Projeto Pessoal: Sistema de Cadastro de Funcionários com MySQL e Modelagem Relacional!

Olá, rede! Tudo bem?

Espero que todos estejam ótimos!

Estou iniciando um projeto pessoal de modelagem de banco de dados relacional utilizando MySQL, com o objetivo de aprimorar minhas habilidades e aprofundar meus conhecimentos nessa área que tanto me encanta. Quando vi essa matéria na faculdade, foi amor à primeira vista e desde então, tenho grandes expectativas de um dia atuar profissionalmente com banco de dados!

Neste primeiro projeto, estou desenvolvendo um sistema de cadastro de funcionários para uma empresa fictícia, permitindo o gerenciamento de informações como cargos, departamentos, endereços, telefones e salários. Está sendo uma experiência incrível aplicar na prática os conceitos que estou aprendendo.

Irei compartilhar aqui na rede a evolução desse trabalho e ficarei muito feliz em receber sugestões ou dicas dos mais experientes!

Fase 1 do Projeto: Introdução ao Sistema de Cadastro de Funcionários.

Fase 2 do Projeto: Desenvolvimento do Modelo Conceitual e Diagrama Entidade-Relacionamento (DER).

Fase 3 do Projeto: Criação do Modelo Lógico.

Fase 4 do Projeto: Implementação do Modelo Físico.

Fase 5 do Projeto: Execução de Consultas e Testes no Banco de Dados.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Fase 1 do Projeto: Introdução ao Sistema de Cadastro de Funcionários /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Objetivo do Projeto: Criar um banco de dados relacional com MySQL para gerenciar:

Cadastro de Funcionários:

Colaborador: nome, CPF, data de nascimento, e-mail, telefone, sexo.

Endereço: rua, CEP, bairro, cidade, estado.

Departamentos: áreas de atuação dos funcionários.

Cargos: funções e responsabilidades.

Escalas de Trabalho: horários e turnos.

Telefone: número e tipo.

Admissão: data.

Demissão: data e motivo.

Salário: valor.

Estado Civil: tipo.

Benefícios: tipos.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Fase 2 do Projeto: Iniciando o Modelo Conceitual /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Lembrando que todo o projeto, incluindo o Diagrama Entidade-Relacionamento (DER), estará disponível no meu GitHub!

Entidades e Relacionamentos:

Colaborador

Atributos: idcolaborador, nome, cpf, data\_nascimento, email, sexo, id\_turno, id\_departamento, id\_cargo

Relacionamentos: vários colaboradores pertencem a um turno, a um departamento e ocupam um cargo.

Turno

Atributos: idturno, periodo, hora\_inicial, hora\_final

Relacionamento: um turno pode ter vários colaboradores.

Departamento

Atributos: iddepartamento, nome\_departamento

Relacionamento: um departamento pode ter vários colaboradores.

Cargo

Atributos: idcargo, nome\_cargo

Relacionamento: um cargo pode ser ocupado por vários colaboradores.

Endereço

Atributos: idendereco, rua, cep, bairro, cidade, estado, numero, id\_colaborador

Relacionamento: cada colaborador reside em um endereço.

Telefone

Atributos: idtelefone, numero, tipo, id\_colaborador

Relacionamento: um colaborador pode ter um ou mais telefones.

Salário

Atributos: idsalario, valor, id\_colaborador

Relacionamento: um colaborador tem um salário.

Estado Civil

Atributos: idestadocivil, tipo, id\_colaborador

Relacionamento: um colaborador tem um estado civil.

Benefício

Atributos: idbeneficio, vale\_transporte, vale\_refeicao, id\_colaborador

Relacionamento: um colaborador pode ter um benefício.

Demissão

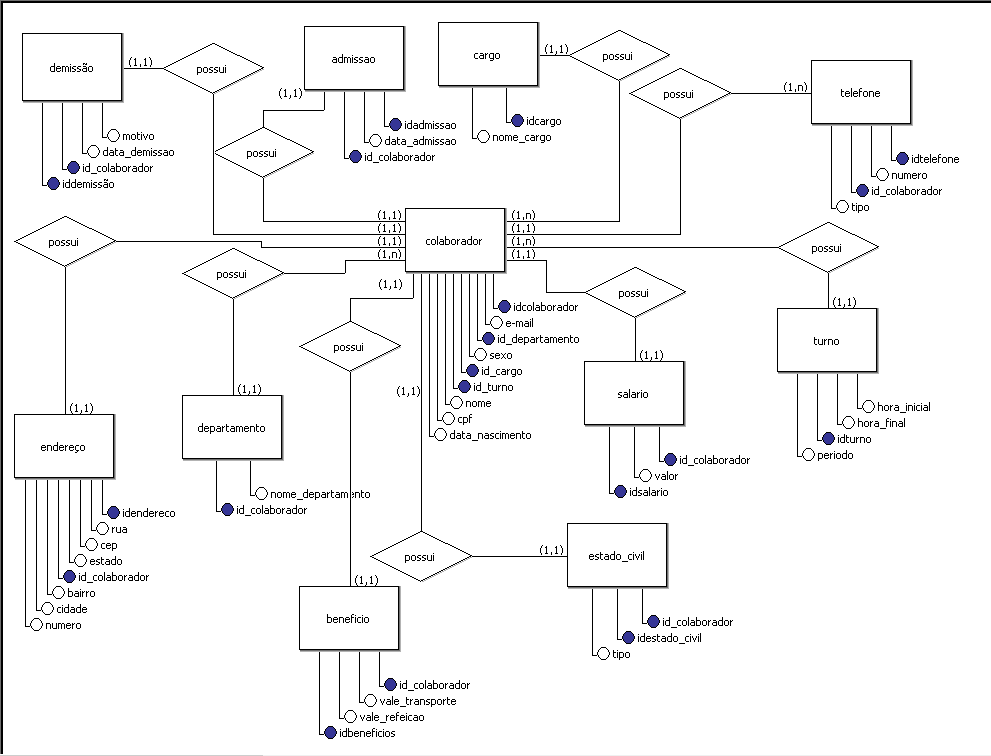
Atributos: iddemissao, data\_demissao, motivo, id\_colaborador

Relacionamento: um colaborador pode ter uma demissão registrada.

Admissão

Atributos: idadmissao, data\_admissao, id\_colaborador

Relacionamento: cada colaborador possui exatamente um registro de admissão no sistema.

Abaixo temos o Diagrama de Entidade e Relacionamentos(DER)

Nos próximos posts, vou compartilhar as etapas da modelagem lógica e física do banco, além das decisões que tomei no caminho.

Lembrando que todo o projeto, incluindo o Diagrama Entidade-Relacionamento (DER) estará disponível. Além disso, estou populando o banco com mais de 100 registros reais de exemplo — uma ótima forma de praticar consultas SQL na prática! Espero que gostem!!! Tudo estará disponível no meu GitHub! : [https://github.com/Eder-Alan](https://github.com/Eder-Alan%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20)  Quase esqueci! Se você tem dicas ou já passou por projetos semelhantes em modelagem de dados com MySQL, vou adorar trocar experiências!

#BancoDeDados #SQL #Modelagem #SistemaDeGestao #DesenvolvimentoDeSistemas #Aprendizado #MySQL #DatabaseDesign #ERDiagram #SQLPractice #DevCommunity

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Fase 3 do Projeto: Criação do Modelo Lógico.

Descrição dos Atributos:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Colaborador: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Colaborador

A entidade Colaboradora nada mais é que a entidade responsável pelas informações cadastrais do colaborador, como cpf, data de nascimento, email e sexo são informações relevantes do colaborador.

Atributos:

idcolaborador: (PK) ou como é mais conhecida Chave primária, que garante a unicidade de cada colaborador no banco de dados. Essencial para referenciar de maneira única qualquer colaborador.

Nome: ou seja, o nome completo do colaborador, lembrando que este campo não pode ser usado com chave estrangeira, isso porque por mais diferente que possa ser o nome das pessoas sempre haverá alguém com um nome igual.

cpf: Utilizado para garantir a unicidade do colaborador, uma vez que o CPF é único no Brasil. Serve como identificador legal, porém não é aconselhável ser utilizando como um chave primária por questões de boas práticas de modelagem.

data\_nascimento: Usada para determinar a idade do colaborador e relacionar com políticas internas da empresas como por exemplo: aposentadoria.

email: Importante para comunicação direta com o colaborador, como por exemplo: envio de notificações e alertas.

sexo: Pode ser usado como análises demográficas da empresa, isso inclui diversidade e inclusão.

Relacionamentos entre Entidades:

Colaborador - Departamento: Vários colaboradores podem ser alocados a um único departamento, refletindo a estrutura organizacional da empresa. Cada colaborador pertence a um único departamento.

Colaborador - Cargo: Vários colaboradores podem ocupar o mesmo cargo, mas cada colaborador tem exatamente um cargo atribuído, que define suas funções e responsabilidades na empresa.

Colaborador - Turno: Vários colaboradores podem estar em um único turno. Cada colaborador pertence a um único turno.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Turno: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Turno

A entidade Turno é responsável por definir as faixas dos horários em que os colaboradores são alocados para trabalhar. Ela organiza os períodos de trabalho, garantindo a eficiência da escala de colaboradores e a gestão de horários.

Atributos:

idturno (PK) ou Chave primária, é um identificador único para cada turno. Essencial para garantir a integridade dos dados e o relacionamento com outras entidades.

periodo: Identifica o tipo de turno (ex: "Manhã", "Tarde", "Noite", "administrativo"). Facilita a categorização dos turnos e a organização das jornadas de trabalho.

hora\_inicial: Hora de início do turno. Ajuda a controlar a entrada dos colaboradores e a organizar as escalas de trabalho.

hora\_final: Hora de término do turno. Essencial para controlar o término do expediente e o total de horas trabalhadas.

Relacionamento: Turno - Colaborador

Cardinalidade: 1:N (em um turno pode conter vários colaboradores)

Descrição: Um turno pode ter vários colaboradores, mas cada colaborador pertence a apenas um turno.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Departamento: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Departamento

A entidade departamento é responsável por definir as repartições de uma empresa, ou seja, os setores de uma organização.

Atributos:

iddepartamento (PK) ou Chave primária, é um identificador único para cada departamento. Essencial para garantir a integridade dos dados e o relacionamento com outras entidades.

nome\_departamento: Nome do departamento, identificando cada departamento da organização.

