Universidade Federal do Mato Grosso
Instituto de Computação

Projeto de Lab. de Banco de Dados

ANTHONY RICARDO RODRIGUES REZENDE
ALAN BRUNO MORAES COSTA
VINICIUS PADILHA VIEIRA

CUIABÁ, MATO GROSSO 2024

ANTHONY RICARDO RODRIGUES REZENDE
ALAN BRUNO MORAES COSTA
VINICIUS PADILHA VIEIRA

Projeto de Lab. de Banco de Dados

Trabalho sobre "Modelagem de Banco de Dados"conforme a disciplina de Lab. de Banco de Dados, apresentado como requerido para a obtenção de nota na Universidade Federal de Mato Grosso - UFMT

Professor: Dr. Josiel Figueiredo

CUIABÁ, MATO GROSSO 2024

Sumário

Resumo

Este trabalho consiste no desenvolvimento de um projeto de Banco de Dados, o qual possui especificação de requisitos, como Modelagem de Banco de Dados desde a parte Conceitual até a Lógico, bem como a construção das tabelas em *script* SQL mais a construção do Diagrama de classe do mesmo.

1 Introdução

O trabalho trata da criação de um sistema de banco de dados que será aplicado a alguma rede farmacêutica, na qual possui múltiplas filiais espalhadas pelo Brasil, o projeto visa aprimorar a eficiência operacional e o gerenciamento de informações. O sistema abrange entidades como filiais, funcionários, fornecedores, produtos, clientes e vendas, promovendo a tomada de decisões embasada em dados colhidos pelo sistema. Isso permite o acompanhamento de estoques, desempenho das filiais e conformidade regulatória, mantendo a qualidade dos produtos e serviços oferecidos. A construção desse banco de dados é essencial para atender às crescentes demandas de uma rede farmacêutica em expansão.

2 Descrição - Sistema para Filiais de Redes Farmacêuticas

Uma Rede Farmacêutica abriu diversas filiais espalhadas pelo Brasil. Além do nome LAV, a rede também possui ID, CNPJ e um site.

Cada filial da rede tem um número exclusivo de identificação, um telefone e um endereço. As filiais possuem funcionários, os quais são identificados por um ID e apresentam um CPF, um nome, número de telefone, cargo, certificado(s) e salário.

A rede farmacêutica também mantém relacionamentos com fornecedores que abastecem suas filiais. Cada fornecedor possui um ID, um CNPJ, um nome, telefone, endereço e portfólio de produtos.

Os produtos da farmácia incluem informações como ID, nome, preço, marca, data de validade e data de fabricação. Dentro da categoria de produtos, há dois tipos: Remédios e Outros. Cada remédio apresenta um ID, o tipo do remédio (original ou genérico), a tarja que indica sua venda, a forma de fabricação, a prescrição médica (com receita ou não), a composição do remédio e a forma de administração. Cada remédio está associado a uma classe específica de medicamentos, que é identificada por um ID e um nome.

Além disso, os remédios contém informações sobre os princípios ativos usados na fabricação deles. Cada princípio ativo armazena um ID e um nome.

Os outros produtos possuem seções que incluem informações sobre as diferentes categorias dentro das filiais da rede farmacêutica, cada uma identificada por um ID e um nome.

Os funcionários da LAV atendem clientes, cada cliente possui CPF, ID, nome e telefone. A cada venda realizada são armazenados a data e o ID desta venda, tais dados servirão para a emissão de relatórios para fins de controle da Rede Farmacêutica.

3 Regras de Negócio

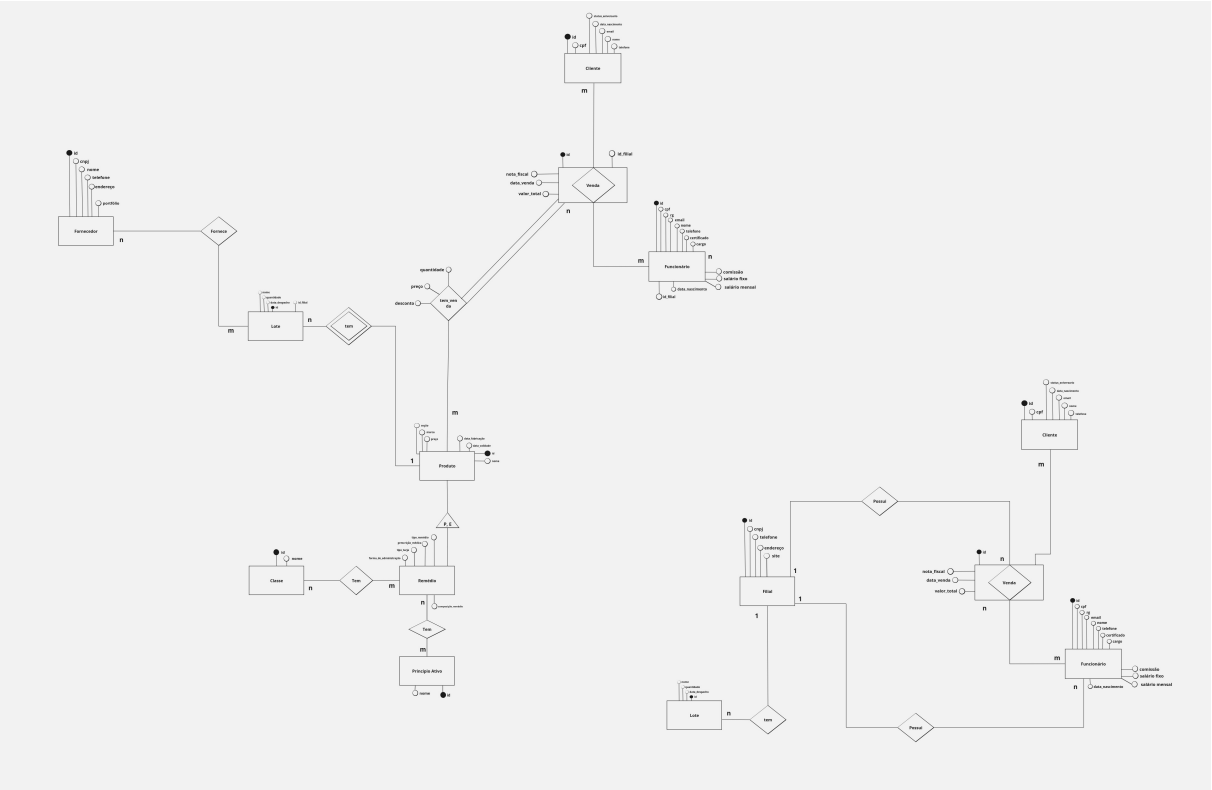
Regras de negócio são diretrizes, condições e lógicas específicas que determinam como os processos de negócio devem ser conduzidos dentro de uma organização. Elas definem os padrões para decisões operacionais, garantindo consistência e eficiência nas operações. As regras de negócio orientam ações como processamento de transações, gerenciamento de dados, e fluxos de trabalho, assegurando que a empresa atue de maneira alinhada aos seus objetivos estratégicos, políticas internas, e regulamentações do setor.

- **Gestão Geral:** A representação do todo é identificado como sendo a matriz, unidade no qual fica encarregada por monitorar e ser o núcleo estratégico das demais filiais.
- **Gestão das Filiais:** As filiais terão acesso de modificação nas bases lote, venda e funcionário. Sendo assim, o restante fica a cargo da matriz realizar o gerenciamento.
- **Gestão de Estoque:** Manter controle de estoque atualizado, alertando quando um produto estiver abaixo do mínimo necessário. Reordenar os produtos de acordo com aqueles mais recentes, de modo que dê prioridade aos produtos mais novos no que diz respeito à venda de produtos do tipo remédio.
- **Gestão de Vendas:** Registrar vendas, associando cada venda a um funcionário e uma filial. Aplicar descontos baseados em promoções ou fidelidade do cliente.
- **Gestão de Clientes:** Manter um registro detalhado dos clientes, incluindo histórico de compras. Oferecer programas de fidelidade com benefícios baseados no histórico de compras e no aniversário do cliente.
- **Controle de Acesso:** Diferentes níveis de acesso para funcionários baseados em suas funções, garantindo segurança dos dados sensíveis.
- **Relatórios Financeiros:** Gerar relatórios de vendas, lucros, e despesas por filial, permitindo análise de desempenho e identificação de áreas para melhoria.
- **Rastreamento de Lotes de Produtos:** Monitorar os lotes de produtos desde o recebimento até a venda, garantindo rastreabilidade em caso de recall.
- **Gestão de Prescrições Médicas:** Associar vendas de medicamentos controlados a prescrições médicas, mantendo um registro seguro e conforme com a legislação.
- **Análise de Tendências de Compra:** Utilizar dados históricos de vendas para identificar tendências e ajustar estoques e promoções de acordo.
- **Dados da tabela classe e principio_ativo:** Os dados sobre os tipos de classe e princípios ativos serão fornecidos pela base legal da ANVISA (Agência Nacional de Vigilância Sanitária). Sendo que ao cadastrar cada produto, à nível de aplicação será disponível com "select box"com pesquisa para adicionar as classes e princípios ativos, na qual um remédio possui.
- **Clientes Aniversariantes:** Com base no valor do atributo status_aniversario seria enviado uma mensagem ao cliente lhe desejando felicitações, além de lhe informar e presentear com o desconto de 30% na primeira compra do dia.
- **Cadastro de Fornecedores:** Os fornecedores se cadastrarão através de um acesso, no qual ele terá que provar que tem todos os selos de garantia de qualidade de produtos. Também fornecerão dados como endereço e telefone para o cadastro efetivo.

4 DER (Diagrama Entidade Relacionamento)

Um DER, ou Diagrama de Entidade e Relacionamento, é uma representação gráfica de um modelo de dados. Ele é usado para descrever as entidades, atributos e relacionamentos entre os dados em um banco de dados. Nesse caso, realizamos um DER baseado na proposta de uma de um sistema de controle e venda de produtos farmacêuticos a partir das filiais de uma determinada rede farmacêutica:

Figura 1: DER - Sistema para Filiais de Redes Farmacêuticas



Fonte: imagem do computador

5 Dicionário de Dados

O dicionário é um livro ou recurso que contém palavras organizadas em ordem alfabética, cada uma acompanhada de informações sobre o significado, pronúncia, uso e, em alguns casos, origem. A principal utilidade de um dicionário é fornecer definições claras e compreensíveis das palavras, ajudando as pessoas a entenderem o significado e o contexto de termos específicos em uma língua. Dado isso, foi elaborado o seguinte dicionário de dados, de modo a facilitar a leitura do DER.

5.1 Dicionário de Entidades & Relacionamentos

Logo abaixo, tem-se todas as informações sobre as Entidades e seus respectivos relacionamentos. Essas informações estão disponíveis para facilitar o entendimento da Modelagem de Dados.

Entidade	Relacionamento C.	Nome do Relac.	Descrição
Filial	Funcionário	Possui	Uma filial possui N funcionários
Filial	Lote	Fornece	Toda Filial recebe um lote de produtos.
Filial	Venda	Possui	Toda Filial tem vendas de produtos.
Fornecedor	Lote	Fornece	Fornecedor se encarrega de fornecer um lote de produtos, como maneira de ocorrer fluxo de mercadorias.
Produto	Venda	item_venda	Na venda de um produto teremos uma informações sobre tudo da venda, cada item.
Produto	Lote	Tem	Todo produto faz parte de um ou mais lotes de produtos.
Funcionário	Filial	Possui	Uma filial possui N funcionários
Funcionário	Cliente	Venda	Um cliente é atendido por um funcionário para realizar a sua compra.
Funcionário	Venda	Venda	Um funcionário faz vendas para um cliente
Cliente	Funcionário	Venda	Um cliente é atendido por um funcionário para realizar a sua compra.
Cliente	Venda	Venda	Na venda de um produto temos um cliente comprador
Venda	Cliente	Venda	Na venda de um produto temos um cliente comprador
Venda	Filial	Possui	Toda venda faz parte de uma filial.
Princípio ativo	Remédio	Tem	Um remédio tem vários princípios ativos.
Classe	Remédio	Tem	Um remédio faz parte de uma ou mais classes de remédio e uma classe pode estar em vários remédios.
Remédio	Principio_ativo	Tem	Um remédio tem vários principios ativos.
Remédio	Classe	Tem	Um remédio faz parte de uma ou mais classes de remédio E uma classe pode estar em vários remédios.
Lote	Fornecedor	Fornece	Fornecedor se encarrega de fornecer um lote de produtos, como maneira de obter lucro em sua atividade.
Lote	Produto	Tem	Todo produto faz parte de um ou mais lotes de produtos.
Lote	Filial	Fornece	Toda Filial recebe um lote de produtos.

Tabela 1: Entidade e Relacionamentos

5.2 Dicionário de Tabelas & Atributos

Logo abaixo, tem-se todas as informações sobre as Tabelas e seus respectivos atributos. Essas informações também estão disponíveis para facilitar o entendimento da Modelagem de Dados.

5.2.1 Tabela e Atributos - Filial

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID da Filial	id	Int	4 bytes	PK NOT NULL	Chave primária de uma filial.
CNPJ da Filial	cnpj	Char	14 bytes	UNIQUE	Número designado pela Receita Federal na abertura da empresa.
Telefone da Filial	telefone	Char	11 bytes		Telefone comercial da filial.
Endereço da Filial	endereco	Varchar	200 bytes		Endereço da localidade da filial.
Site da Filial	site	Varchar	50 bytes		Site para compras online de uma farmácia.

Tabela 2: Tabela e Atributos - filial

5.2.2 Tabela e Atributos - Funcionário

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do Funcionário	id	Int	4 bytes	PK NOT NULL	Chave primária de um funcionário.
CPF do Funcionário	cpf	Char	11 bytes	UNIQUE	Número de documento que identifica o contribuinte perante a Receita Federal.
RG do Funcionário	rg	Char	9 bytes	UNIQUE	Número de documento que identifica o contribuinte perante a SSP.
Nome do Funcionário	nome	Varchar	100 bytes	NOT NULL	Nome completo do funcionário.
Email do Funcionário	email	Varchar	100 bytes	UNIQUE	Email do funcionário.
Endereço do Funcionário	endereco	Varchar	200 bytes	NOT NULL	Endereço residencial do funcionário.
Telefone do Funcionário	telefone	Char	11 bytes		Número de telefone do funcionário.
Certificado do Funcionário	certificado	Boolean	1 byte	NOT NULL	Certificado de curso do funcionário (1 se ele possui e 0 se não possui).
Cargo do Funcionário	cargo	Varchar	15 bytes	NOT NULL	Cargo do funcionário na empresa filial.
Comissão do funcionário	comissao	Decimal	10 bytes	NOT NULL	Comissão fixa recebida pelo funcionário por produto vendido, independente de qual seja.
Salário do funcionário	salario_fixo	Money	8 bytes	NOT NULL	Salário fixo do funcionário.
Salário Mensal do Funcionário	salario_mensal	Money	8 bytes	NOT NULL	Salário mensal (calculado com a comissão) a ser recebido pelo funcionário.
ID da Filial	id_filial	Int	4 bytes	FK NOT NULL	Chave estrangeira para a filial em que o funcionário trabalha.

Tabela 3: Tabela e Atributos - funcionário

5.2.3 Tabela e Atributos - Cliente

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do Cliente	id	Int	4 bytes	PK NOT NULL	Chave primária de um cliente.
CPF do cliente	cpf	Char	11 bytes	UNIQUE	Número do documento que identifica o contribuinte perante a Receita Federal.
Nome do cliente	nome	Varchar	100 bytes	NOT NULL	Nome completo do cliente.
Email do cliente	email	Varchar	100 bytes	UNIQUE	Email do cliente.
Data de nascimento	data_nascimento	Date	4 Bytes		Data de nascimento do cliente.
Endereço do cliente	endereco	Varchar	200 bytes		Endereço do cliente.
Aniversário	status_aniversário	Boolean	1 byte		Variável booleana que marca o aniversário do cliente (1 é aniversário, 0 não é).
Telefone do cliente	telefone	Text	11 bytes		Telefone do cliente.

Tabela 4: Tabela e Atributos - cliente

5.2.4 Tabela e Atributos - Produto

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do produto	id	Int	4 bytes	PK NOT NULL	Chave primária de um produto.
Nome do produto	nome	Varchar	100 bytes	NOT NULL	Nome de um produto da farmácia.
Preço do produto	preco	Money	8 bytes	NOT NULL	Preço para a compra de um produto da farmácia.
Marca de um produto	marca	Varchar	50 bytes	NOT NULL	Marca que criou o produto.
Tipo de remédio	tipo_remedio	Varchar	20 bytes		Tipo que caracteriza o remédio, como Antibiótico, analgésico, etc.
Prescrição médica de um remédio	prescricao_medica	Boolean	1 byte		Atributo booleano para comprovar se o produto necessita de receita.
Tarja de remédio	tipo_tarja	Varchar	20 bytes		As cores das tarjas indicam a classificação de venda do medicamento.
Forma de administração do remédio	forma_de_administracao	Varchar	50 bytes		Diz respeito à forma de usar o remédio. Se deve ser oral, injeção, etc.
Composição do remédio	composicao_remedio	Text	Variável		Serve para especificar se o remédio é original ou genérico.
Data de validade do produto	data_validade	Date	4 bytes	NOT NULL	Data de validade do produto antes de se tornar não utilizável.
Data de fabricação do produto	data_fabricacao	Date	4 bytes	NOT NULL	Data em que o produto foi criado.
Seção de produto	secao	Varchar	10 bytes	NOT NULL	O produto está guardado em alguma seção da filial.

Tabela 5: Tabela e Atributos - produto

5.2.5 Tabela e Atributos - Princípio_ativo

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do princípio ativo	id	Int	4 bytes	PK NOT NULL	Chave primária do princípio ativo de um remédio.
Nome do princípio ativo	nome	Varchar	100 bytes	NOT NULL	Nome do componente farmacologicamente ativo destinado a um medicamento que está presente em algum remédio.

Tabela 6: Tabela e Atributos - princípio_ativo

5.2.6 Tabela e Atributos - tem_venda

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do item$_venda$	id	Int	4 bytes	PK NOT NULL	Chave primária de um item$_venda$.
ID do produto	id_produto	Int	4 bytes	FK NOT NULL	Chave estrangeira do produto, na qual passou por uma determinada venda.
ID da venda	id_venda	Int	4 bytes	FK NOT NULL	Chave estrangeira da venda, na qual vendeu determinado quantidade de um produto.
Quantidade	quantidade	Int	4 bytes	NOT NULL	Quantidade vendida de um determinado produto.
Preço	preco	Money	8 bytes	NOT NULL	Valor referente ao total de um determinada quantidade de um produto.
Desconto	quantidade	Decimal	10 bytes	NOT NULL	Valor referente ao desconto na compra de determinado produto.

Tabela 7: Tabela e Atributos - tem_venda

5.2.7 Tabela e Atributos - Fornece

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID da tabela	fornece	Int	4 bytes	PK NOT NULL	Chave primária para identificar o fornecimento de um lote de produto de um fornecedor
ID do fornecedor	id_fornecedor	Int	4 bytes	FK NOT NULL	Chave estrangeira para relacionar o fonecedor à determinado lote de um produto.
ID do produto	id_produto	Int	4 bytes	FK NOT NULL	Chave estrangeira para relacionar o fornecedor à determinado lote de um determinado produto OBS: id_lote + id_protuto = PK composta em lote
ID do lote	id_lote	Int	4 bytes	FK NOT NULL	Chave estrangeira para relacionar o fornecedor à determinado lote de um determinado produto OBS: id_lote + id_protuto = PK composta em lote

Tabela 8: Tabela e Atributos - fornece

5.2.8 Tabela e Atributos - Produto_tem_principio

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do produto e princípio		Int	0 bytes	PK NOT NULL	Chave primária composta.
ID do produto	id_produto	Int	4 bytes	FK NOT NULL	Chave estrangeira do produto para relacionar com princípio.
ID do princípio	id_principio	Int	4 bytes	FK NOT NULL	Chave estrangeira do principio para relacionar com produto.

Tabela 9: Tabela e Atributos - produto_tem_principio

5.2.9 Tabela e Atributos - Tem_classe

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do produto e classe		Int	0 bytes	PK NOT NULL	Chave primária composta.
ID do produto	id_produto	Int	4 bytes	FK NOT NULL	Chave estrangeira do produto para relacionar com classe.
ID do classe	id_classe	Int	4 bytes	FK NOT NULL	Chave estrangeira de classe para relacionar com produto.

Tabela 10: Tabela e Atributos - tem_classe

5.2.10 Tabela e Atributos - Classe

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID da classe	id	Int	4 bytes	PK NOT NULL	Chave primária da classe de um remédio.
Nome da classe	nome	Varchar	100 bytes	NOT NULL	O nome da classe terapêutica de um medicamento é a categoria que indica para que tipo de doença ou condição ele é indicado.

Tabela 11: Tabela e Atributos - classe

5.2.11 Tabela e Atributos - Fornecedor

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do Fornecedor	id	Int	4 bytes	PK NOT NULL	Chave primária do Fornecedor de uma rede farmacêutica.
CNPJ do Fornecedor	cnpj	Char	14 bytes	UNIQUE	Número designado pela Receita Federal na abertura da empresa.
Nome do Fornecedor	nome	Varchar	100 bytes	NOT NULL	Nome que se refere a empresa fornecedora de produtos.
Telefone do fornecedor	telefone	Char	11 bytes	NOT NULL	Telefone para entrar em contato com o Fornecedor.
Endereço do fornecedor	endereco	Varchar	200 bytes	NOT NULL	Endereço correspondente a posição geográfica da empresa fornecedora.
Portfólio do fornecedor	portfolio	Texto	Variável	NOT NULL	Portfólio se refere a lista de produtos que uma empresa fornecedora fornece.

Tabela 12: Entidade e Atributos - fornecedor

5.2.12 Tabela e Atributos - Venda

Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID da venda	id	Int	4 bytes	PK NOT NULL	Chave primária da venda.
Data de uma venda	data_venda	Date	4 bytes	NOT NULL	Data de quando ocorre uma venda de um funcionário para um cliente.
Valor total da venda	valor_total	Money	8 bytes	NOT NULL	Valor total dos produtos vendidos por um funcionário.
Nota fiscal de uma venda	nfe	Texto	Variável	NOT NULL	Nota fiscal emitida da venda para o cliente.
ID da Filial	id_filial	Int	4 bytes	FK NOT NULL	Chave estrangeira para a filial em que a venda pertence.
ID do funcionário	id_funcionario	Int	4 bytes	FK NOT NULL	Chave estrangeira para o funcionário que realizou a venda.
ID do Cliente	id_cliente	Int	4 bytes	FK NOT NULL	Chave estrangeira para o cliente que realizou a compra.

