



Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Sistemas de Informação

RENATO SULZBACH

MAPEAMENTO DE CONSULTAS SQL
ENTRE SISTEMAS GERENCIADORES DE BANCOS DE DADOS
UTILIZANDO TECNOLOGIA XSLT

FLORIANÓPOLIS
2011



Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Sistemas de Informação

Renato Sulzbach

MAPEAMENTO DE CONSULTAS SQL
ENTRE SISTEMAS GERENCIADORES DE BANCOS DE DADOS
UTILIZANDO TECNOLOGIA XSLT

Trabalho de Conclusão de Curso apresentado
à Universidade Federal de Santa Catarina,
para obtenção do título de Bacharel em
Sistemas de Informação, sob a orientação do
Prof. Dr. Ronaldo dos Santos Melo

Florianópolis
2011

Sulzbach, Renato.

Mapeamento de consultas SQL entre sistemas gerenciadores de bancos de dados utilizando tecnologia XSLT. – Florianópolis, SC: [s.n.], 2011

Orientador: Prof. Dr. Ronaldo dos Santos Melo.

Trabalho de conclusão de curso (graduação) – Universidade Federal de Santa Catarina, Sistemas de Informação.

“Cada país possui seu próprio idioma, mas
os assuntos sobre os quais fala nossa
alma são os mesmos em qualquer lugar”

Tertúlio

SUMÁRIO

LISTA DE ABREVIATURAS	8
RESUMO	9
ABSTRACT	10
1. Introdução	11
2. Fundamentação Teórica	13
2.1. XML	13
2.1.1. XML – Básico	13
2.1.2. XML – Avançado	14
2.2. XML Path (XPath)	15
2.3. XSL Transformation Language (XSLT)	18
3. Trabalhos Relacionados	20
4. Ferramenta Proposta	23
4.1. Visão Geral	23
4.2. Processo de Mapeamento	24
4.2.1. Compilação	25
4.2.2. Transformação	33
5. API	41
6. Estudo de Caso	44
7. Conclusão	47
REFERÊNCIAS BIBLIOGRÁFICAS	49
ANEXO 1 – ARTIGO	51
ANEXO 2 – TESTES DE MAPEAMENTO	58
ANEXO 3 – FONTES	71

LISTA DE FIGURAS

Figura 1	- EXEMPLO DE UM DOCUMENTO XML BEM FORMADO	13
Figura 2	- SWISSQL	20
Figura 3	- ESQUEMA GERAL DO PROCESSO DE MAPEAMENTO ENTRE COMANDOS SQL	24
Figura 4	- ARQUIVO XSL GENÉRICO	34
Figura 5	- ARQUIVO XSL ESPECÍFICO PARA O MICROSOFT SQL SERVER	36
Figura 6	- ARQUIVO XSL ESPECÍFICO PARA O MYSQL	37
Figura 7	- ARQUIVO XSL ESPECÍFICO PARA O POSTGRESQL	38
Figura 8	- ARQUIVO XSL ESPECÍFICO PARA O FIREBIRD	39
Figura 9	- API: ARQUITETURA GERAL	41
Figura 10	- API: DIAGRAMA DE SEQUÊNCIA	42
Figura 11	- API: DIAGRAMA DE CLASSES	43
Figura 12	- EXECUÇÃO DA FERRAMENTA COM OS DADOS DE ENTRADA	44
Figura 13	- ESTRUTURA DO CÓDIGO INTERMEDIÁRIO NO FORMATO XML	45
Figura 14	- EXPRESSÕES SQL EQUIVALENTES	46

LISTA DE TABELAS

Tabela 1	- AS DIFERENTES SINTAXES PARA UM MESMO COMANDO SQL ..	11
Tabela 2	- ENTIDADES PRÉ-DEFINIDAS DE XML	14
Tabela 3	- EIXOS DE XPATH	16
Tabela 4	- TESTES DE NODOS DE XPATH	17
Tabela 5	- OPERADORES DE XPATH	17
Tabela 6	- FUNÇÕES DE XPATH	18
Tabela 7	- ELEMENTOS DE XSLT	19
Tabela 8	- GRAMÁTICA DO MICROSOFT SQL SERVER	25
Tabela 9	- GRAMÁTICA DO MYSQL	26
Tabela 10	- GRAMÁTICA DO POSTGRESQL	27
Tabela 11	- GRAMÁTICA DO FIREBIRD	28
Tabela 12	- AÇÕES SEMÂNTICAS PARA O MICROSOFT SQL SERVER	29
Tabela 13	- AÇÕES SEMÂNTICAS PARA O MYSQL	30
Tabela 14	- AÇÕES SEMÂNTICAS PARA O POSTGRESQL	31
Tabela 15	- AÇÕES SEMÂNTICAS PARA O FIREBIRD	32

LISTA DE ABREVIATURAS

HTML	-	HyperText Markup Language
SGBDR	-	Sistema Gerenciador de Banco de Dados Relacional
SQL	-	Structure Query Language
XML	-	Extensible Markup Language
XSL	-	Extensible Stylesheet Language
XSLT	-	Extensible Stylesheet Language Transformations

RESUMO

O presente trabalho tem como objetivo desenvolver uma ferramenta que converte uma expressão de consulta SQL, escrita na sintaxe específica de um Sistema Gerenciador de Banco de Dados Relacional (SGBDR), para a sintaxe de outro SGBDR. Essa conversão é realizada utilizando técnicas de compilação e o suporte da tecnologia XSLT. A API desenvolvida juntamente com a ferramenta, e que representa o núcleo do processo de conversão, pode ser facilmente incorporada dentro de aplicativos Java/J2EE, permitindo que esses possam trabalhar com vários SGBDRs sem a necessidade de portar consultas SQL específicas de um banco de dados para outro.

Palavras-chave: SGBDR, SQL, XML, XPATH, XSLT, API

ABSTRACT

This research aims at developing a tool that converts an SQL query expression syntax, written in a specific Relational Database Management System (RDBMS), to the syntax of other RDBMS. This conversion is performed using compiling techniques and the XSLT technology support. The API developed along with the tool (the core of the conversion process), can be easily embedded within Java/J2EE applications, allowing it to work with multiple RDBMS without the need to port specific SQL queries from one database to another.

1. Introdução

XML (eXtensible Markup Language) é uma linguagem de marcação de dados que provê um formato designado a descrever dados semi-estruturados. Recomendação da W3C, ela é utilizada por diversos tipos de aplicações principalmente para o compartilhamento de informações através da Internet. Documentos XML são auto-descritivos e possuem uma sintaxe simples. Ao contrário de documentos HTML, *tags* XML não são pré-definidas. O autor pode criar suas próprias marcações a fim de satisfazer necessidades especiais [W3C99].

O gerenciamento de dados XML é realizado por um conjunto de tecnologias provido pela W3C. A tecnologia XSL é uma delas. XSL é uma família de recomendações para a definição de transformações e apresentações de documentos XML. Ela consiste de três partes, compostas pelas tecnologias XPath, XSLT e XSL-FO. XSLT *stylesheet* especifica a apresentação de uma classe de documentos XML, descrevendo como uma instância dessa classe é transformada em um documento XML que utiliza um vocabulário formatado, como (X)HTML ou XSL-FO [W3C07].

No contexto desse trabalho, o problema a ser tratado são expressões SQL que podem assumir diferentes sintaxes dependendo do SGBDR alvo. Vários SGBDRs possuem como característica diferentes formas de disponibilizar uma determinada funcionalidade através de comandos SQL. Essas diferentes formas de implementação trazem dificuldades aos usuários, pois nem sempre uma expressão SQL escrita por um fabricante é válida para outra. Com essa falta de padronização, usuários têm de recorrer a diferentes fontes de pesquisa em busca da tradução correta de uma expressão para o SGBDR ao qual deseja aplicar. Um exemplo dessa heterogeneidade é o comando que especifica o primeiro conjunto de linhas a ser retornado do resultado de uma consulta SQL. Tal comando pode assumir diferenças tanto na grafia quanto na posição na estrutura da expressão SQL. A Tabela 1 lista essas diferenças para quatro SGBDRs.

Tabela 1 – As diferentes sintaxes para um mesmo comando SQL

Microsoft SQL Server	SELECT TOP 10 column FROM table
MySQL	SELECT column FROM table LIMIT 10
PostgreSQL	SELECT column FROM table LIMIT 10
Firebird	SELECT FIRST 10 column FROM table

Assim sendo, esse trabalho tem por objetivo construir uma ferramenta onde uma determinada expressão de consulta SQL de entrada, escrita para um determinado

SGBDR, e representada em um documento XML, pode ser convertida para o dialeto SQL de outro SGBDR. A ferramenta utiliza informações de transformação armazenadas em documentos XSLT específicos de cada SGBDR e que definirão a sintaxe da expressão SQL para o SGBDR desejado. Essas informações de transformação estão relacionadas aos comandos de consulta SQL. A restrição a esse tipo de expressão fez-se necessária uma vez que existe uma grande variedade de comandos SQL. Optou-se por comandos de consulta pois são os tipos mais frequentes de operações que um SGBD executa e os que apresentam mais diferenças sintáticas.

A motivação para o desenvolvimento desta ferramenta vem das limitações existentes nos trabalhos relacionados (SWISSQL, 2011) (SQLWAYS, 2011). Estas limitações dizem respeito à falta de flexibilidade para o mapeamento de consultas entre SGBDs de interesse e a inexistência de soluções gratuitas para a realização desta tarefa.

Os detalhes do trabalho são descritos nos capítulos que se seguem. O capítulo 2 abrange a fundamentação teórica, apresentando os conceitos e tecnologias que foram utilizadas no desenvolvimento. O capítulo 3 apresenta trabalhos relacionados. O capítulo 4 apresenta a visão geral da ferramenta e descreve todas as etapas do processo de mapeamento. O capítulo 5 apresenta a API da ferramenta para integração com outras aplicações. O capítulo 6 apresenta um estudo de caso, utilizando a ferramenta para realizar o processo de mapeamento para uma expressão SQL de consulta. O capítulo 7 é dedicado à conclusão.

2. Fundamentação Teórica

Este capítulo apresenta os conceitos e tecnologias empregadas no desenvolvimento desse trabalho.

2.1. XML

A tecnologia XML (*eXtensible Markup Language*) é projetada para o transporte e armazenamento de dados. Ela define um conjunto de regras, convenções ou diretrizes para criar linguagens de marcação que descrevem dados de praticamente qualquer tipo, de uma forma semi-estruturada. Uma linguagem de marcação é um mecanismo para identificar estruturas em um documento.

2.1.1. XML - Básico

Diferentemente de HTML, onde a semântica e o conjunto de *tags* é fixo, XML provê a facilidade de definição dessas *tags*, bem como o relacionamento estrutural entre elas. Toda a semântica de um documento XML é definida por aplicações processadoras ou por folhas de estilo associadas ao documento.

A Figura 1 mostra um exemplo simples de um documento XML.

```
1  <?xml version="1.0"?>
2
3  <livro isbn = "999-99999-9-X">
4    <titulo>Programação XML</titulo>
5
6    <autor>
7      <primeiroNome>João</primeiroNome >
8      <ultimoNome >Silva</ultimoNome>
9    </autor>
10
11   <capitulos>
12     <prefacio num = "1" paginas = "2">Bem Vindo</prefacio>
13     <prefacio num = "1" paginas = "4">XML Fácil</prefacio>
14     <prefacio num = "2" paginas = "2">Elementos XML</prefacio>
15     <apendice num = "1" paginas = "9">Entidades</apendice>
16   </capitulos>
17
18   <media tipo = "CD"/>
19 </livro>
```

Figura 1 – Exemplo de um documento XML bem formado

O documento inicia com a declaração XML (linha 1). A presença dessa instrução identifica o tipo do documento e sua versão. Todo documento XML deve obrigatoriamente conter exatamente um elemento raiz. Todas as linhas que precedem o elemento raiz representam o prólogo do documento XML. No exemplo, é definido o elemento raiz *livro* (linha 3). Os elementos *titulo*, *autor*, *capítulos* e *media* são considerados elementos filhos do elemento *livro*, porque estão aninhados internamente a ele. Termos como pai, filho e irmão são usados para descrever o relacionamento entre elementos. A linha 19 define o fim do elemento raiz.

As regras sintáticas de um documento XML são simples e lógicas. Todos os elementos de XML possuem uma *tag* (marca) de início e obrigatoriamente a *tag* de fim correspondente. Essas *tags* são *case sensitive*, ou seja, os processadores XML diferenciam letras maiúsculas de letras minúsculas. Dessa forma, as *tags* de abertura e de fechamento devem ser escritas de forma idêntica e estarem corretamente aninhadas. Elementos podem conter conteúdo em forma de texto e atributos, além de outros elementos. Atributos são representados pela forma *nome="valor"*. Os elementos *livro*, *prefacio* e *media* contém os atributos *isbn*, *num* e *paginas* e *tipo*, respectivamente.

Documentos XML são estruturados em forma de uma árvore invertida. Essa árvore inicia com o elemento raiz e ramifica-se através dos seus elementos filhos. Todos os elementos podem ter elementos filhos.

2.1.2. XML - Avançado

Algumas características importantes da linguagem XML são as seguintes:

- *Built-in entities*: a linguagem XML possui entidades pré-definidas para representar caracteres que possuem um significado especial. A Tabela 2 apresenta estas entidades.

Tabela 2 – Entidades pré-definidas da linguagem (XML)

<	<	menor que
>	>	maior que
&	&	e comercial
'	'	Apóstrofe
"	"	Aspas

- *Namespaces*: provê um método para evitar conflitos entre nomes de elementos.
- *CDATA*: textos inseridos em uma seção CDATA são ignorados pelo analisador XML. Dessa forma, tais seções podem conter texto, caracteres espaço em branco e caracteres reservados, como o parêntese à esquerda, parêntese à direita, apóstrofe e aspas, bem como os caracteres `<`, `>` e `&`. Um uso bastante comum para seção CDATA é o código de definição de scripts, como por exemplo *JavaScrip*, os quais frequentemente contém os caracteres `&`, `<` e `>`, criação de ‘ e “. Contudo, as seções CDATA não podem conter o texto `]]>`, pois o mesmo é utilizado para determinar a finalização da seção.
- *Encoding*: utilizado para evitar erros decorrentes da utilização de caracteres não pertencentes ao código ASCII.

2.2. XML Path (*XPath*)

XPath é uma linguagem de definição de expressões de caminho, baseada em *strings*, que fornece uma sintaxe para localizar, de forma eficaz, partes específicas de um documento XML.

Em XPath, um documento XML é conceitualmente visto como uma árvore, onde cada parte do documento é representado como um nodo. Alguns nodos podem conter outros nodos, formando uma estrutura hierarquizada pesquisável. Esses nodos estão classificados em sete tipos:

- Raiz: representa a raiz de um documento XML. Esse nodo existe somente no topo da árvore (e, conseqüentemente, no caminho de busca);
- Elemento: representa um elemento XML;
- Atributo: representa um atributo de um elemento;
- Texto: representa o conteúdo dos dados de caracteres de um elemento;
- Comentário: representa um comentário XML;
- Instrução de processamento: representa uma instrução de processamento XML;
- Espaço de nome: representa um espaço de nomes XML.

A árvore XPath possui um único nodo raiz. O nodo raiz e os nodos elementos possuem listas ordenadas de nodos filhos (descendentes). Todos os nodos, exceto o nodo raiz, possuem um nodo pai, e os nodos pais podem ter um número qualquer de

nodos filhos. Somente os tipos de nodos elementos, comentários, textos e instruções de processamento podem ser caracterizados como nodos filhos. O relacionamento entre um nodo pai e um nodo filho é “estar contido”. Nodos atributos e nodos espaço de nomes não são considerados filhos, pois são usados para fornecer informações descritivas sobre seus nodos pais.

XPath é baseada na noção de caminho de localização. Um caminho de localização é uma expressão, composta de passos de localização, que especifica como navegar uma árvore XPath de um nodo para outro. Cada passo é composto de um “eixo”, um “teste de nodo” e um “predicado” opcional.

- **Eixos:** a pesquisa através de um documento XML começa em um *nodo de contexto* na árvore XPath, e são realizadas em relação a esse nodo. O eixo indica quais nodos, em relação ao nodo contexto, devem ser incluídos na busca. O eixo também dita a ordem dos nodos no conjunto. Os eixos podem ser chamados de *eixos para frente*, quando selecionam nodos que se seguem ao nodo contexto na ordem de documento, e *eixos reversos*, quando selecionam nodos que precedem o nodo contexto na ordem de documento. A Tabela 3 resume os treze eixos XPath.

Tabela 3 – Eixos de XPath

Nome do eixo	Ordenação	Descrição
self	nenhuma	O próprio nodo contexto.
parent	reversa	O pai do nodo contexto, se existir.
child	para frente	Os filhos do nodo contexto, se existirem.
ancestor	reversa	Os ancestrais do nodo contexto, se existirem.
ancestor-or-self	reversa	Os ancestrais do nodo contexto e também ele próprio.
descendant	para frente	Os descendentes do nodo contexto.
descendant-or-self	para frente	Os descendentes do nodo contexto e também ele próprio.
following	para frente	Os nodos no documento XML que se seguem ao nodo contexto, não incluindo descendentes.
following-sibling	para frente	Os nodos irmãos que seguem ao nodo contexto.
preceding	reversa	Os nodos no documento XML precedendo o nodo contexto, não incluindo os ancestrais.
preceding-sibling	reversa	Os nodos irmãos que precedem o nodo contexto.
attribute	para frente	Os nodos atributos do nodo contexto.
namespace	para frente	Os nodos espaços de nomes do nodo contexto.

- **Testes de nodo:** o conjunto de nodos selecionados através do eixo é refinado com os testes de nodos, que dependem do tipo principal de nodo de um eixo para selecionar nodos em um caminho de localização. A Tabela 4 lista alguns tipos de testes de nodos.

Tabela 4 – Testes de nodos de XPath

Teste de nodo	Descrição
*	Seleciona todos os nodos do mesmo tipo principal de nodo.
node ()	Seleciona todos os nodos, independente de seu tipo.
text ()	Seleciona todos os nodos texto.
comment ()	Seleciona todos os nodos comentários.
processing-instruction	Seleciona todos os nodos instrução de processamento.
nome de nodo	Seleciona todos os nodos com o nome de nodo especificado.

- **Predicado:** o predicado é uma expressão booleana utilizada como parte de um caminho de localização para filtrar nodos na busca.

A XPath também fornece operadores e funções que podem ser combinados para formar expressões de caminhos de localização. As Tabelas 5 e 6 mostram operadores e algumas funções sobre conjuntos de nodos.

Tabela 5 – Operadores de XPath

Operadores sobre conjuntos de nodos	Descrição
barra vertical ()	Executa a união de dois conjuntos de nodos.
barra inclinada (/)	Separa passos de localização.
duas barras inclinadas (//)	Abreviação para o caminho de localização.

Tabela 6 – Funções de XPath

Funções sobre conjuntos de nodos	Descrição
last ()	Retorna o último nodo do conjunto de nodos.
position ()	Retorna o número da posição do nodo atual no conjunto de nodos que está sendo testado.
count (<i>conjunto de nodos</i>)	Retorna o número de nodos no conjunto de nodos.
id (<i>string</i>)	Retorna o nodo elemento cujo atributo ID é igual ao valor especificado pelo argumento string.
local-name (<i>conjunto de nodos</i>)	Retorna a parte local do nome expandido para o primeiro nodo no conjunto de nodos.
namespace-uri (<i>conjunto de nodos</i>)	Retorna o espaço de nome URI do nome expandido para o primeiro nodo no conjunto de nodos.
name (<i>conjunto de nodos</i>)	Retorna o nome qualificado para o primeiro nodo no conjunto de nodos.

A linguagem XPath é usada por outras tecnologias XML, tais como XML Pointer (XPointer), que fornece meios para “apontar” para informações dentro de um documento XML e a eXtensible Stylesheet Language (XSLT), que converte ou transforma documentos XML em outros formatos (HTML, XML ou texto). A linguagem XSLT é descrita a seguir.

2.3. XSL Transformation Language (XSLT)

XSLT transforma um documento XML em um documento diferente. O documento resultante pode ser XML, HTML, texto simples ou qualquer outro documento baseado em texto. No processo de transformação, XSLT utiliza XPath para definir partes do documento de origem que devem corresponder a um ou mais modelos predefinidos. Quando uma correspondência for encontrada, XSLT transformará o conteúdo correspondente do documento de origem no conteúdo desejado no documento de resultado. Com XSLT pode-se adicionar ou remover elementos e atributos para ou a partir do fluxo de saída. Pode-se também reorganizar e ordenar elementos, realizar testes e tomar decisões sobre quais elementos mostrar ou ocultar.

O documento XSLT é um documento XML com um elemento raiz *stylesheet* ou *transform*. Em uma transformação XSL, há duas árvores de nodos. A primeira árvore de

nodos é a *árvore de origem*. Os nodos nessa árvore correspondem ao documento XML original ao qual a transformação é aplicada. A segunda árvore de nodos é a *árvore de resultado*. A árvore de resultado contém todos os nodos produzidos pela transformação XSL. Essa árvore de resultado representa o documento produzido pela transformação.

A Tabela 7 lista os elementos utilizados para a construção de um documento XSLT:

Tabela 7 – Elementos de XSLT

<xsl:template>	Utilizado para construir modelos. O atributo <i>match</i> é utilizado para associar um modelo com um elemento XML. O atributo <i>match</i> também pode ser utilizado para definir um modelo para todo o documento XML. O valor do atributo <i>match</i> é uma expressão XPath
<xsl:value-of>	Extraí o valor de um elemento XML selecionado e adiciona ao fluxo de saída da transformação
<xsl:for-each>	Seleciona cada elemento XML de um determinado conjunto de nós
<xsl:sort>	Ordena a saída quando colocado dentro de um elemento <xsl:for-each>
<xsl:if>	Condicional para testar conteúdo XML
<xsl:choose>	Utilizado em conjunto com <xsl:when> e <xsl:otherwise> para expressar múltiplos testes condicionais
<xsl:apply-templates>	Aplica um modelo para o elemento atual ou aos elementos filhos do elemento atual. Adicionando um atributo de seleção, apenas elementos que correspondam ao valor do atributo serão processados. Esse atributo pode ser utilizado para especificar a ordem na qual os elementos filhos são processados
<xsl:copy>	Duplica nodos da árvore de origem na árvore de resultados
<xsl:import> <xsl:include>	Utilizados por um documento XSLT para importar outro documento XSLT
<xsl:variable>	Nome que pode ser vinculado a um valor, para processamento de informações

3. Trabalhos Relacionados

Alguns softwares realizam o mapeamento entre expressões SQL. Porém, a maioria deles se propõe a fazer apenas o mapeamento entre um SGBDR específico para outro SGBDR específico (SQLWays, 2011) (DBCONVERT, 2011).

O SwisSQL (SWISSQL, 2011) é uma ferramenta comercial que realiza as mesmas funções da ferramenta implementada nesse trabalho, sendo o trabalho relacionado mais próximo. Este capítulo compara as duas ferramentas, apresentando as vantagens e desvantagens entre eles.

A Figura 2 apresenta a interface gráfica do SwisSQL.

