



IESB – Centro Universitário de Brasília
Pós-graduação em Banco de Dados com ênfase em SGBD Oracle

Replicação entre Bancos de Dados Heterogêneos
Um estudo de caso para replicação e distribuição de dados entre
os bancos Oracle e PostgreSQL

André Luiz Alves Baracho de Freitas (1031004017)

Brasília-DF
Julho de 2011

André Luiz Alves Baracho de Freitas (1031004017)

**Replicação entre Bancos de Dados Heterogêneos
Um estudo de caso para replicação e distribuição de dados entre
os bancos Oracle e PostgreSQL**

Trabalho de Conclusão de Curso
apresentado ao Curso de Pós-graduação com
ênfase em SGBD Oracle do Instituto de
Educação Superior de Brasília, como
requisito parcial para obtenção do grau de
especialista, sob a orientação do professor
Eder Couto.

Brasília-DF
Julho de 2011

André Luiz Alves Baracho de Freitas (1031004017)

Replicação entre Bancos de Dados Heterogêneos
Um estudo de caso para replicação e distribuição de dados entre
os bancos Oracle e PostgreSQL

Trabalho de Conclusão de Curso apresentado e aprovado em __/__/__, pela banca examinadora constituída pelos professores:

Prof. Esp. Eder Couto - orientador

Prof. Esp. Paulo Lima Machado

Prof. Esp. Rogério Bragança Borges

AGRADECIMENTOS

Agradeço a Deus pelo infinito amor, fonte de inspiração e fazer-me acreditar que todo o sonho é possível de ser realizado.

A minha esposa Patrícia pelo amor, paciência, incentivo e compreensão pelos momentos de lazer que tivemos que abdicar para que este trabalho fosse concluído.

Aos meus pais André e Sônia pelos diversos momentos de carinho, amor, incentivo e ensinamento do caminho a ser seguido, amo vocês.

A minha irmã Anna Paula pelas constantes alegrias e incentivos nos momentos difíceis.

Ao professor Eder Couto, pela atenção, incentivo e paciência.

Aos Diretores da Object Sistemas, pelo apoio, paciência e confiança em mim depositado para que este trabalho fosse realizado.

"Ninguém baterá tão forte quanto à vida.
Porém, não se trata de quão forte pode
bater, se trata de quão forte pode ser
atingido e continuar seguindo em frente. É
assim que a vitória é conquistada."
(ROCK BALBOA)

RESUMO

As organizações a cada dia buscam ampliar seus negócios disseminando a informação, para que isto seja possível a informação precisa chegar de modo que seja rápida, íntegra e que esteja sempre disponível. Portanto, alguns métodos são utilizados com o objetivo de transmitir essa informação, encaminhando ao seu destino e fazendo com que permaneça disponível ao responsável pela sua emissão.

A replicação de banco de dados é um dos métodos que pode auxiliar na disseminação da informação tornando-a sempre acessível e disponível, pois, permite que a informação esteja disponível independente de falhas de rede. Outro fator que a replicação pode auxiliar na implantação de políticas de backup, fazendo com que a informação possa sempre estar disponível a quem tenha direito de acessá-la.

O Objetivo deste trabalho é apresentar a replicação entre as bases de dados heterogêneas. A replicação foi desenvolvida por meio de um estudo de caso entre as bases de dados heterogêneas Oracle e Postgresql. Esta solução de replicação foi implantada no Hospital Geral do Brasil, com o objetivo de melhorar o fluxo de atendimento daquela unidade de saúde.

Palavras-chaves: Replicação de Dados, Oracle, Postgresql, Banco de Dados

ABSTRACT

The organizations seek to expand their every day business disseminating information so that it is possible to get accurate information so that it is rapid, integrated and always available. Therefore, some methods are used in order to provide this information, routing to your destination and making available to remain responsible for their issue.

Replication database is one of the methods that can assist in the dissemination of information made it accessible and always available, because it allows information to be available regardless of network failures. Another factor that replication can help implement backup policies, so that information can always be available to those who have right to access it.

The objective of this study is the replication between heterogeneous databases. Replication has been developed through a case study among heterogeneous databases Oracle and Postgresql. This replication solution was implemented at the Hospital General of Brazil in order to improve the flow of care that health unit.

Keywords: Data Replication, Oracle, PostgreSQL, Database

LISTA DE ILUSTRAÇÕES

Figura 1: Arquitetura do Oracle	19
Figura 2: Arquitetura do Postgresql.....	21
Figura 3: Replicação Passiva.....	25
Figura 4: Replicação ativa.....	25
Figura 5: Processo de captura.....	29
Figura 6: Diagrama Físico do Oracle.....	38
Figura 7: Diagrama Físico do Postgresql.....	38
Figura 8: Criação do Usuário replicador no Oracle	39
Figura 9: Criação de Usuário no Postgresql Filial SP.....	40
Figura 10: Configuração do arquivo de conexão com o Banco de dados.	41
Figura 11: Criação de trigger de replicação do servidor Oracle.....	42
Figura 12: Cadastro dos servidores escravos de replicação.....	42
Figura 13: Cadastro das tabelas e operações DML realizadas.	43
Figura 14: Execução do serviço do replicador.....	44
Figura 15: Criação do Identificador seqüencial no SGBD Oracle da Matriz.	45
Figura 16: Criação do Identificador seqüencial no SGBD Postgresql na Filial Brasília.	46
Figura 17: Criação do Identificador seqüencial no SGBD Postgresql na Filial São Paulo.....	46
Figura 18: Criação de Paciente no SGBD Oracle.....	47
Figura 19: Criação de Paciente no SGBD Postgresql na Filial Brasília.	47
Figura 20: Criação de Paciente no SGBD Postgresql na Filial São Paulo.	48
Figura 21: Atualização de Paciente no SGBD Oracle na Matriz.	48

Figura 22: Atualização de Paciente no SGBD Postgresql na Filial Brasília.....	48
Figura 23: Atualização de Paciente no SGBD Postgresql na Filial São Paulo. .	49
Figura 24: Exclusão de Paciente no SGBD Oracle na Matriz.	49
Figura 25: Exclusão de Paciente no SGBD Postgresql na Filial Brasília.	50
Figura 26: Exclusão de Paciente no SGBD Postgresql na Filial São Paulo.....	50
Figura 27: Topologia dos ativos de rede.....	51
Figura 28: Tabela de conversão de tipo de dados	54

LISTA DE TABELAS

Tabela 1 – comparação entre Objectmmrs e DBMoto.....	36
Tabela 2 - comparação de tipos de dados.....	54

LISTA DE ABREVIATURAS E SIGLAS

CPU – Central Processing Unit

DDL – Data Definition Language

DML – Data Manipulation Language

DTI – Diretoria de Tecnologia da Informação

HD – Hard Disk

HGB – Hospital Geral do Brasil

IBGE – Instituto Brasileiro de Geografia e Estatística

LCR – Logical Change Records

NTP – Network Time Protocol

OBJECTMMRS – Object Multi-Master Replication Server

RAM - Random Access Memory

SGA – System Global Área

SGBD – Sistema Gerenciador de Banco de Dados

SQL – Structured Query Language

SSL – Secure Socket Layer

SUMÁRIO

1	INTRODUÇÃO.....	12
1.1	Delimitações do Tema	12
1.2	Problema	12
1.3	Justificativa	13
1.4	Tema Proposto	14
1.5	Objetivo Geral	14
1.6	Objetivos Específicos	14
1.7	Resultados Esperados	15
1.8	Estrutura do Trabalho.....	15
2	REVISÃO DA LITERATURA.....	17
2.1	Banco de dados	17
2.2	Sistema de Gerenciamento de Banco de Dados	18
2.3	Arquitetura Oracle.....	18
2.4	Arquitetura Postgresql	20
2.5	Transação.....	21
2.6	Controle de concorrência.....	23
2.7	Replicação	23
2.7.1	Técnicas de Replicação.....	24
2.7.2	Modelo de Replicação.....	25
2.7.3	Métodos de Replicação	26
2.7.4	Conflitos de Replicação	27
2.7.5	Replicação utilizada no Oracle e Postgresql	28
2.8	Software de replicação	30

	11
2.8.1 Oracle GoldenGate	30
2.8.2 Oracle Streams.....	30
2.8.3 Postgres-r.....	30
2.8.4 DBMoto.....	31
2.8.5 ObjectMMRS	31
3 METODOLOGIA	32
3.1 Delineamento da Pesquisa	32
3.2 Contextualização do Estudo de Caso.....	33
3.3 Procedimentos de Pesquisa.....	33
4 ESTUDO DE CASO	34
4.1 SGBD Utilizado	34
4.2 Infra-estrutura.....	34
4.3 Descrição da Solução.....	35
4.4 Criação da Replicação	38
4.5 Topologia dos Ativos	50
4.6 Desempenho do Replicador	51
4.7 Funcionamento e Manutenção do Replicador	51
5 ANÁLISES DO ESTUDO DE CASO	53
6 CONCLUSÃO.....	55
7 RECOMENDAÇÕES PARA TRABALHOS FUTUROS.....	56
8 REFERÊNCIAS.....	57
ANEXO A – TRIGGER CRIADA PELO SOFTWARE OBJECTMMRS.....	59
GLOSSÁRIO	61

1 INTRODUÇÃO

1.1 Delimitações do Tema

A informação é o ativo mais importante para as organizações. Importância esta que caso seja perdida ou não acessada por indisponibilidade do sistema pode gerar prejuízos irreparáveis. Estes prejuízos podem ser ocasionados por falhas em rede de computadores, atentados criminosos e desastres naturais como chuvas, terremotos, furacões e etc.

As organizações têm adotado a replicação de banco de dados como solução para minimizar impactos de perda de dados e aumentar a disponibilidade do sistema. Estes dados são transferidos para outros locais em diferentes localizações representando a cópia fiel dos dados, sendo assim, podendo ser acessado de diferentes regiões.

Os Sistemas Gerenciadores de Bancos de dados (SGBD), contam com este auxílio de replicação nativa, portanto, vale lembrar que nem todas as soluções já incorporadas nestes bancos podem atender a necessidade, ser de fácil adequação e entendimento ao usuário deste recurso. Neste estudo de caso será apresentado, um comparativo entre as ferramentas de replicação nativas dos SGBD'S Oracle e PostgreSQL e de um software proprietário de replicação da Object Sistemas, a qual atendeu a necessidade da organização hospitalar.

1.2 Problema

O Hospital Geral do Brasil é uma rede hospitalar com sede no estado do Rio de Janeiro, atualmente possui duas filiais situadas em São Paulo e Brasília que

realizam atendimento de aproximadamente 400 pacientes. Estes atendimentos são realizados através de software único em cada unidade hospitalar.

As unidades hospitalares dispõem de centros de tecnologia subordinados a Diretoria de Tecnologia da Informação – DTI/HGB, que são responsáveis por gerenciar todo o ambiente tecnológico das unidades hospitalares, bem como, realizar treinamento e auditoria dos softwares utilizados e propor soluções de melhoria.

A DTI/HGB ao observar o fluxo de atendimento das unidades e constatar indícios de demora no atendimento, identificou a necessidade de desenvolvimento de um software que utilizasse um prontuário único de atendimento aos seus pacientes, pois a demora estava sendo ocasionada pela existência de softwares independentes em cada unidade hospitalar.

O paciente ao ser transferido para outra unidade e caso não estivesse cadastrado no sistema, era gerado um novo prontuário e migradas as informações da unidade de origem para este. A entrega de exame era realizada de forma equivocada, pois o prontuário que o cliente informava muitas vezes era de outro paciente pela diversificação de softwares utilizados nas unidades hospitalares. Portanto é necessário adotar uma solução de replicação de dados que sincronize as informações do prontuário entre todas as unidades de saúde.

1.3 Justificativa

As unidades hospitalares dispõem de software de cadastro e de atendimento a pacientes independente dos softwares de outras unidades. Desse modo, cada unidade possui as informações dos pacientes que sempre procuram atendimento nas mesmas. Caso um paciente necessite ser transferido para outra unidade da

rede hospitalar e este não possui prontuário no hospital para o qual foi transferido, é aberto um novo prontuário e são migrados os dados do prontuário antigo. Portanto é de fundamental importância a implantação de um software que atenda as diversas unidades hospitalares com uma solução de replicação de banco de dados, podendo assim proporcionar um ambiente com tolerância a falhas de rede, implantação de políticas de backup e fornecimento de informações à alta Administração para possíveis tomadas de decisão.