Tabela 13: Tabela e Atributos - venda

5.2.13 Tabela e Atributos - Lote

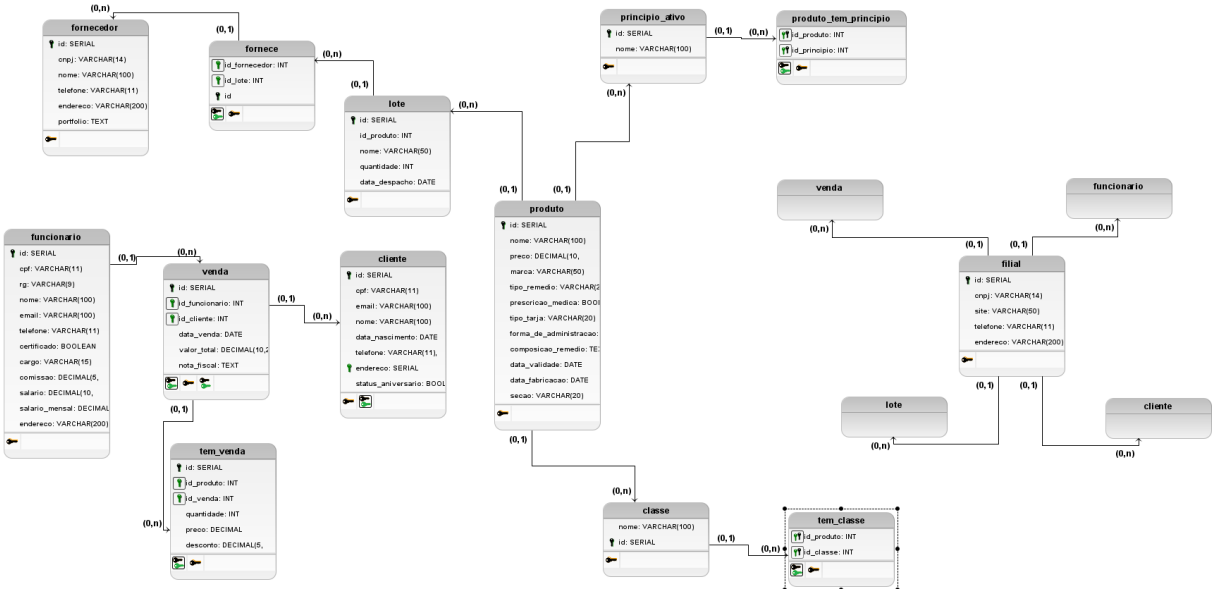
Atributo	Nome do Atributo	Tipo do Atributo	Comprimento	Restrições	Descrição
ID do lote	id	Int	4 bytes	PK NOT NULL	Chave primária do lote de um produto.
Nome do lote	nome	Varchar	100 bytes	NOT NULL	Nome designado para o lote de produtos.
Quantidade do lote	quantidade	Int	4 bytes	NOT NULL	Quantidade de itens dentro de um lote de produtos.
Data de despacho	data	Date	4 bytes	NOT NULL	Data de despacho de uma entrega de determinado lote a uma filial.
ID do produto	id_produto	Int	4 bytes	FK NOT NULL	Chave estrangeira do produto que está dentro do lote.
ID da Filial	id_filial	Int	4 bytes	FK NOT NULL	Chave estrangeira para a filial em que o lote pertence.

Tabela 14: Tabela e Atributos - lote

6 Modelo Lógico

Um modelo lógico é uma representação gráfica de processos que descreve a teoria e o raciocínio de um programa. Ele estabelece a estrutura dos dados e seus relacionamentos, independentemente do banco de dados físico. Os componentes principais são entidades (coisas, pessoas ou conceitos), relacionamentos (associações entre entidades) e atributos (informações descritivas das entidades). O modelo lógico é uma ferramenta essencial na organização dos trabalhos de avaliação e pode ser desenvolvido independentemente do sistema de gerenciamento de banco de dados.

Figura 13: DER - Rede Farmacêutica

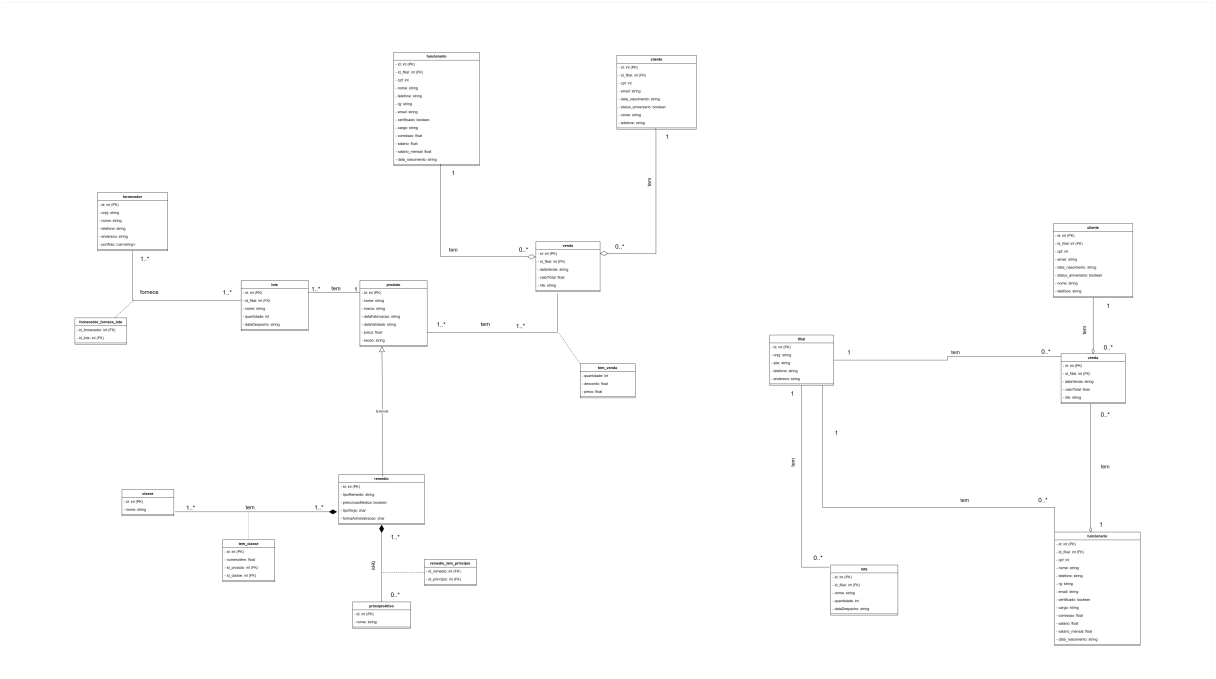


Fonte: imagem do computador

7 Diagrama de Classes

Um diagrama de classe é uma representação gráfica que descreve a estrutura e as relações entre as classes em um sistema orientado a objetos. Ele ilustra as classes do sistema, seus atributos, métodos e como elas estão interconectadas. Sendo assim, dada o contexto de projeto, abaixo está o diagrama de classes referente ao sistema.

Figura 15: Diagrama de Classe



Fonte: imagem do computador

8 Tabelas - SQL

Os códigos SQL que criarão todas as tabelas com as suas respectivas características, de acordo com o modelo relacional do Projeto estão abaixo:

```sql
-- Cria a tabela 'filial'
CREATE TABLE filial (
    id SERIAL,
    cnpj VARCHAR(14) UNIQUE,
    site VARCHAR(50),
    telefone VARCHAR(15),
    endereco VARCHAR(200),
    CONSTRAINT pk_id_filial PRIMARY KEY (id)
);

COMMENT ON TABLE filial IS 'Entidade do DER que representa uma filial.';
COMMENT ON COLUMN filial.id IS 'Chave primária de uma filial.';
COMMENT ON COLUMN filial.cnpj IS 'Número único designado pela Receita Federal na
    abertura legal de uma empresa.';
COMMENT ON COLUMN filial.site IS 'Site para compras online de uma farmácia.';
COMMENT ON COLUMN filial.telefone IS 'Telefone comercial da Filial.';
COMMENT ON COLUMN filial.endereco IS 'Endereço da localidade da filial.';

-- Cria a tabela 'fornecedor'
CREATE TABLE fornecedor (
    id SERIAL,
    cnpj CHAR(14) UNIQUE,
    nome VARCHAR(100) NOT NULL,
    telefone CHAR(11) NOT NULL,
    endereco VARCHAR(200),
    portfolio TEXT NOT NULL,
    CONSTRAINT pk_id_fornecedor PRIMARY KEY (id)
);
```

```
COMMENT ON TABLE fornecedor IS 'Entidade do DER que representa um fornecedor.';
COMMENT ON COLUMN fornecedor.id IS 'Chave primária de um fornecedor.';
COMMENT ON COLUMN fornecedor.cnpj IS 'Número único designado pela Receita Federal
    na abertura legal de uma empresa.';
COMMENT ON COLUMN fornecedor.nome IS 'Nome que se refere a empresa fornecedora de
    produtos.';
COMMENT ON COLUMN fornecedor.telefone IS 'Telefone para entrar em contato com o
    fornecedor.';
COMMENT ON COLUMN fornecedor.endereco IS 'Endereço correspondente a posição geográ
    fica da empresa fornecedora.';
COMMENT ON COLUMN fornecedor.portfolio IS 'Portfólio se refere a lista de produtos
     que uma empresa fornecedora fornece.';

-- Cria a tabela 'funcionario' entidade do DER.
CREATE TABLE funcionario (
    id SERIAL,
    cpf CHAR(11) UNIQUE,
    rg CHAR(9) UNIQUE,
    nome CHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    telefone CHAR(11),
    certificado BOOLEAN NOT NULL,
    cargo VARCHAR(15) NOT NULL,
    comissao DECIMAL(5, 2) NOT NULL,
    salario_fixo MONEY NOT NULL CHECK (salario_fixo > CAST(0.00 AS MONEY)),
    salario_mensal MONEY NOT NULL,
    id_filial INT NOT NULL,
    endereco VARCHAR(200) NOT NULL,
    CONSTRAINT pk_id_funcionario PRIMARY KEY (id),
    CONSTRAINT fk_id_filial_in_funcionario FOREIGN KEY (id_filial) REFERENCES
        filial (id)
);

COMMENT ON TABLE funcionario IS 'Entidade do DER que representa um funcionário.';
COMMENT ON COLUMN funcionario.id IS 'Chave primária de um funcionário.';
COMMENT ON COLUMN funcionario.cpf IS 'Número de documento que identifica o
    contribuinte perante a Receita Federal.';
COMMENT ON COLUMN funcionario.rg IS 'Número de documento que identifica o
    contribuinte perante a SSP.';
COMMENT ON COLUMN funcionario.nome IS 'Nome completo do funcionário.';
COMMENT ON COLUMN funcionario.email IS 'E-mail do funcionário.';
COMMENT ON COLUMN funcionario.telefone IS 'Número de telefone do funcionário.';
COMMENT ON COLUMN funcionario.certificado IS 'Certificado de curso do funcionário
    (1 se ele possui e 0 se não possui).';
COMMENT ON COLUMN funcionario.cargo IS 'Cargo do funcionário na empresa filial.';
COMMENT ON COLUMN funcionario.comissao IS 'Comissão fixa recebida pelo funcionário
     por produto vendido, independente de qual seja.';
COMMENT ON COLUMN funcionario.salario_fixo IS 'Salário fixo do funcionário.';
COMMENT ON COLUMN funcionario.salario_mensal IS 'Salário mensal (calculado com a
    comissão) a ser recebido pelo funcionário.';
COMMENT ON COLUMN funcionario.id_filial IS 'Chave estrangeira para a filial em que
     o funcionário trabalha.';
COMMENT ON COLUMN funcionario.endereco IS 'Endereço residencial do funcionário.';


-- Cria a tabela 'cliente' entidade do DER.
CREATE TABLE cliente (
    id SERIAL,
    cpf CHAR(11) UNIQUE,
```

```
    email VARCHAR(100) UNIQUE,
    nome VARCHAR(100) NOT NULL,
    data_nascimento DATE,
    telefone CHAR(11),
    endereco VARCHAR(200),
    status_aniversario BOOL DEFAULT FALSE,
    CONSTRAINT pk_id_cliente PRIMARY KEY (id)
);

COMMENT ON TABLE cliente IS 'Entidade do DER que representa um cliente.';
COMMENT ON COLUMN cliente.id IS 'Chave primária de um cliente.';
COMMENT ON COLUMN cliente.cpf IS 'Número do documento que identifica o
    contribuinte perante a Receita Federal.';
COMMENT ON COLUMN cliente.email IS 'Email do cliente.';
COMMENT ON COLUMN cliente.nome IS 'Nome completo do cliente.';
COMMENT ON COLUMN cliente.data_nascimento IS 'Data de nascimento do cliente.';
COMMENT ON COLUMN cliente.telefone IS 'Telefone do cliente.';
COMMENT ON COLUMN cliente.endereco IS 'Endereço do cliente.';
COMMENT ON COLUMN cliente.status_aniversario IS 'Verifica se é aniversário do
    cliente.';

-- Cria a tabela 'venda' entidade do DER.
CREATE TABLE venda (
    id SERIAL,
    id_funcionario INT NOT NULL,
    id_cliente INT NOT NULL,
    data_venda DATE NOT NULL DEFAULT CURRENT_DATE,
    valor_total MONEY NOT NULL,
    nota_fiscal TEXT NOT NULL,
    id_filial INT NOT NULL,
    CONSTRAINT pk_id_venda PRIMARY KEY (id),
    CONSTRAINT fk_id_funcionario_in_venda FOREIGN KEY (id_funcionario) REFERENCES
        funcionario (id),
    CONSTRAINT fk_id_cliente_in_venda FOREIGN KEY (id_cliente) REFERENCES cliente
        (id),
    CONSTRAINT fk_id_filial_in_venda FOREIGN KEY (id_filial) REFERENCES filial (id
        )
);

COMMENT ON TABLE venda IS 'Entidade do DER que representa uma venda.';
COMMENT ON COLUMN venda.id IS 'Chave primária de uma venda.';
COMMENT ON COLUMN venda.id_funcionario IS 'Chave estrangeira para o funcionário
    que realizou a venda.';
COMMENT ON COLUMN venda.id_cliente IS 'Chave estrangeira para o cliente que
    realizou a compra.';
COMMENT ON COLUMN venda.data_venda IS 'Data de quando ocorre uma venda de um
    funcionário para um cliente.';
COMMENT ON COLUMN venda.valor_total IS 'Valor total dos produtos vendidos por um
    funcionário.';
COMMENT ON COLUMN venda.nota_fiscal IS 'Nota fiscal emitida da venda para o
    cliente.';
COMMENT ON COLUMN venda.id_filial IS 'Chave estrangeira para a filial em que a
    venda pertence.';


-- Cria a tabela 'produto' entidade DER.
CREATE TABLE produto (
    id SERIAL,
    nome VARCHAR(100) NOT NULL,
    preco MONEY NOT NULL CHECK (preco > CAST(0.00 AS MONEY)),
```

```
    marca VARCHAR(50) NOT NULL,
    tipo_remedio VARCHAR(20),
    prescricao_medica BOOLEAN,
    tipo_tarja VARCHAR(20),
    forma_de_administracao VARCHAR(50),
    composicao_remedio TEXT,
    data_validade DATE NOT NULL,
    data_fabricacao DATE NOT NULL,
    secao VARCHAR(20) NOT NULL,
    CONSTRAINT pk_id_produto PRIMARY KEY (id)
);

COMMENT ON TABLE produto IS 'Entidade DER que representa um produto.';
COMMENT ON COLUMN produto.id IS 'Chave primária de um produto.';
COMMENT ON COLUMN produto.nome IS 'Nome de um produto da farmácia.';
COMMENT ON COLUMN produto.preco IS 'Preço para a compra de um produto da farmácia.
    ';
COMMENT ON COLUMN produto.marca IS 'Marca que criou o produto.';
COMMENT ON COLUMN produto.tipo_remedio IS 'Tipo que caracteriza o remédio, como
    antibiótico ou analgésico.';
COMMENT ON COLUMN produto.prescricao_medica IS 'Atributo booleano para comprovar
    se o produto necessita de receita.';
COMMENT ON COLUMN produto.tipo_tarja IS 'As cores das tarjas indicam a classificaç
    ão de venda do medicamento.';
COMMENT ON COLUMN produto.forma_de_administracao IS 'Diz respeito a forma de usar
    o remédio, se deve ser oral, injeção, etc.';
COMMENT ON COLUMN produto.composicao_remedio IS 'Serve para especificar se o remé
    dio é original ou genérico.';
COMMENT ON COLUMN produto.data_validade IS 'Data de validade do produto antes de
    se tornar não utilizável.';
COMMENT ON COLUMN produto.data_fabricacao IS 'Data em que o produto foi criado.';
COMMENT ON COLUMN produto.secao IS 'O produto está guardado em alguma seção da
    filial.';

-- Cria a tabela 'lote' entidade do DER.
CREATE TABLE lote (
    id SERIAL,
    id_produto INT NOT NULL,
    nome VARCHAR(50) NOT NULL,
    quantidade INT NOT NULL,
    data_despacho DATE NOT NULL DEFAULT CURRENT_DATE,
    id_filial INT NOT NULL,
    CONSTRAINT pk_composta_lote_produto PRIMARY KEY (id, id_produto),
    FOREIGN KEY (id_produto) REFERENCES produto (id),
    CONSTRAINT fk_id_filial_in_lote FOREIGN KEY (id_filial) REFERENCES filial (id)
);

COMMENT ON TABLE lote IS 'Entidade do DER que representa um lote de produtos.';
COMMENT ON COLUMN lote.id IS 'Chave primária de um lote.';
COMMENT ON COLUMN lote.id_produto IS 'Chave estrangeira do produto que está dentro
     do lote.';
COMMENT ON COLUMN lote.nome IS 'Nome designado para o lote de produtos.';
COMMENT ON COLUMN lote.quantidade IS 'Quantidade de itens dentro de um lote de
    produtos.';
COMMENT ON COLUMN lote.data_despacho IS 'Data de despacho de uma entrega de
    determinado lote a uma filial.';
COMMENT ON COLUMN lote.id_filial IS 'Chave estrangeira para a filial em que o lote
     pertence.';

-- Cria a tabela 'principio_ativo' entidade do DER.
```

```
CREATE TABLE principio_ativo (
    id SERIAL,
    nome VARCHAR(100) NOT NULL,
    CONSTRAINT pk_id_principio_ativo PRIMARY KEY (id)
);

COMMENT ON TABLE principio_ativo IS 'Entidade do DER que representa um princípio
    ativo de medicamentos.';
COMMENT ON COLUMN principio_ativo.id IS 'Chave primária de um princípio ativo.';
COMMENT ON COLUMN principio_ativo.nome IS 'Nome do componente farmacologicamente
    ativo destinado a um medicamento.';

-- Cria a tabela 'tem_venda' que vem do relacionamento N-M entre [venda] e [
    produto].
CREATE TABLE tem_venda (
    id SERIAL,
    id_produto INT NOT NULL,
    id_venda INT NOT NULL,
    quantidade INT NOT NULL,
    preco MONEY NOT NULL,
    desconto DECIMAL(5, 2),
    CONSTRAINT pk_id_tem_venda PRIMARY KEY (id),
    CONSTRAINT uq_produto_venda UNIQUE (id_produto, id_venda),
    CONSTRAINT fk_id_venda_in_tem_venda FOREIGN KEY (id_venda) REFERENCES venda (
        id),
    CONSTRAINT fk_id_produto_in_tem_venda FOREIGN KEY (id_produto) REFERENCES
        produto (id)
);

COMMENT ON TABLE tem_venda IS 'Tabela que vem do relacionamento N-M entre venda e
    produto.';
COMMENT ON COLUMN tem_venda.id IS 'Chave primária de um tem_venda.';
COMMENT ON COLUMN tem_venda.id_produto IS 'Chave estrangeira do produto, na qual
    passou por uma determinada venda.';
COMMENT ON COLUMN tem_venda.id_venda IS 'Chave estrangeira da venda, na qual
    vendeu determinada quantidade de um produto.';
COMMENT ON COLUMN tem_venda.quantidade IS 'Quantidade vendida de um determinado
    produto.';
COMMENT ON COLUMN tem_venda.preco IS 'Valor referente ao total de uma determinada
    quantidade de um produto.';
COMMENT ON COLUMN tem_venda.desconto IS 'Valor referente ao desconto na compra de
    determinado produto.';

-- Cria a tabela 'fornece' que vem do relacionamento N-M entre [fornecedor] e [
    lote].
CREATE TABLE fornece (
    id SERIAL,
    id_fornecedor INT NOT NULL,
        id_produto INT NOT NULL,
    id_lote INT NOT NULL,
    CONSTRAINT pk_id_fornece PRIMARY KEY (id),
    CONSTRAINT fk_id_fornecedor_in_fornece FOREIGN KEY (id_fornecedor) REFERENCES
        fornecedor (id),
        CONSTRAINT fk_id_produto_in_fornece FOREIGN KEY (id_lote, id_produto)
            REFERENCES lote (id, id_produto)

);

COMMENT ON TABLE fornece IS 'Tabela que vem do relacionamento N-M entre fornecedor
     e lote.';
```

```
COMMENT ON COLUMN fornece.id IS 'Chave primária para identificar o fornecimento de
     um lote de produto de um fornecedor.';
COMMENT ON COLUMN fornece.id_fornecedor IS 'Chave estrangeira para relacionar o
    fornecedor a determinado lote de um produto.';
COMMENT ON COLUMN fornece.id_produto IS 'Chave estrangeira para relacionar o
    fornecedor a determinado lote de um produto.';
COMMENT ON COLUMN fornece.id_lote IS 'Chave estrangeira para relacionar o
    fornecedor a determinado lote de um produto.';

-- Cria a tabela 'produto_tem_principio' que vem do relacionamento N-M entre [
    produto] e [principio_ativo].
CREATE TABLE produto_tem_principio (
    id_produto INT NOT NULL,
    id_principio INT NOT NULL,
    CONSTRAINT pk_id_produto_tem_principio PRIMARY KEY (id_produto, id_principio),
    CONSTRAINT fk_id_produto_tem_principio FOREIGN KEY (id_produto) REFERENCES
        produto (id),
    CONSTRAINT fk_id_principio_tem_principio FOREIGN KEY (id_principio) REFERENCES
         principio_ativo (id)
);

COMMENT ON TABLE produto_tem_principio IS 'Tabela que vem do relacionamento N-M
    entre produto e princípio ativo.';
COMMENT ON COLUMN produto_tem_principio.id_produto IS 'Chave estrangeira do
    produto para relacionar com princípio.';
COMMENT ON COLUMN produto_tem_principio.id_principio IS 'Chave estrangeira do
    principio para relacionar com produto.';

-- Cria a tabela 'classe' entidade do DER.
CREATE TABLE classe (
    id SERIAL,
    nome VARCHAR(100),
    CONSTRAINT pk_id_classe PRIMARY KEY (id)
);

COMMENT ON TABLE classe IS 'Tabela que representa a classe terapêutica de um
    medicamento.';
COMMENT ON COLUMN classe.id IS 'Chave primária da classe de um remédio.';
COMMENT ON COLUMN classe.nome IS 'O nome da classe terapêutica de um medicamento,
    indica para que tipo de doença ou condição é indicado.';

-- Cria a tabela 'tem_classe' que vem do relacionamento N-M entre [produto] e [
    classe].
CREATE TABLE tem_classe(
    id_produto INT NOT NULL,
    id_classe INT NOT NULL,
    CONSTRAINT pk_id_tem_classe PRIMARY KEY (id_produto, id_classe),
    CONSTRAINT fk_id_produto_in_tem_classe FOREIGN KEY (id_produto) REFERENCES
        produto (id),
    CONSTRAINT fk_id_classe_in_tem_classe FOREIGN KEY (id_classe) REFERENCES
        classe (id)
);

COMMENT ON TABLE tem_classe IS 'Tabela que vem do relacionamento N-M entre produto
     e classe.';
COMMENT ON COLUMN tem_classe.id_produto IS 'Chave estrangeira do produto para
    relacionar com classe.';
COMMENT ON COLUMN tem_classe.id_classe IS 'Chave estrangeira de classe para
    relacionar com produto.';
```

9 Index - Melhoria de Consultas

A otimização das consultas no banco de dados é crucial para melhorar o desempenho e a eficiência das operações de busca. Duas estratégias principais de indexação são índices Hash e índices B-tree. Cada tipo tem suas vantagens específicas dependendo do tipo de consulta executada.

