Departamento de Informática
Curso de Ciência da Computação

FEAR: Uma ferramenta de auxílio ao ensino de Álgebra Relacional usando GALS, Emscripten e WebAssembly

Rodrigo do Nascimento Siqueira

São Luís - MA, 2019

Rodrigo do Nascimento Siqueira

FEAR: Uma ferramenta de auxílio ao ensino de Álgebra Relacional usando GALS, Emscripten e WebAssembly

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, **como parte dos requisitos necessários** para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal do Maranhão

Orientador: Profa. Dra. Simara Vieira da Rocha

São Luís - MA

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Siqueira, Rodrigo do Nascimento.
FEAR: Uma ferramenta de auxílio ao ensino de Álgebra Relacional usando GALS, Emscripten e WebAssembly / Rodrigo do Nascimento Siqueira. – 2019.
53 f.

Orientador(a): Simara Vieira da Rocha.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2019.

1. Álgebra Relacional. 2. Emscripten. 3. GALS. 4. WebAssembly. I. da Rocha, Simara Vieira. II. Título.

Rodrigo do Nascimento Siqueira

FEAR: Uma ferramenta de auxílio ao ensino de Álgebra Relacional usando GALS, Emscripten e WebAssembly

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, **como parte dos requisitos necessários** para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em

Profa. Dra. Simara Vieira da Rocha
Orientador

Prof. Me. Carlos Eduardo Portela Serra de Castro
Examinador

Prof. Dr. Geraldo Braz Junior
Examinador

São Luís - MA
2019

À minha família e aos que contribuíram com essa jornada.

Agradecimentos

Agradeço ao meus pais, Antônio Roque Portela de Araújo, Rosângela de Fátima Medeiros de Araújo e Maria Lúcia Gama do Nascimento, por todo o apoio, paciência e amor dados. E aos meus irmãos Gabriel, Eduardo, Arícia, Ivo e Ricardo, por estarem do meu lado e me apoiarem sempre que possível.

À Rosilda Nogueira e Kamilla Nogueira Soares, por serem uma terceira mãe e uma irmã, respectivamente, e pelos anos de conselhos dados.

Sou muito grato a minha orientadora, professora Simara Vieira da Rocha, por todas as orientações dadas nos últimos meses, sem elas este trabalho não seria possível.

Ao o professor Geraldo Braz Junior, pelos anos de tutoria no PETComp.

Ao PETComp, grupo do qual fiz parte e que possui um grande papel na minha vida acadêmica e pessoal. Graças a ele tive uma nova perspectiva do curso e os anos que passei lá mostraram o caminho profissional que queria seguir.

A todos os amigos que fiz no PET, mais conhecidos como petianos, vocês melhoraram muito minha vida na universidade.

Aos meus amigos e companheiros Anderson Silva, Carolina Lima, Celielma Baldez, Lucas Abreu, Victor Lemos, Marcos Vinícius, Nelia Reis, Paulo Fonseca por toda ajuda, amizade e carinho, e por me aguentarem todos esses anos.

À Eva, pela companhia nas madrugadas em que passava estudando ou fazendo algum trabalho, estando por perto mesmo nos piores momentos.

E agradeço a todas as pessoas que passaram pela minha vida, o que sou hoje é graças a vocês.

"It's never too late to be
what you might have been."
(George Eliot)

"Remember:
Your focus determines your reality."
(Star Wars)

Resumo

As ferramentas de ensino permitem inúmeras possibilidades de tornar a didática mais envolvente e assimilativa. Elas contêm mecanismos que contribuem para captar a atenção do aluno de uma forma a aumentar as chances de um aprendizado de sucesso. Por outro lado, a Álgebra Relacional constitui um conteúdo da disciplina de Banco de Dados e é uma linguagem procedural que consiste em um conjunto de operações que usam relações como entrada e produzem uma nova relação como resultado. Por ter um alto nível de abstração traz dificuldades para o entendimento dos seus conceitos e dos resultados produzidos pelas operações. O objetivo deste trabalho é o desenvolvimento de uma solução *online* voltada para o auxílio no ensino de Álgebra Relacional, onde é possível realizar consultas em Álgebra Relacional, traduzi-las para a linguagem *Structured Query Language* (SQL) e exibir um resultado. No desenvolvimento deste ambiente foi utilizado a GALS que permite a criação de um analisador léxico e sintático, combinada ao compilador Emscripten e à linguagem WebAssembly.

Palavras-chaves: Álgebra Relacional. GALS. Emscripten. WebAssembly.

Abstract

The teaching tools can be multiple possibilities to make more engaging and assimilative didactics. They contain the mechanisms that contribute to the student's attention in order to increase the chances of successful learning. On the other hand, a discussion about the discipline of a database and a procedural language consisting of a set of operations that can serve as a basis and a new relationship as a result. By having a high level of abstraction they bring the difficulty to the understanding of their concepts and their results. The objective of this work is the development of a text solution, intended for the teaching of Algebra, where it is possible to perform queries in algebra, translate it to a structured query language (SQL) and show a result. In the development of this environment we used a GALS that allows the creation of a lexical and syntactic parser in combination with the Sender and the WebAssembly language.

Keywords: Relational Algebra. GALS. Emscripten. WebAssembly.

Lista de ilustrações

Lista de abreviaturas e siglas

σ	Seleção
π	Projeção
ρ	Renomeação
$\cup$	União
$\cap$	Intersecção
-	Diferença
X	Produto Cartesiano
$\bowtie$	Junção
$\overset{\text{T}}{\bowtie}$	Junção Total
$\overset{\text{E}}{\bowtie}$	Junção Esquerda
$\overset{\text{D}}{\bowtie}$	Junção Direita
$\div$	Divisão
$\leftarrow$	Atribuição
IHC	Interação Humano-Computador
LDD	Linguagem de Definição de Dados
LMD	Linguagem de Manipulação de Dados
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language

Sumário

1 Introdução

Em tempos passados, o ensino era um processo estático. Os alunos possuíam como apoio apenas os docentes e os livros, e alguns professores inseriam filmes, músicas ou mídias em geral, para ilustrar aos alunos uma situação do conteúdo ministrado. Atualmente, com a implantação da tecnologia na educação, a busca por informação está menos limitada e possui diversas abordagens diferentes. Com isso, surgiu a necessidade de ferramentas educacionais que venham a complementar o processo de aprendizado. Segundo FIALHO; MATOS (2010) há alguns recursos educacionais que podem ser utilizados como apoio ao trabalho docente enriquecendo sua prática pedagógica e proporcionando momentos de motivação e grande interesse dos alunos, possibilitando reproduções de fenômenos do mundo real e permitindo ao aluno imprimir em seus trabalhos um realismo e qualidade superiores com seu aprendizado, algo difícil de se conseguir nas formas conservadoras de ensino.

Sendo assim, surgiu a necessidade de trazer atividades que possam ser realizadas nas plataformas que mais atraem os alunos, como *smartphones*, *tablets* ou computadores, de forma que o uso desta ferramenta possibilita o processo de ensino e aprendizagem dinâmicas, interativas e contextualizadas com a realidade dos alunos (CHIOFI; OLIVEIRA, 2014). De forma que os alunos continuem seus estudos fora da escola e de forma mais natural, sendo motivados a estudar.

Atualmente, com a facilidade de conexão à internet, muitas ferramentas *online* surgiram com o objetivo de facilitar o uso de determinada funcionalidade pelo usuário. Dessa forma, elas facilitam o alcance de alunos, pois estes podem utilizá-las em casa ou em qualquer outro lugar, de qualquer plataforma em que se possa acessar a internet.

Por outro lado, a Álgebra Relacional, que é uma linguagem procedural, é um conteúdo da disciplina de Banco de Dados. Nela é possível, através de consultas e utilizando operadores, manipular um Banco de Dados, de forma que, dada uma consulta, seja possível obter um resultado sobre um banco de dados. Segundo NAVATHE; ELMASRI (2011) Álgebra Relacional é muito importante por diversos motivos, primeiro, ela oferece um alicerce formal para as operações do modelo relacional, segundo, ela é usada para a implementação e otimização de consultas, terceiro, alguns de seus conceitos são incorporados na linguagem de consulta padrão Structured Query Language (SQL) para Sistema de Gerenciamento de Banco de Dados (SGBD), e, dada sua importância, é essencial o ensino da mesma para o aluno de Ciência da Computação.

Por ser uma linguagem de baixo nível há muita dificuldade em entender o conteúdo, porque não tem como visualizar o que o comando envolvido faz e nem o resultado do

mesmo. Segundo SEIBERT et al. (2016) a dificuldade muito se deve ao fato de o aluno não conseguir visualizar a aplicação de determinados conceitos em situações da vida real, além do mesmo possuir grande dificuldade em interpretar os exercícios propostos. Dessa forma, a construção de uma ferramenta que faça a tradução de Álgebra Relacional para um SQL equivalente - que é uma linguagem que trabalha com a execução de comandos em Bancos de Dados Relacionais - e mostre o resultado é necessária.

Poucas ferramentas implementam a Álgebra Relacional, o que dificulta o seu ensino (PAES, 2007). E devido a esta falta, há a necessidade de um ambiente que permita ao aluno a visualização de como funciona e do que gera uma consulta em Álgebra Relacional, facilitando o entendimento de alunos sobre a matéria e, não obstante, ajudando-os a vivenciar uma prática real além da sala aula, pois, conforme SANTOS (2010), ao interagir com o computador, observando a aplicação prática da Álgebra Relacional, os usuários deixam de levar consigo apenas o conhecimento teórico após o termino da disciplina.

1.1 Objetivos

1.1.1 Geral

Construir uma ferramenta *online* para auxiliar o ensino de Álgebra Relacional.

1.1.2 Específicos

Mais especificamente, pretende-se:

- Realizar um levantamento de ferramentas existentes na literatura para o ensino de Álgebra Relacional;
- Realizar o levantamento de geradores de análise léxica e sintática existentes na literatura;
- Identificar compiladores que possam dar suporte à ferramenta;
- Definir a arquitetura da ferramenta;
- Comparar a ferramenta construída com as soluções existentes na literatura.

1.2 Trabalhos Relacionados

Há algumas soluções que buscam, entre outras coisas, auxiliar o ensino de Álgebra Relacional. Mostrando, a partir de uma consulta, como funciona a Álgebra Relacional e apresentando um resultado respectivo à consulta. As ferramentas foram encontradas através do levantamento bibliográfico e busca *online*.

1.2.1 ProgramAR

O ProgramAR é um ambiente cujo principal objetivo é apoiar o ensino da Álgebra Relacional e do Cálculo Relacional, facilitando o aprendizado do aluno na manipulação das consultas relacionais. A primeira versão do ProgramAR foi desenvolvida pelo aluno de graduação Eric Simão na Universidade Federal da Bahia. Através da ferramenta ProgramAR, o discente pode escrever as expressões em Álgebra Relacional, em Cálculo Relacional ou em SQL e verificar o resultado destas consultas.

A ferramenta funciona executando uma expressão em Álgebra Relacional, seguindo as seguintes etapas:

1. Realiza uma análise léxica e uma análise sintática na expressão para identificar possíveis erros;
2. Caso algum erro seja identificado, é informado ao aluno o tipo de erro e a posição do erro na expressão;
3. Caso não tenham sido encontrados erros na análise sintática, a ferramenta faz a conversão da expressão em Álgebra Relacional para SQL;
4. O SQL, gerado na etapa anterior, é executado no banco de dados e o resultado da consulta é enviado ao ProgramAR;

