

Banco de dados



Sistema de banco de dados



— Definição de banco de dados

- Banco de dados: “é uma coleção de dados relacionados” (ELMASRI; NAVATHE, 2019).
- Origem: termo "**database**" em inglês.
- Dados significam fatos registrados com significado implícito.

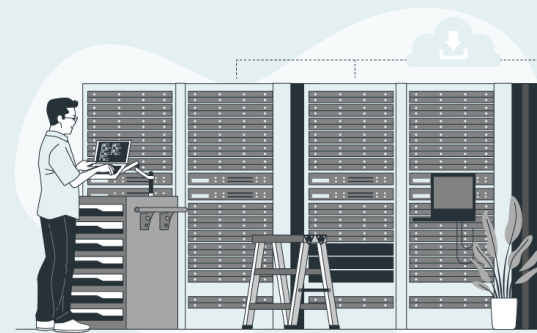


— Definição de banco de dados

- Banco de dados como componente central em **sistemas de informação**.
- Evolução histórica: tecnologia molda implementação.



Antes do computador



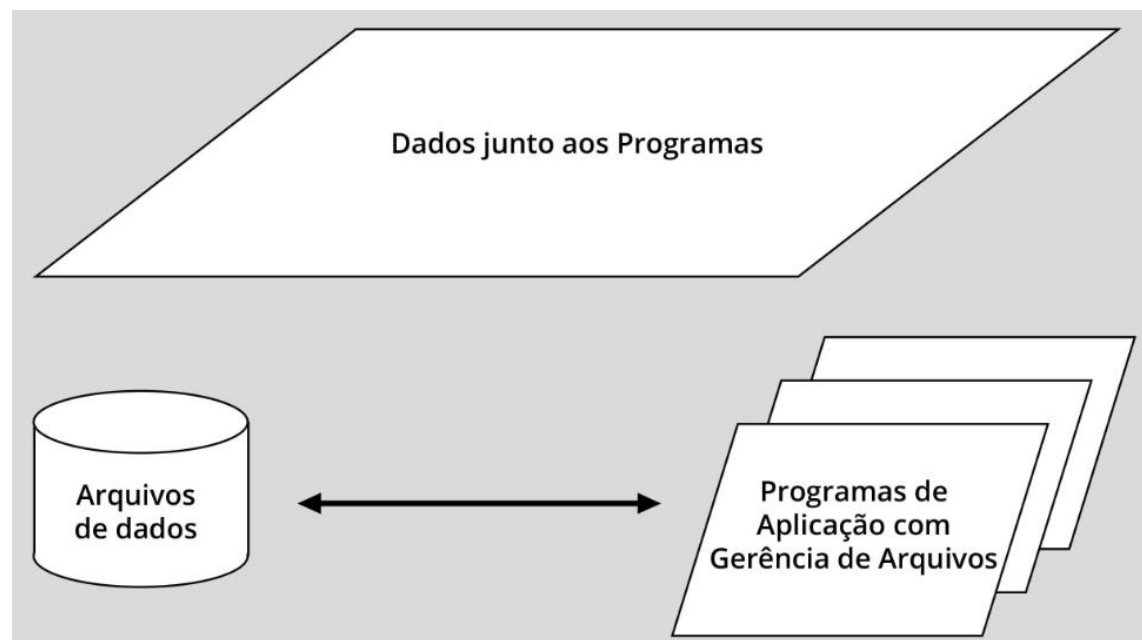
Atualmente

— Evolução dos sistemas de informação em computador

- **Após a Segunda Guerra Mundial:** foi utilizado para cálculos matemáticos complexos.
- **Arquitetura de John von Neumann:** incorpora uma unidade central de processamento capaz de armazenar programas e dados.
- **Advento do disco magnético pela IBM em 1957:** capacidade de leitura direta de dados externos à unidade central de processamento, eliminando a necessidade de leitura sequencial.
- **Centro de Processamento de Dados (CPD):** manipulação de dados armazenados em arquivos hospedados em discos magnéticos, formando o sistema de arquivos utilizado pelo sistema operacional.

— Evolução dos sistemas de informação em computador

Execução em lotes (**batch**):

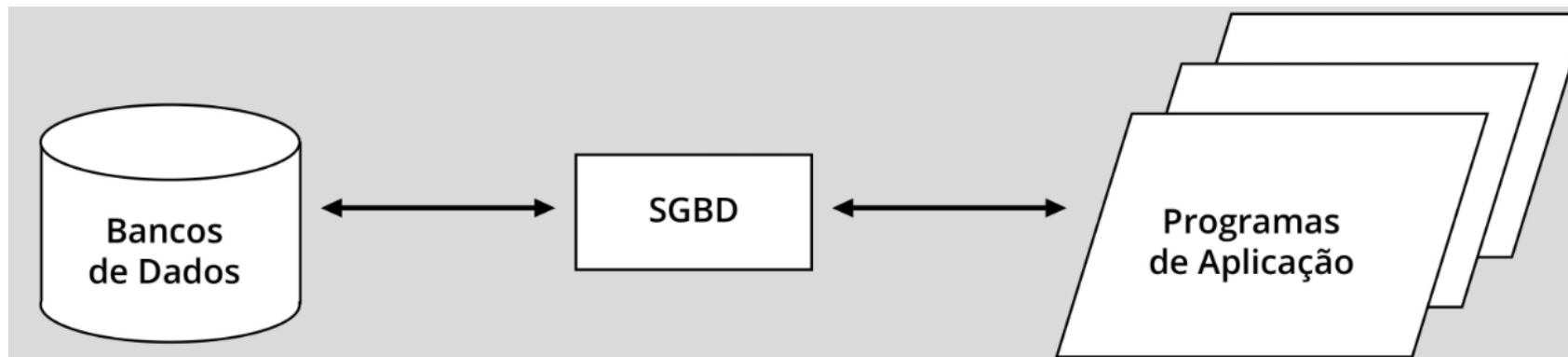


— Evolução dos sistemas de informação em computador

- Modelo de processamento de dados com sistema de arquivos foi amplamente usado em sistemas empresariais no início do uso do computador e persiste em sistemas legados.
- Linguagem COBOL - comum em aplicações empresariais.

— Evolução dos sistemas de informação em computador

Independência entre dados e programas:



— Evolução dos sistemas de informação em computador

- Modularização do sistema: responsabilidades divididas entre programas de aplicação e SGBD.
- SGBD gerencia acesso e manipulação de dados em disco.
- Diferença entre Sistema de Banco de Dados (SBD) e SGBD.
- Eficiência comparativa entre modelo monolítico, sistemas de arquivos e sistemas de bancos de dados.

— Evolução dos sistemas de informação em computador

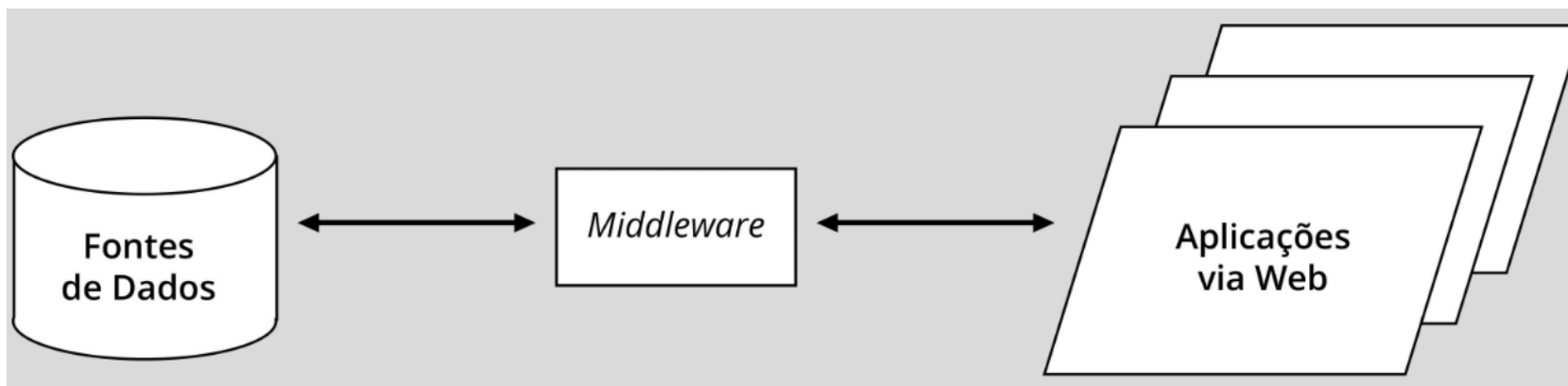
Estágio atual dos sistemas de informação (na Web)

- Revolução tecnológica causada pela **World Wide Web** no final do século XX.
- Surgimento de novas linguagens de programação e novas formas de armazenamento e acesso a dados em fontes com diferentes formatos.
- SGBD - pode ser considerado atualmente como um gênero de software **básico**.

— Evolução dos sistemas de informação em computador

Estágio atual dos sistemas de informação (na Web)

Middleware, em que se incluem servidores de aplicações das diferentes linguagens e ambientes de desenvolvimento Web.



— Evolução dos sistemas de informação em computador

Estágio atual dos sistemas de informação (na Web)

- **Web atual:** fontes de dados abrangem volumes gigantescos em diversos formatos e localizações.
- **Big Data:** conjuntos de dados extremamente grandes e complexos que não podem ser facilmente processados com métodos de processamento de dados tradicionais.
- Diversidade de plataformas digitais, que têm em comum a conexão com a internet e a computação em nuvem (**Cloud Computing**).

— Evolução dos sistemas de banco de dados

Bancos de dados navegacionais

Duas iniciativas independentes, resultaram em dois produtos comerciais:

IDS (*Integrated Data System*)

Monitora atividades de rede ou sistemas em busca de comportamentos maliciosos, alertando sobre possíveis intrusões por meio de padrões ou anomalias.

IMS (*Information Management Systems*)

Sistema de banco de dados hierárquico desenvolvido pela IBM, organizando dados em uma estrutura de árvore para processamento de transações em ambientes empresariais.

— Evolução dos sistemas de banco de dados

Bancos de dados navegacionais

Principais diferenças:

IDS

Usava a estrutura de dados de grafos ou redes, daí a denominação de ***network databases***.



IMS

Adotava a estrutura de dados de árvores, que é um tipo de grafo mais restrito do que as redes, baseado em hierarquias, originando a denominação ***hierarchical databases***.

— Evolução dos sistemas de banco de dados

O modelo relacional de banco de dados

- Revolução nos bancos de dados nas décadas de 1960 e 1970 com o modelo relacional de Edgar Codd.
- Artigo seminal introduziu estrutura baseada em relações matemáticas.
- Álgebra e Cálculo Relacionais fundamentaram a teoria matemática do modelo.

