LAB_04

Participantes: Adriano Vale, Ary Farah, Caroline Assis

4.1 → INSERT: DEFAULT e NULL

```
CREATE DATABASE lab_04;
USE lab_04;

CREATE TABLE Tab_Depto (
ID          INT         AUTO_INCREMENT PRIMARY KEY,
Nome        VARCHAR(60) NOT NULL DEFAULT ('Vendas'),
Localizacao VARCHAR(60) DEFAULT ('Bloco A'),
Sala        CHAR(3)     NOT NULL,
Fone        VARCHAR(20)
);

INSERT Tab_Depto (Sala, Fone) VALUES ('80', '(41)3021-4040');  -- b.1) tratamento de campos omitidos
INSERT Tab_Depto (Sala) VALUES ('100');                        -- b.2) tratamento de campos omitidos
INSERT Tab_Depto (Localizacao, Sala) VALUES (NULL,'200');      -- b.3) tratamento de NULL
INSERT Tab_Depto (Sala) VALUES (NULL);                         -- b.4) tratamento de NULL
INSERT Tab_Depto (Localizacao, Sala) VALUES (NULL, '300');     -- b.5) tratamento de NULL

SELECT * FROM Tab_Depto;
```

O tratamento de campos omitidos significa que se não for especificado nenhum valor no INSERT das colunas, o BD utilizará os valores padrão definidos na estrutura da tabela.

1. **Em b.1), como é feito o tratamento de campos omitidos?**

   O campo ‘Nome’ terá atribuído o valor ‘Vendas’, já o campo ‘Localização’ terá atribuído o valor ‘Bloco A’.

2. **Em b.2), como é feito o tratamento de campos omitidos?**

   Como em b.1), o campo ‘Nome’ terá atribuído o valor ‘Vendas’ e o campo ‘Localização’ terá atribuído o valor ‘Bloco A’. Para o campo ‘Fone‘ foi atribuído um valor NULL, já que não foi especificado no INSERT.

3. **Em b.3), como é feito o tratamento do NULL?**

   O tratamento do NULL é feito colocando o valor DEFAULT no campo ‘Localizacao’.

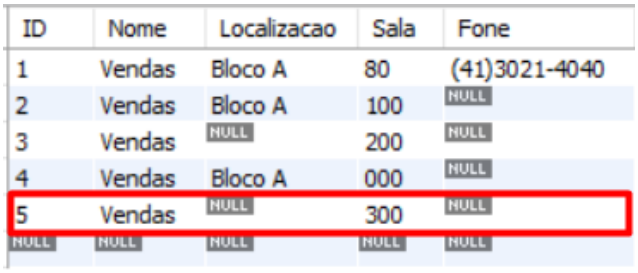
4. **Em b.4), o que acontece neste INSERT do NULL?**

   O INSERT do NULL não funciona, já que o campo ‘Sala’ não pode ter valor vazio.

5. **Em b.5), o que acontece neste INSERT do NULL?**

   O INSERT do NULL atribui o valor DEFAULT no campo ‘Localizacao’.

6. **Em b.5), como ficou povoada a Tab_Depto?**



ID	Nome	Localizacao	Sala	Fone
1	Vendas	Bloco A	80	(41)3021-4040
2	Vendas	Bloco A	100	NULL
3	Vendas	NULL	200	NULL
4	Vendas	Bloco A	000	NULL
5	Vendas	NULL	300	NULL
NULL	NULL	NULL	NULL	NULL

   *Substituí o valor do b.4 por 000 para poder dar continuidade no código, já que esse campo não poderia ser atribuído o valor NULL

4.2 → Ações para manter a IR

```
-- a)
CREATE TABLE Editora (
ID_edit INT AUTO_INCREMENT PRIMARY KEY, -- Tabela PAI
Nome_Edit VARCHAR(60) NOT NULL,
Cidade VARCHAR(60)    NOT NULL,
Estado CHAR(2)        NOT NULL,
Pais VARCHAR(50)      NOT NULL
);

INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Editora AAA', 'São Paulo', 'SP', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Editora Sul', 'Porto Alegre', 'RS', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('LTC', 'São Paulo', 'SP', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('CENGAGE', 'Rio de Janeiro', 'RJ', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Três Estrelas', 'Alagoas', 'CE', 'Brasil');

-------------------------------------------------------------------------------

-- b)
CREATE TABLE Autor (
ID_Autor   INT         AUTO_INCREMENT PRIMARY KEY, -- Tabela FILHO
Nome_Autor VARCHAR(60) NOT NULL,
Dt_Nasc    DATE        NOT NULL,
fk_ID_Edit INT         NULL
);

ALTER TABLE Autor ADD CONSTRAINT FK_Autor_Editora FOREIGN KEY(fk_ID_edit)
REFERENCES Editora (ID_edit)
ON UPDATE CASCADE
ON DELETE CASCADE ;
```

```
ALTER TABLE Autor AUTO_INCREMENT = 100; -- Seed = 100 (início do AUTO_INCREMENT)
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('José', '1956-09-08', 1);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Maria', '1975-04-18', 2);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Antônia', '1954-12-10', 3);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Armínio', '1976-07-28', 5);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Luiza', '1945-11-09', 5);
```

```
-- c)
SELECT * FROM Editora;
SELECT * FROM Autor;

UPDATE Editora
SET ID_edit = 50
WHERE ID_edit = 5;

SELECT * FROM Editora
SELECT * FROM Autor

--------------------------------------------------------------------------------

-- d)
SELECT * FROM Editora;
SELECT * FROM Autor;

DELETE FROM Editora
WHERE ID_edit = 1;

SELECT * FROM Editora
SELECT * FROM Autor
```

