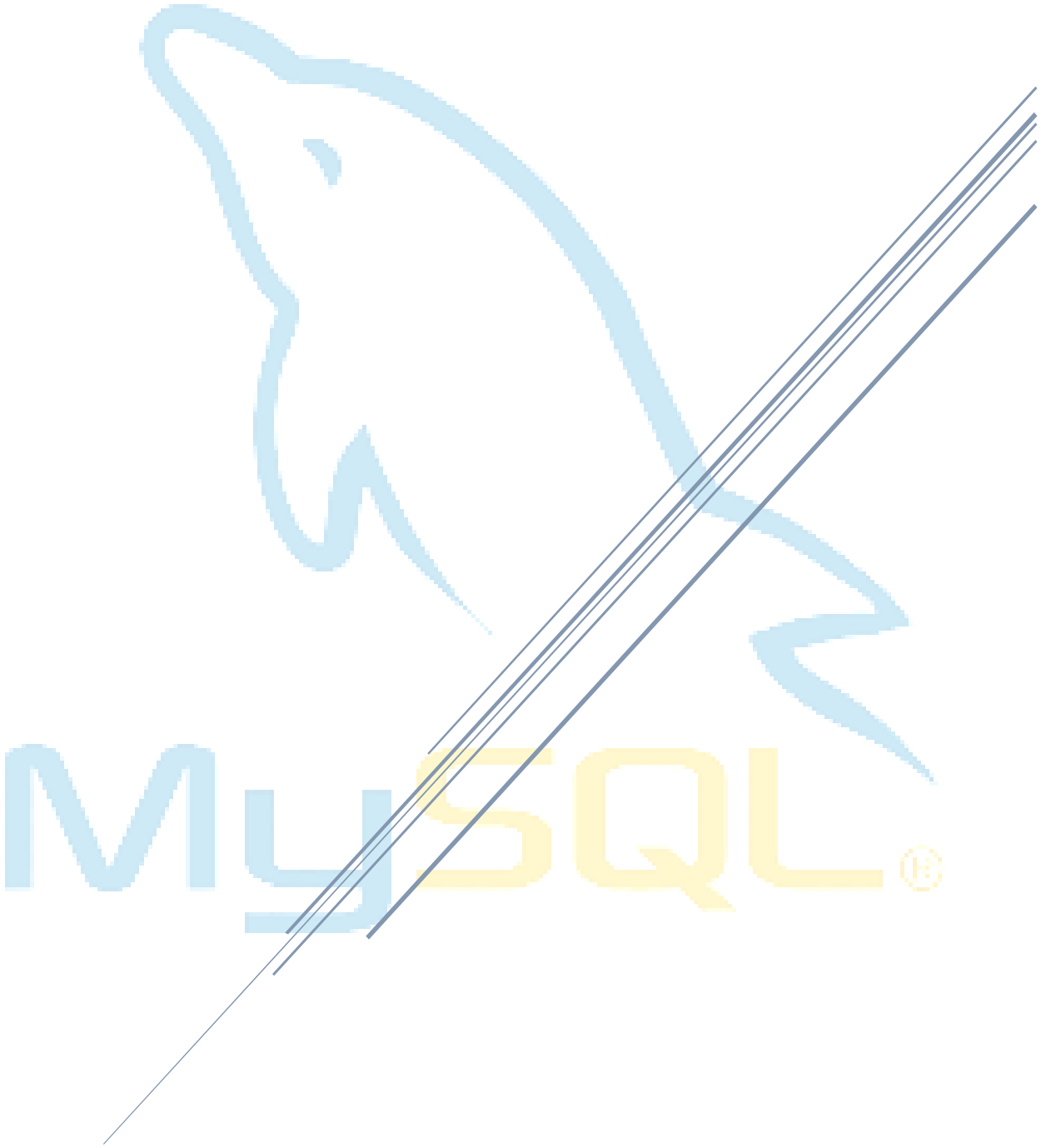
