Universidade de Évora

Mestrado em Engenharia Informática

Pacote de Sincronização de Réplicas de Bases de Dados

Aluno: Rui Filipe Frágua Caeiro

Orientador: Prof. Luís Arriaga Cunha

Évora, Dezembro de 2010

Universidade de Évora

Mestrado em Engenharia Informática

Pacote de Sincronização de Réplicas de Bases de Dados

Aluno: Rui Filipe Frágua Caeiro

Orientador: Prof. Luís Arriaga Cunha

Évora, Dezembro de 2010

Agradecimentos

Esta dissertação é o culminar de todo o meu percurso académico, onde sempre existiu bastante dedicação e esforço, que foi muitas vezes suportada com a ajuda de outras pessoas às quais pretendo deixar um agradecimento especial.

Agradeço profundamente ao meu orientador, Prof. Luís Arriaga Cunha, pelo seu acompanhamento na evolução deste trabalho e também pela sua disponibilidade sempre que era necessária a discussão sobre novos problemas que foram aparecendo ao longo do desenvolvimento desta dissertação.

Agradeço e muito aos meus país e irmão pelo apoio incondicional que me deram ao longo de todo o meu percurso académico.

Um especial agradecimento à Ana, que está sempre ao meu lado, pela sua paciência e encorajamento em todos os momentos.

Resumo

A replicação de base de dados tem como objectivo a cópia de dados entre bases de dados distribuídas numa rede de computadores. A replicação de dados é importante em várias situações, desde a realização de cópias de segurança da informação, ao balanceamento de carga, à distribuição da informação por vários locais, até à integração de sistemas heterogéneos. A replicação possibilita uma diminuição do tráfego de rede, pois os dados ficam disponíveis localmente possibilitando também o seu acesso no caso de indisponibilidade da rede.

Esta dissertação baseia-se na realização de um trabalho que consistiu no desenvolvimento de uma aplicação genérica para a replicação de bases de dados a disponibilizar como *open source software*. A aplicação desenvolvida possibilita a integração de dados entre vários sistemas, com foco na integração de dados heterogéneos, na fragmentação de dados e também na possibilidade de adaptação a várias situações.

Synchronization Package for Database Replication

Abstract

Data replication is a mechanism to synchronize and integrate data between distributed databases over a computer network. Data replication is an important tool in several situations, such as the creation of backup systems, load balancing between various nodes, distribution of information between various locations, integration of heterogeneous systems. Replication enables a reduction in network traffic, because data remains available locally even in the event of a temporary network failure.

This thesis is based on the work carried out to develop an application for database replication to be made accessible as open source software. The application that was built allows for data integration between various systems, with particular focus on, amongst others, the integration of heterogeneous data, the fragmentation of data, replication in cascade, data format changes between replicas, master/slave and multi master synchronization.

Índice

Lista de Figuras

Lista de Quadros

Glossário

2PC	-	Two-phase Commit
ACID	-	Atomicidade, Consistência, Isolamento e Durabilidade
AJAX	-	Asynchronous Javascript And XML
API	-	Application Programming Interface
BD	-	Base de Dados
BSD	-	Berkeley Software Distribution
EE	-	Enterprise Edition
EJB	-	Enterprise Java Bean
FK	-	Chave Estrangeira (Foreign Key)
GPL	-	General Public License
IDE	-	Integrated Development Environment
JAXB	-	Java Architecture for XML Binding
JAX-WS	-	Java API for XML Web Services
JDBC	-	Java Database Connectivity
JSF	-	JavaServer Faces
JSP	-	JavaServer Pages
JVM	-	Java Virtual Machine
ME	-	Mobile Edition
ODBC	-	Open Database Connectivity
PK	-	Chave Primária (Primary Key)
RAM	-	Random Acess Memory
SE	-	Standard Edition
SGBD	-	Sistema de Gestão de Bases de Dados
URI	-	Uniform Resource Identifier
WSDL	-	Web Services Description Language
XML	-	Extensible Markup Language

Capítulo 1

Introdução

1.1 - Motivação

Nos últimos tempos tem-se notado uma enorme evolução no mundo dos sistemas de informação, consequentemente, tem-se acumulado cada vez mais informação em bases de dados. Em muitas situações é necessário recorrer à replicação de dados, para que a informação pretendida esteja distribuída e disponível no local pretendido e à hora certa. Além do excesso de informação existente, em algumas situações os dados têm diferentes origens, aumentando a complexidade quando se pretende garantir a consistência da informação replicada entre diferentes locais.

A nível empresarial é cada vez mais importante o recurso ao armazenamento e análise de dados, sendo este um auxiliar de elevada importância para a tomada de decisões correctas e dentro do período pretendido. Nos dias de hoje são utilizadas algumas aplicações distribuídas tais como, comércio electrónico, controlo de produção e *Data Warehouses*, de modo a auxiliar o negócio de uma empresa. Várias empresas estão divididas por diferentes locais físicos, sendo necessária a troca de informação entre eles. A replicação de dados tem um papel fundamental para que as informações sejam trocadas entre os diferentes locais, mesmo internamente, entre diferentes secções, ou entre diferentes empresas poderá ser importante a troca de informações.

O acumular de muita informação de elevada importância, trouxe a necessidade da criação de cópias de segurança de modo a preservar a informação pretendida. Desta forma é possível recuperar os dados no caso de uma falha ou acidente num sistema de informação. O uso da replicação de dados é importante pois permite a criação de várias cópias de segurança.

1.2 - Objectivos

O trabalho elaborado no âmbito desta dissertação tem como objectivo principal o desenvolvimento de uma aplicação para a sincronização de réplicas entre bases de dados distribuídas, onde é possível integrar dados de diferentes plataformas e diferentes gestores de bases de dados.

Antes do desenvolvimento da aplicação propriamente dita, foram estudadas e analisadas outras aplicações com funcionalidades idênticas, de modo a procurar soluções que pudessem ser

aplicadas no trabalho desenvolvido e também oferecer alguma inovação relativamente ao que existe.

Um dos objectivos é a disponibilização de um sistema de histórico, para que a aplicação obtenha informação sobre as actualizações que ocorrem numa base de dados. Pretende-se também que este sistema de histórico esteja disponível para outras aplicações que não apenas a desenvolvida no âmbito desta dissertação.

A integração entre várias plataformas e gestores de bases de dados é um objectivo que se pretende alcançar, possibilitando a replicação de dados entre plataformas de características diferentes e também entre gestores de bases de dados diferentes.

A implementação de um mecanismo de publicação e subscrição de dados é outro objectivo proposto, proporcionando vários tipos de replicação de dados, ficando o utilizador da aplicação responsável por escolher a situação que lhe é mais conveniente.

Outro objectivo é a possibilidade de filtrar os dados de forma a adaptarem-se a várias situações, como por exemplo, obter apenas uma lista de clientes da localidade Évora ou obter apenas o nome de uma pessoa e o respectivo número de telefone, de uma tabela onde existam mais campos. Pretende-se desta forma melhorar a performance da aplicação e também evitar a replicação de dados desnecessários.

Por fim, pretende-se disponibilizar a aplicação desenvolvida, na comunidade *sourceforge* de modo a oferecer uma nova solução para replicação de dados e também para ajudar na detecção de erros ou falhas que possam ocorrer na sua execução.

Capítulo 2

Estado da Arte

2.1 - Conceitos Envolvidos

Para uma melhor percepção do trabalho desenvolvido no âmbito desta dissertação de mestrado, é importante fazer uma introdução aos conceitos envolvidos. A subsecção 2.1.1 faz uma breve descrição do que é a replicação de dados. Os conceitos sobre fragmentação de dados são descritos em 2.1.2. Os tipos de replicação são enumerados em 2.1.3, a forma como os dados são replicados é descrita em 2.1.4 e os tipos de dados que são actualizados são descritos em 2.1.5.

2.1.1 - Replicação - O que é?

A replicação de dados está associada a um sistema de base de dados distribuídas numa rede de computadores, onde a sua função é copiar dados entre os vários nós disponíveis na rede [1,2]. A replicação deve ser transparente para o utilizador, ou seja, o SGBD ou a aplicação desenvolvida deve ser o responsável por copiar os dados entre os vários sítios [2].

O aumento da disponibilidade dos dados é uma das principais características da replicação de dados, pois é possível aceder a cópias locais, evitando o acesso remoto aos dados, diminuindo assim o tráfego na rede e também a possibilidade da consulta dos dados caso a rede esteja indisponível [1,2]. A probabilidade de colisão de dados é menor, dado que é possível aceder aos mesmos dados em diferentes sítios, possibilitando um balanceamento de carga entre os vários nós [2].

O elevado volume de dados poderá ser um problema para a replicação de dados entre vários nós, além de existir maior probabilidade de redundância de dados, a cópia entre vários nós torna-se bastante lenta. [2]

Na replicação de dados é possível copiar dados entre bases de dados homogéneas e heterogéneas. Consideram-se base de dados homogéneas todas as que usam o mesmo SGBD, ou seja, vão existir vários nós que têm o mesmo SGBD como referência. Nas bases de dados heterogéneas são usados diferentes SGBDs quando se copiam dados entre os nós, sendo necessário um mecanismo de tradução. [2]

2.1.2 - Fragmentação

A fragmentação dos dados consiste na divisão de uma tabela em fragmentos mais pequenos, permitindo reduzir os efeitos negativos da replicação, pois existem menos dados a serem manipulados e também o espaço necessário é menor. Desta forma a fragmentação possibilita mais disponibilidade dos dados, melhor performance e mais confiança nos dados que são replicados. [2]

Nas subsecções 2.1.2.1 e 2.1.2.2 são descritos os dois tipos de fragmentação, horizontal e vertical respectivamente.

2.1.2.1 - Fragmentação Horizontal

A fragmentação horizontal divide os dados de uma tabela através das suas linhas, ou seja, é obtido um subconjunto de tuplos de uma tabela. [2]

ID	Nome	Morada	Localidade	Telefone
1	Rui	Rua X	Évora	123456789
2	João	Rua A	Lisboa	987654321
3	Ana	Rua Q	Lisboa	321456987
4	António	Rua V	Lisboa	789654321
5	Manuel	Rua T	Faro	456789231

Figura 2.1: Exemplo de Fragmentação Horizontal.

Na Figura 2.1 pode-se verificar um exemplo de fragmentação horizontal, onde os dados são divididos pela coluna "Localidade" com o valor "Lisboa", ou seja, todos os elementos que são de Lisboa fazem parte deste subconjunto.

2.1.2.2 - Fragmentação Vertical

Na fragmentação vertical o objectivo é dividir uma tabela em subconjuntos de menor dimensão através da divisão das colunas de uma determinada tabela. [2]

ID	Nome	Morada	Localidade	Telefone
1	Rui	Rua X	Évora	123456789
2	João	Rua A	Lisboa	987654321
3	Ana	Rua Q	Lisboa	321456987
4	António	Rua V	Lisboa	789654321
5	Manuel	Rua T	Faro	456789231

Figura 2.2: Exemplo de Fragmentação Vertical.

Como exemplo de replicação vertical pode-se verificar na Figura 2.2, que os dados a serem divididos estão divididos por três colunas ("ID","Nome", "Morada"). Desta forma existe uma melhor performance no processamento dos dados pretendidos. [2]

2.1.3 - Tipos de Replicação

Existem várias formas para a replicação de dados, em 2.1.2.1 é descrita a replicação síncrona e em 2.1.2.2 é descrita a replicação assíncrona. Nas subsecções 2.1.2.3, 2.1.2.4 e 2.1.2.5 são enunciados a replicação unidireccional, replicação bidireccional e replicação total, parcial e incremental, que são subtipos da replicação síncrona e assíncrona.

2.1.2.1 - Replicação Síncrona

Numa replicação síncrona uma transacção local torna-se também em uma transacção distribuída, isto é, quando existe uma actualização local esta é logo enviada para vários destinos, tudo dentro da mesma transacção. A transacção só termina quando os destinatários responderem à origem que enviou os dados. [3]

As transacções numa replicação síncrona são caracterizadas por serem ACID (Atomicidade, Consistência, Isolamento, Durabilidade), garantido segurança nas transacções efectuadas [2,4]. A atomicidade garante que uma transacção seja processada por completo, reconhecendo uma falha quando a transacção não é inteiramente completa [2]. A consistência é uma propriedade que garante a integridade referencial numa transacção [2]. O isolamento garante que não ocorrem alterações enquanto uma transacção não tiver terminado, garantindo a confiança dos dados [2]. A durabilidade permite recuperar os dados de uma transacção em caso de falha, pois os dados são armazenados em tabelas auxiliares [2].