1. **Em c), qual foi o resultado obtido nas tabelas Editora e Autor, após o UPDATE executado? Por que isso ocorreu?**

ID_edit	Nome_Edit	Cidade	Estado	Pais
1	Editora AAA	São Paulo	SP	Brasil
2	Editora Sul	Porto Alegre	RS	Brasil
3	LTC	São Paulo	SP	Brasil
4	CENGAGE	Rio de Janeiro	RJ	Brasil
50	Três Estrelas	Alagoas	CE	Brasil
NULL	NULL	NULL	NULL	NULL

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
100	José	1956-09-08	1
101	Maria	1975-04-18	2
102	Antônia	1954-12-10	3
103	Armínio	1976-07-28	50
104	Luiza	1945-11-09	50
NULL	NULL	NULL	NULL

O ID_edit de ‘Três Estrelas’ foi substituído de 5 para 50, alterando também em fk_ID_Edit onde os valores eram 5 para 50 devido à ação CASCADE definida na chave estrangeira.

2. **Em d), qual foi o resultado obtido nas tabelas Editora e Autor, após o DELETE executado? Por que isso ocorreu?**

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
101	Maria	1975-04-18	2
102	Antônia	1954-12-10	3
103	Armínio	1976-07-28	50
104	Luiza	1945-11-09	50
NULL	NULL	NULL	NULL

ID_edit	Nome_Edit	Cidade	Estado	Pais
2	Editora Sul	Porto Alegre	RS	Brasil
3	LTC	São Paulo	SP	Brasil
4	CENGAGE	Rio de Janeiro	RJ	Brasil
50	Três Estrelas	Alagoas	CE	Brasil
NULL	NULL	NULL	NULL	NULL

O ID_edit de com valor ‘1’ foi deletado, deletando também os autores associados ao fk_ID_Edit com valor ‘1’ devido à ação CASCADE definida na chave estrangeira.

```
-- e)
UPDATE Editora
SET ID_edit = 5
WHERE ID_edit = 50;

DROP TABLE Autor;
CREATE TABLE Autor -- Alterando Tabela Autor
(
ID_Autor   INT         AUTO_INCREMENT PRIMARY KEY, -- Tabela FILHO
Nome_Autor VARCHAR(60) NOT NULL,
Dt_Nasc    DATE        NOT NULL,
fk_ID_Edit INT         NULL
);

ALTER TABLE Autor ADD CONSTRAINT FK_Autor_Editora FOREIGN KEY(fk_ID_edit)
REFERENCES Editora (ID_edit)
ON UPDATE RESTRICT
ON DELETE RESTRICT ;

ALTER TABLE Autor AUTO_INCREMENT = 100; -- Seed = 100 (início do AUTO_INCREMENT)
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('José', '1956-09-08', 1);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Maria', '1975-04-18', 2);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Antônia', '1954-12-10', 3);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Armínio', '1976-07-28', 5);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Luiza', '1945-11-09', 5);
```

```
-- f)
SELECT * FROM Editora;
SELECT * FROM Autor;

UPDATE Editora
SET ID_edit = 50
WHERE ID_edit = 5;

SELECT * FROM Editora;
SELECT * FROM Autor;

-------------------------------------------------------------------------------
```

```
-- g)
SELECT * FROM Editora;
SELECT * FROM Autor;

DELETE FROM Editora
WHERE ID_edit = 1;

SELECT * FROM Editora;
SELECT * FROM Autor;
```


Ao tentar rodar o código, um erro é exibido na tela, pois devido ao comando RESTRICT, não é possível alterar os valores de uma tabela PAI através de uma chave estrangeira da tabela FILHO.