- **Como usar e porquê usar Index Hash:**

  Índices Hash são utilizados para operações de busca por igualdade. Eles oferecem uma performance de busca rápida e consistente, ideal para colunas com valores únicos como e-mail e telefone.

- **Como usar e porquê usar Index B-tree:**

  Índices B-tree são versáteis e suportam busca por igualdade, buscas de intervalo e ordenações. Eles são recomendados para a maioria das colunas, especialmente aquelas usadas em relações de chave estrangeira e consultas que envolvem ordenação ou intervalos de datas.

9.1 Index - Hash

Índices Hash são recomendados para as seguintes colunas:

```
-- Buscas por email são essenciais para autenticação e identificação rápida de usu
    ários, tornando índices Hash ideais para acelerar essas operações.

CREATE INDEX idx_hash_cliente_email ON cliente USING HASH (email);
CREATE INDEX idx_hash_funcionario_email ON funcionario USING HASH (email);

-- Operações podem envolver buscar por número de telefone, especialmente em
    sistemas de atendimento ao cliente ou gestão de RH.

CREATE INDEX idx_hash_cliente_telefone ON cliente USING HASH (telefone);
CREATE INDEX idx_hash_funcionario_telefone ON funcionario USING HASH (telefone);
```

9.2 Index - B-tree

Índices B-tree são recomendados para as seguintes colunas:

```
-- Facilita as consultas que buscam vendas em um determinado intervalo de tempo,
    seja um dia específico, mês, ano ou entre datas.

CREATE INDEX idx_btree_venda_data ON venda(data_venda);

-- Otimizar a busca por registros de atendimento em determinadas datas.

CREATE INDEX idx_btree_atende_data ON atende(data_atendimento);

-- Permite consultas rápidas para identificar produtos próximos do vencimento,
    facilitando ações de promoção, realocação ou descarte, essenciais na gestão de
     uma farmácia.

CREATE INDEX idx_btree_produto_validade ON produto(data_validade);

-- Pode ser útil para promoções baseadas em aniversário.

CREATE INDEX idx_btree_cliente_data_nascimento ON cliente(data_nascimento);

-- Nome de Fornecedor, Cliente e Funcionário para Buscas e Ordenações.
```

```
CREATE INDEX idx_btree_fornecedor_nome ON fornecedor(nome);
CREATE INDEX idx_btree_cliente_nome ON cliente(nome);
CREATE INDEX idx_btree_funcionario_nome ON funcionario(nome);

-- Filtros de busca em catálogos de produtos, especialmente em um contexto de e-
    commerce ou gestão de estoque.

CREATE INDEX idx_btree_produto_marca ON produto(marca);
CREATE INDEX idx_btree_produto_tipo ON produto(tipo_remedio);
```

9.3 Peso - Index

Conhecer o peso dos índices em um banco de dados é crucial para otimizar o desempenho das consultas, gerenciar eficientemente o espaço de armazenamento e entender a sobrecarga em operações de escrita. Índices eficazes melhoram significativamente as leituras, mas também podem consumir recursos consideráveis, impactando o planejamento de capacidade e a manutenção do sistema.

A análise do tamanho dos índices auxilia na previsão de necessidades futuras de hardware e na otimização de recursos, garantindo escalabilidade e eficiência operacional conforme o volume de dados aumenta.

Index	**Ocupação (bytes)**
filial	38
fornecedor	98
funcionario	269
cliente	250
venda	70
produto	93
lote	32
principio_ativo	14
atende	14
tem_venda	60
fornece	46
produto_tem_principio	32
classe	14
tem_classe	32
Total	**1062**

Tabela 15: Pesos totais de cada Index implementado

10 Ocupação em Disco

A análise da ocupação em disco em relação à modelagem de banco de dados é crucial por várias razões. Primeiramente, uma modelagem eficiente do banco de dados otimiza o espaço em disco, importante em ambientes onde armazenamento é limitado ou caro. Além disso, um banco de dados bem projetado melhora o desempenho das consultas e transações, reduzindo operações de I/O e minimizando tempo de acesso aos dados. Isso também contribui para a escalabilidade do banco de dados, lidando melhor com o crescimento dos dados ao longo do tempo.

Uma modelagem que considera a ocupação em disco facilita operações de manutenção e backup, além de simplificar a administração do banco de dados. Identificar áreas de subutilização de recursos ajuda a economizar recursos de hardware e reduzir custos operacionais. Em resumo, essa análise garante eficiência, desempenho, escalabilidade, facilidade de gerenciamento e economia de recursos no sistema de banco de dados.

10.1 Ocupação Base da Modelagem

Aqui, neste tópico o objetivo é calcular o peso em bytes de cada tabela, de acordo com os tipos de dados usados, além de fazer estimativas e construir projeções de consumo de armazenamento.

- Observação: a informação sobre ocupação em bytes de cada tipo de dado foi obtida diretamente da documentação do PostgreSQL.

10.1.1 Tabela Filial - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
cnpj	CHAR(14)	14
telefone	CHAR(11)	11
endereco	VARCHAR(200)	200
site	VARCHAR(50)	50
Total	**Valor**	**279**

Tabela 16: Peso da tabela Filial

10.1.2 Tabela Fornecedor - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
cnpj	CHAR(14)	14
nome	VARCHAR(100)	100
telefone	CHAR(11)	11
endereco	VARCHAR(200)	200
portfolio	TEXT	variável
Total	**Valor**	**329**

Tabela 17: Peso da tabela fornecedor

10.1.3 Tabela Funcionário - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
cpf	CHAR(11)	11
rg	CHAR(9)	9
nome	VARCHAR(100)	100
email	VARCHAR(100)	100
telefone	CHAR(11)	11
certificado	BOOLEAN	1
cargo	VARCHAR(15)	15
comissao	DECIMAL(5,2)	10
salario_fixo	MONEY	8
salario_mensal	MONEY	8
id_filial	INTEGER	4
Total	**Valor**	**281**

Tabela 18: Peso da tabela funcionário

10.1.4 Tabela Cliente - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
cpf	CHAR(11)	11
nome	VARCHAR(100)	100
email	VARCHAR(100)	100
data_nascimento	DATE	4
status_aniversario	BOOL	1
telefone	CHAR(11)	11
Total	**Valor**	**231**

Tabela 19: Peso da tabela cliente

10.1.5 Tabela Venda - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
id_funcionario	INT	4
id_cliente	INT	4
data_venda	DATE	4
valor_total	MONEY	8
nota_fiscal	TEXT	Variável
id_filial	INT	4
Total	**Valor**	**28**

Tabela 20: Peso da tabela venda

10.1.6 Tabela Produto - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
nome	VARCHAR(100)	100
preco	MONEY	8
marca	VARCHAR(50)	50
tipo_remedio	VARCHAR(20)	20
prescricao_medica	BOOLEAN	1
tipo_tarja	VARCHAR(20)	20
forma_de_administracao	VARCHAR(50)	50
composicao_remedio	TEXT	Variável
data_validade	DATE	4
data_fabricacao	DATE	4
secao	VARCHAR(10)	10
Total	**Valor**	**271**

Tabela 21: Peso da tabela produto

10.1.7 Tabela Lote - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
id_produto	INT	4
nome	VARCHAR(100)	100
quantidade	INT	4
data_despacho	DATE	4
id_filial	INT	4
Total	**Valor**	**120**

Tabela 22: Peso da tabela lote

10.1.8 Tabela Principio_ativo - Peso

Campo	Tipo	Ocupação (bytes)
id	SERIAL	4
nome	VARCHAR(100)	100
Total	**Valor**	**104**

Tabela 23: Peso da tabela principio_ativo

10.1.9 Tabela Tem_venda - Peso

Campo	Tipo	Ocupação (bytes)
id	INT	4
id_venda	INT	4
id_produto	INT	4
quantidade	INT	4
preco	MONEY	8
desconto	DECIMAL(5,2)	10
Total	**Valor**	**34**

Tabela 24: Peso da tabela tem_venda

10.1.10 Tabela Fornece - Peso

Campo	Tipo	Ocupação (bytes)
id	INT	4
id_fornecedor	INT	4
id_lote	INT	4
id_produto	INT	4
Total	**Valor**	**16**

Tabela 25: Peso da tabela fornece

10.1.11 Tabela Produto_tem_principio - Peso

Campo	Tipo	Ocupação (bytes)
id	INT	4
id_venda	INT	4
id_principio	INT	4
Total	**Valor**	**12**

Tabela 26: Peso da tabela produto_tem_principio

10.1.12 Tabela Classe - Peso

Campo	Tipo	Ocupação (bytes)
id	INT	4
nome	VARCHAR(100)	100
Total	**Valor**	**104**

Tabela 27: Peso da tabela classe

10.1.13 Tabela Tem_classe - Peso

Campo	Tipo	Ocupação (bytes)
id	INT	4
id_produto	INT	4
id_classe	INT	4
Total	**Valor**	**12**

Tabela 28: Peso da tabela tem_classe

10.1.14 Peso Total

Logo abaixo, segue-se a uma tabela que relaciona a ocupação de cada "table"em bytes, no que diz respeito a pelo menos 1 única tupla, em relação ao total de ocupação da modelagem.

Tabela	Ocupação (bytes)
Filial	279
Fornecedor	329
Funcionario	281
Cliente	231
Venda	40
Produto	271
Lote	120
Principio_ativo	104
Tem_venda	34
Fornece	16
Produto_tem_principio	12
Classe	104
Tem_classe	12
Total	**1833**

Tabela 29: Pesos totais de cada tabela

Agora, somando-se o peso de pelo menos 1 tupla de cada tabela estima-se que o total seja: 1906 bytes aproximadamente (não levando em consideração a ocupação dos tipos variáveis).

10.2 Projeção para Implantação - MUDAR

OBS: projeção para 1 ano e 2 anos.

10.2.1 Projeção de 1 ano de operação

À medida que a demanda pelos serviços da LAV crescia, também crescia a visão de seus fundadores. Reconhecendo a necessidade de tornar os cuidados de saúde mais acessíveis em todo o país, a LAV iniciou um ambicioso plano de expansão. Inicialmente com 6 filiais, cada uma refletindo o compromisso da marca com a qualidade e a inovação, a farmácia não apenas aumentou sua presença no mercado, mas também solidificou sua reputação como líder no setor farmacêutico.

A decisão de expandir para mais 4 filiais foi impulsionada por uma visão estratégica e pela missão contínua da LAV de promover saúde e bem-estar em todo o país. Analisando cuidadosamente os dados demográficos e as necessidades das comunidades não atendidas, a LAV identificou locais estratégicos para suas novas filiais. Essa expansão não apenas permitiria à LAV atender a uma base de clientes maior e mais diversificada, mas também estabeleceria a marca como uma figura dominante no cenário nacional de farmácias.

Com a expansão, veio a necessidade de uma infraestrutura tecnológica robusta para suportar as operações crescentes. Cada filial da LAV foi equipada com um sistema de gerenciamento avançado, permitindo a eficiência operacional e a continuidade do serviço de alta qualidade pelo qual a LAV é conhecida. No entanto, isso também significou um aumento nos requisitos de armazenamento de dados.

Originalmente, com 6 filiais, o armazenamento base necessário para apenas as tabelas de dados era de 4044 bytes por filial, sendo 24.264 bytes ao total. Ao expandir para 10 filiais, a Farmácia LAV enfrentou o desafio de escalar sua infraestrutura de dados para acomodar o aumento no volume de informações.

Com a expansão para 10 filiais, o armazenamento base necessário para apenas as tabelas de dados nas filiais da Farmácia LAV seria de 40,440 bytes. Esta expansão reflete o compromisso da LAV em levar cuidados de saúde de qualidade e acessíveis a mais pessoas em todo o país, reforçando sua posição como uma marca líder no setor farmacêutico.

10.2.2 Projeção de 2 anos de operação

A expansão da farmácia LAV para **10 filiais** com **400 produtos** diferentes em cada uma resultou em um armazenamento total de **5.272.000 bytes** no banco de dados. Com a decisão estratégica de dobrar o número de produtos para 800 em cada filial, o armazenamento necessário aumentará para **10.544.000 bytes**. Esta expansão significativa reflete não apenas o crescimento físico das filiais, mas também a diversificação dos produtos oferecidos.

A expansão para novos produtos foi impulsionada por uma série de parcerias estratégicas com fornecedores locais e internacionais, marcando uma nova era para a farmácia LAV. A busca por uma variedade maior de produtos surgiu da necessidade de atender a uma demanda crescente por soluções de saúde mais personalizadas e abrangentes. A LAV reconheceu a importância de oferecer aos seus clientes uma gama mais ampla de opções de saúde e bem-estar, desde medicamentos de prescrição até suplementos naturais e itens de cuidados pessoais.

As novas parcerias foram estabelecidas com o objetivo de incorporar inovações no campo da saúde, acessar novos mercados e fortalecer a cadeia de suprimentos. Com fornecedores especializados em produtos orgânicos, medicamentos de alta tecnologia e soluções de saúde digital, a LAV pôde diversificar seu portfólio de produtos, atendendo às necessidades emergentes de seus clientes. Além disso, essas colaborações permitiram à LAV oferecer produtos exclusivos, diferenciando-a de outras farmácias e consolidando sua presença no mercado.

10.3 Crescimento dos Dados - MUDAR

A avaliação do recurso de hardware ideal para armazenar o conteúdo de cada tabela, considerando as regras de negócio da modelagem para a farmácia LAV, deve levar em conta a frequência de operações CRUD (Create,

Read, Update, Delete) que cada tabela sofrerá. Tabelas sujeitas a muitas alterações exigem sistemas de armazenamento que ofereçam alta performance em escrita e leitura, enquanto tabelas com poucas alterações podem ser armazenadas em sistemas mais econômicos, priorizando a estabilidade e os custos de armazenamento a longo prazo.

10.3.1 Tabelas com Poucas Alterações (CRUD)

- **Filial e Fornecedor** Estas tabelas provavelmente terão poucas alterações após a inserção inicial. Filiais e fornecedores são elementos estáveis na operação da farmácia. **Recurso Ideal:** HDDs (Hard Disk Drives) são adequados para este tipo de armazenamento, oferecendo um custo-benefício vantajoso para dados que não exigem frequente atualização ou acesso rápido.
- **Principio ativo e classe** São tabelas que definem categorias ou características essenciais dos produtos, com alterações relativamente raras. **Recurso Ideal:** Armazenamento em HDDs (Hard Disk Drives) de alta durabilidade, considerando que o acesso frequente será predominantemente para leitura.

10.3.2 Tabelas Sujeitas a Muitas Alterações

- **Funcionario, Cliente, Venda, e Atende** Estas tabelas estão no coração das operações diárias e sofrerão constantes inserções e atualizações. **Recurso Ideal:** SSDs de alta performance para suportar um grande volume de transações e acessos rápidos, minimizando latências em operações de escrita e leitura.
- **Produto, Lote, Tem_venda, e Fornece** Inserções, atualizações e consultas serão frequentes devido à natureza dinâmica do estoque, vendas e relacionamentos com fornecedores. **Recurso Ideal:** SSDs de alta performance, possivelmente com tecnologia NVMe (Non-Volatile Memory express) para maximizar a velocidade de acesso e processamento de transações.
- **Produto_tem_principio e Tem_classe** Estas tabelas gerenciam as relações muitos-para-muitos entre produtos, princípios ativos e classes, podendo sofrer alterações frequentes com a introdução de novos produtos ou reclassificação. **Recurso Ideal:** Armazenamento em SSD, proporcionando o equilíbrio entre desempenho para leitura e escrita rápida de dados relacionais complexos.

10.3.3 Tabela de crescimento de dados

A tabela venda possui um peso menor em relação a tabela cliente, por isso existe uma discrepância entre elas, mesmo as duas tendo a mesma quantidade de insert's.

10.3.4 Tabela de Projeção de 1 e 2 anos

10.3.5 Grafico de linhas e barras de crescimento de dados

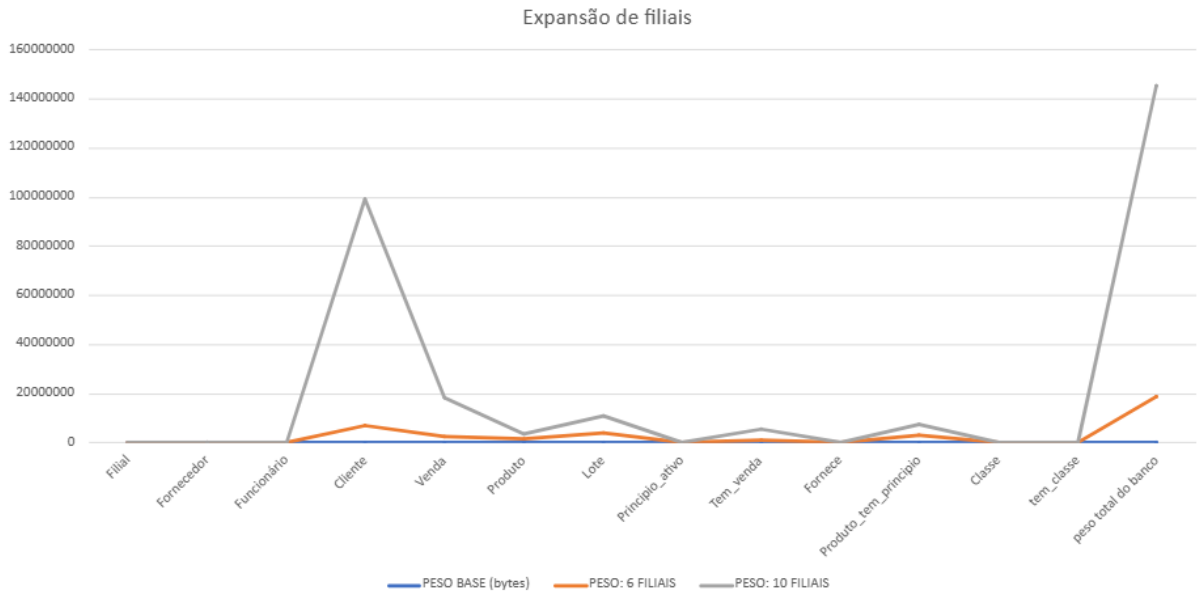
Figura 13: Grafico de linhas - Crescimento do peso do banco por tabela.

TABELA	PESO BASE (bytes)	PESO: 6 FILIAIS	PESO: 10 FILIAIS
Filial	279	1.674	2.790
Fornecedor	329	2.331	23.330
Funcionário	281	10.980	28.100
Cliente	231	6.930.000	92.400.000
Venda	40	2.400.000	16.000.000
Produto	271	1.377.600	2.168.000
Lote	120	4.032.000	6.720.000
Principio_ativo	104	32.400	52.000
Tem_venda	44	1.120.000	4.400.000
Fornece	12	3.528	5.880
Produto_tem_principio	12	2.880.000	4.800.000
Classe	104	7.776	12.480
tem_classe	12	57.600	96.000
peso total do banco	**1.839**	**18.855.889**	**126.708.280**

Tabela 30: Peso dos dados em diferentes cenários

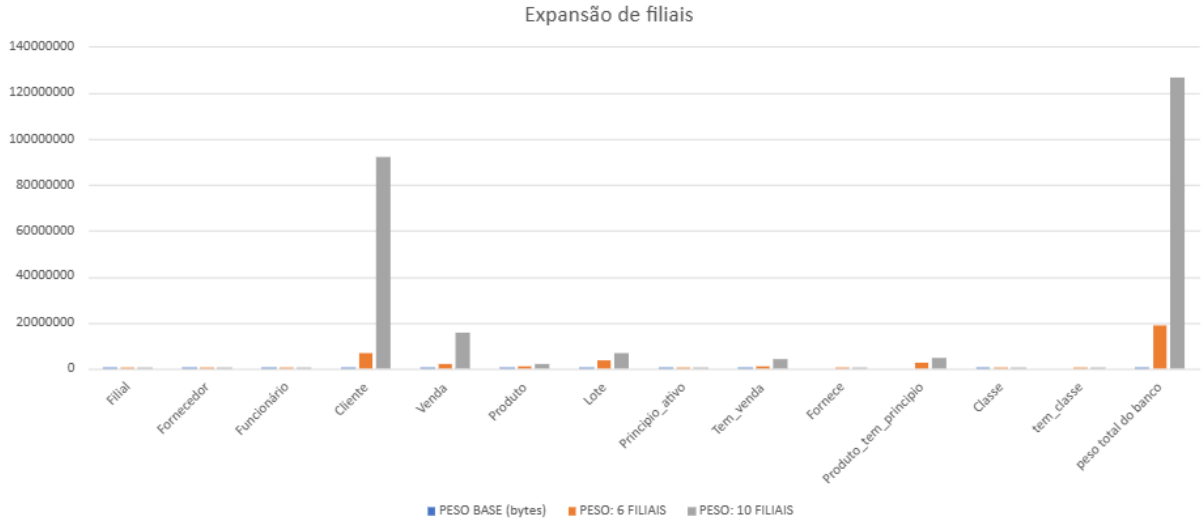
Tabelas	Projeção 1 ano	Projeção 2 anos
Filial	2790	2790
Fornecedor	276360	552720
Funcionário	337200	674400
Cliente	1.108.800.000	2217600000
Venda	192.000.000	384.000.000
Produto	26.016.000	58.536.000
Lote	80.640.000	181.440.000
Principio_ativo	624000	1248000
Tem_venda	67200000	134400000
Fornece	70560	141120
Produto_tem_principio	52.800.000	105.600.000
Classe	149.760	299.520
Tem_classe	1.152.000	2.592.000
Peso total do banco	**1.520.468.670**	**3.082.286.550**

Tabela 31: Peso dos dados em diferentes cenários



Fonte: Excel

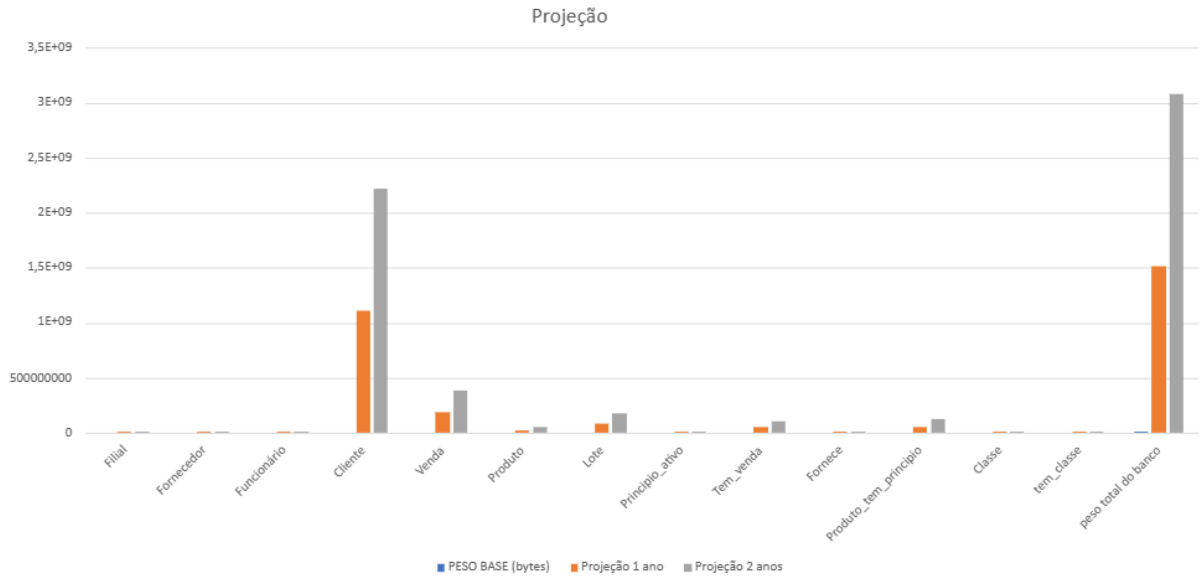
Figura 13: Grafico em barras - Crescimento do peso do banco por tabela.



