UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS e HUMANIDADES
GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Prof. Dr. José de Jesús Pérez Alcázar

Aaron Ferraz do Amaral Martins da Silva - 14554474

Alaina R. dos Santos - 11837352

Lucas Gurgel do Amaral - 14760234

Pedro Henrique Santos Matos - 13684915

Relatório do EP3 de Banco de Dados 1

São Paulo

2024

Requisitos do EP

Consultas - são definidas no documento abaixo. Mostramos quais são os resultados no vídeo com base em nossa popularização do banco de dados.
Trigger - está implementado ao longo do texto, considerando a manutenção do "preco" levando em consideração a empresa e a cidade.
Aplicação - foi feita usando a biblioteca Swing e o driver JDBC, abordaremos melhor no vídeo.
SGBD - Ao realizar esse trabalho optamos por usar o SGBD *MySQL*.
Linguagem - escolhemos o Java.
Criação do Banco de dados - explicitamos o banco de dados que foi criado no SGBD supracitado, além dos dados inseridos para o testamento do sistema.
Execução do programa - demonstramos no vídeo e no relatório abaixo.
Considerações - houveram algumas dúvidas que o enunciado não abordava, decidimos considerar certas situações.

SCRIPT CRIAÇÃO DAS TABELAS

Seguimos o esquema do modelo entidade relacionamento fornecido como resultado da primeira parte do EP. A criação das tabelas levou em consideração a ordem para evitar conflitos de dependência entre as relações.

```
-- Criação do banco de dados e utilização
CREATE database EP3;
USE EP3;

CREATE TABLE Empresa(
  idEmpresa INTEGER NOT NULL,
  nome VARCHAR(20) NOT NULL,
  endereco VARCHAR(30),
  CONSTRAINT ID_Empresa_PK PRIMARY KEY (idEmpresa)
);

CREATE TABLE Cidade (
  idCidade INTEGER NOT NULL,
  nomeCidade VARCHAR(40) UNIQUE NOT NULL,
  estado VARCHAR(2),
  CONSTRAINT ID_Cidade_PK PRIMARY KEY (idCidade)
);

CREATE TABLE Cliente (
  idCliente INTEGER NOT NULL,
```

```
  CPF CHAR(11) UNIQUE NOT NULL,
  RG VARCHAR(12) UNIQUE NOT NULL,
  nomeCompleto VARCHAR(100) NOT NULL,
  endereco VARCHAR(50),
  CONSTRAINT ID_Cliente_PK PRIMARY KEY (idCliente)
);

CREATE TABLE Funcionario (
  CPFfunc CHAR(11) NOT NULL,
  enderecoFunc VARCHAR(50),
  RGfunc VARCHAR(20) UNIQUE NOT NULL,
  salario DECIMAL(10, 2) NOT NULL,
  tipoFunc VARCHAR(50),
  telefoneCont VARCHAR(15),
  nomeCompletoFunc VARCHAR(100) NOT NULL,
  CONSTRAINT ID_CPFfunc_PK PRIMARY KEY (CPFfunc)
);

-- Criação da tabela Servico
CREATE TABLE Servico(
  idServico INTEGER NOT NULL,
  nomeServ VARCHAR(40) UNIQUE NOT NULL,
  servTipo VARCHAR(20) NOT NULL CHECK (servTipo IN ('Guindaste', 'Transporte')),
  CONSTRAINT ID_Servico_PK PRIMARY KEY (idServico)
);

-- Tabelas dependentes de Servico
CREATE TABLE Serv_guindaste (
  idServico INT PRIMARY KEY,
  altura DECIMAL(5, 2),
  tamanhoBase DECIMAL(6, 2),
  percentualBonus DECIMAL(3, 2),
  aumentoBonus DECIMAL(7, 2),
    CONSTRAINT FK_Guindaste_Servico FOREIGN KEY (idServico) REFERENCES