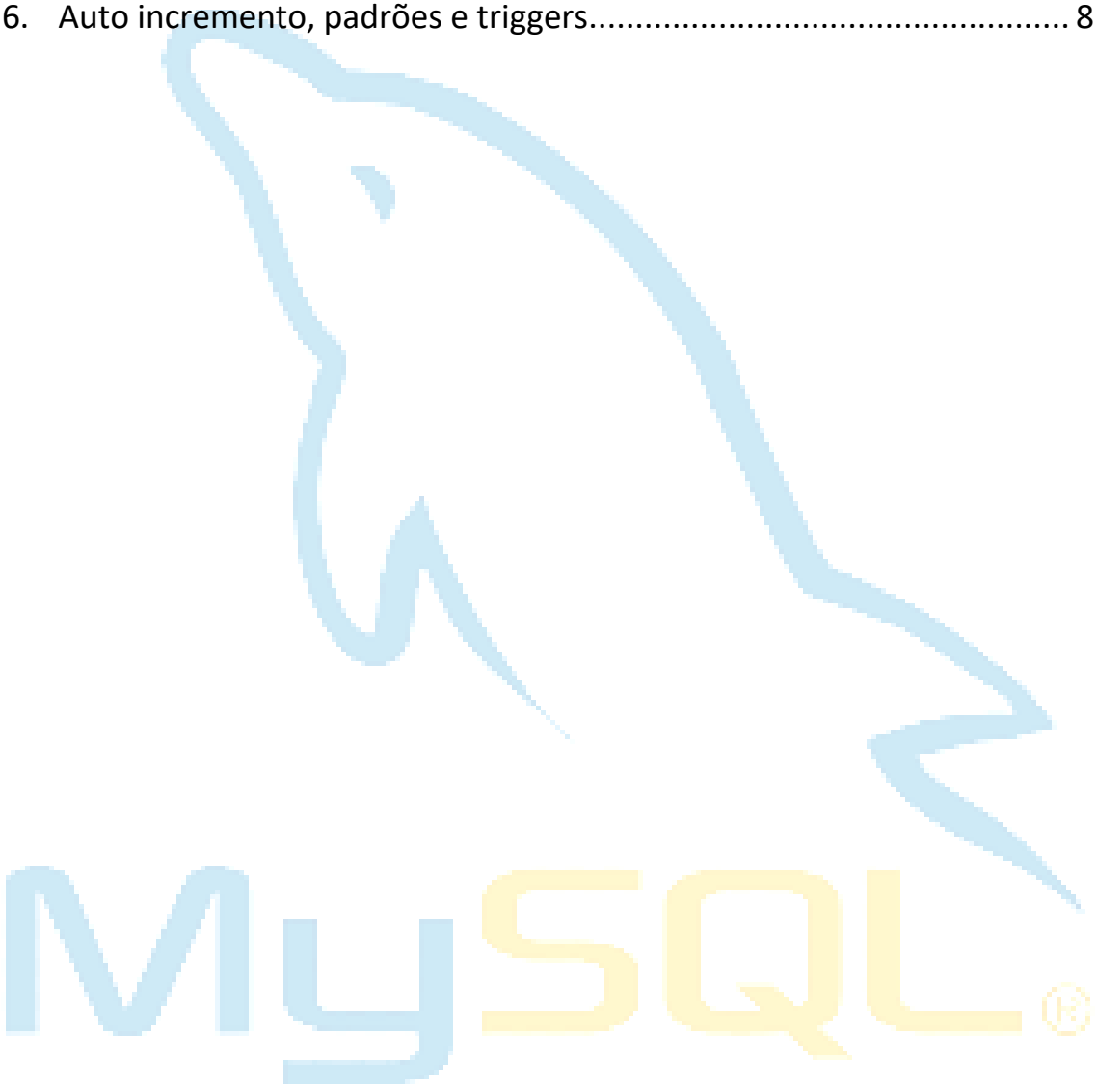
COMANDOS DML

