

Universidade Estadual do Rio Grande do Norte - UERN
Diretoria de Educação a Distância - DEAD
Sistemas para Internet
Banco de dados
Marcelo Henrique Lima Silva

PizzaLab

Sumário

1. Introdução
2. Estrutura do Banco de Dados
3. Procedimentos Armazenados (Stored Procedures)
4. Gatilhos (Triggers)
5. Scripts SQL e Resultados
6. Consultas de Dados (SELECTs)
7. Considerações Finais
8. GitHub do Projeto
9. Referências

1. Introdução

Neste documento, consta o projeto do banco de dados para o sistema da pizzaria PizzaLab, que armazenará todas as informações nele contidas, desde usuários, produtos, estoque, etc. O objetivo é tornar os dados armazenados para o sistema ainda mais eficientes para gestão e usabilidade, garantindo a integridade e disponibilidade dos dados.

O banco de dados possui relacionamento entre as tabelas criadas, respeitando boas práticas de modelagem, além de outros recursos como procedimentos armazenados e gatilhos para automatizar as operações e garantir a consistência das informações.

2. Estrutura do banco de dados

Para gerenciar as operações do PizzaLab, foram modeladas as seguintes tabelas principais:

- **Users:** Armazena as informações dos usuários, como nome, telefone, endereço e tipo de usuário (Administrador, Colaborador ou Cliente).
- **Products:** Contém os dados das pizzas oferecidas no cardápio, incluindo nome, tamanho e preço.
- **Ingredients:** Lista todos os ingredientes disponíveis para a montagem das pizzas.
- **Product_ingredients:** Tabela associativa que relaciona os produtos (pizzas) com seus respectivos ingredientes, incluindo a quantidade utilizada.
- **Stock:** Gerencia o estoque de ingredientes, controlando a quantidade disponível, unidade de medida, data de validade e data da última compra.
- **Orders:** Registra todos os pedidos realizados, com informações sobre o cliente, taxas e descontos.
- **Order_items:** Detalha os itens contidos em cada pedido, especificando o produto, a quantidade e o subtotal.

A estrutura foi desenhada para garantir a normalização e a integridade referencial dos dados.

3. Procedimentos Armazenados (Stored Procedures)

Para automatizar consultas recorrentes e encapsular a lógica de negócio, foram criados os seguintes procedimentos armazenados:

- **PENDING_REQUEST():** Retorna uma lista com todos os pedidos que ainda não foram concluídos ou cancelados.

- **PENDING_REQUEST_CLIENT(ID_CLIENT):** Retorna os detalhes de todos os pedidos de um cliente específico.

4. Gatilhos (Triggers)

Foram implementados gatilhos para automatizar operações e garantir a integridade dos dados, reduzindo a necessidade de intervenção manual e prevenindo inconsistências.

- **PENDING:** Um gatilho que, ao ser inserido um novo pedido, define automaticamente o seu status inicial como "pendente".
- **DELETE_NO:** Garante que nenhum pedido possa ser excluído diretamente do banco de dados, preservando o histórico de operações.

5. Scripts SQL e Resultados

A seguir, são apresentados os scripts SQL utilizados para a criação da estrutura do banco de dados e os resultados de sua execução.

5.1. Criação do Banco de Dados e Tabelas

-- Criação do banco de dados

```
CREATE DATABASE IF NOT EXISTS pizzalab;
```

-- Selecionando o banco de dados

```
USE pizzalab;
```

-- Tabela de usuários

```
CREATE TABLE users(  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  NAME VARCHAR(256) NOT NULL,  
  PHONE VARCHAR(20) NOT NULL UNIQUE,  
  ADDRESS VARCHAR(256) NOT NULL,  
  TYPE TINYINT NOT NULL DEFAULT 2 CHECK (TYPE IN (0, 1, 2)) -- 0: Admin, 1:  
  Colaborador, 2: Cliente  
);
```

-- Tabela de ingredientes

```
CREATE TABLE ingredients(  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  NAME VARCHAR(100) NOT NULL UNIQUE  
);
```

