UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

IMPLEMENTAÇÃO DE UM SIMULADOR DE CONSULTAS EM ÁLGEBRA RELACIONAL

TRABALHO DE GRADUAÇÃO

Larissa Rodrigues Lautert

Santa Maria, RS, Brasil

2010

IMPLEMENTAÇÃO DE UM SIMULADOR DE CONSULTAS EM ÁLGEBRA RELACIONAL

por

Larissa Rodrigues Lautert

Trabalho de Graduação apresentado ao Curso de Ciência da Computação
da Universidade Federal de Santa Maria (UFSM, RS), como requisito
parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Profª Drª Deise de Brum Saccol

Trabalho de Graduação Nº310
Santa Maria, RS, Brasil

2010

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

IMPLEMENTAÇÃO DE UM SIMULADOR DE CONSULTAS EM ÁLGEBRA RELACIONAL

elaborado por
Larissa Rodrigues Lautert

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Profª Drª Deise de Brum Saccol
(Presidente/Orientador)

Profª Drª Andrea Schwertner Charão (UFSM)

Profª Drª Marcia Pasin (UFSM)

Santa Maria, 09 de dezembro de 2010.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

IMPLEMENTAÇÃO DE UM SIMULADOR DE CONSULTAS EM ÁLGEBRA RELACIONAL

Autor: Larissa Rodrigues Lautert
Orientador: Profª Drª Deise de Brum Saccol
Local e data da defesa: Santa Maria, 09 de dezembro de 2010.

Um Banco de Dados Relacional é representado por uma coleção de tabelas, onde cada linha representa uma relação entre um conjunto de valores. Uma de suas vantagens é a facilidade de acesso aos dados, possibilitando que o usuário utilize uma grande variedade de abordagens no tratamento das informações. Para efetuar consultas em bancos de dados relacionais, pode-se utilizar a Álgebra Relacional, uma linguagem de consultas formal. Existem seis operações fundamentais na álgebra relacional, sendo três unárias e três binárias. O resultado de qualquer operação entre relações, ou tabelas, é uma nova relação. Por se tratar de uma linguagem formal, poucas ferramentas interpretam consultas em álgebra relacional, dificultando a aprendizagem dos alunos. Foram analisados os pontos positivos e negativos de duas destas ferramentas para ver o que ainda pode ser feito. O presente trabalho propõe um programa simulador de consultas em álgebra relacional para auxiliar na aprendizagem das operações fundamentais. Foram exploradas as carências dos *software* existentes e implementada uma ferramenta com base nessas deficiências.

Palavras-chave: Álgebra relacional, SQL, banco de dados, Java, ensino.

ABSTRACT

Undergraduate Final Work
Undergraduate Program in Computer Science
Federal University of Santa Maria

IMPLEMENTATION OF A RELATIONAL ALGEBRA QUERY SIMULATOR
Author: Larissa Rodrigues Lautert
Advisor: Prof[a] Dr[a] Deise de Brum Saccol

A Relational Database is represented by a collection of tables, where each tuple represents a relation between a set of values. One of its advantages is the data access facility, enabling the user to use a wide range of approaches for information treatment. To submit queries in relational databases it can be used Relational Algebra, a formal query language. There are six fundamental operations in relational algebra, three unary and three binary. The result of any operation between relations, or tables, is a new relation. Because it is a formal language, few tools interpret queries in relational algebra format, making students learning harder. We analyzed two of these tools strengths and deficiencies in order to see what can still be done. This work proposes a relational algebra query simulator software to support relational algebra operations learning. We analyzed existent software needs and implemented a tool based on these requirements.

Keywords: Relational algebra, SQL, database, Java, teaching.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SGBD	Sistema Gerenciador de Banco de Dados
BDR	Bancos de Dados Relacionais
SQL	Structured Query Language
IDE	Integrated Development Environment
HSQLDB	HyperSQL Database

SUMÁRIO

1 INTRODUÇÃO

No período anterior à informática, dados eram armazenados e organizados através de arquivos de papel, o que era bastante custoso e envolvia um grande número de pessoas conforme a quantidade de dados. Na década de 60, empresas começaram a usar computadores para armazenar seus dados. Nos dois principais modelos existentes, os dados eram representados através de registros e o acesso a eles era feito com operações de baixo nível. Na década de 70, foram desenvolvidos dois protótipos de sistema relacional, cada um com sua linguagem de consulta própria. Nesse período foi proposto o modelo Entidade-Relacionamento para projetos de banco de dados, possibilitando a abstração do armazenamento de dados. Na década seguinte, sistemas relacionais popularizaram-se devido à sua facilidade em relação aos sistemas utilizados anteriormente. Foi também nessa época que a *Linguagem de Consultas Estruturada* (SQL) tornou-se um padrão mundial utilizado até os dias de hoje (REZENDE, 2007).

Atualmente são utilizados programas chamados de Sistemas Gerenciadores de Banco de Dados (SGBD) que realizam o gerenciamento e o acesso aos dados em Bancos de Dados Relacionais (BDR). OS BDR utilizam o Modelo Relacional, pois armazenam seus dados na forma de tabelas. Cada uma destas tabelas possui um atributo (ou atributos) chamado *superchave*, que permite identificar qualquer conjunto de valores presente. Para efetuar consultas em BDR, utiliza-se uma linguagem de consulta processada pelo SGBD.

Álgebra relacional é a linguagem procedural utilizada para efetuar consultas em um BDR. Consiste na base para linguagens de consulta em banco de dados, como a SQL. Outra linguagem utilizada para realizar consultas em um BDR é o Cálculo Relacional. Nele, descreve-se o conjunto de respostas desejado sem especificar como ele será computado, ao contrário da álgebra relacional.

1.1 Justificativas

O estudo e a compreensão do modelo relacional e da álgebra relacional são fundamentais para estudantes de cursos de Banco de Dados. Para facilitar seu aprendizado, é conveniente ter um ambiente para testes de consultas nessa linguagem. Existem alguns *software* que permitem tais testes, dentre os quais destacam-se *LEAP* e *DBTools 2000*. Ambos deixam a cargo do usuário a instalação e configuração do banco da dados, sendo que esta atividade é mais trabalhosa no *DBTools 2000*. O primeiro suporta diversas operações além das fundamentais da álgebra relacional, como criação, edição e remoção de tabelas e até do banco de dados. Porém, possui uma interface não intuitiva e não utiliza os símbolos e a sintaxe adotados pela maioria dos livros de banco de dados. Em oposição, o segundo possui interface bastante amigável para criação de consultas e apresentação de seus resultados. No entanto, durante a execução do programa, através do *prompt* que roda em *background*, percebeu-se que algumas exceções não foram tratadas quando, por exemplo, o usuário digita uma consulta incorreta.

As ferramentas disponíveis analisadas são pouco utilizadas devido aos pontos negativos relatados. Constatou-se a necessidade de um *software* estável, funcional, intuitivo e que siga o padrão de simbologia dos livros. Com ele, os alunos poderão utilizar os conceitos aprendidos em aula para criar consultas e verificar se os resultados correspondem aos esperados. Por isso, este trabalho propõe a implementação de um interpretador de consultas em álgebra relacional com esses requisitos.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho consiste em desenvolver uma ferramenta estável e intuitiva que interprete consultas em álgebra relacional, após o estudo de seus conceitos e operações. Ao final, espera-se um *software* funcional, estável e de fácil utilização para auxiliar no ensino de Banco de Dados.

1.2.2 Objetivos Específicos

- Analisar os *software* simuladores de consultas em álgebra relacional existentes;
- Estudar as operações fundamentais de álgebra relacional;

- Implementar um simulador de consultas em álgebra relacional explorando as carências dos existentes;
- Realizar testes no *software* implementado.

1.3 Organização do Texto

De modo a estruturar o texto, o Capítulo 2 apresenta uma revisão de leitura sobre conceitos envolvidos em Banco de Dados Relacionais. A linguagem de consultas *Álgebra Relacional* e suas principais operações são explicadas com exemplos utilizando a mesma base de dados em que serão feitos os testes da ferramenta proposta. Além disso, faz-se uma análise dos principais *software* simuladores de consultas em álgebra relacional disponíveis, destacando seus pontos positivos e negativos.

O Capítulo 3 contém a proposta da ferramenta simuladora de consultas em álgebra relacional, explicando sua finalidade. A abordagem adotada para simulação destas consultas é apresentada com exemplos.

O Capítulo 4 explica como a ferramenta foi desenvolvida, com uma breve descrição da implementação. Figuras mostram todas as funcionalidades presentes na interface e explica-se como é a utilização do *software* através de exemplos. Após, faz-se uma análise da ferramenta e compara-se com as outras existentes.

Por fim, o Capítulo 5 encerra o trabalho com as considerações finais e sugestões de melhorias para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo destina-se à definição dos conceitos usados em Banco de Dados Relacionais e Álgebra Relacional. Através da apresentação dessas definições, será possível ter a devida compreensão dos fundamentos utilizados na concepção da ferramenta proposta. As operações de álgebra relacional serão explicadas com exemplos, pois elas serão implementadas no *software* proposto.

2.1 Conceitos Fundamentais

Segundo Heuser (2000), os primeiros programas para computadores incorporavam todas as operações de armazenamento de dados. Com o crescimento dos sistemas computacionais distribuídos e aumento de dados, percebeu-se a necessidade de criar um grande repositório compartilhado de dados. Criaram-se então os Bancos de Dados, coleções de dados inter-relacionados dispostos em uma estrutura regular. A partir de então, o programador não precisava mais se preocupar com a forma de armazenamento de dados, pois utilizava um Sistema de Gerência de Banco de Dados (SGBD).

Um SGBD consiste em um *software* que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados. Dessa forma, a manutenção dos *software* desenvolvidos e a produtividade dos programadores melhora, pois os programas utilizam funções já existentes para tratar do armazenamento de dados.

2.2 Banco de Dados Relacionais

Um Banco de Dados Relacional é representado por uma coleção de *tabelas*, também chamadas de *relações*, onde cada linha representa uma relação entre um conjunto de valores (SILBERSCHATZ; KORTH; SUDARSHAN, 2006). Uma de suas vantagens é a facilidade de acesso aos dados, possibilitando que o usuário utilize uma grande variedade

de abordagens no tratamento das informações.