1. **Em f), qual foi o resultado obtido nas tabelas Editora e Autor, após o UPDATE executado? Por que isso ocorreu?**

   Devido ao comando RESTRICT na FK, não é possivel realizar um UPDATE através da tabela filho.

2. **Em g), qual foi o resultado obtido nas tabelas Editora e Autor, após o DELETE executado? Por que isso ocorreu?**

   Devido ao comando RESTRICT na FK, não é possivel realizar um DELETE através da tabela filho.

```
-- h)
UPDATE Editora
SET ID_edit = 5
WHERE ID_edit = 50;

DROP TABLE Autor;
CREATE TABLE Autor (
ID_Autor INT AUTO_INCREMENT PRIMARY KEY, -- Tabela FILHO
Nome_Autor VARCHAR(60) NOT NULL,
Dt_Nasc DATE NOT NULL,
fk_ID_Edit INT NULL
);

ALTER TABLE Autor ADD CONSTRAINT FK_Autor_Editora FOREIGN KEY(fk_ID_edit)
REFERENCES Editora (ID_edit)
ON UPDATE SET NULL
ON DELETE SET NULL;

ALTER TABLE Autor AUTO_INCREMENT = 100; -- Seed = 100 (início do AUTO_INCREMENT)
```

```
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('José', '1956-09-08', 1);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Maria', '1975-04-18', 2);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Antônia', '1954-12-10', 3);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Armínio', '1976-07-28', 5);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Luiza', '1945-11-09', 5);
```

```
-- i)
SELECT * FROM Editora;
SELECT * FROM Autor;

UPDATE Editora
SET ID_edit = 50
WHERE ID_edit = 5;

SELECT * FROM Editora
SELECT * FROM Autor

-------------------------------------------------------------------------------

-- j)
SELECT * FROM Editora;
SELECT * FROM Autor;

DELETE FROM Editora
WHERE ID_edit = 1;

SELECT * FROM Editora
SELECT * FROM Autor
```

1. **Em i), qual foi o resultado obtido nas tabelas Editora e Autor, após o UPDATE executado? Por que isso ocorreu?**

ID_edit	Nome_Edit	Cidade	Estado	Pais
1	Editora AAA	São Paulo	SP	Brasil
2	Editora Sul	Porto Alegre	RS	Brasil
3	LTC	São Paulo	SP	Brasil
4	CENGAGE	Rio de Janeiro	RJ	Brasil
50	Três Estrelas	Alagoas	CE	Brasil

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
100	José	1956-09-08	1
101	Maria	1975-04-18	2
102	Antônia	1954-12-10	3
103	Armínio	1976-07-28	NULL
104	Luiza	1945-11-09	NULL

Na tabela ‘Editora’, o ID de número 5 foi substituído por 50, já na tabela ‘Autor’, os valores das FK com valor igual a 5 foram substituídas por NULL, por causa da constraint SET NULL definida para a chave estrangeira.

2. **Em j), qual foi o resultado obtido nas tabelas Editora e Autor, após o DELETE executado? Por que isso ocorreu?**

ID_edit	Nome_Edit	Cidade	Estado	Pais
2	Editora Sul	Porto Alegre	RS	Brasil
3	LTC	São Paulo	SP	Brasil
4	CENGAGE	Rio de Janeiro	RJ	Brasil
50	Três Estrelas	Alagoas	CE	Brasil

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
100	José	1956-09-08	NULL
101	Maria	1975-04-18	2
102	Antônia	1954-12-10	3
103	Armínio	1976-07-28	NULL
104	Luiza	1945-11-09	NULL

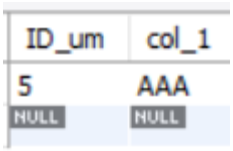
Na tabela ‘Editora’, a coluna com ID de número 1 foi deletada, já na tabela ‘Autor’, o fk_ID_Edit com valor igual a 1 foi substituída por NULL, por causa da constraint SET NULL definida para a chave estrangeira.