Servico(idServico) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Serv_transporte (
  idServico INT PRIMARY KEY,
  percentualAcres DECIMAL(5, 2),
  limiteCarga DECIMAL(7, 2),
    CONSTRAINT FK_Transporte_Servico FOREIGN KEY (idServico) REFERENCES
Servico(idServico) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
-- Tabela Pedido
CREATE TABLE Pedido (
  idPedido INTEGER NOT NULL,
  idCliente INT NOT NULL,
  idEmpresa INT NOT NULL,
  idCidade INT NOT NULL,
  enderecoDest VARCHAR(150),
  enderecoPart VARCHAR(150),
  CONSTRAINT ID_Pedido_PK PRIMARY KEY (idPedido),
      CONSTRAINT FK_Pedido_Cliente FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente) ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT FK_Pedido_Empresa FOREIGN KEY (idEmpresa) REFERENCES Empresa(idEmpresa) ON DELETE CASCADE ON UPDATE CASCADE,
      CONSTRAINT FK_Pedido_Cidade FOREIGN KEY (idCidade) REFERENCES Cidade(idCidade) ON DELETE CASCADE ON UPDATE CASCADE
);

-- Tabela Solicitam
CREATE TABLE Solicitam (
  idServico INTEGER,
  idPedido INTEGER NOT NULL,
  CPFfunc CHAR(11),
  tempoDurac DECIMAL(5, 2) NOT NULL,
  preco DECIMAL(10, 2) NOT NULL CHECK(preco > 0),
  dataFim DATE NOT NULL,
     CONSTRAINT FK_Solicitam_Servico FOREIGN KEY (idServico) REFERENCES Servico(idServico) ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT FK_Solicitam_Pedido FOREIGN KEY (idPedido) REFERENCES Pedido(idPedido) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_Solicitam_CPFfunc FOREIGN KEY (CPFfunc) REFERENCES Funcionario(CPFfunc) ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT Servico_Pedido_CPFfunc_PK PRIMARY KEY(idServico, idPedido, CPFfunc)
);

-- Telefones
CREATE TABLE TelefonesCliente (
  idCliente INT NOT NULL,
  numTelefone VARCHAR(15) NOT NULL,
  CONSTRAINT Cliente_numTelefone_PK PRIMARY KEY(idCliente, numTelefone),
   CONSTRAINT FK_TelefonesCliente_Cliente FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE TelefonesEmpresa (
  idEmpresa INT NOT NULL,
  numTelefone VARCHAR(15) NOT NULL,
  CONSTRAINT Empresa_numTelefone_PK PRIMARY KEY(idEmpresa, numTelefone),
      CONSTRAINT FK_TelefonesEmpresa_Empresa FOREIGN KEY (idEmpresa) REFERENCES Empresa(idEmpresa) ON DELETE CASCADE ON UPDATE CASCADE
);

-- Tabela Oferecem
CREATE TABLE Oferecem (
  idEmpresa INT,
  idCidade INT,
  idServico INT,
  precoHoraSe DECIMAL(6,2) NOT NULL CHECK(precoHoraSe > 0),
    CONSTRAINT FK_Oferecem_Empresa FOREIGN KEY (idEmpresa) REFERENCES Empresa(idEmpresa) ON DELETE CASCADE ON UPDATE CASCADE,
     CONSTRAINT FK_Oferecem_Cidade FOREIGN KEY (idCidade) REFERENCES Cidade(idCidade) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_Oferecem_Servico FOREIGN KEY (idServico) REFERENCES Servico(idServico) ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT Empresa_Servico_Cidade_PK PRIMARY KEY (idEmpresa, idServico, idCidade)
);
```