Para garantir que as transacções sejam ACID existe um protocolo denominado *Two-Phase Commit* (2PC), que tem a função de efectuar uma transacção nos vários destinos pretendidos. Durante a transacção um sistema pode informar que está tudo correcto ou que existiu uma falha. Caso não exista alguma falha, a transacção é bem sucedida, no caso de existir pelo menos uma falha a transacção não é efectuada. [2,4]

Este tipo de replicação poderá causar problemas de performance num sistema, pois é necessário que todos os nós sejam actualizados para a cópia dos dados estar disponível para posteriores acessos. [4]

2.1.2.2 - Replicação Assíncrona

A replicação assíncrona é caracterizada pela cópia dos dados entre vários nós após o registo ser dado como concluído na base de dados fonte. Ao contrário da replicação síncrona, neste tipo

de replicação existe tolerância a falhas, podendo a fonte enviar os dados para um destino apenas quando ele estiver disponível. [3]

Através da replicação assíncrona é possível ter uma melhor performance em relação à replicação síncrona, pois a replicação só é efectuada após a transacção estar concluída localmente. No entanto, podem existir algumas inconsistências, como o atraso no envio de dados e também a maior probabilidade para a colisão dos mesmos dados. [4]

Existem três tipos de replicação assíncrona: logo que possível, a pedido e em momentos regulares. O tipo que mais se aproxima de uma replicação síncrona é o logo que possível, pois os dados são replicados assim que a transacção na fonte termine ou quando o sistema o permitir. Na replicação a pedido, os dados apenas são replicados quando existe um pedido dos mesmos. Em momentos regulares é possível definir um momento mais propício para se efectuar a cópia dos dados. [2]

2.1.2.3 - Replicação Unidireccional

A replicação unidireccional ou "*Master-Slave*" trata de enviar dados entre uma base de dados denominada "Mestre" e várias bases de dados denominadas "Escravas". A base de dados "Mestre" é responsável por receber todas as actualizações e pela propagação das mesmas actualizações pelas restantes bases de dados "Escravas". [2,4]

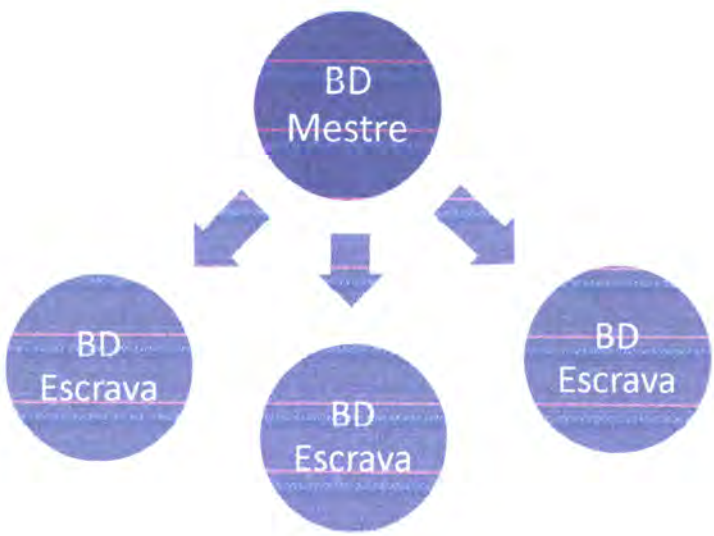

Figura 2.3: Replicação "*Master-Slave*".

Nesta situação, as bases de dados "Escravas" apenas recebem actualizações vindas de uma base de dados "Mestre", que é responsável pelo processamento de toda a informação. Este tipo de replicação pode ser aplicação nas replicações síncrona e assíncrona. [2]

2.1.2.4 - Replicação Bidireccional

A replicação bidireccional também conhecida por "*Multi-Master*" permite a troca de dados entre várias bases de dados "Mestre", ou seja, não existe apenas uma base de dados a receber actualizações. Ao contrário da replicação unidireccional, estamos perante um caso em que cada base de dados recebe actualizações, que são trocadas entre várias bases de dados. [2]

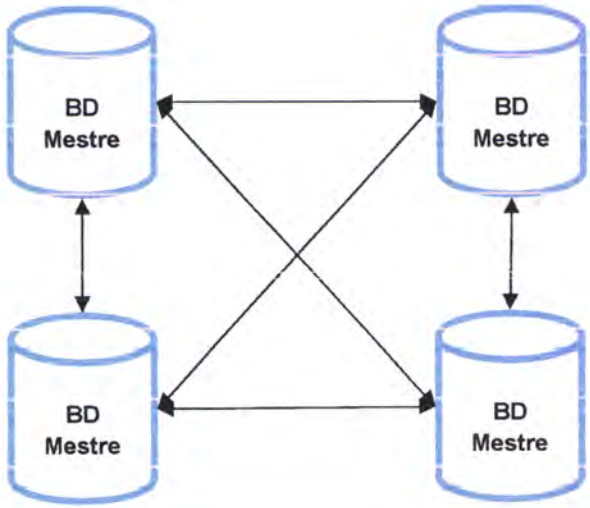

Figura 2.4: Replicação "*Multi-Master*".

Este tipo de replicação torna-se bastante complexo quando existem bastantes nós a replicar dados entre si, principalmente quando se está presente de replicação síncrona, onde é preciso existir alta disponibilidade dos dados. [2]

2.1.2.5 - Replicação Total, Parcial e Incremental

Além do que está descrito nos tópicos anteriores referentes aos tipos de replicação, existem ainda outros tipos que se podem conjugar com os anteriores.

A replicação total implica a cópia total dos dados, ou seja, uma fonte de dados vai transmitir a um destino todos os dados de uma tabela. No caso de existir um grande volume de dados será normal um largo período até a replicação dar-se como concluída. [3]

Na replicação parcial, os dados de uma tabela são divididos em subconjuntos de menor dimensão e depois enviados para o respectivo destino. Este tipo de replicação pode ser utilizado quando é necessário copiar dados desde a última actualização, por exemplo, um sistema esteve inacessível durante um determinado período, quando voltar a estar acessível poderá receber todos os dados desde a última actualização. [3]

A replicação incremental surge quando existem alterações numa base de dados e é necessário replicá-las logo que possível ou na própria transacção inicial. [3]

2.1.4 - Como é Feito?

Os dados ao serem replicados podem ser enviados pela fonte para vários destinos ou serem pedidos pelo destino à fonte. Em 2.1.3.1 é descrita a replicação por envio e em 2.1.3.2 a replicação por pedido. No tópico 2.1.3.3 são caracterizadas as tabelas de histórico, que são um bom auxiliar para o controlo de actualizações.

2.1.3.1 - Por Envio (*Push Replication*)

Os dados podem ser enviados para vários destinatários sem que estes se tenham de preocupar com a actualização dos dados na fonte. A fonte é responsável pelo controlo de actualizações, que podem ser propagadas após uma ordem, incrementalmente ou entre determinados períodos. A responsabilidade dos destinatários é de apenas receber os dados, estando sempre à espera de uma actualização vinda de uma fonte. [3]

2.1.3.2 - Por Pedido (*Pull Replication*)

Ao contrário de uma replicação por envio, existe também a hipótese por pedido, ou seja, o destino dos dados é responsável por fazer um pedido dos dados à fonte. A verificação de actualizações fica a cargo do destinatário, que deve fazer um pedido à fonte numa hora previamente determinada ou através de uma ordem humana. [3]

2.1.3.3 - Tabelas de Histórico

As tabelas de histórico têm uma função muito importante na sincronização de dados, pois é através delas que se obtêm todo o tipo de ocorrências. Por norma, as tabelas de histórico estão associadas a uma ou mais tabelas que recebem actualizações. [3]

Através da analise às tabelas de histórico é possível informar a fonte ou destino dos dados se existem actualizações e quais as actualizações que ocorreram. A fonte dos dados fica encarregue de guardar juntamente com a tabela principal as respectivas tabelas de histórico. Em 4.1 está presente um exemplo de um modelo de tabelas de histórico. [3]

2.1.5 - Dados Actualizados

Existem dois tipos de dados a serem replicados, as transacções que ocorrem numa determinada base de dados e as operações que podem ocorrer numa tabela. [3,4]

Sempre que ocorre uma ou mais actualizações numa base de dados, estas ficam associadas a uma ou mais transacções. Estas transacções podem ser logo replicadas para vários destinos, ou então serem guardadas localmente. [4]

Quando uma transacção termina ficam uma ou várias operações associadas a uma tabela, estas operações podem ser posteriormente replicadas para outros destinos. Ao replicar uma ou mais operações, deve-se considerar a tabela afectada, um registo ou até mesmo um atributo. Ou seja é possível copiar todas as operações que ocorreram numa tabela, um ou mais registos ou apenas um atributo de um determinado registo. [3]

2.2 - Trabalho Relacionado

Nesta subsecção são descritas algumas ferramentas para replicação de bases de dados, disponibilizadas por alguns dos SGBDs mais populares e também desenvolvidas por alguns projectos externos.

2.2.1 - Slony-I

O SGBD PostgreSQL disponibiliza a ferramenta Slony-I para a replicação de dados, esta é caracterizada por disponibilizar replicação unidireccional e assíncrona. É uma ferramenta que funciona à base de *triggers* e ainda possibilita a propagação de dados em cascata. [5]

O Slony-I não coloca restrições na plataforma usada, no entanto só replica dados homogéneos e também não garante consistência nas transacções entre os diversos nós. É uma ferramenta recomendada para casos em que é preciso fazer uma réplica total ou parcial, como é o caso de um *backup* de dados. [5]

2.2.2 - MySQL Replication

A ferramenta para replicação de dados pelo SGBD MySQL denomina-se por MySQL Replication, tal como o Slony-I apenas possibilita a replicação unidireccional e assíncrona. [6]

Nesta ferramenta está disponível um sistema de replicação a pedido, não sendo suportado a propagação de dados em cascata, ou seja, cada destino vai extrair os dados à fonte. Existe a possibilidade de transformação de tipos, no entanto, tal como no Slony-I só é possível replicar dados homogéneos. [6]

2.2.3 - Oracle Replication

A ferramenta disponibilizada pela Oracle apresenta-se muito completa, possibilitando replicação unidireccional e bidireccional e também replicação síncrona e assíncrona. Possibilita ainda a integração de dados com outros SGBDs e garante a consistência das transacções efectuadas. [7]

Não existem restrições de plataforma na ferramenta disponibilizada pela Oracle, existindo ainda a possibilidade de transformação de tipos. [7]

2.2.4 - SQL Server Replication

SQL Server Replication é a ferramenta para replicação de base de dados desenvolvida pela Microsoft. Bastante idêntica à ferramenta desenvolvida pela Oracle, onde existe replicação unidireccional e bidireccional, mas apenas é possível replicar dados assincronamente. [8]

Apesar de existir a possibilidade de replicar dados heterogéneos, apenas é possível usar esta ferramenta na plataforma Windows, podendo restringir o seu uso em algumas situações. [8]

2.2.5 - Daffodil Replicator

Daffodil Replicator é um projecto open source que está disponível na comunidade Source Forge. Esta ferramenta foi desenvolvida na linguagem Java e é baseada em drivers JDBC para a interacção com as bases de dados. [9]

Esta ferramenta possibilita a replicação bidireccional e replicação assíncrona, onde existe aproximação a uma replicação síncrona, ou seja, apesar de não existir replicação síncrona existe uma aproximação sendo possível copiar dados assim que possível. [9]

Este projecto possibilita ainda o seu uso em diferentes plataformas e a integração com diferentes SGBDs. [9]

2.2.6 - DB Replicator

O DB Replicator é outro projecto disponível na comunidade *sourceforge*, onde apresenta bastantes semelhanças com o projecto anteriormente descrito. Não apresenta restrições de plataforma, podendo ainda integrar diferentes SGBDs. [10]

Esta ferramenta possibilita replicação bidireccional e replicação síncrona e tem a particularidade de copiar esquemas de tabelas, caso não existam num respectivo destinatário. [10]

2.2.7 - Tungsten Replicator

O Tungsten Replicator é uma ferramenta *open source*, que permite a replicação de dados. Esta aplicação apenas permite replicação unidireccional e assíncrona e é compatível com os SGBDs Oracle e MySQL. [26]

Esta ferramenta é caracterizada pela sua disponibilidade, sendo tolerante a falhas que possam ocorrer e também pela verificação de consistência nas suas transacções. [26]

Esta aplicação abstém-se da utilização de *triggers* para o registo de histórico, utilizando *logfiles* disponibilizados directamente pelos próprios SGBDs. É ainda possível replicar dados para outros SGBDs além dos referidos como por exemplo o PostgreSQL. [26]

Capítulo 3

Ferramentas Usadas

Neste capítulo são descritas as ferramentas utilizadas para o desenvolvimento do pacote de sincronização de réplicas de bases de dados. A escolha das ferramentas usadas neste projecto foi uma fase importante, porque foi preciso fazer um estudo dos objectivos pretendidos e posteriormente procurar as ferramentas que correspondessem ao que era pretendido.

Os seguintes tópicos fazem uma breve descrição das ferramentas utilizadas ao longo do desenvolvimento deste projecto.

