

S112 – Banco de Dados

Professor MSc. Eng. Márcio José de Lemos

E-mail: marcio.lemos@senairs.org.br

<http://lattes.cnpq.br/4769158065464009>

Projeto de Banco de Dados



SQL

A linguagem **SQL** é basicamente dividida em **três tipos de comandos**:

SQL = DDL + DML + DCL

SQL = DDL + DML + DCL

DDL (definição de dados)

- Comandos: CREATE, DROP, ALTER

DML (manipulação de dados)

- Comandos: SELECT, INSERT, UPDATE e DELETE

DCL (controle de dados)

- Comandos: GRANT e REVOKE

Exercícios SQL

Tabela de CDs			
Campo	Tipo	Tamanho	Chave
Código	+		*
Nome	A	50	
DataCompra	D		
ValorPago	\$		
LocalCompra	A		
Album	L		

Tabela de Musicas			
Campo	Tipo	Tamanho	Chave
CodigCD	1		*
Numero	1		*
Nome	A	50	
Artista	A	50	
Tempo	T		

- 1) Criar o Banco de dados (Discoteca) e posteriormente as tabelas (tab-cd e tab-musica);
- 2) Definir indices para as tabelas;
- 3) Inserir alguns dados;
- 4) Mostrar todos os cds;
- 5) Mostrar os campos nome e data da compra dos cds ordenados por nome;
- 6) Mostrar o total gasto com a compra dos Cds;
- 7) Mostrar todas as músicas (todos os campos) do cds código 1;
- 8) Mostre a quantidade de músicas cadastradas;
- 9) Mostre a média de duração das músicas cadastradas;
- 10) Mostre uma listagens de músicas em ordem alfabética.



Criar o Banco de dados (DBDiscoteca) :

```
CREATE DATABASE DBdiscoteca;
```

Criar as tabelas (tab_cd e tab_musica):

```
CREATE TABLE tab_cd (codigo int(5), nome char(30), datacompra date, valorpago float, localcompra char(20), album boolean, PRIMARY KEY (codigo) );
```

```
CREATE TABLE tab_musica (codigocd int(5), numero int(5), nome char(50), artista char(50), tempo time, PRIMARY KEY (codigocd, numero) );
```


Exercícios SQL

Inserir alguns dados:

```
INSERT INTO tab_cd VALUES (1,"Balada 2015", 2020/05/01, 18.50, "submarino", True);  
INSERT INTO tab_cd VALUES (2,"Swing Total", 2020/05/02, 20.00, "submarino", TRUE);  
INSERT INTO tab_cd VALUES (3,"Amigos para Sempre", 2020/05/03, 40.00,  
"Americanas", false);
```

```
INSERT INTO tab_musica VALUES (1,1, "Estou De Olho", "Guto Boy", 2.55);  
INSERT INTO tab_musica VALUES (1,2, "Virando o Copo", "Guto Boy", 5);  
INSERT INTO tab_musica VALUES (1,2, "Vivendo Cada Dia", "Guto Boy", 8);
```



Exercícios SQL

Mostrar todos os cds;

```
Select * from tab_cd;
```

**Mostrar os campos nome e data da compra dos cds
ordenados por nome;**

```
Select nome, datacompra From tab_cd Order By nome;
```

Mostrar o total gasto com a compra dos Cds;

Select Sum(valorpago) as Total From tab_cd;

Mostrar todas as músicas (todos os campos) do cdscódigo 1;

Select * From tab_musica Where codigocd=1;

Mostre a quantidade de músicas cadastradas;

Select Count(*) as Qtde From tab_musica;

Mostre a média de duração das músicas cadastradas;

Select AVG(tempo) as Media From tab_musica;

Select AVG(tempo) as Media From tab_musica;



Mostre uma listagens de músicas em ordem alfabética;

Select * From tab_musica Order By nome;

Mostrar todos os cdsque são albuns;

Select * From tab_cd Where album=True;

Mostre o cdque custou mais caro.

Select Max(valorpago) From tab_cd;

Triggers em MySQL

Um trigger (“**gatilho**”) é um objeto programável do banco de dados associado a uma tabela. Trata-se de um procedimento que é invocado automaticamente quando um comando **DML** é executado na tabela, sendo executado para cada linha afetada.