INSERÇÃO DAS TUPLAS

```
-- Inserções para atender às consultas solicitadas

-- Empresas
INSERT INTO Empresa (idEmpresa, nome, endereco) VALUES
(1, 'ServTransportes', 'Rua Principal 123'),
(2, 'GuindasteMax', 'Avenida Central 456'),
(3, 'TransMov', 'Rua Terno 190'),
(4, 'RodoMud', 'Rua dos Braulios 789'),
(5, 'AllMoves', 'Avenida Lambda 860'),
(6, 'MudTrans', 'Avenida Cabo da Saudade 123'),
(7, 'TransRodo', 'Rua Dr. Jose 111'),
(8, 'MultiTransp', 'Avenida Água Seca 9790'),
(9, 'EmpSeguTrans', 'Rua Carlotti 142'),
(10, 'Transesp', 'Rua Araucária 777'),
(11, 'TaBaratoTransp', 'Rua Perez 2991'),
(12, 'Metas&Metas', 'Avenida Cariacica 222');
```

```
-- Cidades
INSERT INTO Cidade (idCidade, nomeCidade, estado) VALUES
(1, 'São Paulo', 'SP'),
(2, 'Rio de Janeiro', 'RJ'),
(3, 'Belo Horizonte', 'MG'),
(4, 'Salvador', 'BA'),
(5, 'Rio Branco', 'AC'),
(6, 'Manaus', 'AM'),
(7, 'Porto Alegre', 'RS'),
(8, 'Fortaleza', 'CE'),
(9, 'Recife', 'PE'),
(10, 'Teresina', 'PI'),
(11, 'Belém', 'PA'),
(12, 'Palmas', 'TO');

-- Clientes
INSERT INTO Cliente (idCliente, CPF, RG, nomeCompleto, endereco) VALUES
(1, '1111', '11111', 'Carlos Silva', 'Rua A 100'),
(2, '2221', '222211', 'Ana Santos', 'Rua B 200'),
(3, '3331', '333311', 'José Pereira', 'Rua C 300'),
(4, '4441', '444411', 'José Fidalgo', 'Rua D 400'),
(5, '5551', '555511', 'Bruno Marmelo', 'Rua D 400'),
(6, '6661', '666611', 'João Pinóquio', 'Rua E 700'),
(7, '7771', '777711', 'Felipe Constâncio', 'Rua F 345'),
(8, '8881', '888811', 'Paulo Papudo de Sá', 'Rua G 34567'),
(9, '9991', '999911', 'Maria Clara Pravoika', 'Rua H 192'),
(10, '1010101', '1010101011', 'Joana Medeiros de Castro', 'Rua I 230'),
(11, '1111111', '1111111111', 'Sonia Ruth Campos', 'Rua J 890'),
(12, '1212121', '1212121211', 'Daniela Silva Chaffer', 'Rua K 900');


-- Funcionários
INSERT INTO Funcionario (CPFfunc, enderecoFunc, RGfunc, salario, tipoFunc,
telefoneCont, nomeCompletoFunc) VALUES
('11112', 'Rua D 400', 'FUNC123', 3000.00, 'Motorista', '1111-1111', 'Pedro Souza'),
('22222', 'Rua E 500', 'FUNC456', 3200.00, 'Operador', '2222-2222', 'Maria Oliveira'),
('33332', 'Rua F 600', 'FUNC789', 2800.00, 'Guindasteiro', '3333-3333', 'João Costa'),
('44442', 'Rua X 123', 'FUNC001', 3200.00, 'Operador', '4444-4444', 'Maria Costa Pereira de
Albuquerque Flamino Rocha Loures'),
('55552', 'Rua Y 500', 'FUNC002', 3200.00, 'Motorista', '2222-5555', 'William Jorge
Menezes'),
('66662', 'Rua Z 505', 'FUNC003', 3200.00, 'Montador', '2222-6666', 'Regis Balaclava'),
('77772', 'Rua Alpha 404', 'FUNC004', 3200.00, 'Montador', '2222-7777', 'Marcio Nunes'),
('88882', 'Rua Beta 403', 'FUNC005', 3200.00, 'Operador', '2222-8888', 'Sebastião
Mantovani Ribeiro'),
('99992', 'Rua Gamma 679', 'FUNC006', 3200.00, 'Guindasteiro', '2222-9999', 'Jotacir
Homero'),
```

