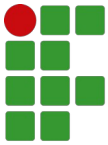


INSTITUTO FEDERAL
Mato Grosso do Sul
Campus Jardim

Licenciatura em Computação
Unidade Curricular: Projeto de Banco de Dados
Prof. Dr. Lucas Hermann Negri

Projeto de Banco de Dados Relacionais com PostgreSQL

Jardim, MS
Setembro de 2024

“Nem todos os que vagueiam estão perdidos”.

– Gandalf

Índice

1 - Introdução

Esta apostila foi escrita para auxiliar estudantes da unidade curricular Projeto de Banco de Dados do curso de Licenciatura em Computação do Instituto Federal de Mato Grosso do Sul, campus Jardim. Seu principal objetivo está na introdução ao projeto de bancos de dados relacionais na prática por meio do PostgreSQL.

Este documento está em constante atualização. Críticas e sugestões podem ser enviadas diretamente para o autor por meio do endereço de e-mail lucas.negri@ifms.edu.br. O formato preferido para impressão desta apostila é o de livreto.

1.1 - Bancos de Dados

No meu ensino médio, quando queria emprestar um livro na biblioteca da escola, a bibliotecária pegava a minha ficha (uma folha de papel) do arquivo (um grande armário com gavetinhas) e verificava se eu estava com alguma pendência. Não havendo, ela anotava o empréstimo na minha ficha e também em uma folha que ficava na contracapa, e eu estava livre para levar o livro. Neste exemplo, o conjunto de fichas de empréstimos de todos os usuários da biblioteca compõem um banco de dados. Com o passar do tempo, os bancos de dados deixaram de ser físicos (papel) e passaram ser digitais. Nesta apostila nos importamos com bancos de dados digitais.

Existem diferentes formas de organizar os dados em um computador, cada forma com suas vantagens e desvantagens. Tudo depende de como que queremos utilizar estes dados: um sistema para uma pequena biblioteca tem requisitos muito diferentes do que o sistema de uma grande rede social com vários milhões de usuários. Quando falamos sobre a forma de organização dos dados, podemos dividir os bancos nas seguintes categorias:

- **Relacionais:** onde os dados são organizados como linhas em tabelas;
- **Chave e valor:** bancos mais simples onde podemos atrelar um valor a uma chave arbitrária;
- **Orientados a documentos:** dados são organizados em documentos, que podem ter uma estrutura mais flexível quando comparado as tabelas e também ter estruturas aninhadas;
- **Outros:** orientados a grafos, orientados a colunas, etc.

Raramente trabalharemos diretamente com os dados armazenados em um banco. Para isto, utilizamos um conjunto de programas responsável por criar, buscar e

gerenciar estes bancos. A este conjuntos de programas dá-se o nome de sistema gerenciador de bancos de dados (SGBD).

O PostgreSQL é um popular SGBD relacional, que pode ser manipulado por meio da linguagem SQL (linguagem de consulta estruturada, do inglês *Structured Query Language*). Os comandos em SQL podem ser enviados por meio do seu *shell* ou por meio de programas externos, como o pgAdmin. Além disto, também pode ser utilizado por linguagens de programação como JavaScript, Python e Java.

1.2 - Exemplos

Esta apostila contém exemplos de modelagem de bancos de dados e de comandos em SQL. Quando possível utilizaremos realce de sintaxe para que o código fique claro e legível. Por padrão, os exemplos devem ser executados diretamente na interface de linha de comando (shell ou pgAdmin) do PostgreSQL.

Lembre-se que alguns exemplos dependem de um contexto. Por exemplo, um comando para buscar um documento em uma dada coleção, para ter um resultado significativo, requer que você tenha inseridos previamente documentos na coleção em questão.

1.3 - Exercícios

Todos os capítulos terminam com uma seção de exercícios relacionados aos temas estudados. A resolução destes exercícios é essencial para a aprendizagem, que necessita da prática para se estabelecer.

A dificuldade dos exercícios é variada, sendo alguns destes marcados como “desafios”, que podem requerer conhecimentos que vão além desta apostila (consulte colegas, professores e outros materiais para resolvê-los).

1.4 - Requisitos

Para executar os exemplos de código e testar os exercícios recomenda-se que o leitor tenha acesso a um computador com um cliente para o PostgreSQL (como o pgAdmin) e acesso a um servidor PostgreSQL (local ou um serviço *web* como o ElephantSQL). Também é possível utilizar o SQL Fiddle, selecionando PostgreSQL 9.6 como SGBD.

Os diagramas, figuras e modelos podem ser feitos à mão utilizando papel e caneta ou qualquer programa para diagramação ou desenho. Recomendo o programa

DrawIO que é gratuito e de código aberto (utilizado para os diagramas desta apostila).

1.5 - Material Complementar

Foram disponibilizadas no Youtube todas as aulas gravadas para a unidade curricular Projeto de Banco de Dados, ministradas em 2021 para o curso de Tecnologia em Análise e Desenvolvimento de Sistemas do IFMS campus Nova Andradina. As videoaulas não estão necessariamente na ordem da apostila, mas podem auxiliar no seu entendimento. O endereço para acesso é https://encurtador.com.br/prvGZ .

Figura 1: Tio projetista de banco de dados te convidando a estudar esta apostila.

2 - Modelo Entidade Relacionamento

Este capítulo apresenta os conceitos básicos do modelo Entidade Relacionamento, contendo a definição dos conceitos de entidade e seus atributos, de relacionamentos e suas cardinalidades e exemplos de modelagem.

2.1 - Princípios de modelagem

O desenvolvimento de um novo banco de dados se inicia na fase de projeto, onde pensamos sobre quais informações queremos armazenar no banco, como que os dados se relacionam entre si, quais restrições devem ser impostas entre outros questionamentos. Para isto, temos como base os requisitos que foram levantados no processo de análise de requisitos, que é estudado na disciplina de engenharia de software.

Como auxílio para esta fase de projeto, podemos utilizar um modelo para expressar as ideias e decisões. Neste contexto destaca-se o modelo Entidade Relacionamento (ER), que nos permite modelar o banco de dados em alto nível, isto é, nos preocupando com os conceitos principais, sem entrar nos preocuparmos com detalhes.

