# APRENDIZADO E FUNCIONAMENTO DE BANCO DE DADOS: ValkyrieDB

Guilherme Luiz Simões Rigon[1]
Marcelo Arantes de Oliveira[2]
Lívia Ferreira Vidal[3]

**Resumo**

No mundo contemporâneo, a aprendizagem de modelagem de banco de dados exige do aluno ferramentas focadas no ensino para que ele possa conhecer conceitos fundamentais de alguma forma no que tange gerenciamento de dados. Enquanto softwares para estes fins são abundantes em aulas de programação, existe uma lacuna na área de banco de dados. Ferramentas de auxílio tendem a facilitar o aprendizado do aluno, trazendo mais uma alternativa para praticar o que foi ensinado em sala de aula. Foi observado que referente as linguagens de programação, as ferramentas existentes foram centralizadas no funcionamento de algum paradigma e em banco de dados podem ser focadas especificamente nos conceitos relacionados ao mesmo. Tendo isso em vista, o artigo tem como objetivo entender a necessidade do uso de ferramentas no ensino de banco de dados relacionais e, ao fim, desenvolver um Compilador que aceite comandos em português para que o aluno possa aprender conceitos de banco de dados em sua linguagem nativa para depois se especializar em uma tecnologia específica. A ferramenta é código aberto, totalmente desenvolvida em *Python*, utilizando como base, bibliografias consagradas e compiladores *Open Source*.

**Palavras-chave:** Banco de dados. Modelagem de dados. Compiladores. Ferramenta de aprendizado. *ValkyrieDB*.

# DATABASE LEARNING AND OPERATION: ValkyrieDB

**Abstract**

Currently, the learning of database modeling requires the student, tools focused on teaching, so that, in some way, he can know fundamental concepts regarding data management. While softwares for these purposes are plentiful in programming

[1]Bacharel em Sistemas de Informação pelo UGB/FERP.
[2]Especialista em Análise de Sistemas pela UNESA.
[3]Mestra em Ensino pelo Centro Universitário de Volta Redonda.

classes, there is a gap in the database area. Helping tools tend to facilitate student learning, bringing to them one more alternative to practice what was taught in the classroom. It was observed that referring to programming languages, the existing tools were focused on the functioning of some paradigm and in databases they can be focused specifically on concepts related to databases. With this in mind, this article aims to understand the need to use tools in the teaching of relational databases and, in the end, to develop a compiler that accepts commands in Portuguese so that the student can learn database concepts in their own language. native language and then specialize in a specific technology. The tool will be open source fully developed in python using established bibliographies and Open Source compilers as a base.

**Keywords:** Database. Data Modeling. Compilers. Learning Tools. ValkyrieDB.

## Introdução

Ferramentas de auxílio no aprendizado de programação são extremamente disseminadas, o mesmo não pode ser dito sobre o ensino de banco de dados. Grande parte das ferramentas voltadas para o ensino de programação tem como principal característica a sintaxe em português, isto acontece para criar um ambiente mais amigável ao estudante que só precisa focar em entender os conceitos e aplicá-los em sua linguagem nativa. Um grande exemplo de ferramenta para o ensino de programação é o Portugol Studio, IDE e compilador com foco no ensino e sintaxe totalmente em português que implementa o paradigma da programação estrutural.

De acordo com ESMIN (1998), programação não é algo difícil de aprender contanto que se domine a lógica de programação, nesse contexto a lógica de programação é o conceito a ser aprendido, em outras palavras, a forma geral de funcionamento do paradigma a ser ensinado. Sendo o SQL uma linguagem de programação com um paradigma de pesquisa declarativa, pode se dizer que o aluno que entende os conceitos do funcionamento do paradigma tende a ter facilidade em lidar com a linguagem SQL.

Sobre o ensino de programação HINTERHOLZ (2009) afirma que disciplinas de programação tem um alto índice de reprovação e/ou evasão, sendo estas disciplinas consideradas crivos durante o curso, sobre as causas ele continua:

> dificuldade de adaptação dos alunos desenvolverem raciocínio lógico quando estão acostumados a decorar o conteúdo; falta de motivação do aluno gerada pelo despreparo e o desânimo quando ele acredita que a disciplina constitui um obstáculo extremamente difícil de ser superado" (HINTERHOLZ, 2009)

Tendo isso em vista é possível assumir que as ferramentas além de focar em ser a ponte entre o aluno e os conceitos, uma vez que tenham êxito em facilitar o ensino do conceito ao aluno pode se afirmar que ferramentas tem grade potencial para diminuir a evasão dos alunos em razão de bloqueio ou receio de se aprofundar no tema após o primeiro contato como evidenciado por ESMIN (1998) e HINTERHOLTZ (2009). Além disso ferramentas podem tornar o ensino mais atrativo com resultados imediatos, dando ao aluno a sensação de recompensa imediata após aprender algo e conseguir visualizar na prática.

