Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Curso Bacharelado em Sistemas de Informação

Itamar Egídio de Moura Fé

Data Migration Performed - Uma ferramenta para migração de dados entre SGBDs diferentes

Picos
2014

Itamar Egídio de Moura Fé

Data Migration Performed - Uma ferramenta para migração de dados entre SGBDs diferentes

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharelado, sob orientação do Professor MSc. Arlino Henrique Magalhães de Araújo.

Picos

2014

Eu, **Itamar Egídio de Moura Fé**, abaixo identificado(a) como autor(a), autorizo a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação abaixo discriminada, de minha autoria, em seu site, em formato PDF, para fins de leitura e/ou impressão, a partir da data de hoje.

Picos-PI 13 de março de 2014.

Itamar Egídio de Moura Fé
Assinatura

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

M929d Moura Fé, Itamar Egídio de.
Data Migration Perfomed: uma ferramenta para migração de dados entre SGBDs diferentes / Itamar Egídio de Moura Fé. - 2013.
CD-ROM : il. ; 4 ¾ pol. (57 p.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí. Picos-PI, 2013.
Orientador(A): Prof. MSc. Arlino Henrique M. de Araújo

1. SGBDs. 2. Migração de Dados 3. Catálogo de Sistema I. Título.

CDD 005.1

Itamar Egídio de Moura Fé

Data Migration Performed - Uma ferramenta para migração de dados entre SGBDs diferentes

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte dos requisitos para obtenção do Grau de Bacharelado, sob orientação do Professor MSc. Arlino Henrique Magalhães de Araújo.

Data de Aprovação: 11/03/2014

Arlino Henrique Magalhães de Araújo Arlino Henrique Magalhães de Araújo UFPI - CSHNB
Dennis Sávio Martins da Silva Dennis Sávio M. da Silva UFPI - CSHNB
Francisco das Chagas Imperes Filho Francisco das Chagas Imperes Filho UFPI - CSHNB

Picos
2014

Dedico esta monografia a toda minha família em especial aos meus pais, Inês de Sousa Moura Fé, Egídio Cirilo de Moura Fé, meus irmãos, Elsa Inês de Sousa Moura Fé Oliveira, Ivan Egídio de Moura Fé e meu cunhado Joserlando José de Oliveira por estarem sempre ao meu lado em todas as minhas conquistas e decisões.

Agradeço primeiramente a Deus, pelo dom da vida, pela minha capacidade de pensar, ter me proporcionado mais uma vitória e por me guiar nessa jornada de conquista.

Agradeço a toda minha família pela paciência, conselhos, ajuda de custo e por acreditar no meu trabalho, mesmo com as dificuldades encontradas, em especial Inês de Sousa Moura Fé, Egídio Cirilo de Moura Fé, Elsa Inês de Sousa Moura Fé Oliveira, Ivan Egídio de Moura Fé, Cirilo (*In Memorian*), Rosa Maria (*In Memorian*), Constâncio (*In Memorian*), Josefa (*In Memorian*), tios, primos e outros.

Ao meu orientador Msc. Arlino Henrique Magalhães de Araújo por ter acreditado e confiado no meu potencial, pela oportunidade de trabalhar na área de Banco de Dados, obrigado pela conclusão desse trabalho.

A todos os professores que compõem o curso de Bacharelado em Sistemas de Informação, que contribuíram através de seus conhecimentos repassados: Arlino Henrique Magalhães de Araújo, Algeir, Dennis Sávio, Érick, Flavio Henrique, Francisca Cosme, Frank César, Fredison Munis, Ismael, Ivanildo, Ivenilton, José Ricardo, Janayna, João Santos, Juliana, Júlio César, Leonardo, Patrícia Medyna, Patricia Vieira, Rayner, Romuere, Ryan e Tadeu.

Aos meus colegas e amigos da Banda Católica Semente de Restauração, Lúcio Sousa, José de Moura Costa Júnior, Francisco Cavalcante Costa, Jackson Cavalcante Costa, Herlindo Barbosa, pelos conselhos e ajuda em todos os momentos.

A minha namorada Jairan Rodrigues pela paciência, amor, atenção, compreensão e companheirismo durante esse trabalho.

Aos meus amigos de infância, em especial Ronildo, Júnior, Deusimar, Willian, Wesley, Valdivan, Laerte, Lismar, Cleber, Leomar, Geraldo, Eduardo, Edimar, Charles, Tiago, Pedro, Welisvania, Cledson, Alielson, Zezim, Fredson, Anderson, Erika, Francimilton, Efim (*In Memorian*), Valnilson (*In Memorian*), Viviane, Rosiane, Rosana, Fernando, flavio, kaline, Joyce, Felipe, Wartin, Walton, Weverton, Jeovane, Igelia, Gilberto, Belinha e outros.

Aos meus amigos da Pastoral da Juventude em especial Andressa Arrais, Júnior Sá, Verônica, Simone, Michele, Carol, Cris, Janaina, Janyele, Brenda, Mimi, João Carlos, Nedinha, Cleinha, Diana, Karla, Barbara, Amanda, Janaina, Ramone, Francisco Júnior, padre Francidilson, padre Adalton, padre Davi Barros, padre Ferdiran e outros.

Aos meus colegas, amigos do curso, da UFPI, em especial a equipe do LIPPO, Allan, Marco Antonio, Wilson, Varton, André, Bruno Rafael, Cliciano, Daniel, Mezzomo, Jonilton, Cavalcante, Ruth, Armando, Abimael, Renan, Klisanderson, Auricélio, Pâmela, Danila, obri-

gado por estarem presente durante toda essa jornada.

Amigos da turma em especial Melksedeck, Rafael Melo, Robson, Jailson, Afonso, Rebelato, Gleyson, Gedilson, Israel, Weder, Thiago José, João Mariano, Leonilio, Waldery, Reginaldo pelos momentos de aprendizagem que compartilhamos no curso.

“Nunca deixe que lhe digam que não vale a pena acreditar nos sonhos que se tem ou que os seus planos nunca vão dar certo ou que você nunca vais ser alguém.”

(Renato Russo)

“O conhecimento torna a alma jovem e diminui a amargura da velhice. Colhe, pois, a sabedoria. Armazena suavidade para o amanhã.”

(Leonardo da Vinci)

“Algumas pessoas acham que foco significa dizer sim para a coisa em que você irá se focar. Mas não é nada disso. Significa dizer não às centenas de outras boas ideias que existem. Você precisa selecionar cuidadosamente.”

(Steve Jobs)

Resumo

A migração de dados entre SGBDs é um processo em que capturam-se os dados de um SGBD origem, e em seguida copia-se os dados capturados para o SGBD destino. Para que seja realizada a migração é necessário obtermos informações do banco de dados como, por exemplo, esquemas, tabelas, colunas. Migrar os dados de forma manual pode acarretar em sucessivas falhas humanas além de um grande desperdício de tempo. Com isso, surge a necessidade de poder migrar automaticamente a base de dados de um SGBD existente para outros SGBDs na intenção de aproveitar os dados, sem haver a necessidade de ter que passar pelo processo de criação manual novamente. Uma das formas de ter acesso às informações sobre os dados do SGBD se dá através do uso do catálogo de sistema do SGBD, que é um componente do SGBD composto na sua maioria de tabelas onde são armazenadas todas as informações sobre os dados armazenados no SGBD. Esse trabalho tem como objetivo o desenvolvimento de uma ferramenta para migração automática de dados entre SGBDs diferentes utilizando o catálogo do sistema.
Palavras-chave: *SGBD*. *Migração de Dados*. Catálogo de Sistema.

Abstract

Data migration between DBMSs is a process that captures the data source DBMS, and then copies the data captured for the target DBMS. For migration to be performed it is necessary to obtain information from the database, for example, schemas , tables, columns. Migrate the data manually may result in successive human error plus a big waste of time . With this comes the need to be able to automatically migrate the database from an existing DBMS to other DBMSs intending to leverage data, without the need of having to go through the manual creation process again. One way to have access to information on the DBMS data is through the use of the DBMS system catalog, which is a component of the DBMS consists mostly of tables which stores all the information about the data stored in the DBMS . This work aims to develop a tool for automatic migration of data between different DBMS using the system catalog.

Keywords: DBMS. Data Migration. System Catalog.

Lista de Figuras

Figura 1 - *Estrutura do catálogo de sistema do PostgreSQL.* 21

Figura 2 - *Parte da estrutura das visões do grupo Information Schema.* 24

Figura 3 - *Exemplo da estrutura do Padrão de Projeto Strategy.* 28

Figura 4 - *Exemplo da Estrutura do padrão de projeto Chain of Responsability.* . . 29

Figura 5 - *Modelo Dimensional do TPC-H.* 30

Figura 6 - *Modelo Relacional do TCP-H.* 31

Figura 7 - *Estrutura da Driver For Catalog Obtainment.* 34

Figura 8 - *Consulta para obter os esquemas do banco de dados no SGBD PostgreSQL.* 35

Figura 9 - *Consulta para obter as tabelas de um determinado esquema no SGBD PostgreSQL.* 35

Figura 10 - *Consulta para obter as colunas de uma determinada tabela de um determinado esquema no SGBD PostgreSQL.* 35

Figura 11 - *Consulta para obter a chave primaria de uma determinada tabela de um determinado esquema no SGBD PostgreSQL.* 36

Figura 12 - *Consulta para obter a chave estrangeira de uma determinada tabela de um determinado esquema no SGBD PostgreSQL.* 36

Figura 13 - *Consulta para obter os esquemas do banco de dados no SGBD SQL Server.* 37

Figura 14 - *Consulta para obter as tabelas de um determinado esquema no SGBD SQL Server.* 37

Figura 15 - *Consulta para obter as colunas de uma determinada tabela de um determinado esquema no SGBD SQL Server.* 37

Figura 16 - *Consulta para obter a chave primaria de uma determinada tabela de um determinado esquema no SGBD SQL Server.* 38

Figura 17 - *Consulta para obter a chave estrangeira de uma determinada tabela de um determinado esquema no SGBD SQL Server.* 38

Figura 18 - *Estrutura da ferramenta para migração de dados entre SGBDs diferentes.* 41

Figura 19 - *Funcionamento da Ferramenta para migração de dados entre SGBDs diferentes.* 42

Figura 20 - *Diagrama de classes do Data Migration Performed.* 43

Figura 21 - *Ferramenta Data Migration Performed.* 44

Figura 22 - *Tela de conexão com o SGBD origem.* 45

Figura 23 - *Tela de conexão do SGBD destino.* 45

Figura 24 - *Tela com a estrutura (Esquema) dos SGBDs origem e destino* 46

Figura 25 - *Tela com a mensagem informando que a estrutura do SGBD foi migrada com sucesso.* 46

Figura 26 - *Tela com a mensagem informando que os dados foram migrados com sucesso.* 46

Figura 27 - *Tela com a captura de toda a estrutura do banco de dados Benchmark TPC-H, tendo como origem SGBD PostgreSQL e destino o SGBD SQL Server.* 48

Figura 28 - Quantidade de dados na tabela customer no SGBD destino após a migração de dados. 49

Figura 29 - Tela com a captura da estrutura do SGBD Benchmark TCP-H com o SGBD SQL Server como origem do dados e o SGBD PostgreSQL com destino dos dados. 50

Figura 30 - Tabela customer no SGBD PostgreSQL após a migração de dados. . . . 50

Lista de Tabelas

Tabela 1 - Uma instancia da relação Cat_Atributos. 19

Tabela 2 - Tabelas do catálogo de sistema do SGBD PostgreSQL 20

Tabela 3 - Visões do catálogo de sistema do SGBD PostgreSQL. 20

Tabela 4 - Tabelas do catálogo de sistema do SGBD SQL Server. 22

Tabela 5 - Tabelas do catálogo sistema do SGBD SQL Server, pertencentes ao banco de dados Master. 23

Tabela 6 - Visões do catálogo de sistemas do SGBD SQL Server. 23

Tabela 7 - Tipos de dados dos SGBDs PostgreSQL e SQL Server. 26

Tabela 8 - Tipos de dados mapeados do SGBD PostgreSQL para o SQL Server. . . 40

Tabela 9 - Tipos de dados mapeados do SGBD SQL Server para o SGBD PostgreSQL. 40

Tabela 10 - Quantidade e dados de todas as tabelas do banco de dados Benchmark TPC-H. 48

Lista de abreviaturas e siglas

ANSI	American National Standard Institute
BD	Banco de Dados
BSD	Berkeley Software Distribuition
DBMS	Data Base Management System
DDL	Data Definition Language
DML	Data Manipulation Language
DMP	Data Migration Performed
HD	Hard Disk
ISO	International Organization for Stardartion
JDBC	Java DataBase Connectivity
JDK	Java Developed Kit
MCC	Multiversion Concurrency Control
RAM	Random Access Memory
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
XML	eXtensible Markup Language

Sumário

1 Introdução **14**

2 Fundamentação Teórica **16**

2.1 Banco de Dados Relacionais 16

2.1.1 SGBD *PostgreSQL* 16

2.1.2 SGBD *SQL Server* 17

2.2 Catálogo de Sistema 17

2.2.1 Catálogo de Sistema do SGBD *PostgreSQL* 19

2.2.2 Catálogo de Sistema do SGBD *SQL Server* 21

2.3 Migração de Dados 24

2.3.1 Migração de Dados entre SGBDs 25

2.4 Padrões de Projeto 26

2.4.1 Padrão de Projeto *Strategy* 28

2.4.2 Padrão de Projeto *Chain of Responsability* 28

2.5 *Benchmark TPC-H* 29

3 Trabalhos Relacionados **32**

3.1 JExodus: Uma ferramenta para migração de dados independente de SGBD. . . 32

3.2 *Data Migration Wizard*: Um assistente para Migração de Dados no *SQL Server*. 32

4 *Data Migration Performed* (DMP): Uma ferramenta para migração de dados entre SGBDs Diferentes. **33**

4.1 Visão Geral 33

4.2 Consultas ao Catálogo de Sistemas 33

4.2.1 Consultas SQL ao catálogo de sistemas do SGBD *PostgreSQL* 34

4.2.2 Consultas SQL ao catálogo de sistemas do SGBD *SQL Server* 36

4.3 Detalhes de Implementação 38

5 A Ferramenta *Data Migration Performed* (DMP) 44

6 Experimento 47

6.1 Ambiente de Execução 47

6.2 Cenários de Teste 47

6.2.1 Cenário 1: Migração de dados *PostgreSQL* para *SQL Server* utilizando a base de dados *Benchmark* TPC-H 47

6.2.2 Cenário 2: Migração de dados do *SQL Server* para o *PostgreSQL* utilizando a base de dados *Benchmark* TPC-H. 49

7 Trabalhos Futuros 51

8 Conclusão 52

Referências 53

1 Introdução

Um Sistema Gerenciador de Banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é um sistema de software de uso geral que facilita o processo de definição, construção, manipulação e compartilhamento de bancos de dados entre diversos usuários e aplicações (ELMASRI; NAVATHE, 2005). Um SGBD é um conjunto de programas responsáveis pelo gerenciamento de uma base de dados, com o objetivo principal de tirar da aplicação cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados.

