# DATABASE APLICATION

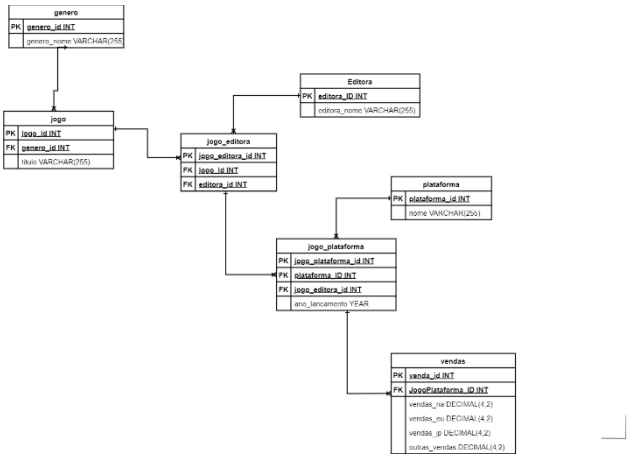
**Sobre o Projeto da equipe Jarvis**

Escolhemos o dataset de vendas de games, tendo dados como número de vendas nas regiões, editora, jogo etc. No projeto foi criado um Docker compose com 3 serviços (containers): mysql, redis e import_data (python).

**Modelo Lógico: diagrama entidade-relacionamento (ERD)**

Tem a de jogo e gênero que é 1:N (um jogo tem um gênero e um gênero tem vários jogos). Um jogo pode ter mais de uma editora, e uma editora mais de um jogo (N:N). O jogo - juntamente com a editora - se liga com a plataforma, ou seja, um jogo de determinada editora pode ter mais de uma plataforma e uma plataforma mais de um jogo. Depois esse jogo, de determinada editora e determinada plataforma, tem os dados da venda.

- **CREATE TABLE**: comando usado para criar uma tabela no banco de dados.
- **genero**, **jogo**, **editora**, **plataforma**, **jogo_editora**, **jogo_plataforma**, **venda**: nomes das tabelas que estão sendo criadas.
- **genero_id**, **jogo_id**, **editora_id**, **plataforma_id**, **jogo_editora_id**, **jogo_plataforma_id**, **venda_id**: colunas de identificação (IDs) para cada tabela. Eles são do tipo INT (inteiro) e são incrementados automaticamente (AUTO_INCREMENT) cada vez que uma nova linha é adicionada à tabela.
- **PRIMARY KEY**: um tipo de restrição que garante que cada valor na coluna seja único e não nulo.
- **genero_nome**, **titulo**, **editora_nome**, **plataforma_nome**: colunas que armazenam informações específicas em cada tabela. Eles são do tipo VARCHAR(255), o que significa que podem armazenar strings de até 255 caracteres.
- **NOT NULL**: uma restrição que garante que a coluna não pode ter um valor nulo.
- **FOREIGN KEY**: uma restrição que garante que o valor na coluna deve corresponder a um valor existente em outra tabela. Por exemplo, *FOREIGN KEY (genero_id) REFERENCES genero(genero_id)* garante que cada genero_id na tabela jogo corresponda a um genero_id existente na tabela genero.
- **vendas_na**, **vendas_eu**, **vendas_jp**, **outras_vendas**: colunas na tabela venda que armazenam informações sobre as vendas em diferentes regiões. Eles são do tipo DECIMAL(6,2), o que significa que podem armazenar números com até 6 dígitos, dos quais 2 são dígitos decimais.



## Containers do Docker Compose

O container de **MySQL** constrói uma imagem situada na pasta ScriptsMySql, na qual roda a DDL, criando as tabelas.

O container de **Python** constrói uma imagem da pasta python que é responsável por baixar as bibliotecas utilizadas e copiar o arquivo .csv e o arquivo de python import_data. O arquivo import_data.py é o mais importante, sendo responsável por inserir os dados do dataset no banco relacional criado, além de extrair do banco relacional e colocar carga no Redis, assim armazenando as consultas mais utilizadas e importantes.

