

IGESTA



Modelagem de Dados - Banco PostgreSQL



INTEGRANTES



Fellipe Meira Brandão

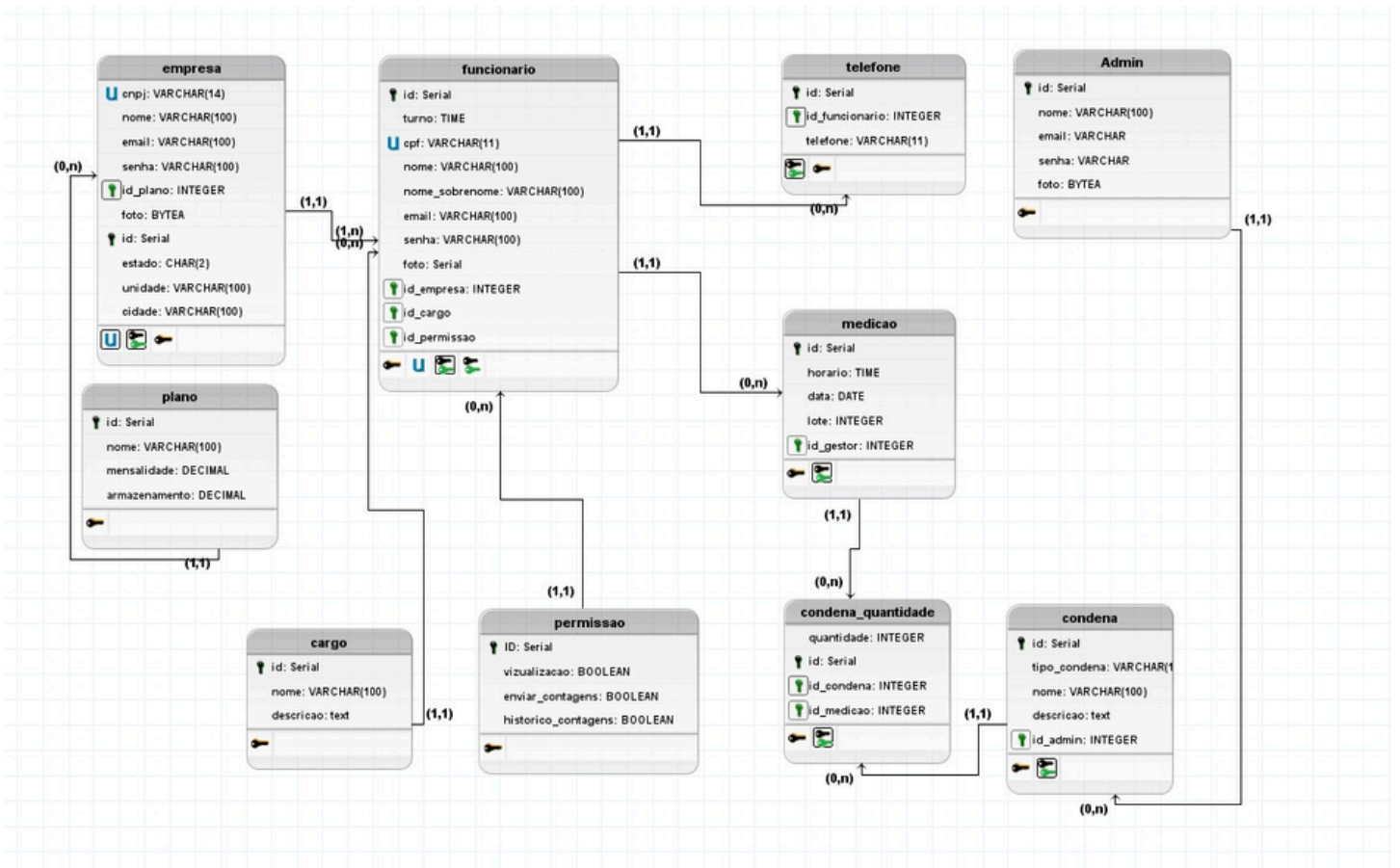


Gabriel Martins

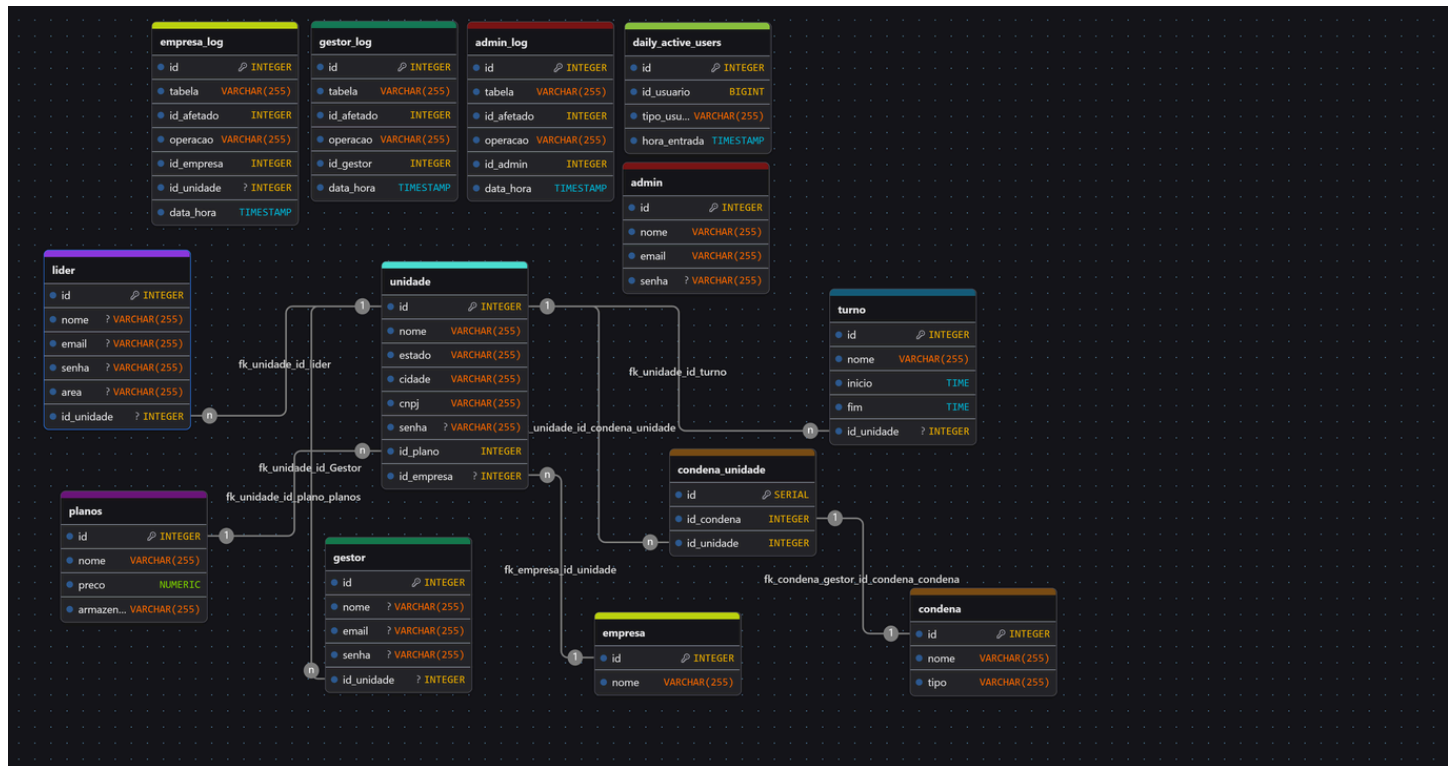


Rafael Cruz

Modelagem do Banco de Dados SQL - Sem normalização



Modelagem do Banco de Dados SQL - Normalizado



Tabelas do modelo normalizado e descrição

planos

- id - ID do plano de armazenamento
- nome - Nome do plano de armazenamento
- preco - Preço do plano
- armazenamento - Espaço de armazenamento do plano (Gigabytes)

lider

- id - ID do líder
- nome - Nome inteiro do líder
- email - Email associado ao líder
- senha - Senha da conta do líder
- cpf - CPF referente ao líder
- area - Área que o líder presta serviço (Halal | Pendura | Sangria | Escaldagem | Evisceração)
- id_unidade - ID da unidade que o líder trabalha

turno

- id - ID do turno
- nome - Nome para referenciar o horário do turno (Manhã | Tarde | Noite)
- inicio - Horário do início do turno (00:00:00)
- fim - Horário de término do turno (00:00:00)
- id_unidade - ID da unidade que irá obedecer tais turnos (Pode ser que os turnos de uma empresa seja diferente da outra)

unidade

- id - ID da unidade
- nome - Nome da unidade de tal empresa
- estado - Estado que essa unidade pertence
- cidade - Cidade que essa unidade pertence
- cnpj - CNPJ correspondente a unidade
- senha - Senha de acesso para a conta da Unidade (CNPJ e senha)
- id_plano - ID do plano que a unidade está usando
- id_empresa - ID da empresa que essa unidade faz parte

empresa

- id - ID da empresa
- nome - Nome da empresa

gestor

- id - ID do gestor
- nome - Nome do gestor
- email - email do gestor
- senha - senha da conta do gestor
- cpf - CPF do gestor
- id_unidade - ID da unidade que o gestor trabalha

condena

- id - ID da condena
- nome - Nome da condena
- tipo - Tipo da condena se é **parcial** ou **total**

condena_unidade

- id_condena - ID da condena que o gestor vai aderir
- id_unidade - ID da unidade que corresponde as condenas escolhidas para uso

admin

- id - ID do administrador do App
- nome - Nome do administrador do App
- email - email do administrador do App