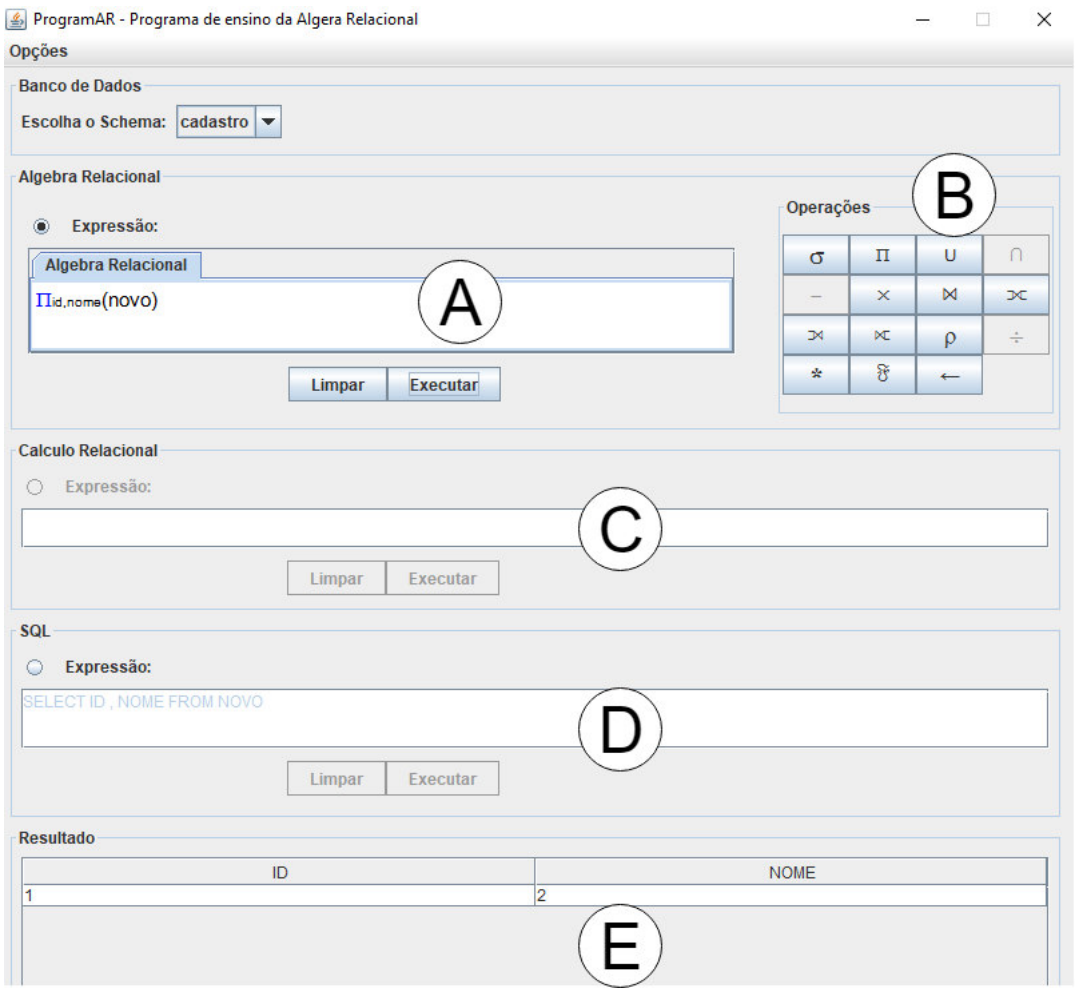
A análise léxica e sintática possui dois módulos, um deles é ativado enquanto o usuário digita, para que em cada caractere digitado haja uma análise do que foi escrito até aquele momento. Já o seu outro módulo é ativado quando é realizado a consulta a um SGBD utilizando uma expressão em Álgebra Relacional. Este módulo recebe como entrada a expressão digitada pelo discente e faz a verificação da sintaxe da expressão (PRATES et al., 2013).

A ferramenta ProgramAR tem como principal vantagem a possibilidade de realizar uma consulta tanto em Álgebra Relacional quanto em Cálculo Relacional e em SQL. Se o usuário faz uma consulta em Álgebra Relacional, como resultado ele recebe um SQL e um resultado equivalente à expressão; se o usuário faz uma consulta em Cálculo Relacional, como resultado ele recebe um SQL e um resultado equivalente à expressão; se o usuário faz uma consulta em SQL, recebe um resultado equivalente à expressão. Sua outra vantagem é que o analisador léxico e sintático analisa cada novo caractere digitado.

Como principal desvantagem há o fato de que três operações não funcionam: Divisão, Intersecção e Diferença. Sendo estes três operadores importantes na Álgebra Relacional. Outra desvantagem é não ser *online*, o usuário só poderá utilizá-la de uma máquina com ela já instalada, o que pode limitar o seu uso. Não é possível ao usuário criar um modelo de dados, de maneira que este terá que possuir um antes de utilizar a ferramenta

A Figura 1 mostra a tela inicial. Conforme percebemos: em A está localizado o espaço em que o usuário pode digitar a expressão em Álgebra Relacional, em D ele visualiza esta mesma expressão em SQL e em E o resultado; em B estão os operadores relacionais que podem ser utilizados em A; no caso de Cálculo Relacional, em C o usuário pode digitar a expressão, e da mesma forma que na Álgebra Relacional, visualizar o SQL em D, e o resultado, em E; no caso de SQL, o usuário pode escrever a expressão em D e ver o resultado em E.

Figura 1 – Tela inicial do ProgramAR.



Fonte: Elaborada pelo Autor

1.2.2 RelaX

A RelaX é uma ferramenta *online* que tem como objetivo o auxílio à aprendizagem de Álgebra Relacional e SQL, de modo que o aluno possa realizar consultas de forma mais prática. Nela, é permitida a execução de Álgebra Relacional, além de ter seus operadores. Também é possível fazer consultas em SQL. A consulta em Álgebra Relacional não é traduzida para SQL e nem de SQL para Álgebra Relacional.

Na execução, é colocada em foco a transparência, de modo que os usuários possam

reconstruir o resultado final, passo-a-passo. Como já foi comentado anteriormente, as instruções não são transformadas em SQL, mas sim interpretadas e executadas diretamente na Álgebra Relacional. Para isto, uma instrução é transformada e visualizada como uma árvore de operadores interativa (KESSLER; TSCHUGGNALL; SPECHT, 2019). Assim, os usuários podem clicar em cada nó da árvore para não apenas inspecionar o resultado intermediário até esse nó, mas também para verificar o correspondente esquema.

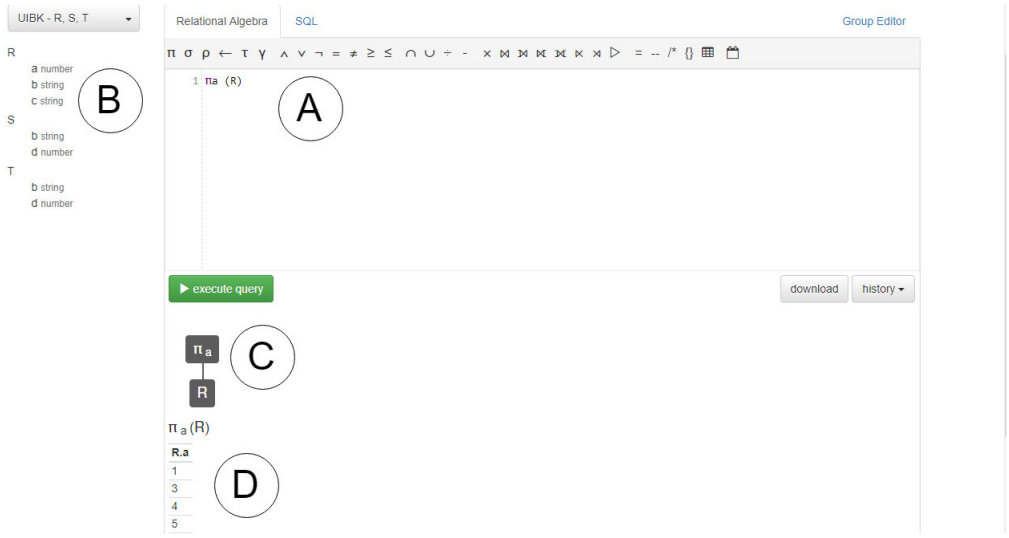
Em relação à análise léxica e sintática no RelaX, assim como no ProgramAR, ocorre em dois momentos, um deles é ativado enquanto o usuário está digitando, assim, em cada caractere digitado ocorre uma análise de tudo que foi escrito até aquele momento. Já o outro, ocorre quando a consulta é realizada. A análise léxica e sintática ocorre tanto na consulta em Álgebra Relacional quanto na consulta em SQL.

Possui como principal vantagem, ser um ambiente *online*. Outra vantagem é o analisador léxico e sintático, que analisa cada novo caractere digitado. Além de possuir todas as operações de Álgebra Relacional disponíveis ao usuário, ele pode construir um modelo de dados, utilizar um modelo fornecido pela própria ferramenta ou um modelo do usuário.

Como desvantagem, há o fato da ferramenta ser pouco intuitiva. Em um primeiro momento, o usuário pode ter um pouco de dificuldade em utilizá-la. Além de que não ocorre uma transformação de Álgebra Relacional para SQL.

A Figura 2 mostra a tela para consulta em Álgebra Relacional. Conforme percebemos: em A está localizado o espaço onde o usuário pode digitar a expressão em Álgebra Relacional, logo acima estão os operadores relacionais, em C é mostrada a árvore de operadores interativa gerada e, em D é mostrado o resultado; em B estão localizadas as informações do modelo escolhido pelo usuário, como os nomes das tabelas e as suas respectivas colunas.

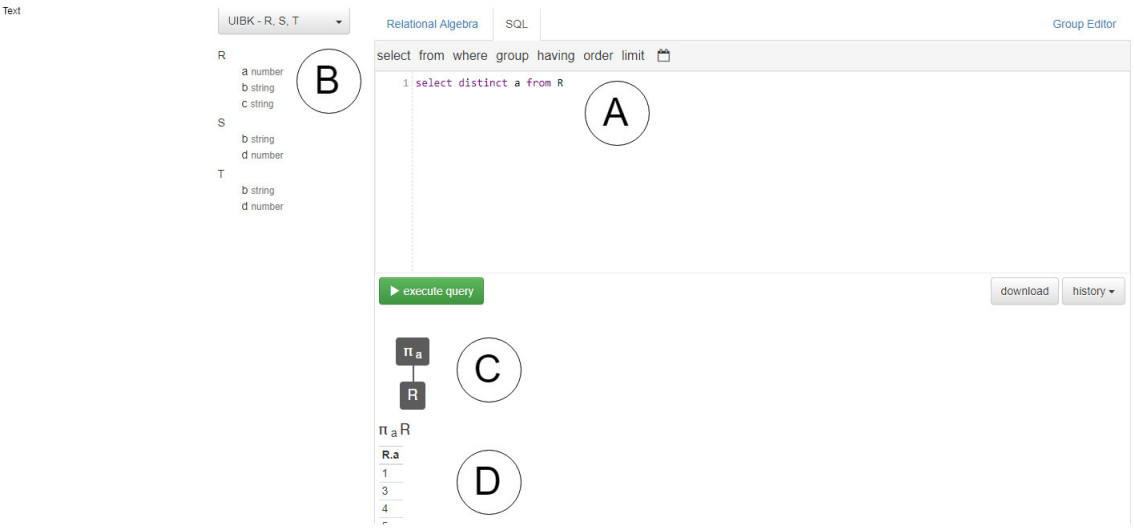
Figura 2 – Tela de consulta de Álgebra Relacional RelaX.



Fonte: Elaborada pelo Autor

A Figura 3 mostra a tela para consulta em SQL. Conforme percebemos: em A está localizado o espaço onde o usuário pode digitar a expressão em SQL, logo acima estão seus principais comandos, em C é mostrada a árvore de operadores interativa gerada e, em D é mostrado o resultado; em B estão localizadas as informações do modelo escolhido pelo usuário, como os nomes das tabelas e suas respectivas colunas.

Figura 3 – Tela de consulta de SQL RelaX.



Fonte: Elaborada pelo Autor

1.2.3 RAT - Relational Álgebra Translator

O RAT é um ambiente que traduz Álgebra Relacional para SQL, cujo objetivo é o auxilio à aprendizagem de Álgebra Relacional, de modo que o aluno possa realizar consultas de forma mais prática. Com ele, é possível escrever uma expressão em Álgebra

Relacional e como resultado receber a expressão transformada em SQL e uma árvore interativa.

Na execução, é colocada em foco a transparência, as instruções são transformadas em SQL e criada uma árvore de operadores interativa, que deve ser capaz de representar graficamente as consultas para facilitar a visualização da solução e construção das consultas (MURILLO; CHAVARRíA; RIVERA, 2012).

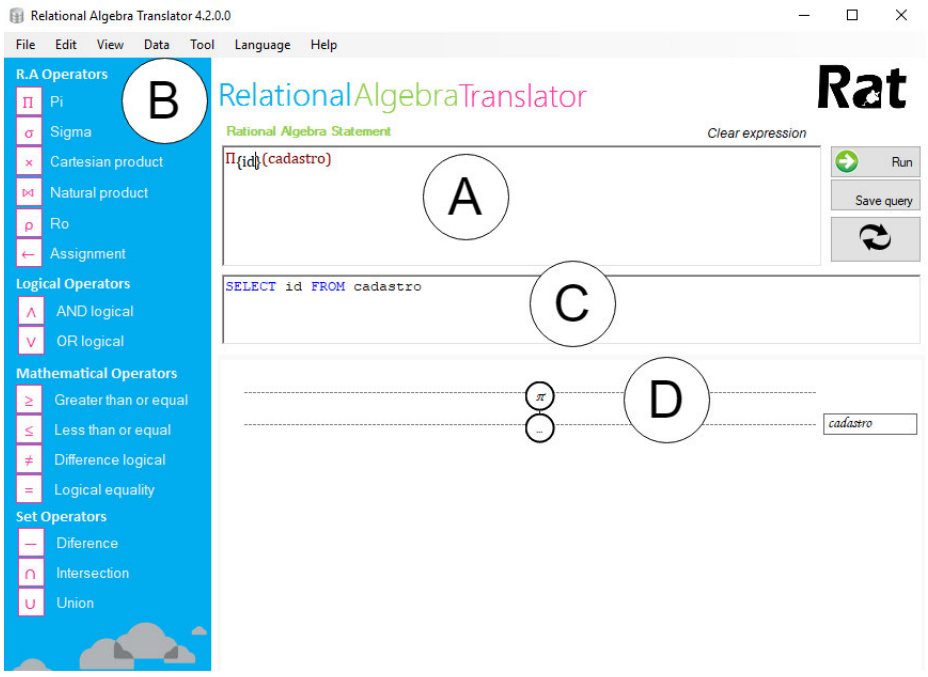
Em relação à análise léxica e sintática no RAT, só ocorre quando a consulta é executada. Já a transformação para SQL e a criação da árvore interativa ocorre acompanhando o que o usuário está digitando, de maneira que a cada informação nova escrita pelo usuário, o SQL e a árvore são atualizados.