Relacionamentos:

Relacionamento: Departamento - Colaborador

Cardinalidade: 1:N (em um departamento pode conter vários colaboradores)

Isso indica um relacionamento de um para muitos (1:N), onde um departamento pode estar relacionado a vários colaboradores, mas cada colaborador está vinculado a um único departamento.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Cargo: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Cargo

A entidade Cargo representa os diferentes papéis ou funções dentro de uma organização. Ela é responsável por categorizar as responsabilidades e as atribuições dos colaboradores, facilitando assim a gestão de recursos humanos e o planejamento de equipe.

Atributos:

idcargo (PK): Identificador único para cada cargo, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

nome\_cargo: Nome do cargo ou função atribuída ao colaborador, como "Analista", "Gerente", "Coordenador", etc. Este atributo facilita a categorização dos cargos dentro da organização.

Relacionamento: Cargo - Colaborador

Cardinalidade: 1:N (Um cargo pode ser ocupado por vários colaboradores)

Descrição: Um cargo pode ser ocupado por vários colaboradores, mas cada colaborador ocupa um único cargo, estabelecendo um relacionamento de um para muitos entre Cargo e Colaborador.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Endereco: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Endereço

A entidade Endereço representa a localização residencial dos colaboradores, armazenando informações necessárias para o envio de correspondências. Ela é fundamental para o gerenciamento de dados de contato e localização dos colaboradores.

Atributos:

idendereco (PK): Identificador único para cada endereço, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

rua: Nome da rua onde o colaborador reside. Fundamental para especificar a localização exata.

cep: Código de Endereçamento Postal (CEP), utilizado para identificar a região de forma precisa.

bairro: Nome do bairro onde o colaborador reside.

cidade: Nome da cidade onde o colaborador reside.

estado: Nome do estado onde o colaborador reside.

numero: Número da residência, indicando o local exato dentro da rua.

id\_colaborador: Nome da chave estrangeira uma (FK) Identificador que estabelece uma relação entre as tabelas, colaborador e endereço.

Relacionamento: Endereço - Colaborador

Cardinalidade: 1:1 (Cada colaborador reside em um único endereço)

Descrição: Cada colaborador possui um único endereço, estabelecendo um relacionamento de um para um entre Endereço e Colaborador. Esse relacionamento ajuda a vincular informações de localização diretamente aos colaboradores.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Telefone: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Telefone

A entidade Telefone representa os números de telefone dos colaboradores, permitindo o contato direto para comunicação, sejam para emergências, avisos ou outros fins administrativos.

Atributos:

idtelefone (PK): Identificador único para cada número de telefone, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

numero: Número de telefone do colaborador. Este atributo armazena o número de contato, que pode ser fixo ou celular, dependendo da necessidade.

tipo: Tipo de telefone, como "Celular", "Residencial", "Comercial", etc. Esse atributo ajuda a categorizar os números de telefone de acordo com sua finalidade.

id\_colaborador: Nome da chave estrangeira uma (FK) este é um identificar de pelo menos um numero de telefone, mas caso um colaborador tenha mais de um telefone isso não representará um problema, visto que a cada telefone adicionado para um colaborador será representado por essa chave, além disto esta (FK) é um identificador que estabelece uma relação entre as tabelas, colaborador e telefone.

Relacionamento: Telefone - Colaborador

Cardinalidade: 1:N (Um colaborador pode possuir um ou mais telefones)

Descrição: Um colaborador pode ter um ou mais números de telefone associados, estabelecendo um relacionamento de um para muitos entre Telefone e Colaborador. Isso permite que um colaborador tenha múltiplos números de contato, mas cada telefone pertença a apenas um colaborador.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Salario: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Salário

A entidade Salário representa a remuneração que um colaborador recebe em troca de seu trabalho. Ela é responsável por armazenar os valores de pagamento e associá-los aos colaboradores de forma organizada e precisa.

Atributos:

idsalario (PK): Identificador único para cada registro de salário, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

valor: Valor do salário do colaborador, indicando a quantia que o colaborador recebe por seu trabalho.

id\_colaborador: Nome da chave estrangeira uma (FK) Identificador que estabelece uma relação entre as tabelas, colaborador e salário.

Relacionamento: Salário - Colaborador

Cardinalidade: 1:1 (Um colaborador tem penas um salário)

Descrição: Cada colaborador tem um único salário associado, estabelecendo um relacionamento de um para um entre Salário e Colaborador. Isso permite associar o valor da remuneração de cada colaborador de forma clara e individual.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Estadocivil: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Estado Civil

A entidade Estado Civil representa a situação civil de um colaborador, indicando se ele é solteiro, casado, divorciado ou viúvo. Essa informação pode ser relevante para questões administrativas e benefícios oferecidos pela organização.

Atributos:

idestadocivil (PK): Identificador único para cada estado civil, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

tipo: Tipo de estado civil, como "Solteiro", "Casado", "Divorciado", "Viúvo", etc. Esse atributo categoriza o estado civil do colaborador.

id\_colaborador: Nome da chave estrangeira uma (FK) Identificador que estabelece uma relação entre as tabelas, colaborador e estado civil.

Relacionamento: Estado Civil - Colaborador

Cardinalidade: 1:1 (Um colaborador ter apenas um estado civil)

Descrição: Cada colaborador possui um único estado civil, estabelecendo um relacionamento de um para um entre Estado Civil e Colaborador. Isso permite associar a situação civil de cada colaborador de forma clara.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Beneficio: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Benefício

A entidade Benefício representa os benefícios fornecidos aos colaboradores, como vale-transporte e vale-refeição. Ela armazena as informações sobre esses auxílios oferecidos pela organização, que são essenciais para o bem-estar e a satisfação dos colaboradores.

Atributos:

idbeneficio (PK): Identificador único para cada benefício, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

vale\_transporte: Indica o valor ou a informação sobre o benefício de vale-transporte concedido ao colaborador.

vale\_refeicao: Indica o valor ou a informação sobre o benefício de vale-refeição concedido ao colaborador.

id\_colaborador: Nome da chave estrangeira uma (FK) Identificador que estabelece uma relação entre as tabelas, colaborador e benefício.

Relacionamento: Benefício - Colaborador

Cardinalidade: 1:1 (Um colaborador ter apenas um conjunto de benefício)

Descrição: Cada colaborador pode ter um único conjunto de benefícios, estabelecendo um relacionamento de um para um entre Benefício e Colaborador. Isso permite associar os benefícios específicos oferecidos a cada colaborador.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Demissao: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Demissão

A entidade Demissão representa o registro de quando um colaborador é desligado da organização, seja por iniciativa própria ou da empresa. Ela armazena informações importantes sobre a data e o motivo da demissão, auxiliando na gestão de recursos humanos.

Atributos:

iddemissao (PK): Identificador único para cada registro de demissão, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

data\_demissao: Data em que o colaborador foi desligado da organização. Esse atributo é importante para o controle administrativo e para fins de registro.

motivo: Motivo da demissão, podendo ser "Pedido de demissão", "Justa causa", "Rescisão contratual", etc. Esse atributo ajuda a categorizar o tipo de demissão.

id\_colaborador: Nome da chave estrangeira uma (FK) Identificador que estabelece uma relação entre as tabelas, colaborador e demissao.

Relacionamento: Demissão - Colaborador

Cardinalidade: 1:1 (Um colaborador possui apenas uma demissão)

Descrição: Cada colaborador tem um único registro de demissão, estabelecendo um relacionamento de um para um entre Demissão e Colaborador. Esse relacionamento facilita o acompanhamento do status do colaborador na organização.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/Admissao: /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Entidade: Admissão

A entidade Admissão representa o registro de quando um colaborador ingressa na organização. Ela armazena a data de início do vínculo empregatício, fundamental para o controle de tempo de serviço e para a administração de benefícios e direitos trabalhistas.

Atributos:

idadmissao (PK): Identificador único para cada registro de admissão, essencial para garantir a integridade dos dados e o relacionamento com outras entidades. Este atributo é a chave primária (PK).

data\_admissao: Data em que o colaborador foi admitido pela organização. Esse atributo é importante para o controle administrativo e para fins de registro.

id\_colaborador: Nome da chave estrangeira uma (FK) Identificador que estabelece uma relação entre as tabelas, colaborador e admissão.

Relacionamento: Admissão - Colaborador

Cardinalidade: 1:1 (Um colaborador possui apenas uma admissão)

Descrição: Cada colaborador tem um único registro de admissão, estabelecendo um relacionamento de um para um entre Admissão e Colaborador. Esse relacionamento facilita o acompanhamento do início do vínculo empregatício e a gestão de dados do colaborador.

Nos próximos posts, vou compartilhar as etapas da modelagem física do banco, além das decisões que tomei no caminho.

Ah, e claro: quem quiser conferir tudo na prática, é só acessar o meu GitHub! Além disso, estou populando o banco com mais de 100 registros reais de exemplo — uma ótima forma de praticar consultas SQL na prática!!! Espero que gostem!!! Meu GitHub! : [https://github.com/Eder-Alan](https://github.com/Eder-Alan%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20)  Quase esqueci! Se você tem dicas ou já passou por projetos semelhantes em modelagem de dados com MySQL, vou adorar trocar experiências!

#BancoDeDados #SQL #Modelagem #SistemaDeGestao #DesenvolvimentoDeSistemas #Aprendizado #MySQL #DatabaseDesign #ERDiagram #SQLPractice #DevCommunity

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Fase 4 do Projeto: Implementação do Modelo Físico.

O modelo físico é fundamental para a construção de um banco de dados eficiente e de fácil manutenção. Aqui estão os principais pontos abordados:

Definição de Tipos de Dados: Para cada atributo das entidades, agora escolhemos os tipos de dados apropriados (por exemplo, VARCHAR, DECIMAL, DATE, CHAR, INT E ETC.).

Chaves Primárias e Estrangeiras: A modelagem inclui as chaves primárias para garantir a unicidade de cada registro (tuplas) e as chaves estrangeiras para estabelecer os relacionamentos entre as tabelas.

Respeitando as Formas Normais: Durante a criação do modelo físico, as formas normais foram respeitadas para garantir a eliminação de redundâncias e dependências indesejadas entre os dados.

