

A photograph of a calm lake with several wooden boats in the foreground. The boats are dark brown and appear to be traditional rowing boats. The water is still, reflecting the sky and the surrounding trees. In the background, there is a dense line of green trees under a clear sky. The overall scene is peaceful and natural.

Stored Procedures e Triggers

Professor: Ricardo Luis dos Santos

IFSUL – Campus Sapucaia do Sul

Agenda

- Rotinas Armazenadas
- Stored Procedures
 - Exemplos
- Triggers
 - Exemplos
- Exercícios

Rotinas Armazenadas

- **Rotinas Armazenadas** são um conjunto de comandos SQL armazenados em SGBD (Sistema Gerenciador de Banco de Dados)
- Rotinas Armazenadas podem ser
 - **Stored Procedures** ou Procedimentos Armazenados
 - **Triggers** ou Gatilhos

Stored Procedures

- Stored Procedures são rotinas armazenadas que devem ser executadas por intermédio de invocações explícitas do usuário
 - COMANDO
 - **CREATE PROCEDURE** minha_procedure()
 - **CALL**
 - CALL EXIBIR_VENDEDORES();
 - CALL MOSTRA_VENDAS();
 - CALL CALCULAR_COMISSAO();

Stored Procedures

- Exemplo de uma stored procedure

```
1 DELIMITER //
```

```
2 • CREATE PROCEDURE exibir_vendedores()
```

```
3   BEGIN
```

```
4       SELECT * FROM VENDEDOR AS V;
```

```
5   END //
```

```
6 DELIMITER ;
```

Stored Procedures

Bloco de comandos que forma o stored procedure

stored procedure

Substitui temporariamente o “;” enquanto símbolo finalizador de comandos SQL

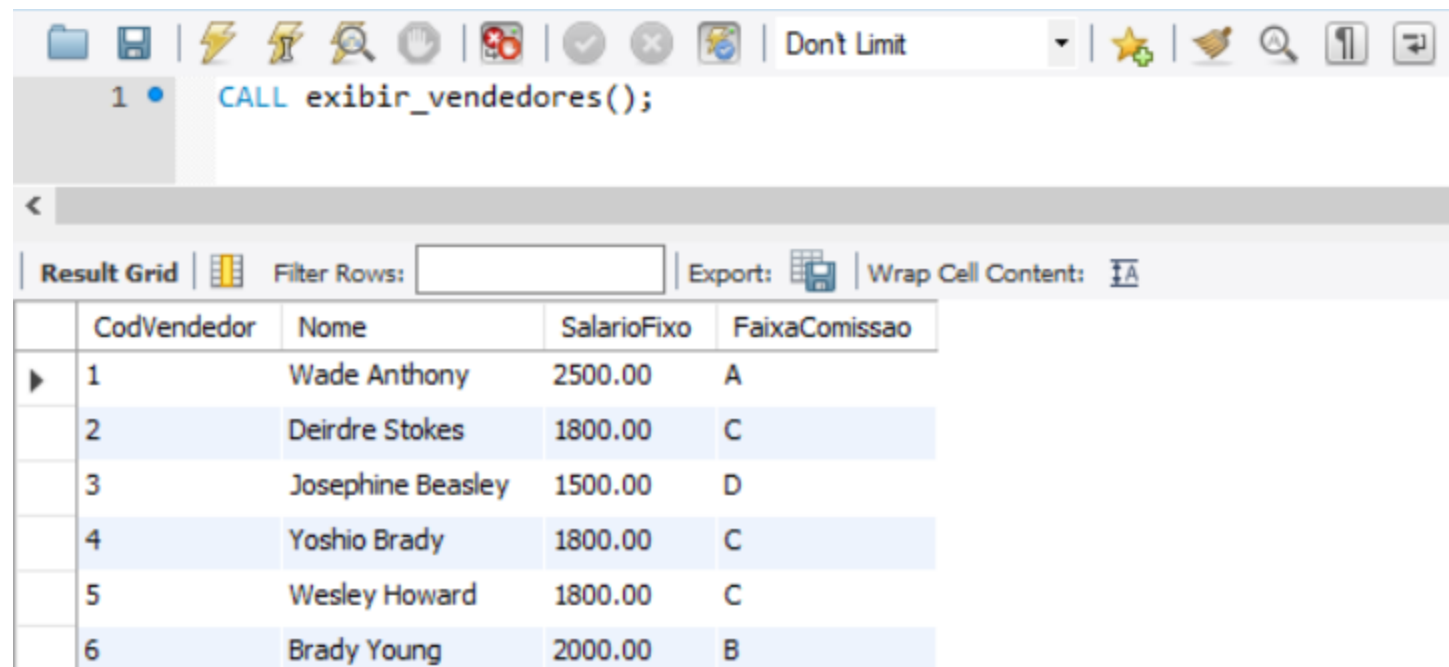
```
1 DELIMITER //
2 CREATE PROCEDURE exibir_vendedores()
3 BEGIN
4     SELECT * FROM VENDEDOR AS V;
5 END //
6 DELIMITER ;
```

