



Estácio

Universidade Estácio de Sá

- DESENVOLVIMENTO FULL STACK- TURMA 23.3 -9003
 - Disciplina: RPG0015 - Vamos Manter as Informações?
 - Semestre Letivo: 2023.2
 - Repositorio Git: <https://github.com/Gregdev22/Missao-2-Mundo-3>
-

- [EMERSON GREGORIO ALVES](#) - MATRICULA: 2022.0908.4986
-

Missão Prática | Nível 2 | Mundo 3

Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

Procedimento 1: Criando o Banco de Dados

Procedimento 2: Alimentando a Base



Objetivos da Prática

- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.

- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
 - Explorar a sintaxe SQL na consulta e manipulação de dados (DML).
 - No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server
-

Códigos

[Procedimento 1: Criando o Banco de Dados](#)

```
create database Loja;
```

```
use Loja;
```

```
create table pessoa(  
    idpessoa int NOT NULL ,  
    nome varchar(255) NOT NULL ,  
    logradouro varchar(255) NOT NULL ,  
    cidade varchar(255) NOT NULL ,  
    estado char(2) NOT NULL ,  
    telefone varchar(11) NOT NULL ,  
    email varchar(255) NOT NULL ,  
    primary key(idpessoa));
```

```
create table pessoa_fisica (  
    idpessoa int NOT NULL,  
    cpf varchar(255) NOT NULL,  
    primary key (idpessoa),  
    foreign key (idpessoa) references pessoa(idpessoa));
```

```
create table pessoa_juridica (  
    idpessoa int NOT NULL,  
    cnpj varchar(255) NOT NULL,  
    primary key (idpessoa),  
    foreign key (idpessoa) references pessoa(idpessoa));
```

```
create table produto (  
    idproduto int NOT NULL ,  
    nome varchar(255) NOT NULL ,  
    quantidade varchar(255) NOT NULL ,  
    preco_venda numeric(5,2) NOT NULL ,  
    primary key(idproduto));
```

```
create table usuario (  
    idusuario int NOT NULL ,
```

```

    login varchar(255)    NOT NULL    ,
    senha varchar(255)    NOT NULL    ,
primary key(idusuario));

create table movimento (
    idmovimento int NOT NULL,
    Usuario_idUsuario int  NOT NULL    ,
    pessoa_idpessoa int  NOT NULL    ,
    produto_idproduto int  NOT NULL    ,
    quantidade int  NOT NULL    ,
    tipo char  NOT NULL    ,
    valorUnitario numeric(5,2)  NOT NULL    ,
primary key(idmovimento),
foreign key (Usuario_idUsuario) references usuario(idusuario),
foreign key (produto_idproduto) references produto(idproduto),
foreign key (pessoa_idpessoa) references pessoa(idpessoa));

create sequence seq_Pessoa
    as numeric
    start with 1
    increment by 1
    no cycle;

```

Procedimento 2: Alimentando a Base

```
use Loja;
```

```

insert into usuario
values (1, 'op1', 'op1'),(2, 'op2', 'op2');

```

```

insert into produto
values (1, 'Banana', 100, 5.00),(3, 'Laranja', 500, 2.00),(4,
'Manga', 800, 4.00);

```

```

insert into pessoa
values (NEXT VALUE FOR seq_Pessoa, 'Joao', 'Rua 12, cas 3,
Quitanda',
'Riacho do Sul', 'PA', '1111-1111','joao@riacho.com');

```

```

insert into pessoa
values (NEXT VALUE FOR seq_Pessoa, 'JJC', 'Rua 11, Centro',
'Riacho do Norte', 'PA', '1212-1212','jjc@riacho.com');

```

```

insert into pessoa_fisica
values (1,'11111111111');

```

```

insert into pessoa_juridica
values (2,'2222222222222');

