TAREFA 22 - BANCO DE DADOS - Solução
Stored Procedures – MySQL
Prof. Dr. Aparecido Freitas

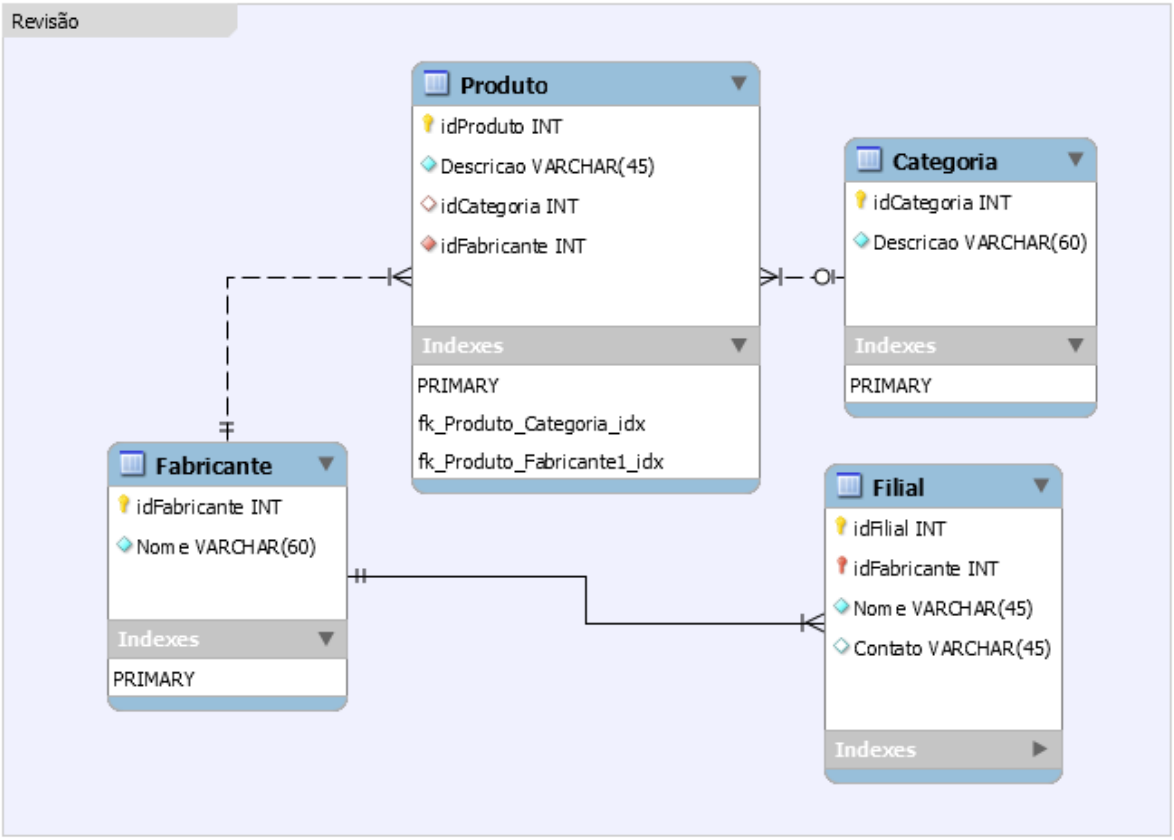
1. Introdução

Para a execução desta atividade iremos considerar um **Banco de Dados** chamado **produtodb**, com as seguintes definições de tabelas:

```
CREATE TABLE Produto (
  idProduto INT(11) NOT NULL AUTO_INCREMENT,
  Descricao VARCHAR(45) NOT NULL,
  idCategoria INT(11) NULL DEFAULT NULL,
  idFabricante INT(11) NOT NULL,
  PRIMARY KEY (idProduto),
  INDEX fk_Produto_Categoria_idx (idCategoria ASC),
  INDEX fk_Produto_Fabricante1_idx (idFabricante ASC),
  CONSTRAINT fk_Produto_Categoria FOREIGN KEY (idCategoria) REFERENCES Categoria (idCategoria),
  CONSTRAINT fk_Produto_Fabricante1 FOREIGN KEY (idFabricante) REFERENCES Fabricante (idFabricante)
);
CREATE TABLE Fabricante (
  idFabricante INT(11) NOT NULL,
  Nome VARCHAR(60) NOT NULL,
  PRIMARY KEY (idFabricante)
);
CREATE TABLE Categoria (
  idCategoria INT(11) NOT NULL,
  Descricao VARCHAR(60) NOT NULL,
  PRIMARY KEY (idCategoria)
);
CREATE TABLE Filial (
  idFilial INT(11) NOT NULL,
  idFabricante INT(11) NOT NULL,
  Nome VARCHAR(45) NOT NULL,
  Contato VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (idFilial, idFabricante),
  INDEX fk_Filial_Fabricante1_idx (idFabricante ASC),
  CONSTRAINT fk_Filial_Fabricante1 FOREIGN KEY (idFabricante) REFERENCES Fabricante (idFabricante)
);
```

```
INSERT INTO fabricante (idFabricante, Nome) VALUES ('1', 'Nestlé');
INSERT INTO fabricante (idFabricante, Nome) VALUES ('2', 'Parmalat');
INSERT INTO fabricante (idFabricante, Nome) VALUES ('3', 'Kelloggs');
INSERT INTO categoria (idCategoria, Descricao) VALUES ('1', 'Leite');
INSERT INTO categoria (idCategoria, Descricao) VALUES ('2', 'Cereais Matinais');
INSERT INTO categoria (idCategoria, Descricao) VALUES ('3', 'Achocolatado');
INSERT INTO produto (Descricao, idCategoria, idFabricante) VALUES ('Leite Integral', '1', '1');
INSERT INTO produto (Descricao, idCategoria, idFabricante) VALUES ('Nescau', '3', '1');
INSERT INTO produto (Descricao, idCategoria, idFabricante) VALUES ('Sucrilhos', '2', '3');
```

Empregaremos o seguinte modelo de dados:



Uma linguagem do tipo **Stored Procedure** corresponde a códigos de programação executados internamente ao Sistema Gerenciamento de Banco de Dados baseados numa linguagem estruturada de script.

Assim, existe uma só linguagem SQL e uma série de PL's incompatíveis entre si que são usados no mercado. A linguagem PL/SQL do MySQL possui algumas características particulares, diferenciando-a dos outros fornecedores.

Para o desenvolvimento de procedimentos em MySQL é necessário o domínio e conhecimento da estrutura da linguagem de script usada pelo MySQL. Stored procedures são rotinas definidas no banco de dados, identificadas por um nome pelo qual podem ser invocadas. Um procedimento desses pode executar uma série de instruções, receber parâmetros e retornar valores.

Desde a versão 5.0 o MySQL suporta a criação e execução de Stored Procedures. As Stored Procedures podem ser usadas para validação de dados, controle de acesso, execução de declarações SQL complexas e muitas outras situações.

Um procedimento armazenado (STORED PROCEDURES) é uma sequência de instruções SQL declarativas armazenada dentro do catálogo de banco de dados. A instrução para criação de uma Stored Procedure é:

```
CREATE PROCEDURE sp_MinhaPrimeira() SELECT 'Olá Mundo!!!';
```

```
mysql> CREATE PROCEDURE sp_MinhaPrimeira() SELECT 'OLá Mundo!!!';
Query OK, 0 rows affected (0.06 sec)

mysql>
```

Um procedimento armazenado pode ser invocado por gatilhos (triggers), outros procedimentos armazenados ou outras aplicações, tais como: **Java, C #, PHP,** etc.;

Uma vez criados, os procedimentos armazenados são compilados e armazenados no banco de dados.

Para invocar uma **Stored Procedure** é necessário utilizar a instrução **CALL**.

```
CALL sp_MinhaPrimeira();
```

```
mysql> CALL sp_MinhaPrimeira();
+--------------+
| OLá Mundo!!! |
+--------------+
| OLá Mundo!!! |
+--------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.05 sec)

mysql>
```

No MySQL procedimentos armazenados são compilados na demanda. Depois de compilar um procedimento armazenado, O MySQL coloca-o em cache. Se um aplicativo usar um procedimento armazenado várias vezes em uma única conexão, a versão compilada é usada.

