UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

SISTEMA PARA APRENDIZADO DE ÁLGEBRA RELACIONAL E LINGUAGEM SQL

ANDRÉ MARTINS DOS SANTOS

BLUMENAU
2010

2010/2-07

ANDRÉ MARTINS DOS SANTOS

SISTEMA PARA APRENDIZADO DE ÁLGEBRA RELACIONAL E LINGUAGEM SQL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Roosevelt dos Santos Junior, Orientador

BLUMENAU
2010

2010/2-07

SISTEMA PARA APRENDIZADO DE ÁLGEBRA RELACIONAL E LINGUAGEM SQL

Por

ANDRÉ MARTINS DOS SANTOS

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Roosevelt dos Santos Junior, Orientador, FURB
	_____
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB
	_____
Membro:	Prof. Marcos Rogério Cardoso, Especialista – FURB

Blumenau, 9 de dezembro de 2010

Dedico este trabalho a minha família e todos os amigos, especialmente aqueles que me incentivaram para realização deste.

AGRADECIMENTOS

A Deus, pela saúde.

À minha mãe, Sumara, pelo apoio e paciência.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, Roosevelt, por ter acreditado na conclusão deste trabalho.

A curiosidade é mais importante do que o conhecimento.

Albert Einstein

RESUMO

Este trabalho descreve o projeto e o desenvolvimento de uma ferramenta educacional voltada para o ensino de álgebra relacional, permitindo que sejam construídos planos de consulta na forma de árvores de expressão algébricas. Os planos de consultas são transformados em linguagem SQL e podem ser executados nos bancos de dados Oracle XE e MySQL 5.5. O desenvolvimento do trabalho foi realizado utilizando a linguagem Java versão 6 em conjunto como compilador Eclipse. O acesso ao banco de dados foi obtido através do *driver* JDBC.

Palavras-chave: Banco de dados. Álgebra relacional. Software de ensino.

ABSTRACT

This work shows the design and development of an educational tool focused on the teaching relational algebra concepts, allowing them to be constructed query plans in the form of algebric expression trees. The query plans are transformed into SQL and can run on Oracle Database XE and MySQL 5.5. The development work was performed using the Java version 6 compiler and Eclipse together. Access to the database was obtained through the JDBC driver.

Key-words: Database. Relational algebra. Teaching software.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

BNF – Backus-Naur *Form*

GALS – Gerador de Analisadores Léxicos e Sintáticos

IBM – *International Business Machines*

JDBC – *Java Database Connectivity*

MVC – *Model View Controller*

SQL – *Standard Query Language*

SGBDR – Sistemas Gerenciadores de Bancos de Dados Relacionais

SEQUEL – *Structured English Query Language*

SDK – *Software Development Kit*

UML – *Unified Modelling Language*

LISTA DE SÍMBOLOS

$\cup$ - união

$\cap$ - intersecção

$-$ - diferença

π - projeção

σ - seleção

ρ - rebatizar

γ - agrupamento

τ - ordenação

$\times$ - produto cartesiano

$\bowtie$ - junção

SUMÁRIO

1 INTRODUÇÃO

O ensino da linguagem SQL é abordado na maioria dos cursos de Ciência da Computação e Sistemas de Informação, conforme pode ser observado abaixo nas diretrizes curriculares destes cursos.

> Os tópicos cobertos devem abordar problemas relativos aos dados propriamente ditos (organização, modelagem, integridade, armazenamento, integração, distribuição e empacotamento) e aos sistemas de gerenciamento de bancos de dados - SGBD (arquitetura, interfaces, **linguagens de interação**, processamento de consultas, controle de concorrência, recuperação, segurança, indexação, gerenciamento de buffers e arquivos). (MINISTÉRIO DA EDUCAÇÃO, 1998, p. 10, grifo nosso).

Em teoria também são lecionados os conceitos da álgebra relacional.

Consultas construídas em SQL podem tornar-se bastante complexas e combinadas com o alto nível da linguagem geralmente dificulta a assimilação dos estudantes. Uso da álgebra relacional pode facilitar o aprendizado fornecendo outras maneiras de resolução e conforme evidenciado por Garcia-Molina, Ullman e Widon (2001, p. 254), “Uma das vantagens de usar álgebra relacional é que ela torna fácil explorar formas alternativas de uma consulta.”

Conhecimentos sobre álgebra relacional nos permitem entender a execução e otimização de consultas em Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR), característica importante apontada por Sumathi e Esakkirajan (2007, p. 72).

Em álgebra relacional pode-se especificar consultas utilizando um conjunto de operadores, cada operador possui um símbolo exclusivo e a concatenação de um ou mais operadores sobre uma relação (tabela) formam uma expressão algébrica. Expressões algébricas possuem uma notação bastante abstrata, contudo podem ser representadas graficamente através de árvores de expressões. As expressões e árvores de expressões são abordadas e exemplificadas respectivamente nas seções 2.2 e 2.3.

Uma alternativa de potencializar o aprendizado seria a união dos conceitos de álgebra relacional e SQL em uma única ferramenta. Propõe-se então construir um editor para auxiliar no ensino de álgebra relacional e facilitar a compreensão da linguagem SQL. A ferramenta proposta será uma opção gráfica para a construção de consultas SQL na forma de árvores de expressões. Além disso, poderá contribuir para melhorar o entendimento do funcionamento de um banco de dados.

Em sala de aula um software educativo auxilia o professor didaticamente proporcionando resultado rápido e concreto para as questões abordadas. Os alunos interagindo

com computador e observando a aplicação prática da álgebra relacional deixam de levar consigo apenas o conhecimento teórico após o termino da disciplina.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é disponibilizar um sistema para aprendizado de álgebra relacional e linguagem SQL (Processo básicos de consulta).

Os objetivos específicos do trabalho são:

a) disponibilizar um editor gráfico para construção de árvores de expressões algébricas;
b) realizar a conversão das árvores de expressões algébricas em linguagem SQL;
c) executar a linguagem SQL nos bancos de dados relacionais Oracle e MySQL.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos. O segundo capítulo corresponde a fundamentação teórica fornecendo base para seu entendimento. O capítulo 3 aborda as etapas do desenvolvimento e também a especificação da aplicação. Por fim, no capítulo 4 é apresentado às conclusões e as possíveis extensões para o mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 MODELO RELACIONAL

De acordo com Sumathi e Esakkirajan (2007, p. 67), o modelo relacional utiliza uma coleção de tabelas para representar os dados e os relacionamentos entre esses dados. Tabelas são estruturas lógicas mantidas por um gerente de banco de dados.

O modelo relacional é composto por três elementos sendo o primeiro o estrutural, o qual define o banco de dados como uma coleção de relações. O segundo é relacionado à integridade dos dados mantido através de chaves primarias e estrangeiras e o terceiro referente à manipulação dos dados.

Dentre as características do modelo relacional pode-se citar:

a) cada linha em uma tabela é chamada de `tupla`;
b) cada coluna em uma tabela é chamada de atributo;
c) a intersecção de uma linha com uma coluna deve conter um valor;
d) no modelo relacional linhas podem estar em qualquer ordem;
e) no modelo relacional colunas podem estar em qualquer ordem;
f) por definição todas as linhas em uma tabela são distintas;
g) a relação precisa possuir uma chave a qual pode ser um conjunto de atributos;
h) para cada coluna de uma tabela deve existir um conjunto de possíveis valores chamado de domínio. O domínio contem todos os valores que podem aparecer na coluna;
i) domínio é um conjunto de valores validos para um atributo;
j) o grau de uma relação é o numero de atributos (colunas) contidos na relação;
k) a cardinalidade de uma relação é o número de `tuplas` (linhas) existentes na relação.

2.2 ÁLGEBRA RELACIONAL

Conforme Ramakrishnan (1998, p. 154), álgebra relacional é uma das duas linguagens formais de consulta associadas ao modelo relacional. Consultas em álgebra são construídas utilizando uma coleção de operadores. Cada operador aceita uma ou duas relações como parâmetro e retorna uma relação como resultado, esta é principal propriedade e torna fácil a composição de consultas complexas.

Aplicações como SGBDRs utilizam a álgebra relacional para representar internamente as consultas SQL construídas por usuários. (GARCIA-MOLINA; ULLMAN; WIDON, 2001, p. 254).

Outra característica importante informada por Elmasri e Navathe (2005, p. 106) é que a álgebra relacional "é usada como uma base para desenvolver e otimizar as consultas em SGDBRs."

2.2.1 Operadores

Em relação às operações utilizadas na álgebra relacional, pode-se defini-las em dois grupos. Um grupo inclui as operações derivadas da teoria dos conjuntos da matemática, quais sejam: união, intersecção, diferença e produto cartesiano. O outro grupo compõe as operações desenvolvidas diretamente para os SGBDRs, como seleção, projeção, junção, rebatizar e divisão. (ELMASRI; NAVATHE, 2005, p. 107).

Existem ainda os operadores classificados como estendidos ou adicionais como ordenação, funções de agrupamento e agregação.

Nas seções subseqüentes são apresentados os operadores. Inicialmente são abordados tradicionais, na seqüência os desenvolvidos para SGBDRs e em seguida os que compreendem a álgebra relacional estendida.

2.2.2 União, intersecção e diferença

Elmasri e Navathe (2005, p. 110) descrevem que os operadores união, intersecção e diferença na álgebra relacional clássica têm seu funcionamento igual ao da teoria dos conjuntos da matemática. Estas são operações binárias, isto é, cada uma é aplicada a duas relações e são definidas da seguinte forma:

a) união: o resultado desta operação, indicada por `R ∪ S`, é uma relação que inclui todas as `tuplas` que estão em `R`, ou em `S`, ou em ambas, `R` e `S`. As `tuplas` repetidas são eliminadas;

b) intersecção: o resultado desta operação indicada por `R ∩ S`, é uma relação que inclui todas as `tuplas` que estão em ambas, `R` e `S`;

c) diferença: o resultado desta operação indicada por `R – S`, é uma relação que inclui todas as `tuplas` que estão em `R`, mas não estão em `S`.

O Quadro 1 demonstra um exemplo de cada uma das operações.

a)

ALUNO	PN	UN
	Susan	Yao
	Ramesh	Shah
	Johnny	Kohier
	Barbara	Jones
	Amy	Ford
	Jimmy	Wang
	Ernest	Gilbert

INSTRUTOR	PNOME	UNOME
	John	Smith
	Ricardo	Browne
	Susan	Yao
	Francis	Johnson
	Ramesh	Shah

b)

PN	UN
Susan	Yao
Ramesh	Shah
Johnny	Kohier
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

c)

PN	UN
Susan	Yao
Ramesh	Shah

d)

PN	UN
Johnny	Kohier
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

e)

PNOME	UNOME
John	Smith
Ricardo	Browne
Francis	Johnson

Fonte: Elmasri e Navathe, 2005 (2005, p. 112).

Quadro 1 – As operações união, intersecção e diferença. a) Duas relações de união compatíveis. b) `ALUNO ∪ INSTRUTOR`. c) `ALUNO ∩ INSTRUTOR`. d) `ALUNO – INSTRUTOR`. e) `INSTRUTOR – ALUNO`

Quando qualquer uma das três operações for utilizada deve-se também respeitar a regra de compatibilidade da união, ou seja, as duas relações devem ter o mesmo número de atributos e estes devem ser do mesmo tipo.

2.2.3 Produto Cartesiano

Segundo Elmasri e Navathe (2005, p. 112) a operação Produto Cartesiano é referenciada pelo símbolo ×, a mesma também atua sobre duas relações, contudo não é necessária a compatibilidade da união para sua utilização. Sua função é combinar as `tuplas` de duas relações na forma combinatória, desta forma o resultado de `R × S` é uma relação `Q` que terá a soma dos atributos de `R` e `S` e o número de `tuplas` igual a quantidade de `tuplas` de `R` multiplicada pela quantidade de `tuplas` de `S`. O Quadro 2 demonstra um exemplo deste operador.

Seja a relação R (a,b) :

a	b
1	João
2	Marcos

Seja a relação S (d,e) :

d	e
1	Maria
2	Claudio

O resultado de R × S :

a	b	d	e
1	João	1	Maria
1	João	2	Claudio
2	Marcos	1	Maria
2	Marcos	2	Claudio

Quadro 2 – Exemplo do operador produto cartesiano

2.2.4 Rebatizar

A operação Rebatizar ou *rename* tem como objetivo rebatizar o nome de uma relação, ou os nomes de seus atributos. Esta técnica pode auxiliar o uso das operações como União e Junção evitando incompatibilidade de operadores e ambigüidade em nomes de atributos.

Pode-se aplicar a operação Rebatizar de três maneiras, sendo elas, `ρS(R)`,

ρ(b1,b2,b3...,bn)(R) ou ρS(b1,b2,b3...,bn)(R), onde símbolo ρ (rho) indica a operação rebatizar, S, o nome da nova relação, e b1,b2...,BN são os novos nomes dos atributos. A primeira expressão rebatiza apenas a relação, a segunda, apenas os atributos da relação R, e a terceira, a relação e seus atributos. (ELMASRI; NAVATHE, 2005, p. 110). O Quadro 3 demonstra um exemplo deste operador.

Seja a relação R (a,b,c) :

a	b	c
1	João	20
2	Marcos	21
3	Maria	23

O resultado de ρ(código,nome,idade) (R):

código	nome	idade
1	João	20
2	Marcos	21
3	Maria	23

Quadro 3 – Exemplo do operador rebatizar

2.2.5 Seleção

O operador Seleção tem como objetivo selecionar um determinado conjunto de tuplas de uma relação respeitando uma condição de seleção. Pode ser visualizada como uma divisão de uma relação em dois conjuntos, aquele conjunto que satisfaça a condição e é apresentado, e o conjunto que não satisfaz a condição e é descartado.

A especificação da operação Seleção tem o formato σ <condição de seleção> (R) onde símbolo σ (sigma) indica a operação, e a condição de seleção é uma expressão booleana definida a partir dos atributos da relação R. Verifica-se que R no seu formato mais simples é diretamente o nome de uma relação do banco de dados, contudo pode ser uma expressão de álgebra relacional cujo resultado é uma relação. (ELMASRI; NAVATHE, 2005, p. 107). O Quadro 4 demonstra um exemplo deste operador.

Seja a relação R (a,b,c) :

a	b	c
1	João	20
2	Marcos	21
3	Maria	23

O resultado de $\sigma_{a > 20}$ (R):

a	b	c
1	Marcos	21
1	Maria	23

Quadro 4 – Exemplo do operador seleção

2.2.6 Projeção

O operador Seleção tem como objetivo selecionar um determinado conjunto de atributos (colunas) de uma relação. Pode ser visualizado como uma divisão dos atributos em dois conjuntos, um com os atributos necessários para a realização da operação, e outro, com os descartados.

A especificação da operação Projeção segue o formato π `<lista de atributos> (R)` onde, o símbolo π (pi) indica a operação e, a lista de atributos, informa quais colunas devem ser apresentadas entre aquelas da relação `R`. Verifica-se que `R` no seu formato mais simples é diretamente o nome de uma relação do banco de dados, contudo pode ser uma expressão de álgebra relacional cujo o resultado é uma relação. (ELMASRI; NAVATHE, 2005, p. 109). O Quadro 5 demonstra um exemplo deste operador.

Seja a relação R (a,b,c) :

a	b	c
1	João	20
2	Marcos	21
3	Maria	23

O resultado de $\pi_{a,b}$ (R):

a	b
1	João
2	Marcos
3	Maria

Quadro 5 – Exemplo do operador projeção

2.2.7 Junção Theta

A operação Junção Theta é utilizada para combinar `tuplas` relacionadas em duas relações dentro de uma `tupla` única. Pode ser definida por um Produto Cartesiano seguido de uma Seleção.

A especificação da operação Junção Theta segue o formato `R <condição de junção> S` onde, o símbolo ⋈ indica a operação e, a condição de junção, informa quais colunas devem ser levadas em consideração na combinação das `tuplas` de `R` e `S`. Apenas as `tuplas` que satisfazem a condição de junção são apresentadas no resultado. Verifica-se que `R` no seu formato mais simples é diretamente o nome de uma relação do banco de dados, contudo pode ser uma expressão de álgebra relacional cujo resultado é uma relação (ELMASRI; NAVATHE, 2005, p. 114). O Quadro 6 demonstra um exemplo deste operador.