Como principal vantagem, há o fato de que a transformação para SQL e a criação da árvore ocorrem junto ao que o usuário está digitando, dando assim um aspecto mais interativo à ferramenta. Além da presença de todas as operações de Álgebra Relacional disponíveis ao usuário.

Como desvantagem principal, há o fato de não mostrar o resultado da operação ao usuário. Assim, não é possível visualizar o que aquela expressão escrita gera, além de não mostrar os nomes das tabelas e suas respectivas colunas.

A Figura 4 reproduz a tela principal onde são realizadas as consultas. Conforme percebemos: em A está localizado o espaço onde o usuário pode digitar a expressão em Álgebra Relacional, em C é possível visualizar a expressão transformada em SQL e em D a árvore de operadores interativa gerada; em B estão os operadores relacionais.

Figura 4 – Tela de consulta RAT.



Fonte: Elaborada pelo Autor

1.2.4 Comparação das soluções encontradas

No levantamento bibliográfico foram encontradas diversas ferramentas, mas algumas ou não estão disponíveis *online*, ou tiveram a instalação impossibilitada devido suas tecnologias estarem ultrapassadas. Sendo assim, apenas foi possível testar as três soluções citadas anteriormente.

A Figura 5 apresenta um comparativo de todas as ferramentas analisadas, referente aos seguintes aspectos:

- Se todas as operações de Álgebra Relacional estão disponíveis ao usuário;
- Se há uma avaliação léxica e sintática, para que, após terminar de digitar uma expressão, o usuário saiba se está ou não correta;
- Se o SQL equivalente é exibido, já que este é um caminho para chegar ao resultado;
- Se o resultado é exibido, pois a visualização deste facilitará o entendimento por parte do usuário;
- Se a ferramenta é intuitiva, pois se forem mais simples ajudam no processo de aprendizagem;
- Se é uma ferramenta WEB, pois ambientes *online* são de mais fácil acesso;

Como é possível observar pela Figura 5, das várias soluções analisadas, nenhuma atende a todos os aspectos desejados, por isso a importância da construção de uma solução mais completa, que englobe todos os aspectos aqui discutidos.

Figura 5 – Tabela de comparação de ferramentas.

Fonte: Elaborada pelo Autor

1.3 Organização do Trabalho

O restante deste trabalho está organizado como segue:

- No capítulo 2 são abordados conceitos relevantes para o desenvolvimento deste trabalho, como: ferramentas educacionais, Álgebra Relacional e seus operadores, sobre SQL e sobre análise léxica e sintática.
- No capítulo 3, apresenta-se as etapas percorridas para a implementação da FEAR.
- No capítulo 4, é realizada a conclusão, bem como sugestões de trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, será apresentada toda a fundamentação teórica necessária para a compreensão dos conceitos que foram utilizados para a implementação da ferramenta, de forma a englobar o conjunto de aspectos apontados no capítulo anterior. Para tanto, abordamos: Ferramentas de Ensino, Álgebra Relacional, SQL, Análise léxica e Análise sintática.

2.1 Ferramentas de Ensino

Para a Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação (BRASSCOM), até 2020 o setor precisará de 750 mil novos trabalhadores qualificados. No entanto, o mercado brasileiro na área de TI vem apresentando um vasto déficit de profissionais. A BRASSCOM aponta ainda que, a cada dez profissionais, apenas dois são aprovados em seus testes técnicos aplicados em processos de seleção dos candidatos. Esses dados, além de alarmantes no que diz respeito ao mercado de TI, acendem o alerta para a necessidade de investir no aperfeiçoamento técnico dos profissionais da área (BRASSCOM, 2019).

Segundo SILVA et al. (2014), as causas para a falta de profissionais referem-se a, desde o baixo interesse dos estudantes brasileiros por ciências exatas até a alta evasão dos cursos ligados a tecnologia. O autor afirma ainda, que um dos fatores que geram os elevados índices de evasão dos cursos da área de computação, reside no fato de os alunos apresentarem dificuldades nas disciplinas que abrangem os conteúdos de raciocínio lógico-abstrato, algoritmos e programação.

Para NETO; IMAMURA (2012), é necessário e justificável, portanto, que se utilizem os recursos tecnológicos, a fim de transformá-los em opções pedagogicamente corretas que realmente somem importantes contribuições ao trabalho dos docentes. Nesse contexto, os softwares educacionais podem ser utilizados como uma boa ferramenta pelos docentes, uma vez que o software interage e prende a atenção do aluno, facilitando a assimilação de informações e tornando o aprendizado mais dinâmico.

Dentro de um contexto pedagógico, existe então, a necessidade de adequação do sistema para que este seja melhor compreendido e manejado pelo usuário, evitando ao máximo que este se confunda e cometa erros. Daí surge o termo Interação Humano-Computador (IHC), que é o estudo do processo de desenvolvimento cujo objetivo principal é promover uma mudança do projeto centrado no software para projeto centrado no usuário (LOPES, 2016).

Segundo FERNANDES; RAABE; BENITTI (2004), baseado nos princípios da IHC, o sistema deve apresentar uma interface bem diagramada, de fácil manejo, interativa e atraente, uma vez que é a interface que realiza o contato com o usuário e é através dela que as informações são comunicadas. É afirmado ainda, que a interface deve possuir uma série de critérios que, juntos, farão o sistema ser bem compreendido, tais como: condução, carga de trabalho, controle explícito do usuário, adaptabilidade, gestão de erros, homogeneidade e coerência, significado dos códigos e denominações e compatibilidade.

De acordo com GLADCHEFF (2001), programas de exercício e prática talvez sejam a maneira mais comum de utilização do computador na educação. Estes programas são utilizados para revisar o material que foi visto em classe, especialmente aquele que envolva memorização e repetição, apresentando exercícios logo de início. Requerem resposta imediata do aluno, propiciando um *feedback*. São também indicados para permitir que os alunos mais avançados possam progredir na matéria em ritmo mais acelerado, ou mesmo permitir que alunos defasados possam alcançar os outros, trabalhando fora do horário normal.

Desta forma, a aprendizagem em um sistema que preza pelo entendimento do usuário se torna muito mais fácil e os ensinamentos melhor compreendidos. As propriedades de usabilidade e aprendizagem caminham juntas na relação Interação Homem-Computador, conferindo à interface qualidade pedagógica a um software educacional. Já os exercícios e práticas proporcionam uma vivencia além daquela na sala de aula.

2.2 Álgebra Relacional

A área de Banco de Dados é fundamental por ser a solução para diversos problemas encontrados nos antigos meios de armazenamento de informação. Segundo KORTH H.F. E SILBERSCHATZ (2006) os sistemas de banco de dados surgiram em resposta aos métodos mais antigos de gerenciamento computadorizado de dados comerciais, de modo que os sistemas de banco de dados apareceram como solução para as desvantagens encontradas nesses antigos métodos, pois estes apresentavam problemas, como a redundância e inconsistência de dados, problemas de integridade, isolamento, dificuldade de acesso a dados, problemas de atomicidade e de segurança. Assim, o ensino de Bancos de Dados, na área de computação, é algo essencial.

Dentro da disciplina de Banco de Dados há uma variedade muito rica de assuntos e uma delas é a Álgebra Relacional. De acordo com DATE (2004) a Álgebra Relacional é um conjunto de operações e relações, no qual cada operação usa uma ou mais relações como seus operandos e produz outra relação como seu resultado. Já para KORTH H.F. E SILBERSCHATZ (2006) define Álgebra Relacional como uma linguagem procedural que consiste em um conjunto de operações que usa uma ou duas relações como entrada e

produzem uma nova relação como resultado, o mesmo define linguagem procedural como aquela em que o usuário instrui o sistema a realizar uma sequência de operações no banco de dados para calcular o resultado desejado. Enquanto NAVATHE; ELMASRI (2011) define que o conjunto básico de operações para o modelo relacional é a Álgebra Relacional; essas operações permitem que um usuário especifique as solicitações de recuperação básicas, como expressões da Álgebra Relacional. O resultado desta expressão é uma nova relação, assim, as operações da álgebra produzem novas relações, que podem ser manipuladas ainda mais, usando operações da mesma álgebra.

Independentemente do autor, quanto à Álgebra Relacional, todos concordam que existe um conjunto de operações fundamentais ou básicas e um conjunto de operações adicionais. As operações fundamentais da Álgebra Relacional são operações específicas para bases de dados relacionais, são elas: seleção, projeção, renomeação, diferença, união e produto cartesiano. Sendo estas subdivididas em operações chamadas unárias, pois operam em uma relação: seleção, projeção e renomeação, e aquelas chamadas binárias, pois operam em pares de relações: união, diferença e produto cartesiano. As operações adicionais são aquelas que não acrescentam qualquer capacidade à álgebra, pois é possível chegar a um mesmo resultado com as operações adicionais, mas simplificam consultas comuns, são elas: intersecção, de divisão e de junções (KORTH H.F. E SILBERSCHATZ, 2006).

Todas as operações estão contempladas na solução desenvolvida, assim, o usuário tem acesso a todo o conteúdo da disciplina disponível e, por tanto, mais facilidade para praticar os conceitos trabalhados. Para uma melhor visão de como funcionam as operações, será explicado cada operador, utilizando-se de exemplos para melhor esclarecimento.

2.2.1 Seleção (σ)

A operação de Seleção, representada na Álgebra Relacional pela letra grega σ (sigma), é usada para escolher um subconjunto das tuplas de uma relação que satisfaça uma condição de seleção. Pode-se considerar que a operação Seleção seja um filtro que mantém apenas as tuplas que satisfazem uma condição qualificadora. Como alternativa, podemos considerar que essa operação restringe as tuplas em uma relação para apenas aquelas que satisfazem a condição (NAVATHE; ELMASRI, 2011).

A operação Seleção é definida pela seguinte sintaxe:

$$\sigma_{\text{<condição seleção>}} \mathrm{R}$$

Em que o símbolo σ é usado para indicar o operador de Seleção, R é uma relação e a <condição de seleção> é uma expressão especificada nos atributos de R, essa pode ser

feita com os operadores =, >, <, >=, <=.

Para efeitos didáticos, visando exemplificar as operações que serão discutidas, iremos considerar o seguinte esquema:

- A tabela de Empregado composta dos seguintes atributos (matrícula, nomeE, endereço, sexo, salário, supervisor, depto), onde matrícula é a chave primária;
- Departamento (codDepto, nomeD, matrGerente), com codDepto como chave primária;
- DepLocalizações(codDepto, Localização);
- Alocação(matrEmp, codProj, numHoras);
- Projeto(codProj, nome, localização, deptoControla), com codProj como chave primária;
- Dependentes(matrEmp, nomeDep, sexo, dataNasc, parentesco).

Como exemplo, imagine que é necessário selecionar as tuplas dos empregados que trabalham no departamento de número 5. A expressão em Álgebra Relacional seria:

$\sigma_{\text{depto} = 5}(\text{Empregado})$

2.2.2 Projeção (π)

A operação de Projeção, representada na Álgebra Relacional pela letra grega π (pi), a operação de Seleção escolhe algumas das linhas da tabela enquanto descarta outras linhas. A operação Projeção , por sua vez, seleciona certas colunas da tabela e descarta outras (NAVATHE; ELMASRI, 2011).

Na operação Projeção há algo conhecido como eliminação de duplicatas, que remove quaisquer tuplas duplicadas, de modo que o resultado dessa operação é um conjunto de tuplas distintas.

A operação Projeção é definida pela seguinte sintaxe:

$$\pi_{<\text{lista atributos}>} \text{R}$$

Onde o símbolo π é usado para indicar o operador de Projeção, R é uma relação e <lista atributos> é a sub-lista desejada de atributos da relação R.