Existe uma divergência de opiniões quando o assunto é pseudolinguagens no ensino dos conceitos gerais de paradigmas de programação, alguns argumentam que pseudolinguagens ajudam a formular o raciocínio lógico e entender o que deve ser feito sem se preocupar com especificidades dos ambientes de produção ou traduzir algum comando especifico para que determinada linha comece a fazer sentido, outros por sua vez argumentam que traduzir uma linguagem não garante que o aluno terá facilidade de aprender na real dificultará ainda mais uma vez que foi criada uma nova barreira para o aprendizado, neste argumento se antes era necessário aprender somente lógica de programação para iniciar em uma linguagem agora é necessário aprender lógica e uma linguagem que só será usada durante o aprendizado.

Sobre o Portugol Studio D'AMATO (2021) diz: "procure um curso superior que valorize o seu investimento em educação, sem te fazer perder tempo à toa aprendendo uma pseudolinguagem sem aplicação prática".

Em contrapartida NOSCHANG el al. (2014) defende a ideia de que o ensino de programação introdutória deve ser focado na capacitação do aluno nos conceitos de programação, dessa forma, IDEs e linguagens profissionais acabam por criar uma barreira dificultando o aprendizado do aluno, em seu artigo, ele cita a alta diversidade de configurações de uma IDE profissional como um fator de dificuldade para o aluno, que em seu primeiro contato não precisa de um emaranhado de configurações extremamente especificas e sim das configurações essenciais, outro fator de dificuldade, principalmente no Brasil, é o inglês por conta da baixa fluência da língua no país, o que não seria um grande problema, porém, segundo a pesquisa levantada pelo Instituto Cultural British Council cerca de 5% da população brasileira sabe se comunicar em inglês, sendo 1% realmente fluente, o que significa que 95% do país ou sabe algumas palavras ou frases ou não sabe nada da língua inglesa, o que acaba por, de certa forma, elitizar o aprendizado em de linguagens de programação.

Outro ponto importantíssimo a se analisar é a experiência empírica uma vez que a mesma é essencial para que o estudante descubra onde e como deseja atuar, isto porque quando o aluno tem seu primeiro contato real, ele pode assumir com mais certeza se deseja atuar naquela área além de colocar em prova seu domínio nela, sendo assim o aluno só terá plena certeza de que quer ser um programador depois de programar algo.

Além disso, a experiência empírica no campo das ciências da computação se torna ainda mais importante uma vez que várias bibliografias são escritas com base nas experiências vividas pelo autor em determinado assunto, um exemplo é a obra "Refatoração" escrito por FOWLER (2020), que usa o livro como um grande registro de vários desafios e como reagiu a eles. Claro que a experiência empírica

não deve ser a única a ser considerada, mas esta tende a deixar o aluno mais confortável durante o aprendizado.

Complementando a ideia:

> O processo de desenvolvimento de software atual se baseia em boas práticas, experiências pessoais e observações gerais, porém apenas esses fatores não podem assegurar a qualidade e conformidade do artefato resultante. (MARCOLINO et al., 2013)

Tendo isso em vista pode se assumir que aulas de banco de dados com foco prático podem reter mais alunos nos cursos e aperfeiçoar o aprendizado do mesmo (não desconsiderando a teoria, mas apresentando-a paralelamente com algo prático), de acordo com ESMIN (1998) “A falta de prática desestimula os alunos a se aprofundar no conteúdo a ser aprendido”.

Por fim, escolher ferramentas para o auxílio do aprendizado pode ser uma faca de dois gumes, caso escolhida ou usada de forma inadequada a ferramenta pode atrapalhar ou atrasar o aprendizado. Por conta disso, muitos professores acabam por não usar. Além disso, como desenvolvido anteriormente, muitas pessoas acabam por achar certas ferramentas como perda de tempo, e professores podem estar inclusos nesses casos.

O que muitas pessoas esquecem é que o aluno não precisa se especializar no uso da ferramenta, ele deve se especializar no conceito que ela apresenta. Voltando aos ensaios de programação, se o aluno consegue entender e praticar os conceitos com a linguagem Python, ele não necessita fazer uso do Portugol, entretanto, caso o aluno encontre alguma dificuldade no aprendizado de tais conceitos por conta da barreira da linguagem, voltar a sua linguagem nativa para entendê-los e então retornar a linguagem de programação cria uma tendencia para que a evolução do aluno seja mais rápida, uma vez que exista a prática do conceito e da execução, tendencia essa descrita por ESMIN (1998) e NOSCHANG el al. (2014).

Com o que foi apresentado até o momento e tendo em vista a escassez de ferramentas que atendam a demanda das matérias relacionadas a banco de dados, o presente artigo tem como objetivo o desenvolvimento de um compilador que compreenda a linguagem SQL adaptada para o português, nomeada de PSQL (Portuguese SQL) sendo o produto nomeado ValkyrieDB.
Durante o desenvolvimento espera-se também

- Criar uma linguagem que facilite o entendimento dos conceitos pelo aluno, tendo como ideia de facilidade as argumentações de Noschang expostas na introdução;
- Manter a construção semântica dos comandos PSQL em conformidade com o que está sendo apresentado como resultado na tela;