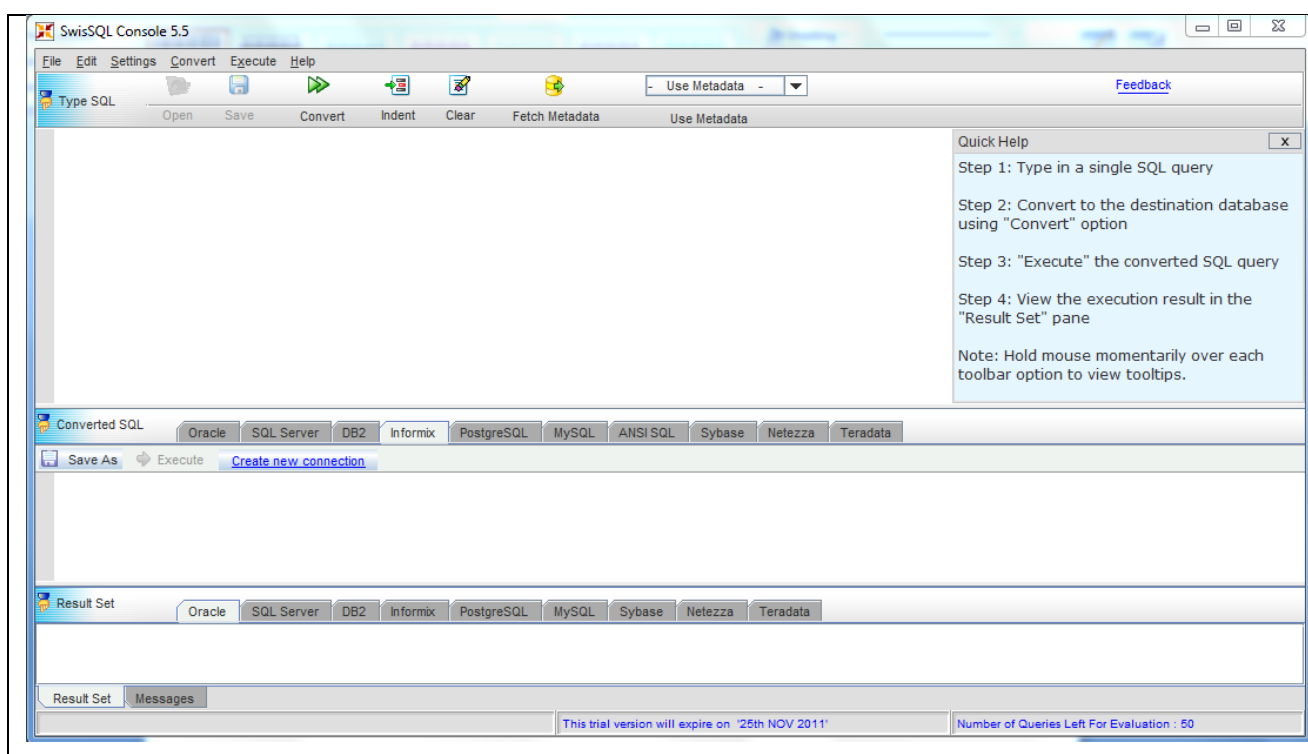


Figura 2 - SwisSQL

As principais diferenças entre a ferramenta proposta neste trabalho e o SwisSQL são:

- **SGBDRs suportados**

O SwisSQL suporta oito SGBDRs para mapeamento. São eles: Oracle, SQL Server, IBM DB2, MySQL, Sybase, PostgreSQL, Informix e Netezza.

A ferramenta implementada nesse trabalho suporta quatro SGBDRs: SQL Server, MySQL, PostgreSQL e Firebird.

- **Parâmetros de entrada**

Para iniciar o processo de mapeamento no SwisSQL, o usuário apenas informa a expressão SQL de entrada. O próprio software se encarrega de validar o comando informado.

Na ferramenta implementada são necessárias três informações de entrada: a expressão SQL de origem, qual o SGBDR que a expressão de origem pertence e o SGBDR de destino.

Embora o número de parâmetros de entrada seja maior, a ferramenta implementada processa mais rapidamente o resultado, pois o SwisSQL mapeia a expressão de entrada para todos os SGBDRs suportados, aumentando o seu custo de processamento. Além disso, nem sempre o usuário deseja mapear para todos os SGBDs destino suportados. Ele pode requerer o mapeamento para apenas um (1) SGBD específico. A ferramenta proposta permite essa flexibilidade.

- **Comandos suportados**

O SwisSQL suporta os comandos SELECT, INSERT, DELETE E UPDATE, enquanto que a ferramenta proposta suporta apenas o SELECT.

- **Execução de expressões mapeadas**

O SwisSQL permite que as expressões mapeadas possam ser executadas caso uma conexão com o SGBDR de destino esteja previamente configurado, enquanto que a ferramenta implementada não disponibiliza essa funcionalidade.

- **API**

Tanto SwisSQL quanto a ferramenta implementada podem ser distribuídos através de uma API para integração com outros softwares.

O que podemos notar com esses comparativos é que o SwisSQL, por ser um software mais maduro e estar já há um tempo no mercado, e conseqüentemente, agregando a cada ano mais funcionalidades, possui um conjunto maior de SGBDRs e comandos possíveis de serem mapeados. Porém, por ser uma ferramenta comercial, seu crescimento está associado às implementações da equipe de desenvolvimento da empresa.

A ferramenta implementada nesse trabalho é uma ferramenta gratuita, de código aberto, e que possibilita a inserção facilitada de novos SGBDRs e comandos para mapeamento. A inserção de novas funcionalidades, como já apresentado nesse trabalho,

é bastante simples e envolve nenhuma ou pouca alteração de código, apenas agregando novas regras de mapeamento ou arquivos de transformação.

4. Ferramenta Proposta

4.1. Visão Geral

Neste trabalho é proposta e implementada uma ferramenta de mapeamento de consultas SQL entre diferentes SGBDRs. O objetivo é, dada uma expressão SQL qualquer, de um determinado SGBDR, a ferramenta possa mapear e transformar para a expressão SQL equivalente de outro SGBDR. Devido à grande variedade de comandos SQL, a ferramenta implementa o mapeamento de comandos de consulta (*SELECT*), uma vez que são os tipos mais frequentes de operações que um SGBD executa e por mais apresentarem diferenças sintáticas.

Para que o mapeamento possa ser realizado, a ferramenta funciona com base em dois processos: **compilação** e **transformação**. O usuário informa a expressão SQL e a qual SGBDR ela pertence. Ele informa também o SGBDR para o qual se deseja que a expressão SQL de entrada seja transformada. A partir dessas informações de entrada, a ferramenta inicia o processo de compilação, analisando os aspectos léxicos e sintáticos da expressão de entrada. Em caso de erro, o sistema informa ao usuário a falha na compilação e indica a posição exata onde ocorreu esse erro. Se nenhum erro ocorrer é iniciada a análise semântica da compilação juntamente com a geração de código intermediário. O código intermediário gerado representa a expressão SQL de entrada estruturada no formato XML. A ferramenta então seleciona o conjunto de regras de transformação e aplica ao código intermediário gerado enviando ao fluxo de saída a expressão SQL do SGBDR de destino.

A Figura 3 mostra a arquitetura geral do processo de mapeamento.

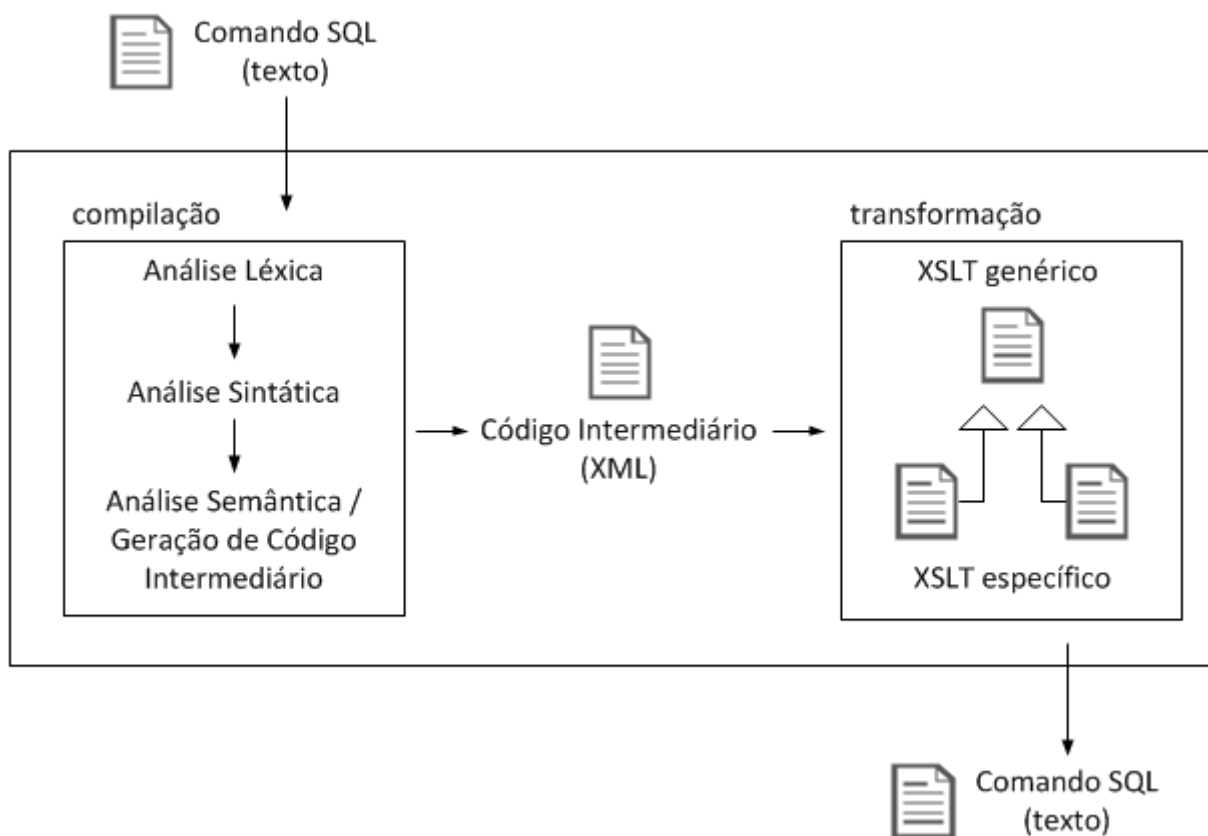


Figura 3 – Esquema geral do processo de mapeamento entre comandos SQL

Para o processo de transformação foi adotada a tecnologia XSLT. A variedade de operadores e funções associadas ao padrão XML para a estruturação de documentos facilita a definição das regras de correspondência e diminui o tempo de desenvolvimento, se comparada ao mesmo tipo de implementação utilizando uma linguagem de alto nível (e.g. Java, C++). Outras vantagens da tecnologia XSLT são a possibilidade da geração de resultado para diferentes formatos (e.g. HTML, XML, texto puro) e tecnologias disponíveis para integração com diferentes linguagens (e.g. Xalan para Java). Em contrapartida, XSLT possui a limitação de apenas receber em seu fluxo de entrada, para processamento, arquivos no formato XML. Para contornar esse problema, a ferramenta, através do processo de compilação, valida e estrutura a expressão SQL de entrada no formato XML.

4.2. Processo de Mapeamento

O processo de mapeamento inicia quando o usuário informa à ferramenta a expressão SQL de entrada, o SGBDR de entrada e o SGBDR de saída, para o qual a expressão de entrada deve ser mapeada. Com base nessas informações, a ferramenta

inicia o processo de compilação, que analisa os aspectos léxicos e sintáticos e aplica as ações semânticas na expressão de entrada, bem como a geração do código intermediário.

4.2.1. Compilação

Para o processo de compilação, foi utilizada a ferramenta GALS (Gerador de Analisadores Léxicos e Sintáticos) (GESSER, 2002). Nela foi inserida uma gramática específica para cada um dos possíveis SGBDRs de entrada. O presente trabalho implementou regras de mapeamento para quatro SGBDRs: Microsoft SQL Server, MySQL, PostgreSQL e Firebird. Entretanto, outros SGBDRs também podem ser integrados à ferramenta, que é facilmente extensível por definir regras de mapeamento em gramáticas e arquivos XSL independentes. As Tabelas 8, 9, 10 e 11 mostram as gramáticas definidas para cada um desses SGBDRs, respectivamente.

Tabela 8 – Gramática do Microsoft SQL Server

<sql> ::=	<select_def> #999 <insert_def> <update_def> <delete_def> ;
<select_def> ::=	<select> <from> <where> ;
<select> ::=	select #100 <distinct> <top> <column> ;
<distinct> ::=	distinct #111 ^ #112 ;
<top> ::=	top int_value #104 ^ ;
<column> ::=	id #101 <alias> <columnlist> <concat> <alias> <columnlist> ;
<columnlist> ::=	"," id #105 <alias> <columnlist> ^ ;
<alias> ::=	as id #113 ^ ;
<concat> ::=	"(" id #118 "+" id #119 ")" ;
<from> ::=	from #102 <table> ;
<table> ::=	id #103 <alias> <tablelist> ;
<tablelist> ::=	"," id #106 <alias> <tablelist> inner join id #115 on #116 id #116 "." #116 id #116 <operator> #116 id #116 "." #116 id #117 <tablelist> ^ ;
<where> ::=	where #107 <conditional> ^ ;
<conditional> ::=	<antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ;
<antecedent>	id ;
<operator> ::=	"=" "<" ">" ">=" "<" "<=" ;
<consequent> ::=	id int_value ;
<conditionallist> ::=	<logical_operator> #114 <antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ^ ;
<logical_operator> ::=	and or ;

Tabela 9 – Gramática do MySQL

<sql> ::=	<select_def> #999 <insert_def> <update_def> <delete_def> ;
<select_def> ::=	<select> <from> <where> <limit> ;
<select> ::=	select #100 <distinct> <column> ;
<distinct> ::=	distinct #111 ^ #112 ;
<limit> ::=	limit int_value #104 ^ ;
<column> ::=	id #101 <alias> <columnlist> <concat> <alias> <columnlist> ;
<columnlist> ::=	"," id #105 <alias> <columnlist> ^ ;
<alias> ::=	as id #113 ^ ;
<concat> ::=	concat "(" id #118 "," id #119 ")" ;
<from> ::=	from #102 <table> ;
<table> ::=	id #103 <alias> <tablelist> ;
<tablelist> ::=	"," id #106 <alias> <tablelist> inner join id #115 on #116 id #116 "." #116 id #116 <operator> #116 id #116 "." #116 id #117 <tablelist> ^ ;
<where> ::=	where #107 <conditional> ^ ;
<conditional> ::=	<antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ;
<antecedent>	id ;
<operator> ::=	"=" "<" ">" ">=" "<" "<=" ;
<consequent> ::=	id int_value ;
<conditionallist> ::=	<logical_operator> #114 <antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ^ ;
<logical_operator> ::=	and or ;

Tabela 10 – Gramática do PostgreSQL

<sql> ::=	<select_def> #999 <insert_def> <update_def> <delete_def> ;
<select_def> ::=	<select> <from> <where> <limit> ;
<select> ::=	select #100 <distinct> <column> ;
<distinct> ::=	distinct #111 ^ #112 ;
<limit> ::=	limit int_value #104 ^ ;
<column> ::=	id #101 <alias> <columnlist> <concat> <alias> <columnlist> ;
<columnlist> ::=	"," id #105 <alias> <columnlist> ^ ;
<alias> ::=	as id #113 ^ ;
<concat> ::=	"(" id #118 " " id #119 ")" ;
<from> ::=	from #102 <table> ;
<table> ::=	id #103 <alias> <tablelist> ;
<tablelist> ::=	"," id #106 <alias> <tablelist> inner join id #115 on #116 id #116 "." #116 id #116 <operator> #116 id #116 "." #116 id #117 <tablelist> ^ ;
<where> ::=	where #107 <conditional> ^ ;
<conditional> ::=	<antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ;
<antecedent>	id ;
<operator> ::=	"=" "<" ">" ">=" "<" "<=" ;
<consequent> ::=	id int_value ;
<conditionallist> ::=	<logical_operator> #114 <antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ^ ;
<logical_operator> ::=	and or ;

Tabela 11 – Gramática do Firebird

<sql> ::=	<select_def> #999 <insert_def> <update_def> <delete_def> ;
<select_def> ::=	<select> <from> <where> ;
<select> ::=	select #100 <distinct> <first> <column> ;
<distinct> ::=	distinct #111 ^ #112 ;
<first> ::=	first int_value #104 ^ ;
<column> ::=	id #101 <alias> <columnlist> <concat> <alias> <columnlist> ;
<columnlist> ::=	"," id #105 <alias> <columnlist> ^ ;
<alias> ::=	as id #113 ^ ;
<concat> ::=	"(" id #118 " " id #119 ")" ;
<from> ::=	from #102 <table> ;
<table> ::=	id #103 <alias> <tablelist> ;
<tablelist> ::=	"," id #106 <alias> <tablelist> inner join id #115 on #116 id #116 "." #116 id #116 <operator> #116 id #116 "." #116 id #117 <tablelist> ^ ;
<where> ::=	where #107 <conditional> ^ ;
<conditional> ::=	<antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ;
<antecedent>	id ;
<operator> ::=	"=" "<" ">" ">=" "<" "<=" ;
<consequent> ::=	id int_value ;
<conditionallist> ::=	<logical_operator> #114 <antecedent> #108 <operator> #109 <consequent> #110 <conditionallist> ^ ;
<logical_operator> ::=	and or ;

Nas regras gramaticais foram inseridos valores numéricos antecidos pelo caracter especial “#”. Esses valores indicam ao analisador semântico a regra semântica a ser processada. Paralelamente ao processo de análise semântica é gerado o código intermediário, no formato XML. As Tabelas 12, 13, 14 e 15 mostram as ações semânticas implementadas para cada um dos SGBDRs.

Tabela 12 – Ações semânticas para o Microsoft SQL Server

#100	Cria elemento <select> , filho do elemento raiz <sql> .
#101	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#102	Cria elemento <from> , filho do elemento raiz <sql> .
#103	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#104	Cria elemento <rows> , filho do elemento raiz <sql> , com o texto de <i>int_value</i> .
#105	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#106	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#107	Cria elemento <where> , filho do elemento raiz <sql> ; Cria elemento <conditional> , filho do elemento <where> ; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#108	Cria elemento <antecedent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#109	Cria atributo operator , com o valor do operador para o elemento <i>currentConditional</i> .
#110	Cria elemento <consequent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#111	Cria atributo distinct , com o valor <i>true</i> , para o elemento <select> .
#112	Cria atributo distinct , com o valor <i>false</i> , para o elemento <select> .
#113	Cria atributo alias , com o valor de <i>id</i> , para o elemento <i>currentParam</i> .
#114	Cria elemento <conditional> , filho do elemento <where> ; Cria atributo logical_operator , com o valor do operador lógico; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#115	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “inner join”; Atribui o elemento <param> criado a <i>currentParam</i> ; Inicializa variável de concatenação para suffix de <i>currentParam</i> .
#116	Concatena valor atual à variável de concatenação.
#117	Concatena valor atual à variável de concatenação; Cria atributo suffix para o elemento <i>currentParam</i> , com o valor da variável de concatenação.
#118	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “1”; Atribui o elemento <param> criado a <i>currentParam</i> .
#119	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “2”; Atribui o elemento <param> criado a <i>currentParam</i> .

Tabela 13 – Ações semânticas para o MySQL

#100	Cria elemento <select> , filho do elemento raiz <sql> .
#101	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#102	Cria elemento <from> , filho do elemento raiz <sql> .
#103	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#104	Cria elemento <rows> , filho do elemento raiz <sql> , com o texto de <i>int_value</i> .
#105	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#106	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#107	Cria elemento <where> , filho do elemento raiz <sql> ; Cria elemento <conditional> , filho do elemento <where> ; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#108	Cria elemento <antecedent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#109	Cria atributo operator , com o valor do operador para o elemento <i>currentConditional</i> .
#110	Cria elemento <consequent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#111	Cria atributo distinct , com o valor <i>true</i> , para o elemento <select> .
#112	Cria atributo distinct , com o valor <i>false</i> , para o elemento <select> .
#113	Cria atributo alias , com o valor de <i>id</i> , para o elemento <i>currentParam</i> .
#114	Cria elemento <conditional> , filho do elemento <where> ; Cria atributo logical_operator , com o valor do operador lógico; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#115	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “inner join”; Atribui o elemento <param> criado a <i>currentParam</i> ; Inicializa variável de concatenação para suffix de <i>currentParam</i> .
#116	Concatena valor atual à variável de concatenação.
#117	Concatena valor atual à variável de concatenação; Cria atributo suffix para o elemento <i>currentParam</i> , com o valor da variável de concatenação.
#118	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “1”; Atribui o elemento <param> criado a <i>currentParam</i> .
#119	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “2”; Atribui o elemento <param> criado a <i>currentParam</i> .

Tabela 14 – Ações semânticas para o PostgreSQL

#100	Cria elemento <select> , filho do elemento raiz <sql> .
#101	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#102	Cria elemento <from> , filho do elemento raiz <sql> .
#103	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#104	Cria elemento <rows> , filho do elemento raiz <sql> , com o texto de <i>int_value</i> .
#105	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#106	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#107	Cria elemento <where> , filho do elemento raiz <sql> ; Cria elemento <conditional> , filho do elemento <where> ; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#108	Cria elemento <antecedent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#109	Cria atributo operator , com o valor do operador para o elemento <i>currentConditional</i> .
#110	Cria elemento <consequent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#111	Cria atributo distinct , com o valor <i>true</i> , para o elemento <select> .
#112	Cria atributo distinct , com o valor <i>false</i> , para o elemento <select> .
#113	Cria atributo alias , com o valor de <i>id</i> , para o elemento <i>currentParam</i> .
#114	Cria elemento <conditional> , filho do elemento <where> ; Cria atributo logical_operator , com o valor do operador lógico; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#115	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “inner join”; Atribui o elemento <param> criado a <i>currentParam</i> ; Inicializa variável de concatenação para suffix de <i>currentParam</i> .
#116	Concatena valor atual à variável de concatenação.
#117	Concatena valor atual à variável de concatenação; Cria atributo suffix para o elemento <i>currentParam</i> , com o valor da variável de concatenação.
#118	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “1”; Atribui o elemento <param> criado a <i>currentParam</i> .
#119	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “2”; Atribui o elemento <param> criado a <i>currentParam</i> .

Tabela 15 – Ações semânticas para o Firebird

#100	Cria elemento <select> , filho do elemento raiz <sql> .
#101	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#102	Cria elemento <from> , filho do elemento raiz <sql> .
#103	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Atribui o elemento <param> criado a <i>currentParam</i> .
#104	Cria elemento <rows> , filho do elemento raiz <sql> , com o texto de <i>int_value</i> .
#105	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#106	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “,” (vírgula); Atribui o elemento <param> criado a <i>currentParam</i> .
#107	Cria elemento <where> , filho do elemento raiz <sql> ; Cria elemento <conditional> , filho do elemento <where> ; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#108	Cria elemento <antecedent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#109	Cria atributo operator , com o valor do operador para o elemento <i>currentConditional</i> .
#110	Cria elemento <consequent> , filho do elemento <i>currentConditional</i> , com o texto de <i>id</i> .
#111	Cria atributo distinct , com o valor <i>true</i> , para o elemento <select> .
#112	Cria atributo distinct , com o valor <i>false</i> , para o elemento <select> .
#113	Cria atributo alias , com o valor de <i>id</i> , para o elemento <i>currentParam</i> .
#114	Cria elemento <conditional> , filho do elemento <where> ; Cria atributo logical_operator , com o valor do operador lógico; Atribui o elemento <conditional> criado a <i>currentConditional</i> .
#115	Cria elemento <param> , filho do elemento <from> , com o texto de <i>id</i> ; Cria atributo prefix com o valor “inner join”; Atribui o elemento <param> criado a <i>currentParam</i> ; Inicializa variável de concatenação para suffix de <i>currentParam</i> .
#116	Concatena valor atual à variável de concatenação.
#117	Concatena valor atual à variável de concatenação; Cria atributo suffix para o elemento <i>currentParam</i> , com o valor da variável de concatenação.
#118	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “1”; Atribui o elemento <param> criado a <i>currentParam</i> .
#119	Cria elemento <param> , filho do elemento <select> , com o texto de <i>id</i> ; Cria atributo <i>concatIndex</i> com o valor “2”; Atribui o elemento <param> criado a <i>currentParam</i> .

O GALS gera, de forma automatizada, todas as classes necessárias para o processo de análise léxica e sintática. A classe de análise semântica é gerada em branco. Nela então são definidos os métodos que implementam cada uma das ações semânticas separadamente.

Conforme comentado anteriormente, a análise semântica gera o código intermediário no formato XML. Levando-se em consideração que o mapeamento dificilmente envolverá grande quantidade de código a ser mapeado, optou-se pelo desempenho à economia de memória. Dessa forma, o código intermediário é gerado utilizando a biblioteca JDOM (JDOM, 2011), que cria uma estrutura em árvore na memória com o conteúdo do XML do código intermediário. Essa estrutura em memória é enviada ao fluxo de entrada do processo de transformação.

4.2.2 Transformação

O código intermediário gerado no processo de compilação possui uma estrutura única, em formato XML, que serve como informação de entrada para regras que transformam o comando SQL de entrada no comando SQL de saída. A ferramenta define essas regras através de arquivos XSL. Essas regras são responsáveis por encontrar as correspondências para cada uma das *tags* encontradas no XML do código intermediário e processar a informação enviando o resultado para o fluxo de saída no formato de texto puro, que corresponde à expressão SQL de saída.