4.3 → INSERT em VIEWS

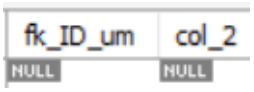
```
-- a)

CREATE TABLE Tab_Um (
ID_um INT PRIMARY KEY NOT NULL,
col_1 CHAR(3) NOT NULL
);

CREATE TABLE Tab_Dois (
fk_ID_um INT PRIMARY KEY NOT NULL,
col_2 CHAR(3) NOT NULL,
FOREIGN KEY (FK_ID_um)
REFERENCES Tab_Um (ID_um)
);

CREATE VIEW JuntaUmDois AS (
SELECT ID_um, col_1, fk_ID_um, col_2
FROM Tab_Um JOIN Tab_Dois
ON (Tab_Um.ID_um = Tab_Dois.fk_ID_um)
);
```

```
-- b)

-- 1º. INSERT
INSERT Tab_Um (ID_um, col_1) VALUES (5, 'AAA');
SELECT * FROM JuntaUmDois;
SELECT * FROM Tab_Um;
SELECT * FROM Tab_Dois;

-- 2º. INSERT
INSERT Tab_Dois(fk_ID_um, col_2) VALUES (5, 'XXX');
SELECT * FROM JuntaUmDois;
SELECT * FROM Tab_Um;
SELECT * FROM Tab_Dois;
```

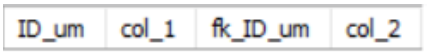
1. **Em b), após o 1º. INSERT, o que foi exibido na VIEW? Por que esse resultado foi apresentado?**



Tab_Um



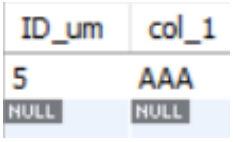
Tab_Dois



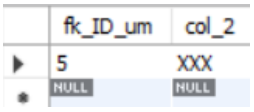
JuntaUmDois

Esse resultado foi apresentado pois foi feito somente um INSERT na Tab_Um, e a VIEW JuntaUmDois usa um JOIN que registra os ID quando forem iguais.

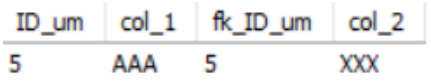
2. **Em b), após o 2º. INSERT, o que foi exibido na VIEW? Por que esse resultado foi apresentado?**



Tab_Um



Tab_Dois

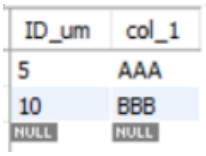


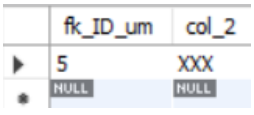
JuntaUmDois

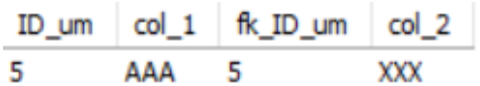
Agora, como foi feito um INSERT na Tab_Dois com o mesmo ID da Tab_Um, a VIEW JuntaUmDois juntou as colunas em uma única tabela.

```
-- c)

-- 1º. INSERT
INSERT JuntaUmDois (ID_Um, col_1) VALUES (10, 'BBB');
SELECT * FROM JuntaUmDois;
SELECT * FROM Tab_Um;
SELECT * FROM Tab_Dois;

-- 2º. INSERT
INSERT JuntaUmDois(fk_ID_um, col_2) VALUES (20, 'YYY');
SELECT * FROM JuntaUmDois;
SELECT * FROM Tab_Um;
SELECT * FROM Tab_Dois;
```

1. **Em c), após o 1º. INSERT, o que foi exibido na VIEW? Por que esse resultado foi apresentado?**







A VIEW foi exibida desse jeito porque ela mostra somente as linhas que tiverem o mesmo ID.

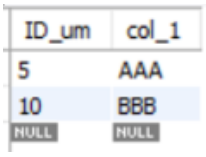
2. **Em c), o 2º. INSERT na VIEW funcionou na Tab_Dois? Por que?**

   Não, pois não é possível alterar uma Child Row, nesse caso, a fk_ID_um.

3. **Em c), como devemos alterar o 2º. INSERT, para que ele funcione na Tab_Dois e também seja exibido na VIEW?**

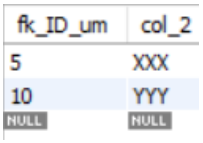
   Para funcionar, o ID a ser adicionado deve ser o mesmo da Tab_Um (10).

4. **Após corrigir o 2º. INSERT, mostre como ficam preenchidas as Tab_Um, Tab_Dois e VIEW JuntaUmDois.**