-- Tabela de produtos (pizzas)

```
CREATE TABLE products(  
  ID INT PRIMARY KEY AUTO_INCREMENT,
```

```
NAME VARCHAR(256) NOT NULL,  
SIZE ENUM('P', 'M', 'G') NOT NULL,  
PRICE DECIMAL(10,2) NOT NULL  
);
```

-- Tabela de associação entre produtos e ingredientes

```
CREATE TABLE product_ingredients(  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  ID_PRODUCT INT NOT NULL,  
  ID_INGREDIENT INT NOT NULL,  
  QUANTITY DECIMAL(5,2) NOT NULL,  
  UNIT VARCHAR(20) NOT NULL,  
  FOREIGN KEY (ID_PRODUCT) REFERENCES products(ID) ON DELETE CASCADE,  
  FOREIGN KEY (ID_INGREDIENT) REFERENCES ingredients(ID) ON DELETE  
  CASCADE  
);
```

-- Tabela de estoque

```
CREATE TABLE stock(  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  ID_INGREDIENT INT NOT NULL,  
  QUANTITY DECIMAL(10,2) NOT NULL CHECK (QUANTITY >= 0),  
  UNIT VARCHAR(20) NOT NULL,  
  EXPIRATION_DATE DATE NOT NULL,  
  LAST_PURCHASE TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (ID_INGREDIENT) REFERENCES ingredients(ID) ON DELETE  
  CASCADE  
);
```

-- Tabela de pedidos

```
CREATE TABLE orders(  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  ID_USER INT NOT NULL,  
  DELIVERY_FEE DECIMAL(10,2) NOT NULL DEFAULT 0.00,  
  DISCOUNT DECIMAL(10,2) NOT NULL DEFAULT 0.00,  
  TOTAL DECIMAL(10,2) NOT NULL DEFAULT 0.00,  
  createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (ID_USER) REFERENCES users(ID) ON DELETE CASCADE  
);
```

-- Tabela de itens do pedido

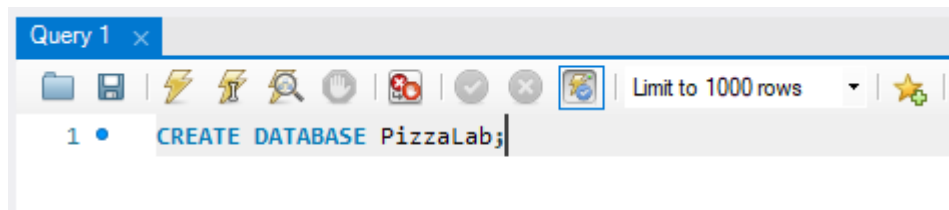
```
CREATE TABLE order_items(  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  ID_ORDER INT NOT NULL,  
  ID_PRODUCT INT NOT NULL,  
  QUANTITY INT NOT NULL CHECK (QUANTITY > 0),  
  UNIT_PRICE DECIMAL(10,2) NOT NULL,  
  SUBTOTAL DECIMAL(10,2) NOT NULL,
```

```
FOREIGN KEY (ID_ORDER) REFERENCES orders(ID) ON DELETE CASCADE,  
FOREIGN KEY (ID_PRODUCT) REFERENCES products(ID) ON DELETE CASCADE  
);
```

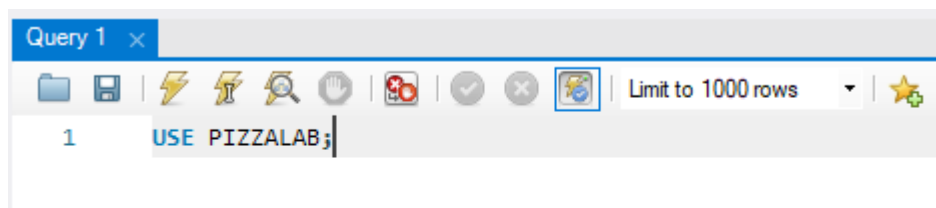
5.2. Criação e Verificação da Estrutura das Tabelas

Aqui, você deve adicionar os prints dos comandos DESC para cada tabela.