('101010102', 'Rua Delta 89', 'FUNC007', 3200.00, 'Motorista', '3333-2222', 'Mara Silva'),
('111111112', 'Rua Ômega 2245', 'FUNC008', 3200.00, 'Motorista', '3333-2222', 'Yuri Marcial'),
('121212122', 'Rua Tetha 424', 'FUNC009', 3200.00, 'Operador', '3333-2222', 'Antônio Vasco');

-- Serviços
INSERT INTO Servico (idServico, nomeServ, servTipo) VALUES
(1, 'Transporte Rodoviário', 'Transporte'),
(2, 'Guindaste Médio', 'Guindaste'),
(3, 'Transporte de Cargas Leves', 'Transporte'),
(4, 'Transporte de Líquidos Corrosivos', 'Transporte'),
(5, 'Guindaste Predial', 'Guindaste'),
(6, 'Guindaste Containers', 'Guindaste'),
(7, 'Transporte de Cargas Vivas', 'Transporte'),
(8, 'Transporte de Especiais', 'Transporte');

-- Serviço de Guindaste
INSERT INTO serv_guindaste (idServico, altura, tamanhoBase, percentualBonus, aumentoBonus) VALUES
(2, 30, 5, 0.2, 100),
(5, 100, 20, 0.3, 2000),
(6, 100, 50, 0.4, 3000);

-- Serviço de Transporte
INSERT INTO serv_transporte (idServico, percentualAcres, limiteCarga) VALUES
(1, 0.1, 500),
(3, 0.1, 250),
(4, 0.5, 50),
(7, 0.6, 500),
(8, 0.3, 200);