Criação do banco e das tabelas

Chegamos à 4ª parte do projeto: O Modelo Físico!

Abaixo teremos a criação do banco de dados e as tabelas.

create database if not exists bd\_empresa

DEFAULT CHARACTER SET utf8mb4

DEFAULT COLLATE utf8mb4\_unicode\_ci;

-- Criando o banco de dados bd\_empresa com charset utf8mb4 para suportar todos os caracteres Unicode e collation utf8mb4\_unicode\_ci para ordenação e comparação de texto correta.

use bd\_empresa; -- acessando o banco de dados criado.

-- criação das tabelas.

-- criação da tabela colaborador.

create table if not exists colaborador(

idcolaborador int primary key auto\_increment,

nome varchar(45) not null,

cpf varchar(15) not null, /\*não deixei o cpf como único para que possa haver possibilidade de uma recontratação de colaborador.\*/

data\_nascimento date not null,

email varchar(35),

sexo enum('M', 'F') not null,

id\_turno int, -- Declarando a coluna de chave estrangeira

id\_departamento int, -- Declarando a coluna de chave estrangeira

id\_cargo int -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

-- criação da tabela endereco.

create table if not exists endereco(

idendereco int primary key auto\_increment,

rua varchar(60) not null,

cep varchar(8) not null,

bairro varchar(30) not null,

cidade varchar(30) not null,

estado char(2) not null,

numero int unsigned,

id\_colaborador int unique -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

-- criação da tabela telefone.

create table if not exists telefone(

idtelefone int primary key auto\_increment,

numero varchar(15) not null,

tipo enum('comerc', 'resid', 'celular') not null,

id\_colaborador int -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

-- criação da tabela turno.

create table if not exists turno(

idturno int primary key auto\_increment,

periodo enum('turno A', 'turno B', 'turno C', 'turno Admin') not null,

hora\_inicial enum('06:00', '14:00', '22:00') not null,

hora\_final enum('14:00', '22:00', '06:00', '17:00') not null

)default charset = utf8mb4;

-- criação da tabela salario.

create table if not exists salario(

idsalario int primary key auto\_increment,

valor decimal(10,2) not null,

id\_colaborador int -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

-- criação da tabela estadocivil.

create table if not exists estadocivil(

idestado\_civil int primary key auto\_increment,

tipo enum('cadado(a)', 'solteiro(a)', 'divorciado(a)',

'viuvo(a)', 'separado(a)', 'companheiro(a)') not null,

id\_colaborador int unique

)default charset = utf8mb4;

-- criação da tabela beneficio.

create table if not exists beneficio(

idbeneficio int primary key auto\_increment,

vale\_transporte enum('sim', 'nao') not null,

vale\_refeição enum('sim', 'nao') not null,

id\_colaborador int unique

)default charset = utf8mb4;

-- criação da tabela departamento.

create table if not exists departamento(

iddepartamento int primary key auto\_increment,

nome\_departamento enum('Desenvolvimento Software',

'Desenvolvimento BD', 'Engenharia',

'Engenharia dados','Gerencia','RH') not null

)default charset = utf8mb4;

-- criação da tabela demissao.

create table if not exists demissao(

iddemissao int primary key auto\_increment,

data\_demissao date not null,

motivo enum('Demissao sem justa causa',

'Demissao por justa causa',

'Pedido de demissao',

'Demissão consensual',

'Demissao indireta') not null,

id\_colaborador int unique -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

-- criação da tabela admissao.

create table if not exists admissao(

idadmissao int primary key auto\_increment,

data\_admissao date not null,

id\_colaborador int unique -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

-- criação da tabela cargo.

create table if not exists cargo(

idcargo int primary key auto\_increment,

nome\_cargo enum('Assitente TI', 'Programador jr', 'Programador pleno',

'Programador senior', 'Analista banco dados',

'Gerente', 'Coordenador', 'Assistente RH', 'Gerente RH',

'Engenheiro banco dados') not null

)default charset = utf8mb4;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Uma breve explicação das decisões tomadas\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Boas Práticas

create database if not exists bd\_empresa

DEFAULT CHARACTER SET utf8mb4

DEFAULT COLLATE utf8mb4\_unicode\_ci;

Usando o IF NOT EXISTS ao criar bancos de dados:

Isso evita erros caso o banco de dados já exista. O uso de IF NOT EXISTS é uma boa prática para evitar falhas de execução caso estiver rodando o script várias vezes ou em diferentes ambientes (produção, desenvolvimento, etc.).

Organizar o nome do banco de dados:

É uma boa prática usar um nome de banco de dados claro e relacionado ao projeto, como bd\_empresa neste caso, para facilitar a identificação.

Usando utf8mb4 como charset:

Fiz o uso do utf8mb4 sendo uma boa prática porque ele suporta todos os caracteres Unicode, incluindo emojis e outros símbolos que não são cobertos pelo utf8 tradicional.

Usando utf8mb4\_unicode\_ci como collation:

utf8mb4\_unicode\_ci é uma colação recomendada, pois é baseada nas regras Unicode para comparação de texto. Ela é case-insensitive (não diferencia maiúsculas de minúsculas) e trata corretamente caracteres acentuados e especiais, proporcionando maior precisão na comparação textual em sistemas que precisam lidar com múltiplos idiomas.

Existem outras colações possíveis, como utf8mb4\_general\_ci, mas a versão unicode\_ci tende a ser mais precisa para comparação de texto em vários idiomas.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- a criação das tabelas.

-- tabela colaborador.

Boas Práticas no Código

create table if not exists colaborador(

idcolaborador int primary key auto\_increment,

nome varchar(45) not null,

cpf varchar(15) not null, /\*não deixei o cpf como único para que possa haver possibilidade de uma recontratação de colaborador.\*/

data\_nascimento date not null,

email varchar(35),

sexo enum('M', 'F') not null,

id\_turno int, -- Declarando a coluna de chave estrangeira

id\_departamento int, -- Declarando a coluna de chave estrangeira

id\_cargo int -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois evita que a tabela seja recriada caso já exista. Isso ajuda a prevenir erros se o script for executado múltiplas vezes ou em diferentes ambientes (desenvolvimento, produção, etc.).

Definir a chave primária:

Chave primária (idcolaborador): Definir uma chave primária (PRIMARY KEY) é uma boa prática, pois ela garante a unicidade dos registros e melhora a eficiência das operações de busca e referência. O uso de auto\_increment é adequado, pois facilita a inserção de novos registros sem a necessidade de fornecer um identificador manualmente.

Escolher tipos de dados adequados:

nome: O tamanho do varchar(45) para nome é uma quantidade razoável para este campo.

email: O tamanho do varchar(35) é razoável para esses campos.

cpf: O tamanho do varchar(15) para esse é o tamanho adequado para armazenar o CPF, incluindo a máscara (por exemplo, XXX.XXX.XXX-XX).

data\_nascimento: date para o tipo de dado date é adequado para armazenar datas de nascimento de colaboradores.

sexo: o uso de enum para sexo é uma boa prática, pois restringe os valores possíveis a 'M' e 'F', o que garante dados consistentes.

Adicionar not null em campos obrigatórios:

nome, data\_nascimento, sexo: Esses campos são essenciais e por este motivo foram declarados com NOT NULL, o que impede que registros incompletos sejam inseridos na tabela.

email: O email não sofreu a restrição de NOT NULL isso porque não é algo obrigatório.

Chaves estrangeiras: as colunas id\_turno, id\_departamento, e id\_cargo são chaves estrangeiras, conhecidas como foreign keys, e as mesmas serão criadas fora das tabelas, o que é uma abordagem que tem sido discutida como boas práticas de design de banco de dados.

Usar o charset adequado:

O uso de DEFAULT CHARSET = utf8mb4 também é conhecido como boa prática, pois permite a armazenagem de todos os caracteres Unicode (incluindo emojis e símbolos de várias línguas). Isso ajuda a garantir que o banco de dados possa armazenar dados de forma completa e sem problemas relacionados a diferentes idiomas ou caracteres especiais.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- tabela endereco.

Boas Práticas no Código

create table if not exists endereco(

idendereco int primary key auto\_increment,

rua varchar(60) not null,

cep varchar(8) not null,

bairro varchar(30) not null,

cidade varchar(30) not null,

estado char(2) not null,

numero int unsigned,

id\_colaborador int unique -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois evita a recriação da tabela caso ela já exista, prevenindo erros durante execuções repetidas do script.

Definição de chave primária:

A chave primária (PRIMARY KEY) é corretamente definida em idendereco, garantindo que os registros sejam identificados de forma única e melhorando a performance das operações de busca e manipulação de dados.

É importante observar o uso do not null, ou seja, são campos obrigatórios.

Uso de varchar com tamanhos adequados:

rua: varchar(60) o tamanho de 60 caracteres para a rua é adequado para o nosso caso.

cep: varchar(10) o CEP geralmente tem 8 caracteres (formato XXXXX-XXX), e o tamanho de 10 caracteres está ideal para esse campo.

bairro: varchar(30) está adequado para o nosso caso.

cidade: varchar(30) está adequado para o nosso caso.

estado: char(2) o uso de char(2) para armazenar a sigla do estado (como "SP" ou "RJ") é adequado, pois o Brasil possui apenas 26 estados e o Distrito Federal, então o campo é sempre fixo com 2 caracteres.

número: int unsigned o uso de int unsigned para o número da residência também é considerada uma boa prática, pois o número de uma residência nunca pode ser negativo. Isso ajuda a garantir dados válidos e economiza espaço.

Chave estrangeira (id\_colaborador):

A coluna id\_colaborador foi definida como única (unique), mas não foi explicitamente declarada como chave estrangeira, isso será tratado mais adiante, além disto, temos um unique o que significa que um colaborador só pode ter um endereço de residência.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 mais uma vez uma boa prática para garantir o suporte a todos os caracteres Unicode, incluindo caracteres especiais, acentos, emojis, etc.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela telefone.

create table if not exists telefone(

idtelefone int primary key auto\_increment,

numero varchar(15) not null,

tipo enum('comerc.', 'resid.', 'celular') not null,

id\_colaborador int -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois evita erros caso a tabela já exista no banco de dados. Isso é útil durante execuções repetidas do script ou quando o ambiente de desenvolvimento.