Triggers em MySQL

Desta forma, as operações que podem disparar um **trigger** são:

- INSERT
- UPDATE
- DELETE

Geralmente, os triggers são empregados para verificar integridade dos dados, fazer validação dos dados e outras operações relacionadas.

Diferença entre Trigger e Procedimento Armazenado

Tanto os **triggers** quanto as **stored procedures** são objetos programáveis de um banco de dados. Porém, eles possuem diferenças importantes entre si, que afetam o modo como são aplicados.



Diferença entre Trigger e Procedimento Armazenado

Algumas das principais diferenças entre trigger e procedimentos armazenados são:

- Um Trigger é associado a uma tabela.
- Os triggers são armazenados no próprio banco de dados como arquivos separados.
- Triggers não são chamados diretamente, sendo invocados automaticamente, ao contrário dos procedimentos armazenados.

Diferença entre Trigger e Procedimento Armazenado

Algumas das principais diferenças entre trigger e procedimentos armazenados são:

- **Procedimentos** armazenados podem trabalhar com parâmetros; **Não** passamos parâmetros aos triggers.
- Os **triggers** não retornam um conjunto de resultados (**resultset**) ao cliente chamador.

Aplicações dos triggers

As principais aplicações dos triggers em bancos de dados são:

- Validação de Dados (tipos de dados, faixas de valores, etc).
- Rastreamento e registro de logs de atividades em tabelas.
- Verificação de integridade de dados e consistência
- Arquivamento de registros excluídos.

Modos de Disparo de um Trigger

Um Trigger em MySQL pode ser disparado de dois modos diferentes:

- **BEFORE** – O trigger é disparado e seu código executado ANTES da execução de cada evento – por exemplo, antes de cada inserção de registros na tabela.
- **AFTER** – O código presente no trigger é executado após todas as ações terem sido completadas na tabela especificada.

Sintaxe para criação de um trigger em MySQL

Para criar um trigger em MySQL usamos a seguinte sintaxe:

CREATE TRIGGER nome timing operação

ON tabela

FOR EACH ROW

Declarações

Sintaxe para criação de um trigger em MySQL

- **timing** pode ser BEFORE ou AFTER
- **operação** pode ser INSERT / UPDATE / DELETE

Exemplo 1 trigger em MySQL

Neste exemplo criaremos uma tabela chamada *Produto*, que conterá os seguintes dados:

- Nome do produto
- Identificação do produto (chave primária)
- Preço normal
- Preço com desconto a ser aplicado

Exemplo 1 trigger em MySQL

Logo após criaremos um **trigger** de nome ***tr_desconto***, cuja função será aplicar um valor de desconto de **10%** à coluna **Preco_Desconto** quando for disparado.

Ou seja, **todos os produtos terão seu preço reduzido em 10%** nesta coluna. O trigger será disparado ao inserir um novo registro na tabela.

Exemplo 1 trigger. Criar a tabela Produto:

Criar a tabela de exemplo 1 trigger :

```
CREATE TABLE Produto (  
  idProduto INT NOT NULL AUTO_INCREMENT,  
  Nome_Produto VARCHAR(45) NULL,  
  Preço_Normal DECIMAL(10,2) NULL,  
  Preço_Desconto DECIMAL(10,2) NULL,  
  PRIMARY KEY (idProduto));
```

Exemplo 1 trigger. Criar a tr_desconto

Criar o Trigger:

```
CREATE TRIGGER tr_desconto BEFORE INSERT  
ON Produto  
FOR EACH ROW  
SET NEW.Preco_Desconto = (NEW.Preco_Normal * 0.90);
```

Exemplo 1 trigger.

Inserção que irá disparar o Trigger:

```
INSERT INTO Produto (Nome_Produto, Preco_Normal)  
VALUES ("DVD", 1.00), ("Pendrive", 18.00);
```



Exemplo 1 trigger.