Muitas vezes é preciso migrar os dados de um SGBD para outro na busca por melhores desempenhos, melhor tratamento de informações, tempo de resposta às consultas SQL executadas, dentre outros benefícios. Pois cada SGBD manipula e tratam seus dados de maneira diferente e particular.

A migração de dados é um processo de transferir dados de um sistema para outro. Segundo Broy (2005) as migrações de dados podem ter dois perfis: a transposição de arquivos ou objetos digitais para uma nova tecnologia entre servidores, e a reutilização dos dados ou atualização de uma estrutura de um sistema legado em um SGBD.

Migrar os dados de um banco de dados para outro de maneira manual pode acarretar em sucessivas falhas humanas além de haver um grande desperdício de tempo, e em se tratando de SGBDs com grande quantidade de dados armazenados, se torna ainda mais evidente o problema de uma migração realizada manualmente. Além disso, existe o problema de compatibilidade entre os tipos de dados dos SGBDs, o que torna ainda mais difícil e árduo o processo de migração.

Com isso, surgi à necessidade de automatizar a migração de dados por meio de uma ferramenta que possa realizar esse trabalho, garantindo ao usuário maior facilidade e praticidade quando desejar que a sua base de dados seja armazenada em outro SGBD. Basicamente o processo de migração de dados consiste nos seguintes passos:

1. Capturar do esquema do banco de dados de um SGBD origem;
2. Criar esse esquema capturado em um SGBD destino;
3. Copiar os dados do SGBD origem para o SGBD destino;

Os SGBDs são formados por esquemas, tabelas, colunas, restrições e relacionamentos existentes, visões, *triggers*, índices, entre outros. Essas informações são armazenadas em um componente do SGBD chamado de catálogo. O catálogo de sistema é o local onde o SGBD armazena todas as informações sobre os esquemas do banco de dados formando então um conjunto de informações sobre as tabelas, colunas, índices, além de informações sobre as tarefas e funções executadas pelo SGBD.

O catálogo, na maioria dos SGBDs, é composto por visões e tabelas de sistema onde são armazenadas e organizadas todas as informações do SGBD. Essas informações contidas no catálogo são chamadas de metadados, ou seja, uma informação a mais sobre os dados armazenados no SGBD. Para ter acesso os metadados é necessário a realização de consultas SQL ao catálogo do SGBD. Porém, a forma de acesso ao catálogo depende do fabricante. Por exemplo, a maneira como é realizada as consultas aos dados do catalogo do SGBD *PostgreSQL* é diferente da forma como são realizadas no SGBD *SQL Server* 2008.

2 Fundamentação Teórica

Este capítulo apresenta a definição sobre os bancos de dados relacionais, o catálogo de sistema dos bancos de dados e a apresentação conceitual da migração de dados, conceitos esses que são a base de todo o desenvolvimento do presente trabalho.

2.1 Banco de Dados Relacionais

Os banco de dados relacionais são baseados no modelo relacional e usam um conjunto de tabelas para representar os dados e as relações entre eles. Eles também incluem uma DML (*Data Manipulation Language*) e uma DDL (*Data Definition Language*) (SILVERSCHATS et al., 2006).

O modelo relacional representa o banco de dados como uma coleção de relações. Cada relação é semelhante a uma tabela de valores ou, até certo ponto, a um arquivo plano de registros. Ele é chamado de arquivo plano pois cada registro tem uma simples estrutura linear ou plana (SILVERSCHATS et al., 2006).

Quando uma relação é considerada uma tabela de valores, cada linha na tabela representa uma coleção de dados relacionais. Uma linha representa um fato que normalmente corresponde a uma entidade ou relacionamento do mundo real. Os nomes da tabela e da coluna são usados para ajudar a interpretar o significado dos valores em cada linha (ELMASRI; NAVATHE, 2005).

2.1.1 SGBD *PostgreSQL*

O *PostgreSQL* é um SGBD relacional de médio e grande porte com algumas implementações e recursos orientados a objetos, é um software de livre de distribuição *open source*, ou seja, possui código-fonte disponível para modificação, sob a licença BSD (*Berkeley Software Distribuition*) (RIBAMAR, 2006).

O *PostgreSQL* é um sistema de banco de dados objeto relacional (SGBDOR), o qual suporta grande parte do padrão SQL-2003 e oferece muitas funcionalidades modernas, tais como: Padrão SQL que é implementado no padrão ANSI SQL (89,92 e 98), além de suportar parte do SQL 2003;

- *Multiversion Concurrency Control* (MCC);
- Tipos de dados definidos pelo usuário;
- *Triggers* (gatilhos) e *Stored Procedures*;
- Funções;
- Estrutura de Índices;
- Linguagem Procedurais.

2.1.2 SGBD *SQL Server*

O *SQL Server* é um sistema de gerenciamento de banco de dados cliente/servidor de alto desempenho com alta integração com o *Windows* NT. Suas características são: integração com os serviços de multithreading [múltiplas linhas], agendamento, Monitor de Desempenho, e log de eventos do *Windows* NT. Um usuário pode se conectar ao *SQL Server* com a mesma senha usada para a rede *Windows* NT (MARTINS, 2002).

Segundo Martins (2002) o *SQL Server* possui uma arquitetura paralela, que executa as funções de banco de dados simultaneamente para diversos usuários e tira proveito de sistemas com múltiplos processadores. Proporciona ainda, o gerenciamento centralizado de todos os servidores através de uma arquitetura de gerenciamento distribuída, com uma interface visual de gerenciamento.

2.2 Catálogo de Sistema

Uma característica fundamental da abordagem de um banco de dados é que o sistema de banco de dados possui não apenas o banco de dados, mas também uma completa definição ou descrição da estrutura desse banco de dados e suas restrições. Essa definição está armazenada no catálogo do SGBD, que contém informações como: A estrutura de cada arquivo, o tipo e o formato de armazenamento de cada item de dado e várias restrições sobre os dados (ELMASRI; NAVATHE, 2005).

Um SGBD relacional mantem as informações sobre cada tabela e índice que contem. As próprias informações descritivas são armazenadas em uma coleção de tabelas especiais chamadas de tabelas de catalogo. Tais informações podem ser descritas como nomes e tamanhos dos arquivos, os nomes e tipos de itens de dados, detalhes de armazenamento de cada arquivo, informações sobre os mapeamentos entre esquemas e restrições, além de muitas outras informações

que usadas frequentemente pelo SGBD (RAMAKRISHNAN; GEHRKE, 2008).

Segundo Ramakrishnan e Gehrke (2008) o catálogo é organizado por tabelas e visões de sistemas onde são incluídas informações a mais sobre os dados armazenados no SGBD. O acesso a essas tabelas e visões do catalogo se dá através de consultas SQL. Com isso, através de consultas ao catalogo de sistema obtemos todas as informações sobre a estrutura e de como está organizado o banco de dados em um SGBD.

A informação armazenada no catálogo é chamada de metadado que descreve a estrutura do banco de dados primário. O catálogo é usado tanto pelo software SGBD como pelos usuários do banco de dados que precisam de informações sobre a estrutura desse banco (MONTEIRO et al., 2007).

De acordo com Monteiro et al. (2007) os metadados são frequentemente descritos como "dados sobre os dados". Ou seja, são informações adicionais que são necessárias para que os dados se tornem uteis. Os metadados são um conjunto de características sobre os dados que não são normalmente incluídas nos dados propriamente ditos.

No trabalho de Monteiro et al. (2007), são descritos ainda os usos mais significativos dos metadados, dentre os principais podemos citar:

- Adicionar contexto e conhecimento sobre os dados acessados pelos usuários.
- Descobrir os tipos de dados, relacionamentos entre os dados, o momento em que um dado foi lido ou atualizado pela última vez.
- Possibilitar a análise do desempenho do SGBD.

Um aspecto interessante de um SGBD relacional é que o catálogo de sistemas é ele próprio uma coleção de tabelas. Por exemplo, poderíamos armazenar informações sobre atributos de tabelas em uma tabela de Cat_Atributos (RAMAKRISHNAN; GEHRKE, 2008):

Cat_Atributos(nome_atrib: String, nome_rel: String, tipo: String, posição: integer).

Supondo que o banco de dados contenha duas tabelas, que são elas:
Marinheiros (id_marin: integer, nome_marin: String, avaliação: integer, idade: real).
Reservas (id_marin: integer, id_barco: integer, dia: date, nome_resp: String).

A Tabela 1 mostra as tuplas da tabela Cat_Atributos que descreve os atributos destas duas tabelas. Observe que, além das tuplas descrevendo Marinheiros e Reservas, outras tuplas (nas primeiras quatro listadas (Tabela 1) descrevem os quatro atributos da própria tabela Cat_Atributos. Estas outras tuplas ilustram uma questão importante: as tabelas de catálogo descrevem todas as tabelas do banco de dados, incluindo as próprias tabelas de catálogo. Quando informações sobre uma tabela são necessárias, elas são obtidas através do catálogo de sistemas (RAMAKRISHNAN; GEHRKE, 2008).

Atrib_name	**Rel_name**	**Tipo**	**Posição**
Nome_atrib	Cat_Atributos	String	1
Nome_rel	Cat_Atributos	String	2
Tipo	Cat_Atributos	String	3
Posição	Cat_Atributos	Integer	4
Id_marin	Marinheiros	Integer	3
Nome_marin	Marinheiros	String	2
Avaliação	Marinheiros	Integer	3
Idade	Marinheiros	Real	4
Id_marin	Reservas	Integer	2
Dia	Reservas	Date	3
Nome_resp	Reservas	String	4

Tabela 1 *– Uma instancia da relação Cat_Atributos.*

2.2.1 Catálogo de Sistema do SGBD *PostgreSQL*

O catálogo do *PostgreSQL* é composto por tabelas de sistema e visões de sistema. A Tabela 2 descreve as tabelas do catálogo do *PostgreSQL*, e a Figura 1 ilustra como são organizadas as tabelas do catálogo de sistema do SGBD *PostgreSQL* (POSTGRESQL, 2013).

Além do catálogo do *PostgreSQL* ser organizado por tabelas, também é composto por visões de sistema (*views*). A partir dessas visões o *PostgreSQL* fornece um acesso conveniente as tabelas do catálogo de sistema mais frequentemente utilizadas. Na tabela 3 é descrito e explicado as visões do catálogo do SGBD *PostgreSQL*.

Nome da tabela do Catálogo	Finalidade
pg_aggregate	Funções de agregação.
pg_am	Métodos de acesso de índice.
pg_amop	Procedimentos de suporte de método de acesso.
pg_amproc	Funções de agregação.
pg_attrdef	Valor padrão das colunas.
pg_attribute	Colunas de tabela ("atributos").
pg_cast	casts (conversões de tipos de dado).
pg_class	Tabelas, índices, sequências, visões ("relações").
pg_constraint	Restrições de verificação, unicidade, chaves
pg_conversion	Informações sobre conversão de codificação.
pg_database	BD que fazem parte deste agrupamento de bancos de dados.
pg_depend	Dependências entre objetos do banco de dados.
pg_description	Descrições ou comentários sobre os objetos do banco de dados.
pg_group	Grupos de usuários do banco de dados.
pg_index	Informações adicionais sobre índices.
pg_inherits	Hierarquia de herança de tabela.
pg_language	Linguagens para escrever funções.
pg_largeobject	Objetos grandes.
pg_listener	Suporte a notificação assíncrona.
pg_namespace	Esquemas.
pg_opclass	Classes de operador de método de acesso de índice.
pg_operator	Operadores.
pg_proc	Funções e procedimentos.
pg_rewrite	Regras de reescrita dos comandos.
pg_shadow	Usuários do banco de dados.
pg_statistic	Estatísticas do planejador.
pg_tablespace	Espaços de tabelas do agrupamento de bancos de dados.
pg_trigger	Tabelas, índices, sequências, visões ("relações").
pg_constraint	Gatilhos.
pg_type	Tipos de dado.

***Tabela 2** – Tabelas do catálogo de sistema do SGBD PostgreSQL*

Nome da Visão	Finalidade
pg_indexes	Índices.
pg_locks	Bloqueios mantidos no momento.
pg_rules	Regras.
pg_settings	Configurações dos parâmetros.
pg_stats	Estatísticas do otimizador.
pg_tables	Tabelas.
pg_user	Usuários do banco de dados.
pg_views	Visões.