— Evolução dos sistemas de banco de dados

Principais SGBDs relacionais

Incluem:

- DB2;
- Oracle Corporation;
- MySQL;
- BSD;
- SQL Server;
- Postgres.

— Evolução dos sistemas de banco de dados

Outros modelos de SGBDs

- No ranking de popularidade dos SGBDs, destacam-se, entre os que adotam o modelo relacional:
 - Oracle;
 - MySQL;
 - Microsoft SQL Server;
 - PostgreSQL;
 - IBM DB2.

— Evolução dos sistemas de banco de dados

Outros modelos de SGBDs

- “Multimodelos” implementam funcionalidades que vão além do modelo relacional.
 - **ORACLE** (documentos, grafos e RDF);
 - **MYSQL** (documentos);
 - **MICROSOFT SQL SERVER** (documentos e grafos);
 - **POSTGRESQL** (documentos);
 - **IBM DB2** (documentos e RDF).

— Evolução dos sistemas de banco de dados

E o que são esses outros modelos de banco de dados, além do relacional?

- O modelo relacional - amplamente utilizado em sistemas empresariais devido à sua popularidade, robustez e padronização, usando a linguagem SQL para consulta e manipulação de dados.
- Aplicações modernas (Web, ciência de dados e Big Data) demandam recursos que vão além das capacidades das tabelas do modelo relacional.
- Surgiram os bancos de dados NoSQL, uma alternativa que não adota o modelo relacional, não utiliza exclusivamente a linguagem SQL e atende às necessidades de armazenamento e processamento de grandes volumes de dados em formatos diversos.

— Diferenças entre sistema de arquivos e sistema de banco de dados

- Independência de dados: deriva da arquitetura de três esquemas no banco de dados.
- Arquitetura ANSI/SPARC apresenta três níveis (externo, conceitual e interno).

Externo

Integra visões dos usuários.

Conceitual

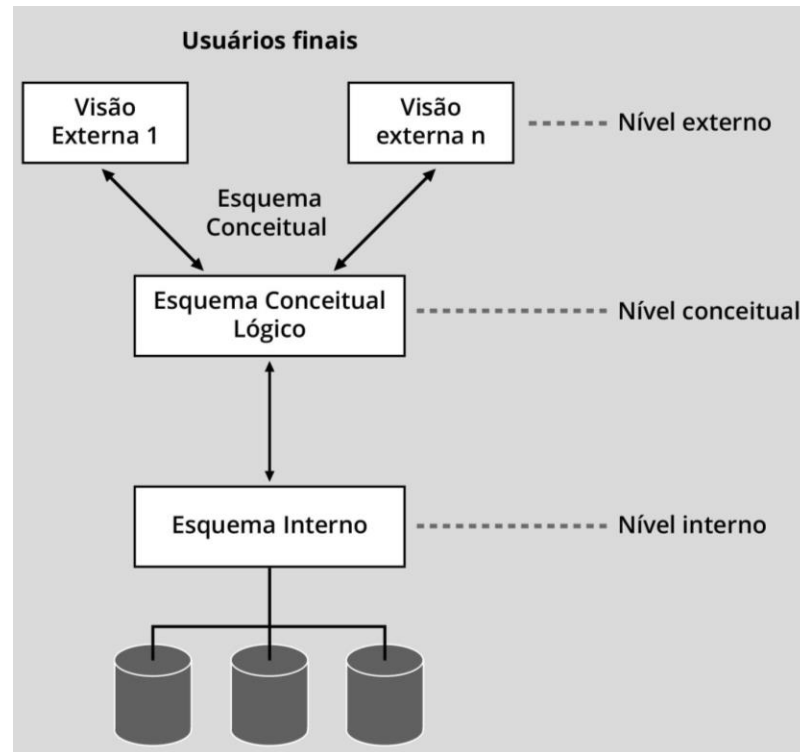
Estrutura lógica.

Interno

Estrutura física no disco.

— Diferenças entre sistema de arquivos e sistema de banco de dados

Separação entre aplicações e usuários finais do banco de dados físico:



— Diferenças entre sistema de arquivos e sistema de banco de dados

Dois tipos de independência de dados:

Independência lógica de dados

Modificar esquema conceitual lógico sem impactar visões externas.

Independência física de dados

Alterar esquema interno (reorganização física) sem afetar esquema lógico e externo.

— Diferenças entre sistema de arquivos e sistema de banco de dados

- **Natureza autocontida:** SBD inclui descrição completa das estruturas e restrições (metadados no catálogo).
- **Abstração de dados:** representação conceitual oculta detalhes de armazenamento, suportando múltiplas visões lógicas e independência de dados.
- **Suporte ao compartilhamento e processamento simultâneo:** permite acesso simultâneo de múltiplos usuários e processamento de transações concorrentes.

— Diferenças entre sistema de arquivos e sistema de banco de dados

- **Modelos físicos:** descrevem como os dados são armazenados no computador, incluindo tipos de arquivos e caminhos de acesso.
- **Modelos lógicos:** representam abstratamente a implementação dos bancos de dados, como o modelo relacional de Edgar Codd.
- **Modelos conceituais:** refletem a visão do usuário final, como o modelo ER de Peter Chen, usado na modelagem conceitual de dados.

— Vantagens e desvantagens da abordagem de banco de dados

Funcionalidades que conferem vantagens adicionais em relação a sistemas sem banco de dados:

- Controle de redundância;
- Compartilhamento de dados;
- Múltiplas interfaces;
- Cumprimento de restrições de integridade;
- Backup e recuperação de dados;
- Compartilhamento de dados simultâneo.

— Vantagens e desvantagens da abordagem de banco de dados

É responsabilidade do SGBD garantir as propriedades das transações, conhecidas pela sigla:

- **Atomicidade** (*Atomicity*);
- **Consistência** (*Consistency*);
- **Isolamento** (*Isolation*);
- **Durabilidade** (*Durability*).

— Vantagens e desvantagens da abordagem de banco de dados

Vantagens

- Estabelecimento de padrões de uso dos dados na organização.
- Redução do tempo de desenvolvimento de aplicações.
- Flexibilidade na manutenção dos dados.
- Disponibilidade de dados atualizados em toda a organização.
- Economia de escala, entre outras.



Desvantagens

- Sobrecarga no desempenho do sistema devido à presença do SGBD.
- Custo e esforço adicionais na capacitação e implementação de funcionalidades sofisticadas.

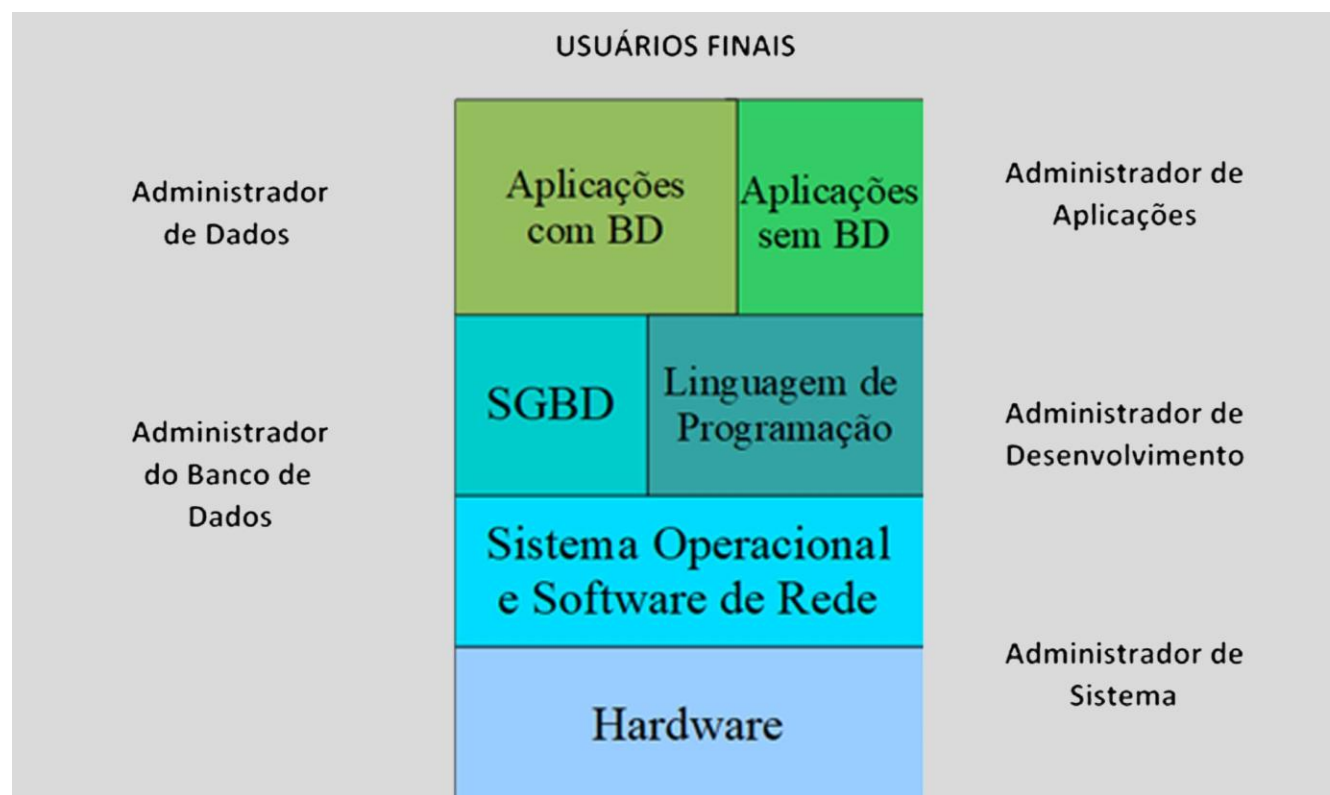
— Vantagens e desvantagens da abordagem de banco de dados

Quando a abordagem de SBD não é indicada:

- Aplicações simples com dados estáticos e bem definidos.
- Aplicações de tempo real com requisitos rígidos incompatíveis com o overhead do SGBD.
- Sistemas embarcados com poucos dados e requisitos estritos de tempo real.
- Sistemas monousuários sem concorrência, típicos de aplicações em desktop.
- Aplicações que exigem dados volumosos e dinâmicos, como IoT, podem ser incompatíveis com SGBDs tradicionais, mas NoSQL busca atender a essas demandas.

— Papéis em sistemas de bancos de dados

Camadas de um sistema de computação.