1.4 Tema Proposto

O tema proposto avalia os replicadores nativos dos bancos de dados Oracle 10g e Postgresql, e de softwares proprietários de replicação de dados, buscando propor uma solução de replicação para atendimento da necessidade das unidades hospitalares.

1.5 Objetivo Geral

Avaliar as ferramentas nativas dos bancos de dados Oracle 10g e Postgresql, e de softwares proprietários de replicação de dados, e propor a implantação do qual melhor atender a necessidade das unidades hospitalares.

1.6 Objetivos Específicos

- Identificar as funcionalidades dos replicadores nativos dos bancos de dados Oracle e Postgresql e de softwares proprietários de replicação de dados;
- Propor implantação do replicador que atenda a necessidade das unidades hospitalares;

- Descrever o ambiente de estudo de caso com a replicação entre os bancos de dados Oracle e Postgresql.
- Analisar os dados replicados entre os bancos de dados heterogêneos.

1.7 Resultados Esperados

Após implantação de replicação de dados, é esperada uma melhora no atendimento, alta disponibilidade em termos de tolerância a falhas de rede, e dados sempre atualizados para os gestores que necessitem de informações a nível gerencial para uma possível tomada de decisão de forma rápida e precisa.

1.8 Estrutura do Trabalho

Este trabalho está estruturado em seis capítulos divididos da seguinte forma:

O capítulo um é a parte introdutória identificando o problema encontrado, a justificativa para o tema escolhido, seguindo com a apresentação dos objetivos gerais e específicos e os resultados esperados com o desenvolvimento do trabalho.

O capítulo dois aborda a revisão literária apresentando os conceitos de banco de dados, sistemas de gerenciamento de banco de dados, arquiteturas do Oracle e Postgresql, transação, controle de concorrência, replicação e software de replicação.

O capítulo três define os procedimentos de pesquisa utilizada para desenvolvimento deste estudo de caso.

O capítulo quatro apresenta o estudo de caso de implantação de um mecanismo de replicação nas unidades hospitalares do Hospital Geral do Brasil, utilizando um software de replicação Objectmms.

O capítulo cinco apresenta a análise do estudo de caso, identificando os procedimentos adotados para realizar a replicação.

O capítulo seis apresenta a conclusão deste trabalho os benefícios encontrados e os impactos minimizados com a implantação deste mecanismo.

E por fim, no capítulo sete é apresentado às oportunidades de trabalhos futuros a serem realizados.

2 REVISÃO DA LITERATURA

2.1 Banco de dados

Date (2004) descreve um banco de dados como uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa. O termo empresa, aqui, é simplesmente um termo genérico para qualquer organização comercial, científica, técnica ou outra organização razoavelmente autônoma. Uma empresa poderia ser um único indivíduo com um pequeno banco de dados pessoal, ou uma corporação ou grande empresa completa com um enorme banco de dados compartilhado, ou qualquer coisa entre esses extremos.

Garcia-Molina, Ullman e Widom (2001), definem que o banco de dados é uma coleção de dados organizados para facilitar o acesso e a modificação, preservada durante um longo período.

Segundo Elmasri e Navathe (2005), o banco de dados representa alguns aspectos do mundo real, sendo chamado, às vezes, de minimundo ou de universo de discurso. As mudanças no minimundo são refletidas em um banco de dados. Este é uma coleção lógica e coerente de dados com algum significado inerente. Uma organização de dados ao acaso não pode ser corretamente interpretada como um banco de dados. Portanto, um banco de dados é projetado, construído e povoado por dados, atendendo a uma proposta específica. Possui um grupo de usuários definidos e algumas aplicações preconcebidas, de acordo com o interesse desse grupo de usuários.

2.2 Sistema de Gerenciamento de Banco de Dados

Silberschartz, Korth e Sudarshan (2005), explicam que um Sistema Gerenciador de Banco de Dados (SGBD), é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados.

Conforme Date (2004), O SGBD é o software que trata de todo o acesso ao banco de dados. Conceitualmente, o que ocorre é o seguinte:

- Um usuário faz um pedido de acesso usando uma determinada sublinguagem de dados, SQL.
- O SGBD intercepta o pedido e o analisa.
- O SGBD, por sua vez, inspeciona o esquema externo ou as versões objeto desse esquema para esse usuário, o mapeamento externo/conceitual correspondente, o esquema conceitual, o mapeamento conceitual/interno e a definição do banco de dados armazenado.
- O SGBD executa as operações necessárias sobre o banco de dados armazenado.

2.3 Arquitetura Oracle

Watson (2010) explica que o servidor Oracle é composto de duas entidades: a instância e o banco de dados de acordo com as definições abaixo.

Instância – é composta de estruturas de memória e processos, sua existência é temporária, na memória RAM e nas CPU's, quando é desligada a instância em execução, todos os vestígios da sua existência são perdidos, os processos que compõem a instância são conhecidos como processos de segundo plano ou

processos de background, por que estão presentes e em execução o tempo todo enquanto a instância está ativa. As estruturas de memória, implementadas em segmentos de memória compartilhados fornecidos pelo sistema operacional, são conhecidos como a área de sistema global ou SGA, essa área é alocada na inicialização da instância e liberada no desligamento. As sessões de usuário são compostas de um processo de usuário executando localmente na máquina do usuário, conectando a um processo de servidor executando localmente na máquina do servidor.

Banco de dados – é composto de arquivos físicos no disco. Esteja ele em execução ou parado, os arquivos permanecem. Estes arquivos são os arquivos de dados, os arquivos de redo log e os arquivos de controle.

A Figura 1 apresenta com mais detalhes a arquitetura do SGBD Oracle

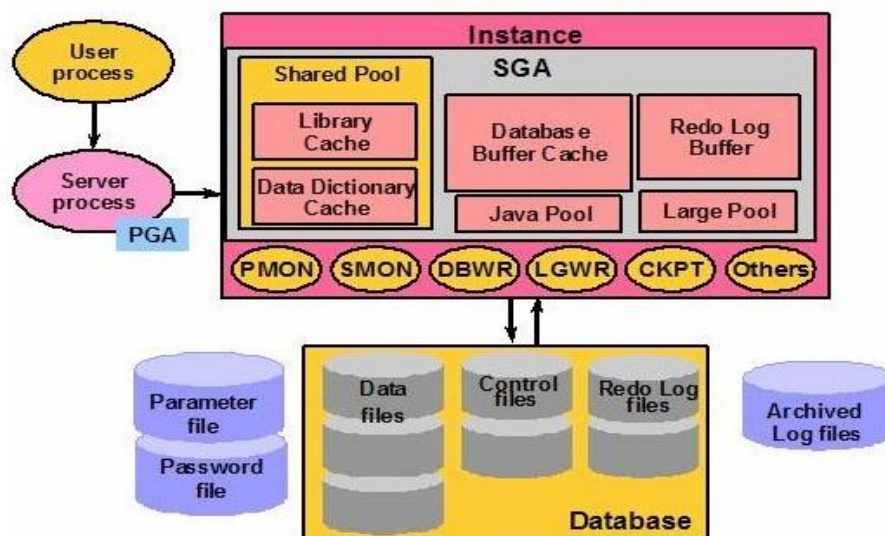


Figura 1: Arquitetura do Oracle

2.4 Arquitetura Postgresql

Conforme documentação oficial, o Postgresql utiliza o modelo cliente-servidor.

Uma sessão do Postgresql consiste nos seguintes processos cooperando entre si:

- Um processo servidor, que gerencia os arquivos de banco de dados, recebe conexões dos aplicativos cliente com o banco de dados, e executa ações no banco de dados em nome dos clientes. O programa servidor de banco de dados se chama postmaster.

- O aplicativo cliente do usuário (frontend) que deseja executar operações de banco de dados. Os aplicativos cliente podem ter naturezas muito diversas: o cliente pode ser uma ferramenta no modo caractere, um aplicativo gráfico, um servidor Web que acessa o banco de dados para mostrar páginas Web, ou uma ferramenta especializada para manutenção do banco de dados.

O servidor Postgresql pode tratar várias conexões simultâneas de clientes. Para esta finalidade é iniciado um novo processo para cada conexão. Deste ponto em diante, o cliente e o novo processo servidor se comunicam sem intervenção do processo postmaster original. Portanto, o postmaster está sempre executando e aguardando por novas conexões dos clientes, enquanto os clientes e seus processos servidor associados surgem e desaparecem.

Todos os processos postgres compartilham duas áreas de memória:

- O buffer cache armazena blocos lidos ou modificados de tabelas e índices.
- O write-ahead log armazena temporariamente o log de transações, até que ele possa ser armazenado em disco.

Além disso, cada processo postgres tem uma área individual para operações de ordenação.

A figura 2 apresenta com mais detalhes a arquitetura do SGBD Postgresql.

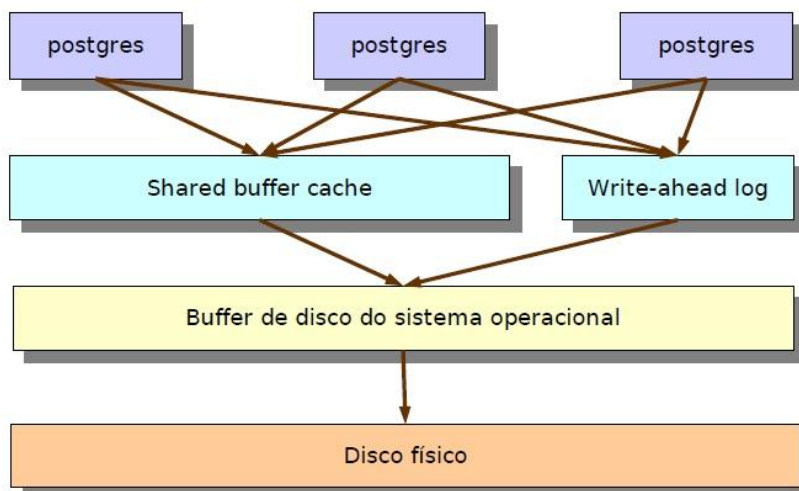


Figura 2: Arquitetura do PostgreSQL.

2.5 Transação

Segundo Coulouris, Dollimore e Kindberg (2007), o objetivo das transações é garantir que todos os objetos gerenciados por um servidor permaneçam em um estado consistente ao serem acessados por várias transações na presença de falhas do servidor. As transações tratam falhas por colapso de processos e falhas por omissão na comunicação, mas não como qualquer tipo de comportamento arbitrário.

Os objetos que podem ser recuperados depois que seu servidor falha são chamado de objetos recuperáveis. Em geral, os objetos gerenciados por um servidor podem ser armazenados em memória volátil, ou em memória persistente. Mesmo que os objetos sejam armazenados em memória volátil, o servidor pode usar memória persistente para armazenar informações suficientes para que, no caso de falhas do processo servidor, o estado dos objetos seja recuperado. Isso permite que os servidores tornem os objetos recuperáveis. Uma transação é especificada por um cliente como um conjunto de operações sobre os objetos a serem executadas como

uma unidade indivisível pelos servidores que estão gerenciando esses objetos.

Ozsu e Valduriez (2001) afirmam que a transação é uma unidade de computação consistente e confiável. Assim, intuitivamente, uma transação toma um banco de dados, executa uma ação sobre ele e gera uma nova versão do banco de dados, provocando uma transição de estado. Isso é semelhante ao que faz uma consulta, mas, se o banco de dados era considerado consistente antes da execução da transação, independente do fato de que a transação pode ter sido executada de forma concorrente com outras e podem ter ocorrido falhas durante sua execução. Em geral, considera-se que a transação é constituída por uma seqüência de operações de leitura e gravação sobre o banco de dados, juntamente com etapas de computação. Nesse sentido, uma transação pode ser imaginada como um programa com consultas de acesso ao banco de dados incorporado.

