RPG0015 - Vamos manter as informações!

Iggor Motta de Oliveira. Matricula: 202303067879

Campus Virtual - SIA
Disciplina: Vamos manter as informações? - Semestre Letivo 2024.3

Objetivo 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.

Código - 1º Procedimento:

```
CREATE SEQUENCE orderPessoa

AS INT

START WITH 1

INCREMENT BY 1;

CREATE TABLE Pessoa(

  idPessoa INTEGER NOT NULL,

  nome VARCHAR(255),

  cpf VARCHAR(11) NOT NULL,

  endereco VARCHAR(255),

  cidade VARCHAR(255),

  estado CHAR(2),

  telefone VARCHAR(12),
```

```
  email VARCHAR(255)
CONSTRAINT PK_Pessoa PRIMARY KEY CLUSTERED(idPessoa ASC)
);
CREATE TABLE PessoaFisica(
  FK_Pessoa_idPessoa INTEGER NOT NULL,
  cpf VARCHAR(11) NOT NULL,
  CONSTRAINT PK_PessoaFisica PRIMARY KEY CLUSTERED(FK_Pessoa_idPessoa ASC),
  CONSTRAINT FK_Pessoa_PessoaFisica FOREIGN KEY(FK_Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
CREATE TABLE PessoaJuridica(
  FK_Pessoa_idPessoa INTEGER NOT NULL,
  cnpj VARCHAR(14) NOT NULL,
  CONSTRAINT PK_PessoaJuridica PRIMARY KEY CLUSTERED(FK_Pessoa_idPessoa ASC),
  CONSTRAINT FK_Pessoa_PessoaJuridica FOREIGN KEY(FK_Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
CREATE TABLE Usuario(
  idUsuario INTEGER NOT NULL IDENTITY,
  loginName VARCHAR(255) NOT NULL,
  senha VARCHAR(255) NOT NULL,
```

```
  CONSTRAINT PK_Usuario PRIMARY KEY CLUSTERED(idUsuario ASC)
);
CREATE TABLE Produto(
  idProduto INTEGER NOT NULL IDENTITY,
  nome VARCHAR(255) NOT NULL,
  quantidade INTEGER,
  precoVenda DECIMAL,
  CONSTRAINT PK_Produto PRIMARY KEY CLUSTERED(idProduto ASC)
);
CREATE TABLE Movimento(
  idMovimento INTEGER  NOT NULL IDENTITY,
  FK_Usuario_idUsuario INTEGER NOT NULL,
  FK_Pessoa_idPessoa INTEGER NOT NULL,
  FK_Produto_idProduto INTEGER NOT NULL,
  quantidade INTEGER,
  tipo CHAR(1),
  precoUnitario DECIMAL,
  CONSTRAINT PK_Movimento PRIMARY KEY CLUSTERED(idMovimento ASC),
  CONSTRAINT FK_Usuario_Movimento FOREIGN KEY(FK_Usuario_idUsuario) REFERENCES Usuario(idUsuario)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  CONSTRAINT FK_Pessoa_Movimento FOREIGN KEY(FK_Pessoa_idPessoa) REFERENCES Pessoa(idPessoa)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
```

CONSTRAINT FK_Produto_Movimento FOREIGN KEY(FK_Produto_idProduto) REFERENCES Produto(idProduto)

ON UPDATE CASCADE

ON DELETE CASCADE);

Análise e Conclusão:

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

- 1x1: Adicionar uma chave estrangeira em uma das tabelas referenciando a chave primária da outra tabela .Isso garante que cada linha na tabela tenha no máximo uma linha associada na outra tabela.
- 1xN: Chaves estrangeiras são usadas para associar as tabelas pai e filho .Assim cada instância da entidade pai se relacione com muitas instâncias da entidade filho.
- NxN: Requer uma tabela de junção separada para armazenar todas as associações entre pares de instâncias. Essa tabela armazena todas as entidades relacionadas.

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

- <<Include>>
- <<extend>>

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

- Interface intuitiva facil de mexer, integra com outras ferrementas como o Visual Studio e o Azure assim temos mais opções de Recursos, Sugere código que ajuda . Tudo isso facilita o desenvolvimento.