***Tabela 3** – Visões do catálogo de sistema do SGBD PostgreSQL.*

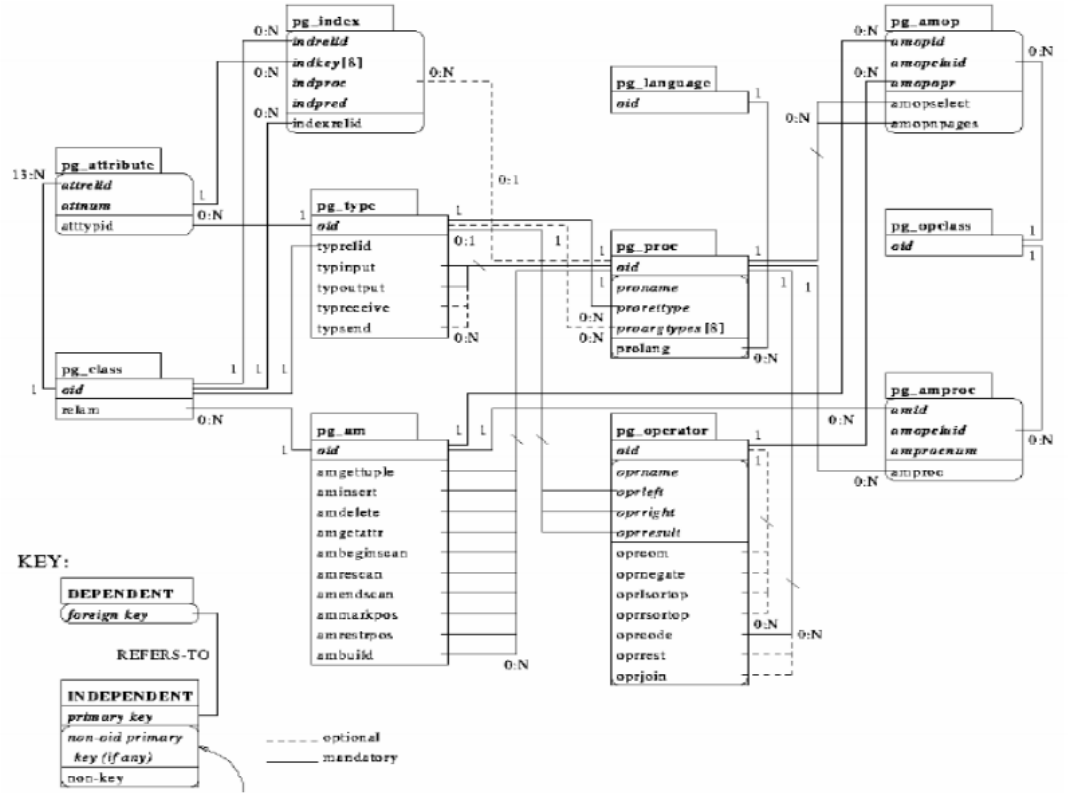

Figura 1 *– Estrutura do catálogo de sistema do PostgreSQL.*

2.2.2 Catálogo de Sistema do SGBD *SQL Server*

Quando criamos uma tabela no *SQL Server*, com suas colunas, índices, tipos de dados e etc., essas informações (metadados) são armazenadas no *Database Catalog*, que é um conjunto de tabelas e visões que armazenam informações sobre os objetos que o usuário criou no banco de dados, como definição de tabela, código fonte de uma *Stored Procedure* ou definição de uma *view* (MONTEIRO et al., 2007). A tabela 4 descreve as tabelas do catálogo do *SQL Server*.

O SGBD *SQL Server* também é composto de tabelas de sistema localizadas apenas no banco de dados Master, que é um banco de dados do próprio *SQL Server*, e que também forma o catálogo do SGBD *SQL Server*. A tabela 5 descreve as tabelas pertencentes ao banco de dados Master:

O catálogo do SGBD *SQL Server*, assim como o catálogo do SGBD *PostgreSQL*, também é composto por visões. Dentre elas, podemos citar o information *Schema Views*, *compatibility views*, *catalog views*, *replication views*, *Dinamic Management and Functios*. Na Figura 2 é apresentado parte da estrutura das visões do grupo *Information Schema*

Nome do Catálogo	Finalidade
syscolumns	Informação sobre cada coluna de cada tabela, e cada parâmetro de procedimento.
syscomments	Para cada objeto de banco de dados (visão, regra, default, trigger, procedimento) contém o texto de sua definição.
sysconstraints	Inclui informações sobre todas as restrições usadas no banco de dados.
sysdepends	Registra as dependências entre objetos do banco de dados.
sysfilegroups	Tem uma linha para cada grupo de arquivos armazenado em um banco de dados.
Sysfiles	Informações sobre cada arquivo de um banco de dados.
sysforeignkeys	Informações sobre todas as restrições de chaves estrangeiras encontradas em todas as tabelas de um banco de dados.
Sysfulltextcatatalogs	Lista todos os catálogos de texto completo para esse banco de dados.
Sysindexes	Informação para cada índice criado e para cada tabela sem índices, além de informações para cada tabela que possui colunas text ou image.
Sysindexkeys	Informação sobreas chaves e as colunas de um índice.
Sysmembers	Informações sobre os membros de cada papel.
Sysobjects	Informação sobre cada objeto do banco de dados (tabelas, visões, procedimentos, regras, defaults e gatilhos).
Syspermissions	Informação sobre permissões atribuídas a usuários, grupos e papéis em um banco de dados.
Sysprotects	Permissões atribuídas à contas de segurança.
Sysreferences	Informação sobre toda restrição de integridade referencial usada numa coluna ou tabela de um banco de dados.
Systypes	Informação sobre cada tipo de dados (do sistema ou definido pelo usuário).
Sysusers	Informação sobre cada usuário que pode ter acesso ao banco de dados.

Tabela 4 *– Tabelas do catálogo de sistema do SGBD SQL Server.*

No trabalho de Monteiro et al. (2007), são descritas as visões do grupo *Information Schemas Views* que baseiam-se nas especificações de catálogo, do padrão SQL-92. É descrito,

Nome do Catálogo	Finalidade
Sysallocations	Informações sobre cada unidade de alocação gerenciada pelo *SQL Server*.
Sysaltfiles	Informações sobre cada arquivo gerenciado pelo SQL Server.
Syscharsets	Informação sobre conjuntos de caracteres [*character sets*] e ordens de classificação [*sort orders*].
sysconfigures, syscurconfigs	Parâmetros de configuração do SQL Server.
Sysdatabases	Informação sobre os bancos de dados existentes.
Sysdevices	Informação sobre os dispositivos, tais como o dispositivo de fita.
Syslanguages	Idiomas suportados pelo servidor.
Syslockinfo	Travas (locks) ativas.
Syslogins	Contas de login.
Sysmessages	Mensagens de erro do sistema.
Sysoledbusers	Contém uma linha para cada usuário e senha mapeados em um servidor.
Sysperfinfo	Informação sobre os monitores de performance.
Sysprocesses	Processos em execução.
Sysremotelogins	Contas de login remotas.
Sysservers	Servidores remotos conhecidos.

Tabela 5 *– Tabelas do catálogo sistema do SGBD SQL Server, pertencentes ao banco de dados Master.*

ainda, que essas visões apresentam as informações do catálogo em um formato independente das tabelas de sistema que armazenam os metadados. Com isso, as visões de sistema são mais simples no que diz respeito à organização dos dados e também a forma como disponibilizam esses dados para o acesso. A tabela 6 descreve as principais visões de sistema do grupo *Information Schemas Views*.

Nome do Catálogo	Finalidade
INFORMATION_SCHEMA.CONSTRAINTS	Obter informações sobre as restrições de verificação que são de instalação em um banco de dados.
INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE	Obter informações sobre os tipos de dados das colunas do banco de dados.
INFORMATION_SCHEMA.COLUMNS	Obter colunas das tabelas.
INFORMATION_SCHEMA.SCHEMATA	Obter informações sobre os esquemas.
INFORMATION_SCHEMA.TABLES	Obter tabelas.

Tabela 6 *– Visões do catálogo de sistemas do SGBD SQL Server.*

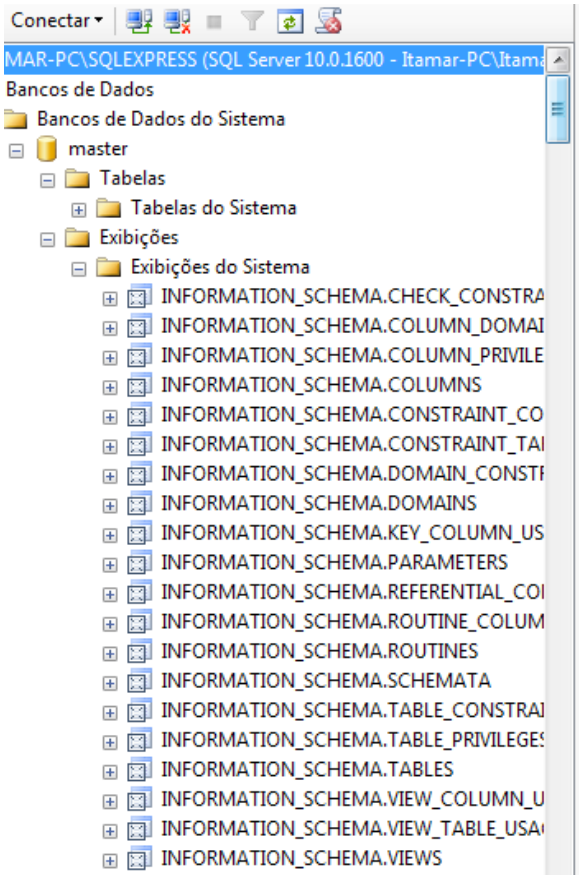

Figura 2 *– Parte da estrutura das visões do grupo Information Schema.*

2.3 Migração de Dados

Frequentemente, os dados que uma nova aplicação necessita vêm de outras aplicações existentes. Caso estes dados estejam disponíveis em sistemas existentes e seu volume seja muito grande, eles devem ser migrados das aplicações existentes para a nova aplicação, ao invés de recriados. O processo de transferir dados de um sistema para outro é chamado de migração de dados (WU; VALTCHEV, 2005).

A migração de dados resumidamente é implementada em duas etapas: a extração e a carga de dados.

A extração de dados é o processo de extrair dados de um sistema existente e armazená-lo em um arquivo. Se o volume dos dados for relativamente pequeno, estes podem ser extraídos para um arquivo que será referenciado diretamente pela aplicação destino. No entanto, se o volume dos dados for grande e os sistemas utilizam diferentes ambientes computacionais, estes devem ser armazenados em alguma mídia, e então, carregados na aplicação destino (WU; VALTCHEV, 2005).

Segundo Wu e Valtchev (2005) a carga de dados é o processo de transferir o conteúdo do arquivo gerado na etapa de extração para a base de dados da aplicação destino. Se os domínios

dos dois sistemas possuem uma interpretação comum entre valores e os relacionamentos entre os dados são bem definidos, então o processo de mapeamento é relativamente direto. Caso os domínios sejam diferentes ou os relacionamentos não estejam bem definidos, é necessário passar por um processo de transformação dos dados, que pode ocorrer na etapa de extração ou de carga.

2.3.1 Migração de Dados entre SGBDs

Para realizar a migração de dados entre SGBDs, é necessário capturar o esquema de um SGBD origem, em seguida criar o esquema capturado no SGBD destino e copiar os dados do SGBD origem para o SGBD destino, de forma automática. Esse processo evita o esforço manual e o tempo que é necessário para a migração de dados.

O esquema de banco de dados é formado por tabelas, colunas, chaves e restrições. Com isso, para que o processo de migração seja concebido corretamente, é necessário que essas informações sejam capturadas no SGBD de origem escolhido e em seguida o mesmo esquema seja criado com as mesmas definições no SGBD destino escolhido. Porém, para que o esquema do SGBD, ou seja, suas tabelas, colunas, chaves primaria e estrangeira possam ser migrados, é indispensável que seja feito um mapeamento da estrutura que será migrada.

O mapeamento em banco de dados consiste no processo de verificação de todos os tipos de dados do SGBD. É utilizado principalmente durante o processo de migração de dados entre SGBDs com estruturas diferentes pois os dados de cada campo de uma tabela podem variar em tamanho (precisão binária) ou formato. Embora existam padrões de tipos de dados no SQL ANSI/ISSO, os desenvolvedores de cada SGBD criaram diversos objetos de apoio ao gerenciamento de estrutura e dados, com isso são gerados novos formatos independentes em cada SGBD.

Os SGBDs diferem quanto à forma em que armazenam os seus dados. Apesar de conter tipos de dados semelhantes, também são constituídos de tipos de dados particulares. Por exemplo, o SGBD *PostgreSQL* possui tipos de dados que não são encontrados no SGBD *SQL Server* ou que não representam o mesmo tipo de dados como é o caso do *NUMERIC*, que no SGBD *PostgreSQL* é relacionado ao tipo de dado da classe decimal, ou seja, tamanho variante. Já no SGBD *SQL Server* o tipo de dado *NUMERIC* está na classe de dados exatos, ou seja, seu tamanho é fixo não podendo então variar. A Tabela 7 descreve os principais tipos de dados dos SGBDs *PostgreSQL* e *SQL Server*.

