Faculdade de Tecnologia de Sorocaba

Tecnologia em Análise e Desenvolvimento de Sistemas

Sistema de Gerenciamento de Serviços

Projeto Final - Base de Dados

Prof. Maria Angélica Calixto de Andrade Cardieri

Disciplina: Laboratório de Base de Dados

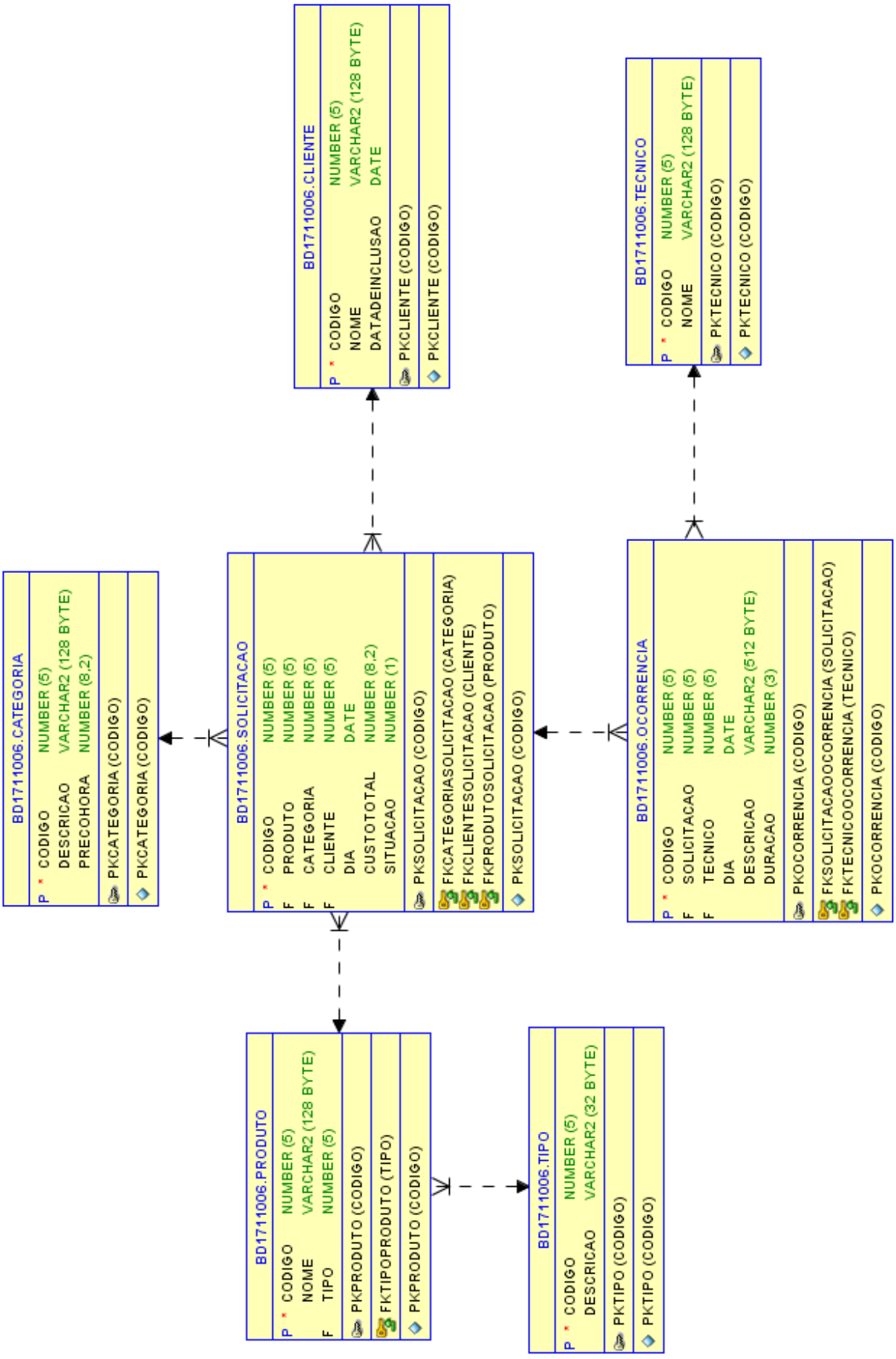
Pedro Bernardo de SOUSA	0030481711006
Weuller Júnior Souza Bessa	0030481621040
Vítor Andrade Marques da Silva	0030481511040
Walter Pereira Mendes Junior	0030481521040