2.2.1 Modelo Relacional

O Modelo Relacional foi proposto por Edgard Frank Codd como um modelo genérico de dados (CODD, 1970). Neste modelo amplamente utilizado, todos os dados são armazenados em tabelas. Cada tabela possui um nome único e cada linha da tabela representa um relacionamento entre um conjunto de valores. Dessa forma, é possível estabelecer um método declarativo para especificar dados e consultas, ou seja, ir direto ao local do banco de dados que possui determinada informação desejada. O Sistema Gerenciador de Banco de Dados toma conta de como processar a consulta feita e retornar a resposta correta.

O conceito básico para estabelecer relações entre linhas de tabelas de um banco de dados relacional é o da *chave*. Em um banco de dados relacional, há ao menos três tipos de chaves a considerar: a *chave primária*, a *chave alternativa* e a *chave estrangeira*. Uma *chave primária* é uma coluna, ou combinação de colunas, cujo valor distingue uma linha das demais dentro de determinada tabela. Por exemplo, a coluna *codEmp* em uma tabela *Empregado*. Em alguns casos, mais de uma coluna, ou combinações de colunas, pode servir para distinguir uma linha das demais. Uma das colunas, ou combinação de colunas, é escolhida como chave primária e as demais são denominadas *chaves alternativas*. Já a *chave estrangeira* é uma coluna, ou combinação de colunas, cujo valor aparece necessariamente na chave primária de uma tabela. A chave estrangeira é o mecanismo que permite a implementação de relacionamentos em um banco de dados relacional (HEUSER, 2000).

Para efetuar consultas em bancos de dados relacionais, pode-se utilizar álgebra relacional ou cálculo relacional, linguagens de consultas formais. A álgebra relacional consiste em uma linguagem de consulta onde o procedimento para encontrar a resposta desejada é descrito, ou seja, é uma linguagem procedural. Em oposição, no cálculo relacional descreve-se o conjunto de respostas esperadas sem a necessidade de explicitar como este conjunto é computado, pois se trata de uma linguagem declarativa. Essas duas linguagens possuem poder de expressão equivalentes, ou seja, toda consulta válida em cálculo relacional pode ser escrita em álgebra relacional e vice-versa.

As linguagens anteriormente citadas não são usadas em aplicações reais. Para esse fim existe a *Linguagem de Consulta Estruturada* (SQL), inspirada em muitas características

da Álgebra Relacional. Seu propósito inicial era demonstrar a viabilidade da implementação do modelo relacional. Devido à sua simplicidade e facilidade de uso, tornou-se um padrão de banco de dados. Consiste em uma linguagem de pesquisa declarativa, pois especifica a forma do resultado, e não como obtê-lo.

2.2.2 Álgebra Relacional

A álgebra relacional é uma linguagem de consulta formal onde o usuário fornece as instruções que representam operações na base de dados para calcular o resultado. As operações fundamentais na álgebra relacional são seleção, projeção, união, diferença, produto cartesiano e renomeação. Existem também as operações adicionais, definidas a partir das fundamentais. O resultado de qualquer operação entre relações, também chamadas de tabelas, é uma nova relação. Uma linguagem de manipulação de dados completa inclui ainda operações de modificação, inserção e exclusão de dados (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

Para exemplificar as operações da Álgebra Relacional, serão consideradas as seguintes tabelas:

- Projeto (idProjeto, titulo, tipo, coordenador) onde *idProjeto* é chave primária.
- Aluno (matricula, nome, dataNascimento, email) onde *matricula* é chave primária.
- Participantes (matricula, idProjeto, dataInicio, dataFim) onde *matricula* e *idProjeto* compõe a chave primária e são chaves estrangeiras. O atributo *matricula* referencia a *matricula* da tabela *Aluno* e *idProjeto* referencia o *idProjeto* da tabela *Projeto*.
- BolsistaPET (matricula, dataIngresso, dataSaida) onde *matricula* é chave primária e estrangeira. Ela referencia o atributo *matricula* da tabela *Aluno*.

As Tabelas 2.1, 2.2, 2.3 e 2.4 mostram uma instância no Banco de Dados, ou seja, uma "foto" dos dados armazenados em determinado momento.

Tabela 2.1: Tabela Projeto.

idProjeto	titulo	tipo	coordenador
0	Monitoria em Algoritmos	ensino	Luiz
1	Inteligência artificial em jogos	pesquisa	Luiz
2	Universitar: o desafio populacional	extensão	Antônio

As seções a seguir detalham as operações fundamentais.

Tabela 2.2: Tabela Aluno.

matricula	nome	dataNascimento	email
1257	Leandro	1989-08-15	leandro@mail.com
1282	Vinicius	1989-02-24	vinicius@mail.com
1315	Daniela	1990-01-07	daniela@mail.com
1322	Paulo	1990-09-26	paulo@mail.com
1485	Fernando	1988-03-15	fernando@mail.com

Tabela 2.3: Tabela Participantes.

matricula	idProjeto	dataInicio	dataFim
1257	2	2009-10-01	2010-10-01
1282	0	2010-03-01	2010-07-15
1315	1	2010-08-10	2010-12-15

Tabela 2.4: Tabela BolsistaPET.

matricula	dataIngresso	dataSaida
1282	2008-07-22	2010-09-04
1315	2009-03-30	2010-10-23
1322	2009-10-01	2010-08-15

2.2.2.1 Seleção

A operação de *seleção*, representada pelo operador σ, obtém tuplas de acordo com um critério informado. Esta operação é classificada como unária, pois se aplica a apenas uma relação. Seu resultado será uma tabela com todos os registros que satisfizerem à condição declarada. Como exemplo, a operação que seleciona tuplas na tabela *Bolsista-PET* de acordo com a condição *dataIngresso >= 2009-01-01*, isto é, todos os alunos que ingressaram no grupo PET no ano de 2009 ou depois. Essa operação é escrita da seguinte forma em álgebra relacional:

$$\sigma_{dataIngresso>=2009-01-01}(BolsistaPET)$$

Como resposta, tem-se a relação mostrada na tabela 2.5.

Tabela 2.5: Resultado da operação de Seleção.

matricula	dataIngresso	dataSaida
1315	2009-03-30	2010-10-23
1322	2009-10-01	2010-08-15

2.2.2.2 *Projeção*

A operação de projeção, representada pelo operador π, é unária e filtra os atributos desejados na relação resultante. Como citado anteriormente, o resultado de uma operação de álgebra relacional é uma relação, de forma que pode-se aplicar outra operação a essa relação resultante. Tal propriedade é mostrada no exemplo abaixo, onde a operação de projeção é aplicada ao resultado de uma seleção. A operação de seleção tem como resultado os projetos coordenados pelo professor Luiz. A operação de projeção aplicada a seguir resulta no título dos projetos do professor Luiz.

$$\pi_{titulo}(\sigma_{coordenador=''Luiz''}(Projeto))$$

Como resposta, tem-se a relação mostrada na tabela 2.6.

Tabela 2.6: Resultado da operação de Projeção.

titulo
Inteligência artificial em jogos
Monitoria em Algoritmos

2.2.2.3 *Renomeação*

A terceira e última operação unária fundamental é a renomeação, representada pelo operador ρ. Através dela, pode-se renomear uma tabela para referências na mesma consulta. Deve ser usada quando precisa-se, para cada tupla gerada no resultado, acessar mais de uma tupla da mesma tabela de entrada. No exemplo, dá-se o nome de *ProjetosLuiz* à tabela que contém os seus projetos.

$$\rho_{ProjetosLuiz}(\sigma_{coordenador=''Luiz''}(Projeto))$$

Como resposta, tem-se a relação mostrada na tabela 2.7.

Tabela 2.7: Tabela ProjetosLuiz, resultado da operação de Renomeação.

idProjeto	titulo	tipo	coordenador
0	Monitoria em Algoritmos	ensino	Luiz
1	Inteligência artificial em jogos	pesquisa	Luiz

2.2.2.4 União

Ao contrário das anteriores, a operação de união, representada pelo operador $\cup$, é binária. A relação resultante contém todos os elementos pertencentes às relações originais sem duplicidade de tuplas. Para realizar essa operação, as duas relações precisam ser *união compatíveis*, ou seja, devem possuir o mesmo número de colunas e os domínios das respectivas colunas devem ser os mesmos. Como exemplo, a união entre os alunos vinculados a projetos e os alunos bolsistas do programa PET. Como as tabelas não são compatíveis, primeiro fez-se uma projeção no atributo *matricula* de cada tabela.

$$\pi_{matricula}(Participantes) \cup \pi_{matricula}(BolsistaPET)$$

Como resposta, tem-se a relação mostrada na tabela 2.8.

Tabela 2.8: Resultado da operação de União.

matricula
1282
1315
1322
1257

2.2.2.5 Diferença

A operação de diferença, representada pelo operador $-$, é aplicada a duas relações e resulta nas tuplas presentes na primeira e ausentes na segunda. Assim como na operação anterior, as duas relações precisam ter estruturas compatíveis. No exemplo desta operação deseja-se descobrir quais alunos nunca foram bolsistas do programa PET. Como na operação anterior, primeiro aplicou-se a operação de projeção no atributo *matricula* de cada tabela.

$$\pi_{matricula}(Aluno) - \pi_{matricula}(BolsistaPET)$$

Como resposta, tem-se a relação mostrada na tabela 2.9.

Tabela 2.9: Resultado da operação de Diferença.

matricula
1257
1485

2.2.2.6 Produto Cartesiano

O produto cartesiano, representado pelo operador $\times$, é uma operação cujo resultado contém uma combinação de todas as tuplas entre as duas relações de entrada. É utilizado quando se deseja cruzar dados de duas relações diferentes. No exemplo, realizou-se o produto cartesiano das tabelas *Aluno* e *BolsistaPET*. Antes da operação de produto cartesiano, é feita uma projeção sobre os atributos *matricula* e *nome* da tabela *Aluno* e *matricula* e *dataIngresso* da tabela *BolsistaPET* a fim de obter uma relação mais concisa como resposta.

$$\pi_{matricula,nome}(Aluno) \times \pi_{matricula,dataIngresso}(BolsistaPET)$$

Como resposta, tem-se a relação mostrada na tabela 2.10.

Tabela 2.10: Resultado da operação de Produto Cartesiano.