```

```
insert into movimento
```

```

values (1,1,1,1,20,'S',4.00),
(4,1,1,3,15,'S',2.00),
(5,2,1,3,10,'S',3.00),
(7,1,2,3,15,'E',5),
(8,1,2,4,20,'E',4.00);

-- Dados completos de pessoas físicas.
select *
from pessoa, pessoa_fisica
where pessoa.idpessoa = pessoa_fisica.idpessoa;

--Dados completos de pessoas jurídicas.
select *
from pessoa, pessoa_juridica
where pessoa.idpessoa = pessoa_juridica.idpessoa;

--Movimentações de entrada, com produto, fornecedor, quantidade,
preço unitário e valor total.
select idmovimento, produto_idproduto, produto.nome as
'Produto',pessoa_idpessoa, pessoa.nome as 'Fornecedor',
movimento.quantidade, valorUnitario,
(movimento.quantidade * valorUnitario) as valor_total
from movimento
join pessoa
on movimento.pessoa_idpessoa = pessoa.idpessoa
join produto
on movimento.produto_idproduto = produto.idproduto
where movimento.tipo = 'E';

--Movimentações de saída, com produto, comprador, quantidade,
preço unitário e valor total
select idmovimento, produto_idproduto, produto.nome as
'Produto',pessoa_idpessoa, pessoa.nome as 'Comprador',
movimento.quantidade, valorUnitario,
(movimento.quantidade * valorUnitario) as valor_total
from movimento
join pessoa
on movimento.pessoa_idpessoa = pessoa.idpessoa
join produto
on movimento.produto_idproduto = produto.idproduto
where movimento.tipo = 'S';

--Valor total das entradas agrupadas por produto.
select produto.nome, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL ENTRADAS'
from movimento
JOIN produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'E'

```

```
group by produto.nome;

--Valor total das saídas agrupadas por produto.
select produto.nome, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL SAIDAS'
from movimento
JOIN produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'S'
group by produto.nome;

--Operadores que não efetuaram movimentações de entrada
(compra).
select movimento.Usuario_idUsuario AS 'ID DO OPERADOR'
from movimento
except
select movimento.Usuario_idUsuario
from movimento
where movimento.tipo = 'E';

--Valor total de entrada, agrupado por operador.
select usuario.login AS OPERADOR, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL ENTRADAS'
from movimento
JOIN usuario
on usuario.idusuario = movimento.Usuario_idUsuario
where movimento.tipo = 'E'
group by usuario.login;

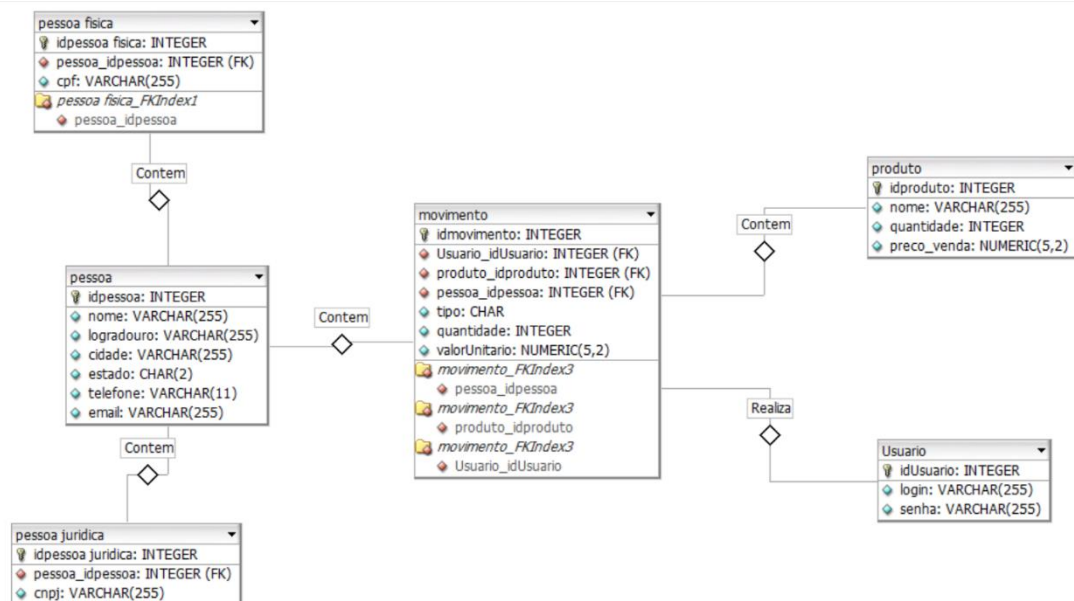
--Valor total de saída, agrupado por operador.
select usuario.login AS OPERADOR, SUM (movimento.quantidade *
movimento.valorUnitario) AS 'VALOR TOTAL SAIDAS'
from movimento
JOIN usuario
on usuario.idusuario = movimento.Usuario_idUsuario
where movimento.tipo = 'S'
group by usuario.login;

--Valor médio de venda por produto, utilizando média ponderada.
select produto.nome, SUM (movimento.quantidade *
movimento.valorUnitario) / SUM(movimento.quantidade) as 'Valor
médio de venda'
from movimento
JOIN produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'S'
group by produto.nome;
```