-- Pedidos
INSERT INTO Pedido (idPedido, idCliente, idEmpresa, idCidade, enderecoDest, enderecoPart) VALUES
(1, 1, 1, 1, 'Rua Z 900', 'Rua X 800'),
(2, 1, 2, 2, 'Avenida W 700', 'Avenida V 600'),
(3, 1, 3, 3, 'Rua A 500', 'Rua U 400'),
(4, 3, 12, 12, 'Avenida W 700', 'Avenida V 600'),
(5, 2, 9, 3, 'Rua Y 500', 'Rua U 400'),
(6, 11, 9, 11, 'Rua Z 900', 'Rua X 800'),
(7, 10, 9, 12, 'Avenida W 700', 'Avenida V 600'),
(8, 3, 12, 8, 'Rua Y 500', 'Rua U 400'),
(9, 5, 10, 8, 'Rua Z 900', 'Rua X 800'),
(10, 6, 5, 8, 'Avenida W 700', 'Avenida V 600'),
(11, 7, 6, 2, 'Rua Y 500', 'Rua U 400'),
(12, 10, 6, 2, 'Rua Z 900', 'Rua X 800'),
(13, 9, 3, 6, 'Avenida W 700', 'Avenida V 600'),

```
(14, 12, 3, 6, 'Rua Y 500', 'Rua U 400'),
(15, 5, 9, 11, 'Rua Z 900', 'Rua X 800'),
(16, 5, 1, 1, 'Avenida W 700', 'Avenida V 600'),
(17, 3, 4, 9, 'Rua Y 500', 'Rua U 400'),
(18, 2, 6, 2, 'Rua Z 900', 'Rua X 800'),
(19, 10, 10, 8, 'Avenida W 700', 'Avenida V 600'),
(20, 4, 2, 2, 'Rua Y 500', 'Rua U 400'),
(21, 4, 1, 1, 'Rua Z 900', 'Rua X 800'),
(22, 8, 1, 1, 'Avenida W 700', 'Avenida V 600'),
(23, 9, 3, 6, 'Rua Y 500', 'Rua U 400'),
(24, 9, 6, 2, 'Rua Z 900', 'Rua X 800'),
(25, 2, 1, 1, 'Avenida W 700', 'Avenida V 600'),
(26, 3, 3, 6, 'Rua Y 500', 'Rua U 400');

-- Solicitações
INSERT INTO Solicitam (idServico, idPedido, CPFfunc, tempoDurac, preco, dataFim)
VALUES
(1, 1, '11112', 5.0, 1500.00, '2024-01-10'),
(2, 2, '44442', 4.5, 1800.00, '2023-01-15'),
(4, 3, '44442', 6.0, 1600.00, '2023-02-20'),
(4, 4, '22222', 5.0, 1500.00, '2022-02-10'),
(4, 5, '77772', 4.5, 1800.00, '2020-03-15'),
(2, 6, '88882', 6.0, 1600.00, '2022-03-20'),
(7, 7, '99992', 5.0, 1500.00, '2017-04-10'),
(7, 8, '55552', 4.5, 1800.00, '2015-04-15'),
(7, 9, '66662', 6.0, 1600.00, '2023-05-20'),
(2, 10, '11112', 5.0, 1500.00, '2010-05-10'),
(3, 11, '121212122', 4.5, 1800.00, '2022-06-15'),
(3, 12, '66662', 6.0, 1600.00, '2018-06-20'),
(8, 13, '66662', 5.0, 1500.00, '2018-07-10'),
(8, 14, '99992', 4.5, 1800.00, '2018-07-15'),
(2, 15, '55552', 6.0, 1600.00, '2019-08-20'),
(1, 16, '22222', 5.0, 1500.00, '2019-08-10'),
(5, 17, '88882', 4.5, 1800.00, '2015-09-15'),
(3, 18, '111111112', 6.0, 1600.00, '2015-09-20'),
(7, 19, '33332', 5.0, 1500.00, '2015-10-10'),
(2, 20, '11112', 4.5, 1800.00, '2016-10-15'),
(1, 21, '44442', 6.0, 1600.00, '2024-11-20'),
(1, 22, '66662', 5.0, 1500.00, '2024-11-10'),
(8, 23, '77772', 4.5, 1800.00, '2023-12-15'),
(3, 24, '44442', 6.0, 1600.00, '2023-12-20'),
(1, 25, '77772', 5.0, 1500.00, '2022-11-10'),
(8, 26, '99992', 4.5, 1800.00, '2020-10-15');

-- Telefones de Empresas
INSERT INTO TelefonesEmpresa (idEmpresa, numTelefone) VALUES
(1, '4444-4444'),
(1, '4444-1111'),
```

```
(2, '5555-5555'),
(2, '5555-1111'),
(3, '6666-6666'),
(3, '6666-1212'),
(4, '6666-1111'),
(5, '6666-2222'),
(6, '6666-3333'),
(7, '6666-4444'),
(8, '6666-5555'),
(9, '6666-7777'),
(10, '6666-8888'),
(11, '6666-9999'),
(12, '6666-0000');

-- Telefones de Clientes
INSERT INTO TelefonesCliente (idCliente, numTelefone) VALUES
(1, '7777-7777'),
(1, '7777-1235'),
(2, '8888-1010'),
(2, '8888-8888'),
(3, '9999-9999'),
(4, '7777-1111'),
(5, '8888-2222'),
(6, '9999-3333'),
(7, '7777-4444'),
(8, '8888-5555'),
(9, '9999-6666'),
(10, '7777-9999'),
(11, '8888-0000'),
(12, '9999-0001');

-- Serviços Oferecidos pelas empresas em determinadas cidades e seu valor hora
INSERT INTO Oferecem (idEmpresa, idCidade, idServico, precoHoraSe) VALUES
(1, 1, 1, 300.00),
(2, 2, 2, 400.00),
(3, 3, 3, 350.00),
(4, 3, 3, 890.00),
(4, 12, 6, 300.00),
(5, 12, 6, 400.00),
(5, 3, 6, 350.00),
(10, 8, 7, 300.00),
(11, 4, 2, 400.00),
(12, 3, 3, 350.00),
(2, 10, 1, 300.00),
(2, 12, 2, 400.00),
(7, 3, 5, 350.00),
(3, 5, 2, 350.00),
(8, 2, 7, 400.00),
```