Aluno.matricula	nome	Bolsista.matricula	dataIngresso
1282	Vinicius	1282	2008-07-22
1257	Leandro	1282	2008-07-22
1315	Daniela	1282	2008-07-22
1322	Paulo	1282	2008-07-22
1485	Fernando	1282	2008-07-22
1282	Vinicius	1315	2009-03-30
1257	Leandro	1315	2009-03-30
1315	Daniela	1315	2009-03-30
1322	Paulo	1315	2009-03-30
1485	Fernando	1315	2009-03-30
1282	Vinicius	1322	2009-10-01
1257	Leandro	1322	2009-10-01
1315	Daniela	1322	2009-10-01
1322	Paulo	1322	2009-10-01
1485	Fernando	1322	2009-10-01

2.2.2.7 Junção

Com o objetivo de associar dados de tabelas relacionadas, pode-se realizar uma combinação das operações de seleção e produto cartesiano. Como resultado, tem-se a junção, uma operação adicional da álgebra relacional representada pelo operador $\bowtie$. A seguir são descritos três tipos de junção com o mesmo exemplo para fins de comparação. Nele, deseja-se encontrar o *nome* de todos os alunos que foram bolsistas do grupo PET. A relação resultante é a mesma para as três consultas e está representada na tabela 2.11.

2.2.2.7.1 Junção *Theta*

A junção *theta* é uma junção genérica, pois pode-se utilizar qualquer condição na seleção. A consulta a seguir mostra a sintaxe desta operação aplicada ao exemplo citado anteriormente.

$$\pi_{nome}(Aluno \bowtie (Aluno.matricula = BolsistaPET.matricula)BolsistaPET)$$

2.2.2.7.2 Equijunção

A equijunção realiza a operação de igualdade para comparar uma ou mais colunas indicadas. A consulta a seguir mostra a sintaxe desta operação aplicada ao exemplo citado anteriormente.

$$\pi_{nome}(Aluno \bowtie (matricula), (matricula)BolsistaPET)$$

2.2.2.7.3 Junção Natural

A junção natural compara colunas de mesmo nome, sem precisar expecificá-las. A consulta a seguir mostra a sintaxe desta operação aplicada ao exemplo citado anteriormente.

$$\pi_{nome}(Aluno \bowtie BolsistaPET)$$

Tabela 2.11: Resultado das operações de Junção.

nome
Vinicius
Daniela
Paulo

2.2.2.8 Intersecção

Assim como a operação anterior, a intersecção é uma operação adicional da álgebra relacional e é representada pelo operador $\cap$. Deve ser aplicada a duas relações compatíveis a fim de obter uma relação resultante com os registros em comum. No exemplo, deseja-se encontrar a *matricula* de todos alunos que foram bolsistas do grupo PET e participaram de algum projeto.

$$\pi_{matricula}(Participantes) \cap \pi_{matricula}(BolsistaPET)$$

Como resposta, tem-se a relação mostrada na tabela 2.12.

Tabela 2.12: Resultado da operação de Intersecção.

matricula
1282
1315

2.3 Análise de Ferramentas de Consulta em Álgebra Relacional

Foram analisados dois *software* livres simuladores de consultas em álgebra relacional. O primeiro, *LEAP* (LEYTON; HILL, 1995), foi desenvolvido na Escócia. A versão testada, com interface para Windows, foi lançada em 1998. O segundo *software* testado foi *DBTools 2000* (DBTOOLS, 2000), desenvolvido em Georgia Tech. A seguir serão analisados seus pontos positivos e negativos considerando principalmente suas funcionalidades e usabilidade.

2.3.1 Análise do *Software LEAP*

2.3.1.1 Instalação

A instalação do *LEAP* é rápida e fácil. A execução do arquivo "install.bat" constrói o banco de dados necessário e roda o programa.

2.3.1.2 Operações Suportadas

O *software* testado suporta todas as operações fundamentais da álgebra relacional (seleção, projeção, união, intersecção, produto cartesiano e renomeação), diferença, junção e junção natural. Além destas operações de consulta, o *LEAP* permite criação, edição e remoção de tabelas e bancos de dados.

2.3.1.3 Interface e Usabilidade

A interface do *LEAP* é somente textual, como mostra a figura 2.1. Para usar o programa, é preciso ler a documentação existente para saber quais operações são suportadas e qual é o formato esperado de entrada, já que não adota a sintaxe e os símbolos da álgebra relacional utilizados na maioria dos livros clássicos de banco de dados. Para realizar

a operação de seleção, por exemplo, ao invés de utilizar a letra grega σ, o usuário deve digitar “select”. Após ler a documentação, a utilização fica mais simples. O *software* possui comandos para listar as tabelas presentes no banco de dados, listar o conteúdo de determinada tabela, realizar consultas em álgebra relacional e alterar tabelas, como citado na seção anterior. Permite a criação de um novo banco de dados e construção de todas as tabelas deste. Oferece a possibilidade de editar algumas variáveis do programa para, por exemplo, acompanhar o que está ocorrendo na execução de cada consulta.

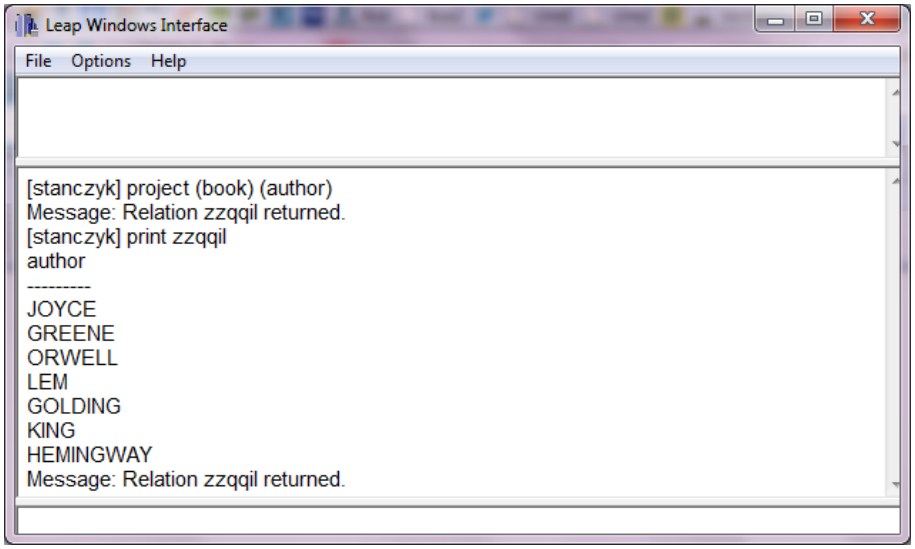

Figura 2.1: Interface da ferramenta *LEAP*.

2.3.1.4 Pontos Positivos

O *software* suporta diversas operações além das fundamentais da álgebra relacional, como criação, edição e remoção de tabelas e até do banco de dados.

2.3.1.5 Pontos Negativos

A interface não é nada intuitiva, pois exige que o usuário leia a documentação para saber como usar o programa. O fato de não utilizar a sintaxe e os símbolos adotados pelos livros também constitui outro ponto negativo, pois dificulta o uso e a compreensão por parte dos alunos.

2.3.2 Análise do *Software DBTools 2000*

2.3.2.1 Instalação

O *DBTools 2000* deixa a cargo do usuário a instalação de um Sistema de Gerenciamento de Banco de Dados e a criação de uma conexão com o banco de dados. Para testar o programa, instalou-se Microsoft Access em ambiente Windows.

2.3.2.2 Operações Suportadas

O *software* permite execução de consultas em SQL e álgebra relacional. O presente relatório analisa apenas a segunda parte. Suporta todas as operações fundamentais da álgebra relacional: seleção, projeção, união, diferença, produto cartesiano e renomeação. Além destas, suporta ainda a operação de intersecção.

2.3.2.3 Interface e Usabilidade

A interface do *DBTools 2000* possui espaço para digitação de consultas, visualização de consultas anteriores, resultados, tabelas presentes no banco e seus atributos. A interface para realização de consultas, mostrada na figura 2.2, possui botões com os símbolos das operações suportadas e legendas com o formato esperado para cada operação. Por exemplo, o botão σ possui a legenda "select: σ(cond)(R)". Possui ainda um botão para a visualização da consulta em SQL equivalente à em álgebra relacional digitada.

2.3.2.4 Pontos Positivos

O *software* apresenta uma interface com o usuário bastante intuitiva, tanto na criação da consulta em álgebra relacional, com um botão para cada operação, quanto na apresentação dos resultados.

2.3.2.5 Pontos Negativos

Os pontos negativos do programa já começam na instalação, pois fica a cargo do usuário a parte de configuração do banco de dados. Durante a execução do programa, através do *prompt* que roda em *background*, percebeu-se que algumas exceções não foram tratadas quando, por exemplo, gera-se o SQL de uma consulta incorreta. Outro ponto negativo é a interação com o usuário nas mensagens de erro, que não especificam o que houve de errado. Além disso, alguns erros são mostrados apenas no *prompt*, por exemplo

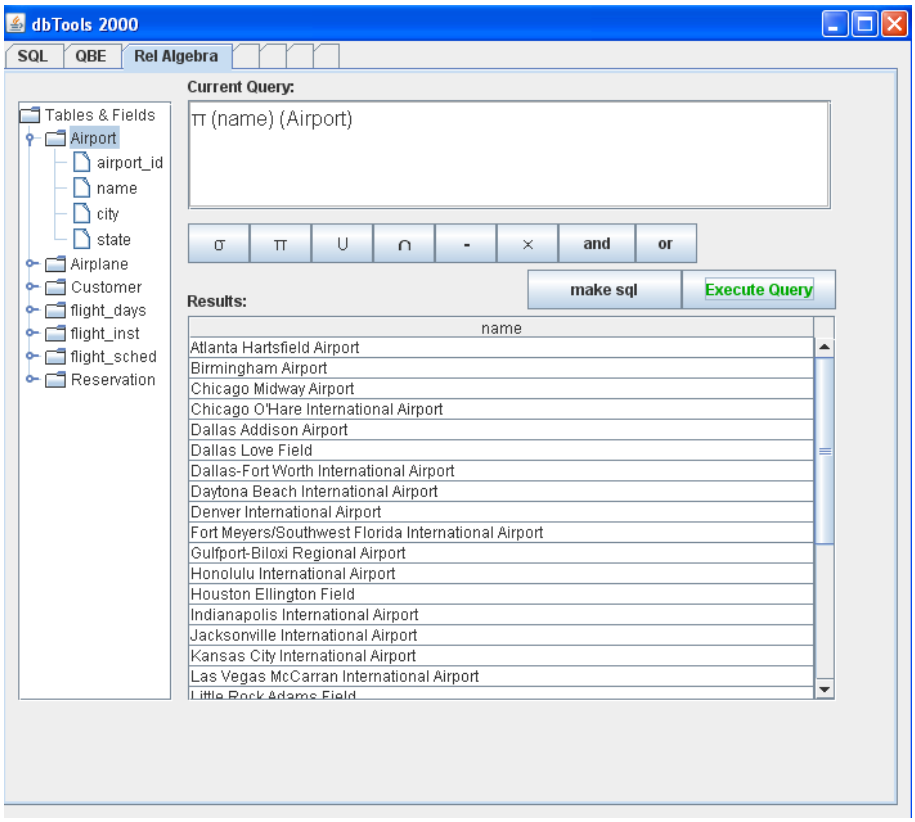

Figura 2.2: Interface de consulta da ferramenta *DBTools 2000*.

quando o usuário tenta executar uma consulta incompleta. Falta ainda um menu com ajuda ao usuário. A documentação existente pode ser acessada apenas através do site. Foi relatado que o programa é instável, porém este fato não foi observado.