Seja a relação R (a,b,c) :

a	b	c
1	João	2
2	Marcos	2
3	Maria	3

Seja a relação S (d,e) :

d	e
1	Vendas
2	Desenvolvimento
3	Recursos Humanos

O resultado de R ⋈ r.c == s.d S:

a	b	c	d	e
1	João	2	2	Desenvolvimento
2	Marcos	2	2	Desenvolvimento
3	Maria	3	3	Recursos Humanos

Quadro 6 – Exemplo do operador junção

2.2.8 Operador estendido Agrupamento e Funções Agregadas

Para agrupar `tuplas` de uma determinada relação tendo como base seus atributos foi definida uma operação que é indicada pelo símbolo γ (gama), a qual também fornece suporte a funções de agregação.

A especificação desta operação segue o formato `<atributos de agrupamento>` γ

<lista de função> R, em que <atributos de agrupamento> é uma lista de atributos da relação R, e <lista de função> é uma lista de pares (<função> <atributo>). Para cada par, <função> é permitida uma das funções matemáticas de agrupamento como SOMA(SUM), MEDIA(AVERAGE), MÁXIMO(MAX), MÍNIMO(MIN), CONTAR(COUNT) seguido pelo <atributo> da relação definida por R (ELMASRI; NAVATHE, 2005, p. 119). O Quadro 7 demonstra um exemplo deste operador.

Seja a relação R (a,b,c) :

a	b	c
1	João	20
2	Marcos	21
3	Maria	23
4	Carlos	20

a) O resultado de c γ (R):

c
20
21
23

b) O resultado de c γ count(b) (R):

c	b
20	2
21	1
23	1

Quadro 7 – Exemplo do operador agrupamento sem função de grupo (a) e com função de grupo (b)

2.2.9 Operador estendido Ordenação

Para realizar a ordenação ou classificação de tuplas de uma determinada relação tendo como base seus atributos foi definida uma operação que é indicada pelo símbolo τ (tau).

A especificação desta operação segue o formato γ <lista de atributos> R, em que <lista de atributos> é uma lista de atributos da relação R indicando a ordem de classificação que será aplicada (GARCIA-MOLINA; ULLMAN; WIDON, 2001, p. 269). O Quadro 8 demonstra um exemplo deste operador.

Seja a relação R (a,b,c) :

a	b	c
1	João	20
2	Marcos	21
3	André	24
4	Carlos	20

O resultado de τ b (R):

a	b	c
3	André	24
4	Carlos	20
1	João	20
2	Marcos	21

Quadro 8 – Exemplo do operador ordenação

2.3 EXPRESSÕES ALGÉBRICAS

Conforme apresentado por Elmasri e Navathe (2005, p. 106, grifo do autor), "Uma sequência de operações de álgebra relacional forma uma **expressão de álgebra relacional** cujos resultados também serão uma relação que representa o resultado de uma consulta de banco de dados (ou solicitação de recuperação)".

Um exemplo simples de uma expressão de álgebra relacional é apresentado no Quadro 9. Neste exemplo foram utilizados dois operadores, a seleção(σ) e projeção (π). A seleção tem como objetivo remover linhas de uma relação e a projeção remover colunas.

Seja a relação StarsIn (title,year,starName) :

title	year	starName
A caçada	1996	Laurence Fishburn
A Vida é Bela	1997	Roberto Benigni
A negociação	1998	Samuel L. Jackson

O resultado da expressão algébrica $\pi_{starName}(\sigma_{year = 1996}(\pi_{starName,year}(StarsIn)))$ é :

starName
Laurence Fishburn

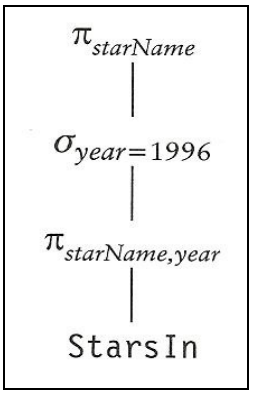
Quadro 9 – Exemplo de expressão algébrica

O fluxo de execução de uma expressão inicia no comando mais interno. Considerando o exemplo do Quadro 9, é processado $\pi_{starName,\ year}(StarsIn)$ ou seja é realizada a projeção das colunas `starName` e `year` sobre relação `StarsIn`. A partir deste resultado, em seguida é efetuado o comando $\sigma_{year} = 1996$ que consiste na seleção das linhas onde `year` é igual a 1996. Por fim é realizada a projeção da coluna `starName`.

2.4 ÁRVORES DE EXPRESSÕES ALGÉBRICAS

> Podemos combinar vários operadores da álgebra relacional em uma expressão aplicando um operador ao(s) resultado(s) de um ou mais operadores diferentes. Desse modo, como para qualquer álgebra, poderemos representar a aplicação de diversos operadores como uma **árvore de expressões**. As folhas dessa árvore são nomes de relações, e cada um dos interiores é identificado por um operador que faz sentido quando aplicado à (s) relação(ões) representadas por seu filho ou filhos. (GARCIA-MOLINA; ULLMAN; WIDON, 2001, p. 269, grifo nosso).

Um exemplo de árvore de expressão algébrica é apresentado na figura 2 o mesmo corresponde à expressão algébrica abordada na seção anterior.



Fonte: Garcia-Molina, Ullman e Widon (2001, p. 373).

Figura 2 – Exemplo de árvore de expressão algébrica

2.5 LINGUAGEM SQL

Conforme Ramakrishnan (1998, p. 181), SQL é a linguagem de banco de dados mais utilizada comercialmente. Foi originalmente desenvolvida pela *International Business Machines* (IBM) e inicialmente chamada de *Structured English Query Language* (SEQUEL) ou linguagem de pesquisa em inglês estruturado, sendo projetada como uma interface para um sistema experimental de um banco de dados relacional.

SQL é uma linguagem de banco de dados abrangente. A mesma possui comandos para definição, consultas e atualização. Assim, o comando *select* e sua declaração básica,

conforme Elmasri e Navathe (2005, p. 157), tem a forma apresentada no Quadro 10.

SELECT `<lista de atributos>`
FROM `<lista de tabelas>`
WHERE `<condição>` **;**

onde:

`<lista de atributos>` é uma lista com os nomes dos atributos cujos valores serão recuperados;

`<lista de tabelas>` é uma lista com os nomes das tabelas que serão necessárias para a realização da consulta;

`<condição>` é uma expressão condicional que identifica quais linhas devem ser recuperadas.

Fonte: adaptado de Elmasri e Navathe (2005, p. 157).

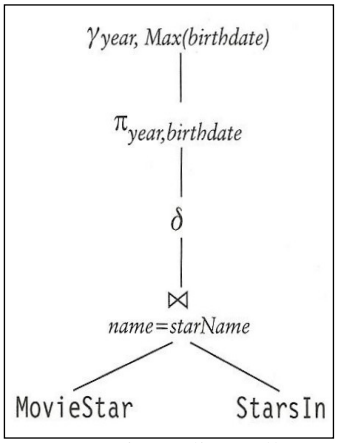
Quadro 10 – Comando `select`

2.6 RELAÇÃO ENTRE ÁRVORES DE EXPRESSÕES ALGÉBRICAS E LINGUAGEM SQL

As árvores de expressão algébricas definem o plano lógico de consulta que o SGBDR executará, sendo assim as árvores de expressão tem relação direta com o comando *select* da linguagem SQL (GARCIA-MOLINA; ULLMAN; WIDON, 2001, p. 353).

Como abordado por Ramakrishnan (1998, p. 154), a linguagem SQL é uma forma especializada de realizar consultas ao SGBDR enquanto consultas em álgebra relacional descrevem passo a passo como será o procedimento de execução para atender a uma determinada pergunta.

Na Figura 2 é apresentada uma árvore de expressão algébrica e no Quadro 11 uma consulta correspondente construída na linguagem SQL.



Fonte: Garcia-Molina, Ullman e Widon (2001, p. 377).

Figura 2 – Árvore de expressão algébrica

```
SELECT YEAR, MAX(BIRTHDATE) FROM STARSIN,MOVIESTAR WHERE STARNAME
= NAME GROUP BY YEAR;
```

Quadro 11 – Comando `select` correspondente

2.7 TRABALHOS CORRELATOS

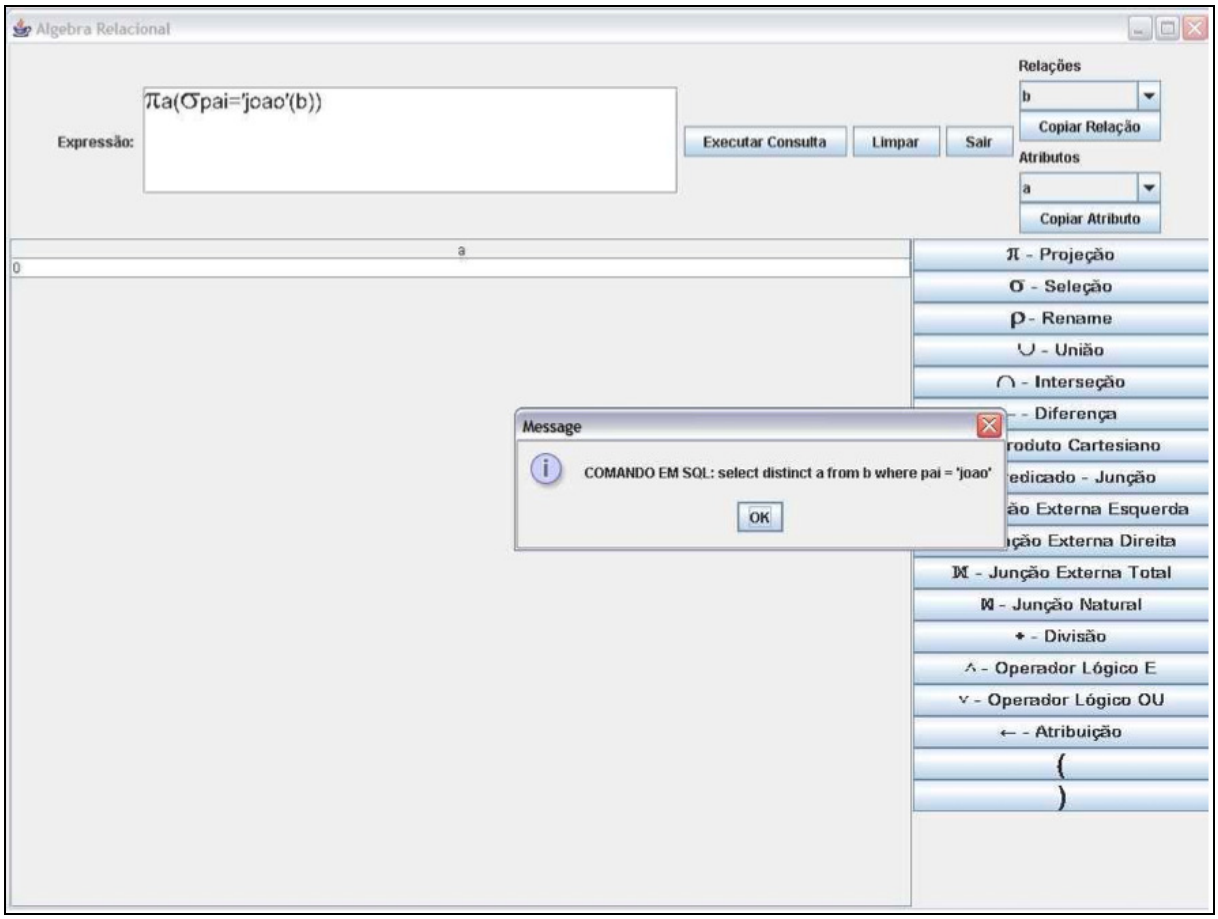
Em relação a softwares voltados ao ensino de álgebra relacional é possível encontrar trabalhos acadêmicos e artigos com esta finalidade. Dentre eles, foram estudados dois. O primeiro, EnsinAR (PAES, 2007), é uma ferramenta didática para o ensino de álgebra relacional e o segundo, iDFQL (APPEL; TRAINA JUNIOR, 2004), é uma ferramenta de apoio ao processo de ensino-aprendizagem da álgebra relacional baseado no construcionismo.

2.7.1 EnsinAR

Paes (2007, p. 1) afirma que EnsinAR baseia-se na conversão de expressões escritas em álgebra relacional para a linguagem SQL. A expressão resultante em SQL é então exibida e executada. Este trabalho não permite a construção de árvores de expressão algébricas.

Para realizar a conversão da expressão algébrica para linguagem SQL, o autor desenvolveu um compilador utilizando o Gerador de Analisadores Léxicos e Sintáticos (GALS). O compilador tem como entrada a expressão algébrica, analisa lexicamente e sintaticamente o comando e, passando com sucesso, gera o código SQL.

Esta ferramenta não possibilita a leitura de um esquema de banco de dados para escolher uma determinada relação. O usuário precisa primeiramente criar as tabelas através da própria aplicação. Após esta atividade, pode realizar a construção das expressões algébricas.



Fonte: Paes (2007, p. 16).

Figura 3 - Interface da ferramenta EnsinAR

A Figura 3 demonstra o uso da ferramenta EnsinAR. Pode-se identificar a expressão algébrica criada no topo da tela e, logo abaixo, o resultado da execução no banco de dados. O comando SQL gerado a partir da expressão esta apresentado na mensagem no centro da tela.

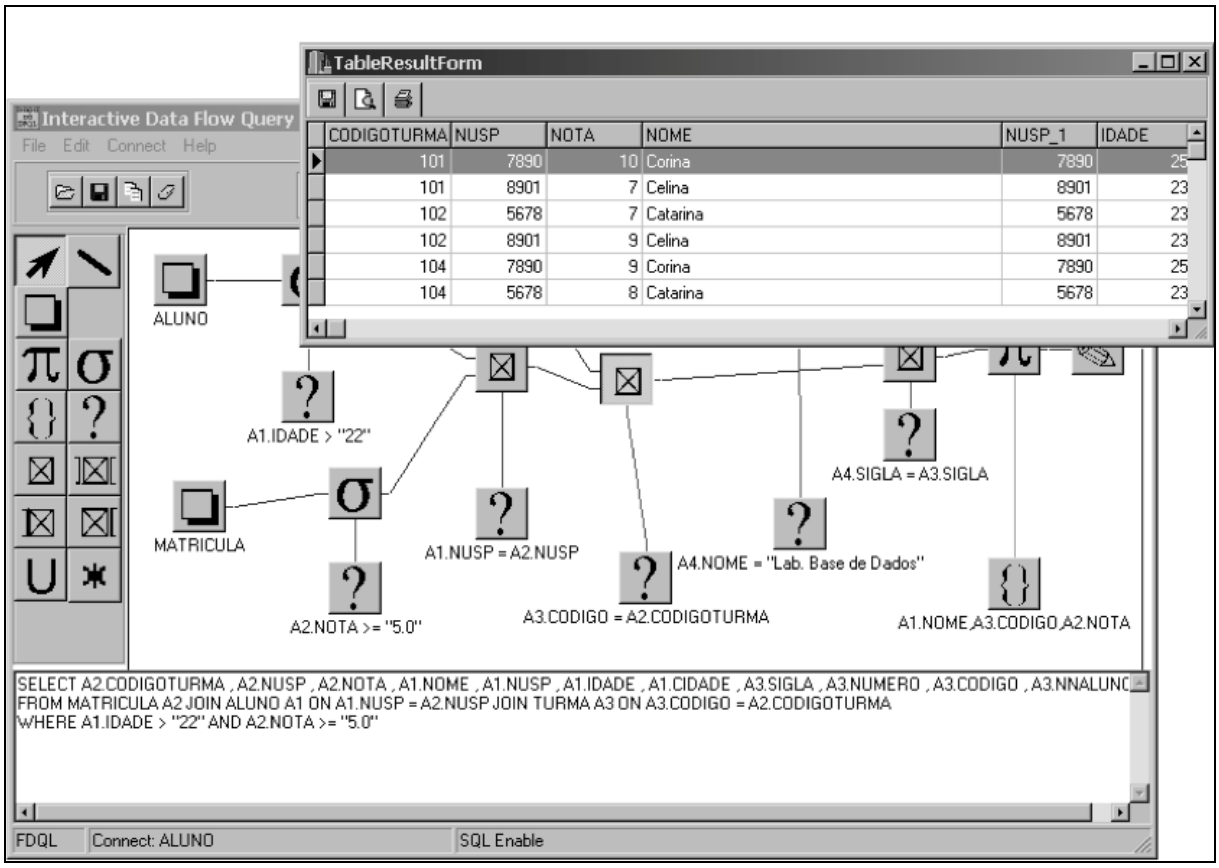
2.7.2 iDFQL

O principal objetivo de iDFQL, segundo Appel e Traina Junior (2004), é combinar o uso de ícones para representar os operadores da álgebra relacional e diagramas de fluxo para representar a consulta.

