UFCD 5410: Laboratório #1

Nelson Santos
nelson.santos.0001376@edu.atec.pt

ATEC - April 3, 2025

Introdução

Na última sessão teórica foi apresentada a sintaxe relativamente a manipular uma Base de Dados(BD), nomeadamente como criar ou eliminar uma BD.

Nesta sessão o objetivo será a criação de uma BD que irá representar um loja online, bem como as tabelas que a irá constituir.

As tabelas e colunas serão apresentadas no enunciado, no entanto o tipo de cada coluna vai ser estabelecido pelo aluno, numa tentativa critica da escolha do tipo adequado de cada coluna face aos tipos disponíveis no MySQL e apresentados na aula teórica.

1 Conceitos Base

Nesta secção, e inicialmente, vamos explorar dois conceitos fundamentais em BD relacionais: chaves primárias/*primary key* (PK) e chaves estrangeiras/*foreign key* (FK). Estes conceitos apresentam uma importância essencial na organização e na ligação dos dados dentro de uma BD, garantindo tanto a **integridade** quanto a **eficiência** das operações realizadas sobre os mesmos. Vamos detalhar o que cada uma dessas chaves representa e como elas são utilizadas para estabelecer relações entre tabelas, proporcionando uma compreensão mais clara de como os dados são guardados e acedidos num Sistema de Gestão de Base de Dados (SGBD) relacional.

Assim:

- **Chave primária - PK**: Uma chave primária é um campo/coluna (ou conjunto de campos) de uma tabela da BD que serve como identificador exclusivo/único para cada linha/tuplo na tabela. Cada tabela normalmente tem uma chave primária que garante que não haja tuplos duplicados e que cada tuplo seja único. A chave primária é usada como índice (*index*) para indexar os dados na tabela e otimizar operações de pesquisa, atualização e remoção de tuplos
- **Chave estrangeira - FK**: Uma chave estrangeira é um campo numa tabela que estabelece uma relação com a chave primária de outra tabela. A chave estrangeira cria um vínculo entre as duas tabelas, permitindo que os dados de uma tabela sejam relacionados aos dados de outra tabela. A chave estrangeira geralmente representa uma referência a uma chave primária de outra tabela

Resumo dos principais tipos de dados:

- **TEXT**: Guarda conjunto de caracteres de tamanho grande, com capacidade de até 65.535 bytes. Útil para armazenar grandes blocos de texto, como descrições longas, entre outros
- **CHAR**: Guarda conjunto de caracteres de comprimento fixo. É necessário especificar um comprimento máximo para a string. Preenche com espaços à direita para atingir o comprimento máximo, se necessário. Útil quando os dados terão sempre o mesmo comprimento
- **VARCHAR**: Guarda conjunto de caracteres de comprimento variável. É especificado um comprimento máximo para a string, mas o espaço ocupado depende do comprimento real da string. Útil para guardar strings de comprimentos variáveis, economizando assim espaço em disco

- **INT**: Guarda números inteiros de precisão fixa. Tamanho variável dependendo da precisão necessária. Útil para guardar números inteiros como IDs, quantidades, etc
- **DECIMAL**: Guarda números decimais exatos com uma precisão fixa. É especificado a precisão total e a casas decimais. Útil para situações onde a precisão é importante, como em cálculos, etc
- **FLOAT**: Guarda números de ponto flutuante de precisão simples. Ocupa 4 bytes. Útil para guardar números com casas decimais num intervalo relativamente amplo
- **DOUBLE**: Guarda números de ponto flutuante de precisão dupla. Ocupa 8 bytes. Oferece uma precisão maior que FLOAT, sendo útil para números com casas decimais num intervalo ainda mais amplo e com maior precisão

2 Base de Dados 1

De seguida podemos encontrar um script `SQL` que nos permite criar uma BD e a tabela `user` com os tipos de dados adequados.

Execute o `script` seguinte para criar a BD e a tabela `users`.

script.sql

```
#Create a database if it doesn't exist
CREATE DATABASE IF NOT EXISTS user;

#Drop the database if it exists
DROP DATABASE IF EXISTS user;

#Create a database if it doesn't exist
CREATE DATABASE IF NOT EXISTS user;

#Use the newly created database
USE user;

#Create a user table
CREATE TABLE user (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

#Create table role
CREATE TABLE role (
    role_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    role_name VARCHAR(50),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Analise cada query[1] e verifique se os tipos de dados estão adequados a cada coluna.

3 Desafio 1

Vamos experimentar adicionar novos tuplos ou novas linhas na nossa BD `user`:

[1] Em SQL, uma `query` é uma instrução ou comando utilizado para manipular uma BD. A `query` serve para consultar, inserir, atualizar ou excluir dados de tabelas dentro de um SGBD.

script.sql

```
INSERT INTO user (username, email, password)
    VALUES ('john_doe', 'john@example.com', 'password123');

INSERT INTO user (username, email, password)
    VALUES ('jane_smith', 'jane@example.com', 'password456');

INSERT INTO role (user_id, role_name)
    VALUES (1, 'admin');