3 A FERRAMENTA *SIMALG*

Este capítulo apresenta uma proposta para criação de uma ferramenta simuladora de consultas em Álgebra Relacional, visando suprir as necessidades das ferramentas analisadas na seção 2.3.

3.1 Visão Geral

A ferramenta *SimAlg* tem como finalidade a simulação de consultas em álgebra relacional. As operações suportadas são seleção, projeção, união, intersecção, diferença, produto cartesiano e junção natural. Ao contrário do *software LEAP*, onde a interface é somente textual, o *SimAlg* possui uma interface gráfica para interação com o usuário e utiliza a mesma simbologia e sintaxe adotadas pela maioria dos livros. A fim de melhorar os pontos negativos do *software DBTools 2000*, foi utilizado um banco de dados que não requer instalação nem configuração, facilitando a utilização da ferramenta. No tratamento das exceções, mensagens de erro são exibidas para informar ao usuário o problema ocorrido. Além destas, o menu de ajuda auxilia em caso de dúvidas quanto à utilização do programa e à álgebra relacional.

A arquitetura proposta está representada na figura 3.1. Primeiramente o usuário compõe a consulta em álgebra relacional na interface do *SimAlg*. Ao executar a consulta, o *software* a converte de álgebra relacional para SQL através do mapeamento explicado na seção 3.2. O comando em SQL é enviado para o banco de dados processar a consulta. Após o processamento, os resultados são mostrados ao usuário na interface.

3.2 Mapeamento de Álgebra Relacional para SQL

A ferramenta se baseia no mapeamento de uma consulta em álgebra relacional (AR) para sua correspondente em SQL. Desta forma, o usuário escreve a consulta em álgebra

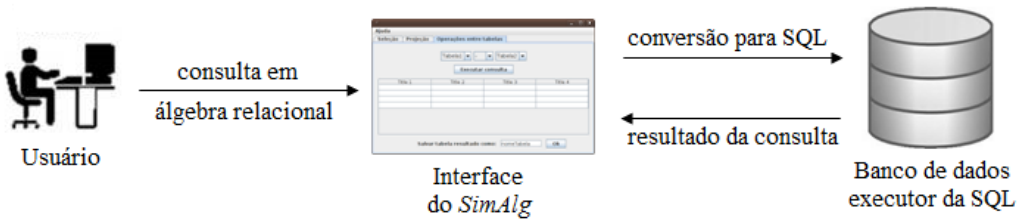

Figura 3.1: Fluxo de funcionamento.

relacional, a ferramenta converte esta consulta para SQL, envia o comando SQL para o banco de dados processar a consulta e os resultados são mostrados na tabela presente na interface. Os mapeamentos utilizados nas operações suportadas pela ferramenta podem ser vistos a seguir, primeiro de forma genérica e depois aplicando para os exemplos dados anteriormente.

1. A operação de *seleção* da álgebra relacional é mapeada para a cláusula *where* da SQL:

   - AR: $\sigma_{coluna=1}(tabela)$
   - SQL: SELECT * FROM tabela WHERE coluna = 1
   - AR: $\sigma_{dataIngresso>=2009-01-01}(BolsistaPET)$
   - SQL: SELECT * FROM BolsistaPET WHERE dataIngresso >= 2009-01-01

2. A operação de *projeção* da álgebra relacional é mapeada para a cláusula *select* da SQL:

   - AR: $\pi_{coluna}(tabela)$
   - SQL: SELECT coluna FROM tabela
   - AR: $\pi_{titulo}(\sigma_{coordenador="Luiz"}(Projeto))$
   - SQL: SELECT titulo FROM Projeto WHERE coordenador = “Luiz”

3. A operação de *união* da álgebra relacional é mapeada para a cláusula *union* da SQL:

   - AR: $tabela1 \cup tabela2$
   - SQL: SELECT * FROM tabela1 UNION SELECT * FROM tabela2
   - AR: $\pi_{matricula}(Participantes) \cup \pi_{matricula}(BolsistaPET)$

- SQL: SELECT matricula FROM Participantes UNION SELECT matricula FROM BolsistaPET

4. A operação de *diferença* da álgebra relacional é mapeada para a cláusula *minus* da SQL:

   - AR: $tabela1 - tabela2$
   - SQL: SELECT * FROM tabela1 MINUS SELECT * FROM tabela2
   - AR: $\pi_{matricula}(Aluno) - \pi_{matricula}(BolsistaPET)$
   - SQL: SELECT matricula FROM Aluno MINUS SELECT matricula FROM BolsistaPET

5. A operação de *produto cartesiano* da álgebra relacional é mapeada para a cláusula *cross join* da SQL:

   - AR: $tabela1 \times tabela2$
   - SQL: SELECT * FROM tabela1 A CROSS JOIN tabela2 B
   - AR: $\pi_{matricula,nome}(Aluno) \times \pi_{matricula,dataIngresso}(BolsistaPET)$
   - SQL: SELECT A.matricula, A.nome, B.matricula, B.dataIngresso FROM Aluno A CROSS JOIN BolsistaPET B

6. A operação de *junção natural* da álgebra relacional é mapeada para a cláusula *natural join* da SQL:

   - AR: $tabela1 \bowtie tabela2$
   - SQL: SELECT * FROM tabela1 NATURAL JOIN tabela2
   - AR: $\pi_{nome}(Aluno \bowtie BolsistaPET)$
   - SQL: SELECT nome FROM Aluno NATURAL JOIN BolsistaPET

7. A operação de *intersecção* da álgebra relacional é mapeada para a cláusula *intersect* da SQL:

   - AR: $tabela1 \cap tabela2$
   - SQL: SELECT * FROM tabela1 INTERSECT SELECT * FROM tabela2

- AR: $\pi_{matricula}(Participantes) \cap \pi_{matricula}(BolsistaPET)$
- SQL: SELECT matricula FROM Participantes INTERSECT SELECT matricula FROM BolsistaPET

4 DESENVOLVIMENTO DA APLICAÇÃO

Este capítulo explica a utilização da ferramenta *SimAlg* e detalhes relevantes da implementação. Ao final, é feita uma análise do *software* desenvolvido.

4.1 Utilização do Programa

A aplicação desenvolvida permite a execução de sete operações da álgebra relacional, além da visualização do conteúdo existente no banco de dados. A fim de facilitar a utilização, o programa vem com as tabelas e os dados citados na seção 2.2.2.

4.1.1 Execução de Consultas

As três primeiras abas da interface gráfica são destinadas à realização de operações da álgebra relacional, sendo a primeira para operação de seleção, a segunda para operação de projeção e a terceira para operações binárias. As seções a seguir exemplificam a utilização da ferramenta para as consultas apresentadas na seção 2.2.2.

4.1.1.1 Seleção

Na interface de seleção, o usuário tem possibilidade de executar uma consulta com até duas condições. Primeiro deve-se selecionar a tabela que deseja realizar a consulta através do menu *dropdown* mais à direita. Selecionada a tabela, suas colunas aparecem como opções nos menus *dropdown* das condições. Se a coluna selecionada for do tipo *varchar*, apenas o operador de igualdade é exibido no menu *dropdown* da operação. Em caso contrário, as opções são igualdade, maior, menor, maior ou igual e menor ou igual, pois a coluna será um número ou uma data. Escolhido o operador, basta digitar o valor para comparação na caixa de texto. Caso o usuário queira escolher uma segunda condição, seleciona o operador de conjunção ou disjunção (*and* ou *or*), o menu *dropdown* de tabela e o segundo campo do valor de comparação. Após selecionar a(s) condição(ões)

desejada(s), o usuário deve clicar no botão “Executar operação” para conferir o resultado na tabela abaixo. O SQL equivalente aparecerá abaixo do botão. O sistema permite salvar a tabela resultante da consulta. Através deste recurso, é possível executar consultas em que o resultado de uma operação é a entrada de outra.

A figura 4.1 mostra a consulta exemplificada na seção 2.2.2.1, que seleciona os alunos que ingressaram no grupo PET após o início do ano de 2009.

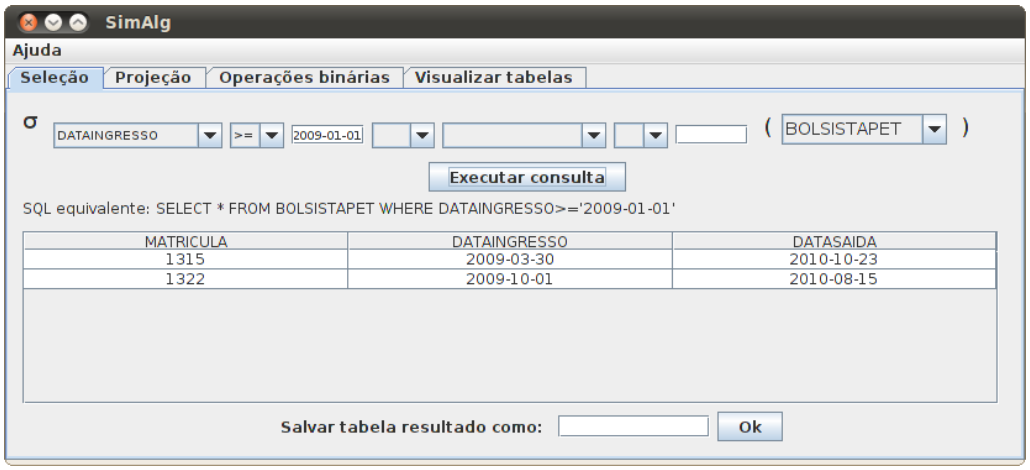

Figura 4.1: Seleção dos alunos que ingressaram no grupo PET após o início do ano de 2009.