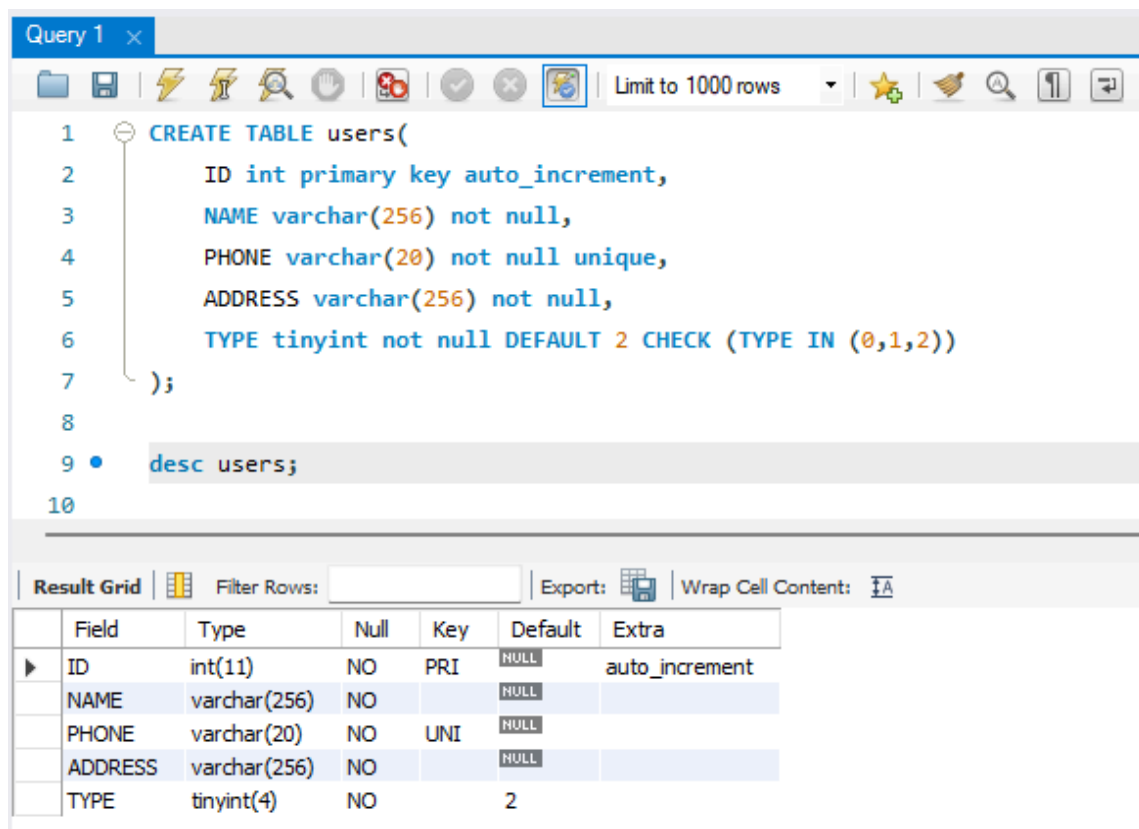
Criando o banco de dados:



Seleciona o banco de dados:



Criando a tabela **users**:



Criando tabela **products**:

```
Query 1 x
Limit to 1000 rows

1 • CREATE TABLE products(
2     ID int primary key auto_increment,
3     NAME varchar(256) not null,
4     SIZE enum('P', 'M', 'G') not null,
5     PRICE decimal(10,2) not null
6 );
7
8 • desc products;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
NAME	varchar(256)	NO		NULL	
SIZE	enum('P','M','G')	NO		NULL	
PRICE	decimal(10,2)	NO		NULL	

Criando a tabela **ingredients**:

```
Query 1 x
Limit to 1000 rows

1 • CREATE TABLE ingredients(
2     ID int primary key auto_increment,
3     NAME varchar(100) not null unique
4 );
5
6 • desc ingredients;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
NAME	varchar(100)	NO	UNI	NULL	

Criando a tabela **product_ingredients**:

Query 1 x

Limit to 1000 rows

```
1 • CREATE TABLE product_ingredients(  
2     ID int primary key auto_increment,  
3     ID_PRODUCT int not null,  
4     ID_INGREDIENT int not null,  
5     QUANTITY decimal(5,2) not null,  
6     UNIT varchar(20) not null,  
7     FOREIGN KEY (ID_PRODUCT) REFERENCES products(ID) ON DELETE CASCADE,  
8     FOREIGN KEY (ID_INGREDIENT) REFERENCES ingredients(ID) ON DELETE CASCADE  
9 );  
10  
11 • desc product_ingredients;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
ID_PRODUCT	int(11)	NO	MUL	NULL	
ID_INGREDIENT	int(11)	NO	MUL	NULL	
QUANTITY	decimal(5,2)	NO		NULL	
UNIT	varchar(20)	NO		NULL	

Criando a tabela **stock**:

Query 1 x

Limit to 1000 rows

```
1 • CREATE TABLE stock(  
2     ID int primary key auto_increment,  
3     ID_INGREDIENT int not null,  
4     QUANTITY decimal(10,2) not null CHECK (QUANTITY >= 0),  
5     UNIT varchar(20) not null,  
6     EXPIRATION_DATE DATE not null,  
7     LAST_PURCHASE TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
8     FOREIGN KEY (ID_INGREDIENT) REFERENCES ingredients(ID) ON DELETE CASCADE  
9 );  
10  
11 • desc stock;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
ID_INGREDIENT	int(11)	NO	MUL	NULL	
QUANTITY	decimal(10,2)	NO		NULL	
UNIT	varchar(20)	NO		NULL	
EXPIRATION_DATE	date	NO		NULL	
LAST_PURCHASE	timestamp	NO		current_timestamp()	

Criando a tabela **orders**:

Query 1

```
1 • CREATE TABLE orders(  
2     ID int primary key auto_increment,  
3     ID_USER int not null,  
4     DELIVERY_FEE decimal(10,2) not null DEFAULT 0.00,  
5     DISCOUNT decimal(10,2) not null DEFAULT 0.00,  
6     TOTAL decimal(10,2) not null DEFAULT 0.00,  
7     createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
8     FOREIGN KEY (ID_USER) REFERENCES users(ID) ON DELETE CASCADE  
9 );  
10  
11 • desc orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
ID_USER	int(11)	NO	MUL	NULL	
DELIVERY_FEE	decimal(10,2)	NO		0.00	
DISCOUNT	decimal(10,2)	NO		0.00	
TOTAL	decimal(10,2)	NO		0.00	
createdAt	timestamp	NO		current_timestamp()	

Criando a tabela **order_items**:

Query 1

```
1 • CREATE TABLE order_items(  
2     ID int primary key auto_increment,  
3     ID_ORDER int not null,  
4     ID_PRODUCT int not null,  
5     QUANTITY int not null CHECK (QUANTITY > 0),  
6     UNIT_PRICE decimal(10,2) not null,  
7     SUBTOTAL decimal(10,2) not null,  
8     FOREIGN KEY (ID_ORDER) REFERENCES orders(ID) ON DELETE CASCADE,  
9     FOREIGN KEY (ID_PRODUCT) REFERENCES products(ID) ON DELETE CASCADE  
10 );  
11  
12 • desc order_items;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	auto_increment
ID_ORDER	int(11)	NO	MUL	NULL	
ID_PRODUCT	int(11)	NO	MUL	NULL	
QUANTITY	int(11)	NO		NULL	
UNIT_PRICE	decimal(10,2)	NO		NULL	
SUBTOTAL	decimal(10,2)	NO		NULL	

Cadastrando usuários:

```
Query 1 x
Limit to 1000 rows
1 /* Criando um cliente padrão */
2 • INSERT INTO users (NAME, PHONE, ADDRESS) VALUES ('Carlos Souza', '999888777', 'Rua B, 456');
3 /* Criando um colaborador */
4 • INSERT INTO users (NAME, PHONE, ADDRESS, TYPE) VALUES ('Ana Pereira', '998877665', 'Rua C, 789', 1);
5 /* Criando um administrador */
6 • INSERT INTO users (NAME, PHONE, ADDRESS, TYPE) VALUES ('Dono da Pizzaria', '990011223', 'Rua D, 000', 0);
7
```

