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:

ID	Tipo	Nome	Preço	Marca	Cor	Categoria
1	Roupa	Camiseta	40	Nike	Preto	
2	Roupa	Meia	10	Lupo	Branco	
3	Eletrônico	DVD	80	Sony		Imagem
4	Roupa	Calça	60	Levi's	Marrom	

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:

ID	Nome	Preço	Marca
1	Camiseta	40	Nike
2	Meia	10	Lupo
3	DVD	80	Sony
4	Calça	60	Levi's

- Tabela Roupa:

ID	Cor
1	Preto
2	Branco
4	Marrom

- Tabela Eletrônico:

ID	Categoria
3	Imagem

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:

ID	Nome	Preço	Marca	Cor
1	Camiseta	40	Nike	Preto
2	Meia	10	Lupo	Branco
3	DVD	80	Sony	Preto
4	Calça	60	Levi's	Marrom

- Tabela Roupa:

ID	Nome	Preço	Marca	Cor
1	Camiseta	40	Nike	Preto
2	Meia	10	Lupo	Branco
4	Calça	60	Levi's	Marrom

- Tabela Eletrônico:

ID	Nome	Preço	Marca	Categoria
3	Imagem	80	Sony	Imagem

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.

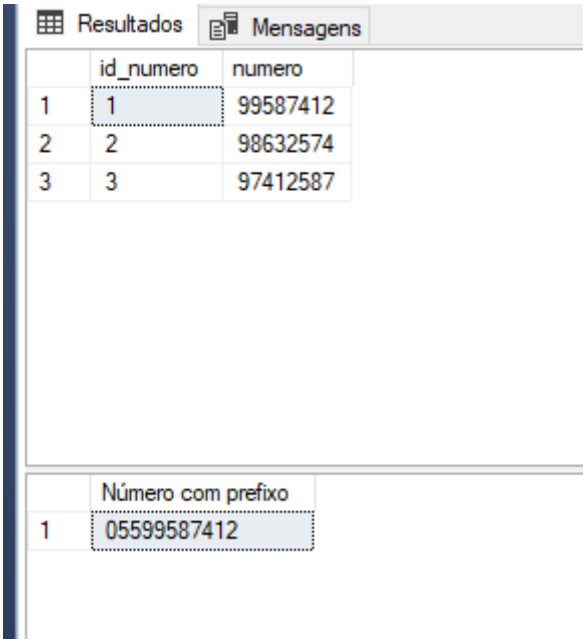
7- Um Data Mart é um repositório de dados menores e descentralizados, sendo assim uma parte de uma Data Warehouse completa. Data Marts focam em apenas uma área de interesse, podendo ser montado mais rápido e com menos custos que uma Data Warehouse.

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';
```



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]
```

	titulo	preco_random
1	Samsung Galaxy S7	996,46
2	Xperia ZQ	996,46
3	Samsung Galaxy S6	996,46

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
```

	id	titulo	preco_arredondado
1	1	Samsung Galaxy S7	2798.99
2	2	Xperia ZQ	999.99
3	3	Samsung Galaxy S6	1761.99

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';
```

	concat_funcao
1	Maria Antonieta - Samsung Galaxy S7

14-

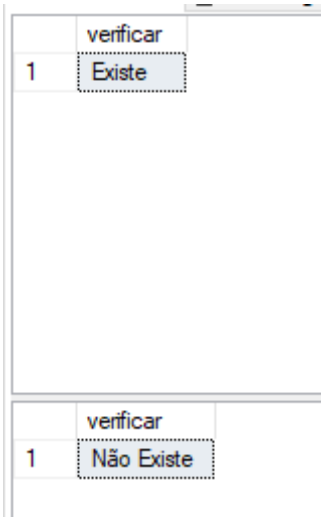
```
SELECT FORMAT(CURRENT_TIMESTAMP, 'yyyy-MM-dd hh:mm:ss tt') AS data_corrente;
```

	data_corrente
1	2023-05-16 10:06:15

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';
```



	verificar
1	Existe

	verificar
1	Não Existe

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;
```

	titulo
1	Samsung Galaxy S7
2	Xperia ZQ
3	Xperia ZQ

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)
```

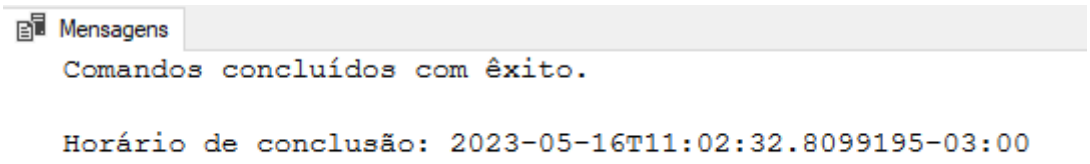
```
   AND (@id_produto IS NOT NULL AND p.id = @id_produto)
   AND (@preco_minimo IS NOT NULL AND p.preco >=
@preco_minimo);
END
GO

EXECUTE busca_produtos_nao_vendidos @id_produto = 1,
@preco_minimo = 500;
```

	id	titulo	descricao	preco	nome_imagem
1	1	Samsung Galaxy S7	Câmera Dual Pixel 12 MP. Resistência à água e po...	2799.00	4512

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)
);
```



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
```

	colhidoId	planta	datacolheita	quantidade	peso
1	1	Planta 1	2023-05-15	10	0.500
2	3	Planta 3	2023-05-13	12	0.600