Comando para criação de um stored procedure

Comandos SQL que compõem o stored procedure

Stored Procedures

- Utilizamos o comando CALL para executar um stored procedure



The screenshot shows a database IDE interface. At the top, there is a toolbar with various icons for file operations, execution, and search. Below the toolbar, a SQL command is entered in a text area: `CALL exibir_vendedores();`. Below the command, a "Result Grid" is displayed, showing the output of the stored procedure. The grid has five columns: "CodVendedor", "Nome", "SalarioFixo", and "FaixaComissao". It contains six rows of data, each representing a salesperson.

	CodVendedor	Nome	SalarioFixo	FaixaComissao
▶	1	Wade Anthony	2500.00	A
	2	Deirdre Stokes	1800.00	C
	3	Josephine Beasley	1500.00	D
	4	Yoshio Brady	1800.00	C
	5	Wesley Howard	1800.00	C
	6	Brady Young	2000.00	B

Stored Procedures

- Stored procedures são criados para serem utilizados diversas vezes no futuro
- Mas se, por algum motivo, você precisar apagar o stored procedure, pode usar o comando DROP

DROP PROCEDURE EXIBIR_VENDEDORES;

Stored Procedures

- Assim como as funções, um stored procedure pode receber utilizar **parâmetros**, utilizando a sintaxe **(MODO nome TIPO ...)**
 - **nome** - Nome do parâmetro, segue as mesmas regras das variáveis
 - **tipo** - Tipo do parâmetro (Int, Varchar, Decimal, entre outros)
 - **MODO**
 - **IN** – Indica que o parâmetro é de entrada de dados
 - **OUT** – Indica que o parâmetro é de retorno de dados
 - **INOUT** – Indica que o parâmetro será utilizado para entrada e retorno dos dados

Stored Procedures

- Exemplo de Stored Procedure com IN

```
1  DELIMITER //
```

```
2  ● CREATE PROCEDURE exibir_produtos(IN QUANTIDADE INT)
```

```
3  ┌ BEGIN
```

```
4    SELECT
```

```
5      *
```

```
6    FROM
```

```
7      PRODUTO AS P
```

```
8    LIMIT QUANTIDADE;
```

```
9  └ END //
```

```
10 DELIMITER ;
```