Os aspectos de consistência e confiabilidade das transações se devem a quatro propriedades:

- **Atomicidade** – se refere ao fato que uma transação é tratada como uma unidade de operação. Assim, ou todas as ações da transação são concluídas, ou nenhuma delas se completa.
- **Consistência** – Um banco de dados está em um estado consistente se ele obedece a todas as restrições de consistência definidas sobre ele.
- **Isolamento** – é a propriedade das transações que exige que cada transação veja um banco de dados consistente em todos os momentos, uma transação em execução não pode revelar seus resultados a outras transações concorrentes antes de se consolidar.
- **Durabilidade** – se refere à propriedade das transações que assegura que, uma vez que a transação se consolida ou se completa, seus resultados se

tornam permanentes e não podem ser apagados do banco de dados. Por conseguinte, o SGBD assegura que os resultados de uma transação sobreviverão após a ocorrência de falhas do sistema.

2.6 Controle de concorrência

Conforme Ozsu e Valduriez (2001), controle de concorrência trata das propriedades de isolamento e consistência das transações. O mecanismo de controle distribuído da concorrência de um SGBD distribuído assegura que a consistência do banco de dados, seja mantida em um ambiente distribuído multiusuário. Se as transações são consistentes internamente, isto é, não violam nenhuma das restrições de consistência, a maneira mais simples de alcançar esse objetivo é executar cada transação sozinha, uma após a outra.

2.7 Replicação

Coulouris, Dollimore e Kindberg (2007) descrevem a replicação como a chave para prover alta disponibilidade e tolerância a falhas em sistemas distribuídos. A alta disponibilidade é um tópico de crescente interesse principalmente com a atual tendência em direção à computação móvel e, conseqüentemente, à operação desconectada. A tolerância a falhas é uma preocupação permanente dos serviços fornecidos em sistemas onde a segurança é crítica e em outros tipos importantes de sistemas.

As motivações para a replicação são:

- **Melhoria de desempenho** – a colocação dos dados na cache em clientes e servidores é uma maneira conhecida de melhorar o desempenho,

navegadores e servidores proxies colocam na cache cópias de recursos web para evitar a latência da busca desses recursos no servidor de origem, além disso, às vezes os dados são replicados de forma transparente entre vários servidores de origem no mesmo domínio.

- **Maior disponibilidade** – os usuários exigem que os serviços sejam de alta disponibilidade, isto é, a proporção do tempo durante a qual o serviço está acessível com tempo de resposta razoável deve ser próxima a 100%. Fora os atrasos decorrentes dos conflitos do controle de concorrência pessimista, os fatores relevantes para a alta disponibilidade são falhas no servidor e/ou particionamento da rede e operação desconectada.

2.7.1 Técnicas de Replicação

Segundo Coulouris, Dollimore e Kindberg (2007), esta técnica pode ser apresentada da seguinte forma:

- Replicação Passiva - nesta forma, os front ends se comunicam somente com o gerenciador de réplica primário para obterem o serviço. O gerenciador de réplica primário executa as operações e envia cópias dos dados atualizados para os backups. Se o primário falhar, um dos backups será promovido para atuar como primário (figura 3).

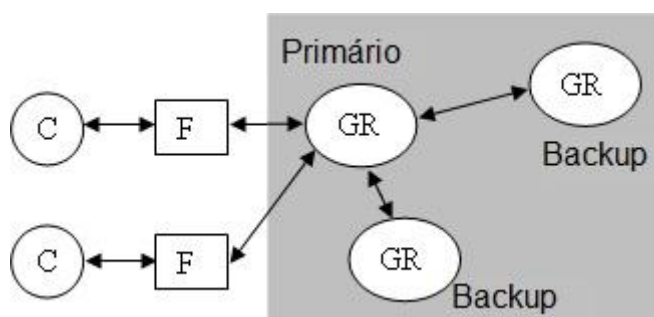


Figura 3: Replicação Passiva

Replicação Ativa – na replicação ativa para tolerância a falhas os gerenciadores de réplica são máquinas de estado que desempenham papéis equivalentes e são organizados com um grupo. Os front ends enviam suas requisições por multicast para o grupo de gerenciadores de réplica, e todos os gerenciadores de réplica processam a requisição independentemente, mas de forma idêntica, e respondem. Se qualquer gerenciador de réplica falhar, isso não tem nenhum impacto sobre o desempenho do serviço, pois os gerenciadores de réplica restantes continuam a responder normalmente (figura 4).

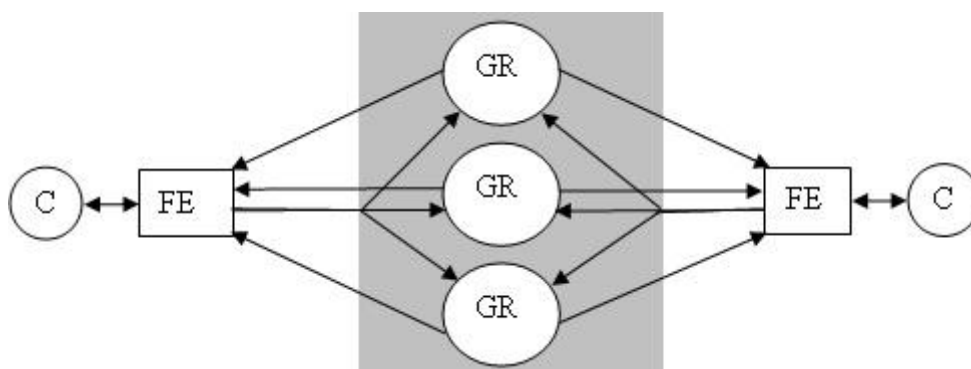


Figura 4: Replicação ativa.

2.7.2 Modelo de Replicação

Segundo Oliveira (2006), além da técnica de replicação, outro ponto importante na replicação de dados é o modelo adotado. Existem dois modelos básicos de replicação:

- **Replicação síncrona:** a replicação de dados é feita no instante em que

uma determinada cópia é alterada. Nessa situação, todas as cópias ou replicações de dados serão feitas no instante da sincronização e consistência. Se alguma cópia do banco é alterada, essa mudança será imediatamente aplicada a todos os outros bancos dentro da transação. Os servidores replicados cooperam usando estratégias sincronizadas e protocolos especializados de réplica para manter os conjuntos de dados replicados coerentes. Esse modelo de replicação demanda conexão constante e com eventual intermitência do meio de comunicação. Não toleram atrasos na propagação das atualizações.

- **Replicação assíncrona:** nesse tipo de replicação, somente a réplica alterada é tratada em tempo real; a cópia dos dados fica fora de sincronia entre os bancos de dados. Se uma base de dados é modificada, a alteração será propagada e aplicada para outra base de dados num segundo passo dentro de uma transação separada, sendo que essa poderá ocorrer segundos, minutos, horas ou até dias depois. A cópia poderá ficar temporariamente fora de sincronia, provocada pela interrupção de comunicação entre as réplicas, mas quando a sincronização ocorrer, os dados convergirão para todos os locais especificados.

2.7.3 Métodos de Replicação

Oliveira (2006), explica que em sistemas de bancos de dados, toda replicação requer um método para realizar as alterações feitas em dados replicados. Os dois métodos usados ou com maior frequência são:

- **Arquivo de registro:** cada transação (insert, update e delete) feita no banco de dados cria uma entrada na tabela de log. Em tempo determinado, essa

entrada é processada e a transação é propagada para todas as cópias do dado alterado; e

- Gatilhos transacionais: permitem que os usuários do banco de dados criem suas próprias regras para o tratamento de dados replicados. Esses gatilhos são disparados de acordo com evento determinado.

Arquivos de log e gatilhos podem ser usados individual ou conjuntamente em um SGBD, dependendo das características de cada sistema e de suas aplicações. Um ponto favorável ao uso de gatilhos para a realização de replicação de dados é que esse esquema demanda menos tempo e consome menos recursos do que arquivos de log.

A replicação de dados baseada em log implicaria que, em intervalos de tempo definidos, um arquivo fosse gravado com todas as alterações realizadas e, posteriormente, fosse processado em cada um dos sites envolvidos na replicação. Através de gatilhos e procedimentos, o mesmo processo pode ser realizado de forma mais rápida e consumindo menos recursos – uma alteração em uma réplica implica na execução de um procedimento responsável por enviar aos demais sites às modificações realizadas naquela época, e esperar por uma resposta afirmativa ou negativa de cada um dos sites em relação à efetivação da alteração em suas réplicas locais.

2.7.4 Conflitos de Replicação

Conforme Coutinho (2002) deve-se sempre criar um ambiente de replicação que evite a possibilidade de conflitos. E utilizando algumas técnicas isso é possível, não em sua totalidade, mas sim para que se permita um ambiente acessível. Alguns tipos de conflitos podem ocorrer de acordo com a replicação

utilizada, estes conflitos são:

- **Update** – ocorre quando se utiliza replicação ativa, pois somente nela se consegue fazer alterações. Ele pode acontecer quando duas transações, originadas de bancos diferentes, atualiza a mesma linha no mesmo instante. Este conflito poder ser evitado. Quanto menor for o tempo de atualização da réplica, ou seja, menor intervalo de atualização, menor será a possibilidade de ocorrer conflitos de atualização.
- **Delete** – este conflito é muito parecido com o conflito anterior. Imagine as duas bases A e B deletando a mesma linha no mesmo momento. Quando houver a sincronização das bases o conflito surgirá, pois haverá uma tentativa de apagar uma linha que já não mais existe.
- **Chaves únicas e seqüenciais** – ocorre quando a replicação de uma linha viola a integridade de uma tabela Primary Key ou Unique Key, quando duas transações originadas de lugares diferentes tentam inserir um registro em uma mesma tabela (réplica) utilizando o mesmo valor para primary key.

2.7.5 Replicação utilizada no Oracle e Postgresql

2.7.5.1 Oracle

De acordo com a documentação oficial da Oracle, a replicação funciona através de arquivo de registro da seguinte forma: um processo de captura armazena eventos do banco de dados, tais como alterações feitas em tabelas, esquemas, ou um banco de dados inteiro. As alterações registradas no arquivo de redo log de cada banco são

armazenadas em um registro chamado registro de mudança lógica (LCR). As regras usadas por este processo de captura determinam as alterações que ele capta, essas mudanças capturadas são chamadas de mensagens capturadas. O banco de dados onde as mudanças são geradas no redo log é chamado de banco de dados fonte. Um processo de captura pode capturar mudanças localmente no banco de origem, ou pode capturar alterações remotamente em um banco de dados downstream (figura 5).

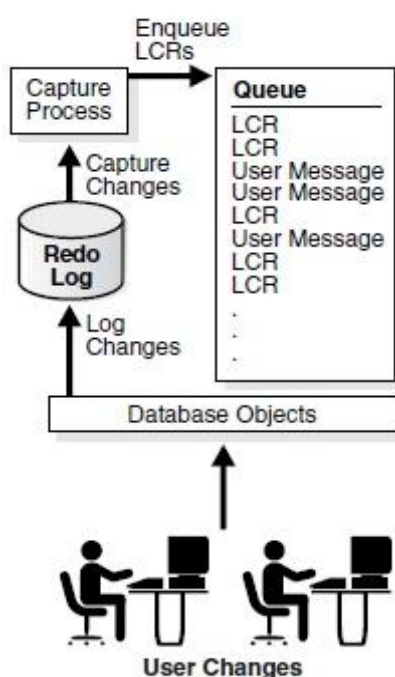


Figura 5: Processo de captura.

2.7.5.2 Postgresql

Segundo a documentação oficial do Postgresql versão 8.4.3.2 não existe uma solução nativa de replicação pelo fato deste banco ser de código aberto e facilmente estendido, uma série de empresas criaram várias soluções de replicação e recursos de balanceamento de carga. Portanto utilizaremos uma solução de replicação chamada de Slony-I.

Slony-I é um replicador do tipo Mestre-Escravo, funcionando da

seguinte forma:

1. Um servidor Slony-I age como mestre e um ou mais servidores Slony-I agem como escravos, replicando os dados de forma assíncrona.
2. Estes dados são capturados através de triggers e armazenados em tabelas de log.
3. Os logs armazenados são enviados aos servidores escravos.

2.8 Software de replicação

2.8.1 Oracle GoldenGate

Software de replicação adquirido pela Oracle em Julho de 2009 da GoldenGate, este software é capaz de permitir integração em tempo real de dados, garantindo contínua disponibilidade através da captura e entrega de informações baseadas em log de acordo como as mudanças que vão ocorrendo, proporcionando sincronização dos dados em ambientes heterogêneos como DB2, MySQL, SQL Server e Sybase.