(12, 11, 3, 300.00),
(7, 1, 4, 200.00),
(2, 12, 5, 250.00),
(6, 4, 8, 400.00),
(10, 9, 6, 300.00),
(5, 7, 1, 300.00),
(4, 3, 2, 250.00),
(9, 6, 7, 350.00),
(3, 2, 3, 500.00),
(6, 2, 1, 700.00),
(8, 1, 8, 780.00),
(8, 1, 7, 800.00),
(8, 1, 1, 859.00),
(2, 2, 3, 950.00),
(4, 7, 2, 869.00),
(2, 7, 5, 500.00),
(5, 8, 2, 700.00),
(9, 3, 4, 800.00),
(9, 11, 2, 700.00),
(12, 8, 7, 900.00),
(10, 8, 2, 800.00),
(4, 9, 5, 800.00),
(4, 9, 1, 950.00),
(1, 10, 3, 840.00),
(12, 12, 4, 600.00),
(9, 12, 7, 700.00),
(6, 2, 3, 800.00),
(3, 6, 8, 900.00),
(7, 6, 3, 890.00)

CONSULTAS:

Enunciado: 1. Que tipo de serviços um determinado cliente X solicitou no último mês.

```
SELECT se.servTipo
FROM servico as se, solicitam as s, pedido as p, cliente as c
WHERE s.idPedido = p.idPedido
        AND se.idServico = s.idServico
        AND p.idCliente = c.idCliente
        AND c.idCLiente = X
        AND MONTH(s.dataFim) = (MONTH(CURDATE()) - 1)
        AND YEAR(s.dataFim) = YEAR(CURDATE());
```

Enunciado: 2. Qual é a empresa que mais ofereceu mais serviços à cidade de Y no estado de Z

```
SELECT emp.nome
FROM empresa as emp, oferecem as ofe, cidade as cid
WHERE emp.idEmpresa = ofe.idEmpresa
        AND cid.idCidade = ofe.idCidade
        AND cid.nomeCidade = 'Y'
        AND cid.estado = 'Z'
GROUP BY emp.nome
ORDER BY COUNT(ofe.idServico) DESC
LIMIT 1;
```

Enunciado: 3. Quais funcionários (nome e sobrenome) trabalharam para o cliente X no mês Y do ano Z.

```
SELECT DISTINCT func.nomeCompletoFunc
FROM funcionario as func, solicitam as sol, pedido as ped,  cliente as cli
WHERE func.CPFfunc = sol.CPFfunc
        AND ped.idPedido = sol.idPedido
        AND cli.idCLiente = ped.idCliente
        AND cli.idCLiente = X
        AND MONTH(sol.dataFim) = Y
        AND YEAR(sol.dataFim) = Z;
```

Enunciado: 4. Listar as solicitações foram feitas no último ano, nome do cliente que as realizou, municípios de origem e destino (se houver) e preço total de cada solicitação.

```
SELECT ped.idPedido, cli.nomeCompleto, ped.enderecoPart, ped.enderecoDest, sol.preco
FROM pedido as ped, cliente as cli, solicitam as sol
WHERE sol.idPedido = ped.idPedido
        AND ped.idCliente = cli.idCliente
        AND YEAR(sol.dataFim) = YEAR(CURDATE()) - 1;
```

Enunciado: 5. Listar o faturamento das empresas por mês em um ano X.

```
SELECT emp.nome, MONTH(sol.dataFim) as mes, SUM(sol.preco)
FROM empresa as emp, solicitam sol, pedido as ped
WHERE sol.idPedido = ped.idPedido
```

```
        AND emp.idEmpresa = ped.idEmpresa
        AND YEAR(sol.dataFim) = X
GROUP BY emp.nome, MONTH(sol.dataFim)
ORDER BY emp.nome;
```