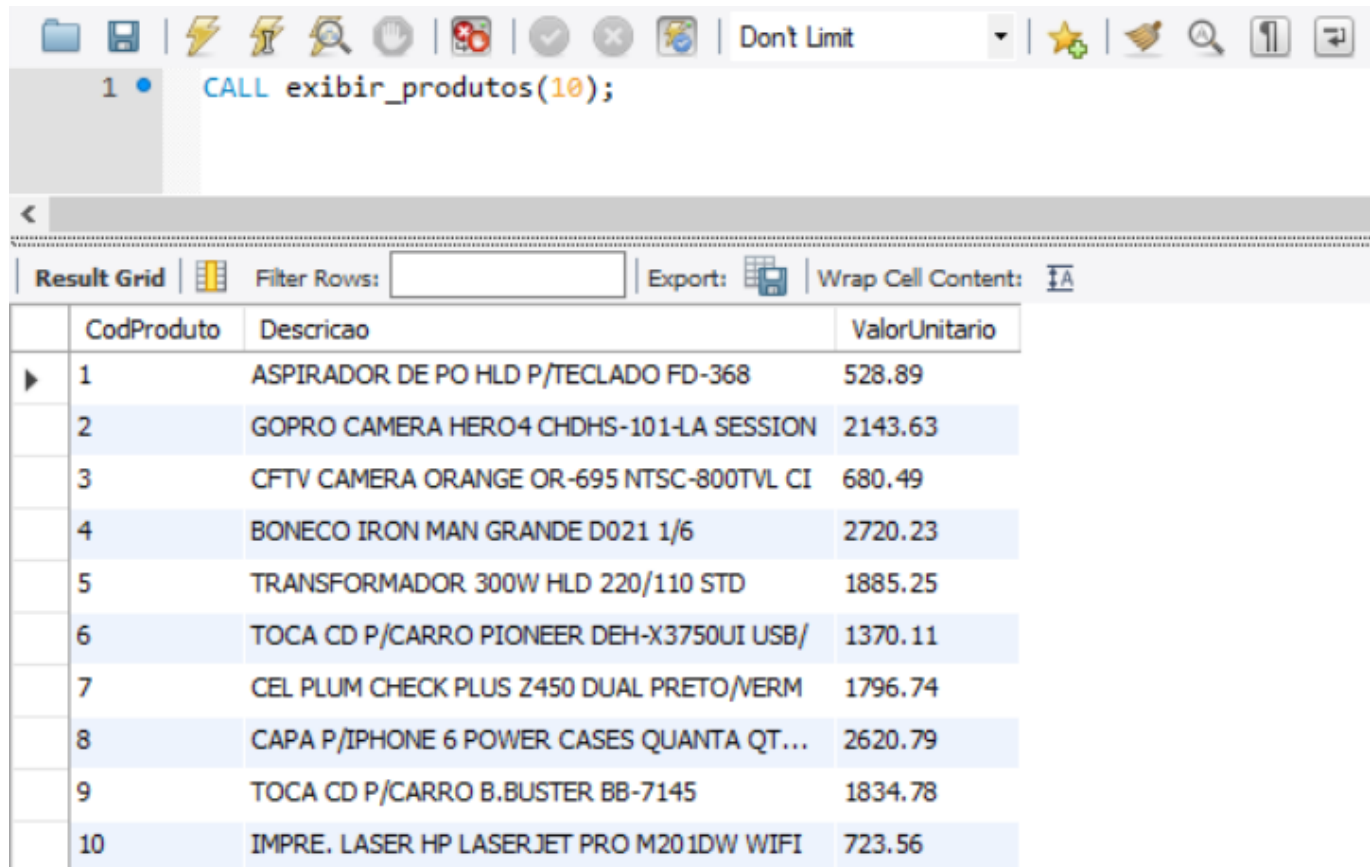
```
11
```

```
12
```

O que esse stored procedure faz?

Stored Procedures

- Exemplo de Stored Procedure com IN



The screenshot shows a database IDE interface. At the top, there is a toolbar with various icons for file operations, execution, and navigation. Below the toolbar, a SQL query is entered in a text area: `CALL exibir_produtos(10);`. Below the query, a "Result Grid" is displayed, showing the results of the stored procedure call. The grid has four columns: "CodProduto", "Descricao", and "ValorUnitario". The results are listed in a table with 10 rows, each representing a product. The first row is highlighted with a mouse cursor.

	CodProduto	Descricao	ValorUnitario
▶	1	ASPIRADOR DE PO HLD P/TECLADO FD-368	528.89
	2	GOPRO CAMERA HERO4 CHDHS-101-LA SESSION	2143.63
	3	CFTV CAMERA ORANGE OR-695 NTSC-800TVL CI	680.49
	4	BONECO IRON MAN GRANDE D021 1/6	2720.23
	5	TRANSFORMADOR 300W HLD 220/110 STD	1885.25
	6	TOCA CD P/CARRO PIONEER DEH-X3750UI USB/	1370.11
	7	CEL PLUM CHECK PLUS Z450 DUAL PRETO/VERM	1796.74
	8	CAPA P/IPHONE 6 POWER CASES QUANTA QT...	2620.79
	9	TOCA CD P/CARRO B.BUSTER BB-7145	1834.78
	10	IMPRE. LASER HP LASERJET PRO M201DW WIFI	723.56

Stored Procedures

- Exemplo de Stored Procedure com OUT

```
1 DELIMITER //
```

```
2 • CREATE PROCEDURE contar_produtos(OUT QUANTIDADE INT)
```

```
3 BEGIN
```

```
4     SELECT
```

```
5         COUNT(*) INTO QUANTIDADE
```

```
6     FROM
```

```
7         PRODUTO AS P;
```

```
8 END //
```

```
9 DELIMITER ;
```

```
10
```

Stored Procedures

- Exemplo de Stored Procedure com OUT

```
1 DELIMITER //
```

```
2 • CREATE PROCEDURE contar_produtos(OUT QUANTIDADE INT)
```

```
3 BEGIN
```

```
4     SELECT
```

```
5         COUNT(*) INTO QUANTIDADE
```

```
6     FROM
```

```
7         PRODUTO AS P;
```

```
8 END //
```