Manipulação de dados com MySQL



Alura
MySQL

Sumário



1. Modelagem do Banco de dados.

1.1 Revisão – entidades

Uma entidade é uma representação de um conjunto de informações sobre determinado conceito do sistema. Toda entidade possui ATRIBUTOS, que são as informações que referenciam a entidade.

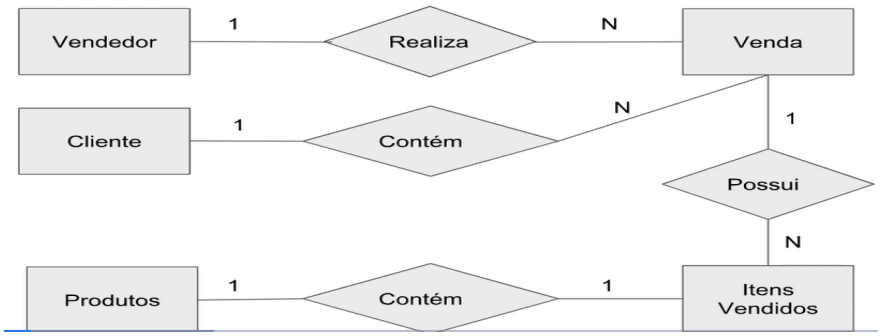
Um banco MySQL possui dentro da sua estrutura do seu servidor um conjunto de uma entidade que nós chamamos de banco de dados.

Tabela.
índice.
Visão.
Procedures.
Trigger.

1.2 Modelagem

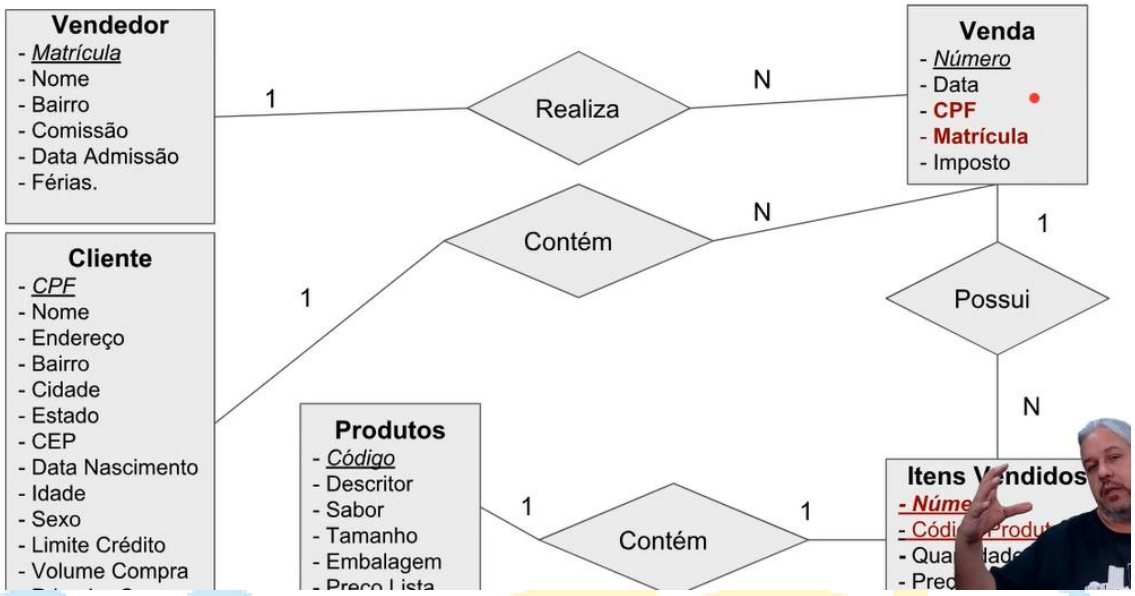
Para montar um banco de dados, precisamos primeiramente conhecer bem o projeto que será executado e que enviara as informações para nosso banco de dados, para isso precisamos seguir alguns passos.

- **Análise dos requisitos:**
  Entendimento das regras de negócio;
  Entender as regras de negócio da empresa e do software que enviara as informações para nosso banco de dados.
  Efetuar atividades de entrevistas e reuniões;
  Essa parte do processo é muito importante para saber quais vão ser as informações que serão necessárias inserir no sistema.
  Desenho de modelo mais fiel a realidade;
  Tentar reproduzir um modelo o mais fiel possível da realidade do cliente e software.
- **Modelo conceitual**
  Construção do diagrama de entidade e relacionamento;



Estabelecer a cardinalidade das entidades;

- **Atributos das entidades**
  Estabelece características de cada entidade;



- **Transformar o diagrama de entidades em banco de dados**
  Transformar cada entidade em uma ou mais tabelas físicas do banco de dados.
  Cada relação da modelagem será um relacionamento nas tabelas do banco de dados.