4.1.1.2 Projeção

A segunda aba é destinada a operações de projeção, onde o usuário pode fazer uma consulta que projete até três colunas. Semelhante à interface de seleção, primeiro deve-se selecionar a tabela que deseja-se realizar a consulta através do menu *dropdown* mais à direita. Selecionada a tabela, suas colunas aparecem como opções nos três menus *dropdown* de coluna. A execução da consulta, visualização do SQL equivalente e opção de salvar a tabela resultante são iguais para as três abas.

O exemplo apresentado na seção 2.2.2.2, onde é projetado o título dos projetos coordenados pelo professor Luiz, deve ser realizada em duas etapas. Primeiramente o usuário realiza uma operação de seleção sobre a tabela *Projeto* para obter os projetos do professor Luiz e salva a tabela resultante, como destacado na figura 4.2. Após, projeta a coluna *titulo* nesta tabela salva, conforme figura 4.3.

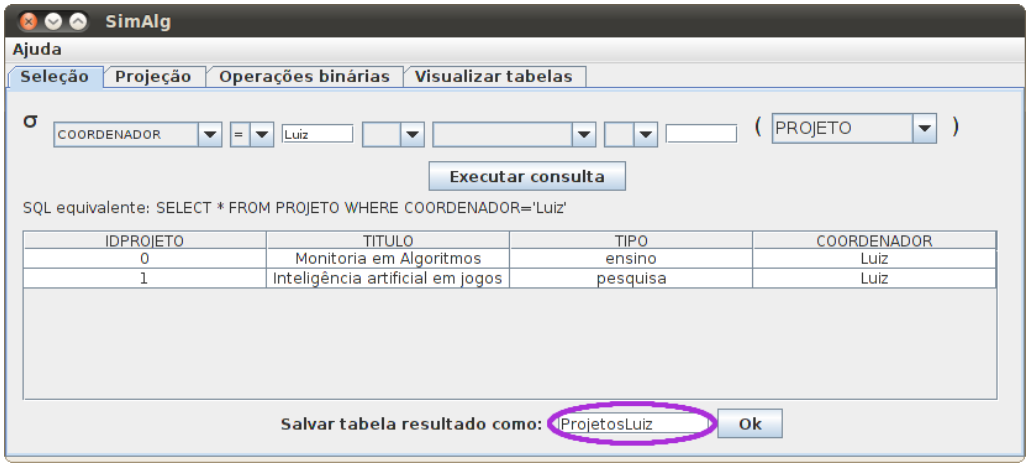

Figura 4.2: Seleção dos projetos coordenados pelo professor Luiz.

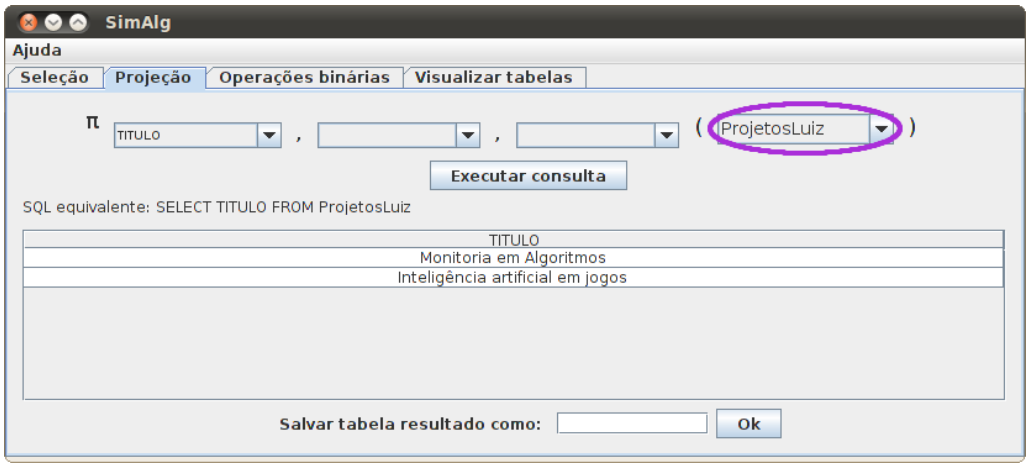

Figura 4.3: Projeção do atributo *titulo* na tabela resultante da operação de seleção.

4.1.1.3 Operações Binárias

A terceira aba possui interface para as operações binárias de união, intersecção, diferença, produto cartesiano e junção natural. O usuário seleciona livremente duas tabelas nos menus *dropdown* e a operação entre elas. Caso a operação exija compatibilidade entre tabelas, como união, e elas não forem compatíveis, o usuário é avisado após tentar executar a consulta. Caso contrário, o resultado aparece na tabela abaixo. As seções a seguir mostram a execução dos exemplos apresentados na seção 2.2.2.

4.1.1.3.1 União

O exemplo de união mostrado na seção 2.2.2.4, que retorna a matrícula dos alunos participantes de projetos ou do grupo PET, deve ser realizada em três etapas. Primeiro executa-se uma projeção no atributo *matricula* da tabela *Participantes* e salva-se a tabela

resultante da operação (figura 4.4). Após, projeta-se o atributo *matricula* da tabela *BolsistaPET* e salva-se o resultado (figura 4.5). A operação de união é executada com as tabelas salvas nas duas primeiras etapas, conforme a figura 4.6.

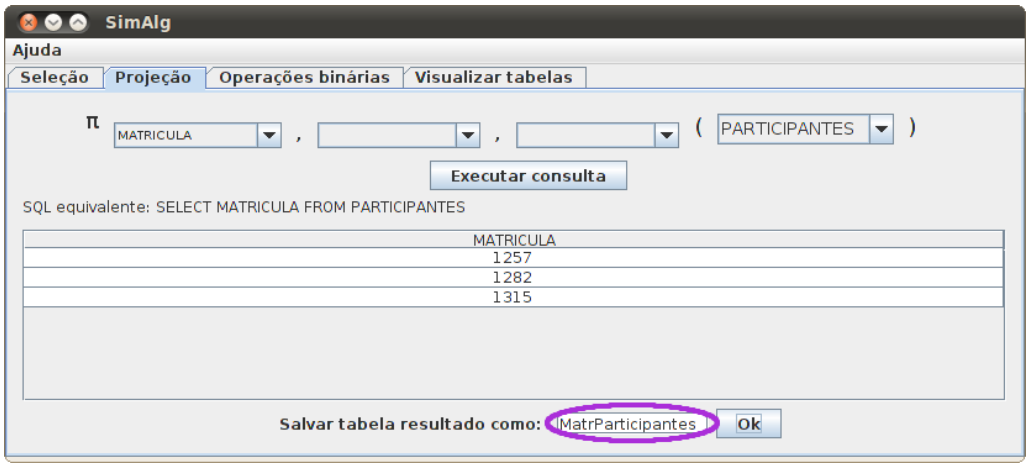

Figura 4.4: Projeção sobre o atributo *matricula* da tabela *Participantes*.

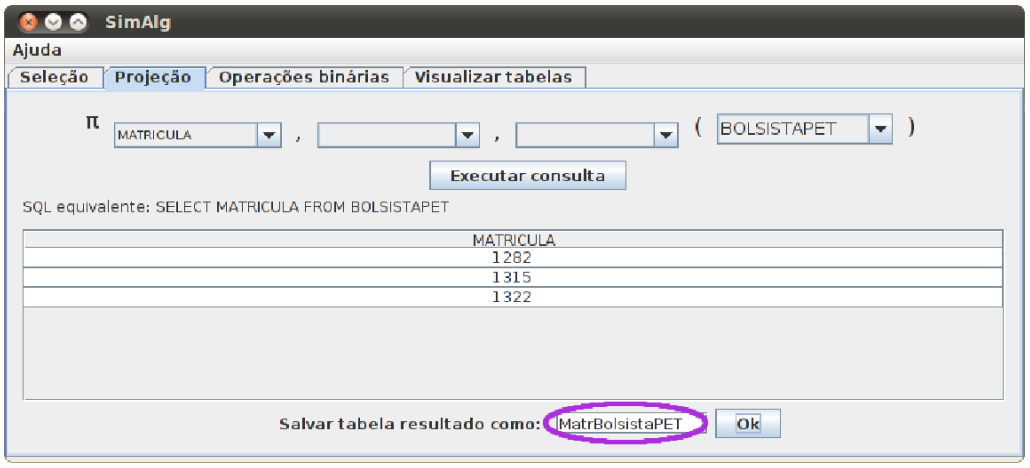

Figura 4.5: Projeção sobre o atributo *matricula* da tabela *BolsistaPET*.

4.1.1.3.2 Diferença

A operação de diferença exemplificada na seção 2.2.2.5, que retorna os alunos que nunca participaram do grupo PET, é executada em três etapas, como o exemplo dado para a operação de união. Primeiro deve-se realizar a operação de projeção sobre o atributo *matricula* da tabela *Aluno* e depois, da tabela *BolsistaPET*. As duas tabelas devem ser salvas para a posterior execução da operação de diferença entre elas, conforme a figura 4.7.