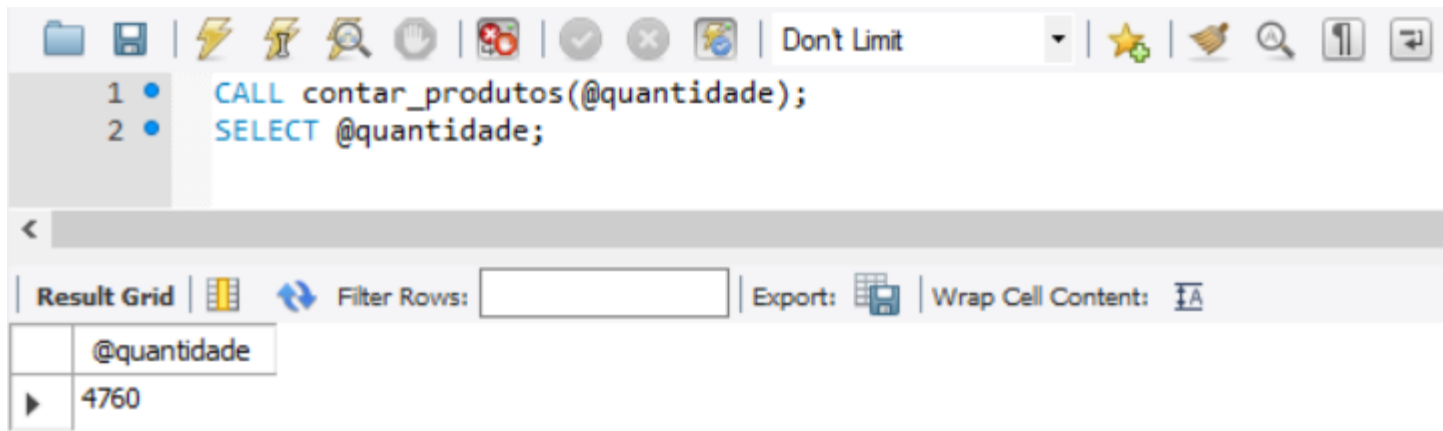
```
9 DELIMITER ;
```

```
10
```

Essa instrução nos permite armazenar o resultado em uma determinada variável

Stored Procedures

- Exemplo de Stored Procedure com OUT



The screenshot shows a database client interface with a toolbar at the top. The main area displays two lines of SQL code:

```
1 CALL contar_produtos(@quantidade);  
2 SELECT @quantidade;
```

Below the code, there is a "Result Grid" section. It includes a "Filter Rows" input field, an "Export" button, and a "Wrap Cell Content" checkbox. The result grid itself contains two columns: the first column is labeled "@quantidade" and the second column contains the value "4760".

	@quantidade
▶	4760

Stored Procedures

- Exemplo de Stored Procedure com INOUT

```
1 DELIMITER //
```

```
2 • CREATE PROCEDURE POTENCIA2(INOUT VALOR INT)
```

```
3 BEGIN
```

```
4     SET VALOR = VALOR * VALOR;
```