Os arquivos XSL são específicos para cada um dos SGBDRs possíveis de serem mapeados, porém, são especializados através de um arquivo XSL genérico, que define regras de mapeamento comuns a todos eles, bem como variáveis necessárias para seus processamentos. A Figura 4 mostra a estrutura do arquivo XSL genérico.

```

<?xml version = "1.0"?>
<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:xalan="http://xml.apache.org/xalan">

  <xsl:output method="xml" indent="yes" encoding="UTF-8" xalan:indent-amount="5"
    omit-xml-declaration="yes"/>

  <!-- Declaraco de variáveis globais -->
  <xsl:variable name="rows" select="/sql/rows"/>

  <xsl:template match = "select">SELECT
    <xsl:if test="@distinct = 'true'">
      DISTINCT
    </xsl:if>
    <xsl:for-each select="param">
      <xsl:value-of select="@prefix"/>
      <xsl:value-of select="."/>
      <xsl:value-of select="@suffix"/>
      <xsl:choose>
        <xsl:when test="@alias">
          AS <xsl:value-of select="@alias"/>
        </xsl:when>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match = "from"> FROM
    <xsl:for-each select="param">
      <xsl:value-of select="@prefix"/>
      <xsl:value-of select="."/>
      <xsl:value-of select="@suffix"/>
      <xsl:choose>
        <xsl:when test="@alias">
          AS <xsl:value-of select="@alias"/>
        </xsl:when>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match = "where"> WHERE
    <xsl:for-each select="conditional">
      <xsl:choose>
        <xsl:when test="@logical_operator">
          <xsl:value-of select="@logical_operator"/>
        </xsl:when>
      </xsl:choose>
      <xsl:value-of select="./antecedent"/>
      <xsl:value-of select="@operator"/>
      <xsl:value-of select="./consequent"/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>

```

Figura 4 – Arquivo XSL genérico

O XSL genérico possui regras básicas para o mapeamento direto de uma expressão de consulta SQL padrão. Se em um processo de transformação forem aplicadas apenas as regras de correspondência do XSL genérico, a expressão SQL de saída será idêntica à expressão SQL de entrada. Nesse arquivo são definidas as regras para:

- Armazenamento do valor da limitação de *tuplas* nos resultados de pesquisa através da definição da variável global *rows*;
- Comando *SELECT* com uma ou mais colunas;
- Comando *SELECT* com ou sem definição de *alias* para as colunas;
- Comando *SELECT* com ou sem a definição de distinção de *tuplas* nos resultados de pesquisa;
- Comando *FROM* definindo uma ou mais tabelas;
- Comando *FROM* com ou sem definição de *alias* para as tabelas;
- Comando *WHERE* definindo uma ou mais condicionais;
- Operadores *AND* e *OR* para condicionais do comando *WHERE*.

Os arquivos XSL específicos para cada SGBDR importam o arquivo XSL genérico. Dessa forma, as regras de correspondência definidas genericamente podem ser sobrescritas pelos arquivos específicos, modificando ou agregando valores à expressão SQL de entrada especificada. As Figuras 5, 6, 7 e 8 mostram as estruturas dos arquivos XSL específicos de cada um dos SGBDRs.

```

<?xml version = "1.0"?>

<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:xalan="http://xml.apache.org/xalan">

  <xsl:import href = "generic/Sql.xsl"/>
  <xsl:output method="text" indent="no" encoding="UTF-8"/>

  <xsl:template match = "sql">
    <xsl:apply-templates select="select"/>
    <xsl:apply-templates select="from"/>
    <xsl:apply-templates select="where"/>
  </xsl:template>

  <xsl:template match = "select">SELECT
    <xsl:if test="@distinct = 'true'">
      DISTINCT
    </xsl:if>
    <xsl:if test="$rows > -1">TOP
      <xsl:value-of select="$rows"/>
    </xsl:if>
    <xsl:for-each select="param">
      <xsl:choose>
        <xsl:when test="@concatIndex = 1"></xsl:when>
        <xsl:when test="@concatIndex = 2">+</xsl:when>
      </xsl:choose>
      <xsl:value-of select="@prefix"/>
      <xsl:value-of select="."/>
      <xsl:value-of select="@suffix"/>
      <xsl:choose>
        <xsl:when test="@concatIndex = 2"></xsl:when>
      </xsl:choose>
      <xsl:choose>
        <xsl:when test="@alias">
          AS <xsl:value-of select="@alias"/>
        </xsl:when>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>

```

Figura 5 – Arquivo XSL específico para o Microsoft SQL Server

```

<?xml version = "1.0"?>

<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:xalan="http://xml.apache.org/xalan">

  <xsl:import href = "generic/Sql.xsl"/>
  <xsl:output method="text" indent="no" encoding="UTF-8"/>

  <xsl:template match = "sql">
    <xsl:apply-templates select="select"/>
    <xsl:apply-templates select="from"/>
    <xsl:apply-templates select="where"/>
    <xsl:apply-templates select="rows"/>
  </xsl:template>

  <xsl:template match = "select">SELECT
    <xsl:if test="@distinct = 'true'">
      DISTINCT
    </xsl:if>
    <xsl:for-each select="param">
      <xsl:choose>
        <xsl:when test="@concatIndex = 1">CONCAT(</xsl:when>
          <xsl:when test="@concatIndex = 2">,</xsl:when>
        </xsl:choose>
        <xsl:value-of select="@prefix"/>
        <xsl:value-of select="."/>
        <xsl:value-of select="@suffix"/>
        <xsl:choose>
          <xsl:when test="@concatIndex = 2">)</xsl:when>
        </xsl:choose>
      <xsl:choose>
        <xsl:when test="@alias">
          AS <xsl:value-of select="@alias"/>
        </xsl:when>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match = "rows">
    <xsl:if test="$rows > -1"> LIMIT
      <xsl:value-of select="$rows"/>
    </xsl:if>
  </xsl:template>

</xsl:stylesheet>

```

Figura 6 – Arquivo XSL específico para o MySQL

```

<?xml version = "1.0"?>

<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:xalan="http://xml.apache.org/xalan">

  <xsl:import href = "generic/Sql.xsl"/>
  <xsl:output method="text" indent="no" encoding="UTF-8"/>

  <xsl:template match = "sql">
    <xsl:apply-templates select="select"/>
    <xsl:apply-templates select="from"/>
    <xsl:apply-templates select="where"/>
    <xsl:apply-templates select="rows"/>
  </xsl:template>

  <xsl:template match = "select">SELECT
    <xsl:if test="@distinct = 'true'">
      DISTINCT
    </xsl:if>
    <xsl:for-each select="param">
      <xsl:choose>
        <xsl:when test="@concatIndex = 1"></xsl:when>
        <xsl:when test="@concatIndex = 2">||</xsl:when>
      </xsl:choose>
      <xsl:value-of select="@prefix"/>
      <xsl:value-of select="."/>
      <xsl:value-of select="@suffix"/>
      <xsl:choose>
        <xsl:when test="@concatIndex = 2"></xsl:when>
      </xsl:choose>
      <xsl:choose>
        <xsl:when test="@alias">
          AS <xsl:value-of select="@alias"/>
        </xsl:when>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match = "rows">
    <xsl:if test="$rows > -1"> LIMIT
      <xsl:value-of select="$rows"/>
    </xsl:if>
  </xsl:template>

</xsl:stylesheet>

```

Figura 7 – Arquivo XSL específico para o PostgreSQL

```

<?xml version = "1.0"?>

<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  xmlns:xalan="http://xml.apache.org/xalan">

  <xsl:import href = "generic/Sql.xsl"/>
  <xsl:output method="text" indent="no" encoding="UTF-8"/>

  <xsl:template match = "sql">
    <xsl:apply-templates select="select"/>
    <xsl:apply-templates select="from"/>
    <xsl:apply-templates select="where"/>
  </xsl:template>

  <xsl:template match = "select">SELECT
    <xsl:if test="$rows > -1">FIRST
      <xsl:value-of select="$rows"/>
    </xsl:if>
    <xsl:if test="@distinct = 'true'">
      DISTINCT
    </xsl:if>
    <xsl:for-each select="param">
      <xsl:choose>
        <xsl:when test="@concatIndex = 1"></xsl:when>
        <xsl:when test="@concatIndex = 2">||</xsl:when>
      </xsl:choose>
      <xsl:value-of select="@prefix"/>
      <xsl:value-of select="."/>
      <xsl:value-of select="@suffix"/>
      <xsl:choose>
        <xsl:when test="@concatIndex = 2"></xsl:when>
      </xsl:choose>
      <xsl:choose>
        <xsl:when test="@alias">
          AS <xsl:value-of select="@alias"/>
        </xsl:when>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>

```

Figura 8 – Arquivo XSL específico para o Firebird

A sequência das regras de correspondência são definidas através do elemento `<xsl:template>`, em cada um dos arquivos XSL específicos, aplicado à *tag* raiz do XML que representa o código intermediário. Quando um valor é atribuído a “match” de `<xsl:template>`, o processador XSLT procura a regra de correspondência no próprio documento. Se a mesma não for encontrada, a procura é feita no arquivo XSL genérico importado.

Os arquivos XSL são responsáveis pela correspondência e manipulação de cada uma das *tags* do arquivo XML que representa a expressão SQL de entrada. Paralelamente a esse processo, é formatada a expressão SQL correspondente no formato de texto simples. Esse texto é então enviado ao fluxo de saída.

5. API

A API é o mecanismo de análise e mapeamento de consultas SQL que representa o núcleo do funcionamento da ferramenta. Ela permite que aplicações baseadas em Java possam trabalhar com vários SGBDRs sem a necessidade de portar consultas SQL específicas de um SGBD para outro, convertendo instruções *SELECT* em tempo real entre os SGBDs suportados. A arquitetura geral da API é ilustrada na Figura 9.

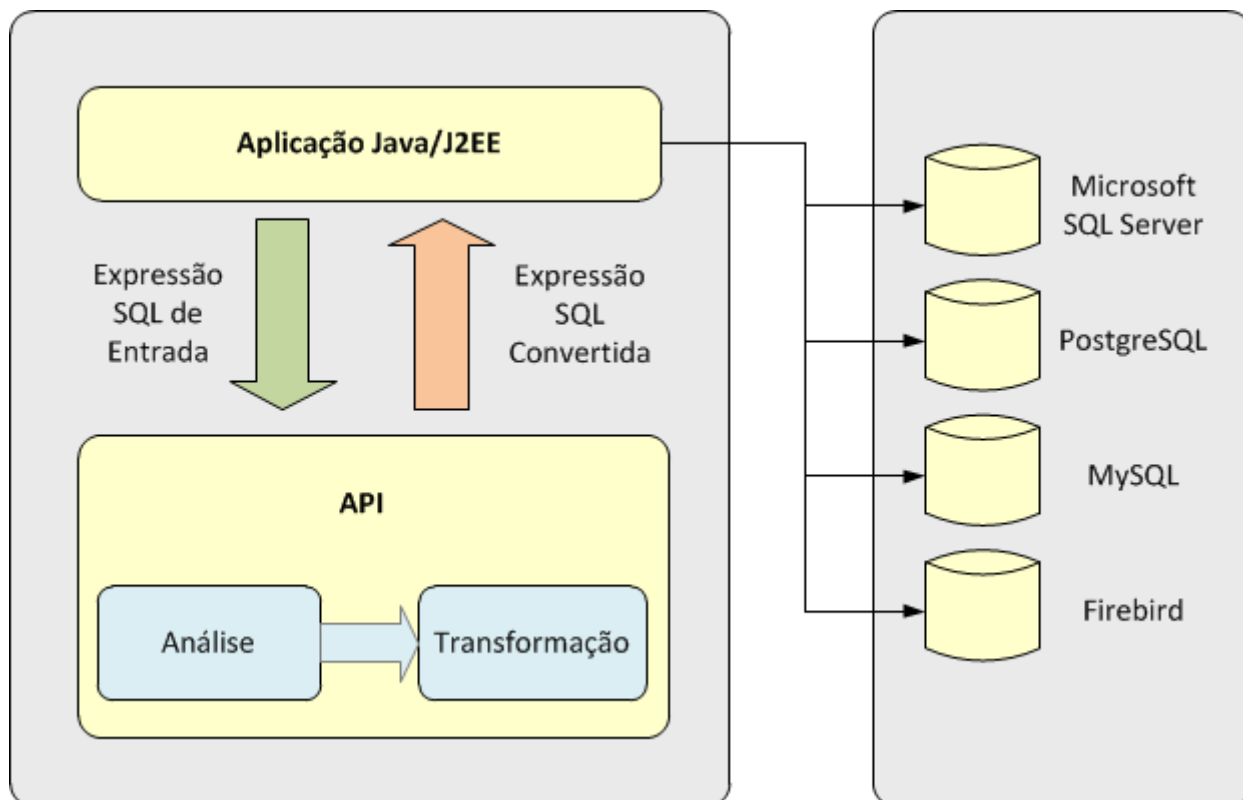


Figura 9 – API: arquitetura geral

A interação entre uma aplicação Java/J2EE e a API é bastante simples. A Figura 10 mostra o diagrama de sequência com as classes, métodos e parâmetros envolvidos no processo.

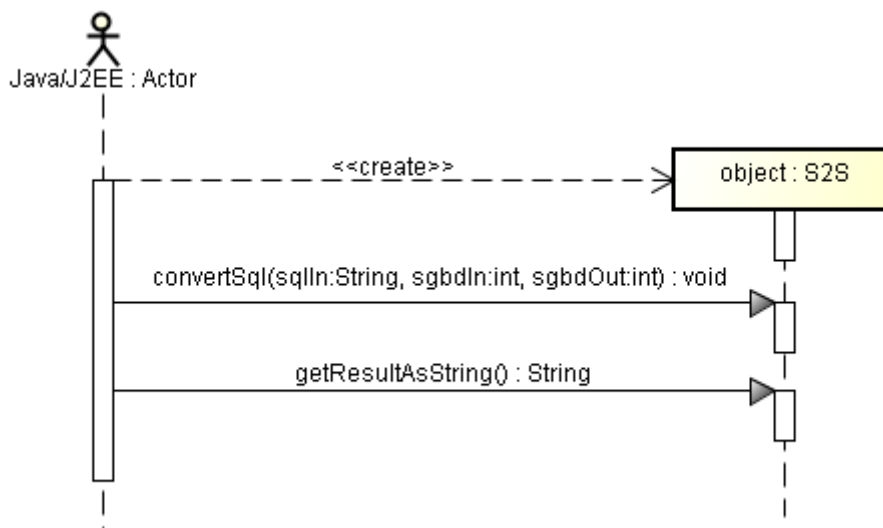


Figura 10 – API: diagrama de sequência

Uma aplicação Java/J2EE, representada na forma do *Actor* na Figura X, instancia um objeto da classe S2S que representa a interface de interação com a API. Através do objeto instanciado o método *convertSql* é invocado, recebendo como parâmetros o comando de consulta SQL a ser mapeado e os SGBDRs de origem e destino. Por fim, o método *getResultAsString* é executado a fim de se obter o resultado do mapeamento na forma de texto puro.

Através dos parâmetros de entrada do método de conversão do comando de consulta SQL, a API carrega dinamicamente as classes e arquivos específicos dos SGBDRs envolvidos no processo de mapeamento. A Figura 11 mostra o diagrama com as principais classes da API.

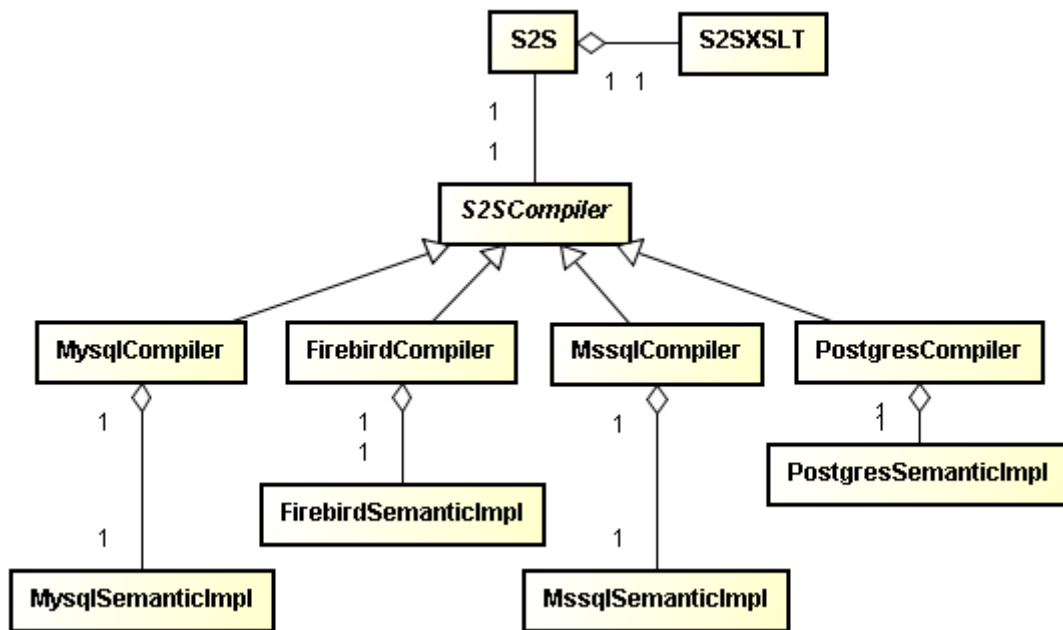


Figura 11 – API: diagrama de classes

Cada SGBDR define sua própria classe de compilação, representadas pelas classes `MysqlCompiler`, `FirebirdCompiler`, `MssqlCompiler` e `PostgresCompiler`. Também para cada SGBDR é definida uma classe que implementará suas próprias regras semânticas. Essas classes são representadas por `MysqlSemanticImpl`, `FirebirdSemanticImpl`, `MssqlSemanticImpl` e `PostgresSemanticImpl`. Quando o processo de compilação é finalizado com sucesso, um objeto da classe `S2SXLST` é instanciado e através dele as regras de mapeamento XSLT são aplicadas e o resultado é informado.

A API fornece uma maneira simples e fácil de realizar a análise e mapeamento das consultas SQL diretamente dentro de aplicativos Java/J2EE. Esse recurso garante que os aplicativos estejam no controle de quando uma consulta SQL precisa ser convertida para um dialeto diferente.

6. Estudo de Caso

Esta seção apresenta um exemplo da execução do mapeamento de um comando SQL de consulta entre 2 SGBDRs através do uso da ferramenta. O estudo de caso é breve, pois a interface gráfica é simples, assim como a interação com o usuário. A Figura 12 mostra a tela inicial da ferramenta com a expressão SQL de entrada, o SGBDR de entrada e o SGBDR de saída, para o qual se deseja mapear a expressão informada.

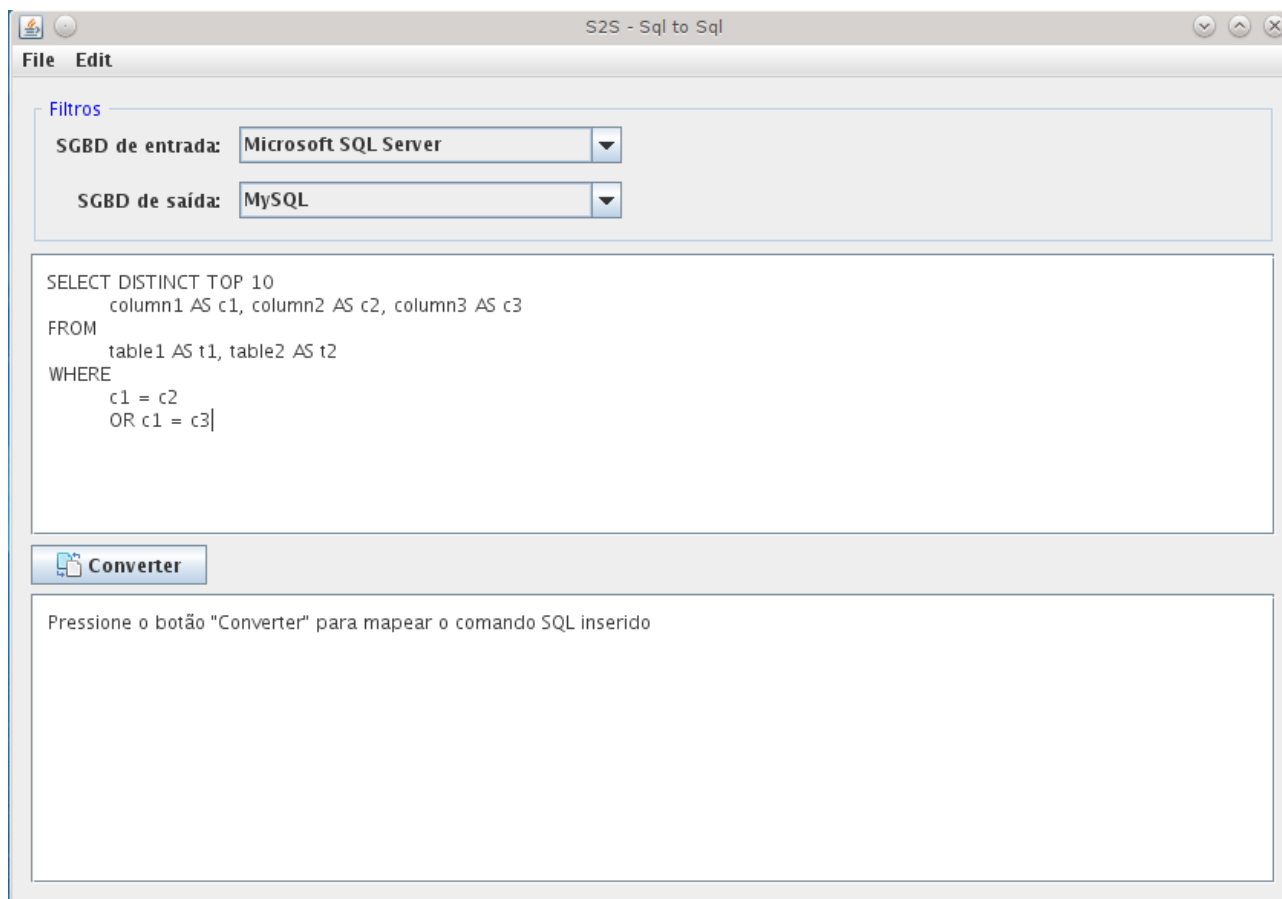


Figura 12 – Execução da ferramenta com os dados de entrada

Quando o botão “Converter” é acionado, a compilação é iniciada. Após o processo de análise léxica e sintática, a análise semântica gera o código intermediário. A Figura 13 mostra a estrutura da árvore XML que representa esse código e que será enviada ao fluxo de entrada do processo de transformação.

```

<?xml version="1.0" encoding="UTF-8"?>
<sql>
  <rows>10</rows>
  <select distinct="true">
    <param alias="c1">column1</param>
    <param prefix="," alias="c2">column2</param>
    <param prefix="," alias="c3">column3</param>
  </select>
  <from>
    <param alias="t1">table1</param>
    <param prefix="," alias="t2">table2</param>
  </from>
  <where>
    <conditional operator="=">
      <antecedent>c1</antecedent>
      <consequent>c2</consequent>
    </conditional>
    <conditional logical_operator=" OR " operator="=">
      <antecedent>c1</antecedent>
      <consequent>c3</consequent>
    </conditional>
  </where>
</sql>

```

Figura 13 – Estrutura do código intermediário no formato XML

O elemento `<rows>` é a variável global que especifica a quantidade de tuplas a serem retornadas da consulta. Esse elemento nem sempre está presente na estrutura.

O elemento `<select>` possui o atributo `distinct` com o valor `true`, que define como verdadeira a opção de retornar apenas as tuplas não repetidas. Esse atributo sempre está presente na estrutura e pode assumir os valores `true` ou `false`.

Os elementos `<param>`, filhos dos elementos `<select>`, representam as colunas especificadas na expressão SQL de entrada. Pelo menos um elemento `<param>` de `<select>` estará presente na estrutura. Eles podem ter os atributos `alias` (define um `alias` para a coluna), `prefix` (valores que aparecerão antes da coluna) e `suffix` (valores que aparecerão após a coluna).

Os elementos `<param>`, filhos do elemento `<from>`, seguem a mesma lógica dos elementos `<param>` filhos de `<select>`, e também pelo menos um elemento `<param>` estará presente na estrutura.

Os elementos *<conditional>* definem os operadores dos elementos *<antecedent>* e *<consequent>*, operandos da expressão SQL, através do atributo *operator*. O atributo *logical_operator* define o operador lógico entre esses condicionais. O elemento *<where>* pode não estar presente na estrutura, porém, caso esteja, ao menos um elemento *<conditional>* deverá estar.