— Papéis em sistemas de bancos de dados

Diferentes papéis de um sistema de computação corporativo de grandes organizações:

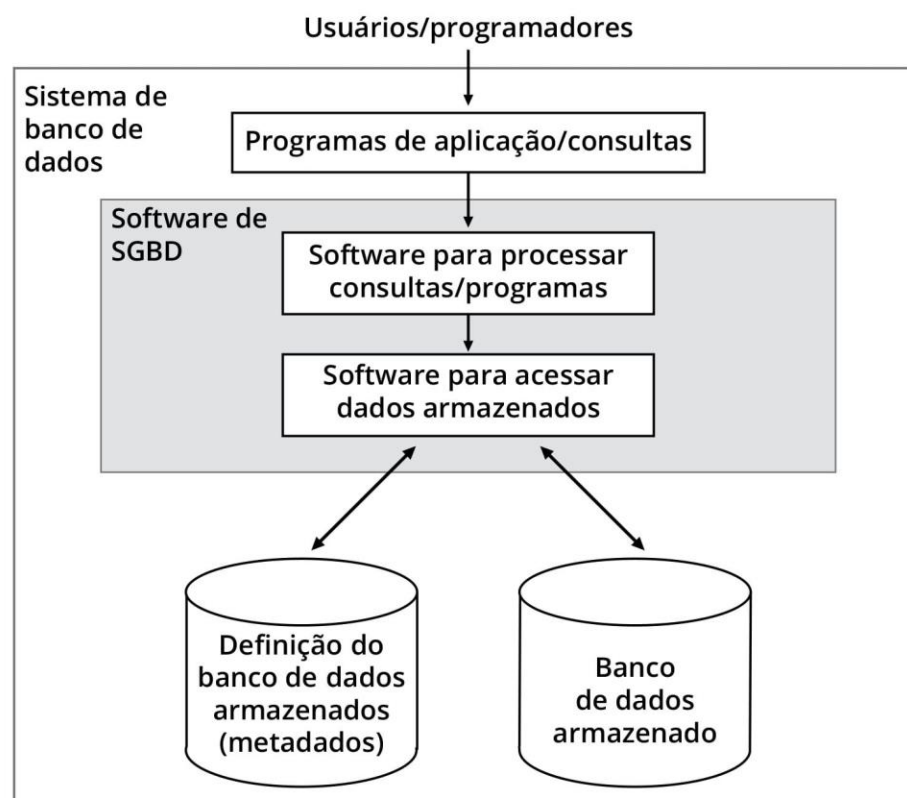
- **Usuários Finais:** beneficiários dos sistemas, independentemente da abordagem de banco de dados.
- **Administrador de Aplicações:** gerencia e suporta sistemas de aplicação, como o administrador de ERP.
- **Administrador de Desenvolvimento:** supervisiona equipes de desenvolvimento de sistemas, utilizando ferramentas disponíveis.
- **Administrador de Dados:** define políticas e regras para o ambiente de dados, orientando o desenvolvimento.
- **Administrador do Banco de Dados:** cria e mantém bancos de dados no SGBD, apoiando equipes de desenvolvimento.
- **Administrador de Sistema:** mantém o sistema de computação, focando em hardware, sistema operacional e interfaces.

— Componentes de um sistema de banco de dados

- Ambiente de banco de dados: programas de aplicação e consultas acessam dados e metadados via SGBD.
- SGBD consiste em software para processar consultas e acessar dados/metadados armazenados em disco.
- Diferença entre SBD e SGBD: SGBD é componente do SBD, incluindo programas, consultas, dados e metadados.
- Fronteira do SBD abrange programas de aplicação, consultas, metadados no catálogo e conteúdo no banco de dados.

— Componentes de um sistema de banco de dados

Visão simples do ambiente de banco de dados:



— Componentes de um sistema de banco de dados

- Metadados definidos como esquema do banco de dados, seguindo o modelo lógico de implementação.
- Modelar conforme o modelo lógico significa esquematizar com os construtores desse modelo.
- Exemplo no modelo relacional: esquema composto por **tabelas** e **colunas**.
- Comandos de definição de dados alteram o esquema ao criar, alterar ou remover tabelas.
- Estado ou instância: conteúdo do banco de dados em um momento específico.
- Manipulações com comandos de inserção, atualização ou remoção provocam mudança de estado.
- Cada ação gera uma nova instância do banco de dados.
- Estado reflete o conteúdo atual do banco de dados.

— Módulos de um sistema de gerência de banco de dados

Módulos que compõem um SGBD:

Parte superior

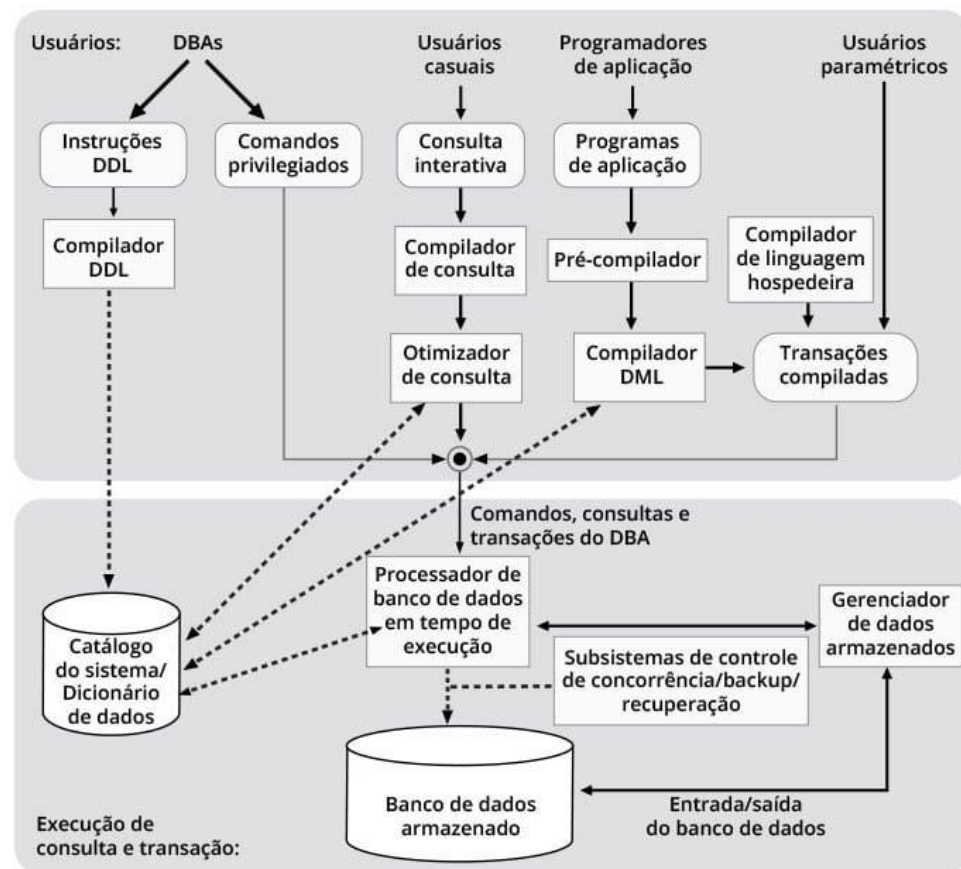
Correspondente ao processamento das consultas e aplicações.

Parte inferior

Corresponde ao acesso a metadados e dados armazenados (a base de dados).

— Módulos de um sistema de gerência de banco de dados

Módulos que compõem um SGBD:



— Um exemplo de SGBD relacional: PostgreSQL

- PostgreSQL originou-se do projeto Ingres na década de 1970, evoluindo de Postgres para PostgreSQL.
- Inicialmente, usava QueL em vez de SQL, competindo com a linguagem da IBM.
- SQL tornou-se padrão, e PostgreSQL implementou a linguagem.
- Reconhecido como banco de dados relacional avançado, com documentação globalmente referenciada.
- Amplamente utilizado no ensino e comercialmente por sua licença livre e código aberto.

— Um exemplo de SGBD relacional: PostgreSQL

O PostgreSQL é um SGBD relacional-objeto, combinando a estrutura de dados de tabelas com extensões orientadas a objetos. Ele suporta o padrão SQL e implementa recursos como consultas complexas, gatilhos, visões materializadas atualizáveis e controle de concorrência multiversionado, permitindo extensões personalizadas, como novos tipos de dados, funções e linguagens procedurais.

— Um exemplo de SGBD relacional: PostgreSQL

- PostgreSQL usa modelo cliente/servidor com back end e front end cooperativos.

Back end (postgres)

Gerencia arquivos, aceita conexões e executa ações no SBD.

Front end

Aplicações clientes diversas (psql, pgadmin) para operar no banco de dados.

- Sessão PostgreSQL composta por interação entre back end e front end.

— Um exemplo de SGBD relacional: PostgreSQL

Recursos avançados do PostgreSQL

- Controle de concorrência multiversão (MVCC), PiTR, tablespaces.
- Replicação assíncrona, transações aninhadas, online/hot backups.
- Planejador/otimizador de consultas, write ahead logging (WAL).

— Um exemplo de SGBD relacional: PostgreSQL

Características avançadas do PostgreSQL

Herança de Tabelas

- Adiciona sabor orientado a objetos, permite derivar tabelas como classes base.
- Suporta herança simples e múltipla.

Sistema de Regras

- Sistema de reescrita de consultas.
- Permite criar regras para transformar operações específicas dinamicamente.

Sistema de Eventos

- Comunicação interprocessos via LISTEN e NOTIFY.
- Facilita peer-to-peer e coordenações avançadas entre eventos de banco de dados.