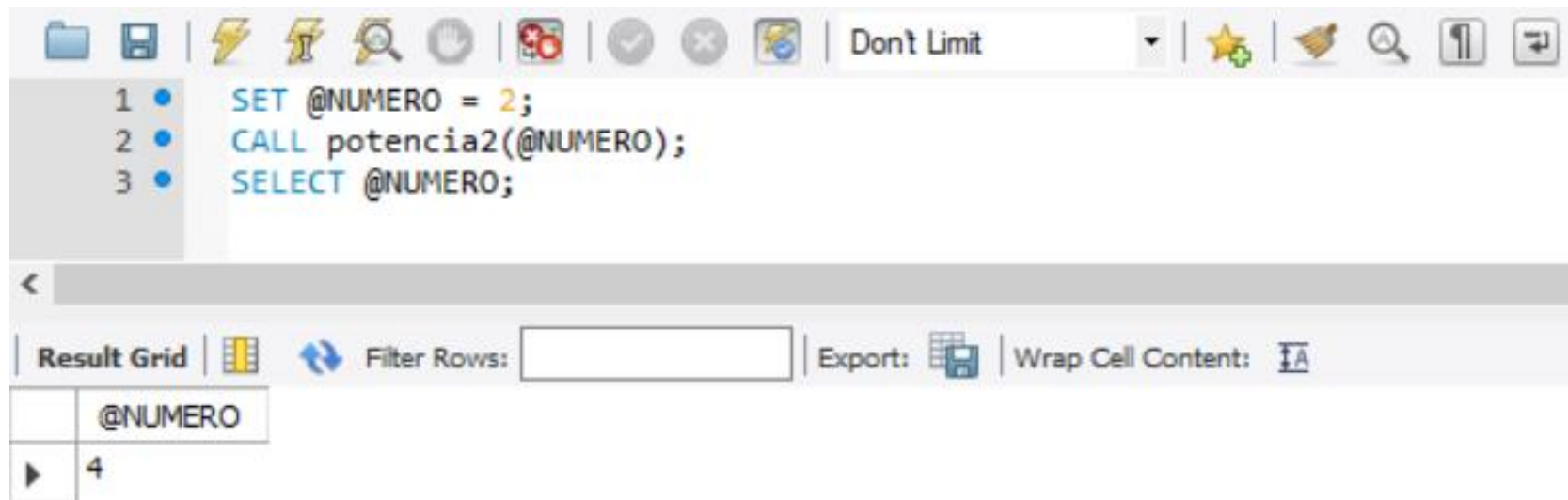
```
5 END //
```

```
6 DELIMITER ;
```

```
7
```

Stored Procedures

- Exemplo de Stored Procedure com INOUT



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Don't Limit' dropdown. The main editor area contains three lines of SQL code:

```
1 SET @NUMERO = 2;  
2 CALL potencia2(@NUMERO);  
3 SELECT @NUMERO;
```

Below the editor is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays a single row with the column header '@NUMERO' and the value '4'.

@NUMERO
4

Stored Procedures

- Exemplo de Stored Procedure com diversos parâmetros




```
1  DELIMITER //
2  CREATE PROCEDURE ANALISE_PRODUTOS(
3      OUT MAIOR_PRECO DECIMAL(10,2),
4      OUT MENOR_PRECO DECIMAL(10,2),
5      OUT QUANTIDADE INT
6  )
7  BEGIN
8      SELECT MAX(P.VALORUNITARIO) INTO MAIOR_PRECO FROM PRODUTO AS P;
9      SELECT MIN(P.VALORUNITARIO) INTO MENOR_PRECO FROM PRODUTO AS P;
10     SELECT COUNT(P.CODPRODUTO) INTO QUANTIDADE FROM PRODUTO AS P;
11 END //
12 DELIMITER ;
13
```

Stored Procedures

- Exemplo de Stored Procedure com diversos parâmetros

```
1 • CALL ANALISE_PRODUTOS(@MAIOR, @MENOR, @COUNT);  
2 • SELECT @MAIOR, @MENOR, @COUNT;
```

<

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

	@MAIOR	@MENOR	@COUNT
▶	2999.80	7.10	4760

Stored Procedures

- Obviamente podemos encapsular um stored procedure dentro de outro
- Podemos inclusive fazer a retirada da “chata” passagem de parâmetros

```
1  DELIMITER //
```

```
2  CREATE PROCEDURE ANALISE_PRODUTOS_SEM_PARAMETROS()
```

```
3  BEGIN
```

```
4      SET @MAIOR = 0;
```

```
5      SET @MENOR = 0;
```

```
6      SET @COUNT = 0;
```

```
7      CALL ANALISE_PRODUTOS(@MAIOR, @MENOR, @COUNT);
```

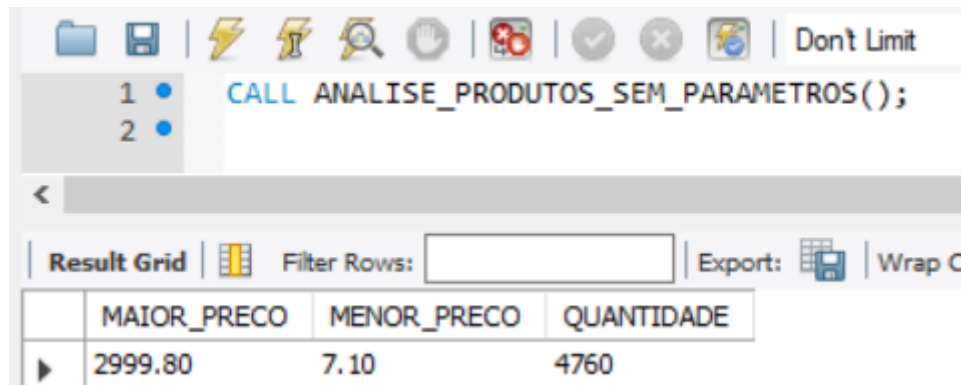
```
8      SELECT @MAIOR AS MAIOR_PRECO, @MENOR AS MENOR_PRECO, @COUNT AS QUANTIDADE;
```

```
9  END //
```

```
10 DELIMITER ;
```