De forma resumida, a utilidade do modelo ER está na sua capacidade de visualizar e comunicar a estrutura e o relacionamento dos dados em um sistema de banco de dados de forma clara e compreensível. Ele permite aos projetistas de banco de dados capturar as informações essenciais do domínio e criar um esquema de banco de dados que atenda aos requisitos do sistema.

2.2 - Entidades

Como evidenciado por seu nome, o modelo ER é composto por entidades e relacionamentos entre as entidades. Uma entidade é uma coisa ou objeto do mundo real que possui características distintas e é considerada importante o suficiente para ser representada no banco de dados. Por exemplo, em um sistema de gerenciamento de uma biblioteca, as entidades podem incluir *livro*, *autor* e *editora*.

As entidades possuem atributos que a descrevem. No exemplo da biblioteca, um usuário pode possuir os atributos CPF, nome e e-mail. Note que escolhemos quais atributos utilizar de acordo com a sua importância para o sistema, portanto, no sistema da biblioteca, não precisamos saber qual o time de futebol favorito do usuário.

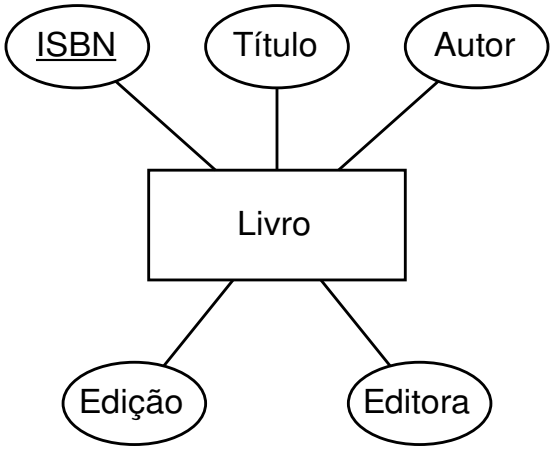


Figura 2: Entidade Livro no contexto de um sistema de gerenciamento de biblioteca.

A Figura 2 mostra, em forma de diagrama (também chamado de Diagrama Entidade Relacionamento ou DER), a entidade *Livro* de um sistema de gerenciamento de biblioteca. A entidade é representada pelo retângulo, enquanto seus atributos são representados pelas ovais. O atributo ISBN é o atributo chave, evidenciado pelo texto sublinhado.

Atributos chave são aqueles que identificam unicamente uma instância da entidade. Por exemplo, cada livro possui um código ISBN único. Existe a possibilidade que uma entidade possua um atributo chave composto, isto é, formado por dois ou mais atributos simples. Um exemplo é o número de RG de uma pessoa: é necessário saber, além do número do RG, qual foi o órgão que fez a expedição do mesmo.

Os atributos podem ser naturais (que existem no mundo real) ou artificiais (gerados pelo sistema). Por exemplo, é comum que um sistema gere um código interno único (chave artificial) para servir de identificador para usuários de um sistema (usuário 1, usuário 42, etc) em vez de usar o CPF (que seria uma chave natural) para reduzir a chance de exposição de dados pessoais ou para a simplificação do sistema.

2.3 - Relacionamentos

Entidades podem ser ligadas por meio de relacionamentos, que indicam como as entidades estão conectadas umas às outras. Por exemplo, um relacionamento chamado *Escreve* pode estabelecer a ligação entre um autor e seu livro.

A Figura 3 mostra um DER que modela o relacionamento *Escreve* (*Autor Escreve Livro*), que liga a entidade *Autor* com a entidade *Livro*. Enquanto que as entidades são representadas como retângulos, o relacionamento é representado como um losango. As

arestas que ligam o relacionamento às entidades conterão valores que podem ser 1 ou N (que significa *vários*). Estes valores representam a cardinalidade do relacionamento. Neste exemplo, um *Autor* pode escrever N (vários) livros, mas um livro é escrito por somente 1 autor. Os relacionamentos mais comuns são os 1 para 1, 1 para N e N para N.

No modelo da Figura 3, a entidade *Livro* deixou de ter o atributo *Autor*, pois a informação de autoria passou a ser representada pelo relacionamento entre *Autor* e *Livro* .

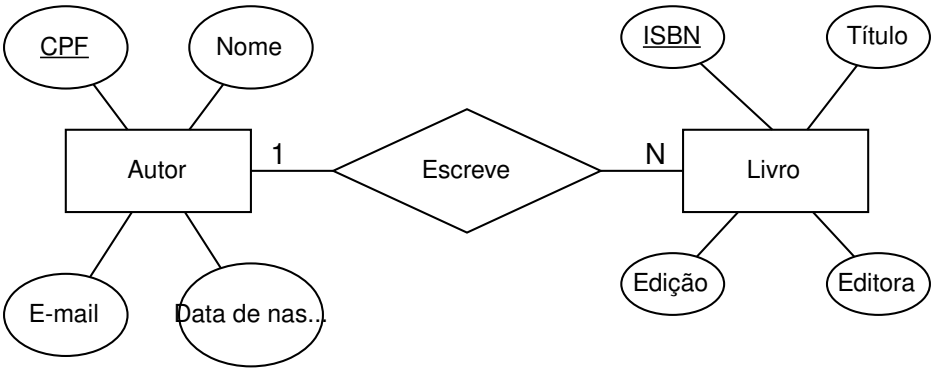


Figura 3: Exemplo de relacionamento entre as entidades Livro e Autor.

Os relacionamentos também podem ter atributos próprios, diferentemente das entidades, não podem ter atributos chave. Também é possível que existam relacionamentos envolvendo mais de duas entidades, além de relacionamentos que envolvam uma entidade com ela mesma (auto relacionamento).

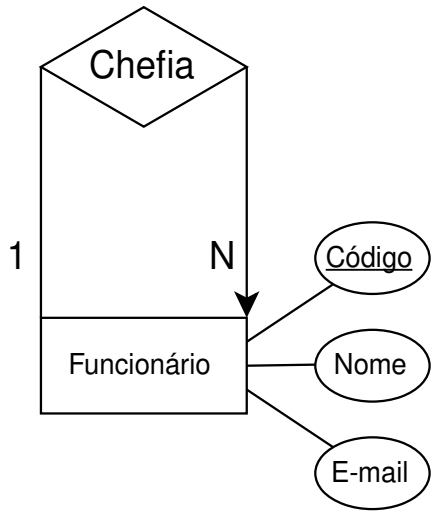


Figura 4: Exemplo de auto relacionamento, onde um funcionário pode ser chefe de um grupo de funcionários.