Sorocaba

junho de 2019

Sumário

1 Diagrama de Entidades e Relacionamentos



2 Scripts de Criação do Esquema

2.1 Criação de Tabelas

schema ▸ create_tables.sql ▸ ...

```sql
CREATE TABLE Tipo ( /* do produto */
    codigo NUMBER(5, 0), /* PK */
    descricao VARCHAR2(32)
);

CREATE TABLE Produto (
    codigo NUMBER(5, 0), /* PK */
    nome VARCHAR2(128),
    tipo NUMBER(5, 0) /* FK para Tipo.codigo */
);

CREATE TABLE Cliente (
    codigo NUMBER(5, 0), /* PK */
    nome VARCHAR2(128)
);

CREATE TABLE Tecnico (
    codigo NUMBER(5, 0), /* PK */
    nome VARCHAR2(128)
);
```

```
21
22  CREATE TABLE Categoria (
23      codigo NUMBER(5, 0), /* PK */
24      descricao VARCHAR2(128),
25      precoHora NUMBER(8, 2) /* 999999.99 */
26  );
27
28  CREATE TABLE Ocorrencia (
29      codigo NUMBER(5, 0), /* PK */
30      solicitacao NUMBER(5, 0), /* FK para Solicitacao.codigo */
31      tecnico NUMBER(5, 0), /* FK para Tecnico.codigo */
32      dia DATE,
33      descricao VARCHAR2(512),
34      duracao NUMBER(3)
35  );
36
37  CREATE TABLE Solicitacao (
38      codigo NUMBER(5, 0), /* PK */
39      produto NUMBER(5, 0), /* FK para Produto.codigo */
40      categoria NUMBER(5, 0), /* FK para Categoria.codigo */
41      cliente NUMBER(5, 0), /* FK para Cliente.codigo */
42      dia DATE,
43      custoTotal NUMBER(8, 2), /* 999999.99 */
44      situacao NUMBER(1) /* 1 | 2 | 3 = não atendida | aguardando peças | atendida */
45  );
46
```

2.2 Script de Criação de Constraints

schema ▸ create_constraints.sql

```sql
ALTER TABLE Cliente
    ADD CONSTRAINT pkCliente PRIMARY KEY (codigo);

ALTER TABLE Tecnico
    ADD CONSTRAINT pkTecnico PRIMARY KEY (codigo);

ALTER TABLE Tipo
    ADD CONSTRAINT pkTipo PRIMARY KEY (codigo);

ALTER TABLE Produto
    ADD CONSTRAINT pkProduto PRIMARY KEY (codigo)
    ADD CONSTRAINT fkTipoProduto FOREIGN KEY (tipo) REFERENCES Tipo;

ALTER TABLE Categoria
    ADD CONSTRAINT pkCategoria PRIMARY KEY (codigo);

ALTER TABLE Solicitacao
    ADD CONSTRAINT pkSolicitacao PRIMARY KEY (codigo)
    ADD CONSTRAINT fkProdutoSolicitacao FOREIGN KEY (produto) REFERENCES Produto
    ADD CONSTRAINT fkCategoriaSolicitacao FOREIGN KEY (categoria) REFERENCES Categoria
    ADD CONSTRAINT fkClienteSolicitacao FOREIGN KEY (cliente) REFERENCES Cliente;

ALTER TABLE Ocorrencia
    ADD CONSTRAINT pkOcorrencia PRIMARY KEY (codigo)
    ADD CONSTRAINT fkTecnicoOcorrencia FOREIGN KEY (tecnico) REFERENCES Tecnico
    ADD CONSTRAINT fkSolicitacaoOcorrencia FOREIGN KEY (solicitacao) REFERENCES Solicitacao;

```

3 Populando O Banco de Dados

3.1 Clientes (64)

dataset ▸ insert_clients.sql

```sql
INSERT INTO Cliente (codigo, nome) VALUES (1, 'Amble Spilsburie');
INSERT INTO Cliente (codigo, nome) VALUES (2, 'Micky Skirling');
INSERT INTO Cliente (codigo, nome) VALUES (3, 'Mada Leer');
INSERT INTO Cliente (codigo, nome) VALUES (4, 'Becky Rutherforth');
INSERT INTO Cliente (codigo, nome) VALUES (5, 'Butch Pollak');
INSERT INTO Cliente (codigo, nome) VALUES (6, 'Julieta Seligson');
INSERT INTO Cliente (codigo, nome) VALUES (7, 'Armin Boreland');
```

[...]

```sql
INSERT INTO Cliente (codigo, nome) VALUES (57, 'Ainsley MacKenney');
INSERT INTO Cliente (codigo, nome) VALUES (58, 'Alice Glitherow');
INSERT INTO Cliente (codigo, nome) VALUES (59, 'Matthieu Harriot');
INSERT INTO Cliente (codigo, nome) VALUES (60, 'Ludvig Gregoraci');
INSERT INTO Cliente (codigo, nome) VALUES (61, 'Corilla Cheeseman');
INSERT INTO Cliente (codigo, nome) VALUES (62, 'Gideon Mattaus');
INSERT INTO Cliente (codigo, nome) VALUES (63, 'Chelsae Dunkinson');
INSERT INTO Cliente (codigo, nome) VALUES (64, 'Link Rockliffe');
```

3.2 Tipos (6)

```sql
INSERT INTO Tipo (codigo, descricao) VALUES (1, 'impressora');
INSERT INTO Tipo (codigo, descricao) VALUES (2, 'cabo');
INSERT INTO Tipo (codigo, descricao) VALUES (3, 'teclado');
INSERT INTO Tipo (codigo, descricao) VALUES (4, 'monitor');
INSERT INTO Tipo (codigo, descricao) VALUES (5, 'áudio');
INSERT INTO Tipo (codigo, descricao) VALUES (6, 'mouse');
```