Exemplo, buscar as informações de sexo e de salário de todos os empregados:

$\pi_{\text{sexo, salário}}(\text{Empregados})$

2.2.3 Renomeação (ρ)

A operação de Renomeação, representada na Álgebra Relacional pela letra grega ρ (rho), podemos definir uma operação Renomeação - que pode renomear o nome da relação ou os nomes de atributos, ou ambos - como um operador unário (NAVATHE; ELMASRI, 2011).

A operação Renomeação é definida pelas seguintes sintaxes:

$$\rho_{\text{S(B1, B2, ..., Bn)}}(\text{R})$$

Onde o símbolo ρ é usado para indicar o operador Renomeação, S é o novo nome da relação, B1, B2, ..., Bn que são os novos nomes dos atributos e R é o nome da relação. Nesta primeira expressão é possível renomear tanto a relação quanto seus atributos.

$$\rho_{\text{S}}(\text{R})$$

Em que o símbolo ρ é usado para indicar o operador Renomeação, S é o novo nome da relação e R é o nome da relação. Nesta segunda expressão é possível renomear apenas o nome da relação.

$$\rho_{\text{(B1, B2, ..., Bn)}}(\text{R})$$

Onde o símbolo ρ é usado para indicar o operador Renomeação, B1, B2, ..., Bn que são os novos nomes dos atributos e R é o nome da relação. Nesta terceira expressão é possível renomear apenas os atributos.

Exemplo, para renomear a tabela DepLocalizações para DepLocal e os seus atributos codDepto e Localização para codDep e Local, respectivamente:

$\rho_{\text{DepLocal(codDep, Local)}}(\text{DepLocalizações})$

Exemplo, renomear a tabela Dependentes para Família:

$\rho_{\text{Família}}(\text{Dependentes})$

Exemplo, renomear os atributos de Alocação para mEmp, cProj e nHoras:

$\rho_{(\text{mEmp, cProj, nHoras})}(\text{Alocação})$

2.2.4 União (∪)

A operação União, representada na Álgebra Relacional pelo símbolo matemático ∪ (união teórica de conjuntos), é uma operação que faz parte do grupo de operações matemáticas padrão sobre conjuntos. Essa operação resulta em uma nova relação, que inclui todas as tuplas que estão nas duas relações envolvidas, descartando as tuplas duplicadas.

Como é uma operação binária, esta operação é aplicada a dois conjuntos (de tuplas), quando adaptadas aos bancos de dados relacionais, as duas relações precisam ter o mesmo tipo de tuplas, isso significa que as duas relações precisam ter a mesma aridade e cada par correspondente de atributos precisa ter o mesmo domínio (NAVATHE; ELMASRI, 2011).

A operação União é definida pela seguinte sintaxe:

$$R \cup S$$

Em que o símbolo ∪ é usado para indicar o operador União e onde R e S são duas relações.

Exemplo, para obter a matrícula dos empregados que trabalham no departamento 5 ou supervisionam empregados que trabalham no departamento 5:

$\pi_{\text{matricula}}(\sigma_{\text{depto} = 5}(\text{Empregado})) \cup \pi_{\text{supervisor}}(\sigma_{\text{depto} = 5}(\text{Empregado}))$

2.2.5 Intersecção (∩)

A operação de Intersecção, representada na Álgebra Relacional pelo símbolo matemático ∩ (intersecção teórica de conjuntos), é uma operação que faz parte do grupo de operações matemáticas padrão sobre conjuntos. Essa operação, resulta em uma nova relação que inclui todas as tuplas que estão tanto nas duas relações envolvidas.

Assim como na operação de União, na operação de Intersecção é preciso que seja realizada entre duas relações compatíveis.

A operação Intersecção é definida pela seguinte sintaxe:

$$R \cap S$$

Em que o símbolo ∩ é usado para indicar o operador Intersecção e onde R e S são duas relações.

Exemplo, para obter a matrícula dos empregados que trabalham no departamento 5 e são supervisores:

$\pi_{\text{matricula}}(\sigma_{\text{depto} = 5}(\text{Empregado})) \cap \pi_{\text{supervisor}}(\text{Empregado})$

2.2.6 Diferença (-)

A operação de Diferença, representada na Álgebra Relacional pelo símbolo matemático -, é uma operação que faz parte do grupo de operações matemáticas padrão sobre conjuntos. Essa operação resulta em uma nova relação, que inclui todas as tuplas que estão em uma relação, mas não estão em outra.

Assim como na operação de União, na operação de Diferença é preciso que seja realizada entre duas relações compatíveis.

A operação Intersecção é definida pela seguinte sintaxe:

$$R - S$$

Onde o símbolo - é usado para indicar o operador Intersecção e onde R e S são duas relações.

Exemplo, para obter a matrícula dos empregados que trabalham no departamento 5 e não são supervisores. A expressão em Álgebra Relacional seria:

$\pi_{\text{matricula}}(\sigma_{\text{depto} = 5}(\text{Empregado})) - \pi_{\text{supervisor}}(\text{Empregado})$

2.2.7 Produto Cartesiano (X)

A operação de Produto Cartesiano, representada na Álgebra Relacional pelo símbolo X. Essa operação resulta em uma nova relação, que inclui todas as tuplas que estão em duas relações.

Apesar de ser uma operação binária, assim como a operação União, as relações sobre as quais ela é utilizada não precisam ser compatíveis. É produzido um novo elemento

combinando cada membro (tupla) de uma relação (conjunto) com cada membro (tupla) da outra relação (conjunto) (NAVATHE; ELMASRI, 2011).

A operação Produto Cartesiano é definida pela seguinte sintaxe:

$$R \times S$$

Onde o símbolo X é usado para indicar o operador Produto Cartesiano e onde R e S são duas relações.

Exemplo, para obter para cada empregado do sexo feminino, uma lista dos nomes de seus dependentes:

$\pi_{\text{nomeE, nomeDep}}\ (\sigma_{\text{matr = matrEmp}}\ (\sigma_{\text{sexo='F'}}(\text{Empregado})\ \text{X Dependentes}))$

2.2.8 Junção (⋈)

A operação de Junção, representada na Álgebra Relacional pelo ⋈, é utilizada quando se deseja combinar tuplas relacionadas de duas relações em uma única tupla.

Pode-se chegar ao mesmo resultado da operação Junção através de uma operação Produto Cartesiano seguida de uma operação Seleção, mas a Junção é importante, pois é usada frequentemente quando se especifica consultas de bancos de dados (NAVATHE; ELMASRI, 2011).

A operação Junção é definida pela seguinte sintaxe:

$$R \bowtie_{\text{<condição junção>}} S$$

Em que o símbolo ⋈ é usado para indicar o operador Junção, R e S são duas relações e <condição junção> é a condição que a combinação que R e S tem que satisfazer, essa condição pode ser feita com os operadores =, >, <, >=, <=.

Exemplo, obter o nome do gerente de cada departamento:

$\pi_{\text{nomeD, nomeE}}\ (\text{Departamento} \bowtie_{\text{matrGerente = Matricula}} \text{Empregado})$

2.2.9 Junção Externa

As operações de Junções Externas são extensões da operação de Junção que são necessárias para especificar certos tipos de consultas. A operação de Junção descrita

anteriormente casam as tuplas que satisfizerem a condição de junção. Nelas as tuplas sem uma tupla correspondente (ou relacionada) são eliminadas do resultado de Junção. As tuplas com valores *null* nos atributos também são eliminadas. Isso equivale à perda de informação quando se espera que o resultado gere um relatório baseado em todas as informações das relações componentes.

Um conjunto de operações, chamado Junções Externas, pode ser utilizado quando queremos manter todas as tuplas em R, ou todas aquelas em S, ou todas aquelas em ambas as relações no resultado da Junção, independentemente se elas têm ou não tuplas correspondentes na outra relação. Isso satisfaz a necessidade de consultas nas quais as tuplas de duas tabelas são combinadas pelo casamento de linhas correspondentes, mas sem a perda de qualquer tupla por falta de valores casados (NAVATHE; ELMASRI, 2011).

2.2.9.1 Junção Externa à Esquerda (⟕)

A operação de Junção Externa à Esquerda, representada na Álgebra Relacional pelo símbolo ⟕, é uma extensão da operação Junção e é utilizada quando se deseja manter cada tupla da relação à esquerda, R, se nenhuma tupla correspondente for encontrada em S, então os atributos de S no resultado da junção são preenchidos com valores *null* (NAVATHE; ELMASRI, 2011).

A operação Junção Externa à Esquerda é definida pela seguinte sintaxe:

$$R ⟕_{\text{<condição junção>}} S$$

Em que o símbolo ⟕ é usado para indicar o operador Junção Externa à Esquerda, R e S são duas relações e <condição junção> é a condição que a combinação que R e S tem que satisfazer, essa condição pode ser feita com os operadores =, >, <, >=, <=.

Exemplo, é preciso obter as informações de todos os Empregados, bem como as informações dos departamentos que eles gerenciam, se eles gerenciarem um departamento:

Empregado ⟕ $_{\text{Matricula = matrGerente}}$ Departamento

2.2.9.2 Junção Externa à Direita (⟖)

A operação de Junção Externa à Direita, representada na Álgebra Relacional pelo símbolo ⟖, é uma extensão da operação Junção e é utilizada quando se deseja manter cada tupla da relação à direita, S, se nenhuma tupla correspondente for encontrada em R, então os atributos de R no resultado da junção são preenchidos com valores *null* (NAVATHE; ELMASRI, 2011).

A operação Junção Externa à Direita é definida pela seguinte sintaxe:

$$R ⟖_{\text{<condição junção>}} S$$

Em que o símbolo ⟖ é usado para indicar o operador Junção Externa à Direita, R e S são duas relações e <condição junção> é a condição que a combinação que R e S tem que satisfazer, essa condição pode ser feita com os operadores =, >, <, >=, <=.

Exemplo, obter todas as informações de Departamento, bem como a de seus Empregados, se tiverem empregados:

Empregado $⟖_{\text{Matricula = matrGerente}}$ Departamento

2.2.9.3 Junção Externa Total (⟗)

A operação de Junção Externa Total ou Junção Externa Completa, representada na Álgebra Relacional pelo símbolo ⟗, é uma extensão da operação Junção e é utilizada quando se deseja manter todas as tuplas nas relações da esquerda e da direita quando nenhuma tupla correspondente for encontrada, preenchendo-as com valores *null* (NAVATHE; ELMASRI, 2011).

A operação Junção Externa Total é definida pela seguinte sintaxe:

$$R ⟗_{\text{<condição junção>}} S$$

Em que o símbolo ⟗ é usado para indicar o operador Junção Externa Total, R e S são duas relações e <condição junção> é a condição que a combinação que R e S tem que satisfazer, essa condição pode ser feita com os operadores =, >, <, >=, <=.

Exemplo, obter informações de todos os Empregados e de todos os departamentos:

Empregado $⟗_{\text{Matricula = matrGerente}}$ Departamento

2.2.10 Divisão (÷)

A operação de Divisão, representada na Álgebra Relacional pelo símbolo ÷, é definida por conveniência para lidar com consultas que envolvem quantificação universal. Para uma tupla aparecer no resultado da operação Divisão, os valores da tupla precisam aparecer na primeira relação em combinação com cada tupla da segunda relação(NAVATHE; ELMASRI, 2011).

A operação Divisão é definida pela seguinte sintaxe:

$$R \div S$$

Na qual, o símbolo ÷ é usado para indicar o operador Divisão, R e S são relações.

Como exemplo, suponha que é preciso recuperar os nomes dos empregados que trabalham em todos os projetos:

$\pi_{\text{codProj}}(\text{Projeto}) \div \pi_{\text{matrEmp, codProj}}\ (\text{Alocação})$

2.2.11 Atribuição (←)

A operação de Atribuição, representada na Álgebra Relacional pelo símbolo ←, é conveniente para escrever uma expressão de Álgebra relacional atribuindo partes dela a variáveis de relação temporárias. (KORTH H.F. E SILBERSCHATZ, 2006).

A operação Atribuição é definida pela seguinte sintaxe:

$$\begin{aligned} &temp1 \leftarrow R \\ &temp2 \leftarrow S \\ &temp1 \div temp2 \end{aligned}$$

Na qual, o símbolo ← é usado para indicar o operador Atribuição, R e S são relações, temp1 e temp2 são variáveis temporárias e ÷ representa a operação de Divisão, mas esta pode ser qualquer outra operação binária.

Como exemplo, será utilizado o mesmo contexto do exemplo da operação Divisão, em que era preciso recuperar os nomes dos empregados que trabalham em todos os projetos:

$temp1 \leftarrow \pi_{\text{codproj}}(\text{Projeto})$

$temp2 \leftarrow \pi_{\text{matrEmp, codProj}}(\text{Alocação})$

$temp1 \div temp2$

2.3 SQL

A linguagem *Strutured Query Language* (SQL) é geralmente referida como de consulta estruturada em banco de dados, no entanto, segundo KORTH H.F. E SILBERSCHATZ (2006), ela pode fazer muito mais do que simplesmente consultar um banco de

dados. Esta linguagem pode definir a estrutura dos dados, modificar dados no banco de dados e especificar restrições de segurança. Já (NAVATHE; ELMASRI, 2011) aponta para o fato de que o Cálculo Relacional é considerado a base para a linguagem SQL, e a Álgebra Relacional é usada nos detalhes internos de muitas implementações de banco de dados para processamento e otimização de consulta, e ainda salienta que a linguagem SQL é o padrão para Sistemas de Gerenciamento de Banco de Dados (SGBD) relacionais. SGBDs são programas responsáveis pelo gerenciamento do banco de dados.