2.8.2 Oracle Streams

Software de replicação da Oracle que possivelmente não será mais utilizado pela Oracle nas próximas versões de bancos de dados após a aquisição do Oracle GoldenGate. Replicação baseada em log podendo ser configurado para utilização de ambientes heterogêneos suportando somente os bancos de dados Sybase, Informix, SQL Server e DB2.

2.8.3 Postgres-r

É uma extensão de replicação do banco de dados Postgresql, provendo

eficiente, rápido e consistente replicação entre os bancos de dados, foi projetado para ser o mais transparente possível para o usuário.

O postgres-r foi desenvolvido especificamente para balanceamento de carga e alta disponibilidade, utiliza o modelo de replicação síncrona, suporta apenas bancos de dados Postgresql.

2.8.4 DBMoto

Software de replicação da Hit Software, realiza a replicação de dados através de log, possui um conjunto de ferramentas de integração de dados. Suporte a bancos de dados heterogêneos tais como: DB2, Oracle, Postgresql, SQL Server, Ingress, Firebird, MySQL, Gupta e Solid.

2.8.5 ObjectMMRS

Software de replicação da Object Sistemas utiliza modelos de replicação assíncrona e síncrona, entrega e captura de dados através de log ou gatilhos suporte a bancos de dados heterogêneos tais como: DB2, Oracle, Postgresql, SQL Server, Firebird, MySQL, Apache Derby, HSQLDB e SQLite.

3 METODOLOGIA

3.1 Delineamento da Pesquisa

O tipo de pesquisa utilizado neste trabalho pode ser classificado da seguinte forma:

É uma pesquisa de natureza aplicada, pois o investigador é movido pela necessidade de contribuir para fins práticos mais ou menos imediatos, buscando soluções para problemas concretos de acordo com Cervo e Bervian (2005).

É também uma pesquisa bibliográfica quanto aos procedimentos, pois foram utilizados livros e artigos como referência para elaboração deste trabalho, de acordo com Gil (2010) a pesquisa bibliográfica é elaborada com base em material já publicado. Tradicionalmente, esta modalidade de pesquisa inclui material impresso, como livros, revistas, jornais, teses, dissertações e anais de eventos científicos.

Quanto aos objetivos, segundo Andrade (2006) é exploratória, pois o primeiro passo de todo trabalho científico. São finalidades de uma pesquisa exploratória, sobretudo quando bibliográfica, proporcionar maiores informações sobre determinado assunto; facilitar a delimitação de um tema de trabalho; definir os objetivos ou formular as hipóteses de uma pesquisa ou descobrir novo tipo de enfoque para o trabalho que se tem em mente.

É um estudo de caso, pois foram levantadas informações sobre a empresa, bem como, os processos internos determinando os pontos de falhas para que estes processos fossem melhorados utilizando replicação de banco de dados. Silva e Siveira (2008) explicam que um estudo de caso, trata-se de um objeto bem restringido individual sobre o qual se levanta o maior número de informações possíveis. Assim uma cidade, um fenômeno ou mesmo um evento podem ser

objetos de estudo de caso.

3.2 Contextualização do Estudo de Caso

Trata-se de uma rede hospitalar denominada, Hospital Geral do Brasil, onde foram encontrados pela Diretoria de Tecnologia da Informação – DTI/HGB, algumas necessidades como implantação de um sistema único hospitalar abrangendo todas as unidades, e que os dados trafegados neste sistema, fossem replicados entre todas as unidades. A sede das unidades hospitalares utiliza o SGBD Oracle 10g, e as filiais Postgresql 8.4.2.3. A replicação será assíncrona abrangendo as informações de prontuário dos pacientes.

3.3 Procedimentos de Pesquisa

Os procedimentos executados foram realizados com referência na revisão de literatura e execução de análises para preparação de ambiente, execução de testes e auditoria.

4 ESTUDO DE CASO

O objetivo deste estudo de caso é apresentar a implantação de um software de replicação de banco de dados entre o SGBD Oracle e Postgresql na rede hospitalar, Hospital Geral do Brasil.

4.1 SGBD Utilizado

O SGBD utilizado na matriz é o Oracle versão 10g Enterprise Edition Release 2, e nas filiais Postgresql 8.4.2.3.

4.2 Infra-estrutura

Os servidores que compõem o parque tecnológico das unidades hospitalares, onde estarão às bases de dados são definidas da seguinte forma:

Matriz

Servidor de Banco de dados

- SGBD: Oracle 10g
- Processador Intel Core 2 duo 2.33 Ghz
- Memória: 32 GB
- HD: 500GB
- Sistema Operacional: Red Hat Enterprise Linux 5

Filial

Servidor de Banco de dados

- SGBD: PostgreSQL 8.4.2.3
- Processador Intel Core 2 duo 2.33 Ghz

- Memória: 32 GB
- HD: 500GB
- Sistema Operacional: GNU/Linux Debian 5

4.3 Descrição da Solução

A solução encontrada para aperfeiçoar o atendimento oferecido pelas unidades da rede hospitalar é a criação de um mecanismo de replicação de banco de dados, pois, a partir do momento de sua criação o prontuário será replicado para todas as unidades hospitalares.

De acordo com o apresentado neste estudo de caso atualmente existem vários softwares de replicação de dados que poderiam ser implantados nas unidades hospitalares. Entretanto, para o completo atendimento das necessidades de cada unidade foram seguidos alguns procedimentos de avaliação, conforme descrito abaixo:

1. Verificação e análise das ferramentas de replicação nativa dos bancos de dados.
2. Verificação e análise de alguns softwares de replicação existentes no mercado.

Após análise dos replicadores nativos foi observado que não atenderiam por falta de interoperabilidade entre os bancos de dados utilizados nas unidades hospitalares.

Após a verificação das funcionalidades complementares de alguns softwares de replicação, foram escolhidas duas ferramentas que condizem com as necessidades do hospital. Segue a matriz de comparação que levou a escolha da ferramenta.

RECURSO	OBJECTMMRS	DBMoto
Compatível com o Oracle	SIM	SIM
Compatível com o MSSQLServer	SIM	SIM
Compatível com o Ingres	NÃO	SIM
Compatível com o Sybase	SIM	SIM
Compatível com o DB2	SIM	SIM
Compatível com o PostgreSQL	SIM	SIM
Compatível com o MySQL	SIM	SIM
Compatível com o Apache Derby	SIM	NÃO
Compatível com o HSQLDB	SIM	NÃO
Compatível com o SQLite	SIM	NÃO
Compatível com o Gupta	NÃO	SIM
Compatível com o Solid	NÃO	SIM
Replicação de banco de dados Multi-Master	SIM	SIM
Replicação de banco de dados Assíncrono	SIM	SIM
Deteção e controle de conflitos de update	SIM	??
Configuração de link redundante entre internet e intranet	SIM	NÃO
Instruções reduzidas de versão de mecanismos de replicação para dispositivos móveis	SIM	NÃO
Uso de criptografia SSL	SIM	SIM
Captura de alteração dos dados baseados em triggers padrão do banco de dados	SIM	NÃO
Compressão dos dados	SIM	??
Ferramenta de sincronização de tabelas	SIM	SIM
Replicação de DDL	SIM	NÃO
Ferramenta para execução de comandos DDL em todos os servidores	SIM	NÃO
Proteção de usuários e senhas de banco de dados, não expondo os arquivos de configuração ou setup de tabelas	SIM	SIM
Necessita de um ou mais log's de tabelas para cada tabela de replicação de banco de dados	NÃO	SIM
Ferramenta de monitoramento que pode enviar emails e sms alertando os mecanismos de replicação de banco de dados indisponíveis	SIM	??
Arquitetura Multi-thread	SIM	??
Arquitetura multi-processo para atingir o máximo de performance na replicação de banco de dados entre milhares de servidores	SIM	??
compatível com o Linux	SIM	??
compatível com o Windows	SIM	SIM
Ferramenta de monitoramento enterprise com interface web habilitada para controle de vários projetos diferentes de replicação	SIM	SIM
Customização fácil de classe java permitindo controle completo, por exemplo é permitido replicação de dados para destinatários web-services	SIM	??
Característica de replicação que permite decisões de replicação baseada em comparação de dados entre bancos de dados	SIM	NÃO

de origem e destino		
Modo de funcionamento externo incorporado que permite uma aplicação java iniciar o mecanismo de replicação de controle, parar, ciclo de replicação, etc	SIM	??
Capacidade de customização de conversão de tipos de dados	SIM	??

Tabela 1 – comparação entre Objectmmrs e Dbmoto.

Após análise dos replicadores disponíveis de acordo com a matriz de comparação, o software mais adequado é o Objectmmrs da Object Sistemas. Este software será responsável em replicar toda e qualquer informação manipulada nos bancos de dados Oracle e Postgresql.

Para enviar informações entre os bancos de dados, o software de replicação utiliza tabelas de dicionário de dados do replicador, criadas no banco de dados. Estas informações são armazenadas temporariamente nestas tabelas e são descartadas logo após a efetivação da transferência destes registros.

A criação de replicação de banco de dados será realizada segundo o ambiente tecnológico disponibilizado pelo Hospital.

O modelo de dados das unidades hospitalares é composto de 200 tabelas participantes da replicação de dados, segue abaixo a criação do mecanismo de replicação nas quatro tabelas principais (figuras 6 e 7).

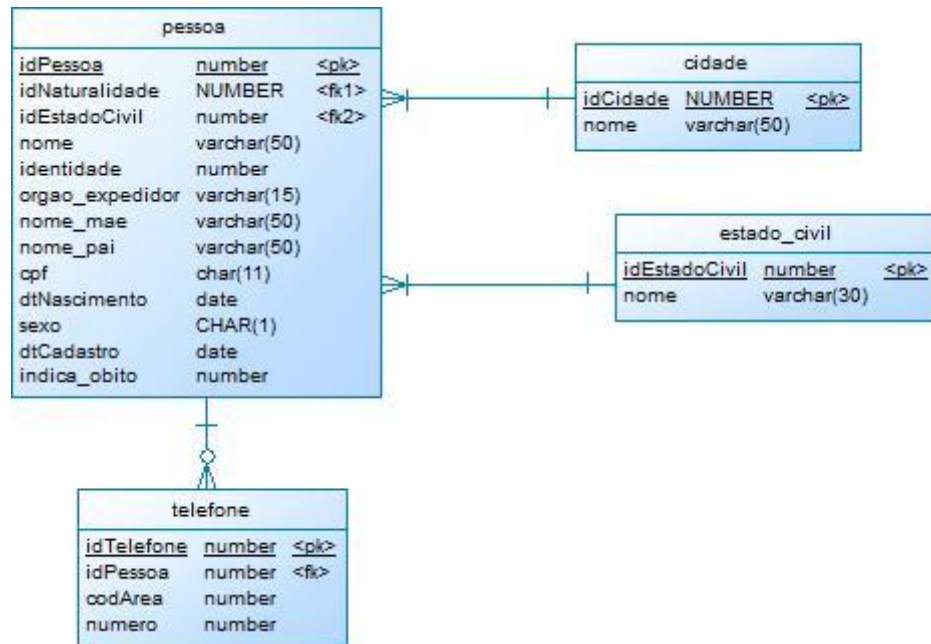


Figura 6: Diagrama Físico do Oracle

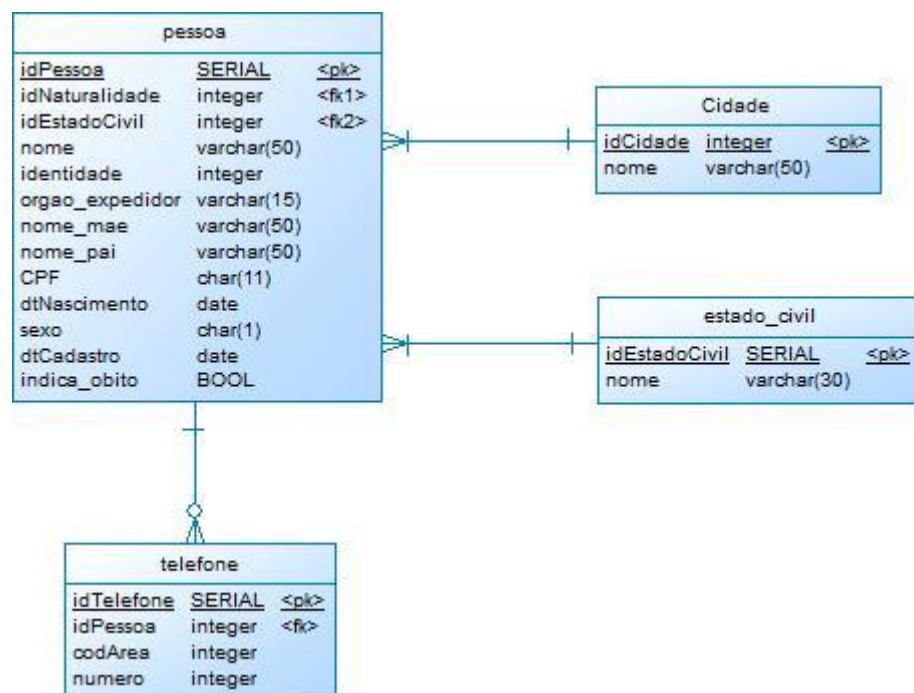


Figura 7: Diagrama Físico do Postgresql

4.4 Criação da Replicação

Para instalação do replicador Objectmmrs foram descompactados os arquivos de instalação em um diretório denominado: “/objectmmrs”, no servidor da matriz. Logo após, foram criados três usuários no SGBD Oracle: um servirá como dono das

tabelas de dicionário de dados do replicador e os outros dois foram utilizados pelo usuário do replicador para enviar os dados para os servidores das filiais. No SGBD das filiais foram criados dois usuários: um servirá como dono das tabelas de dicionário de dados do replicador e o outro para enviar os dados para o servidor da matriz. Após a criação dos usuários serão concedidos privilégios para sua utilização.