3.3 Produtos (20)

```
/* mice */
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    1, 'LOGITECH M117', 6);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    2, 'LOGITECH MX Vertical Advanced Ergonomic Mouse', 6);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    3, 'LOGITECH MX Master 2S', 6);

/* keyboards */
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    4, 'LOGITECH CRAFT Advanced Keyboard with Creative Input Dial', 3);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    5, 'LOGITECH Wireless Illuminated Keyboard K800 Hand-proximity backlight', 3);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    6, 'LOGITECH Bluetooth Illuminated Keyboard K810 Easy-Switch multi-device keyboard', 3);

/* printers */
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    7, 'Canon Pro 1000 imagePROGRAF', 1);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    8, 'HP Ink Tank 116 Jato de Tinta Colorida Bivolt 3UM87A', 1);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    9, 'Plotter - Yes Printer Modelo 1801w Cabeça Xp600', 1);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    10, 'Termica Jly-58 58mm Tickts Pc', 1);

/* cables */
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    11, 'Ethernet 20 Metros', 2);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    12, 'Coaxial para Antena RG6 - 75OHMS', 2);

/* monitor */
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    13, 'LG LED 238 Widescreen Full HD IPS HDMI 24MK430H', 4);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    14, 'Dell de 21.5” P2219H (somente painel) - 210-arxl - 210-arxl', 4);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    15, 'LG 19,5" Led - 20M37Aa', 4);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    16, 'Philips 18,5" LED HD VGA Widescreen 193V5LSB2', 4);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    17, 'AOC LED 185 Widescreen VGA E970SWNL', 4);

/* audio */
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    18, 'ALTO-FALANTES LOGITECH Z120 - a7271767 - a7271767', 5);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    19, 'Caixa de Som C3 Tech 20 Portátil 3W RMS Preta SP 301BK', 5);
INSERT INTO Produto (codigo, nome, tipo) VALUES (
    20, 'Caixa de Som C3 Tech 20 Portátil 3W RMS Branca SP 301WH', 5);
```

3.4 Técnicos (10)

```
INSERT INTO Tecnico (codigo, nome) VALUES (1, 'Maria Angélica');
INSERT INTO Tecnico (codigo, nome) VALUES (2, 'César Munari');
INSERT INTO Tecnico (codigo, nome) VALUES (3, 'Jefferson Blaitt');
INSERT INTO Tecnico (codigo, nome) VALUES (4, 'Carolina Camargo');
INSERT INTO Tecnico (codigo, nome) VALUES (5, 'Tadeu Maffeis');
INSERT INTO Tecnico (codigo, nome) VALUES (6, 'Dimas Cardoso');
INSERT INTO Tecnico (codigo, nome) VALUES (7, 'Sérgio Bernardo');
INSERT INTO Tecnico (codigo, nome) VALUES (8, 'Fernando Miranda');
INSERT INTO Tecnico (codigo, nome) VALUES (9, 'Paulo Edson Alves');
INSERT INTO Tecnico (codigo, nome) VALUES (10, 'Tony Stark');
```

3.5 Categoria (2)

```
INSERT INTO Categoria (codigo, descricao, precohora) VALUES (1, 'Software', 30.0);
INSERT INTO Categoria (codigo, descricao, precohora) VALUES (2, 'Hardware', 20.0);
```

3.6 Solicitações (128)

dataset ▸ insert_solicitacoes.sql

```sql
INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (1, 18, 2, 34, '8/4/2019', 0, 2);

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (2, 13, 1, 49, '6/1/2019', 0, 1);

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (3, 9, 1, 1, '9/3/2019', 0, 2);

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (4, 10, 2, 38, '6/2/2019', 0, 1);

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (5, 15, 1, 6, '25/6/2019', 0, 1);

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (6, 19, 2, 1, '9/7/2018', 0, 1);

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (7, 19, 1, 59, '18/10/2018', 0, 1);
```

[...]

```sql

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (127, 19, 2, 53, '28/3/2019', 0, 1);

INSERT INTO Solicitacao
(codigo, produto, categoria, cliente, dia, custototal, situacao)
VALUES (128, 19, 1, 14, '8/9/2018', 0, 1);