INSERT INTO role (user_id, role_name)
    VALUES (2, 'user');

INSERT INTO role (user_id, role_name)
    VALUES (3, 'guest');
```

Verifique se os dados foram inseridos com sucesso. Para isso execute as seguinte `queries`:

script.sql

```
#Show all user
SELECT * FROM user;

#Show all role
SELECT * FROM role;

#Show user with email
SELECT * from user where email = 'john@example.com';
```

4 Desafio 2

Vamos agora criar uma BD nova: `shop`. O objetivo deste desafio será criar a BD `shop` bem como adicionar as tabelas e a escolha dos tipos adequados a cada coluna. Não existe qualquer problema em não escolher o tipo de dados adequados.

As tabelas a introduzir serão as seguintes:

script.sql

```
+--------------+        +----------------+         +--------------+
|   user       |        |   product      |         |    order     |
+--------------+        +----------------+         +--------------+
| user_id (PK) |        | product_id (PK)|         | order_id (PK)|
| name         |        | product_name   |         | user_id (FK) |
| email        |        | product_desc   |         | order_date   |
| password     |        | price          |         | order_status |
| address      |        | category       |         +--------------+
| phone        |        | stock_quantity |
+--------------+        +----------------+
```

script.sql

```
+------------------+         +------------+
|   order_item     |         |  payment   |
+------------------+         +------------+
| order_item_id(PK)|         | payment_id |
| order_id (FK)    |         | order_id   |
| product_id (FK)  |         | method     |
| quantity         |         | amount     |
| unit_price       |         | date       |
| total_price      |         +------------+
+------------------+
```

5 Diagrama Entidade-Relação

Um diagrama Entidade-Relação (ER) é uma ferramenta de modelagem de dados usada para representar visualmente as relações entre diferentes entidades de um SGBD. Este diagrama é constituído por: entidades, atributos e relações entre as entidades. O diagrama ER auxilia a compreender a estrutura de um SGBD, permitindo analisar como os dados estão organizados e como se relacionam.

Assim um diagrama ER é constituído por:

- **Entidades**: Entidades, normalmente representadas por retângulos, poderão-se designar de substantivos (tais como objetos, pessoas, conceitos ou eventos). Também podemos designar de 'objetos do mundo real'. Se quisermos modelar uma biblioteca, as entidades poderão ser, por exemplo, livro, autor, leitor, etc.
- **Atributos**: São características ou propriedades das entidades. Cada entidade tem seus próprios atributos e que descrevem as informações que são armazenadas sobre ela. Por exemplo, um livro pode ter atributos como título, autor, ano de Publicação, isbn, etc.
- **Relações**: representam as associações entre diferentes entidades. Continuando o nosso exemplo da biblioteca, a relação entre autor e livro seria 'escrito por': livro —> escrito por —> autor. Podemos quantificar a relação referindo que um autor pode ter escrito vários livros, e um livro poderá ter um ou vários autores. Graficamente as relações são representadas por linhas que conectam as entidades.