Um exemplo de auto relacionamento (*Funcionário chefia funcionário*) pode ser visto na Figura 4, onde um funcionário pode chefiar vários outros funcionários, mas possui até no máximo 1 chefe, que também é um funcionário. Neste caso, podemos adicionar uma seta para auxiliar na leitura do sentido, necessária para entender a cardinalidade.

A Figura 5 mostra um exemplo de relacionamento 1 para 1. Neste trecho de modelo do sistema de gestão das escolas de um município, uma escolha possui somente um funcionário diretor, e um funcionário só pode ser diretor de uma única escola.

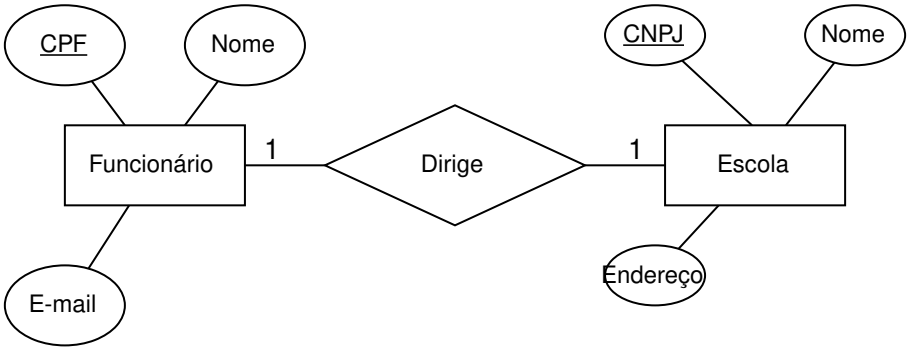


Figura 5: Exemplo de relacionamento 1 para 1 - funcionário dirige escola.

Já na Figura 6 temos um trecho de modelagem de um sistema de comércio eletrônico, onde um usuário pode marcar vários produtos como favorito, e um produto pode ser marcado por vários usuários, ou seja, um relacionamento N para N.

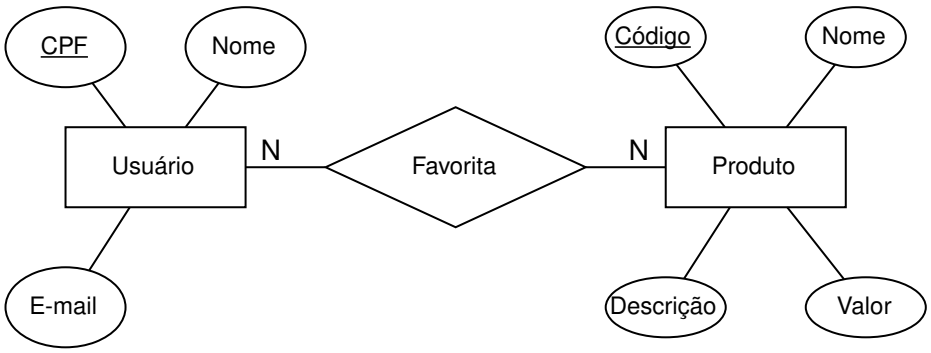


Figura 6: Exemplo de relacionamento N para N - usuário favorita (marca como favorito) produto.

2.4 - Exemplo de Modelagem

De forma a exemplificar o processo de modelagem ER, desenvolveremos nesta seção o modelo de um sistema de gestão de biblioteca com base em uma lista de requisitos que já havia sido obtida com a direção do estabelecimento.

Requisitos funcionais:

- A biblioteca possui uma vasta coleção de livros de literatura que podem ser emprestados para usuários cadastrados;
- O sistema funciona localmente, possuindo um computador para consultas (que todos podem utilizar para buscar por obras) e um computador para uso do(a) bibliotecário(a), que registrará os empréstimos de livros;
- Deve-se guardar o CPF, nome, e-mail e ano de cadastro de cada usuário;
- O sistema de busca deve permitir busca por ISBN, título, por autor e por editora;
- Armazena-se o e-mail dos autores para possíveis contatos, assim como a sua data de nascimento para envio automático de e-mails de “feliz aniversário”;
- Armazena-se o CNPJ, o nome e o e-mail de contato das editoras;
- Todos os empréstimo possuem duração de até 15 dias, devendo-se registrar a data da retirada;
- Os empréstimos são controlados individualmente, por livro;
- Cada livro é escrito por apenas um autor principal e publicado por somente uma editora.

Com base nos requisitos funcionais, chegamos no modelo ER descrito pelo diagrama da Figura 7. Não existe somente uma modelagem correta, portanto seria possível que um outro profissional obtivesse um modelo diferente que também atendesse aos requisitos listados.