```

3.7 Ocorrências (513)

dataset ▸ insert_ocorrencias.sql

```
1   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
    (1, 10, 8, '1/11/2018', 'Curabitur at ipsum ac tellus semper interdum. Mauris
    ullamcorper purus sit amet nulla. Quisque arcu libero, rutrum ac, lobortis vel,
    dapibus at, diam.', 8);
2   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
    (2, 65, 8, '18/3/2019', 'Aenean fermentum. Donec ut mauris eget massa tempor
    convallis. Nulla neque libero, convallis eget, eleifend luctus, ultricies eu, nibh.',
    35);
3   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
    (3, 18, 7, '28/1/2019', 'Pellentesque at nulla. Suspendisse potenti. Cras in purus eu
    magna vulputate luctus.', 26);
4   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
    (4, 107, 9, '17/8/2018', 'Curabitur gravida nisi at nibh. In hac habitasse platea
    dictumst. Aliquam augue quam, sollicitudin vitae, consectetuer eget, rutrum at,
    lorem.', 32);
5   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
    (5, 61, 5, '25/9/2018', 'Proin eu mi. Nulla ac enim. In tempor, turpis nec euismod
    scelerisque, quam turpis adipiscing lorem, vitae mattis nibh ligula nec sem.', 37);
6   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
    (6, 26, 1, '4/5/2019', 'Aenean lectus. Pellentesque eget nunc. Donec quis orci eget


      [...]


509   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
      (509, 66, 7, '28/9/2018', 'Curabitur gravida nisi at nibh. In hac habitasse platea
      dictumst. Aliquam augue quam, sollicitudin vitae, consectetuer eget, rutrum at,
      lorem.', 14);
510   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
      (510, 34, 5, '30/6/2018', 'Cras non velit nec nisi vulputate nonummy. Maecenas
      tincidunt lacus at velit. Vivamus vel nulla eget eros elementum pellentesque.', 12);
511   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
      (511, 54, 5, '18/6/2018', 'Sed ante. Vivamus tortor. Duis mattis egestas metus.', 14);
512   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
      (512, 14, 2, '27/6/2018', 'Phasellus in felis. Donec semper sapien a libero. Nam
      dui.', 12);
513   INSERT INTO Ocorrencia (codigo, solicitacao, tecnico, dia, descricao, duracao) VALUES
      (513, 11, 10, '27/6/2018', 'Can Jarvis solve it?', 1);
514
```

4 Comandos SQL para realizar consultas

4.1 Listar o código do cliente, o nome do cliente e todas as solicitações efetuados por ele no mês de junho/2019.

exercicios ▸ 4.1.sql

```sql
SELECT * from Cliente INNER JOIN Solicitacao
ON Cliente.codigo = Solicitacao.cliente
WHERE to_char(dia, 'MM-YYYY') = '06-2019';

```

4.2 Listar o produto que possui o maior número de solicitações cadastradas já atendidas.

exercicios ▸ 4.2.sql

```sql
select * from Produto where Produto.codigo = (
    select Solicitacao.produto from Solicitacao
    group by Solicitacao.produto
    having count(*) = (
        select max(count(*)) from Solicitacao group by produto
    )
);

```

4.3 Listar o número de solicitações existentes para cada tipo de produto e a descrição do tipo.

exercicios ▸ 4.3.sql

```sql
SELECT descricao, solicitacoes FROM Tipo INNER JOIN (
    SELECT tipo, count(*) as solicitacoes FROM Solicitacao INNER JOIN Produto
    ON Solicitacao.produto = Produto.codigo
    INNER JOIN Tipo
    ON Tipo.codigo = tipo
    GROUP BY tipo
) ON Tipo.codigo = tipo;

```

4.4 Criar uma visão com o custo total das manutenções realizadas para cada cliente. Considere apenas as solicitações atendidas. Esta visão é atualizável? Por quê?

exercicios ▸ 4.4.sql ▸ ...

```sql
CREATE OR REPLACE VIEW CustoPorCliente AS
SELECT Solicitacao.cliente, SUM(Ocorrencia.duracao * Categoria.precoHora) as custo
FROM Categoria INNER JOIN Solicitacao ON Solicitacao.categoria = Categoria.codigo
INNER JOIN Ocorrencia ON Ocorrencia.solicitacao = Solicitacao.codigo
WHERE Solicitacao.situacao = 3 GROUP BY Solicitacao.cliente;

SELECT * FROM CustoPorCliente;

```

4.5 Listar todas as ocorrências das solicitações não atendidas.

exercicios ▸ 4.5.sql

```sql
SELECT
    Ocorrencia.codigo,
    Ocorrencia.dia,
    Ocorrencia.descricao,
    Ocorrencia.duracao,
    Ocorrencia.tecnico,
    Ocorrencia.solicitacao
FROM Solicitacao INNER JOIN Ocorrencia
ON Solicitacao.codigo = Ocorrencia.solicitacao
WHERE Solicitacao.situacao = 1;

```

4.6 Listar a descrição da categoria de problema que mais ocorreu nos últimos 2 meses.

exercicios ▸ 4.6.sql

```sql
SELECT descricao FROM Categoria
WHERE codigo = (
    SELECT codigo FROM (
        SELECT Categoria.codigo, COUNT(Categoria.codigo) as c FROM Categoria INNER JOIN
        Solicitacao ON Categoria.codigo = Solicitacao.categoria
        WHERE Solicitacao.dia >= SYSDATE-60
        GROUP BY Categoria.codigo
    ) WHERE c = (
        SELECT MAX(COUNT(Categoria.codigo)) FROM Categoria INNER JOIN Solicitacao ON
        Categoria.codigo = Solicitacao.categoria
        WHERE Solicitacao.dia >= SYSDATE-60
        GROUP BY Categoria.codigo
    )
);

```

4.7 Liste o código do produto que nunca teve uma solicitação de manutenção.

exercicios ▸ 4.7.sql

```sql
SELECT codigo FROM Produto
WHERE codigo NOT IN (
    SELECT produto FROM Solicitacao
);

```

4.8 Listar o nome dos técnicos que tenham solicitações parcialmente atendidas e para quem que já exista mais de 2 ocorrências para a solicitação.