Este trabalho apresenta suporte à construção de diagramas equivalentes a árvores relacionais, porém, operadores como agregação e ordenação que são amplamente utilizados na linguagem SQL não são suportados.

A ferramenta iDFQL modificou operadores da álgebra relacional que necessitam de parâmetros para sua execução, desta forma foram desenvolvidos dois novos operadores

chamados respectivamente de Condição e Lista de Atributos. A função da operação Condição é permitir a entrada de regras para que os operadores de junção e seleção possam ser executados. Já o operador Lista de Atributos, define um conjunto de colunas para a operação de projeção. A separação da condição de execução e atributos dos operadores torna a árvore de expressões mais legível como um todo.



Fonte: Appel e Traina Junior (2004, p. 10).

Figura 4 - Interface da ferramenta iDFQL

A Figura 4 demonstra o uso da ferramenta iDFQL. Pode-se identificar a árvore de expressões criada na tela principal, logo abaixo, o comando SQL gerado e o resultado da execução no banco de dados na tela a frente.

3 DESENVOLVIMENTO

Este capítulo descreve as etapas do desenvolvimento da aplicação. São abordados os requisitos, a especificação e o desenvolvimento do mesmo, apontando as técnicas utilizadas. Por fim, são apresentados os resultados obtidos.

Os requisitos apresentados abaixo se encontram classificados em Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF).

O sistema deverá:

a) permitir a construção de árvores de expressão algébricas utilizando *drag-and-drop* (RF);
b) suportar os operadores: União, Diferença, Projeção, Seleção, Junção Theta, Produto Cartesiano, Agrupamento/Agregação e Reordenação (RF);
c) permitir salvar e editar as árvores construídas (RF);
d) transformar as árvores em comandos SQL padrão (RF);
e) permitir conexão com os bancos de dados Oracle XE e MySQL 6.0 (RF);
f) listar as relações de *schemas* do banco em que esta conectado (RF);
g) executar e apresentar o resultado dos comandos SQL gerados nos bancos de dados Oracle e MySQL (RF);
h) utilizar a linguagem de programação Java (RNF);
i) utilizar o ambiente de programação Eclipse (RNF).

3.1 ESPECIFICAÇÃO

A especificação deste trabalho foi desenvolvida utilizando o diagrama de casos de uso e o diagrama de classes da *Unified Modelling Language* (UML) , a ferramenta utilizada para a elaboração foi o *Enterprise Architec*.

3.1.1 Diagrama de casos de uso

A Figura 5 demonstra os casos de uso. Nas seções seguintes os mesmos são detalhados através de seus respectivos cenários.

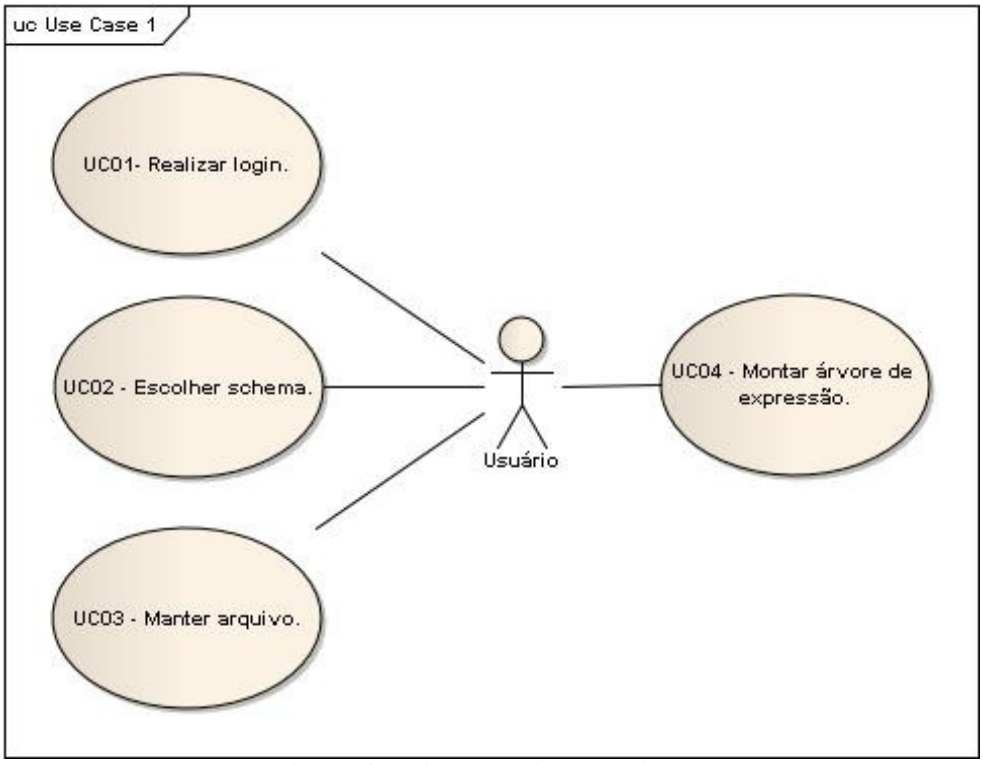


Figura 5 – Diagrama de casos de uso

3.1.1.1 UC01 - Realizar login

Este caso de uso descreve como o usuário realizará o login na aplicação. Os dados para o *login* são referentes à conexão com o SGBDR escolhido. O Quadro 12 descreve seus cenários.

UC01 - Realizar login	
Pré-condição	A ferramenta deve estar instalada
Cenário Principal	1. O usuário aciona a aplicação. 2. O sistema apresenta a tela login. 3. O usuário seleciona o banco informa o nome do banco, usuário e senha e aciona o botão conectar. 4. O sistema realiza a conexão com o banco de dados. 5. O sistema apresenta a mensagem "Conexão realizada com sucesso".
Exceção	No passo 4, caso não for possível realizar a conexão o sistema exibe a mensagem : Os dados informados para login estão incorretos.".
Pós-condição	Aplicação conectada ao banco de dados selecionado

Quadro 12 – Cenários do UC01

3.1.1.2 UC02 – Selecionar *schema*

Demonstra como o usuário seleciona um *schema* do SGDBR conectado. Permite que as relações possam ser acessadas e utilizadas na confecção das árvores de expressões algébricas. O Quadro 13 descreve seus cenários.

UC02 – Selecionar schema	
Pré-condição	O usuário deve ter executado o UC01.
Cenário Principal	1. O usuário seleciona um *schema* na aplicação. 2. O sistema carrega as tabelas na Lista de Relações.
Exceção	No passo 2, caso não exista tabelas no *schema* selecionado o sistema apresenta a mensagem "Não existem tabelas no *schema* escolhido".
Pós-condição	Tabelas carregadas na Lista de Relações.

Quadro 13 – Cenários do UC02

3.1.1.3 UC03 – Manter arquivo

O objetivo desse caso de uso é manter os arquivos gerados pela ferramenta. O Quadro 14 descreve seus cenários.

UC03 – Manter arquivo	
Pré-condição	O usuário deve ter executado o UC01.
Cenário Principal	1. O usuário acessa o menu “Arquivo” e seleciona a opção "Salvar". 2. O usuário informa o diretório e o nome para o arquivo e aciona o botão “Ok”. 3. O sistema salva o arquivo no diretório escolhido.
Cenário Alternativo	1. No passo 1 do cenário principal o usuário seleciona a opção "Abrir" do menu "Arquivo". 2. O usuário seleciona o arquivo desejado e aciona o botão "Ok". 3. O sistema carrega o arquivo. 4. O usuário acessa o menu “Arquivo” e seleciona a opção "Salvar". 5. O fluxo retorna para o passo 2 do cenário principal.
Exceção	No passo 2 do cenário principal, caso o arquivo já foi salvo em diretório o sistema não apresenta a tela e atualiza o arquivo.
Exceção 1	No passo 3 do cenário alternativo, caso o arquivo selecionado seja inválido o sistema apresenta a mensagem : "O arquivo escolhido não pode ser carregado pois possui formato inválido". O fluxo retorna para o passo 2.
Pós-condição	Arquivo salvo no diretório escolhido.

Quadro 14 – Cenários do UC03

3.1.1.4 UC04 – Montar árvores de expressão

O UC04 é detalhado através dos casos de uso UC05, UC06, UC07 e UC08 apresentados na seção 3.2.4 Operacionalidade da implementação.

3.1.2 Diagrama de Pacotes

Esta seção apresenta a arquitetura da aplicação representada através do diagrama de pacotes (Figura 7). O desenvolvimento da ferramenta foi baseado no padrão MVC (*Model View Controller*). Conforme a metodologia do MVC, a aplicação é composta pelos pacotes *Model*, *View* e *Controller* o pacote *Resource* é comum aos demais pacotes este por sua vez possui classes de constantes, arquivos de configuração e mensagens.

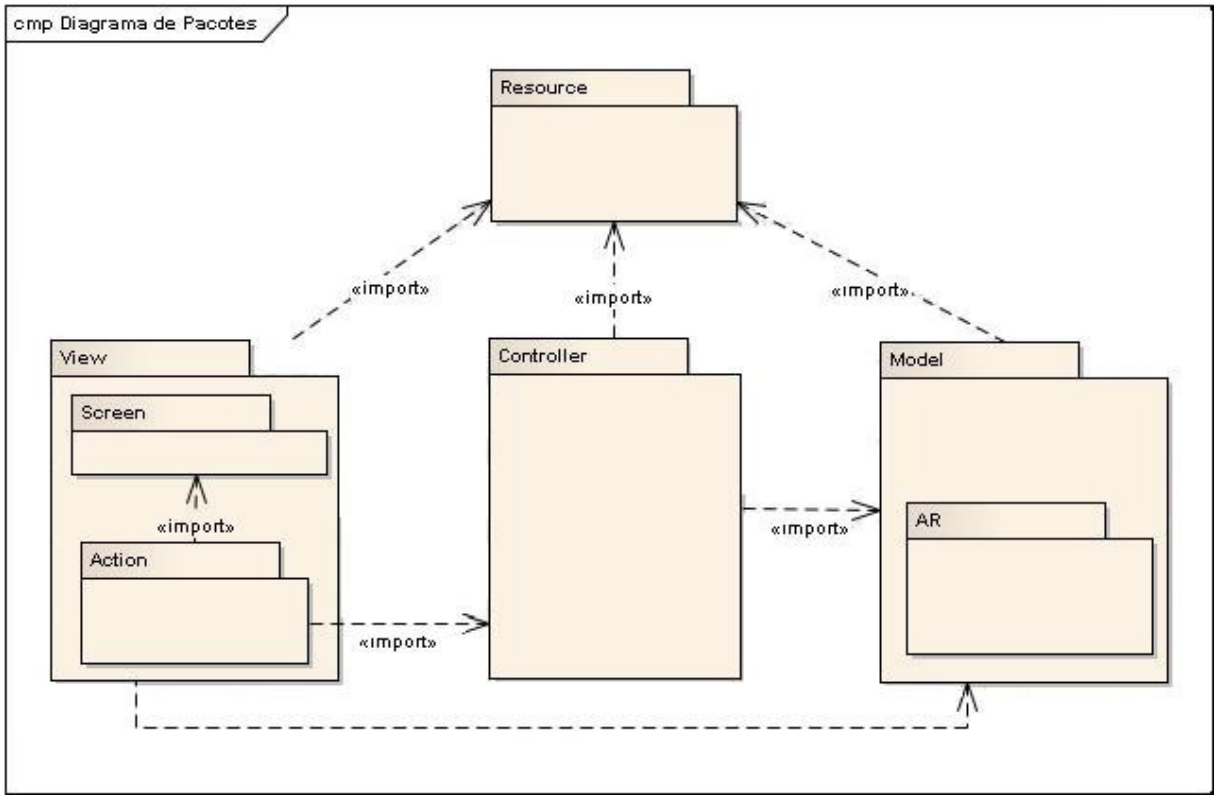


Figura 7 – Diagrama de pacotes

3.1.2.1 Pacote *View*

O pacote View agrupa as classes referentes a interface gráfica da aplicação, a figura 8 demonstra o diagrama de classes.

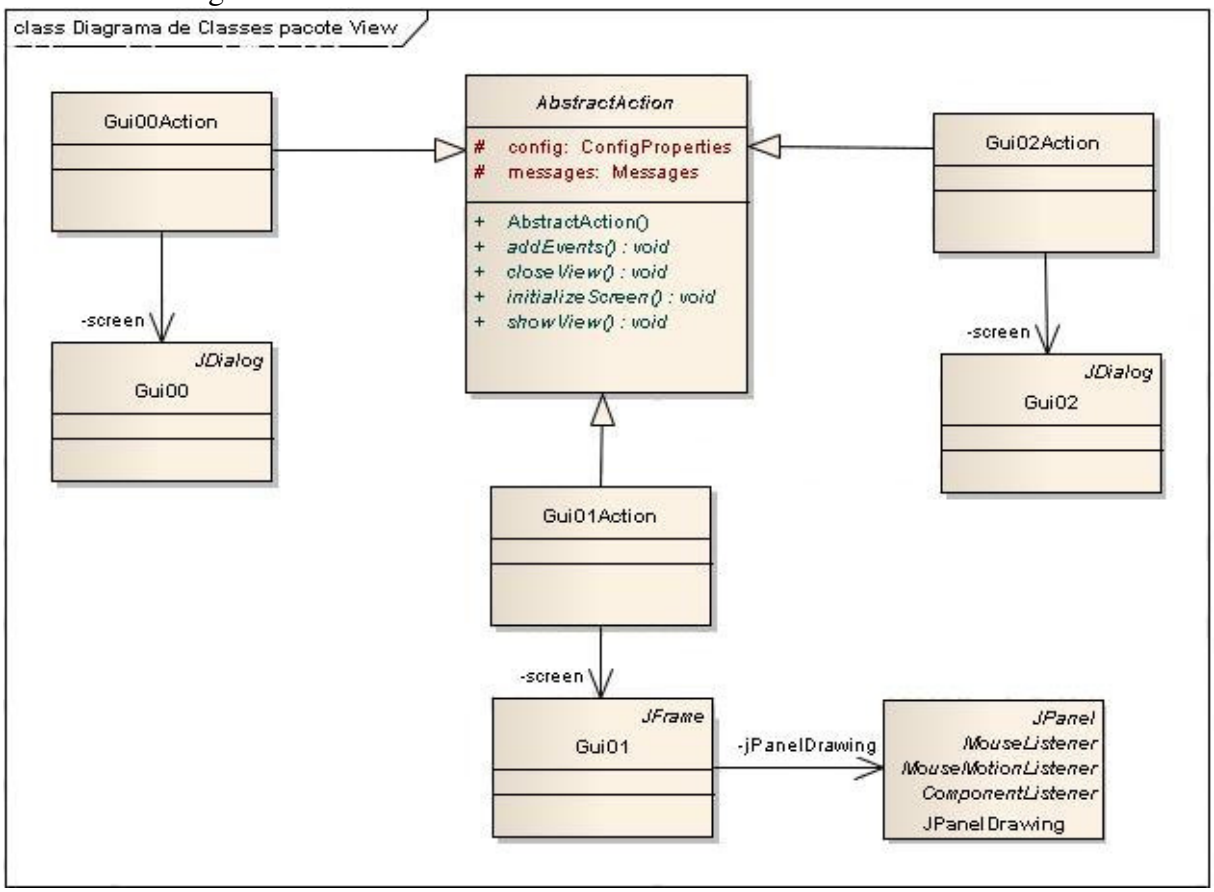


Figura 8 – Diagrama de classes pacote *View*

As classes com prefixo `Gui` seguido de um valor numérico são responsáveis pelo desenho das telas, as classes com pós-fixo `Action` são responsáveis pelas ações executadas. A classe `AbstractAction` define métodos abstratos para padronizar as classes `Action`.