Criação do usuário dono do dicionário de dados do replicador, usuário Objectmmrs (figura 8).

```
CREATE USER objectmmrs IDENTIFIED BY object123;  
GRANT CONNECT, CREATE ANY TRIGGER TO objectmmrs;  
GRANT SELECT, INSERT, UPDATE, DELETE ON cidade TO objectmmrs;  
GRANT SELECT, INSERT, UPDATE, DELETE ON estado_civil TO objectmmrs;  
GRANT SELECT, INSERT, UPDATE, DELETE ON pessoa TO objectmmrs;  
GRANT SELECT, INSERT, UPDATE, DELETE ON telefone TO objectmmrs;
```

Figura 8: Criação do Usuário replicador no Oracle

Após a criação do usuário dono do dicionário de dados foi realizada a conexão no banco de dados com este usuário e executado o script “or_master_schema.sql” localizado no diretório: “/objectmmrs/sql/oracle”.

É importante ser realizada a criação do dicionário de dados, pois nestas tabelas são armazenadas todas as informações referentes à replicação como tabelas cadastradas, log de eventos realizados, servidores replicados dentre outras.

Após a criação do dicionário de dados foi seguida a configuração do replicador com a criação dos usuários repdf e repsp utilizados pelo replicador (figura 9).

```

CREATE USER repdf IDENTIFIED BY rep 123;
CREATE USER repsp IDENTIFIED BY rep 123;

GRANT CONNECT, CREATE ANY TRIGGER TO repdf;
GRANT CONNECT, CREATE ANY TRIGGER TO repsp;

GRANT SELECT, INSERT, UPDATE, DELETE ON cidade TO repdf;
GRANT SELECT, INSERT, UPDATE, DELETE ON estado_civil TO repdf;
GRANT SELECT, INSERT, UPDATE, DELETE ON pessoa TO repdf;
GRANT SELECT, INSERT, UPDATE, DELETE ON telefone TO repdf;

GRANT SELECT, INSERT, UPDATE, DELETE ON cidade TO repsp;
GRANT SELECT, INSERT, UPDATE, DELETE ON estado_civil TO repsp;
GRANT SELECT, INSERT, UPDATE, DELETE ON pessoa TO repsp;
GRANT SELECT, INSERT, UPDATE, DELETE ON telefone TO repsp;

GRANT SELECT ON objectmmrs.or_tb_subscription TO repsp;
GRANT SELECT ON objectmmrs.or_tb_log_output_t TO repsp;
GRANT SELECT ON objectmmrs.or_tb_outputtemp TO repsp;

GRANT SELECT ON objectmmrs.or_tb_subscription TO repsp;
GRANT SELECT ON objectmmrs.or_tb_log_output_t TO repsp;
GRANT SELECT ON objectmmrs.or_tb_outputtemp TO repsp;

```

Figura 9: Criação de Usuário no Postgresql Filial SP

Após a criação do dicionário de dados e dos usuários de replicação, foi configurado o arquivo de conexão com as bases de dados, localizado no diretório “/objectmmrs/objectdb.properties”, para a configuração deste arquivo é necessário adicionar ao arquivo o usuário master, os usuários escravos e o endereço de cada servidor de banco de dados. O software de replicação utiliza este arquivo para realizar a conexão com as bases de dados e executar a replicação (figura 10).

```
# Databases configuration parameters of OBJECT Multi-Master Replication Suite
#
#

databases=master
#DBCP database pool properties
master.dtbType=Oracle
master.dtbDriver=oracle.jdbc.driver.OracleDriver
master.dtbUrl=jdbc:oracle:thin:@mmrs1:1521:orcl
master.dtbDAC=XI59UfgZyf+cjC7rf3pn3XilhyHX9gOK83PcabdbMPU=
master.afterLogin=
master.pool=DBCP
master.DBCP.maxActive=5
master.DBCP.maxIdle=3
master.DBCP.minIdle=1
master.DBCP.maxWait=3000
master.DBCP.connectionTestStatement=SELECT 1 FROM DUAL
master.DBCP.testOnBorrow=false
master.DBCP.stmtPoolSize=0
master.DBCP.accessToUnderlyingConnectionAllowed=false
master.DBCP.loginTimeout=0

databases=slave1
#DBCP database pool properties
slave1.dtbType=PostgreSQL
slave1.dtbDriver=org.postgresql.Driver
slave1.dtbUrl=jdbc:postgresql://mmrs2:5432/filialdf?charset=utf8
slave1.dtbDAC=pAzgDLuPvDeWqKDOGEXWpPNz3Gm3WzD1
slave1.afterLogin=
slave1.pool=DBCP
slave1.DBCP.maxActive=5
slave1.DBCP.maxIdle=3
slave1.DBCP.minIdle=1
slave1.DBCP.maxWait=3000
#slave1.DBCP.connectionTestStatement=SELECT 1
slave1.DBCP.testOnBorrow=false
slave1.DBCP.stmtPoolSize=0
slave1.DBCP.accessToUnderlyingConnectionAllowed=false
slave1.DBCP.loginTimeout=0
```

Figura 10: Configuração do arquivo de conexão com o Banco de dados.

Após a configuração do arquivo de conexão foi editado o arquivo “/objectmmrs/publicar.sh”, neste arquivo foram cadastradas as tabelas que serão replicadas. O replicador utiliza trigger para capturar os dados, armazená-los em tabelas temporárias e replicá-los para os servidores cadastrados, após a edição do arquivo, este foi executado gerando o arquivo publicar.sql no diretório raiz “/objectmmrs” (figura 11).


```
#!/bin/sh
echo
echo "Trigger Generator Utility - Publish"
echo "Copyright 2002,2009 OBJECT Sistemas"
echo "=====
echo
java br.com.object.replication.util.TriggerGenerator
DTB "objectmmrs.properties" "templates/trigger/oracle.vm" OBJECTMMRS ORACLE "CIDADE,ESTADO_CIVIL,PESSOA,TELEFONE" publicar.sql UPDCOL K
```

Figura 11: Criação de trigger de replicação do servidor Oracle.

O próximo passo foi executar o arquivo publicar.sql com o script de criação das triggers no servidor Oracle utilizando o usuário Objectmmrs (Anexo A).

Após, foram realizados dois insert's na tabela or_tb_client para cadastro dos servidores que atuam como servidores escravos (figura 12).

```
INSERT INTO or_tb_client
(CLI_ID, CLI_NAME, CLI_ORDER, CLI_DTBALIAS, CLI_SYNCTYPE,
CLI_SERVERTYPE, CLI_COMMTYPE, CLI_ENGINE, CLI_STATUS, CLI_LASTCYCLE,
CLI_LASTCYCLEBTS, CLI_LOGALIAS, CLI_LOG_TABLE, CLI_FILETRANSTRIGGER,
CLI_WSHOST, CLI_WSPORT
)
VALUES
(2, 'MMRS2', 1, 'SLAVE1', 'A', 'N', 'J', 1, 'A', NULL, NULL, NULL, 'REPDP', NULL,
'OR_TB_LOG_FILIAL_DF', 0, NULL, 0);

INSERT INTO or_tb_client
(CLI_ID, CLI_NAME, CLI_ORDER, CLI_DTBALIAS, CLI_SYNCTYPE,
CLI_SERVERTYPE, CLI_COMMTYPE, CLI_ENGINE, CLI_STATUS, CLI_LASTCYCLE,
CLI_LASTCYCLEBTS, CLI_LOGALIAS, CLI_LOG_TABLE, CLI_FILETRANSTRIGGER,
CLI_WSHOST, CLI_WSPORT
)
VALUES
(3, 'MMRS3', 2, 'SLAVE2', 'A', 'N', 'J', 1, 'A', NULL, NULL, NULL, 'REPS', NULL,
'OR_TB_LOG_FILIAL_SP', 0, NULL, 0);

COMMIT;
```

Figura 12: Cadastro dos servidores escravos de replicação.

Após o cadastro dos servidores escravos foram cadastradas as tabelas que estes servidores receberão os dados e quais operações DML serão realizadas (figura 13).

```
INSERT INTO or_tb_subscription  
(pub_id, cli_id, sub_doinserts, sub_doupdates, sub_dodeletes, sub_stype)  
SELECT pub_id, 2, 'Y', 'Y', 'Y', 'D' FROM or_tb_publish  
  
INSERT INTO or_tb_subscription  
(pub_id, cli_id, sub_doinserts, sub_doupdates, sub_dodeletes, sub_stype)  
SELECT pub_id, 3, 'Y', 'Y', 'Y', 'D' FROM or_tb_publish  
  
COMMIT;
```

Figura 13: Cadastro das tabelas e operações DML realizadas.

Após estas configurações de replicação o serviço do replicador foi iniciado com a execução do comando no Linux, “./run.sh” (figura 14), no diretório padrão de instalação do replicador. Estas configurações realizadas para criação de replicação na matriz também foram executadas nos servidores das filiais.