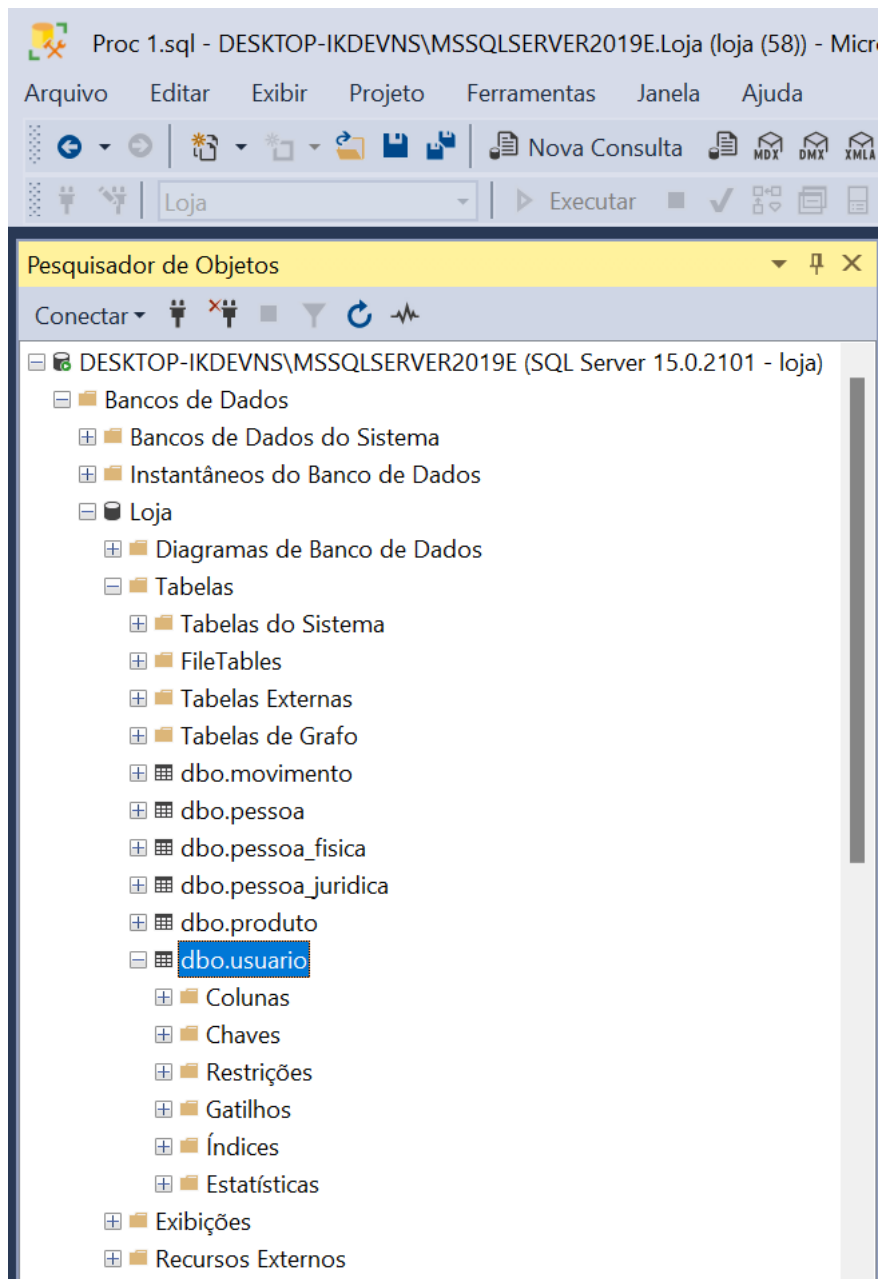
Resultados:

► Procedimento 1:

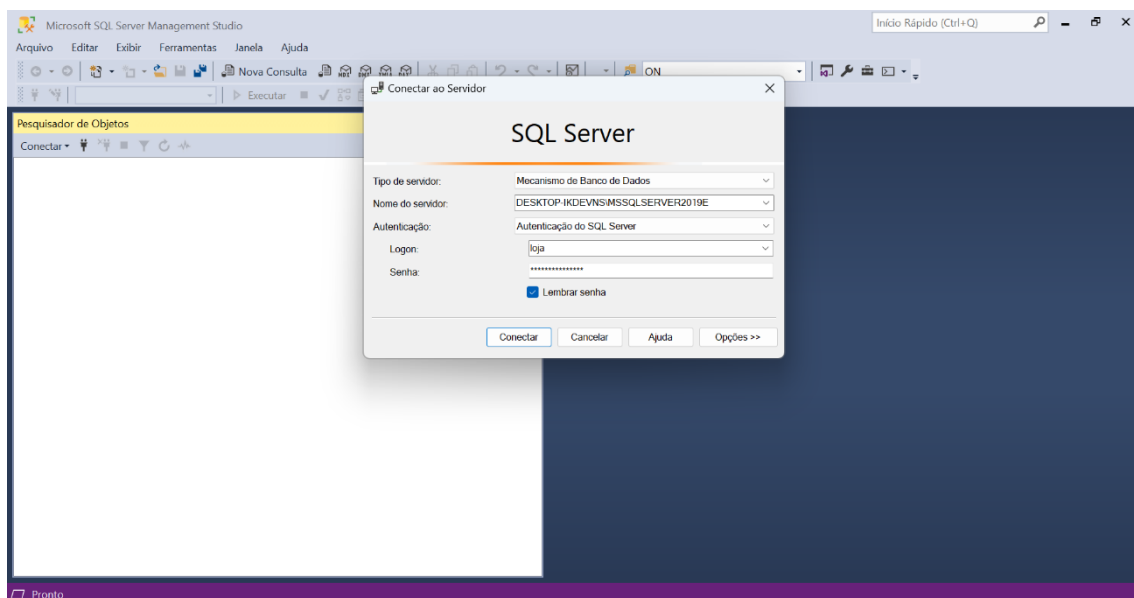
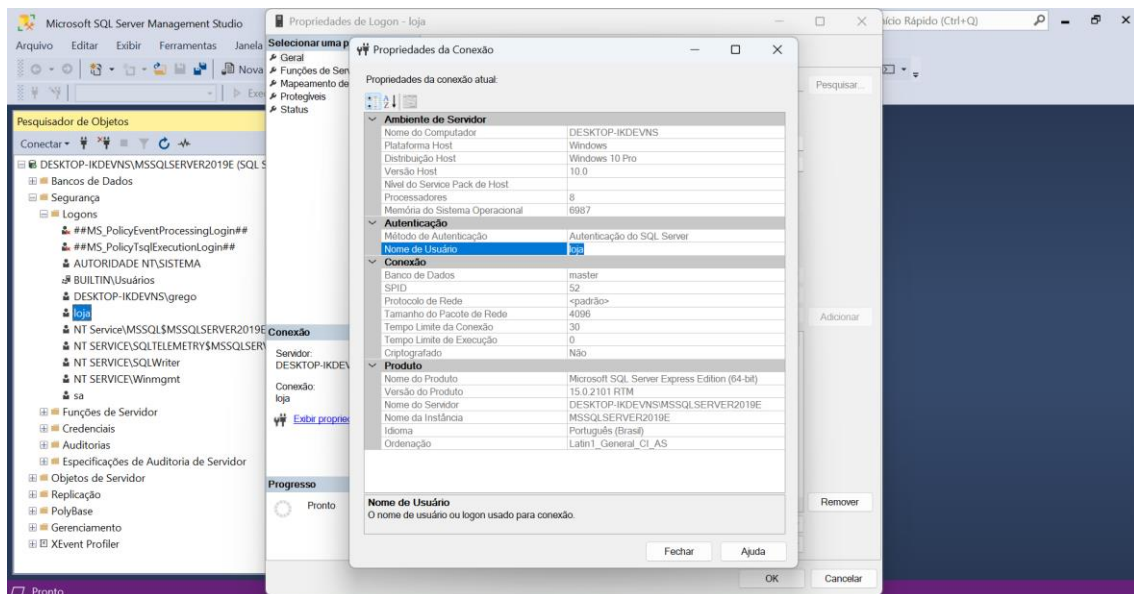
Modelo