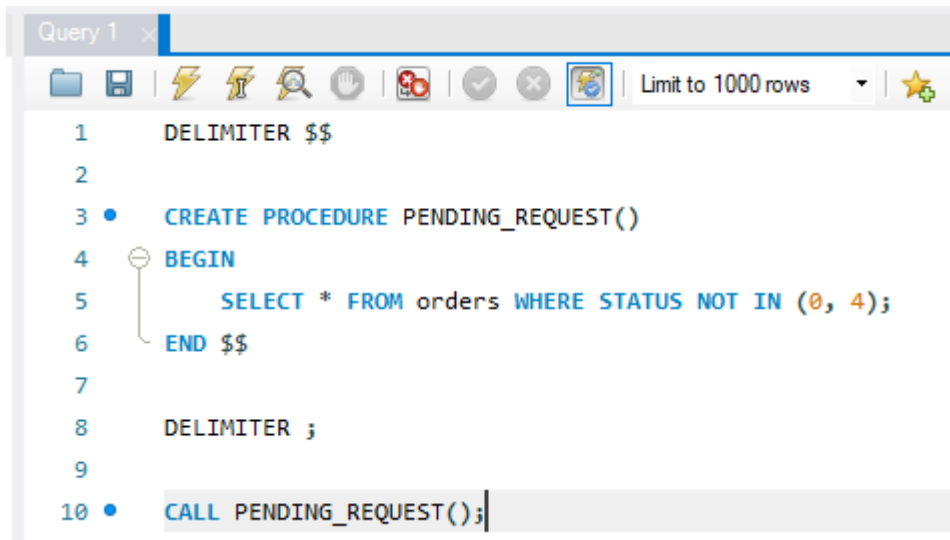
Cadastro e relacionamento do produto e ingrediente:

```
Query 1 x
Limit to 1000 rows
1 /* Cadastro de ingredientes */
2 • INSERT INTO ingredients (NAME) VALUES ('Queijo Mussarela'), ('Molho de Tomate'), ('Calabresa'), ('Orégano');
3 /* Cadastrando um produto */
4 • INSERT INTO products (NAME, SIZE, PRICE) VALUES ('Pizza de Calabresa', 'Média', 39.90);
5 /* Relacionando ingredientes com o produto */
6 • INSERT INTO product_ingredients (ID_PRODUCT, ID_INGREDIENT, QUANTITY, UNIT) VALUES
7 (1, 1, 150, 'g'), -- 150g de Queijo Mussarela
8 (1, 2, 100, 'ml'), -- 100ml de Molho de Tomate
9 (1, 3, 80, 'g'), -- 80g de Calabresa
10 (1, 4, 5, 'g'); -- 5g de Orégano
11
```

Criando pedido e adicionando itens:

```
Query 1 x
Limit to 1000 rows
1 /* Criando um pedido */
2 • INSERT INTO orders (ID_USER, DELIVERY_FEE, DISCOUNT, TOTAL)
3 VALUES (1, 5.00, 3.00, 76.80);
4 /* Adicionando itens ao pedido */
5 • INSERT INTO order_items (ID_ORDER, ID_PRODUCT, QUANTITY, UNIT_PRICE, SUBTOTAL)
6 VALUES (1, 1, 2, 39.90, 79.80);
7
```