A classe `Gui01` é a principal interface da aplicação e é composta pela classe `JPanelDrawing`. `JPanelDrawing` disponibiliza uma área de desenho onde serão construídas as árvores de expressão algébricas.

3.1.2.2 Pacote *Controller*

O pacote *Controller* contem as classes utilitárias que responsáveis por executar tarefas especificas invocadas pelo pacote *View*. A Figura 9 demonstra o diagrama de classes.

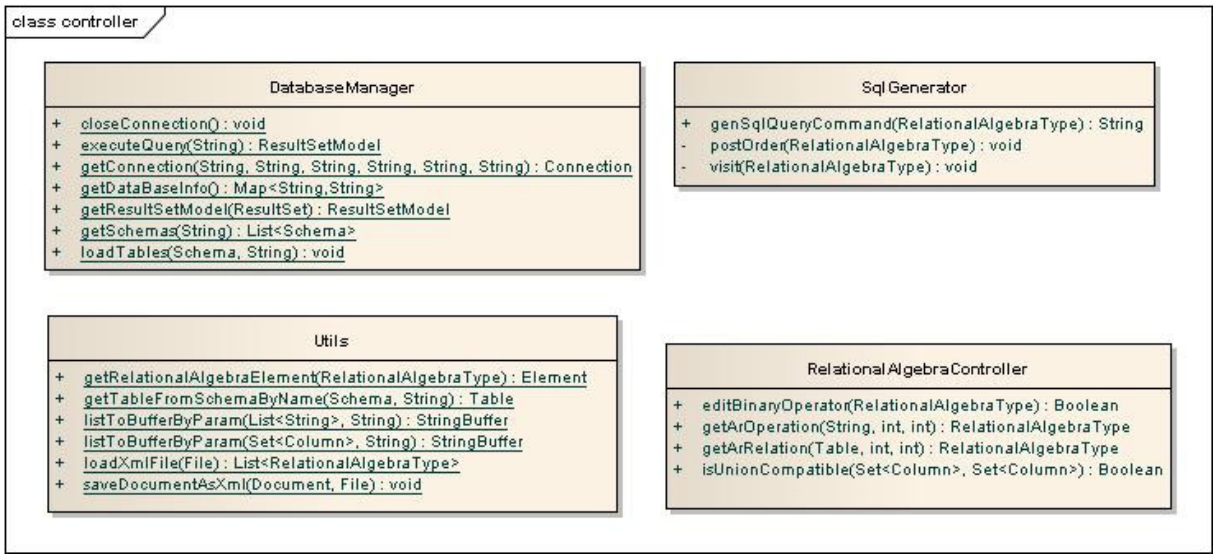


Figura 9 – Diagrama de classes pacote *Controller*

A classe `DataBaseManager` é responsável pelo acesso ao banco de dados, possui métodos para execução de consultas e para retornar informações do banco a qual esta conectado.

A classe `RelationaAlgebraController` tem como objetivo agrupar tarefas relativas a álgebra relacional como exemplo a verificação de compatilidade de união e criação de operadores como intersecção e projeção.

Por fim a classe `SqlGenerator` tem a finalidade realizar a geração de comandos de consulta na linguagem SQL.

3.1.2.3 Pacote *Model*

O pacote `Model` possui as classes de modelo conhecidas como `javaBeans` e são utilizadas pelos pacotes `View` e `Controller` a figura 10 apresenta seu diagrama de classes.

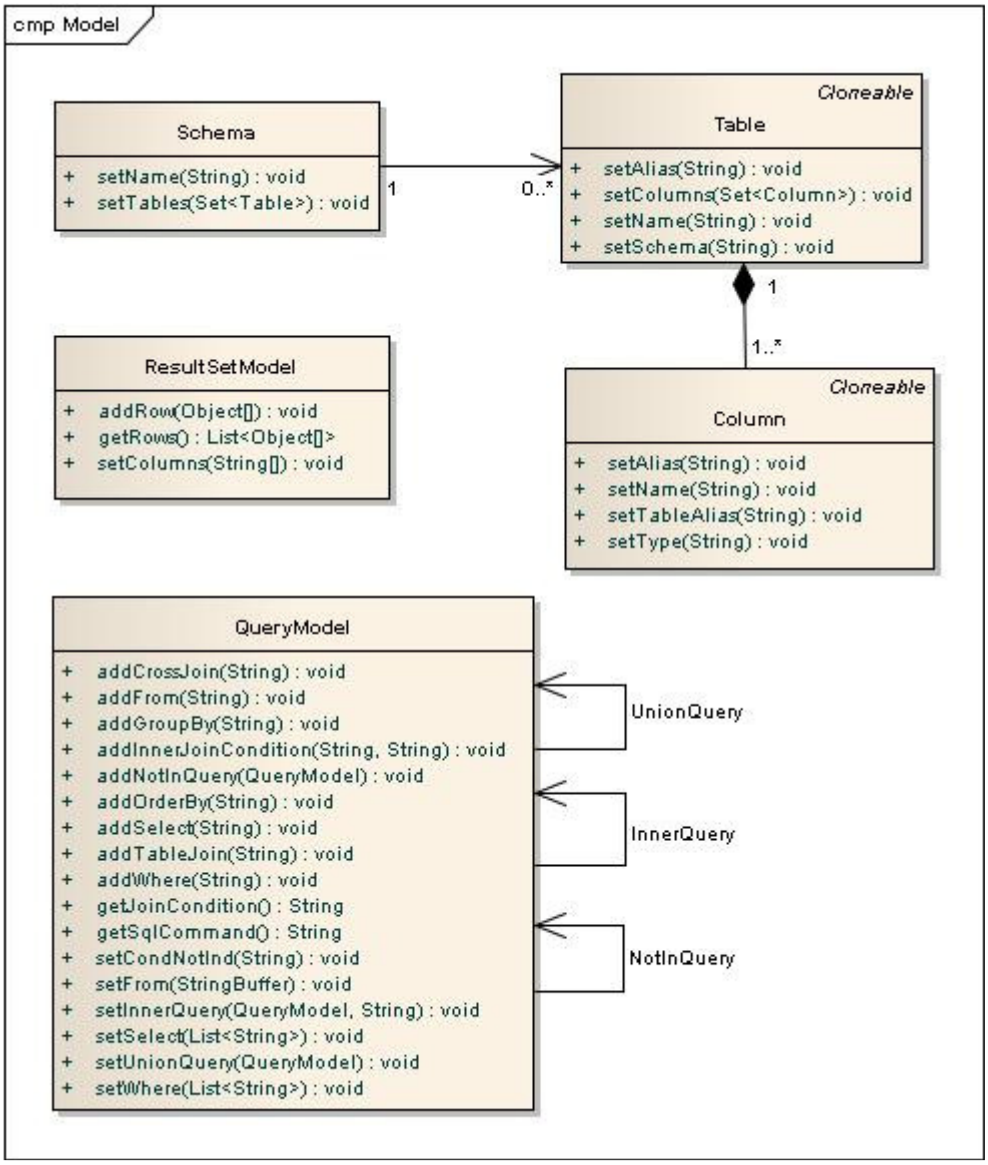


Figura 10 – Diagrama de classes pacote `Model`

A classe `Schema` representa um esquema de banco de dados que por sua vez é composto por tabelas, as tabelas são representas pela classe `Table` e as colunas de cada tabelas são representadas pela classe `Column`.

A classe `QueryModel` representa a estrutura de um comando de consulta SQL.

Para expressar o resultado de uma consulta SQL realizada é utilizada a classe `ResultSetModel`.

O pacote *Model* possui um pacote interno chamado `AR`, esse contem os tipos de objetos relacionados a álgebra relacional a Figura 11 apresenta o diagrama de classes correspondente.

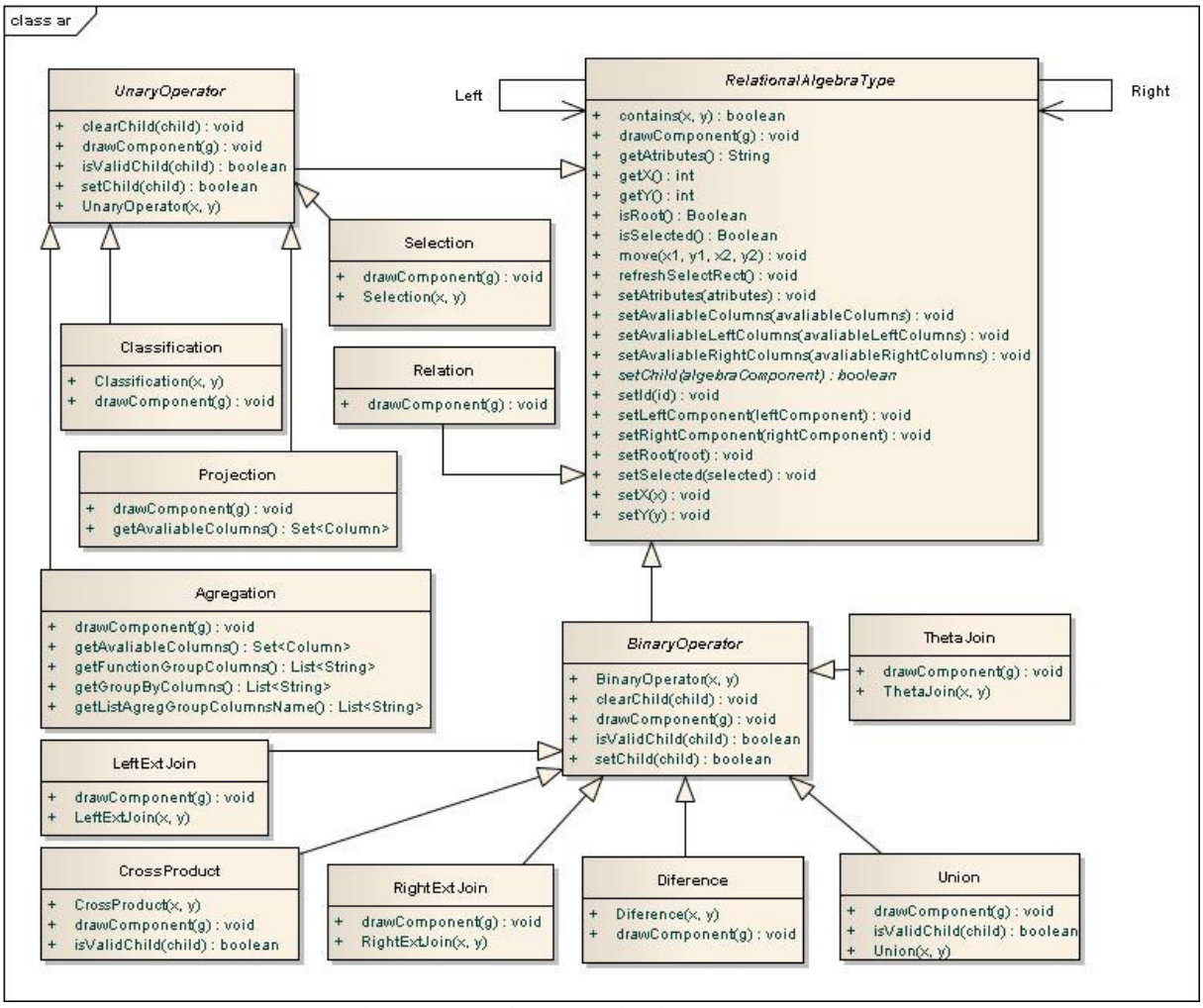


Figura 11 – Diagrama de classes pacote AR

A classe abstrata `RelationalAlgebraComponent` determina os métodos base para os operadores da álgebra relacional, a estruturação em árvore também é de sua responsabilidade portanto essa classe possui duas associações consigo que representam respectivamente os filhos da esquerda e direta.

As classes `UnaryOperator` e `BinaryOperator` representam as operações unárias e binárias, cada uma possui métodos de controle para política de alocação de filhos na estrutura, ambas herdam de `RelationalAlgebraComponent`.

Por fim cada operação é representada pelo seu tipo correspondente como por exemplo a classe `Projection` que identifica a operação π (projeção) classificada como operação unaria portanto herda de `UnaryOperator`.

3.2 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.2.1 Técnicas e ferramentas utilizadas

Para a implementação deste trabalho foi utilizada a linguagem de programação Java versão 6. O compilador para a edição do código fonte foi o Eclipse versão 3.2.

O acesso aos SGBDRs suportados pela ferramenta foi implementado utilizando o *driver* JDBC (*Java Database Connectivity*).

3.2.2 Leitura dos *schemas* e tabelas

A classe `DatabaseManager` é responsável pelas operações realizadas no banco de dados, antes de realizar a leitura dos *schemas* é necessário realizar a conexão utilizando o método `getConnection`. Após a conexão conseguida o método `getSchemas` pode ser invocado e realizara a leitura dos `schemas` o Quadro 20 demonstra seu código fonte.

A interface `DatabaseMetaData` utilizada na linha 91 é implementa pelo fornecedor do do driver JDBC, a mesma define métodos para leitura da estrutura do banco do qual esta conectado e no caso da conexão com o banco Oracle, a instancia da classe retornada pelo método `getMetaData` é especifica para o mesmo.

Para realizar a leitura dos *schemas* do banco Oracle o método `getSchemas` da interface `DatabaseMetaData` é invocado, já para o banco MySQL o método `getCatalogs` foi chamado, isso ocorre pelo fato de que o MySQL não trabalha com o conceito de *schemas* para agrupar suas tabelas e sim catálogos.

```
89   public static List<Schema> getSchemas(String dataBase) throws Exception{
90      List<Schema> listSchema = new ArrayList<Schema>();
91      DatabaseMetaData dbmdt = (DatabaseMetaData) connection.getMetaData();
92      ResultSet rsSchemas = null;
93      String schemaColumn = "";
94
95      if(dataBase.equals(Constants.ORACLE_DB)){
96         rsSchemas = dbmdt.getSchemas();
97         schemaColumn = "TABLE_SCHEM";
98      }else if (dataBase.equals(Constants.MYSQL_DB)){
99         rsSchemas = dbmdt.getCatalogs();
100        schemaColumn = "TABLE_CAT";
101     }
102
103     String tableCat;
104     Schema schema = null;
105
106     while(rsSchemas.next()){
107        tableCat = rsSchemas.getString(schemaColumn);
108        schema = new Schema(tableCat);
109        listSchema.add(schema);
110     }
111
112     return listSchema;
113  }
```

Quadro 20 – Método `getSchemas`

Para carregar as tabelas de um determinado *schema* ou catalogo o método `loadTables` da classe `DatabaseManager` é invocado, o Quadro 21 apresenta seu código.

```
120  public static void loadTables(Schema schema, String dataBase)
121        throws Exception {
122
123     DatabaseMetaData dbmdt = (DatabaseMetaData) connection.getMetaData();
124     ResultSet rsTables;
125     ResultSet rsColumns;
126
127     String tableSchema = schema.getName();
128     Set<Table> tableSet = new HashSet<Table>();
129
130     rsTables = null;
131     String[] types = { "TABLE" };
132
133     if (dataBase.equals(Constants.ORACLE_DB)) {
134        rsTables = dbmdt.getTables(null, tableSchema, null, types);
135     } else if (dataBase.equals(Constants.MYSQL_DB)) {
136        rsTables = dbmdt.getTables(tableSchema, null, null, types);
137     }
138
139     while (rsTables.next()) {
140        String tableName = rsTables.getString("TABLE_NAME");
141        Table table = new Table(tableSchema, tableName);
142        LinkedHashSet<Column> columnSet = new LinkedHashSet<Column>();
143        rsColumns = dbmdt.getColumns(tableSchema, null, tableName, null);
144        while (rsColumns.next()) {
145           String columnName = rsColumns.getString("COLUMN_NAME");
146           String columnType = rsColumns.getString("TYPE_NAME");
147           columnSet.add(new Column(columnName, columnType));
148        }
149        rsColumns.close();
150        table.setColumns(columnSet);
151        tableSet.add(table);
152     }
153     rsTables.close();
154     schema.setTables(tableSet);
155  }
```

Quadro 21 – Método `loadTables`

O método `getTables` (linha 134) é responsável por buscar o conjunto de tabelas de cada *schema*, para isso o mesmo recebe como parâmetro o nome do *schema* ou catálogo e também um filtro `types` para retornar somente objetos do tipo tabela.