A linguagem SQL se tornou um padrão para banco de dados relacionais. Para NAVATHE; ELMASRI (2011) ela é dividida em dois grandes grupos de comandos: Linguagem de Manipulação de Dados (LMD), que versa sobre comandos de manipulação, e Linguagem de Definição de Dados (LDD), sobre comandos de definição, por não ser o objeto de estudo, não iremos detalhar esse grupo de comandos.

2.3.1 LMD - Linguagem de Manipulação de Dados

A linguagem SQL é composta por um conjunto de comandos de manipulação. A LMD da SQL inclui uma linguagem de consulta baseada na Álgebra Relacional e no Cálculo Relacional de tupla. Ela também inclui comandos para inserir, excluir e modificar tuplas no banco de dados (KORTH H.F. E SILBERSCHATZ, 2006).

A LMD possui os seguintes comandos: SELECT, INSERT, UPDATE e DELETE, destes iremos apenas detalhar o SELECT, pois o foco está voltado para formulação de comandos de recuperação.

2.3.1.1 SELECT

O comando SELECT é usado para listar atributos desejados no resultado de uma consulta. Normalmente é acompanhado por FROM, e às vezes por WHERE também. O comando FROM lista as relações a serem lidas e o comando WHERE lista certas condições envolvendo atributos da relação (KORTH H.F. E SILBERSCHATZ, 2006).

Definimos a sintaxe do comando SELECT como:

```
SELECT <lista atributos>
FROM <lista tabelas>
WHERE <condição>
```

Em que <lista atributos> é uma lista de nomes de atributo cujos valores devem ser recuperados pela consulta, <lista tabelas> é uma lista dos nomes de relação exigidos para processar a consulta e <condição> é uma expressão condicional que identifica as tuplas a serem recuperadas pela consulta.

Como exemplo, selecione as informações dos empregados que trabalham no departamento de número 5:

SELECT *

FROM Empregado

WHERE depto = 5

Isso corresponde à mesma consulta de seleção em Álgebra Relacional, que foi utilizada como exemplo no item 2.2.1 deste Capítulo.

2.4 Análise léxica e Análise sintática

Como a ideia é escrever um comando em Álgebra Relacional e mostrar o equivalente em SQL, é preciso que se faça uma análise léxica e uma análise sintática, de modo que, ao ser executado, a ferramenta possa avaliar se o que foi escrito, está ou não correto.

Segundo AHO et al. (2007), o analisador léxico é a primeira fase de um compilador. Sua principal tarefa é a de ler os caracteres de entrada e produzir uma sequência de *tokens*, que são sequências de caracteres tendo um significado coletivo, que o *parser*, que é um analisador sintático, utiliza para a análise sintática. Então, ao analisarmos uma palavra, o analisador léxico é a forma de verificar se os caracteres dessa palavra fazem parte de determinado alfabeto.

A importância da análise léxica na solução aqui proposta deve-se ao fato de que será necessário verificar se cada elemento introduzido pelo usuário faz parte do alfabeto, como por exemplo, verificar se o σ (ou o que estiver representando este) faz parte da linguagem de Álgebra Relacional. Isso é de suma importância, pois para ver se a estrutura de consulta está correta, precisamos antes verificar se seus elementos fazem parte desta linguagem.

Conforme AHO et al. (2007), o analisador sintático obtém uma cadeia de *tokens* provenientes do analisador léxico e verifica se a mesma pode ser gerada pela gramática da linguagem, espera-se que o analisador sintático relate quaisquer tipos de erros de sintaxe de uma forma inteligível. Então, a análise sintática pode ser definida como o processo que analisa a sequência de entrada, de modo a verificar se a estrutura da frase está correta.

A análise sintática é necessária para verificar se a estrutura da consulta está correta, como por exemplo, averiguar se a expressão $\sigma_{\text{nome = 'Paulo'}}(\text{Empregado})$ está correta. A sua importância deve-se ao fato de que uma consulta não pode ocorrer de forma adequada, se sua estrutura estiver incorreta.

2.4.1 Geradores Automáticos de Analisadores

Como há a necessidade de que o ambiente a ser criado realize uma análise léxica e uma análise sintática, é preciso então construir um analisador léxico e sintático. Devido ao grande esforço necessário para a construção de um analisador léxico e sintático, existem geradores que auxiliam na construção de um, dentre eles:

2.4.1.1 LEX/FLEX: Gerador de Analisador Léxico

A ferramenta Lex ou Flex, permite especificar um analisador léxico, definindo expressões regulares para descrever padrões para os *tokens*. Ela é distribuída como parte do pacote de compilação GNU, produzido pela *Free Software Foundation*, e está disponível gratuitamente em diversos endereços na internet (PAXSON, 1998).

Lex é um programa que recebe como entrada um arquivo de texto contendo expressões regulares, juntamente com as ações associadas a cada expressão. A notação de entrada para Lex é chamada de linguagem Lex, e a ferramenta em si é o compilador Lex.

Como vantagem, há o fato de ser gratuita e ter sido a primeira neste estilo, mas possui várias desvantagens, como o fato de gerar só analisadores léxicos em C ou em C++, precisando assim, de outro gerador para construção de um analisador sintático. Além de não ter interface gráfica, de modo que tudo é feito por linha de comando.

2.4.1.2 YACC - Yet Another Compiler-Compiler

O YACC é um gerador de analisadores sintáticos que roda no ambiente UNIX. Aceita como entrada uma especificação das características sintáticas da linguagem, com ações semânticas embutidas, e gera uma rotina em C para a análise sintática (BROWN; LEVINE, 1992).

O analisador gerado é feito de modo a trabalhar em conjunto com uma rotina de análise léxica gerada pelo LEX, permitindo uma fácil integração entre os analisadores léxico (gerado pelo LEX) e sintático (gerado pelo YACC). Todos são gerados no mesmo ambiente (UNIX) e na mesma linguagem (C). A especificação é constituída de uma gramática livre de contexto.

Como vantagem, tem o fato de ser gratuita, mas possui várias desvantagens, como o fato de gerar só analisadores sintáticos na linguagem C e só no ambiente UNIX, precisando assim, de outra ferramenta para a construção de um analisador léxico (normalmente o YACC trabalha em conjunto com o LEX). Não tem interface gráfica, de modo que tudo é feito por linha de comando.

2.4.1.3 CUP - Java Based Constructor of Useful Parsers

A CUP é um sistema para gerar analisadores sintáticos a partir de especificações simples. Tem o mesmo papel do programa amplamente utilizado YACC e na verdade possui a maioria das características do YACC. No entanto, CUP é escrito em Java, utiliza especificações incluindo código Java embutido, e produz analisadores que são implementadas em Java (HUDSON, 2013).

O uso do CUP envolve a criação de uma especificação simples baseada na gramática para qual será feito o reconhecimento, juntamente com a construção de um scanner capaz de ler tokens a partir da entrada.

Como vantagem, há o fato de ser gratuita, mas, da mesma forma que o YACC, possui várias desvantagens, como o fato de gerar apenas analisadores sintáticos na linguagem Java, precisando assim, de outra ferramenta para construir um analisador léxico. Além de não ter interface gráfica, de modo que tudo é feito por linha de comando.

2.4.1.4 GALS - Gerador de Analisadores Léxicos e Sintáticos

O GALS é um ambiente para a geração de analisadores léxicos e sintáticos, desenvolvido em Java. GALS é um Software Livre que permite a criação de um analisador léxico e sintático, permitindo que o usuário defina se os analisadores serão separados ou se serão integrados um ao outro. Pode-se escolher a linguagem em que o analisador será gerado, podendo escolher entre: C++, Java ou Delphi (GESSER, 2003).

No GALS, os aspectos léxicos de uma especificação são definidos pela declaração dos *Tokens*, que define quais construções léxicas serão aceitas, na Figura 6 é demonstrado na posição A onde são declarados os *Tokens*. A forma geral da definição de um *token* é:

<TOKEN> : <EXPRESSÃO REGULAR>

Só é permitida a definição de um *token* por linha. São aceitos comentários, da mesma forma que nas definições regulares. Aqui, *TOKEN* pode ser um identificador, ou uma sequencia de caracteres entre “” (GESSER, 2003).

Os aspectos léxicos também são definidos pela declaração de Definições Regulares, que são expressões auxiliares para a definição de *Tokens*, na Figura 6, é demonstrado na posição B, onde são declaradas as Definições Regulares. Todas as definições regulares são da forma:

<IDENTIFICADOR> : <EXPRESSÃO REGULAR>

Só é permitida uma definição por linha. Identificadores repetidos são considerados erros, que são reportados ao usuário. Pode-se ainda inserir comentários de linha, que se iniciam com "//" e se estendem até o fim da linha (GESSER, 2003).

Os aspectos sintáticos são compostos pela declaração de Símbolos Terminais (em projetos, com Analisadores Léxico e Sintático conjuntos, os *Tokens* declarados na Especificação Léxica são tomados como Símbolos Terminais), Símbolos Não-Terminais, na Figura 6, é demonstrado na posição C, onde são declarados estes símbolos, e Gramática, na Figura 6, é demonstrado em D onde é declarada a gramática.

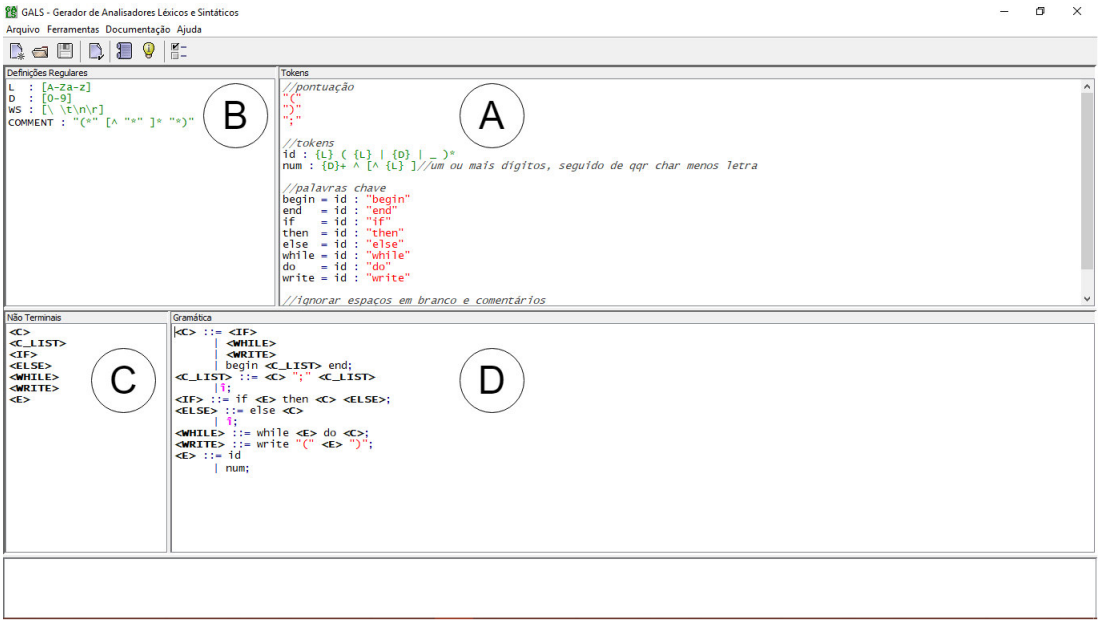
Os Símbolos Não-Terminais são definidos por identificadores entre < e >. O primeiro símbolo é considerado o símbolo inicial da gramática.

A construção da Gramática segue a forma:

<NT> ::= <LISTA DE SÍMBOLOS> | <LISTA DE SÍMBOLOS> | ... ;

Onde NT é um símbolo não-terminal e LISTA DE SÍMBOLOS é uma lista de símbolos terminais e não terminais e de ações semânticas (a serem vistas mais tarde). Todos os símbolos utilizados aqui devem ter sido previamente definidos. O símbolo terminal é representado por î.

Figura 6 – Ambiente do GALS.



Fonte: Elaborada pelo Autor

Como vantagem, há o fato de o GALS possuir interface gráfica, ter diferentes opções de linguagem em que o analisador será gerado e a possibilidade de construir analisador léxico e sintático integrado. A desvantagem principal é fato de o usuário precisar de um

estudo prévio da linguagem que o programa utiliza para entrada dos dados, para inserir os dados da gramática a ser analisada.