Trigger foi disparado observando o preço com desconto:

```
SELECT * FROM Produto;
```


Como excluir (delete) um trigger

Para excluir um trigger em MySQL usamos a declaração **DROP TRIGGER**, seguida do nome do trigger, como no exemplo:

```
DROP TRIGGER tr_desconto;
```

RESUMO

Triggers (“gatilhos” em português) são objetos do banco de dados que, relacionados a certa tabela, permitem a realização de processamentos em consequência de uma determinada ação como, por exemplo, a inserção de um registro.

RESUMO

Os **triggers** podem ser **executados ANTES ou DEPOIS** das operações de **INSERT, UPDATE e DELETE** de registros.

- Observação: o suporte a triggers foi incluído na versão 5.0.2 do MySQL.

RESUMO

Prós e Contras dos Triggers

- Os principais pontos positivos sobre os triggers são:
- Parte do processamento que seria executado na aplicação passa para o banco, poupando recursos da máquina cliente.
- Facilita a manutenção, sem que seja necessário alterar o código fonte da aplicação.



RESUMO

Prós e Contras dos Triggers

- Requer maior conhecimento de manipulação do banco de dados (**SQL**) para realizar as operações internamente.

EXEMPLO 2 trigger

- Um mercado que, ao realizar vendas, precisa que o estoque dos produtos seja automaticamente reduzido.
- A devolução do estoque deve também ser automática no caso de remoção de produtos da venda.

EXEMPLO 2 trigger

- Como se trata de um ambiente hipotético, teremos apenas duas tabelas de estrutura simples.

EXEMPLO 2 trigger

```
CREATE TABLE Produtos
(
    Referencia    VARCHAR(3) PRIMARY KEY,
    Descricao     VARCHAR(50) UNIQUE,
    Estoque       INT NOT NULL DEFAULT 0
);
```

EXEMPLO 2 trigger

```
INSERT INTO Produtos VALUES ('001', 'Feijão', 10);  
INSERT INTO Produtos VALUES ('002', 'Arroz', 5);  
INSERT INTO Produtos VALUES ('003', 'Farinha', 15);
```

EXEMPLO 2 trigger

```
CREATE TABLE ItensVenda  
(  
    Venda          INT,  
    Produto VARCHAR(3),  
    Quantidade     INT  
);
```