- **Construir o banco de dados**
  Podemos usar ferramentas CASE para isso.
  Ferramentas onde desenhamos graficamente as tabelas e ele nos retornara os códigos para criação das mesmas dentro do ambiente SQL.

2. Criando a estrutura do banco de dados

2.1 Criando e deletando um banco de dados

Para criar um novo banco de dados dentro do Mysql digite o comando

```
CREATE DATABASE vendas_sucos2 DEFAULT CHARACTER SET utf8;
```

Onde DEFAULT CHARACTER SET uft8 seria a linguagem utilizada pelo o nosso banco de dados.

Caso tiver dúvida se o banco de dados a ser criado existe ou não, podemos usar o comando IF NOT EXISTS. Este comando é usado em diversas entidades do MYSQL e é usado para que o processo de criação aconteça apenas se o componente não existe.

Para apagar um banco de dados.

3. Incluindo dados nas tabelas

Temos algumas maneiras de inserir dados dentro de uma tabela, a mais simples e utilizando o comando **INSERT INTO** de uma forma bem genérica:

```
USE vendas_sucos;


INSERT INTO PRODUTOS (CODIGO, DESCRITOR, SABOR, TAMANHO, EMBALAGEM,
PRECO_LISTA)

VALUES ('1040107', 'Light - 350 ml - Melância', 'Melância', '350 ml',
'Lata', 4.56);
```

Também temos como inserir mais de um dado em somente um **INSERT**:

```
INSERT INTO PRODUTOS

VALUES ('1040110', 'Light - 350 ml - Jaca', 'Jaca', '350 ml', 'Lata',
6.00),

       ('1040111', 'Light - 350 ml - Manga', 'Manga', '350 ml',
'Lata', 3.50);
```

Podemos também acessar da base Vendas_sucos uma tabela da base Sucos_vendas utilizando o comando:

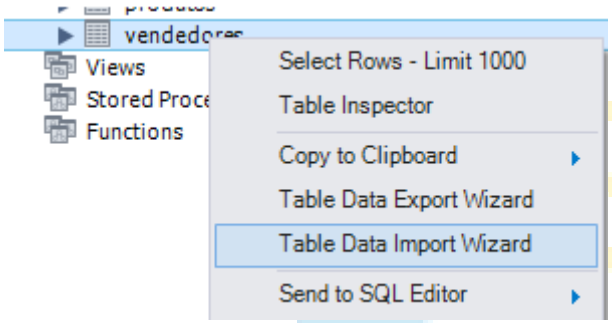
```
USE vendas_sucos;

SELECT * FROM sucos_vendas.tabela_de_produtos;
```

Utilizando essa questão podemos montar um SELECT consultando uma tabela de outro banco ou do mesmo e logo depois adicionar esse SELECT a um comando INSERT, dessa forma inserindo todos os dados do SELECT na tabela definida no INSERT:

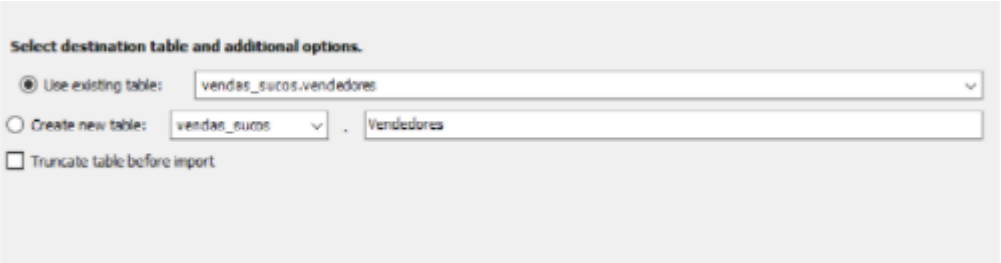
```
INSERT INTO produtos

SELECT CODIGO_DO_PRODUTO AS CODIGO, NOME_DO_PRODUTO AS DESCRITOR,

SABOR, TAMANHO, EMBALAGEM,  PRECO_DE_LISTA AS PRECO_LISTA

FROM sucos_vendas.tabela_de_produtos

WHERE CODIGO_DO_PRODUTO NOT IN (SELECT CODIGO FROM produtos);
```

OBS: Para isso e necessário que os campos tenham o mesmo nome, caso os nomes sejam diferentes basta colocar um AS (alias(apelido)), para que fiquem igual, uma outra coisa e que é necessário que estejam organizados na mesma ordem dos campos da tabela que deseja inserir.