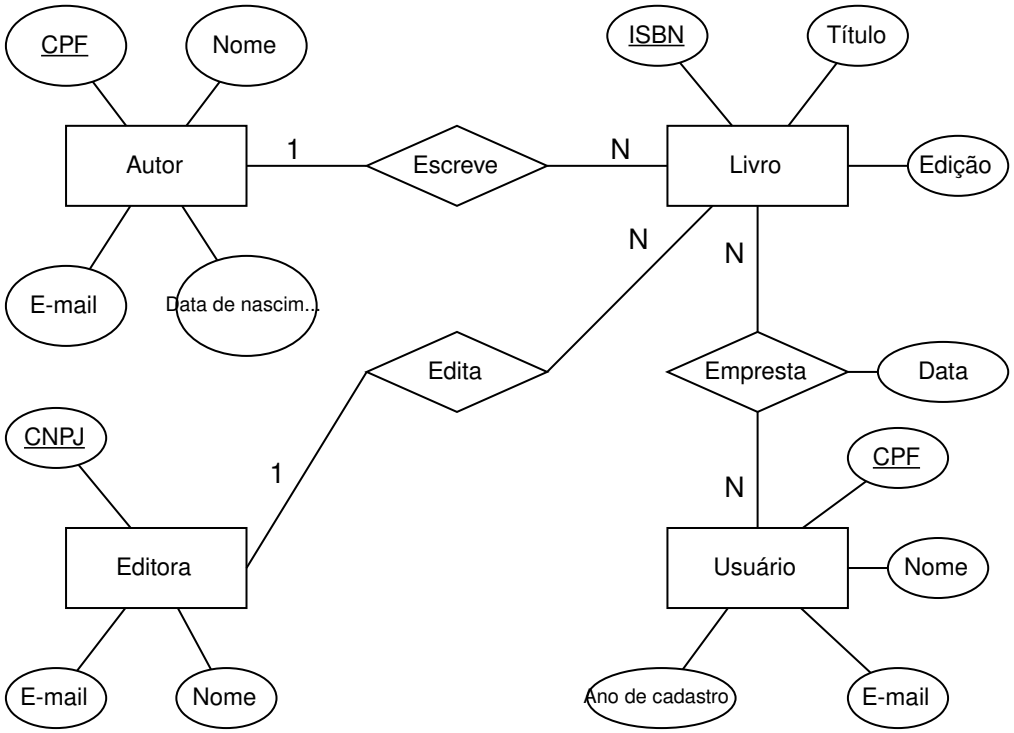


Figura 7: Diagrama ER do sistema de gerenciamento de biblioteca.

2.5 - Exercícios

1. Considere um sistema de gerenciamento de uma escola. Identifique 3 entidades relevantes e seus atributos, identificando o atributo chave de cada entidade.

2. Desenhe um diagrama ER para um sistema de uma locadora de jogos de videogame (afinal, locadoras de filmes já não existem mais!). Indique as entidades, os atributos de cada entidade e os relacionamentos entre elas.

3. Em uma universidade, os estudantes precisam fazer um trabalho de conclusão de curso, podendo ter até múltiplos professores orientadores simultaneamente. Considerando que um professor pode orientar vários trabalhos, faça um diagrama ER que mostre as entidades Estudante e Professor, além do relacionamento de orientação. Utilize atributos tanto nas entidades quanto no relacionamento resultante para armazenar as informações relevantes. Não se esqueça de escolher a cardinalidade adequada para o relacionamento!

4. Você foi contratado para projetar o sistema de um comércio eletrônico. Desenha um diagrama ER, modelando o sistema.

5. Faça a modelagem de um sistema para a gestão de uma universidade, de acordo com os seguintes requisitos:

    a) A universidade é dividida em departamentos, sendo que cada departamentos possui um nome único, um professor diretor e uma série de cursos ofertados;

    b) Cada curso possui um nome único, uma data de início de oferta, um professor coordenador e um conjunto de disciplinas ofertadas;

    c) Cada docente possui um código, nome, e-mail, telefone e área de atuação, e é vinculado a um departamento, mas pode ministrar disciplinas de diferentes cursos;

    d) Um discente possui um código único de matrícula, nome, data de nascimento, telefone e e-mail e é matriculado em no máximo um curso simultaneamente, e pode estar matriculado em várias disciplinas;

    e) Uma disciplina possui um código único, um nome, uma carga horária, o nível e uma descrição e é ofertada em uma determinada data por um único professor, podendo ter vários estudantes matriculados.

    f) O sistema deverá armazenar a nota e a frequência final do estudante por disciplina cursada (um estudante pode vir a cursar diferentes ofertas de uma mesma disciplina, até obter aprovação!).

3 - Modelo Relacional

O modelo relacional, proposto originalmente pelo cientista da computação Edgar F. Codd em 1970[1] é amplamente utilizado em sistemas de banco de dados. Neste modelo, os dados são organizados em tabelas, também conhecidas como relações. Cada tabela é composta por linhas, chamadas de tuplas, e colunas.

As tabelas em um sistema relacional podem ter relacionamentos, representados por meio de chaves. Uma chave primária é usada para identificar exclusivamente cada tupla em uma tabela. Além disso, chaves estrangeiras são utilizadas para estabelecer relacionamentos entre diferentes tabelas, garantindo a integridade referencial.

A utilidade do modelo relacional na modelagem de sistemas de banco de dados é sua capacidade de representar relações complexas de maneira organizada e eficiente. Ele permite que os dados sejam armazenados de forma estruturada, garantindo a consistência, integridade e segurança das informações.

Neste capítulo aprenderemos como obter o modelo relacional de um sistema a partir do seu modelo entidade relacionamento, para posterior implementação por meio da linguagem SQL.

3.1 - Tabelas

No modelo relacional, uma tabela possui colunas e linhas. A Figura 8 é um exemplo de tabela que armazena os dados de livros. Esta modelagem é equivalente àquela feita por meio do modelo ER visto na Figura 2. Com objetivo didático, mostramos algumas linhas como exemplos de livros que poderiam ser cadastrados.

ISBN	**Título**	**Autor**	**Edição**	**Editora**
8533613377	O Senhor dos Anéis	J. R. R. Tolkien	1	Martins Fontes
7777777777	Introdução ao PostgreSQL	Lucas H. Negri	2	Editora IFMS
1231231234	Tudo sobre Elefantes	E. L. Fante	5	Mundo Animal
345678697	Manual do Programador das Galáxias	Arthur Dent	18	SuperDev

Figura 8: Tabela Livro.

1 CODD, Edgar F. A relational model of data for large shared data banks. Communications of the ACM, v. 13, n. 6, p. 377-387, 1970.

3.2 - Chaves Primárias e Estrangeiras

Em uma tabela, a chave primária identifica unicamente as entradas (linhas) da tabela, e é equivalente ao atributo chave visto no modelo ER. Uma chave primária pode ser composta por uma única coluna ou por múltiplas colunas (chave primária composta).

Como exemplo, poderíamos escolher a coluna CPF como chave primária para a tabela Pessoa, pois cada pessoa tem um CPF diferente. Poderíamos também utilizar o número do RG e o órgão expedidor como uma chave primária composta (não poderíamos usar só o RG pois o número em si não é único, pois existem vários órgãos diferentes que fazem a emissão, cada um com seu registro próprio).

Na modelagem relacional também temos o conceito de chave estrangeira, que referencia a chave primária de uma tabela. Utilizamos a chave estrangeira quando queremos nos referir a uma entrada presente em outra tabela, sem precisar repetir todas as informações já cadastradas. Por exemplo, a Figura 9 mostra a tabela Avaliação, que possui como chave primária a coluna código (identificada por estar sublinhada) e o ISBN como chave estrangeira (referenciando a coluna ISBN da tabela Livro da Figura 8). Neste caso, a avaliação de código 1 está avaliando o livro "O Senhor dos Anéis, que é referenciado pelo seu ISBN (8533613377).