Projeto de banco de dados: modelagem conceitual



— Projeto de banco de dados

Fases do projeto de banco de dados:



Levantamento de Requisitos



Projeto Conceitual



Projeto Lógico



Projeto Físico

— Levantamento de requisitos

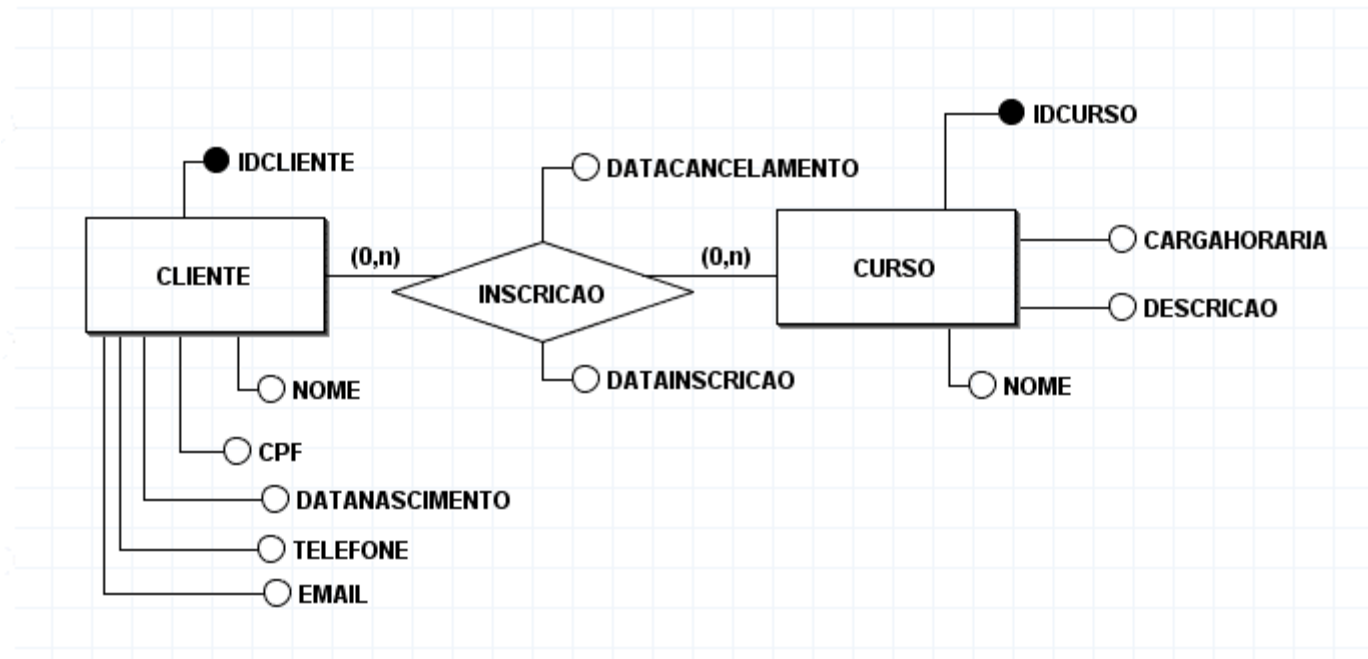
- Entrevistas com usuários para entender o negócio.
- Documentação completa e detalhada dos requisitos de dados.
- Transição para a próxima fase: projeto conceitual.

— Projeto conceitual

- Construção de modelo de dados de alto nível.
- Baseado nos requisitos obtidos no levantamento.
- Utiliza Diagrama de Entidade e Relacionamento (DER).
- Elementos: **entidades, relacionamentos e atributos**.
- Foco na estrutura, não na implementação física.
- Representação gráfica visual com DER.

— Projeto conceitual

DER construído a partir dos requisitos de dados obtidos na etapa de levantamento de requisitos.



DER construído a partir dos requisitos de dados da Escola.

— Projeto conceitual

- Modelo inclui duas entidades: **CLIENTE** e **CURSO**.
- **CLIENTE** armazena dados de clientes, **CURSO** armazena dados de cursos.
- Relacionamento estabelecido: cliente pode se inscrever em um ou mais cursos.
- Estrutura básica do modelo conceitual identificada no DER.

— Projeto conceitual

“

O esquema conceitual de alto nível (DER) pode ser utilizado como uma referência para garantir que todos os requisitos de dados dos usuários sejam atendidos e que não estejam em conflito.

(ELMASRI; NAVATHE, 2019)

— Projeto conceitual

- Facilita aprendizado preciso sobre o funcionamento do negócio.
- Comparado aos requisitos de dados, DER oferece visualização mais clara.
- Permite compreender relações entre entidades e suas características.
- Melhora a compreensão da estrutura e dinâmica dos dados no contexto do negócio modelado.

— Projeto conceitual

Outra notação para DER: diagrama de classes UML

- Falta de uma notação-padrão; depende de preferências ou regras da empresa.
- Ferramentas CASE adotam diversas notações; brModelo usa uma próxima à original.
- Uso comum da UML em projetos de software para representar conceitos do DER.

— Projeto conceitual

Outra notação para DER: diagrama de classes UML

Classes do diagrama UML:

Superior

Nome da classe.

Central

Atributos, opcionalmente com
tipo de dados.

Inferior

Operações associadas aos
objetos da classe.

— Projeto conceitual

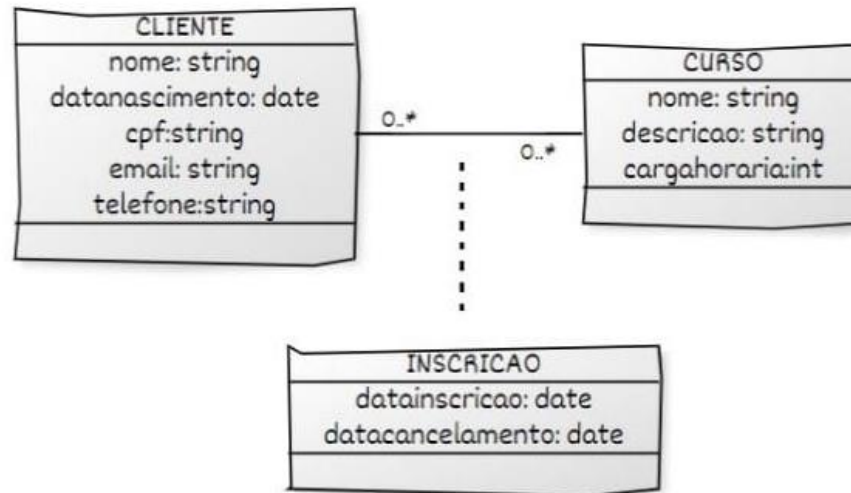
Outra notação para DER: diagrama de classes UML

- Na UML, relacionamento entre classes é chamado de **associação**.
- Representado por linha conectando as classes participantes.
- Atributos dos relacionamentos em caixa conectada por linha tracejada.

— Projeto conceitual

Outra notação para DER: diagrama de classes UML

Exibição do DER como diagrama de classes UML:



Esquema conceitual Escola na notação do diagrama de classes UML.

— Projeto conceitual

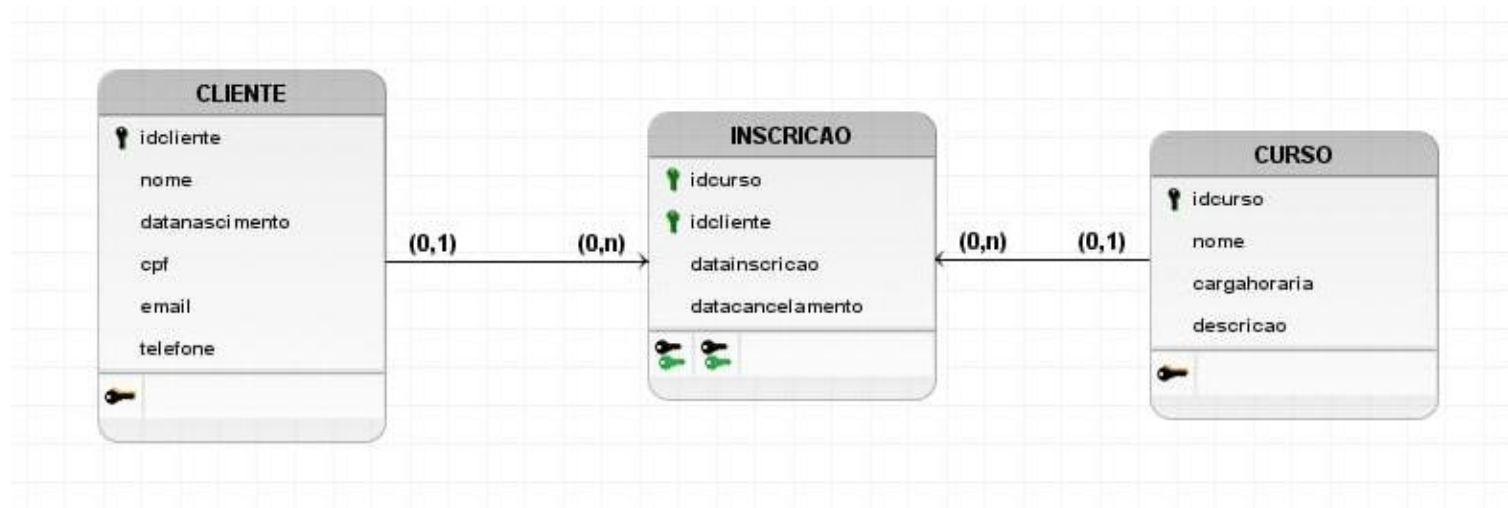
Outra notação para DER: diagrama de classes UML

- Após levantamento e DER, próxima etapa é o projeto lógico.
- Construção de modelo de dados mais detalhado.
- Depende da escolha do SGBD (Sistema Gerenciador de Banco de Dados).
- Etapa crucial antes da implementação real do banco de dados.

— Projeto lógico

- Transforma modelo conceitual em modelo lógico, dependendo do SGBD escolhido.
- Diversos modelos lógicos (rede, hierárquico, relacional, orientado a objeto, grafos, chave-valor, XML).
- Modelo relacional é o mais **popular**.
- SGBD que fazem uso do modelo relacional: Oracle, MySQL, PostgreSQL, SQLite, SQL Server.
- Conversão DER para modelo lógico relacional envolve regras específicas.
- Tabelas representam dados em estrutura relacional.
- Ferramentas de modelagem auxiliam, mas é crucial entender os princípios da conversão.

— Projeto lógico



Tabelas originadas do DER da Escola.

— Projeto lógico

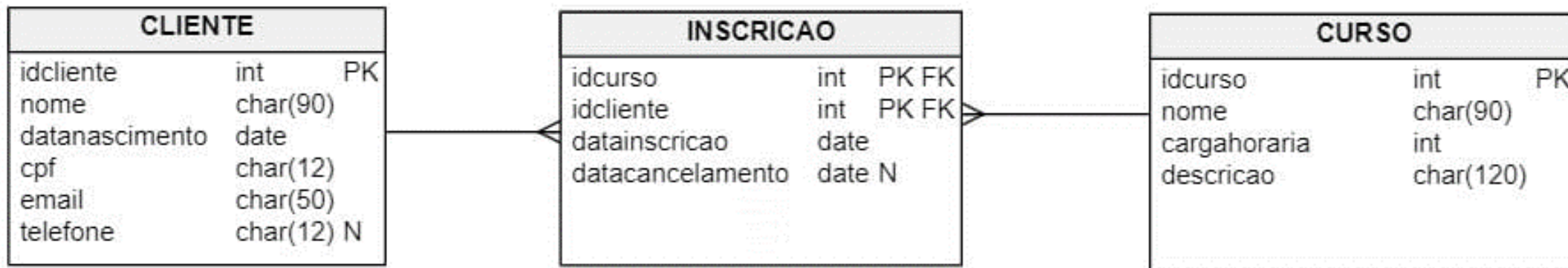
- Entidades do DER representadas como tabelas no modelo relacional.
- Exemplo: tabelas **CLIENTE**, **CURSO** e **INSCRICAO**.
- Atributos vinculados, mas características como tipos de dados não definidas.
- Representação textual das tabelas é possível além da visual.
- Projeto ainda não inclui detalhes como tipos e tamanhos de dados.