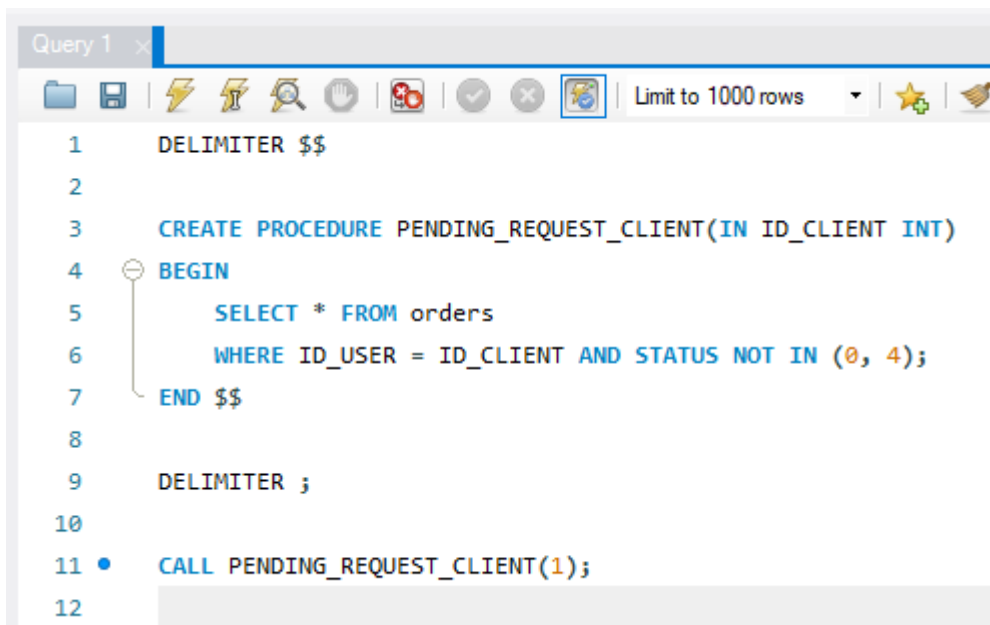
PENDING_REQUEST:



The screenshot shows a SQL query editor window titled "Query 1". The toolbar includes icons for file operations, execution, and a "Limit to 1000 rows" dropdown. The SQL code is as follows:

```
1 DELIMITER $$
2
3 • CREATE PROCEDURE PENDING_REQUEST()
4 BEGIN
5     SELECT * FROM orders WHERE STATUS NOT IN (0, 4);
6 END $$
7
8 DELIMITER ;
9
10 • CALL PENDING_REQUEST();
```

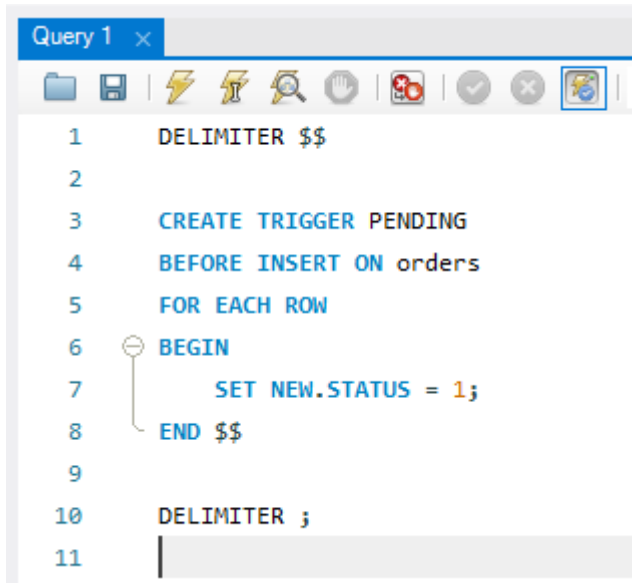
PENDING_REQUEST_CLIENT:



The screenshot shows a SQL query editor window titled "Query 1". The toolbar includes icons for file operations, execution, and a "Limit to 1000 rows" dropdown. The SQL code is as follows:

```
1 DELIMITER $$
2
3 CREATE PROCEDURE PENDING_REQUEST_CLIENT(IN ID_CLIENT INT)
4 BEGIN
5     SELECT * FROM orders
6     WHERE ID_USER = ID_CLIENT AND STATUS NOT IN (0, 4);
7 END $$
8
9 DELIMITER ;
10
11 • CALL PENDING_REQUEST_CLIENT(1);
12
```