Código	**ISBN**	**Comentário**	**Nota**
1	8533613377	Adorei o personagem Gollum, muito fofo!	9
2	8533613377	Bom livro, mas o Gandalf é OP	7
3	1231231234	Não gosto de elefantes	5
4	345678697	Nem li ainda, quando ler mudo a nota	1

Figura 9: Tabela Avaliação.

3.3 - Algoritmo de Transformação do Modelo ER para o Modelo Relacional

Uma prática comumente utilizada na modelagem de bancos de dados está em criar o modelo ER, que é mais simples de entender e por ser mais abstrato pode contar com a participação de profissionais de áreas diferentes, e então obter o modelo relacional a partir do primeiro.

Podemos obter o modelo relacional a partir de um modelo ER já existente, seguindo o seguinte algoritmo:

I. Transformar cada entidade existente em uma tabela. Os atributos da entidade se transformam em colunas das tabelas, sendo o atributo chave transformado em chave primária;

II. Implementar os relacionamentos de acordo com sua cardinalidade:

   a) Se o relacionamento for 1 para 1: escolher uma das tabelas e adicionar uma chave estrangeira para a outra. Se existir atributos no relacionamento, adicioná-los como colunas na tabela escolhida;

   b) Se o relacionamento for 1 para N: fazer o mesmo que no item II a), mas adicionando as colunas obrigatoriamente na tabela do lado do N;

   c) Se o relacionamento for N para N: criar uma tabela intermediária e adicionar nela uma chave estrangeira para as duas tabelas originais, adicionando nesta tabela também, em forma de coluna, os atributos presentes no relacionamento.

3.4 - Exemplo

Para exemplificar o uso do algoritmo descrito na Seção 3.3 iremos transformar o modelo ER visto na Figura 7(modelo da biblioteca) para o modelo relacional. Iniciaremos criando uma tabela para cada uma das entidades (Autor, Livro, Editora e Usuário), obtendo como resultado o modelo da Figura 10.

CPF	**Nome**	**E-mail**	**Data de nascimento**

Tabela Autor

ISBN	**Título**	**Edição**

Tabela Livro

CNPJ	**Nome**	**E-mail**

Tabela Editora

CPF	**Nome**	**E-mail**	**Ano de cadastro**

Tabela Usuário

Figura 10: Modelo relacional (em construção) do sistema de gerenciamento de biblioteca.

Na sequência, implementamos os relacionamentos:

a) "Autor Escreve Livro" é implementado adicionando uma chave estrangeira (Autor) em Livro;

b) "Editora Edita Livro" é implementado adicionando uma chave estrangeira (Editora) em Livro;

c) “Usuário Empresta Livro” é implementado com a criação da tabela intermediária Empresta, que conterá as chaves estrangeiras (Usuário e Livro) e a coluna Data.

A Figura 11 mostra o modelo relacional final. Note que desta vez não colocamos entradas para servir de exemplo, estando apenas interessados nas tabelas em si.

CPF	**Nome**	**E-mail**	**Data de nascimento**

Tabela Autor

ISBN	**Título**	**Edição**	**Autor**	**Editora**

Tabela Livro

CNPJ	**Nome**	**E-mail**

Tabela Editora

CPF	**Nome**	**E-mail**	**Ano de cadastro**

Tabela Usuário

Usuário	**Livro**	**Data**

Tabela Empresta

Figura 11: Modelo relacional (final) do sistema de gerenciamento de biblioteca.

3.5 - Exercícios

1. Considerando o modelo relacional do sistema de gerenciamento de biblioteca (Figura *11*), adicione 2 linhas na tabela Autor, 2 linhas na tabela Editora, 3 linhas na tabela Livro, 2 linhas na tabela Usuário e 4 linhas na tabela empresta.

2. Ainda no contexto da questão anterior, responda o motivo pelo qual devemos inserir os autores antes de inserir os empréstimos.

3. Transforme o modelo ER visto na Figura 5 para o modelo relacional.

4. Transforme o modelo ER visto na Figura 6 para o modelo relacional. Será necessário criar uma tabela intermediária.

5. Transforme o modelo ER visto na Figura 4 para o modelo relacional. Aqui seguimos o mesmo procedimento feito quando o relacionamento é entre duas entidades distintas.

6. Obtenha o modelo relacional do sistema de gestão de universidade modelado originalmente nos exercícios do capítulo 2.

4 - Linguagem SQL: criação de tabelas e inserção de dados

Nos capítulos anteriores aprendemos como fazer a modelagem de um banco de dados com diagramas entidade relacionamento e então como transformá-los em tabelas. Agora aprenderemos como criar de fato as tabelas no banco, utilizando para isto a linguagem SQL (*Structured Query Language*).

A SQL é uma linguagem padronizada utilizada na maioria dos SGBDs relacionais, incluindo o PostgreSQL. Entretanto, cada SGBD possui suas peculiaridades, e por isso o código que escreveremos para o PostgreSQL nesta apostila precisaria de alterações para funcionar com o MySQL, por exemplo.

4.1 - Criação de tabelas com o CREATE TABLE

A criação de tabelas é realizada utilizando o comando CREATE TABLE. Neste comando, especificamos o nome da tabela a ser criada, assim como as suas colunas. Cada coluna possui um nome e um tipo de dados, podendo ter também alguma restrição de integridade.

A Figura 12 mostra um exemplo de uso do comando CREATE TABLE, que é utilizado para criar a tabela Autor. Cada coluna possui um tipo de dados, e a coluna CPF é marcada como sendo uma chave primária por meio da palavra-chave PRIMARY KEY. Ser uma chave primária implica em duas restrições de integridade: a) o valor nunca pode ficar vazio e b) o valor deve ser único, isto é, não pode se repetir na tabela.

Por padrão as colunas são opcionais, logo podemos usar a palavra-chave NOT NULL para tornar obrigatória a coluna que quisermos. Note que toda coluna marcada como PRIMARY KEY já é automaticamente NOT NULL também!

```
CREATE TABLE Autor (
        CPF TEXT PRIMARY KEY,
        nome TEXT NOT NULL,
        email TEXT,
        dataNascimento DATE
);
```

Figura 12: Exemplo de uso do comando CREATE TABLE: criação da tabela Autor.

A Tabela 1 mostra alguns tipos de dados mais utilizados no PostgreSQL.