Com relação a conceitos de banco de dados foi escolhido o livro Introdução a Sistemas de Bancos de Dados de 2004 escrito por Christopher J. Date, umas das pessoas mais influentes quando o assunto é banco de dados, este livro é de onde foram retirados os conceitos básicos de funcionamento de um banco de dados. A linguagem SQL é uma escolha necessária uma vez que o intuito do trabalho é facilitar o aprendizado dos conceitos para usá-la junto a um banco de dados SQL sigla para *Structured Query Language* ou Linguagem de Consulta Estruturada é a linguagem de pesquisa declarativa usada nos bancos de dados, a linguagem SQL tem 5 subconjuntos de comandos, sendo estes:

- **DML** (*Data Manipulation Language* ou Linguagem de Manipulação de Dados): Se trata dos comandos que executam ações que causam alguma alteração no conjunto de dados, sendo estes comandos o **INSERT**, **UPDATE** e **DELETE**, para inserção, atualização e deleção respectivamente;

- **DDL** (*Data Definition Language* ou Linguagem de Definição de Dados): Comandos que permitem manipular tabelas e elementos associados, no SQL clássico somente os comandos **CREATE** e **DROP** faziam parte deste subconjunto e a função de cada um é criar um objeto no banco e deletar um objeto do banco respectivamente, porém em alguns sistemas de banco de dados existe o comando ALTER que permite o usuário alterar um objeto no banco de dados;
- **DCL** (*Data Control Language* ou Linguagem de Controle de Dados): Comandos relacionados ao controle de acesso a informação do usuário no banco de dados, neste subconjunto existem dois comandos sendo eles **GRANT** e **REVOKE** que autoriza e restringe o usuário respectivamente;
- **DTL** (*Data Transaction Language* ou Linguagem de Transação de Dados): Permitem gerenciar transações ocorrendo no banco, transações nada mais são que alterações no disco que ainda não foram confirmadas, neste conjunto existem três comandos sendo eles **BEGIN TRAN** (ou **BEGIN TRANSACTION**), **COMMIT** e **ROLLBACK** que respectivamente delimita o início de uma transação, confirma todos os dados da transação no disco, desfaz as alterações feitas pela transação;
- **DQL** (*Data Query Language* ou Linguagem de Consulta de dados): Permite o usuário consultar dados no banco de dados tendo somente um comando conhecido por **SELECT** que permite fazer consultas no banco.

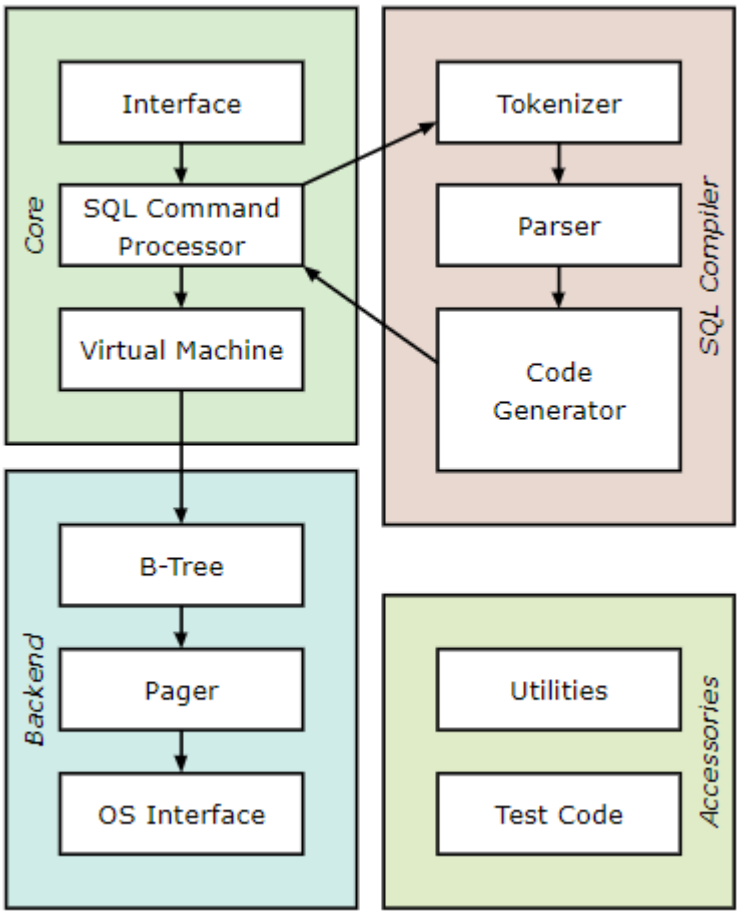
O *SQLite*, desenvolvido e lançado por Dwayne Richard Hipp em 17 de agosto de 2000, foi escolhido como modelo a ser seguido por se tratar de um banco de dados extremamente poderoso, utilizado em vários cenários, mas que não deixa de ser extremamente simples e direto ao ponto. O *SQLite* é código aberto o que facilita o entendimento do seu funcionamento em baixo nível.

A arquitetura do SQLite é composta por um *Core*, que pode ser entendido como o coração da aplicação, que divide o processamento dos comandos em

Entrada e Saída ou *Interface*, Compilação no *SQL Command Processor* e na camada *SQL Compiler* e manipulação do disco através da *backend*.