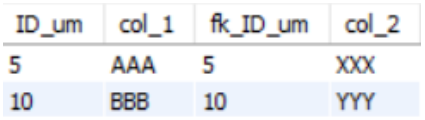
ID_um	col_1
5	AAA
10	BBB
NULL	NULL

Tab_Um



fk_ID_um	col_2
5	XXX
10	YYY
NULL	NULL

Tab_Dois



ID_um	col_1	fk_ID_um	col_2
5	AAA	5	XXX
10	BBB	10	YYY

JuntaUmDois

4.4 → Variáveis

```
-- a)
CREATE TABLE Empresa (
ID INT PRIMARY KEY AUTO_INCREMENT,
Nome VARCHAR(20),
Atuacao VARCHAR(50),
Cidade VARCHAR(20),
Estado VARCHAR(2)
);

INSERT Empresa (Nome, Atuacao, Cidade, Estado) VALUES
('ACME Corp.','Cartoons','São Paulo','SP'),
('Estrela Ltda.', 'Transporte passageiros','Campinas','SP'),
('Aurora','Panificadora','Belo Horizonte','MG'),
('Azul','Aviação','São Paulo','SP'),
('Leão Ltda.','Bebidas','Curitiba','PR'),
('Petit S.A.','Queijos e frios', 'Uberlândia','MG'),
('Barreados Corp.','Alimentos congelados','Morretes','PR');

SELECT * FROM Empresa;

-----------------------------------------------------------------

-- b)
```

```
CREATE TABLE Estoque (
ID INT PRIMARY KEY AUTO_INCREMENT,
Nome VARCHAR(20),
Qtde INT DEFAULT 10,
ValUnit DECIMAL(10,2)
);

INSERT Estoque (Nome, Qtde, ValUnit) VALUES
('caderno',200,15.00),
('borracha',50,6.50),
('caneta',300, 5.50),
('régua 30cm',80, 10.00),
('lápis',500, 4.00),
('bloco A4',35, 18.45);

SELECT * FROM Estoque;
```

ID	Nome	Atuacao	Cidade	Estado
1	ACME Corp.	Cartoons	São Paulo	SP
2	Estrela Ltda.	Transporte passageiros	Campinas	SP
3	Aurora	Panificadora	Belo Horizonte	MG
4	Azul	Aviação	São Paulo	SP
5	Leão Ltda.	Bebidas	Curitiba	PR
6	Petit S.A.	Queijos e frios	Uberlândia	MG
7	Barreados Corp.	Alimentos congelados	Morretes	PR
NULL	NULL	NULL	NULL	NULL

ID	Nome	Qtde	ValUnit
1	caderno	200	15.00
2	borracha	50	6.50
3	caneta	300	5.50
4	régua 30cm	80	10.00
5	lápis	500	4.00
6	bloco A4	35	18.45
NULL	NULL	NULL	NULL

```
-- c)\
SET @nome_produto = 'none'; -- Declara e inicializa variáveis de sessão
SET @total_produtos = -1;

SELECT nome INTO @nome_produto FROM Estoque
WHERE ID = 3;

SELECT COUNT(*) INTO @total_produtos FROM Estoque;

SELECT @nome_produto AS 'Produto com ID = 3';
SELECT @total_produtos AS 'Total de Produtos Cadastrados';
```

Total de Produtos Cadastrados
6

Produto com ID = 3
caneta

1. **Nas linhas 1. e 2., o que acontece se não inicializarmos as variáveis?**

   Elas não terão valores iniciais definidos e serão consideradas nulas até que atribuam valores a elas em algum ponto posterior no código. Isso pode fazer com que resultados indesejados sejam obtidos, como a quantidade errada de produtos cadastrados.

2. **É preciso executar de uma única vez todos os comandos do script apresentado, para exibir o conteúdo final das variáveis de sessão do exemplo? Por que?**

   Não, pode executar cada comando separadamente, desde que o faça na ordem correta. As variáveis de sessão são mantidas na sessão de execução do SQL Server e podem ser acessadas em qualquer momento após a declaração, desde que a sessão esteja ativa.

```
-- d)
DELIMITER $$
CREATE PROCEDURE proc_demo1() -- Cria procedure proc_demo1(), sem parâmetros
BEGIN
  DECLARE i INT DEFAULT 0; -- Declara e inicializa variáveis locais
  DECLARE output VARCHAR(100) DEFAULT 'Saída = ';
  WHILE i < 10 DO
    SET output = CONCAT(output, i , ', ');
    SET i = i + 1;
  END WHILE;
  SELECT output;
END $$
DELIMITER ; -- Retorna ao delimitador padrão da linguagem
CALL proc_demo1();
```

1. **Em que linhas do código estão os delimitadores do bloco de comandos da procedure?**

   Os delimitadores estão nas linhas 2 e 12, com os símbolos '$$'.

2. **O que é feito nas linhas 5. e 6.?**

   São declaradas as variáveis, com seu tipo e o valor DEFAULT.

