Disciplina: **Missão Prática | Nível 4 | Mundo 4**
Semestre: 2º - 2024
Aluno: DAVIDSON PEREIRA DE OLIVEIRA
Matrícula: 2022.11.54585-5
Repositório no GIT: dev-davidson/Missão-Pratica-Nível-4-Mundo-4 (github.com)

Relatório discente de acompanhamento

RPG0026 - Tirando proveito da nuvem para projetos de software

Design do Banco de Dados:

1. Definir a Arquitetura do Banco de Dados

Arquitetura Geral:

1.1. **Objetivo:** O objetivo é criar um banco de dados eficiente e escalável para a LogiMove Transportes, que lida com motoristas, clientes e pedidos. A arquitetura deve suportar a digitalização das operações e permitir a coordenação e rastreamento das remessas.

1.2. **Entidades e Tabelas:** As principais entidades do banco de dados são:

- **Motoristas (Drivers):** Armazena informações sobre motoristas, como suas qualificações e histórico de viagens.
- **Clientes (Clients):** Contém detalhes dos clientes, como contato e histórico de pedidos.
- **Pedidos (Orders):** Registra informações sobre os pedidos, incluindo detalhes, status e cronograma de entrega.

1.3. **Relacionamentos:**

- **Clientes e Pedidos:**
  - Um cliente pode fazer vários pedidos, mas cada pedido é feito por um único cliente.
  - Relacionamento: 1 (um para muitos)
- **Motoristas e Pedidos:**
  - Um motorista pode estar associado a vários pedidos, mas cada pedido é atribuído a um único motorista.
  - Relacionamento: 1(um para muitos)

1.4. **Considerações de Escalabilidade:**

- **Indexação:** Criar índices nas colunas frequentemente usadas em consultas, como `ClientID` e `DriverID`.
- **Segurança:** Configurar permissões adequadas para diferentes usuários e grupos.
- **Backup e Recuperação:** Implementar políticas de backup regulares para garantir a recuperação dos dados.

2. Criar um Diagrama de Entidade-Relacionamento (ER)

O diagrama ER visualiza as entidades, atributos e relacionamentos entre as tabelas. Aqui está uma descrição detalhada do diagrama ER para o banco de dados da LogiMove Transportes:

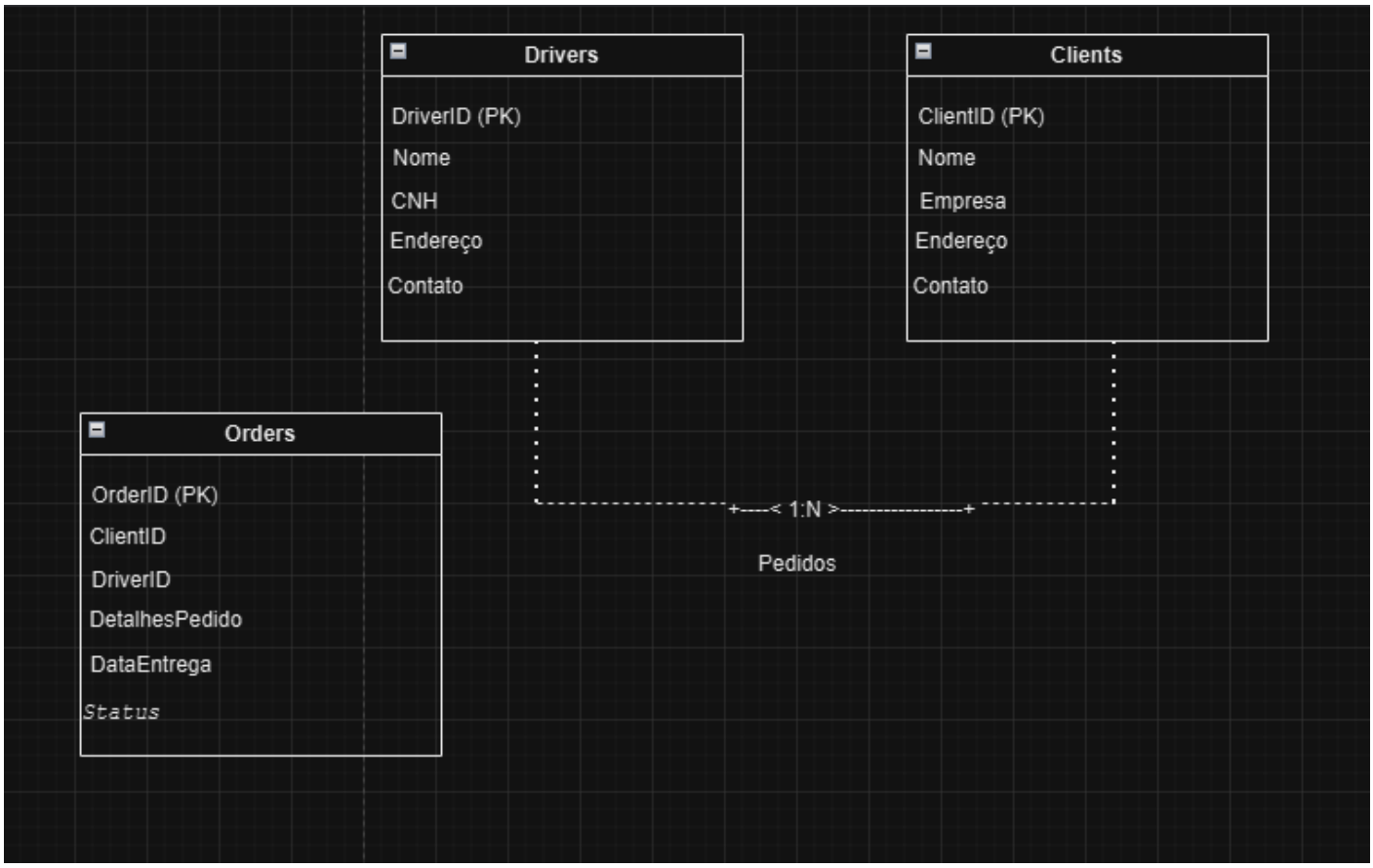
Entidades e Seus Atributos:

- **Motoristas (Drivers):**
  - `DriverID` (INT, PK, Identity)
  - `Nome` (VARCHAR(100))
  - `CNH` (VARCHAR(20))
  - `Endereço` (VARCHAR(200))
  - `Contato` (VARCHAR(50))
- **Clientes (Clients):**
  - `ClientID` (INT, PK, Identity)
  - `Nome` (VARCHAR(100))
  - `Empresa` (VARCHAR(100))
  - `Endereço` (VARCHAR(200))
  - `Contato` (VARCHAR(50))
- **Pedidos (Orders):**
  - `OrderID` (INT, PK, Identity)
  - `ClientID` (INT, FK)
  - `DriverID` (INT, FK)
  - `DetalhesPedido` (TEXT)
  - `DataEntrega` (DATE)
  - `Status` (VARCHAR(50))

Relacionamentos:

- **Clientes a Pedidos:**
  - Um cliente pode ter vários pedidos (`1:N`).
  - Representado como uma linha com um diamante `1` conectado à tabela de clientes e um diamante `N` conectado à tabela de pedidos.
- **Motoristas a Pedidos:**
  - Um motorista pode estar associado a vários pedidos (`1:N`).
  - Representado como uma linha com um diamante `1` conectado à tabela de motoristas e um diamante `N` conectado à tabela de pedidos.

Diagrama ER Visual:



Draw.io

1. Criação das Tabelas

Tabela de Motoristas (Drivers)

```
CREATE TABLE Drivers (
    DriverID INT PRIMARY KEY IDENTITY(1,1),  -- Chave primária com incremento automático
    Nome VARCHAR(100) NOT NULL,
    CNH VARCHAR(20) NOT NULL,
    Endereço VARCHAR(200),
    Contato VARCHAR(50)
);
```

Tabela de Clientes (Clients)

```
CREATE TABLE Clients (
    ClientID INT PRIMARY KEY IDENTITY(1,1),  -- Chave primária com incremento automático
    Nome VARCHAR(100) NOT NULL,
    Empresa VARCHAR(100),
    Endereço VARCHAR(200),
    Contato VARCHAR(50)
);
```

Tabela de Pedidos (Orders)