Os métodos `getTables`, `getSchemas` ou `getCatalogos` fornecidos pela interface `DatabaseMetaData` possuem como retorno uma classe chamada `ResultSet` que representa uma estrutura de linhas e colunas. Para recuperar os dados de um `ResultSet` deve-se realizar uma iteração sobre o mesmo, sendo que, cada iteração representa uma linha que contem um conjunto de colunas, e para adquirir o valor de cada coluna pode-se utilizar o método `getString` passando como parâmetro o nome da coluna desejada, a linha 139 apresenta a iteração sobre o `ResultSet rsTables`.

3.2.3 Geração dos comandos SQL

A geração dos comandos SQL acontece em duas fases:

1. Visitação da árvore em pós-ordem alimentando um modelo de *query*;
2. Recuperação do comando através do modelo.

A classe `SqlGenerator` é responsável pela geração dos comandos SQL, a mesma executa a primeira fase da geração através do método `genSqlQueryCommand`. O método `genSqlQueryCommand` recebe como parâmetro o nodo do qual se deseja gerar o código, esse nodo é considerado o raiz da árvore e a partir dele é iniciado o processo de geração, o quadro 22 apresenta o trecho do código.

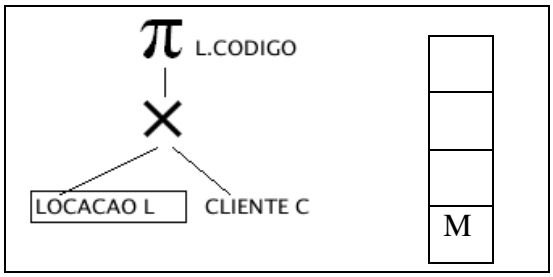
```
27  public String genSqlQueryCommand(RelationalAlgebraComponent root) {
28      postOrder(root);
29      return sqlCommand;
30  }
31
32  private void postOrder(RelationalAlgebraComponent n) {
33      if (n != null) {
34          postOrder(n.getLeftComponent());
35          postOrder(n.getRightComponent());
36          visit(n);
37      }
38  }
```

Quadro 22 – Métodos `genSqlQueryCommand` e `postOrder`

O modelo (*template*) de *query* é representado pela classe `QueryModel`, sua função consiste em receber as informações adquiridas na visitação da árvore e organizá-los conforme a sintaxe do comando de consulta SQL.

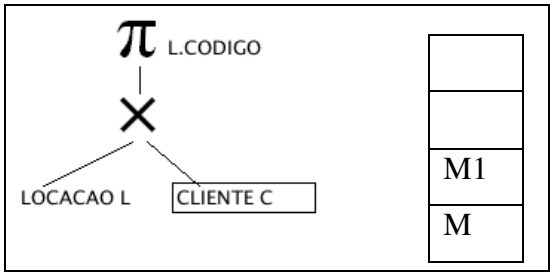
Para melhor demonstrar o algoritmo de geração será realizado um passo a passo com uma árvore de expressão de exemplo. Cada passo será detalhado descritivamente e conterá

um quadro de apoio com a árvore de exemplo utilizada e o estado da pilha. O quadro 23 representa o inicio do algoritmo, conforme o varredura em pós-ordem o primeiro nó visitado é a relação `LOCACAO L`, para este tipo de nodo é criado um modelo `M` de consulta SQL, e ao mesmo é aplicado a relação `LOCACAO L`, o modelo `M` é empilhado.



Quadro 23 – Passo 1 da geração do comando SQL

O próximo nó visita é a relação `CLIENTE C`, novamente é criado um modelo `M1` e aplicado ao mesmo a relação `CLIENTE C`, `M1` é empilhado. O Quadro 24 demonstra o passo.

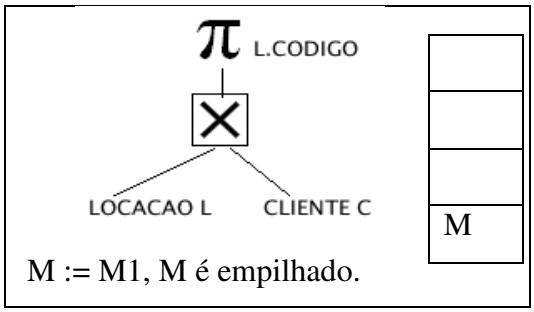


Quadro 24 – Passo 2 da geração do comando SQL

O próximo nó visitado é o operador produto cartesiano, como trata-se de um operador binário os modelos `M` e `M1` são desempilhados (linhas 317 e 318 do Quadro 25), neste momento é realizada uma fusão entre os modelos, a fusão consiste em atribuir ao modelo `M` todo o modelo `M1` isso é possível pois pode-se representar em uma única consulta SQL o produto cartesiano entre duas tabelas. O modelo `M` é empilhado (linhas 317 e 318 do Quadro 30), o Quadro 26 contextualiza o passo 3.

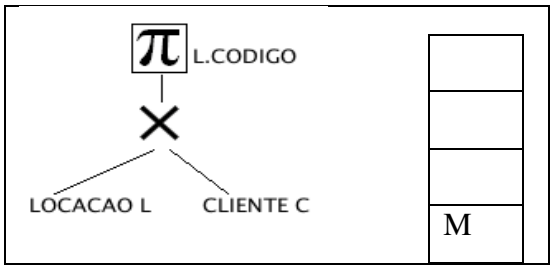
```
316     case CROSS_PRODUCT:
317        rModel = stack.pop();
318        lModel = stack.pop();
319        CrossProduct cross = (CrossProduct)n;
320
321        //Retorna as colunas disponiveis do operador direita do produto cartesiano.
322        itRight = cross.getAvaliableRightColumns().iterator();
323        listRightColumns = new ArrayList<String>();
324        while (itRight.hasNext()) {
325           listRightColumns.add(itRight.next().getNameWithTableAlias());
326        }
327
328        //Retorna as colunas disponiveis do operador esquerda do produto cartesiano.
329        itLeft = cross.getAvaliableLeftColumns().iterator();
330        listLeftColumns = new ArrayList<String>();
331        while (itLeft.hasNext()) {
332           listLeftColumns.add(itLeft.next().getNameWithTableAlias());
333        }
334
335        if (rModel.nivel == 0) {
336
337           lModel.getSelect().clear();
338           lModel.getSelect().addAll(listLeftColumns);
339           lModel.getSelect().addAll(listRightColumns);
340
341           lModel.getWhere().addAll(rModel.getWhere());
342           lModel.getListCrossJoin().addAll(rModel.getListCrossJoin());
343           lModel.getListJoin().addAll(rModel.getListJoin());
344           lModel.getListNotInQuery().addAll(rModel.getListNotInQuery());
345
346           if (rModel.getCondNotInd() != null) {
347              lModel.setCondNotInd(rModel.getCondNotInd());
348           }
349           lModel.addCrossJoin(rModel.getFrom().toString());
350           lModel.getListTableJoin().addAll(rModel.getListTableJoin());
351        }
352        stack.push(lModel);
353        break;
```

Quadro 25 – Código fonte referente ao passo 3



Quadro 26 – Passo 3 da geração do comando SQL

O próximo nó visitado é o operador projeção, é desempilhado o modelo `M` e atribuído a projeção ao mesmo, o modelo `M` é novamente empilhado. O Quadro 27 contextualiza o passo 4.



Quadro 27 – Passo 4 da geração do comando SQL

Neste momento é identificado que nodo atual é a raiz da árvore, é então desempilhado o modelo `M` e invocado o método `getSqlCommand` do mesmo. O método `getSqlCommand` é responsável por recuperar todos os atributos que o modelo recebeu durante a varredura da árvore e montar o comando SQL.

Para garantir que os comandos SQL gerados rodassem nos bancos de dados Oracle e MySQL o método `getSqlCommand` respeita a sintaxe defina na norma SQL ANSI 92.

3.2.4 Operacionalidade da implementação

Esta seção apresenta a operacionalidade da ferramenta desenvolvida através de um estudo de caso. A figura 6 especifica os casos de uso utilizados para teste.

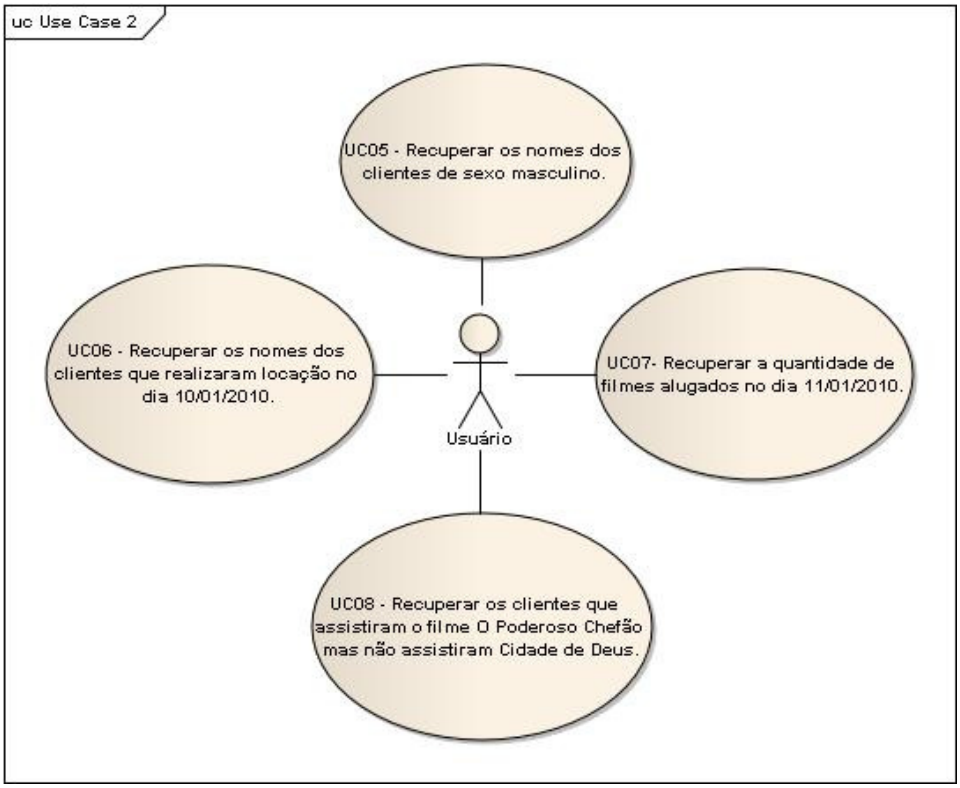


Figura 6 – Diagrama de casos de uso 02

Nas seções seguintes cada caso uso é detalhado através de seus respectivos cenários, cada um possuirá uma pós-condição que determina o resultado da execução da árvore de expressão gerada. Para melhor entendimento foi utilizado um modelo de dados apresentado no Quadro 15.

FILME	
CODIGO	NOME
1	Em nome da Rosa
2	O Exterminador do Futuro
3	Titanic
4	Cidade de Deus
5	O Poderoso Chefão

CLIENTE		
CODIGO	NOME	SEXO
1	João	M
2	Maria	F
3	José	M
4	Miguel	M
5	Marco	M

LOCAÇÃO_FILME	
CODIGO_LOCACAO	CODIGO_FILME
1	1
1	2
2	3
3	5
4	4
4	5
4	2
5	3

LOCAÇÃO			
CODIGO	DATA_INICIO	DATA_FIM	CODIGO_CLIENTE
1	10/01/10	12/01/10	1
2	10/01/10	11/01/10	2
3	10/01/10	11/01/10	3
4	11/01/10	14/01/10	4
5	11/01/10	12/01/10	1

Quadro 15 – Modelo de dados para os casos de uso 5, 6, 7 e 8

3.2.4.1 UC06 - Recuperar os nomes dos clientes de sexo masculino.

O quadro 16 descreve os cenários do caso de uso 05.

<table>
<tr><th colspan="2">UC05 - Recuperar os nomes dos clientes de sexo masculino</th></tr>
<tr><td>Pré-condição</td><td>O usuário deve ter executado o UC01.</td></tr>
<tr><td>Cenário Principal</td><td>1. O usuário insere a relação CLIENTE na a área de edição.
2. O usuário insere os operadores Seleção e Projeção.
3. O usuário efetua a ligação da relação Cliente com o operador Seleção.
4. O usuário informa a condição de seleção : SEXO = 'M'.
5. O usuário efetua a ligação do operador Seleção com o operador Projeção.
6. O usuário informa o atributo NOME para efetuar projeção.
7. O usuário aciona o botão executar.</td></tr>
<tr><td>Exceção</td><td>No passo 4 o usuário não informa condição de seleção, o sistema apresenta a mensagem "É necessário informar a condição de seleção.", o fluxo retorna para o passo 4.</td></tr>
<tr><td>Pós-condição</td><td>O sistema deve apresentar a seguinte relação como resultado:
<table><tr><th>NOME</th></tr><tr><td>Joao</td></tr><tr><td>José</td></tr><tr><td>Miguel</td></tr><tr><td>Marco</td></tr></table></td></tr>
</table>

Quadro 16 – Cenários do UC05

3.2.4.2 UC06 - Recuperar os nomes dos clientes que realizaram locação no dia 10/01/2010.

Demonstra como o usuário realiza a construção de uma árvore de expressão para recuperar os nomes dos clientes que realizaram locação no dia 10/01/2010. O Quadro 17 descreve seus cenários.

<table>
<tr><th colspan="2">UC06 - Recuperar os nomes dos clientes que realizaram locação no dia '10/01/2010'.</th></tr>
<tr><td>Pré-condição</td><td>O usuário deve ter executado o UC01.</td></tr>
<tr><td>Cenário Principal</td><td>1. O usuário insere as relações LOCACAO e CLIENTE para a área de edição.
2. O usuário insere os operadores Seleção,Projeção e Junção.
3. O usuário efetua a ligação da relação LOCACAO com o operador Seleção.
4. O usuário informa a condição de seleção : DATA_INICIAL : '10/01/2010'.
5. O usuário efetua a ligação do operador Seleção com o operador Junção.
6. O usuário efetua a ligação da relação CLIENTE com o operador Junção.
7. O usuário informa a condição de junção 'c.codigo = l.cod_cliente'.
8. O usuário efetua a ligação do operador Junção com o operador Projeção.
9. O usuário informa o atributo NOME para efetuar a projeção.
10. O usuário aciona o botão executar.</td></tr>
<tr><td>Exceção</td><td>No passo 4 o usuário não informa condição de seleção, o sistema apresenta a mensagem "É necessário informar a condição de seleção.", o fluxo retorna para o passo 4.</td></tr>
<tr><td>Exceção 2</td><td>No passo 7 o usuário não informa condição de junção, o sistema apresenta a mensagem "É necessário informar a condição de junção.", o fluxo retorna para o passo 7.</td></tr>
<tr><td>Pós-condição</td><td>O sistema deve apresentar a seguinte relação como resultado:
<table><tr><td>NOME</td></tr><tr><td>Joao</td></tr><tr><td>Maria</td></tr><tr><td>José</td></tr></table></td></tr>
</table>

Quadro 17 – Cenários do UC06

3.2.4.3 UC07- Recuperar a quantidade de filmes alugados no dia 11/01/2010.

Demonstra como o usuário realiza a construção de uma árvore de expressão para recuperar a quantidade de filmes alugados no dia 11/01/2010. O Quadro 18 descreve seus cenários.