exercicios ▸ 4.8.sql

```sql
SELECT nome FROM Tecnico WHERE codigo IN (
    SELECT Ocorrencia.tecnico FROM Solicitacao INNER JOIN Ocorrencia
    ON Ocorrencia.solicitacao = Solicitacao.codigo INNER JOIN Tecnico
    ON Ocorrencia.tecnico = Tecnico.codigo
    WHERE situacao = 2
    GROUP BY Ocorrencia.tecnico
    HAVING COUNT(*) > 2
);

```

4.9 Acrescente uma coluna nova “data de inclusão” no formato DATE, na primeira tabela criada. Altere o valor desta coluna colocando a data do sistema.

exercicios ▸ 4.9.sql

```
/*
    No nosso caso, a primeira tabela criada foi Tipo. Decidimos alterar a tabela
    Cliente, no lugar de Tipo, por nos parecer que, para um sistema como este,
    registrar a data de inserção de um cliente é mais útil do que a dos tipos
    recorrentes de problemas.
*/

ALTER TABLE Cliente ADD dataDeInclusao DATE;
UPDATE Cliente SET dataDeInclusao = SYSDATE;

```

4.10 Explique para que serve a claúsula GROUP BY e dê 1 exemplo de sua utilização.

exercicios ▸ 4.10.sql

```
/*
    A cláusula GROUP BY serve para regrupar as linhas de uma tabela em conjuntos que
    tem algo em comum. Aquilo que as linhas agrupadas tem em comum é o conteúdo da
    coluna informada logo após o comando GROUP BY. Por exemplo, no comando SQL abaixo,
    agrupamos as linhas da tabela Ocorrencia por técnico, de forma que subconjuntos de
    occorências, com o técnico em comum, são criados em memória principal. Projetamos o
    código do técnico (Ocorrencia.tecnico) e o número de ocorrências em cada
    subconjunto.
*/

SELECT Ocorrencia.tecnico, COUNT(*) FROM Ocorrencia
GROUP BY Ocorrencia.tecnico;

```

4.11 Explique para que serve a claúsula HAVING e dê 1 exemplo de sua utilização.

exercicios ▸ 4.11.sql

```sql
/*
    A cláusula HAVING serve para limitar o conjunto resultante de uma pesquisa que
    contém uma função de agregação. Por exemplo, se utilizarmos uma função SUM, que é
    de agregação, na cláusula SELECT, podemos filtrar com base na adição fornecida pela
    função SUM, através da cláusula HAVING. Por exemplo, na requisição abaixo,
    regrupamos as ocorrências em suconjuntos que tem o técnico em comum, contamos os
    elementos em cada subconjunto e selecionamos apenas os conjuntos que tem mais de 50
    ocorrências, através da cláusula HAVING.
*/

SELECT tecnico, COUNT(*) as ocorrencias
FROM Ocorrencia
GROUP BY tecnico
HAVING COUNT(*) > 50;

```

4.12 Dê exemplo de um comando utilizando subconsultas que utilize o operador IN.

exercicios ▸ 4.12.sql

```sql
/*
    O operador IN serve para verificar se um determinado dado se encontra em um
    conjunto de dados do mesmo tipo. Por exemplo, na consulta abaixo, usamos o
    resultado unidimencional da subconsulta (que retorna uma lista de códigos de
    tecnicos com mais de 50 ocorrências), para apresentar os nomes desses técnicos. O
    operador IN verifica, para cada linha da tabela Tecnico, se o código figura dentro
    da lista de códigos retornados pela subconsulta.
*/

SELECT nome FROM Tecnico WHERE codigo IN (
    SELECT tecnico FROM Ocorrencia
    GROUP BY tecnico
    HAVING COUNT(*) > 50
);

```

4.13 Dê exemplo de um comando utilizando subconsultas que utilize o operador NOT IN.

exercicios ▸ 4.13.sql

```sql
/*
    O operador NOT IN verifica se um determinado dado (à esquerda do operador) não se
    encontra na lista de dados do mesmo tipo (à direita do operador). Se o dado não
    estiver no conjunto, o operador retorna TRUE. Basicamente, este operador faz o
    inverso do que IN faz. Por exemplo, na consulta abaixo, o operador NOT IN verifica,
    para cada linha da tabela Tecnico, se o valor da coluna codigo não está na tabela
    unidimencional retornada pela subconsulta. Como a subconsulta retorna uma lista de
    códigos de tecnicos que tem mais de 50 ocorrências, listamos os nomes dos técnicos
    que atenderam até 50 occorências.
*/

SELECT nome FROM Tecnico WHERE codigo NOT IN (
    SELECT tecnico FROM Ocorrencia
    GROUP BY tecnico
    HAVING COUNT(*) > 50
);