Fonte: Excel

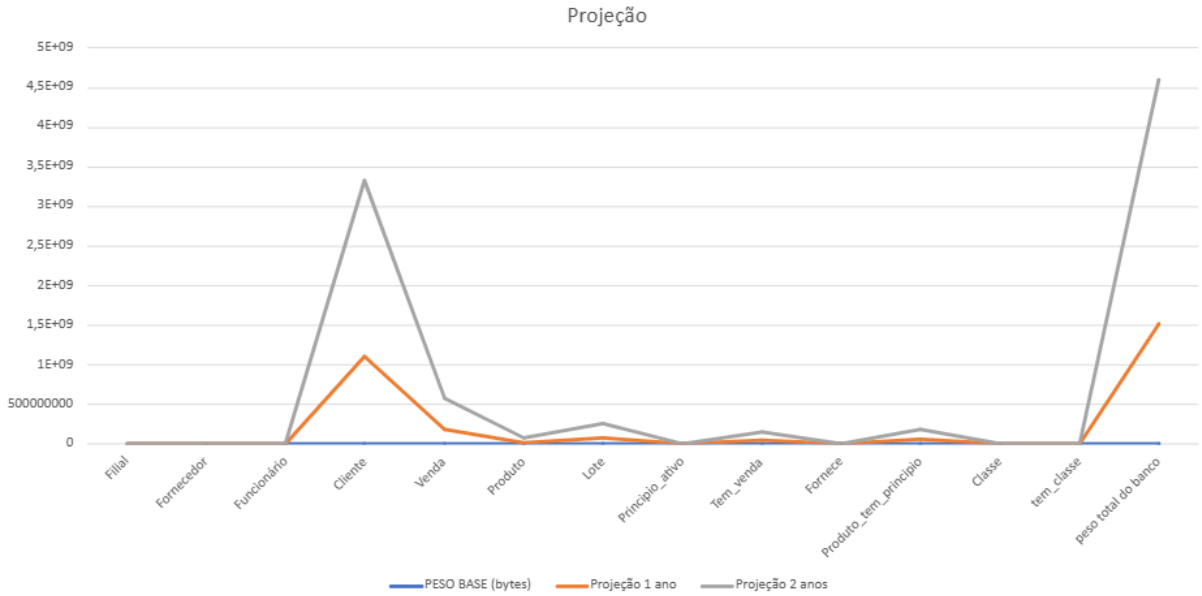
10.3.6 Grafico de linhas e barras sobre projeção 1 e 2 anos

Figura 13: Grafico de barras - Crescimento do peso do banco por tabela.



Fonte: Excel

Figura 13: Grafico em linha - Projeção do peso do banco por tabela.



Fonte: Excel

- **Clientes, Vendas e tem_vendas:** Com 10.000 clientes atualmente, o peso do banco de dados aumentou de 3.564.000 (2.000 clientes) para 20.790.000. Esse crescimento deve-se à abertura de novas filiais e à adição de novos produtos ao nosso comércio. Quanto à tabela tem_vendas, cada produto tem atualmente, em média, 187 vendas registradas, resultando em um aumento no volume de dados salvos de 10 milhões para 33 milhões.

- **Produtos e Lotes:** O crescimento na tabela "Produto", de 2.636.000 para 10.544.000, devido ao aumento de 400 para 800 produtos. Esse aumento se deve às novas parcerias estabelecidas. A projeção ilustra como essas adições podem expandir significativamente os dados em nosso banco.

- **Relações Fornecedor-Lote e Funcionário-Cliente** Conforme as filiais aumentam os fornecedores devem fornecer uma quantia maior de lotes por produto. E conforme as filiais aumentam teremos mais funcionários e mais clientes. À medida que fornecedores fornecem lotes e funcionários atendem clientes, o peso do banco aumenta para essas relações, passando de 115.200 para 403.200 na tabela "Fornece"e de 3.104.640 para 10.496.640 na tabela "Atende".

- **Classe e tem_classe** Estas tabelas tiveram um crescimento mínimo, pois estão relacionadas às classes dos remédios. Portanto, o ajuste foi apenas devido à quantidade de remédios que tivemos (400 -> 800). A tabela Classe manteve a quantidade (com somente 3 classes, independentemente da quantidade de remédios), e tem_classe saltou de 12.000 para 24.000 bytes.

- **Filial e Fornecedor:** Estas tabelas tiveram baixo crescimento. O fornecedor se manteve com cerca de 10 mil, já que eles já distribuíam alguns itens das novas parcerias, razão pela qual essa parceria foi tão bem-sucedida. A tabela Filial saltou de 3.348 para 5.580 devido à abertura de novas filiais (De 6 filiais para 10 filiais).

10.3.7 Considerações Finais sobre Crescimento de Dados

- **Backup e Recuperação:** Independentemente do hardware escolhido, uma estratégia robusta de backup e recuperação é crucial. Isso pode incluir armazenamento em nuvem para backups regulares e recuperação de desastres.

- **Segurança dos Dados:** Proteção com criptografia e sistemas de segurança adequados são necessários para proteger informações sensíveis, especialmente dados pessoais de clientes e funcionários.
- **Escalabilidade:** A infraestrutura de armazenamento deve ser facilmente escalável para acomodar o crescimento futuro, tanto em termos de capacidade de armazenamento quanto de performance.

Selecionar o hardware de armazenamento apropriado para cada tabela envolve um equilíbrio entre custo, desempenho e requisitos operacionais, sempre alinhado às prioridades e estratégias de negócios da farmácia LAV.

OBS: o crescimento é em vista das regras de negócio.