Definir chave primária:

A definição de idtelefone como a chave primária (PRIMARY KEY) é essencial para garantir a unicidade de cada telefone e melhorar a performance de buscas e outras operações.

telefone: Uso de VARCHAR para o número de telefone.

numero: VARCHAR(15): com 15 caracteres são suficientes para armazenar números de telefone no Brasil, incluindo o DDD e possíveis símbolos como parênteses e traços por exemplo, (11) 98765-4321 outra observação muito importante o uso do not null ou seja são campos obrigatórios.

telefone: o uso de ENUM para o tipo de telefone:

tipo ENUM('comerc.', 'resid.', 'celular'): O uso de ENUM é adequado para tipos fixos de telefone.

Chave estrangeira para id\_colaborador:

A coluna id\_colaborador está indicada como uma chave estrangeira, mas não foi declarada explicitamente como tal no código, isso será tratado mais adiante.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 mais uma vez uma boa prática para garantir o suporte a todos os caracteres Unicode, incluindo caracteres especiais, acentos, emojis e caracteres especiais.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela turno.

create table if not exists turno(

idturno int primary key auto\_increment,

periodo enum('turno A', 'turno B', 'turno C', 'turno Admin') not null,

hora\_inicial enum('06:00', '14:00', '22:00') not null,

hora\_final enum('14:00', '22:00', '06:00', '17:00') not null

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois evita erros de recriação da tabela, permitindo que o script seja executado várias vezes sem problemas.

Definir chave primária:

A definição de idturno como a chave primária (PRIMARY KEY) é adequada, pois garante a unicidade dos registros e melhora o desempenho nas consultas que buscam dados específicos.

periodo: o uso de ENUM para periodo

ENUM para periodo: Usar ENUM para os diferentes períodos (turno A, B, C, Admin) isto é uma boa escolha, pois os valores são limitados e conhecidos, tornando os dados consistentes.

hora\_inicial e hora\_final:

A utilização de ENUM para horários fixos também é adequada no caso em que os horários são bem definidos e limitados, como neste caso.

Aqui vai apenas uma observação:

Para uma flexibilidade e expansão: Se no futuro precisássemos de mais horários ou se os horários precisarem de ajustes mais dinâmicos, talvez um campo TIME ou DATETIME fosse mais adequado. O ENUM pode ser rígido e dificultar alterações no futuro, caso os turnos ou horários se tornem mais variados, mas para o projeto em questão está sobe medida.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 mais uma vez uma boa prática para garantir o suporte a todos os caracteres Unicode, incluindo caracteres especiais, acentos, emojis e caracteres especiais.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela salario.

create table if not exists salario(

idsalario int primary key auto\_increment,

valor decimal(10,2) not null,

id\_colaborador int -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática para evitar erros durante a execução repetida do script, garantindo que a tabela só será criada se ela não existir.

Definição de chave primária:

A coluna idsalario foi definida como chave primária (PRIMARY KEY). Isso assegura que cada registro de salário tenha uma identificação única, melhorando a performance das operações de busca e manipulação de dados.

salário: o uso de DECIMAL para o valor do salário:

valor DECIMAL(10,2) foi uma escolha adequada para representar valores monetários, pois o tipo DECIMAL é exato, o que significa que não há problemas com arredondamento, que podem ocorrer com o tipo FLOAT ou DOUBLE.

Um exemplo para ficar mais claro o entendimento:

10: Define até 10 dígitos no total, o que é mais do que suficiente para valores de salários (por exemplo, até R$ 99.999.999,99).

2: Define 2 casas decimais para representar centavos, que é o formato esperado para valores monetários.

Chave estrangeira (id\_colaborador):

A coluna id\_colaborador é uma chave estrangeira que se refere à tabela colaborador, mas ela não está explicitamente definida como chave estrangeira no código, isso será tratado mais adiante.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 mais uma vez uma boa prática para garantir o suporte a todos os caracteres Unicode, incluindo caracteres especiais, acentos, emojis e caracteres especiais.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela estadocivil.

create table if not exists estadocivil(

idestado\_civil int primary key auto\_increment,

tipo enum('cadado(a)', 'solteiro(a)', 'divorciado(a)',

'viuvo(a)', 'separado(a)', 'companheiro(a)') not null,

id\_colaborador int unique

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática para evitar erros de recriação da tabela, permitindo que o script seja executado várias vezes sem problemas.

Definir chave primária:

A coluna idestado\_civil é definida como chave primária (PRIMARY KEY), o que é adequado para garantir que cada estado civil tenha uma identificação única.

tipo: o uso de ENUM para o tipo de estado civil:

ENUM para o campo tipo é uma boa prática, pois os valores possíveis são limitados e conhecidos. Isso garante que os valores sejam consistentes e evita erros de digitação ao inserir os dados.

Uso de UNIQUE em id\_colaborador:

A coluna id\_colaborador está definida como UNIQUE, o que significa que cada colaborador pode ter apenas um estado civil registrado na tabela. Isso foi feito para esta regra de negócios, ou seja, um colaborador só pode ter um estado civil por vez, além disso, é uma chave estrangeira que se refere à tabela colaborador, mas ela não está explicitamente definida como chave estrangeira no código, isso será tratado mais adiante.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 é uma boa prática para garantir que o banco de dados suporte todos os caracteres Unicode, incluindo caracteres especiais, emojis e outros símbolos.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela beneficio.

create table if not exists beneficio(

idbeneficio int primary key auto\_increment,

vale\_transporte enum('sim', 'nao') not null,

vale\_refeição enum('sim', 'nao') not null,

id\_colaborador int unique

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois impede erros de recriação da tabela e garante que o script possa ser executado várias vezes sem problemas.

Definir chave primária:

A coluna idbeneficio é corretamente definida como chave primária (PRIMARY KEY), garantindo a unicidade dos registros e permitindo um acesso eficiente.

Uso de ENUM para os benefícios:

ENUM para vale\_transporte e vale\_refeicao esta escolha foi feita devido esta regra de negócio, pois esses campos têm apenas dois valores possíveis: 'sim' ou 'não'. Usar ENUM garante que os valores sejam consistentes e evita erros de digitação.

Chave única (UNIQUE) para id\_colaborador:

A coluna id\_colaborador é definida como UNIQUE, o que significa que um colaborador pode ter nesta regra de negócios apenas um conjunto de benefícios (vale-transporte e vale-refeição) por vez, além disso, é uma chave estrangeira que se refere à tabela colaborador, mas ela não está explicitamente definida como chave estrangeira no código, isso será tratado mais adiante.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 é uma boa prática para garantir que o banco de dados suporte todos os caracteres Unicode, incluindo caracteres especiais, emojis e outros símbolos.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela departamento.

create table if not exists departamento(

iddepartamento int primary key auto\_increment,

nome\_departamento enum('Desenvolvimento Software',

'Desenvolvimento BD', 'Engenharia',

'Gerencia','RH') not null

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática para evitar erros ao recriar a tabela, permitindo que o script seja executado várias vezes sem problemas.

Definir chave primária:

A coluna iddepartamento é corretamente definida como chave primária (PRIMARY KEY), garantindo a unicidade e permitindo um acesso eficiente.

Uso de ENUM para os departamentos:

O tipo ENUM para o campo nome\_departamento é uma boa prática, isso porque os departamentos são valores fixos e limitados. O uso de ENUM garante que os valores serão consistentes e evita erros de digitação.

O uso de DEFAULT CHARSET = utf8mb4 é uma boa prática para garantir que o banco de dados suporte todos os caracteres Unicode, incluindo caracteres especiais, emojis e outros símbolos.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela demissao.

create table if not exists demissao(

iddemissao int primary key auto\_increment,

data\_demissao date not null,

motivo enum('Demissao sem justa causa',

'Demissao por justa causa',

'Pedido de demissao',

'Demissão consensual',

'Demissao indireta') not null,

id\_colaborador int unique -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois garante que o código não tente recriar a tabela se ela já existir, evitando erros de execução repetida do script.

Definir chave primária:

A coluna iddemissao está definida como chave primária (PRIMARY KEY), o que é uma boa prática para garantir a unicidade e facilitar as operações de busca e atualização.

data\_demissao: temos o tipo date é adequado para armazenar datas de demissão dos colaboradores.

Uso de ENUM para o motivo da demissão:

O tipo ENUM para o campo motivo é uma boa prática, pois os valores possíveis são limitados e conhecidos, garantindo que apenas motivos válidos sejam inseridos.

Uso de UNIQUE em id\_colaborador:

Chave estrangeira para id\_colaborador:

A coluna id\_colaborador está definida como UNIQUE, o que significa que um colaborador pode ter apenas um registro de demissão. Isso pode é uma boa prática para esta regra de negócios que é: um colaborador só pode ser demitido uma vez, além disso, é uma chave estrangeira que se refere à tabela colaborador, mas ela não está explicitamente definida como chave estrangeira no código, isso será tratado mais adiante.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 é uma boa prática para garantir que o banco de dados suporte todos os caracteres Unicode, incluindo caracteres especiais, emojis e outros símbolos.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela admissao.

create table if not exists admissao(

idadmissao int primary key auto\_increment,

data\_admissao date not null,

id\_colaborador int unique -- Declarando a coluna de chave estrangeira

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois garante que o código não tente recriar a tabela se ela já existir, evitando erros de execução repetida do script.

Definir chave primária:

A coluna idadmissao está corretamente definida como chave primária (PRIMARY KEY), garantindo a unicidade e facilitando a recuperação de dados relacionados a admissões específicas.

data\_admissao: temos o tipo date é adequado para armazenar datas de admissão dos colaboradores.

Uso de UNIQUE em id\_colaborador:

Chave estrangeira para id\_colaborador:

A coluna id\_colaborador está definida como UNIQUE, o que significa que um colaborador pode ter apenas um registro de admissao. Isso pode é uma boa prática para esta regra de negócios que é: um colaborador só pode ser admitido uma vez, além disso, é uma chave estrangeira que se refere à tabela colaborador, mas ela não está explicitamente definida como chave estrangeira no código, isso será tratado mais adiante.