```

4.14 Dê exemplo de um comando utilizando subconsultas que utilize o operador EXISTS.

exercicios ▸ 4.14.sql

```sql
/*
    O operador EXISTS deve ser utilizado na cláusula WHERE e deve ser seguido de uma
    subconsulta do tipo SELECT. Se a subconsulta retornar uma ou mais linhas, a
    cláusula WHERE que precede EXISTS vai ser avaliada em TRUE. Por exemplo, a consulta
    abaixo lista, no dataset deste trabalho, os 54 clientes que estão associados a pelo
    menos uma solicitação. A subconsulta vai retornar um certo número de linhas para
    cada Cliente, com base em quantas ocorrências daquele cliente tem na tabela
    Solicitacao. O conjunto de resultados será zero quando o cliente da vez não tiver
    nenhum solicitação. Cada vez que isso acontece, o operador EXISTS faz com que a
    cláusula WHERE retorne FALSE e a linha da tabela Cliente não é selecionada.
*/

SELECT nome FROM Cliente
WHERE EXISTS (
    SELECT 1 FROM Solicitacao
    WHERE Solicitacao.cliente = Cliente.codigo
);

```

4.15 Dê exemplo de um comando utilizando subconsultas que utilize o operador NOT EXISTS.

exercicios ▸ 4.15.sql

```sql
/*
    O operador NOT EXISTS funciona de maneira inversa ao EXISTS. Seu posicionamento é similar, insto é, na cláusua WHERE, seguido de uma subconsulta, e ele retorna TRUE apenas quando a subconsulta não retorna linha alguma. No exemplo abaixo, a subconsulta retorna uma linha para cada solicitação cujo código do cliente corresponde a o cliente da vez, de forma que são mostrados os nomes dos clientes que nunca fizerem nenhum pedido.
*/

SELECT nome FROM Cliente
WHERE NOT EXISTS (
    SELECT 1 FROM Solicitacao
    WHERE Solicitacao.cliente = Cliente.codigo
);

```

4.16 Dê exemplo de uma subconsulta utilizada dentro de um comando UPDATE.

exercicios ▸ 4.16.sql

```sql
/*
    No comando UPDATE abaixo, 3 subconsultas são utilizadas.
    A mais interna (a primeira a ser executada) identifica a categoria da Solicitação 10.
    O resultado dela é utilizado para selecionar o preço por hora daquela categoria.
    Este valor é multiplicado pelo resultado de uma terceira subconsulta, que retorna a soma de todas as ocorrências associadas à solicitação 10.
    O resultado dessa multiplicação é ustilizado pelo UPDATE, para registrar, na coluna custoTotal da solicitação 10, seu custo total.

    Note que o único valor constante neste update é o código 10. Esta consulta é, portanto, utilizável por uma PROCEDURE que recebe o código da Solicitação.
*/

UPDATE Solicitacao SET custoTotal = (
    (
        SELECT precoHora FROM Categoria WHERE codigo = (
            SELECT categoria FROM Solicitacao WHERE codigo = 10
        )
    ) * (
        SELECT SUM(duracao) FROM Ocorrencia WHERE solicitacao = 10
    )
) WHERE codigo = 10;

```

4.17 Dê exemplo de uma subconsulta utilizada dentro de um comando DELETE.

exercicios ▸ 4.17.sql

```sql
/*
    A comando DELETE abaixo apaga qualquer linha na tabela Produto que não seja
    referenciada pela tabela Solicitacao
*/

DELETE Produto WHERE NOT EXISTS (
    SELECT produto FROM Solicitacao WHERE Solicitacao.produto = Produto.codigo
);

```

4.18 Dê exemplo de uma consulta utilizando a cláusula MINUS.

exercicios ▸ 4.18.sql

```sql
/*
    A consulta abaixo apresenta os códigos das solicitações do cliente 5 que não dizem
    respeito ao produto 19. No dataset deste trabalho, há 5 solicitações do cliente 5 e
    12 solicitações para o produto 19. Com a cláusula MINUS, podemos eliminar do
    resultado a interceção entre esses dois conjuntos.
*/

SELECT * FROM (
    SELECT Solicitacao.codigo as solicitacao, Solicitacao.produto, Solicitacao.cliente
    FROM Solicitacao INNER JOIN Produto
    ON Solicitacao.produto = Produto.codigo
    INNER JOIN Cliente
    ON Solicitacao.cliente = Cliente.codigo
    WHERE Cliente.codigo = 5

    MINUS

    SELECT Solicitacao.codigo as solicitacao, Solicitacao.produto, Solicitacao.cliente
    FROM Solicitacao INNER JOIN Produto
    ON Solicitacao.produto = Produto.codigo
    INNER JOIN Cliente
    ON Solicitacao.cliente = Cliente.codigo
    WHERE Produto.codigo = 19
);
```

4.19 Dê exemplo de uma consulta utilizando a cláusula INTERSECT.

exercicios ▸ 4.19.sql

```sql
/*
    As mesmas seleções do exercício 4.18 são utilizadas aqui para demonstrar a cláusula
    INTERSECT. Nota-se que a obrigatoriedade de operar sobre tabelas de estrutura
    idência é compartilhada entre estes dois recusos.
    O resultado neste caso é a seleção da interseção entre o conjunto de solicitações
    do cliente 5 e o conjunto de solicitações sobre o produto 19.
    No dataset destre trabalho, apenas duas linhas pertencem a interseção
*/

SELECT * FROM (
    SELECT Solicitacao.codigo as solicitacao, Solicitacao.produto, Solicitacao.cliente
    FROM Solicitacao INNER JOIN Produto
    ON Solicitacao.produto = Produto.codigo
    INNER JOIN Cliente
    ON Solicitacao.cliente = Cliente.codigo
    WHERE Cliente.codigo = 5
INTERSECT
    SELECT Solicitacao.codigo as solicitacao, Solicitacao.produto, Solicitacao.cliente
    FROM Solicitacao INNER JOIN Produto
    ON Solicitacao.produto = Produto.codigo
    INNER JOIN Cliente
    ON Solicitacao.cliente = Cliente.codigo
    WHERE Produto.codigo = 19
);
```

