FACULDADE DE TECNOLOGIA DE BAURU

TECNOLOGIA EM BANCO DE DADOS

Titulo do trabalho de pesquisa

Equipe:
Sandro César Fernandes

Bauru/SP
2022

Titulo do trabalho de pesquisa

Equipe:
Sandro César Fernandes

Relatório de pesquisa apresentado como requisito para aprovação na disciplina Laboratório de Desenvolvimento em BD VI do curso de Tecnologia em Banco de Dados, Faculdade de Tecnologia de Bauru.

Prof. Lucas Chasseraux Tauil

Bauru/SP
2022

Sumário

1. INTRODUÇÃO

A gestão eficiente de dados desempenha um papel essencial em todos os setores, e a área de segurança rodoviária não é exceção. Neste contexto, a conversão de dados armazenados em um banco de dados relacional para um banco de dados não relacional é uma tarefa de grande relevância. Este trabalho tem como objetivo explorar a complexa e crucial transição de dados fictícios relativos a acidentes ocorridos em vias estaduais de um sistema de gerenciamento de banco de dados tradicional para uma estrutura não relacional.

A coleta e análise de dados sobre acidentes rodoviários são fundamentais para a prevenção e gestão eficaz da segurança viária. No entanto, com a crescente quantidade de informações sendo geradas em tempo real, os sistemas de banco de dados tradicionais muitas vezes enfrentam desafios em termos de escalabilidade, flexibilidade e velocidade de processamento necessários para lidar com esses dados de maneira eficiente. É nesse contexto que surge a necessidade de migrar para uma abordagem não relacional.

2. OBJETIVOS

A conversão de dados de banco de dados relacionais para não relacionais é uma tarefa que constantemente é realizada por diversos fatores pois são amplamente conhecidos pela sua escalabilidade horizontal, sua modelagem de dados flexível, seu desempenho com leitura e gravação, são ideais para lidar com dados semiestruturados e não estruturados, tem redundância e tolerância a falhas, baixo custo de manutenção, o que o faz a escolha perfeita para implementação de aplicação web, entre outras. Este trabalho examinará os motivos para a migração, os desafios associados à conversão de dados e como a adoção de um banco de dados não relacional pode proporcionar uma melhor capacidade de lidar com grandes volumes de dados de acidentes rodoviários, permitindo análises mais precisas e aprimoramento das estratégias de segurança nas vias estaduais. Ao longo deste estudo, serão exploradas as etapas do processo de conversão, as vantagens e desvantagens das abordagens envolvidas e como essa mudança impacta a eficácia do gerenciamento de dados e, por fim, a segurança rodoviária.

3. MATERIAIS E MÉTODOS

Foi escolhida, além do ***SQL*** para a elaboração do trabalho, a linguagem ***Python*** como ferramenta para coleta e conversão de dados, tendo em vista que dados relativos a acidentes de trânsito onde é exposto o emplacamento do veículo, torna-se dado sensível e de uso exclusivo do proprietário ou da corporação, para extração de dados estatísticos.

O banco de dados selecionado por conveniência é o ***Sqlite***, nativo do ***Python***, oferece flexibilidade e compatibilidade com outras grandes ferramentas de banco de dados, como o ***MySQL***. Conforme (SQLite, 2023) ***SQLite*** é um banco de dados relacional que se destaca por sua simplicidade e eficiência. Ele é frequentemente utilizado em aplicativos de software devido à sua facilidade de incorporação e ao fato de ser uma solução "embutida", o que significa que não requer um servidor de banco de dados separado para funcionar.
Usando a linguagem SQL (*Structured Query Language*), os desenvolvedores podem criar, consultar, atualizar e excluir dados em bancos de dados SQLite de maneira semelhante a outros sistemas de gerenciamento de banco de dados relacionais. Uma característica importante do SQLite é sua capacidade de manter a integridade dos dados, seguindo os princípios das transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Isso significa que as operações de banco de dados são confiáveis mesmo em situações de falha.

Além disso, o SQLite é conhecido por sua eficiência em termos de uso de recursos, tornando-o uma escolha popular para aplicativos que precisam de alto desempenho em operações de leitura e gravação. Ele está disponível em várias plataformas, incluindo Windows, macOS, Linux e sistemas operacionais móveis como Android e iOS. Devido à sua simplicidade, versatilidade e licença de código aberto (domínio público), o SQLite é amplamente utilizado em uma variedade de aplicativos, desde aplicativos móveis simples até navegadores da web populares como o Google Chrome. Sua ampla disponibilidade e facilidade de uso o tornam uma opção valiosa para desenvolvedores que desejam armazenar e gerenciar dados de forma confiável em seus aplicativos.

Para o armazenamento após a conversão, escolhemos o ***MongoDB,*** um banco de dados NoSQL altamente popular e amplamente utilizado, conhecido por suas várias vantagens. Uma das principais vantagens do ***MongoDB*** é sua capacidade de armazenar dados flexíveis e não estruturados, o que o torna uma escolha ideal para aplicativos que lidam com dados variáveis, como redes sociais e aplicativos de IoT. Além disso, o ***MongoDB*** oferece escalabilidade horizontal eficiente, permitindo que os desenvolvedores adicionem servidores facilmente à medida que o tráfego do aplicativo cresce, garantindo assim um desempenho confiável e rápido.