3.1 - MySQL

O MySQL começou a ser desenvolvido no inicio de 1994, por Michael Widenius e David Axmark e a primeira versão foi lançada em Janeiro de 1998. [6]

O MySQL é um SGBD bastante popular, que é destinado para sistemas com elevados níveis de produção. É um sistema multi-plataforma, ou seja, é suportado por vários sistemas operativos, tais como, Linux, MAC OS X, Windows, entre outros. Além disso, são disponibilizadas várias APIs, que facilitam a integração com outras linguagens de programação (C, Java, PHP, Ruby, etc). Possibilita ainda, a criação de *stored procedures* e *triggers*. Não necessita de muitos recursos de hardware e é facilmente manipulado. [6,11]

Além das características já referenciadas, outra das razões para o uso deste SGBD, deve-se ao facto de ser um software livre, com base na licença GNU GPL e apresentar um bom desempenho e estabilidade. [6]

3.2 - PosgreSQL

O PostgreSQL começou a ser desenvolvido no inicio do ano de 1986, num projecto liderado por Michael Stonebraker. A primeira versão deste SGDB foi lançada em Junho de 1989, e no decorrer do tempo é possível constatar a sua evolução a nível de novas funcionalidades e também das diversas designações que lhe foram atribuídas ao longo do seu desenvolvimento. [12]

As principais características deste SGBD são: a facilidade de integração com outras linguagens de programação, integridade transaccional, o suporte para vários sistemas

operativos, a criação de *stored procedures* e *triggers* e a integração de várias linguagens procedimental, tais como, PL/pgSQL, PL/Java e PL/Tcl. [12]

O PostgreSQL é um software de acesso livre, com base na licença GNU BSD. [12]

3.2.1 - PL/Tcl

PL/Tcl é uma linguagem procedimental, que foi desenvolvida para o SGDB PosgreSQL, sendo possível o desenvolvimento de funções e *triggers* através da linguagem Tcl. [12]

3.3 - Java

Java nasceu num projecto iniciado em 1991 por James Gosling e Patrick Naughton, em 1995 foi lançada a primeira versão, onde o principal lema era "*Write Once, Run Anywhere*" (WORA). Esta tecnologia é uma linguagem de programação orientada a objectos, com suporte para vários sistemas operativos. Ao contrário das linguagens convencionais, a linguagem Java é compilada para um *bytecode* que é executado numa máquina virtual (JVM). [13,14]

A tecnologia Java disponibiliza três plataformas distintas para o desenvolvimento de software, sendo elas, Java SE, Java EE e Java ME. Neste projecto foi utilizada a plataforma Java EE, que herda todas as propriedades da plataforma Java SE e ainda tem a capacidade para o desenvolvimento de software distribuído. [13,14]

3.3.1 - Java Database Connectivity (JDBC)

JDBC é uma API disponível para a linguagem de programação Java, que possibilita a ligação entre uma aplicação e uma base de dados, o envio de instruções SQL e o processamento de resultados. [15]

A ligação entre uma aplicação e uma base de dados é efectuada através de um *driver*, que está dividido em quatro categorias [15]:

- Tipo 1: É a ponte entre JBDC e ODBC, onde métodos JDBC são convertidos em chamadas ODBC. Esta solução só deve ser utilizada quando não existe um outro tipo de driver disponível. [15]
- Tipo 2: É um *driver* API-Nativo, que converte as chamadas JBDC em chamadas da API do SGBD que está a ser utilizado. [15]
- Tipo 3: A ligação a um SGBD é feita através de um protocolo Middleware, que faz a ponte entre a aplicação e o SGBD. [15]
- Tipo 4: É um *driver* de ligação directa ao SGBD, ou seja, as chamadas feitas por uma aplicação são directamente enviadas para o SGBD. [15]

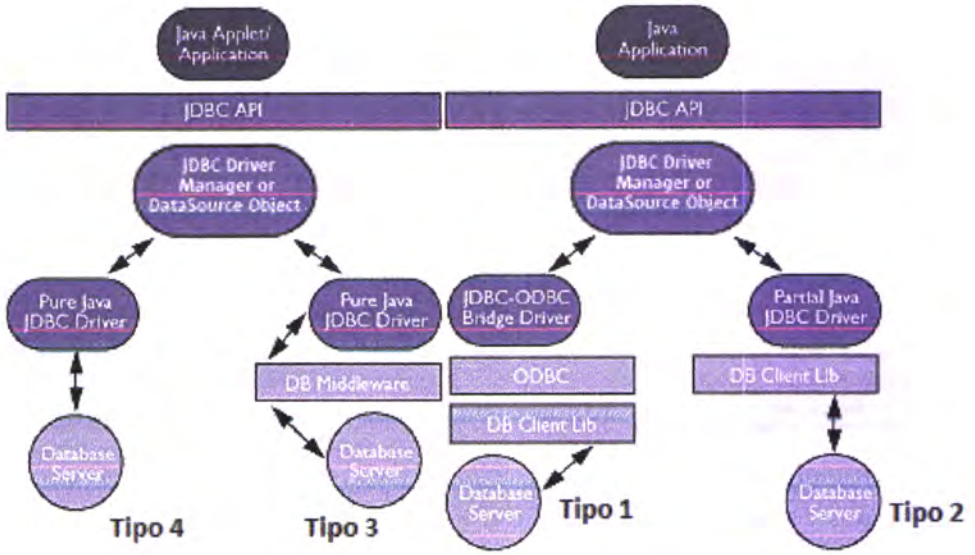

Figura 3.1: Arquitectura JDBC. [15]

No desenvolvimento deste projecto foram utilizados dois *drivers* do tipo 4, um para a conexão ao SGBD MySQL e outro ao SGBD PostgreSQL. Estes drivers são disponibilizados pelos próprios desenvolvedores dos referidos SGBDs. [16,18]

3.3.2 - Java Server Pages (JSP)

JSP é uma tecnologia Java, que facilita a criação de conteúdo *Web* dinâmico e além disso é possível ser executado em diferentes sistemas operativos. Esta tecnologia é constituída por código HTML, que misturado com etiquetas especiais é possível chamar conteúdos Java. [22]

3.3.3 - Java API for XML Web Services (JAX-WS)

JAX-WS é uma API Java, que é utilizada para o desenvolvimento de *Web Services*. Esta API faz parte do projecto GlassFish e tem como objectivo facilitar o desenvolvimento de aplicações Web [18]. Esta API define um contrato WSDL para garantir a interoperabilidade entre aplicações, através de chamadas de procedimentos remotas, ou seja, invocar funções remotas como se tratassem de funções locais [19].

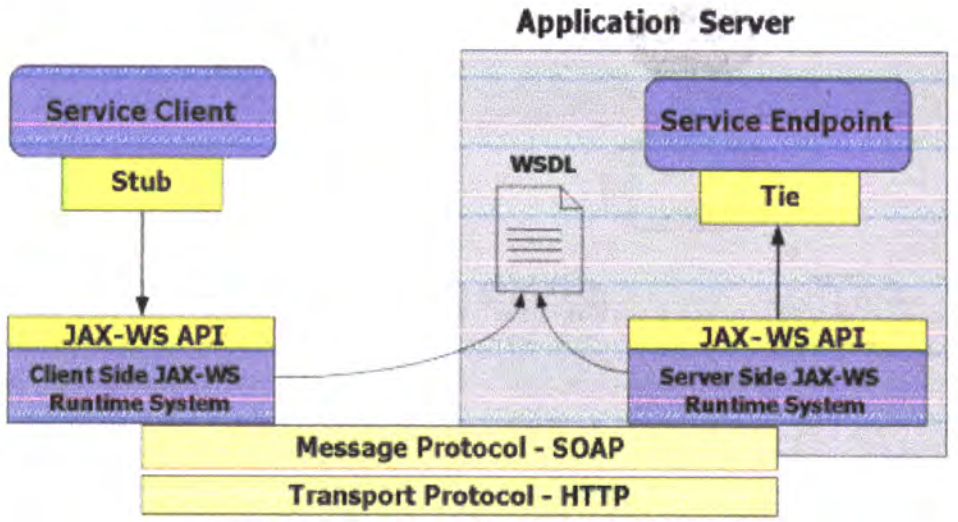

Figura 3.2: Estrutura JAX-WS. [19]

3.4 - GlassFish

GlassFish é um servidor de aplicações desenvolvido pela Sun, que foi anunciado em Junho de 2005, sendo a primeira versão lançada em Maio de 2006. Inicialmente tinha como objectivo oferecer total suporte à plataforma Java EE, no entanto, até ao dia de hoje tem vindo a apresentar novas funcionalidades. É um software distribuído livremente através das licenças GNU GPL e CDDL. [21]

As principais características do GlassFish, são: a alta performance, a compatibilidade com *frameworks* populares, o suporte para AJAX, o suporte total para Java EE e o suporte para linguagens dinâmicas como Ruby ou Groovy. [21]

Este servidor de aplicações é uma boa referência dentro da plataforma Java EE, permitindo o desenvolvimento de aplicações em diferentes tecnologias, tais como, JSP, JSF, *Servlet*, EJB, JAX-WS, JAXB, entre outras. [20]

3.5 - NetBeans

O NetBeans é um IDE que possibilita a criação de aplicações de desktop, Web, empresariais e móveis e ao mesmo tempo permite o uso das linguagens de programação, Java, PHP, JavaScript, Ruby, C/C++, entre outras. Este IDE é distribuído gratuitamente e pode ser executado em diferentes sistemas operativos. [23]

Através do NetBeans é possível desenvolver aplicações bastante sólidas, pois são disponibilizadas um grande conjunto de bibliotecas e documentação bem organizada. Além disso são proporcionadas ferramentas para auxiliar no desenvolvimento de código de uma aplicação. [24]

No desenvolvimento deste projecto, o uso do NetBeans foi bastante importante, pois facilitou a manipulação do servidor de aplicações GlassFish. A facilidade para gerar os ficheiros necessários para a construção do *Web Service* foi outro factor importante. [24]

Capítulo 4

Descrição do Trabalho Realizado

O presente capítulo faz a descrição de todo o trabalho desenvolvido para serem atingidos os objectivos propostos. Os seguintes tópicos estão divididos pelas diferentes fases deste projecto.

Estes tópicos foram analisados e discutidos antes de serem implementados, com base no estudo do Estado da Arte, na leitura de alguns livros e artigos sobre replicação de bases de dados.

Este projecto consiste no desenvolvimento de uma aplicação para replicação de base de dados, à qual foi dada o nome de "PIP Database Replicator". O principal objectivo, e tal como o nome indica, é a replicação de dados entre vários sistemas espalhados numa rede de computadores. Estão implementados mecanismos que facilitam a integração entre sistemas de diferentes plataformas, ou seja, não existe dependência a nível do sistema operativo utilizado. É também possível a integração de dados heterogéneos, ou seja, é possível copiar dados entre SGBDs diferentes. Estão, também, disponíveis várias formas de replicar os dados e de propagação de dados, podendo escolher o utilizador aquilo que pretende.

Em 4.1 são descritas as tabelas de histórico que têm a função de auxiliar a aplicação desenvolvida; na subsecção 4.2 são enumeradas as configurações disponíveis na aplicação e a sua função. A comunicação entre sistemas é detalhada em 4.3 e na subsecção 4.4 são descritos os modelos de publicação e subscrição de dados. Na subsecção 4.5 é descrita a verificação de actualizações; em 4.6 são enumeradas as formas como os dados são propagados, a integração de dados que foi desenvolvida na aplicação é descrita em 4.7. A fragmentação horizontal e vertical desenvolvida neste projecto é detalhada em 4.8, em 4.9 onde são descritas as formas para transformação de dados; a monitorização do sistema é descrita em 4.10 e a interface para o utilizador é apresentada em 4.11.

4.1 - Tabelas de Histórico

As tabelas de histórico têm um papel bastante importante no funcionamento da aplicação desenvolvida, pois é através delas que se obtém toda a informação sobre as actualizações numa dada tabela. São duas tabelas que constituem o histórico de uma tabela, a primeira regista a data de actualização, a operação realizada (*INSERT*, *UPDATE* ou *DELETE*), o utilizador que efectuou a operação e a chave primária da tabela actualizada. A segunda tabela de histórico,

apenas guarda informação no caso de uma operação ser um *UPDATE* ou um *DELETE* e além da data de actualização, guarda os dados de todos os campos da tabela que foi actualizada.

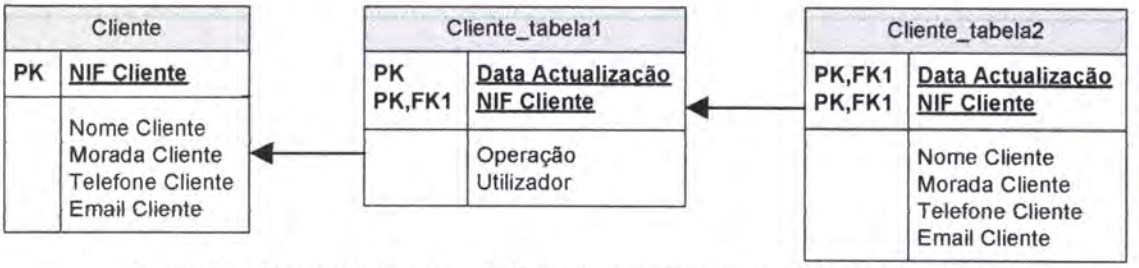

Figura 4.1: Exemplo de uma tabela mais as respectivas tabelas de histórico.

Para o SGBD PostgreSQL, foi criado um *trigger* que chama um *stored procedure* para registar a informação necessária nas tabelas de histórico, ou seja, sempre que ocorre uma actualização numa tabela é accionado um *trigger* que vai chamar um *stored procedure*. O *trigger* tem a função de actuar sempre que ocorre uma operação (*INSERT*, *UPDATE* ou *DELETE*). O *stored procedure* recebe a informação necessária para ser guardada nas tabelas de histórico.

No SGBD MySQL, a actualização de tabelas de histórico é baseada apenas em três *triggers*, cada um correspondente a uma operação que possa ocorrer numa tabela. Cada *trigger* é responsável de inserir nas tabelas de histórico a informação referente à operação que ocorreu numa dada tabela.