5 Função Útil

Escreva uma função que seja útil para a lógica de negócios de seu sistema e indique o contexto de sua utilização

exercicios ▸ 5.sql ▸ ...

```
/*
    A função abaixo recebe o código de um produto e, através da junção das tabelas Solicitacao e
    Ocorrencia, calcula a média de duração das ocorrências para aquele produto. Isso pode ser útil
    para prever, com base em ocorrências passadas, quanto tempo uma nova ocorrência pode levar.
*/

CREATE OR REPLACE FUNCTION tempoMedioDeOcorrenciaPorProduto(p IN Produto.codigo%TYPE)
RETURN NUMBER
AS
duracaoMedia NUMBER;
BEGIN
    SELECT AVG(duracao) INTO duracaoMedia FROM Solicitacao INNER JOIN Ocorrencia
    ON Ocorrencia.solicitacao = Solicitacao.codigo
    WHERE produto = p;

    RETURN duracaoMedia;
END tempoMedioDeOcorrenciaPorProduto;
/

/* TESTE: produto 1 -> 26,2608696 */
SELECT tempoMedioDeOcorrenciaPorProduto(1) FROM Dual;

```

6 Trigger para Muitas Ocorrências

Escreva um trigger que, ao incluir-se uma ocorrência, se já houver mais de 3 ocorrências da mesma solicitação, grava em uma tabela de log a mensagem "Situação Grave – grande número de ocorrências <cod_solicitacao> <nome_cliente> <qtde>".

exercicios ▸ 6.sql ▸ ...

```sql
/* Criar tabela de logs */
CREATE TABLE Log (
    codigo NUMBER(5, 0),
    mensagem VARCHAR2(256)
);
ALTER TABLE Log ADD CONSTRAINT pkLog PRIMARY KEY (codigo);
CREATE SEQUENCE LogSeq START WITH 1;

/* Criar trigger */
CREATE OR REPLACE TRIGGER MuitasOcorrencias
BEFORE INSERT ON Ocorrencia
FOR EACH ROW
DECLARE
n NUMBER;
cod_cli Cliente.codigo%TYPE;
nome_cli Cliente.nome%TYPE;
BEGIN
    SELECT COUNT(*) INTO n FROM Ocorrencia WHERE solicitacao = :NEW.solicitacao;
    SELECT cliente INTO cod_cli FROM Solicitacao WHERE codigo = :NEW.solicitacao;
    SELECT nome INTO nome_cli FROM Cliente WHERE codigo = cod_cli;

    IF n > 3 THEN
        INSERT INTO Log (codigo, mensagem) VALUES (LogSeq.nextval, 'Situação Grave – grande número de
        ocorrências ' || :NEW.solicitacao || ' ' || nome_cli || ' ' || (n+1));
    END IF;
END MuitasOcorrencias;
/
```

```sql
/* TESTES */
/*
    A solicitação 71 tem apenas uma ocorrência. Incluir mais uma não deve acionar o trigger. PASSANDO
*/
INSERT INTO Ocorrencia
(codigo, solicitacao, tecnico, dia, descricao, duracao)
VALUES
(514, 71, 8, '1/11/2018', 'abc', 1);
SELECT * FROM Log;

/*
    A solicitação 6 tem 3 ocorrências. Incluir uma 4ª não deve acionar o trigger, mas incluir uma 5ª
    deve.
*/
/* +1 PASSANDO */
INSERT INTO Ocorrencia
(codigo, solicitacao, tecnico, dia, descricao, duracao)
VALUES
(515, 6, 10, '1/11/2018', 'abc', 1);
SELECT * FROM Log;

/* +2 PASSANDO */
INSERT INTO Ocorrencia
(codigo, solicitacao, tecnico, dia, descricao, duracao)
VALUES
(516, 6, 10, '1/11/2018', 'abc', 1);
SELECT * FROM Log;

 /*
    A solicitação 52 tem 4 ocorrências. Incluir mais uma deve acionar o trigger. PASSANDO
 */
 INSERT INTO Ocorrencia
 (codigo, solicitacao, tecnico, dia, descricao, duracao)
 VALUES
 (517, 52, 10, '1/11/2018', 'abc', 1);
 SELECT * FROM Log;

 /*
    A solicitação 64 tem 5 ocorrências. Incluir mais uma deve acionar o trigger. PASSANDO
 */
 INSERT INTO Ocorrencia
 (codigo, solicitacao, tecnico, dia, descricao, duracao)
 VALUES
 (518, 64, 10, '1/11/2018', 'abc', 1);
 SELECT * FROM Log;

 /*
    A solicitação 50 tem 7 ocorrências. Incluir mais uma deve acionar o trigger. PASSANDO
 */
 INSERT INTO Ocorrencia
 (codigo, solicitacao, tecnico, dia, descricao, duracao)
 VALUES
 (519, 50, 10, '1/11/2018', 'abc', 1);
 SELECT * FROM Log;
```

