. *Implementando Integridade Referencial no MySQL.* 2004. [on-line]. URL: http://www.sqlmagazine.com.br/Colunistas/eber/06_IntegridadeReferencial.asp
Arquivo capturado em 10 de Setembro de 2006.

Institucional Itec / Grupo Itautec. *Cluster por Replicação de Dados*. 2002. [on-line]. URL: http://www.itec.com.br/semanal/NEWAS400Clustering1.htm
Arquivo capturado em 10 de Setembro de 2006.

Gordon, S.R, Gordon, J.R. (2006). *Sistemas de Informação: Uma abordagem gerencial* / Steven R. Gordon, Judith R. Gordon; tradução Oscar Rudy Kronmeyer Filho. Rio de Janeiro: LTC.

Mello, R.S. *SGBD*. 2006. [on-line]. URL: http://www.inf.ufsc.br/~ronaldo/ine5323/2-sgbd.pdf
Arquivo capturado em 10 de Setembro de 2006.

MySQL AB. *Sítio Oficial.* 2006. [on-line]. URL: http://www.mysql.com

MySQL AB. *Sítio Oficial / Fóruns.* 2006. [on-line]. URL: http://www.forums.mysql.com

MySQL AB. *Sítio Oficial / Manual de Referência do MySQL 4.1*. 2006. [on-line]. URL: http://dev.mysql.com/doc/refman/4.1/pt/todo.html

Oliveira, A.F. *Estudo Adaptativo Para Incorporação de Transações no Sistema de Comunicação Aberto – DiretoGNU*. Maio de 2002. [on-line]. URL: http://www.inf.lasalle.tche.br/direto/docs/Anexo2Transacoes.PDF Arquivo capturado em 10 de Setembro de 2006.

Ricarte, I.L.M. *Programação Orientada a Objetos: Uma abordagem com Java.* 15 de Outubro de 2002. [on-line]. URL: http://www.dca.fee.unicamp.br/cursos/PooJava/javadb/bdrel.html. Arquivo capturado em 10 de Setembro de 2006.

Souza, M.A. (2004). *SQL, PL/SQL, SQL*PLUS: manual de referência completo e objetivo.* Rio de Janeiro: Editora Ciência Moderna Ltda.

Suehring, Steve (2002). *MySQL, a Bíblia* / Steve Suehring ; tradução Edson Furmankiewics. Rio de Janeiro: Elsevier.

Spenik, Mark; Sledge, Orryn (1996). *Microsoft SQL Server DBA Survival Guide*, Indianapolis: Sams Publishing.

Simões, I.E; Torres, G.M. (2003). *Alta Disponibilidade em Servidores.* 27 de Fevereiro de 2003. [on-line]. URL: http://www.eee.ufg.br/cepf/pff/2002/pf2002_04.pdf Arquivo capturado em 10 de Setembro de 2006.

Tanenbaum, A.S. (2003). *Redes de Computadores* / Andrew S. Tanenbaum; tradução Vandenberg D. de Souza. Rio de Janeiro: Elsevier.