Stored Procedures

- Invocando



- Note

- Agora não precisamos criar as variáveis de output
- Não precisamos fazer o select das variáveis

Stored Procedures

- Vamos criar um novo banco de dados

```
1 • CREATE DATABASE BANCO_DO_BUTUCA;
2
3 • USE BANCO_DO_BUTUCA;
4
5 • CREATE TABLE CONTA_CORRENTE (
6     NUMERO INTEGER NOT NULL PRIMARY KEY,
7     SALDO DECIMAL(10,2) NOT NULL,
8     NOME VARCHAR(100) NOT NULL);
9
10 • INSERT INTO CONTA_CORRENTE VALUES
11     (1001, 900.60, "JAMELÃO TROMBONNI"),
12     (1002, 200.05, "ALCIONE TROMBONNI"),
13     (1003, 1688.55, "ELSA SOARES TROMBONNI");
14
15 • SELECT * FROM CONTA_CORRENTE;
```

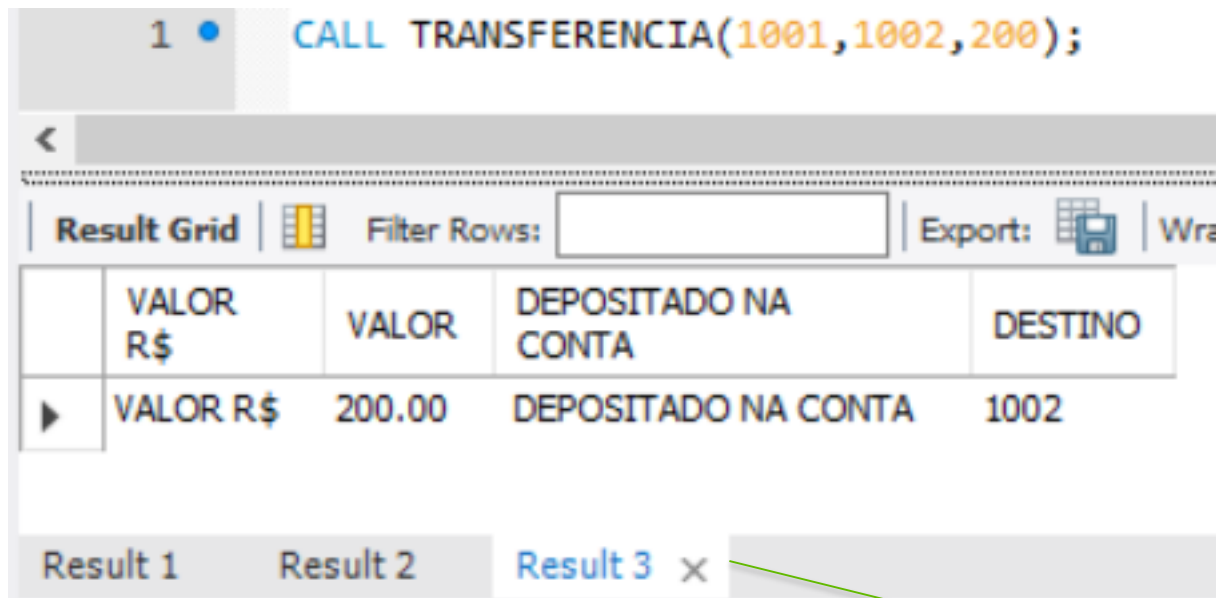
Stored Procedures

- Crie a seguinte stored procedure

```
1  DELIMITER //
2  • CREATE PROCEDURE TRANSFERENCIA(
3      IN ORIGEM INTEGER,
4      IN DESTINO INTEGER,
5      IN VALOR DECIMAL(10,2)
6  )
7  BEGIN
8      DECLARE SALDO_ORIGEM DECIMAL(10,2);
9      DECLARE SALDO_DESTINO DECIMAL(10,2);
10
11     SELECT SALDO INTO SALDO_ORIGEM FROM CONTA_CORRENTE WHERE NUMERO = ORIGEM;
12     SELECT SALDO INTO SALDO_DESTINO FROM CONTA_CORRENTE WHERE NUMERO = DESTINO;
13
14     IF SALDO_ORIGEM < VALOR THEN
15         SELECT "SALDO INSUFICIENTE PARA TRANSFERENCIA";
16     ELSE
17         SELECT "TRANSFERENCIA AUTORIZADA";
18         UPDATE CONTA_CORRENTE SET SALDO = (SALDO_ORIGEM - VALOR) WHERE NUMERO = ORIGEM;
19         SELECT "VALOR R$", VALOR, " SACADO DA CONTA ", ORIGEM;
20         UPDATE CONTA_CORRENTE SET SALDO = (SALDO_DESTINO + VALOR) WHERE NUMERO = DESTINO;
21         SELECT "VALOR R$", VALOR, " DEPOSITADO NA CONTA ", DESTINO;
22     END IF;
23 END //
24 DELIMITER ;
```

Stored Procedures

- Crie a seguinte stored procedure



The screenshot shows a database management tool interface. At the top, a SQL query is entered: `CALL TRANSFERENCIA(1001,1002,200);`. Below the query, there is a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap'. The main area displays a table with the following data:

	VALOR R\$	VALOR	DEPOSITADO NA CONTA	DESTINO
▶	VALOR R\$	200.00	DEPOSITADO NA CONTA	1002

At the bottom, there are tabs for 'Result 1', 'Result 2', and 'Result 3'. A green arrow points from the 'Result 3' tab to a green box containing text.