3. **Quais os delimitadores do laço de repetição WHILE?**

   Os delimitadores do laço WHILE são 'WHILE … DO' e 'END WHILE'

4. **O que é feito nas linhas 8., .9 e .15?**

   8/9 → é definido o que deve acontecer enquanto a condição do WHILE for verdadeira

   15 → é chamada a PROCEDURE criada

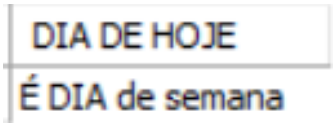
4.5 → IF / CASE

```
-- a)
```

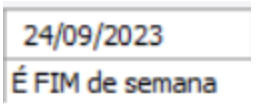
```
-- IF: testa de uma CONDIÇÃO é TRUE ou FALSE, apenas
-- SINTAXE: IF(condition, value_if_true, value_if_false)

SELECT IF (WEEKDAY(NOW()) IN (5, 6), 'É FIM de semana', 'É DIA de semana') AS 'DIA DE HOJE';
SELECT IF (WEEKDAY('2023-09-24') IN (5, 6), 'É FIM de semana', 'É DIA de semana') AS '24/09/2023';

-----------------------------------------------------------------------------------------

-- b)
SELECT ID AS 'Código', Nome, Qtde AS 'Quantidade',
IF (Qtde < 100, 'Baixo (menor que 100)', 'Em boa quantidade') AS 'Nível Estoque'
FROM Estoque;
```

O comando WEEKDAY retorna o dia da semana em número: segunda = 0, terça = 1, quarta = 2

1. **Em a), qual a diferença entre os comandos SELECT? Qual o resultado?**



DIA DE HOJE
É DIA de semana

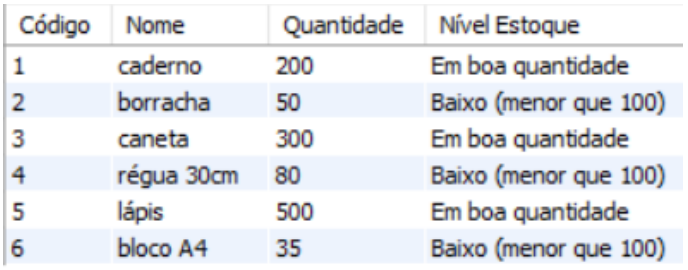


24/09/2023
É FIM de semana

O primeiro SELECT seleciona o dia de hoje, e se o resultado for igual a 5 ou 6, retorna o resultado como ‘É FIM de semana’, caso contrário retorna como ‘É DIA de semana’. Nesse caso, como estou fazendo em uma quinta-feira (3), o resultado obtido foi o segundo.

O segundo SELECT seleciona a data 24/09/2023 e utiliza as mesmas condições do primeiro. Nesse caso, o dia 24/09/2023 caiu em um domingo (6), retornarndo ‘É FIM de semana’.

2. **Em b), apresente e explique o que aparece na coluna ‘Nível Estoque’ do SELECT?**



Código	Nome	Quantidade	Nível Estoque
1	caderno	200	Em boa quantidade
2	borracha	50	Baixo (menor que 100)
3	caneta	300	Em boa quantidade
4	régua 30cm	80	Baixo (menor que 100)
5	lápis	500	Em boa quantidade
6	bloco A4	35	Baixo (menor que 100)

Esse comando seleciona a tabela ‘Estoque’ já criada, e em ‘Nível Estoque’ é utilizado uma condição que analisa a quantidade de cada item e retorna ‘Em boa quantidade’ caso seja maior ou igual a 100 e ‘Baixo (menor que 100)’ caso seja menor que 100.

```
-- c)

-- CASE: retorna valor que pode ser atribuído
-- SINTAXE:
-- CASE
```

```
-- WHEN condition1 THEN result1
-- WHEN condition2 THEN result2
-- WHEN conditionN THEN resultN
-- ELSE result
-- END;

INSERT INTO Estoque (Nome, ValUnit) VALUES ('cola bastão', 15.00); -- 1º. INSERT
INSERT INTO Estoque (Nome, Qtde, ValUnit) VALUES ('tesoura', NULL, 15.00); -- 2º. INSERT

SELECT ID AS 'Código', Nome, Qtde AS 'Quantidade',
CASE
WHEN Qtde < 100 THEN 'BAIXO'
WHEN Qtde BETWEEN 100 AND 300 THEN 'OK'
WHEN Qtde > 300 THEN 'ALTO'

ELSE 'DESCONHECIDO'
END AS 'Nível Estoque'
FROM Estoque
ORDER BY Nome;
```

1. **No 1º. INSERT, não foi especificado um valor de Qtde. Qual o Nível de Estoque apresentado no SELECT? Por que esse valor foi exibido?**

   O valor exibido foi 10. Isso aconteceu porque na criação da tabela foi definido como DEFAULT de Qtde o valor 10.

