Normalização e Modelagem de Banco de Dados

Fevereiro 11, 2026

Contents

1 Introdução

2 Regras de Negócio Identificadas

3 Normalização (1FN,2FNe3FN)

3.1 Primeira Forma Normal (1FN) ....
3.2 Segunda Forma Normal (2FN). . . . .
3.3 Terceira Forma Normal(3FN)..... . . . . .

4 Modelo Conceitual (resumo)

4.1 Entidades principais . . . .
4.2 Relacionamentos principais..... . . . . .

5 Modelo Lógico (resumo)

5.1 Tabelas principais . . . .

6 Modelo Físico (SQL)

7 Justificativas Técnicas

7.1 Porque existe a tabela Item_Pedido ? .. . .
7.2 Porque o Rastreamento está separado?. . . . .
7.3 Porque foi utilizado CHECK no campo status ? . . . . . .

8 Escalabilidade

9 Conclusão

1 Introdução

A empresa **Soluções Express** está em fase de expansão e enfrenta um desafio comum a muitas organizações: a substituição de planilhas desorganizadas por uma solução de banco de dados estruturada e eficiente. O objetivo desta iniciativa é garantir a integridade dos dados, reduzir a redundância, facilitar consultas rápidas e proporcionar escalabilidade para suportar o crescimento futuro da empresa. Uma modelagem cuidadosa e a aplicação rigorosa das formas nor- mais são fundamentais para alcançar esse cenário de qualidade e ro- bustez no sistema de armazenamento.

2 Regras de Negócio Identificadas

A análise do problema permitiu definir claramente as regras de negócio que guiarão o design do banco de dados. Seguem as principais:

- Um cliente pode ser **Pessoa Física** (identificado por CPF) ou **Pessoa Jurídica** (identificado por CNPJ).
- Um cliente pode realizar múltiplos pedidos.
- Cada pedido possui um código único, data, status e valor total.
- A relação entre pedido e produto é de muitos para muitos(N:N), ou seja, um pedido pode conter vários produtos e um produto pode estar em vários pedidos.

- Paracadaitemdopedido,énecessárioarmazenaraquantidade e o preço unitário vigente na data da compra.
- ExistementregadoresautônomosidentificadospeloCPF,veículo e placa.
- Cada pedido deve possuir um rastreamento com histórico contendo status, data/hora e localização.

3 Normalização(1FN,2FNe3FN)

A normalização é essencial para garantir a qualidade do mod- elo de dados, evitando redundâncias e inconsistências. A seguir, descrevemos a aplicação das formas normais neste projeto.

3.1 Primeira Forma Normal (1FN)

A 1FN exige que os atributos sejam atômicos, isto é, não devem conter múltiplos valores, e que não existam grupos repetitivos em uma mesma tabela.

Aplicação no projeto:

- Os produtos relacionados a um pedido não foram armazenados em uma única coluna com valores múltiplos. Para isso, foi criada a tabela Item_Pedido, que representa cada item individualmente.

- O rastreamento, que possui histórico, não foi consolidado em um campo único; em vez disso, foi criada a tabela Rastreamento para registrar múltiplas atualizações.

3.2 Segunda Forma Normal (2FN)

A 2FN requer que não haja atributos que dependam apenas de parte da chave composta.

Aplicação no projeto: AtabelaItem_Pedido possui uma chave primária composta pelos campos id_pedido e id_produto. Os atributos quantidade e preco_unitario_na_compra dependem integralmente dessa chave composta, atendendo à 2FN.

3.3 Terceira Forma Normal (3FN)

A 3FN determina que não deve haver dependências transitivas entre atributos que não sejam chave.

Aplicação no projeto:

- A tabela Categoria foi separada da tabela Produto para evitar repetição das informações da categoria.
- A separação entre Cliente e Pedido impede a duplicação dos dados pessoais dos clientes em múltiplos pedidos.
- O rastreamento foi segmentado em tabela própria para manter o histórico sem redundância.

4 ModeloConceitual(resumo)

O modelo conceitual apresenta as principais entidades e seus relacionamentos, destacando a estrutura lógica do sistema.

4.1 Entidades principais

- **Cliente**
- **Pedido**
- **Produto**
- **Categoria**
- **Item_Pedido**
- **Entregador**
- **Entrega**
- **Rastreamento**

4.2 Relacionamentos principais

Relacionamento	**Descrição** Um cliente pode
Cliente (1) □ (N) Pedido	fazer vários pedidos.
Pedido (N) □ (N) Produto	Relacionamento N:N resolvido pela tabela Item_Pedido.
Categoria (1) □ (N) Produto	Uma categoria pode agrupar vários produtos.
Pedido (1) □ (N) Rastreamento	Cada pedido possui múltiplos registros de rastreamento histórico. Um entregador
Entregador (1) □ (N) Entrega	pode realizar diversas entregas.
Pedido (1) □ (1) Entrega	Cada pedido possui uma entrega associada.

5 Modelo Lógico (resumo)

O modelo lógico detalha as tabelas, suas chaves primárias e estrangeiras, representando fielmente o modelo conceitual em estruturas relacionais.

5.1 Tabelas principais

Tabela	Chaves
Cliente	PK id_cliente
Pedido	PK id_pedido ,FK id_cliente
Produto	PK id_produto ,FK id_categoria
Categoria	PK id_categoria
Item_Pedido	PK composta (id_pedido, id_produto)
Entregador	PK id_entregador
Entrega	PK id_entrega ,FK id_pedido ,FK id_entregador
Rastreamento	PK id_rastreamento, FK id_pedido

6 Modelo Físico (SQL)

A implementação física foi realizada no SQL Server, contemplando restrições que asseguram a integridade dos dados.