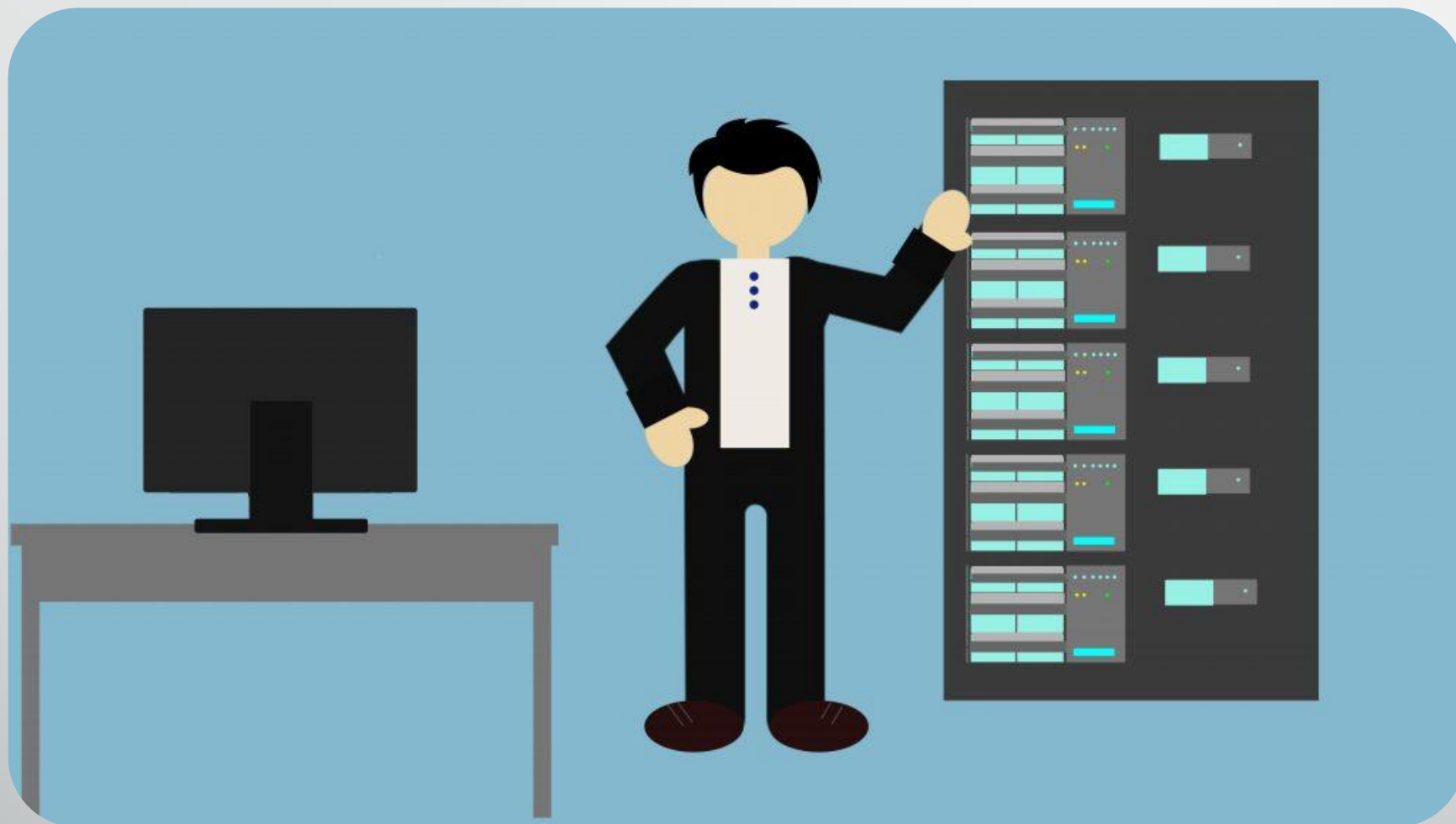
EXEMPLO 2 trigger

- Ao inserir e remover registro da tabela **ItensVenda**, o estoque do produto referenciado deve ser **alterado na tabela Produtos**.

EXEMPLO 2 trigger

Para isso, serão criados **dois triggers**:

- um ***AFTER INSERT*** para dar baixa no estoque e
- um ***AFTER DELETE*** para fazer a devolução da quantidade do produto.



Criando os gatilhos

```
DELIMITER $
```

```
CREATE TRIGGER Tgr_ItensVenda_Insert AFTER INSERT  
ON ItensVenda  
FOR EACH ROW  
BEGIN  
    UPDATE Produtos SET Estoque = Estoque - NEW.Quantidade  
WHERE Referencia = NEW.Produto;  
END$
```

```
END$
```

```
MHEKE B6f6L6UCf9 = IEM'pLoqnf0?
```

Criando os gatilhos

```
DELIMITER $  
CREATE TRIGGER Tgr_ItensVenda_Delete AFTER DELETE  
ON ItensVenda  
FOR EACH ROW  
BEGIN  
    UPDATE Produtos SET Estoque = Estoque + OLD.Quantidade  
WHERE Referencia = OLD.Produto;  
END$  
DELIMITER ;
```

Criando os gatilhos Explicação

- No **primeiro gatilho**, foi utilizado o registro **NEW** para obter as informações da linha que está sendo inserida na tabela.
- O mesmo é feito no **segundo gatilho**, onde se obtém os dados que estão sendo apagados da tabela através do registro **OLD**.

Criando os gatilhos –TESTANDO

- Tendo criado os **triggers**, podemos testá-los inserindo dados na tabela **ItensVenda**.

TESTE PRÁTICO!!