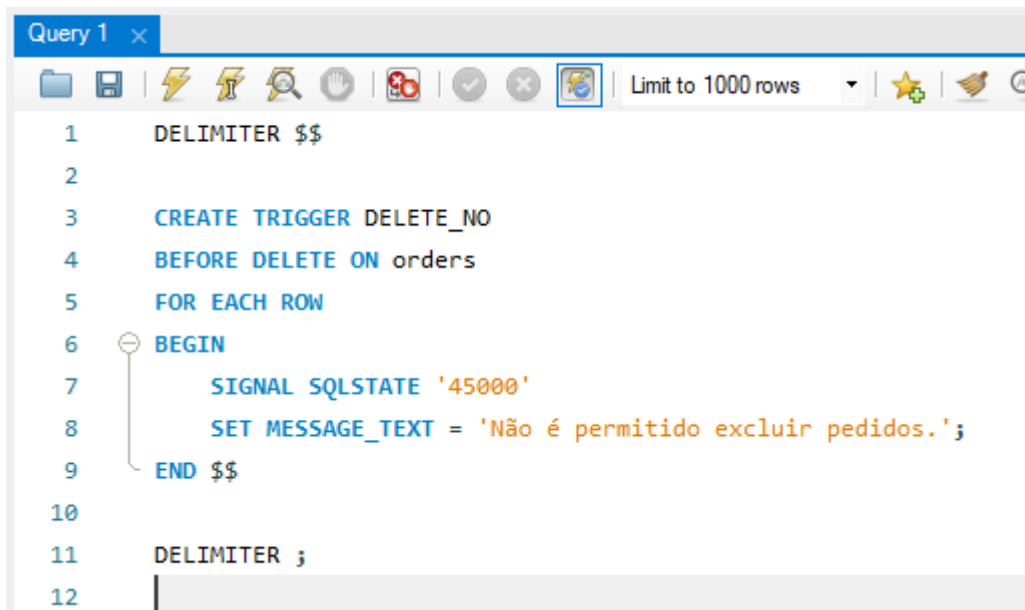
PENDING:



The screenshot shows a SQL query editor window titled "Query 1". The toolbar includes icons for file operations, execution, and debugging. The SQL code is as follows:

```
1 DELIMITER $$
2
3 CREATE TRIGGER PENDING
4 BEFORE INSERT ON orders
5 FOR EACH ROW
6 BEGIN
7     SET NEW.STATUS = 1;
8 END $$
9
10 DELIMITER ;
11
```

DELETE_NO:



The screenshot shows a SQL query editor window titled "Query 1". The toolbar includes icons for file operations, execution, and debugging, along with a "Limit to 1000 rows" dropdown. The SQL code is as follows:

```
1 DELIMITER $$
2
3 CREATE TRIGGER DELETE_NO
4 BEFORE DELETE ON orders
5 FOR EACH ROW
6 BEGIN
7     SIGNAL SQLSTATE '45000'
8     SET MESSAGE_TEXT = 'Não é permitido excluir pedidos.';
9 END $$
10
11 DELIMITER ;
12
```

6. Consultas de Dados (SELECTs)

Para verificar os dados inseridos e o funcionamento do banco, podem ser utilizadas as seguintes consultas básicas.

```
-- Consultar todos os usuários cadastrados  
SELECT * FROM users;
```

```
-- Consultar todos os ingredientes  
SELECT * FROM ingredients;
```

```
-- Consultar todos os produtos (pizzas)  
SELECT * FROM products;
```

```
-- Consultar a relação entre pizzas e ingredientes  
SELECT * FROM product_ingredients;
```

```
-- Consultar o estado atual do estoque  
SELECT * FROM stock;
```

```
-- Consultar todos os pedidos realizados  
SELECT * FROM orders;
```

```
-- Consultar todos os itens de pedidos  
SELECT * FROM order_items;
```

7. Considerações Finais

O banco de dados desenvolvido para o sistema PizzaLab atende de forma robusta às necessidades operacionais de uma pizzaria moderna. Com o uso de uma modelagem relacional bem definida, procedimentos armazenados e gatilhos, foi possível criar um sistema que garante a integridade e a consistência dos dados, ao mesmo tempo que automatiza tarefas críticas.

Futuras implementações podem incluir a otimização de consultas para relatórios gerenciais, a integração com um sistema de controle de caixa e o desenvolvimento de uma API para dar suporte a aplicativos mobile e web.

8. GitHub do Projeto

[Link projeto](#)

9. Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: Informação e documentação — Referências — Elaboração. Rio de Janeiro, 2018.