Outra vantagem notável é a capacidade do ***MongoDB*** de lidar com grandes volumes de dados, o que o torna adequado para aplicativos com demandas de alta carga de trabalho. Sua arquitetura distribuída e a capacidade de dimensionar horizontalmente garantem que os aplicativos possam crescer de maneira sustentável ao longo do tempo. Além disso, o ***MongoDB*** oferece uma flexibilidade excepcional na modelagem de dados, permitindo que os desenvolvedores ajustem rapidamente o esquema conforme as necessidades do aplicativo evoluem. Além disso, o ***MongoDB*** oferece uma ampla gama de recursos de consulta e indexação, tornando-o eficiente na recuperação de dados e permitindo consultas complexas. Com sua comunidade ativa e documentação extensa, o ***MongoDB*** é uma escolha sólida para desenvolvedores que desejam uma solução de banco de dados flexível, escalável e de alto desempenho para seus aplicativos modernos.

Para elaboração do MER/DER foi utilizado o aplicativo gratuito ***Dia,*** um software de código aberto usado para criar diagramas e gráficos técnicos, como diagramas de fluxo, organogramas e esquemas. Ele oferece uma interface intuitiva e recursos para ajudar na criação de representações visuais de informações e processos. O ***Dia*** é amplamente utilizado em engenharia, ciência da computação e outras áreas para criar diagramas de forma simples e eficaz.

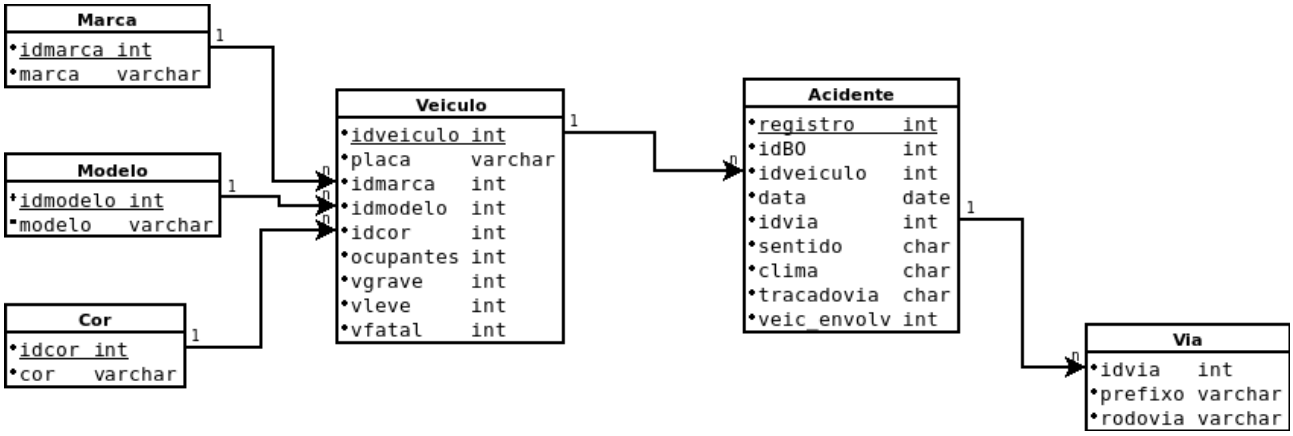

Figura 1: Diagrama do Banco de dados relacional. Fonte: o autor

Toda a programação foi realizada no aplicativo ***VSCODE***, um ambiente de desenvolvimento integrado (IDE) altamente popular, amplamente utilizado por desenvolvedores de software em todo o mundo. Este IDE de código aberto oferece uma experiência de desenvolvimento poderosa e versátil, adequada para uma variedade de linguagens de programação e plataformas. O ***VSCode*** está disponível gratuitamente e é compatível com sistemas operacionais *Windows, macOS* e *Linux*, garantindo acessibilidade a um amplo público de desenvolvedores.