Charset utf8mb4:

O uso de DEFAULT CHARSET = utf8mb4 é uma boa prática, pois isso permite que a tabela armazene todos os caracteres Unicode, incluindo caracteres especiais, emojis e outros símbolos.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- criação da tabela cargo.

create table if not exists cargo(

idcargo int primary key auto\_increment,

nome\_cargo enum('Assitente TI', 'Programador jr', 'Programador pleno',

'Programador senior', 'Analista banco dados',

'Gerente', 'Coordenador', 'Assistente RH', 'Gerente RH',

'Engenheiro banco dados') not null

)default charset = utf8mb4;

Boas Práticas no Código

Uso de IF NOT EXISTS:

O uso de IF NOT EXISTS é uma boa prática, pois garante que o código não tente recriar a tabela se ela já existir, evitando erros de execução repetida do script.

Definir chave primária:

A coluna idcargo está corretamente definida como chave primária (PRIMARY KEY), garantindo a unicidade de cada cargo e facilitando operações de busca, inserção e manipulação de dados.

Uso de ENUM para nome\_cargo:

O uso de ENUM para o campo nome\_cargo está adequado isso porque o número de cargos é fixo e limitado. Isso garante que os cargos sejam consistentes e evita erros de digitação ou valores inválidos.

Apenas uma Observação: Caso precisássemos adicionar novos cargos no futuro ou modificar os cargos existentes, o uso de ENUM poderia ser limitante, pois para adicionar ou remover um cargo nós precisaríamos alterar a definição da tabela, o que pode ser mais difícil de gerenciar em um sistema em produção.

O que poderia ser feito: criar uma tabela separada para armazenar os cargos, permitindo adicionar ou remover cargos sem alterar a estrutura da tabela cargo. Isso pode ser mais flexível, especialmente se os cargos mudarem com frequência, porém para esta regra de negócio o modelo atual está atendendo bem nossas necessidades.

Usar utf8mb4 como charset para garantir compatibilidade com caracteres especiais e Unicode.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- Definindo todas as chaves estrangeiras para garantir a integridade referencial

-- foreign keys da tabela colaborador, faz referencia a tabela turno

alter table colaborador

add constraint fk\_colaborador\_turno

foreign key (id\_turno) references turno(idturno);

-- foreign keys da tabela colaborador, faz referencia a tabela departamento

alter table colaborador

add constraint fk\_colaborador\_departamento

foreign key (id\_departamento) references departamento(iddepartamento);

-- foreign keys da tabela colaborador, faz referencia a tabela cargo

alter table colaborador

add constraint fk\_colaborador\_cargo

foreign key (id\_cargo) references cargo(idcargo);

-- foreign keys da tabela endereco, faz referencia a tabela colaborador

alter table endereco

add constraint fk\_endereco\_colaborador

foreign key (id\_colaborador) references colaborador(idcolaborador);

-- foreign keys da tabela telefone, faz referencia a tabela colaborador

alter table telefone

add constraint fk\_telefone\_colaborador

foreign key (id\_colaborador) references colaborador(idcolaborador);

-- foreign keys da tabela demissao, faz referencia a tabela colaborador

alter table demissao

add constraint fk\_demissao\_colaborador

foreign key (id\_colaborador) references colaborador(idcolaborador);

-- foreign keys da tabela admissao, faz referencia a tabela colaborador

alter table admissao

add constraint fk\_admissao\_colaborador

foreign key (id\_colaborador) references colaborador(idcolaborador);

-- foreign keys da tabela salario, faz referencia a tabela colaborador

alter table salario

add constraint fk\_salario\_colaborador

foreign key (id\_colaborador) references colaborador(idcolaborador);

-- foreign keys da tabela estado\_civil, faz referencia a tabela colaborador

alter table estadocivil

add constraint fk\_estadocivil\_colaborador

foreign key (id\_colaborador) references colaborador(idcolaborador);

-- foreign keys da tabela beneficio, faz referencia a tabela colaborador

alter table beneficio

add constraint fk\_beneficio\_colaborador

foreign key (id\_colaborador) references colaborador(idcolaborador);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Uma breve explicação das decisões tomadas\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- Para que serve esta tal de foreign key?

Uma foreign key (ou chave estrangeira) é um conceito fundamental em bancos de dados relacionais.

Ela é um campo (ou conjunto de campos) em uma tabela que faz referência à primary key (chave primária) de outra tabela. Em outras palavras, a foreign key serve para criar um relacionamento entre tabelas, garantindo que os dados de uma tabela estejam sempre relacionados de forma consistente com os dados de outra.

Abaixo estão as principais finalidades e benefícios de usar uma foreign key:

Manter a integridade referencial:

A foreign key garante que os valores de uma coluna em uma tabela correspondam a valores válidos na tabela referenciada. Por exemplo, evita que seja inserido um pedido com um código de cliente que não existe na tabela de clientes.

Criar relacionamentos entre tabelas:

Por meio das foreign keys, é possível representar relações entre diferentes entidades. Por exemplo, uma tabela de Pedidos pode ter uma foreign key que aponta para a tabela de Clientes, indicando a qual cliente cada pedido pertence.

Evitar dados órfãos:

Ajuda a prevenir registros "órfãos" — ou seja, registros que fazem referência a dados que não existem mais. Por exemplo, ao excluir um cliente, todos os pedidos relacionados podem ser automaticamente excluídos (dependendo das regras definidas), garantindo que a tabela de Pedidos não contenha registros inválidos.

Por que criar as foreign keys separadamente (fora da definição da tabela)?

Em alguns casos, é considerado uma boa prática criar as foreign keys separadamente após a criação das tabelas. Os motivos incluem:

Respeito à ordem de dependências:

Quando existe uma relação de dependência entre várias tabelas (por exemplo, tabela\_1, tabela\_2, e tabela\_3), pode haver situações em que uma foreign key aponta para uma tabela que ainda não foi criada.

Por exemplo:

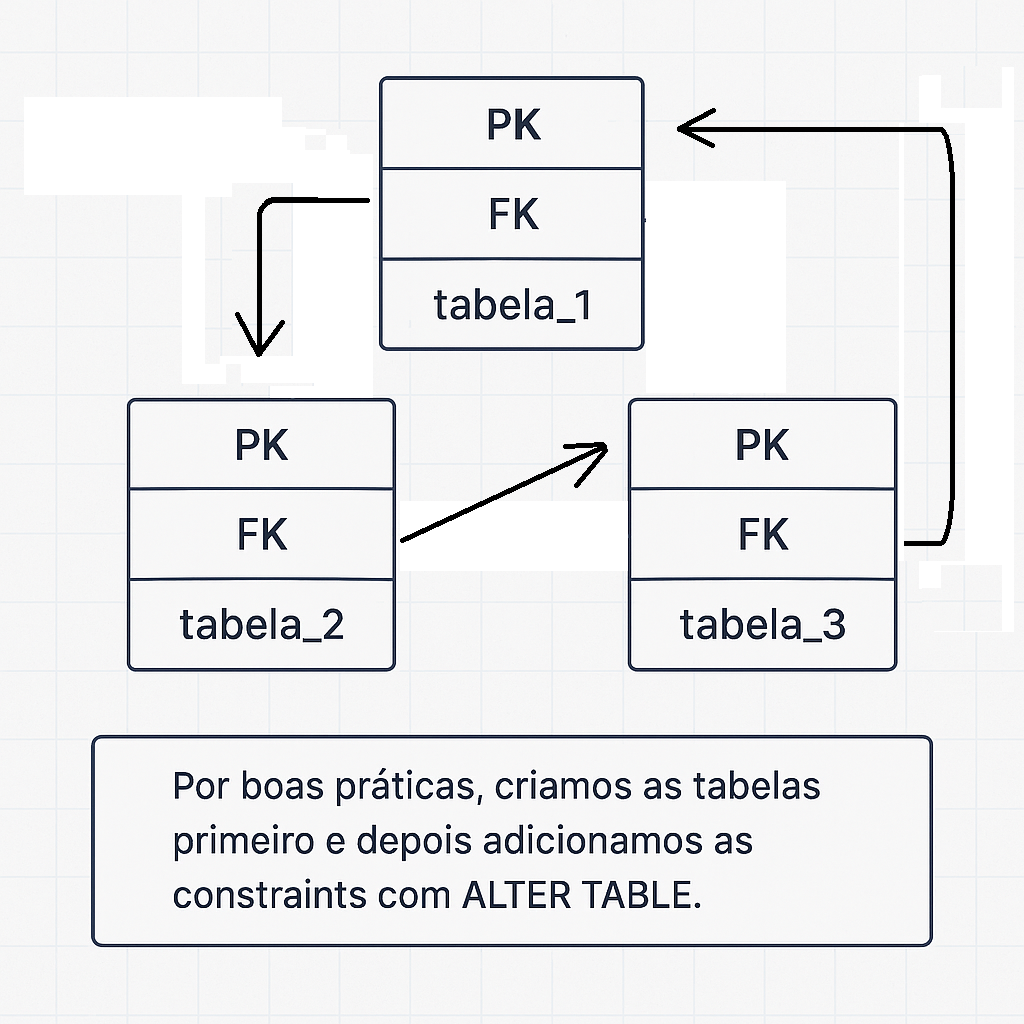
A tabela\_1 depende da tabela\_2;

A tabela\_2 depende da tabela\_3;

E a tabela\_3 volta a depender da tabela\_1.

Isso cria um ciclo de dependência que impede a criação direta com foreign keys já na definição da tabela, pois o banco de dados exige que a tabela referenciada já exista.

Abaixo uma figura que facilitará o entendimento:

Facilidade de Manutenção e Flexibilidade: 

Se você precisar alterar ou remover uma chave estrangeira em um banco de dados sem modificar diretamente a estrutura das tabelas, a definição de foreign keys separadas pode tornar o processo mais fácil e menos propenso a erros.

Essa abordagem é útil em bancos de dados dinâmicos ou grandes sistemas, onde exista a necessidade de ajustar os relacionamentos entre tabelas de forma flexível, sem causar grandes impactos na estrutura dos dados.

Desempenho e Escalabilidade:

Em grandes sistemas distribuídos ou em bancos de dados que exigem alta performance, a definição explícita de foreign keys dentro da tabela pode ser um obstáculo. Já criar as chaves estrangeiras de forma separada pode ajudar a melhorar o desempenho, pois o banco de dados pode não precisar verificar as restrições o tempo todo.

Por que definir as Foreign Keys por fora da tabela?