A arquitetura do *SQLite* é disponibilizada em seu site oficial e pode ser descrita a partir da Figura 1:

Figura 1. Arquitetura do Banco de Dados *SQLite*




Fonte: *SQLite* (2000)

Uma vez que o objetivo é o desenvolvimento de um compilador que compreenda o SQL adaptado para o português ou PSQL, e por sua natureza de ensino, considerando que após o entendimento dos conceitos o aluno queira migrar

os dados gravados no ValkyrieDB para uma ferramenta profissional, o compilador terá sua saída em SQL que será executado dentro do banco de dados SQLite.
Por conta disso o *Core* e o *SQL Compiler* serão os alvos do desenvolvimento, não havendo necessidade de pensar na camada *Backend*.

Com relação ao desenvolvimento do PSQL foi escolhido o livro Compiladores: Princípios, técnicas e ferramentas, mais especificamente sua 2° edição da editora Pearson escrito por AHO (2007). Um compilador é um programa que a partir de um código fonte em uma linguagem de programação X cria um programa semanticamente equivalente em uma linguagem Y, dessa forma é possível escrever instruções de um programa em uma linguagem de fácil interpretação humana, chamadas de linguagens de alto nível, enquanto o computador interpreta as mesmas instruções em uma linguagem de difícil interpretação humana, chamadas de linguagens de baixo nível. A compilação é dividida em seis fases, sendo elas:

- **Análise Léxica ou Tokenização**: É a primeira fase da compilação e se refere a leitura de um código fonte, cada conjunto de caracteres com algum significado nesta fase é denominado um lexema, o papel do analisador léxico é encontrar todos os lexemas do código fonte fornecido e classificar seu tipo de acordo com o contexto em que o mesmo está disposto no código, a junção do lexema e seu tipo é chamada de *token* e o resultado final desta fase é uma lista de *tokens*;
- **Análise Sintática**: É o processo que determina se uma lista de *tokens* fornecida pode gerar um programa válido, enquanto o analisador léxico está preocupado em obter os tokens que compõem o programa, o analisador sintático se preocupa em determinar se os tokens obtidos satisfazem as regras gramaticais da linguagem, as validações são feitas a partir autômatos finitos determinísticos, a saída desta fase é uma arvore de sintaxe que representa a estrutura sintática da lista de *tokens* fornecida;

- **Análise Semântica**: A partir da arvore de sintaxe e da tabela de símbolos está fase é responsável validar a coerência semântica do código fonte fornecido, se preocupando com o escopo das variáveis, tipos, declarações e toda validação que não foi possível realizar nas etapas anteriores;
- **Geração de Código Intermediário**: Nesta fase a arvore de sintaxe é transformada em uma representação intermediária do código, a principal diferença do código intermediário e do código final é a não especificação da máquina alvo, está fase possibilita certas otimizações no código de modo a deixar o código final mais eficiente;
- **Otimização de Código**: Esta fase usa o código intermediário para fazer algumas transformações no código a fim de se obter um código final melhor, um exemplo para facilitar a visualização desta fase seria imaginar uma função que arredonda o número para duas casas decimais sendo usada no número 3.14159, na otimização de código ele verificaria a possibilidade de trocar o valor da constante para 3.14 e retirar a execução da função do código final;
- **Geração do Código Final**: Através do código intermediário é mapeado e gerado o código final, com todas as especificações da máquina alvo, manipulações de memória e sistema operacional de destino, sendo essas especificações dependentes da linguagem de destino da compilação.

O desenvolvimento foi orientado pela metodologia de gerenciamento de projetos *Kanban*, para fazer uma ordem de prioridade de desenvolvimento eficiente, a metodologia *Kanban* se baseia na construção de um quadro de sinalização, para controlar os fluxos de produção de um determinado projeto. Através do *Kanban* foi possível montar uma fila de prioridade das tarefas além de possibilitar uma visão geral do que deveria ser desenvolvido no projeto.

Por fim, o ValkyrieDB foi desenvolvido utilizando a linguagem Python, desenvolvida e lançada por Guido van Rossum em 1991, por se tratar de uma

linguagem amplamente utilizada para o processamento de grandes massas de dados, além de ser uma linguagem extremamente prática e simples.

## Materiais e Métodos

O primeiro passo foi a construção de uma interface CLI (*Command-Line Interface* ou Interface de Linha de Comando), para facilitar a integração com editores de textos, após finalizar os preparativos do projeto foi iniciado a construção do compilador PSQL, um detalhe importante é que por se tratar de um MVP (*Minimum Viable Product*) ou Produto Minimamente Viável o escopo de comandos a serem implementados se delimitou em três dos cinco subconjuntos sendo eles: **DML** (*Data Manipulation Language*), **DDL** (*Data Definition Language*) e **DQL** (*Data Query Language*). Dessa forma, em linhas gerais, o ValkyrieDB fará o famoso CRUD, sigla para *CREATE*, *READ*, *UPDATE* e *DELETE*, ou seja criação, leitura alteração e deleção de dados.