O processo de transformação é então iniciado e a ferramenta busca o arquivo XSL específico para o SGBDR MySQL, definido como parâmetro e para o qual se deseja mapear a expressão SQL de entrada. Ao final desse processo, é enviada à interface da ferramenta a expressão SQL de saída correspondente, conforme mostra a Figura 14.

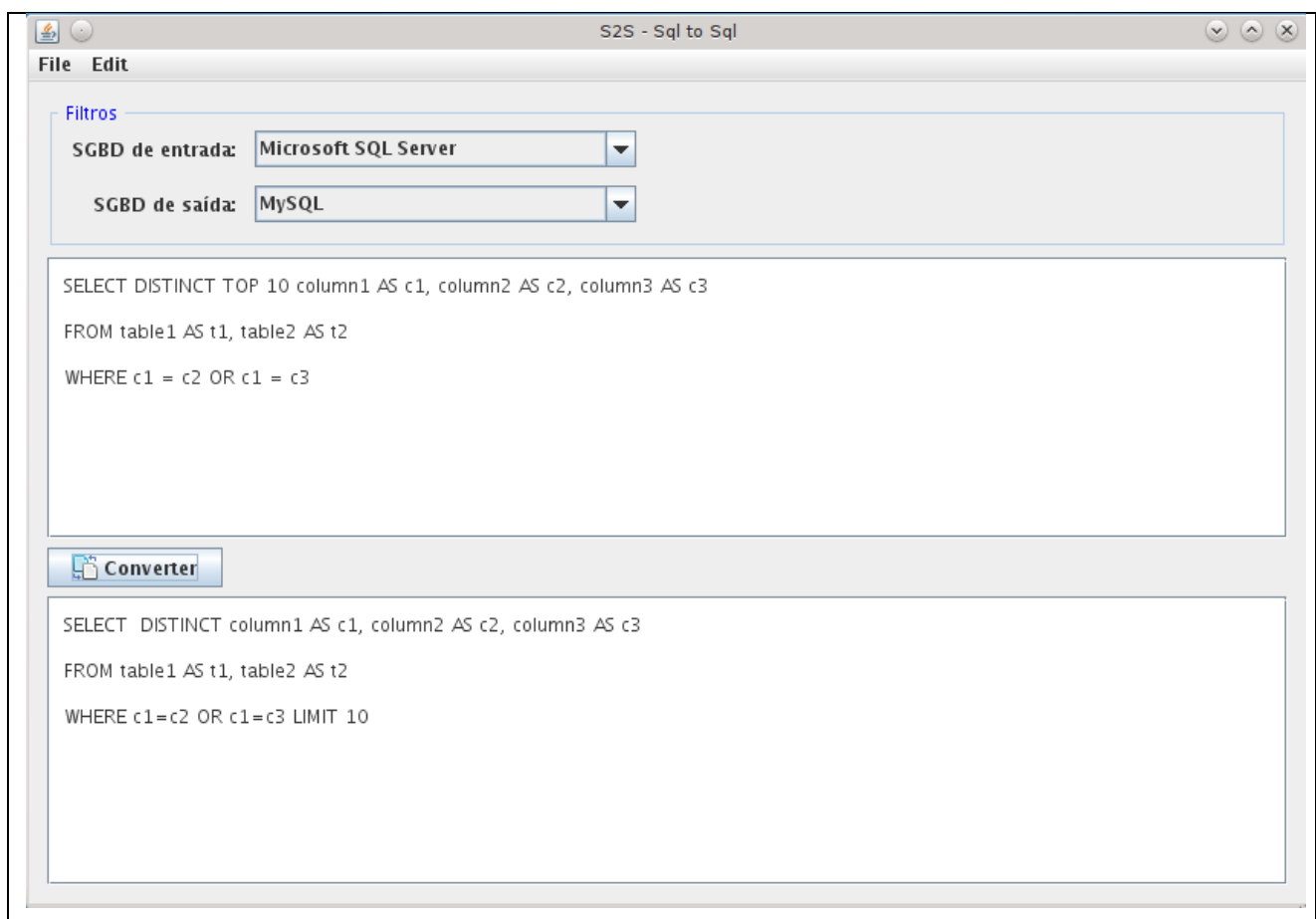


Figura 14 – Expressões SQL equivalentes

7. Conclusão

O presente trabalho teve como objetivo desenvolver uma ferramenta que converte uma expressão de consulta SQL, escrita na sintaxe específica de um SGBDR, para a sintaxe de outro SGBDR.

O objetivo pretendido foi alcançado. A ferramenta é capaz de receber uma expressão de consulta SQL em formato texto puro, identificar, através de parâmetros informados, as características de mapeamento e devolver a expressão de consulta SQL equivalente em modo texto. Todo esse processo envolve a intervenção mínima do usuário final, o que faz com que sua execução seja simplificada.

O grande diferencial da ferramenta é a fácil inclusão de novas regras de mapeamento, sejam elas para outros comandos SQL quanto para outros SGBDRs. Os SGBDRs que servem como parâmetro de entrada para o processo de mapeamento são independentes dos SGBDRs que servem como parâmetro de saída. Para adicionar um SGBDR passível de ser mapeado, é necessário apenas definir a sua própria gramática e ações semânticas para geração do código intermediário. Para adicionar um SGBDR que se deseje mapear, apenas as suas próprias definições das regras de transformação são necessárias. Essas regras são armazenadas em arquivos distintos por SGBDR. Dessa forma nada do que já está definido é alterado, ou seja, não se corre o risco de desestruturar informações de mapeamentos já existentes.

Porém, apesar da facilidade de inclusão de novas regras de mapeamento na ferramenta, essa tarefa pode ser facilitada ao usuário através de implementações que podem ser consideradas como trabalhos futuros:

- Implementar novos comandos: além das expressões de consulta, implementar também as expressões de manipulação de dados (e.g. UPDATE, DELETE);
- Validação semântica: atualmente a ferramenta apenas faz a validação léxica e sintática, aceitando as expressões de consulta SQL como semanticamente corretas.
- Compilar expressão SQL de saída: após todo o processo de compilação estar implementado (análise léxica, sintática e semântica), adicionar o processo de compilação no resultado do mapeamento. Isso é importante pois semanticamente algumas expressões são válidas apenas em determinados contextos (e.g. alguns comandos não funcionam em subconsultas);
- Importação de novos arquivos XSLT: permitir que a ferramenta possa receber, através de uma opção simples, novos arquivos de mapeamento;

- Ferramenta de edição para arquivos XSLT de mapeamento de comandos SQL: a definição das regras de mapeamento nesses arquivos é feita de forma manual, através de um editor de textos qualquer. Uma ferramenta criada para esse propósito facilitaria a edição, podendo ter como características a visualização de arquivos XSLT importados para evitar replicação de código, inserção automática de blocos de comandos previamente definidos quando apenas parte desses comandos necessitar de alteração além da montagem automática da estrutura básica que um arquivo de mapeamento deve ter.

REFERÊNCIAS BIBLIOGRÁFICAS

1. ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 4. ed. São Paulo: Addison Wesley, 2005.
2. DEITEL, Harvey M, et al. **XML Como Programar**. 1. ed. São Paulo: Bookman, 2003.
3. FURTADO, Olinto José Varela. **Apostila de Linguagens Formais e Compiladores**. Florianópolis: UFSC, 1992.
4. WORLD WIDE WEB CONSORTIUM. **Extensible Markup Language (XML)**. Disponível na Internet no endereço: <http://www.w3.org/XML>. Acesso em: Novembro de 2011.
5. W3SCHOOLS. **XML Tutorial**. Disponível na Internet no endereço: <http://www.w3schools.com/xml>. Acesso em: Novembro de 2011.
6. W3SCHOOLS. **XPath Tutorial**. Disponível na Internet no endereço: <http://www.w3schools.com/xpath>. Acesso em: Novembro de 2011.
7. WORLD WIDE WEB CONSORTIUM. **XSL Transformations (XSLT) Version 1.0**. Disponível na internet no endereço: <http://www.w3.org/TR/xslt>. Acesso em: Novembro de 2011.
8. W3SCHOOLS. **XSLT Tutorial**. Disponível na Internet no endereço: <http://www.w3schools.com/xsl>. Acesso em: Novembro de 2011.
9. SWISSQL. **SwisSQL Console 5.5**. Disponível na Internet no endereço: <http://www.swissql.com/products/sql-translator/sql-converter.html>. Acesso em: Novembro de 2011.
10. SQLWAYS. **SQLWays**. Disponível na Internet no endereço: <http://www.ispirer.com/products>. Acesso em: Novembro de 2011.
11. DBCONVERT. **DBConvert**. Disponível na Internet no endereço: <http://dbconvert.com/>. Acesso em: Novembro de 2011.
12. JDOM. **Jdom**. Disponível na Internet no endereço: <http://www.jdom.org/>. Acesso em: Novembro de 2011.
13. MICROSOFT SQL SERVER, **Microsoft SQL Server**. Disponível na Internet no

endereço: <http://www.microsoft.com/sqlserver/en/us/default.aspx>. Acesso em: Novembro de 2011.

14. MYSQL, **MySQL**. Disponível na Internet no endereço: <http://www.mysql.com/>. Acesso em: Novembro de 2011.

15. POSTGRESQL, **PostgreSQL**. Disponível na Internet no endereço: <http://www.postgresql.org/>. Acesso em: Novembro de 2011.

16. FIREBIRD, **Firebird**. Disponível na Internet no endereço: <http://www.firebirdsql.org/>. Acesso em: Novembro de 2011.

17. MICROSOFT SQL SERVER, **SELECT Clause (Transact-SQL)**. Disponível na Internet no endereço: <http://msdn.microsoft.com/en-us/library/ms176104.aspx>. Acesso em: Novembro de 2011.

18. MYSQL, **Sintaxe SELECT**. Disponível na Internet no endereço: <http://dev.mysql.com/doc/refman/4.1/pt/select.html>. Acesso em: Novembro de 2011.

19. POSTGRESQL, **SELECT**. Disponível na Internet no endereço: <http://www.postgresql.org/docs/9.0/static/sql-select.html>. Acesso em: Novembro de 2011.

20. FIREBIRD, **SELECT**. Disponível na Internet no endereço: <http://www.firebirdsql.org/refdocs/langrefupd21-select.html>. Acesso em: Novembro de 2011.

21. GESSER, Carlos Eduardo. **Ambiente para geração de analisadores léxicos e sintáticos**. Trabalho de Conclusão de Curso - UFSC, 2002.

ANEXO 1 – ARTIGO

MAPEAMENTO DE CONSULTAS SQL ENTRE SISTEMAS GERENCIADORES DE BANCOS DE DADOS UTILIZANDO TECNOLOGIA XSLT

Renato Sulzbach, Ronaldo dos Santos Mello

Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Florianópolis – Santa Catarina – Brasil

Resumo

O presente trabalho tem como objetivo desenvolver uma ferramenta que converte uma expressão de consulta SQL, escrita na sintaxe específica de um Sistema Gerenciador de Banco de Dados Relacional (SGBDR), para a sintaxe de outro SGBDR. Essa conversão é realizada utilizando técnicas de compilação e o suporte da tecnologia XSLT. A API desenvolvida juntamente com a ferramenta, e que representa o núcleo do processo de conversão, pode ser facilmente incorporada dentro de aplicativos Java/J2EE, permitindo que esses possam trabalhar com vários SGBDRs sem a necessidade de portar consultas SQL específicas de um banco de dados para outro.

Palavras-chave: SGBDR, SQL, XML, XPATH, XSLT, API

1. Introdução

XML (eXtensible Markup Language) é uma linguagem de marcação de dados que provê um formato designado a descrever dados semi-estruturados. Recomendação da W3C, ela é utilizada por diversos tipos de aplicações principalmente para o compartilhamento de informações através da Internet. Documentos XML são auto-descritivos e possuem uma sintaxe simples. Ao contrário de documentos HTML, *tags* XML não são pré-definidas. O autor pode criar suas próprias marcações a fim de satisfazer necessidades especiais [W3C99].

O gerenciamento de dados XML é realizado por um conjunto de tecnologias provido pela W3C. A tecnologia XSL é uma delas. XSL é uma família de recomendações para a definição de transformações e apresentações de documentos XML. Ela consiste de três partes, compostas pelas tecnologias XPath, XSLT e XSL-FO. XSLT *stylesheet* especifica a apresentação de uma classe de documentos XML, descrevendo como uma instância dessa classe é transformada em um documento XML que utiliza um vocabulário formatado, como (X)HTML ou XSL-FO [W3C07].

No contexto desse trabalho, o problema a ser tratado são expressões SQL que podem assumir diferentes sintaxes dependendo do SGBDR alvo. Vários SGBDRs possuem como característica diferentes formas de disponibilizar uma determinada funcionalidade através de comandos SQL. Essas diferentes formas de implementação trazem dificuldades aos usuários, pois nem sempre uma expressão SQL escrita por um fabricante é válida para outra. Com essa falta de padronização, usuários têm de recorrer a diferentes fontes de pesquisa em busca da tradução correta de uma expressão para o

SGBDR ao qual deseja aplicar. Um exemplo dessa heterogeneidade é o comando que especifica o primeiro conjunto de linhas a ser retornado do resultado de uma consulta SQL. Tal comando pode assumir diferenças tanto na grafia quanto na posição na estrutura da expressão SQL.

1.1. Objetivo

Construir uma ferramenta onde uma determinada expressão de consulta SQL de entrada, escrita para um determinado SGBDR, e representada em um documento XML, pode ser convertida para o dialeto SQL de outro SGBDR

1.2. Motivação

Limitações existentes nos trabalhos relacionados. Estas limitações dizem respeito à falta de flexibilidade para o mapeamento de consultas entre SGBDs de interesse e a inexistência de soluções gratuitas para a realização desta tarefa.

2. Ferramenta proposta

Neste trabalho é proposta e implementada uma ferramenta de mapeamento de consultas SQL entre diferentes SGBDRs. O objetivo é, dada uma expressão SQL qualquer, de um determinado SGBDR, a ferramenta possa mapear e transformar para a expressão SQL equivalente de outro SGBDR. Devido à grande variedade de comandos SQL, a ferramenta implementa o mapeamento de comandos de consulta (*SELECT*), uma vez que são os tipos mais frequentes de operações que um SGBD executa e por mais apresentarem diferenças sintáticas.

Para que o mapeamento possa ser realizado, a ferramenta funciona com base em dois processos: **compilação** e **transformação**. O usuário informa a expressão SQL e a qual SGBDR ela pertence. Ele informa também o SGBDR para o qual se deseja que a expressão SQL de entrada seja transformada. A partir dessas informações de entrada, a ferramenta inicia o processo de compilação, analisando os aspectos léxicos e sintáticos da expressão de entrada. Em caso de erro, o sistema informa ao usuário a falha na compilação e indica a posição exata onde ocorreu esse erro. Se nenhum erro ocorrer é iniciada a análise semântica da compilação juntamente com a geração de código intermediário. O código intermediário gerado representa a expressão SQL de entrada estruturada no formato XML. A ferramenta então seleciona o conjunto de regras de transformação e aplica ao código intermediário gerado enviando ao fluxo de saída a expressão SQL do SGBDR de destino.

A Figura 1 mostra a arquitetura geral do processo de mapeamento.

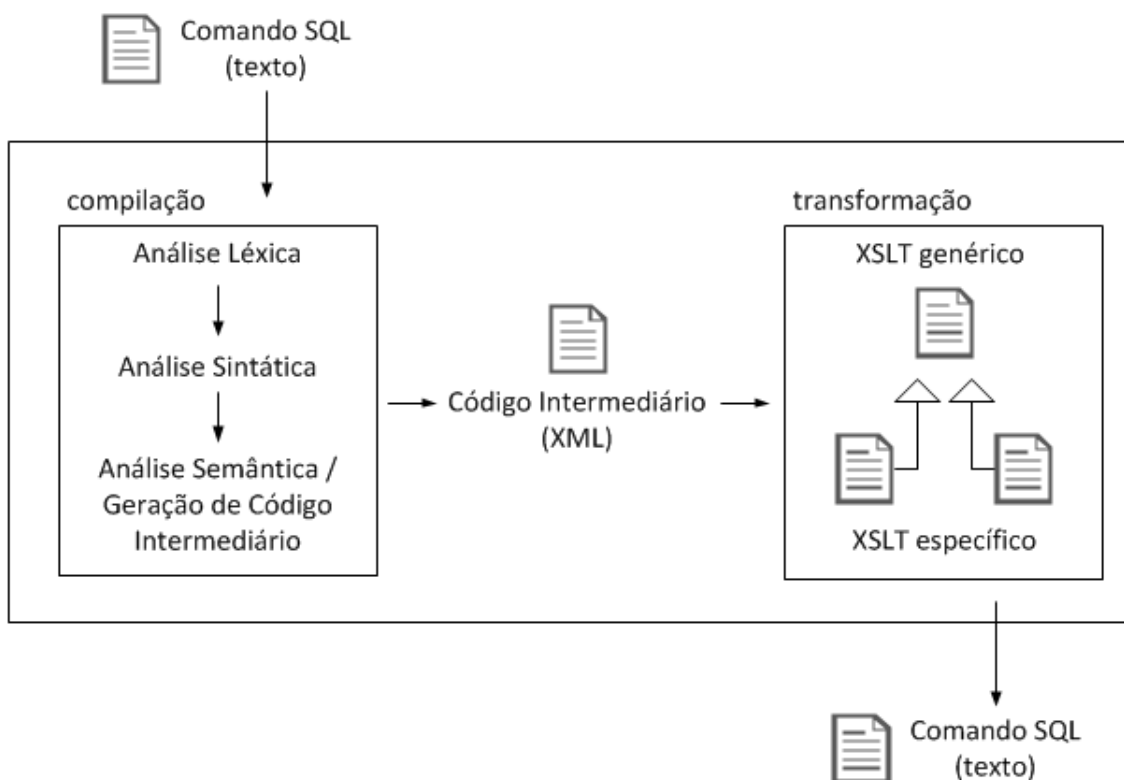


Figura 1 – Esquema geral do processo de mapeamento entre comandos SQL

2.1. Compilação

Para o processo de compilação, foi utilizada a ferramenta GALS (Gerador de Analisadores Léxicos e Sintáticos) (GESSER, 2002). Nela foi inserida uma gramática específica para cada um dos possíveis SGBDRs de entrada. O presente trabalho implementou regras de mapeamento para quatro SGBDRs: Microsoft SQL Server, MySQL, PostgreSQL e Firebird. Entretanto, outros SGBDRs também podem ser integrados à ferramenta, que é facilmente extensível por definir regras de mapeamento em gramáticas e arquivos XSL independentes.

O GALS gera, de forma automatizada, todas as classes necessárias para o processo de análise léxica e sintática. A classe de análise semântica é gerada em branco. Nela então são definidos os métodos que implementam cada uma das ações semânticas separadamente.

Conforme comentado anteriormente, a análise semântica gera o código intermediário no formato XML. Levando-se em consideração que o mapeamento dificilmente envolverá grande quantidade de código a ser mapeado, optou-se pelo desempenho à economia de memória. Dessa forma, o código intermediário é gerado utilizando a biblioteca JDOM (JDOM, 2011), que cria uma estrutura em árvore na memória com o conteúdo do XML do código intermediário. Essa estrutura em memória é enviada ao fluxo de entrada do processo de transformação.

2.2. Transformação

O código intermediário gerado no processo de compilação possui uma estrutura única, em formato XML, que serve como informação de entrada para regras que transformam o comando SQL de entrada no comando SQL de saída. A ferramenta define essas regras através de arquivos XSL. Essas regras são responsáveis por encontrar as correspondências para cada uma das *tags* encontradas no XML do código intermediário e processar a informação enviando o resultado para o fluxo de saída no formato de texto

puro, que corresponde à expressão SQL de saída.

Os arquivos XSL são específicos para cada um dos SGBDRs possíveis de serem mapeados, porém, são especializados através de um arquivo XSL genérico, que define regras de mapeamento comuns a todos eles, bem como variáveis necessárias para seus processamentos.

O XSL genérico possui regras básicas para o mapeamento direto de uma expressão de consulta SQL padrão. Se em um processo de transformação forem aplicadas apenas as regras de correspondência do XSL genérico, a expressão SQL de saída será idêntica à expressão SQL de entrada.

Os arquivos XSL específicos para cada SGBDR importam o arquivo XSL genérico. Dessa forma, as regras de correspondência definidas genericamente podem ser sobrescritas pelos arquivos específicos, modificando ou agregando valores à expressão SQL de entrada especificada.

3. API

A API é o mecanismo de análise e mapeamento de consultas SQL que representa o núcleo do funcionamento da ferramenta. Ela permite que aplicações baseadas em Java possam trabalhar com vários SGBDRs sem a necessidade de portar consultas SQL específicas de um SGBD para outro, convertendo instruções *SELECT* em tempo real entre os SGBDs suportados. A arquitetura geral da API é ilustrada na Figura 2.

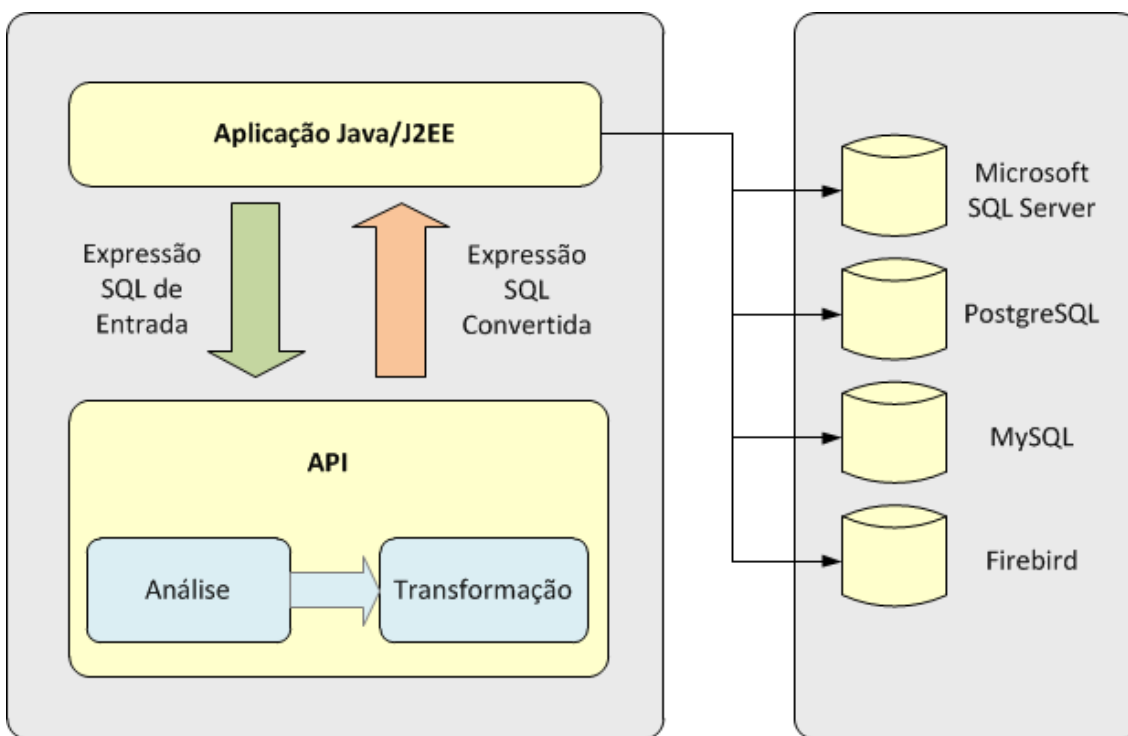


Figura 2 – API: arquitetura geral

A API fornece uma maneira simples e fácil de realizar a análise e mapeamento das consultas SQL diretamente dentro de aplicativos Java/J2EE. Esse recurso garante que os aplicativos estejam no controle de quando uma consulta SQL precisa ser convertida para um dialeto diferente.

4. Conclusão

O presente trabalho teve como objetivo desenvolver uma ferramenta que converte

uma expressão de consulta SQL, escrita na sintaxe específica de um SGBDR, para a sintaxe de outro SGBDR.