## Dockerfiles

É uma maneira de criar imagens para execução de containers Docker, com instruções personalizadas e que podem ser modificadas posteriormente, conforme a necessidade. O arquivo **DockerFile** é uma estrutura contendo instruções executadas em top-down, que ao final da execução, deixam o container pronto para uso.

```
# Criação da imagem para usar no container de python import_data
FROM python:latest

RUN pip install mysql-connector-python pandas redis

# Define o diretório de trabalho dentro do contêiner como /app/script
WORKDIR /app/script

# Copie o arquivo import_data.py e video_games_sales.csv para o diretório de trabalho dentro do contêiner
COPY import_data.py ./

COPY video_games_sales.csv ./
```

```
# Criação da imagem para usar no container mysql
FROM mysql:latest

# Copia o arquivo script.sql para o diretório /docker-entrypoint-initdb.d/ dentro do contêiner
# Este diretório é usado pelo MySQL para executar scripts SQL durante a inicialização do contêiner
COPY ./script.sql /docker-entrypoint-initdb.d/
```

## importa_data.py

- **import pandas as pd**: comando que importa a biblioteca Pandas e a renomeia como pd. (Pandas é uma biblioteca de manipulação e análise de dados em Python.)
- **from mysql_connection import connect_to_db**: comando que importa a função usada para estabelecer uma conexão com o banco de dados MySQL.

```
import pandas as pd
from mysql_connection import connect_to_db
```

- **def insert_data(conn, table_name, data)**: definição de uma função chamada insert_data que insere dados em uma tabela específica no banco de dados MySQL. A função recebe três argumentos: conn (a conexão com o banco de dados), table_name (o nome da tabela onde os dados serão inseridos) e data (os dados a serem inseridos).

```
def insert_data(conn, table_name, data):
    cursor = conn.cursor()
    placeholders = ', '.join(['%s'] * len(data))
    columns = ', '.join(data.keys())
    sql = f"INSERT INTO {table_name} ({columns}) VALUES ({placeholders})"
    cursor.execute(sql, list(data.values()))
    conn.commit()
    return cursor.lastrowid
```

- **try, except, finally**: bloco de código que tenta executar comandos e captura exceções se ocorrerem. O bloco finally é executado independentemente de uma exceção ser levantada ou não.
- **conn = connect_to_db()**: chama a função connect_to_db para estabelecer uma conexão com o banco de dados MySQL e armazena a conexão na variável conn.
- **df = pd.read_csv('video_games_sales.csv', nrows=1000)**: lê os primeiros 1000 registros de um arquivo CSV chamado 'video_games_sales.csv' e armazena os dados em um DataFrame do Pandas chamado df.
- **df['year'] = df['year'].apply(clean_year)**: aplica a função clean_year a cada valor na coluna 'year' do DataFrame df.
- **df.dropna(subset=['year', 'genre', 'publisher', 'platform'], inplace=True)**: remove quaisquer linhas do DataFrame df que tenham valores ausentes nas colunas 'year', 'genre', 'publisher' ou 'platform'.

```
conn = connect_to_db()
if conn:
    print("Conexão com o banco de dados MySQL estabelecida com sucesso!")

    # Alterar para o caminho do arquivo do dataset local
    df = pd.read_csv('video_games_sales.csv', nrows=1000)

    # Remover valores não numéricos da coluna 'year' e converter para inteiro
    df['year'] = df['year'].apply(clean_year)

    # Remover linhas com valores ausentes em outras colunas
    df.dropna(subset=['year', 'genre', 'publisher', 'platform'], inplace=True)
```

## Dataset (Kaggle) e Dicionário de dados

**Dataset - Video Games Sales**

| Nome | Descrição | Tipo de Dado | Tamanho |
|---|---|---|---|
| Rank | Classificação das vendas totais | Inteiro | - |
| Name | Nome do jogo | Varchar | Variável |
| Platform | Plataforma do jogo | Varchar | Variável |
| Year | Ano de lançamento do jogo | Inteiro | - |
| Genre | Gênero do jogo | Varchar | Variável |
| Publisher | Editora do jogo | Varchar | Variável |
| NA_Sales | Vendas na América do Norte (em milhões) | Decimal | 6,2 |
| EU_Sales | Vendas na Europa (em milhões) | Decimal | 6,2 |
| JP_Sales | Vendas no Japão (em milhões) | Decimal | 6,2 |
| Other_Sales | Vendas no resto do mundo (em milhões) | Decimal | 6,2 |

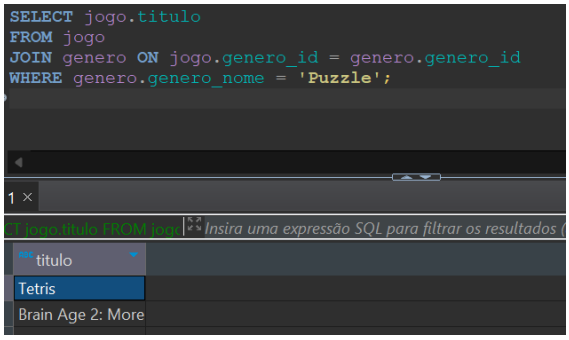
## Comandos que ligam o repositório ao VSCode, e logo em seguida ao DBeaver