2. **No 2º. INSERT, foi especificado que de Qtde = NULL. Qual o Nível de Estoque apresentado no SELECT? Por que esse valor foi exibido?**

   O nível de estoque apresentado foi ‘DESCONHECIDO’. Isso aconteceu por causa do ELSE, que agiria caso o valor fosse nulo (não se encaixasse no CASE (<100, 100< Qtde < 300, > 300)).

3. **Quando a opção ELSE do CASE é a executada?**

   A opção ELSE é executada quando ‘Qntd’ não se encaixa em nenhuma das opções do CASE (Qtde < 100, 100 < Qtde < 300, Qtde > 300), ou seja, fosse nulo, retornando o valor ‘DESCONHECIDO’

4. **Apresente o resultado do SELECT.**

Código	Nome	Quantidade	Nível Estoque
6	bloco A4	35	BAIXO
2	borracha	50	BAIXO
1	caderno	200	OK
3	caneta	300	OK
7	cola bastão	10	BAIXO
5	lápis	500	ALTO
4	régua 30cm	80	BAIXO
8	tesoura	NULL	DESCONHECIDO

4.6 → Comparação

```
-- a)
SELECT nome, Atuacao, Cidade, Estado
FROM Empresa
WHERE
((Cidade <> 'São Paulo' AND Estado <> 'SP')
OR
(Cidade <> 'Morretes' AND Estado <> 'PR')
);

--------------------------------------------------------------------------------

-- b)
SELECT nome, Atuacao, Cidade, Estado
FROM Empresa
WHERE NOT (
((Cidade = 'São Paulo' AND Estado = 'SP')
OR
(Cidade = 'Morretes' AND Estado = 'PR'))
);
```

PROBLEMA: Encontrar as empresas que não estão nem em São Paulo, SP nem em Morretes, PR.

1. **Apresente o resultado de cada SELECT.**

nome	Atuacao	Cidade	Estado
ACME Corp.	Cartoons	São Paulo	SP
Estrela Ltda.	Transporte passageiros	Campinas	SP
Aurora	Panificadora	Belo Horizonte	MG
Azul	Aviação	São Paulo	SP
Leão Ltda.	Bebidas	Curitiba	PR
Petit S.A.	Queijos e frios	Uberlândia	MG
Barreados Corp.	Alimentos congelados	Morretes	PR

a)

nome	Atuacao	Cidade	Estado
Estrela Ltda.	Transporte passageiros	Campinas	SP
Aurora	Panificadora	Belo Horizonte	MG
Leão Ltda.	Bebidas	Curitiba	PR
Petit S.A.	Queijos e frios	Uberlândia	MG

b)

2. **Qual consulta de SELECT satisfaz o enunciado? Por que?**

   O segundo SELECT satisfaz o enunciado, pois ele não retorna as empresas presentes em São Paulo (SP) ou Morretes (PR).

4.7 → Funções Matemáticas

```
-- a)
SET @angle = PI()/4; -- 45º em rad
SELECT CONCAT( 'O SENO do ângulo: ' ,
 CONVERT(ROUND(@angle,3), CHAR) ,
 ' rad = ' ,
CONVERT(ROUND(SIN(@angle),3), CHAR)) AS 'SENO 45º (ou PI/4 rad)';
```

1. **Qual é a variável utilizada no exemplo e como ela foi definida?**

   A variável utilizada foi ‘angle’, sendo definida como 45º, porém em radianos (PI()/4).

2. **O que faz a função PI()?**

   A função PI() retorna o valor de **π** (3,14159…).

3. **O que faz a função CONCAT()?**

   É uma função utilizada para concatenar mais de uma string, transformando-as em uma só.

4. **O que faz a função CONVERT()?**

   É uma função utilizada para converter valores, nesse exemplo transformando INT em CHAR.

5. **Apresente o resultado do comando SELECT do exemplo.**

SENO 45º (ou PI/4 rad)
O SENO do ângulo: 0.785 rad = 0.707

```
-- b)
DROP PROCEDURE IF EXISTS proc_demo2;

DELIMITER \\
```

```
CREATE PROCEDURE proc_demo2(IN angle FLOAT, OUT output VARCHAR (100))
BEGIN
  SET output = '';
  SET output = CONCAT (output,
  ' [ ANGULO_GRAUS = ', CONVERT(ROUND(@angle * 180 / PI (), 3), CHAR), ']',
  ' [ ANGULO_RAD = ', CONVERT(ROUND(@angle, 3 ), CHAR), ']',
  ' [ SENO = ', CONVERT(ROUND(SIN(@angle),3 ), CHAR), ']',
  ' [ COSSENO = ', CONVERT(ROUND(COS(@angle),3 ), CHAR), ']',
  ' [ TANGENTE = ', CONVERT(ROUND(TAN(@angle),3 ), CHAR), ']');
END \\
DELIMITER ;

--------------------------------------------------------------------------------

-- c)
SET @angle = PI()/3;
SET @resp = '';
CALL proc_demo2(@angle, @resp);
SELECT @resp AS 'RESPOSTA';

SET @angle = PI()/4;
SET @resp = '';
CALL proc_demo2(@angle, @resp);
SELECT @resp AS 'RESPOSTA';

SET @angle = PI()/6;
SET @resp = '';
CALL proc_demo2(@angle, @resp);
SELECT @resp AS 'RESPOSTA';
```