O grande diferencial da ferramenta é a fácil inclusão de novas regras de mapeamento, sejam elas para outros comandos SQL quanto para outros SGBDRs. Os SGBDRs que servem como parâmetro de entrada para o processo de mapeamento são independentes dos SGBDRs que servem como parâmetro de saída. Para adicionar um SGBDR passível de ser mapeado, é necessário apenas definir a sua própria gramática e ações semânticas para geração do código intermediário. Para adicionar um SGBDR que se deseje mapear, apenas as suas próprias definições das regras de transformação são necessárias. Essas regras são armazenadas em arquivos distintos por SGBDR. Dessa forma nada do que já está definido é alterado, ou seja, não se corre o risco de desestruturar informações de mapeamentos já existentes.

Porém, apesar da facilidade de inclusão de novas regras de mapeamento na ferramenta, essa tarefa pode ser facilitada ao usuário através de implementações que podem ser consideradas como trabalhos futuros, como novos comandos (e.g. UPDATE, DELETE), validação semântica, compilação da expressão SQL de saída, importação de novos arquivos XSLT e uma ferramenta de edição para arquivos XSLT de mapeamento de comandos SQL

5. Referências bibliográficas

1. ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 4. ed. São Paulo: Addison Wesley, 2005.
2. DEITEL, Harvey M, et al. **XML Como Programar**. 1. ed. São Paulo: Bookman, 2003.
3. FURTADO, Olinto José Varela. **Apostila de Linguagens Formais e Compiladores**. Florianópolis: UFSC, 1992.
4. WORLD WIDE WEB CONSORTIUM. **Extensible Markup Language (XML)**. Disponível na Internet no endereço: <http://www.w3.org/XML>. Acesso em: Novembro de 2011.
5. W3SCHOOLS. **XML Tutorial**. Disponível na Internet no endereço: <http://www.w3schools.com/xml>. Acesso em: Novembro de 2011.
6. W3SCHOOLS. **XPath Tutorial**. Disponível na Internet no endereço: <http://www.w3schools.com/xpath>. Acesso em: Novembro de 2011.
7. WORLD WIDE WEB CONSORTIUM. **XSL Transformations (XSLT) Version 1.0**. Disponível na internet no endereço: <http://www.w3.org/TR/xslt>. Acesso em: Novembro de 2011.
8. W3SCHOOLS. **XSLT Tutorial**. Disponível na Internet no endereço: <http://www.w3schools.com/xsl>. Acesso em: Novembro de 2011.
9. SWISSQL. **SwisSQL Console 5.5**. Disponível na Internet no endereço: <http://www.swissql.com/products/sql-translator/sql-converter.html>. Acesso em: Novembro de 2011.
10. SQLWAYS. **SQLWays**. Disponível na Internet no endereço: <http://www.ispirer.com/products>. Acesso em: Novembro de 2011.
11. DBCONVERT. **DBConvert**. Disponível na Internet no endereço: <http://dbconvert.com/>. Acesso em: Novembro de 2011.
12. JDOM. **Jdom**. Disponível na Internet no endereço: <http://www.jdom.org/>. Acesso em: Novembro de 2011.
13. MICROSOFT SQL SERVER, **Microsoft SQL Server**. Disponível na Internet no

endereço: <http://www.microsoft.com/sqlserver/en/us/default.aspx>. Acesso em: Novembro de 2011.

14. MYSQL, **MySQL**. Disponível na Internet no endereço: <http://www.mysql.com/>. Acesso em: Novembro de 2011.

15. POSTGRESQL, **PostgreSQL**. Disponível na Internet no endereço: <http://www.postgresql.org/>. Acesso em: Novembro de 2011.

16. FIREBIRD, **Firebird**. Disponível na Internet no endereço: <http://www.firebirdsql.org/>. Acesso em: Novembro de 2011.

17. MICROSOFT SQL SERVER, **SELECT Clause (Transact-SQL)**. Disponível na Internet no endereço: <http://msdn.microsoft.com/en-us/library/ms176104.aspx>. Acesso em: Novembro de 2011.

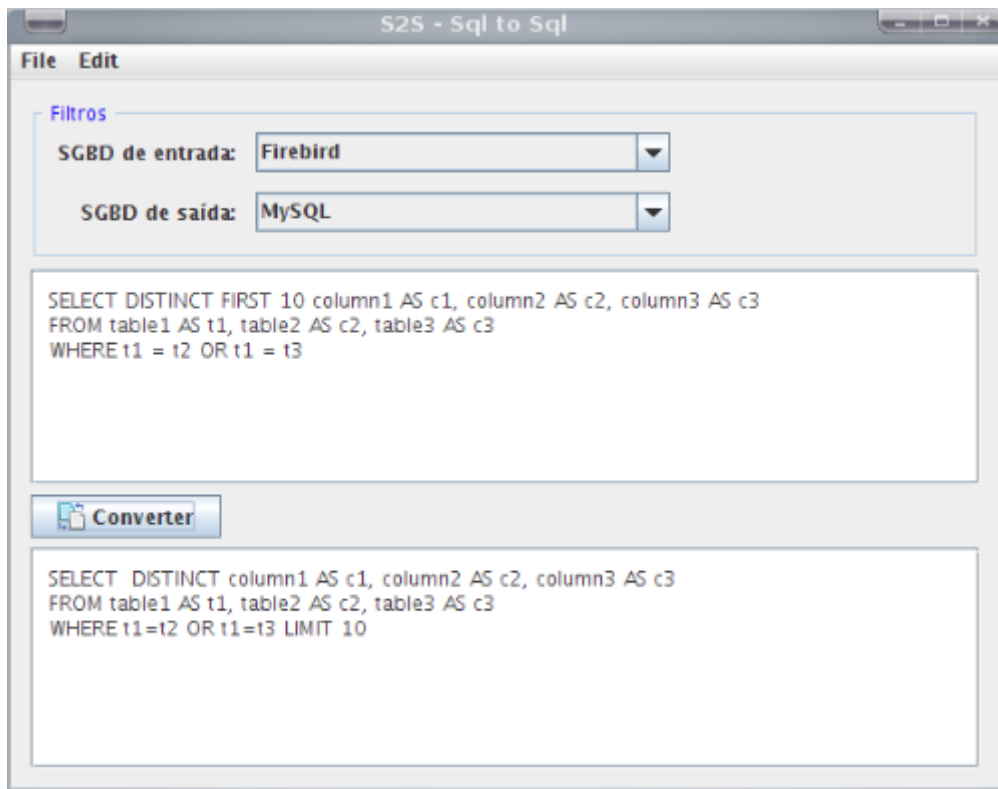
18. MYSQL, **Sintaxe SELECT**. Disponível na Internet no endereço: <http://dev.mysql.com/doc/refman/4.1/pt/select.html>. Acesso em: Novembro de 2011.

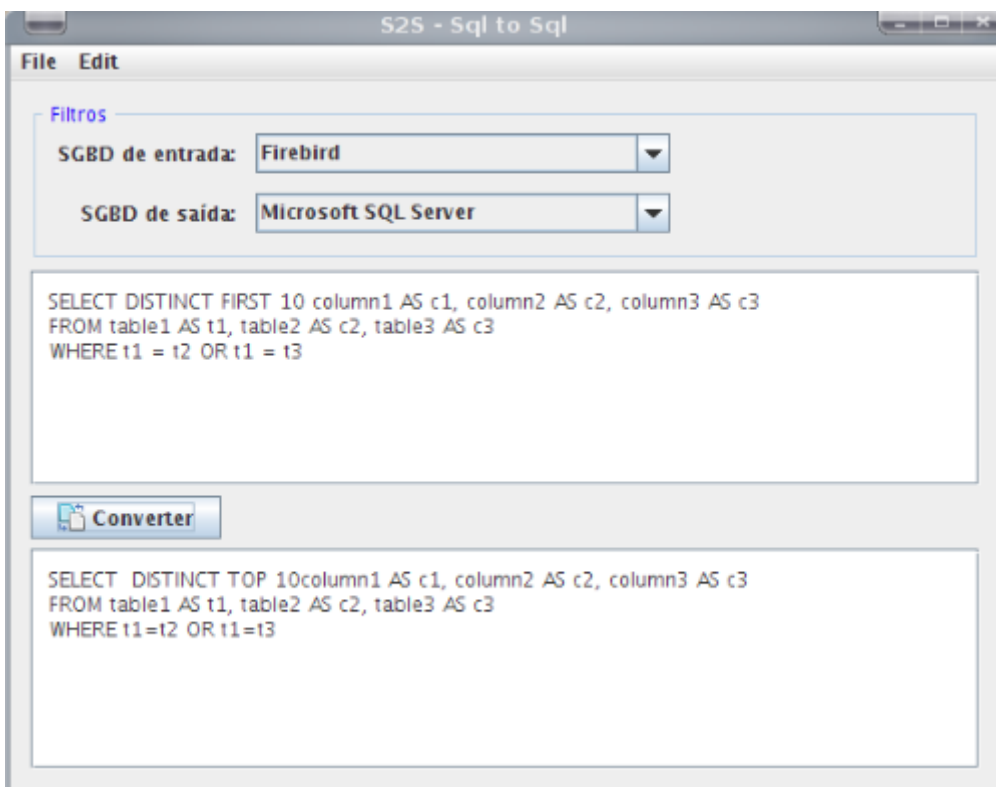
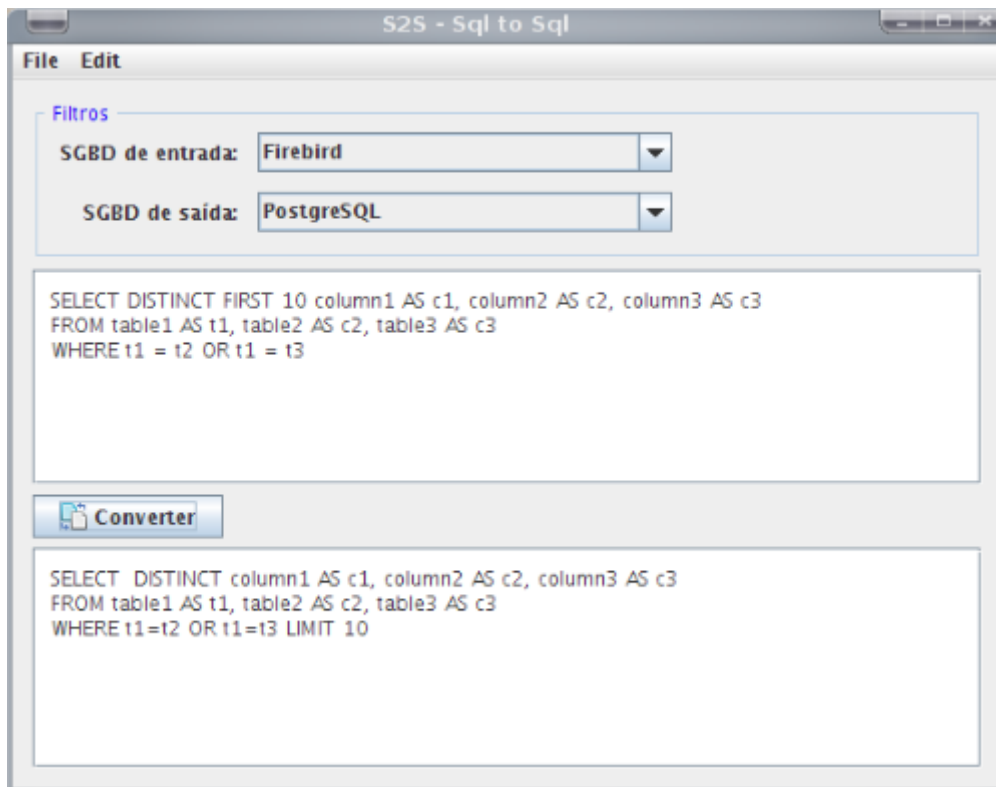
19. POSTGRESQL, **SELECT**. Disponível na Internet no endereço: <http://www.postgresql.org/docs/9.0/static/sql-select.html>. Acesso em: Novembro de 2011.

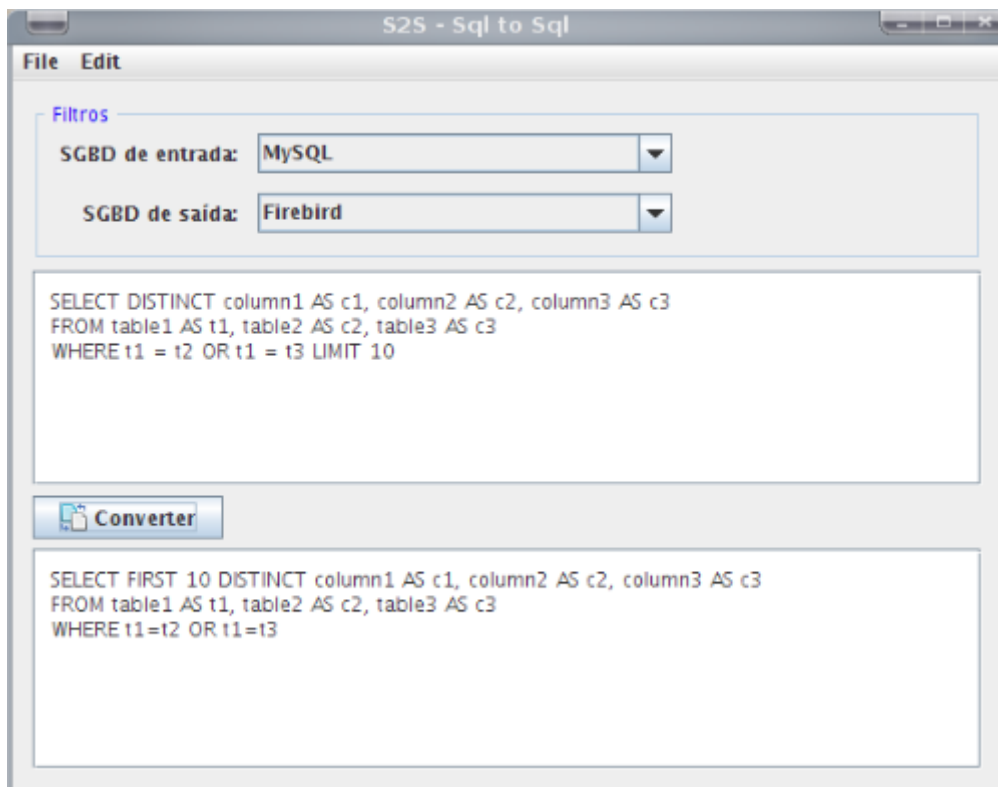
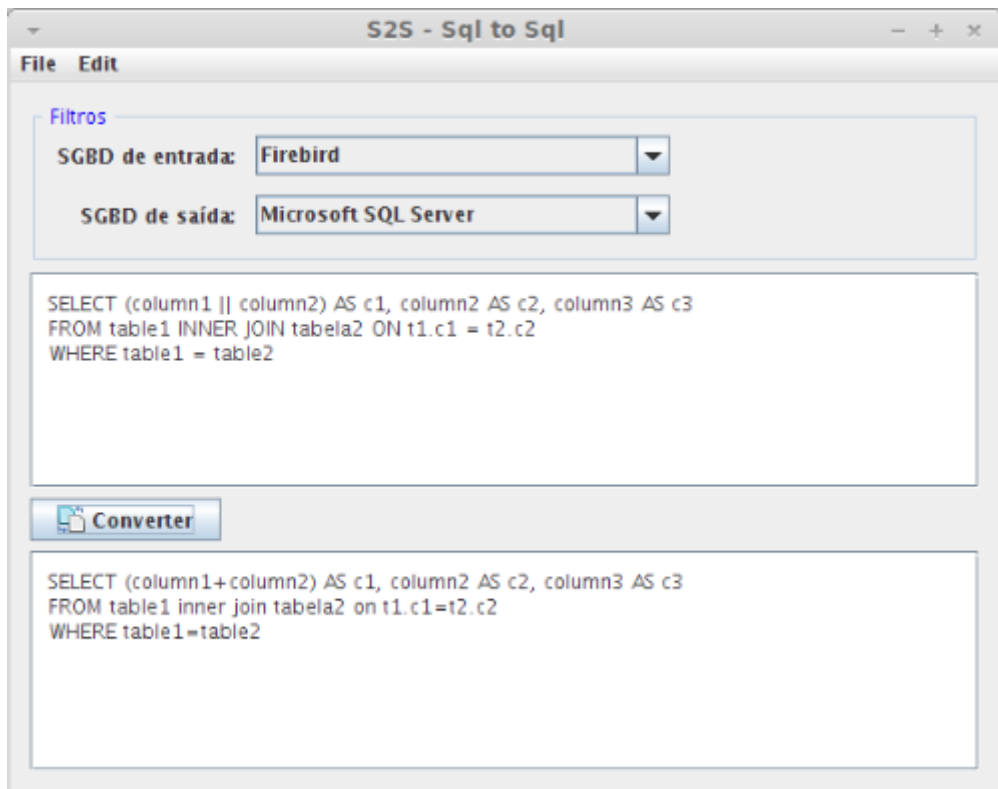
20. FIREBIRD, **SELECT**. Disponível na Internet no endereço: <http://www.firebirdsql.org/refdocs/langrefupd21-select.html>. Acesso em: Novembro de 2011.

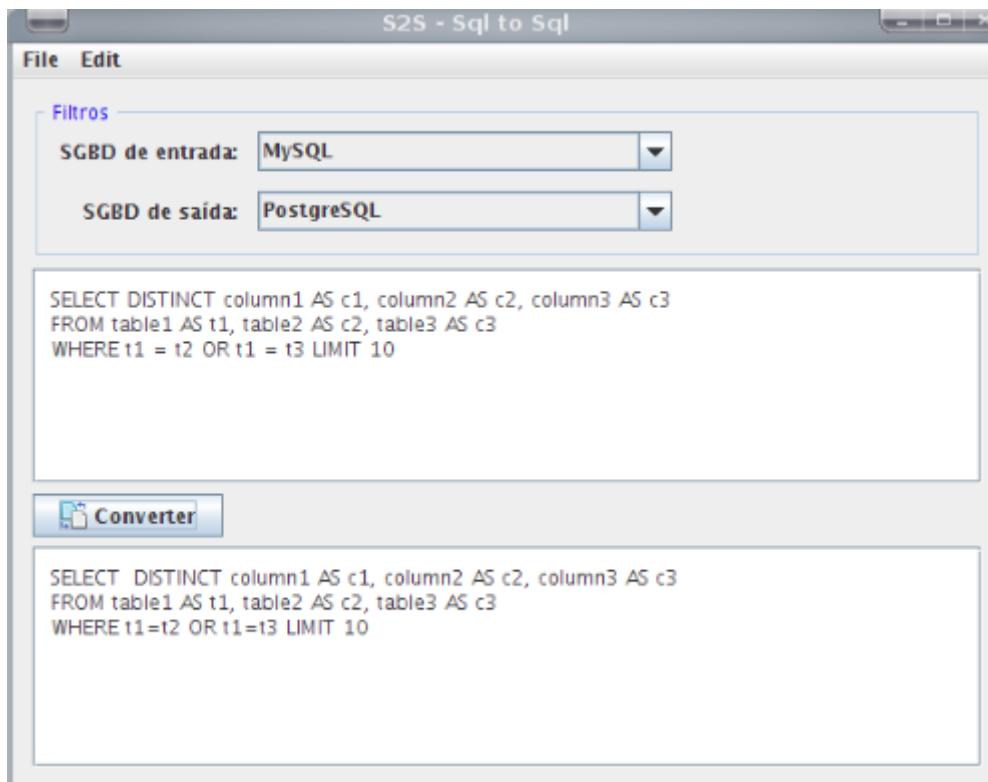
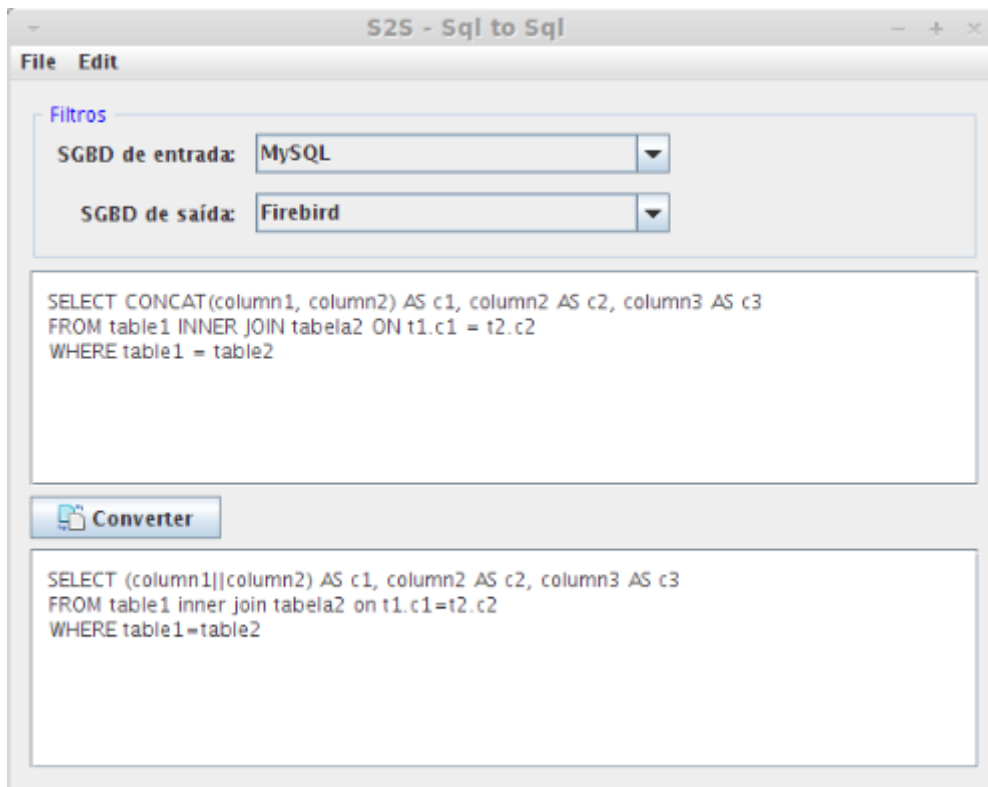
21. GESSER, Carlos Eduardo. **Ambiente para geração de analisadores léxicos e sintáticos**. Trabalho de Conclusão de Curso - UFSC, 2002.