As tabelas de histórico podem ser criadas através do interface da aplicação, de uma forma bastante simples. Para o SGBD PostgreSQL existe a excepção de que é preciso ter previamente instalado o *stored procedure* na base de dados a que pertence a tabela pretendida. Para o correcto funcionamento da aplicação é importante que as tabelas a serem replicadas tenham instalados as tabelas de histórico correspondentes.

4.2 - Configurações

Em certos momentos é preciso indicar à aplicação certos parâmetros para a execução de uma determinada tarefa, por isso foi criado no sistema, a possibilidade do utilizador escolher as configurações da aplicação consoante o pretendido.

As configurações do sistema referem-se às configurações para a conexão aos SGBDs disponíveis e também a parâmetros para um correcto funcionamento da aplicação. Pode-se então verificar as seguintes configurações:

- **Utilizador PostgreSQL**: Nome do utilizador para conexão ao SGBD PostgreSQL.
- **PostgreSQL *Hostname***: Nome do anfitrião mais o porto para conexão ao SGBD PostgreSQL (Ex: localhost:8080).

- ***Password* PostgreSQL**: *Password* de acesso ao SGBD PosgreSQL.
- **Utilizador MySQL**: Nome do utilizador para conexão ao SGBD PosgreSQL.
- **MySQL *Hostname***: Nome do anfitrião mais o porto para conexão ao SGBD MySQL.
- ***Password* MySQL**: *Password* de acesso ao SGBD MySQL.
- **SGBD do Sistema**: Indica o SGBD que tem a base de dados de sistema, que contem as informações para a replicação de dados ou monitorização do sistema.
- **Frequência para verificação de subscrições**: Indica o período de tempo em que se verificam as actualizações nas subscrições submetidas.
- **Frequência para verificação de publicações**: Indica o período de tempo em que se verificam as actualizações nas publicações submetidas.
- **Tipo de Replicação**: Refere como os dados devem ser replicados, de forma normal ou em cascata.
- **Hora para replicações não instantâneas**: Indica a hora em que se verificam as subscrições a serem replicadas de forma não instantânea.
- **Tempo de Resposta**: Refere o tempo máximo de resposta que outro sistema tem para responder.
- **Número de Tentativas**: Refere o número de tentativas para verificar a disponibilidade de um sistema.

Todas as configurações aqui descritas são guardadas num ficheiro encriptado, de modo a garantir, também alguma segurança. O utilizador da aplicação poderá manipular as configurações através do interface disponibilizado.

4.3 - Comunicação entre Sistemas

Uma das fases mais importantes do desenvolvimento deste projecto foi a forma como os vários sistemas disponíveis comunicavam entre si. Foi então desenvolvido um *Web Service*, de forma a ser possível a comunicação entre sistemas. Uma das razões para a escolha deste meio de comunicação foi a facilidade de integração com diferentes plataformas. [25]

Os *Web Services* comunicam entre si através de mensagens XML, onde existe um ficheiro WSDL, que é um espécie de contrato onde são definidos os métodos que podem ser executados por uma aplicação remota. [25]

Para o desenvolvimento deste projecto e com o auxílio do IDE NetBeans e da API JAX-WS, foi gerado um ficheiro WSDL, para permitir a comunicação entre os vários sistemas disponíveis. No ficheiro WSDL da aplicação estão disponíveis os seguintes métodos:

- **Verifica Disponibilidade**: Um sistema responde positivamente, caso tenha recebido correctamente uma mensagem. Este método serve para verificar se um sistema está ou não disponível.
- **Obtém Bases de Dados**: Neste método a aplicação local informa a aplicação remota quais as bases de dados que tem disponíveis.
- **Obtém Tabelas**: São devolvidas todas as tabelas de uma dada base de dados.
- **Obtém Campos**: São devolvidos todos os campos de uma dada base de dados e uma tabela.
- **Nova Subscrição**: A aplicação remota informa que subscreveu uma nova tabela.
- **Recebe Dados**: Este método é responsável por receber os dados de replicação, enviados por uma aplicação remota.
- **Obtém Publicações**: A lista de tabelas publicadas é disponível através deste método.
- **Verifica Actualizações de uma Publicação**: Através deste método, a aplicação remota verifica se existem novas actualizações para uma dada bases de dados e uma tabela.
- **Obtém Dados de uma Publicação**: A aplicação remota obtém os dados de uma publicação através deste método.
- **Actualização de Subscrição bem Sucedida**: Através deste método a aplicação local é informada que uma subscrição foi replicada com sucesso.
- **Obtém Dependências**: Devolve as dependências de uma dada base de dados e uma tabela.

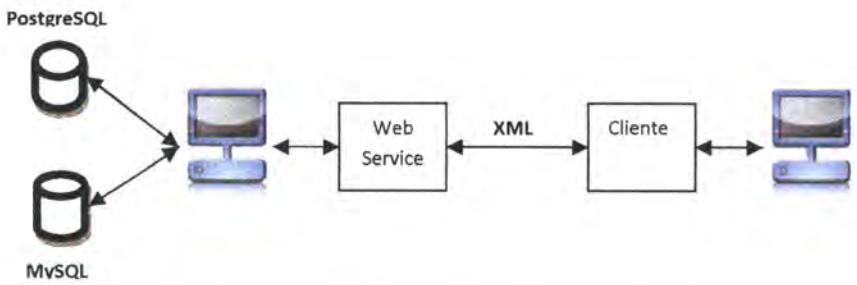

Figura 4.2: Exemplo da comunicação "Cliente-Servidor".

Estes métodos podem ser chamados através de um módulo "cliente" disponível na aplicação, que pode executar qualquer um dos métodos descritos anteriormente, tal como é exemplificado na figura 4.2. O "servidor" responsável pelo envio dos dados, não tem informações sobre o destino dos dados que envia, o "cliente" que recebe os dados é que deve ter a capacidade de distribuir correctamente a informação que recebe.

Para que a aplicação possa reconhecer as máquinas "vizinhas", que permitem enviar e receber dados, foi criada uma tabela de sistema onde são guardadas as informações necessárias para que o sistema reconheça as maquinas que podem exercer essa funcionalidade.

Database Group	
PK	**Hostname**
	Port

Figura 4.3: Tabela de sistema para identificação de sistemas "vizinhos".

4.4 - Subscrição e Publicação de Dados

O utilizador tem duas hipóteses para replicar os dados de uma base de dados, através de subscrições ou publicações, ou seja, o sistema é informado como deve enviar e receber os dados.

Numa subscrição, o utilizador informa uma fonte de dados, quais as tabelas que pretende copiar para o seu sistema. O sistema que envia os dados regista na base de dados de sistema que existe um novo destino para replicar os dados de uma determinada tabela. O sistema que fez o pedido de subscrição é também informado que existe uma réplica de dados a ser recebida.

Outra forma para copiar dados é através de publicações, onde uma fonte de dados pública as tabelas que pretende replicar. O sistema que pretenda obter uma publicação deve informar quais as publicações que pretende receber, de modo a obter uma cópia dos dados posteriormente.

Além do que já foi mencionado, uma das grandes diferenças entre uma subscrição e uma publicação é a forma como os dados são replicados. Numa subscrição a fonte envia os dados para um destino (Replicação Enviada). Numa publicação o destino faz um pedido dos dados à fonte (Replicação e Pedido).

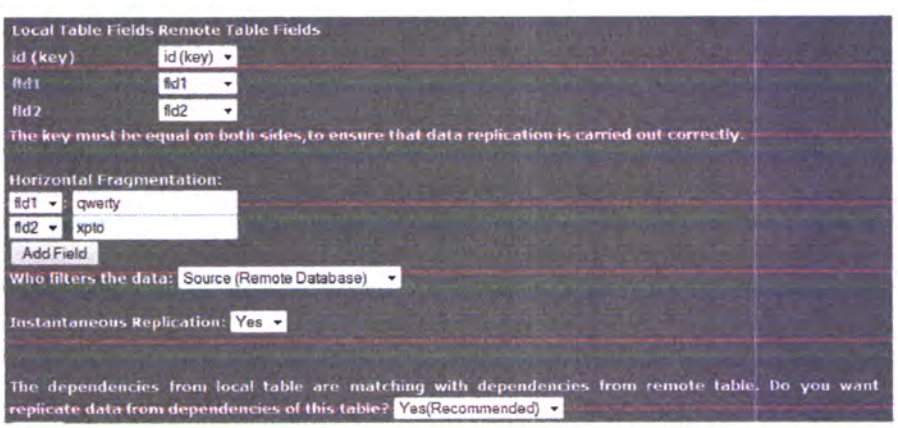

Figura 4.4: Exemplo de uma subscrição.

Numa subscrição ou publicação foi implementada a hipótese de fragmentação horizontal e vertical, ou seja, o utilizador pode escolher como deseja que os dados sejam filtrados. Para a fragmentação vertical deve-se definir a correspondência de campos pretendida e para a fragmentação horizontal deve-se adicionar os campos pretendidos e o respectivo valor, como se pode constar na Figura 4.4.

Também é possível definir quem filtra os dados, ou seja, quem fica encarregue da fragmentação horizontal e vertical definida pelo utilizador. Esta opção possibilita ao utilizador escolher o melhor para o seu sistema, evitando a sobrecarga quer da fonte ou do destino dos dados.

Quando existem dependências nas tabelas a serem subscritas ou publicadas, existe a hipótese de, também, serem replicadas as dependências da tabela principal a ser replicada. Isto é, caso seja pedido o sistema vai também enviar os dados das tabelas "pai" que correspondem à tabela pretendida.

Caso se trate de uma subscrição, é possível definir se a replicação é ou não instantânea. Caso seja instantânea o sistema vai enviar os dados assim que possível, no caso de ser não instantânea, o sistema envia os dados numa hora previamente definida pelo utilizador.

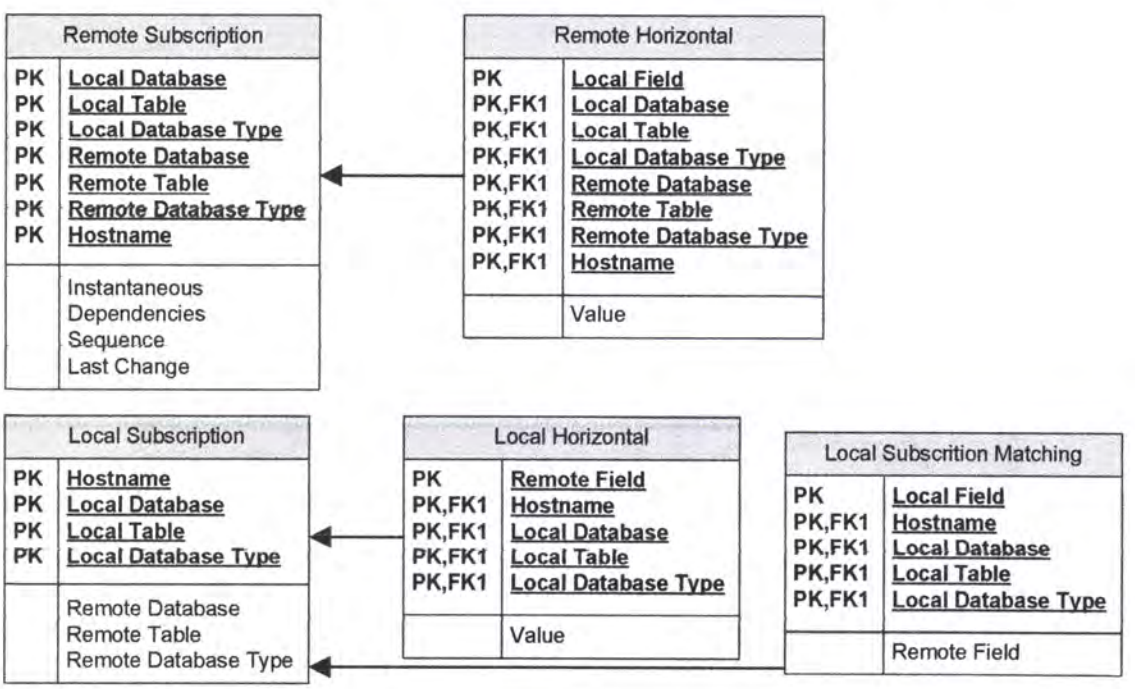

Figura 4.5: Tabelas de Sistema para Subscrições.

Na Figura 4.5, estão representadas as tabelas de sistema para guardarem os dados referentes às subscrições que possam existir. A tabela "Remote Subscription" tem a informação das subscrições a serem enviadas, onde "Hostname" é o nome do anfitrião que vai receber os dados,

"Sequence" define a ordem de envio de uma subscrição e "Last Change" contém a data do último volume de dados enviado. A tabela "Remote Horizontal" só é utilizada caso seja a fonte dos dados a filtrar os dados a serem enviados. A tabela "Local Subscription" contém os dados das subscrições a serem recebidas, para que o sistema identifique a base de dados e a tabela que vai receber os dados enviados pela fonte. "Local Horizontal" é a tabela utilizada para guardar os dados caso seja o destino a filtrar os dados. A tabela "Local Subscription Matching" trata da correspondência de campos que possa existir, ou seja, tem as informações necessárias para a fragmentação vertical.