Tipo	Descrição
TEXT	Sequência de caracteres. Utilizar para textos e também para valores numéricos que não serão utilizados em cálculos (como o número do CPF, por exemplo!). É preferível utilizar o TEXT em vez do CHAR e VARCHAR. Os valores textuais são definidos entre aspas simples. Exemplos: 'IFMS', 'Gosto do PostgreSQL', '123.456.789-01'.
INT	Número inteiro (isto é, "sem vírgula"). Exemplos: 5, -8, 92.
REAL	Números "com vírgula". Exemplos: 5.2, 8, 3.14.
DATE	Representa uma data. Representado no formato ANO-MES-DIA, entre aspas simples. Exemplo: '2023-08-25'.
TIMESTAMP	Representa uma data com a informação de horário, no formato ANO-MES-DIA HORA:MINUTO:SEGUNDO, entre aspas simples. Exemplo: '2023-08-25 12:45:18'.
SERIAL	Número inteiro semelhante ao INT, mas que, por padrão, assume valores em uma sequência (1, 2, 3, ...). Utilizado principalmente quando você quer que o próprio banco de dados gere um código único para cada linha da tabela.

Tabela 1: Operadores de atualização mais utilizados

A Figura 13 mostra a criação da tabelas Editora e Livro, exemplificando o uso de chaves estrangeiras que são declaradas por meio da restrição de integridade REFERENCES, que referencia uma coluna de uma tabela. Neste caso, só podemos inserir dados na chave estrangeira que existam na tabela referenciada. Só é possível referenciar uma tabela que já exista, logo a ordem de execução do CREATE TABLE importa (devemos criar primeiro as tabelas Autor e Editora, para aí então criar a tabela Livro).

Como regra, as chaves estrangeiras que criaremos sempre "apontam" para chaves primárias. No exemplo da Figura 13, temos duas chaves estrangeiras na tabela Livro: a chave estrangeira autor, que referencia a coluna CPF da tabela Autor (uma chave primária) e a chave estrangeira editora, que referencia a coluna CNPJ da tabela Editor (uma chave primária).

```
CREATE TABLE Editora (
        CNPJ TEXT PRIMARY KEY,
        nome TEXT NOT NULL,
        email TEXT
);

CREATE TABLE Livro (
        ISBN TEXT PRIMARY KEY,
        titulo TEXT NOT NULL,
        edicao INT,
        autor TEXT REFERENCES Autor(CPF),
        editora TEXT REFERENCES Editora(CNPJ)
);
```

Figura 13: Exemplo de uso do comando CREATE TABLE: criação das tabelas Editora e Livro.

4.2 - Inserção de dados com o INSERT

O comando CREATE TABLE cria tabelas que estão inicialmente vazias. Para armazenar informações em uma tabela, utilizamos o comando INSERT INTO. A sintaxe do comando pode ser vista no exemplo da Figura 14, onde inserimos uma nova pessoa na tabela Autor e uma nova editora na tabela Editora.

```
INSERT INTO autor VALUES ('123.456.789-01', 'Joãozinho Escriba',
'jao@escriba.com.br', '1950-11-20');

INSERT INTO editora VALUES ('11.222.333/0001-23', 'NA comics',
'na.comics@gmail.com');
```

Figura 14: Exemplo de uso do comando INSERT INTO para inserir um novo autor e uma nova editora.

No exemplo da Figura 15 vemos como inserir uma linha na tabela Livro, especificando valores para as duas chaves estrangeiras.

```
INSERT INTO livro VALUES ('123-1-1234-1234-1', 'Escrevendo poesias
com JavaScript', 1, '123.456.789-01', '11.222.333/0001-23');
```

Figura 15: Exemplo de uso do comando INSERT INTO para inserir um novo livro.

Algumas colunas, como é o caso daquelas com o tipo SERIAL, possuem um valor padrão e, para utilizá-lo, basta especificarmos DEFAULT como valor. As colunas que forem opcionais (ou seja, que não são marcadas como NOT NULL ou PRIMARY KEY) podem ser deixadas vazias especificando NULL como valor.

A Figura 16 mostra um exemplo de uso do tipo serial: criamos a tabela produto e inserimos itens, deixando o próprio banco de dados gerar os valores para *codigo* (serão atribuídos os valores 1, 2, 3, ...). A coluna estoque determina quantos produtos estão atualmente em estoque, e o valor é o preço em reais[2].

```
CREATE TABLE produto (
    codigo SERIAL PRIMARY KEY,
    nome TEXT NOT NULL,
    estoque INT NOT NULL,
    valor REAL NOT NULL
);

INSERT INTO produto(nome, estoque, valor) VALUES ('Mouse Gamer', 10,
500.35);

INSERT INTO produto(nome, estoque, valor) VALUES ('Teclado RGB', 22,
750.12);

INSERT INTO produto VALUES (DEFAULT, 'Monitor 240Hz', 5, 1620.21);
```

Figura 16: Exemplo de uso do tipo SERIAL. Os produtos terão seus códigos atribuídos automaticamente, de forma sequencial, pelo banco de dados.

4.3 - Exercícios

1. Considerando o modelo relacional do sistema de gerenciamento de biblioteca (Figura *11*), escreva o código SQL para criação das tabelas.

2. Continuando o exercício anterior, escreva o código SQL para inserir valores em todas as tabelas criadas.

2 Existem tipos mais adequados para armazenamento de valores monetários, como o MONEY. Utilizaremos o REAL por simplicidade.

3. Considere o modelo relacional do sistema de gestão de universidade originalmente modelo nos exercícios do capítulo 2 e 3. Crie as tabelas utilizando CREATE TABLE, e insira ao menos 2 linhas em cada tabela utilizando o comando INSERT INTO.

Figura 17: Um monte de dados

5 - Linguagem SQL: consultas simples

No capítulo anterior, aprendemos como criar tabelas e inserir valores. Neste capítulo aprenderemos como recuperar (também chamado de consultar ou buscar) os valores já salvos em um tabela. Em alguns casos iremos recuperar todos os valores que foram inseridos, já em outros iremos pesquisar por entradas que satisfaçam uma determinada condição. Por fim, aprenderemos como ordenar os resultados de um consulta.