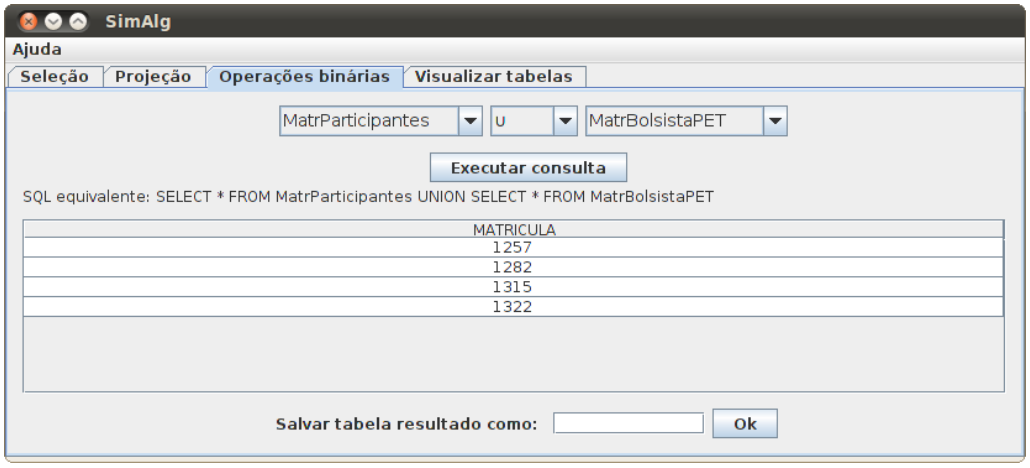

Figura 4.6: União entre as tabelas *MatrParticipantes* e *MatrBolsistaPET*.

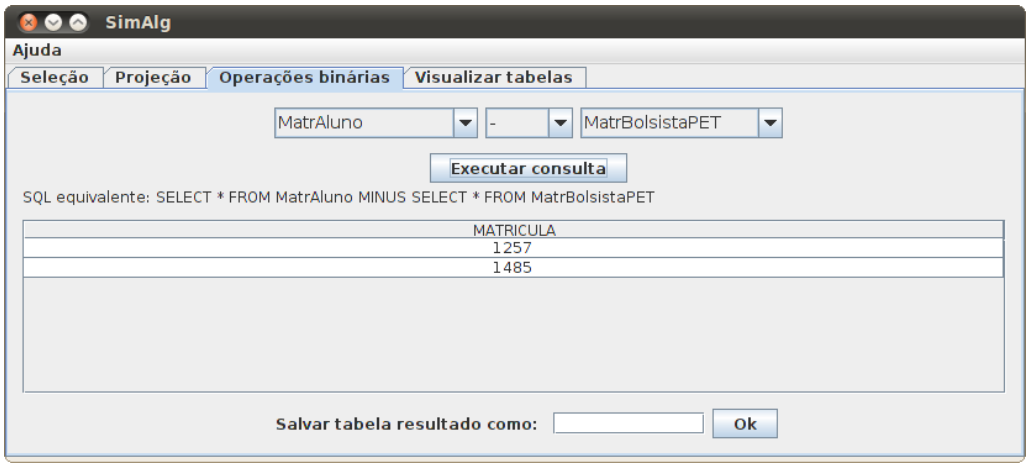

Figura 4.7: Diferença entre as tabelas *MatrAluno* e *MatrBolsistaPET*.

4.1.1.3.3 Produto cartesiano

Em oposição às operações binárias citadas anteriormente, o produto cartesiano não exige compatibilidade entre as tabelas envolvidas. Porém, a fim de obter uma tabela resultado mais compacta, realizou-se a operação de projeção sobre dois atributos de cada uma das tabelas envolvidas. Primeiro projetam-se os atributos *matricula* e *nome* da tabela *Aluno*. Após salvar a tabela, projetam-se os atributos *matricula* e *dataIngresso* de *BolsistaPET* e o resultado é salvo. Com essas duas tabelas, executa-se a operação de produto cartesiano, como mostra a figura 4.8.

4.1.1.3.4 Junção Natural

A junção natural, que simplifica a resposta do produto cartesiano com operação de seleção, é mostrada na figura 4.9 entre as tabelas *Aluno* e *BolsistaPET*. No exemplo dado

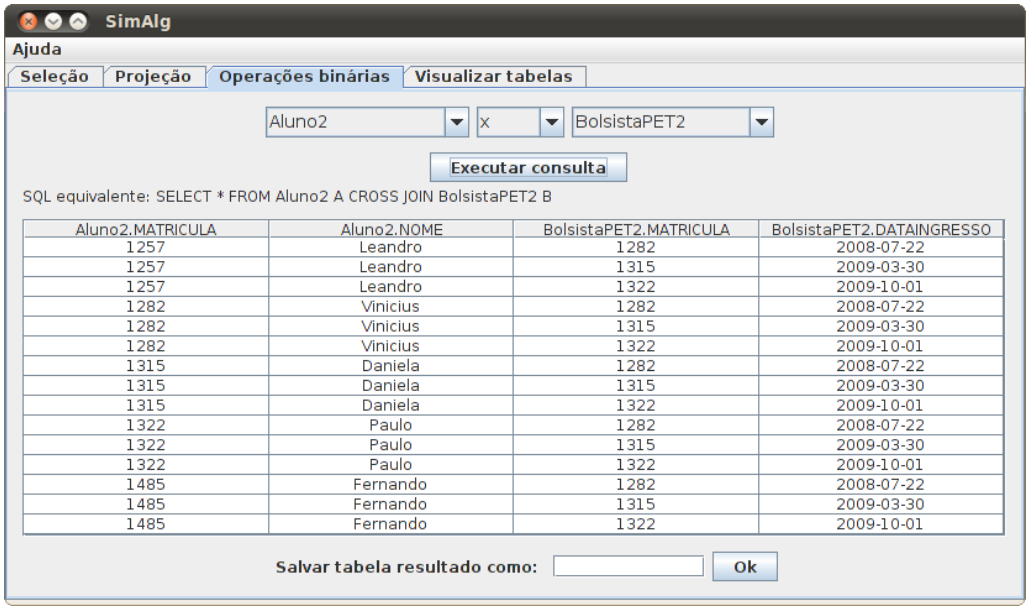

Figura 4.8: Produto cartesiano entre duas tabelas resultantes de projeção.

na seção 2.2.2.7.3, é aplicada ainda uma projeção sobre o atributo *matricula* da tabela resultante. Para isso, deve-se salvar o resultado da junção natural e aplicar a operação de projeção na segunda aba do programa.

Figura 4.9: Junção natural entre as tabelas *Aluno* e *BolsistaPET*.

4.1.1.3.5 Intersecção

O exemplo da operação de intersecção visto na seção 2.2.2.8 é mostrado na figura 4.10. Antes de realizar a intersecção, deve-se salvar as projeções sobre os atributos *matricula* das tabelas *Participantes* e *BolsistaPET*, como visto anteriormente nas figuras 4.4 e 4.5.

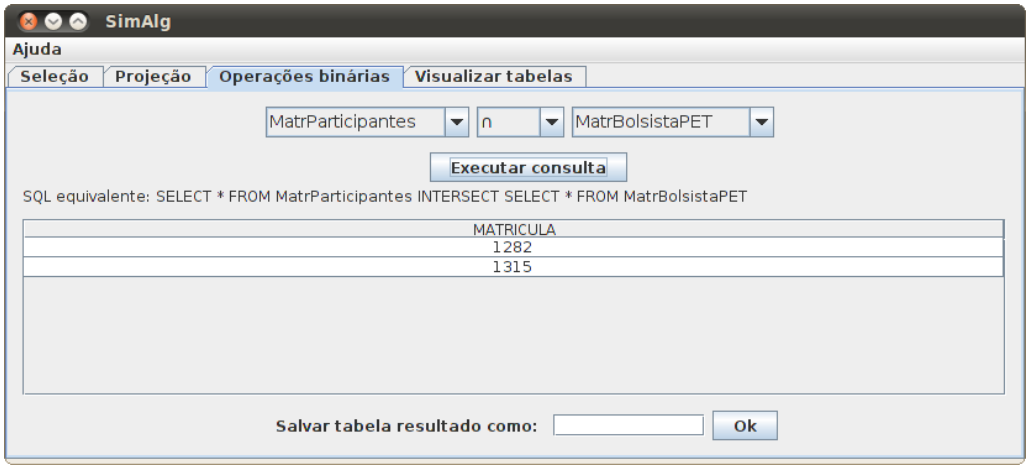

Figura 4.10: Intersecção entre as tabelas *MatrParticipantes* e *MatrBolsistaPET*.

4.1.2 Visualização de Tabelas

A fim de facilitar a construção das consultas, a quarta aba do programa permite ao usuário saber quais tabelas e dados estão presentes no banco de dados. Ao selecionar no menu *dropdown* a tabela que deseja-se visualizar, seu conteúdo é exibido na tabela abaixo. Tabelas criadas durante a execução do programa também podem ser visualizadas nesta aba. A figura 4.11 mostra a tabela *Aluno*.

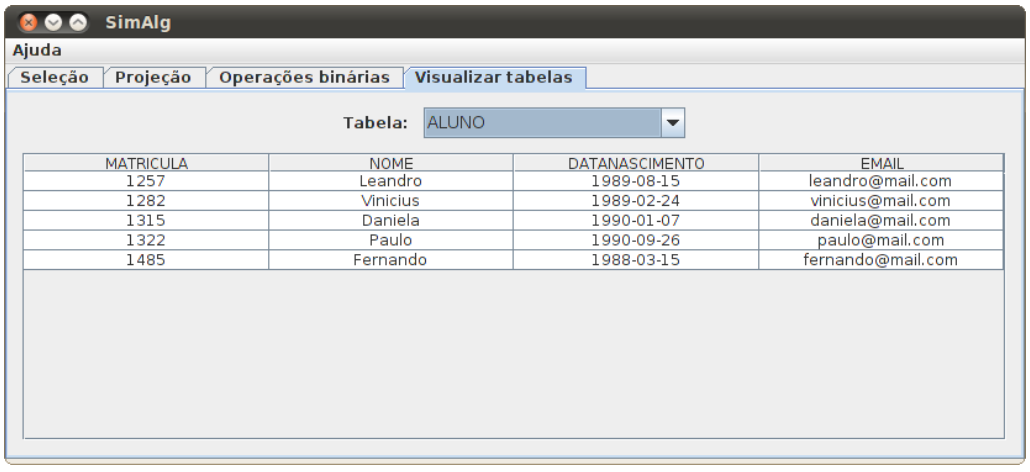

Figura 4.11: Visualização da tabela *Aluno*.