2.4.1.5 Comparação das soluções encontradas

Com o objetivo de decidir qual solução seria utilizada para a construção de um analisador léxico e sintático, ferramentas cujo propósito é auxiliar na construção de compiladores foram analisadas. Desta forma, foram descobertos os pontos fracos e fortes de cada uma.

Em relação à interface com o usuário nos sistemas LEX, YACC e CUP apresentados, a interação entre o usuário e o programa é através da linha de comando. Isto obriga o usuário a especificar um compilador em um arquivo texto, que é enviado para a entrada do programa. Esta característica constitui uma dificuldade para o uso das ferramentas. Dos ambientes analisados o GALS é a única a possuir interface gráfica.

Em relação ao que é gerado nos sistemas LEX, YACC e CUP apresentados, o resultado final é ou um analisador léxico ou um analisador sintático. Isto obriga o usuário a procurar uma outra ferramenta para suprir a falta do outro analisador, caso este usuário queira um analisador léxico e sintático. O GALS é o único que gera um analisador léxico e sintático integrado.

Após comparações, chegou-se à conclusão que o GALS é o que melhor supre as necessidades do trabalho aqui proposto. Assim, o GALS foi utilizado para gerar um analisador léxico e sintático de Álgebra Relacional. A linguagem escolhida do analisador gerado foi C++.

2.4.2 Compiladores

Na perspectiva de que a solução aqui proposta deverá ser *online*, e que o analisador léxico e sintático da mesma estará na linguagem C++, que não é uma linguagem para web, tem-se a necessidade de traduzir o código do analisador para web, ou seja, é necessário compilar o código.

Segundo AHO et al. (2007), um compilador é um programa que lê um programa escrito numa linguagem - a linguagem fonte - e o traduz num programa equivalente numa outra linguagem - linguagem alvo. Apesar de existir uma variedade muito grande de compiladores, além do compilador Emscripten, não foi encontrado nenhum outro que compile o código C/C++ para uma linguagem web.

2.4.2.1 Emscripten

O Emscripten é um compilador de *Low Level Virtual Machine* (LLVM) - é uma infraestrutura de compilador escrita em C++ para web. Ele recebe *bytecode* LLVM e

compila para web. O Emscripten funciona apenas por linha de comando. Ao utilizar o Emscripten, é possível compilar um código em C ou C++ para web, compilar qualquer outro código que pode ser traduzido em *bytecode* de LLVM para web e compilar *runtimes* C/C++ de outras linguagens para web, e então executar o código nessas outras linguagens de forma indireta.

O Emscripten torna o código nativo imediatamente disponível na web. A vantagem do Emscripten é que, desenvolvedores de C/C++ não tem o alto custo de portar código, variando de jogos de alta performance até *frameworks* de aplicativos, manualmente para web. A linguagem alvo do Emscripten pode ser JavaScript ou WebAssembly. No trabalho aqui proposto, a linguagem alvo do Emscripten vai ser o WebAssembly. A linguagem WebAssembly foi escolhida devido ser um novo tipo de linguagem, que além de permitir o uso de programas que estão em C/C++ na web, também pode ser trabalhado em conjunto com o JavaScript (EMSCRIPTEN, 2015).

2.4.2.2 WebAssembly

O WebAssembly (abreviado Wasm) é um formato de instrução binária para uma máquina virtual baseada em pilha. Wasm é projetado como um destino portátil para compilação de linguagens de alto nível como C, C ++ e Rust, permitindo a implementação na web.

O WebAssembly é um novo tipo de código que pode ser executado em *browsers* modernos e trata-se de uma linguagem de baixo nível como o *assembly*, com um formato binário compacto que executa com performance quase nativa e que fornece um novo alvo de compilação para linguagens como C/C++, para que possam ser executadas na web. Também foi projetado para executar em conjunto com o JavaScript, permitindo que ambos trabalhem juntos.

O WebAssembly tem enormes implicações para a plataforma web — ele fornece uma maneira de executar na web códigos escritos em diversas linguagens em velocidade quase nativa, com *apps* que não conseguiriam fazer isso antes (WEBASSEMBLY, 2019).

Para melhor exemplificar o funcionamento e a ligação do Emscripten com o WebAssembly, temos a Figura 7 como um exemplo, segundo o qual a compilação é feita por linha de comando. Conforme percebemos: em A está localizado o comando em que entramos no ambiente do Emscripten (emsdk), em B cria-se um arquivo com o nome ”hello” e em C entramos neste arquivo; em D é criado um arquivo na linguagem C com o nome ”hello.c” e em E está seu código, que é basicamente um ”Hello, world!”; Ao colocarmos o comando ”emcc hello.c -s WASM=1 -o hello.html” em F, estamos dizendo ao compilador para pegar o arquivo em C e compilá-lo para HTML, que é uma linguagem para web.

O resultado será um arquivo HTML, um arquivo JS e um arquivo WASM, criando

assim uma página genérica HTML, como demonstrado na Figura 8, em A percebemos que é mostrado "Hello, world!".

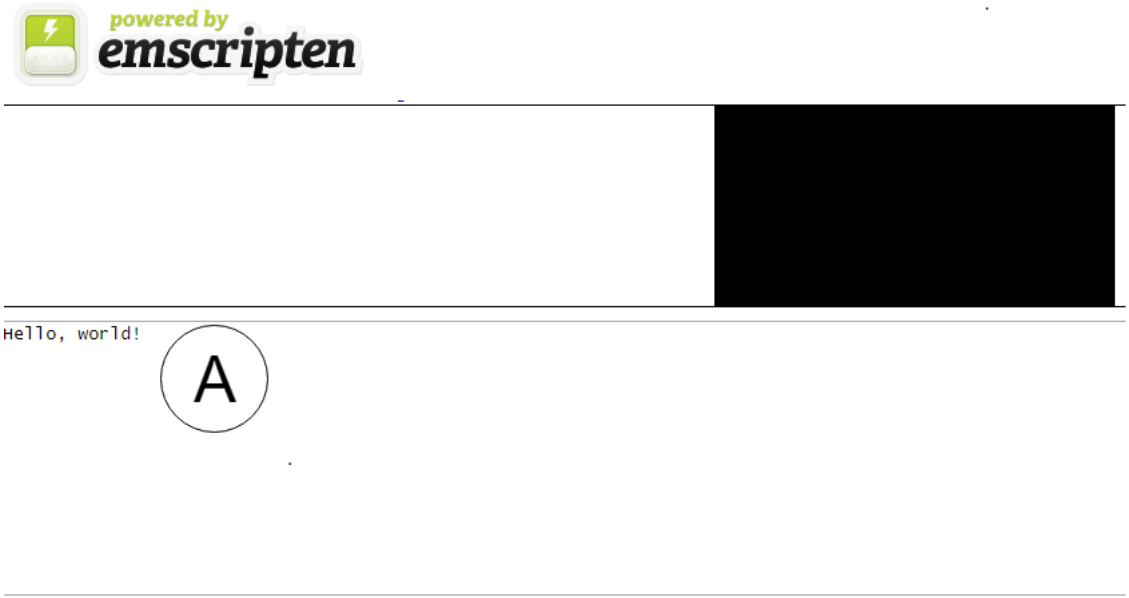
Figura 7 – Funcionamento do Emscripten e do WebAseembly.

```
rod@DESKTOP-FVJFV4S:~$ cd emsdk Ⓐ
rod@DESKTOP-FVJFV4S:~/emsdk$ mkdir hello Ⓑ
rod@DESKTOP-FVJFV4S:~/emsdk$ cd hello Ⓒ
rod@DESKTOP-FVJFV4S:~/emsdk/hello$ cat <<EOF>> hello.c Ⓓ
> #include <stdio.h>
> int main(int argc, char ** argv) {
>   printf("Hello, world!\n");
> }
> EOF
rod@DESKTOP-FVJFV4S:~/emsdk/hello$ emcc hello.c -s WASM=1 -o hello.html Ⓕ
```

Ⓔ

Fonte: Elaborada pelo Autor

Figura 8 – Página genérica.



Fonte: Elaborada pelo Autor

No trabalho aqui proposto, o Emscripten e o WebAssembly foram empregados para compilar o analisador léxico e sintático gerado no GALS em C++ para web.

2.5 Considerações finais

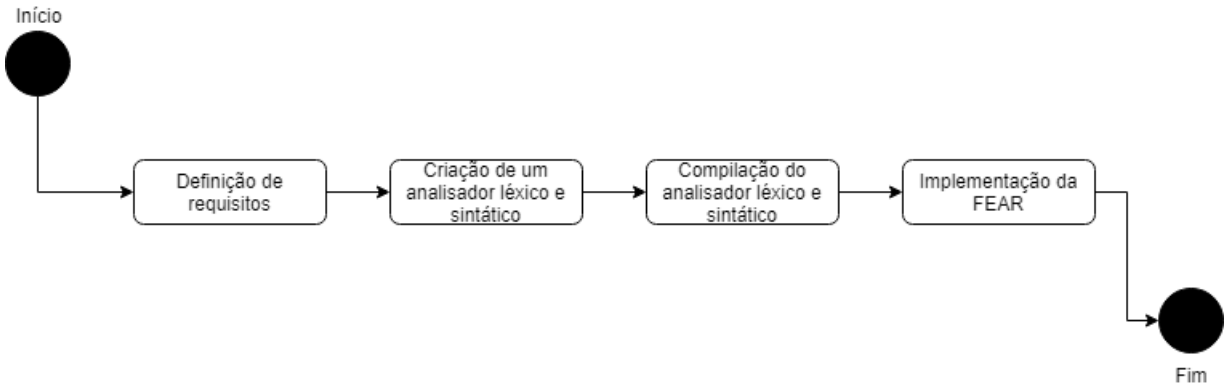
Este capítulo tratou da fundamentação teórica necessária para a compreensão dos conceitos abordados na ferramenta. No próximo capítulo, detalharemos a implementação da FEAR, com o objetivo de demonstrar as tecnologias utilizadas.

3 FEAR: Ferramenta de Ensino a Álgebra Relacional

Este capítulo trata da implementação da ferramenta que foi desenvolvida, com base nos conceitos detalhados no capitulo anterior. A GALS foi utilizada para desenvolver um analisador léxico e sintático. O Emscripten, para compilar o código do analisador criado em GALS para uma linguagem web e o WebAssembly foi a linguagem web escolhida.

A Figura 9 mostra as etapas percorridas para a implementação da FEAR. As etapas são: Definição de Requisitos, Criação de um analisador léxico e sintático, Compilação do analisador léxico e sintático e Implementação da FEAR.

Figura 9 – Esquema de etapas do desenvolvimento da ferramenta.



Fonte: Elaborada pelo Autor

3.1 Definição de Requisitos

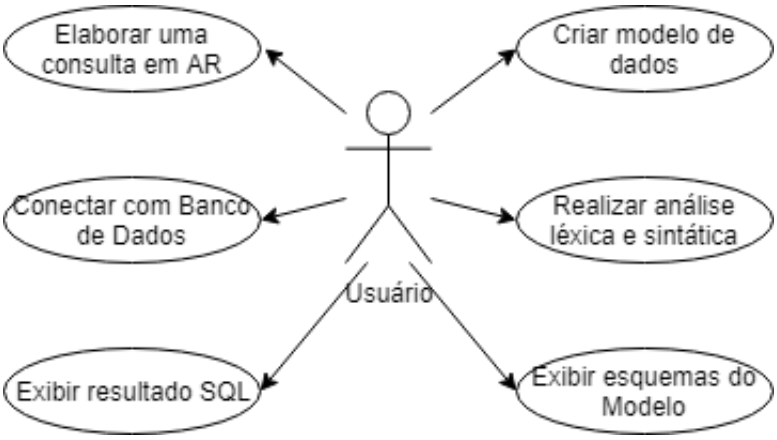
Dentre os principais requisitos elicitados para a ferramenta, destacamos os seguintes:

- Que o ambiente seja online;
- Permitir a criação de um modelo de dados gráfico, que contenha a possibilidade de inserir todos os atributos, seus tipos, tamanhos e índices;
- Listar as relações de esquemas do modelo em que está conectado;
- Suportar as operações: Seleção, Projeção, Renomeação, União, Diferença, Intersecção, Produto Cartesiano, Junção, Junção Externa à Esquerda, Junção Externa à Direita, Junção Externa Total e Divisão;
- Permitir que o usuário possa realizar uma consulta em Álgebra Relacional;

- Realizar uma análise léxica e sintática do que foi escrito pelo usuário;
- Traduzir a consulta feita em Álgebra Relacional para SQL e apresentar esse SQL;
- Executar e apresentar o resultado dos comandos SQL gerados nos modelos de dados;

Para os principais requisitos encontrados, a Figura 10 traz um Diagrama Caso de Uso:

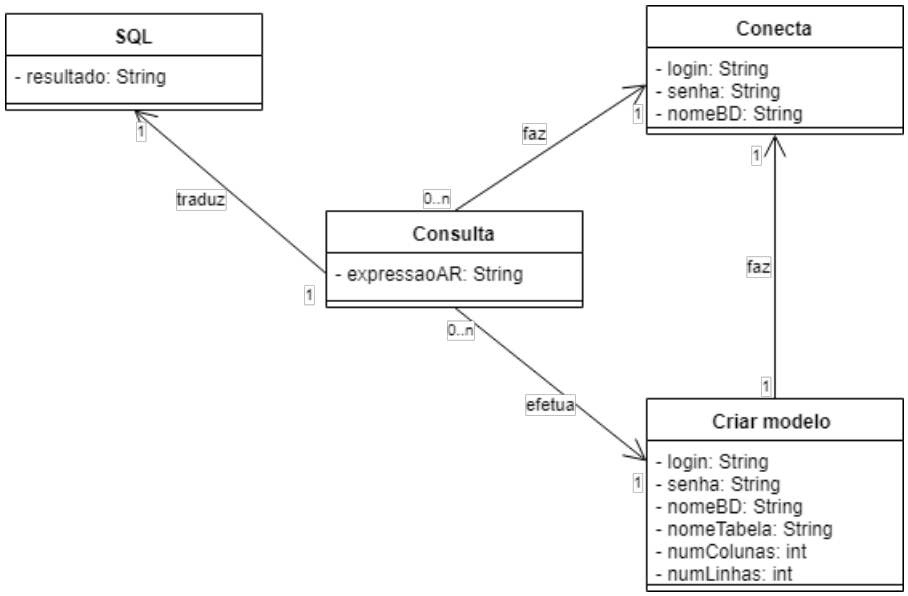
Figura 10 – Diagrama Caso de Uso da FEAR.