5.1 - Consultas em tabelas com o SELECT

O comando SELECT é utilizado para recuperar valores de uma tabela. A sua sintaxe básica é simples: especificamos quais são as colunas a serem recuperadas e então o nome da tabela. A Figura 18 mostra um exemplo: queremos recuperar as colunas ISBN, titulo e edição da tabela chamada *livro*.

Se quisermos recuperar todas as colunas de uma tabela sem ter que descrever coluna por coluna, podemos substituir a listagem de colunas por um asterisco (*), como visto na Figura 19.

```
SELECT ISBN, titulo, edicao FROM livro;
```

Figura 18: Uso do SELECT para recuperar os dados inseridos na tabela livro, mas somente das colunas ISBN, titulo e edicao.

```
SELECT * FROM livro;
```

Figura 19: Uso do SELECT para recuperar os dados inseridos na tabela livro, considerando todas as colunas.

5.2 - Filtragem de dados com o WHERE

Podemos criar consultas que recuperem somente as linhas que satisfaçam uma condição específica, isto é, podemos fazer buscas com filtros. Um filtro pode ser especificado por meio da palavra-chave WHERE.

Um exemplo de uso de filtro pode ser visto na Figura 20, onde buscamos pelo livro cujo ISBN seja igual a um determinado valor. Aqui sabemos que a consulta poderá ter no máximo 1 resultado, pois a coluna ISBN é uma chave primária, logo não há valores repetidos.

```
SELECT titulo FROM livro WHERE ISBN = '123-1-1234-1234-1';
```

Figura 20: Uso do WHERE para recuperar o título do livro que tenha um determinado ISBN .

Na Figura 21 já procuramos por todos os livros que estejam na quinta edição, como especificado na condição "edicao = 5". Aqui poderemos ter várias linhas como resultado.

```
SELECT ISBN, titulo FROM livro WHERE edicao = 5;
```

Figura 21: Uso do WHERE para recuperar o ISBN e o título dos livros que possuem quinta edição .

Podemos criar filtros mais complexos unindo condições por meio dos operadores AND e do OR (operadores lógicos E e OU). O AND exige que ambas as condições sejam satisfeitas, enquanto que o OU exige que somente uma das condições seja satisfeita para que a linha faça parte da resposta. A Figura 22 mostra um exemplo de uso do operador AND.

```
SELECT titulo FROM livro WHERE edicao >= 2 AND edicao <= 5;
```

Figura 22: Uso do WHERE para recuperar o título dos livros cuja edição está entre 2 e 5.

O operador de igualdade '=' compara se dois valores são exatamente iguais. Entretanto, se quisermos verificar se um texto está contido dentro de outro, devemos utilizar os operadores LIKE ou ILIKE. Como exemplo, a Figura 23 mostra uma consulta que verifica todos os livros que contenham o 'dinossauro' no título. Neste caso, o '%dinossauro %' significa que estamos procurando por dinossauro, e que pode existir qualquer texto antes ou depois de dinossauro. Se trocássemos para '%dinossauro' (perceba que não há % no final), então o título deveria terminado por dinossauro.

```
SELECT * FROM livro WHERE titulo ILIKE '%dinossauro%';
```

Figura 23: Uso do operador ILIKE para recuperar os livros que contenham a palavra dinossauro em qualquer parte do título.

Tabela 2: Operadores comumente utilizados na especificação de filtros em consultas.

Operador	Descrição
=	Verifica se um valor é igual a outro
>	Compara se um valor é maior que outro
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
LIKE	Se um texto está contido em outro
ILIKE	Se um texto está contido em outro, ignorando diferenças de caixa alta e caixa baixa
AND	Operador booleano 'E'
OR	Operador booleano 'OU'
IN	Verifica se um valor está contido em uma sequência de valores
NOT IN	Verifica se um valor não está contido em uma sequência de valores

A Tabela 2 mostra os operadores mais comumente utilizados na escrita de consultas com filtros.

5.3 - Ordenação de valores com o ORDER BY

Vimos que podemos utilizar o SELECT para fazer uma pesquisa e recuperar dados armazenados em uma tabela. Entretanto, as linhas resultantes estarão em uma ordem arbitrária, normalmente na ordem em que foram inseridas. Se quisermos ordenar os resultados podemos utilizar o ORDER BY.

O uso do ORDER BY é simples: adicionamos a palavra-chave ORDER BY, seguida pela coluna que queremos fazer a ordenação, como visto no exemplo da Figura 24.

```
SELECT * FROM livro ORDER BY edicao;
```

Figura 24: Uso do ORDER BY para mostrar os livros ordenados somente pela edição (da menor para a maior edição).

Por padrão, a ordenação será em ordem crescente (ASC), isto é, se estivermos ordenando números, seguirá do menor para o maior, se for um texto, em ordem alfabética,

se for uma data, da mais antiga para a mais recente, e assim por diante. Se quisermos inverter a ordem (ordenar do maior para o menor ou do z para o a), adicionamos DESC (descendente) no final, como visto na Figura 25.

```
SELECT ISBN, titulo FROM livro ORDER BY titulo DESC;
```

Figura 25: Uso do ORDER BY para mostrar os livros em ordem alfabética inversa (do z para o a).

Podemos ordenar os resultados de uma consulta de acordo com uma coluna, e, em caso de empate, desempatar por uma segunda coluna. O exemplo da Figura 26 mostra esta situação: ordenamos primeiro pelo título do livro e, em sequência, pela sua edição.

```
SELECT ISBN, titulo FROM livro ORDER BY titulo, edicao;
```

Figura 26: Uso do ORDER BY para mostrar os livros em ordem alfabética. Em caso de empate, mostra da menor para a maior edição.

5.4 - Exercícios

1. Considerando o sistema de gestão de biblioteca desenvolvido nos capítulos anteriores, escreva consultas para pesquisar:

    a) O título de todos os livros cadastrados no sistema;

    b) Os nomes e e-mails dos usuários da biblioteca, ordenando pelo seu nome em ordem alfabética;

    c) O nome e e-mail das editoras cujo e-mail seja do domínio gmail (simplificando para '@gmail.com');

    d) Todas as colunas dos usuários cujo primeiro nome seja João e que tenha se cadastrado em 2013;

    e) Todas as colunas dos autores que tenham nascido entre 1º de janeiro de 2000 e 25 de março de 2002, ordenando o resultado do autor mais velho para o mais

novo.