A estrutura do banco de dados é idêntica nos três hospitais. Foram necessários alguns ajustes para impedir erros de duplicação de chaves primarias, devido os identificadores das tabelas serem controlados por números seqüenciais, exceto a tabela cidade que segue a identificação padrão adotada pelo IBGE. Para solução deste problema foram criados identificadores seqüenciais incrementados pelo total de banco de dados que farão parte da replicação, segue abaixo a criação dos identificadores seqüenciais no SGBD Oracle e Postgresql (figuras 15, 16 e 17).

```

-rw-r--r-- 1 root root 302 Mai 2 2011 3rd_licenses.txt
-rw-r--r-- 1 root root 64 Mai 2 2011 commons-logging.properties
-rw-r--r-- 1 root root 103 Mai 2 2011 CONTEST.BAT
-rw-r--r-- 1 root root 107 Mai 2 2011 contest.sh
-rw-r--r-- 1 root root 89 Mai 2 2011 DACGEN.BAT
-rw-r--r-- 1 root root 94 Mai 2 2011 dacgen.sh
-rw-r--r-- 1 root root 32 Mai 2 2011 DERBY.BAT
-rw-r--r-- 1 root root 128 Mai 2 2011 derby.properties
-rw-r--r-- 1 root root 31 Mai 2 2011 derby.sh
drwxr-xr-x 2 root root 4096 Jun 22 14:52 lib
-rw-r--r-- 1 root root 658 Out 24 17:11 license.dat
-rw-r--r-- 1 root root 443 Out 24 17:11 license.key1
-rw-r--r-- 1 root root 46 Out 24 17:11 license.key2
-rw-r--r-- 1 root root 2194 Jun 22 15:07 log4j.properties
drwxr-xr-x 2 root root 4096 Out 25 23:59 logs
-rw-r--r-- 1 root root 324953 Mai 2 2011 MMRSServer.jar
-rw-r--r-- 1 root root 2313 Jun 22 16:01 objectdb.properties
-rw-r--r-- 1 root root 3708 Mai 2 2011 objectdb_template.properties
-rw-r--r-- 1 root root 2340 Jun 15 17:22 objectmmrs.properties
-rw-r--r-- 1 root root 2326 Mai 2 2011 objectmmrs_template.properties
drwxr-xr-x 2 root root 4096 Jun 8 09:21 output
-rw-r--r-- 1 root root 295 Mai 2 2011 PUBLICAR.BAT
-rw-r--r-- 1 root root 342 Jun 15 10:08 publicar.sh
-rw-r--r-- 1 root root 36985 Jun 15 11:35 publicar.sql
-rw-r--r-- 1 root root 289 Mai 2 2011 REFRESH.BAT
-rw-r--r-- 1 root root 317 Mai 2 2011 refresh.sh
-rw-r--r-- 1 root root 37522 Mai 2 2011 release_notes_ptBR.txt
-rw-r--r-- 1 root root 341 Mai 2 2011 RUN.BAT
-rwxr-xr-x 1 root root 396 Mai 2 2011 run.sh
-rw-r--r-- 1 root root 217 Mai 2 2011 SERVERINFO.BAT
-rw-r--r-- 1 root root 226 Mai 2 2011 serverinfo.sh
-rw-r--r-- 1 root root 440 Mai 2 2011 SETCLASSPATH.BAT
-rw-r--r-- 1 root root 489 Jun 8 11:52 setclasspath.sh
-rw-r--r-- 1 root root 440 Mai 2 2011 SETCLASSPATH_TEMPLATE.BAT
-rw-r--r-- 1 root root 440 Mai 2 2011 setclasspath_template.sh
drwxr-xr-x 11 root root 4096 Jun 8 09:21 sql
drwxr-xr-x 5 root root 4096 Jun 8 09:21 templates
drwxr-xr-x 2 root root 4096 Jun 8 09:21 tmp
-rw-r--r-- 1 root root 9383 Mai 2 2011 upgrade_notes_ptBR.txt
-rw-r--r-- 1 root root 5719 Jun 15 11:35 velocity.log
drwxr-xr-x 3 root root 4096 Jun 8 09:21 windows
root@debian:/objectmmrs# ./run.sh

```

Figura 14: Execução do serviço do replicador.

```
CREATE SEQUENCE seq_pessoa
START WITH 1
INCREMENT BY 3;

CREATE SEQUENCE seq_telefone
START WITH 1
INCREMENT BY 3;

CREATE SEQUENCE seq_estado_civil
START WITH 1
INCREMENT BY 3;

GRANT SELECT, ALTER ON seq_pessoa TO objectmmrs;
GRANT SELECT, ALTER ON seq_telefone TO objectmmrs;
GRANT SELECT, ALTER ON seq_estado_civil TO objectmmrs;

GRANT SELECT, ALTER ON seq_pessoa TO repdf;
GRANT SELECT, ALTER ON seq_telefone TO repdf;
GRANT SELECT, ALTER ON seq_estado_civil TO repdf;

GRANT SELECT, ALTER ON seq_pessoa TO repsp;
GRANT SELECT, ALTER ON seq_telefone TO repsp;
GRANT SELECT, ALTER ON seq_estado_civil TO repsp;
```

Figura 15: Criação do Identificador seqüencial no SGBD Oracle da Matriz.


```
CREATE SEQUENCE seq_pessoa
START WITH 2
INCREMENT BY 3;

CREATE SEQUENCE seq_telefone
START WITH 2
INCREMENT BY 3;

CREATE SEQUENCE seq_estado_civil
START WITH 2
INCREMENT BY 3;

GRANT SELECT, UPDATE ON seq_pessoa TO objectmmrs;
GRANT SELECT, UPDATE ON seq_telefone TO objectmmrs;
GRANT SELECT, UPDATE ON seq_estado_civil TO objectmmrs;

GRANT SELECT, UPDATE ON seq_pessoa TO repmatriz;
GRANT SELECT, UPDATE ON seq_telefone TO repmatriz;
GRANT SELECT, UPDATE ON seq_estado_civil TO repmatriz;
```

Figura 16: Criação do Identificador sequencial no SGBD Postgresql na Filial Brasília.

```
CREATE SEQUENCE seq_pessoa
START WITH 3
INCREMENT BY 3;

CREATE SEQUENCE seq_telefone
START WITH 3
INCREMENT BY 3;

CREATE SEQUENCE seq_estado_civil
START WITH 3
INCREMENT BY 3;

GRANT SELECT, UPDATE ON seq_pessoa TO objectmmrs;
GRANT SELECT, UPDATE ON seq_telefone TO objectmmrs;
GRANT SELECT, UPDATE ON seq_estado_civil TO objectmmrs;

GRANT SELECT, UPDATE ON seq_pessoa TO repmatriz;
GRANT SELECT, UPDATE ON seq_telefone TO repmatriz;
GRANT SELECT, UPDATE ON seq_estado_civil TO repmatriz;
```

Figura 17: Criação do Identificador sequencial no SGBD Postgresql na Filial São Paulo.

Após a criação dos identificadores seqüenciais foram realizadas operações de insert, update e delete e verificado a integridade dos dados após a transferência dos dados entre os servidores.

As figuras 18, 19 e 20, apresentam as operações de insert realizadas nos servidores da matriz, filial Brasília e filial São Paulo respectivamente.

Enter SQL Statement:

```

1 INSERT INTO pessoa (IDPESSOA, IDNATURALIDADE, IDESTADOCIVIL, NOME, IDENTIDADE, ORGAO_EXPEDIDOR, NOME_MAE, NOME_PAI, CPF, DTNASCIMENTO, SEXO)
2 VALUES (SEQ_PESSOA.NEXTVAL, 1, 1, 'Andre', 22222, 'SSP/DF', 'Sônia', 'Andre', '00000000000', '19/04/1985', 'M', sysdate, 0);
3
4 COMMIT;
5
6 SELECT * FROM pessoa

```

Results: Script Output Explain Autotrace DBMS Output OWA Output

IDPESSOA	IDNATURALIDADE	IDESTADOCIVIL	NOME	IDENTIDADE	ORGAO_EXPEDIDOR	NOME_MAE	NOME_PAI	CPF	DTNASCIMENTO
1	1	1	Andre	22222 SSP/DF	Sônia	Andre	00000000000	19/04/85	

Figura 18: Criação de Paciente no SGBD Oracle.

```

INSERT INTO rh.pessoa (IDNATURALIDADE, IDESTADOCIVIL, NOME, IDENTIDADE, ORGAO_EXPEDIDOR, NOME_MAE, NOME_PAI, CPF, DTNASCIMENTO, SEXO, DTCADASTRO, INDICA_OBITO)
VALUES (1, 1, 'Patricia', 22222, 'SSP/DF', 'Inerilda', 'Isaias', '00000000000', '12/12/1987', 'F', now(), 'F');

select * from rh.pessoa

```

Output pane

Output Explain Messages History

	idpessoa bigint	idnaturalidade integer	idestadocivil integer	nome character varying(50)	identidade integer	orgao_expedidor character varying(15)	nome_mae character varying(50)	nome_pai character varying(50)	cpf character(11)	dtnascimento date	sexo character(1)
1	1	1	1	Andre	22222	SSP/DF	Sônia	Andre	00000000000	1985-04-19	M
2	2	1	1	Patricia	22222	SSP/DF	Inerilda	Isaias	00000000000	1987-12-12	F

Figura 19: Criação de Paciente no SGBD Postgresql na Filial Brasília.

```

INSERT INTO rh.pessoa (IDNATURALIDADE, IDESTADOCIVIL, NOME, IDENTIDADE, ORGAO_EXPEDIDOR, NOME_MAE, NOME_PAI, CPF, DTNASCIMENTO, SEXO, DTCADASTRO, INDICA_OBITO)
VALUES (1,1,'Anna Paula',22222,'SSP/DF','Sonia','Andre','00000000000','23/10/1982','F', now(), 'F');

select * from rh.pessoa

```

	idpessoa bigint	idnaturalidade integer	idestadocivil integer	nome character varying(50)	identidade integer	orgao_expedidor character varying(15)	nome_mae character varying(50)	nome_pai character varying(50)	cpf character(11)	dtnascimento date	sexo char
1	1	1	1	Andre	22222	SSP/DF	Sônia	Andre	00000000000	1985-04-19	M
2	2	1	1	Patricia	22222	SSP/DF	Inerilda	Isaias	00000000000	1987-12-12	F
3	3	1	1	Anna Paula	22222	SSP/DF	Sonia	Andre	00000000000	1982-10-23	F

Figura 20: Criação de Paciente no SGBD Postgresql na Filial São Paulo.

As figuras 21, 22 e 23, apresentam as operações de update realizadas nos servidores da matriz, filial Brasília e filial São Paulo respectivamente.

```

Enter SQL Statement:
1 SELECT * FROM pessoa
2
3 update pessoa set cpf='98755445321'
4 where idpessoa=3

```

	IDPESSOA	IDNATURALIDADE	IDESTADOCIVIL	NOME	IDENTIDADE	ORGAO_EXPEDIDOR	NOME_MAE	NOME_PAI	CPF	DTNASCIMENTO
1	2	1	1	Patricia	22222	SSP/DF	Inerilda	Isaias	00000000000	12/12/87
2	1	1	1	Andre	22222	SSP/DF	Sônia	Andre	00000000000	19/04/85
3	3	1	1	Anna Paula	22222	SSP/DF	Sonia	Andre	98755445321	23/10/82

Figura 21: Atualização de Paciente no SGBD Oracle na Matriz.

```

update rh.pessoa set cpf='55245643445'
where idpessoa=2

select * from rh.pessoa

```

	idpessoa bigint	idnaturalidade integer	idestadocivil integer	nome character varying(50)	identidade integer	orgao_expedidor character varying(15)	nome_mae character varying(50)	nome_pai character varying(50)	cpf character(11)	dtnascimento date	sexo character(1)	dtcadastro date	indica_obito boolean
1	3	1	1	Anna Paula	22222	SSP/DF	Sonia	Andre	98755445321	1982-10-23	F	2011-06-23	f
2	2	1	1	Patricia	22222	SSP/DF	Inerilda	Isaias	55245643445	1987-12-12	F	2011-06-23	f
3	1	1	1	Andre	22222	SSP/DF	Sônia	Andre	00000000000	1985-04-19	M	2011-06-23	f

Figura 22: Atualização de Paciente no SGBD Postgresql na Filial Brasília.

<pre>update rh.pessoa set cpf='03256732378' where idpessoa=1 select * from rh.pessoa</pre>													
<div>Output pane</div> <div> <div>Output</div> <div>Explain</div> <div>Messages</div> <div>History</div> </div>													
	idpessoa	idnaturalidade	idestadocivil	nome	identidade	orgao_expedidor	nome_mae	nome_pai	cpf	dtascimento	sexo	dtcadastro	indica_obito
	bigint	integer	integer	character varying(50)	integer	character varying(15)	character varying(50)	character varying(50)	character(11)	date	character(1)	date	boolean
1	3	1	1	Anna Paula	22222	SSP/DF	Sonia	Andre	98755445321	1982-10-23	F	2011-06-23	f
2	1	1	1	Andre	22222	SSP/DF	Sônia	Andre	03256732378	1985-04-19	M	2011-06-23	f
3	2	1	1	Patricia	22222	SSP/DF	Inerilda	Isaias	55245643445	1987-12-12	F	2011-06-23	f

Figura 23: Atualização de Paciente no SGBD Postgresql na Filial São Paulo.

As figuras 24, 25 e 26, apresentam as operações de delete realizadas nos servidores da matriz, filial Brasília e filial São Paulo respectivamente.