Banco e Tabelas

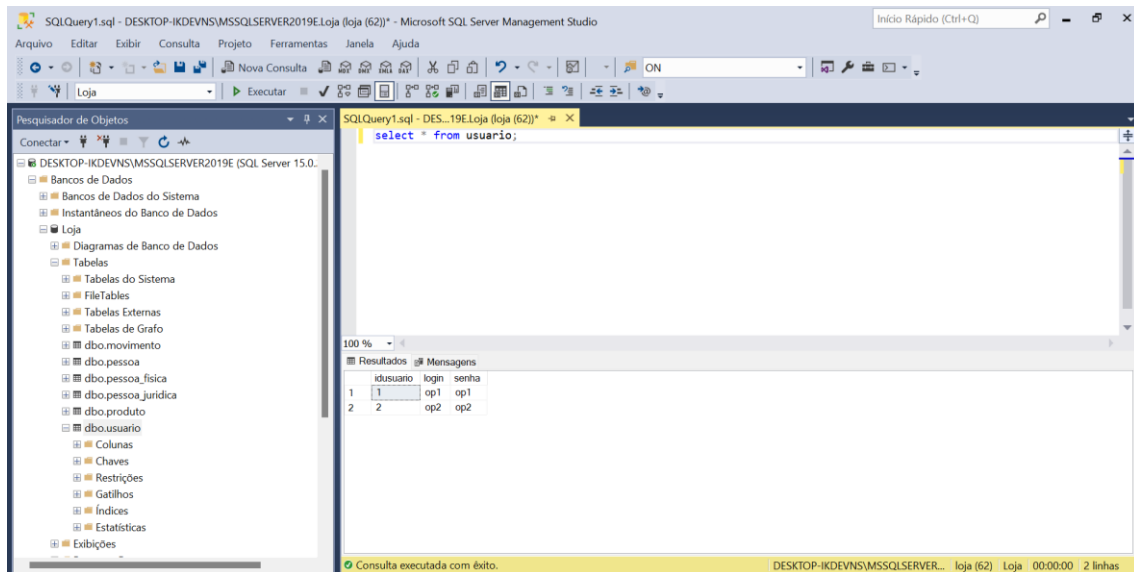


Usuario/Logon Loja

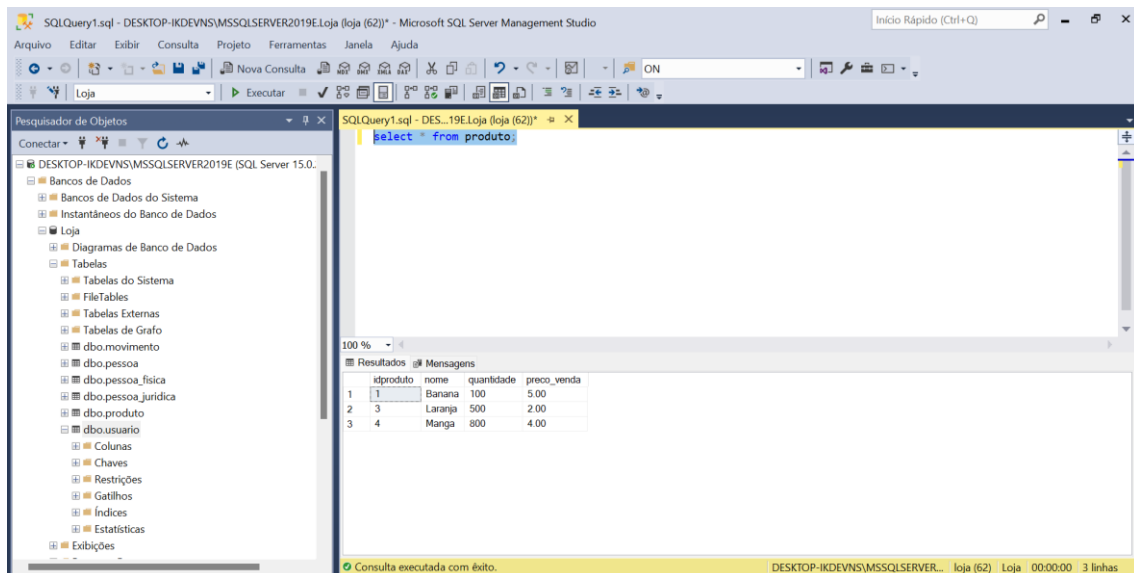


► Procedimento 2:

Usuarios



Produtos



Movimentos

Proc 2.sql - DESKTOP-IKDEVNS\MSSQLSERVER2019E Loja (58)* - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja

Pesquisador de Objetos

Conectar

DESKTOP-IKDEVNS\MSSQLSERVER2019E (SQL Server 15.0.)

Bancos de Dados

Bancos de Dados do Sistema

Instantâneos do Banco de Dados

Loja

Diagramas de Banco de Dados

Tabelas

Tabelas do Sistema

FileTables

Tabelas Externas

Tabelas de Grafo

dbo.movimento

dbo.pessoa

dbo.pessoa_fisica

dbo.pessoa_juridica

dbo.produto

dbo.usuario

Colunas

Chaves

Restrições

Gatilhos

Índices

Estadísticas

Exibições

Proc 2.sql - DESKTO...19E Loja (58)*

```
--Valor médio de venda por produto, utilizando média ponderada.
select produto.nome, SUM (movimento.quantidade * movimento.valorUnitario) / SUM(movimento.quantidade) as 'Valor mé
from movimento
JOIN produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'S'
group by produto.nome;

select * from movimento;
```

100 %

Resultados Mensagens

idmovimento	Usuario_idUsuario	pessoa_idpessoa	produto_idproduto	quantidade	tipo	valorUnitario
1	1	1	1	20	S	4.00
2	4	1	3	15	S	2.00
3	5	2	1	10	S	3.00
4	7	1	3	15	E	5.00
5	8	1	2	20	E	4.00

Consulta executada com êxito. DESKTOP-IKDEVNS\MSSQLSERVER... loja (58) Loja 00:00:00 5 linhas

Dados completos de pessoas físicas e jurídicas

Proc 2.sql - DESKTOP-IKDEVNS\MSSQLSERVER2019E Loja (52)* - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja

Pesquisador de Obj...

Conectar

DESKTOP-IKDEVNS\MSSQLS...

Bancos de Dados

Bancos de Dados do Si...

Instantâneos do Banco...