Há algumas vantagens e considerações em colocar as foreign keys fora da definição da tabela. Isso envolve principalmente flexibilidade no momento de manutenção do banco de dados e gestão das dependências. Aqui estão alguns pontos chave:

Vantagens de definir foreign keys fora da tabela:

Facilidade na modificação da estrutura:

Ao não definir as chaves estrangeiras diretamente nas tabelas durante a criação, fica mais fácil ajustar a estrutura das tabelas posteriormente. Você pode adicionar ou remover chaves estrangeiras com comandos ALTER TABLE, sem a necessidade de modificar a estrutura da tabela diretamente. Isso pode ser útil em ambientes de produção onde a tabela já contém dados, e modificar a estrutura da tabela pode ser mais complexo.

Mais controle sobre a integridade referencial:

Colocar as chaves estrangeiras fora das tabelas pode ser vantajoso quando você precisa de mais controle sobre o relacionamento entre as tabelas, especialmente se houver regras complexas de exclusão, atualização ou manipulação de dados relacionados. Por exemplo, você pode adicionar restrições específicas de ação (como ON DELETE CASCADE, ON UPDATE RESTRICT) que podem ser facilmente ajustadas sem precisar alterar a tabela inteira.

Desvantagens de definir foreign keys fora da tabela, pois é nem tudo são flores existem as desvantagens também abaixo alguma delas:

Potencial aumento da complexidade:

Gerenciar as dependências entre tabelas separadas pode aumentar a complexidade da administração do banco de dados, pois você precisará de scripts separados para definir ou modificar as chaves estrangeiras. Além disso, pode ser mais difícil entender rapidamente todas as relações entre as tabelas, especialmente em bancos de dados com muitas tabelas.

Consistência:

Quando você coloca as chaves estrangeiras fora das tabelas, pode ocorrer a situação de não garantir a consistência referencial durante o desenvolvimento ou manutenção do banco de dados. Isso ocorre especialmente caso esqueça de adicionar a chave estrangeira no futuro. Em comparação, se a chave estrangeira estiver diretamente na tabela, a integridade referencial é mais garantida.

Aumento de complexidade nos scripts de migração:

Quando você precisa migrar dados entre diferentes ambientes (como desenvolvimento, staging, e produção), o processo pode se tornar mais complexo se as foreign keys estiverem declaradas separadamente. Isso pode exigir scripts adicionais para garantir que as dependências sejam configuradas corretamente.

E ai qual o método escolher??? resposta: tudo depende!!!

Bem vejamos foreign key dentro das tabelas:

Isso é ideal quando queremos garantir a integridade referencial automaticamente e tiver um banco de dados com uma estrutura mais simples. Esse método também é mais indicado para projetos menores ou sistemas que não exigem uma manutenção constante da estrutura do banco de dados.

Foreign keys fora da tabela:

Esta abordagem é útil para sistemas mais complexos ou ambientes em que a estrutura do banco de dados pode ser frequentemente alterada. Colocar as foreign keys fora das tabelas proporciona maior flexibilidade, permitindo adicionar ou modificar chaves estrangeiras sem alterar a definição da tabela.

Então basicamente:

Foreign keys dentro da tabela:

Facilitam o controle de integridade referencial diretamente na definição da tabela, mas limitam flexibilidade caso seja necessário fazer mudanças na estrutura do banco.

Foreign keys fora da tabela:

Proporcionam mais flexibilidade e controle ao adicionar ou alterar as chaves estrangeiras sem modificar a estrutura das tabelas, mas podem aumentar a complexidade e o risco de inconsistências.

Aplicação no seu projeto:

No nosso projeto, estamos adotando a abordagem de definir as foreign keys fora das tabelas, utilizando um recurso conhecido como constraint. Essa escolha permite um controle mais flexível e a possibilidade de alterar os relacionamentos entre tabelas conforme necessário, sem interferir diretamente na estrutura das mesmas.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- inserção de dados.

/\*uma observação muito importante: para adicionar os registros temos que respeitar a ordem de dependência, ou seja, dentro da tabela colaborador consta 3 chaves estrangeiras, que se referem as chaves primárias das tabelas turno, departamento e cargo, sendo assim caso eu tente inserir os dados da tabela colaborador primeiro que os outros registros, os dados simplesmente não serão inseridos.\*/

insert into turno(idturno, periodo, hora\_inicial, hora\_final)

values(null, 'turno B', '14:00', '22:00'),

(null, 'turno A', '06:00', '14:00'),

(null, 'turno Admin', '06:00', '17:00'),

(null, 'turno B', '14:00', '22:00'),

(null, 'turno A', '06:00', '14:00'),

(null, 'turno B', '14:00', '22:00'),

(null, 'turno A', '06:00', '14:00'),

(null, 'turno Admin', '06:00', '17:00'),

(null, 'turno B', '14:00', '22:00'),

(null, 'turno A', '06:00', '14:00');

select \* from turno;

insert into departamento(iddepartamento, nome\_departamento)

values(null, 'Desenvolvimento Software'),

(null, 'Desenvolvimento Software'),

(null, 'Desenvolvimento Software'),

(null, 'Desenvolvimento Software'),

(null, 'Desenvolvimento Software'),

(null, 'Desenvolvimento BD'),

(null, 'Engenharia'),

(null, 'Engenharia dados'),

(null, 'RH'),

(null, 'Gerencia');

select \* from departamento;

insert into cargo(idcargo, nome\_cargo)

values(null, 'Programador jr'),

(null, 'Analista banco dados'),

(null, 'Assitente TI'),

(null, 'Programador jr'),

(null, 'Programador jr'),

(null, 'Programador senior'),

(null, 'Analista banco dados'),

(null, 'Programador jr'),

(null, 'Assistente RH'),

(null, 'Gerente');

select \* from cargo;

insert into colaborador(idcolaborador, nome, cpf, data\_nascimento, email, sexo, id\_turno, id\_departamento, id\_cargo)

values(null,"Breanna Moore","117-99-4967","1960-06-07","lynndavenport@example.org","F",1,1,1),

(null,"Alejandro Zimmerman","509-09-9072","1986-11-11","lpowers@example.net","M",2,2,2),

(null,"Michelle Page","512-45-5206","1975-04-04","psawyer@example.com","F",3,3,3),

(null,"Dr. Edward Lee","501-96-5261","1985-09-26","shermanjudith@example.net","M",4,4,4),

(null,"Ryan Smith","783-94-8884","1990-05-04","dbates@example.com","M",5,5,5),

(null,"Martin Castillo","722-95-3843","1983-08-25","robertmiller@example.com","M",6,6,6),

(null,"Samuel Marquez","695-61-8259","1968-09-22","donna46@example.com","M",7,7,7),

(null,"Brittany Hernandez","773-24-2166","1950-01-25","johnanderson@example.com","M",8,8,8),

(null,"Nicholas Walsh","321-23-1317","1989-10-18","hodgesaaron@example.net","M",9,9,9),

(null,"Daniel Smith","582-88-2893","1984-02-06","zgrant@example.com","M",10,10,10);

select \* from colaborador;

/\*id\_turno int, -- Declarando a coluna de chave estrangeira

id\_departamento int, -- Declarando a coluna de chave estrangeira

id\_cargo int -- Declarando a coluna de chave estrangeira

\*/

insert into admissao(idadmissao, data\_admissao, id\_colaborador)

values(null,"1996-09-13",1),

(null,"1990-08-28",2),

(null,"2010-11-04",5),

(null,"2010-03-09",6),

(null,"1990-10-12",8),

(null,"2014-09-17",10),

(null,"1998-01-09",3),

(null,"1992-12-23",4),

(null,"2000-02-16",7),

(null,"1988-12-25",9);

select \* from admissao;

insert into demissao(iddemissao, data\_demissao, motivo, id\_colaborador)

values(null,"2000-06-09","Pedido de demissao",1),

(null,"2001-02-16","Demissao por justa causa",2),

(null,"2019-07-20","Demissao consensual",5),

(null,"2015-05-06","Demissao por justa causa",6),

(null,"1998-05-20","Demissao sem justa causa",8),

(null,"2024-02-05","Demissao consensual",10);

select \* from demissao;

insert into endereco(idendereco, rua, cep, bairro, cidade, estado, numero, id\_colaborador)

values(null, "Parque Anhangabaú", "01007040", "CENTRO", "São Paulo", "SP", 70,1),

(null, "Passarela Carlos Drummond de Andrade","01007060", "CENTRO", "São Paulo", "SP", 71,2),

(null, "Viaduto Doutor Eusébio Stevaux", "01007040", "CENTRO", "São Paulo", "SP", 73,3),

(null, "Parque Anhangabaú", "01007040", "CENTRO", "São Paulo", "SP", 70,4),

(null, "Largo Fernando Gallego", "01007070", "CENTRO", "São Paulo", "SP", 74,5),

(null, "Rua São Benedito", "02315060", "Vila Progredior", "São Paulo", "SP", 45,6),

(null, "Avenida Ipiranga", "01037000", "Centro", "São Paulo", "SP", 150,7),

(null, "Rua dos Três Irmãos", "05698030", "Jardim Rio Bonito", "São Paulo", "SP", 33,8),