Enter SQL Statement:

1 SELECT * FROM pessoa

2

3 delete from pessoa

4 where idpessoa=1;

5

6 commit;

Results

Script Output

Explain

Autotrace

DBMS Output

OWA Output

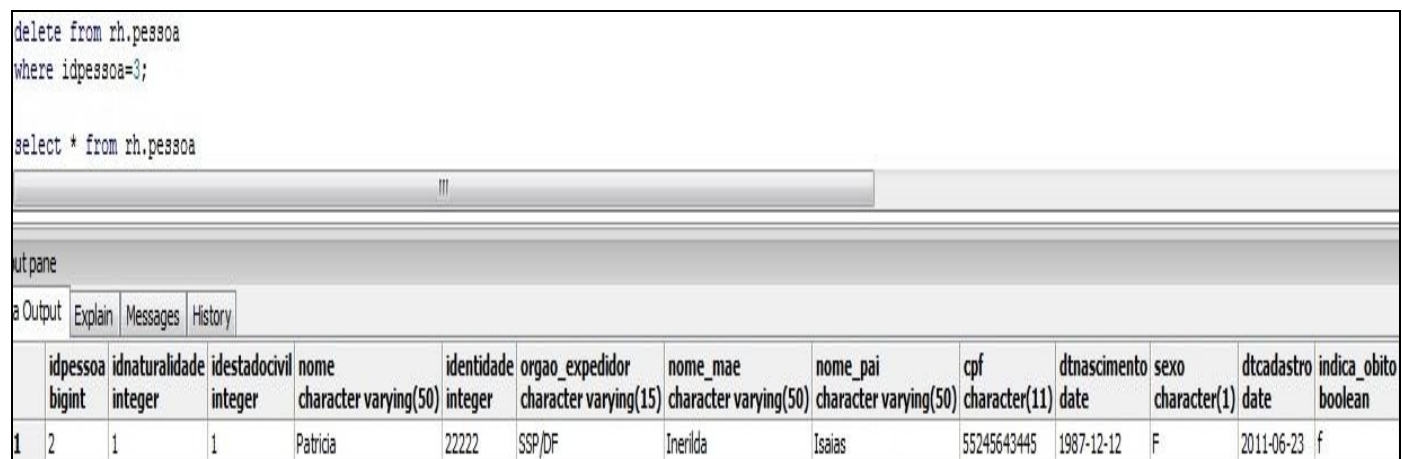
Results:

	IDPESSOA	IDNATURALIDADE	IDESTADOCIVIL	NOME	IDENTIDADE	ORGAO_EXPEDIDOR	NOME_MAE	NOME_PAI	CPF	DTNASCIMENTO
1	2	1	1	Patricia	22222 SSP/DF	Inerilda	Isaias	55245643445	12/12/87	
2	3	1	1	Anna Paula	22222 SSP/DF	Sonia	Andre	98755445321	23/10/82	

Figura 24: Exclusão de Paciente no SGBD Oracle na Matriz.

```
delete from rh.pessoa
where idpessoa=3;

select * from rh.pessoa
```

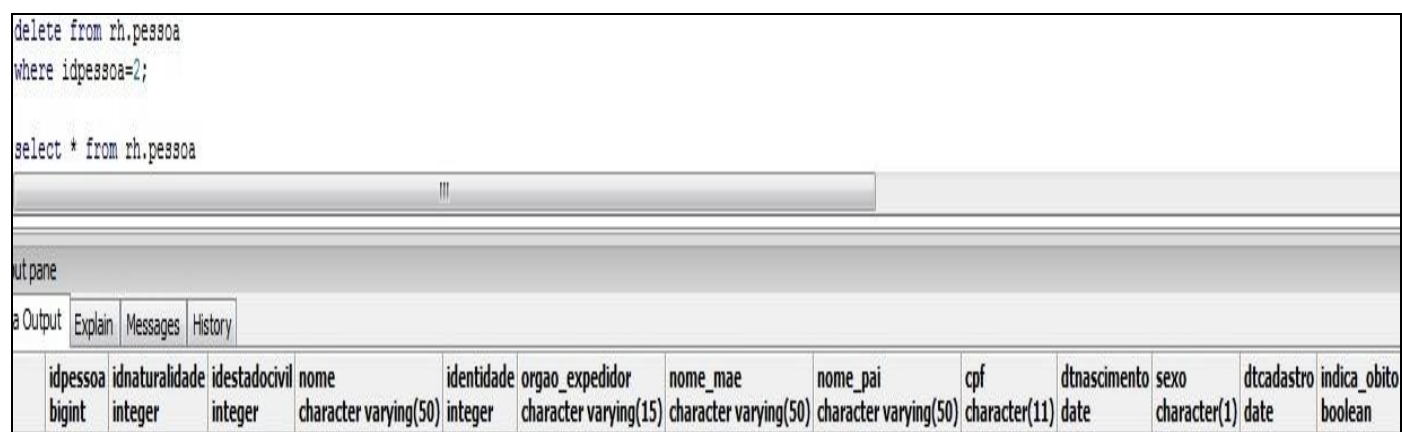


	idpessoa	idnaturalidade	idestadocivil	nome	identidade	orgao_expedidor	nome_mae	nome_pai	cpf	dtnascimento	sexo	dtcadastro	indica_obito
	bigint	integer	integer	character varying(50)	integer	character varying(15)	character varying(50)	character varying(50)	character(11)	date	character(1)	date	boolean
1	2	1	1	Patricia	22222	SSP/DF	Inerilda	Isaias	55245643445	1987-12-12	F	2011-06-23	f

Figura 25: Exclusão de Paciente no SGBD Postgresql na Filial Brasília.

```
delete from rh.pessoa
where idpessoa=2;

select * from rh.pessoa
```



	idpessoa	idnaturalidade	idestadocivil	nome	identidade	orgao_expedidor	nome_mae	nome_pai	cpf	dtnascimento	sexo	dtcadastro	indica_obito
	bigint	integer	integer	character varying(50)	integer	character varying(15)	character varying(50)	character varying(50)	character(11)	date	character(1)	date	boolean
1	2	1	1	Patricia	22222	SSP/DF	Inerilda	Isaias	55245643445	1987-12-12	F	2011-06-23	f

Figura 26: Exclusão de Paciente no SGBD Postgresql na Filial São Paulo.

4.5 Topologia dos Ativos

A figura 27 apresenta a topologia dos ativos das unidades de saúde após a instalação e configuração do replicador.

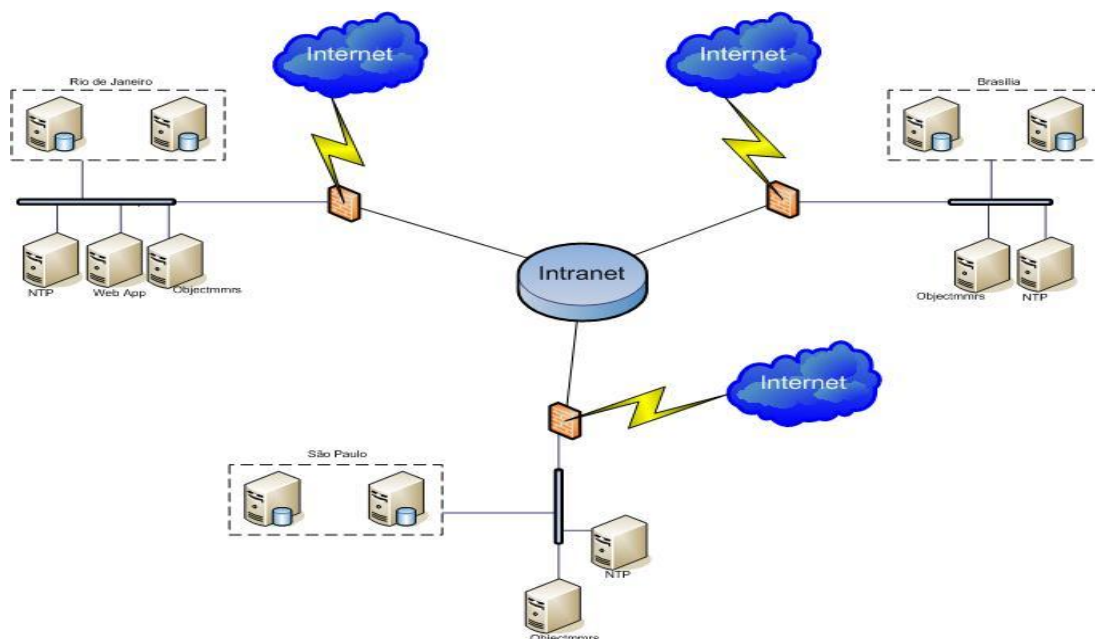


Figura 27: Topologia dos ativos de rede

4.6 Desempenho do Replicador

Para verificação do comportamento do replicador foram simulados os seguintes testes:

1. Simulação de 100 usuários efetuando 30 operações de insert, update e delete em cada servidor.
2. Verificação da velocidade do tráfego de dados na rede.

Após a realização dos testes foram alcançados os seguintes resultados:

1. Simulação com os 100 usuários foi obtida uma movimentação de 9000 registros nos três servidores com duração de 12 minutos de replicação.
2. A velocidade de transmissão de dados ficou entre 129 e 931 kbits/segundo.

4.7 Funcionamento e Manutenção do Replicador

Após instalação e configuração do replicador o serviço permanece ligado constantemente e só em caso de manutenção no serviço do replicador este será desligado. Enquanto estiver sendo realizada a manutenção no replicador os

dados manipulados no banco de dados são armazenados em tabelas temporárias para que não haja perda de dados e logo após a finalização da manutenção o serviço é restabelecido e os dados destas tabelas temporárias são replicados para os bancos de dados cadastrados. Entretanto, caso ocorra perda dos dados no momento de manutenção da replicação, pode ser utilizado aplicativos internos do replicador para execução de sincronização de tabelas para que os dados permaneçam idênticos.

O serviço do replicador pode ser inicializado junto com os serviços do sistema operacional ou através de comando de execução do arquivo “run.sh” no diretório objectmmrs.

O servidor de replicação é passível de algumas falhas que possam ser causados por eventos internos (falhas de hardware) ou externos (falta de energia, enchente dentre outros), neste caso não haverá impacto na replicação, pois os dados são armazenados em tabelas temporárias e após o restabelecimento do serviço a replicação é realizada normalmente.

A replicação ao ser iniciada não realiza nenhuma verificação nas tabelas para saber se estão nulas ou se existe diferença de quantidade de registros de uma tabela para outra, caso ocorra alguma operação de delete ou update em registros que só existe em determinada tabela o replicador registrará o erro no arquivo de log e continuará realizando a replicação até que algum administrador intervenha e solucione o problema.

O replicador antes de enviar algum registro para os bancos de dados verifica se está operacional, caso não esteja, os registros que serão enviados para este banco será armazenado em tabelas temporárias até que volte a ser operacional, após o restabelecimento do serviço o replicador envia os dados que ficaram na

fila.

Segundo a documentação do replicador Objectmmrs não existe rotina de manutenção a ser seguida, apenas é recomendado que seja verificado nos arquivos de log's eventuais erros na replicação entre os servidores.

5 ANÁLISES DO ESTUDO DE CASO

Após a criação e configuração do mecanismo de replicação de banco de dados foram identificadas as tabelas principais que teriam os dados replicados entre as unidades hospitalares.

O primeiro procedimento foi realizar um planejamento de forma a atender as necessidades dos hospitais e o cenário tecnológico disponibilizado pela DTI/HGB. Deste modo foram definidas quais medidas seriam adotadas e implantadas.

A primeira medida adotada foi criação de redundância de link's para replicação via intranet ou internet, por padrão a replicação seria via intranet, caso perdesse a comunicação por este modo, automaticamente o replicador altera o modo de replicação para internet e após o restabelecimento do serviço via intranet, voltaria para o modo de replicação padrão.

A segunda medida adotada foi a criação de identificadores seqüenciais para cada tabela envolvida na replicação com incremento inicial 3, evitando erro de duplicação de chave primária, comumente ocorrida em replicadores multi-master.

A terceira medida adotada foi à criação de servidores NTP (Network Time Protocol), para garantir que à hora dos servidores de replicação estejam sincronizados de acordo com a hora atual do observatório nacional, com esta sincronização pode ser evitado conflitos de update, insert e delete, comumente

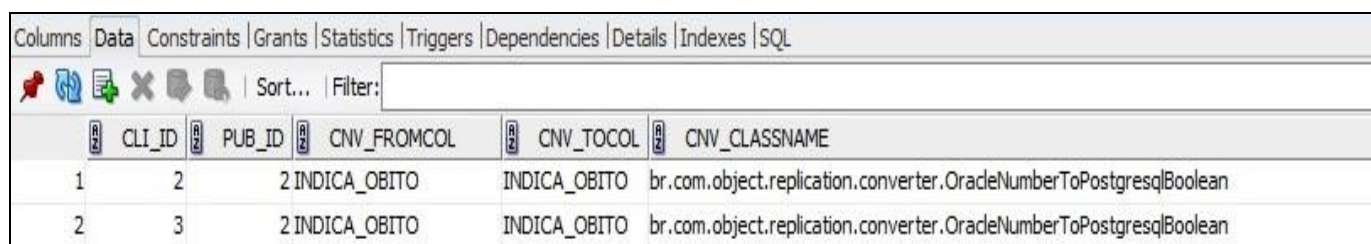
ocorrido em replicadores multi-master.

