**BANCO DE DADOS II**

**LISTA 2 - RESOLUÇÃO**

**Aluno:** Vitor de Azambuja Ribeiro Franco

**R.A:** 5153344

**1-a)** Em Tabela Única por Tipo de Classes, todas as classes na hierarquia se encontram em uma única tabela, incluindo todos os atributos de todas as classes. Sendo especificados somente por um campo “tipo” na tabela.

Exemplo:

Tabela

Descrição gerada automaticamente

**b)** Em Tabela por Subclasse, a hierarquia é dada em uma tabela separada e provendo o ID para ser usado como chave estrangeira nas tabelas de subclasse. Desse modo, as tabelas de subclasses contêm os atributos específicos de cada tipo.

Exemplo:

- Tabela Produto:

Tabela

Descrição gerada automaticamente

- Tabela Roupa:

Tabela

Descrição gerada automaticamente

- Tabela Eletrônico:



**c)** Em Tabela por Classe Concreta, a hierarquia é dada em uma tabela separada e as tabelas de subclasse tem os atributos da classe pai e os específicos de sua própria classe.

Exemplo:

- Tabela Produto:

Tabela

Descrição gerada automaticamente

- Tabela Roupa:

Tabela

Descrição gerada automaticamente

- Tabela Eletrônico:



**2-**

V

F

F

Letra A

**3-a)**

Performance, automação (reutilização de código).

**b)**

Manutenção, dependente da sintaxe do Banco de Dados.

**4-** O papel da mineração de dados é tratar e organizar os dados de uma forma que sejam úteis para prever um comportamento ou um conjunto de comportamentos e dessa forma, aplicar em estratégias de marketing e semelhantes.

**5-** A mineração de dados no processo KDD é um processo crucial após a transformação de dados para a descoberta de padrões ocultos que resultará na obtenção de conhecimento após uma avaliação.

**6-a)** Sup(Alimentação) = 3

**b)** Sup(Impostos Diversos) = 3

**c)** Sup({Alimentação} → {Impostos Diversos}) = 2

**d)** Conf({Alimentação} → {Impostos Diversos}) = 2/3 = 66,66%

**e)** Conf({Impostos Diversos} → {Alimentação}) = 2/3 = 66,66%

**f)** Conf({Matéria Prima} → {Impostos Diversos}) = 0/3 = 0%

**g)** Sim, com as regras de associação pode-se descobrir comportamentos e causas para um determinado gasto como por exemplo, impostos que podem aplicar-se diretamente à vales alimentação e desse modo estruturar um plano para a redução de tais gastos.

**7-** Um Data Mart é um repositório de dados menores e descentralizados, sendo assim uma parte de uma Data Warehouse completa. Data Marts focam em apenas uma área de interesse, podendo ser montado mais rápido e com menos custos que uma Data Warehouse.

**8-** Algumas características dos Bancos de Dados Transacionais (para OLTP) são: possuem poucos registros acessados de cada vez, possibilidade de consulta e atualização, o tamanho do banco de dados pode variar de 100MB até 100GB, suporta milhares de usuários e não possui redundância de dados. Por outro lado, tem-se as características das Data Warehouses (para OLAP): Volume alto de acessos por vez, possibilidade basicamente de consultas, tamanhos que variam de 100GB até poucos terabytes, suporta centenas de usuários e dados redundantes podem estar presentes.

**9-**

CREATE TABLE telefone(

id\_numero int not null PRIMARY KEY,

numero varchar(15) not null

);

INSERT INTO telefone

VALUES

(1,'99587412'),

(2,'98632574'),

(3,'97412587');

select \* from telefone;

GO

CREATE FUNCTION insere\_prefixo(@id\_numero INT)

RETURNS VARCHAR(15)

AS

BEGIN

DECLARE @numero\_prefix VARCHAR(15)

SELECT @numero\_prefix = '055'+numero

FROM telefone

WHERE @id\_numero = id\_numero

RETURN @numero\_prefix

END

GO

SELECT dbo.insere\_prefixo(1) AS 'Número com prefixo';

Interface gráfica do usuário, Tabela

Descrição gerada automaticamente

**10-**

GO

create function ajuste\_precos (@valor decimal(10,2))

returns decimal(10,2)

begin

return abs(@valor \* 1.05)

end

GO

select dbo.ajuste\_precos(-100) as 'Ajuste';



**11-**

GO

create view [VI\_RANDOMVALOR]

as

select titulo,round(rand()\*1000,2) AS preco\_random from produto

GO

GO

create function preco\_random (@valor decimal(10,2))

returns decimal(10,2)

begin

return rand() \* 1000

end

GO

select \* from [VI\_RANDOMVALOR]

Tabela

Descrição gerada automaticamente

**12-**

GO

CREATE FUNCTION concat\_funcao(@id\_cliente int, @id\_produto int)

RETURNS VARCHAR(125)