Enunciado: 6. Verificar qual o serviço mais solicitado no último mês entre todas empresas

```
SELECT serv.servTipo, COUNT(serv.servTipo) AS totalSolicitacoes
FROM servico as serv, solicitam as sol
WHERE serv.idServico = sol.idServico
          AND MONTH(sol.dataFim) = MONTH(CURDATE()) - 1
          AND YEAR(sol.dataFim) = YEAR(CURDATE())
GROUP BY serv.servTipo
ORDER BY totalSolicitacoes DESC
LIMIT 1;
```

Enunciado: 7. Listar o serviço (nome) mais solicitado, e o número de solicitações para cada empresa

```
WITH ranked_services AS (
   SELECT emp.nome, se.nomeServ, COUNT(sol.idServico) AS numSolicitacoes,
                ROW_NUMBER() OVER (PARTITION BY emp.idEmpresa ORDER BY
COUNT(sol.idServico) DESC) AS service_rank
   FROM empresa AS emp
   JOIN pedido AS ped ON emp.idEmpresa = ped.idEmpresa
   JOIN solicitam AS sol ON ped.idPedido = sol.idPedido
   JOIN servico AS se ON se.idServico = sol.idServico
   GROUP BY emp.nome, se.nomeServ, emp.idEmpresa
)
SELECT nome, nomeServ, numSolicitacoes
FROM ranked_services
WHERE service_rank = 1
ORDER BY nome;
```

Enunciado: 8. Verificar em qual a cidade houve o maior número de solicitações.

```
SELECT cid.nomeCidade, COUNT(ped.idPedido) as numSolicitacoes
FROM cidade as cid, pedido as ped, solicitam as sol
WHERE ped.idCidade = cid.idCidade AND sol.idPedido = ped.idPedido
GROUP BY cid.nomeCidade
ORDER BY numSolicitacoes DESC
```

```
LIMIT 1;
```

Enunciado: 9. Verificar qual a cidade destino que é mais referenciada nos pedidos e a sua quantidade de pedidos.

```
SELECT cid.nomeCidade, COUNT(ped.idPedido) as quantPedidos
FROM cidade as cid, pedido as ped
WHERE cid.idCidade = ped.idCidade
GROUP BY cid.idCidade, cid.nomeCidade
ORDER by quantPedidos DESC
LIMIT 1;
```

Enunciado: 10. Listar para cada empresa o seu faturamento total.

```
SELECT emp.nome, SUM(sol.preco) as faturamento
FROM solicitam as sol, empresa as emp, pedido as ped
WHERE sol.idPedido = ped.idPedido AND ped.idEmpresa = emp.idEmpresa
GROUP BY emp.nome
ORDER BY faturamento ASC;
```

TRIGGERS:

```
DELIMITER $$

CREATE OR REPLACE TRIGGER trigger_preco
AFTER UPDATE ON Oferecem
FOR EACH ROW
BEGIN
  -- Atualiza o campo preco na tabela Solicitam
  UPDATE Solicitam s
  SET s.preco = NEW.precoHoraSe * s.tempoDurac
  WHERE EXISTS (
    SELECT 1
    FROM Pedido p
    WHERE p.idPedido = s.idPedido
      AND p.idEmpresa = NEW.idEmpresa
      AND p.idCidade = NEW.idCidade
  );
END $$
DELIMITER ;
```

```
DELIMITER $$

CREATE TRIGGER trigger_atualizar_preco
BEFORE INSERT OR UPDATE ON Solicitam
FOR EACH ROW
BEGIN
   DECLARE preco_hora DECIMAL(10, 2);

   -- Buscar o PrecoHoraSe na tabela Oferecem
   SELECT precoHoraSe INTO preco_hora
   FROM Oferecem
       WHERE idEmpresa = (SELECT idEmpresa FROM Pedido WHERE idPedido = NEW.idPedido)
       AND idServico = NEW.idServico
    AND idCidade = (SELECT idCidade FROM Pedido WHERE idPedido = NEW.idPedido);

   -- Calcular o preço do serviço solicitado
   SET NEW.preco = preco_hora * NEW.tempoDurac;
END$$

DELIMITER ;
```