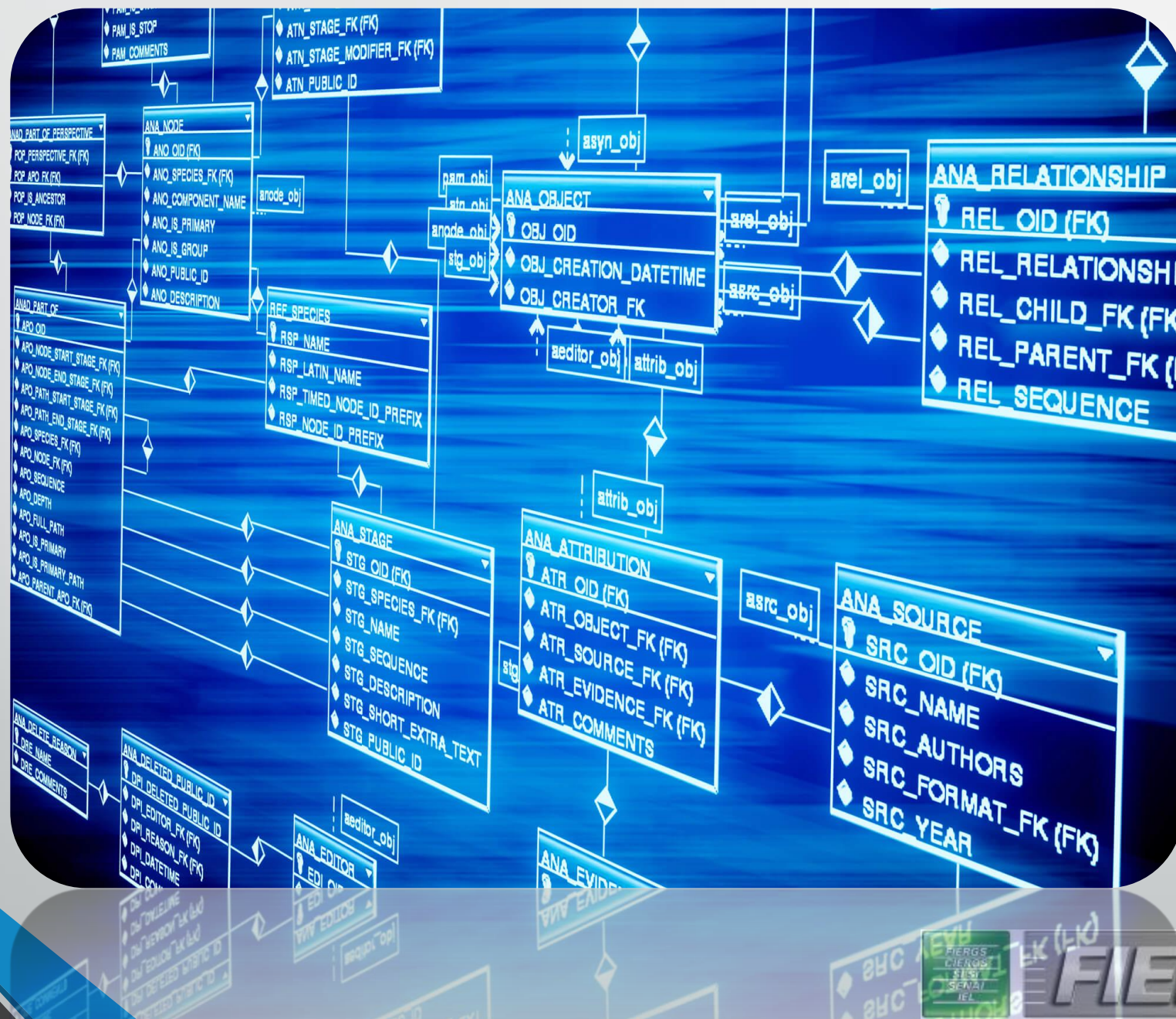
- Nesse caso, vamos simular uma
- **venda de número 1** que contem **três** unidades do produto **001**,
- uma unidade do produto **002** e
- **cinco** unidades do produto **003**.