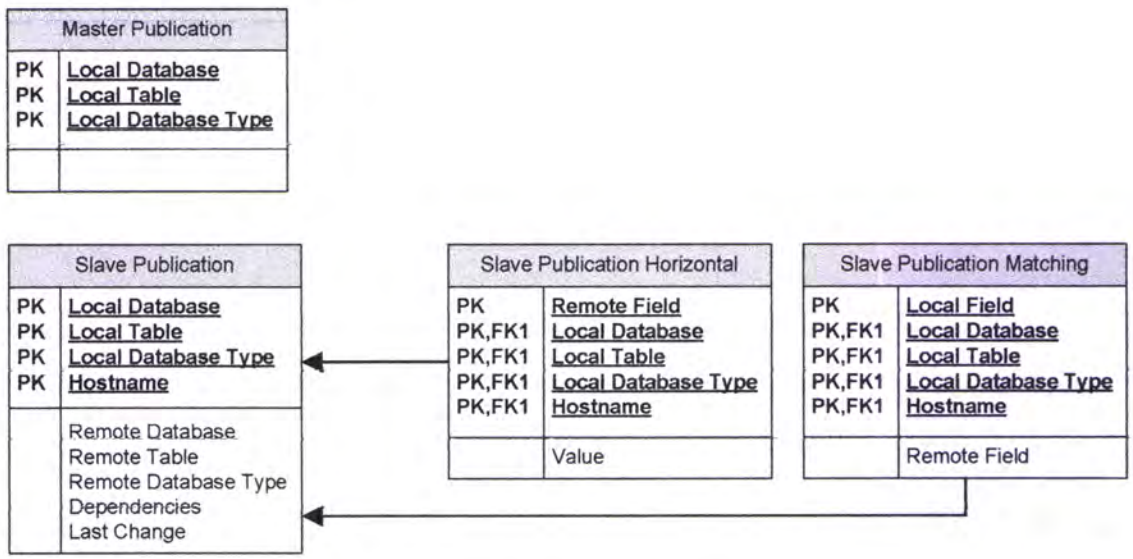

Figura 4.6: Tabelas de Sistema para Publicações.

As informações sobre as publicações também estão guardadas numa tabela de sistema como se pode constar na Figura 4.6. A tabela "Master Publication" regista todas as tabelas que estão publicadas numa fonte de dados. "Slave Publication" é a tabela que guarda as informações das publicações que foram subscritas, sendo "Hostname" o nome do anfitrião de onde vão ser extraídos os dados e "Last Change" a data da última alteração. As tabelas "Slave Publication Horizontal" e "Slave Publication Matching" têm a mesma função das tabelas "Local Subscription Horizontal" e "Local Subscription Matching", respectivamente.

4.5 - Verificação de Actualizações

Um dos momentos importantes durante a execução da aplicação é a percepção se existem ou não actualizações. Como foi referido no tópico anterior, quer as subscrições quer as publicações têm um campo "Last Change", que refere a data da última actualização de cada subscrição ou publicação.

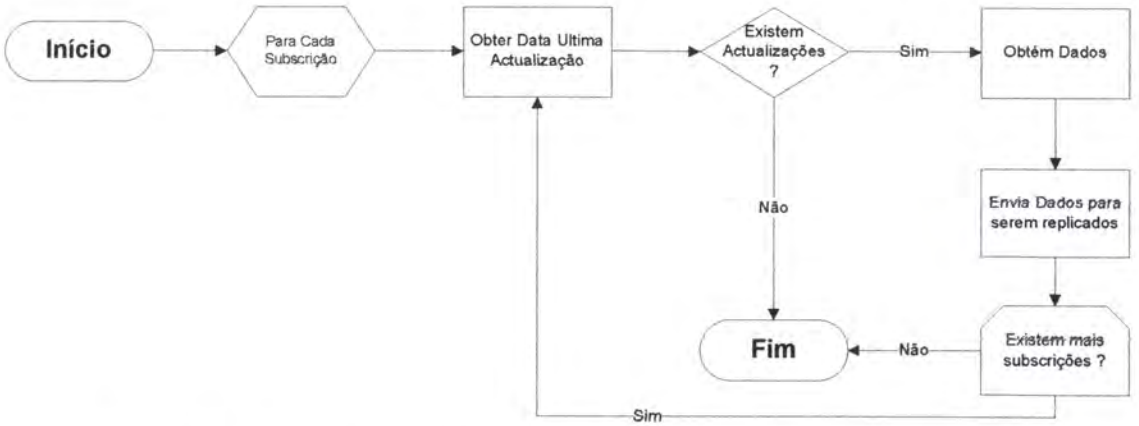

Figura 4.7: Fluxograma para verificação de actualizações numa subscrição.

Para as subscrições, a fonte de dados está constantemente, num período de tempo definido pelo utilizador, à procura de actualizações. Este procedimento está representado na Figura 4.7, através de um fluxograma.

Nas publicações a verificação de actualizações envolve a comunicação entre a fonte e o destino dos dados. O destino dos dados verifica para todas as publicações subscritas a data da última actualização, que será comunicado à fonte de dados. A fonte de dados verifica se existem actualizações, caso existam, estes serão enviados para o destino.

4.6 - Propagação dos Dados

Como já foi referido no tópico 4.2, existem duas formas de propagar os dados, uma mais comum denominada "Normal", onde uma fonte envia para vários destinos e outra denominada "Cascata" onde uma fonte envia para um destino que envia para outro destino e assim sucessivamente.

Figura 4.8: Esquema de uma propagação de dados "Normal".

Numa propagação de dados "Normal", a aplicação pode enviar o mesmo volume de dados para vários destinos ao mesmo tempo. Esta opção é viável, no entanto, pode sobrecarregar a fonte de dados caso sejam enviados muitos dados ao mesmo tempo. Em caso de falha de comunicação a fonte não enviar os dados para um destino, que apenas receberá os dados quando estiver disponível.

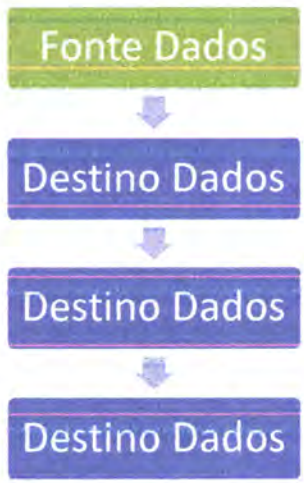

Figura 4.9: Esquema de uma propagação de dados em "Cascata".

Numa propagação de dados em "Cascata", como é ilustrado na Figura 4.9, a fonte de dados envia os dados para um destino, que fica encarregue e enviar para outro destino e assim sucessivamente. Caso um destino não esteja contactável, de modo a receber os dados, os dados serão enviados para destino a seguir, quando todos os destinos tiverem recebido a cópia dos dados, o último destino a receber os dados vai tentar enviar novamente para os destinos que tenham falhado durante o ciclo normal da propagação em "Cascata".

A comunicação entre uma fonte e um destino é feita através do *Web Service* descrito no tópico 4.3, cada destino para onde são enviados os dados tem de responder dentro do tempo definido pelo utilizador. Em caso de falha, a fonte insiste novamente até esgotar o número de tentativas definidas pelo utilizador, se ainda assim o sistema não responder, este será dado como incontactável. Os dados que são enviados entre os sistemas, são colocados dentro de uma lista para que possam ser enviados através do *Web Service*.

Tudo o que está descrito neste tópico refere-se à propagação dos dados numa subscrição, no caso de uma publicação e como se trata de um sistema de replicação a pedido, o destino dos dados pede os dados à fonte. Este pedido é feito através do *Web Service* implementado, o destino faz o pedido à fonte de dados enviando a data da última actualização, caso existam actualizações a fonte responde enviando os dados actualizados, como resposta.

4.7 - Integração de Dados Heterogéneos

Desde o início do desenvolvimento deste projecto que foi planeada a integração de dados heterogéneos, isto é, a cópia de dados entre SGBDs diferentes, onde o objectivo é demonstrar que é possível copiar informação não só entre diferentes plataformas, mas também entre SGBDs diferentes.

No tópico 4.4 pode verificar-se nas tabelas de sistema das subscrições e publicações um campo denominado "Local Database Type" ou "Remote Database Type", que servem para identificar o tipo de base de dados com que o sistema está a lidar.

Como foi referido no capítulo 3, no desenvolvimento deste projecto foram utilizados os SGBDs PostgreSQL e MySQL, onde é possível integrar dados entre eles. Na aplicação, foram criados dois módulos, um para manipular o SGBD PostgreSQL e outro o SGBD MySQL, desta forma é possível responder aos vários pedidos quando a aplicação está presente sobre um destes tipos.

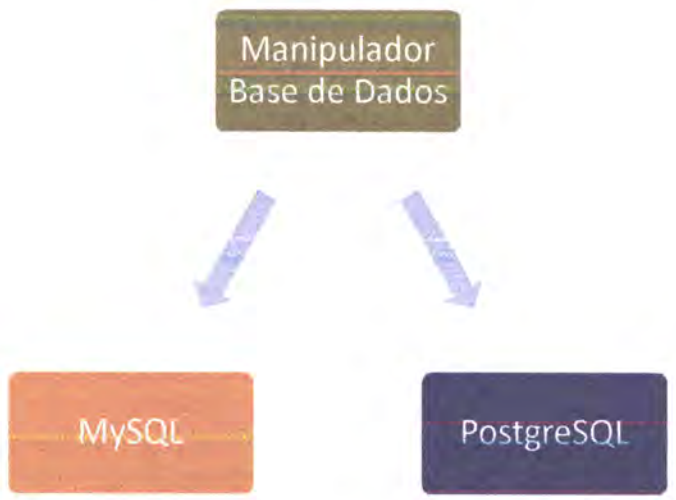

Figura 4.10: Exemplo de reconhecimento do SGBD a ser usado.

Para distinguir os dados PostgreSQL e MySQL, foi desenvolvido um módulo para reconhecimento da base de dados que está a ser utilizada, que ao reconhecer o tipo de base de dados encaminha os dados a serem manipulados para o módulo do SGBD correspondente.

Em suma, quando a aplicação recebe ou envia dados, estes são determinados não só pela base de dados e tabela, mas também pelo tipo (MySQL ou PostgreSQL), que determina o módulo a ser usado pela aplicação.

4.8 - Fragmentação Horizontal e Vertical

Tal como foi descrito em 2.1.2, a fragmentação horizontal e vertical trata de filtrar os dados. Neste projecto foi implementado uma solução para fragmentação horizontal e vertical.

Existem duas soluções implementadas para a fragmentação horizontal, uma quando é a fonte a filtrar os dados e outra quando é o destino a filtrar os dados. Servem estas soluções para o utilizador adaptar o sistema consoante as necessidades, ou seja, pode definir a fonte como responsável da filtragem dos dados, caso não exista uma grande sobrecarga ou então enviar todos os dados para o destino, evitando perder tempo a filtrar dados.

No caso dos dados serem filtrados na fonte, a aplicação reconhece quais os campos a serem filtrados e os respectivos valores. Quando a extracção dos dados é feita, a fonte transmite ao respectivo SGBD, os dados que pretende e a respectiva filtragem. Na situação em que o destino filtra os dados, é verificado se existem filtragens a realizar. Caso existam, o destino analisa os dados recebidos e filtra-os consoante os campos e valores anteriormente definidos pelo utilizador.

Numa fragmentação vertical, são seleccionadas as colunas que o utilizador definiu. Ao definir as colunas pretendidas é ainda possível definir uma correspondência de colunas entre a fonte e o destino de dados. Este tipo de filtragem é feito apenas pelo destino dos dados, que ao receber dados provenientes de uma fonte vai analisar e ordenar as colunas que o utilizador pretende copiar.

A fragmentação horizontal e vertical pode ser definidas através da submissão de novas subscrições ou publicações.

4.9 - Transformação de Dados

A transformação de dados é outra funcionalidade implementada no projecto desenvolvido, que trata de transformar os dados vindos de uma fonte num formato ou tipo adequado ao destino dos dados. Existem duas formas para transformação de dados, através de uma tabela de conversão ou de uma função.

Uma tabela de conversão é uma forma simples de transformar o valor de um determinado campo noutro valor pretendido por quem recebe um determinado volume de dados. Para a aplicação ter o conhecimento de quais os dados a transformar, foi criada uma tabela de sistema associada às subscrições e publicações, assim o utilizador informa o sistema do que pretende converter, como se pode constar na figura 4.11.

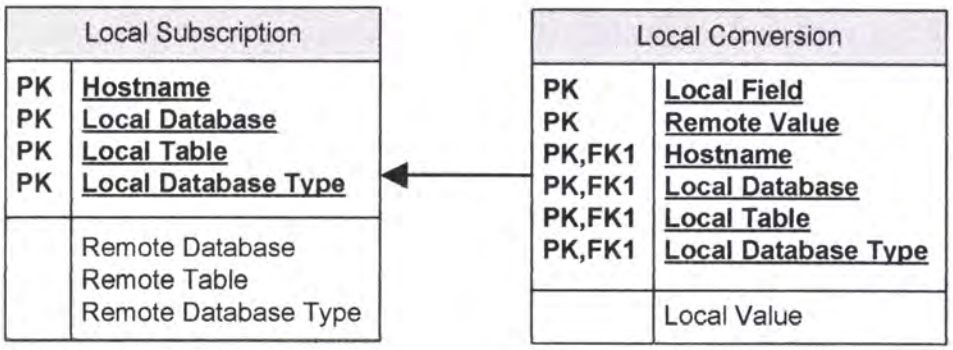

Figura 4.11: Tabela de Conversão associada a uma subscrição.