BEGIN

DECLARE @nome\_cliente VARCHAR(125);

DECLARE @titulo\_produto VARCHAR(125);

SET @nome\_cliente = (SELECT nome FROM cliente WHERE id = @id\_cliente);

SET @titulo\_produto = (SELECT titulo FROM produto WHERE id = @id\_produto);

RETURN CONCAT(@nome\_cliente, ' - ', @titulo\_produto);

END

GO

Interface gráfica do usuário, Tabela

Descrição gerada automaticamente

**13-**

GO

CREATE FUNCTION concat\_funcao(@id\_cliente int, @id\_produto int)

RETURNS VARCHAR(125)

BEGIN

DECLARE @nome\_cliente VARCHAR(125);

DECLARE @titulo\_produto VARCHAR(125);

SET @nome\_cliente = (SELECT nome FROM cliente WHERE id = @id\_cliente);

SET @titulo\_produto = (SELECT titulo FROM produto WHERE id = @id\_produto);

RETURN CONCAT(@nome\_cliente, ' - ', @titulo\_produto);

END

GO

SELECT dbo.concat\_funcao(2,1) AS 'concat\_funcao';



**14-**

SELECT FORMAT(CURRENT\_TIMESTAMP, 'yyyy-MM-dd hh:mm:ss tt') AS data\_corrente;



**15-**

GO

CREATE FUNCTION verificar\_tabela (@nomeTabela VARCHAR(128))

RETURNS VARCHAR(10)

AS

BEGIN

DECLARE @existe VARCHAR(10);

IF EXISTS (

SELECT 1

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_NAME = @nomeTabela

)

BEGIN

SET @existe = ('Existe');

END

ELSE

BEGIN

SET @existe = ('Não Existe');

END

RETURN @existe;

END;

GO

SELECT dbo.verificar\_tabela('cliente') AS 'verificar';

SELECT dbo.verificar\_tabela('teste') AS 'verificar';

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

**16-a)**

GO

CREATE PROCEDURE busca\_produto\_cliente(@id\_cliente int)

AS

BEGIN

SELECT p.titulo

FROM produto p

INNER JOIN item\_pedido itp ON itp.id\_produto = p.id

INNER JOIN pedido pe ON pe.id = itp.id\_pedido

INNER JOIN cliente c ON c.id = pe.id\_cliente AND c.id = @id\_cliente;

END

GO

EXECUTE dbo.busca\_produto\_cliente @id\_cliente = 1;

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**b)**

GO

CREATE PROCEDURE busca\_produtos\_nao\_vendidos(@id\_produto INT, @preco\_minimo DECIMAL(10, 2))

AS

BEGIN

SELECT \*

FROM produto p

WHERE p.id IN (SELECT id\_produto FROM item\_pedido)

AND (@id\_produto IS NOT NULL AND p.id = @id\_produto)

AND (@preco\_minimo IS NOT NULL AND p.preco >= @preco\_minimo);

END

GO

EXECUTE busca\_produtos\_nao\_vendidos @id\_produto = 1, @preco\_minimo = 500;

Interface gráfica do usuário

Descrição gerada automaticamente

**17-a-b)**

CREATE TABLE canteiro(

canteiroId int PRIMARY KEY,

nome varchar(20),

luzdiaria decimal(4,3),

aguadiaria decimal(4,3)

);

CREATE TABLE funcionario(

funcId int PRIMARY KEY,

nome varchar(20),

idade int

);

CREATE TABLE planta(

id int PRIMARY KEY,

nome varchar(20),

luzdiaria decimal(4,3),

agua decimal(4,3),

peso decimal(4,3)

);

CREATE TABLE colhido(

colhidoId int PRIMARY KEY,

plantaId int,

funcId int,

canteiroId int,

datacolheita date,

quantidade int,

peso decimal(4,3),

FOREIGN KEY (plantaId) REFERENCES planta(id),

FOREIGN KEY (canteiroId) REFERENCES canteiro(canteiroId),

FOREIGN KEY (funcId) REFERENCES funcionario(funcId)

);

CREATE TABLE plantio(

plantioId int PRIMARY KEY,

plantaId int,

funcId int,

canteiroId int,

dataplantio date,

sementes int,

FOREIGN KEY (plantaId) REFERENCES planta(id),

FOREIGN KEY (funcId) REFERENCES funcionario(funcId),

FOREIGN KEY (canteiroId) REFERENCES canteiro(canteiroId)

);

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**c)**

GO

CREATE PROCEDURE buscar\_colheitas(@funcionarioId INT)

AS

BEGIN

SELECT c.colhidoId, p.nome AS planta, c.datacolheita, c.quantidade, c.peso

FROM colhido c

INNER JOIN planta p ON p.id = c.plantaId

WHERE c.funcId = @funcionarioId;

END

GO

EXECUTE buscar\_colheitas @funcionarioID = 1

Tabela

Descrição gerada automaticamente com confiança média