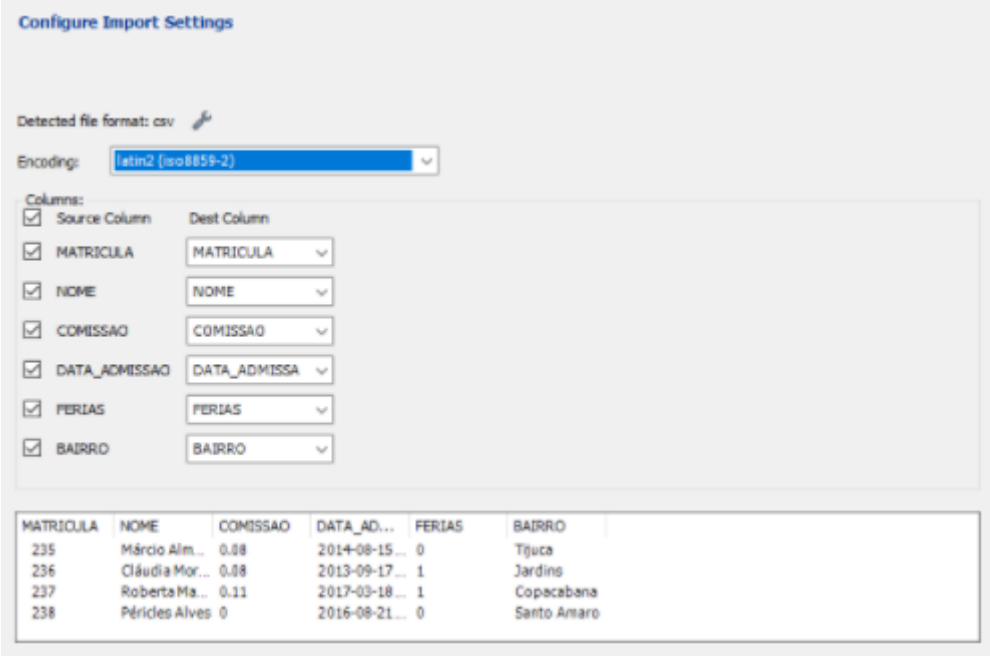
Uma outra forma de inclusão em lote seria incluir dados de um documento externo, para isso basta seguir os passos a baixo:



Vá em cima do nome da tabela de deseja adiciona e clique em “table data import wizard” logo após selecione o documento desejado, CSV, XML, ETC....



Aqui temos algumas opções sendo elas para inserir em uma tabela já existente, criar uma tabela antes da inserção ou apagar todos os registros de uma tabela existente antes de inserir comando truncate.



Nessa parte do lado esquerdo temos os campos de nosso documento e do lado direito os campos da tabela SQL, vincule cada item com seu respectivo campo.

Depois é só finalizar e os dados vão ser inseridos.

4. Alterando e excluindo dados existentes

Alterando dados:

Temos o jeito tradicional e mais simples, que seria:

```
UPDATE produtos SET PRECO_LISTA = 5 WHERE CODIGO = '1000889';
```

Alterando em lote:

```
UPDATE produtos SET EMBALAGEM = 'PET', TAMANHO = '1 Litro', DESCRITOR
=

'Sabor da Montanha - 1 Litro - Uva' WHERE CODIGO = '1000889';
```

Temos que saber nessa funcionalidade que so conseguimos alterar em lote dados da mesma linha ou de linhas deferentes quando possuem o mesmo dado.