11 Inserção de Dados

A inserção de dados em uma tabela *SQL* é o processo de adicionar novas linhas de dados a uma tabela. A seguir há as inserções necessárias para o banco de dados do projeto.

```
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780001', 'site1
    .com', '11987654321', 'Endereço da Filial 1');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780002', 'site2
    .com', '21987654321', 'Endereço da Filial 2');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780003', 'site3
    .com', '31987654321', 'Endereço da Filial 3');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780004', 'site4
    .com', '41987654321', 'Endereço da Filial 4');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780005', 'site5
    .com', '51987654321', 'Endereço da Filial 5');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780006', 'site6
    .com', '61987654321', 'Endereço da Filial 6');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780007', 'site7
    .com', '71987654321', 'Endereço da Filial 7');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780008', 'site8
    .com', '81987654321', 'Endereço da Filial 8');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780009', 'site9
    .com', '91987654321', 'Endereço da Filial 9');
INSERT INTO filial (cnpj, site, telefone, endereco) VALUES ('123456780010', '
    site10.com', '101987654321', 'Endereço da Filial 10');

INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    87456321000189', 'LAB. bate bem', '11987654321', 'Rua das Tecnologias, 500, Sã
    o Paulo, SP', 'Coracao bem.');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    76849531000155', 'LAB. PRINCIPA', '21987654322', 'Estrada dos Agricultores,
    120, Campinas, SP', 'Nos temos a solucao.');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    65849321000122', 'VIVABEM LAB.', '31987654323', 'Avenida dos Construtores,
    750, Belo Horizonte, MG', 'menos dor');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    54938210001299', 'BioHealth Equipamentos', '41987654324', 'Rua dos Médicos,
    200, Rio de Janeiro, RJ', 'Equipamentos médicos e hospitalares.');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    43021198000188', 'LAB. VIVA', '51987654325', 'Rua da Moda, 450, Fortaleza, CE'
    , ' viva bem ');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    32109876000199', 'LAB. dorme bem', '61987654326', 'Parque Tecnológico, 1080,
    Salvador, BA', 'Solucoes para o sono');
```

```
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    21098765000177', 'FARMALEV', '71987654327', 'Boulevard Gourmet, 330, Porto
    Alegre, RS', 'FARMA');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    21098765000173', 'Fini brazil', '11387654321', 'Rua das lagrimas , 230, Porto
    Alegre, RS', 'Suprimento de doces');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    23231942388322', 'Come bem', '51287154331', 'Rua das Parques , 130, Mato-Groso
    , MT', 'Doces & bebidas');
INSERT INTO fornecedor (cnpj, nome, telefone, endereco, portfolio) VALUES ('
    34573236842233', 'LAB. sobreviva', '33112496954', 'Rua das Parques , 131, Mato
    -Groso, MT', 'SEM DOR');

INSERT INTO funcionario (cpf, rg, nome, email, telefone, certificado, cargo,
    comissao, salario_fixo, salario_mensal, id_filial, endereco)
VALUES
('12345678901', '987654321', 'João Silva', 'joao.silva@email.com', '11987654321',
    TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 1, 'Endereço 1'),
('23456789012', '876543219', 'Maria Oliveira', 'maria.oliveira@email.com', '
    21987654321', TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 1, 'Endereço 2'),
('34567890123', '765432198', 'Ana Costa', 'ana.costa@email.com', '31987654321',
    FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 1, 'Endereço 3'),
('45678901234', '654321987', 'Carlos Dias', 'carlos.dias@email.com', '41987654321'
    , FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 1, 'Endereço 4'),
('56789012345', '543219876', 'Teresa Gonçalves', 'teresa.goncalves@email.com', '
    51987654321', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 1, 'Endereço 5'),
('56789012145', '543219176', 'Teresa', 'teresa.@email.com', '51987654321', FALSE,
    'Atendente', 2.00, 2000.00, 2040.00, 2, 'Endereço 5'),
('45678901134', '654321187', 'Carlos ', 'carlos.@email.com', '41987654321', FALSE,
     'Atendente', 2.00, 2000.00, 2040.00, 2, 'Endereço 4'),
('34567890223', '765432298', 'Ana ', 'ana.@email.com', '31987654321', FALSE, '
    Faxineira', 0.00, 1200.00, 1200.00, 2, 'Endereço 3'),
('23456789312', '876543119', 'Maria ', 'maria.@email.com', '21987654321', TRUE, '
    Gerente', 10.00, 5000.00, 5500.00, 2, 'Endereço 2'),
('12345678301', '987654121', 'João ', 'joao.@email.com', '11987654321', TRUE, '
    Farmaceutico', 5.00, 3000.00, 3050.00, 2, 'Endereço 1'),
('50789012145', '513219176', 'Teresa dias', 'teresa.dias@email.com', '51980650121'
    , FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 4, 'Endereço 5'),
('45678911134', '611321187', 'Carlos dias ', 'carlos.diaSs@email.com', '
    41087014321', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 4, 'Endereço 4'),
('34560090123', '715432218', 'Ana dias', 'ana.dias@email.com', '31987604021',
    FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 4, 'Endereço 3'),
('23456080312', '176543119', 'Maria dias', 'maria.dias@email.com', '21081043211',
    TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 4, 'Endereço 2'),
('1234071301', '917154121', 'João dias', 'joao.dias@email.com', '11907604321',
    TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 4, 'Endereço 1'),
('56789011145', '541219176', 'Teresa triz', 'teresa.triz@email.com', '51917654321'
    , FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 3, 'Endereço 5'),
('45678902134', '651321187', 'Carlos triz', 'carlos.triz@email.com', '41917654321'
    , FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 3, 'Endereço 4'),
('34567892223', '761432298', 'Ana triz', 'ana.triz@email.com', '31987654311',
    FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 3, 'Endereço 3'),
('23456782312', '871543119', 'Maria triz', 'maria.triz@email.com', '21987154321',
    TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 3, 'Endereço 2'),
('12345672301', '981654121', 'João triz', 'joao.triz@email.com', '11987651321',
    TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 3, 'Endereço 1'),
('56789712145', '503219176', 'vitor', 'vitor.@email.com', '51987614321', FALSE, '
    Atendente', 2.00, 2000.00, 2040.00, 5, 'Endereço 5'),
('45678701134', '604321187', 'gabriel ', 'gabriel.@email.com', '41982654321',
    FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 5, 'Endereço 4'),
```

```
('34567790223', '705432298', 'vitoria ', 'vitoria.@email.com', '31982654321',
    FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 5, 'Endereço 3'),
('23456779312', '806543119', 'jucelino ', 'jucelino.@email.com', '21927654321',
    TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 5, 'Endereço 2'),
('10345678301', '907654121', 'junior ', 'junior.@email.com', '11987624321', TRUE,
    'Farmaceutico', 5.00, 3000.00, 3050.00, 5, 'Endereço 1'),
('56389712145', '303239176', 'vitor dias', 'vitor.dias@email.com', '51987614321',
    FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 5, 'Endereço 5'),
('45373701134', '634323187', 'gabriel dias', 'gabriel.dias@email.com', '
    41912614321', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 5, 'Endereço 4'),
('34563790223', '735433298', 'vitoria dias', 'vitoria.dias@email.com', '
    31982154321', FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 5, 'Endereço 3'),
('23453779312', '836543119', 'jucelino dias', 'jucelino.dias@email.com', '
    21127154321', TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 5, 'Endereço 2'),
('11145678301', '903634121', 'junior dias', 'junior.dias@email.com', '11987624121'
    , TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 5, 'Endereço 1'),
('56789412145', '540009176', 'vitor santos', 'vitor.santos@email.com', '
    51987614421', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 6, 'Endereço 5'),
('45674701134', '644321187', 'gabriel santos', 'gabriel.santos@email.com', '
    41984654321', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 6, 'Endereço 4'),
('34567490223', '704432298', 'vitoria santos', 'vitoria.santos@email.com', '
    31984654321', FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 6, 'Endereço 3'),
('23454779312', '806543149', 'jucelino santos', 'jucelino.santos@email.com', '
    21427654321', TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 6, 'Endereço 2'),
('12344678301', '907644121', 'junior santos', 'junior.santos@email.com', '
    14987424321', TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 6, 'Endereço 1'),
('56789717145', '503719176', 'isabela', 'isabela.@email.com', '51987614321', FALSE
    , 'Atendente', 2.00, 2000.00, 2040.00, 7, 'Endereço 5'),
('45678707134', '607321187', 'davi ', 'davi.@email.com', '41982657321', FALSE, '
    Atendente', 2.00, 2000.00, 2040.00, 7, 'Endereço 4'),
('34567797223', '707432298', 'giovanna ', 'giovanna.@email.com', '31972654321',
    FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 7, 'Endereço 3'),
('23456777312', '807543119', 'fernanda ', 'fernanda.@email.com', '21977654321',
    TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 7, 'Endereço 2'),
('12345677301', '002654121', 'caio ', 'caio.@email.com', '11987724321', TRUE, '
    Farmaceutico', 5.00, 3000.00, 3050.00, 7, 'Endereço 1'),
('56789787145', '503719876', 'isabela santos', 'isabela.santos@email.com', '
    51988614321', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 8, 'Endereço 5'),
('45678807134', '607821187', 'davi santos', 'davi.santos@email.com', '41982658321'
    , FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 8, 'Endereço 4'),
('34567798223', '708482298', 'giovanna santos', 'giovanna.santos@email.com', '
    31982654321', FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 8, 'Endereço 3'),
('23456778312', '808543119', 'fernanda santos', 'fernanda.santos@email.com', '
    21987654321', TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 8, 'Endereço 2'),
('01345678301', '908654121', 'caio santos', 'caio.santos@email.com', '11988724321'
    , TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 8, 'Endereço 1'),
('56789717115', '544419176', 'isabela dias', 'isabela.dias@email.com', '
    51997614321', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 9, 'Endereço 5'),
('45678709134', '607441197', 'davi dias', 'davi.dias@email.com', '41982657391',
    FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 9, 'Endereço 4'),
('34567797213', '707442298', 'giovanna dias', 'giovannadias.@email.com', '
    31926543211', FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 9, 'Endereço 3'),
('23456793111', '804493119', 'fernanda dias', 'fernanda.dias@email.com', '
    21976543211', TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 9, 'Endereço 2'),
('12345679301', '904459121', 'caio dias', 'caio.dias@email.com', '19877243211',
    TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 9, 'Endereço 1'),
('56789713145', '303719176', 'isabela dias C', 'isabela.diasC@email.com', '
    51997634321', FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 10, 'Endereço 5'),
('45678703134', '307321197', 'davi dias C', 'davi.diasC@email.com', '41982637391',
     FALSE, 'Atendente', 2.00, 2000.00, 2040.00, 10, 'Endereço 4'),
```

```
('34567793223', '307932298', 'giovanna dias C', 'giovannadias.C@email.com', '
    31923543211', FALSE, 'Faxineira', 0.00, 1200.00, 1200.00, 10, 'Endereço 3'),
('2345679312', '307593119', 'fernanda dias C', 'fernanda.diasC@email.com', '
    21936543211', TRUE, 'Gerente', 10.00, 5000.00, 5500.00, 10, 'Endereço 2'),
('12345673301', '307659121', 'caio dias C', 'caio.diasC@email.com', '19873243211',
     TRUE, 'Farmaceutico', 5.00, 3000.00, 3050.00, 10, 'Endereço 1');

INSERT INTO cliente (cpf, email, nome, data_nascimento, telefone, endereco,
    status_aniversario) VALUES
('12345678901', 'cliente1@email.com', 'Cliente Um', '1985-01-01', '11987654321', '
    Endereço Cliente 1', False),
('23456789012', 'cliente2@email.com', 'Cliente Dois', '1990-02-02', '21987654322',
     'Endereço Cliente 2', False),
('34567890123', 'cliente3@email.com', 'Cliente Três', '1995-03-03', '31987654323',
     'Endereço Cliente 3', False),
('45678901234', 'cliente4@email.com', 'Cliente Quatro', '2000-04-04', '41987654324
    ', 'Endereço Cliente 4', False),
('56789012345', 'cliente5@email.com', 'Cliente Cinco', '2005-05-05', '51987654325'
    , 'Endereço Cliente 5', False),

('1234567213', 'cliente13@email.com', 'Cliente Um', '19850101', '11987654211', '
    Endereço Cliente 2', False),
('2345672822', 'cliente66@email.com', 'Cliente Dois', '19900202', '21987643221', '
    Endereço Cliente 2', False),
('3456789232', 'client11@email.com', 'Cliente Três', '19950303', '31198765323', '
    Endereço Cliente 2', False),
('4567891242', 'cliente22@email.com', 'Cliente Quatro', '20000404', '41986543214',
     'Endereço Cliente 2', False),
('5678912252', 'cliente21@email.com', 'Cliente Cinco', '20050505', '51987514325',
    'Endereço Cliente 2', False),

('1234563213', 'cliente113@email.com', 'Cliente 11', '1985-10-23', '11987165421',
    'Endereço Cliente 2', False),
('2345372822', 'cliente616@email.com', 'Cliente 11', '1993-02-22', '12198764322',
    'Endereço Cliente 2', False),
('3156789232', 'client111@email.com', 'Cliente 11', '1995-03-23', '11981765323', '
    Endereço Cliente 2', False),
('4167391242', 'cliente221@email.com', 'Cliente 11', '2004-03-24', '41986543124',
    'Endereço Cliente 2', False),
('5178312252', 'cliente211@email.com', 'Cliente 11', '2005-02-05', '51987543125',
    'Endereço Cliente 2', False),

('1244563214', 'cliente112@email.com', 'Cliente 31', '1985-04-23', '11947654121',
    'Endereço Cliente 2', False),
('2445372824', 'cliente612@email.com', 'Cliente 31', '1993-04-22', '21947643122',
    'Endereço Cliente 2', False),
('3146789242', 'client121@email.com', 'Cliente 31', '1995-04-23', '11947653123', '
    Endereço Cliente 2', False),
('4147391242', 'cliente222@email.com', 'Cliente 31', '2004-04-24', '41946541324',
    'Endereço Cliente 2', False),
('5478312254', 'cliente212@email.com', 'Cliente 31', '2005-12-05', '51947543125',
    'Endereço Cliente 2', False),

('7234563213', 'cliente11w@email.com', 'Cliente 113', '1985-10-23', '11981765421',
     'Endereço Cliente 11', False),
('7345372822', 'cliente61w@email.com', 'Cliente 616', '1993-02-22', '21987641322',
     'Endereço Cliente 616', False),
('7456789232', 'client11w@email.com', 'Cliente 111', '1995-03-23', '13198765323',
    'Endereço Cliente 111', False),
('7567391242', 'cliente22w@email.com', 'Cliente 221', '2004-03-02', '14198654324',
```

```
    'Endereço Cliente 221', False),
('7678312252', 'cliente21w@email.com', 'Cliente 211', '2003-10-25', '15198754325',
    'Endereço Cliente 211', False),

('7234663213', 'clientetaew@email.com', 'Cliente 113', '1985-10-23', '11987651421'
    , 'Endereço Cliente 11', False),
('7346372822', 'clientetdwe@email.com', 'Cliente 616', '1993-02-22', '21987164322'
    , 'Endereço Cliente 616', False),
('7456789632', 'client1tsew@email.com', 'Cliente 111', '1995-03-23', '31987653231'
    , 'Endereço Cliente 111', False),
('7567391642', 'clientetwewd@email.com', 'Cliente 221', '2004-03-02', '41986543241
    ', 'Endereço Cliente 221', False),
('7668612252', 'clientetaww@email.com', 'Cliente 211', '2003-10-25', '51987543251'
    , 'Endereço Cliente 211', False),

('7274673213', 'clientetaesw@email.com', 'Cliente 113', '1985-10-23', '11947644121
    ', 'Endereço Cliente 11', False),
('7347372722', 'clientetdwse@email.com', 'Cliente 616', '1993-02-22', '21984643122
    ', 'Endereço Cliente 616', False),
('7457789732', 'client1tsesw@email.com', 'Cliente 111', '1995-03-23', '31947614323
    ', 'Endereço Cliente 111', False),
('7577391742', 'clientetweswd@email.com', 'Cliente 221', '2004-03-02', '
    41486544124', 'Endereço Cliente 221', False),
('7678672272', 'clientetawsw@email.com', 'Cliente 211', '2003-10-25', '51987441425
    ', 'Endereço Cliente 211', False),

('7234863283', 'clienteta8djw@email.com', 'Cliente 113', '1985-10-23', '
    11987651821', 'Endereço Cliente 11', False),
('7386872822', 'clientetd8dde@email.com', 'Cliente 616', '1993-02-22', '
    21918864322', 'Endereço Cliente 616', False),
('7456889832', 'client1t8hewd@email.com', 'Cliente 111', '1995-03-23', '
    31987165823', 'Endereço Cliente 111', False),
('7567381842', 'clientetw8ehswd@email.com', 'Cliente 221', '2004-03-02', '
    41881684324', 'Endereço Cliente 221', False),
('7668612852', 'clientetaa8w@email.com', 'Cliente 211', '2003-10-25', '51981758325
    ', 'Endereço Cliente 211', False),

('7294669213', 'cliente9ew@email.com', 'Cliente 113', '1985-10-23', '11997169421',
    'Endereço Cliente 11', False),
('734697922', 'clientet9we@email.com', 'Cliente 616', '1993-02-22', '21991769322',
    'Endereço Cliente 616', False),
('7496789632', 'client19s9w@email.com', 'Cliente 111', '1995-03-23', '31981965923'
    , 'Endereço Cliente 111', False),
('75693942', 'clientetw9d@email.com', 'Cliente 221', '2004-03-02', '41981954324',
    'Endereço Cliente 221', False),
('7968912952', 'cliente9w@email.com', 'Cliente 211', '2003-10-25', '51997159325',
    'Endereço Cliente 211', False),

('7294339313', 'cliente9ew3@email.com', 'Cliente 113', '1985-10-23', '11913991421'
    , 'Endereço Cliente 11', False),
('7334637322', 'clientet9w3e@email.com', 'Cliente 616', '1993-02-22', '21936932112
    ', 'Endereço Cliente 616', False),
('7496738332', 'client193s9w@email.com', 'Cliente 111', '1995-03-23', '31983311923
    ', 'Endereço Cliente 111', False),
('7536939342', 'clientetw93d@email.com', 'Cliente 221', '2004-03-02', '41931134324
    ', 'Endereço Cliente 221', False),
('7968932352', 'cliente39w@email.com', 'Cliente 211', '2003-10-25', '51199715325',
    'Endereço Cliente 211', False);

INSERT INTO venda (id_funcionario, id_cliente, data_venda, valor_total,
```

```
   nota_fiscal, id_filial) VALUES
(1, 1, '2024-03-23', 299.99, 'NF123156', 1),
(1, 2, '2024-03-23', 299.99, 'NF113456', 1),
(1, 3, '2024-03-23', 299.99, 'NF123416', 1),
(1, 4, '2024-03-23', 299.99, 'NF123156', 1),
(1, 5, '2024-03-23', 299.99, 'NF113456', 1),
(2, 6, '2024-03-23', 299.99, 'NF123451', 1),
(2, 7, '2024-03-23', 299.99, 'NF121456', 1),
(2, 8, '2024-03-23', 299.99, 'NF123456', 1),
(2, 9, '2024-03-23', 299.99, 'NF113456', 1),
(2, 10, '2024-03-23', 299.99, 'NF123156', 1),
(2, 11, '2024-03-23', 299.99, 'NF123156', 1),
(3, 12, '2024-03-23', 299.99, 'NF113416', 1),
(3, 13, '2024-03-23', 299.99, 'NF113456', 1),
(3, 14, '2024-03-23', 299.99, 'NF123156', 1),
(3, 15, '2024-03-23', 299.99, 'NF123456', 1),
(3, 16, '2024-03-23', 299.99, 'NF121456', 1),
(3, 17, '2024-03-23', 299.99, 'NF123156', 1),
(4, 18, '2024-03-23', 299.99, 'NF121456', 1),
(4, 19, '2024-03-23', 299.99, 'NF123456', 1),
(4, 20, '2024-03-23', 299.99, 'NF123451', 1),
(2, 21, '2025-03-23', 299.99, 'NF123456', 2),
(2, 22, '2025-03-13', 299.99, 'NF113156', 2),
(2, 23, '2025-03-23', 299.99, 'NF121456', 2),
(2, 24, '2025-03-13', 299.99, 'NF123456', 2),
(3, 25, '2025-01-23', 299.99, 'NF113456', 3),
(3, 26, '2025-01-23', 299.99, 'NF123156', 3),
(3, 27, '2025-01-23', 299.99, 'NF123156', 3),
(4, 18, '2024-03-23', 299.99, 'NF121456', 4),
(4, 19, '2024-03-23', 299.99, 'NF123456', 4),
(4, 20, '2024-03-23', 299.99, 'NF123451', 4),
(2, 21, '2025-03-23', 299.99, 'NF123456', 4),
(2, 22, '2025-03-13', 299.99, 'NF113156', 4),
(2, 23, '2025-03-23', 299.99, 'NF121456', 4),
(2, 24, '2025-03-13', 299.99, 'NF123456', 4),
(3, 25, '2025-01-23', 299.99, 'NF113456', 4),
(3, 26, '2025-01-23', 299.99, 'NF123156', 4),
(3, 26, '2025-04-24', 299.99, 'NF123136', 4),
(3, 26, '2025-04-24', 299.99, 'NF123126', 4),
(3, 26, '2025-04-24', 299.99, 'NF123116', 4),
(1, 1, '2024-04-02', 299.99, 'NF122056', 1),
(1, 2, '2024-04-02', 299.99, 'NF111056', 1),
(1, 3, '2024-04-02', 299.99, 'NF124016', 1),
(1, 4, '2024-04-02', 299.99, 'NF121056', 1),
(3, 27, '2025-01-23', 299.99, 'NF123156', 4);

INSERT INTO produto (nome, preco, marca, tipo_remedio, prescricao_medica,
   tipo_tarja, forma_de_administracao, composicao_remedio, data_validade,
   data_fabricacao, secao) VALUES
 ('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
 ('Amoxicilina', 25.90, 'Eurofarma', 'Antibiótico', TRUE, 'Tarja Vermelha', 'Oral'
    , 'Amoxicilina Triidratada 500mg', '2026-02-01', '2023-02-01', 'Medicamentos'
    ),

 ('Omeprazol', 18.50, 'Aché', 'Antiulceroso', FALSE, 'Sem Tarja', 'Oral', '
    Omeprazol 20mg', '2026-03-01', '2023-03-01', 'Medicamentos'),
 ('Paracetamol', 10.00, 'Cimed', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', '
    Paracetamol 500mg', '2026-04-01', '2023-04-01', 'Medicamentos'),
 ('Losartana Potássica', 22.90, 'Sandoz', 'Anti-hipertensivo', FALSE, 'Sem Tarja',
```

```
    'Oral', 'Losartana Potássica 50mg', '2026-05-01', '2023-05-01', '
    Medicamentos'),
('Shampoo Anticaspa', 25.90, 'ClearMen', NULL, FALSE, NULL, NULL, 'Zinco
    Piritiona', '2025-12-31', '2023-01-01', 'Higiene'),
('Creme Dental', 12.50, 'Colgate', NULL, FALSE, NULL, NULL, 'Fluoreto de Sódio',
    '2025-12-31', '2023-01-01', 'Higiene'),
('Desodorante Roll-On', 15.00, 'Rexona', NULL, FALSE, NULL, NULL, 'Aqua, Aluminum
    Chlorohydrate', '2025-12-31', '2023-01-01', 'Higiene'),
('Protetor Solar FPS 50', 49.90, 'Nivea', NULL, FALSE, NULL, NULL, 'Avobenzone,
    Octocrylene', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
('Dipirona', 15.00, 'Medley', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', 'Dipirona
    Sódica 500mg', '2026-01-01', '2023-01-01', 'Medicamentos'),

('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
('Amoxicilina', 25.90, 'Eurofarma', 'Antibiótico', TRUE, 'Tarja Vermelha', 'Oral'
    , 'Amoxicilina Triidratada 500mg', '2026-02-01', '2023-02-01', 'Medicamentos'
    ),
('Omeprazol', 18.50, 'Aché', 'Antiulceroso', FALSE, 'Sem Tarja', 'Oral', '
    Omeprazol 20mg', '2026-03-01', '2023-03-01', 'Medicamentos'),
('Paracetamol', 10.00, 'Cimed', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', '
    Paracetamol 500mg', '2026-04-01', '2023-04-01', 'Medicamentos'),
('Losartana Potássica', 22.90, 'Sandoz', 'Anti-hipertensivo', FALSE, 'Sem Tarja',
    'Oral', 'Losartana Potássica 50mg', '2026-05-01', '2023-05-01', '
    Medicamentos'),
('Shampoo Anticaspa', 25.90, 'ClearMen', NULL, FALSE, NULL, NULL, 'Zinco
    Piritiona', '2025-12-31', '2023-01-01', 'Higiene'),
('Creme Dental', 12.50, 'Colgate', NULL, FALSE, NULL, NULL, 'Fluoreto de Sódio',
    '2025-12-31', '2023-01-01', 'Higiene'),
('Desodorante Roll-On', 15.00, 'Rexona', NULL, FALSE, NULL, NULL, 'Aqua, Aluminum
    Chlorohydrate', '2025-12-31', '2023-01-01', 'Higiene'),
('Protetor Solar FPS 50', 49.90, 'Nivea', NULL, FALSE, NULL, NULL, 'Avobenzone,
    Octocrylene', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
('Dipirona', 15.00, 'Medley', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', 'Dipirona
    Sódica 500mg', '2026-01-01', '2023-01-01', 'Medicamentos'),

('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
('Amoxicilina', 25.90, 'Eurofarma', 'Antibiótico', TRUE, 'Tarja Vermelha', 'Oral'
    , 'Amoxicilina Triidratada 500mg', '2026-02-01', '2023-02-01', 'Medicamentos'
    ),
('Omeprazol', 18.50, 'Aché', 'Antiulceroso', FALSE, 'Sem Tarja', 'Oral', '
    Omeprazol 20mg', '2026-03-01', '2023-03-01', 'Medicamentos'),
('Paracetamol', 10.00, 'Cimed', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', '
    Paracetamol 500mg', '2026-04-01', '2023-04-01', 'Medicamentos'),
('Losartana Potássica', 22.90, 'Sandoz', 'Anti-hipertensivo', FALSE, 'Sem Tarja',
    'Oral', 'Losartana Potássica 50mg', '2026-05-01', '2023-05-01', '
    Medicamentos'),
('Shampoo Anticaspa', 25.90, 'ClearMen', NULL, FALSE, NULL, NULL, 'Zinco
    Piritiona', '2025-12-31', '2023-01-01', 'Higiene'),
('Creme Dental', 12.50, 'Colgate', NULL, FALSE, NULL, NULL, 'Fluoreto de Sódio',
    '2025-12-31', '2023-01-01', 'Higiene'),
('Desodorante Roll-On', 15.00, 'Rexona', NULL, FALSE, NULL, NULL, 'Aqua, Aluminum
    Chlorohydrate', '2025-12-31', '2023-01-01', 'Higiene'),
('Protetor Solar FPS 50', 49.90, 'Nivea', NULL, FALSE, NULL, NULL, 'Avobenzone,
    Octocrylene', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
```

```
 ('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
 ('Dipirona', 15.00, 'Medley', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', 'Dipirona
     Sódica 500mg', '2026-01-01', '2023-01-01', 'Medicamentos'),

 ('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
 ('Amoxicilina', 25.90, 'Eurofarma', 'Antibiótico', TRUE, 'Tarja Vermelha', 'Oral'
    , 'Amoxicilina Triidratada 500mg', '2026-02-01', '2023-02-01', 'Medicamentos'
    ),
 ('Omeprazol', 18.50, 'Aché', 'Antiulceroso', FALSE, 'Sem Tarja', 'Oral', '
    Omeprazol 20mg', '2026-03-01', '2023-03-01', 'Medicamentos'),
 ('Paracetamol', 10.00, 'Cimed', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', '
    Paracetamol 500mg', '2026-04-01', '2023-04-01', 'Medicamentos'),
 ('Losartana Potássica', 22.90, 'Sandoz', 'Anti-hipertensivo', FALSE, 'Sem Tarja',
     'Oral', 'Losartana Potássica 50mg', '2026-05-01', '2023-05-01', '
    Medicamentos'),
 ('Shampoo Anticaspa', 25.90, 'ClearMen', NULL, FALSE, NULL, NULL, 'Zinco
    Piritiona', '2025-12-31', '2023-01-01', 'Higiene'),
 ('Creme Dental', 12.50, 'Colgate', NULL, FALSE, NULL, NULL, 'Fluoreto de Sódio',
    '2025-12-31', '2023-01-01', 'Higiene'),
 ('Desodorante Roll-On', 15.00, 'Rexona', NULL, FALSE, NULL, NULL, 'Aqua, Aluminum
     Chlorohydrate', '2025-12-31', '2023-01-01', 'Higiene'),
 ('Protetor Solar FPS 50', 49.90, 'Nivea', NULL, FALSE, NULL, NULL, 'Avobenzone,
    Octocrylene', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
 ('Hidratante Corporal', 29.90, 'Dove', NULL, FALSE, NULL, NULL, 'Glycerin,
    Paraffinum Liquidum', '2025-12-31', '2023-01-01', 'Cuidados Pessoais'),
 ('Dipirona', 15.00, 'Medley', 'Analgésico', FALSE, 'Sem Tarja', 'Oral', 'Dipirona
     Sódica 500mg', '2026-01-01', '2023-01-01', 'Medicamentos');


INSERT INTO lote (id_produto, nome, quantidade, data_despacho, id_filial)
VALUES
(1, 'Lote A1', 100, '2024-03-23', 1),
(2, 'Lote B2', 150, '2024-03-24', 1),
(3, 'Lote C3', 200, '2024-03-25', 1),
(4, 'Lote D4', 250, '2024-03-26', 1),
(5, 'Lote E5', 300, '2024-03-27', 1),
(6, 'Lote F6', 350, '2024-03-28', 1),
(7, 'Lote G7', 400, '2024-03-29', 1),
(8, 'Lote A1', 100, '2024-03-23', 1),
(9, 'Lote B2', 150, '2024-03-24', 1),
(10, 'Lote C3', 200, '2024-03-25', 1),
(11, 'Lote D4', 250, '2024-03-26', 1),
(12, 'Lote A1T', 100, '2024-03-23', 2),
(13, 'Lote B2T', 150, '2024-03-24', 2),
(14, 'Lote C3T', 200, '2024-03-25', 2),
(15, 'Lote D4T', 250, '2024-03-26', 2),
(16, 'Lote E5T', 300, '2024-03-27', 2),
(17, 'Lote F6T', 350, '2024-03-28', 2),
(18, 'Lote G7T', 400, '2024-03-29', 2),
(19, 'Lote A1T', 100, '2024-03-23', 2),
(20, 'Lote B2T', 150, '2024-03-24', 2),
(21, 'Lote C3T', 200, '2024-03-25', 2),
(22, 'Lote D4T', 250, '2024-03-26', 2),
(23, 'Lote A1R', 100, '2024-03-23', 3),
(24, 'Lote B2R', 150, '2024-03-24', 3),
(25, 'Lote C3R', 200, '2024-03-25', 3),
(26, 'Lote D4R', 250, '2024-03-26', 3),
(27, 'Lote E5R', 300, '2024-03-27', 3),
```

```
(28, 'Lote F6R', 350, '2024-03-28', 3),
(29, 'Lote G7R', 400, '2024-03-29', 3),
(30, 'Lote A1R', 100, '2024-03-23', 3),
(31, 'Lote B2R', 150, '2024-03-24', 3),
(32, 'Lote C3R', 200, '2024-03-25', 3),
(33, 'Lote D4R', 250, '2024-03-26', 3),
(34, 'Lote A1W', 100, '2024-03-23', 4),
(35, 'Lote B2W', 150, '2024-03-24', 4);

INSERT INTO principio_ativo (nome) VALUES
('Paracetamol'),
('Amoxicilina'),
('Zinco Piritiona'),
('Fluoreto de Sódio'),
('Aluminum Chlorohydrate');




INSERT INTO tem_venda (id_produto, id_venda, quantidade, preco, desconto)
VALUES
(1, 1, 2, 25.90, 0.00),
(2, 2, 1, 12.50, 1.00),
(3, 3, 3, 15.00, 0.00),
(4, 4, 1, 49.90, 5.00),
(5, 5, 2, 29.90, 2.00),
(6, 6, 3, 15.00, 0.00),
(7, 7, 1, 49.90, 5.00),
(8, 8, 7, 29.90, 2.00),
(9, 9, 2, 29.90, 2.00),
(10, 10, 7, 29.90, 2.00),
(11, 11, 2, 29.90, 2.00),
(30, 30, 2, 25.90, 0.00),
(18, 18, 7, 29.90, 2.00),
(19, 19, 7, 29.90, 2.00),
(20, 20, 7, 29.90, 2.00),
(21, 21, 7, 29.90, 2.00),
(31, 31, 2, 25.90, 0.00),
(32, 32, 1, 12.50, 1.00),
(33, 33, 3, 15.00, 0.00),
(34, 34, 7, 49.90, 5.00),
(28, 28, 71, 29.90, 2.00),
(29, 29, 17, 29.90, 2.00),
(24, 20, 8, 29.90, 2.00),
(25, 21, 8, 29.90, 2.00),
(1, 41, 3, 30.00, 0),
(2, 40, 1, 40.00, 5.00),
(2, 42, 5, 25.90, 0.00),
(3, 43, 5, 12.50, 1.00);

INSERT INTO fornece (id_fornecedor, id_produto, id_lote) VALUES
(1, 1, 1),
(2, 2, 2),
(3, 3, 3),
(4, 4, 4),
(5, 5, 5),
(6, 6, 6),
(7, 7, 7),
(1, 8, 8),
(2, 9, 9),
```

```
(3,10,10),
(4,11,11),
(5,12,12),
(6,13,13),
(7,14,14),
(1,15,15),
(2,16,16),
(3,17,17),
(4,18,18),
(5,19,19),
(6,20,20),
(7,21,21),
(1,25,25),
(2,26,26),
(3,27,27),
(4,28,28),
(5,29,29),
(6,30,30),
(7,31,31),
(1,35,35);

INSERT INTO produto_tem_principio (id_produto, id_principio) VALUES
(1,1),
(2,2),
(3,3),
(4,4),
(5,5),
(6,1),
(7,2),
(8,3),
(9,4),
(10,5),
(11,1),
(12,2),
(13,3),
(14,4),
(15,5),
(16,1),
(17,2),
(18,3),
(19,4),
(20,5),
(21,1),
(22,2),
(23,3),
(24,4),
(25,5),
(26,1),
(27,2),
(28,3),
(29,4),
(30,5),
(31,1),
(32,2),
(33,3),
(34,4),
(35,5),
(36,1);

INSERT INTO classe (nome) VALUES
```

```
('inotrópicos'),
('vasodilatadores'),
('anticoagulantes'),
('antipiréticos'),
('analgésicos'),
('paracetamol'),
('Higiene Pessoal'),
('Cuidados com a Pele');

INSERT INTO tem_classe (id_produto, id_classe) VALUES
(1,1),
(2,2),
(3,3),
(4,4),
(5,5),
(6,6),
(7,7),
(8,8),
(9,1),
(10,2),
(11,3),
(12,4),
(13,5),
(14,6),
(15,7),
(16,8),
(17,1),
(18,2),
(19,3),
(20,4),
(21,5),
(22,6),
(23,7),
(24,8),
(25,1),
(26,2),
(27,3),
(28,4),
(29,5),
(30,6),
(31,7),
(32,8),
(33,1),
(34,2),
(35,3),
(36,4);
```

12 Consulta/Relatórios

Consultas *SQL* de agregação são consultas que são usadas para calcular agregações de dados, como somas, médias e contagens. Essas consultas são úteis para obter uma visão geral de grandes conjuntos de dados. Um relatório em *SQL* é um conjunto de dados formatado e organizado para apresentação. Os relatórios do projeto servirão para o controle da empresa. Aqui, tem-se consultas simples sem análise performática da mesma.

12.1 Mensal

Abaixo estão as consultas mensais que julgamos ser de crucial importância de serem realizadas durante o mês, a fim de monitorar o desempenho e o dia a dia das filiais.