Para a transformação de dados é também possível aplicar uma função para transformar o valor de um determinado campo. Para este tipo de transformação estão pré-definidas algumas funções, que o utilizador pode escolher. As funções disponíveis na aplicação desenvolvida, são:

- **LowerCase**: Converte todos os caracteres de uma *string* em minúsculas.
- **UpperCase**: Converte todos os caracteres de uma *string* em maiúsculas.
- **Compact**: Remove os espaços em branco de uma *string* (Exemplo: "AA ␣BC" em "AABC").
- **FirstUpperCase**: O primeiro carácter de uma *string* é convertido em maiúscula.
- **Clean**: Remove os caracteres especiais de uma *string* (Exemplo: ã, ç, é).
- **Trim**: Remove os espaços iniciais de uma *string*, caso existam.

Tal como nas tabelas de conversão, para informar o sistema qual a função pretendida para transformar o valor de um campo, foi criada uma tabela de sistema como está representado na figura 4.12. Ao informar qual o campo pretendido a função vai transformar todos os valores associados.

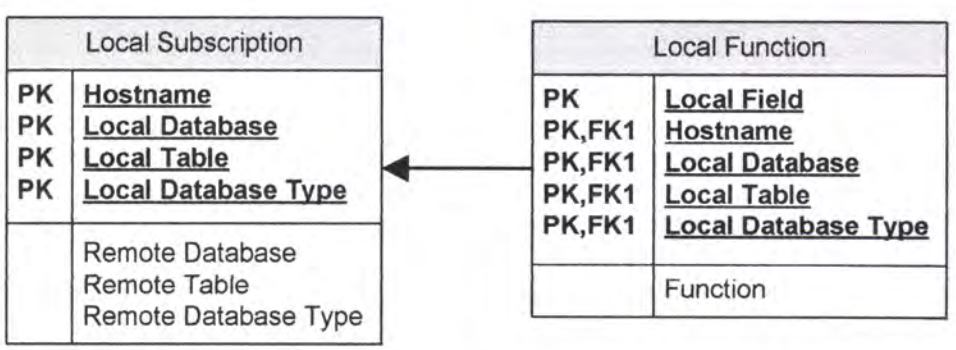

Figura 4.12: Tabela de transformação através de funções associada a uma subscrição.

As transformações de dados mencionadas neste tópico são aplicadas na recepção dos dados e não no envio, de modo a não sobrecarregar a fonte de dados.

4.10 - Monitorização do Sistema

Para o controlo dos dados enviados ou recebidos e de erros que possam ocorrer, foi criado um módulo para monitorizar o sistema. A função deste módulo é apenas visualizar, em modo geral, as ocorrências no sistema para ajudar o utilizador a tomar alguma decisão caso seja necessário.

Para a monitorização do sistema foram criadas tabelas de sistema, onde são guardadas as informações sobre subscrições e publicações enviadas ou recebidas e também de erros que possam surgir durante a execução da aplicação. Na Figura 4.13 podem-se verificar as tabelas de sistemas para auxiliar na monitorização do sistema.

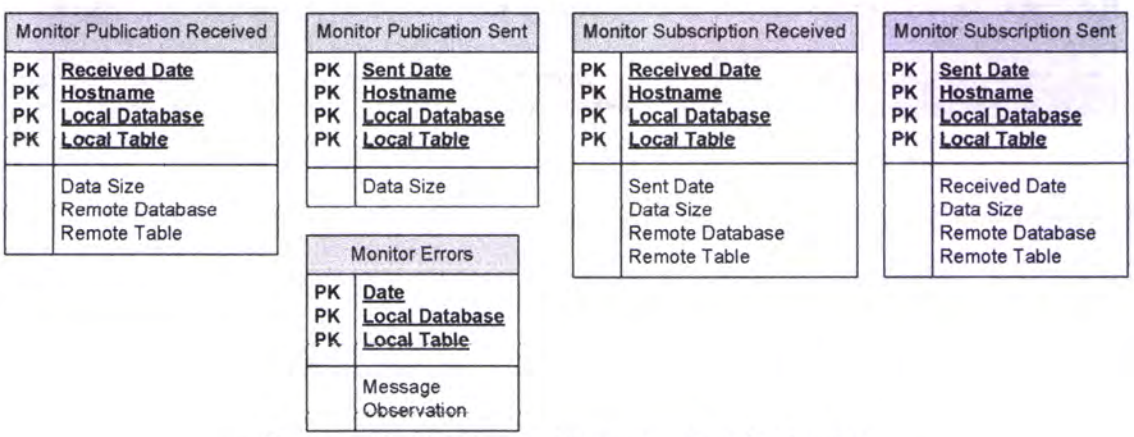

Figura 4.13: Tabelas de sistema para monitorização.

Sempre que ocorre um envio ou recepcionamento de dados, a aplicação regista a informação necessária na tabela de sistema correspondente à ocorrência. No caso de um erro, sempre que ocorre um erro na aplicação ou na comunicação entre sistemas é registado na tabela de monitorização correspondente.

4.11 - Interface para o Utilizador

Ao longo deste capítulo foram descritas várias características da aplicação desenvolvida, no entanto ainda não foi descrito a base do funcionamento da aplicação. O interface desenvolvido para esta aplicação é uma ferramenta bastante útil, pois é através dela que se gere todo o sistema de replicação.

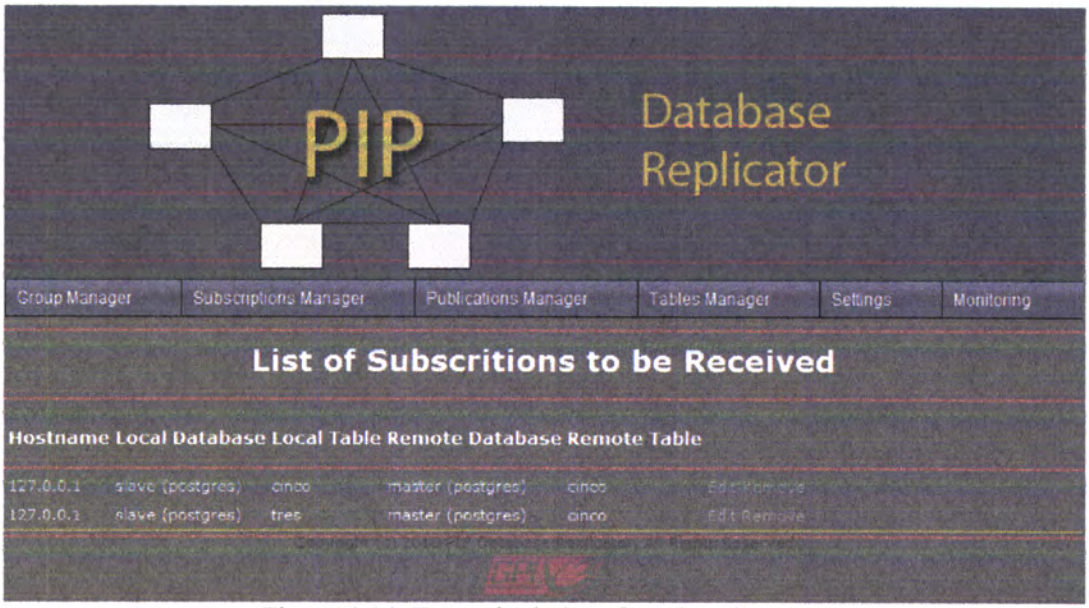

Figura 4.14: Exemplo do interface da aplicação.

O interface permite ao utilizador gerir as subscrições, gerir as publicações, gerir as tabelas de uma determinada base de dados, gerir o grupo de sistemas, alterar as definições e monitorização do sistema.

Este interface pode ser executado através de qualquer Web Browser, não existindo, também, restrições a nível de plataforma usada.

Capítulo 5

Avaliação de Resultados

Neste capítulo, são analisados os resultados obtidos após a finalização do projecto, bem como o seu funcionamento perante casos reais. São descritas as mais-valias ou funcionalidades que o projecto oferece, a facilidade de integração entre diferentes SGBDs, o peso do sistema quando tem dados para replicar, a tolerância a falhas e também serão demonstrados alguns testes realizados.

O projecto desenvolvido no âmbito desta dissertação de mestrado está disponível para a comunidade *Open Source* e pode ser consultado em, http://pipdbreplicator.sourceforge.net/.

5.1 - Funcionalidades

Após o desenvolvimento do projecto é possível analisar as funcionalidades disponíveis consoante o que é pretendido pelo utilizador final. Todas as funcionalidades implementadas foram previamente analisadas, de modo a serem disponibilizadas segundo o pretendido numa aplicação de replicação de base de dados.

O sistema de histórico é uma das funcionalidades base disponível, pois sem ele é impossível verificar as ocorrências numa determinada tabela. Além disso é possível aproveitar o sistema de histórico para o uso em outras aplicações que manipulem as mesmas bases de dados, pois as tabelas de histórico são criadas na mesma base de dados onde se encontra a tabela principal.

A possibilidade de replicar dados entre bases de dados mestre é outra das funcionalidades que se destacam, pois assim é possível manter todas as bases de dados distribuídas actualizadas entre si.

As variadas formas de distribuição dos dados são uma funcionalidade que se pode destacar, pois o utilizador da aplicação poderá escolher aquilo que lhe é mais conveniente consoante as suas necessidades. O sistema oferece uma solução de Replicação Enviada, onde a fonte de dados recebe subscrições dos destinos, para depois enviar os dados para os respectivos receptores, esta distribuição poderá ser realizada em cascata, caso seja pretendido pelo utilizador. No sistema também está presente uma solução de Replicação a Pedido, onde a fonte de dados pública as tabelas pretendidas e os receptores fazem um pedido dos dados publicados.

A fragmentação horizontal e vertical é outra solução apresentada ao utilizador, onde é possível filtrar os dados pretendidos e adaptá-los ao receptor dos respectivos dados.

O *Web Service* desenvolvido possibilita a integração com outras aplicações que possam vir a ser desenvolvidas, podendo o utilizador adaptar-se a uma solução mais conveniente para o seu sistema.

	PIP Database Replicator
Multi-Master	Sim
Master-Slave	Sim
Assíncrono	Sim
Síncrono	Não
Dados Heterogéneos	Sim
Multi-Plataforma	Sim
Transformação de Tipos	Sim
Fragmentação Horizontal	Sim
Fragmentação Vertical	Sim
Vários Schemas	Sim
Transacções distribuídas e Two Phase commit	Não
Distribuição por Cascata	Sim
Push ou Pull Replication	Sim

Tabela 5.1: Funcionalidades da aplicação desenvolvida.

A Tabela 5.1 faz um resumo de todas as funcionalidades disponíveis, com base no estudo realizado no capítulo 2. Como se pode verificar, a aplicação disponibiliza quase todas as funcionalidades que outras aplicações podem ou não oferecer. Apesar de a aplicação não disponibilizar replicação síncrona, existe a possibilidade de copiar dados logo que possível, de forma a aproximar-se o máximo de uma replicação síncrona. Como não existe replicação síncrona, não poderá também existir transacções distribuídas, no entanto uma transacção ocorrida localmente poderá ser totalmente copiada e posteriormente enviada para outros sistemas. Apesar de não existir nenhum módulo de 2PC directamente disponível na aplicação, o sistema reconhece quando uma réplica ocorreu totalmente ou não. Caso não tenha ocorrido com sucesso a réplica dos dados, o sistema enviará de novo todos os dados de modo a garantir que a cópia é efectivamente realizada.

	PIP	TUNGSTEN	DAFFODIL
Replicação e Clustering de Base de Dados	Sim	Sim	Sim
Master-Slave	Sim	Sim	Não
Multi-Master	Sim	Não	Não
Replicação Síncrona	Não	Não	Não
Replicação Assíncrona	Sim	Sim	Sim
Multi SGBBs	Sim	Sim	Sim
Multi Plataforma	Sim	Sim	Sim
Transformação de Dados	Sim	Sim	Sim
Fragmentação Horizontal	Sim	Não	Sim
Fragmentação Vertical	Sim	Não	Sim
Vários Schemas	Sim	Sim	Sim
Push ou Pull replication	Sim	Não	Sim
Replicação em Cascata	Sim	Não	Não
Base em mecanismo de histórico	Sim	Não	Não
Base em logfiles	Não	Sim	Sim
Verificação de Consistência	Sim	Sim	Sim
Tolerância a Falhas	Sim	Sim	Sim
Conflitos - detecção e resolução	Não	Sim	Sim
Segurança	Não	Sim	Sim
Calendarização	Sim	Sim	Sim
On demand	Sim	Não	Não
Log e report de erros	Sim	Sim	Sim

Tabela 5.2: Comparação entre aplicações

Na tabela 5.2 é feita uma comparação de funcionalidades entre a aplicação desenvolvida nesta dissertação e outras duas aplicações *open source* mais representativas.

5.2 - Adaptação a diferentes SGBDs

Uma das características chave da aplicação desenvolvida é a facilidade de integração com diferentes SGBDs. O utilizador apenas tem de se preocupar em escolher o que pretende e como usar.

Foi referido no Capítulo 3 que apenas os SGBDs MySQL e PostgreSQL estão disponíveis neste projecto, no entanto, será fácil a integração de outros SGBDs, desde que exista um driver JDBC compatível para a manipulação do SGBD pretendido. Depois será necessário adaptar os métodos desenvolvidos para a interacção com o SGBD.