(null, "Rua Martins Fontes", "01050000", "Centro", "São Paulo", "SP", 120,9),

(null, "Avenida Paulista", "01310100", "Bela Vista", "São Paulo", "SP", 500,10);

select \* from endereco;

insert into telefone(idtelefone, numero, tipo, id\_colaborador)

values(NULL, '956657712', 'celular', 1),

(NULL, '64785241', 'resid', 1),

(NULL, '969412525', 'comerc', 1),

(NULL, '985451236', 'celular', 2),

(NULL, '996451236', 'celular', 3),

(NULL, '996451521', 'celular', 4),

(NULL, '64581263', 'resid', 5),

(NULL, '956965231', 'celular', 5),

(NULL, '996984236', 'celular', 6),

(NULL, '945789696', 'celular', 7),

(NULL, '996965336', 'celular', 9),

(NULL, '996451236', 'resid', 9),

(NULL, '978478533', 'celular', 8),

(NULL, '978964513', 'celular', 10);

select \* from telefone;

insert into salario(idsalario, valor, id\_colaborador)

values(null, 1750.75, 1),

(null, 2780.56, 2),

(null, 1982.75, 3),

(null, 2256.22, 4),

(null, 1660.55, 5),

(null, 3750.75, 6),

(null, 12230.62, 7),

(null, 13261.20, 8),

(null, 2548.63, 9),

(null, 18500.40, 10);

select \* from salario;

insert into estadocivil(idestado\_civil, tipo, id\_colaborador)

values(null, 'cadado(a)', 1),

(null, 'solteiro(a)', 2),

(null, 'divorciado(a)', 3),

(null, 'cadado(a)', 4),

(null, 'divorciado(a)', 5),

(null, 'cadado(a)', 6),

(null, 'cadado(a)', 7),

(null, 'cadado(a)', 8),

(null, 'cadado(a)', 9),

(null, 'viuvo(a)', 10);

select \* from estadocivil;

insert into beneficio(idbeneficio, vale\_transporte, vale\_refeição, id\_colaborador)

values(null, 'sim', 'sim',1),

(null, 'nao', 'sim',2),

(null, 'sim', 'nao',3),

(null, 'sim', 'nao',4),

(null, 'sim', 'sim',9),

(null, 'nao', 'nao',10),

(null, 'sim', 'sim',7),

(null, 'sim', 'nao',5),

(null, 'sim', 'sim',8),

(null, 'sim', 'sim',6);

select \* from beneficio;

-- select de todas as tabelas.

select \* from colaborador;

select \* from admissao;

select \* from demissao;

select \* from endereco;

select \* from telefone;

select \* from turno;

select \* from salario;

select \* from departamento;

select \* from cargo;

select \* from estadocivil;

select \* from beneficio;

Nos próximos posts, será o momento da Execução de Consultas e Testes no Banco de Dados

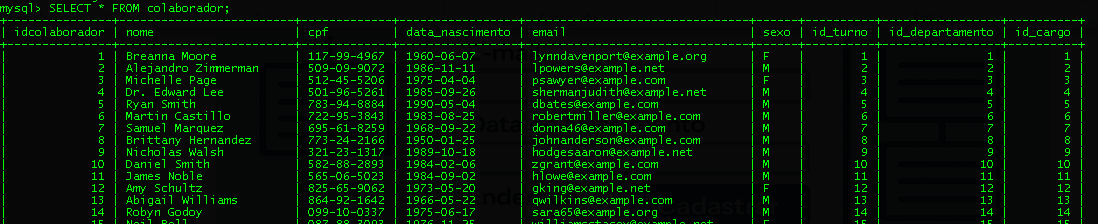
Além disso, estou populando o banco com mais de 100 registros reais de exemplo — uma ótima forma de praticar consultas SQL na prática!!! Espero que gostem!!! Tudo estará disponível no meu GitHub!: [https://github.com/Eder-Alan](https://github.com/Eder-Alan%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20)  Quase esqueci! Se você tem dicas ou já passou por projetos semelhantes em modelagem de dados com MySQL, vou adorar trocar experiências!

#BancoDeDados #SQL #Modelagem #SistemaDeGestao #DesenvolvimentoDeSistemas #Aprendizado #MySQL #DatabaseDesign #ERDiagram #SQLPractice #DevCommunity

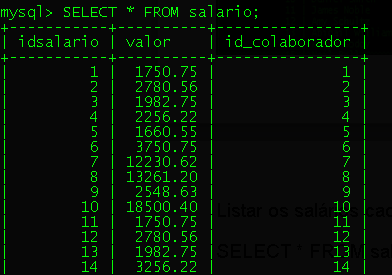
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Fase 5 do Projeto: Execução de Consultas e Testes no Banco de Dados.

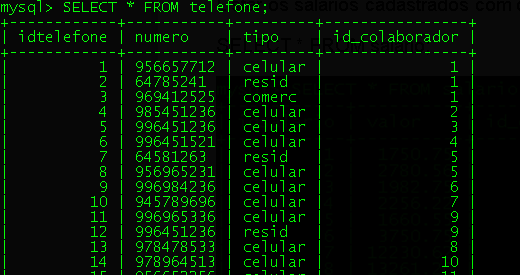
Iremos verificar o cadastro dos colaboradores e para isso iremos listar todos os colaboradores cadastrados, com o seguinte comando:

SELECT \* FROM colaborador;

Iremos agora listar os salários cadastrados como o seguinte comando: SELECT \* FROM salario:



Agora listar todos os telefones cadastrados: SELECT \* FROM telefone;



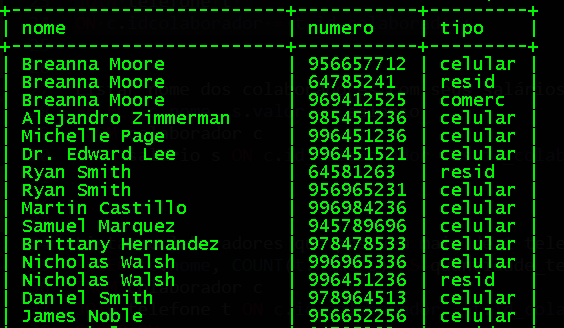
Desafio 01: listar o nome dos colaboradores com o número de seus telefones e tipos:

SELECT c.nome, t.numero, t.tipo

FROM colaborador c

INNER JOIN telefone t

ON c.idcolaborador = t.id\_colaborador;

 Resultado: Acima o resultado da consulta, que como pode ser observada permite verificar rapidamente todos os contados cadastrados de todos os colaboradores.

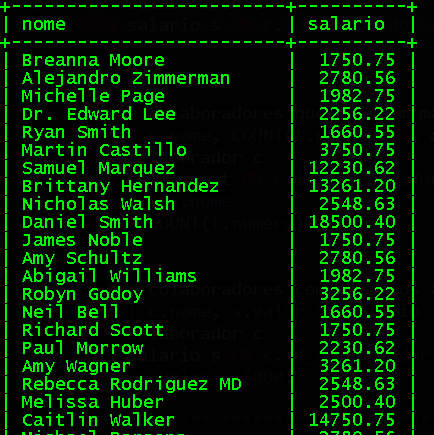
Desafio 02: listar o nome dos colaboradores com seus respectivos salários:

SELECT c.nome, s.valor AS salario

FROM colaborador c

INNER JOIN salario s

ON c.id\_colaborador = s.id\_colaborador;

 Resultado: Acima o resultado da consulta que permite de forma rápida a verificação de todos os colaboradores e seus respectivos salários, informação esta muito importante para o setor de RH.

Desafio 03: listar os colaboradores que possuem mais de 1 telefone:

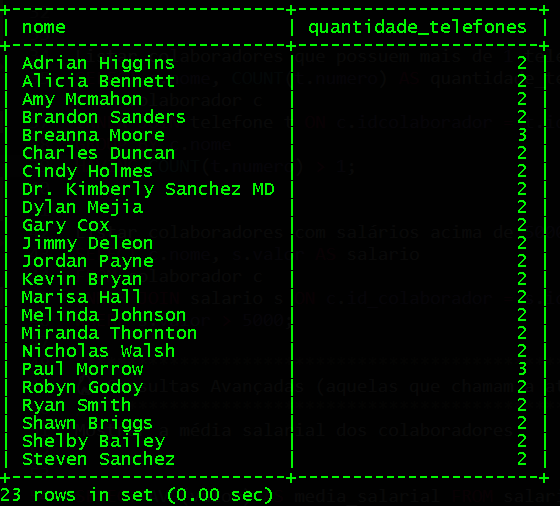
SELECT c.nome, COUNT(t.numero) AS quantidade\_telefones

FROM colaborador c

INNER JOIN telefone t

ON c.idcolaborador = t.id\_colaborador

GROUP BY c.nome

HAVING COUNT(t.numero) > 1; Resultado: Acima o resultado da consulta que permite verificar a quantidade de telefone por colaborador.

Desafio 04: listar os colaboradores com salários acima de R$5.000,00:

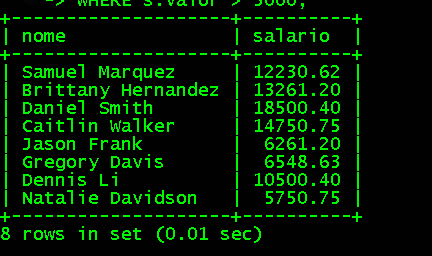
SELECT c.nome, s.valor AS salario

FROM colaborador c

INNER JOIN salario s

ON c.idcolaborador = s.id\_colaborador

WHERE s.valor > 5000;

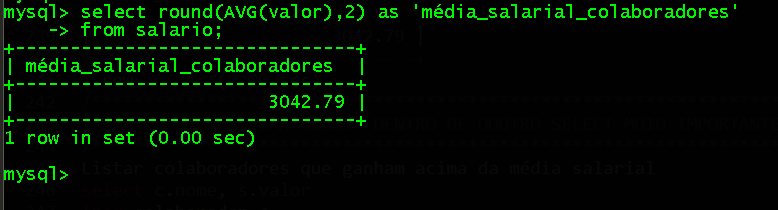


Resultado: Como pode ser observado o resultado da consulta, acima a lista dos colaboradores que ganha acima de R$5.000,00.

Desafio 05: média salarial dos colaboradores:

select round(AVG(valor),2) as 'méida\_salarial\_colaboradores'

from salario;

 Resultado: Temos acima o resultado da consulta que nos informa a média dos salários dos colaboradores.