7 Trigger de Exclusão em Cascata

Escreva um trigger que, ao excluir-se uma solicitação, exclua também as suas ocorrências.

exercicios ▸ 7.sql ▸ TRIGGER excluirOcorrenciasDependentes

```sql
CREATE OR REPLACE TRIGGER excluirOcorrenciasDependentes
BEFORE DELETE ON Solicitacao
FOR EACH ROW
BEGIN
    DELETE FROM Ocorrencia WHERE solicitacao = :OLD.codigo;
END excluirOcorrenciasDependentes;
/

/* TESTE
    A solicitação 50 tem 8 ocorrências. Zero devem existir após sua exclusão.
PASSANDO */
SELECT COUNT(*) FROM Ocorrencia WHERE Ocorrencia.solicitacao = 50; /* 8 */
DELETE FROM Solicitacao WHERE codigo = 50;
SELECT COUNT(*) FROM Ocorrencia WHERE Ocorrencia.solicitacao = 50; /* 0 */

```

8 Procedure para Custo de Manutenção

Escreva uma procedure que calcule o custo de uma manutenção. Esta procedure deve receber como parâmetro o código da Solicitação e somar as horas de todas as ocorrências realizadas para esta solicitação. Considerar que a unidade é sempre horas inteiras (desconsiderar minutos). O custo base é

Se tipoProd = ´HW´ custo = R$ 20,00 por hora
Se tipoProd = ´SW´ custo = R$ 30,00 por hora

O custo total não pode ser menor do que o preço mínimo para a categoria.

exercicios ▸ 8.sql

```sql
CREATE OR REPLACE PROCEDURE custoManutencao(pCod Solicitacao.codigo%TYPE)
AS
custoHora Categoria.precoHora%TYPE;
totalDuracao NUMBER;
cTotal NUMBER(8, 2);
BEGIN
    SELECT precoHora INTO custoHora FROM Categoria
    WHERE codigo = (SELECT categoria FROM Solicitacao WHERE codigo = pCod);

    SELECT SUM(Ocorrencia.duracao) INTO totalDuracao FROM Ocorrencia WHERE solicitacao = pCod;

    cTotal := custoHora * totalDuracao;

    UPDATE Solicitacao SET custoTotal = cTotal WHERE pCod = Solicitacao.codigo;
END custoManutencao;
/

```

9 Procedure para Classificar Solicitação

Escreva uma procedure que receba como parâmetro o código do produto e verifique quantas requisições existem (em qualquer situação) e classifique:

Se qtde de requisições >= 15 “Produto Ruim – não recomendar”
Se qtde de requisições >= 5 e < 15 “Produto a ser verificado”
Se qtde de requisições < 5 e > 0 “Produto Bom”
Se qtde de requisições = 0 “Produto Excelente – recomendar”

Gravar uma linha na tabela de Mensagem com: codproduto, nomeproduto e a classificação atribuída acima.

exercicios ▸ 9.sql

```sql
/* Criar tabela de mensagens */
CREATE TABLE Mensagem (
    codigo NUMBER(5, 0),
    codproduto NUMBER(5, 0),
    nomeproduto VARCHAR2(128 BYTE),
    classificacao VARCHAR2(128 BYTE)
);
ALTER TABLE Mensagem ADD CONSTRAINT codigo PRIMARY KEY (codigo);
CREATE SEQUENCE menSeq START WITH 1;

/* Criar PROCEDURE */
CREATE OR REPLACE PROCEDURE classificarSolicitacao(pCod Produto.codigo%TYPE)
AS
total NUMBER;
n Produto.nome%TYPE;
BEGIN
    SELECT nome INTO n FROM Produto WHERE Produto.codigo = pCod;
    SELECT COUNT(*) INTO total FROM Solicitacao WHERE Solicitacao.produto = pCod;

    IF total >= 15 THEN
        INSERT INTO Mensagem VALUES (menSeq.nextVal, pCod, n, 'Produto Ruim - não recomendar');
    ELSIF total >=5 AND total < 15 THEN
        INSERT INTO Mensagem VALUES (menSeq.nextVal, pCod, n, 'Produto a ser verificado');
    ELSIF total > 0 AND total < 5 THEN
        INSERT INTO Mensagem VALUES (menSeq.nextVal, pCod, n, 'Produto Bom');
    ELSIF total = 0 THEN
        INSERT INTO Mensagem VALUES (menSeq.nextVal, pCod, n, 'Produto Excelente - recomendar');
    END IF;

END classificarSolicitacao;
/

/* Executar PROCEDURE para os 10 primeiros produtos */
BEGIN
    FOR a IN 1 .. 10 LOOP
        classificarSolicitacao(a);
    END LOOP;
END;
/

/* Ler resultados */
SELECT * FROM Mensagem;

```

10 Considerações Finais

Para executar o projeto:

1) git clone https://github.com/bernardodesousa/SGS.git
2) executar main.sql

Mais informações em

https://github.com/bernardodesousa/SGS

* * *