2º Procedimento | Alimentando a Base

Código - 2º Procedimento:

```
INSERT INTO Pessoa(idPessoa,nome,endereco,cidade,estado,telefone,email)
 VALUES (NEXT VALUE FOR orderPessoa, 'Joao','Rua A, 11','Riacho do sul','PA','1111-1111','joao@riacho.com'),
  (NEXT VALUE FOR orderPessoa, 'JJC','Rua B, Centro','Riacho do Norte','PA','1212-1212','jjc@riacho.com');

INSERT INTO PessoaFisica(FK_Pessoa_idPessoa,cpf)
 VALUES (1,'11111111111');

INSERT INTO PessoaJuridica(FK_Pessoa_idPessoa,cnpj)
 VALUES (2,'22222222222222');


INSERT INTO Usuario(loginName,senha)
 VALUES ('op1','op1'),
  ('op2','op2');

INSERT INTO Produto(nome,quantidade,precoVenda)
 VALUES ('Banana',100,'5.00'),
  ('Laranja',500,'2.00'),
  ('Manga',800,'4.00');

INSERT INTO Movimento(FK_Usuario_idUsuario,FK_Pessoa_idPessoa,FK_Produto_idProduto,quantidade,tipo,precoUnitario)
 VALUES (1,1,1,10,'E',5.00),
  (2,2,2,20,'S',2.00),
  (1,3,3,30,'E',4.00);

SELECT p.*, pf.cpf
FROM Pessoa p
INNER JOIN PessoaFisica pf ON p.idPessoa = pf.FK_Pessoa_idPessoa;
```

```
SELECT p.*, pj.cnpj
FROM Pessoa p
INNER JOIN PessoaJuridica pj ON p.idPessoa = pj.FK_Pessoa_idPessoa;

SELECT m.*, p.nome as fornecedor, pr.nome as Produto, m.quantidade, m.precoUnitario, (m.quantidade * m.precoUnitario) as total
FROM Movimento m
INNER JOIN Pessoa p ON p.idPessoa = m.FK_Pessoa_idPessoa
INNER JOIN Produto pr ON pr.idProduto = m.FK_Produto_idProduto
WHERE m.tipo = 'E';

SELECT m.*, p.nome as comprador, pr.nome as Produto, m.quantidade, m.precoUnitario, (m.quantidade * m.precoUnitario) as total
FROM Movimento m
INNER JOIN Pessoa p ON m.FK_Pessoa_idPessoa = p.idPessoa
INNER JOIN Produto pr ON m.FK_Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S';

SELECT pr.nome, SUM(m.quantidade * m.precoUnitario) as compras
FROM Movimento m
INNER JOIN Produto pr ON m.FK_Produto_idProduto = pr.idProduto
WHERE m.tipo = 'E'
GROUP BY pr.nome;

SELECT pr.nome, SUM(m.quantidade * m.precoUnitario) as vendas
FROM Movimento m
INNER JOIN Produto pr ON m.FK_Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;

SELECT u.*
FROM Usuario u
LEFT JOIN Movimento m ON u.idUsuario = m.FK_Usuario_idUsuario AND m.tipo = 'E'
WHERE m.idMovimento IS NULL;
```

```
SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as compras
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'E'
GROUP BY u.loginName;

SELECT u.loginName, SUM(m.precoUnitario * m.quantidade) as vendas
FROM Movimento m
INNER JOIN Usuario u ON m.FK_Usuario_idUsuario = u.idUsuario
WHERE m.tipo = 'S'
GROUP BY u.loginName;

SELECT pr.nome, SUM(m.precoUnitario * m.quantidade) / SUM(m.quantidade) as media
FROM Movimento m
INNER JOIN Produto pr ON m.FK_Produto_idProduto = pr.idProduto
WHERE m.tipo = 'S'
GROUP BY pr.nome;
```

Análise e Conclusão:

a. Quais as diferenças no uso de *sequence* e *identity*?

- Sequence é um objeto independente, que pode ser utilizado pra preencher qualquer coluna, de uma ou mais tabelas. Identity Só é associado a uma coluna de uma tabela.
- Sequence pode ser definido valor mínimo e máximo (Ex: De 1 a 100). Já o Identity o valor máximo é o limite do tipo de dado da coluna.
- Sequence pode ser gerado um novo sequencial em comandos de UPDATE, caso necessário. Já o identity sequencial é gerado apenas no INSERT dos dados.

- Sequence o valor atual do sequencial pode ser o valor atual do sequencial pode ser consultando através da view sys.sequences. Identity consulta através da view sys.identity_columns.
- Sequence o valor da sequencia pode ser reiniciado. Identity O valor da sequencia NÃO pode ser reiniciado.
- Sequence ao ser iniciada em uma tabela já populada, os registros anteriores não serão alterados. No identity ao ser iniciada na tabela, a coluna inteira é populada com o sequential.

b. Qual a importância das chaves estrangerias para a consistência do banco?

- A chave estrangeiras ajudam a manter a consistência no caso de falhas.
- Ajuda na extração de dados consistentes entre tabelas para análise e relatórios,em geral facilita na manutenção.
- As chaves estrangeiras evitam que registros façam referência a dados inexistentes em outra tabela,ajunda o sitema a ter confiabilidade e consistência dos dados armazenados.

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

- Operadores da álgebra relacional: SELECT, UNION,EXCEPT, JOIN.
- EXISTS, IN, ANY, ALL são operadores de subconsultas ou seja para cálculo relacional.

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

- Utilizando a cláusula GROUP BY.É Obrigatório que todas as colunas que aparecem na cláusula SELECT estaja incluídas no GROUP BY.

Conclusão

Este trabalho foi bem desafiador, exigiu que eu buscasse muito conhecimento fora da plataforma, mas acho que deu certo. Encontrei algumas dificuldades ao longo do processo.Acredito que, ao enfrentar esses desafios, construí uma experiência prática que será importante para minha atuação no mercado de trabalho. Essa jornada reforçou minhas habilidades e me motivou a continuar me desenvolvendo, com o objetivo de me tornar um profissional cada vez mais preparado.