Portanto, quando no processo de migração a base de dados origem e destino são diferentes é necessário haver uma comparação e conversão de dados para os tipos apropriados para cada SGBD. Com isso as etapas do processo de migração se tornam completas, no que diz respeitos aos dados, considerando que o tratamento dos tipos de dados que serão migrados

Tipos de dados	PostgreSQL	SQL Server
INTEIRO	SMALLINT, INTEGER, BIGINT	TINYINT, SMALLINT, INT, BIGINT
REAL	REAL,DOUBLE PRECISION	REAL,FLOAT
DECIMAL	DECIMAL,NUMERIC	DECIMAL, SMALLMONEY, MONEY
TEXTO	CHAR,VARVHAR,TEXT	NVARCHAR, VARCHAR, NCHAR, CHAR, TEXT, NTEXT
BINÁRIO	BYTEA	BYTE
DATA/TEMPO	DATE, TIME, TIMESTAMP	SMALLDATETIME, DATETIME
LÓGICO	BOOLEAN	BOOLEAN

***Tabela 7** – Tipos de dados dos SGBDs PostgreSQL e SQL Server.*

é uma das partes mais importantes no processo de migração de dados entre bancos de dados diferentes.

2.4 Padrões de Projeto

Um padrão descreve um conjunto composto por um contexto, um problema e uma solução. Em outras palavras, pode-se descrever um padrão como solução para um determinado problema em um contexto. Porém um padrão não descreve qualquer solução, mas uma solução que já tenha sido utilizada com sucesso em mais de um contexto. Exatamente por esse motivo que a descrição dos padrões normalmente sempre descreve alguns de seus usos conhecidos. Um padrão não descreve soluções novas, mais soluções consolidadas (GUERRA, 2012).

De acordo com Guerra (2012) para ser um padrão, uma solução não basta ser recorrente, mais precisa ser uma boa solução. Além disso é comum que ele traga outras partes, como a estrutura da solução, como funciona sua dinâmica e as consequências positivas e negativas de sua aplicação. Outra parte importante é o relacionamento com os outros padrões, pois mostra outros padrões que oferecem uma outra alternativa, ou padrões que podem complementar essa solução para compensar suas desvantagens.

Ainda segundo Guerra (2012) padrões não refletem em pedaços de código ou componentes que são reutilizados de forma igual em diversas aplicações, eles são um conhecimento que deve estar na cabeça dos desenvolvedores. A partir desse conhecimento, os desenvolvedores devem avaliar o contexto e o problema que querem resolver e adaptar e combinar padrões de forma a resolver o problema equilibrando as forças envolvidas. É a partir desse conhecimento que os padrões ajudam os desenvolvedores a desenvolver sua habilidade em modelagem orientada a objetos.

Alexander (2007) estabelece que um padrão deve ter as seguintes características:

- **Encapsulamento**: um padrão encapsula um problema ou solução bem definida. Ele deve ser independente, específico e formulado de maneira a ficar claro onde ele se aplica.
- **Generalidade**: todo padrão deve permitir a construção de outras realizações a partir deste padrão.
- **Equilíbrio**: quando um padrão é utilizado em uma aplicação, o equilíbrio dá a razão, relacionada com cada uma das restrições envolvidas, para cada passo do projeto. Uma análise racional que envolva uma abstração de dados empíricos, uma observação da aplicação de padrões em artefatos tradicionais, uma série convincente de exemplos e uma análise de soluções ruins ou fracassadas pode ser a forma de encontrar este equilíbrio.
- **Abstração**: os padrões representam abstrações da experiência empírica ou do conhecimento cotidiano.
- **Abertura**: um padrão deve permitir a sua extensão para níveis mais baixos de detalhe.
- **Combinatoriedade**: os padrões são relacionados hierarquicamente. Padrões de alto nível podem ser compostos ou relacionados com padrões que endereçam problemas de nível mais baixo.

Além das características de um padrão de projeto, Alexander (2007) define também o formato que a descrição de um padrão deve ter, essas descrições são divididas em cinco partes:

- **Nome**: uma descrição da solução, mais do que do problema ou do contexto.
- **Exemplo**: uma ou mais figuras, diagramas ou descrições que ilustrem um protótipo de aplicação.
- **Contexto**: a descrição das situações sob as quais o padrão se aplica.
- **Problema**: uma descrição das forças e restrições envolvidos e como elas interagem.
- **Solução**: relacionamentos estáticos e regras dinâmicas descrevendo como construir artefatos de acordo com o padrão, frequentemente citando variações e formas de ajustar a solução segundo as circunstâncias. Inclui referências a outras soluções e o relacionamento com outros padrões de nível mais baixo ou mais alto.

2.4.1 Padrão de Projeto *Strategy*

O *Strategy* é um padrão que deve ser utilizado quando uma classe possuir diversos algoritmos que possam ser utilizados de forma intercambiável. A solução proposta pelo padrão consiste em delegar a execução do algoritmo para uma instancia que compõe a classe principal. Dessa forma, quando a funcionalidade for invocada, no momento da execução do algoritmo, será invocado um método da instancia que a compõe (WIKIPEDIA, 2014b).

Uma das partes principais de um padrão diz respeito às suas consequências, pois é a partir delas que o desenvolvedor vai avaliar se essa é uma boa escolha ou não para seus requisitos. No caso do *Strategy*, a principal consequência positiva é justamente o fato do algoritmo poder ser alterado sem a modificação da classe. A partir dessa estrutura, novas implementações do algoritmo podem ser criadas e introduzidas posteriormente (WIKIPEDIA, 2014b). A figura 3 apresenta a estrutura do padrão de projeto *Strategy*.

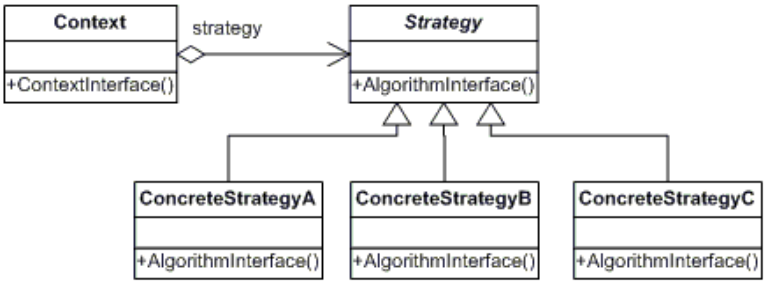

Figura 3 *– Exemplo da estrutura do Padrão de Projeto Strategy.*

2.4.2 Padrão de Projeto *Chain of Responsability*

Quando em um software uma funcionalidade é executada, normalmente existem diversos passos que precisam ser executados em sequência. A dificuldade ocorre quando precisa-se de flexibilidade na configuração desses passos. Por exemplo, pode ser necessária a modificação da ordem dos passos e até mesmo para a inserção de novos passos. Um outro objetivo seria a possibilidade de reutilização desses passo sem outros processamentos ou para outras aplicações (WIKIPEDIA, 2014a).

Chain of Responsibility é um padrão de projeto que cria uma cadeia de execução onde cada elemento processa as informações e em seguida delega a execução ao próximo da sequência. Em sua implementação tradicional, os elementos são percorridos até que um deles façam o tratamento da requisição, encerrando a execução depois disso. Como alternativa, também é possível criar uma cadeia de execução onde cada um executa sua funcionalidade até que a cadeia termine ou ela seja explicitamente finalizada por um dos elementos (WIKIPEDIA, 2014a). Com esse padrão cria-se uma cadeia de objetos que examinam sequencialmente uma solici-

tação. Após examinar a solicitação, cada objeto pode processa-la ou passa-la adiante para o próximo objeto na cadeia. A figura 4 monstra como é formada a estrutura do padrão de projeto *Chain of Responsability*.

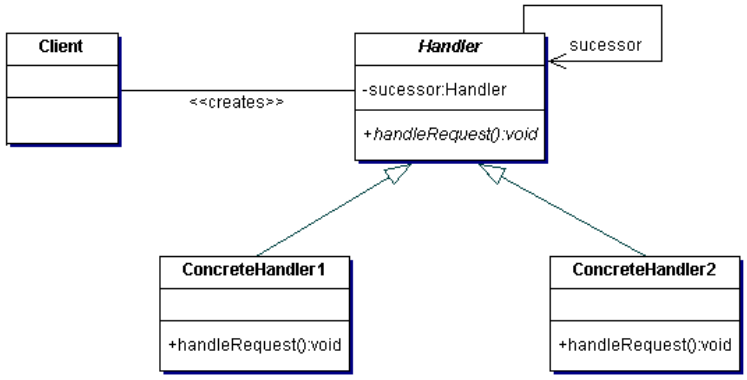

Figura 4 *– Exemplo da Estrutura do padrão de projeto Chain of Responsability.*

2.5 *Benchmark TPC-H*

O TPC-H é um *benchmarck* de suporte a decisões que consiste em um conjunto de consultas ad-hoc voltadas para os negócios e modificações de dados simultâneos. Tem por finalidade simular e avaliar o desempenho de um ambiente de *Data Warehouse*. *Data Warehouse* é um grande repositório de dados coletados de diversas fontes que se destina a gerar informações para o nível gerencial sendo fonte para tomadas de decisão.

Os testes do método TPC-H são realizados sob uma estrutura padrão composta por oito tabelas. Destas, seis tabelas dimensionais (*Region*, *Nation*, *Supplier*, *Part*, *Customer* e *Partsupp*) e duas de fatos (*Orders* e *Lineitem*).

As tabelas de fato estão no centro do modelo dimensional (modelo para suporte à decisão), seus valores são usados para medir o desempenho do negócio. As tabelas de dimensão são áreas ou focos de negócio e fornecem um método geral de organizar a informação corporativa provendo múltiplas perspectivas dos dados. As tabelas de fato armazenam os fatos ocorridos e as chaves características correspondentes nas tabelas dimensionais. as consultas ocorrem inicialmente nas tabelas de dimensão e depois nas tabelas de fatos, assegurando a precisão dos dados.

A Figura 5 mostra o modelo dimensional do benchmark TPC-H. já a Figura A.2 ilustra o modelo relacional do benchmark TPC-H.

Algumas observações devem ser ressaltadas, uma vez analisada a Figura A.2 (MORELLI, 2006):

- O número que aparece logo abaixo do nome da tabela representa sua cardinalidade. Esta pode ser fixa (tabelas *region* e *nation*), ou variável, onde multiplica-se uma constante por

um scale fator determinado no momento da criação da base. Por exemplo, caso SF valha 3, haverá 450.000 tuplas na tabela de clientes (*customer*).

- Estão apresentadas cinco regiões (continentes), que congregam vinte e cinco nações (tabelas *region* e *nation*, respectivamente). Clientes e fornecedores (tabelas *supplier* e *customer*) estão associadas às nações. Enquanto os primeiros realizam pedidos de compras (tabela orders), os segundos fornecem componentes (tabela part) de itens de compra. Como um fornecedor pode oferecer vários itens e um item pode ser disponibilizado por vários fornecedores, existe uma tabela para registrar esta relação N x N (*partsupp*). Finalmente, a tabela mais volumosa do modelo (*lineitem*) associa itens de compra a pedidos.

- Os parênteses ao lado dos nomes das tabelas indicam o prefixo utilizado para denominar os campos da tabela em questão. Desta forma, a chave primaria da tabela de fornecedores chama-se S_SUPPKEY.

- As flechas indicam as associações entre chaves primarias e estrangeiras. Assim, vemos que a chave primaria da tabela nation está vinculada ao campo *nationkey* na tabela de fornecedores (tabela *supplier*).

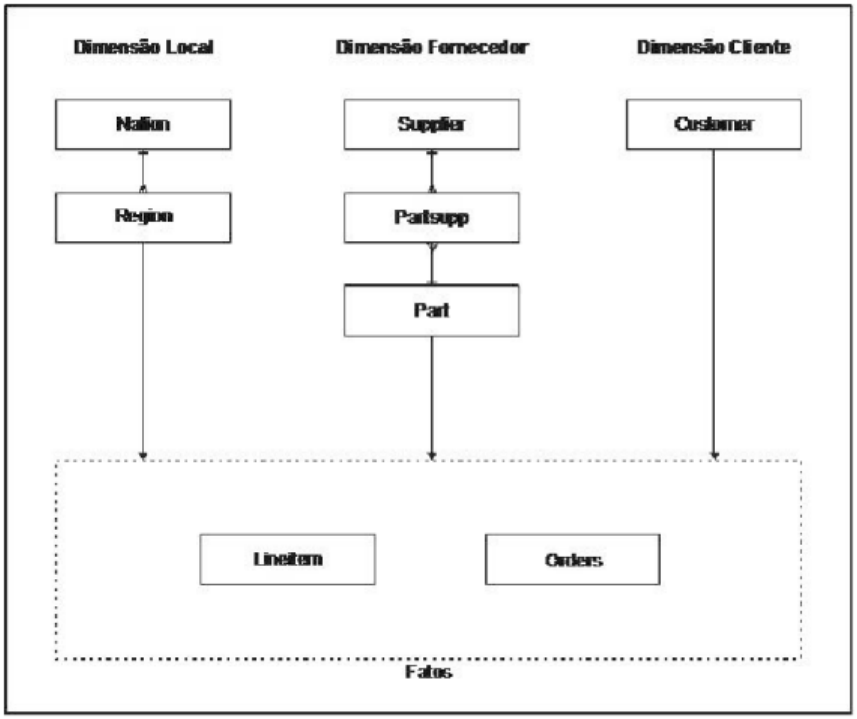

Figura 5 *– Modelo Dimensional do TPC-H.*

O tamanho total da base de dados depende do fator de escalabilidade (SF – *Scale factor*). Para SF=1, a base completa ocupa aproximadamente 1 GB e os volumes de cada tabela são os

que aparecem na Figura A.2. Já para SF=30, a base de dados ocupará 30 GB e a tabela de itens de pedidos de compra possuirá cento e oitenta milhões de tuplas (30 x 6 milhões) (MORELLI, 2006).