— Projeto físico

- Detalhes de implementação são definidos para objetos do banco de dados.
- Escolha de tipos de dados, tamanho de colunas, obrigatoriedade/opcionalidade.
- Relacionamentos definidos por chaves estrangeiras.
- Projeto físico muitas vezes feito com ferramentas gráficas de modelagem.
- Uso da ferramenta Vertabelo para modelagem PostgreSQL.

— Projeto físico

Modelo físico enriquecido com detalhes de implementação compatíveis com o SGBD escolhido.



Modelo físico do estudo de caso Escola.

— Projeto físico

- Cada coluna especificada com detalhes como tipo de dados e restrições (FK, PK, N).
- Diferente do modelo lógico, aqui detalhes são específicos.
- Uso de linguagem declarativa SQL na criação do esquema.
- Parte da SQL responsável por isso é a Linguagem de Definição de Dados (DDL).

— Projeto físico

Script DDL SQL compatível com o modelo escola:

```
1 CREATE TABLE CLIENTE (  
2     idcliente int NOT NULL,  
3     nome char(90) NOT NULL,  
4     datanascimento date NOT NULL,  
5     CPF char(12) NOT NULL,  
6     email char(50) NOT NULL,  
7     telefone char(12) NULL,  
8     PRIMARY KEY (idcliente) );  
9 CREATE TABLE CURSO (  
10    idcurso int NOT NULL,  
11    nome char(90) NOT NULL,  
12    cargahoraria int NOT NULL,  
13    descricao char(120) NOT NULL,  
14    PRIMARY KEY (idcurso) );  
15 CREATE TABLE INSCRICAO (  
16    idcurso int NOT NULL,  
17    idcliente int NOT NULL,  
18    datainscricao date NOT NULL,  
19    datacancelamento date NULL,  
20    PRIMARY KEY (idcurso,idcliente),  
21    FOREIGN KEY (idcliente) REFERENCES CLIENTE (idcliente),  
22    FOREIGN KEY (idcurso) REFERENCES CURSO (idcurso) );
```

— Projeto físico

Propósito script SQL DDL:

- Cada tabela iniciada com **CREATE TABLE** <<nometabela>>.
- Identificadores únicos para **CLIENTE** (idcliente) e **CURSO** (idcurso).
- Identificador único composto para **INSCRICAO** (idcurso, idcliente).
- Restrições (linhas 21 e 22) garantem referência a clientes e cursos existentes.

— Entidade

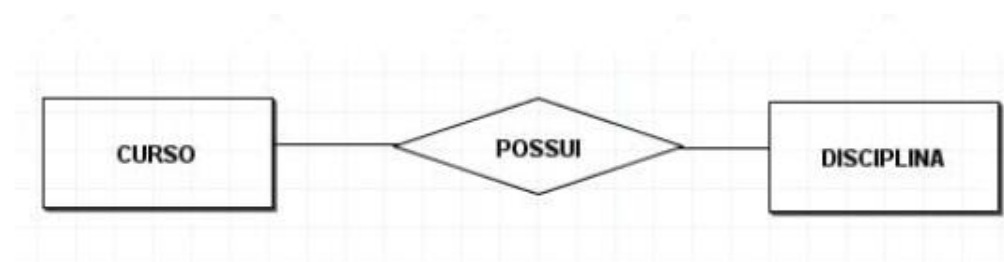
- Segundo Heuser (2009), entidade representa conjunto de objetos da realidade a ser mantido no banco de dados.
- No DER, entidade é representada por um retângulo com nome definido dentro dele.
- Representação de entidade em DER.



Exemplo de representação gráfica de entidade.

— Relacionamento

- Segundo Heuser (2009), relacionamento especifica associações entre objetos.
- No DER, relacionamento é representado por losango ligado por linhas conectadas a entidades.
- Representação de relacionamento em DER.



Exemplo de representação gráfica de relacionamento.

— Relacionamento

Autorrelacionamento

- Relacionamento envolve ocorrências de uma mesma entidade.
- Diferenciar o papel que cada ocorrência da entidade cumpre no contexto do relacionamento em questão.

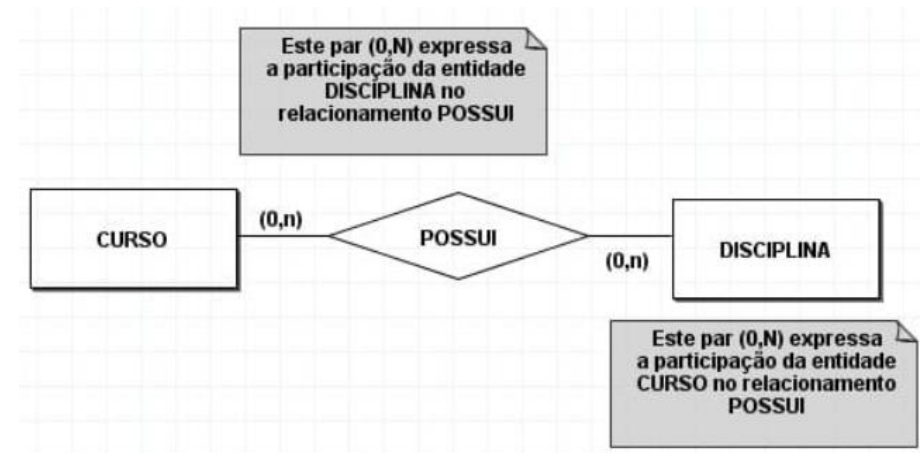


Exemplo de representação gráfica de autorrelacionamento.

— Relacionamento

Cardinalidade de relacionamentos

- Perguntas sobre relacionamento CURSO e DISCIPLINA abordadas com cardinalidade.
- Respostas expressas como pares ordenados (mínima, máxima).
- Representação da cardinalidade no DER.

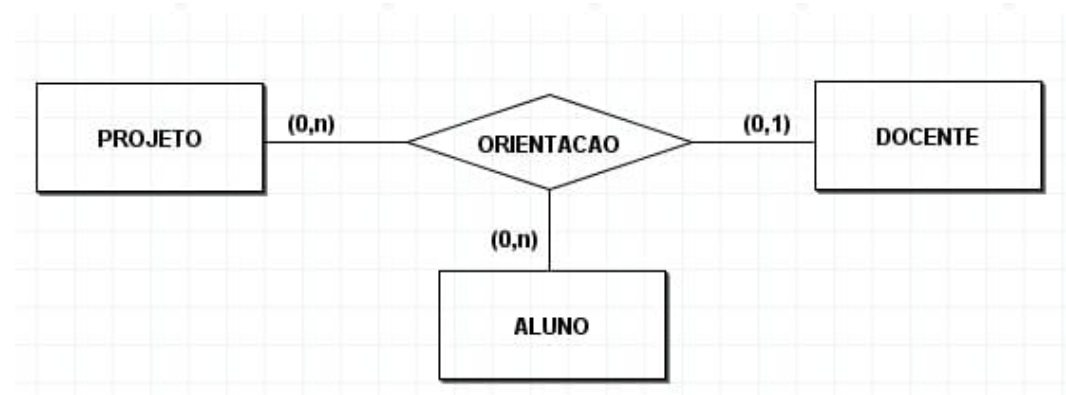


Exemplo de representação gráfica do relacionamento POSSUI, com as cardinalidades definidas.

— Relacionamento

Relacionamento ternário

- Modelagem de orientações de alunos em projetos por docentes.
- Relacionamento ternário ORIENTACAO envolvendo entidades PROJETO, ALUNO e DOCENTE.
- Cardinalidade expressa em pares ordenados (mínima, máxima).



Exemplo de relacionamento ternário.

— Relacionamento

Relacionamento ternário

Detalhes de cada par de cardinalidade:

Cardinalidade máxima 1

Aluno pode ser orientado por no máximo um docente em um projeto.

Cardinalidade máxima N

Docente pode orientar vários alunos em um projeto.

Cardinalidade máxima N

Aluno e docente podem participar de vários projetos.

— Atributos

“

Atributo corresponde a um dado que é associado a cada ocorrência de uma entidade ou de um relacionamento.

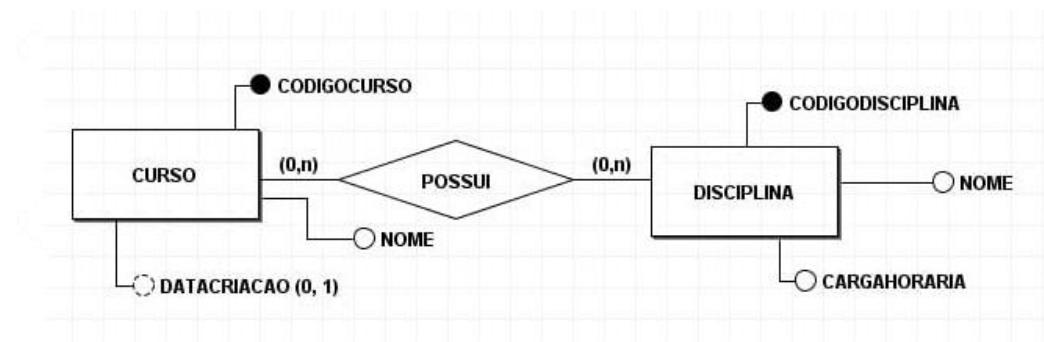
(HEUSER, 2009)

— Atributos

Propriedades para as entidades CURSO e DISCIPLINA:

- Todo **curso** possui um código único, nome e, opcionalmente, data de criação.
- Toda **disciplina** possui um código único, nome e carga horária.

Parte do DER contemplando esses requisitos de dados.



DER contemplando atributos das entidades CURSO e DISCIPLINA.

— Atributos

Cardinalidade em atributo

- No DER, ao lado do atributo DATACRIACAO, há cardinalidade (0,1).
- Significado especial da cardinalidade em atributos:

Cardinalidade Mínima

- 0;
- 0;
- 1;
- 1.