Observe os vários resultados gerados, olhe para cada um deles



TRIGGERS

Triggers

- Triggers ou Gatilhos são rotinas armazenadas associadas a eventos percebidos automaticamente pelo SGBD
 - Não há a necessidade de invocarmos explicitamente tais rotinas para estas serem executadas
 - Quando um determinado evento programado ocorre, a trigger (gatilho) é disparado automaticamente

Triggers

- **DELETE FROM PRODUTO WHERE CODPRODUTO = 1;**
 - Quando a tabela ItemPedido foi criado, foi definida uma chave estrangeira para Produto
 - Foi utilizada uma cláusula “ON DELETE RESTRICT” foi utilizada
 - Assim um erro é gerado ao tentar excluir um produto que possua pedidos cadastrados

Triggers

- Criando um trigger que dispara sempre que ocorre uma exclusão em PRODUTO

```
1 DELIMITER //
2 • CREATE TRIGGER EXCLUSAO_DE_PRODUTO
3   BEFORE DELETE ON PRODUTO
4     FOR EACH ROW
5     BEGIN
6         DELETE FROM ITEMPEDIDO WHERE CODPRODUTO = OLD.CODPRODUTO;
7     END //
8 DELIMITER ;
```

Triggers

- Criando um trigger que dispara uma exclusão em PRODUTO

Instrução que permite criar uma trigger

```
1 DELIMITER //
2 • CREATE TRIGGER EXCLUSAO_DE_PRODUTO
3   BEFORE DELETE ON PRODUTO
4     FOR EACH ROW
5     BEGIN
6         DELETE FROM ITEMPEDIDO WHERE CODPRODUTO = OLD.CODPRODUTO;
7     END //
8 DELIMITER ;
```

Código que será executado quando a trigger é disparada

Triggers

- Criando uma exceção que ocorre

Define qual evento disparará a trigger. Os valores aceitos são:

INSERT, REPLACE, DELETE ou UPDATE

```
1 DELIMITER //  
2 CREATE TRIGGER EXCLUSAO_DE_PRODUTO  
3 BEFORE DELETE ON PRODUTO  
4 FOR EACH ROW  
5 BEGIN  
6     DELETE FROM ITEMPEIDIDO WHERE CODPRODUTO = OLD.  
7 END //
```

Tabela na qual a trigger ficará monitorando eventos

Define que a trigger será executada antes da operação que a disparou. Os valores aceitos são:

BEFORE: executa a trigger antes da operação

AFTER: executa a trigger depois da operação

Triggers

- Criando um trigger que dispara sempre que ocorre uma exclusão em PRODUTO

```
1 DELIMITER //
2 • CREATE TRIGGER EXCLUSAO_DE_PRODUTO
3 BEFORE DELETE ON PRODUTO
4 FOR EACH ROW
5 BEGIN
6     DELETE FROM ITEMPEDIDO WHERE CODPRODUTO = OLD.CODPRODUTO;
7 END //
8 DELIMITER ;
```