Ao longo do Capítulo 4 foram descritas algumas tabelas de sistema, isto é, que são auxiliares da aplicação para tomar uma decisão. Estas tabelas podem ser integradas numa base de dados MySQL ou PostgreSQL. O utilizador deve criar uma base de dados no SGBD pretendido, instalar as tabelas de sistema e depois informar na aplicação, qual o SGBD que tem a base de dados e tabelas de sistema.

Nos testes onde se pretende demonstrar a integração entre duas bases de dados, foram consideradas as seguintes opções:

- (**A**) Simples: consiste a replicação total dos dados.
- (**B**) Fragmentação Horizontal com filtragem na fonte
- (**C**) Fragmentação Horizontal com filtragem no destino
- (**D**) Fragmentação Vertical
- (**E**) Fragmentação Horizontal e Vertical com filtragem na fonte
- (**F**) Fragmentação Horizontal e Vertical com filtragem no destino
- (**G**) Dependências: consiste na replicação de uma tabela mais respectivas dependências.

As opções estão identificadas por letras entre A e G, de forma a identificar as opções nas tabelas de resultados que estão descritas nesta subsecção. Na fragmentação horizontal são filtrados 10% dos dados totais, ou seja, se uma tabela tiver 10000 registos apenas vão ser replicados 1000 registos. Na fragmentação vertical foram filtradas três colunas de uma tabela de cinco colunas. Na replicação com dependências é utilizada uma tabela que tem 10% de registos de uma tabela "pai", ou seja, vão ser replicados os 10% dos registos da tabela "filho" mais 10% dos registos da tabela "pai". Os valores apresentados nas tabelas de resultados estão em segundos.

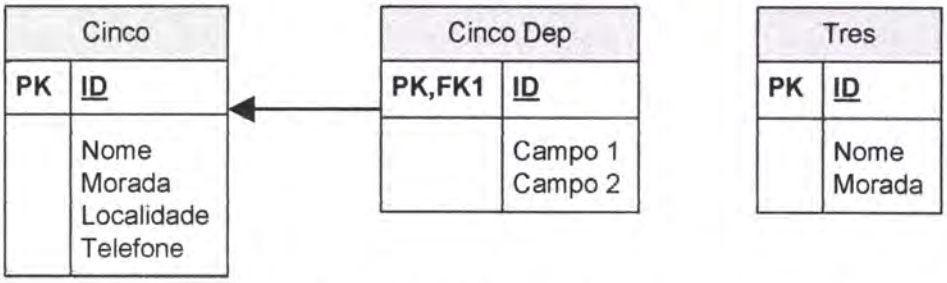

Figura 5.1: Tabelas utilizadas para nos testes realizados.

As tabelas representadas na Figura 5.1 serviram de auxílio para a realização dos testes. A tabela principal é a "Cinco" que tem a maioria dos dados a serem replicados; a tabela "Cinco Dep" tem dados que são dependentes da tabela "Cinco"; por fim a tabela "Tres" apenas será usada na fragmentação vertical e por isso estará apenas presente no destino dos dados.

Para a realização destes testes foram utilizados os computadores pessoais PC1 e PC2.

Volume Dados	A	B	C	D	E	F	G
5000	106	16	16	67	11	11	22
20000	404	51	52	260	37	39	92
50000	1048	133	124	613	89	89	286
100000	2786	285	289	1452	173	173	642

Tabela 5.3: Resultados da integração entre duas BDs PostgreSQL.

A replicação de dados entre duas bases de dados PostgreSQL mostra um desempenho esperado, consoante o aumento do volume de dados mais tempo demora a replicação. Na filtragem de dados, quando se trata de fragmentação horizontal o tempo de execução é bastante menor do que quando se copiam a totalidade de dados, não existindo diferenças relevantes na distinção se é a fonte ou o destino quem filtra os dados. Na fragmentação vertical, apesar de serem todos os dados copiados, apenas o tempo de execução é reduzido para quase metade do que quando se replica uma tabela inteira com cinco colunas. Na replicação de uma tabela com dependências os resultados estão dentro do esperado.

Volume Dados	A	B	C	D	E	F	G
5000	106	20	23	165	20	20	39
20000	657	81	81	683	81	79	155
50000	1645	187	183	2014	95	90	418
100000	3556	373	369	3416	677	672	1615

Tabela 5.4: Resultados da integração entre uma BD PostgreSQL (fonte) e uma BD MySQL (destino).

A integração entre uma BD PostgreSQL e uma BD MySQL apresenta alguns resultados fora do normal, nomeadamente quando se trata de fragmentação vertical, onde o tempo de execução é quase sempre superior ao que se verifica durante uma replicação de dados total. Seria esperado que na filtragem de dados o tempo de execução fosse menor. Nos restantes testes realizados não existe nenhum resultado considerado fora do normal.

Volume Dados	A	B	C	D	E	F	G
5000	162	20	19	97	20	19	37
20000	628	71	72	645	75	73	190
50000	1625	171	170	1608	182	179	701
100000	3374	352	356	3148	351	362	1990

Tabela 5.5: Resultados da integração entre duas BDs MySQL.

Entre duas bases de dados MySQL os resultados continuam dentro do normal, à excepção de quando se trata de fragmentação vertical, onde os tempos obtidos continuam a ser bastante idênticos àqueles que se verificam numa cópia total de dados.

Volume Dados	A	B	C	D	E	F	G
5000	121	13	14	65	10	10	26
20000	391	47	48	249	34	33	125
50000	1029	127	121	624	85	83	519
100000	2721	237	233	1500	158	164	1666

Tabela 5.6: Resultados da integração entre uma BD MySQL (fonte) e uma BD PostgreSQL (destino).

Quando se trata da integração entre uma base de dados MySQL e uma base de dados PostgreSQL, onde a base de dados PostgreSQL é a responsável por guardar a cópia dos dados, os resultados obtidos retornam ao normal. Isto é, todos os resultados apresentados estão dentro do esperado e além disso não existe problemas com a fragmentação vertical.

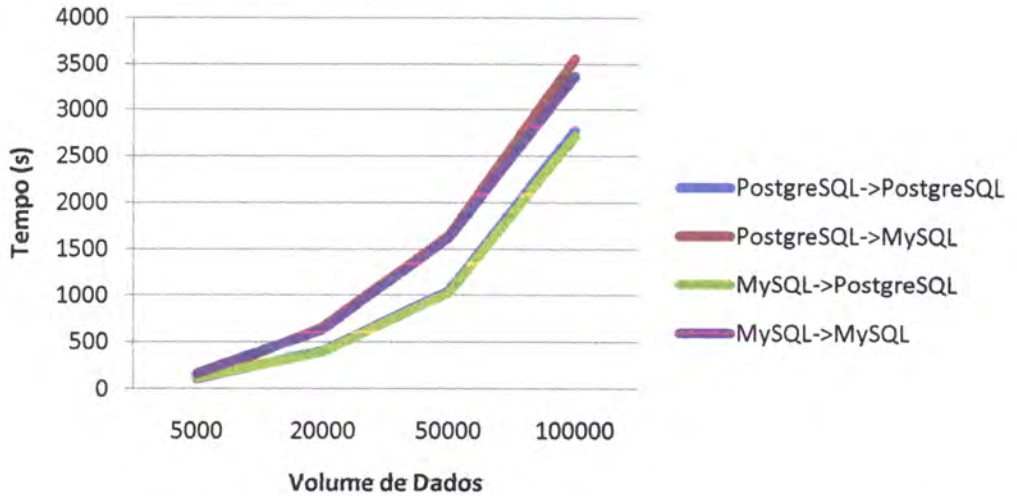

Figura 5.2: Gráfico comparativo dos tempos obtidos numa replicação total.

Através da interpretação das tabelas apresentadas nesta subsecção e com auxílio do gráfico da Figura 5.2, pode-se constatar que a integração de dados onde o SGBD PostgreSQL é o receptor, tem melhor desempenho do que quando o SGBD MySQL é o receptor. Desde o inicio dos testes realizados, que se verificaram sempre diferenças entre os tempos obtidos, onde o SGBD PostgreSQL apresenta sempre melhores resultados, quando é o receptor. Pode-se concluir, que tanto o SGBD e PostgreSQL e MySQL não apresentam diferenças no processamento dos dados antes do envio para os vários destinos. No entanto, o SGBD MySQL apresenta piores tempos em relação ao SGBD PostgreSQL quando se trata de recepcionar dados.

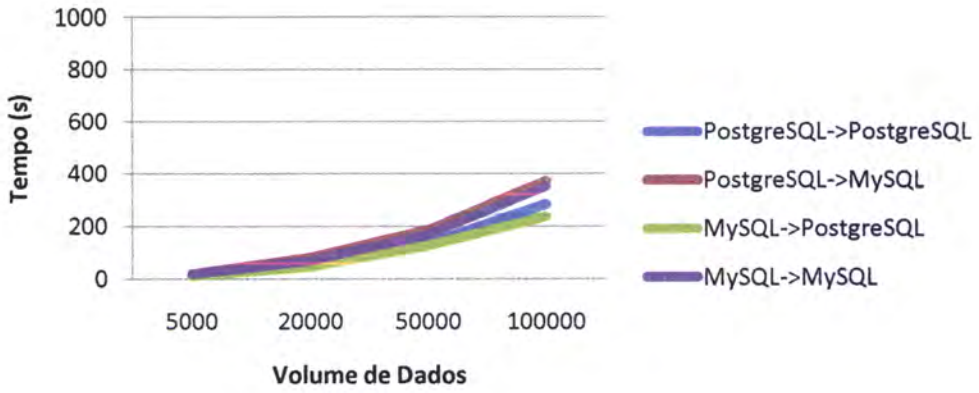

Figura 5.3: Gráfico comparativo dos tempos obtidos numa replicação com fragmentação horizontal e filtragem na fonte.

Como seria de esperar, na fragmentação horizontal os tempos obtidos são menores em comparação com uma replicação total, pois os dados são filtrados diminuindo assim o volume de dados. Nesta situação não existem grandes diferenças de desempenho entre o SGBD PostgreSQL e MySQL. Através da análise das tabelas apresentadas, pode-se também constar que não existem grandes diferenças na distinção de quem filtra os dados, ou seja, os tempos são idênticos quando a fonte ou o destino filtram os dados.

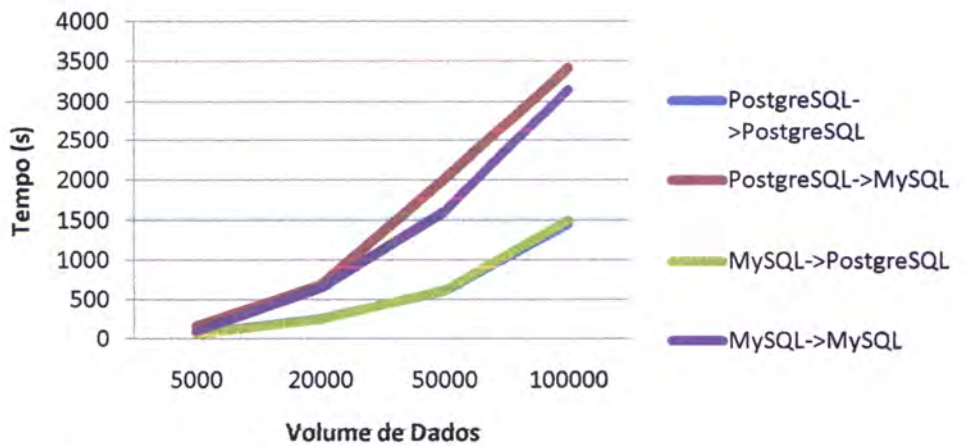

Figura 5.4: Gráfico comparativo dos tempos obtidos numa replicação com fragmentação vertical.

Também na fragmentação vertical é possível verificar a redução de tempo do processamento dos dados, apesar do volume de dados ser igual ao de uma replicação total, sendo que a diferença está no número menor de colunas. O SGBD MySQL mostra novamente problemas e quando trata de filtrar os dados de modo a obter apenas três colunas das cinco enviadas, demora quase sempre o mesmo tempo do que uma replicação total correspondente.

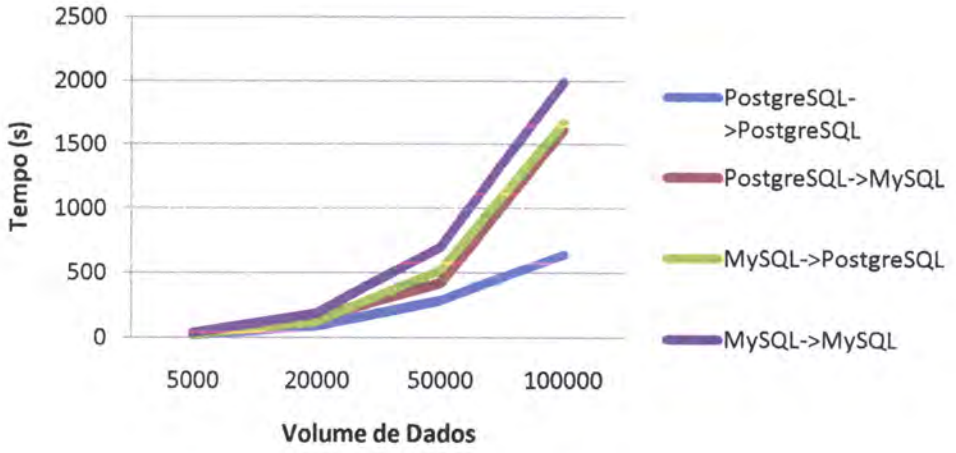

Figura 5.5: Gráfico comparativo dos tempos obtidos numa replicação de uma tabela mais as respectivas dependências.