COMO EXECUTAR A APLICAÇÃO

Abordaremos a forma como nós criamos a aplicação que pode ser interessante para executar o mais próximo possível de como nós executamos. Utilizamos a IDE Eclipse, basta abrir uma workspace com a pasta do projeto. Ao acessá-lo deve se atentar a classe “Conexao” responsável por fornecer uma conexão com o banco de dados.

```
private static final String url = "jdbc:mysql://localhost:3306/ep3";
private static final String user = "root";
private static final String password = "b234";
```

Utilizamos um driver jdbc para o MySQL, sendo o próprio PC o local que vai estar o banco de dados. O “user” e “password” corresponde a respectivamente o usuário e sua senha no SGBD MySQL, fundamental para poder acessar o banco de dados. Antes de rodar é necessário se certificar da criação do banco de dados e das tabelas, assim como dos inserts especificados acima do relatório. Após criar o banco de dados, digite “USE nomebancodedados” nesse caso o nosso se chama ep3, isso é importante para setar a utilização do banco. Em seguida, execute a classe App com a classe main que vai

rodar as interfaces. A partir disso basta selecionar algumas das 10 opções (consultas) na estrutura do JComboBox da biblioteca Java Swing e colocar valores caso seja requerido e apertar o botão “Consultar”.

CONSIDERAÇÕES

A principal consideração é o fato de no enunciado não ter sido requisitado a normalização do banco, implicando em alguns conflitos com os enunciados de certas consultas.

Para evitarmos deixar a query longa evitamos ficar nomeando atributos sem nome próprio. Optamos por fazer isso com apenas alguns atributos. Isso acabou sendo tratado na interface da aplicação.

Consulta 3: De acordo com o modelo fornecido pelo professor, não apresentava a primeira forma normal, então consideramos o nomeCompleto como nome e sobrenome.

Consulta 4: Não havia a existência de municípios de origem e destino, então tomamos como origem o endereço de partida e o destino como endereço de destino da relação Pedido.

Consulta 7: Usamos algumas cláusulas que não foram ensinadas em aula, estávamos com dificuldade em fazer essa consulta, pesquisamos algumas coisas e achamos isso. Decidimos colocar uma explicação mais técnica aqui. O código utiliza uma subconsulta nomeada (WITH ranked_services) para calcular informações sobre os serviços mais solicitados por cada empresa. Dentro dessa subconsulta, são feitas várias junções entre tabelas (empresa, pedido, solicitam e servico) para associar empresas aos serviços solicitados e contar o número de solicitações por serviço utilizando COUNT. A função de janela ROW_NUMBER() é usada com a cláusula PARTITION BY emp.idEmpresa para reiniciar a contagem de classificação por empresa e organizar os serviços dentro de cada empresa em ordem decrescente de solicitações. A consulta externa filtra apenas o serviço mais solicitado por cada empresa (service_rank = 1) e exibe os resultados ordenados pelo nome da empresa (ORDER BY nome). Isso permite identificar os serviços mais populares para cada empresa em uma única operação SQL eficiente.

Consulta 9: Como não há essa informação no esquema, assumimos que a cidade referenciada pelo idCidade no pedido é também o destino.

Trigger: Devido a falta de informações acerca disso, decidimos ignorar a parte de poder aumentar o preço do serviço em virtude de algumas situações específicas de cada serviço, pois seria necessário uma análise para isso.

Link do Google Drive:
https://drive.google.com/drive/folders/1cZdxnYo2XddtnwiDU5vZNCvko0bddIrB?usp=sharing