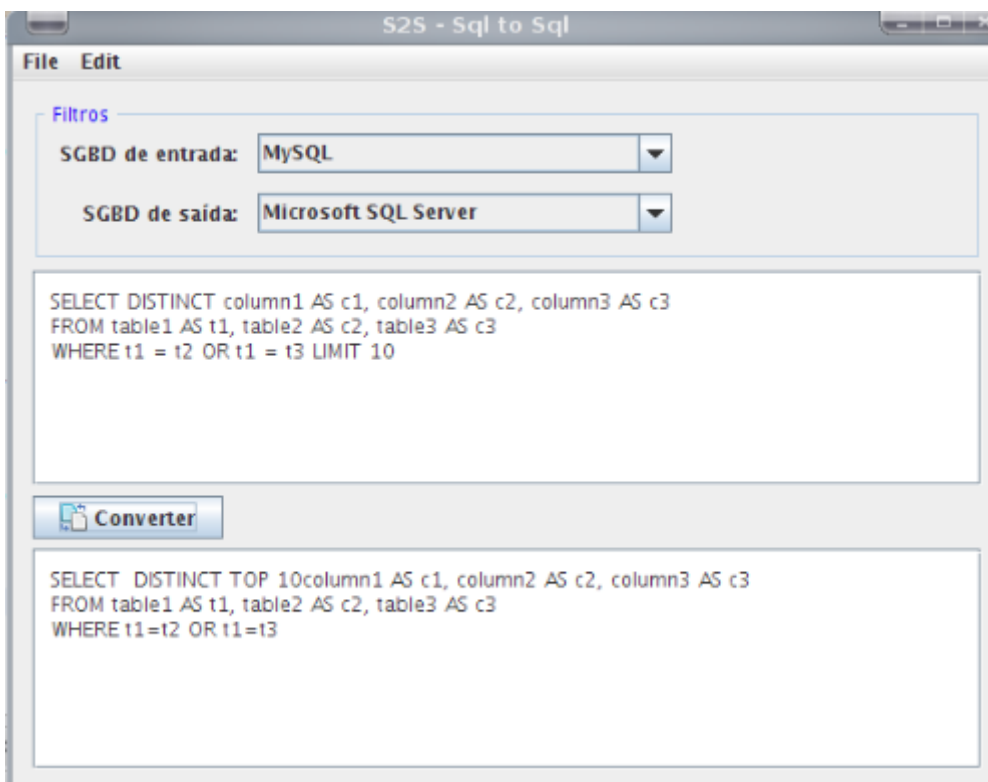
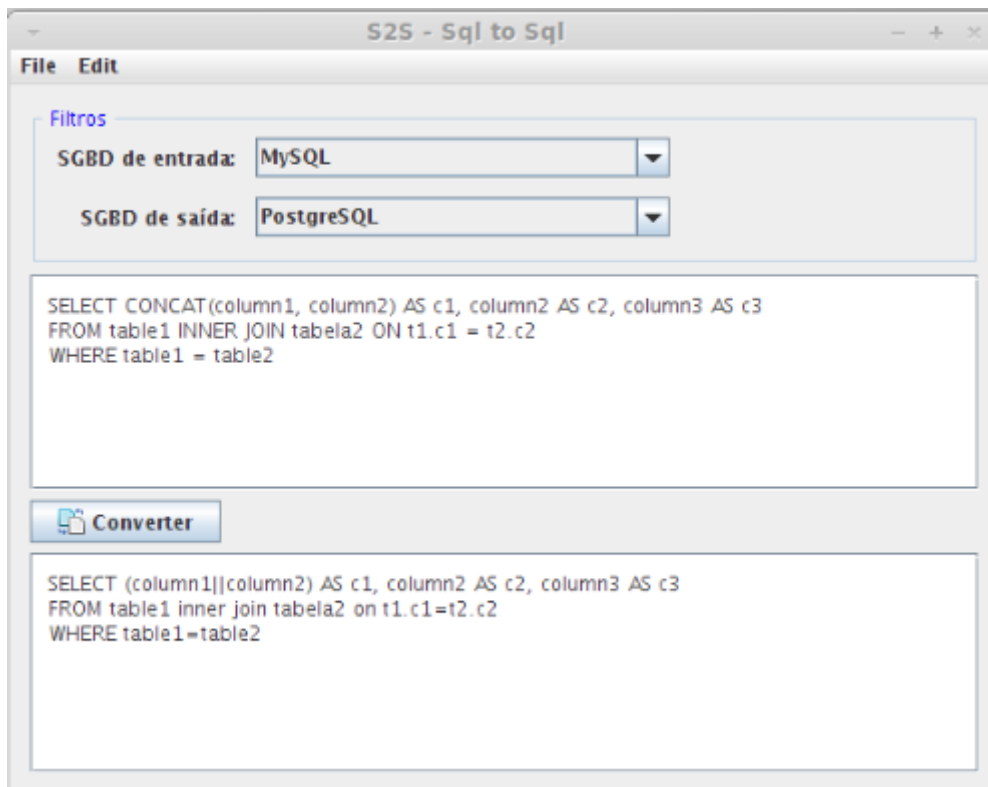
ANEXO 2 – TESTES DE MAPEAMENTO

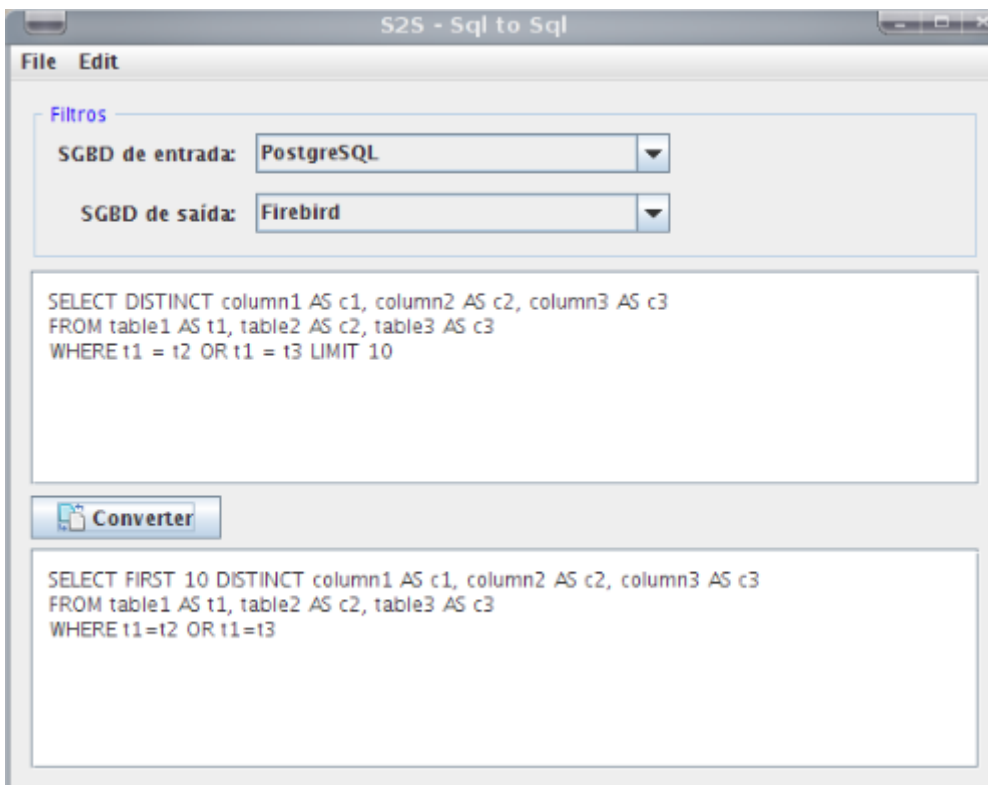
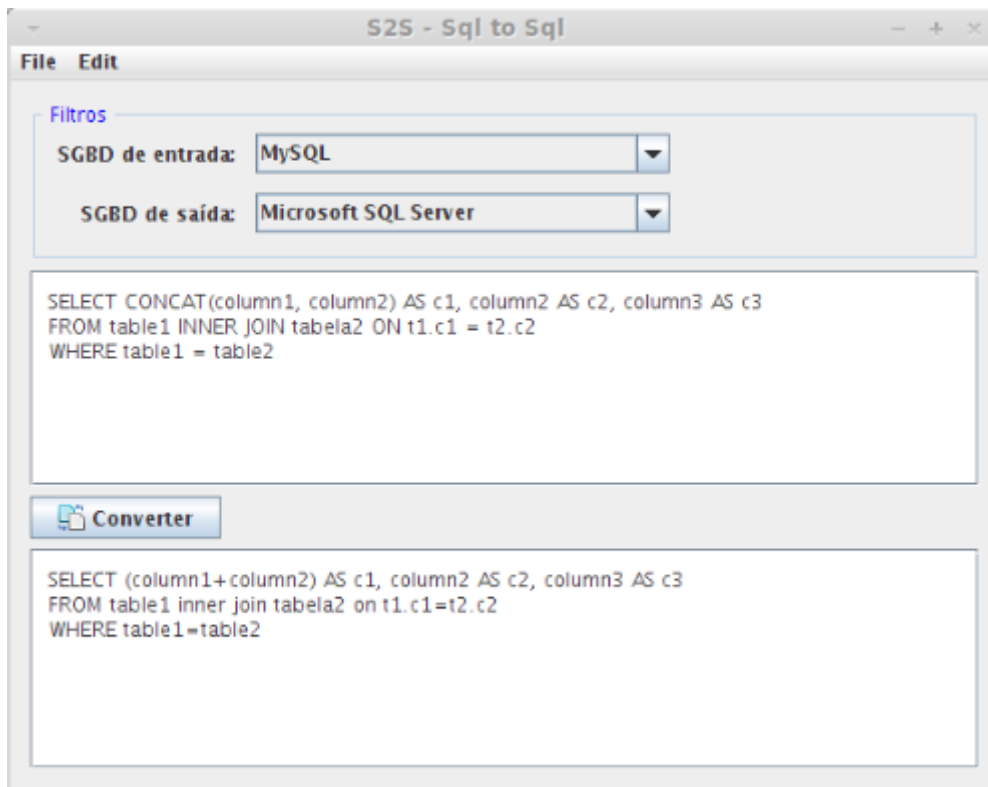


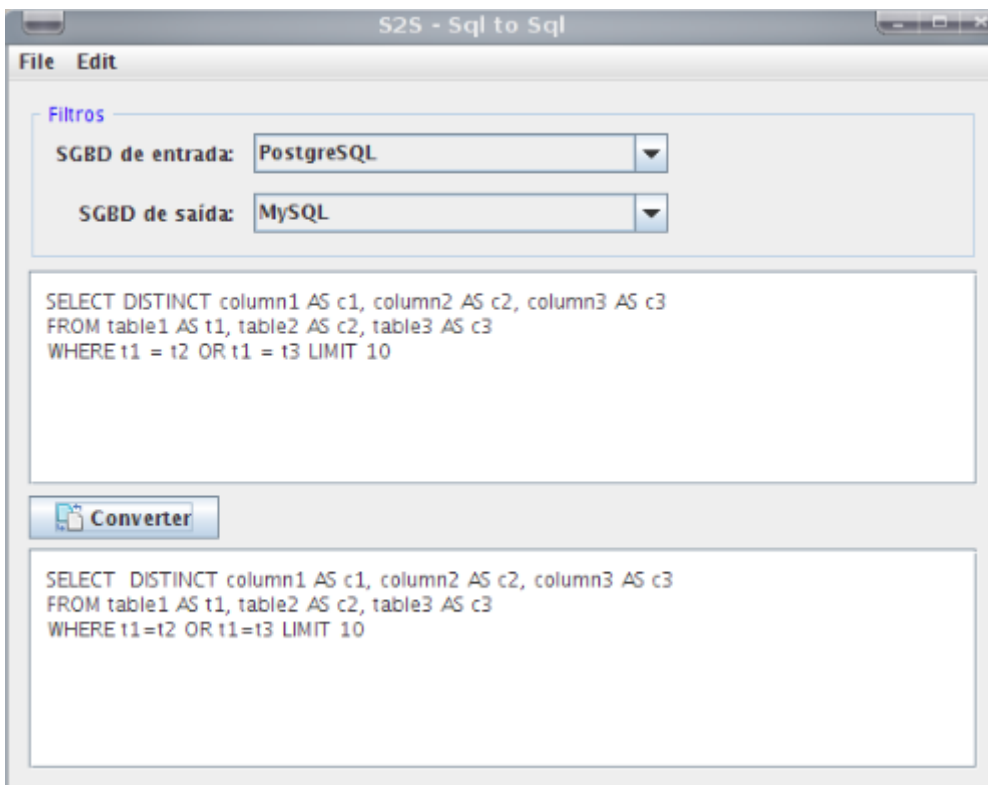
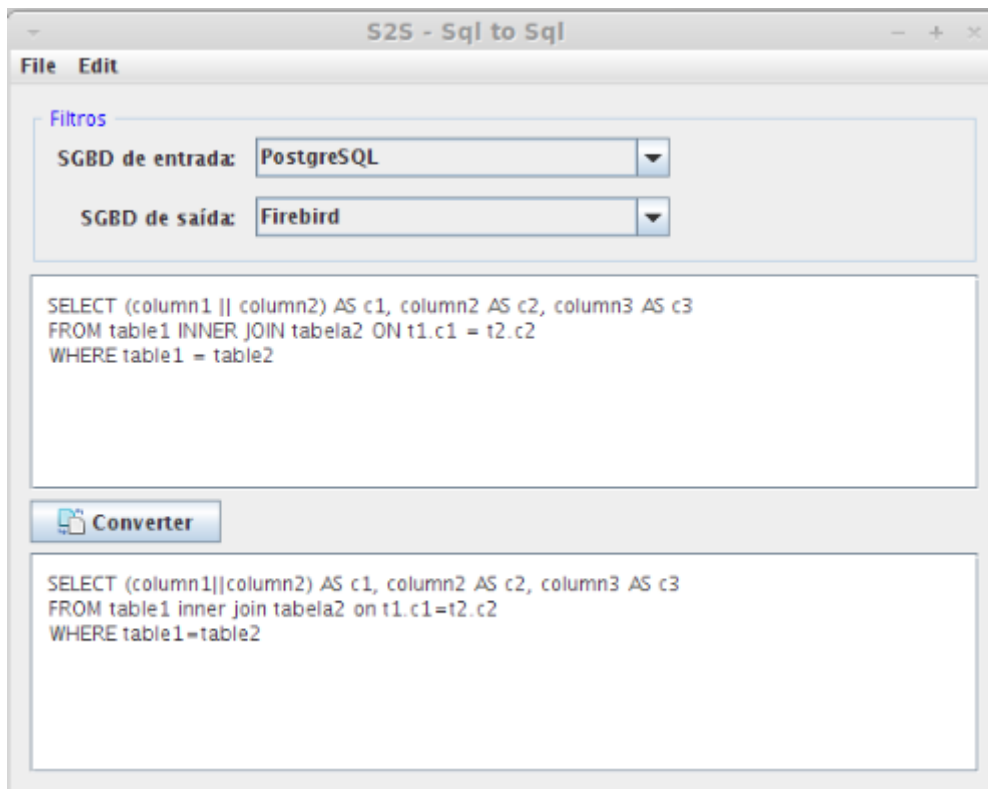


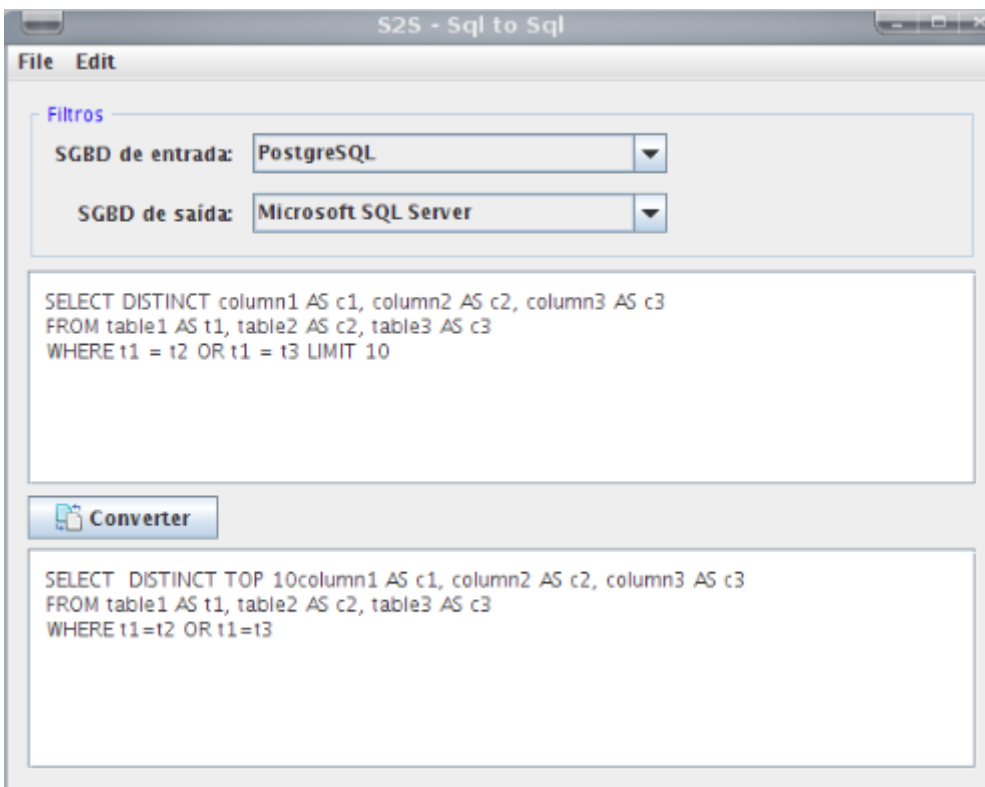
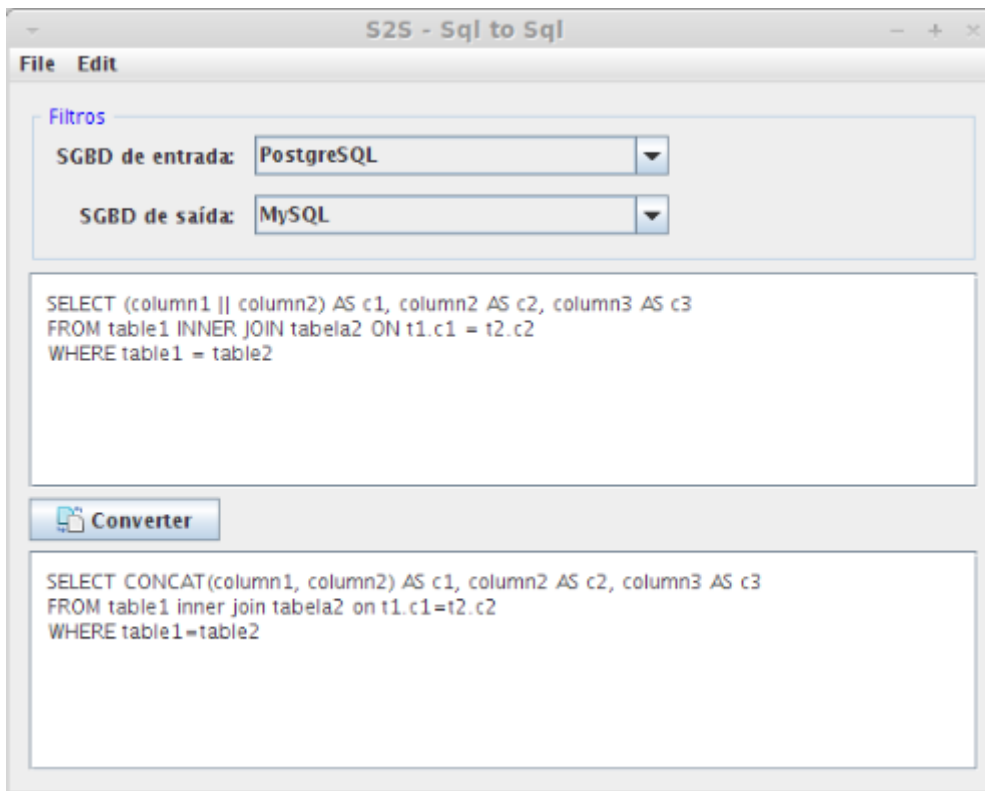


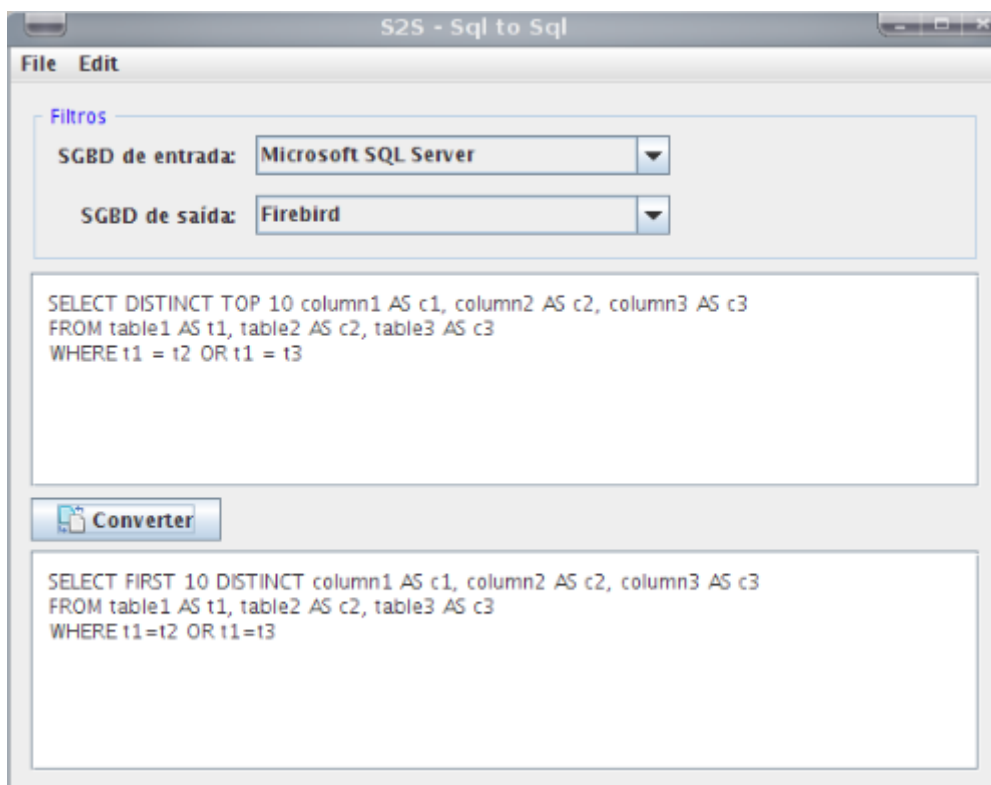


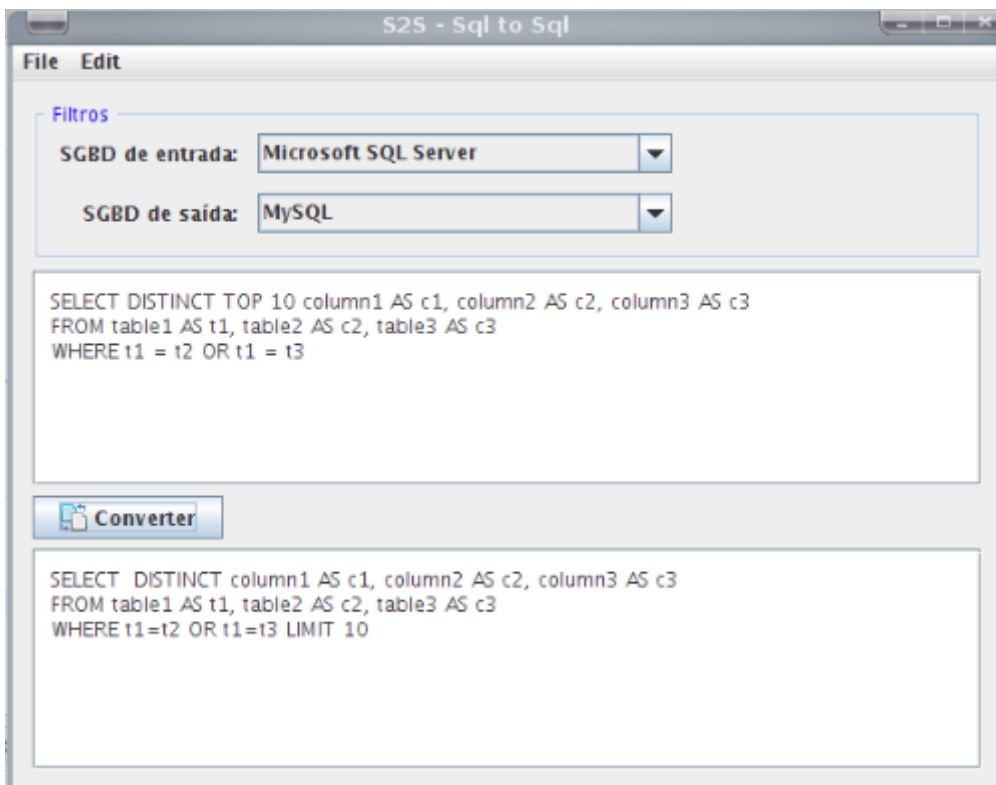
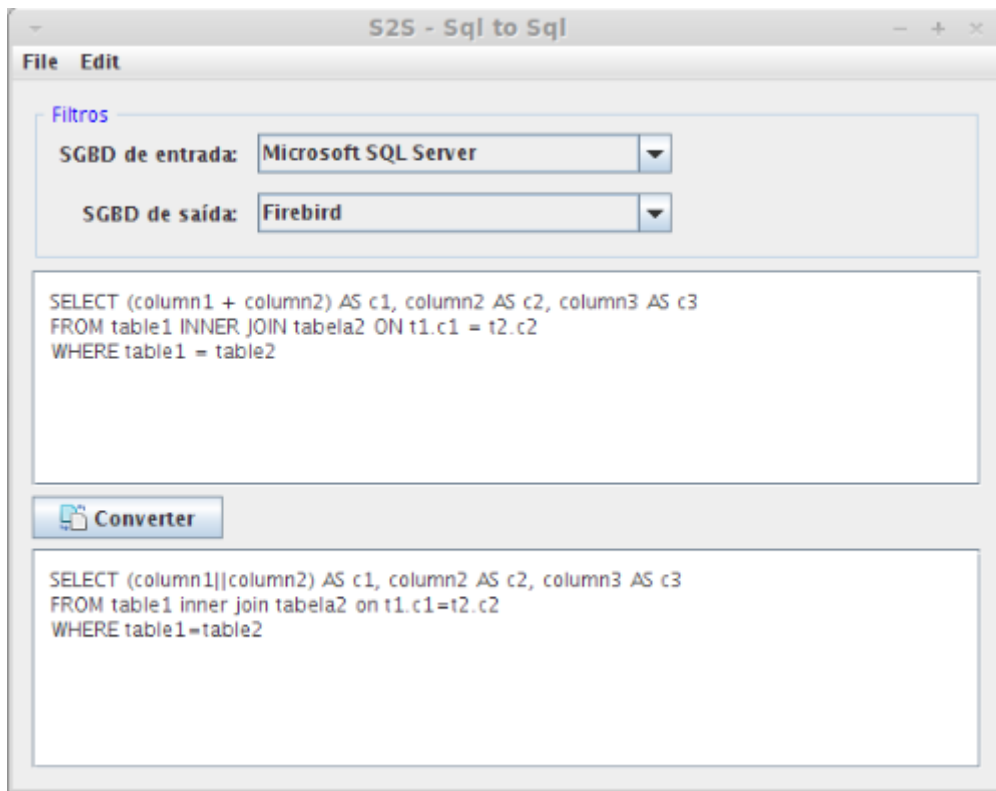