Com relação ao compilador PSQL, seu desenvolvimento foi divido em quatro partes:

**Análise Léxica:** Momento em que foi desenvolvido um analisador léxico do compilador com o objetivo de gerar a lista de *tokens*.

A classificação dos lexemas, isto é, a determinação de seu tipo, é feita com base no contexto da palavra na sentença do comando, desta forma ao encontrar um lexema entre aspas considera-se que ele é um texto (*string*), se o lexema inicia com números tendo no máximo um separador de casas decimal considera-se um valor numérico (*number*), caso o lexema se encontre no arquivo de palavras chave, sua classificação será o próprio tipo e seu valor vazio, por fim se o lexema inicia com letras e não esteja entre aspas classificamos o mesmo como um identificador (id), que tem como objetivo identificar algum dado como uma coluna ou tabela.

Ao final de toda lista de tokens é colocado um token do tipo EOF (*End Of File*) que simboliza o fim do arquivo de código. Dessa forma quando executamos o comando:

DELETE DE Usuarios ONDE id = 1

Temos a seguinte lista de *tokens* (representados por <Tipo, Valor>):

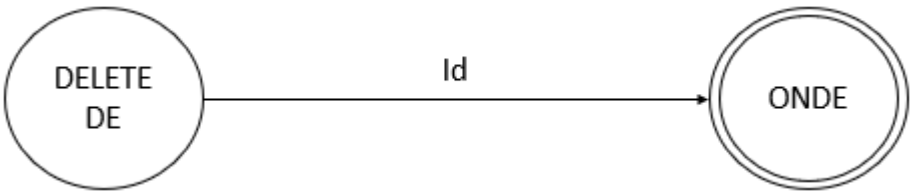
**<DELETE,>**,**<DE,>**,**<id, USUARIOS>**, **<ONDE,>**, **<id, ID>**, **<=,>**, **<Number, 1>**

## Análise Sintática

Como desenvolvido anteriormente, aqui é onde verificamos a validade da lista de *tokens* fornecida pelo analisador léxico, nesta etapa foi desenvolvido um autômato para cada comando, este autômato verifica a sintaxe e caso a sequência de tokens fornecida seja válida ele inicia o processo de geração de código final, é importante ressaltar que como o PSQL se trata de uma linguagem de tradução sem acesso aos metadados do disco, ele não consegue fazer a análise semântica dos *tokens* informados, sendo esta parte realizada pelo próprio SQLite.

O funcionamento geral dos autômatos se baseia em uma lista de tokens válidos para cada comando, dessa forma para o comando **DELETE** tem-se o autômato apresentado na Figura 2:

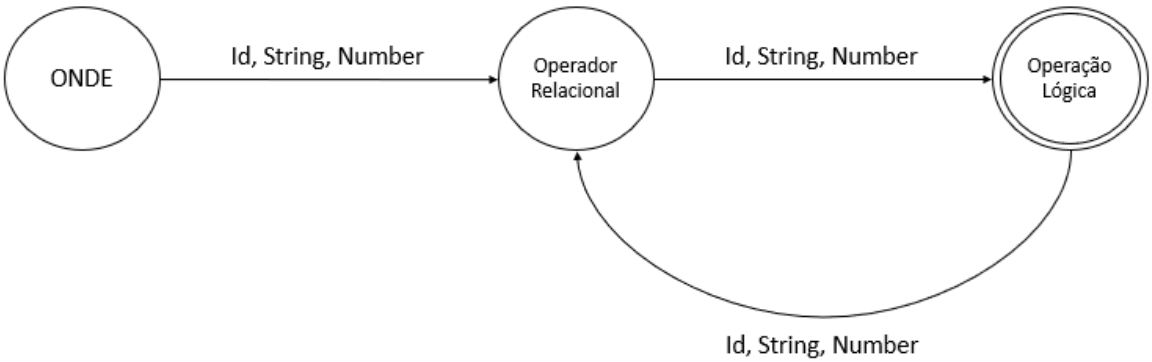
Figura 2. Autômato referente ao comando DELETE




Fonte: Pesquisa dos Autores.

O autômato do comando **DELETE** garante que após as palavras chaves **DELETE DE** o próximo *token* deve obrigatoriamente ser um identificador (id), além disso ele deixa opcional o uso da palavra-chave **ONDE** que tem seu autômato apresentado pela Figura 3:

Figura 3. Autômato referente a palavra-chave ONDE




Fonte: Pesquisa dos Autores.

Considera-se:

- Operadores Relacionais: **>**, **<**, **>=**, **<=**, **=** e **<>**.
- Operações Lógicas: **E**, **OU** e **NÃO**

A partir das verificações dos autômatos é possível fazer a verificação sintática da lista de *tokens* e iniciar o processo de geração de código SQL.

**Geração de Código Final**

Após passar pelas verificações do analisador sintático o compilador pega as cadeias de tokens e inicia o processo de geração de código através de equivalência de palavras-chave. Dessa forma a palavra-chave DELETE do PSQL é equivalente a palavra-chave DELETE do SQL, SELECIONE do PSQL equivale a SELECT do SQL e assim por diante.