As principais estruturas adotadas foram:

- PRIMARY KEY para garantir unicidade dos registros.
- FOREIGN KEYparamanteraintegridadereferencialentretabelas.
- CHECK para validar os valores do campo status.
- UNIQUE para garantir a singularidade de CPF, CNPJ e placa dos veículos.

O script SQL completo segue abaixo:

```
CREATE TABLE Categoria (
    id_categoria INT PRIMARY KEY IDENTITY,
    nome VARCHAR(100) NOT NULL
);

CREATE TABLE Cliente (
    id_cliente INT PRIMARY KEY IDENTITY,
    nome VARCHAR(150) NOT NULL,
    tipo CHAR(1) NOT NULL CHECK (tipo IN ('F', 'J')),
    cpf VARCHAR(11) UNIQUE NULL,
    cnpj VARCHAR(14) UNIQUE NULL,
    endereco VARCHAR(255) NULL
);

CREATE TABLE Produto (
    id_produto INT PRIMARY KEY IDENTITY,
    nome VARCHAR(150) NOT NULL,
    id_categoria INT NOT NULL,
```

```
    preco DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (id_categoria) REFERENCES Categoria(id_categoria)
);

CREATE TABLE Pedido (
    id_pedido INT PRIMARY KEY IDENTITY,
    id_cliente INT NOT NULL,
    data_pedido DATETIME NOT NULL DEFAULT GETDATE(),
    status VARCHAR(20) NOT NULL CHECK (status IN ('pendente', 'em_tra
    valor_total DECIMAL(12,2) NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente)
);

CREATE TABLE Item_Pedido (
    id_pedido INT NOT NULL,
    id_produto INT NOT NULL,
    quantidade INT NOT NULL CHECK (quantidade > 0),
    preco_unitario_na_compra DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (id_pedido, id_produto),
    FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido),
    FOREIGN KEY (id_produto) REFERENCES Produto(id_produto)
);

CREATE TABLE Entregador (
    id_entregador INT PRIMARY KEY IDENTITY,
    nome VARCHAR(150) NOT NULL,
    cpf VARCHAR(11) UNIQUE NOT NULL,
```

```
    veiculo VARCHAR(50) NOT NULL,
    placa VARCHAR(10) UNIQUE NOT NULL
);

CREATE TABLE Entrega (
    id_entrega INT PRIMARY KEY IDENTITY, id_pedido INT NOT NULL UNIQUE
    id_entregador INT NOT NULL, data_entrega DATETIME NULL, FOREIGN KEY
    (id_pedido) REFERENCES Pedido(id_pedido), FOREIGN KEY (id_entregador
    REFERENCES Entregador(id_entregador)


);

CREATE TABLE Rastreamento (
    id_rastreamento INT PRIMARY KEY IDENTITY,
    id_pedido INT NOT NULL,
    status VARCHAR(20) NOT NULL CHECK (status IN ('pendente', 'em_tra
    data_hora DATETIME NOT NULL DEFAULT GETDATE(),
    localizacao VARCHAR(255) NULL,
    FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido)
);
```

7 Justificativas Técnicas

7.1 Por que existe a tabela Item_Pedido?

A relação entre Pedido e Produto é de muitos para muitos, o que impossibilita armazenar diretamente os produtos dentro da tabela Pedido sem criar redundância ou grupos repetitivos. A tabelaItem_Pedido resolve essa questão ao representar cada produto individualmente dentro de um pedido, permitindo registrar tanto a **quantidade** quanto o **preço unitário** vigente no momento da compra, informações essenciais para controle e faturamento.

7.2 Por que o Rastreamento está separado?

O rastreamento necessita armazenar um histórico completo das etapas por que um pedido passa. Consolidar essas informações em um único campo levaria à perda de dados ou à necessidade de formatos complexos e pouco eficientes. Com a tabela Rastreamento, cada atualização de status, data/hora e localização é registrada como um novo registro, garantindo rastreabilidade detalhada e organizada.

7.3 Por que foi utilizado CHECK no campo status?

A constraint CHECK assegura que o campo status aceite somente valores válidos: *pendente*, *em_transporte* e *entregue*. Essa restrição evita erros de digitação, inconsistências e mantém o padrão dos dados, facilitando consultas e integrações futuras.

8 Escalabilidade

Pensando no crescimento futuro do sistema, foram planejadas estratégias para garantir flexibilidade e extensibilidade:

- Possibilidade de adicionar novos status criando uma tabela Status, caso os valores se tornem dinâmicos.
- Inclusão de formas de pagamento através de uma tabela Pagamento, permitindo múltiplos métodos.
- Suporte a múltiplas entregas por pedido, como tentativas de entrega, por meio de ajustes na tabela Entrega.
- Desmembramento do endereço para uma tabela própria caso haja necessidade de maior detalhamento e reutilização.

Essas previsões asseguram que o banco de dados acompanhe o crescimento e a complexidade do negócio sem comprometer a eficiência e a organização.

9 Conclusão

O banco de dados desenvolvido para a **Soluções Express** foi modelado rigorosamente seguindo as normas de normalização (1FN, 2FN, 3FN), o que proporcionou uma estrutura robusta, eficiente e escalável. A redução da redundância e a manutenção da integridade dos dados garantem a qualidade do sistema, enquanto a clara organização fa- cilita consultas e atualizações. Com a modelagem adequada e as im- plementações técnicas apropriadas, a empresa está preparada para

um crescimento sustentável e uma melhor gestão das operações por meio do banco de dados.