O MySQL Workbench permite gerar um diagrama ER de forma muito rápida e eficiente.
Passos a seguir:

- **Step1:** Confirmar que existe a BD com tabelas
- **Step2:** Clicar no menu superior **Database** -> **Reverse Engineer**
- **Step3:** Selecionar e confirmar os dados da conexão. De seguida selecione **Continue**
- **Step4:** Após a realização da conexão com sucesso, selecionar **Continue**
- **Step5:** Selecionar a base de dados do MySQL Server que queremos gerar o diagrama ER. De seguida selecione **Continue**
- **Step6:** Após a obtenção da informação sobre os objetos, selecione **Next**
- **Step7:** Verificar sinal 'check' na opção de importar as tabelas MySQL e selecione **Execute**
- **Step8:** Se importação bem realizada, selecione **Continue**
- **Step9:** Após aparecer o resumo, selecione **Close**

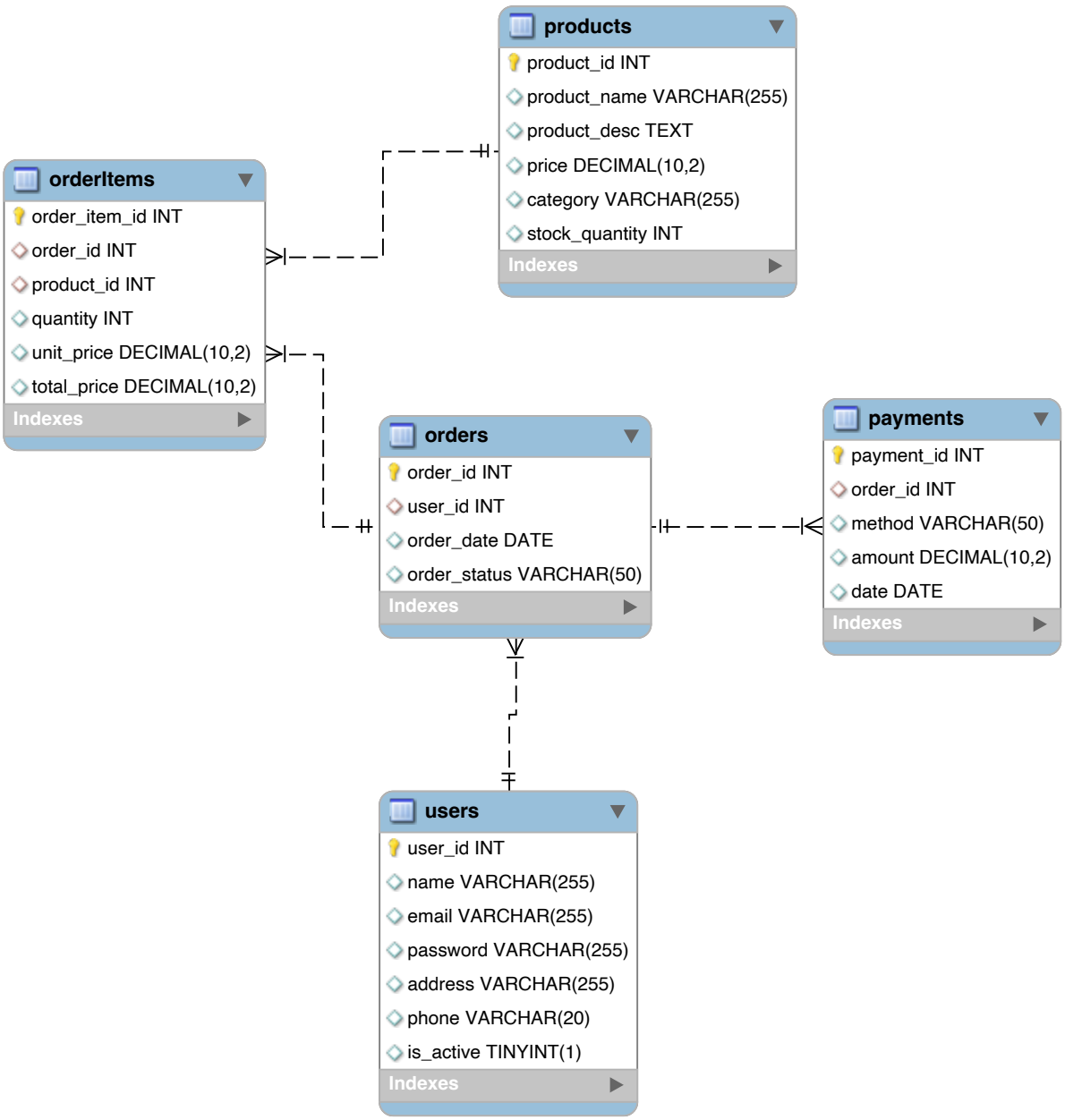

Figure 1: Diagrama ER loja online.

6 Submissão Diagrama ER

Após o término do **Desafio 2** por favor submeta o diagrama ER da sua implementação no assignment criado pelo formador para o efeito.

7 Atualização de uma tabela

Durante o processo de gestão de uma BD, a necessidade te atualizar uma tabela poderá surgir, sendo indispensável o domínio do uso da *statement* `ALTER TABLE`.

`ALTER TABLE` possibilita a alteração da estrutura de uma tabela existente, através da modificação, acrescento ou remoção de colunas e restrições (*constraints*). Esta funcionalidade permite uma grande flexibilidade de adaptar o esquema (*schema*) de uma BD, através da alteração dos requisitos iniciais sem que seja necessário recriar a tabela.

As operações mais comuns consiste em adicionar novas colunas numa tabela, modificar o tipo de uma coluna ou as restrições, e ainda a remoção de uma coluna.

Execute as alterações presentes no script seguinte. Estas representam as três operações mais comuns com a *statement* `ALTER TABLE`:

- Acrescentar uma nova coluna
- Update de uma *constraint* ou restrição
- Update de um tipo de uma coluna

script.sql

```
# Add new column to user table
ALTER TABLE user
    ADD COLUMN is_active BOOLEAN DEFAULT TRUE;

#Add new FK in order_items table
ALTER TABLE order_item
ADD CONSTRAINT fk_product_id
    FOREIGN KEY (product_id) REFERENCES products(product_id);

#Update column type
ALTER TABLE product
    MODIFY COLUMN price FLOAT;
```

8 Remoção de uma tabela

Por vezes torna-se necessário a remoção de uma determinada tabela de uma BD. Esta operação é possível com a utilização da *statement* `DROP TABLE`.

No entanto há que usar com muito cuidado esta *statement* pois as consequencias são irreversíveis, pois os dados associados a esta tabela serão definitivamente eliminados. Estes dados consistem os tuplos da tabela, *schema*, índices, *triggers* e/ou restrições ou *constraints*.

Execute o script seguinte. Este cria uma tabela e de seguida a mesma será apagada.

script.sql

```
#Create a temporary table
CREATE TABLE temporary (id INT PRIMARY KEY,
    name VARCHAR(255), description TEXT);

#Drop the temporary table
    DROP TABLE temporary;
```

Bom trabalho!