Tabelas de Log e DAU

empresa_log

- id - ID do log
- tabela - Tabela que sofreu alteração da conta de empresa
- id_afetado - ID que foi modificado de alguma maneira
- operacao - Consta se foi um INSERT, UPDATE ou DELETE
- id_empresa - ID da empresa que fez a operação
- id_unidade - ID da unidade que fez a operação
- data_hora - Data e horário que foi executado a ação da empresa

gestor_log

- id - ID do log
- tabela - Tabela que sofreu alteração da conta de gestor
- id_afetado - ID que foi modificado de alguma maneira
- operacao - Consta se foi um INSERT, UPDATE ou DELETE
- id_gestor - ID do gestor que fez a operação
- data_hora - Data e horário que foi executado a ação do gestor

admin_log

- id - ID do log
- tabela - Tabela que sofreu alteração de uma conta do admin
- id_afetado - ID que foi modificado de alguma maneira
- operacao - Consta se foi um INSERT, UPDATE ou DELETE
- id_admin - ID do admin que fez a operação
- data_hora - Data e horário que foi executado a ação do admin

daily_active_users

- id - ID que representa a entrada de um usuário

- id_usuario - ID do usuário que entrou no app.
- tipo_usuario - Representa o tipo do usuário (Gestor | Líder | Unidade)
- hora_entrada - Data e horário que o usuário entrou no app.

Functions

tipo_usuario

Essa função identifica o tipo de usuário (**Gestor**, **Líder** ou **Unidade**) no momento do login no aplicativo, com base nas credenciais informadas: **e-mail** (para **gestores** e **líderes**) ou **CNPJ** (para **unidades**) e a senha.

verificar_empresa_existente

Verifica se uma empresa já existe no banco de dados com base nos oito primeiros dígitos do CNPJ da unidade, que representam a raiz do CNPJ utilizada para identificar empresas e suas respectivas unidades. Facilita o reconhecimento do relacionamento quando uma nova unidade é inserida.

log_admin

Função utilizada por uma trigger para registrar na tabela "**admin_log**" todos os dados inseridos ou alterados por administradores.

log_gestor

Função utilizada por uma trigger para registrar na tabela "**gestor_log**" todos os dados inseridos ou alterados por gestores.

log_empresa

Função utilizada por uma trigger para registrar na tabela "**empresa_log**" todos os dados inseridos ou alterados por empresas/unidades.

Procedures

inserir_unidade_por_nome

Insere uma nova unidade no banco de dados e verifica se a empresa correspondente já existe, utilizando a função **“verificar_empresa_existente”** antes de realizar a inserção.

atualizar_condenas_unidade

Atualiza a tabela **“condena_unidade”** quando uma unidade passa a utilizar um novo tipo de condena ou deixa de utilizá-la, garantindo que as informações de vínculo entre unidades e tipos de condena permaneçam atualizadas no banco de dados.

atualizar_plano_unidade

Atualiza o plano selecionado pela unidade, registrando a nova escolha no banco de dados e substituindo o plano anteriormente associado

Triggers

Admin

- **trg_log_admin** - chamada após Insert/Update/Delete feito por um admin na tabela **“admin”**.
- **trg_log_admin_planos** - chamada após Insert/Update/Delete feito por um admin na tabela **“planos”**.
- **trg_log_admin_condena** - chamada após Insert/Update/Delete feito por um admin na tabela **“condena”**.

Gestor

- **trg_log_gestor** - chamada após Insert/Update/Delete feito por um gestor na tabela **“gestor”**.
- **trg_log_gestor_lider** - chamada após Insert/Update/Delete feito por um gestor na tabela **“lider”**.
- **trg_log_gestor_condena** - chamada após Insert/Update/Delete feito por um admin na tabela **“condena_unidade”**.

Empresa/Unidade

- **trg_log_empresa**- chamada após Insert/Update/Delete feito por uma unidade na tabela "empresa".
- **trg_log_empresa_unidade** - chamada após Insert/Update/Delete feito por uma unidade na tabela "unidade".
- **trg_log_unidade_turno** - chamada após Insert/Update/Delete feito por uma unidade na tabela "turno".

Índices

MongoDB

db.registros.createIndex({unidade:1})

O index acima cria um índice simples no campo "unidade" da coleção "registros".

O motivo para criar esse índice é melhorar o desempenho das consultas que filtram ou ordenam os documentos com base no campo "unidade", já que ele é muito utilizado nas APIs de consulta.

db.registros.createIndex({unidade: 1, data:-1})

O índice composto acima cria um índice nos campos "unidade" (ordem crescente) e "data" (ordem decrescente) da coleção "registros".

O motivo para criar esse índice composto é otimizar consultas que filtram por "unidade" e ordenam os resultados por "data" em ordem decrescente. Esses campos são muito utilizados nas APIs que consultam os registros e nas consultas que o chatbot realiza para fornecer respostas rápidas e precisas aos usuários.