De acordo com Morelli (2006) a base de dados sofre acessos de um conjunto de consultas possuindo características *ad-hoc*, ou seja, não se conhece nem a ordem de execução, nem os parâmetros de cada uma das 22 consultas (leitura) e duas funções (uma insere dados e a outra elimina).

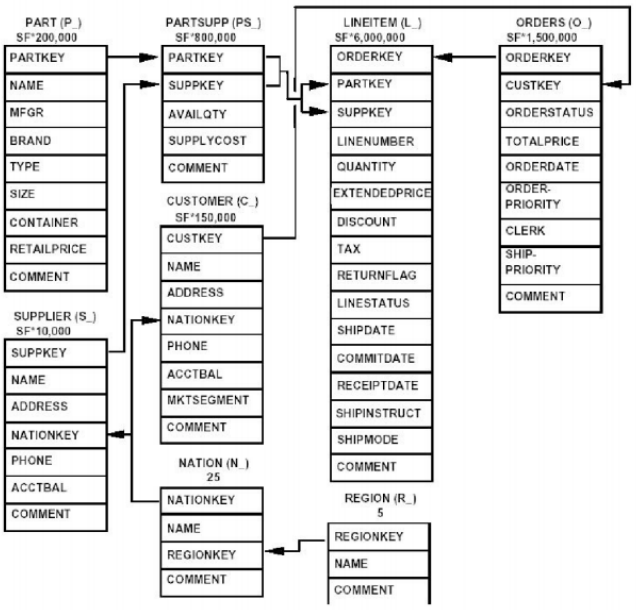

Figura 6 *– Modelo Relacional do TCP-H.*

3 Trabalhos Relacionados

Nesta seção são apresentados alguns trabalhos relacionados que tratam da migração de dados entre SGBDs diferentes.

3.1 JExodus: Uma ferramenta para migração de dados independente de SGBD.

Essa ferramenta permite que atributos e tabelas do esquema do banco de dados antigo possam ser mapeados para o novo esquema de acordo com os requisitos da nova modelagem, além disso, os valores de cada atributo de uma tabela podem ser corrigidos ou transformados ao mesmo tempo em que a migração é executada (NETO; PASSOS, 2006).

Para realizar a migração, essa ferramenta utiliza o framework Hibernate e a API JDBC. Os dados são informados via XML. Com isso, o *JExodus* monta um SQL (*select*, *insert* e *update*) baseado no dialeto do banco utilizado pelo *Hibernate*. Quando a conexão entre as bases é estabelecida, inicia-se a migração.

3.2 *Data Migration Wizard*: Um assistente para Migração de Dados no *SQL Server*.

Ferramenta que permite migrar os dados do banco de dados entre bases de dados no SGBD *SQL Server*, com um foco em sistemas legados (BARRETO, 2008).

Segundo o Barreto (2008), o *Data Migration Wizard* foi concebido para superar os pontos fracos identificados na ferramneta Data *Importer/Explorer* da *Microsoft*.

Algumas características dessa ferramenta são: foi criado para a realização de migração de dados entre bases de dados *SQL Server* de aplicações que sofrem atualizações e suas versões; divide o processo de migração em iterações (identificadas automaticamente), de modo que todos os dados podem ser migrados de uma única vez, já que não violam a restrição de integridade referencial da base; permite a criação e execução de scripts SQL pelo usuário.

4 *Data Migration Performed* (DMP): Uma ferramenta para migração de dados entre SGBDs Diferentes.

4.1 Visão Geral

Este trabalho propõe a criação de uma ferramenta para realizar a migração de dados entre banco de dados diferentes, para isso foi estudado o catálogo de sistemas dos bancos de dados *PostgreSQL* e *SQL Server* para elaborar e executar consultas SQL que retornam os dados necessárias para a migração de dados de um banco de dados origem para um banco de dados destino de maneira automática. A ferramenta foi desenvolvida utilizando a linguagem de programação Java com a IDE de desenvolvimento *Netbeans* 7.1.

4.2 Consultas ao Catálogo de Sistemas

Através de consultas SQL ao catálogo podemos capturar os dados necessários para realizar o processo de migração de dados, pois no catálogo são armazenadas todas as informações sobre o SGBD.

Nesse trabalho foram elaboradas consultas SQL ao catálogo dos bancos de dados *PostgreSQL* e *SQL Server*. Essas consultas retornam toda a estrutura do banco de dados, ou seja, os esquemas, tabelas, colunas, e as chaves primárias e estrangeiras. Com isso são realizadas consultas SQL ao catálogo do banco de dados origem, para obter e montar toda a estrutura do banco de dados que será migrado. Após essa etapa, um componente da implementação fica responsável por montar um *Script* de criação para o banco de dados destino. Em seguida, são executadas outras consultas SQL ao catálogo do banco de dados origem para capturar as informações contidas em todas as tabelas do banco de dados origem, com essas informações capturadas, outro componente executa um *Update* no banco de dados destino povoando todas as tabelas do banco de dados destino, com as mesmas informações contidas no banco de dados origem. Com a estrutura do banco de dados montada no SGBD destino e os dados inseridos nas suas tabelas, a migração de dados é finalizada com sucesso.

Nesse trabalho também foram desenvolvidos dois *drivers*, um *driver* para o SGBD

PostgreSQL e um outro *driver* para o SGBD *SQL Server*. Um *driver* consiste na implementação dos métodos de uma *interface Java* permitindo que a ferramenta capture todas as informações do catálogo de um SGBD, referentes a migração de dados. A Figura 7 apresenta a estrutura da *interface* Java utilizada no projeto, denominada de *Driver for Catalog Obtainment*.

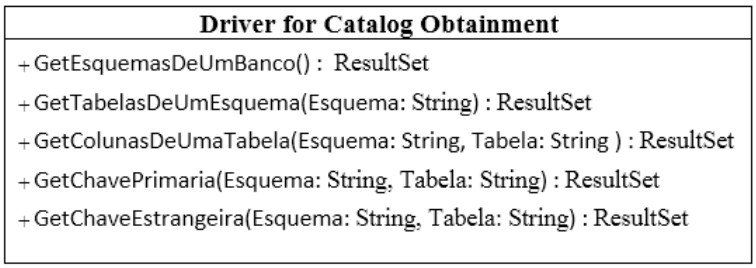

***Figura 7** – Estrutura da Driver For Catalog Obtainment.*

O primeiro método da classe *Driver For Catalog Obtainment* é o *GetEsquemasDeUmBanco* que executa uma consulta SQL ao catálogo de sistema do SGBD, retornando os esquemas do banco de dados envolvido. O método *GetTabelasDeUmEsquema* executa uma consulta SQL a catálogo de sistema do SGBD tendo como parâmetro o nome de um esquema, retornando as tabelas do banco de dados referentes ao esquema. O método *GetColunasDeUmaTabela* executa uma consulta SQL ao catálogo de sistema do SGBD tendo como parâmetros os nomes de um esquema e uma tabela, retornando as colunas do banco de dados referentes ao esquema e a tabela determinados. O método *GetChavePrimaria* executa uma consulta SQL ao catálogo de sistema do SGBD tendo como parâmetros os nomes de um esquema e uma tabela, retornando as chaves primarias do banco de dados referentes ao esquema e a tabela determinados. O método *GetChaveEstrangeira* executa uma consulta SQL ao catálogo de sistema do SGBD tendo como parâmetros um esquema e uma tabela, retornando as chaves estrangeiras do banco de dados referentes ao esquema e a tabela determinados.

4.2.1 Consultas SQL ao catálogo de sistemas do SGBD *PostgreSQL*

A seguir são apresentadas as consultas SQL ao catálogo de sistema do banco de dados *PostgreSQL* que foram desenvolvidas neste projeto e que retornam os dados necessários para a que seja possível a implementação do *driver* de consulta ao catálogo de sistema do SGBD *PostgreSQL* e com isso consequentemente a realização da migração de dados, são elas: Consulta SQL que captura os esquemas do banco de dados (Figura 8), consulta que captura as tabelas de um determinado esquema (Figura 9), consulta que obtém as colunas de uma determinada tabela de um determinado esquema (Figura 10), consulta SQL que obtém a chave primaria de uma determinada tabela de um determinado esquema (Figura 11) e a consulta que retorna a chave estrangeira, ou as chaves estrangeiras, que existem em uma determinada tabela de um

determinado esquema (Figura 12). As consultas são apresentadas tanto em forma de *view* como também em forma de *Function* para uma melhor compreensão.

```
CREATE VIEW Get_Esquemas AS
SELECT DISTINCT schemaname
FROM pg_tables WHERE schemaname NOT IN
('pg_catalog','information_schema','pg_toast')
ORDER BY schemaname;
```

Figura 8 *– Consulta para obter os esquemas do banco de dados no SGBD PostgreSQL.*

```
CREATE FUNCTION Get_TabelasDeUmEsquema(esquema varchar(50))
RETURNS SETOF name AS
'SELECT t.tablename
FROM pg_tables t, pg_namespace n
WHERE  n.nspname = t.schemaname AND n.nspname LIKE esquema'
 language 'sql';
```

Figura 9 *– Consulta para obter as tabelas de um determinado esquema no SGBD PostgreSQL.*

```
CREATE FUNCTION Get_Colunas(esquema varchar(100),tabela varchar(100))
RETURNS SETOF RECORD AS $$
BEGIN
RETURN QUERY SELECT DISTINCT a.attname, t.typname, a.atttypmod-4
as tamanho, a.atthasdef as serial
FROM pg_namespace n, pg_class c, pg_attribute a, pg_tables tt, pg_type t
WHERE n.oid = c.relnamespace
AND a.attrelid = c.oid
AND n.nspname = tt.schemaname
AND a.attnum > 0
AND t.typname not in ('point','line')
AND c.relname = tt.tablename
AND a.atttypid = t.typelem
AND schemaname = esquema
AND tablename = tabela;
RETURN;
END;
$$ language 'plpgsql';
```

Figura 10 *– Consulta para obter as colunas de uma determinada tabela de um determinado esquema no SGBD PostgreSQL.*

```
CREATE FUNCTION Get_Chave_Primaria(esquemavarchar(100), tabela varchar(100))
RETURNS SETOF name AS
'SELECT a.attname
FROM pg_namespace n, pg_tables t, pg_class c
INNER JOIN pg_attribute a ON (c.oid = a.attrelid)
INNER JOIN pg_index i ON (c.oid = i.indrelid)
WHERE n.oid = c.relnamespace and a.attrelid = c.oid
AND n.nspname = t.schemaname
AND a.attnum > 0 and c.relname = t.tablename
AND i.indkey[0] = a.attnum  and i.indisprimary = "t"
AND schemaname = esquema
AND tablename = tabela'
language 'sql';
```

Figura 11 *– Consulta para obter a chave primaria de uma determinada tabela de um determinado esquema no SGBD PostgreSQL.*

```
CREATE FUNCTION Get_ChaveEstrangeira(tabela varchar(50), esquema varchar(50))
RETURNS SETOF name AS
'SELECT   a.attname
FROM pg_tables t, pg_catalog.pg_attribute a
JOIN pg_catalog.pg_class cl ON (a.attrelid = cl.oid AND cl.relkind = "r")
JOIN pg_catalog.pg_namespace n ON (n.oid = cl.relnamespace)
JOIN  pg_catalog.pg_constraint  ct  ON  (a.attrelid  =  ct.conrelid  AND   ct.confrelid  !=  0  AND
ct.conkey[1] = a.attnum)
JOIN pg_catalog.pg_class clf ON (ct.confrelid = clf.oid AND clf.relkind = "r")
JOIN pg_catalog.pg_namespace nf ON (nf.oid = clf.relnamespace)
JOIN pg_catalog.pg_attribute af ON (af.attrelid = ct.confrelid AND af.attnum = ct.confkey[1])
WHERE   cl.relname = t.tablename and n.nspname = t.schemaname
and schemaname = esquema and cl.relname = tabela'
language 'sql';
```

Figura 12 *– Consulta para obter a chave estrangeira de uma determinada tabela de um determinado esquema no SGBD PostgreSQL.*

4.2.2 Consultas SQL ao catálogo de sistemas do SGBD *SQL Server*

Abaixo são apresentadas as consultas SQL ao catálogo de sistema do banco de dados *SQL Server* que foram desenvolvidas nesse projeto e que retornam os dados necessários para

que seja possível a implementação do *driver* de consulta ao catálogo do SGBD *SQL Server* e consequentemente a realização da migração de dados, são elas: Consulta SQL para obter os esquemas do banco de dados do SGBD (Figura 13), consulta SQL que capturam as tabelas de um determinado esquema (Figura 14), consulta SQL para obter as colunas de uma determinada tabela de um determinado esquema (Figura 15), consulta SQL que obtém a chave primaria de uma determinada tabela de um determinado esquema (Figura 16) e a consulta SQL que retorna a chave estrangeira ou as chaves estrangeiras de uma determinada tabela de um determinado esquema (Figura 17). As consultas são apresentadas em forma de *view* ou em forma de *Function*, para uma melhor compreensão.

```
CREATE VIEW GetEsquemasSQLServver AS
SELECT DISTINCT s.name as esquema
FROM sys.tables t, sys.schemas s
WHERE t.schema_id = s.schema_id AND t.name NOT IN ('sysdiagrams')
```

Figura 13 *– Consulta para obter os esquemas do banco de dados no SGBD SQL Server.*

```
CREATE FUNCTION GetTabelasDeumEsquema(@Esquema nvarchar(50))
RETURNS TABLE AS
RETURN (
        SELECT t.name FROM sys.tables t, sys.schemas s
        WHERE t.schema_id = s.schema_id AND s.name = @Esquema
        AND t.name not in ('sysdiagrams')
            )
```