Fonte: Elaborada pelo Autor

Já a Figura 11 apresenta o Diagrama de Classe em nível conceitual para a ferramenta desenvolvida.

Figura 11 – Diagrama de Classe em nível conceitual da FEAR.



Fonte: Elaborada pelo Autor

3.2 Criação de um analisador léxico e sintático

Conforme detalhado no item 2.4 do Capitulo 2, um analisador léxico e sintático é um programa que analisa um determinado texto sobre a perspectiva de uma linguagem , de modo a saber se está correto. No contexto deste trabalho foi utilizado a ferramenta GALS, por possuir interface gráfica e por gerar analisadores léxicos e sintáticos integrados.

Como demonstrado na Figura 12, o analisador foi construído da seguinte maneira: no espaço indicado por A, foram definidos os tokens, como por exemplo ”sel”, que na construção da Gramática representará o comando Seleção; em B foram estabelecidos as Definições Regulares, como por exemplo ”L”, que é definida como uma sequência de caracteres e será utilizada na definição dos tokens. Já em D, é escrita a gramática, onde são definidas as regras de sintaxe da Álgebra Relacional, explicadas no item 2.2 do Capitulo 2, assim o analisador a ser construído obedecerá a estas regras. Na posição C, são mostrados os Símbolos Não-Terminais que serão usados na Gramática. A ferramenta, ao ser executada, gerará um analisador léxico e sintático na linguagem C++.

Figura 12 – GALS da FEAR.

Fonte: Elaborada pelo Autor

3.3 Compilação do analisador léxico e sintático

Conforme detalhado no item 2.4.2 do Capitulo 2, o processo de compilação é basicamente ler um programa escrito numa linguagem e traduzi-lo num programa equivalente numa outra linguagem. No contexto deste trabalho, foi utilizado o compilador Emscripten, nele foi pego o código do analisador léxico e sintático desenvolvido no GALS e este foi compilado para a linguagem WebAssembly, que, como foi dito no item 2.4.2.2 do Capitulo

2, é uma nova linguagem para web. Como o código do analisador léxico e sintático é muito extenso, não será possível mostrar todo o processo, mas este é o mesmo do explicado no item 2.4.2.2 do Capitulo 2, mudando apenas o último comando, como mostrado na Figura 13.

Figura 13 – Emscripten e WebAssembly da FEAR.

```
>          cout<<e.getMessage();
>        }
>        catch ( SemanticError &e ) Ⓐ
>        {
>          cout<<e.getMessage();
>        }
>
> }
> EOF
rod@DESKTOP-FVJFV4S:~/emsdk/Compilador3$ emcc -s WASM=1 \ -s DISABLE_EXCEPTION_CATCHING=0 \-s NO_EXIT_RUNTIME=0 -o compi
lador.js compilador.cpp Ⓑ
```

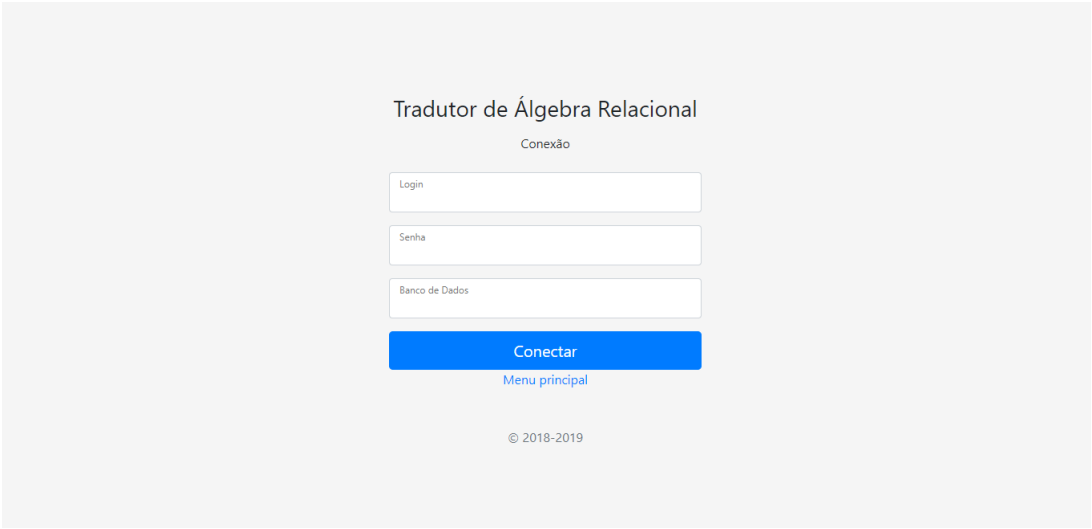
Fonte: Elaborada pelo Autor

Na posição A da Figura 13 vemos o final do código em C++, gerada pela GALS e em B o comando ”emcc -s WASM=1 -s DISABLE_EXCEPTION_CATCHING=0 s NO_EXIT_RUNTIME=0 -o compilador.js compilador.cpp”, em que o ”DISABLE_EXCEPTION_CATCHING=0” ativa as exceções no código otimizado e o ”NO_EXIT_TIME=0” significa que incluímos o código para encerrar o tempo de execução. O arquivo o ”compilador.cpp”, é compilado para um arquivo que possa ser utilizado na web, o ”compilador.js”. Este processo gera dois arquivos, um arquivo WASM e um arquivo js.

3.4 Implementação da FEAR: Ferramenta de Ensino á Álgebra Relacional

Conforme os requisitos citados no item 3.1 no presente Capítulo, a ferramenta teria que permitir conexão com os bancos de dados. Para tanto, como demonstrado na Figura 14, foi criada uma página para a conexão com Banco de Dados na ferramenta. Para conectar-se ao Banco de Dados, como login e senha, será necessário utilizar o mesmo login e senha do SGBD, além de informar o nome do Banco de Dados ao qual deseja estar conectado. Caso não ocorra nenhum erro, após a conexão, o usuário é levado de volta a página inicial. Caso ocorra um erro, é mostrado uma mensagem de erro, e o usuário permanece na mesma página.

Figura 14 – Página para conexão.



Fonte: Elaborada pelo Autor

Conforme os requisitos definidos, a ferramenta teria que permitir a criação de um modelo de dados, para o caso este ainda não tenha sido construído. Para isso, como demonstrado na Figura 15, foi feita uma página para criação de um modelo de dados na ferramenta. Assim como no processo de conexão ao Banco de Dados, para a construção de modelo de dados é necessário informar o login e senha do SGBD. O usuário terá que informar o nome que o modelo de dados e nome de uma tabela. Caso ocorra um erro, é mostrado uma mensagem de erro, e o usuário permanece na mesma página. Em caso de não ocorrer nenhum erro, o usuário é levado para uma outra página, em que possa inserir colunas na tabela.

Figura 15 – Página para criar Modelo de Dados.

Fonte: Elaborada pelo Autor

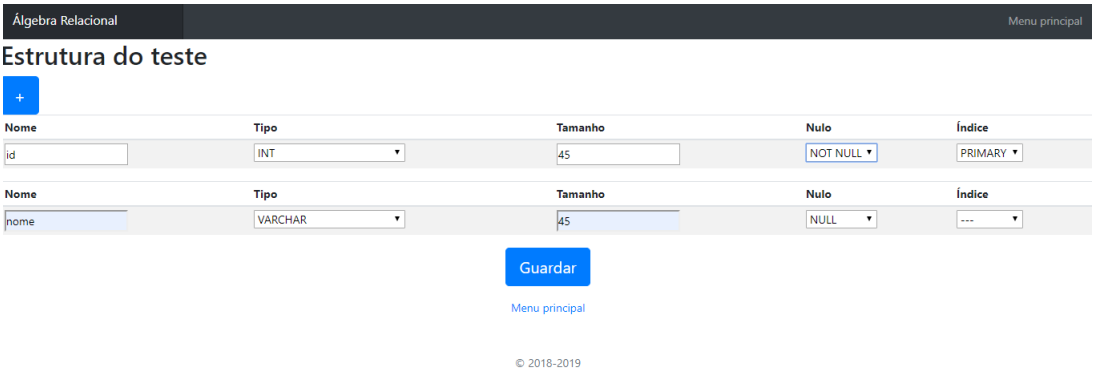
Como demonstrado na Figura 16, foi criada uma página para inserção de colunas

na tabela, onde cada linha representa uma coluna. Para criar uma coluna na tabela, o usuário precisa informar: o nome daquela coluna, o tipo de dados dela (podem ser INT, VARCHAR, etc.), o tamanho, o índice (podem ser PRIMARY KEY, UNIQUE KEY, etc.) e se é ou não nula, nos seus respectivos espaços. O usuário pode acrescentar quantas colunas desejar, clicando no símbolo "+".

No exemplo da Figura 16, temos duas colunas sendo criadas: a primeira é uma coluna de nome id, do tipo inteira (INT) e que possui o tamanho 45, este tamanho é máximo de dígitos que uma variável dessa coluna pode ter, também é determinado que ela não aceitará valores nulos e que como é uma chave primária, seu valor não pode ser repetido. A segunda é uma coluna de chamada nome, do tipo palavra (VARCHAR) e de tamanho 45, também é determinado que ela aceitará valores nulos e vemos que, como Índice, na foi determinado.

Caso ocorra um erro, é mostrado uma mensagem de erro, e o usuário permanece na mesma página. Em caso de não ocorrer nenhum erro, o usuário é levado para uma outra página, em que possa inserir tuplas na tabela.

Figura 16 – Página para adicionar colunas na tabela.

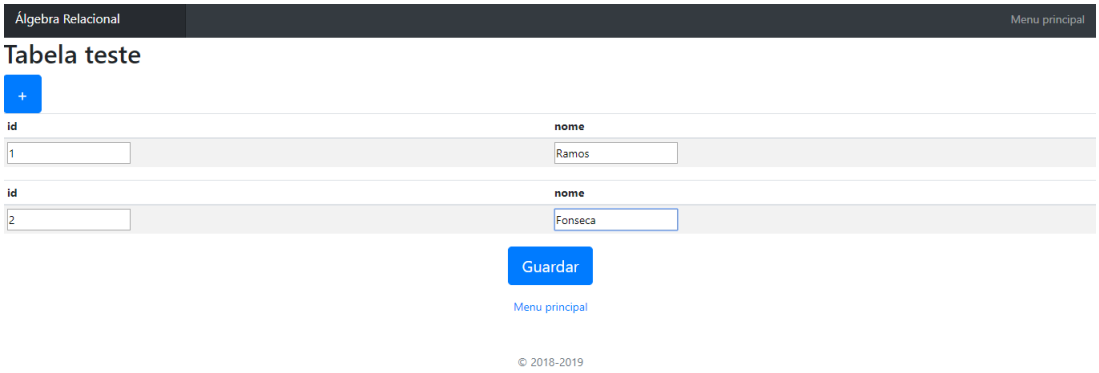


Fonte: Elaborada pelo Autor

Como demonstrado na Figura 17, foi criada uma página para inserção de tuplas na tabela, onde cada linha representa uma tupla. Para criar uma tupla na tabela, o usuário precisa preencher cada linha, respeitando as condições determinadas de cada atributo. Continuando o exemplo anterior, onde foram criadas as colunas id e nome, e respeitando suas restrições, temos duas tuplas sendo criadas: a primeira é uma como id de valor 1 e como nome Ramos e a segunda com id de valor 2 e como nome Fonseca.