12.1.1 Relatório - Número total de remédios vendidos por classe no mês atual

```
SELECT c.nome AS nome_classe, SUM(tv.quantidade) AS quantidade_vendida
FROM tem_venda tv
JOIN venda v ON tv.id_venda = v.id
JOIN produto p ON tv.id_produto = p.id
JOIN tem_classe tc ON p.id = tc.id_produto
JOIN classe c ON tc.id_classe = c.id
WHERE EXTRACT(MONTH FROM v.data_venda) = EXTRACT(MONTH FROM CURRENT_DATE)
AND EXTRACT(YEAR FROM v.data_venda) = EXTRACT(YEAR FROM CURRENT_DATE)
GROUP BY c.nome
ORDER BY quantidade_vendida DESC;
```

12.1.2 Relatório - Produto mais vendido em cada filial no mês atual

```
WITH vendas_mes_atual AS (
    SELECT
        v.id_filial,
        tv.id_produto,
        SUM(tv.quantidade) AS total_vendido
    FROM venda v
    JOIN tem_venda tv ON v.id = tv.id_venda
    WHERE EXTRACT(MONTH FROM v.data_venda) = EXTRACT(MONTH FROM CURRENT_DATE)
        AND EXTRACT(YEAR FROM v.data_venda) = EXTRACT(YEAR FROM CURRENT_DATE)
    GROUP BY v.id_filial, tv.id_produto
),
ranked_produtos AS (
    SELECT
        vma.id_filial,
        vma.id_produto,
        p.nome AS nome_produto,
        vma.total_vendido,
        RANK() OVER(PARTITION BY vma.id_filial ORDER BY vma.total_vendido DESC) AS
            rank
    FROM vendas_mes_atual vma
    JOIN produto p ON vma.id_produto = p.id
)
SELECT
    rp.id_filial,
    rp.nome_produto,
    rp.total_vendido
FROM ranked_produtos rp
WHERE rp.rank = 1;
```

12.1.3 Relatório - Quantidade total vendida de remédios por funcionário em cada filial no mês atual

```
SELECT
    f.id_filial,
    v.id_funcionario,
```

```
    f.nome AS nome_funcionario,
    SUM(tv.quantidade) AS total_remedios_vendidos
FROM venda v
JOIN tem_venda tv ON v.id = tv.id_venda
JOIN produto p ON tv.id_produto = p.id
JOIN funcionario f ON v.id_funcionario = f.id
WHERE p.secao = 'Medicamentos'
    AND EXTRACT(MONTH FROM v.data_venda) = EXTRACT(MONTH FROM CURRENT_DATE)
    AND EXTRACT(YEAR FROM v.data_venda) = EXTRACT(YEAR FROM CURRENT_DATE)
GROUP BY f.id_filial, v.id_funcionario, f.nome
ORDER BY f.id_filial, total_remedios_vendidos DESC;
```

12.1.4 Relatório - Funcionário que mais vendeu no mês atual em cada uma das filiais

```
WITH vendas_funcionario AS (
    SELECT
        v.id_filial,
        v.id_funcionario,
        f.nome AS nome_funcionario,
        SUM(v.valor_total) AS valor_vendido
    FROM venda v
    JOIN funcionario f ON v.id_funcionario = f.id
    WHERE EXTRACT(MONTH FROM v.data_venda) = EXTRACT(MONTH FROM CURRENT_DATE)
        AND EXTRACT(YEAR FROM v.data_venda) = EXTRACT(YEAR FROM CURRENT_DATE)
    GROUP BY v.id_filial, v.id_funcionario, f.nome
),
ranked_vendas AS (
    SELECT
        vf.*,
        RANK() OVER(PARTITION BY vf.id_filial ORDER BY vf.valor_vendido DESC) AS
            rank
    FROM vendas_funcionario vf
)
SELECT
    rv.id_filial,
    rv.id_funcionario,
    rv.nome_funcionario,
    rv.valor_vendido
FROM ranked_vendas rv
WHERE rv.rank = 1
ORDER BY rv.id_filial, rv.valor_vendido DESC;
```

12.1.5 Relatório - Quantidade total de produtos não remédios vendidos por seção no mês atual

```
SELECT
    p.secao,
    SUM(tv.quantidade) AS total_quantidade

FROM tem_venda tv

JOIN produto p ON tv.id_produto = p.id
JOIN venda v ON tv.id_venda = v.id
WHERE p.tipo_remedio IS NULL AND v.data_venda BETWEEN DATE_TRUNC('month',
    CURRENT_DATE) AND CURRENT_DATE
GROUP BY p.secao
```

12.2 Semestral

Logo, abaixo segue-se alguns relatórios anuais necessários e convenientes para analisar o desempenho e nuances das filiais ao longo de 6 meses de operação.

12.2.1 Relatório - Ranking dos Remédios mais vendidos em cada Filial nos últimos 6 meses

```
WITH vendas_recentes AS (
    SELECT
        v.id_filial,
        p.id AS id_produto,
        p.nome AS nome_remedio,
        SUM(tv.quantidade) AS total_vendido
    FROM venda v
    JOIN tem_venda tv ON v.id = tv.id_venda
    JOIN produto p ON tv.id_produto = p.id
    WHERE p.secao = 'Medicamentos'
        AND v.data_venda >= CURRENT_DATE - INTERVAL '6 months'
    GROUP BY v.id_filial, p.id
),
ranked_remedios AS (
    SELECT
        vr.id_filial,
        vr.id_produto,
        vr.nome_remedio,
        vr.total_vendido,
        RANK() OVER(PARTITION BY vr.id_filial ORDER BY vr.total_vendido DESC) AS
            ranking
    FROM vendas_recentes vr
)
SELECT
    rr.id_filial,
    rr.id_produto,
    rr.nome_remedio,
    rr.total_vendido,
    rr.ranking
FROM ranked_remedios rr
ORDER BY rr.id_filial, rr.ranking;
```

12.2.2 Relatório - 3 Filiais que mais venderam Produtos não remédios nos últimos 6 meses

```
SELECT
    v.id_filial,
    SUM(tv.quantidade * tv.preco) AS valor_total_vendas
FROM venda v
JOIN tem_venda tv ON v.id = tv.id_venda
JOIN produto p ON tv.id_produto = p.id
WHERE p.secao != 'Medicamentos'
    AND v.data_venda >= CURRENT_DATE - INTERVAL '6 months'
GROUP BY v.id_filial
ORDER BY valor_total_vendas DESC
LIMIT 3;
```

12.2.3 Relatório - 2 Filiais que mais venderam Remédios nos últimos 6 meses

```sql
SELECT
    v.id_filial,
    SUM(tv.quantidade * tv.preco) AS valor_total_vendas
FROM venda v
JOIN tem_venda tv ON v.id = tv.id_venda
JOIN produto p ON tv.id_produto = p.id
WHERE p.secao = 'Medicamentos'
    AND v.data_venda >= CURRENT_DATE - INTERVAL '6 months'
GROUP BY v.id_filial
ORDER BY valor_total_vendas DESC
LIMIT 2;
```

12.3 Anual

Logo, abaixo segue-se alguns relatórios anuais necessários e convenientes para analisar o desempenho e nuances das filiais ao longo de 12 meses de operação.

12.3.1 Relatório - Filial que mais vendeu produtos no geral em 12 meses

```sql
SELECT
    filial.id AS id_filial,
    SUM(tem_venda.quantidade) AS total_vendido

FROM venda

JOIN filial ON venda.id_filial = filial.id
JOIN tem_venda ON venda.id = tem_venda.id_venda
WHERE venda.data_venda >= CURRENT_DATE - INTERVAL '12 months'
GROUP BY filial.id
ORDER BY total_vendido DESC
LIMIT 1;
```

12.3.2 Relatório - 2 Filiais que menos venderam produtos no geral em 12 meses

```sql
SELECT
    filial.id AS id_filial,
    SUM(tem_venda.quantidade) AS total_vendido

FROM venda

JOIN filial ON venda.id_filial = filial.id
JOIN tem_venda ON venda.id = tem_venda.id_venda
WHERE venda.data_venda >= CURRENT_DATE - INTERVAL '12 months'
GROUP BY filial.id
ORDER BY total_vendido ASC
LIMIT 2;
```

12.3.3 Relatório - 2 Funcionários que mais venderam em cada filial em 12 meses

```sql
WITH VendasPorfuncionario AS (
    SELECT
```

```
        filial.id AS filial_id,
        funcionario.id AS funcionario_id,
        funcionario.nome AS nome_funcionario,
        SUM(tem_venda.quantidade) AS total_vendido,
        RANK() OVER (PARTITION BY filial.id ORDER BY SUM(tem_venda.quantidade)
            DESC) AS rank

    FROM venda

    JOIN funcionario ON venda.id_funcionario = funcionario.id
    JOIN tem_venda ON venda.id = tem_venda.id_venda
    JOIN filial ON venda.id_filial = filial.id
    WHERE venda.data_venda >= CURRENT_DATE - INTERVAL '12 months'
    GROUP BY filial.id, funcionario.id
)

SELECT filial_id, nome_funcionario, total_vendido
FROM VendasPorfuncionario
WHERE rank <= 2;
```

12.3.4 Relatório - 2 Funcionários que menos venderam em cada filial em 12 meses

```
WITH VendasPorfuncionario AS (
    SELECT
        filial.id AS filial_id,
        funcionario.id AS funcionario_id,
        funcionario.nome AS nome_funcionario,
        SUM(tem_venda.quantidade) AS total_vendido,
        RANK() OVER (PARTITION BY filial.id ORDER BY SUM(tem_venda.quantidade) ASC
            ) AS rank

    FROM venda

    JOIN funcionario ON venda.id_funcionario = funcionario.id
    JOIN tem_venda ON venda.id = tem_venda.id_venda
    JOIN filial ON venda.id_filial = filial.id
    WHERE venda.data_venda >= CURRENT_DATE - INTERVAL '12 months'
    GROUP BY filial.id, funcionario.id
)

SELECT filial_id, nome_funcionario, total_vendido
FROM VendasPorfuncionario
WHERE rank <= 2;
```

12.3.5 Relatório - Todas todas as vendas de produtos de todos as filiais juntas em 12 meses

```
SELECT
    SUM(tem_venda.quantidade) AS total_vendido

FROM venda

JOIN tem_venda ON venda.id = tem_venda.id_venda
WHERE venda.data_venda >= CURRENT_DATE - INTERVAL '12 months';
```

12.3.6 Relatório - Todas as vendas de produtos não remédios de todos as filiais juntas em 12 meses

```
SELECT
    SUM(tem_venda.quantidade) AS total_vendido

FROM venda

JOIN tem_venda ON venda.id = tem_venda.id_venda
JOIN produto ON tem_venda.id_produto = produto.id
WHERE venda.data_venda >= CURRENT_DATE - INTERVAL '12 months'
AND produto.tipo_remedio IS NULL;
```

12.3.7 Relatório - Todas as vendas de remédios de todos as filiais juntas em 12 meses

```
SELECT
    SUM(tem_venda.quantidade) AS total_vendido

FROM venda

JOIN tem_venda ON venda.id = tem_venda.id_venda
JOIN produto ON tem_venda.id_produto = produto.id
WHERE venda.data_venda >= CURRENT_DATE - INTERVAL '12 months'
AND produto.tipo_remedio IS NOT NULL;
```

13 Tabelas Disponíveis no Nível de Aplicação

A otimização e organização das consultas no nível da aplicação são fundamentais para melhorar a experiência do usuário e a eficiência do sistema. Isso pode ser alcançado através da implementação de views e materialized views, que simplificam o acesso aos dados e melhoram o desempenho de consultas repetitivas.

13.1 Implementação de View

Views são implementadas para proporcionar acesso simplificado a consultas complexas, agregando dados de múltiplas tabelas ou simplificando a lógica de negócios no banco de dados. Abaixo, alguns exemplos:

- View simplificada para incluir e disponibilizar somente dados disponíveis diretamente de cliente e agregados de venda:

```
CREATE VIEW detalhes_cliente AS
SELECT
    c.id,
    c.nome,
    c.email,
    c.telefone,
    SUM(v.valor_total) AS valor_total
FROM
    cliente c
LEFT JOIN
    venda v ON c.id = v.id_cliente
GROUP BY
    c.id, c.nome, c.email, c.telefone;
```

```
COMMENT ON VIEW detalhes_cliente IS $$
Esta view combina informações de clientes com detalhes de suas compras
    para fornecer um panorama completo de cada cliente. Inclui dados
    pessoais e a soma total das vendas, facilitando análises detalhadas
    de comportamento do cliente e histórico de compras. Útil para aná
    lises financeiras e de marketing.
$$;
```

- View para controle de estoque e validade de produtos: Facilita a visualização dos produtos próximos da data de validade.

```
CREATE VIEW estoque_produto_validade AS

SELECT p.nome, p.marca, p.data_validade, COUNT(l.id) AS
    quantidade_estoque

FROM produto p

JOIN lote l ON p.id = l.id_produto
WHERE p.data_validade > CURRENT_DATE
GROUP BY p.id;
```

13.2 Implementação de Materialized View

Materialized views são úteis para melhorar o desempenho de consultas que não exigem dados em tempo real, armazenando os resultados da consulta no banco de dados. Abaixo, alguns exemplos:

- Materialized view para relatório de vendas mensais: Armazena os totais de vendas mensais, reduzindo a carga de consulta repetitiva em dados históricos.

```
CREATE MATERIALIZED VIEW vendas_mensais AS

SELECT EXTRACT(MONTH FROM data_venda) AS mes,
       EXTRACT(YEAR FROM data_venda) AS ano,
       SUM(valor_total) AS total_vendas

FROM venda

GROUP BY EXTRACT(MONTH FROM data_venda), EXTRACT(YEAR FROM data_venda);
```

- Materialized view para análise de desempenho de funcionários: Armazena informações sobre as vendas realizadas por cada funcionário, facilitando a avaliação de desempenho.

```
CREATE MATERIALIZED VIEW desempenho_funcionario AS

SELECT f.id, f.nome, COUNT(v.id) AS total_vendas, SUM(v.valor_total) AS
    valor_vendas

FROM funcionario f

JOIN venda v ON f.id = v.id_funcionario
GROUP BY f.id;
```

Essas implementações permitem um acesso mais eficiente e organizado aos dados, melhorando a performance das consultas e a usabilidade no nível da aplicação.

14 Functions

Este tópico se dedica a abordar as diversas funcionalidades que o sistema necessita para operar de maneira eficiente e automatizada. Através do uso de funções no banco de dados, é possível implementar uma série de processos automáticos que otimizam as operações diárias, garantem a integridade dos dados e melhoram a experiência do usuário.

14.1 Necessidades do Sistema

Functions (nome)	Necessidade
atualizar_salario_mensal_funcionarios()	Atualizar automaticamente o salário mensal com base nas comissões das vendas.
setar_status_aniversario()	Fornecer aos clientes cadastrados desconto de 30% caso seja o seu aniversário.
resetar_status_aniversario_ano_novo()	Resetar o "status_aniversario"para NULL de todos os clientes a cada novo ano.
aplicar_desconto_proximos_vencimento()	Vender produtos que estão a 45 dias para vencimento com 30% de desconto.
descartar_produtolote_vencimento()	Descartar os produtos e lotes correspondentes que estão a 30 dias para vencer.

Tabela 32: Legenda da tabela

14.2 Function - 01

A ideia central dessa "Function"é atualizar automaticamente o salário mensal com base nas comissões das vendas, sabendo-se que este cálculo deve ser realizado todo início de mês.

```
CREATE OR REPLACE FUNCTION atualizar_salario_mensal_funcionarios()
RETURNS VOID AS $$
DECLARE
    funcionario_record RECORD;
BEGIN
    FOR funcionario_record IN SELECT id, salario_fixo, comissao FROM funcionario
        LOOP
        DECLARE
            total_vendas DECIMAL(10,2) DEFAULT 0;
            comissao_total DECIMAL(10,2);
        BEGIN
            -- Calcula o total de vendas do funcionário no mês atual
            SELECT COALESCE(SUM(valor_total), 0) INTO total_vendas
            FROM venda
            WHERE id_funcionario = funcionario_record.id
            AND EXTRACT(MONTH FROM data_venda) = EXTRACT(MONTH FROM CURRENT_DATE)
            AND EXTRACT(YEAR FROM data_venda) = EXTRACT(YEAR FROM CURRENT_DATE);

            -- Calcula o total da comissão
            comissao_total := total_vendas * funcionario_record.comissao / 100;

            -- Atualiza o salário mensal do funcionário
            UPDATE funcionario
            SET salario_mensal = funcionario_record.salario_fixo + comissao_total
            WHERE id = funcionario_record.id;
        END;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

COMMENT ON FUNCTION atualizar_salario_mensal_funcionarios() IS $$
Esta função calcula e atualiza automaticamente o salário mensal dos funcionários
com base nas comissões obtidas das vendas do mês anterior. É ideal executar esta
```

```
função no início de cada mês para assegurar que os salários sejam ajustados
conforme o desempenho de vendas. $$;
```

14.2.1 Necessidade da Aplicação - pgAgent

A automação de tarefas recorrentes é essencial para a manutenção eficiente de sistemas complexos de banco de dados. No contexto do nosso sistema de gestão de farmácias, há várias operações que necessitam ser executadas periodicamente. Entre elas, destaca-se a atualização mensal do salário dos funcionários com base em suas comissões de vendas. Esta tarefa, não apenas é tediosa e suscetível a erros se realizada manualmente, mas também é crítica para a satisfação e motivação dos funcionários.

Para automatizar este processo, utilizamos o *pgAgent*, um agente de agendamento de tarefas para o PostgreSQL. O *pgAgent* permite configurar a execução automática de scripts SQL, procedimentos armazenados ou comandos em intervalos especificados, reduzindo significativamente a possibilidade de esquecimento ou erros na execução de tarefas críticas.

Configuração do pgAgent para Atualização de Salários Para implementar a atualização automática dos salários mensais dos funcionários, é necessário configurar um *job* no pgAgent que execute a função **atualizar_salario_mensal_funcionarios()** no primeiro dia de cada mês. O processo de configuração envolve os seguintes passos:

1. Instalação e configuração do *pgAgent* no ambiente PostgreSQL, caso ainda não esteja configurado.
2. Criação de um *job* no pgAgent com um nome descritivo, como "Atualização de Salários Mensais".
3. Adição de um *step* ao *job* criado para executar a função SQL **atualizar_salario_mensal_funcionarios()**, assegurando que a operação seja realizada no contexto do banco de dados correto.
4. Configuração do agendamento (*schedule*) do *job* para que ocorra no dia 1 de cada mês, preferencialmente durante um período de baixa atividade do sistema, para minimizar o impacto sobre outras operações.

Este procedimento assegura que os salários dos funcionários sejam atualizados automaticamente, refletindo com precisão as comissões ganhas no período anterior, sem necessidade de intervenção manual, otimizando operações do sistema e garantindo a acurácia dos dados.

Benefícios da Automatização A utilização do pgAgent para automatizar a atualização de salários mensais dos funcionários apresenta vários benefícios:

- **Redução de Erros:** Minimiza erros humanos associados à atualização manual de salários.
- **Eficiência Operacional:** Libera a equipe de TI para se concentrar em tarefas mais estratégicas, ao invés de operações repetitivas.
- **Satisfação dos Funcionários:** Garante que os salários sejam atualizados pontualmente e com precisão, contribuindo para a satisfação e motivação da equipe.

Assim, o uso do pgAgent é uma estratégia chave na otimização da administração do sistema, alinhando tecnologia e necessidades operacionais de forma eficaz.

14.3 Function - 02

A ideia central dessa "Function"é fornecer aos clientes cadastrados um desconto de 30% em todos os produtos, caso seja o seu aniversário na primeira compra. A aplicação desse desconto é controlada pelo **status_aniversario** no registro do cliente, que é setado para **TRUE** diariamente para os clientes que fazem aniversário naquela data, permitindo que recebam o desconto em sua primeira compra do dia. Essa funcionalidade é realizada em conjunto com a trigger 02, que se encarrega de aplicar o desconto e atualizar o **status_aniversario** para **FALSE** após a primeira compra, prevenindo descontos subsequentes no mesmo dia.

```
CREATE OR REPLACE FUNCTION setar_status_aniversario()
RETURNS VOID AS $$

BEGIN
    -- Atualiza status_aniversario para TRUE para clientes que fazem aniversário
       hoje
    UPDATE cliente
    SET status_aniversario = TRUE
    WHERE EXTRACT(MONTH FROM data_nascimento) = EXTRACT(MONTH FROM CURRENT_DATE)
      AND EXTRACT(DAY FROM data_nascimento) = EXTRACT(DAY FROM CURRENT_DATE);
END;

$$ LANGUAGE plpgsql;

COMMENT ON FUNCTION setar_status_aniversario() IS $$
Atualiza o status_aniversario para TRUE para todos os clientes que fazem
aniversário na data atual. Isso permite que descontos de aniversário sejam
aplicados automaticamente na primeira compra do cliente no seu dia de aniversário.
    $$;
```

14.3.1 Necessidade da Aplicação - pgAgent

Para garantir que os clientes elegíveis recebam o desconto de 30% em suas compras no dia do seu aniversário, a execução da função `setar_status_aniversario()` precisa ser automatizada para ocorrer diariamente. O uso do pgAgent, uma ferramenta de agendamento de tarefas para o PostgreSQL, facilita essa automação de maneira eficaz e confiável.

Configurar essa automação com o pgAgent envolve os seguintes passos:

1. No pgAdmin, navegue até o servidor PostgreSQL onde deseja criar o job e expanda o nó correspondente.
2. Localize e expanda o nó "pgAgent Jobs", clique com o botão direito do mouse e selecione "Create» "Job".
3. Dê um nome ao job, como "ResetarStatusAniversarioDiariamente".
4. Crie um step para o job: com o job selecionado, clique com o botão direito e escolha "Create» "Step". Insira um nome para o step, selecione o tipo como "sql", e no campo "Definition", insira o comando **SELECT setar_status_aniversario();**.
5. Configure o agendamento do job: com o job ainda selecionado, clique com o botão direito e escolha "Create» "Schedule". Configure o agendamento para ser executado diariamente, escolhendo a frequência apropriada e definindo a hora de início preferida.

Essa configuração assegura a execução diária da função, aplicando o desconto aos clientes que fazem aniversário no dia atual, sem a necessidade de intervenção manual. É uma solução prática e eficaz para melhorar a experiência do cliente e promover vendas adicionais em datas comemorativas.

14.4 Function - 03

A ideia central dessa "Function"é resetar o "status_aniversario"para NULL (ou FALSE, assumindo que a estrutura da tabela utilize um BOOLEAN) de todos os clientes no primeiro dia de cada novo ano, com o intuito de poder fornecer descontos aos aniversariantes novamente. Essa abordagem garante que o benefício de aniversário seja renovado anualmente para cada cliente.

```
CREATE OR REPLACE FUNCTION resetar_status_aniversario_ano_novo()
RETURNS VOID AS $$

BEGIN
    -- Atualiza status_aniversario para NULL (ou FALSE) para todos os clientes
    UPDATE cliente
    SET status_aniversario = FALSE;
END;

$$ LANGUAGE plpgsql;

COMMENT ON FUNCTION resetar_status_aniversario_ano_novo() IS $$
Esta função zera o status_aniversario de todos os clientes no primeiro dia de cada
ano novo. Garante que todos os clientes sejam elegíveis para receber descontos de
aniversário novamente no próximo ano. $$;
```

14.4.1 Necessidade da Aplicação - pgAgent

Para garantir que a função **resetar_status_aniversario_ano_novo()** seja executada automaticamente no primeiro dia de cada novo ano, é necessário configurar um agendamento no pgAgent. Esse processo automatiza a execução da função, assegurando que todos os clientes sejam elegíveis para receber os descontos de aniversário no início de cada ano.

Configurar essa automação com o pgAgent envolve os seguintes passos:

1. No pgAdmin, navegue até o servidor PostgreSQL onde deseja criar o job e expanda o nó correspondente.
2. Localize e expanda o nó "pgAgent Jobs", clique com o botão direito do mouse e selecione "Create» "Job".
3. Nomeie o job, por exemplo, "ResetarStatusAniversarioAnoNovo".
4. Crie um step para o job: com o job selecionado, clique com o botão direito e escolha "Create» "Step". Dê um nome ao step, defina o tipo como "sql", e no campo "Definition", insira o comando **SELECT resetar_status_aniversario_ano_novo();**.
5. Configure o agendamento do job: com o job ainda selecionado, clique com o botão direito e escolha "Create» "Schedule". Configure o agendamento para ser executado anualmente, no primeiro dia do ano, ajustando a frequência e a hora de início conforme necessário.