Figura 14 *– Consulta para obter as tabelas de um determinado esquema no SGBD SQL Server.*

```
CREATE FUNCTION GetColunasDeUmaTabela(@Esquema nvarchar(50),@Tabela
nvarchar(50))
RETURNS TABLE AS
    RETURN (
            SELECT distinct c.name, ty.name as tipo,c.max_length ,c.is_nullable
            FROM sys.tables t, sys.columns c, sys.schemas s, sys.types ty
            WHERE t.object_id = c.object_id
            AND t.name like @Tabela
            AND t.name NOT IN ('sysdiagrams')
            AND ty.name NOT IN ('sysname')
            AND t.schema_id = s.schema_id
            AND s.name like @Esquema
            AND ty.system_type_id = c.system_type_id
                )
```

Figura 15 *– Consulta para obter as colunas de uma determinada tabela de um determinado esquema no SGBD SQL Server.*

```
CREATE FUNCTION GetChavePrimaria(@Esquema nvarchar(50),@Tabela nvarchar(50))
RETURNS TABLE  AS
RETURN(
        SELECT  COLUMN_NAME
        FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
        WHERE TABLE_NAME = @Tabela
        AND TABLE_SCHEMA = @Esquema
        AND CONSTRAINT_NAME = (select name from sys.key_constraints
        WHEREparent_object_id =
                        (select object_id from sys.tables where name =@Tabela))
      )
```

Figura 16 *– Consulta para obter a chave primaria de uma determinada tabela de um determinado esquema no SGBD SQL Server.*

```
CREATE FUNCTION GetChaveEstrangeira(@Esquema nvarchar(50),@Tabela nvarchar(50))
RETURNS TABLE AS
      RETURN(
      SELECT cc.COLUMN_NAME as nome
      FROM sys.foreign_keys f,INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE cc
      WHERE cc.TABLE_NAME = @Tabela
      AND cc.TABLE_SCHEMA = @Esquema
      AND cc.CONSTRAINT_NAME = f.name
      AND parent_object_id = (select object_id from sys.tables where name = @Tabela)
            )
```

Figura 17 *– Consulta para obter a chave estrangeira de uma determinada tabela de um determinado esquema no SGBD SQL Server.*

4.3 Detalhes de Implementação

As consultas SQL às tabelas do catálogo de sistemas de um SGBD, são responsáveis por obter os dados de um banco de dados. Esses dados capturados são manipulados de maneira que sejam utilizados por componentes durante o processo de migração de dados. Os componentes são módulos de execução de tarefas durante a migração de dados, esses módulos são classes *Java* que utilizam-se das consultas SQL ao catálogo de sistemas e realizam a migração de dados propriamente dita.

Um ponto muito importante na hora da realização da migração de dados entre SGBDs diferentes é fazer o mapeamento dos tipos de dados. O mapeamento consiste no processo de verificação de todos os tipos de dados dos SGBDs envolvidos durante a migração de dados. Uma má elaboração desse mapeamento pode acarretar em erros de compatibilidade de tipos de dados entre os SGBDs.

Os SGBDS na sua maioria seguem o padrão ANSI/ISO para representar os seus tipos

de dados, mais além dos tipos de dados tradicionais, também possui outras maneira de representação de um mesmo tipo de dado, por exemplo, no SGBD *PostgreSQL* temos o tipo de dado *Caracter varying(n)* que é um tipo de dado para representar caracteres com um comprimento variável com limite de tamanho, enquanto que no SGBD *SQL Server*, o mesmo tipo de dado é representado pelo tipo *nvarchar(n)*. Por essas diferenças entre dados dos SGBDS, se torna indispensável o mapeamento desses tipos de dados durante o processo de migração de dados. A Tabela 8 apresenta os tipos de dados que foram mapeados do SGBD *PostgreSQL* para o *SQL Server* e a Tabela 9 apresenta os tipos de dados que foram mapeados do SGBD *SQL Server* para o SGBD *PostgreSQL*. Para realização desse mapeamento foi utilizada uma técnica de comparação dos tipos de dados e foi realizado o estudo do tamanho dos tipos de dados e com isso, os dados são mapeados de acordo com os respectivos tipos no SGBD destino.

Na ferramenta Data Migration Performed o processo de migração foi divido em duas partes:

1. É migrada toda a estrutura do banco de dados, ou seja, as tabelas, colunas, tipos de colunas e chaves primaria e estrangeira.

2. São migrados os dados das tabelas, ou seja, todos os dados das tabelas em que existam dados inseridos.

Tipos de dados do SGBD PostgreSQL	**Como o tipo de dado é aceito no SGBD SQL Server**
Smallint	Smallint
Integer	Int
Bigint	Bigint
Decimal	Money
Numeric	Int
Real	Real
Double Precision	Float
Caracter varying(n)	Nvarchar(n)
Varchar(n)	Varchar(n)
Character(n)	Nchar(n)
Char(n)	Char(n)
Text	nText
Timestamp without Time Zone, Timestamp with Time Zone	Smalldatetime
Date	Datetime

Tabela 8 *– Tipos de dados mapeados do SGBD PostgreSQL para o SQL Server.*

Tipos de dados do SGBD SQL Server	**Como o tipo de dado é aceito no SGBD PostgreSQL**
Int	Integer
Smalldatetime	Smallint
Money	Decimal
Real	Real
Varchar(n)	Varchar(n)
Nvarchar(n)	Caracter verying(n)
Nchar(n)	Caracter(n)
Char(n)	Char(n)
Ntext	Text
Smalldatetime	Timestamp without Time Zone
Datetime	Date
Bigint	Bigint

Tabela 9 *– Tipos de dados mapeados do SGBD SQL Server para o SGBD PostgreSQL.*

A seguir é descrito, com maiores detalhes, a função de cada componente da ferramenta *Data Migration Performed* no processo de migração de dados, e a Figura 18 apresenta genericamente a maneira como cada componente interage.

1. **Driver for Catalog Obtainment** – contém as consultas SQL ao catálogo de sistemas e é responsável por capturar os dados do SBGD origem necessários para a migração de

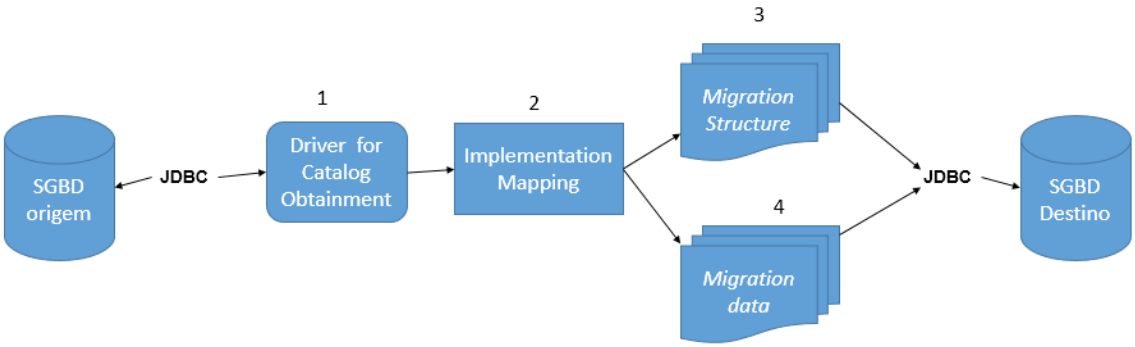

Figura 18 *– Estrutura da ferramenta para migração de dados entre SGBDs diferentes.*

dados.

2. **Implementation Mapping** - Responsável pelo mapeamento sobre os dados do SGBD origem para o SGBD destino.

3. **Migration Structure** – recebe o esquema mapeado e monta a estrutura (esquema do SGBD) em um *script* do tipo *create*. Em seguida executa esse *script* no SGBD destino.

4. **Migration Data** – recebe os dados do SGBD origem mapeados e monta um *script* do tipo *create*. Em seguida executa esse *script* no SGBD destino.

Sendo definido o SGBD origem que será migrado, é feita uma conexão via JDBC entre o *driver*, que contém as consultas SQL ao catálogo de sistemas e o SGBD. Com o esquema do SGBD origem capturado, o módulo **Implementation Mapping** executa o mapeamento dos dados para o SGBD destino ao qual será realizada a migração. Após o mapeamento dos dados, é feita a conexão via JDBC ao SGBD destino e então é iniciada a migração dos dados que divide-se em duas etapas: Primeiro, o módulo **Migration Structure** monta toda a estrutura do esquema do SGBD destino e executa a migração de dados, ou seja, são migrados: os esquemas, as tabelas, as colunas e as chaves primaria e estrangeiras para o SGBD origem para o SGBD destino. Segundo, o módulo **Migration Data** é responsável por capturar e montar um *script* do tipo *insert* com os dados do SGBD origem, em seguida o **Migration Data** executa a migração de todos os dados do SGBD origem para o SGBD destino. A Figura 19 apresenta a sequência de execução das etapas da ferramenta *Data Migration Performed* durante a migração de dados.

1. O **Driver for Catalog Obtainment** se conecta via JDBC ao banco de dados origem e faz consultas ao esquema do SGBD origem.

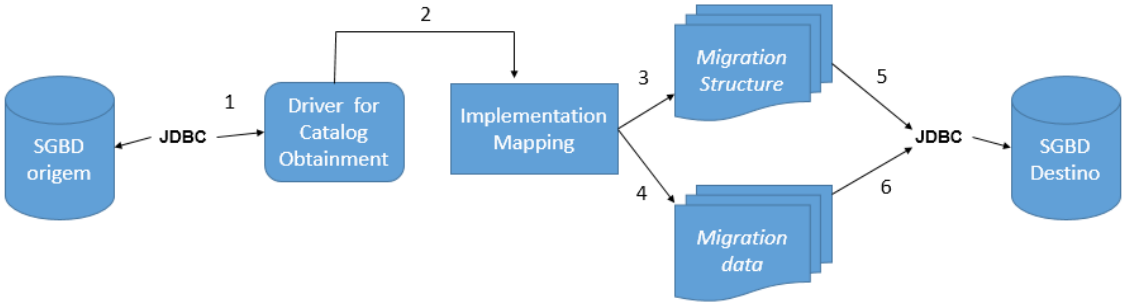

Figura 19 *– Funcionamento da Ferramenta para migração de dados entre SGBDs diferentes.*

2. O módulo **Implementation Mapping** recebe os dados do módulo driver e faz a execução do mapeamento sobre os dados do SGBD origem para o SGBD destino.

3. O **Migration Structure** recebe do módulo **Implementation Mapping** a estrutura (esquema) do banco de dados origem com os dados já mapeados de acordo com as definições do SGBD destino.

4. O módulo **Migration Data**, recebe os dados (povoamento das tabelas) do SGBD já mapeados pelo módulo **Implementation Mapping**.

5. O **Migration Structure** cria e executa um *script* do tipo *create* para o SGBD destino, ou seja, é criado toda a estrutura do SGBD origem no SGBD destino.

6. O **Migration Data** executa a migração de todos os dados do SGBD origem para o SGBD destino, criando um *script* do tipo *insert* com os dados do SGBD origem e executa esse *script* no SGBD destino, ou seja, os dados do SGBD origem são inseridos no SGBD destino.

Para o desenvolvimento desse projeto e melhor compreensão do seu funcionamento, foram utilizados os padrões de projeto, são eles, padrão Strategy e padrão Chain Of Responsability. Os padrões de projeto são descritos com uma solução para um determinado problema e um determinado contexto. A Figura 20 mostra o diagrama de classes do seguinte projeto, onde são utilizados dois padrões de projeto.

O primeiro padrão utilizado no desenvolvimento desse projeto foi o *Strategy*, que trata-se de um padrão que deve ser utilizado quando uma classe possui diversos métodos que possam ser utilizados por outra classe, ou seja, uma ou mais classes implementam os métodos de uma terceira classe. No diagrama da Figura (20) podemos observar que existe uma classe (**Driver**

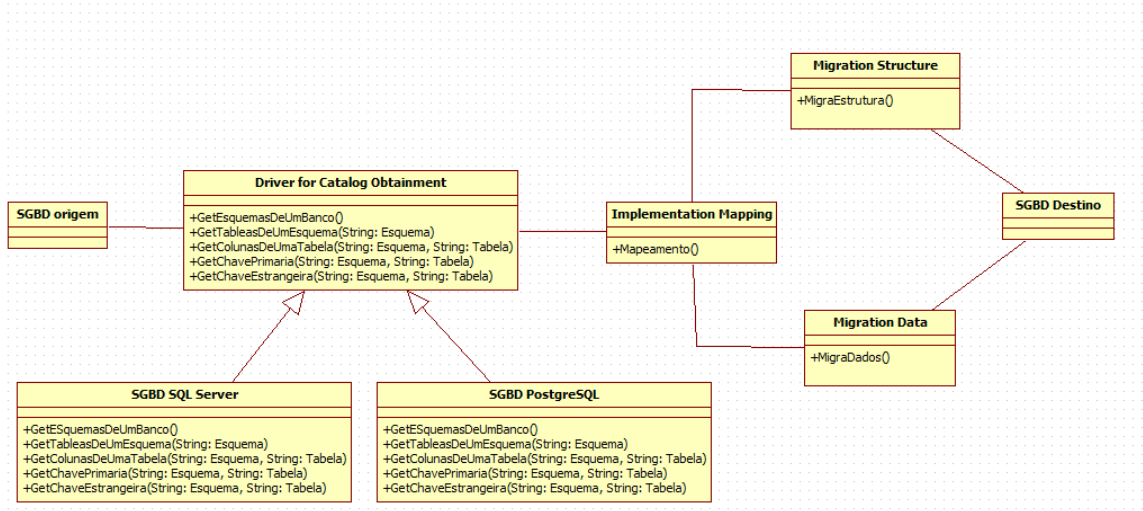

Figura 20 *– Diagrama de classes do Data Migration Performed.*

for Catalog Obtainment) com diversos métodos que são as consultas SQL ao catálogo de sistema de um SGBD, e que esses métodos deverão ser implementados pelas classes que irão acessar o catálogo de sistema do SGBD (classes **SGBD PostgreSQL** e **SGBD SQL Server**). A partir dessa estrutura, são realizadas novas implementações dos métodos, podendo ser criadas e introduzidas posteriormente.