Da mesma forma que incluímos os dados na tabela baseado nos dados de uma outra tabela, podemos também alterar dados desta mesma maneira:

```
UPDATE VENDEDORES A INNER JOIN SUCOS_VENDAS.TABELA_DE_VENDEDORES B

ON A.MATRICULA = SUBSTRING(B.MATRICULA,3,3)

SET A.FERIAS = B.DE_FERIAS;
```

Apagando os dados:

Para realizar uma exclusão de dados, utilizamos o comando:

```
DELETE FROM PRODUTOS WHERE CODIGO = '1001000';
```

Ele também podemos colocar filtros, não só podemos como devemos colocar filtros:

```
DELETE FROM PRODUTOS WHERE TAMANHO = '1 Litro' AND
SUBSTRING(DESCRITOR,1,15) = 'Sabor dos Alpes';
```

Outra forma é apagar usando a seleção de dados de outra tabela:

```
DELETE FROM PRODUTOS WHERE CODIGO NOT IN ( SELECT CODIGO_DO_PRODUTO
FROM SUCOS_VENDAS.TABELA_DE_PRODUTOS);
```

5. Utilizando roolback e commit

Esses comandos são utilizados para segurança da informação onde o comando só e salvo após executar um commit ou cancelado após utilizado o roolback.

Para utilizar ele precisa executar o comando:

```
START TRANSACTION;
```

Após esse comando executado podemos executar todas as querys sem ter efetivamente salvo essas alterações.

Para salvar as alterações utilizamos o comando COMMIT:

```
COMMIT;
```

Para cancelar as alterações e voltar ao estado anterior, utilizamos o comando ROOLBACK:

```
ROLLBACK;
```

6. Auto incremento, padrões e triggers

O campo do tipo auto incremento cria uma sequência numérica de números inteiros em um campo. Para definir este campo precisamos configurá-lo na criação da tabela. Logo digite o comando abaixo e execute:

```
CREATE TABLE TAB_IDENTITY (ID INT AUTO_INCREMENT, DESCRITOR
VARCHAR(20), PRIMARY KEY(ID));
```

Para inserir um registro não precisamos nos referenciar ao campo auto incremento no comando INSERT. Digite e execute:

```
INSERT INTO TAB_IDENTITY (DESCRITOR) VALUES ('CLIENTE1');
```

Podemos definir padrões para os campos. Com isto um campo pode ter um valor default caso não seja referenciado no comando INSERT. Digite e execute:

```
CREATE TABLE TAB_PADRAO

(ID INT AUTO_INCREMENT,

DESCRITOR VARCHAR(20),

ENDERECO VARCHAR(100) NULL,

CIDADE VARCHAR(50) DEFAULT 'Rio de Janeiro',

DATA_CRIACAO TIMESTAMP DEFAULT CURRENT_TIMESTAMP(),

PRIMARY KEY(ID));
```

Criando uma trigger:

```
DELIMITER //

CREATE TRIGGER TG_CALCULA_FATURAMENTO_INSERT AFTER INSERT ON
ITENS NOTAS
FOR EACH ROW BEGIN

  DELETE FROM TAB_FATURAMENTO;

  INSERT INTO TAB_FATURAMENTO

  SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VENDA FROM
  NOTAS A INNER JOIN ITENS_NOTAS B
  ON A.NUMERO = B.NUMERO
  GROUP BY A.DATA_VENDA;
END//
```

Triggs são utilizados como gatilhos para quando e necessário executar um comando logo que alguma tabela sofre algum tipo de alteração dentro do banco.

Podemos utilizar um trigger para INSERT, UPDATE e DELETE, dentro desse trigger colocamos os comando para serem executados após a ação dentro da tabela.

```
DELIMITER //

CREATE TRIGGER TG_CALCULA_FATURAMENTO_UPDATE AFTER UPDATE ON
ITENS_NOTAS
FOR EACH ROW BEGIN
  DELETE FROM TAB_FATURAMENTO;

INSERT INTO TAB_FATURAMENTO
  SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VENDA FROM
  NOTAS A INNER JOIN ITENS_NOTAS B
  ON A.NUMERO = B.NUMERO
  GROUP BY A.DATA_VENDA;
END//
DELIMITER //

CREATE TRIGGER TG_CALCULA_FATURAMENTO_DELETE AFTER DELETE ON
ITENS_NOTAS
FOR EACH ROW BEGIN
  DELETE FROM TAB_FATURAMENTO;

INSERT INTO TAB_FATURAMENTO
  SELECT A.DATA_VENDA, SUM(B.QUANTIDADE * B.PRECO) AS TOTAL_VENDA FROM
  NOTAS A INNER JOIN ITENS_NOTAS B
  ON A.NUMERO = B.NUMERO
  GROUP BY A.DATA_VENDA;
END//
```