3.1 CRIAÇÃO DO BANCO DE DADOS

```
CREATE TABLE MARCA
(
        IDMARCA INTEGER PRIMARY KEY AUTOINCREMENT,
        MARCA VARCHAR(20)
);

CREATE TABLE MODELO
(
        IDMODELO INTEGER PRIMARY KEY AUTOINCREMENT,
        MODELO VARCHAR(20)
);

CREATE TABLE COR
(
        IDCOR INTEGER PRIMARY KEY AUTOINCREMENT,
        COR VARCHAR(20)
```

```
);

CREATE TABLE VEICULO (
   IDVEICULO INTEGER PRIMARY KEY AUTOINCREMENT,
   PLACA VARCHAR(7) NOT NULL,
   IDMARCA INTEGER NOT NULL,
   IDMODELO INTEGER NOT NULL,
   IDCOR INTEGER NOT NULL,
   OCUPANTES INTEGER NOT NULL,
   VGRAVE INTEGER NOT NULL,
   VLEVE INTEGER NOT NULL,
   VFATAL INTEGER NOT NULL,
   FOREIGN KEY (IDMARCA) REFERENCES MARCA(IDMARCA),
   FOREIGN KEY (IDMODELO) REFERENCES MODELO(IDMODELO),
   FOREIGN KEY (IDCOR) REFERENCES COR(IDCOR)
);

CREATE TABLE VIA
(
        IDVIA INTEGER PRIMARY KEY AUTOINCREMENT,
        PREFIXO VARCHAR(5),
        RODOVIA VARCHAR(200)
);

CREATE TABLE ACIDENTE
(
        REGISTRO INTEGER PRIMARY KEY AUTOINCREMENT,
        IDBO INTEGER NOT NULL,
        IDVEICULO INTEGER NOT NULL,
        DATA DATE,
        IDVIA INTEGER NOT NULL,
        SENTIDO CHAR(1) NOT NULL,
        CLIMA CHAR(1) NOT NULL,
        TRACADOVIA CHAR(1)NOT NULL,
        VEICENVOLVIDO INTEGER NOT NULL,
        FOREIGN KEY (IDVIA) REFERENCES VIA(IDVIA),
```

```
        FOREIGN KEY (IDVEICULO) REFERENCES VEICULO(IDVEICULO),
        CHECK (SENTIDO IN('N','S','L','O')),
        CHECK (CLIMA IN('C','N','S')),
        CHECK (TRACADOVIA IN('R','C','A','D'))
);
```

O banco de dados foi populado com dados fictícios o suficiente para realização do trabalho e visualização de resultados, por isso deixo de inserir tal procedimento.
A conversão dos dados poderia ter sido realizada em sites específicos, disponíveis de forma gratuita na internet, porém, com dados sensíveis e a insegurança que a internet nos trás não é lógico que tal conversão se dê de outra forma sem ser via código. Para tal, foi desenvolvido o simples código em ***Python*** que realiza a tarefa de converter e enviar a um banco de dados não relacional, aqui utilizado o ***MongoDB***.

```
##############################################################################
# Alunos: ERICK CÉSAR DOMINGOS BARBOSA
#         SANDRO CESAR FERNANDES
# pesquisa.py - converte base de dados relacional em não relacional
##############################################################################


import sqlite3
from pymongo import MongoClient


cliente = MongoClient("localhost", 27017)
db = cliente["LabVI"]
colecao = db["Acidentes"]


conn = sqlite3.connect('/home/sandro/Documentos/Projetos/Lab VI/BD/Untitled')
cursor = conn.cursor()
query = """SELECT
V.PLACA,
M.MARCA,
MD.MODELO,
C.COR,
V.OCUPANTES,
V.VGRAVE,
```

```
V.VLEVE,
V.VFATAL,
A.IDBO,
A.DATA,
VI.RODOVIA,
A.IDVIA,
A.SENTIDO,
A.CLIMA,
A.TRACADOVIA,
A.VEICENVOLVIDO
FROM VEICULO V
INNER JOIN ACIDENTE A ON V.IDVEICULO = A.IDVEICULO
INNER JOIN MARCA M ON V.IDMARCA = M.IDMARCA
INNER JOIN MODELO MD ON V.IDMODELO = MD.IDMODELO
INNER JOIN COR C ON V.IDCOR = C.IDCOR
INNER JOIN VIA VI ON A.IDVIA = VI.IDVIA;
"""

cursor.execute(query)
documentos = []

for linha in cursor.fetchall():
documento = {
"PLACA": linha[0],
"MARCA": linha[1],
"MODELO": linha[2],
"COR": linha[3],
"OCUPANTES": linha[4],
"VITIMA_GRAVE": linha[5],
"VITIMA_LEVE": linha[6],
"VITIMA_FATAL": linha[7],
"BOLETIM": linha[8],
```

"DATA": linha[9],

"RODOVIA": linha[10],

"CLIMA": linha[13],

"TRAÇADO_VIA": linha[14],

"VEICULOS_ENVOLVIDOS": linha[15],

}

documentos.append(documento)

colecao.insert_many(documentos)

conn.close()

4. RESULTADOS E DISCUSSÃO

Nesse item é feita a apresentação dos resultados da pesquisa, fundamentados teoricamente. Os resultados obtidos são apresentados de forma objetiva e lógica, acompanhados de uma análise crítica dos mesmos. Devem ser apresentados na forma de tabelas e gráficos. São comparados com o que era esperado (com base na teoria descrita na introdução) e com resultados de outros experimentos já publicados na área. Se os resultados diferem do que era esperado, na discussão deve-se procurar explicar o motivo. Ilustrar com fotos, quando possível.

1. CONCLUSÕES

Síntese pessoal (do grupo) sobre as conclusões alcançadas com o trabalho. Enumere os resultados mais significativos do trabalho.

2. REFERÊNCIAS

SQLite, 2023. SQLTeam. https://www.sqlite.org, 2023, Disponível em: <https://www.sqlite.org/about.html>. Acesso: 08 set. 2023.