Loja

lojateste

Segurança

Objetos de Servidor

Replicação

PolyBase

Gerenciamento

XEvent Profiler

Proc 1.sql - DESKTO...E-master (loja (55))

Proc 2.sql - DESKTO...19E Loja (52)*

```
(8,1,2,4,20,'E',4.00);

-- Dados completos de pessoas físicas.
select *
from pessoa, pessoa_fisica
where pessoa.idpessoa = pessoa_fisica.idpessoa;

-- Dados completos de pessoas jurídicas.
select *
from pessoa, pessoa_juridica
where pessoa.idpessoa = pessoa_juridica.idpessoa;
```

90 %

Resultados Mensagens

idpessoa	nome	logradouro	cidade	estado	telefone	email	idpessoa	cpf
1	Joao	Rua 12, cas 3, Quitanda	Riachão do Sul	PA	1111-1111	joao@riachao.com	1	111111111111

idpessoa	nome	logradouro	cidade	estado	telefone	email	idpessoa	cnpj
2	JJC	Rua 11, Centro	Riachão do Norte	PA	1212-1212	jgc@riachao.com	2	222222222222

Consulta executada com êxito. DESKTOP-IKDEVNS\MSSQLSERVER... loja (52) Loja 00:00:00 2 linhas

Movimentações de entrada e saída, com produto, comprador, fornecedor, quantidade, preço unitário e valor total

Proc 2.sql - DESKTOP-IKDEVNS\MSSQLSERVER2019E\Loja (loja (58)) - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja

Executar

91 %

Proc 2.sql - DESKTOP...19E\Loja (loja (58))

```
--Movimentações de entrada, com produto, fornecedor, quantidade, preço unitário e valor total.
select idmovimento, produto_idproduto, produto.nome as 'Produto', pessoa_idpessoa, pessoa.nome as 'Fornecedor', movimento.quantidade, valorUnitario,
(movimento.quantidade * valorUnitario) as valor_total
from movimento
join pessoa
on movimento.pessoa_idpessoa = pessoa.idpessoa
join produto
on movimento.produto_idproduto = produto.idproduto
where movimento.tipo = 'E';

--Movimentações de saída, com produto, comprador, quantidade, preço unitário e valor total
select idmovimento, produto_idproduto, produto.nome as 'Produto', pessoa_idpessoa, pessoa.nome as 'Comprador', movimento.quantidade, valorUnitario,
(movimento.quantidade * valorUnitario) as valor_total
from movimento
join pessoa
on movimento.pessoa_idpessoa = pessoa.idpessoa
join produto
on movimento.produto_idproduto = produto.idproduto
where movimento.tipo = 'S';
```

Resultados

idmovimento	produto_idproduto	Produto	pessoa_idpessoa	Fornecedor	quantidade	valorUnitario	valor_total
7	3	Laranja	2	JJC	15	5.00	75.00
8	4	Manga	2	JJC	20	4.00	80.00

idmovimento	produto_idproduto	Produto	pessoa_idpessoa	Comprador	quantidade	valorUnitario	valor_total
1	1	Banana	1	Joao	20	4.00	80.00
4	3	Laranja	1	Joao	15	2.00	30.00
5	3	Laranja	1	Joao	10	3.00	30.00

Consulta executada com êxito.

DESKTOP-IKDEVNS\MSSQLSERVER... loja (58) Loja 00:00:00 5 linhas

Valor total das entradas e saídas agrupadas por produto

Proc 2.sql - DESKTOP-IKDEVNS\MSSQLSERVER2019E\Loja (loja (58)) - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja

Executar

91 %

Proc 2.sql - DESKTOP...19E\Loja (loja (58))

```
where movimento.tipo = 'S';

--Valor total das entradas agrupadas por produto.
select produto.nome, SUM (movimento.quantidade * movimento.valorUnitario) AS 'VALOR TOTAL ENTRADAS'
from movimento
join produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'E'
group by produto.nome;

--Valor total das saídas agrupadas por produto.
select produto.nome, SUM (movimento.quantidade * movimento.valorUnitario) AS 'VALOR TOTAL SAIDAS'
from movimento
join produto
on produto.idproduto = movimento.produto_idproduto
where movimento.tipo = 'S'
group by produto.nome;

--Operadores que não efetuaram movimentações de entrada (compra).
```

Resultados

nome	VALOR TOTAL ENTRADAS
Laranja	75.00
Manga	80.00

nome	VALOR TOTAL SAIDAS
Banana	80.00
Laranja	60.00

Consulta executada com êxito.