4.1.3 Menu de Ajuda

A interface possui um menu de ajuda com três itens. O primeiro auxilia na utilização da ferramenta, conforme figura 4.12. Nesta, explica-se como realizar operações de seleção no *SimAlg*. O segundo item do menu de ajuda contém uma breve descrição das operações de álgebra relacional suportadas. Na figura 4.13 pode-se observar a explicação

sobre a operação de seleção. Por fim, o terceiro item mostra o nome da desenvolvedora e o *link* onde podem ser encontradas mais informações sobre o programa.

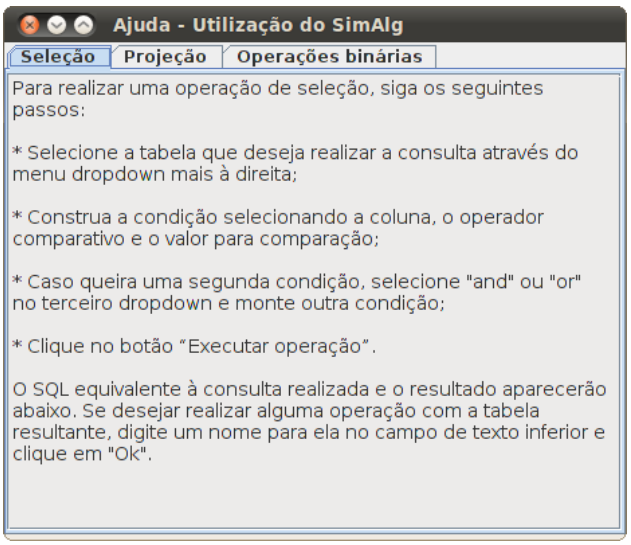

Figura 4.12: Janela de ajuda na utilização do *SimAlg*.

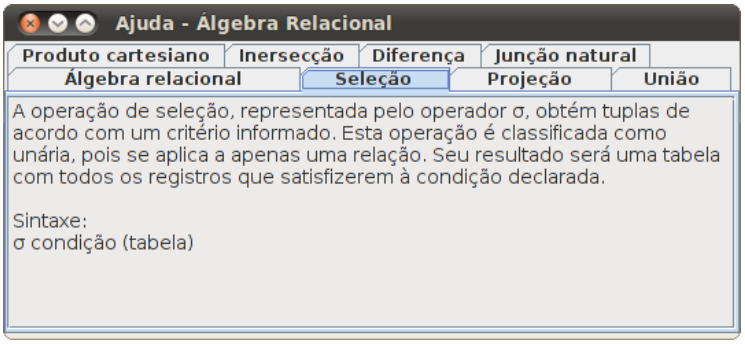

Figura 4.13: Janela de ajuda com explicação das operações de álgebra relacional suportadas pela ferramenta.

4.1.4 Tratamento de Erros

Nos testes realizados com o programa, foram identificados os erros que o usuário pode cometer. O primeiro é nas condições da operação de seleção. Se ele tentar executar sem preencher a condição, a consulta simplesmente não será executada. Se a segunda condição estiver parcialmente preenchida, será desconsiderada. No caso de condições com colunas dos tipos *varchar* e *date*, aspas simples são acrescentadas no início e no final do valor a ser comparado, caso o usuário não coloque. Se o formato da data for diferente do esperado (aaaa-mm-dd), o usuário é informado do formato correto. Outro erro possível é na execução de operações binárias que exigem compatibilidade entre as tabelas. Por fim,

o último caso de erro observado foi na opção de salvar uma tabela, pois o usuário pode digitar um nome inválido (contendo caracteres que não forem alfa-numéricos ou nome já utilizado para outra tabela). Em todos esses casos, mensagens de erro em janelas *pop-up* alertam o usuário sobre o problema e sobre como proceder.

4.2 Implementação

A ferramenta foi desenvolvida em linguagem Java no ambiente de desenvolvimento integrado (IDE) NetBeans 6.8 (NETBEANS, 2010) devido à sua facilidade para composição de interface gráfica. Para gerenciar o banco de dados, foi utilizado o SGBD HSQLDB (HSQLDB, 2010) por não precisar de instalação no computador em que o programa será executado, tornando o uso da ferramenta mais prático.

O diagrama de classes apresentado na figura 4.14 mostra como o programa está organizado. Os atributos das classes foram omitidos por motivos de simplificação. As nove classes do programa e os principais aspectos da implementação são detalhadas nas seções a seguir.

4.2.1 Classe *Gui*

Essa classe contém a *main* e todos os componentes visuais do programa.

4.2.1.1 Execução de Consultas

Quando um dos botões de execução de consulta é pressionado, a composição da consulta em álgebra relacional é enviada para um método da classe *GuiDAO*. Esse método pode ser *executeSelectOperation*, *executeProjectOperation* ou *executeBinaryOperation*, dependendo da aba que o usuário está. Como resposta, recebe o resultado da consulta e o exibe na tabela da interface gráfica. O SQL equivalente também é obtido através de um método da classe *GuiDAO*.

4.2.1.2 Criação de Tabelas

Na classe *Gui*, a funcionalidade de salvar a tabela resultante das operações consiste apenas na invocação do método *createTable* da classe *GuiDAO*. Seu retorno indica o sucesso ou não da operação de criação da tabela. No caso de sucesso, os menus *dropdown* da interface gráfica são atualizados.

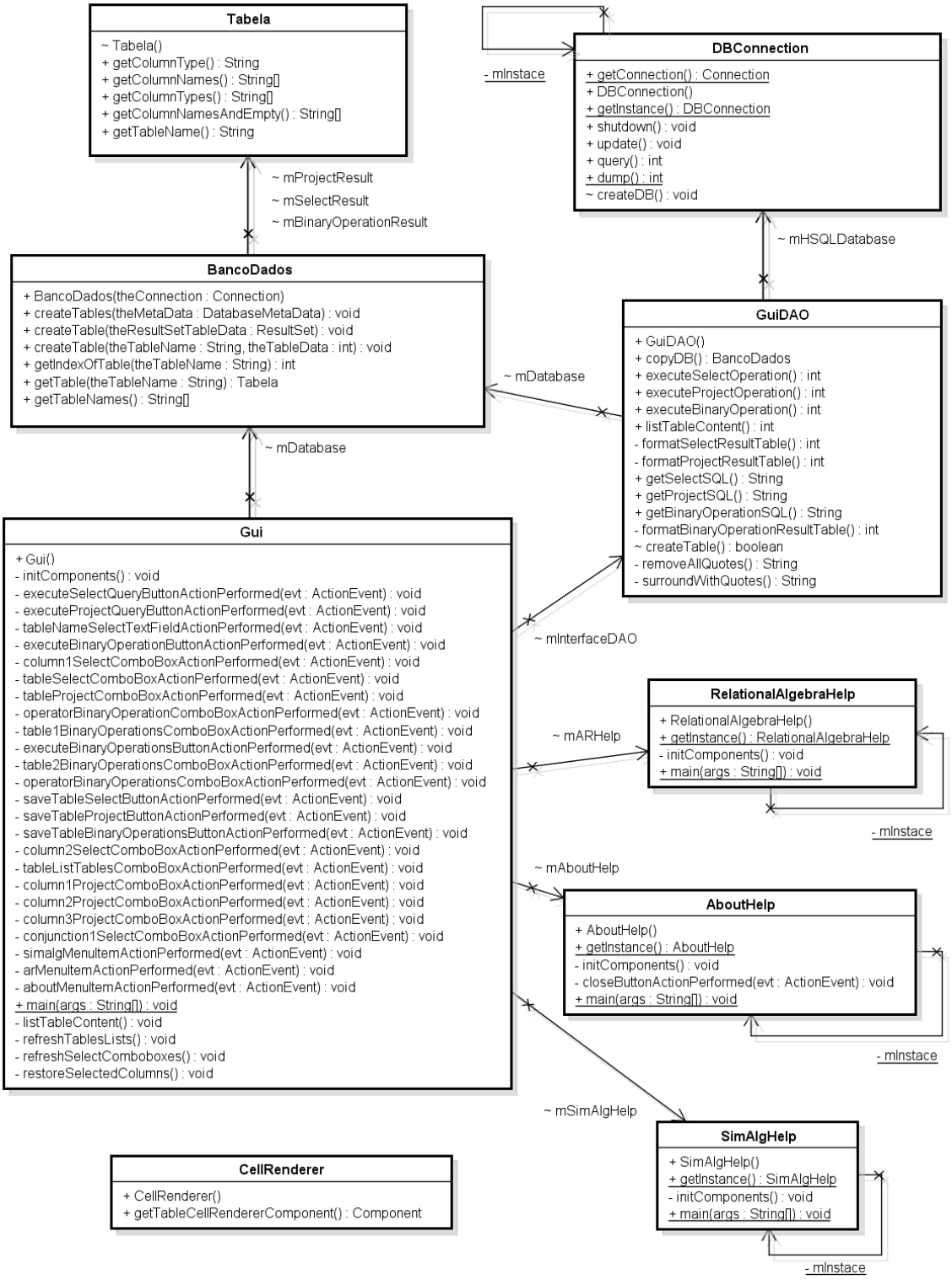

Figura 4.14: Diagrama de classes do *SimAlg*.

4.2.1.3 Visualização do Conteúdo Presente no Banco de Dados

A tabela com o conteúdo de todas as relações do banco de dados, presente na quarta aba da interface gráfica, é construída com informação proveniente da invocação do mé-

todo *listTableContent*, implementado na classe *GuiDAO*.

4.2.2 Classe *GuiDAO*

Essa classe é responsável pela criação dos códigos em SQL e invocação dos métodos que os executam. Tem por objetivo a abstração da forma que as consultas são realizadas.

4.2.2.1 Execução de Consultas

Os métodos *executeSelectOperation*, *executeProjectOperation* e *executeBinaryOperation* recebem *Strings* com as tabelas, colunas e valores envolvidos na consulta em questão. Estes métodos realizam o mapeamento da álgebra relacional para SQL, conforme visto na seção 3.2, e executam o SQL através do método *query*, presente na classe *DBConnection*. O resultado da consulta, composto pelos nomes, tipos e valores das colunas, é retornado. O SQL é salvo como variável de classe e pode ser obtido através do método *getSelectSQL*, *getProjectSQL* ou *getBinaryOperationSQL*, dependendo da operação.

4.2.2.2 Criação de Tabelas

O método *createTable* compõe o SQL para criação da nova tabela através do seu nome e seu conteúdo (valores, nomes e tipos de colunas), passados por parâmetro. Esse SQL é executado no banco de dados pelo método *update*, de *DBConnection*.