1. **Em a., Para que servem os comandos das linhas 3. e 14.?**

   Para mudar o delimitador da linguagem. Na linha 3, é alterado para ‘\\’ e na linha 14 o delimitador volta a ser o padrão da linguage ‘;’.

2. **Quais são e como são definidos os parâmetros da procedure proc_demo2()?**

   Os parâmetros da procedure proc_demo2() são ‘IN angle FLOAT’, aceitando um valor float e ‘OUT output VARCHAR(100)’, retornando uma string de até 100 caracteres.

3. **O que está sendo feito na atribuição que inicia na linha 7.?**

   Está sendo reatribuídos valores à variável ‘output’ com as informações desejadas.

4. **Em c., apresente e explique o resultado o de cada um dos 3 conjuntos de comandos.**

É retornado o valor do ãngulo em graus e em radianos, juntamente ao seu SENO, COSSENO e TANGENTE

RESPOSTA
[ANGULO_GRAUS = 60] [ANGULO_RAD = 1.047] [SENO = 0.866] [COSSENO = 0.5] [TANGENTE = 1.732]

60º	1.047 rad	sen: 0.866	cos: 0.5	tg: 1.732

```
RESPOSTA
[ ANGULO_GRAUS = 45] [ ANGULO_RAD = 0.785] [ SENO = 0.707] [ COSSENO = 0.707] [ TANGENTE = 1]
```

45º	0.785 rad	sen: 0.707	cos: 0.707	tg: 1

```
RESPOSTA
[ ANGULO_GRAUS = 30] [ ANGULO_RAD = 0.524] [ SENO = 0.5] [ COSSENO = 0.866] [ TANGENTE = 0.577]
```

30º	0.524 rad	sen: 0.5	cos: 0.866	tg: 0.577

4.8 → Stores Procedures Recursivas

```
-- a)
DROP PROCEDURE IF EXISTS fatorial;

DELIMITER //
CREATE PROCEDURE fatorial(IN param INT, OUT total INT)
BEGIN
  DECLARE param_menos INT DEFAULT NULL ;
  DECLARE tmp_total INT DEFAULT -1;
  SET @@max_sp_recursion_depth = 50;
    IF (param IS NULL) OR (param < 0) OR (param > 12)
    THEN SET total = -1;

    ELSEIF (param = 0) OR (param = 1)
    THEN SET total = 1;

    ELSE
    SET param_menos = param - 1;
    CALL fatorial(param_menos, tmp_total);

      IF (tmp_total = -1)
      THEN SET total = -1;

      ELSE SET total = tmp_total * param;
      END IF;
    END IF;
END //
DELIMITER ;
```

1. **O que é feito no comando da linha 8.?**

   É definido o valor da variável global para 50.

2. **Se o parâmetro de entrada param não tiver valor (= NULL) qual será o valor do parâmetro de saída total?**

   O valor retornado será -1.

3. **Qual o valor máximo que podemos utilizar para calcular o fatorial, no exemplo passado?**

   O valor máximo que podemos utilizar é 12, já que acima disso será retornado o valor -1.

```
-- b)
-- conjunto commandos 1:
SET @resp = -1;
CALL fatorial (0, @resp);
SELECT @resp;

-- conjunto commandos 2:
CALL fatorial (13, @resp);
SELECT @resp;

-- conjunto commandos 3:
CALL fatorial (4, @resp);
SELECT @resp;

-- conjunto commandos 4:
CALL fatorial (-4, @resp);
SELECT @resp;

-- conjunto commandos 5:
CALL fatorial (6, @resp);
SELECT @resp;
```

1. **Mostre o resultado da execução do conjunto de comandos, indicando qual o valor numérico que está sendo calculado o fatorial.**

1) 0

@resp
1

2) 13

@resp
-1

3) 4

@resp
24

4) -4

@resp
-1

5) 6

@resp
720

2. **Justifique, de acordo com a stored procedure, o resultado apresentado**

Para os valores nulos, menores que 0 ou maiores que 12 o retono será -1, de acordo com o que foi definido dentro da procedure, os demais valores retornarão seu fatorial