UC07- Recuperar a quantidade de filmes alugados no dia 11/01/2010.	
Pré-condição	O usuário deve ter executado o UC01.
Cenário Principal	1. O usuário insere as relações LOCACAO e LOCACAO_FILME para a área de edição. 2. O usuário insere os operadores Seleção, Agregação e Junção. 3. O usuário efetua a ligação da relação LOCACAO com o operador Seleção. 4. O usuário informa a condição de seleção : DATA_INICIAL : '11/01/2010'. 5. O usuário efetua a ligação do operador Seleção com o operador Junção. 6. O usuário efetua a ligação da relação LOCACAO_FILME com o operador Junção. 7. O usuário informa a condição de junção : 'l.codigo = lc.cod_locacao'. 8. O usuário efetua a ligação do operador Junção com o operador Agregação. 9. O usuário informa a função COUNT e como parâmetro o atributo codigo. 10. O usuário aciona o botão executar;
Exceção	No passo 4 o usuário não informa condição de seleção, o sistema apresenta a mensagem "É necessário informar a condição de seleção.", o fluxo retorna para o passo 4.
Exceção 2	No passo 7 o usuário não informa condição de junção, o sistema apresenta a mensagem "É necessário informar a condição de junção.", o fluxo retorna para o passo 7.
Exceção 3	No passo 9 o usuário não informa a função COUNT, o sistema apresenta a mensagem "É necessário informar a função de agrupamento.", o fluxo retorna para o passo 9.
Pós-condição	O sistema apresenta a relação como resultado: COUNT(codigo) 4

Quadro 18 – Cenários do UC07

3.2.4.4 UC08- Recuperar os clientes que assistiram o filme 'O Poderoso Chefão' mas não assistiram Cidade de Deus.

Demonstra como o usuário realiza a construção de uma árvore de expressão para recuperar os clientes que assistiram ao filme 'O Poderoso Chefão' mas não assistiram Cidade de Deus. O Quadro 19 descreve seus cenários.

<table>
<tr><th colspan="2">UC08- Recuperar os clientes que assistiram o filme 'O Poderoso Chefão' mas não assistiram Cidade de Deus.</th></tr>
<tr><td>Pré-condição</td><td>O usuário deve ter executado o UC01.</td></tr>
<tr><td>Cenário Principal</td><td>1. O usuário insere a relação LOCACAO_FILME e os operadores seleção e projeção para a área de edição.
2. O usuário efetua a ligação da relação LOCACAO_FILME com o operador seleção.
3. O usuário informa a condição de seleção : CODIGO_FILME = 4.
4. O usuário efetua a ligação do operador seleção com o operador projeção informando a coluna CODIGO_LOCACAO como parâmetro do mesmo.
5. O usuário realiza novamente os passo de 1 a 4 criando uma segunda árvore substituindo a condição de seleção do passo 3 para : CODIGO_FILME = 5.
6. O usuário insere a operação diferença.
7. O usuário efetua a ligação da primeira e da segunda árvore com a operação diferença.
8. O usuário insere a relação LOCACAO e o operador junção theta na área de edição.
9. O usuário realiza a ligação do operador diferença com o operador junção theta.
10. O usuário realiza a ligação da relação LOCACAO com o operador junção theta e informa a condição de junção l.codigo_locacao = l1.codigo.
11. O usuário insere a relação CLIENTE e o operador junção theta na área de edição.
12. O usuário realiza a ligação do operador junção theta inserida no passo 8 com a junção theta inserida no passo 10.
13. O usuário realiza a ligação da relação CLIENTE com a junção theta inserida no passo 11 e informa a condição de junção l1.codigo_cliente = c.codigo.
14. O usuário insere o operador projeção na área de edição.
15. O usuário realiza a ligação do operador junção theta inserido no passo 11 com o operador projeção informando a coluna NOME como parâmetro do mesmo.
16. O usuário aciona o botão executar.</td></tr>
<tr><td>Exceção</td><td>No passo 3 o usuário não informa condição de seleção, o sistema apresenta a mensagem "É necessário informar a condição de seleção.", o fluxo retorna para o passo 3.</td></tr>
<tr><td>Exceção 2</td><td>No passo 10 ou 13 o usuário não informa condição de junção, o sistema apresenta a mensagem "É necessário informar a condição de junção.", o fluxo retorna para o passo origem da exceção.</td></tr>
<tr><td>Pós-condição</td><td>O sistema deve apresentar a seguinte relação como resultado:
<table><tr><td>NOME</td></tr><tr><td>José</td></tr></table></td></tr>
</table>

Quadro 19 – Cenários do UC08

3.2.5 Demonstração da ferramenta

Para a demonstração do funcionamento será construída uma árvore de expressão respeitando os passos do caso de uso UC08- Recuperar os clientes que assistiram o filme O Poderoso Chefão mas não assistiram Cidade de Deus.

Inicialmente o usuário deve realizar o *login* na aplicação informando qual o banco desejado, o endereço do mesmo (IP e Porta), o nome do esquema de tabelas no caso do banco MySQL, usuário e senha. A Figura 12 demonstra o *login* na aplicação.

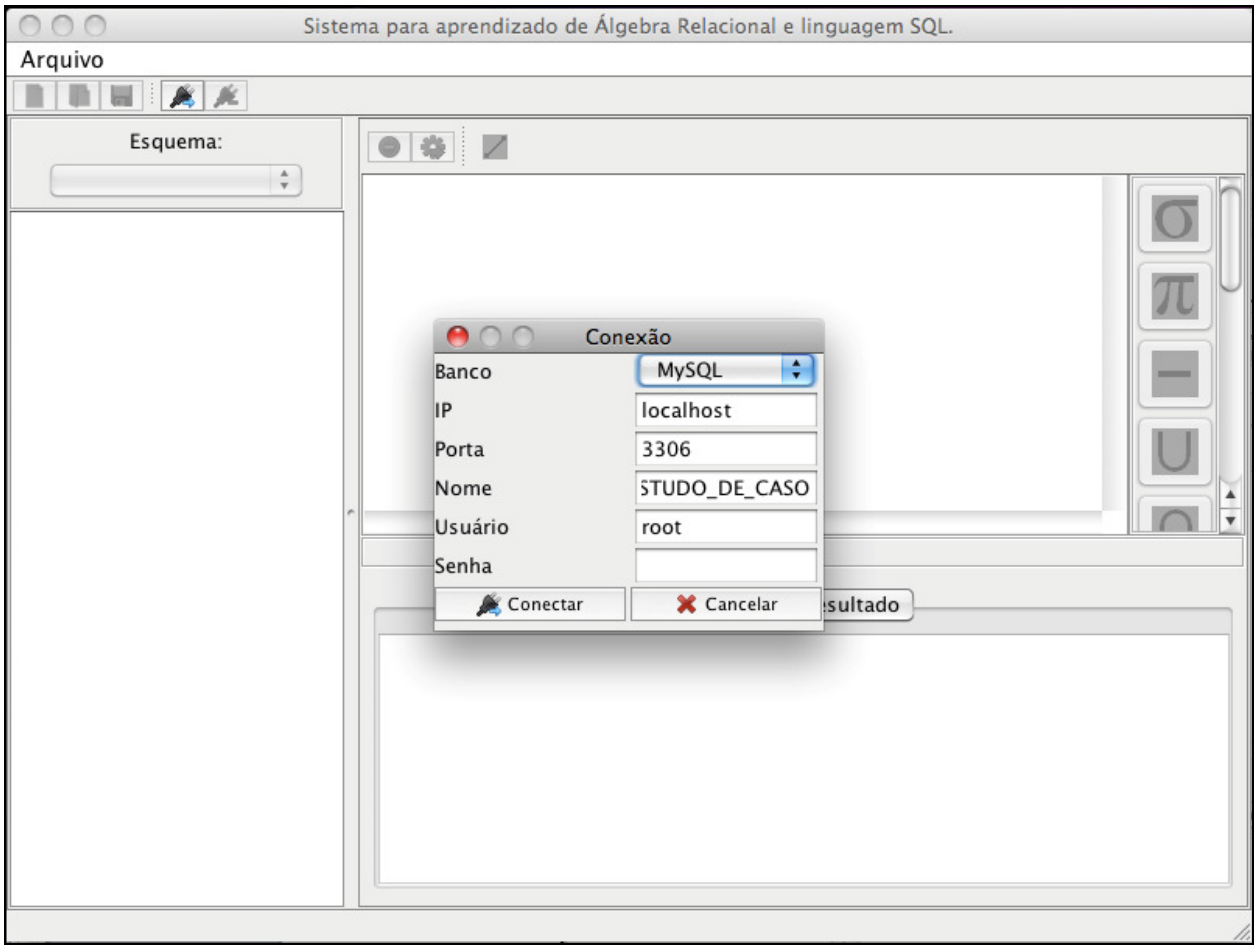


Figura 12 – Realizar *login*

Realizado o *login* a aplicação carrega as tabelas e libera os botões para construção das árvores de expressão a Figura 13 apresenta as áreas da ferramenta e os principais botões.

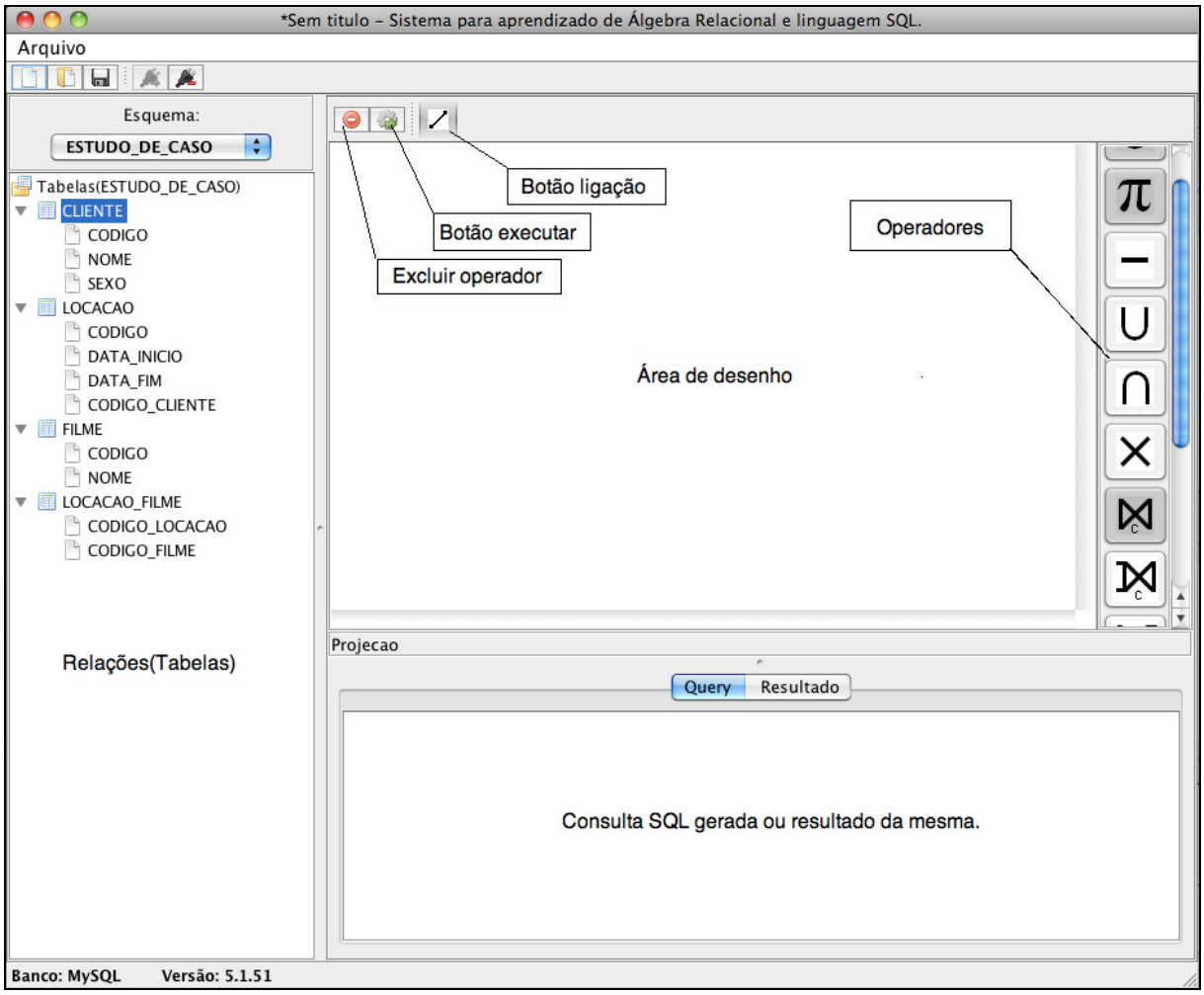


Figura 13 – Apresentação das áreas da ferramenta

Após a conexão realizada o usuário pode começar a formular o plano de consulta para o problema proposto no UC08. Deve-se então realizar os seguintes passos:

a) Passo 1: inserir a relação LOCACAO_FILME e os operadores seleção e projeção na área de edição;
b) Passo 2: efetuar a ligação da relação LOCACAO_FILME com o operador seleção;
c) Passo 3: informar a condição de seleção : CODIGO_FILME = 5;
d) Passo 4: efetuar a ligação do operador seleção com o operador projeção informando a coluna CODIGO_LOCACAO como parâmetro do mesmo.

Para inserir os operadores e relações na área de desenho o usuário deve clicar sobre o operador ou relação desejado, arrastar e soltar dentro da área de desenho.

Para os passos que consistem em efetuar ligação entre os operadores deve-se anteriormente ativar o botão ligação no topo do editor a Figura 14 apresenta a execução dos procedimentos 1, 2, 3 e 4.

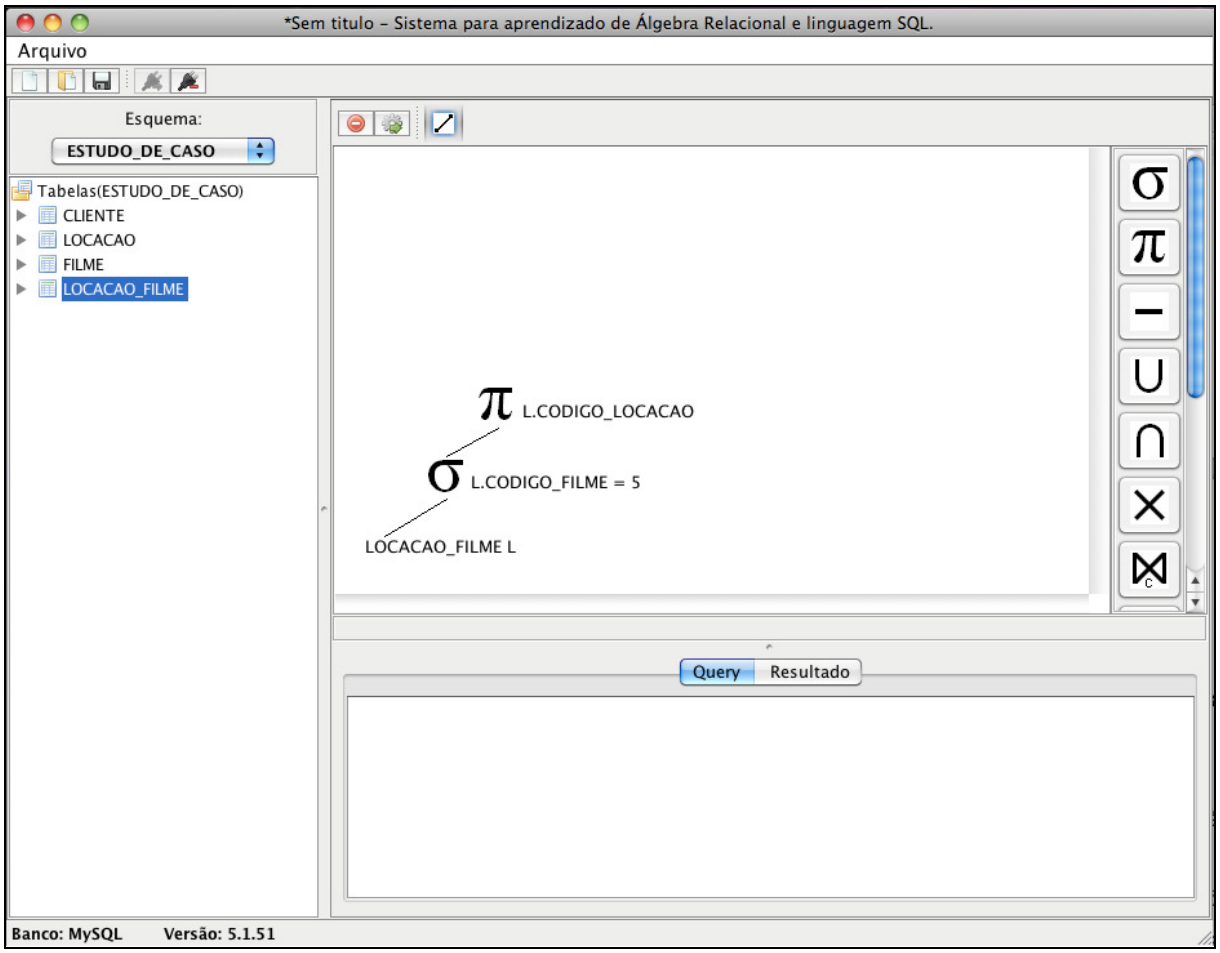


Figura 14 – Execução dos passos 1, 2, 3 e 4

O próximo passo (5) consiste em realizar novamente os passo de 1 a 4 criando uma segunda árvore substituindo a condição de seleção do passo 3 para `CODIGO_FILME` = 4.