- docker-compose up -d mysql redis
- docker-compose up import_data
- docker ps

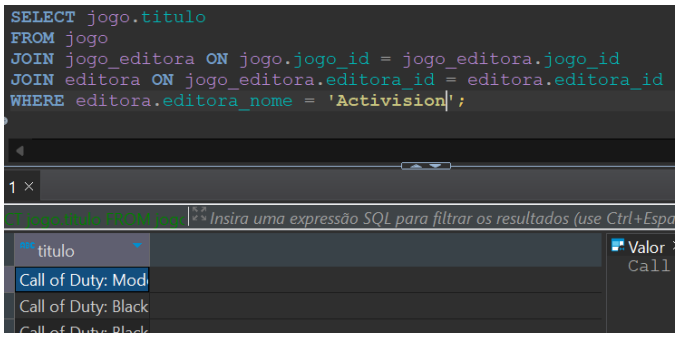
## Consultas no DBeaver

- select * from [crtl + espaço]
- select * from (nome da tabela)

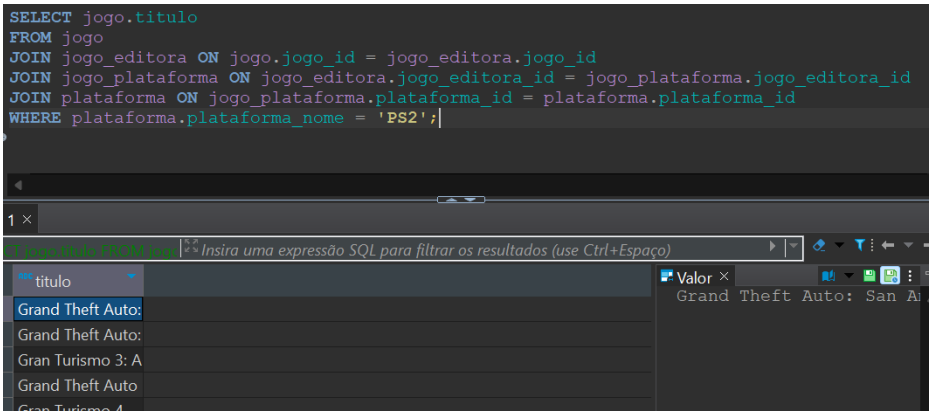

- Jogos de um determinado gênero




- Jogos de uma determinada editora




- Jogos de uma determinada plataforma




- Vendas globais de um determinado jogo

```sql
SELECT jogo.titulo, (venda.vendas_na + venda.vendas_eu + venda.vendas_jp + venda.outras_vendas) AS vendas_globais
FROM jogo
JOIN jogo_editora ON jogo.jogo_id = jogo_editora.jogo_id
JOIN jogo_plataforma ON jogo_editora.jogo_editora_id = jogo_plataforma.jogo_editora_id
JOIN venda ON jogo_plataforma.jogo_plataforma_id = venda.jogo_plataforma_id
WHERE jogo.titulo = 'Super Mario 64';
```

| titulo | vendas_globais |
|---|---|
| Super Mario | 11,9 |

Valor: Super Mario 64

- Vendas na América do Norte de um determinado jogo

```sql
SELECT jogo.titulo, venda.vendas_na
FROM jogo
JOIN jogo_editora ON jogo.jogo_id = jogo_editora.jogo_id
JOIN jogo_plataforma ON jogo_editora.jogo_editora_id = jogo_plataforma.jogo_editora_id
JOIN venda ON jogo_plataforma.jogo_plataforma_id = venda.jogo_plataforma_id
WHERE jogo.titulo = 'Tetris';
```

| titulo | vendas_na |
|---|---|
| Tetris | 23,2 |

Valor: Tetris

- Jogos de um determinado ano

```sql
SELECT jogo.titulo
FROM jogo
JOIN jogo_editora ON jogo.jogo_id = jogo_editora.jogo_id
JOIN jogo_plataforma ON jogo_editora.jogo_editora_id = jogo_plataforma.jogo_editora_id
WHERE jogo_plataforma.ano_lancamento = 2007;
```

| titulo |
|---|
| Wii Fit |
| Halo 3 |
| Super Mario Galaxy |

Valor: Wii Fit