4.2.2.3 Visualização do Conteúdo Presente no Banco de Dados

Esse recurso é implementado fazendo a operação *SELECT * FROM tabela*, onde *tabela* é uma *String* passada por parâmetro ao método *listTableContent*. O resultado é retornado para exibição na interface gráfica.

4.2.3 Classe *DBConnection*

A classe *DBConnection* cria o banco de dados HSQLDB na memória principal, ou seja, os dados são armazenados apenas durante a execução do programa. Essa abordagem foi utilizada para evitar o acúmulo de tabelas salvas em execuções anteriores. Nessa classe estão as chamadas para os métodos que, de fato, efetuam as consultas e atualizações na base de dados. Parte do código foi reaproveitada de um trabalho realizado na disciplina de Fundamentos de Banco de Dados, no segundo semestre de 2008.

4.2.3.1 Consultas

O método *query* executa o código *createStatement().executeQuery(expression)* a partir da conexão, do tipo *Connection*, onde *expression* é o SQL passado por parâmetro. Dessa operação, um objeto *ResultSet* é retornado e enviado para o método *dump*, que formata o resultado em uma *LinkedList<LinkedList<String>>*. A primeira *LinkedList<String>* possui os nomes das colunas, a segunda possui os tipos e a partir da terceira estão as tuplas do resultado da operação.

4.2.3.2 Atualizações

O método *update* executa o mesmo código que o método *query* para a execução do SQL. A *String expression* pode ser um SQL de criação ou exclusão de tabela, inserção de conteúdo ou atualização de tupla. No caso do *SimAlg*, apenas criações de tabelas são executadas.

4.2.3.3 Criação da Base de Dados

O método *createDB* cria as tabelas e insere os dados que irão compor o banco de dados, conforme apresentado na seção 2.2.2. Como o programa é de código aberto, o usuário pode alterar esse método para criar suas próprias tabelas.

4.2.4 Classe *BancoDados*

A classe *BancoDados* possui uma *LinkedList* de objetos do tipo *Tabela*, que armazena a estrutura de cada uma das tabelas existentes no banco de dados. No início da execução do programa, todas as tabelas existentes são inseridas nessa lista. Após, sempre que uma tabela é criada no banco de dados, é invocado um método da classe *BancoDados* que insere a estrutura da nova tabela na lista. Através desta classe, pode-se obter a estrutura de qualquer tabela pelo método *getTable* sem precisar consultar o banco de dados. O método *getTableNames* é chamado no início do programa para compor os menus *dropdown* da interface gráfica que possuem nomes de tabelas.

4.2.5 Classe *Tabela*

Essa classe possui uma abstração para a estrutura da tabela armazenada no banco de dados. No seu construtor, são inicializadas as variáveis que armazenam o nome da tabela, os tipos e os nomes das colunas. O método *getColumnNames*, que retorna um *array* de

String com os nomes das colunas, é utilizado para composição dos menus *dropdown*. O método *getColumnTypes* retorna um *array* de *String* com os tipos das colunas, utilizado para gerar o SQL de criação de tabela. Possui ainda métodos para retornar o tipo de determinada coluna e o nome da tabela.

4.2.6 Classe *CellRenderer*

A classe *CellRenderer* herda *DefaultTableCellRenderer* e sobrescreve o método *getTableCellRendererComponent*. Esse método centraliza o texto em uma *JTable* e foi utilizado para melhorar a visualização das tabelas.

4.2.7 Classes *SimAlgHelp*, *RelationalAlgebraHelp* e *AboutHelp*

Estas três classes, que herdam *JFrame*, compõem a interface das janelas de ajuda.

4.3 Análise do *SimAlg*

Esta seção apresenta uma análise as funcionalidades implementadas na ferramenta *SimAlg*. Após, faz-se uma comparação com os programas semelhantes previamente analisados.

4.3.1 Execução de Consultas

A construção de consultas possui uma interface com formato propositalmente pouco flexível para cada operação. Essa característica facilita a utilização da ferramenta por estudantes iniciantes, pois mostra qual é a estrutura correta das operações e obriga que consultas encadeadas sejam feitas por partes, permitindo o acompanhamento dos resultados parciais. Porém, para estudantes mais avançados pode constituir um ponto negativo, já que limita a construção de consultas mais complexas. Essa abordagem foi utilizada para simplificar o mapeamento das consultas de álgebra relacional para SQL.

4.3.2 Comparação entre *LEAP*, *DBTools 2000* e *SimAlg*

A tabela 4.1 mostra uma comparação entre as ferramentas analisadas na seção 2.3 e o *SimAlg*. Os tópicos presentes na tabela serão detalhados a seguir.

Por utilizar HSQLDB, a ferramenta *SimAlg* não necessita de nenhuma instalação ou configuração relativa ao banco de dados. Para utilizar o programa, basta ter a máquina virtual Java instalada em qualquer sistema operacional e executar o arquivo *SimAlg.jar*.

Tabela 4.1: Comparação entre ferrramentas que simulam consultas em álgebra relacional.

Característica	LEAP	DBTools 2000	SimAlg
Instalação fácil	×		×
Interface intuitiva		×	×
Liberdade para construção de consultas	×	×	
Sintaxe e simbologia tradicionais		×	×
Visualização do SQL equivalente		×	×
Visualização do conteúdo existente no BD			×
Tratamento de erros	×		×
Estabilidade	×		×
Criação de BD durante a execução	×		
Menu de ajuda dentro do programa	×		×

Sua instalação e execução constituem uma vantagem sobre o *LEAP* e o *DBTools 2000*. A interface gráfica do programa, com espaço para visualização do conteúdo existente no banco de dados, é outra vantagem sobre as ferramentas analisadas, pois permite analisar se o resultado obtido corresponde ao esperado. Assim como o *DBTools 2000*, o *SimAlg* segue a sintaxe e a simbologia utilizadas nos livros, facilitando a associação das consultas executadas no programa com o conteúdo de álgebra relacional aprendido. A fim de auxiliar também na aprendizagem de SQL, o *SimAlg*, assim como o *DBTools 2000*, mostra qual é o SQL equivalente à consulta realizada em álgebra relacional. O tratamento de erros, juntamente com os tópicos de ajuda, constituem outra vantagem, pois guiam o usuário quando necessário ou desejado. No entanto, o *SimAlg* não permite a execução de consultas mais complexas como nas outras ferramentas, nem a criação e troca de banco de dados, possível no *LEAP*.

Os códigos fonte e a aplicação estão disponíveis em `http://www.inf.ufsm.br/~llautert/SimAlg`.

5 CONCLUSÃO E TRABALHOS FUTUROS

A necessidade da implementação de uma nova ferramenta simuladora de consultas em álgebra relacional foi constatada através da análise dos programas existentes que executam tal tarefa. As ferramentas analisadas, *LEAP* e *DBTools*, possuem características que dificultam sua utilização, como interface pouco intuitiva, tratamento inadequado de erros, instabilidade e não utilização da simbologia e sintaxe adotados nos livros tradicionais de banco de dados. Após constatadas estas deficiências, fez-se uma proposta de uma nova ferramenta que suprisse tais necessidades.

Implementou-se, então, o *SimAlg*, tendo como objetivos principais a estabilidade e a facilidade de utilização por parte dos alunos. A segunda característica foi obtida através da interface gráfica com simbologia e sintaxe tradicionais, além de não necessitar instalação nem configuração prévia para executar o programa. Espera-se que estas vantagens sirvam de incentivo para a propagação do uso do *SimAlg* como ferramenta de ensino nos cursos de Banco de Dados.

Após a finalização da implementação, foram realizados vários testes bem sucedidos com as sete operações suportadas. Pela análise dos resultados obtidos nestes testes e comparação entre as funcionalidades dos programas *LEAP*, *DBTools* e *SimAlg*, concluiu-se que a ferramenta desenvolvida atingiu com sucesso seu principal objetivo de suprir as necessidades dos programas disponíveis existentes.

Entretanto, ainda há o que ser feito. Sugere-se, para trabalhos futuros, que seja suportada a criação e alternância de banco de dados para a realização das consultas durante a execução do programa. Atualmente, estas operações apenas podem ser realizadas alterando o código da aplicação. Deste modo, apenas usuários que saibam utilizar linguagem SQL conseguem criar tabelas e inserir dados nestas. Sugere-se a implementação de uma interface que permita tais operações durante a execução do programa.

Outra melhoria a ser realizada diz respeito à construção de consultas encadeadas na interface, sem a necessidade de uso de tabelas temporárias. Com este recurso, usuários avançados poderão realizar testes mais complexos não compreendidos na interface inicial proposta para o *SimAlg*. A fim de continuar com as vantagens da interface pouco flexível atual, uma boa ideia é mantê-la disponível como *modo iniciante* e implementar outra que permita liberdade na construção de consultas como *modo avançado*. Desta forma, usuários com pouco conhecimento em álgebra relacional poderão familiarizar-se com sua sintaxe utilizando o programa no modo iniciante e após, passariam a construir consultas livremente no modo avançado. Implementados estes recursos, o *SimAlg* será uma ferramenta mais completa.

REFERÊNCIAS

CODD, E. F. A Relational Model of Data for Large Shared Data Banks. **Commun. ACM**, Nova Iorque, v.13, n.6, p.377–387, 1970.

DBTOOLS. Disponível em: http://www.cc.gatech.edu/computing/Database/dbTools. Acesso em: agosto de 2010.

HEUSER, C. A. **Projeto de banco de dados**. 3.ed. Porto Alegre: Editora Sagra Luzzatto, 2000.

HSQLDB. Disponível em: http://hsqldb.org. Acesso em: setembro de 2010.

LEYTON, R.; HILL, C. **Leap**. Disponível em: http://leap.sourceforge.net. Acesso em: agosto de 2010.

NETBEANS. Disponível em: http://netbeans.org. Acesso em: setembro de 2010.

REZENDE, R. **Conceitos Fundamentais de Banco de Dados - Parte 2**. Disponível em: http://www.devmedia.com.br/post-1678-Conceitos-Fundamentais-de-Banco-de-Dados-Parte-2.html. Acesso em: outubro de 2010.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de Banco de Dados**. 5.ed. Rio de Janeiro: Editora Campus, 2006.