2. Considerando o sistema de gestão de universidade desenvolvido, escreva consultas para recuperar:

   a) O nome do departamento dirigido pelo professor de código 92;

   b) O nome dos cursos que iniciaram sua oferta antes do ano de 2000, ordenados alfabeticamente;

   c) O código, nome e e-mail dos docentes cuja área de atuação é “Ciência da Computação”;

   d) Todas as colunas dos discentes cujo nome contenha a palavra “Rambo”, não diferenciando entre maiúsculas e minúsculas.

3. O sistema de um comércio eletrônico possui uma tabela *produto* que foi criada com o código SQL abaixo.

```
CREATE TABLE produto (
    codigo SERIAL PRIMARY KEY,
    nome TEXT NOT NULL,
    quantidade INT NOT NULL,
    preco REAL NOT NULL
);
```

   Escreva consultas para pesquisar:

   a) O nome dos produtos de código 1, 5 e 8;
   b) Todos os produtos da tabela, ordenados do mais barato ao mais caro;
   c) Os produtos que estão em estoque, ordenados pelo nome;
   d) Os produtos que tenham menos de 10 itens em estoque e custem mais do que R$ 50,00;
   e) Os produtos que custam entre R$ 100,00 e R$ 500,00 ou aqueles que possuem estoque maior que 25 itens.

6 - Atualização e remoção

Em um SGBD relacional como o PostgreSQL, os dados inseridos em uma tabela serão guardados de forma persistente, isto é, são armazenadas na memória secundária e sobrevivem mesmo se o computador for reiniciado. Existem comandos que podemos utilizar para modificar as linhas já inseridas nas tabelas, assim como também podemos remover os dados que não queremos mais. Neste capítulo aprenderemos como modificar dados existentes com o UPDATE, e utilizaremos o DELETE e o DROP TABLE para remoção de dados e de tabelas inteiras.

6.1 - Atualização de dados com o UPDATE

O comando UPDATE pode atualizar o valor de uma ou mais colunas para um subconjunto de dados de uma tabela. As modificações são especificadas após a palavra-chave SET, enquanto que utilizamos o WHERE (o mesmo WHERE que vimos quando estudamos o SELECT) para filtrar quais linhas que serão modificadas. Um nota importante: se realizarmos um comando UPDATE sem WHERE iremos modificar TODAS as linhas da tabela. O autor desta apostila agradou o chefe com um desses erros de principiante (acabou configurando o nome de todos os clientes no banco parra 'João'), mas felizmente não chegou a ser demitido.

A Figura 27 mostra um exemplo de uso do update. Nesta atualização, modificamos o título do livro cujo ISBN é igual a '123-1-1234-1234-1' para `'Meu novo título'`. Repare como é o WHERE que delimita quais linhas que serão alteradas.

```
UPDATE livro SET titulo = 'Meu novo título' WHERE ISBN = '123-1-1234-
1234-1';
```

Figura 27: Uso do UPDATE para atualizar o título de um livro específico.

Na Figura 28 podemos ver um comando que altera o autor para 1 e a edição para 3 para todos os livros que contenham a palavra 'mistério' no título. Neste exemplo vemos como é possível fazer alterações em múltiplas colunas simultaneamente, além de estarmos modificando várias linhas em um mesmo comando.

```
UPDATE livro SET autor = 1, edicao = 3 WHERE titulo ILIKE '%mistério
%';
```

Figura 28: Uso do UPDATE para configurar o autor de todos os livros cujo nome contém mistério no título.

6.2 - Remoção de dados com o DELETE

Para removermos linhas de uma tabela utilizamos o comando DELETE. Sua sintaxe é simples: especificamos a tabela que será afetada e qual a condição para que a linha seja removida. Aqui temos o mesmo perigo do UPDATE, isto é, se não especificarmos um filtro com o WHERE, removeremos todas as linhas da tabela.

A Figura 29 mostra como utilizar o delete para remover todos os livros cuja edição seja menor ou igual a 2. Já a Figura 30 mostra como remover todos os livros da tabela.

```
DELETE FROM livro WHERE edicao <= 2;
```

Figura 29: Remove todas as 1ªs e 2ªs edições de livros da tabela.

```
DELETE FROM livro;
```

Figura 30: Remove todos os livros, mas mantém a tabela.

Se quisermos remover a tabela em si, e não somente o seu conteúdo, podemos utilizar o comando DROP TABLE como visto na Figura 31.

```
DROP TABLE livro;
```

Figura 31: Remove a tabela livro e todo o seu conteúdo.

6.3 - Exercícios

1) Desenvolva um comando que remova todos os livros que estão na quinta edição ou superior;

2) Escreva um comando que remova os autores que nasceram depois do dia 02 de abril de 2002;

3) Considere a tabela produto da seção de exercícios do capítulo anterior e escreva comandos para:

a) Atualizar a quantidade do produto de código 102 para 20 unidades;
b) Remover os produtos cujo valor seja maior do que R$ 1.000,00;
c) Configurar a quantidade de todos os produtos que custem entre R$ 20,00 e R$ 30,00 para 50;
d) Dobrar o preço de todos os produtos.

4) Escreva um comando que remova todas as tabelas criadas para o sistema de gestão de universidade. Não se esqueça de seguir a ordem correta: o inverso da ordem de criação (lembre-se que não podemos remover uma tabela que ainda está sendo referenciada por uma chave estrangeira);

7 - Conclusão

Esta apostila apresentou os conceitos básico de bancos de dados relacionais e da linguagem SQL por meio do PostgreSQL. Muita coisa foi deixada de fora ou simplificada por motivos didáticos.

Apesar de existirem outros paradigmas de banco de dados sendo utilizados hoje em dia (como os orientados a documentos e a chave/valor), os bancos relacionais que usam SQL ainda são amplamente utilizados e continuam sendo os mais adequados para grande parte das demandas. Se você se interessou pelo assunto, recomendo a leitura da documentação oficial[3] ou um bom tutorial[4].

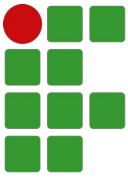
Espero que tenha gostado do curso, lembrando que esta apostila foi concebida para uso simultâneo com aulas presenciais ou remotas. Até a próxima!

Figura 32: Futuras instalações do "FIMS" JD.

3 https://www.postgresql.org/docs/

4 https://www.postgresqltutorial.com/

Anotações



INSTITUTO FEDERAL
Mato Grosso do Sul
Campus Jardim