Caso ocorra um erro, é mostrado uma mensagem de erro, e o usuário permanece na mesma página. Em caso de não ocorrer nenhum erro, o usuário é levado para a página principal.

Figura 17 – Página para adicionar tuplas.

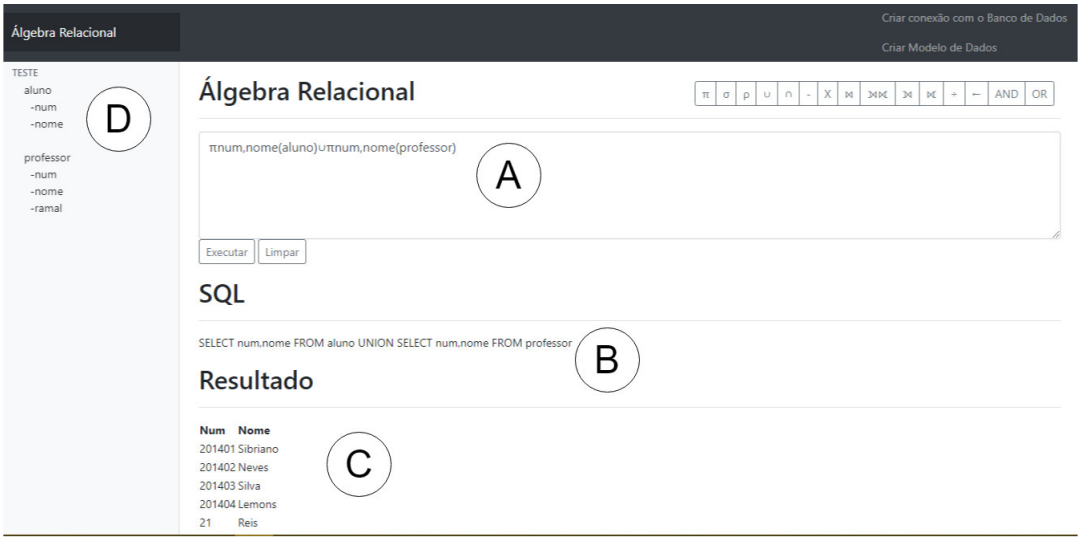


Fonte: Elaborada pelo Autor

Conforme os requisitos citados no item 3.1 no presente Capítulo, a ferramenta teria que permitir que o usuário pudesse realizar uma consulta em Álgebra Relacional, suportar os operadores relacionais, realizar uma análise léxica e sintática do que foi escrito pelo usuário, listar as relações de schemas do banco em que está conectado, traduzir a consulta feita em Álgebra Relacional para SQL e apresentar esse SQL, executar e apresentar o resultado dos comandos SQL gerados nos bancos de dados.

Para isto, como demonstrado na Figura 18, foi feita uma página em que o usuário pudesse escrever uma consulta em Álgebra Relacional, como mostrado na posição A, e utilizar os operadores relacionas, em D podemos ver que é listado as relações e suas respectivas colunas, de modo que o usuário terá uma visualização do que poderá usar. Ao ser executada uma consulta, esta é analisada pelo analisador léxico e sintático, para saber se está léxica e sintaticamente correta. Após passar pela análise léxica e sintática, a ferramenta é traduzida para SQL e o comando SQL resultante é exibido, como mostrado em B. O comando SQL, resultado da tradução da consulta, é executado e são mostrados os resultados deste comando, como mostrado em C.

Figura 18 – Página inicial da ferramenta.



Fonte: Elaborada pelo Autor

3.5 Comparação com os Trabalhos Relacionados

Devido aos trabalhos apresentados no item 1.2 do Capitulo 1 possuírem tecnologias diferentes entre si, é difícil compara-los por esse aspecto. Por tanto, nos ateremos aos aspectos de funcionalidades.

Em relação as operações de álgebra relacional, a ProgramAR é a única que não apresenta todas as operações, faltando as operações de Divisão, Intersecção e Diferença. Já o RelaX e o RAT possuem todas operações disponível para uso. No FEAR, todas essas operações foram implementadas.

Na questão da avaliação léxica e sintática, todas possuem uma forma de analisar a consulta feita pelo usuário. O FEAR também possui um analisador para consulta, este foi feito através da GALS.

Em relação ao SQL, RelaX é única a não mostrar um comando SQL feito a partir de uma consulta em álgebra relacional. Já o ProgramAR e o RAT exibem um SQL, no FEAR também é exibido um comando SQL equivalente a consulta.

Apesar de não mostrar um comando SQL, o RelaX exibe um resultado equivalente a consulta feita, já o RAT, apesar de exibir o SQL não mostra o resultado. ProgramAR apresenta tanto o SQL quanto o resultado. Da mesma forma que ProgramAR, o FEAR também o SQL e o resultado.

No que diz respeito a interface, o RAT e o ProgramAR são bem intuitivas, de maneira que ao utilizar pela primeira vez, um usuário não terá problemas para usa-las, já o RelaX é pouco intuitiva. O FEAR foi desenvolvido de maneira a ser o mais intuitivo possível, não gerando problemas para o usuário utiliza-lo.

No que se refere a ser uma ferramenta web, o RelaX é a única que é web. Neste contexto, o FEAR também é web.

Como pode-se observar na Figura 19, a ferramenta construída FEAR, engloba todas as variáveis foram citadas no item 1.2.4 no Capitulo 1.

Figura 19 – Tabela de comparação com Trabalhos Relacionados.

Fonte: Elaborada pelo Autor

3.6 Considerações finais

Nesse capitulo foram descritas as etapas percorridas para a implementação da FEAR, no próximo capitulo faremos as considerações finais e falaremos dos trabalhos futuros.

4 Conclusão

Este trabalho propôs desenvolver uma ferramenta web para o auxílio no ensino da Álgebra Relacional, usando as tecnologias GALS para a construção de uma analisador léxico e sintático, o Emscripten para compilar o código do analisador gerado e o WebAssembly como linguagem alvo para que o analisador seja utilizado na web.

Para tanto, fez-se um levantamento das ferramentas existentes, onde se observou que nenhuma delas incorporavam todos os aspectos desejados: possuir todas as operações de Álgebra Relacional, fazer uma avaliação léxica e sintática, exibir um SQL e um resultado equivalente à consulta feita, serem ambientes intuitivos e serem web.

No desenvolvimento da FEAR percorreram-se as etapas de Definição de Requisitos, construção de um analisador léxico e sintático, compilação deste analisador e a implementação da ferramenta.

Em relação à questão da análise léxica e sintática, que é de suma importância para a solução, foram procurados geradores automáticos de analisadores léxicos e analisadores sintáticos. Como base no estudo feito, a GALS mostrou-se ser a melhor opção, pois gera ambos os analisadores e possui interface gráfica.

Após a construção do analisador léxico e sintático, considerando todos os aspectos de cada operação da Álgebra Relacional, foi necessário encontrar uma maneira para este analisador ser usado no ambiente web. Para tanto, usamos o compilador Emscripten, que compila uma linguagem de alto nível para web, como linguagem alvo utilizamos a WebAssembly, que é uma linguagem binária que trabalha em conjunto com outras linguagens web. O código em C++, contendo o analisador desenvolvido em GALS, foi compilado no Emscripten para WebAssembly e JavaScript, este último sendo a linguagem que trabalha em conjunto com o WebAssembly.

Como dificuldades encontradas durante o desenvolvimento da ferramenta temos a construção da gramática na GALS em que fossem incluídas todas as regras de Álgebra Relacional, a manipulação do código no Emscripten e na integração dos códigos gerados na solução, a tradução de Álgebra Relacional para SQL e a construção das páginas onde o usuário pudesse criar modelos de dados de maneira gráfica.

Como sugestão de trabalhos futuros temos a integração da possibilidade de fazer consultas em Cálculo Relacional e em SQL, assim o usuário poderá realizar consultas nas três linguagens aprendidas na disciplina de Banco de Dados; que a cada novo caractere inserido, faça-se uma análise de tudo que foi escrito até aquele momento, para que o usuário possa visualizar se há algum erro antes de executar o comando; caso tenha algum

erro na consulta, que seja mostrado a posição dele ao usuário e colocar a ferramenta em teste em sala de aula, para que possam ser validadas questões de usabilidade, interface e performance.

Referências

AHO, A. V. et al. *Compiladores: princípios, técnicas e ferramentas*. [S.l.]: Pearson, 2007. v. 2. Citado 2 vezes nas páginas 22 e 26.

BRASSCOM. 2019. Disponível em: <https://brasscom.org.br/>. Acesso em: 10 de junho de 2019. Citado na página 10.

BROWN, D.; LEVINE, J. R. *Lex e yacc – second edtion*. [S.l.: s.n.], 1992. Citado na página 23.

CHIOFI, L. C.; OLIVEIRA, M. R. F. de. O uso das tecnologias educacionais como ferramentas didÁticas no processo de ensino e aprendizagem. *OS DESAFIOS DA ESCOLA PÚBLICA PARANAENSE NA PERSPECTIVA DO PROFESSOR PDE*, 2014. Citado na página 1.

DATE, C. *Introdução a Sistemas de Bancos de Dados*. [S.l.]: Editora Campus, 2004. v. 8. Citado na página 11.

EMSCRIPTEN. 2015. Disponível em: <https://emscripten.org/docs/introducing_emscripten/index.html>. Acesso em: 10 de junho de 2019. Citado na página 27.

FERNANDES, L. S.; RAABE, A. L. A.; BENITTI, F. B. V. Interface de software educacional: Desafios de design gráfico. *Congresso Brasileiro de Computação*, 2004. Citado na página 11.

FIALHO, N. N.; MATOS, E. L. M. A arte de envolver o aluno na aprendizagem de ciências utilizando softwares educacionais. *Educar em Revista*, 2010. Citado na página 1.

GESSER, C. E. Gals - gerador de analisadores léxicos e sintáticos. 2003. Citado 2 vezes nas páginas 24 e 25.

GLADCHEFF, A. P. Um instrumento de avaliação da qualidade para software educacional de matemática. 2001. Citado na página 11.

HUDSON, S. *CUP User's Manual*. 2013. Disponível em: <https://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html#changes>. Acesso em: 10 de junho de 2019. Citado na página 24.

KESSLER, J.; TSCHUGGNALL, M.; SPECHT, G. Relax: A webbased execution and learning tool for relational algebra. *Datenbanksysteme für Business, Technologie und Web*, 2019. Citado na página 5.

KORTH H.F. E SILBERSCHATZ, A. *Sistemas de Bancos de Dados*. [S.l.]: Editora Makron Books, 2006. v. 5. Citado 4 vezes nas páginas 11, 12, 20 e 21.

LOPES, A. G. Using research methods in human computer interaction to design technology for resilience. *Journal of Information Systems and Technology Management*, 2016. Citado na página 10.

MURILLO, J. V.; CHAVARRíA, S. B.; RIVERA, S. M. Herramienta asistida por computadora para la enseÑanza del Álgebra relacional en bases de datos. *UNICIENCIA*, 2012. Citado na página 7.

NAVATHE, S.; ELMASRI, R. *Sistemas de Bancos de Dados*. [S.l.]: Editora Addison-Wesley, 2011. v. 6. Citado 9 vezes nas páginas 1, 12, 13, 14, 15, 17, 18, 19 e 21.

NETO, J. C.; IMAMURA, M. M. Uma abordagem dos tipos de ferramentas computacionais utilizados para auxiliar o processo ensino-aprendizagem da matemática. 2012. Citado na página 10.

PAES, E. L. Ensinar: Ferramenta didÁtica para o ensino de Álgebra relacional. 2007. Citado na página 2.

PAXSON, V. *Flex – A scanner Generator*. 1998. Disponível em: <http://www.gnu.org/software/flex/>. Acesso em: 10 de junho de 2019. Citado na página 23.

PRATES, A. et al. Programar - ferramenta para auxiliar o ensino em Álgebra relacional. 2013. Citado na página 3.

SANTOS, A. M. D. Sistema para aprendizado de Álgebra relacional e linguagem sql. 2010. Citado na página 2.

SEIBERT, J. A. R. et al. Cuda-ra: Uma ferramenta de interpretação de álgebra relacional e estrutura de dados para gpu. 2016. Citado na página 2.

SILVA, E. G. da et al. Análise de ferramentas para o ensino de computação na educação básica. *Congresso da Sociedade Brasileira de Computação*, 2014. Citado na página 10.

WEBASSEMBLY. 2019. Disponível em: <https://webassembly.org/>. Acesso em: 10 de junho de 2019. Citado na página 27.