## Execução do SQLite (Análise Semântica)

Trata-se da parte final do processo de compilação do PSQL na qual, caso não tenha encontrado erro algum, ele encaminha o comando em SQL para uma conexão SQLite, executa o comando e retorna as mensagens do que foi feito, retorno de consultas ou erros semânticos do SQLite.

Todos os comandos que passam pelos três passos de compilação do PSQL podem retornar três saídas, sendo elas:

- Número de linhas alteradas, no caso dos comandos DML;
- Uma mensagem de “Comando executado com sucesso!”, no caso dos comandos DDL;
- Uma tabela gerada pela biblioteca *tabulate* com o resultado de uma consulta, no caso dos comandos DQL;
- Por fim uma classe de erro que se caracteriza pelo início “[Erro do SQLite]...” representa os erros internos do SQLite como problemas durante a análise semântica. Exemplo: Uma tabela que não existe.

Dessa forma, o ValkyrieDB trabalha com manipulação de dados e tabelas, com ênfase nos comandos DML e DQL englobando seleções de múltiplas tabelas, condições e tokens do gênero.

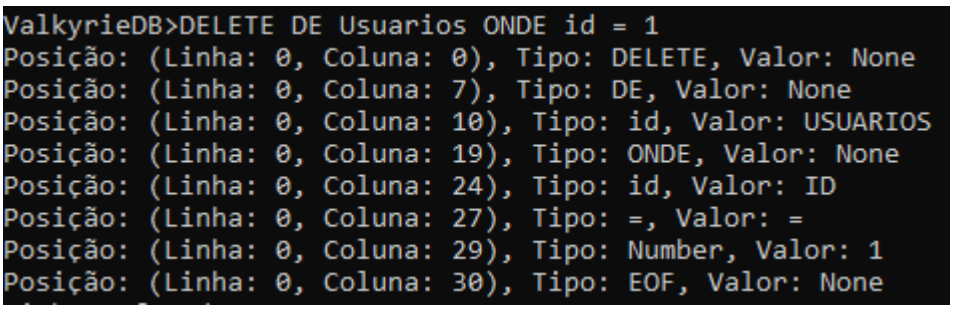
## Resultados

O resultado obtido durante o desenvolvimento do ValkyrieDB foi um software que entendesse a linguagem proposta, PSQL, de modo a compilar o PSQL em uma saída SQL e executar dentro do banco de dados SQLite. Cada um dos componentes do compilador PSQL atingiu seus próprios resultados. O analisador léxico foi construído de modo a retornar uma lista de tokens com o tipo, valor e posição do lexema, dessa forma a execução do comando:

DELETE DE Usuarios ONDE id = 1

Gera a seguinte lista de tokens:

Figura 4. Resultado da Análise Léxica




Fonte: Pesquisa dos Autores.

A posição foi adicionada na representação dos *tokens*, uma vez que ela tem extrema importância durante a análise sintática para a localização dos *tokens* não esperados. O analisador sintático foi construído de modo a interagir com a lista de *tokens* e a partir do primeiro *token* definir o comando que se deseja executar, definido o comando é chamado o autômato referente para realizar a análise sintática de fato. O processo de geração de código final é feito durante a execução do autômato e caso a sintaxe do comando em processamento não corresponda com gramática da linguagem a geração de código é pausada e um erro é mostrado ao usuário, caso contrário o código gerado é levado a fase de execução no SQLite. O autômato do comando **EXCLUA** (equivalente ao comando **DROP** do SQL) é representado pela Figura 5:

Figura 5. Autômato referente ao comando EXCLUA

```
def __drop(self, tokens, index = 0):
    iSintaxe = 0
    sintaxe = {
        1 : Keywords.EXCLUA,
        2 : Keywords.TABELA,
        3 : "id"
    }
    i = index
    command = ""
    if tokens[i].getType() == Keywords.EXCLUA:
        while i < len(tokens) and not self.__hasErros and iSintaxe < 3:
            iSintaxe += 1
            if tokens[i].getType() != sintaxe[iSintaxe]:
                thk = tokens[i]
                self.__showMessageError(f'[PSQL#22] Token `{Keywords.GetPSQL(thk.getValue())}` '
                + f'não esperado próximo a {thk.getPosition()}')
            if tokens[i].getType() != "EOF":
                command += f"{self.__writeToken(tokens[i])} "
            i += 1
    if not self.__hasErros:
        return (command, i)
    else:
        return ('', index)
```

Fonte: Pesquisa dos Autores.