Quando se trata da replicação de uma tabela mais as respectivas dependências de outra tabela, o SGBD PostgreSQL continua a ter melhor desempenho do que o SGBD MySQL. Existe a particularidade na situação em que o SGBD MySQL envia dados para o SGBD PostgreSQL, onde se consta que o tempo de execução é maior do quando se envia entre duas bases de dados PostgreSQL e entre uma base de dados PostgreSQL e outra MySQL. Na outra situação em que o SGBD PostgreSQL recebe dados, o tempo de execução é bastante menor, quer isto dizer que o SGBD MySQL é mais lento quando tem de extrair informação da tabela principal mais as respectivas dependências.

Capítulo 6

Conclusões e Trabalho Futuro

Esta dissertação teve como tema principal a replicação de bases de dados, onde foram descritos alguns conceitos chave, bem como a aplicação para a replicação de dados que foi desenvolvida. Neste capítulo é analisado o trabalho realizado e são descritas algumas melhorias a serem implementadas para o correcto funcionamento da aplicação. No seguinte ponto 6.1 são apresentadas as conclusões após a realização deste projecto e em 6.2 é abordado o trabalho futuro a ser implementado neste projecto.

6.1 - Conclusões

Ao longo desta dissertação foram apresentados conceitos, ferramentas e metodologias que constituem a aplicação desenvolvida, com o objectivo de replicar bases de dados distribuídas. Nesta subsecção são descritos os objectivos alcançados e também as conclusões obtidas.

O objectivo principal, a replicação de dados entre vários sistemas, foi alcançado, sendo possível a troca de informação entre qualquer sistema. A adaptabilidade foi um dos aspectos a ter em conta, ou seja, a aplicação desenvolvida pode adaptar-se a várias situações possíveis quando se pretende replicar dados. O utilizador poderá optar por obter todos os dados de uma só vez, incrementalmente ou parcialmente, consoante a situação que mais desejar. Pode ainda escolher como propagar os dados, de forma a controlar a carga de um sistema e permitir replicar dados entre tabelas com esquemas diferentes permitindo a realização de transformações de modo a ser possível uma adaptação à tabela que recebe os dados. Na situação em que uma tabela tem dependências, ou seja, tem informação proveniente de outras tabelas é possível replicar juntamente a tabela principal mais as respectivas dependências, garantindo a integridade dos dados.

Além da replicação de dados propriamente dita, foram implementadas mais soluções para facilitar a integração de informação entre os vários sistemas. As tabelas de histórico, desenvolvidas neste projecto têm um papel fundamental, pois é através delas que é possível a verificação de actualizações numa determinada tabela. Desta forma, a aplicação sabe qual os dados que deve processar para posteriormente serem enviados para outros destinos. Além do que já foi descrito, as tabelas de histórico podem ser utilizadas por outras aplicações que pretendam aceder ao histórico de uma tabela.

O *Web Service* apresentado neste projecto possibilita a integração da aplicação desenvolvida com outras aplicações que possam surgir, desde que seja utilizado o contrato WSDL desenvolvido. Além disso, a aplicação desenvolvida não depende da plataforma em uso, ou seja, poderá ser executada em qualquer plataforma, desde que exista uma JVM e um servidor de aplicações instalados.

Outro dos objectivos propostos que foi cumprido é a integração de dados heterogéneos, sendo possível replicar dados entre diferentes SGBDs. No caso deste projecto apenas foram usados os SGBDs MySQL e PostgreSQL, no entanto, existe a possibilidade de integrar mais SGBDs com a aplicação desenvolvida, complementando ainda mais a aplicação.

Ainda abordando a adaptabilidade da aplicação, existe a possibilidade de enviar dados ou pedir dados, isto é, o sistema pode enviar dados para vários destinos sem que ninguém faça um pedido ou um destinatário efectuar um pedido dos dados que pretende. Desta forma o utilizador da aplicação pode adaptá-la à situação que lhe for mais conveniente.

Em determinadas situações pode ser necessário efectuar uma filtragem de dados, de forma a não serem copiados dados desnecessários, aumentando assim o rendimento de um sistema. A fragmentação horizontal e vertical é de certa forma uma solução para filtrar os dados, sendo possível a selecção específica de determinadas colunas ou de um determinado domínio de dados de uma tabela. Como ficou provado no ponto 5.5, quando é aplicada fragmentação horizontal ou vertical num determinado volume de dados, o tempo de execução de uma réplica é bastante menor em comparação com uma cópia total de dados sem qualquer filtragem.

O módulo de monitorização disponível na aplicação tem um papel importante durante a execução da aplicação, informando o utilizador quando são efectuadas réplicas e também quando ocorre um erro. Desta forma, o utilizador está bem auxiliado quando precisa de tomar uma decisão, de forma a melhorar o desempenho do sistema ou então na ocorrência de um erro.

Após a realização dos testes na aplicação, pode-se constatar que ao efectuar uma cópia total dos dados o tempo de execução é cada vez maior à medida que o volume de dados aumenta, chegando a um ponto em que poderá ser insustentável para a aplicação replicar um elevado volume de dados. Assim e através dos resultados obtidos, a utilização desta aplicação deverá focar mais na replicação incremental ou parcial de dados, evitando a replicação total de dados de elevado volume.

Ainda em relação aos testes realizados, na comparação entre os tempos obtidos entre as filtragens na fonte ou destino dos dados, a diferença existente é mínima. No entanto, quantas mais replicações a fonte tiver de suportar, maior é a complexidade e desta forma o rendimento será menor, por isso será aconselhável o destino dos dados filtrar os dados de modo a não

sobrecarregar a fonte. A propagação dos dados em cascata pode também evitar a sobrecarga dos dados na fonte, distribuindo a responsabilidade de replicação pelos vários destinatários.

Em geral os objectivos definidos para este projecto, foram alcançados, apesar de existirem algumas melhorias a serem feitas, de forma a disponibilizar mais opções à aplicação e também melhorar o desempenho em determinadas funcionalidades já disponíveis.

6.2 - Trabalho Futuro

Ao longo do desenvolvimento desta dissertação e na fase de testes foram identificados alguns aspectos a serem implementados ou melhorados, de forma a optimizar a aplicação desenvolvida.

Desde a fase de testes realizada na aplicação houve a necessidade de serem executados testes mais exaustivos e variados, que acabaram por não se concretizar devido à falta de recursos. Para que sejam detectados prováveis erros e ainda melhorar o desempenho da aplicação deveriam ser realizados mais testes não especificamente com um volume de dados variáveis, mas focando-se nas funcionalidades específicas da aplicação, tais como, a integração de dados heterogéneos, fragmentação horizontal e vertical, transformação de tipos, entre outras. No decorrer desta dissertação, apenas foi utilizada a plataforma Windows para a realização de testes, no entanto, devem ser feitos noutras plataformas tais como Linux ou Mac OSx, de forma a poder ser analisada a influência de uma plataforma na sua execução, bem como realizar os respectivos melhoramentos no caso de surgirem erros.

Em comparação com outras aplicações de replicação de bases de dados seria útil implementar a funcionalidade de copiar o esquema das tabelas, ou seja, além de serem copiados os dados de uma tabela o seu esquema também seria copiado. Como a fonte dos dados nem sempre conhece os esquemas de tabelas existentes no destino, no caso de não existir o esquema da tabela pretendida seria útil copiar também o esquema garantindo que os dados seriam correctamente guardados.

Actualmente a aplicação apresentada, replica dados entre máquinas, identificando o receptor apenas como uma só máquina, mas seria benéfico para a aplicação enviar os dados para um grupo de máquinas e não para várias máquinas soltas. Ou seja, a aplicação deveria reconhecer um determinado número de máquinas como uma só e apenas realizar um envio de dados, evitando mais envios de forma a não levar ao sobre carregamento do sistema. Assim realizaria menos envios, ficando disponível para outras funções. O grupo de máquinas ficaria responsável por receber os dados e distribuí-los pelas máquinas constituintes.

Na replicação total de dados com um elevado volume reparou-se que o processamento era muito mais lento e que às vezes existiam problemas de memória nas máquinas onde foram

realizados os testes, pois os dados antes de serem enviados são processados na sua globalidade e depois enviados. Para evitar problemas de memória seria benéfico dividir os dados os dados em parcelas de menor dimensão e envia-las sequencialmente para os vários destinos, evitando assim o maior desgaste na fonte dos dados.

A segurança foi um assunto não abordado nesta dissertação e também não foi implementada na aplicação desenvolvida, no entanto, caso seja necessária a replicação de dados de elevada importância será necessário criar mecanismos de segurança evitando a captação de dados quando estes são replicados para outros destinos. Uma solução possível passa pela encriptação dos dados na fonte e desencriptação no destino correcto.

Referências Bibliográficas

[1] - Ramakrishnan, R. e Gehrke, J. *Database Management Systems*. 2nd Edition. McGraw-Hill

[2] - Ozsu, M. e Valduriez, P. 1999. *Principles of Distributed Database Systems*. 2nd Edition. Prentice-Hall. New Jersey.

[3] - Paul, S. 2009. *Pro SQL Server 2008 Replication*. Springer

[4] - Wiesmann, M. Pedone, F. Schiper, A. Kemme, B. Alonso, G. *Understanding Replication in Databases and Distributed Systems*.

[5] - *Slony-I 2.0.4 Documentation* [Em Linha]. [Acedido: 19 de Abril de 2010]. Disponível em: http://www.slony.info/documentation/index.html.

[6] - *MySQL 5.0 Reference Manual* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://dev.mysql.com/doc/refman/5.0/en/replication.html.

[7] - *Oracle Database 11g: Oracle Streams Replication* [Em Linha]. [Acedido: 19 de Abril de 2010]. Disponível em: http://www.oracle.com/technology/products/dataint/pdf/twp_streams_replication_11gr1.pdf.

[8] - *SQL Server Replication* [Em Linha]. [Acedido: 19 de Abril de 2010]. Disponível em: http://msdn.microsoft.com/en-us/library/ms151198.aspx

[9] - *Daffodil Replicator - Developer's Guide* [Em Linha]. [Acedido: 21 de Abril de 2010]. Disponível em: http://sourceforge.net/projects/daffodilreplica/.

[10] - Tilma, B. 2008. *DBReplicator Manual* [Em Linha]. [Acedido: 21 de Abril de 2010]. Disponível em: http://sourceforge.net/projects/dbreplicator/.

[11] - Wikipédia. *MySQL* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://en.wikipedia.org/wiki/MySQL.

[12] - *PostgreSQL 8.3.11 Documentation* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://www.postgresql.org/docs/8.3/static/index.html.

[13] - Oracle. *Java Technology* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://www.oracle.com/us/technologies/java/index.html.

[14] - Wikipédia. *Java (programming language)* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://en.wikipedia.org/wiki/Java_%28programming_language%29.

[15] - Oracle. *JDBC Overview* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://www.oracle.com/technetwork/java/overview-141217.html.

[16] - *PostgreSQL JDBC Driver* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://jdbc.postgresql.org/.

[17] - *MySQL Connector/J* [Em Linha]. [Acedido: 19 de Agosto de 2010]. Disponível em: http://dev.mysql.com/doc/refman/5.1/en/connector-j.html.

[18] - Wikipédia. *Java API for XML Web Services* [Em Linha]. [Acedido: 20 de Agosto de 2010]. Disponível em: http://en.wikipedia.org/wiki/Java_API_for_XML_Web_Services.

[19] - Docentes de Sistemas Distribuídos, DEI, IST, UTL. *JAX-WS* [Em Linha]. [Acedido: 20 de Agosto de 2010]. Disponível em: http://disciplinas.ist.utl.pt/leic-sod/2009-2010/labs/05-web-services-2/jax-ws/index.html.

[20] - Vasiliev, Y. 2008. *Beginning Database-Driven Aplication Development in Java EE: Using GlassFish.* Springer.

[21] - *GlassFish Overview* [Em Linha]. [Acedido: 20 de Agosto de 2010]. Disponível em: https://glassfish.dev.java.net/faq/v2/GlassFishOverview.pdf.

[22] - Wikipédia. *JavaServer Pages* [Em Linha]. [Acedido: 20 de Agosto de 2010]. Disponível em: http://en.wikipedia.org/wiki/JavaServer_Pages.

[23] - *NetBeans IDE 6.9.1 Release Information* [Em Linha]. [Acedido: 20 de Agosto de 2010]. Disponível em: http://netbeans.org/community/releases/69/.

[24] - Wikipédia. *NetBeans* [Em Linha]. [Acedido: 20 de Agosto de 2010]. Disponível em: http://en.wikipedia.org/wiki/NetBeans.

[25] - Wikipédia. *Web Service* [Em Linha]. [Acedido: 23 de Agosto de 2010]. Disponível em: http://en.wikipedia.org/wiki/Web_service.

[26] – Tungsten Replicator Guide [Em Linha]. [Acedido: 18 Dezembro de 2010]. Disponível em: http://www.continuent.com/images/stories/pdfs/tungsten-replicator-guide.pdf.