A quarta medida adotada foi elaborar uma matriz de comparação com alguns tipos de dados utilizados no SGBD's Oracle e PostgreSQL (tabela 2).

ORACLE	POSTGRESL
DATE	TIMESTAMP
LONG	TEXT
CLOB	TEXT
NCLOB	TEXT
BLOB	BYTEA
BFILE	TEXT
BINARY_FLOAT	DOUBLE PRECISION
BINARY_FLOAT	DOUBLE PRECISION
NÃO EQUIVALENTE	SERIAL
NÃO EQUIVALENTE	BOOLEAN

Tabela 2 – comparação de tipos de dados.

Após a verificação dos tipos de dados utilizados nos dois bancos foi percebido que a tabela pessoa possui uma coluna indica_obito no SGBD Postgresql definida como Boolean e no Oracle definida como number, portanto, para que estas diferenças não causassem impacto na replicação, foi configurado no replicador utilizando a tabela de dicionário de dados or_tb_converter as tabelas e colunas com estas diferenças, a partir daí uma classe interna do software de replicação fará a conversão dos tipos de dados (figura 28). Ao final da replicação os dados foram replicados sem erros permanecendo íntegros e confiáveis.



CLI_ID	PUB_ID	CNV_FROMCOL	CNV_TOCOL	CNV_CLASSNAME
1	2	2 INDICA_OBITO	INDICA_OBITO	br.com.object.replication.converter.OradeNumberToPostgresqlBoolean
2	3	2 INDICA_OBITO	INDICA_OBITO	br.com.object.replication.converter.OradeNumberToPostgresqlBoolean

Figura 28: Tabela de conversão de tipo de dados

6 CONCLUSÃO

Conforme este estudo sobre implantação de replicação entre bases de dados heterogêneas nas unidades hospitalares do Hospital Geral do Brasil foi identificado à necessidade desta organização de implantar replicação de banco de dados.

Nas unidades hospitalares, o SGBD Oracle é utilizado na matriz e o SGBD Postgresql nas duas filiais, portanto, para que fosse implantada a replicação, foi verificado que a replicação nativa destes SGBDS não atenderia a necessidade do hospital através da utilização da matriz de comparação com os dois SGBDS, pois, além do fato destes não haver interoperabilidade entre si, não atenderia de forma completa, sendo assim, a replicação foi implantada utilizando um replicador proprietário da Object Sistemas, denominado de Objectmmrs.

A replicação utilizando este software atendeu as necessidades da organização sendo possível melhorar o fluxo de atendimento, garantindo alta disponibilidade em termos de tolerância a falhas de rede, pois, além de poder utilizar todos os recursos do software de replicação, de forma simples e prática foi solicitado ao desenvolvedor deste, que adicionasse link redundante de conexão, fazendo com que o hospital aumentasse o tempo de disponibilidade do sistema utilizado naquele hospital.

7 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Após a implantação e execução da replicação de banco de dados permanecem algumas recomendações a serem executadas futuramente como descritas abaixo:

1. Aquisição do software de replicação com interface web para instalação e monitoramento dos dados replicados.
2. Implantação de ferramentas de monitoramento dos SGBDS Oracle e Postgresql.

8 REFERÊNCIAS

ANDRADE, Maria Margarida de. **Introdução à metodologia do trabalho científico**. 7. ed. São Paulo: Editora Atlas S. A, 2006.

CERVO, Amado Luiz; BERVIAN, Pedro Alcino. **Metodologia Científica**. 5. ed. São Paulo: Pearson Prentice Hall, 2002.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Sistemas Distribuídos: Conceitos e Projetos**. 4. ed. Porto Alegre: Brookman, 2007.

COUTINHO, Willian Pires. **Dados Distribuídos com replicação em Oracle**. 2002. Disponível em: < <https://www.computacao.unitri.edu.br> > Acesso em: 19 março 2011.

DATE, C. J. **Introdução a Sistemas de Banco de Dados**. 8. ed. Rio de Janeiro: Elsevier, 2004.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Bancos de Dados**. 4. ed. São Paulo: Person Addison Wesley, 2005.

GARCIA-MOLINA, Hector; ULLMAN, Jeffrey D.; WIDOM, Jennifer. **Implementação de Sistemas de Bancos de Dados** Rio de Janeiro: Campus, 2001.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**. 5. ed. São Paulo: Editora Atlas S. A, 2010.

OLIVEIRA, Alexandre Pereira de. **Modelo de Replicação de Dados entre SGBD Heterogêneos**. Monografia (Graduação em Ciências da Computação). 2006. Gravataí. Universidade Luterana do Brasil.

ORACLE. **Streams Concepts and Administration 10g Release 2**. Disponível em: < <http://www.oracle.com> > Acesso em: 30 de agosto de 2011.

POSTGRESQL. **PostgreSQL 8.4.5 Documentation**. Disponível em: < <http://www.postgresql.org/docs/manuals/>> Acesso em: 19 de março de 2011.

RAMALHO, José Antonio. **Oracle 10g**. São Paulo: Pioneira Thompson Learning, 2005.

SILBERSCHARTZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 3. ed. São Paulo: Pearson Makron Books, 2005.

SILVA, José Maria da; SILVEIRA, Emerson Sena da. **Apresentação de Trabalhos Acadêmicos: Normas e Técnicas**. 3. ed. Rio de Janeiro: Editora Vozes, 2008.

SILVA, Robson Soares. **Oracle Database Express Edition 10g: Guia de Instalação, Configuração e Administração**. 1. ed. São Paulo: Érica, 2007.

SLONY-I. **Slony-I 2.1.0 Documentation** . Disponível em: <<http://slony.info/documentation/2.1/requirements.html>> Acesso em: 30 de agosto de 2011.

OZSU, M. Tamer; VALDURIEZ, Patrick. **Princípios de Sistemas de Bancos de Dados Distribuídos**. 2. ed. Rio de Janeiro: Campus, 2001.

WATSON, John. **OCA Oracle Database 11g Administração I: Guia do Exame 1Z0-052**. Porto Alegre: Brookman, 2010.

ANEXO A – TRIGGER CRIADA PELO SOFTWARE OBJECTMMRS

Exemplo de Trigger criada pelo software objectmmrs para ser executada no Oracle:

```
--
-- OBJECT Multi-Master Replication Server v_6_2
-- Publishing script for ORACLE.TELEFONE
--
INSERT INTO OBJECTMMRS.or_tb_publish
(pub_schema, pub_tablename, pub_order, pub_loglevel, pub_keyname, pub_keydatatype, pub_enabled, pub_urnolog, pub_
VALUES
('ORACLE', 'TELEFONE', 0, '0',
 'IDTELEFONE', 'N', 'Y', '', '0');

commit;

CREATE OR REPLACE TRIGGER OBJECTMMRS.or_trg_TELEFONE AFTER INSERT OR UPDATE OR DELETE
ON ORACLE.TELEFONE FOR EACH ROW
DECLARE
campos VARCHAR2(4000);
buf VARCHAR2(4000);
colupd SMALLINT;
todolog NUMBER(5);
seq NUMBER(12);
isenabled CHAR(1);
nologuser VARCHAR2(32);
updcmode CHAR(1);
updccol VARCHAR2(32);
updlog SMALLINT;
serverid SMALLINT;
```

Exemplo de Trigger criada pelo software objectmmrs para ser executada no PostgreSQL:

```
--
-- OBJECT Multi-Master Replication Server v_6_2
-- Publishing script for rh.cidade
--
INSERT INTO public.or_tb_publish
    (pub_schema, pub_tablename, pub_order, pub_loglevel, pub_keyname, pub_keydatatype, pub_enabled, pub_urnolog, pub_
VALUES
    ('rh', 'cidade', 0, '0',
    'idcidade', 'N', 'Y', '', '0');

CREATE FUNCTION public.or_f_cidade() RETURNS TRIGGER AS
$BODY$
    DECLARE
        campos TEXT;
        buf TEXT;
        colupd SMALLINT;
        todolog SMALLINT;
        isenabled CHAR(1);
        nologuser VARCHAR(32);
        updcmode CHAR(1);
        updccol VARCHAR(32);
        updlog SMALLINT;
```

GLOSSÁRIO

BACKUP: é a cópia dos dados de um dispositivo de armazenamento a outra para que possa ser recuperado em caso de perda dos dados originais.

BINARY FLOAT: tipo de dado numérico utilizado exclusivamente para ponto flutuante pode armazenar até 4 Bytes.

BFILE: tipo de dado utilizado para armazenar dados binários não estruturados em arquivos do sistema fora do banco de dados.

BLOB: Binary Large Object, um tipo de dado que armazena até 4 GB de fotos, vídeo, áudio, gráficos, mapas e etc.

BYTEA: um tipo de dado binário com tamanho máximo de armazenamento de até 1 GB.

BOOLEAN: tipo de dado que armazena somente um dos dois estados: Verdadeiro ou Falso.

CHAVE PRIMÁRIA: restrição que exige que cada linha ou grupo de linhas seja identificado por um valor único.

CHAVE ÚNICA: restrição que exige que cada valor ou grupo de valores seja único em todos os registros da tabela.

CLOB: Character Large Object, armazena dados do tipo caractere de até 4 GB.

CPU: Central Processing Unit, ou do português, Unidade Central de Processamento, é um dispositivo multifuncional programável que aceita dados digitais como entrada. Processa de acordo com as instruções armazenadas em sua memória, e fornece resultados como saída.

CREATE: comando utilizado para criar objetos em bancos de dados como: tabelas, usuários e etc.

DATE: tipo de dado utilizado para armazenar data e hora, pode armazenar data ou hora ou ambos, para cada valor DATE a Oracle armazena as informações de século, ano, mês, data, hora, minuto e segundo.

DDL: Data Definition Language, do português, Linguagem de Definição de Dados, é utilizada para definir, alterar e remover objetos de banco de dados.

DELETE: comando SQL utilizado para deletar registros em tabelas de banco de dados.

DML: Data Manipulation Language, do português, Linguagem de Definição de Dados, é utilizada para inclusão, remoção e modificação de informações em bancos de dados.

DOUBLE PRECISION: um tipo de dado numérico de ponto flutuante de precisão de 15 dígitos decimais

GATILHOS(TRIGGERS): linguagem procedural executadas automaticamente quando ocorre algum evento vinculado a uma tabela ou visão.

GRANT: comando utilizado para autorizar algum usuário a executar operações em bancos de dados.

HARDWARE: Componentes físicos e eletrônicos do computador.

IDENTIFICADORES SEQUENCIAIS: é utilizado dentro do banco de dados para criar campos de autonumeração.

INSERT: comando SQL utilizado para inserir registros em tabelas de banco de dados.

LOG DE DADOS: arquivos de registros utilizados para armazenar eventos importantes em sistemas computacionais.

LONG: um tipo de dado que armazena dados de até 2 GB de tamanho, com restrição de que somente uma coluna por tabela pode ser definido com esse tipo.

NCLOB: um tipo de dado utilizado para armazenar dados do tipo Unicode até 4 GB.

NTP: Network Time Protocol, é um protocolo para sincronização dos relógios dos computadores.

RAM: Random Access Memory, do português, Memória de Acesso Aleatório, é uma forma de armazenamento de dados do computador.

SERIAL: um tipo de dado numérico com propriedade de auto incremento.

SISTEMA OPERACIONAL: é um conjunto de programas que inicializa o hardware do computador, fornece rotinas para controle de dispositivos, gerencia, escalona e mantém a integridade de sistema.

SSL: Secure Socket Layer é o protocolo usado para criar páginas seguras, encriptando toda a transmissão entre o cliente e o servidor.

SOFTWARE: é uma seqüência de instruções a serem seguidas ou executadas, na manipulação, redirecionamento ou modificação de um dado ou acontecimento.

TEXT: um tipo de dados de caracteres com tamanho máximo de armazenamento de até 1 GB.

TIMESTAMP: tipo de dado que armazena dados do tipo data e hora.

UPDATE: comando SQL utilizado para atualizar registros em tabelas de banco de dados.