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY IDENTITY(1,1),  -- Chave primária com incremento automático
    ClientID INT NOT NULL,
    DriverID INT NOT NULL,
    DetalhesPedido TEXT,
    DataEntrega DATE,
    Status VARCHAR(50),
    FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),
    FOREIGN KEY (DriverID) REFERENCES Drivers(DriverID)
);
```

2. Definição de Índices

Índices para a Tabela de Motoristas

```
-- Índice para a coluna CNH, útil para buscas rápidas por CNH
CREATE INDEX IDX_Drivers_CNH ON Drivers(CNH);
```

Índices para a Tabela de Clientes

```
-- Índice para a coluna Nome, útil para buscas rápidas por nome
CREATE INDEX IDX_Clients_Nome ON Clients(Nome);
```

Índices para a Tabela de Pedidos

```
-- Índice para a coluna Status, útil para filtragem por status de pedidos
CREATE INDEX IDX_Orders_Status ON Orders(Status);

-- Índice composto para as colunas ClientID e DriverID, útil para consultas filtradas por cliente e motorista
CREATE INDEX IDX_Orders_ClientID_DriverID ON Orders(ClientID, DriverID);
```

3. Revisão e Validação

Verificar a Estrutura das Tabelas:

```
-- Verificar estrutura da tabela Drivers
EXEC sp_help 'Drivers';

-- Verificar estrutura da tabela Clients
EXEC sp_help 'Clients';

-- Verificar estrutura da tabela Orders
EXEC sp_help 'Orders';
```

Testar a Inserção de Dados:

```
-- Inserir dados de teste
INSERT INTO Drivers (Nome, CNH, Endereço, Contato) VALUES
('João Silva', '123456789', 'Rua A, 123', '1234-5678'),
('Maria Oliveira', '987654321', 'Rua B, 456', '8765-4321');

INSERT INTO Clients (Nome, Empresa, Endereço, Contato) VALUES
('Empresa X', 'Logística', 'Av. Central, 789', '1122-3344'),
('Cliente Y', 'Comércio', 'Rua do Comércio, 101', '5566-7788');

INSERT INTO Orders (ClientID, DriverID, DetalhesPedido, DataEntrega, Status) VALUES
(1, 1, 'Entrega de mercadorias para a Empresa X', '2024-08-30', 'Pendente'),
(2, 2, 'Entrega de produtos para o Cliente Y', '2024-08-25', 'Entregue');
```

Executar Consultas de Teste:

```
-- Consultar todos os pedidos
SELECT * FROM Orders;

-- Consultar pedidos de um cliente específico
SELECT * FROM Orders WHERE ClientID = 1;

-- Consultar motoristas com pedidos pendentes
SELECT d.Nome, o.Status FROM Drivers d
JOIN Orders o ON d.DriverID = o.DriverID
WHERE o.Status = 'Pendente';
```

Inserção e Gestão de Dados

```
INSERT INTO Drivers (Nome, CNH, Endereço, Contato) VALUES ('João Silva', '123456789', 'Rua A, 123', '1234-5678');
INSERT INTO Clients (Nome, Empresa, Endereço, Contato) VALUES ('Empresa X', 'Logística', 'Av. Central, 789', '1122-3344');
INSERT INTO Orders (ClientID, DriverID, DetalhesPedido, DataEntrega, Status) VALUES (1, 1, 'Entrega para Empresa X', '2024-08-30', 'Pendente');
```

Execução e Validação de Consultas

```
-- Consultar todos os pedidos
SELECT * FROM Orders;

-- Consultar pedidos por status
SELECT * FROM Orders WHERE Status = 'Pendente';

-- Consultar detalhes do motorista para um pedido específico
SELECT d.Nome, d.Contato FROM Drivers d
JOIN Orders o ON d.DriverID = o.DriverID
WHERE o.OrderID = 1;
```

Operações CRUD Eficientes

- **Create:** Inserir novos registros.

```
INSERT INTO Orders (ClientID, DriverID, DetalhesPedido, DataEntrega, Status) VALUES (2, 1, 'Entrega urgente', '2024-09-05', 'Pendente');
```

- **Read:** Ler registros existentes.

```
SELECT * FROM Orders WHERE Status = 'Pendente';
```

- **Update:** Atualizar registros existentes.

```
UPDATE Orders SET Status = 'Entregue' WHERE OrderID = 1;
```

- **Delete:** Excluir registros.

```
DELETE FROM Orders WHERE OrderID = 2;
```