DESKTOP-IKDEVNS\MSSQLSERVER... loja (58) Loja 00:00:00 4 linhas

Operadores que não efetuaram movimentações de entrada (compra). Valor total de entrada e saída agrupado por operador

Proc 2.sql - DESKTOP-IKDEVNS\MSSQLSERVER2019E\Loja (loja (58)) - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja

Executar

75 %

Resultados Mensagens

1 ID DO OPERADOR

2

OPERADOR VALOR TOTAL ENTRADAS

1 op1 155.00

OPERADOR VALOR TOTAL SAIDAS

1 op1 110.00

2 op2 30.00

Consulta executada com êxito. DESKTOP-IKDEVNS\MSSQLSERVER... loja (58) Loja 00:00:00 4 linhas

Valor médio de venda por produto, utilizando média ponderada

Proc 2.sql - DESKTOP-IKDEVNS\MSSQLSERVER2019E\Loja (loja (58)) - Microsoft SQL Server Management Studio

Arquivo Editar Exibir Consulta Projeto Ferramentas Janela Ajuda

Loja

Executar

91 %

Resultados Mensagens

1 nome Valor médio de venda

2 Banana 4.000000

3 Laranja 2.400000

Consulta executada com êxito. DESKTOP-IKDEVNS\MSSQLSERVER... loja (58) Loja 00:00:00 2 linhas

Análise e Conclusão

- Como são implementadas as diferentes cardinalidades, basicamente 1X1, 1XN ou NxN, em um banco de dados relacional?

Atraves dos graus de relação que entidades ou tabelas têm entre si

- Que tipo de relacionamento deve ser utilizado para representar o uso de herança em bancos de dados relacionais?

1x1

- Como o SQL Server Management Studio permite a melhoria da produtividade nas tarefas relacionadas ao gerenciamento do banco de dados?

Atraves de um editor de consultas, monitoramento de desempenho e segurança e gerenciamento de permissões.

- Quais as diferenças no uso de sequence e identity?

AS SEQUENCES são acionadas sempre quando forem necessárias, sem dependência de tabelas e campos no banco, onde pode ser chamada diretamente por aplicativos. AS SEQUENCES, nós podemos obter o novo valor antes de usá-lo em um comando, diferente do IDENTITY, onde não podemos obter um novo valor. Além disso, com o IDENTITY não podemos gerar novos valores em uma instrução UPDATE, enquanto que com SEQUENCE, já podemos. Com SEQUENCES, podemos definir valores máximos e mínimos, além de termos a possibilidade de informar que a mesma irá trabalhar de forma cíclica e com cache, além de podemos obter mais valores em sequencia de um só vez, utilizando para isso a procedure SP_SEQUENCE_GET_RANGE, onde então é permitido atribuímos os valores individuais para aumentar então o desempenho no uso da SEQUENCE. Uma das grandes utilidades em IDENTITY está no fato de podermos trabalhar com o mesmo na utilização de TRANSAÇÕES de INSERT, pois, só iremos gerar um próximo valor a partir do momento que o comando for executado, ou seja, que a transação for aceita, ao contrário de uma SEQUENCE, que uma vez chamado seu próximo valor, mesmo que ocorra um erro de transação, o valor é alterado.

- Qual a importância das chaves estrangeiras para a consistência do banco?

A utilização da chave estrangeira possibilita a implementação da integridade de dados diretamente no banco de dados, conhecida como integridade referencial. Uma chave estrangeira é a representação de um relacionamento entre tabelas.

- Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

Operadores do SQL pertencem à álgebra relacional:

SELEÇÃO, RESTRIÇÃO, PROJEÇÃO, UNIÃO, INTERSECÇÃO, DIFERENÇA DE CONJUNTOS, PRODUTO CARTESIANO, JUNÇÃO, DIVISÃO, RENOMEAÇÃO, ATRIBUIÇÃO;

Operadores do SQL pertencem AO cálculo relacional:

Igual, diferente, maior, menor, maior ou igual, menor ou igual.

- Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

O agrupamento em consultas É FEITO UTILIZANDO O "GRUPO BY".