Inserindo dados na tabela

```
INSERT INTO ItensVenda VALUES (1, '001',3);  
INSERT INTO ItensVenda VALUES (1, '002',1);  
INSERT INTO ItensVenda VALUES (1, '003',5);
```

Excluindo dados da tabela ItensVenda

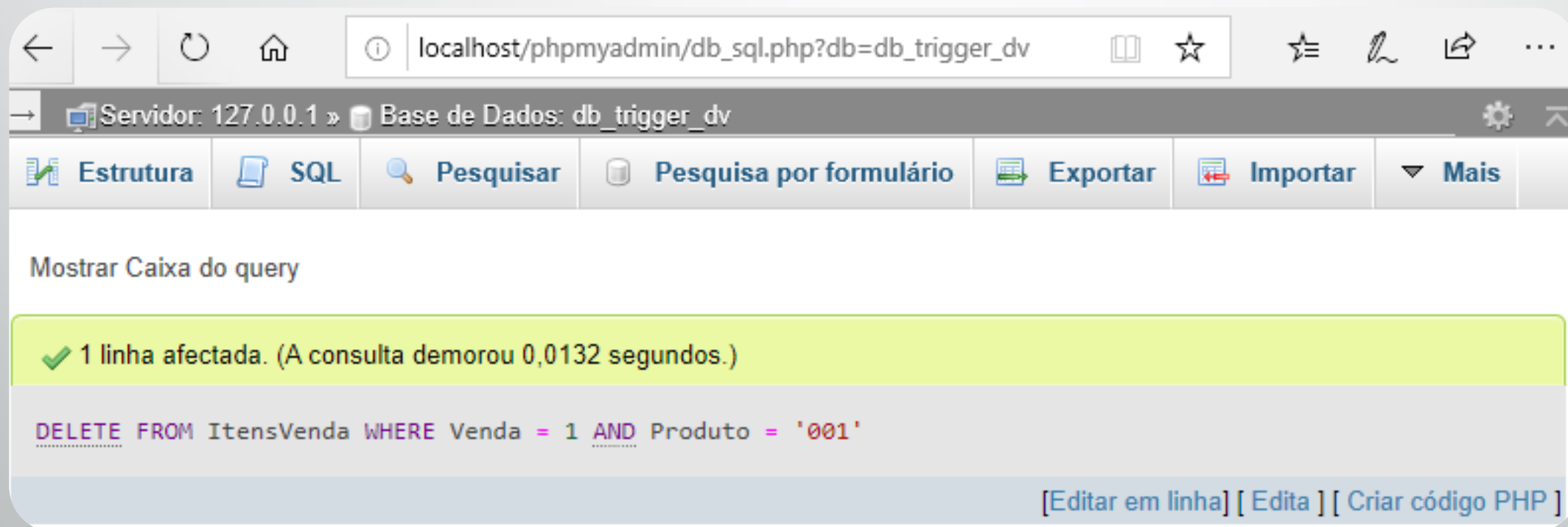
```
DELETE FROM ItensVenda WHERE Venda = 1 AND Produto = '001';
```



Excluindo dados da tabela ItensVenda



Excluindo dados da tabela ItensVenda



The screenshot shows the phpMyAdmin interface in a web browser. The address bar displays 'localhost/phpmyadmin/db_sql.php?db=db_trigger_dv'. The interface shows the 'Base de Dados: db_trigger_dv' selected. The 'SQL' tab is active, and the query 'DELETE FROM ItensVenda WHERE Venda = 1 AND Produto = '001'' has been executed. A green message box indicates '1 linha afectada. (A consulta demorou 0,0132 segundos.)'. Below the message, the executed SQL query is shown. At the bottom right, there are links for '[Editar em linha]', '[Edita]', and '[Criar código PHP]'.