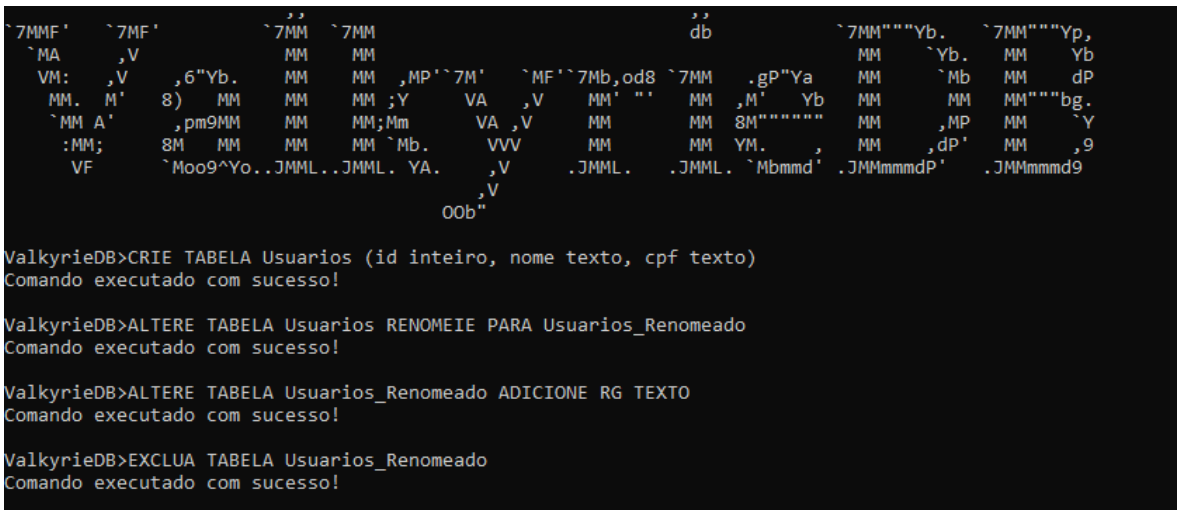
A seguir, a demonstração de todos os comandos disponíveis no PSQL, a Figura 6 demonstra o funcionamento dos comandos DQL (Linguagem de Consulta de Dados):

Figura 6. Execução dos comandos DQL


Fonte: Pesquisa dos Autores.

A Figura 7 demonstra o funcionamento e os resultados dos comandos DDL (Linguagem de Definição de Dados) adaptados para o PSQL:

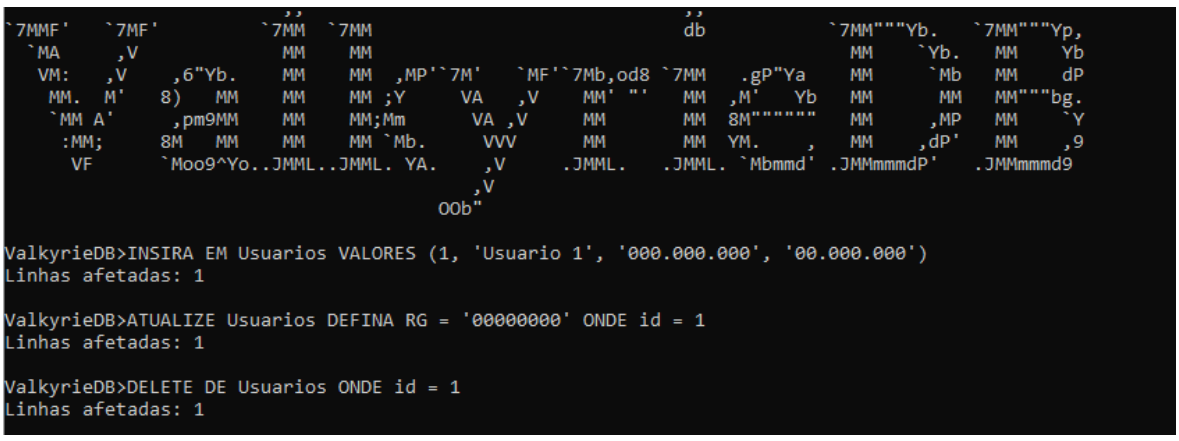
Figura 7. Execução dos comandos DDL




Fonte: Pesquisa dos Autores

A Figura 8 demonstra os comandos e resultados do conjunto de instruções DML (Linguagem de Manipulação de Dados)

Figura 8. Execução dos comandos DML




Fonte: Pesquisa do Autor

Analisando os objetivos específicos, é possível dizer que a linguagem PSQL implementa as facilitações evidenciadas por Noschang no sentido da barreira linguística. Avaliando a construção semântica dos comandos, é possível afirmar que a leitura do PSQL para um brasileiro é a mesma leitura de um comando SQL para um norte americano, uma vez que a construção semântica dos comandos se manteve em conformidade que o SQL, de modo que o comando que está sendo executado possa ser lido como uma frase imperativa dando ao usuário a sensação de estar dando ordens para a máquina.

Por fim, manipulando o arquivo de palavras chaves é possível adaptar o ValkyrieDB para qualquer linguagem contanto que o usuário que modifique conheça a linguagem para renomear as palavras chaves a fim de manter a construção semântica das frases.

## Discussão

Durante o desenvolvimento do software, foram percebidas algumas dificuldades, sendo a principal a não manipulação do disco por parte do ValkyrieDB, pois uma vez que o software não manipula o disco, este não pode iniciar um processo de análise semântica, uma vez que para analisar a semântica do comando é necessário acessar previamente o disco para saber se determinadas tabelas ou campos existem ou não, por conta disso está fase é realizada pelo próprio SQLite.