Cardinalidade Máxima

- 1;
- N;
- 1;
- N.

Significado

- Opcional, Monovalorado;
- Opcional, Multivalorado;
- Obrigatório, Monovalorado;
- Obrigatório, Multivalorado.

1. Sistema de Gerenciamento de Biblioteca

Descrição: Gerencia livros, seus exemplares, usuários da biblioteca e empréstimos.

2. Sistema de Reservas de Hotéis

Descrição: Gerencia hóspedes, reservas, quartos e serviços oferecidos.

3. Sistema de Gerenciamento de Consultas Médicas

Descrição: Gerencia pacientes, médicos, consultas e pagamentos.

Tabelas e Relacionamentos:

- Usuário (ID_Usuário, Nome, Email, Telefone)
- Livro (ID_Livro, Título, Autor, ISBN)
- Exemplar (ID_Exemplar, ID_Livro_FK, Localização, Status)
- ID_Livro_FK é chave estrangeira que faz referência à tabela Livro.
- Empréstimo (ID_Empréstimo, ID_Usuário_FK, ID_Exemplar_FK, Data_Empréstimo, Data_Devolução)
- ID_Usuário_FK é chave estrangeira que faz referência à tabela Usuário.

Tabelas e Relacionamentos:

- Hóspede (ID_Hóspede, Nome, Documento, Telefone)
- Reserva (ID_Reserva, ID_Hóspede_FK, Data_Início, Data_Fim)
 - ID_Hóspede_FK é chave estrangeira que faz referência à tabela Hóspede.
- Quarto (ID_Quarto, Tipo, Preço_Diária, Status)
- Serviço (ID_Serviço, Nome_Serviço, Preço)
- Reserva_Serviço (ID_Reserva_FK, ID_Serviço_FK)
 - ID_Reserva_FK é chave estrangeira que faz referência à

Tabelas e Relacionamentos:

- Paciente (ID_Paciente, Nome, Data_Nascimento, Telefone)
- Médico (ID_Médico, Nome, Especialidade, CRM)
- Consulta (ID_Consulta, ID_Paciente_FK, ID_Médico_FK, Data, Diagnóstico)
 - ID_Paciente_FK é chave estrangeira que faz referência à tabela Paciente.
 - ID_Médico_FK é chave estrangeira que faz referência à tabela Médico.

— Atributos

Atributo obrigatório e monovalorado

- Na construção de DER, a maioria dos atributos é monovalorado e **obrigatório**.
- Cardinalidade (1,1) não é expressa para atributos monovalorados e obrigatórios por questões de legibilidade.

— Atributos

Atributo composto

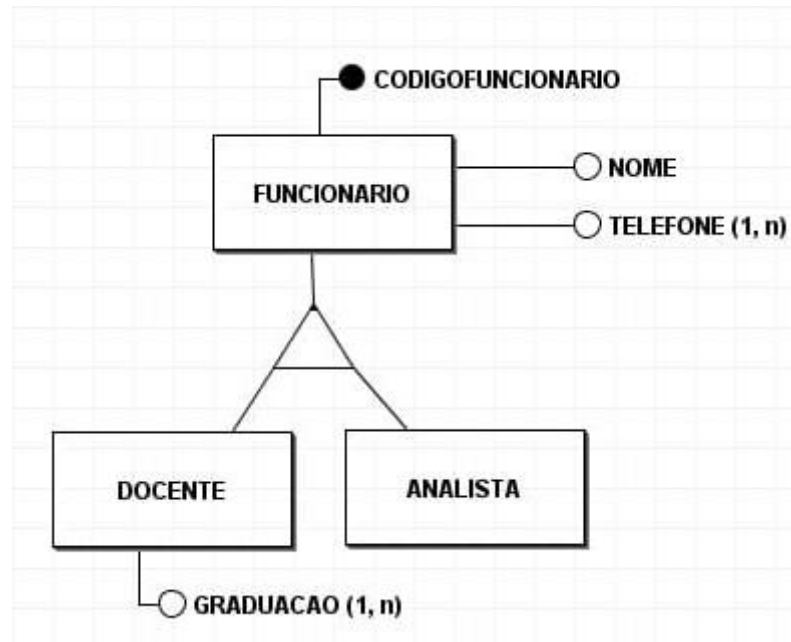
- São atributos complexos, subdivididos em partes menores.
- Atributo endereço pode ser subdividido em **logradouro**, **complemento**, **CEP** e **cidade**.

— Especialização/generalização

- Mecanismo representado por um triângulo.
- Funcionário é entidade mais genérica (posição superior na hierarquia).
- Docente e Analista são entidades especializadas (posição inferior).
- No DER, cada entidade especializada herda propriedades da entidade mais genérica.

— Especialização/generalização

DER com os novos requisitos de dados:



DER com mecanismo de especialização/generalização.

— Especialização/generalização

- **Funcionário**: código único, nome e pelo menos um telefone.
- **Docente** (especializado): herda propriedades de Funcionário, possui atributo obrigatório GRADUACAO.
- **Analista** (especializado): herda propriedades de Funcionário.

— Especialização/generalização

Classificações para especialização/generalização

Perguntas analisadas no DER:

- Pode existir funcionário que não seja docente nem analista?
- Pode existir funcionário que seja docente e analista?

— Entidade associativa

- Utilizada para vincular entidade a algum relacionamento.
- Exemplo: relacionamento (OFERTA) entre TURMA e DISCIPLINA.

Características

- Representada por losango dentro de um retângulo.
- Resolve impasses ao associar entidades a relacionamentos complexos.

— Entidade associativa



TURMA

Código, descrição, data de criação.



DISCIPLINA

Pode ser ofertada em várias turmas.



OFERTA (Entidade Associativa)

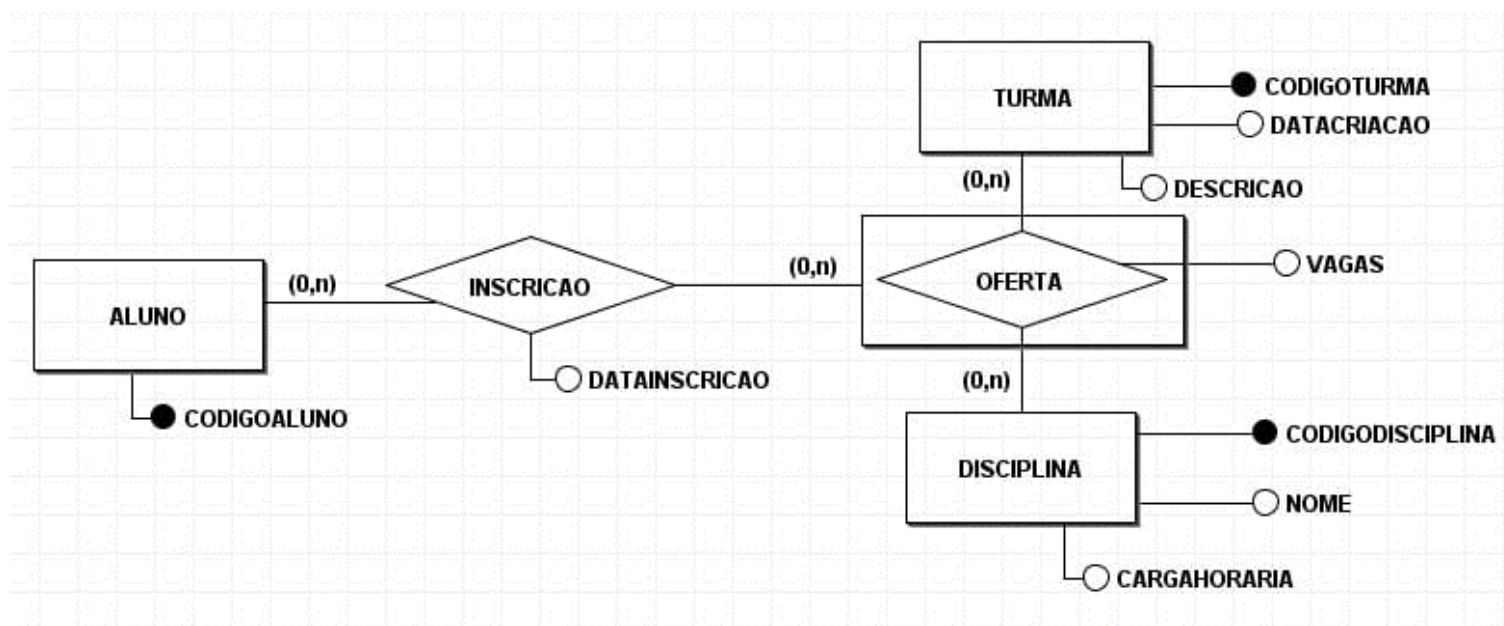
Relacionamento entre TURMA e DISCIPLINA. Atributos: Número de vagas, data de inscrição do aluno.



INSCRIÇÃO (Relacionamento)

Associado a OFERTA e ALUNO.

— Entidade associativa



DER com os novos requisitos de dados, fazendo uso de entidade associativa.

— Entidade associativa

OFERTA sob duas perspectivas:

Relacionamento

OFERTA possui atributo VAGAS, útil no planejamento das turmas e disciplinas que serão ofertadas para inscrição.

Entidade

OFERTA útil para identificar a turma e a disciplina escolhida pelo aluno quando do momento de uma inscrição.