localhost/phpmyadmin/db_sql.php?db=db_trigger_dv

Servidor: 127.0.0.1 » Base de Dados: db_trigger_dv

Estrutura SQL Pesquisar Pesquisa por formulário Exportar Importar Mais

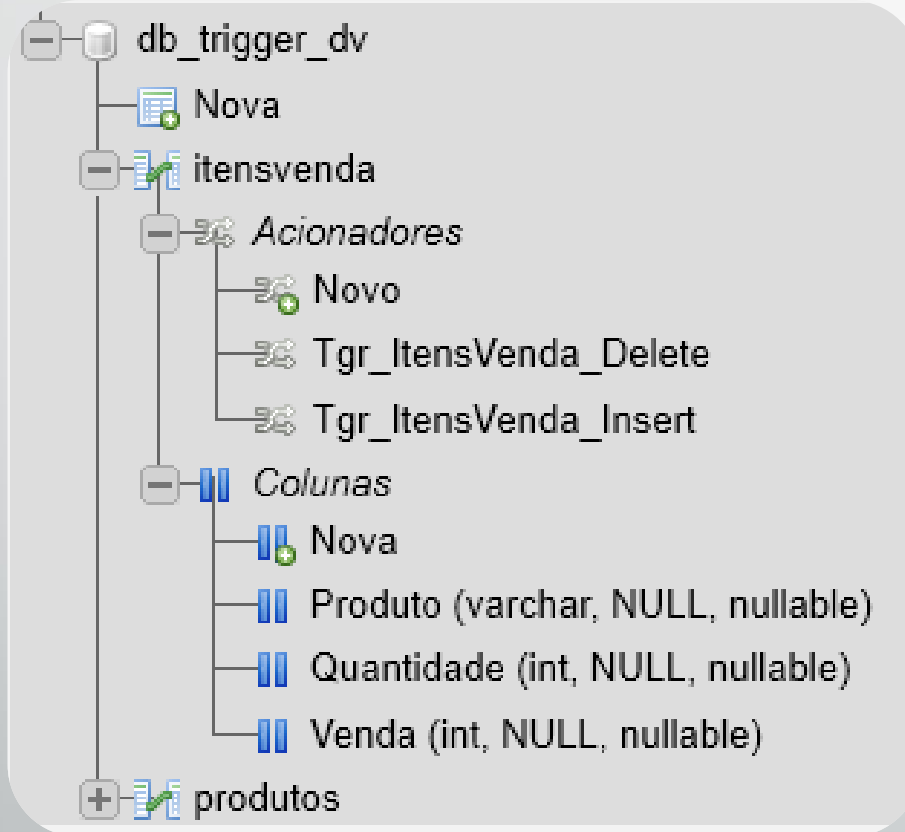
Mostrar Caixa do query

✓ 1 linha afectada. (A consulta demorou 0,0132 segundos.)

```
DELETE FROM ItensVenda WHERE Venda = 1 AND Produto = '001'
```

[Editar em linha] [Edita] [Criar código PHP]

Estrutura do Banco



Estrutura da Trigger- Tgr_ItensVenda_Delete

The screenshot displays the phpMyAdmin interface. On the left, a tree view shows the database structure, including a trigger named 'Tgr_ItensVenda_Delete' under the 'db_trigger_dv' database. The main panel shows the details of this trigger:

- Nome do gatilho:** Tgr_ItensVenda_Delete
- Tabela:** itensvenda
- Tempo:** AFTER
- Evento:** DELETE
- Definição:**

```
1 BEGIN
2   UPDATE Produtos SET Estoque =
   Estoque + OLD.Quantidade
3 WHERE Referencia = OLD.Produto;
4 END
```
- Definidor:** root@localhost

Buttons for 'Executar' (Execute) and 'Fechar' (Close) are visible at the bottom of the details panel.

Estrutura da Trigger- Tgr_ItensVenda_Insert

The screenshot displays the phpMyAdmin interface for a database named 'db_trigger_dv'. The left sidebar shows the database structure, including tables like 'db_trigger', 'db_trigger_10_porcento', 'produto', and 'itensvenda'. The 'itensvenda' table is selected, and its triggers are listed: 'Acionadores', 'Novo', 'Tgr_ItensVenda_Delete', and 'Tgr_ItensVenda_Insert'. The 'Tgr_ItensVenda_Insert' trigger is selected, and its details are shown in the main panel.