O segundo padrão utilizado para o desenvolvimento desse projeto foi o padrão *Chain Of Responsability*, que é um padrão de projeto que cria uma ordem de execução onde cada elemento processa as informações e em seguida delega a execução ao próximo elemento da sequência. No processo de migração de dados são seguidos várias etapas até que o mesmo seja concluído. Cada etapa da migração é realizada por um componente (elemento), desde a parte de captura dos dados do catálogo de sistema do SGBD origem até a migração dos dados propriamente ditos no SGBD destino. Na Figura (20) podemos observar que a execução das etapas são realizadas seguindo uma ordem de execução: Primeiro o **Driver for Catalog Obtainment** consulta o catálogo do SGBD origem, através das consultas SQL, e obtém os dados que são necessários para o processo de migração. Em seguida, após o **Driver for Catalog Obtainment** capturar os dados do SGBD origem, a classe **Implementation Mapping** vai receber esses dados e realizar o mapeamento dos mesmos para o SGBD destino determinado. Após as consultas serem mapeadas, as próximas etapas são as de migração da estrutura do SGBD e migração dos dados propriamente ditos do SGBD. Nesse caso, primeiro a classe **Migration Structure** recebe a estrutura do SGBD origem mapeada em seguida migra a estrutura para o SGBD destino. Após a migração da estrutura, é então realizada a última etapa do processo de migração de dados que é realizada pela classe **Migration Data**, onde os dados propriamente ditos são inseridos no SGBD destino e com isso finalizando o processo de migração de dados.

5 A Ferramenta *Data Migration Performed* (DMP)

A ferramenta para executar a migração de dados entre SGBDs diferentes utilizando o catálogo de sistemas, denominada **Data Migration Performed** (Figura 21), foi escrita na linguagem *Java* utilizando a plataforma de desenvolvimento IDE *NetBeans* na sua versão 7.1 juntamente com o JDK e as bibliotecas JDBC para que seja feita a conexão entre a aplicação e os SGBDs. O protótipo possui uma interface gráfica intuitiva e que permite ao usuário interagir durante o processo de migração de dados. Contudo, para isso, é necessário fornecer a ferramenta as informações de acesso aos SGBDs origem e destino (tais como, endereço do servidor, usuário, senha, nome do banco de dados). Essas informações devem ser fornecidas no painel de conexões do SGBD origem (Figura 22) e do SGBD destino (Figura 23).

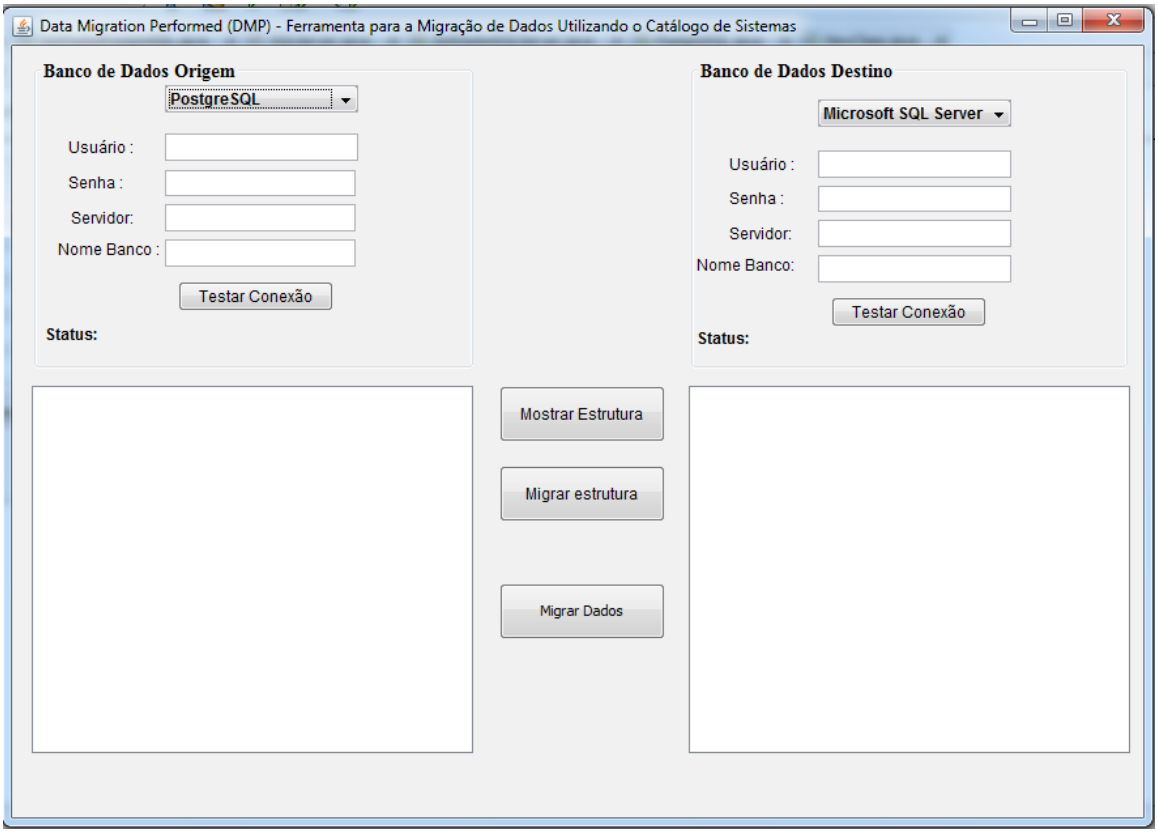

Figura 21 *– Ferramenta Data Migration Performed.*

Considerando que as informações de conexão tenham sido fornecidas corretamente, o

Figura 22 *– Tela de conexão com o SGBD origem.*

Figura 23 *– Tela de conexão do SGBD destino.*

processo de migração de dados pode ser iniciado. Inicia-se primeiramente clicando no botão *Mostrar Estrutura* que mostrará no *JtextArea* a esquerda a estrutura (esquema) do SGBD origem e no *JTextArea*,a direita a estrutura (esquema) já mapeada com os padrões do SGBD destino (Figura 24).

Com a estrutura do SGBD origem já montada, o próximo passo a ser executado é o da migração da mesma para o SGBD destino. Para isso, basta clicar no botão Migrar Dados. Supondo que a migração da estrutura do SGBD foi realizada com sucesso, na tela será mostrada uma mensagem informando que a migração da estrutura foi executada com sucesso (Figura 25).

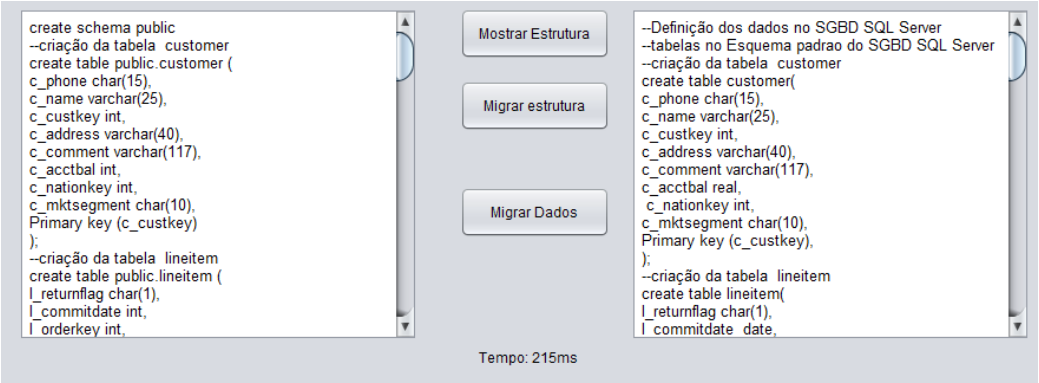

Figura 24 *– Tela com a estrutura (Esquema) dos SGBDs origem e destino*

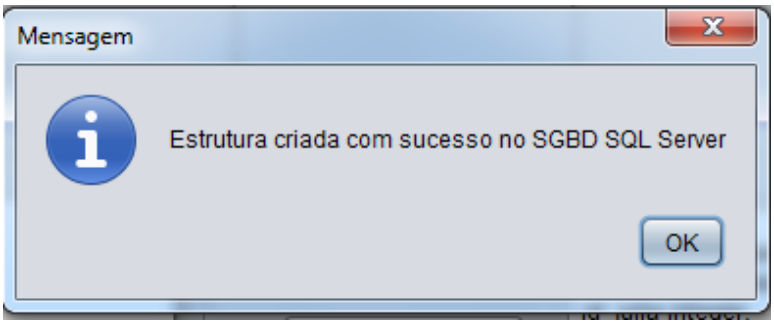

Figura 25 *– Tela com a mensagem informando que a estrutura do SGBD foi migrada com sucesso.*

Considerando que que a estrutura do SGBD foi migrada, ou seja, criada no SGBD destino, o próximo passo é a migração dos dados das tabelas do SGBD origem para o SGBD destino. Para isso, basta clicar no botão Migrar Dados. Esse processo pode ter variação de tempo de execução dependendo do número de dados que estão sendo migrados. Se essa etapa for concluída corretamente, é exibida uma mensagem na tela (Figura 26) informando que os dados foram migrados com sucesso.

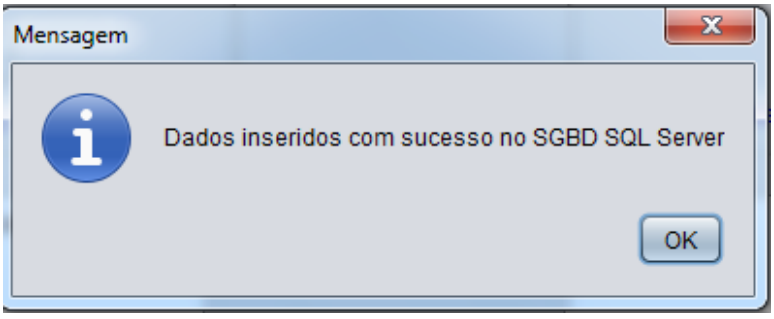

Figura 26 *– Tela com a mensagem informando que os dados foram migrados com sucesso.*

6 Experimento

6.1 Ambiente de Execução

Para a execução dos testes foi utilizado um ambiente composto por uma estação *Core(TM) i3-350 2.27GHz* com 3GB de RAM e 500 de HD, e com o sistema operacional *Windows 7 Utimate*. Os SGBDs utilizados nos testes foram o *PostgreSQL* 9.2 e o *SQL Server 2008*.

6.2 Cenários de Teste

A fim de provar a eficácia da ferramenta para migração de dados proposta, foi utilizado dois cenários diferentes de testes com os SGBDs *PostgreSQL* e *SQL Server*. Os cenários utilizam o *benckmark* TPC-H, o qual consiste em um *benckmark* voltado para aplicações de suporte a decisão. O TPC-H é formado por um conjunto de 22 consultas ad-hoc (além de dois comandos DDL utilizados para a criação e a remoção de uma visão) e possui uma estrutura padrão composta por oito tabelas. Destas, seis são tabelas dimensionais (*Region*, *Nation*, *Supplier*, *Part*, *Customer* e *Partsupp*) e duas de fatos (*Orders* e *Lineitem*). Para a realização dos testes foi utilizada uma base de dados de 1GB, onde a tabela *Liniitem* tem 6.000.000 de tuplas.

Para os cenários, os seguintes testes foram realizados: i) executou-se a migração dos dados tendo como origem o SGBD *PostgreSQL* e como destino o SGBD *SQL Server*; ii) executou-se a migração dos dados tendo como SGBD origem o *SQL Server* e como destino o SGBD *PostgreSQL*.

6.2.1 Cenário 1: Migração de dados *PostgreSQL* para *SQL Server* utilizando a base de dados *Benchmark* TPC-H

Para a realização dos testes com o *Benckmark* TPC-H, primeiramente foi escolhido aleatoriamente a ordem dos SGBDs origem e destino. Primeiro utilizou-se o SGBD *PostgreSQL* como origem dos dados e o SGBD *SQL Server* como destino dos dados. Foi feito o *backup* dos dados da base de dados *Benckmark* TPC-H no SGBD *PostgreSQL* (origem) e no SGBD destino, para receber os dados da migração, foi criado um banco de dados chamado *TPC-H_-SQLSERVER*.

A Figura 27 mostra a estrutura da base de dados *Benckmark* TPC-H capturada e montada

(tanto nas definições do SGBD origem quanto nos definições do SGBD destino) pela **Data Migration Performed**(DMP).