Por mais que o ValkyrieDB implemente ideias defendidas anteriormente na introdução, não é possível afirmar sua eficácia uma vez que não foi feito um teste de campo. Segue como sugestão um teste com grupos de alunos, onde dois grupos de mesma quantidade de alunos são ensinados pelo mesmo professor, no primeiro grupo o professor utiliza sua metodologia sem alteração e no segundo ele altera sua metodologia a fim de incluir o ValkyrieDB como suporte ao ensino. Ao final de

seis meses analisar qual turma teve o melhor rendimento tendo como principal objetivo avaliar os conceitos acerca da modelagem de dados prática.

Outro ponto a ser levantado é com relação ao suporte ao ensino uma vez que o aluno não precisa se especializar na ferramenta e sim no conceito que ela apresenta, o aluno que compreende os conceitos deve fazer a transição para uma ferramenta profissional o mais rápido possível.

É esperado que o ValkyrieDB se comporte como um facilitador, uma ponte entre os conceitos e o aluno, e que assim que o aluno se sentir confortável ele possa fazer a transição para um *software* de banco de dados profissional. O tempo sugerido para a apresentação dos conceitos é de duas semanas, após isso somente alunos que sintam dificuldades acerca dos conceitos devem utilizar a ferramenta, de modo a resolver exercícios ou desafios criados em sala, a fim de entender os conceitos e fazer a transição para uma ferramenta profissional.

## Considerações Finais

Por fim, o ValkyrieDB, em uma versão futura, poderia ter um desenvolvimento de uma camada de *backend* própria, dessa forma deixando de ser uma interface com um compilador e se tornando um SGBD completo, isso traria algumas vantagens com relação ao processo de compilação como a possibilidade da análise semântica e a melhor especificação de mensagens de erro.

Outro ponto que poderia ser desenvolvido em versões futuras seria uma IDE que auxiliasse o aluno durante o desenvolvimento, completando os comandos e fazendo sugestões inteligentes para a construção do script.

Com relação aos objetivos do artigo que visavam a facilitação do aprendizado dos conceitos por parte do aluno, o ValkyrieDB consegue implementar bem as teorias propostas pelos autores desenvolvidos na introdução, porém como não houve um teste de campo, não é possível comprovar a eficácia do ValkyrieDB

dentro das salas de aula, só é possível dizer que ele atende as ideias propostas pelos autores desenvolvidos ficando para um futuro teste dizer se o software atende ou não as expectativas esperadas.

## Referências


ARCHITECTURE of SQLite. 2000. Disponível em: https://www.sqlite.org/arch.html. Acesso em: 8 nov. 2021.

AHO, Alfred Vaino; LAM, Monica *Sin-Ling*; SETHI, Ravi; ULLMAN, Jeffrey David. **COMPILADORES**: Princípios, Técnicas e Ferramentas. 2. ed. 2007. 648 p. ISBN 8588639246.

D'AMATO; Guilherme Brügger. Por que as Universidades Brasileiras Ainda Ensinam Portugol em 2021? Disponível em: https://vaiprogramar.com/porque-universidades-brasileiras-ensinam-portugol/. Acesso em: 10 set. 2021.

DATE, Christopher J. **Introdução a Sistemas de Bancos de Dados**. 1. ed. GEN LTC, 2004. ISBN 8535212736.

ESMIN, Ahmed Ali Abdalla. PORTUGOL/PLUS: UMA FERRAMENTA DE APOIO AO ENSINO DE LÓGICA DE PROGRAMAÇÃO BASEADO NO PORTUGOL. **IV Congresso RIBIE, Brasilia 1998**, 1998. Disponível em: http://www.ufrgs.br/niee/eventos/RIBIE/1998/pdf/com_pos_dem/118.pdf. Acesso em: 7 nov. 2021.

FOWLER, Martin. **Refatoração**: Aperfeiçoando o Design de Códigos Existentes. 2. ed. Novatec Editora, 2020. 456 p. ISBN 8575227246.

HINTERHOLZ JR, Ornélio. Tepequém: uma nova Ferramenta para o Ensino de Algoritmos nos Cursos Superiores em Computação. **XVII-Anais do Workshop sobre Educação em Informática**, v. 20, ed. 21, 2009. Disponível em: http://csbc2009.inf.ufrgs.br/anais/pdf/wei/st02_04.pdf. Acesso em: 7 maio 2022.

MARCOLINO, Anderson Da Silva *et al*. A IMPORTÂNCIA DA EXPERIMENTAÇÃO EMPÍRICA NA PESQUISA EM CIÊNCIA DA COMPUTAÇÃO. ***Journal of Exact Sciences***, ano 2014, v. 3, n. 1, p. 21-27, 8 nov. 2013. Disponível em: https://www.mastereditora.com.br/periodico/20141123_163016.pdf. Acesso em: 7 nov. 2021.

NOSCHANG, Luiz F. *el al*. Uma ide para iniciantes em programação. **Anais do XXII *Workshop* sobre Educação em Computação**, SBC, p. 1-10, 2014. Disponível em: https://sol.sbc.org.br/index.php/wei/article/view/10954/10824. Acesso em: 16 mar. 2022.

PORTUGOL Studio. 7 jul. 2009. Disponível em: http://lite.acad.univali.br/portugol/. Acesso em: 10 set. 2021.