Apresentação da Solução de Banco de Dados

1. Descrição da Solução Adotada

Este projeto implementa uma solução de banco de dados para coletar, armazenar e visualizar informações sobre empresas de desenvolvimento de jogos (especificamente FromSoftware e Bethesda Game Studios), seus jogos, expansões, plataformas e publicações. A solução envolve:

- **Web Scraping**: Coleta de dados de páginas da Wikipedia.
- **Transformação de Dados**: Processamento e formatação dos dados coletados.
- **Banco de Dados MySQL**: Armazenamento estruturado das informações.
- **Análise e Visualização**: Geração de insights e gráficos a partir dos dados armazenados.

2. Esquema do Banco de Dados

O banco de dados `fromsoftware_db` é composto pelas seguintes tabelas e seus relacionamentos:

```
 DROP DATABASE IF EXISTS fromsoftware_db;
CREATE DATABASE fromsoftware_db;
USE fromsoftware_db;

CREATE TABLE atividEmp (
    atividade VARCHAR(255) PRIMARY KEY
);

CREATE TABLE tipoEmp(
    tipo VARCHAR(255) PRIMARY KEY
);

CREATE TABLE empresa (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    fundacao DATE,
    sede VARCHAR(255),
    head VARCHAR(255), -- Pessoa no comando
    atividade VARCHAR(255),
    tipo VARCHAR(255),
```

```
    numEmpregados INT,
    FOREIGN KEY(atividade) REFERENCES atividEmp(atividade),
    FOREIGN KEY(tipo) REFERENCES tipoEmp(tipo)
);

CREATE TABLE jogo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    idEmpDev INT NOT NULL,
    FOREIGN KEY(idEmpDev) REFERENCES empresa(id)
);

CREATE TABLE publisher(
    idPublisher INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255)
);

CREATE TABLE publicacoes(
    idJogo INT,
    idPublisher INT,
    ano YEAR,
    FOREIGN KEY(idJogo) REFERENCES jogo(id),
    FOREIGN KEY(idPublisher) REFERENCES publisher(idPublisher),
    PRIMARY KEY(idJogo, idPublisher)
);

CREATE TABLE plataforma(
    idPlataforma INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255)
);

CREATE TABLE compatibilidade(
    idJogo INT,
    idPlataforma INT,
    FOREIGN KEY(idJogo) REFERENCES jogo(id),
    FOREIGN KEY(idPlataforma) REFERENCES
plataforma(idPlataforma),
    PRIMARY KEY(idJogo, idPlataforma)
);

CREATE TABLE expansao(
    idExpansao INT AUTO_INCREMENT,
    idJogo INT,
    nome VARCHAR(255),
    ano_lancamento YEAR,
    FOREIGN KEY(idJogo) REFERENCES jogo(id),
    PRIMARY KEY(idExpansao)
);
```

3. Esquema Gráfico para o Modelo ER

O modelo Entidade-Relacionamento (ER) representa visualmente a estrutura do banco de dados e os relacionamentos entre as entidades. O arquivo `modelo_bd.brM3` contém o modelo ER, que pode ser visualizado com a ferramenta MySQL Workbench.

4. Execução do Processo de Criação e Alimentação do Banco

O processo de criação e alimentação do banco de dados é orquestrado pelo script `main.py` e utiliza os seguintes componentes:

- **`fromsoftware_db.sql`**: Script SQL para criar o esquema do banco de dados.
- **`webscraper_pd.py`**: Módulo responsável por realizar o web scraping das páginas da Wikipedia e transformar os dados em DataFrames do Pandas.
- **`main.py`**: Script principal que coordena a execução do web scraping, a inserção dos dados no banco de dados e a chamada para as análises e visualizações.

Para executar o processo:

1. Certifique-se de ter um servidor MySQL em execução e as credenciais configuradas corretamente no `main.py` e `database.py`.

2. Execute o script `main.py`:

   `bash python3 main.py`

   Este script irá: * Verificar e criar o banco de dados `fromsoftware_db`. * Raspar dados das páginas da Wikipedia para FromSoftware e Bethesda Game Studios. * Inserir os dados das empresas, jogos, publishers, plataformas e expansões nas tabelas correspondentes.

5. Execução das Consultas SQL que Geraram os Gráficos

As visualizações de dados são geradas a partir de consultas SQL executadas no banco de dados. As principais consultas utilizadas para gerar os gráficos são:

```sql
/* SQL Queries from data_visualization.py */

-- Query 1: Companies by country
SELECT country_of_origin, COUNT(*) AS company_count
```

```sql
FROM companies
GROUP BY country_of_origin
ORDER BY company_count DESC
LIMIT 10;

-- Query 2: Products per company
SELECT c.name, COUNT(p.id) AS product_count
FROM products p
JOIN companies c ON p.company_id = c.id
GROUP BY c.name
ORDER BY product_count DESC
LIMIT 10;

-- Query 3: Product launches per year
SELECT release_year, COUNT(*) AS launch_count
FROM products
GROUP BY release_year
ORDER BY release_year;


/* SQL Queries from data_analysis.py */

-- Query for product expansion data
SELECT
    p.id AS id_jogo,
    p.nome AS jogo,
    e.nome AS expansao
FROM produto p
LEFT JOIN expansao e ON p.id = e.idJogo
ORDER BY p.nome;

-- Query for release year data
SELECT nome, ano_lancamento, 'Jogo Base' AS tipo FROM produto
UNION ALL
SELECT nome, ano_lancamento, 'Expansao' AS tipo FROM expansao;
```