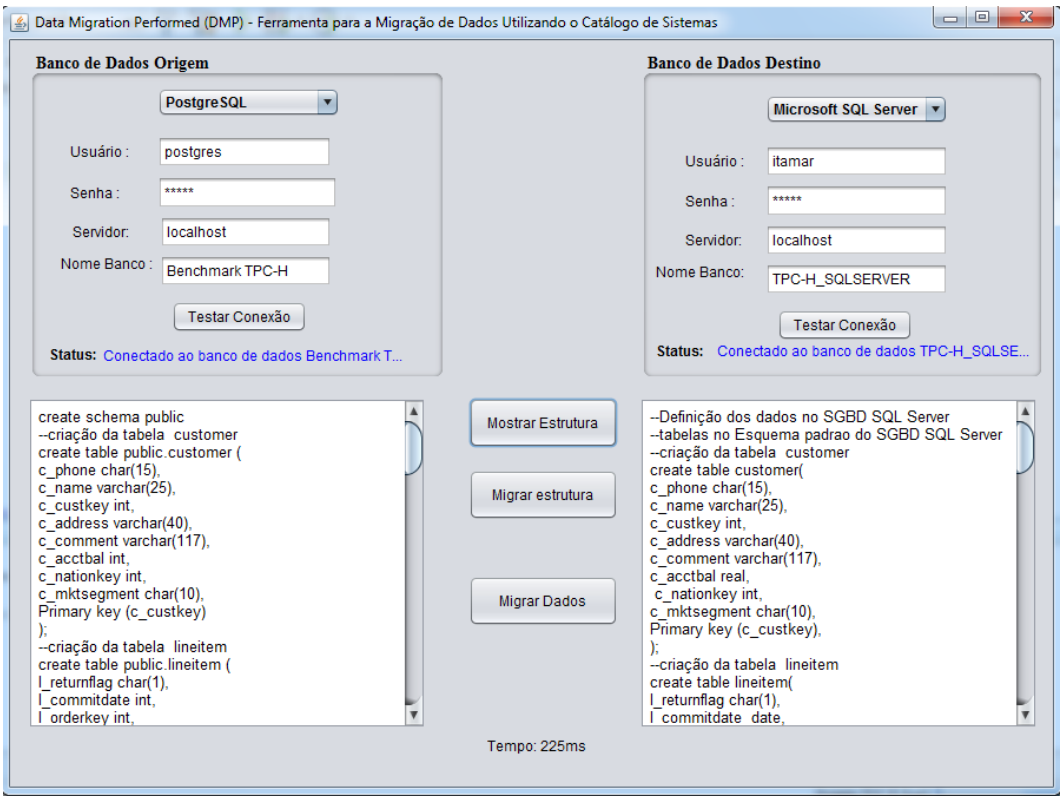

Figura 27 *– Tela com a captura de toda a estrutura do banco de dados Benchmark TPC-H, tendo como origem SGBD PostgreSQL e destino o SGBD SQL Server.*

O *Benckmark* TPC-H, como foi dito na seção anterior, possui as seguintes tabelas: *Region*, *Nation*, *Supplier*, *Part*, *Customer*, *Partsupp*, *Orders* e *Lineitem*, sendo que as mesmas possuem diferentes quantidades de dados armazenados. A Tabela 10 apresenta a quantidade de dados que cada tabela possui.

Tabela	**Quantidade de dados**
Region	5
Nation	25
Supplier	10000
Part	200000
Customer	150000
Partsupp	800000
Orders	150000
Lineitem	6001240
Total de Dados	7311270

Tabela 10 *– Quantidade e dados de todas as tabelas do banco de dados Benchmark TPC-H.*

Como a base de dados do *Benchmark* TPC-H possui uma quantidade de dados grande se torna inviável contarmos dado por dado de todas as tabelas no SGBD que recebeu a migração

de dados. Com isso para comprovar a eficácia da ferramenta executou a contagem (*Count*) da quantidade de registros existentes no SGBD destino (que recebeu a migração de dados) e foi comparado com a quantidade de dados existentes no SGBD origem.

A Figura 28 apresenta um exemplo da contagem e dados da tabela *customer* no SGBD destino. Com isso, podemos verificar que tivemos a mesma quantidade de dados nas tabelas origem e destino.

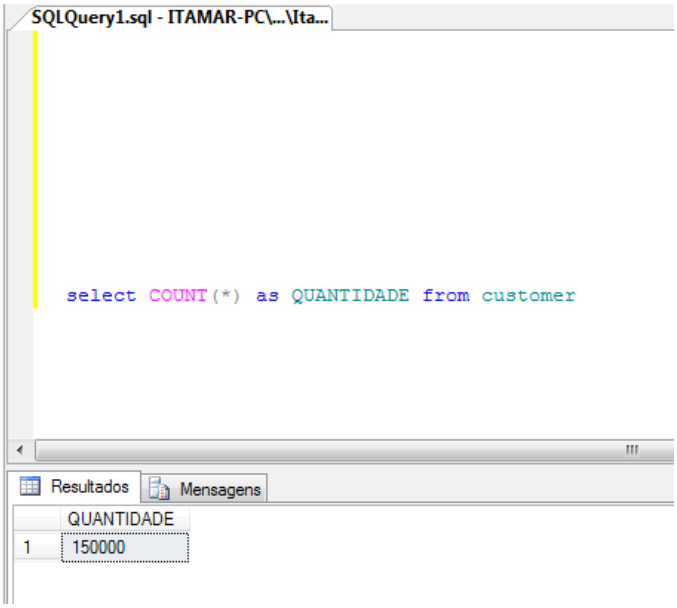

Figura 28 *– Quantidade de dados na tabela customer no SGBD destino após a migração de dados.*

6.2.2 Cenário 2: Migração de dados do *SQL Server* para o *PostgreSQL* utilizando a base de dados *Benchmark* TPC-H.

Semelhante ao teste anteriormente apresentado, esse teste utilizou também a base de dados *Benchmark* TPC-H. Foi criada um banco de dados com os dados do *Benchmark* TPC-H no *SQL Server* (SGBD origem) e para receber os dados foi criado um banco de dados no SGBD Destino *PostgreSQL* chamado de *TPC-H_POSTGRESQL*. A Figura 29 apresenta a tela do **Data Migration Performed** capturando a estrutura do banco de dados *Benchmark* TCP-H do SGBD *SQL Server*.

Da mesma forma como foi apresentado no teste anterior (Seção 6.2.1), os dados foram migrados de um SGBD para outro de forma correta. Levando em consideração que a base de dados TCP-H possui uma quantidade muito grande de dados, seria inviável contar e mostrar dado por dado que foi migrado do SGBD *SQL Server* para o SGBD *PostgreSQL*. Utilizando a mesma abordagem anteriormente citada, foi executada a contagem (*Count*) da quantidade de registros existentes no SGBD destino (que recebeu a migração de dados) e foi comparado com

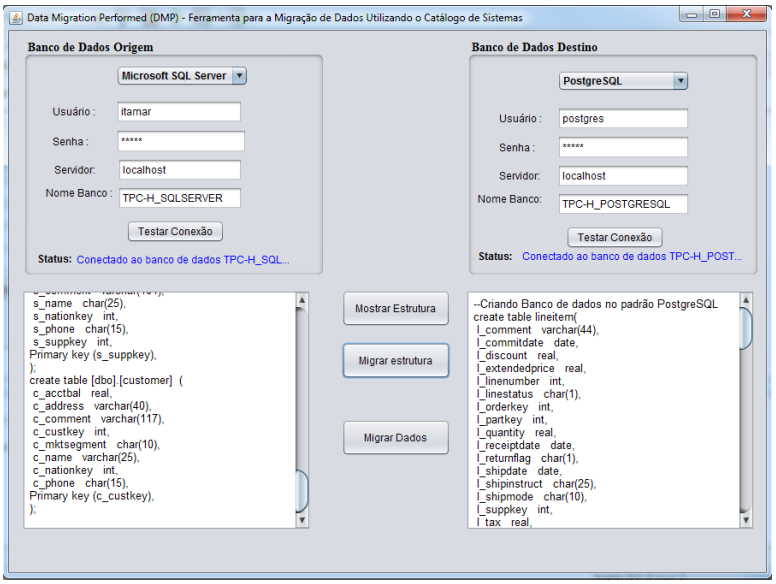

Figura 29 *– Tela com a captura da estrutura do SGBD Benchmark TCP-H com o SGBD SQL Server como origem do dados e o SGBD PostgreSQL com destino dos dados.*

a quantidade de dados existentes no SGBD origem. A Figura 30 apresenta a contagem de dados da tabela *customer* após a migração dos dados.

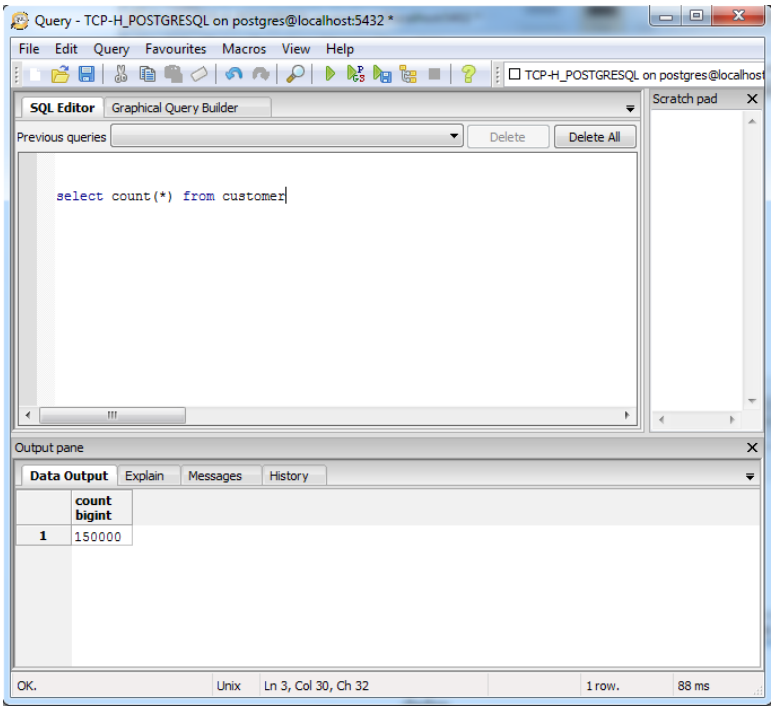

Figura 30 *– Tabela customer no SGBD PostgreSQL após a migração de dados.*

7 Trabalhos Futuros

A seguir são descritas algumas possibilidades para trabalhos futuros:

- Estender o protótipo implementado nesse trabalho com outros drivers de SGBDs, como o *MySQL*, *Oracle*, *DB2*, entre outros.
- Tornar possível também a transferência de outras funções dos SGBDs tais como: *Stored Procedures*, *Functions*, *Triggers*.

8 Conclusão

O processo de migração de dados entre SGBDs diferentes é um procedimento complexo, levando em consideração que SGBDs diferentes apresentam diferentes tipos de dados e variações em relação ao padrão ANSI/SQL. Apesar dessa complexidade, pode-se observar que o **Data Migration Performed** lida com essas questões e contornam esses problemas durante a migração de dados entre SGBDs diferentes.

O **Data Migration Performed** utiliza o catálogo de sistemas do próprio SGBD envolvido na migração de dados para obter os dados e informações sobre os mesmos. E ainda, faz com que a migração de dados se torne mais flexível, tratando as heterogeneidades dos SGBDs envolvidos.

Com isso, em se tratando de ambientes que possuem grandes quantidades de dados e desejam que os mesmos sejam migrados para outros SGBDs, essa ferramenta é uma solução viável para facilitar a transferência dos dados mantendo a integridade e a semântica dos mesmos.

Referências

ALEXANDER, Christopher. **An Introduction for Object – Oriented Design**. 2007.

BARRETO, Rodrigo Lumack do Monte. **Data Migration Wizard: Um assistente de Migração de Dados para Bancos de Dados no SQL Server.** Centro de Informática da Universidade Federal do Pernambuco UFPE, Pernambuco, 2008.

BROY, M. Legacy. **PostgreSQL prático: versão 8.1.4.** Migrationsstrategien.Hauptseminar Management von Softwaresystemen., München, 2005.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados.** 4. ed. São Paulo: Pearson Addison Wesley, 2005.

GUERRA, Eduardo. **Design Patterns com Java – Projeto Orientado a Objetos guiado por padrões.** São Paulo, 2012.

MARTINS, Renato de Almeida. **Apostila Microsoft SQL Server 7.0 – Instalação, configuração e gerenciamento**. 2002.

MONTEIRO, José Maria; LIFSCHITZ, Sérgio; BRAYNER. **Extraindo Metadados de SGBDs.** Rio de Janeiro, 2007.

MORELLI, E. M. T. **Automatic Index Recriation in a Relational DBMS (in portuguese)**. Department of Informatics, PUC-Rio, 2006.

NETO, Josino R.; PASSOS, Erick B. **JExodus: uma ferramenta para migração de dados independente de SGBD.** Instituto Federal de Educação Ciência e Tecnologia do Piauí (IFPI), Piauí, 2006.

POSTGRESQL. **Documentação do PostgreSQL.** 2013. Disponível em: <http://pgdocptbr.sourceforge.net/pg80/catalogs.html>. Acesso em: 03 fev. 2013.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Sistemas de Banco de dados.** São Paulo: McGraw-Hill, 2008.

RIBAMAR, F. S. **PostgreSQL prático: versão 8.1.4.** São Paulo, p. 157, 2006.

SILVERSCHATS, Abraham; KORTH, Henry; SURDARSHAM, S. **Sistema de Banco de Dados.** 5. ed. Rio de Janeiro: Elsevier, 2006.

WIKIPEDIA. **Padrão de Projeto Chain of Responsability**. 2014. Disponível em: <http://pt.wikipedia.org/wiki/ChainofResponsability>. Acesso em: 03 fev. 2014.

WIKIPEDIA. **Padrão de Projeto Strategy**. 2014. Disponível em: <http://pt.wikipedia.org/wiki/strategy>. Acesso em: 03 fev. 2014.

WU, H. Sahraoui L.; VALTCHEV, P. **Coping with Legacy System Migration Complexity**. 10th IEEE Internacional Conference on Engineering of Complex Computer Systems, p. 600–609, 2005.