Essa configuração garante a execução anual da função, renovando a elegibilidade para descontos de aniversário para todos os clientes sem necessidade de intervenção manual.

14.5 Function - 04

A ideia central é vender produto a 30% de desconto caso falte 45 dias para, só que quando chegar a 30 dias descartar produto.

- **Vender a 30% de desconto:**

```
CREATE OR REPLACE FUNCTION descartar_produtolote_vencimento()
RETURNS VOID AS $$
BEGIN
    UPDATE produto SET preco = preco * 0.7
    WHERE data_validade BETWEEN CURRENT_DATE + INTERVAL '1 day' AND
        CURRENT_DATE + INTERVAL '45 days';
END;
$$ LANGUAGE plpgsql;

COMMENT ON FUNCTION descartar_produtolote_vencimento() IS $$
Esta função atualiza o preço dos produtos que estão entre 1 e 45 dias
antes da data de validade, aplicando um desconto de 30%. Destina-se a
incentivar a venda de produtos próximos ao vencimento e minimizar
perdas. $$;
```

- **Descartar quando chegar a 30 dias do vencimento (produto e lote):**

```
CREATE OR REPLACE FUNCTION
    descartar_completamente_produtolote_vencimento()
RETURNS VOID AS $$
BEGIN
    -- Primeiro, exclui os lotes dos produtos que serão descartados
    DELETE FROM lote
    WHERE id_produto IN (
        SELECT id FROM produto
        WHERE data_validade BETWEEN CURRENT_DATE + INTERVAL '1 day'
        AND CURRENT_DATE + INTERVAL '30 days'
    );

    -- Em seguida, exclui os produtos próximos do vencimento
    DELETE FROM produto
    WHERE data_validade BETWEEN CURRENT_DATE + INTERVAL '1 day'
    AND CURRENT_DATE + INTERVAL '30 days';
END;
$$ LANGUAGE plpgsql;

COMMENT ON FUNCTION descartar_produtolote_vencimento() IS $$
Esta função exclui produtos e lotes que estão entre 1 e 30 dias antes
da data de validade. A função é usada para garantir que produtos
expirados sejam removidos do estoque e não sejam vendidos aos
consumidores. $$;
```

14.5.1 Necessidade da Aplicação - pgAgent

Para executar essas funções automaticamente, você pode usar o pgAgent ou um cron job no sistema operacional do servidor de banco de dados.

- **Usando pgAgent:** Você precisaria criar um job no pgAgent através do PgAdmin, configurando-o para executar as funções diariamente.
- **Usando cron (Linux):** Você pode agendar um cron job para chamar essas funções usando o comando psql. Por exemplo, para executar a função de aplicar desconto todos os dias à meia-noite:

```
0 0 * * * psql -d projetoredefarmacia -c 'SELECT
    aplicar_desconto_proximos_vencimento();'
```

15 Triggers

Neste tópico, visa-se explorar todas as necessidades do sistema, no que diz respeito às automações que podem ser realizadas utilizando "triggers"no banco de dados. Desde então, foi elaborado todas as cruciais para o pleno funcionamento do sistema, tanto quanto especificado a necessidade de cada uma.

15.1 Necessidades do Sistema

Triggers (nome)	Necessidade
diminuir_quantidade_no_lote()	Atualizar a quantidade de produtos no lote, de acordo com as vendas.
atualizar_valor_total_venda()	Atualizar o valor total da venda com relação ao desconto e ao aniversário do cliente.
verificar_atualizar_contato_funcionario()	Inserir e atualizar corretamenta telefone e e-mail.
permissao_venda()	Permitir a venda apenas para funcionários de cargo vendedor e farmacêutico.
verificar_atualizar_contato_cliente()	Inserir e atualizar corretamenta telefone e e-mail.
verifica_validade_produto	Garantir que produtos vencidos não sejam adicionados ao estoque.

Tabela 33: Legenda da tabela

15.2 Trigger - 01

A ideia central dessa trigger é diminuir a quantidade de produtos no lote, de acordo com as vendas.

```
CREATE OR REPLACE FUNCTION diminuir_quantidade_no_lote()
RETURNS TRIGGER AS $$
DECLARE
    loteRecente DATE;
BEGIN
    -- Verifica se a quantidade atual no lote é suficiente
    IF (SELECT quantidade FROM lote WHERE id_produto = NEW.id_produto ORDER BY
        data_despacho DESC LIMIT 1) >= NEW.quantidade THEN

        -- Obtendo a data do despacho mais recente
        SELECT data_despacho INTO loteRecente FROM lote WHERE id_produto = NEW.
            id_produto ORDER BY data_despacho DESC LIMIT 1;

        -- Diminui a quantidade do lote mais recente
        UPDATE lote
        SET quantidade = quantidade - NEW.quantidade
        WHERE id_produto = NEW.id_produto AND data_despacho = loteRecente;

    ELSE
        -- Lança um erro se a venda exceder a quantidade disponível no lote mais
            recente
        RAISE EXCEPTION 'Quantidade insuficiente no lote para o produto %', NEW.
            id_produto;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_diminuir_quantidade_no_lote
BEFORE INSERT ON tem_venda
FOR EACH ROW
EXECUTE FUNCTION diminuir_quantidade_no_lote();
```

```
COMMENT ON FUNCTION diminuir_quantidade_no_lote() IS $$
Esta trigger é responsável por atualizar a quantidade de produtos em estoque no
lote mais recente após uma venda ser realizada. Ela verifica se a quantidade
disponível é suficiente e diminui essa quantidade no lote de acordo com a
quantidade vendida. Um erro é lançado caso a venda exceda a quantidade disponível.
    $$;
```

15.3 Trigger - 02

A ideia central dessa trigger é atualizar o valor total da venda com relação ao desconto e com relação ao aniversário do cliente.

```
CREATE OR REPLACE FUNCTION
    atualizar_valor_total_venda_e_aplicar_desconto_aniversario()
RETURNS TRIGGER AS $$
DECLARE
    valorDesconto DECIMAL(10,2) := 0.00; -- Inicializa o desconto com zero
    valorProduto MONEY;
    novoValorTotal MONEY;
    aniversario BOOL;
BEGIN
    -- Verifica se hoje é aniversário do cliente e se o status_aniversario é TRUE
    SELECT (EXTRACT(MONTH FROM data_nascimento) = EXTRACT(MONTH FROM CURRENT_DATE)
            AND EXTRACT(DAY FROM data_nascimento) = EXTRACT(DAY FROM CURRENT_DATE)
            AND status_aniversario = TRUE)
    INTO aniversario
    FROM cliente
    WHERE id = (SELECT id_cliente FROM venda WHERE id = NEW.id_venda);

    -- Determina o desconto aplicável
    IF aniversario THEN
        valorDesconto := 0.30; -- Desconto de 30% para aniversário

        -- Atualiza o status_aniversario para FALSE após aplicar o desconto
        UPDATE cliente
        SET status_aniversario = FALSE
        WHERE id = (SELECT id_cliente FROM venda WHERE id = NEW.id_venda);
    ELSE
        valorDesconto := NEW.desconto; -- Utiliza o desconto original da venda, se
            não for aniversário
    END IF;

    -- Calcula o valor do produto vendido multiplicado pela quantidade e aplica o
        desconto
    SELECT preco INTO valorProduto FROM produto WHERE id = NEW.id_produto;
    novoValorTotal = valorProduto * NEW.quantidade * (1 - valorDesconto);

    -- Atualiza o valor total na venda com ou sem desconto de aniversário
    UPDATE venda
    SET valor_total = valor_total + novoValorTotal
    WHERE id = NEW.id_venda;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_atualizar_valor_total_venda_e_aplicar_desconto_aniversario
AFTER INSERT ON tem_venda
```

```
FOR EACH ROW
EXECUTE FUNCTION atualizar_valor_total_venda_e_aplicar_desconto_aniversario();

COMMENT ON FUNCTION atualizar_valor_total_venda_e_aplicar_desconto_aniversario()
    IS $$
Esta trigger é acionada após a inserção de uma venda e verifica se o cliente está
fazendo aniversário, aplicando um desconto de 30% se for o caso. Ela também
atualiza o valor total da venda, considerando os descontos aplicados, e reseta o
status de aniversário do cliente para evitar descontos múltiplos no mesmo dia. $$;
```

15.4 Trigger - 03

Este trigger pode ser invocado após uma inserção ou atualização na tabela funcionário, para assegurar que o telefone e o e-mail seguem um formato específico ou para realizar alguma ação corretiva.

```
CREATE OR REPLACE FUNCTION verificar_atualizar_contato_funcionario()
RETURNS TRIGGER AS $$
BEGIN
    -- Exemplo de verificação de formato de telefone
    IF NEW.telefone !~ '^\d{11}$' THEN
        RAISE EXCEPTION 'Formato de telefone inválido para funcionário: %', NEW.
            telefone;
    END IF;

    -- Exemplo de verificação de formato de email
    IF NEW.email NOT LIKE '%@%' THEN
        RAISE EXCEPTION 'Formato de email inválido para funcionário: %', NEW.email
            ;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER verificar_contato_funcionario
BEFORE INSERT OR UPDATE ON funcionario
FOR EACH ROW EXECUTE FUNCTION verificar_atualizar_contato_funcionario();

COMMENT ON FUNCTION verificar_atualizar_contato_funcionario() IS $$
Esta trigger é usada para garantir que os dados de contato dos funcionários, como
    telefone e e-mail, estejam em formatos válidos antes de serem inseridos ou
    atualizados na tabela de funcionários. A função impede a inserção ou atualizaç
    ão de dados inválidos, lançando uma exceção se os formatos não atenderem aos
    critérios especificados. $$;
```

15.5 Trigger - 04

A ideia central dessa trigger é permitir a venda apenas para funcionários de cargo vendedor e farmacêutico.

```
CREATE OR REPLACE FUNCTION permissao_venda() RETURNS TRIGGER AS $$
DECLARE
        func RECORD;
BEGIN
        SELECT * INTO func FROM funcionario WHERE id = new.id_funcionario;
        IF func.cargo = 'Faxineira' THEN
                raise notice 'Uma faxineira não pode realizar as vendas da farmá
                    cia';
```

```
                RETURN NULL;
        ELSE
                RETURN NEW;
        END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_permissao_venda
BEFORE INSERT ON venda
FOR EACH ROW
EXECUTE FUNCTION permissao_venda();

COMMENT ON FUNCTION permissao_venda() IS $$
Esta trigger verifica o cargo do funcionário antes de permitir que uma venda seja
    registrada.
Se o funcionário tiver um cargo não autorizado a realizar vendas, como 'Faxineira'
    , a venda será
bloqueada e um aviso será emitido. A venda prossegue normalmente para cargos
    autorizados, como
vendedores e farmacêuticos. $$;
```

15.6 Trigger - 05

Este trigger pode ser invocado após uma inserção ou atualização na tabela cliente, para assegurar que o telefone e o e-mail seguem um formato específico ou para realizar alguma ação corretiva.

```
CREATE OR REPLACE FUNCTION verificar_atualizar_contato_cliente()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificação de formato de telefone
    IF NEW.telefone !~ '^\d{11}$' THEN
        RAISE EXCEPTION 'Formato de telefone inválido para cliente: %', NEW.
            telefone;
    END IF;

    -- Verificação de formato de email
    IF NEW.email NOT LIKE '%@%' THEN
        RAISE EXCEPTION 'Formato de email inválido para cliente: %', NEW.email;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER verificar_contato_cliente
BEFORE INSERT OR UPDATE ON cliente
FOR EACH ROW EXECUTE FUNCTION verificar_atualizar_contato_cliente();

COMMENT ON FUNCTION verificar_atualizar_contato_cliente() IS $$
Esta trigger é utilizada para garantir que os dados de contato dos clientes, como
    telefone e e-mail, estejam em formatos válidos antes de serem inseridos ou
    atualizados na tabela de clientes. A função impede a inserção ou atualização
    de dados inválidos, lançando uma exceção se os formatos não atenderem aos crit
    érios especificados. $$;
```

15.7 Trigger - 06

Para garantir que produtos vencidos não sejam adicionados ao estoque, uma trigger que verifica a data de validade dos produtos contra a data atual antes de inserir um novo lote.

```
CREATE OR REPLACE FUNCTION verifica_validade_produto()
RETURNS TRIGGER AS $$
BEGIN
    -- Verifica se a data de despacho é posterior    data de validade do produto
    IF NEW.data_despacho > (SELECT data_validade FROM produto WHERE id = NEW.
        id_produto) THEN
        RAISE EXCEPTION 'Não é possível inserir lote com produtos vencidos.';
    END IF;
    RETURN NEW; -- Retorna o novo registro para ser inserido
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER verifica_validade_antes_inserir_lote
BEFORE INSERT ON lote
FOR EACH ROW
EXECUTE FUNCTION verifica_validade_produto();

COMMENT ON FUNCTION verifica_validade_produto() IS $$
Esta trigger é acionada antes de inserir um novo lote na tabela de lotes. Ela
    verifica se a data de despacho do lote é anterior    data de validade do
    produto associado. Caso a data de despacho seja posterior    data de validade,
     a inserção é bloqueada e uma exceção é lançada para prevenir o armazenamento
    de produtos vencidos. $$;
```

16 Controle de Concorrência

Para abordar o controle de concorrência com base no projeto da farmácia, precisamos considerar os aspectos de como os dados são acessados e modificados nas tabelas, bem como o ambiente em que essas operações ocorrem (por exemplo, uma aplicação de alto volume de transações).

16.1 Identificação das Tabelas

As tabelas que mais provavelmente necessitam de um controle de concorrência refinado são aquelas que sofrem frequentes inserções, atualizações ou deleções. Têm relacionamentos chave com outras tabelas, especialmente em operações que envolvem atualizações em cascata ou restrições de chave estrangeira. São acessadas concorrentemente por múltiplos usuários ou processos, podendo causar condições de corrida ou inconsistências. Com base no esquema fornecido, as tabelas críticas para o controle de concorrência incluem:

- **lote:** A tabela lote exige o nível REPEATABLE READ ou SERIALIZABLE devido à sua alta frequência de inserções e atualizações, as quais podem levar a conflitos de concorrência, como a inserção de lotes duplicados ou a atualização inconsistente de quantidades de lotes. SERIALIZABLE é particularmente crítico aqui para prevenir leituras fantasmas e garantir a consistência durante a inserção de novos lotes ou a atualização de lotes existentes, mantendo a integridade do estoque.
- **produto:** Para a tabela produto, utilizar REPEATABLE READ pode prevenir a leitura não repetível, assegurando que as informações do produto, como preço e estoque, permaneçam consistentes durante a transação. O uso de SERIALIZABLE, no entanto, é essencial para operações que envolvem a atualização crítica ou a inserção de produtos, onde é imperativo garantir a total consistência e evitar conflitos de

concorrência, como atualizações simultâneas do estoque ou inserções que poderiam afetar a unicidade dos dados.

16.2 Mudanças em Níveis de Isolamento

Para controlar a concorrência e prevenir problemas como sujeira de leitura, leituras não repetíveis e fantasmas, pode-se ajustar os níveis de isolamento das transações. Os níveis de isolamento variam de Read Uncommitted a Serializable, cada um com seus trade-offs entre consistência e performance.

Considerações para as tabelas identificadas:

- **Read Committed:** Pode ser suficiente para a maioria das operações que não exigem a leitura de dados não confirmados, reduzindo o risco de leituras sujas. Este nível pode ser a escolha padrão para um sistema com requisitos de consistência moderados.
- **Read:** Necessário para cenários onde as transações precisam garantir que os dados lidos não mudem durante a transação. Útil para operações em venda e produto onde a consistência dos dados entre leituras é crucial.
- **Serializable:** O nível mais alto de isolamento, evitando a maioria dos problemas de concorrência, mas com o maior impacto na performance. Recomendado para transações críticas que envolvem múltiplas tabelas, como tem_venda, onde a consistência e a integridade dos dados são prioritárias.

16.3 REPEATABLE READ

Para evitar leituras não repetíveis, o nível de isolamento REPEATABLE READ pode ser utilizado:

- **Tabela lote:**

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
INSERT INTO lote (id_produto, nome, quantidade, data_despacho, id_filial)
VALUES (2, 'Lote Exemplo REPEATABLE READ', 150, '2024-04-15', 2);
COMMIT;
```

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
UPDATE lote SET quantidade = quantidade + 30 WHERE id = 2;
COMMIT;
```

- **Tabela produto:**

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
INSERT INTO produto (nome, preco, marca, tipo_remedio, prescricao_medica,
    tipo_tarja, forma_de_administracao, composicao_remedio, data_validade,
    data_fabricacao, secao)
VALUES ('Remédio', '99.99', 'Marca', 'Analgesico', FALSE, 'Sem Tarja', 'Oral'
    , 'Composicao', '2025-12-31', '2024-01-01', 'Medicamentos');
COMMIT;
```

```
BEGIN;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

UPDATE produto
SET preco = '150.00'
```

```
WHERE id = 1;

COMMIT;
```

16.4 SERIALIZABLE

Para garantir a consistência total e evitar leituras fantasmas, o nível de isolamento SERIALIZABLE é recomendado:

- **Tabela lote:**

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
INSERT INTO lote (id_produto, nome, quantidade, data_despacho, id_filial)
VALUES (1, 'Lote Novo Exemplo', 100, '2024-04-10', 1);
COMMIT;
```

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE lote SET quantidade = quantidade + 50 WHERE id = 1;
COMMIT;
```

- **Tabela produto:**

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
INSERT INTO produto (nome, preco, marca, tipo_remedio, prescricao_medica,
    tipo_tarja, forma_de_administracao, composicao_remedio, data_validade,
    data_fabricacao, secao)
VALUES ('Remédio', '99.99', 'Marca', 'Analgesico', FALSE, 'Sem Tarja', 'Oral'
    , 'Composicao', '2025-12-31', '2024-01-01', 'Medicamentos');
COMMIT;
```

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE produto
SET preco = '150.00'
WHERE id = 1;
```

17 Configuração de Captura de Logs

Para garantir uma monitoração eficaz do sistema, diversas configurações de log devem ser ativadas no PostgreSQL.

17.1 Ativando Logs Detalhados

Editar o arquivo `postgresql.conf` para incluir:

```
 Log de consultas lentaslog_min_duration_statement = 5000  Milissegundos

 Logs de bloqueiolog_lock_waits = on Log de conexões e desconexões
log\_connections = on
log\_disconnections = on
```

Após ajustar as configurações, é necessário reiniciar o serviço do PostgreSQL.

18 Configuração de Arquivos WAL

O gerenciamento apropriado dos arquivos WAL é crucial para a recuperação de dados e a replicação.

18.1 Definindo o Nível de WAL

```
wal\_level = logical
```

18.2 Habilitando o Arquivamento de WAL

```
archive\_mode = on
archive\_command = 'cp %p /path_to_archive/%f'
```

Certifique-se de substituir `/path_to_archive/` pelo caminho desejado para armazenamento dos arquivos WAL.

18.3 Ajustes Adicionais de WAL

Outros parâmetros, como `max_wal_size` e `min_wal_size`, podem ser configurados para otimizar o desempenho do banco de dados.

19 Backup

19.1 Contexto

Como estamos em um contexto empresarial, todos os dados são criticos. O manejo e armazenamento de dados sensíveis, particularmente informações relacionadas às entradas e saídas de medicamentos, são de extrema importância para a farmácia. Estes dados não só requerem rigorosa precisão para a gestão do estoque e controle de qualidade, mas também são essenciais para a prestação de contas a entidades reguladoras e auditores. Um backup confiável garante a integridade dos dados de forma continua afim de analisar e projetar a empresa para um futuro lucrativo. Manter dados dos nossos clientes seguros e salvos, para entender a nossa base de clientes e expandir a empresa. Então salvamos o banco todo, de forma segura para que se caso ocorra alguma tipo de desastre, possamos recuperar.

19.2 Quais Estratégias

Usamos o **pg_dump** para realizar o backup do nosso banco de dados. Essa escolha nos permite ter uma flexibilidade na restauração dos dados, seja em nivel de banco completo ou em objetos objetos do banco. automatizamos isso criando um arquivo chamado .bat e criando uma tarefa pelo windows. Obtamos por esse metodo devido a quantidade de dados e a simplicidade no uso. Porém para grandes volumes de dados seguiriamos com uma estratégia um pouco diferente. Primeiro iriamos fazer backups em nuvem, utilizariamos o **pg_basebackup** que transforma os dados em binario, que ajudaria no volume dos dados.

19.3 Como fazer com Postgresql

Existe varias maneiras de fazer um backup no banco de dados no PostgreSQL. O método utilizado foi o automatizado. O passo a seguir serve apenas para sistema windows

- **Passo - 1:** Abrir um bloco de notas
- **Passo - 2:** e colocar o seguinte texto (alterado para seu sistema):