Pode-se visualizar Stored Procedures por meio do comando:

```
SHOW PROCEDURE STATUS LIKE 'sp_MinhaPrimeira';
```

```
mysql> SHOW PROCEDURE STATUS LIKE 'SP_MINHAPRIMEIRA';
+-----------+-----------------+-----------+----------------+---------------------
--+---------------------+---------------+---------+----------------------+------
---------------+--------------------+
| Db        | Name            | Type      | Definer        | Modified
  | Created             | Security_type | Comment | character_set_client | colla
tion_connection | Database Collation |
+-----------+-----------------+-----------+----------------+---------------------
--+---------------------+---------------+---------+----------------------+------
---------------+--------------------+
| produtodb | sp_MinhaPrimeira | PROCEDURE | root@localhost | 2018-10-03 07:56:2
3 | 2018-10-03 07:56:23 | DEFINER       |         | cp850                | cp850
_general_ci     | utf8_general_ci    |
+-----------+-----------------+-----------+----------------+---------------------
--+---------------------+---------------+---------+----------------------+------
---------------+--------------------+
1 row in set (0.00 sec)

mysql>
```

Vamos agora criar uma **procedure** chamada **SP_ConsultaMaiorFabricante**, que ao ser invocada, deve exibir o fabricante com o **maior número de produtos vinculados**. A procedure deve exibir:

- ✓ **Código do Produto;**
- ✓ **Descrição do Produto;**
- ✓ **Nome do Fabricante;**

As estruturas das tabelas envolvidas devem ser:

```
mysql> describe produto;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| idProduto    | int(11)     | NO   | PRI | NULL    | auto_increment |
| Descricao    | varchar(45) | NO   |     | NULL    |                |
| idCategoria  | int(11)     | YES  | MUL | NULL    |                |
| idFabricante | int(11)     | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.02 sec)

mysql>
```

```
mysql> describe fabricante;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| idFabricante | int(11)     | NO   | PRI | NULL    |       |
| Nome         | varchar(60) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

Os dados atuais das tabelas são:

```
mysql> select * from produto;
+-----------+----------------+-------------+--------------+
| idProduto | Descricao      | idCategoria | idFabricante |
+-----------+----------------+-------------+--------------+
|         9 | Leite Integral |           1 |            1 |
|        10 | Nescau         |           3 |            1 |
|        11 | Sucrilhos      |           2 |            3 |
+-----------+----------------+-------------+--------------+
3 rows in set (0.02 sec)

mysql>
```

```
mysql> select * from fabricante;
+--------------+----------+
| idFabricante | Nome     |
+--------------+----------+
|            1 | Nestlé   |
|            2 | Parmalat |
|            3 | Kelloggs |
+--------------+----------+
3 rows in set (0.00 sec)

mysql>
```

A query deverá gerar as seguintes tuplas:

Result Grid | Filter Rows: | Export:

	IDFABRICANTE	NOME	TOTALPRODUTOS
▶	1	Nestlé	2

Defina a PROCEDURE SP_CONSULTAMAIORFABRICANTE(), conforme solicitado:

```
CREATE PROCEDURE SP_CONSULTAMAIORFABRICANTE()
  SELECT P.IDFABRICANTE , F.NOME , COUNT(*) 'TOTALPRODUTOS'
  FROM PRODUTO P
  INNER JOIN FABRICANTE F USING (IDFABRICANTE)
  GROUP BY P.IDFABRICANTE
  ORDER BY TOTALPRODUTOS DESC
  LIMIT 1;
```

Em seguida, execute o comando para a chamada da procedure:

```
CALL SP_CONSULTAMAIORFABRICANTE();
```

Execute o comando para visualizar a Store Procedure criada:

```
SHOW PROCEDURE STATUS LIKE 'SP_CONSULTAMAIORFABRICANTE';
```

Execute o comando que retorna a String exata que pode ser usada para se recriar a rotina:

```
SHOW CREATE PROCEDURE SP_CONSULTAMAIORFABRICANTE;
```

Execute o comando para excluir a procedure SP_CONSULTAMAIORFABRICANTE():

```
DROP PROCEDURE SP_CONSULTAMAIORFABRICANTE;
```