<title>localhost / 127.0.0.1 / db_trigger_dv | phpMyAdmin 5.0.1</title>

Detalhes

Nome do gatilho	Tgr_ItensVenda_Insert
Tabela	itensvenda
Tempo	AFTER
Evento	INSERT

Definição

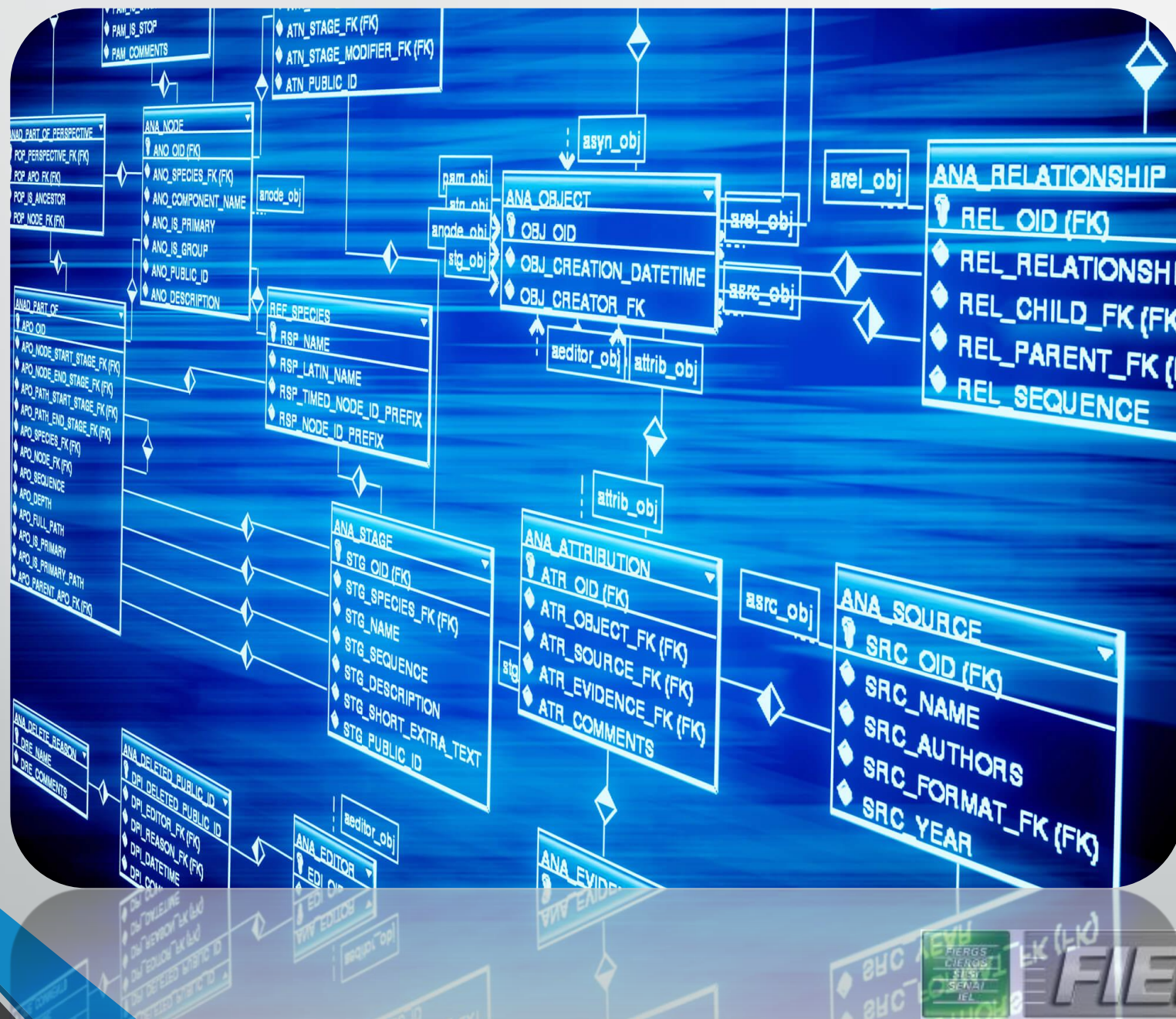
```
1 BEGIN
2     UPDATE Produtos SET Estoque =
    Estoque - NEW.Quantidade
3 WHERE Referencia = NEW.Produto;
4 END
```

Definidor root@localhost

Executar Fechar

Agora, fazendo uma consulta à tabela Produtos

```
SELECT * FROM 'produtos';
```

Obrigado pela atenção

Sigo à disposição pelo e-mail:

marcio.lemos@senairs.org.br