```
@echo off
set PGPASSWORD=sua_senha
"C:\caminho\para\postgresql\bin\pg\_dump.exe" -U nome\_usuario -h
    localhost -p 5432 nome\_do\_banco > nome\_do\_banco\_backup.sql
```

- **Passo - 3** feito isso salve o arquivo em '.bat'.
- **Passo - 4** Criar uma tarefa no windows
  - **4.1 - Abrir o Agendador de Tarefas:** Pressione 'Windows + R', digite 'taskschd.msc'
  - **4.2 - Criar Tarefa:** No painel à direita, clique em "Criar Tarefa..."
  - **4.3 - Gatilhos:** Gatilhos: Vá para a aba "Gatilhos" e clique em "Novo...". Configure quando você quer que o backup seja executado (diariamente, semanalmente, etc.).
  - **4.4 - Ações:** Vá para a aba "Ações" e clique em "Novo...". Em "Programa/script", clique em "Procurar..." e selecione o arquivo .bat que você criou
- **Salve**
- **Lembrete** Troque os caminhos e a senha.

Para fazer o processo de recuperação podemos criar um arquivo **'.bat'**. Mas recomendamos fazer esse processo manualmente.

19.4 Quais ferramentas usar para automatizar

As ferramentas foi o Sistema Operacional Windows e o próprio PostgreSQL.

20 TABLESPACE

No PostgreSQL, um tablespace é uma localização no sistema de arquivos onde o banco de dados armazena seus dados. O conceito de tablespace é útil para administradores de banco de dados por várias razões, principalmente para a gestão de armazenamento dos dados. Aqui estão alguns pontos sobre tablespaces:

- **Organização Física:** Tablespaces permitem que os administradores definam uma localização no sistema de arquivos para os arquivos de banco de dados, o que pode ajudar na organização dos dados em diferentes dispositivos de armazenamento.
- **Desempenho:** Ao distribuir dados entre múltiplos dispositivos de armazenamento (por exemplo, diferentes discos ou arrays de discos), é possível melhorar o desempenho do banco de dados, já que você pode reduzir os gargalos de I/O ao distribuir a carga de leitura e escrita.
- **Segurança e Backup:** É possível aumentar a segurança e facilitar o backup separando dados críticos em tablespaces diferentes, o que pode simplificar a aplicação de políticas de segurança e estratégias de backup/restauração.
- **Flexibilidade de Armazenamento:** Com tablespaces, os administradores podem gerenciar o espaço de armazenamento de maneira mais flexível, como adicionar mais espaço conforme necessário sem afetar o restante do sistema.

- **Quotas de Espaço:** Embora o PostgreSQL não suporte quotas de espaço de disco nativamente, é possível implementar funcionalidades semelhantes a quotas utilizando tablespaces em conjunto com as limitações do sistema de arquivos do sistema operacional.

20.1 Tabelas com TABLESPACE

Com base na análise detalhada e nos dados extraídos sobre o crescimento projetado, identificamos tabelas-chave para o foco de nossa infraestrutura. Essa seleção foi guiada por projeções estratégicas, permitindo priorizar recursos e otimizar o desempenho, antecipando o crescimento dos dados.

Projeção de 1 ano de operação

- **Filial e Fornecedor:** Estas tabelas, após a inserção inicial, tendem a experimentar um número relativamente baixo de atualizações. Isso se deve principalmente à natureza estática de seus dados, como informações de contato e detalhes cadastrais, que mudam com pouca frequência. Portanto, alocá-las em um tablespace hospedado em HDD (Hard Disk Drive) tradicional é uma escolha eficaz, pois esses dispositivos oferecem uma capacidade de armazenamento superior por um custo menor em comparação aos SSDs. Embora os HDDs tenham velocidades de acesso e leitura/escrita mais lentas do que os SSDs, esse compromisso é aceitável para dados que não exigem acesso constante ou imediato, garantindo assim um armazenamento econômico enquanto mantém o desempenho adequado para as operações esperadas nessas tabelas.

Projeção de 2 anos de operação

- **Funcionario, Cliente, Venda:** Ao centro das operações diárias, essas tabelas são caracterizadas por um alto volume de transações, incluindo frequentes atualizações e consultas. A natureza dessas tabelas demanda um acesso rápido e eficiente para suportar a performance necessária nas operações diárias da empresa. A alocação dessas tabelas em tablespaces em SSDs (Solid-State Drives) é estratégica, pois os SSDs proporcionam tempos de acesso e taxas de transferência significativamente mais rápidos em comparação aos HDDs. Especialmente para operações que envolvem grandes volumes de dados e exigem respostas quase imediatas, como transações de venda e atualizações de dados de funcionários ou clientes. A tecnologia SSD, especialmente quando empregada com NVMe (Non-Volatile Memory Express), que oferece uma interface de acesso direto aos dados via PCIe, pode drasticamente reduzir latências e aumentar a velocidade das transações, tornando-as ideais para essas tabelas dinâmicas.

- **Produto, Lote, Tem_venda, e Fornece:** As operações relacionadas ao gerenciamento de estoque e vendas necessitam de processamentos ágeis devido à sua natureza dinâmica, envolvendo constante atualização e consulta de dados para refletir com precisão o estado atual do estoque e registrar transações de venda em tempo real. Semelhantemente às tabelas mencionadas anteriormente, alocar essas tabelas em tablespaces em SSDs, preferencialmente com tecnologia NVMe, é crucial. Isso não apenas garante a velocidade necessária para operações frequentes e intensivas mas também apoia a integridade e confiabilidade dos dados em um ambiente de rápido movimento. A vantagem de utilizar SSDs NVMe aqui é notável, pois sua capacidade de suportar um número maior de operações de entrada/saída por segundo (IOPS) e a redução nas latências de acesso aos dados podem significativamente melhorar o desempenho das operações de banco de dados que são críticas para o gerenciamento eficiente de estoques e a execução suave das vendas.

Para a criação do tablespace podemos seguir os seguintes passos:

- **Passo - 1: Criar caminho para guarda os dados:**

```
  CREATE TABLESPACE meus_tablespace_HD
  OWNER postgres
  LOCATION '/path/to/tablespace';

  -- PARA SDD
  CREATE TABLESPACE meus_tablespace_SSD
  OWNER postgres
  LOCATION '/path/to/tablespace';
```

- **Passo - 2: Definir quais tabelas iram para o tablespace:**

```
CREATE TABLE filial (
    id SERIAL,
    cnpj VARCHAR(14) UNIQUE,
    site VARCHAR(50),
    telefone VARCHAR(15),
    endereco VARCHAR(200),
    CONSTRAINT pk_id_filial PRIMARY KEY (id)
) TABLESPACE meus_tablespace_HD;

CREATE TABLE fornecedor (
    id SERIAL,
    cnpj CHAR(14) UNIQUE,
    nome VARCHAR(100) NOT NULL,
    telefone CHAR(11) NOT NULL,
    endereco VARCHAR(200),
    portfolio TEXT NOT NULL,
    CONSTRAINT pk_id_fornecedor PRIMARY KEY (id)
)TABLESPACE meus_tablespace_HD;

CREATE TABLE funcionario (
    id SERIAL,
    cpf CHAR(11) UNIQUE,
    rg CHAR(9) UNIQUE,
    nome CHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    telefone CHAR(11),
    certificado BOOLEAN NOT NULL,
    cargo VARCHAR(15) NOT NULL,
    comissao DECIMAL(5, 2) NOT NULL,
    salario_fixo MONEY NOT NULL CHECK (salario_fixo > CAST(0.00 AS MONEY)),
    salario_mensal MONEY NOT NULL,
    id_filial INT NOT NULL,
    endereco VARCHAR(200) NOT NULL,
    CONSTRAINT pk_id_funcionario PRIMARY KEY (id),
    CONSTRAINT fk_id_filial_in_funcionario FOREIGN KEY (id_filial) REFERENCES
        filial (id)
)TABLESPACE meus_tablespace_SSD;

CREATE TABLE cliente (
    id SERIAL,
    cpf CHAR(11) UNIQUE,
    email VARCHAR(100) UNIQUE,
    nome VARCHAR(100) NOT NULL,
    data_nascimento DATE,
    telefone CHAR(11),
    endereco VARCHAR(200),
    status_aniversario BOOL DEFAULT FALSE,
```

```
    CONSTRAINT pk_id_cliente PRIMARY KEY (id)
)TABLESPACE meus_tablespace_SSD;

CREATE TABLE venda (
    id SERIAL,
    id_funcionario INT NOT NULL,
    id_cliente INT NOT NULL,
    data_venda DATE NOT NULL DEFAULT CURRENT_DATE,
    valor_total MONEY NOT NULL,
    nota_fiscal TEXT NOT NULL,
    id_filial INT NOT NULL,
    CONSTRAINT pk_id_venda PRIMARY KEY (id),
    CONSTRAINT fk_id_funcionario_in_venda FOREIGN KEY (id_funcionario)
        REFERENCES funcionario (id),
    CONSTRAINT fk_id_cliente_in_venda FOREIGN KEY (id_cliente) REFERENCES
        cliente (id),
    CONSTRAINT fk_id_filial_in_venda FOREIGN KEY (id_filial) REFERENCES
        filial (id)
)TABLESPACE meus_tablespace_SSD;


CREATE TABLE produto (
    id SERIAL,
    nome VARCHAR(100) NOT NULL,
    preco MONEY NOT NULL CHECK (preco > CAST(0.00 AS MONEY)),
    marca VARCHAR(50) NOT NULL,
    tipo_remedio VARCHAR(20),
    prescricao_medica BOOLEAN,
    tipo_tarja VARCHAR(20),
    forma_de_administracao VARCHAR(50),
    composicao_remedio TEXT,
    data_validade DATE NOT NULL,
    data_fabricacao DATE NOT NULL,
    secao VARCHAR(20) NOT NULL,
    CONSTRAINT pk_id_produto PRIMARY KEY (id)
)TABLESPACE meus_tablespace_SSD;

CREATE TABLE lote (
    id SERIAL,
    id_produto INT NOT NULL,
    nome VARCHAR(50) NOT NULL,
    quantidade INT NOT NULL,
    data_despacho DATE NOT NULL DEFAULT CURRENT_DATE,
    id_filial INT NOT NULL,
    CONSTRAINT pk_composta_lote_produto PRIMARY KEY (id, id_produto),
    FOREIGN KEY (id_produto) REFERENCES produto (id),
    CONSTRAINT fk_id_filial_in_lote FOREIGN KEY (id_filial) REFERENCES filial
         (id)
) TABLESPACE meus_tablespace_SSD;

CREATE TABLE tem_venda (
    id SERIAL,
    id_produto INT NOT NULL,
    id_venda INT NOT NULL,
    quantidade INT NOT NULL,
    preco MONEY NOT NULL,
    desconto DECIMAL(5, 2),
    CONSTRAINT pk_id_tem_venda PRIMARY KEY (id),
    CONSTRAINT uq_produto_venda UNIQUE (id_produto, id_venda),
    CONSTRAINT fk_id_venda_in_tem_venda FOREIGN KEY (id_venda) REFERENCES
```

```
        venda (id),
    CONSTRAINT fk_id_produto_in_tem_venda FOREIGN KEY (id_produto) REFERENCES
         produto (id)
)TABLESPACE meus_tablespace_SSD;


CREATE TABLE fornece (
    id SERIAL,
    id_fornecedor INT NOT NULL,
        id_produto INT NOT NULL,
    id_lote INT NOT NULL,
    CONSTRAINT pk_id_fornece PRIMARY KEY (id),
    CONSTRAINT fk_id_fornecedor_in_fornece FOREIGN KEY (id_fornecedor)
        REFERENCES fornecedor (id),
        CONSTRAINT fk_id_produto_in_fornece FOREIGN KEY (id_lote, id_produto)
            REFERENCES lote (id, id_produto)
) TABLESPACE meus_tablespace_SSD;
```

21 Configuração de Usuários (Roles)

Descreve a criação de roles no PostgreSQL correspondendo às categorias de usuários do sistema.

21.1 Criação de Usuários

```
CREATE ROLE admin WITH LOGIN PASSWORD 'senha_admin';
CREATE ROLE funcionario_filial WITH LOGIN PASSWORD 'senha_funcionario';
CREATE ROLE gestao_filial WITH LOGIN PASSWORD 'senha_gestao';
CREATE ROLE fornecedor WITH LOGIN PASSWORD 'senha_fornecedor';
```

21.2 Atribuição de Permissões

Define as permissões específicas para cada role.

21.2.1 Administração

```
GRANT ALL PRIVILEGES ON DATABASE farmacia TO admin;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA rede_farmaceutica TO admin;
```

21.2.2 Funcionários das Filiais

```
GRANT SELECT, INSERT, UPDATE ON TABLE rede_farmaceutica.venda TO
    funcionario_filial;
GRANT SELECT ON TABLE rede_farmaceutica.produto, rede_farmaceutica.cliente TO
    funcionario_filial;
```

21.2.3 Gestão de Filiais

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE rede_farmaceutica.filial,
    rede_farmaceutica.funcionario, rede_farmaceutica.venda TO gestao_filial;
GRANT SELECT ON TABLE rede_farmaceutica.cliente, rede_farmaceutica.produto TO
    gestao_filial;
```

21.2.4 Fornecedores

```
GRANT SELECT, UPDATE ON TABLE rede_farmaceutica.produto TO fornecedor;
GRANT SELECT ON TABLE rede_farmaceutica.fornecedor TO fornecedor WHERE id =
    CURRENT_USER_ID();
```

21.3 Observação

Esse controle embora expresso no documento como pode ser realizado à nível de banco de dados não está completamente desenvolvido

22 Organização dos Dados por Esquemas

```
CREATE SCHEMA rede_farmaceutica AUTHORIZATION admin;

ALTER TABLE public.venda SET SCHEMA rede_farmaceutica;
ALTER TABLE public.cliente SET SCHEMA rede_farmaceutica;
ALTER TABLE public.produto SET SCHEMA rede_farmaceutica;
ALTER TABLE public.funcionario SET SCHEMA rede_farmaceutica;
ALTER TABLE public.filial SET SCHEMA rede_farmaceutica;
-- Continuando assim para outras demais tabelas conforme necessário
```

23 Auditoria do Sistema

Para garantir a integridade e a segurança do sistema de banco de dados para a rede farmacêutica, é crucial implementar um mecanismo de auditoria robusto. Este mecanismo consistirá na criação de uma tabela de auditoria dedicada, que vai registrar todas as interações e edições realizadas nas tabelas do banco de dados.

23.1 Objetivos da Auditoria

A tabela de auditoria tem vários objetivos importantes:

- Monitorar e registrar todas as ações de CRUD (Create, Read, Update, Delete) realizadas no banco de dados.
- Manter um histórico de quem realizou cada ação, incluindo data e hora da operação.
- Assegurar a rastreabilidade de todas as modificações para facilitar a análise de eventos inesperados ou não autorizados.
- Contribuir para a conformidade regulatória, fornecendo registros detalhados das atividades de dados.

23.2 Estrutura da Tabela de Auditoria

A tabela de auditoria, denominada **auditoria_log**, terá a seguinte estrutura:

```
CREATE TABLE auditoria_log (
    ID BIGSERIAL PRIMARY KEY,
    Tabela_Afetada VARCHAR(255),
```

```
    Tipo_Operacao VARCHAR(255) CHECK (Tipo_Operacao IN ('CREATE', 'READ', 'UPDATE'
        , 'DELETE')),
    Timestamp_Operacao TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    Usuario VARCHAR(255),
    Dados_Antigos TEXT,
    Dados_Novos TEXT
);

COMMENT ON COLUMN auditoria_log.ID IS 'Identificador único para cada registro de
    auditoria.';
COMMENT ON COLUMN auditoria_log.Tabela_Afetada IS 'Nome da tabela onde a operação
    foi realizada.';
COMMENT ON COLUMN auditoria_log.Tipo_Operacao IS 'Tipo de operação realizada (
    CREATE, READ, UPDATE, DELETE).';
COMMENT ON COLUMN auditoria_log.Timestamp_Operacao IS 'Data e hora exatas em que a
     operação foi realizada.';
COMMENT ON COLUMN auditoria_log.Usuario IS 'Identificação do usuário que realizou
    a operação.';
COMMENT ON COLUMN auditoria_log.Dados_Antigos IS 'Representação dos dados antes da
     operação, para operações de UPDATE e DELETE.';
COMMENT ON COLUMN auditoria_log.Dados_Novos IS 'Representação dos dados após a
    operação, para operações de CREATE e UPDATE.';
```

23.3 Descrição dos Campos

- **ID**: Identificador único para cada registro de auditoria.
- **Tabela_Afetada**: Nome da tabela onde a operação foi realizada.
- **Tipo_Operacao**: Tipo de operação realizada (criação, leitura, atualização, exclusão).
- **Timestamp_Operacao**: Data e hora exatas em que a operação foi realizada.
- **Usuario**: Identificação do usuário que realizou a operação.
- **Dados_Antigos**: Representação dos dados antes da operação, para operações de UPDATE e DELETE.
- **Dados_Novos**: Representação dos dados após a operação, para operações de CREATE e UPDATE.

23.4 Implementação

A implementação dessa tabela de auditoria requer a criação de triggers para cada tabela do banco de dados que se deseja auditar. Esses triggers serão responsáveis por inserir registros na tabela

auditoria_log sempre que uma operação de CRUD for realizada. Este processo garante um sistema de auditoria abrangente e automatizado, essencial para a segurança e conformidade do sistema de banco de dados. Por exemplo, para capturar efetivamente as interações e alterações nas tabelas do banco de dados, serão implementadas triggers para cada operação de CRUD relevante. A seguir, exemplos de triggers para a tabela **produto**.

23.4.1 Trigger de Inserção

```
-- Criação da função do trigger
CREATE OR REPLACE FUNCTION produto_after_insert()
RETURNS TRIGGER AS $$
BEGIN
```

```
    INSERT INTO auditoria_log (Tabela_Afetada, Tipo_Operacao, Usuario, Dados_Novos
        )
    VALUES ('produto', 'CREATE', CURRENT_USER, CONCAT(NEW.ID, ', ', NEW.nome));
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Comentário sobre a função do trigger
COMMENT ON FUNCTION produto_after_insert() IS $$
Função trigger que captura eventos de inserção na tabela Produto. Registra
    detalhes da operação de criação na tabela de auditoria, incluindo o usuário
    que realizou a operação e os dados inseridos.
$$;

-- Criação do trigger
CREATE TRIGGER produto_log_insert
AFTER INSERT ON produto
FOR EACH ROW
EXECUTE FUNCTION produto_after_insert();
```

23.4.2 Trigger de Atualização

```
-- Criação da função do trigger para atualizações
CREATE OR REPLACE FUNCTION produto_after_update()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO meu_schema.auditoria_log (
        Tabela_Afetada,
        Tipo_Operacao,
        Usuario,
        Dados_Antigos,
        Dados_Novos
    ) VALUES (
        'produto',
        'UPDATE',
        CURRENT_USER,
        CONCAT(OLD.ID, ', ', OLD.nome),
        CONCAT(NEW.ID, ', ', NEW.nome)
    );
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Comentário sobre a função do trigger
COMMENT ON FUNCTION produto_after_update() IS $$
Esta função trigger é acionada após cada atualização na tabela Produto. Ela
    registra detalhes da operação de atualização na tabela de auditoria, incluindo
     o usuário que realizou a operação e uma comparação dos dados antigos e novos.
$$;

-- Criação do trigger para atualizações
CREATE TRIGGER Produto_After_Update
AFTER UPDATE ON produto
FOR EACH ROW
EXECUTE FUNCTION produto_after_update();
```

23.4.3 Trigger de Exclusão

```
-- Criação da função do trigger para exclusões
CREATE OR REPLACE FUNCTION produto_after_delete()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO auditoria_log (
        Tabela_Afetada,
        Tipo_Operacao,
        Usuario,
        Dados_Antigos
    ) VALUES (
        'produto',
        'DELETE',
        CURRENT_USER,
        CONCAT(OLD.ID, ', ', OLD.nome)
    );
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

-- Comentário sobre a função do trigger
COMMENT ON FUNCTION produto_after_delete() IS $$
Esta função trigger é acionada após cada exclusão na tabela Produto. Ela registra
    detalhes da operação de exclusão na tabela de auditoria, incluindo o usuário
    que realizou a operação e os dados que foram excluídos.
$$;

-- Criação do trigger para exclusões
CREATE TRIGGER Produto_After_Delete
AFTER DELETE ON produto
FOR EACH ROW
EXECUTE FUNCTION produto_after_delete();
```

Estas triggers garantem que todas as operações de inserção, atualização e exclusão realizadas na tabela **produto** sejam registradas na tabela **auditoria_log**, fornecendo um histórico detalhado de todas as alterações para propósitos de auditoria e conformidade. Lembre-se de adaptar as triggers conforme necessário para outras tabelas que você deseja auditar.

24 Requisitos não Funcionais

Requisitos não funcionais de um projeto ou sistema são critérios que especificam critérios gerais de qualidade, desempenho, segurança, e outras operações essenciais que o sistema deve atender, mas que não estão diretamente relacionados às funcionalidades específicas que o sistema irá executar. Eles servem para definir como o sistema faz o que deve fazer, em contraste com os requisitos funcionais que definem o que o sistema deve fazer.

- **Interface:**

  Uma interface gráfica amigável e intuitiva é fundamental para a adoção do sistema pelos usuários finais. Usar princípios de design UX/UI, como a consistência na interface, feedback visual claro, e minimização da carga cognitiva, pode ajudar a atingir esse objetivo. Ferramentas de design como Adobe XD ou Sketch, juntamente com frameworks front-end como React, Angular, ou Vue.js, podem ser empregadas para criar interfaces responsivas que se adaptam bem a diversos dispositivos.

- **Suporte de Desenvolvimento:**

A escolha de linguagens de programação modernas e frameworks atualizados, como Python com Django ou Flask para back-end e React ou Angular para front-end, facilita a manutenção e expansão do sistema. A documentação é vital para o suporte e desenvolvimento contínuo do sistema; ferramentas como Swagger para APIs RESTful e ferramentas de documentação de código como Doxygen podem ser utilizadas para manter a documentação atualizada.

- **Plataforma de Execução:**

  A compatibilidade com sistemas operacionais variados e a capacidade de hospedagem em ambientes de cloud computing são essenciais para a flexibilidade e escalabilidade do sistema. Contêineres Docker e orquestradores como Kubernetes podem ser usados para garantir a portabilidade e a fácil implantação em diferentes ambientes, incluindo provedores de cloud como AWS, Google Cloud Platform e Azure.

- **Robustez, Integridade, Segurança:**

  Para garantir a integridade dos dados e a segurança, o sistema deve incorporar backups regulares, recuperação de desastres e controle de acesso baseado em perfis de usuário. O uso de HTTPS para comunicação segura, juntamente com sistemas de gerenciamento de identidade e acesso como OAuth e JWT para autenticação e autorização, são práticas recomendadas. Além disso, a implementação de firewalls de aplicativos web (WAF) e testes de penetração regulares podem ajudar a fortalecer a segurança do sistema.

- **Performance:**

  A performance é crítica para a experiência do usuário e a eficiência operacional. Técnicas de otimização, como caching, indexação eficiente no banco de dados e uso de Content Delivery Networks (CDNs) para ativos estáticos, podem melhorar significativamente a velocidade de resposta do sistema. Ferramentas de monitoramento de performance, como New Relic ou Prometheus, podem ser usadas para identificar e resolver gargalos de performance.

  Cada um desses requisitos não funcionais traz suas próprias complexidades e desafios, e a sua implementação eficaz é crucial para o sucesso do sistema. Eles devem ser considerados desde o início do projeto, com escolhas tecnológicas e de design que alinhem a arquitetura do sistema com esses objetivos de qualidade.

25 Conclusão

Ao longo deste processo de modelagem de dados, tornou-se evidente a importância de um banco de dados bem estruturado para aprimorar as operações e facilitar decisões estratégicas em projetos complexos, como é o caso de uma rede de farmácias. Inicialmente, a etapa mais crítica envolveu a identificação clara dos requisitos do sistema, focando nas necessidades essenciais de um sistema farmacêutico e na compreensão de suas relações e dos dados necessários para armazenamento.

Utilizou-se um Diagrama Entidade Relacionamento (DER) para mapear as componentes do sistema e suas interconexões, fundamentais para estabelecer a estrutura inicial do banco de dados. Esse diagrama foi a base para o desenvolvimento de um modelo operacional, onde foram definidas as tabelas, seus atributos críticos e as conexões entre elas. Essas tabelas, implementadas em SQL, formam a espinha dorsal do modelo, especificando claramente que dados cada uma deve conter para uma organização eficiente das informações.

Exemplos de consultas ao banco foram elaborados para extrair dados valiosos sobre vendas, estoques de produtos e o desempenho dos funcionários, informações cruciais para a gestão de uma rede farmacêutica.

Em resumo, a eficiência de um banco de dados é vital para o sucesso de qualquer organização que busca otimizar suas operações e fundamentar decisões críticas em dados robustos. Este projeto de banco de dados, focado no setor farmacêutico, exemplifica a importância estratégica dessa ferramenta para o sucesso de negócios modernos.