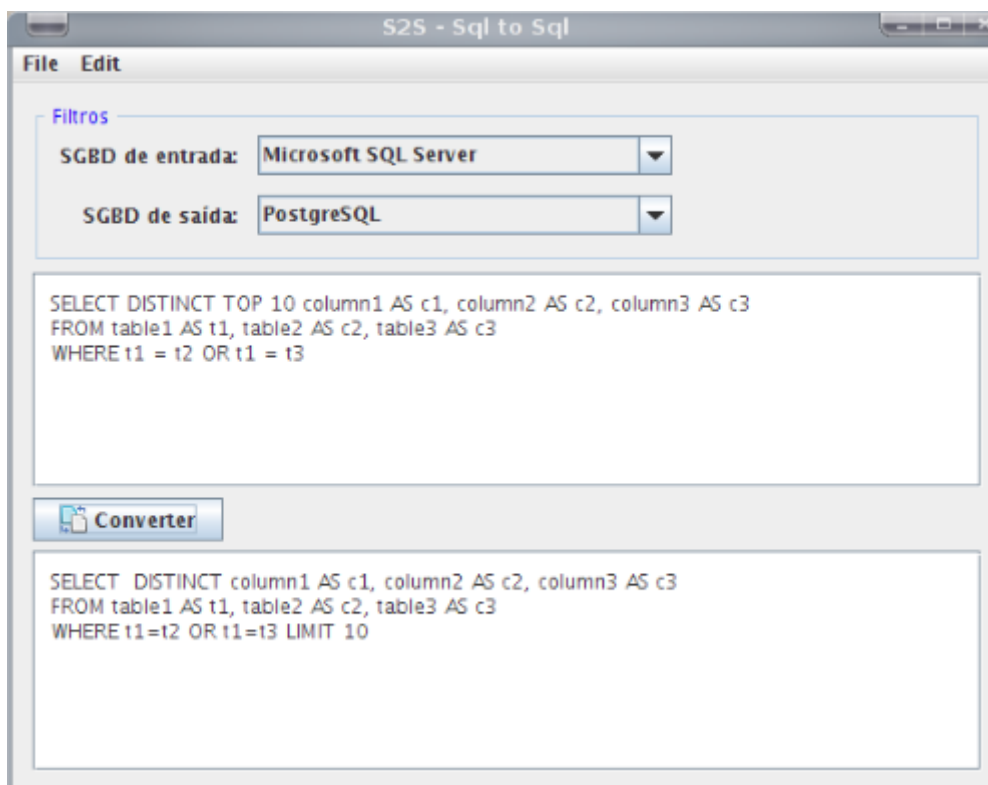
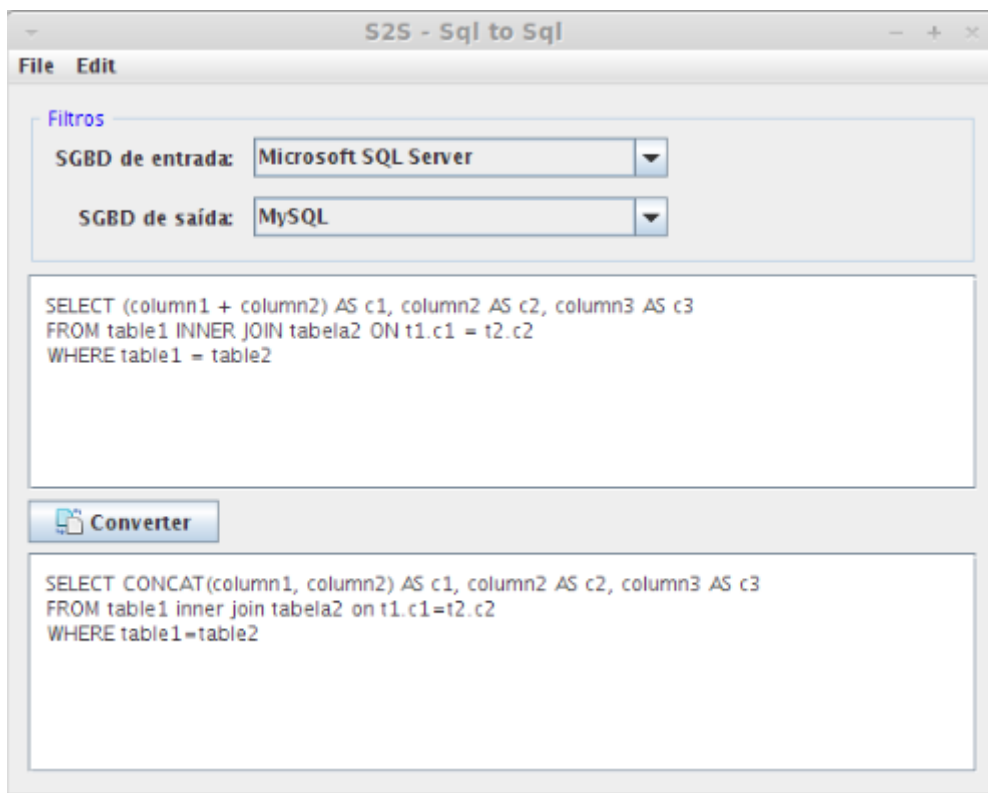


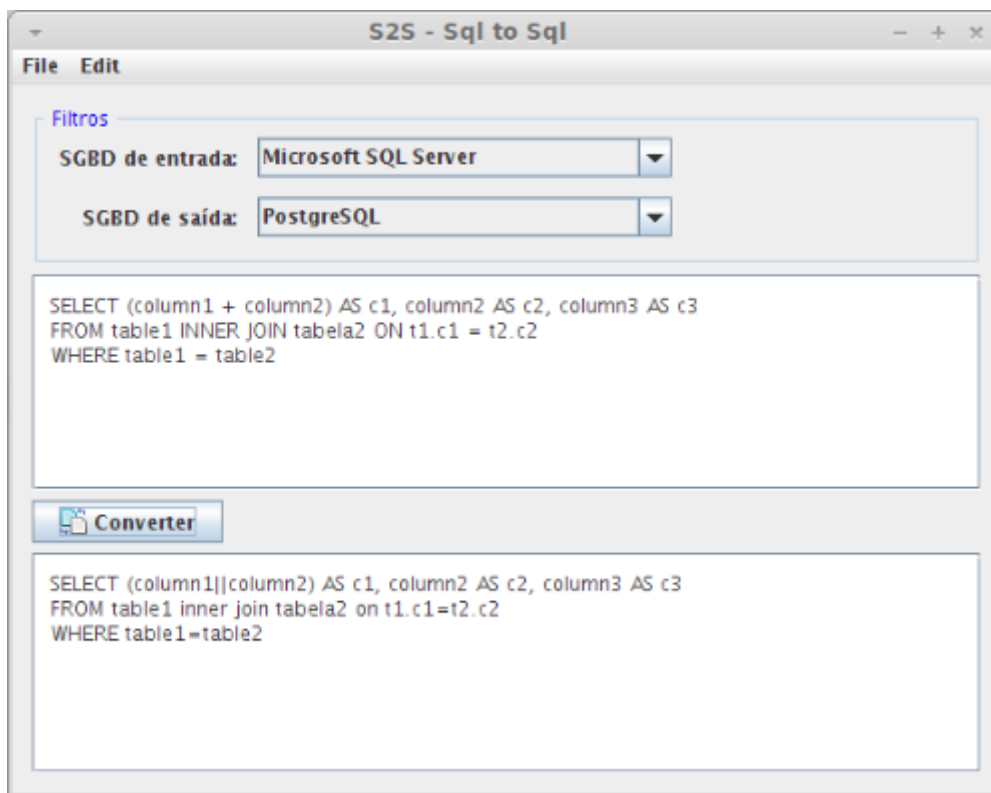












ANEXO 3 – FONTES

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * S2SView.java
 *
 * Created on 06/07/2011, 20:54:44
 */
package br.ufsc.s2s.view;

import br.ufsc.s2s._C;
import br.ufsc.s2s.application.S2SControl;
import java.io.File;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/**
 *
 * @author Renato
 */
public class S2SView extends javax.swing.JFrame {

    private S2SControl control = null;
    private final String cbxItemSelect = "Selecione...";
    private final String cbxItemFirebird = "Firebird";
    private final String cbxItemMssql = "Microsoft SQL Server";
    private final String cbxItemMySql = "MySQL";
    private final String cbxItemPostgre = "PostgreSQL";
    private final String cbxSgbdItems[] = {cbxItemSelect, cbxItemFirebird, cbxItemMssql, cbxItemMySql,
cbxItemPostgre};
    private final String invalidParameter = "Parâmetro inválido";
    private final String invalidSqlIn = "Comando SQL inválido";
    private final String selectSgbdIn = "Selecione o SGBD de ENTRADA";
    private final String selectSgbdOut = "Selecione o SGBD de SAÍDA";
    private final String insertSqlIn = "Insira o COMANDO SQL a ser mapeado";
    private final String txtSqlInEmptyText = "Insira aqui o comando SQL a converter";
    private final String txtSqlOutEmptyText = "Pressione o botão \"Converter\" para mapear o comando SQL
inserido";

    /** Creates new form S2SView */
    public S2SView() {
        this.control = new S2SControl(this);

        initComponents();

        this.setLocationRelativeTo(null);

        this.txtSqlIn.setText(this.txtSqlInEmptyText);
        this.txtSqlOut.setText(this.txtSqlOutEmptyText);

        DefaultComboBoxModel cbxSgbdInModel = new DefaultComboBoxModel(this.cbxSgbdItems);
        DefaultComboBoxModel cbxSgbdOutModel = new DefaultComboBoxModel(this.cbxSgbdItems);
        this.cbxSgbdIn.setModel(cbxSgbdInModel);
        this.cbxSgbdOut.setModel(cbxSgbdOutModel);
    }

    public int getSgbdIn() {
        int in = -1;
        int index = this.cbxSgbdIn.getSelectedIndex();

```



```

    if (this.cbxDatabases[index].equals(this.cbxDatabasesFirebird))
        in = _C.SGBD_FIREBIRD;
    else if (this.cbxDatabases[index].equals(this.cbxDatabasesMssql))
        in = _C.SGBD_MSSQL;
    else if (this.cbxDatabases[index].equals(this.cbxDatabasesMySQL))
        in = _C.SGBD_MYSQL;
    else if (this.cbxDatabases[index].equals(this.cbxDatabasesPostgre))
        in = _C.SGBD_POSTGRES;

    return in;
}

public int getSgbdOut() {
    int out = -1;
    int index = this.cbxDatabases.getSelectedIndex();

    if (this.cbxDatabases[index].equals(this.cbxDatabasesFirebird))
        out = _C.SGBD_FIREBIRD;
    else if (this.cbxDatabases[index].equals(this.cbxDatabasesMssql))
        out = _C.SGBD_MSSQL;
    else if (this.cbxDatabases[index].equals(this.cbxDatabasesMySQL))
        out = _C.SGBD_MYSQL;
    else if (this.cbxDatabases[index].equals(this.cbxDatabasesPostgre))
        out = _C.SGBD_POSTGRES;

    return out;
}

public String getSqlIn() {
    return this.txtSqlIn.getText();
}

public void setSqlOut(String sqlOut) {
    this.txtSqlOut.setText(sqlOut);
}

public void setSqlIn(String sqlIn) {
    this.txtSqlIn.setText(sqlIn);
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jScrollPane1 = new javax.swing.JScrollPane();
    txtSqlIn = new javax.swing.JTextArea();
    jScrollPane2 = new javax.swing.JScrollPane();
    txtSqlOut = new javax.swing.JTextArea();
    grpFilters = new javax.swing.JPanel();
    lblSgbdIn = new javax.swing.JLabel();
    cbxDatabasesIn = new javax.swing.JComboBox();
    lblSgbdOut = new javax.swing.JLabel();
    cbxDatabasesOut = new javax.swing.JComboBox();
    btnConvert = new javax.swing.JButton();
    menuBar = new javax.swing.JMenuBar();
    menuFile = new javax.swing.JMenu();
    menuItemOpen = new javax.swing.JMenuItem();
    jSeparator1 = new javax.swing.JPopupMenu.Separator();

```

```

menuItemQuit = new javax.swing.JMenuItem();
menuEdit = new javax.swing.JMenu();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("S2S - Sql to Sql");
setIconImage(this.getToolkit().createImage("/br/ufsc/s2s/resources/sqltosql.png"));

txtSqlIn.setColumns(20);
txtSqlIn.setRows(5);
txtSqlIn.setBorder(javax.swing.BorderFactory.createEmptyBorder(10, 10, 10, 10));
txtSqlIn.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        txtSqlInFocusGained(evt);
    }
    public void focusLost(java.awt.event.FocusEvent evt) {
        txtSqlInFocusLost(evt);
    }
});
jScrollPane1.setViewportViewView(txtSqlIn);

txtSqlOut.setColumns(20);
txtSqlOut.setEditable(false);
txtSqlOut.setRows(5);
txtSqlOut.setMargin(new java.awt.Insets(10, 10, 10, 10));
jScrollPane2.setViewportViewView(txtSqlOut);

grpFilters.setBorder(javax.swing.BorderFactory.createTitledBorder(null, " Filtros ",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma", 0, 12),
java.awt.Color.blue)); // NOI18N

lblSgbdIn.setText("SGBD de entrada:");

lblSgbdOut.setText("SGBD de saída:");

javax.swing.GroupLayout grpFiltersLayout = new javax.swing.GroupLayout(grpFilters);
grpFilters.setLayout(grpFiltersLayout);
grpFiltersLayout.setHorizontalGroup(
    grpFiltersLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(grpFiltersLayout.createSequentialGroup()
            .addComponent(lblSgbdIn)
            .addComponent(lblSgbdOut)
            .addGap(10, 10, 10)
            .addGroup(grpFiltersLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(cbxSgbdIn, javax.swing.GroupLayout.PREFERRED_SIZE, 250,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(cbxSgbdOut, javax.swing.GroupLayout.PREFERRED_SIZE, 250,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(10, 10, 10))
        );
grpFiltersLayout.setVerticalGroup(
    grpFiltersLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(grpFiltersLayout.createSequentialGroup()
            .addComponent(lblSgbdIn)
            .addComponent(cbxSgbdIn, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(lblSgbdOut)
            .addComponent(cbxSgbdOut, javax.swing.GroupLayout.PREFERRED_SIZE,

```

```

javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
);

    btnConvert.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/br/ufsc/s2s/resources/convert.png"))); // NOI18N
    btnConvert.setText("Converter");
    btnConvert.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnConvertActionPerformed(evt);
        }
    });

    menuFile.setText("File");

    menuItemOpen.setText("Open...");
    menuItemOpen.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            menuItemOpenActionPerformed(evt);
        }
    });
    menuFile.add(menuItemOpen);
    menuFile.add(jSeparator1);

    menuItemQuit.setText("Quit");
    menuItemQuit.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            menuItemQuitActionPerformed(evt);
        }
    });
    menuFile.add(menuItemQuit);

    menuBar.add(menuFile);

    menuEdit.setText("Edit");
    menuBar.add(menuEdit);

    setJMenuBar(menuBar);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 573,
Short.MAX_VALUE)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 573,
Short.MAX_VALUE)
                    .addComponent(grpFilters, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(btnConvert))
                .addContainerGap()
            );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(grpFilters, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 130,

```

```

Short.MAX_VALUE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(btnConvert)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 133,
Short.MAX_VALUE)
    .addContainerGap()
);

pack();
} // </editor-fold>

private void txtSqlInFocusLost(java.awt.event.FocusEvent evt) {
    if (this.txtSqlIn.getText().length() == 0) {
        this.txtSqlIn.setText(this.txtSqlInEmptyText);
    }
}

private void txtSqlInFocusGained(java.awt.event.FocusEvent evt) {
    if (this.txtSqlIn.getText().equalsIgnoreCase(this.txtSqlInEmptyText)) {
        this.txtSqlIn.setText("");
    }
}

private void btnConvertActionPerformed(java.awt.event.ActionEvent evt) {
    boolean error = false;
    String title = null;
    String message = null;

    if (this.cbxSgbdIn.getSelectedIndex() == 0) {
        error = true;
        title = this.invalidParameter;
        message = this.selectSgbdIn;
    } else if (this.cbxSgbdOut.getSelectedIndex() == 0) {
        error = true;
        title = this.invalidParameter;
        message = this.selectSgbdOut;
    } else if (this.txtSqlIn.getText().length() == 0 ||
this.txtSqlIn.getText().equalsIgnoreCase(this.txtSqlInEmptyText)) {
        error = true;
        title = this.invalidSqlIn;
        message = this.insertSqlIn;
    }

    if (error) {
        JOptionPane.showMessageDialog(null, message, title, JOptionPane.ERROR_MESSAGE);
    }
    else {
        String result = this.control.convertSql(this.getSqlIn(), this.getSgbdIn(), this.getSgbdOut());
        this.setSqlOut(result);
    }
}

private void menuItemOpenActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser jfc = new JFileChooser();
    jfc.showOpenDialog(this);
    File file = jfc.getSelectedFile();

    if (file != null) {
        this.setSqlIn(this.control.getFileContent(file));
    }
}

```

```

private void menuItemQuitActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {
            new S2SView().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton btnConvert;
private javax.swing.JComboBox cbxSgbdIn;
private javax.swing.JComboBox cbxSgbdOut;
private javax.swing.JPanel grpFilters;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JLabel lblSgbdIn;
private javax.swing.JLabel lblSgbdOut;
private javax.swing.JMenuBar menuBar;
private javax.swing.JMenu menuEdit;
private javax.swing.JMenu menuFile;
private javax.swing.JMenuItem menuItemOpen;
private javax.swing.JMenuItem menuItemQuit;
private javax.swing.JTextArea txtSqlIn;
private javax.swing.JTextArea txtSqlOut;
// End of variables declaration
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.application;

import br.ufsc.s2s.domain.S2S;
import br.ufsc.s2s.service.Persistence;
import br.ufsc.s2s.view.S2SView;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;

/**
 *
 * @author Renato
 */
public class S2SControl {

    private S2SView view;

    public S2SControl(S2SView view) {
        this.view = view;
    }

    public String getFileContent(File file) {
        String content = null;
        try {
            content = new Persistence().getFileContent(file);
        }
        catch(FileNotFoundException fnfe) {
            System.err.println(fnfe);
        }
        catch (IOException ioe) {
            System.err.println(ioe);
        }

        return content;
    }

    public String convertSql(String sqlIn, int sgbdIn, int sgbdOut) {
        S2S s2s = new S2S();
        String result = null;
        try {
            s2s.convertSql(sqlIn, sgbdIn, sgbdOut);
            result = s2s.getResultAsString();
        }
        catch(Exception error) {
            result = error.getMessage();
            System.err.println(error);
        }

        return this.format(result);
    }

    private String format(String result) {
        String rv = result;

        rv = rv.replaceAll("\n", " ");
        rv = rv.replaceAll("\t", "");

        rv = rv.replace("FROM", "\nFROM");
        rv = rv.replace("WHERE", "\nWHERE");
    }
}

```

```
    rv = rv.replaceAll(", ", ", ");  
    return rv;  
}
```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s;

/**
 *
 * @author Renato
 */
public class _C {

    public static final int MIN_SGBD_ID = 1;
    public static final int MAX_SGBD_ID = 4;

    public static final int SGBD_MSSQL = 1;
    public static final int SGBD_MYSQL = 2;
    public static final int SGBD_POSTGRES = 3;
    public static final int SGBD_FIREBIRD = 4;

    //TODO definir uma variável para a aplicação
    // public static final String XSL_FILE_PATH = "D:/Softdev/NetBeansProjects/S2S/files/xslt/";
    public static final String XSL_FILE_PATH = "/home/renato/SoftDev/NetBeansProjects/S2S/files/xslt/";
    public static final String XSL_MSSQL_FILE_PATH = XSL_FILE_PATH + "SqlServer.xml";
    public static final String XSL_MYSQL_FILE_PATH = XSL_FILE_PATH + "MySql.xml";
    public static final String XSL_POSTGRES_FILE_PATH = XSL_FILE_PATH + "Postgres.xml";
    public static final String XSL_FIREBIRD_FILE_PATH = XSL_FILE_PATH + "Firebird.xml";

    public static final String INVALID_SGBD_IN = "Parâmetro \"sgbdIN\" inválido";
    public static final String INVALID_SGBD_OUT = "Parâmetro \"sgbdOUT\" inválido";
}

```



```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain;

import br.ufsc.s2s._C;
import br.ufsc.s2s.domain.compiler.S2SCompiler;
import br.ufsc.s2s.domain.exception.S2SException;
import br.ufsc.s2s.domain.xslt.S2SXSLT;
import org.jdom.Document;

/**
 *
 * @author Renato
 */
public class S2S {

    private int sgbdOut;
    private Document intermediateCode;

    public String getResultAsString() throws Exception {
        String file = null;

        if (this.sgbdOut == _C.SGBD_FIREBIRD)
            file = _C.XSL_FIREBIRD_FILE_PATH;
        else if (this.sgbdOut == _C.SGBD_MSSQL)
            file = _C.XSL_MSSQL_FILE_PATH;
        else if (this.sgbdOut == _C.SGBD_MYSQL)
            file = _C.XSL_MYSQL_FILE_PATH;
        else if (this.sgbdOut == _C.SGBD_POSTGRES)
            file = _C.XSL_POSTGRES_FILE_PATH;

        if (file == null)
            throw new S2SException(_C.INVALID_SGBD_OUT);

        S2SXSLT parser = new S2SXSLT();
        return parser.transform(this.intermediateCode, file);
    }

    public void convertSql(String sql, int sgbdIn, int sgbdOut) throws Exception {

        if (!(sgbdIn >= _C.MIN_SGBD_ID && sgbdIn <= _C.MAX_SGBD_ID))
            throw new S2SException(_C.INVALID_SGBD_IN);

        if (!(sgbdOut >= _C.MIN_SGBD_ID && sgbdOut <= _C.MAX_SGBD_ID))
            throw new S2SException(_C.INVALID_SGBD_OUT);

        this.sgbdOut = sgbdOut;
        S2SCompiler compiler = S2SCompiler.getInstance(sgbdIn);
        this.intermediateCode = compiler.compile(sql);
    }
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.xslt;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;
import java.util.Properties;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;
import org.jdom.transform.JDOMResult;
import org.jdom.transform.JDOMSource;

/**
 * @author Renato
 */
public class S2SXSLT {

    public void transform(String xmlInFilePath, String xsltInFilePath, String outFilePath)
        throws TransformerException, TransformerConfigurationException, FileNotFoundException {

        String KEY = "javax.xml.transform.TransformerFactory";
        String VALUE = "org.apache.xalan.xsltc.trax.TransformerFactoryImpl";

        Properties props = System.getProperties();
        props.put(KEY, VALUE);
        System.setProperties(props);

        TransformerFactory tFactory = TransformerFactory.newInstance();
        Transformer transformer = tFactory.newTransformer(new StreamSource(xsltInFilePath));

        transformer.transform(new StreamSource(xmlInFilePath), new StreamResult(new
        FileOutputStream(outFilePath)));
    }

    public String transform(Document inputDoc, String xsltInFilePath)
        throws IOException, JDOMException, TransformerConfigurationException, TransformerException {

        SAXBuilder builder = new SAXBuilder();
        Document xsltdoc = builder.build(xsltInFilePath);
        JDOMResult jdomResult = new JDOMResult();
        Transformer obj = TransformerFactory.newInstance().newTransformer(new JDOMSource(xsltdoc));
        obj.transform(new JDOMSource(inputDoc), jdomResult);
        XMLOutputter out = new XMLOutputter(Format.getPrettyFormat());
        List result = jdomResult.getResult();

        return out.outputString(result);
    }
}