— Objetivos ao construir um DER

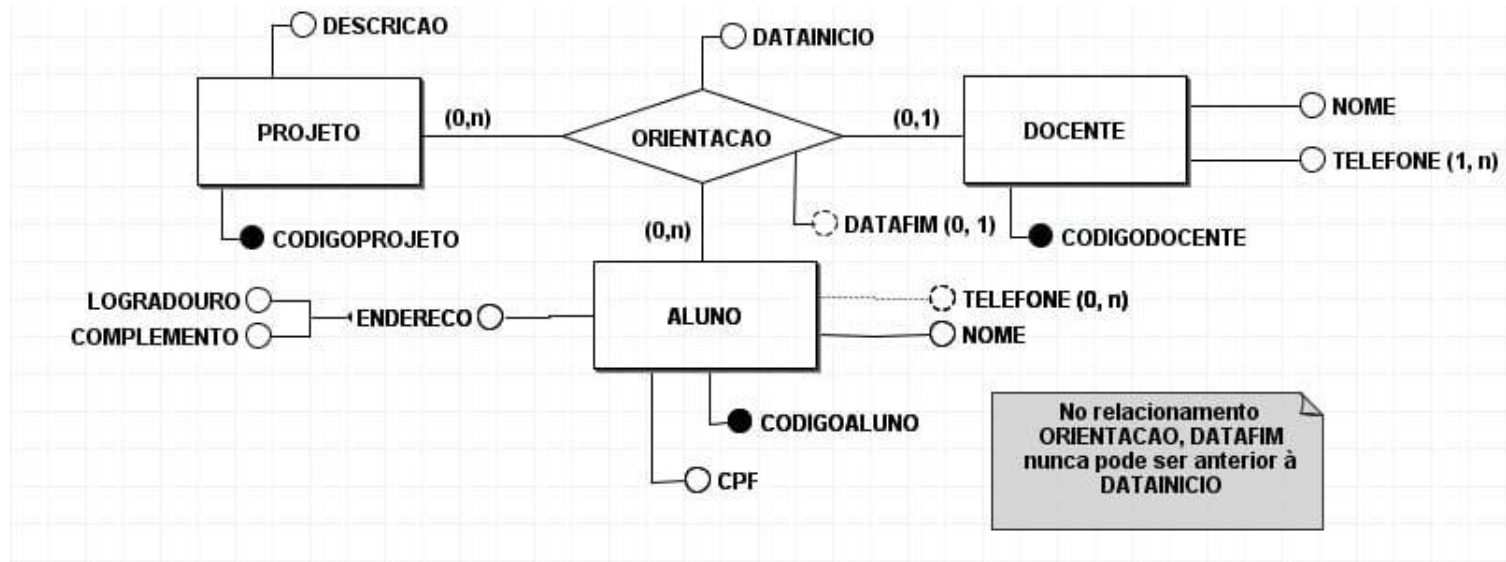
Importância do DER

- Captura partes essenciais do negócio.
- Facilita entendimento uniforme.
- Atualização minimiza curva de aprendizado em TI.

Restrições de integridade

- Além de expressas no DER, algumas em linguagem natural.
- Asseguram conformidade do SGBD com as necessidades do negócio.

Objetivos ao construir um DER



DER com uma restrição semântica adicionada sob o formato de um texto.

— Modelagem de entidades e relacionamentos

Equivalência entre modelos

“

[...] Para fins de projeto de banco de dados, dois modelos ER são equivalentes quando ambos geram o mesmo esquema de banco de dados.

(HEUSER, 2009)

— Modelagem de entidades e relacionamentos

Equivalência entre modelos

Transformação de relacionamento N:N em entidade:

- **Etapa 1:** representar o relacionamento N:N como uma entidade.
- **Etapa 2:** relacionar a entidade criada na etapa 1 às entidades participantes do relacionamento original.
- **Etapa 3:** adicionar à entidade criada na etapa 1 o(s) atributo(s) do relacionamento original.
- **Etapa 4:** a entidade criada será identificada pelos relacionamentos com as entidades originais.
- **Etapa 5:** estabelecer a cardinalidade (1,1) da entidade criada para cada relacionamento vinculado a ela.

— Modelagem de entidade isolada

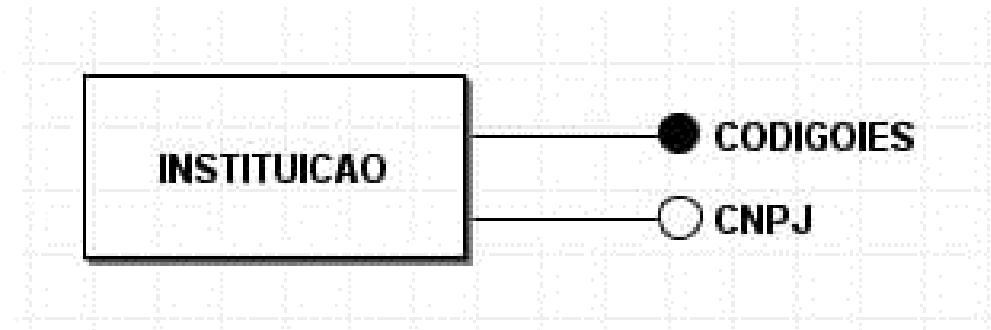
“

[...] Uma entidade isolada é uma entidade que não apresenta relacionamento com outras entidades.

(HEUSER, 2009)

— Modelagem de entidade isolada

- Entidade isolada para modelar IES (DER).
- Adição da entidade INSTITUICAO ao modelo.
- Permanece isolada devido a requisitos de dados.
- A entidade e seus atributos (código, CNPJ)



DER com entidade isolada IES.

— Modelagem de entidade isolada

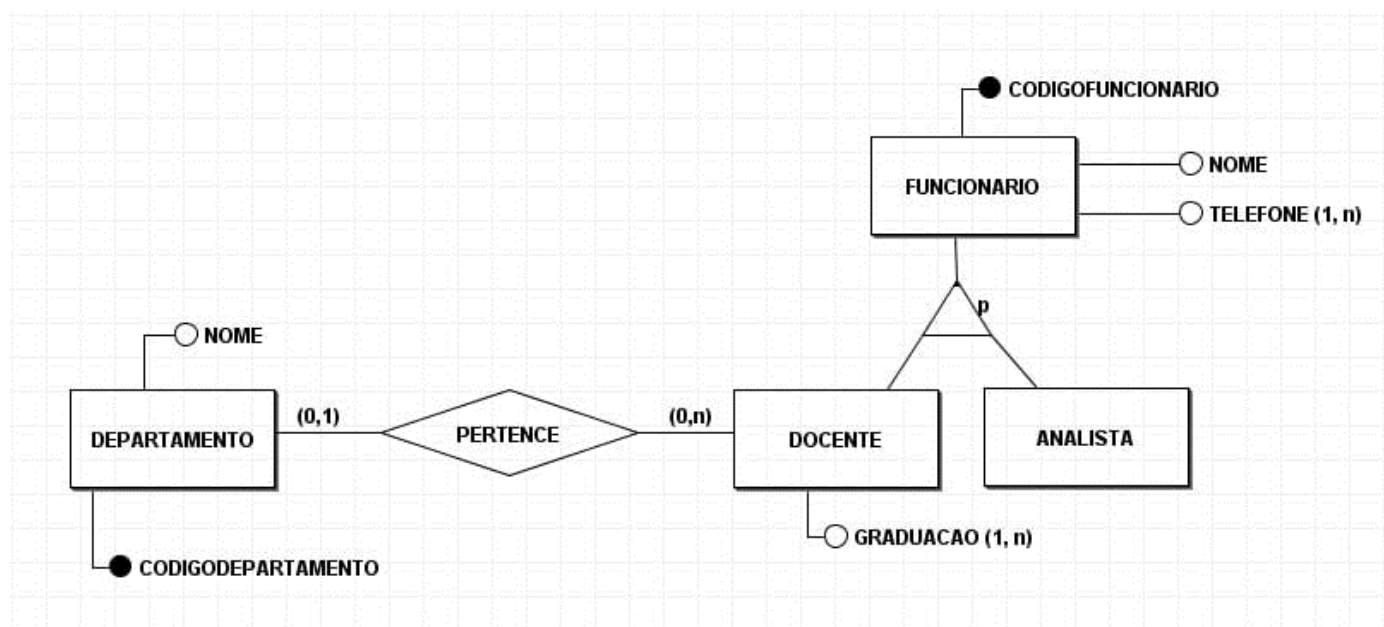
Quando manter histórico

- Associação docente-departamento no modelo acadêmico.
- Requisitos de dados: código e nome para departamentos.
- Docente associado a no máximo um departamento.

— Modelagem de entidade isolada

Quando manter histórico

DER construído com base nos requisitos:



DER associando docente a departamento.

— Modelagem de entidade isolada

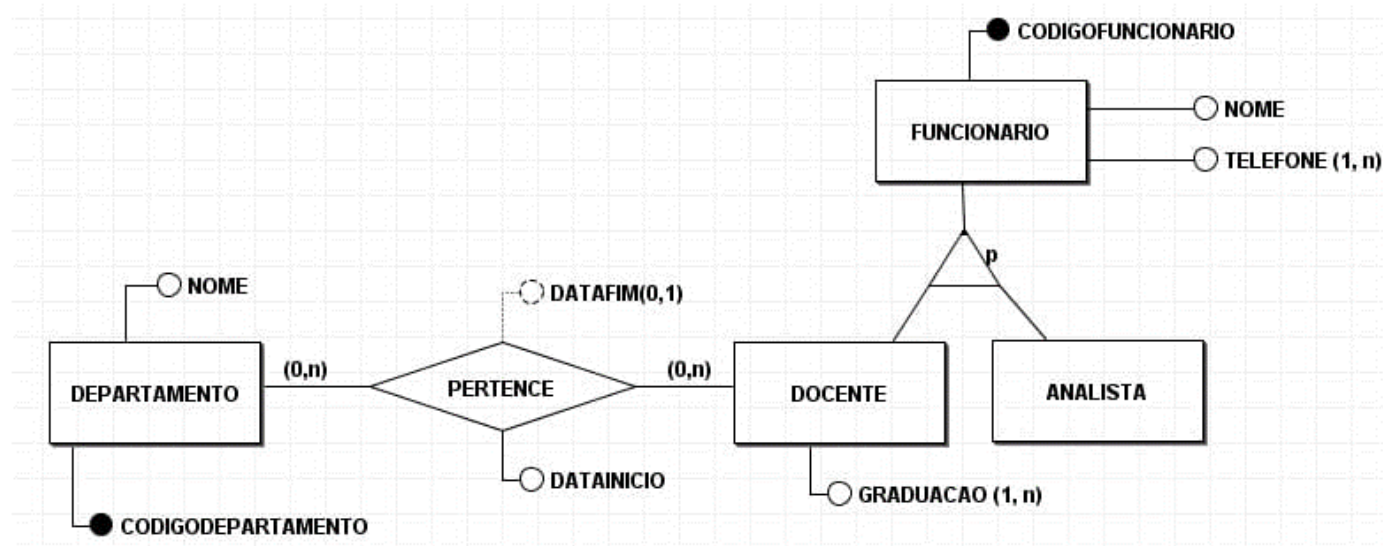
Quando manter histórico

- DER atual reflete requisitos de dados com info nos departamentos e restrição de associação docente-departamento.
- Planejamento estratégico é vital para adaptar o DER a mudanças, como a necessidade de registrar movimentações de docentes.
- Novo requisito inclui armazenamento de movimentações com datas de início e fim, visando histórico institucional.

— Modelagem de entidade isolada

Quando manter histórico

DER contemplando o novo requisito:



DER associando docente a departamento, contendo histórico das movimentações.