Desafio 06: listar os colaboradores que ganham acima da média salarial:

select c.nome, s.valor

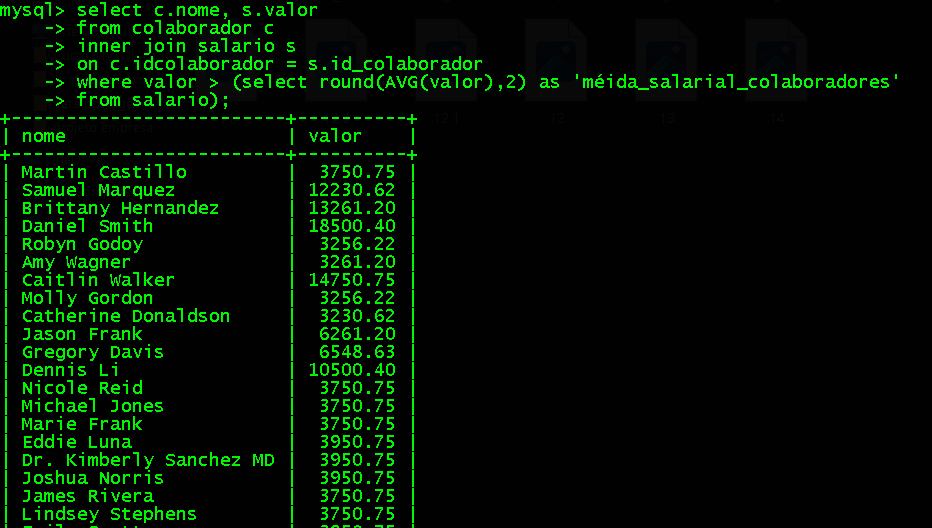
from colaborador c

inner join salario s

on c.idcolaborador = s.id\_colaborador

where s.valor > (select round(AVG(valor),2) as 'méida\_salarial\_colaboradores'

from salario);



Resultado: O resultado dessa consulta demonstra que consigo identificar facilmente os colaboradores que ganham acima da média, uma análise importante para o RH da empresa fictícia.

Desafio 07: Listar nome, telefone, tipo do telefone, salário e o cargo de todos os colaboradores numa tabela só.

SELECT c.nome,

t.numero AS telefone, t.tipo,

s.valor AS salario,

g.nome\_cargo

FROM colaborador c

INNER JOIN telefone t

ON c.idcolaborador = t.id\_colaborador

INNER JOIN salario s

ON c.idcolaborador = s.id\_colaborador

INNER JOIN cargo g

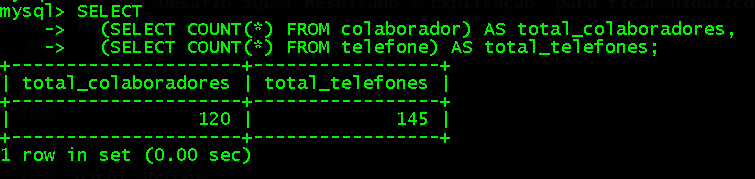
ON c.id\_cargo = g.idcargo;

Resultado: acima temos o resultado da consulta, com todas as informações solicitadas.

Desafio 8: Contar quantos colaboradores estão cadastrados e a quantidade total de telefones registrados.

SELECT

(SELECT COUNT(\*) FROM colaborador) AS total\_colaboradores,

(SELECT COUNT(\*) FROM telefone) AS total\_telefones;

Resultado: Acima temos a quantidade de colaboradores cadastrados e a quantidade de telefones registrados.

Desafio 9: Listar colaboradores por ordem de salário (do maior para o menor)

SELECT c.nome, s.valor AS salario

FROM colaborador c

JOIN salario s

ON c.idcolaborador = s.id\_colaborador

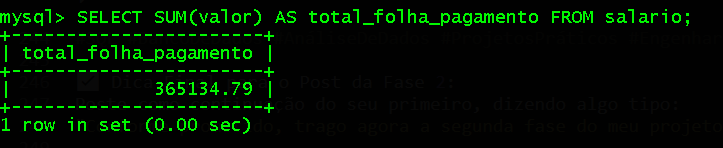
ORDER BY s.valor DESC;



Resultado: uma listagem de todos os colaboradore e seus respectivos salários, do maior para o menor valor.

Desafio 10: Desafio 9: Mostrar a soma total dos salários pagos pela empresa.

SELECT SUM(valor) AS total\_folha\_pagamento FROM salario;



Resultado: O valor total que a empresa investe mensalmente em salários. Consulta estratégica para o departamento financeiro e planejamento de custos.

**Conclusão e considerações Finais:**

Ao desenvolver este projeto pessoal, tive a oportunidade de aprender e evoluir muito. No começo, tudo parecia distante eu estava olhando o projeto de longe ainda, sem saber exatamente por onde começar ou até onde conseguiria chegar. Mas no momento que comecei a dar os primeiros passos, com a elaboração da modelagem conceitual, foi ai que tudo começou a ganhar forma. Nesta fase, construí o modelo entidade-relacionamento (MER), onde defini as entidades principais do sistema, identifiquei os atributos como nome, endereço, estado civil, cargos e etc. E não esquecendo também as relações entre cada um deles. Na sequência foi a vez do Diagrama de Entidade Relacional (DER), que nada mais é que uma representação visual da modelagem, pois utiliza símbolos e notações para representar as entidades, atributos e relacionamentos de forma gráfica, esta ferramenta visual facilita a compreensão do modelo de dados e a comunicação entre os desenvolvedores e os stakeholders do projeto.

Na sequência, avancei para a modelagem lógica, que foi um passo fundamental para transformar a ideia inicial em algo mais concreto. E foi neste momento que estruturei o banco de dados definindo claramente as tabelas, as chaves primárias que garantem a unicidade dos registros e as chaves estrangeiras que mantêm a integridade referencial entre as tabelas. Também refinei os tipos de dados, padronizei os nomes e preparei toda a base para que pudesse ser implementada de forma eficiente.

Depois, parti para a modelagem física, onde toda a estrutura desenhada anteriormente foi finalmente criada no banco de dados real. Nesta etapa, executei os comandos SQL para gerar as tabelas, configurei o ambiente para que tudo funcionasse de maneira otimizada, logo depois avancei em popular o banco inserindo os dados de cada tabela. Ver cada tabela criada e cada relacionamento funcionando foi uma experiência muito gratificante, pois era a prova de que todo aquele planejamento não foi em vão, e o projeto estava quase se concretizando, faltando apenas parte final, as consultas.

Por fim, cheguei na fase mais prática: a consulta dos dados. Neste momento, pude validar tudo o que foi construído, realizando queries SQL para buscar informações específicas, e verificar que tudo estava funcionando como o planejado, nossa como é gratificante! Cruzar dados entre diferentes tabelas e gerar respostas que comprovassem que o banco estava estruturado corretamente! Cada consulta bem-sucedida foi uma confirmação de que o projeto estava funcional e coerente com os objetivos planejados deste o início.

Sinto-me muito feliz por ter realizado este projeto. Sou um apaixonado por dados e, desde que tive meu primeiro contato com essa área na faculdade, senti que era algo que me completava. Não vou negar: enfrentei dificuldades durante o desenvolvimento, sim com certeza, afinal, foi meu primeiro projeto dessa natureza. Nunca havia feito isso antes, e justamente por isso, cada obstáculo superado foi uma conquista.

Aprendi que vídeos, cursos, livros, palestras e professores são valiosos e nos ajudam muito! Mas a verdadeira transformação acontece quando temos dentro de nós a vontade de aprender. É essa chama viva, esse desejo ardente de crescer e evoluir que nos impulsiona. Este projeto não foi apenas um exercício técnico, mas também uma prova do quanto posso me desenvolver quando uno conhecimento com paixão.

“Aprender é a única coisa de que a mente nunca se cansa, nunca tem medo e nunca se arrepende”

Leonardo da Vinci.

REFERÊNCIAS:

alura - A linguagem padrão para trabalhar com banco de dados relacionais! Disponível em: <https://www.alura.com.br/artigos/o-que-e-sql> Acesso em: 05/04/2025

alura, 2025 Modelagem Conceitual, Lógica e Física Disponível em: <https://cursos.alura.com.br/forum/topico-modelagem-conceitual-logica-e-fisica-clube-do-livro-273520> Acesso em: 06/03/2025

alura, 2025 Diferenças entre modelo lógico e modelo físico Disponível em: <https://www.alura.com.br/artigos/diferencas-modelo-logico-modelo-fisico> Acesso em: 20/03/2025

alura, 2025 Conceito de chave estrangeira 2025 Disponível em: <https://cursos.alura.com.br/forum/topico-duvida-conceito-de-chave-estrangeira-267937> Acesso em: 25/04/2025

BLOG TECNOLÓGICO, 2025 SQL : Tipos de chaves e modelos de dados Disponível em: <https://www.eng.com.br/artigo.cfm?id=7558&post=sql-:-tipos-de-chaves-e-modelos-de-dados> Acesso em: 05/05/2025

DEVEMEDIA, 2025 Introdução a modelagem conceitual Disponível em: <https://www.devmedia.com.br/introducao-a-modelagem-conceitual/10793> Acesso em: 05/03/2025

DEVEMEDIA, 2025 SQL: Aprenda a utilizar a chave primária e a chave estrangeira Disponível em: <https://www.devmedia.com.br/sql-aprenda-a-utilizar-a-chave-primaria-e-a-chave-estrangeira/37636> Acesso em: 10/05/2025

DEVEMEDIA, 2025 Modelagem de Dados Tutorial Disponível em: <https://www.devmedia.com.br/modelagem-de-dados-tutorial/20398> Acesso em: 07/03/2025

DEVEMEDIA, 2025 SQL: Aprenda a utilizar a chave primária e a chave estrangeira Disponível em: <https://www.devmedia.com.br/sql-aprenda-a-utilizar-a-chave-primaria-e-a-chave-estrangeira/37636> Acesso em: 20/04/2025

IBM, 2025 O que é SQL (structured query language)? Disponível em: <https://www.ibm.com/br-pt/think/topics/structured-query-language> Acesso em: 07/04/2025

wikipedia, 2025 Modelo físico Disponível em: <https://pt.wikipedia.org/wiki/Modelo_f%C3%ADsico> Acesso em: 20/03/2025

wikipedia - Modelagem de dados Disponível em: <https://pt.wikipedia.org/wiki/Modelagem_de_dados> Acesso em: 05/04/2025

wikipedia, 2025 SQL Disponível em: <https://pt.wikipedia.org/wiki/SQL> Acesso em: 05/04/2025

wikipedia, 2025 Chave primária Disponível em: <https://pt.wikipedia.org/wiki/Chave_prim%C3%A1ria> Acesso em: 26/04/2025

wikipedia, 2025 Chave estrangeira Disponível em: <https://pt.wikipedia.org/wiki/Chave_estrangeira> Acesso em: 04/04/2025

wikipedia, 2025 Modelagem conceitual Disponível em: <https://pt.wikipedia.org/wiki/Modelagem_conceitual> Acesso em: 05/03/2025

#BancoDeDados #SQL #Modelagem #SistemaDeGestao #DesenvolvimentoDeSistemas #Aprendizado

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/