```

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.exception;

/**
 *
 * @author Renato
 */
public class S2SException extends RuntimeException {

    public S2SException(String message) {
        super(message);
    }
}
```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler;

import br.ufsc.s2s._C;
import br.ufsc.s2s.domain.compiler.firebird.FirebirdCompiler;
import br.ufsc.s2s.domain.compiler.mssql.MssqlCompiler;
import br.ufsc.s2s.domain.compiler.mysql.MysqlCompiler;
import br.ufsc.s2s.domain.compiler.postgresql.PostgresCompiler;
import org.jdom.Document;

/**
 *
 * @author Renato
 */
public abstract class S2SCompiler {

    public static S2SCompiler getInstance(int sgbdCompiler) {
        S2SCompiler instance = null;

        if (sgbdCompiler == _C.SGBD_FIREBIRD) {
            instance = new FirebirdCompiler();
        }
        else if (sgbdCompiler == _C.SGBD_MSSQL) {
            instance = new MssqlCompiler();
        }
        else if (sgbdCompiler == _C.SGBD_MYSQL) {
            instance = new MysqlCompiler();
        }
        else if (sgbdCompiler == _C.SGBD_POSTGRES) {
            instance = new PostgresCompiler();
        }

        return instance;
    }

    public abstract Document compile(String sourceCode) throws Exception;
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.firebird;

import br.ufsc.s2s.domain.compiler.S2SCompiler;
import br.ufsc.s2s.domain.compiler.firebird.gals.Lexical;
import br.ufsc.s2s.domain.compiler.firebird.gals.Syntactic;
import org.jdom.Document;

/**
 *
 * @author Renato
 */
public class FirebirdCompiler extends S2SCompiler {

    @Override
    public Document compile(String sourceCode) throws Exception {
        Document xmlDocument = null;

        Lexical lexical = new Lexical();
        Syntactic syntactic = new Syntactic();

        lexical.setInput(sourceCode);

        FirebirdSemanticImpl semantic = new FirebirdSemanticImpl();
        syntactic.parse(lexical, semantic);
        xmlDocument = semantic.getXmlDocument();

        return xmlDocument;
    }
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.firebird;

import br.ufsc.s2s.domain.compiler.firebird.gals.*;
import java.io.FileWriter;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.output.XMLOutputter;

/**
 *
 * @author Renato
 */
public class FirebirdSemanticImpl extends Semantic {

    private Token token = null;

    private Document xmlDocument = null;
    private Element elSql = null;
    private Element elSelect = null;
    private Element elFrom = null;
    private Element elWhere = null;

    private Element currentParam = null;
    private Element currentConditional = null;
    private Element currentConcat = null;

    private StringBuilder tempString = null;

    public FirebirdSemanticImpl() {
        this.elSql = new Element("sql");
        this.xmlDocument = new Document(this.elSql);
        this.xmlDocument.setRootElement(this.elSql);
    }

    public Document getXmlDocument() {
        return this.xmlDocument;
    }

    @Override
    public void executeAction(int action, Token token) throws SemanticError {
        this.token = token;

        switch (action) {
            case 100: this.executeAction_100(); break;
            case 101: this.executeAction_101(); break;
            case 102: this.executeAction_102(); break;
            case 103: this.executeAction_103(); break;
            case 104: this.executeAction_104(); break;
            case 105: this.executeAction_105(); break;
            case 106: this.executeAction_106(); break;
            case 107: this.executeAction_107(); break;
            case 108: this.executeAction_108(); break;
            case 109: this.executeAction_109(); break;
            case 110: this.executeAction_110(); break;
            case 111: this.executeAction_111(); break;
            case 112: this.executeAction_112(); break;
            case 113: this.executeAction_113(); break;
            case 114: this.executeAction_114(); break;
        }
    }
}

```

```

        case 115: this.executeAction_115(); break;
        case 116: this.executeAction_116(); break;
        case 117: this.executeAction_117(); break;
        case 118: this.executeAction_118(); break;
        case 119: this.executeAction_119(); break;
        case 999: this.executeAction_999(); break;
    }
}

private void executeAction_100() {
    this.elSelect = new Element("select");
    this.elSql.addContent(this.elSelect);
}

private void executeAction_101() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_102() {
    this.elFrom = new Element("from");
    this.elSql.addContent(this.elFrom);
}

private void executeAction_103() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_104() {
    Element elRows = new Element("rows");
    elRows.setText(this.token.getLexeme());
    this.elSql.addContent(0, elRows);
}

private void executeAction_105() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_106() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_107() {
    this.elWhere = new Element("where");
    Element elConditional = new Element("conditional");
    this.elWhere.addContent(elConditional);
    this.elSql.addContent(this.elWhere);
    this.currentConditional = elConditional;
}

```

```

private void executeAction_108() {
    Element elAntecedent = new Element("antecedent");
    elAntecedent.setText(this.token.getLexeme());
    this.currentConditional.addContent(elAntecedent);
}

private void executeAction_109() {
    this.currentConditional.setAttribute("operator", this.token.getLexeme());
}

private void executeAction_110() {
    Element elAntecedent = new Element("consequent");
    elAntecedent.setText(this.token.getLexeme());
    this.currentConditional.addContent(elAntecedent);
}

private void executeAction_111() {
    this.elSelect.setAttribute("distinct", "true");
}

private void executeAction_112() {
    this.elSelect.setAttribute("distinct", "false");
}

private void executeAction_113() {
    this.currentParam.setAttribute("alias", this.token.getLexeme());
}

private void executeAction_114() {
    Element elConditional = new Element("conditional");
    elConditional.setAttribute("logical_operator", " " + this.token.getLexeme().toUpperCase() + " ");
    this.elWhere.addContent(elConditional);
    this.currentConditional = elConditional;
}

private void executeAction_115() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", " inner join ");
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
    this.tempString = new StringBuilder();
}

private void executeAction_116() {
    String lex = (this.token.getLexeme().equalsIgnoreCase("on") ? " on " : this.token.getLexeme());
    this.tempString.append(lex);
}

private void executeAction_117() {
    this.tempString.append(this.token.getLexeme());
    this.currentParam.setAttribute("suffix", this.tempString.toString());
}

private void executeAction_118() {
    Element elParam = new Element("param");
    this.elSelect.addContent(elParam);
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("concatIndex", "1");
    this.currentParam = elParam;
}

private void executeAction_119() {

```



```

        Element elParam = new Element("param");
        this.elSelect.addContent(elParam);
        elParam.setText(this.token.getLexeme());
        elParam.setAttribute("concatIndex", "2");
        this.currentParam = elParam;
    }

private void executeAction_999() {
    // ***** Escrevendo a árvore xml em um arquivo *****
    try{
        XMLOutputter out = new XMLOutputter();
        FileWriter writer = new FileWriter("/home/renato/exemplo.xml");

        out.output(this.xmlDocument, writer);
        writer.flush();
        writer.close();
    } catch(Exception e) {
        System.err.println(e);
    }
}
}
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.mssql;

import br.ufsc.s2s.domain.compiler.S2SCompiler;
import br.ufsc.s2s.domain.compiler.mssql.gals.Lexical;
import br.ufsc.s2s.domain.compiler.mssql.gals.Syntactic;
import org.jdom.Document;

/**
 *
 * @author Renato
 */
public class MssqlCompiler extends S2SCompiler {

    @Override
    public Document compile(String sourceCode) throws Exception {
        Document xmlDocument = null;

        Lexical lexical = new Lexical();
        Syntactic syntactic = new Syntactic();

        lexical.setInput(sourceCode);

        MssqlSemanticImpl semantic = new MssqlSemanticImpl();
        syntactic.parse(lexical, semantic);
        xmlDocument = semantic.getXmlDocument();

        return xmlDocument;
    }
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.mssql;

import br.ufsc.s2s.domain.compiler.mssql.gals.*;
import java.io.FileWriter;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.output.XMLOutputter;

/**
 *
 * @author Renato
 */
public class MssqlSemanticImpl extends Semantic {

    private Token token = null;

    private Document xmlDocument = null;
    private Element elSql = null;
    private Element elSelect = null;
    private Element elFrom = null;
    private Element elWhere = null;

    private Element currentParam = null;
    private Element currentConditional = null;

    private StringBuilder tempString = null;

    public MssqlSemanticImpl() {
        this.elSql = new Element("sql");
        this.xmlDocument = new Document(this.elSql);
        this.xmlDocument.setRootElement(this.elSql);
    }

    public Document getXmlDocument() {
        return this.xmlDocument;
    }

    @Override
    public void executeAction(int action, Token token) throws SemanticError {
        this.token = token;

        switch (action) {
            case 100: this.executeAction_100(); break;
            case 101: this.executeAction_101(); break;
            case 102: this.executeAction_102(); break;
            case 103: this.executeAction_103(); break;
            case 104: this.executeAction_104(); break;
            case 105: this.executeAction_105(); break;
            case 106: this.executeAction_106(); break;
            case 107: this.executeAction_107(); break;
            case 108: this.executeAction_108(); break;
            case 109: this.executeAction_109(); break;
            case 110: this.executeAction_110(); break;
            case 111: this.executeAction_111(); break;
            case 112: this.executeAction_112(); break;
            case 113: this.executeAction_113(); break;
            case 114: this.executeAction_114(); break;
            case 115: this.executeAction_115(); break;
        }
    }
}

```

```

        case 116: this.executeAction_116(); break;
        case 117: this.executeAction_117(); break;
        case 118: this.executeAction_118(); break;
        case 119: this.executeAction_119(); break;
        case 999: this.executeAction_999(); break;
    }
}

private void executeAction_100() {
    this.elSelect = new Element("select");
    this.elSql.addContent(this.elSelect);
}

private void executeAction_101() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_102() {
    this.elFrom = new Element("from");
    this.elSql.addContent(this.elFrom);
}

private void executeAction_103() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_104() {
    Element elRows = new Element("rows");
    elRows.setText(this.token.getLexeme());
    this.elSql.addContent(0, elRows);
}

private void executeAction_105() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_106() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_107() {
    this.elWhere = new Element("where");
    Element elConditional = new Element("conditional");
    this.elWhere.addContent(elConditional);
    this.elSql.addContent(this.elWhere);
    this.currentConditional = elConditional;
}

private void executeAction_108() {

```

```

        Element elAntecedent = new Element("antecedent");
        elAntecedent.setText(this.token.getLexeme());
        this.currentConditional.addContent(elAntecedent);
    }

    private void executeAction_109() {
        this.currentConditional.setAttribute("operator", this.token.getLexeme());
    }

    private void executeAction_110() {
        Element elAntecedent = new Element("consequent");
        elAntecedent.setText(this.token.getLexeme());
        this.currentConditional.addContent(elAntecedent);
    }

    private void executeAction_111() {
        this.elSelect.setAttribute("distinct", "true");
    }

    private void executeAction_112() {
        this.elSelect.setAttribute("distinct", "false");
    }

    private void executeAction_113() {
        this.currentParam.setAttribute("alias", this.token.getLexeme());
    }

    private void executeAction_114() {
        Element elConditional = new Element("conditional");
        elConditional.setAttribute("logical_operator", " " + this.token.getLexeme().toUpperCase() + " ");
        this.elWhere.addContent(elConditional);
        this.currentConditional = elConditional;
    }

    private void executeAction_115() {
        Element elParam = new Element("param");
        elParam.setText(this.token.getLexeme());
        elParam.setAttribute("prefix", " inner join ");
        this.elFrom.addContent(elParam);
        this.currentParam = elParam;
        this.tempString = new StringBuilder();
    }

    private void executeAction_116() {
        String lex = (this.token.getLexeme().equalsIgnoreCase("on") ? " on " : this.token.getLexeme());
        this.tempString.append(lex);
    }

    private void executeAction_117() {
        this.tempString.append(this.token.getLexeme());
        this.currentParam.setAttribute("suffix", this.tempString.toString());
    }

    private void executeAction_118() {
        Element elParam = new Element("param");
        this.elSelect.addContent(elParam);
        elParam.setText(this.token.getLexeme());
        elParam.setAttribute("concatIndex", "1");
        this.currentParam = elParam;
    }

    private void executeAction_119() {
        Element elParam = new Element("param");

```

```
this.elSelect.addContent(elParam);
elParam.setText(this.token.getLexeme());
elParam.setAttribute("concatIndex", "2");
this.currentParam = elParam;
}

private void executeAction_999() {
// ***** Escrevendo a árvore xml em um arquivo *****
try{
XMLOutputter out = new XMLOutputter();
FileWriter writer = new FileWriter("/home/renato/exemplo.xml");

out.output(this.xmlDocument, writer);
writer.flush();
writer.close();
} catch(Exception e) {
System.err.println(e);
}
}
}
```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.mysql;

import br.ufsc.s2s.domain.compiler.S2SCompiler;
import br.ufsc.s2s.domain.compiler.mysql.gals.Lexical;
import br.ufsc.s2s.domain.compiler.mysql.gals.Syntactic;
import org.jdom.Document;

/**
 *
 * @author Renato
 */
public class MysqlCompiler extends S2SCompiler {

    @Override
    public Document compile(String sourceCode) throws Exception {
        Document xmlDocument = null;

        Lexical lexical = new Lexical();
        Syntactic syntactic = new Syntactic();

        lexical.setInput(sourceCode);

        MysqlSemanticImpl semantic = new MysqlSemanticImpl();
        syntactic.parse(lexical, semantic);
        xmlDocument = semantic.getXmlDocument();

        return xmlDocument;
    }
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.mysql;

import br.ufsc.s2s.domain.compiler.mysql.gals.*;
import java.io.FileWriter;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.output.XMLOutputter;

/**
 *
 * @author Renato
 */
public class MysqlSemanticImpl extends Semantic {

    private Token token = null;

    private Document xmlDocument = null;
    private Element elSql = null;
    private Element elSelect = null;
    private Element elFrom = null;
    private Element elWhere = null;

    private Element currentParam = null;
    private Element currentConditional = null;

    private StringBuilder tempString = null;

    public MysqlSemanticImpl() {
        this.elSql = new Element("sql");
        this.xmlDocument = new Document(this.elSql);
        this.xmlDocument.setRootElement(this.elSql);
    }

    public Document getXmlDocument() {
        return this.xmlDocument;
    }

    @Override
    public void executeAction(int action, Token token) throws SemanticError {
        this.token = token;

        switch (action) {
            case 100: this.executeAction_100(); break;
            case 101: this.executeAction_101(); break;
            case 102: this.executeAction_102(); break;
            case 103: this.executeAction_103(); break;
            case 104: this.executeAction_104(); break;
            case 105: this.executeAction_105(); break;
            case 106: this.executeAction_106(); break;
            case 107: this.executeAction_107(); break;
            case 108: this.executeAction_108(); break;
            case 109: this.executeAction_109(); break;
            case 110: this.executeAction_110(); break;
            case 111: this.executeAction_111(); break;
            case 112: this.executeAction_112(); break;
            case 113: this.executeAction_113(); break;
            case 114: this.executeAction_114(); break;
        }
    }
}

```



```

        case 115: this.executeAction_115(); break;
        case 116: this.executeAction_116(); break;
        case 117: this.executeAction_117(); break;
        case 118: this.executeAction_118(); break;
        case 119: this.executeAction_119(); break;
        case 999: this.executeAction_999(); break;
    }
}

private void executeAction_100() {
    this.elSelect = new Element("select");
    this.elSql.addContent(this.elSelect);
}

private void executeAction_101() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_102() {
    this.elFrom = new Element("from");
    this.elSql.addContent(this.elFrom);
}

private void executeAction_103() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_104() {
    Element elRows = new Element("rows");
    elRows.setText(this.token.getLexeme());
    this.elSql.addContent(0, elRows);
}

private void executeAction_105() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_106() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_107() {
    this.elWhere = new Element("where");
    Element elConditional = new Element("conditional");
    this.elWhere.addContent(elConditional);
    this.elSql.addContent(this.elWhere);
    this.currentConditional = elConditional;
}

```

```

private void executeAction_108() {
    Element elAntecedent = new Element("antecedent");
    elAntecedent.setText(this.token.getLexeme());
    this.currentConditional.addContent(elAntecedent);
}

private void executeAction_109() {
    this.currentConditional.setAttribute("operator", this.token.getLexeme());
}

private void executeAction_110() {
    Element elAntecedent = new Element("consequent");
    elAntecedent.setText(this.token.getLexeme());
    this.currentConditional.addContent(elAntecedent);
}

private void executeAction_111() {
    this.elSelect.setAttribute("distinct", "true");
}

private void executeAction_112() {
    this.elSelect.setAttribute("distinct", "false");
}

private void executeAction_113() {
    this.currentParam.setAttribute("alias", this.token.getLexeme());
}

private void executeAction_114() {
    Element elConditional = new Element("conditional");
    elConditional.setAttribute("logical_operator", " " + this.token.getLexeme().toUpperCase() + " ");
    this.elWhere.addContent(elConditional);
    this.currentConditional = elConditional;
}

private void executeAction_115() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", " inner join ");
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
    this.tempString = new StringBuilder();
}

private void executeAction_116() {
    String lex = (this.token.getLexeme().equalsIgnoreCase("on") ? " on " : this.token.getLexeme());
    this.tempString.append(lex);
}

private void executeAction_117() {
    this.tempString.append(this.token.getLexeme());
    this.currentParam.setAttribute("suffix", this.tempString.toString());
}

private void executeAction_118() {
    Element elParam = new Element("param");
    this.elSelect.addContent(elParam);
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("concatIndex", "1");
    this.currentParam = elParam;
}

private void executeAction_119() {

```

```

        Element elParam = new Element("param");
        this.elSelect.addContent(elParam);
        elParam.setText(this.token.getLexeme());
        elParam.setAttribute("concatIndex", "2");
        this.currentParam = elParam;
    }

private void executeAction_999() {
    // ***** Escrevendo a árvore xml em um arquivo *****
    try{
        XMLOutputter out = new XMLOutputter();
        FileWriter writer = new FileWriter("/home/renato/exemplo.xml");

        out.output(this.xmlDocument, writer);
        writer.flush();
        writer.close();
    } catch(Exception e) {
        System.err.println(e);
    }
}
}
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.postgresql;

import br.ufsc.s2s.domain.compiler.S2SCompiler;
import br.ufsc.s2s.domain.compiler.postgres.gals.Lexical;
import br.ufsc.s2s.domain.compiler.postgres.gals.Syntactic;
import org.jdom.Document;

/**
 *
 * @author Renato
 */
public class PostgresCompiler extends S2SCompiler {

    @Override
    public Document compile(String sourceCode) throws Exception {
        Document xmlDocument = null;

        Lexical lexical = new Lexical();
        Syntactic syntactic = new Syntactic();

        lexical.setInput(sourceCode);

        PostgresSemanticImpl semantic = new PostgresSemanticImpl();
        syntactic.parse(lexical, semantic);
        xmlDocument = semantic.getXmlDocument();

        return xmlDocument;
    }
}

```

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.domain.compiler.postgresql;

import br.ufsc.s2s.domain.compiler.postgres.gals.*;
import java.io.FileWriter;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.output.XMLOutputter;

/**
 *
 * @author Renato
 */
public class PostgresSemanticImpl extends Semantic {

    private Token token = null;

    private Document xmlDocument = null;
    private Element elSql = null;
    private Element elSelect = null;
    private Element elFrom = null;
    private Element elWhere = null;

    private Element currentParam = null;
    private Element currentConditional = null;

    private StringBuilder tempString = null;

    public PostgresSemanticImpl() {
        this.elSql = new Element("sql");
        this.xmlDocument = new Document(this.elSql);
        this.xmlDocument.setRootElement(this.elSql);
    }

    public Document getXmlDocument() {
        return this.xmlDocument;
    }

    @Override
    public void executeAction(int action, Token token) throws SemanticError {
        this.token = token;

        switch (action) {
            case 100: this.executeAction_100(); break;
            case 101: this.executeAction_101(); break;
            case 102: this.executeAction_102(); break;
            case 103: this.executeAction_103(); break;
            case 104: this.executeAction_104(); break;
            case 105: this.executeAction_105(); break;
            case 106: this.executeAction_106(); break;
            case 107: this.executeAction_107(); break;
            case 108: this.executeAction_108(); break;
            case 109: this.executeAction_109(); break;
            case 110: this.executeAction_110(); break;
            case 111: this.executeAction_111(); break;
            case 112: this.executeAction_112(); break;
            case 113: this.executeAction_113(); break;
            case 114: this.executeAction_114(); break;
        }
    }
}

```

```

        case 115: this.executeAction_115(); break;
        case 116: this.executeAction_116(); break;
        case 117: this.executeAction_117(); break;
        case 118: this.executeAction_118(); break;
        case 119: this.executeAction_119(); break;
        case 999: this.executeAction_999(); break;
    }
}

private void executeAction_100() {
    this.elSelect = new Element("select");
    this.elSql.addContent(this.elSelect);
}

private void executeAction_101() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_102() {
    this.elFrom = new Element("from");
    this.elSql.addContent(this.elFrom);
}

private void executeAction_103() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_104() {
    Element elRows = new Element("rows");
    elRows.setText(this.token.getLexeme());
    this.elSql.addContent(0, elRows);
}

private void executeAction_105() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elSelect.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_106() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", ",");
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
}

private void executeAction_107() {
    this.elWhere = new Element("where");
    Element elConditional = new Element("conditional");
    this.elWhere.addContent(elConditional);
    this.elSql.addContent(this.elWhere);
    this.currentConditional = elConditional;
}

```

```

private void executeAction_108() {
    Element elAntecedent = new Element("antecedent");
    elAntecedent.setText(this.token.getLexeme());
    this.currentConditional.addContent(elAntecedent);
}

private void executeAction_109() {
    this.currentConditional.setAttribute("operator", this.token.getLexeme());
}

private void executeAction_110() {
    Element elAntecedent = new Element("consequent");
    elAntecedent.setText(this.token.getLexeme());
    this.currentConditional.addContent(elAntecedent);
}

private void executeAction_111() {
    this.elSelect.setAttribute("distinct", "true");
}

private void executeAction_112() {
    this.elSelect.setAttribute("distinct", "false");
}

private void executeAction_113() {
    this.currentParam.setAttribute("alias", this.token.getLexeme());
}

private void executeAction_114() {
    Element elConditional = new Element("conditional");
    elConditional.setAttribute("logical_operator", " " + this.token.getLexeme().toUpperCase() + " ");
    this.elWhere.addContent(elConditional);
    this.currentConditional = elConditional;
}

private void executeAction_115() {
    Element elParam = new Element("param");
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("prefix", " inner join ");
    this.elFrom.addContent(elParam);
    this.currentParam = elParam;
    this.tempString = new StringBuilder();
}

private void executeAction_116() {
    String lex = (this.token.getLexeme().equalsIgnoreCase("on") ? " on " : this.token.getLexeme());
    this.tempString.append(lex);
}

private void executeAction_117() {
    this.tempString.append(this.token.getLexeme());
    this.currentParam.setAttribute("suffix", this.tempString.toString());
}

private void executeAction_118() {
    Element elParam = new Element("param");
    this.elSelect.addContent(elParam);
    elParam.setText(this.token.getLexeme());
    elParam.setAttribute("concatIndex", "1");
    this.currentParam = elParam;
}

private void executeAction_119() {

```

```
        Element elParam = new Element("param");
        this.elSelect.addContent(elParam);
        elParam.setText(this.token.getLexeme());
        elParam.setAttribute("concatIndex", "2");
        this.currentParam = elParam;
    }

private void executeAction_999() {
    // ***** Escrevendo a árvore xml em um arquivo *****
    try{
        XMLOutputter out = new XMLOutputter();
        FileWriter writer = new FileWriter("/home/renato/exemplo.xml");

        out.output(this.xmlDocument, writer);
        writer.flush();
        writer.close();
    } catch(Exception e) {
        System.err.println(e);
    }
}
}
```



```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.ufsc.s2s.service;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

/**
 *
 * @author renato
 */
public class Persistence {

    public String getFileContent(File file) throws FileNotFoundException, IOException {
        StringBuilder sb = null;

        if (file != null) {
            sb = new StringBuilder();

            BufferedReader bf = new BufferedReader(new FileReader(file));
            while (bf.ready()) {
                sb.append(bf.readLine());
                sb.append("\n");
            }
            bf.close();
        }

        return sb == null ? null : sb.toString();
    }
}

```