A Figura 15 demonstra a execução do passo 5.

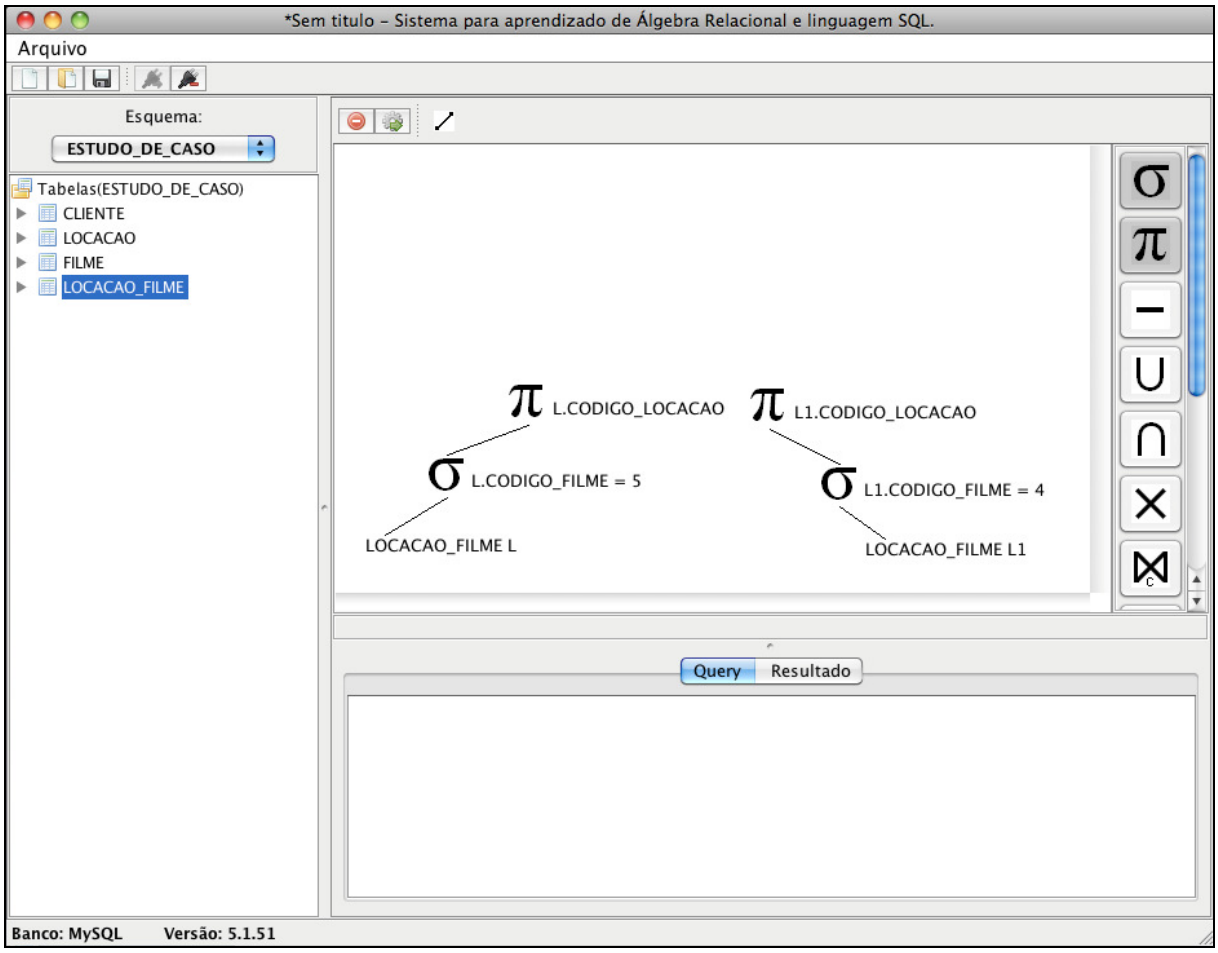


Figura 15 – Execução do passo 5

Os próximos procedimentos consistem em:

a) Passo 6: inserir a operação diferença;

b) Passo 7: efetuar a ligação da primeira e da segunda árvore com a operação diferença;

c) Passo 8: inserir a relação `LOCACAO` e o operador junção *theta* na área de edição;

d) Passo 9: realizar a ligação do operador diferença com o operador junção *theta*;

e) Passo 10: realizar a ligação da relação `LOCACAO` com o operador junção *theta* e informa a condição de junção `l.codigo_locacao = l2.codigo`.

A Figura 16 demonstra a execução dos passos 6, 7, 8, 9 e 10.

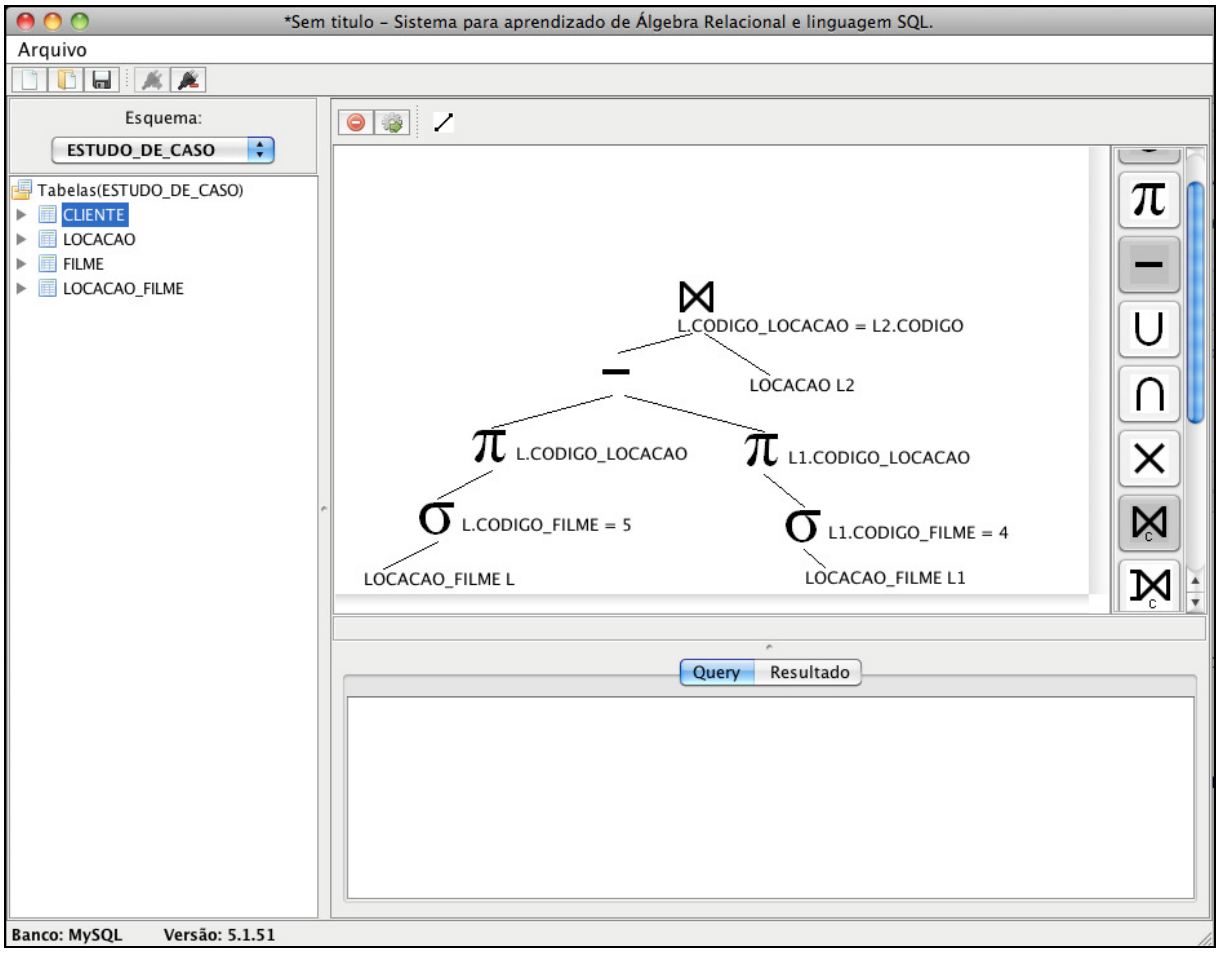


Figura 16 – Execução dos passos 6, 7, 8, 9 e 10

Conforme a sequência os próximos passos consistem em :

a) Passo 11: inserir a relação `CLIENTE` e o operador junção *theta* na área de edição;

b) Passo 12: realizar a ligação do operador junção *theta* inserida no passo 8 com a junção *theta* inserida no passo 10;

c) Passo 13: realizar a ligação da relação `CLIENTE` com a junção *theta* inserida no passo 11 e informa a condição de junção `l1.codigo_cliente = c.codigo`.

A Figura 17 demonstra a execução dos passos 11, 12 e 13.

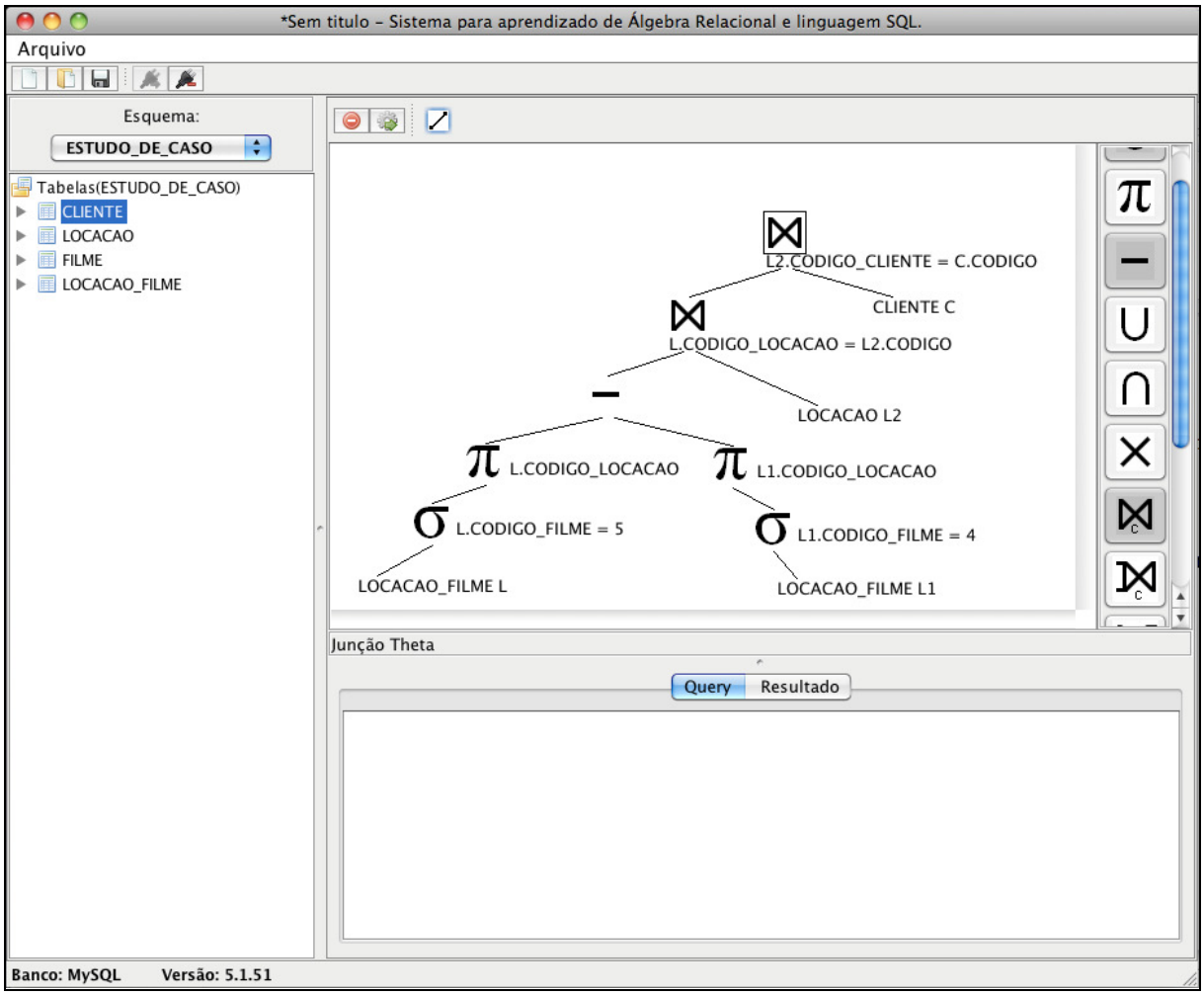


Figura 17 – Execução dos passos 11, 12 e 13

Finalizando a construção da árvore deve-se :

a) Passo 14: inserir o operador projeção na área de edição;

b) Passo 15: realizar a ligação do operador junção theta inserido no passo 11 com o operador projeção informando a coluna NOME como parâmetro do mesmo.

A Figura 18 demonstra os passos 14 e 15.