Dados utilizados nas instruções **OLD (dados antigos)** e **NEW (novos dados)**. Lembrando que:

- Triggers sobre **INSERT** possuem apenas **NEW**
- Triggers sobre **DELETE** possuem apenas **OLD**
- Triggers sobre **UPDATE** ou **REPLACE** possuem ambos **OLD** e **NEW**

Triggers

- Criando um trigger que dispara sempre que ocorre uma exclusão em PRODUTO

```
1 DELIMITER //
2 • CREATE TRIGGER EXCLUSAO_DE_PRODUTO
3   BEFORE DELETE ON PRODUTO
4     FOR EACH ROW
5     BEGIN
6         DELETE FROM ITEMPEDIDO WHERE CODPRODUTO = OLD.CODPRODUTO;
7     END //
8 DELIMITER ;
```

Dados utilizados nas instruções **OLD (dados antigos)** e **NEW (novos dados)**. Lembrando que:

- Triggers sobre **INSERT** possuem apenas **NEW**
- Triggers sobre **DELETE** possuem apenas **OLD**
- Triggers sobre **UPDATE** ou **REPLACE** possuem ambos **OLD** e **NEW**

Triggers

- Vamos fazer algumas alterações no BANCO_DO_BUTUCA

```
1 • USE BANCO_DO_BUTUCA;
2
3 • CREATE TABLE MOVIMENTACAO(
4     NUMERO INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
5     CONTA INTEGER NOT NULL,
6     SALDO_ANTERIOR DECIMAL(10,2) NOT NULL,
7     SALDO_ATUAL DECIMAL(10,2) NOT NULL
8 );
```


Triggers

- Vamos fazer algumas alterações no BANCO_DO_BUTUCA

```
1  DELIMITER //
2
3  • CREATE TRIGGER ATUALIZA_MOVIMENTACAO
4  AFTER UPDATE ON CONTA_CORRENTE
5  FOR EACH ROW
6  BEGIN
7      INSERT INTO MOVIMENTACAO
8          (CONTA, SALDO_ANTERIOR, SALDO_ATUAL)
9      VALUES
10         (OLD.NUMERO, OLD.SALDO, NEW.SALDO);
11  END
12  //
13
14  DELIMITER ;
```

O que esse trigger faz?

Triggers

- Vamos fazer algumas alterações no BANCO_DO_BUTUCA

```
1 • CALL TRANSFERENCIA(1002,1001,100);
2
3 • SELECT * FROM MOVIMENTACAO;
```

<

Result Grid

Filter Rows:

Edit:

	NUMERO	CONTA	SALDO_ANTERIOR	SALDO_ATUAL
▶	1	1002	600.05	500.05
	2	1001	500.60	600.60
*	NULL	NULL	NULL	NULL

EXERCÍCIOS

- Utilizando o BD compubras crie os seguintes stored procedures
 1. Exiba a quantidade total vendida dos produtos (group by por codProduto) (exibir id, nome do produto e quantidade)
 2. Exiba todas as vendas efetuadas (número do pedido, cliente, total da venda e o vendedor)
 3. Calcule a comissão para todos os vendedores com base nas vendas do mês/ano (mês e ano serão parâmetros IN)

EXERCÍCIOS

- Utilizando o BD banco_do_butuca crie os seguintes stored procedures
 1. Efetuar o saque em uma conta corrente
 2. Efetuar o depósito em uma conta corrente
 3. Listar contas com saldo maior que o valor informado
 4. Setar o saldo de todas as contas (valor informado)
 - REMOVER O SAFE UPDATES NAS CONFIGS

EXERCÍCIOS

- Utilizando o BD banco_do_butuca crie os seguintes triggers
 1. Ao deletar uma conta apague também as movimentações salvas (AFTER)
 2. Ao criar uma conta salvar um registro em movimentação com saldo antigo e atual igual a 0 (AFTER)
 3. Antes de cadastrar uma nova conta verifique se o saldo informado é igual a 0, se não for setar para 0 (BEFORE)
 4. Antes de deletar uma conta, salvar um registro em movimentação de que a conta está zerada (BEFORE)

A photograph of a calm lake with several wooden boats in the foreground. The boats are dark brown and appear to be traditional rowing boats. The water is still, reflecting the sky and the surrounding trees. In the background, there is a dense line of green trees under a clear sky. The overall atmosphere is peaceful and natural.

Stored Procedures e Triggers

Professor: Ricardo Luis dos Santos

IFSUL – Campus Sapucaia do Sul