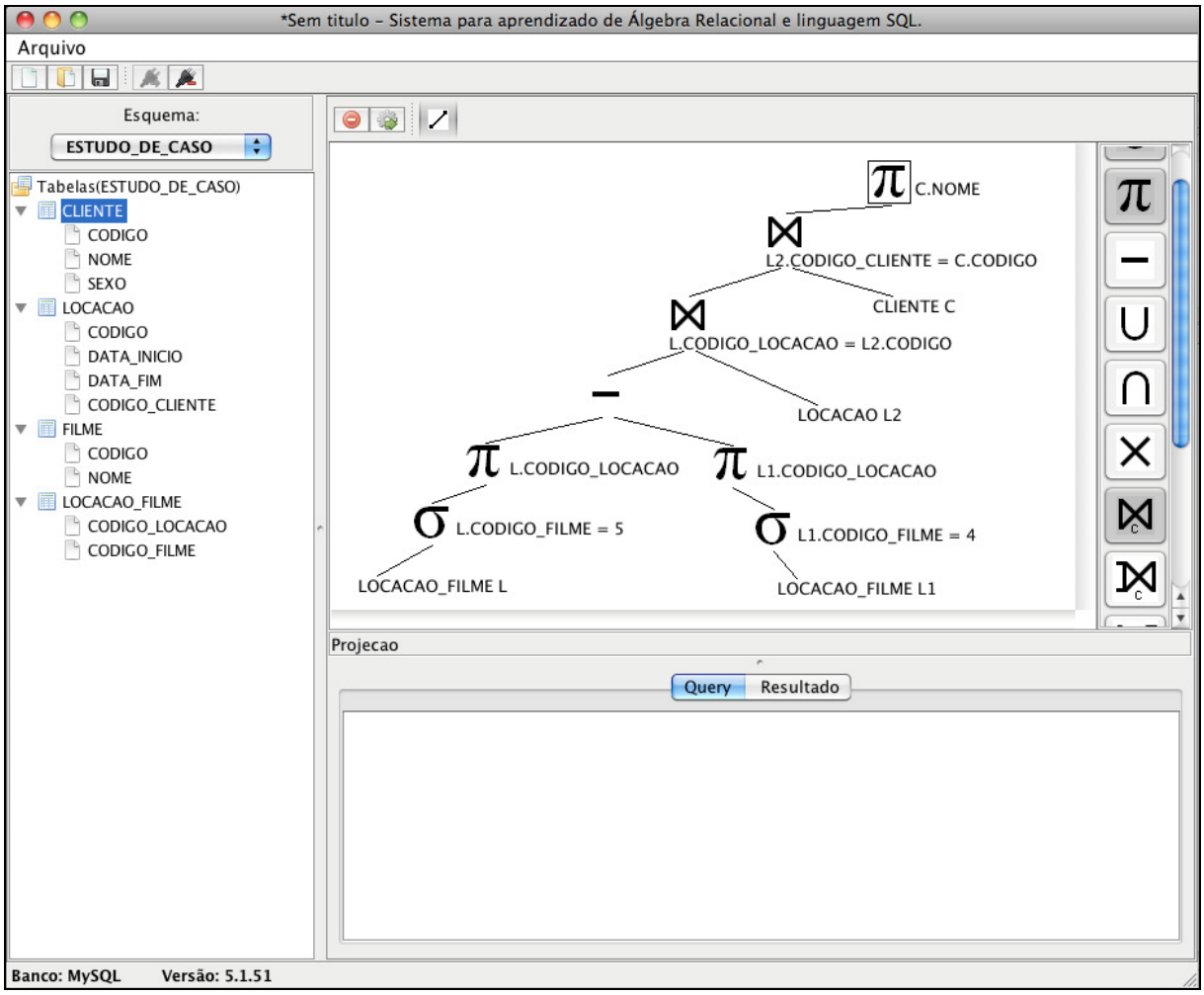


Figura 18 – Execução dos passos 14 e 15

Por fim o usuário aciona o botão executar, a Figura 19 apresenta o resultado do plano de consulta e a Figura 20 a consulta SQL gerada.

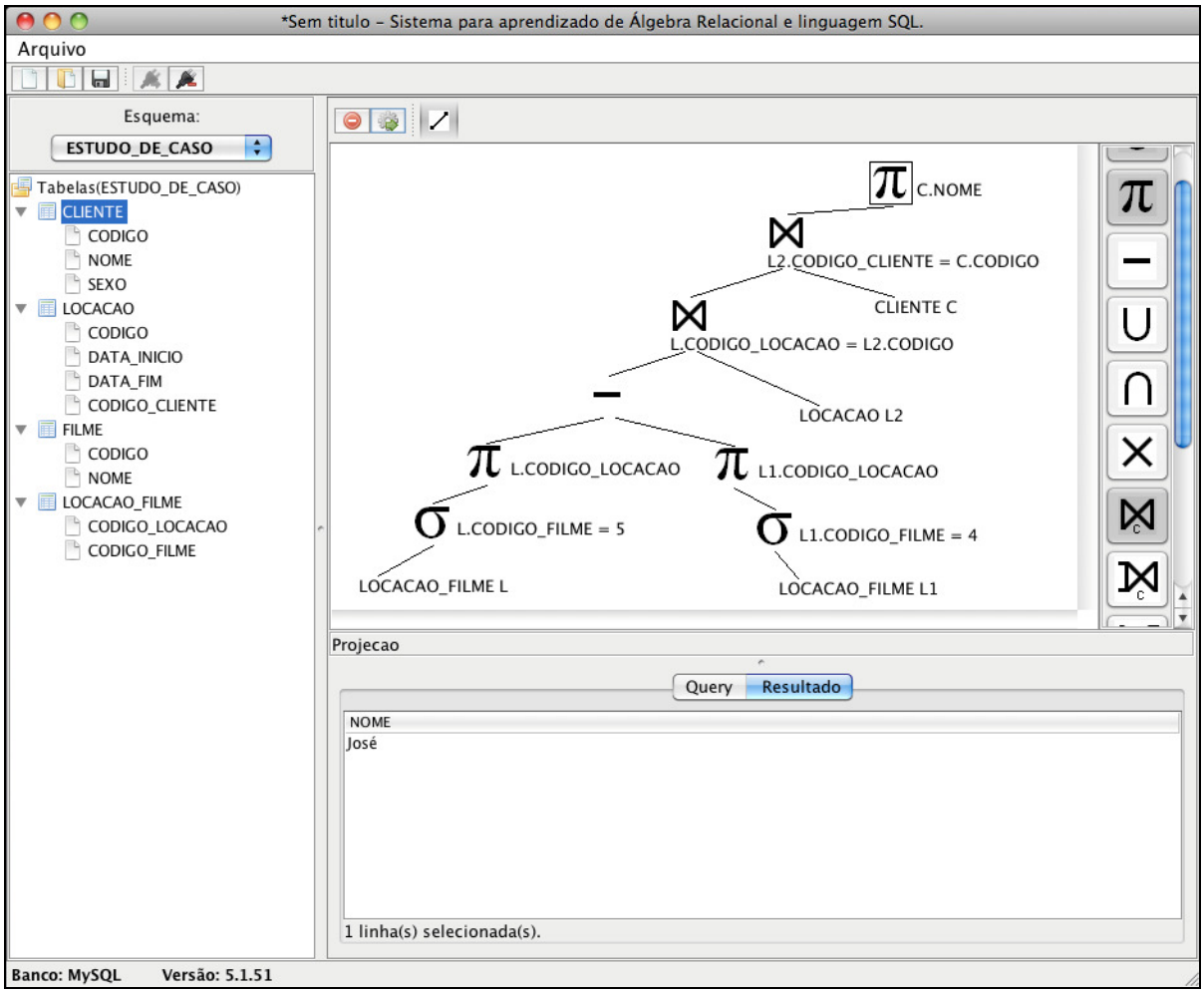


Figura 19 – Resultado do plano de consulta

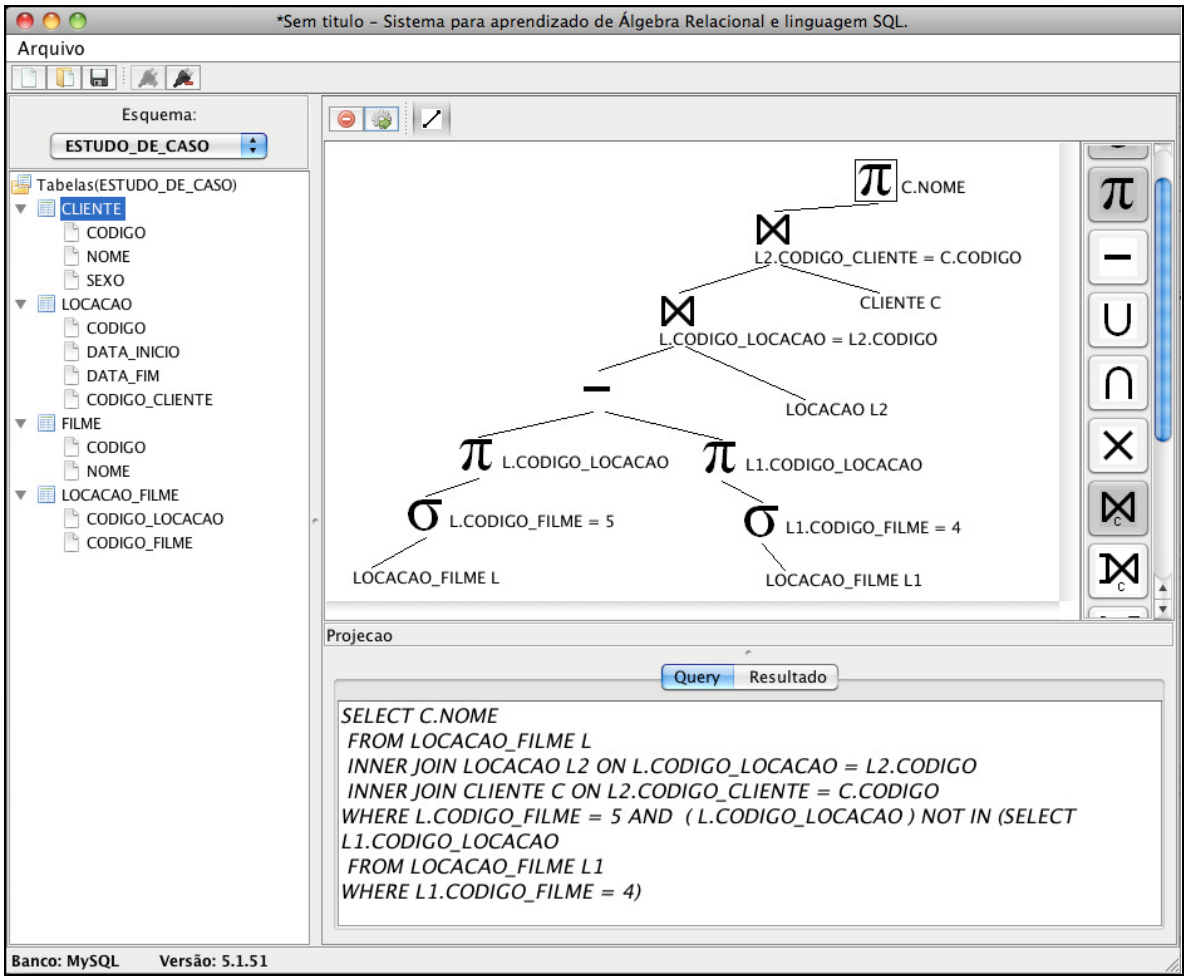


Figura 20 – Comando SQL gerado

3.3 RESULTADOS E DISCUSSÃO

Os resultados obtidos pela ferramenta foram satisfatórios. A mesma apresentou boa funcionalidade e usabilidade, é de fácil uso não o sendo necessária grande capacitação para seus usuários. Permite que sejam construídas árvores complexas conforme evidenciado no estudo de caso.

Em relação aos conceitos da álgebra relacional a aplicação trabalha com a metodologia nomeada álgebra sobre sacolas (conjuntos múltiplos), ou seja, duplicatas não são removidas como na álgebra relacional clássica.

A geração da linguagem SQL é feita realizando uma varredura na árvore construída e com o apoio de uma pilha e um *template* (modelo) de consulta SQL, os detalhes da geração são apresentados na seção 3.2.3. O uso de uma BNF (Backus-Naur Form) não foi adotado pelo motivo de que a entrada dos dados não é de forma textual e sim como uma estrutura em árvore. Outro fator determinante é que a sintaxe para construção de expressões algébricas ou árvores de expressões é a concatenação sucessiva de operadores partindo-se de uma relação. Não existem regras que definem, por exemplo, se um operador pode ou não ser ligado a outro.

Os testes realizados na ferramenta visaram principalmente a coerência do resultado das consultas SQL em relação às árvores de origem.

Em relação aos trabalhos correlatos o Quadro 28 demonstra um comparativo dos mesmos com este trabalho.

	IDFQL	EnsinAr	Ferramenta
Árvores de expressões algébricas	Sim	Não	Sim
Expressões algébricas	Não	Sim	Não
Manter arquivos	Sim	Não	Sim
Acesso a tabelas de diversos *schemas*	Não	Não	Sim
Álgebra Estendida	Não	Não	Sim

Quadro 28 – Comparativo da ferramenta com os trabalhos correlatos

4 CONCLUSÕES

Durante o desenvolvimento do trabalho foram adotadas algumas estratégias para implementar o gerador SQL. Inicialmente foi construída uma gramática no Gerador de Analisadores Léxicos e Sintáticos (GALS) para validar expressões algébricas e gerar o código, contudo seria necessária a conversão da árvore para uma expressão na forma textual, e na fase semântica realizar um reagrupamento da expressão levando em consideração a prioridade dos operadores para depois realizar a geração do código, em decorrência desses motivos esse método não foi utilizado.

Partiu-se então para varredura direta na árvore gerando uma consulta externa para cada operador, desta forma o comando SQL gerado expressava exatamente a árvore de expressão construída. Esta técnica quando aplicada a árvores muito extensas gerava comandos SQL ilegíveis e ocorriam erros de colunas com nome duplicados mesmo utilizando a sintaxe `tabela.nome_colula`.

Optou-se então por construir um *template* que expressa um comando SQL, e com o auxílio de uma pilha o mesmo é preenchido durante a visitação da árvore de expressão, esta técnica apresentou bons resultados e qualidade dos comandos, entretanto não foi possível garantir uma geração de qualquer árvore construída pelo fato de que não existe restrição ou sintaxe para construção das árvores de expressão, porém, a linguagem fim SQL possui sintaxe, sendo assim alguns tipos de construções utilizando operadores binários após o operador união podem gerar consultas SQL não funcionais.

Em relação aos objetivos inicialmente propostos a maioria foi atendido. Ficou parcialmente realizada a geração dos comandos SQL.

O presente trabalho fornece uma ferramenta de apoio ao ensino da álgebra relacional permitindo aos usuários formular planos de consulta através de árvores de expressões algébricas. A aplicação gera consultas em linguagem SQL a partir das árvores de expressão, desta forma pode-se comparar um determinado plano de consulta em álgebra relacional com uma consulta correspondente em linguagem SQL. Propicia também melhor entendimento da maneira com que os SGBDRS realizam a recuperação de dados.

No âmbito da educação, uma ferramenta que possibilite uma forma diferenciada de visão para resolver problemas e assimilar assuntos como álgebra relacional e linguagem SQL agregam sem dúvida maior qualidade e desempenho no ensino.

Durante o desenvolvimento, foram adicionados os operadores junção a esquerda e

junção a direita aumentando assim o número de operadores inicialmente propostos.

4.1 EXTENSÕES

Ficam como sugestões para trabalhos futuros os seguintes itens:

a) disponibilizar os operadores rebatizar, atribuição, junção natural e intersecção;
b) disponibilizar suporte a expressões de álgebra relacional;
c) incluir uma interface que realize a conversão de comandos SQL em árvores de expressões;
d) aprimorar a entrada de dados para os operadores, realizar validações para os nomes de colunas informados e condições de junção e seleção.

REFERÊNCIAS BIBLIOGRÁFICAS

APPEL, Ana P.; TRAINA JUNIOR, Caetano. iDFQL: uma ferramenta de apoio ao processo de ensino-aprendizagem da álgebra relacional baseado no construcionismo. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 6., 2004, Vitória. **Anais Eletrônicos…** Vitória: SBC, 2004. Não paginado. Disponível em: <http://bibliotecadigital.sbc.org.br/bibliotecadigital/download.php?paper=47>. Acesso em: 20 mar. 2010.

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

GARCIA-MOLINA, Hector; ULLMAN, Jeffrey D.; WIDOM, Jennifer. **Implementação de sistemas de bancos de dados**. Tradução Vandenberg D. de Souza. Rio de Janeiro: Campus, 2001.

MINISTÉRIO DA EDUCAÇÃO. **Diretrizes curriculares dos cursos da área de computação e informática**. [Porto Alegre], [1998?]. Disponível em: <http://www.inf.ufrgs.br/ecp/docs/diretriz.pdf >. Acesso em: 27 mar. 2010.

PAES, Enderson L. **EnsinAR:** ferramenta didática para o ensino de álgebra relacional. 2007. 151 f. Trabalho de Conclusão de Curso (Curso de Ciência da Computação) - Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://projetos.inf.ufsc.br/projetos.php?busca=sim&titulo=&autor=Ederson+Luiz+Paes&resp=&orientacao=&curso=&disciplina=&semestre_busca=&concluido_busca=&keywords=>. Acesso em: 31 mar. 2010.

SUMATHI, S; ESAKKIRAJAN, S. **Fundamentals of relational database management systems.** Springer, 2007. Disponível em: <http://books.google.com.br/books?id=RjnNA0GW0wsC&printsec=frontcover#v=onepage&q&f=false>. Acesso em: 24 out. 2010.

RAMAKRISHNAN, Raghu. **Database management systems**. Boston: WCB McGraw-Hill, 1998.