— Modelagem de entidade isolada

Quando manter histórico

- Novo DER indica que um docente pode passar por vários departamentos ao longo da carreira (cardinalidade máxima N).
- Atuação do docente em um departamento é definida por datas de início e fim.
- Alteração no DER: mudança na cardinalidade máxima e adição de atributos no relacionamento.

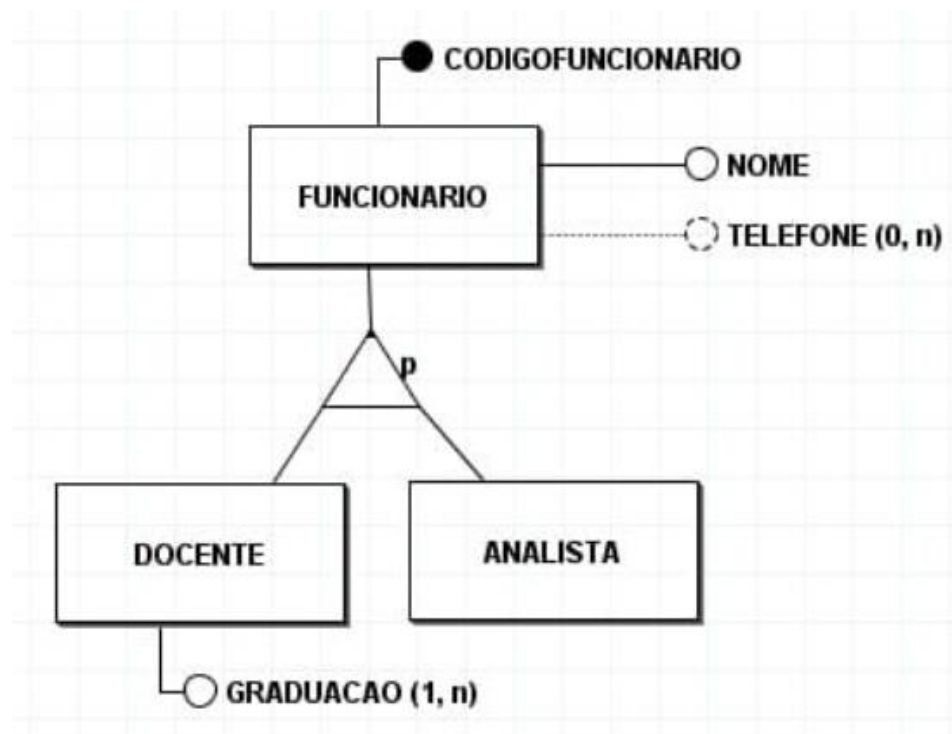
— Mais sobre modelagem de entidades e relacionamentos

- Construção de DER é incremental, não ocorrendo em uma única etapa.
- Estratégia descendente usada: modelagem de conceitos abstratos, identificação de entidades, relacionamentos e atributos.
- Etapas utilizadas para construção do diagrama de entidade e relacionamento: modelo **inicial**, **detalhado** e **validação**.
- Possibilidade de retornar a qualquer etapa durante a construção do DER.

— Atributo x entidade

- Em alguns casos, modelar um objeto como atributo pode não ser a melhor escolha.
- Necessidade de considerar alternativas de modelagem para garantir requisitos específicos.

— Atributo x entidade



DER associando docente a departamento, contendo histórico das movimentações.

— Atributo x entidade

Graduação do docente expressa como atributo, não entidade.

Cardinalidade mínima 1

Informação obrigatória.

Cardinalidade máxima N

Possibilidade de registrar
vários cursos associados ao
docente.

— Atributo x especialização

- Decisão entre atributo e especialização: se objeto tem atributo(s) ou relacionamento(s), opta-se por especialização.
- Entidade **FUNCIONARIO** especializada; escolha coerente ao considerar relacionamento com entidade **GRADUACAO**.

— Atributo opcional



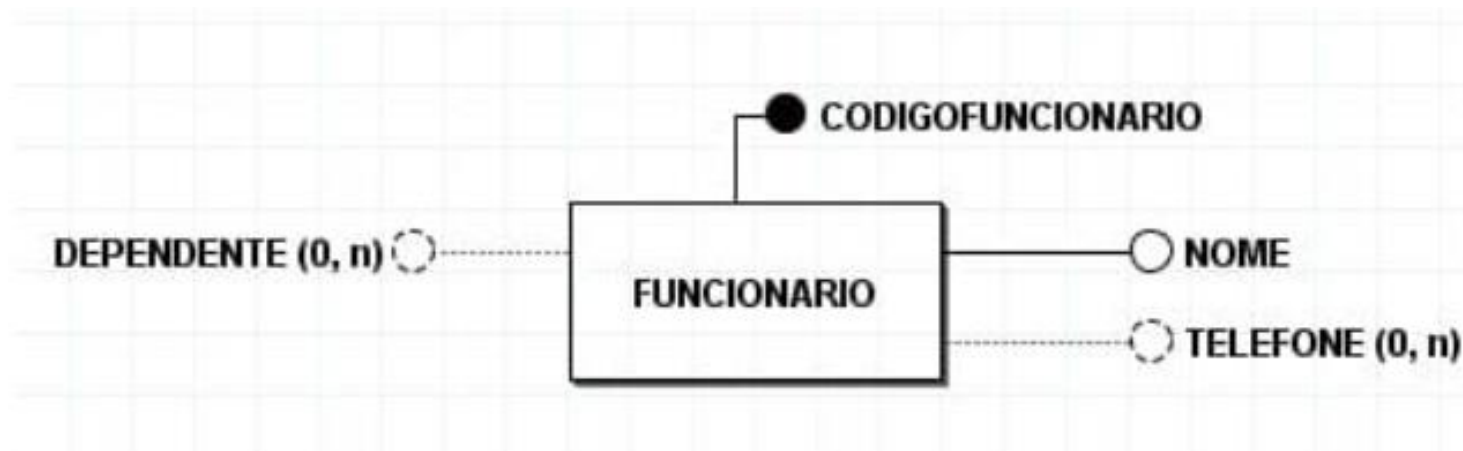
Atenção!

Para possíveis entidades especializadas, caso surjam vários atributos opcionais.

— Atributo multivalorado

- Necessidade de modelar dependente(s) para funcionário no DER.
- Consideração inicial: adicionar atributo opcional na entidade FUNCIONARIO.

— Atributo multivalorado



DER associando docente a departamento, contendo histórico das movimentações

— Atributo multivalorado

- Evitar modelagem com atributos multivalorados devido as limitações no modelo físico de SGBD relacional.
- Atributos multivalorados geralmente escondem detalhes e relacionamentos importantes.
- Desejável permitir controle individual de informações, como nome, data de nascimento e tipo/categoria de cada dependente ou telefone.

— Atributo x redundância

- Atributos redundantes são comuns na modelagem de dados.
- Redundância pode levar a inconsistências e duplicações de informação.
- Importante identificar e eliminar atributos redundantes para garantir integridade e eficiência no banco de dados.

— Atributo composto

Pode ser dividido em:

Subpartes

**Atributos básicos com
significados próprios**

Projeto de banco de dados: modelagem lógica e física



— Modelo relacional

Termo "relação" é utilizado formalmente em banco de dados, sendo informalmente chamado de "tabela" no contexto comercial.

“

O modelo relacional representa o banco de dados como uma coleção de relações.

(ELMASRI; NAVATHE, 2019)

— Componentes de uma tabela

- Tabela corresponde a um conjunto não ordenado de linhas (tuplas).
- Linhas divididas em campos ou colunas (atributos).
- Campos são nomeados para facilitar interpretação dos dados.

— Componentes de uma tabela



Componentes de uma tabela de banco de dados.

— Componentes de uma tabela

Nome de tabelas

- Devem ter nomes únicos e representar entidades do DER.
- Nome da tabela deve refletir claramente o objeto modelado.

— Componentes de uma tabela

Colunas de tabelas

- São monovaloradas, com nome único, e possuem valores atômicos.
- Nome de coluna ajuda a entender sua finalidade.
Ex.: **DTNASCIMENTO** representa **data de nascimento**.
- Colunas não admitem subdivisões.
Ex.: **CODIGOALUNO** não pode ser dividido em outros campos.

— Componentes de uma tabela

Colunas de tabelas

- Ao implementar uma tabela, é necessário definir um tipo de dado para cada coluna (caractere, numérico, data, booleano).
- Alguns SGBDs permitem definição personalizada do tipo de dados.
- O conjunto de valores que uma coluna pode assumir é chamado de **domínio**.
- Valor de coluna pode ser opcional ou obrigatório; opcional permite valores vazios (NULL).

— Componentes de uma tabela

Linhas de tabelas

- Representam itens de informação no banco de dados.
- Cada linha corresponde a uma unidade básica para armazenamento e recuperação de dados.
- Tabela de cadastro de alunos com 10.000 linhas armazena dados de 10.000 alunos.
- Linhas devem armazenar dados de acordo com a semântica do objeto.

— Chave primária

- Propriedades da chave primária incluem: valor não repetido, no máximo um valor por linha, e nenhum valor vazio (restrição de integridade de entidade).
- Alguns SGBDs permitem a propriedade "autoincremento" para gerar automaticamente valores únicos.

— Chave primária

- Chave **mínima**: todas as colunas que a compõem devem ser necessárias e suficientes para diferenciar linhas. A chave primária composta (CODIGOFUNCAIONARIO, NRDEPENDENTE) na tabela DEPENDENTE é mínima.
- Chave **candidata**: mais de uma coluna pode servir como chave primária. Ex.: CODIGOALUNO e CPF são chaves candidatas na tabela ALUNO.
- Chave **alternativa**: coluna que poderia ter sido escolhida como chave primária, mas não foi.
- Algumas vezes, escolhe-se uma chave primária artificial para manter independência das colunas e preservar informações de negócios.

— Chave estrangeira

- Modelo relacional é composto por tabelas e relacionamentos entre elas.
- Relacionamento entre FUNCIONARIO e DEPENDENTE exemplificado.
- Cada dependente associado a um funcionário (1 para N).
- Um funcionário pode ter vários dependentes, ou nenhum.

— Chave estrangeira

FUNCIONARIO

CODIGOFUNCIONARIO	NOME	SEXO	CPF	DTNASCIMENTO
1	José Maciel	M	23456239999	13/09/1982
2	Pedro Antônio	M	98789345552	13/11/1986
3	Debora Silva	F	12332149048	15/02/1998
4	Amanda de Miranda	F	60989893223	15/05/1997

DEPENDENTE

CODIGOFUNCIONARIO	NRDEPENDENTE	NOME	DTNASCIMENTO
1	1	Andrey Campos	13/07/2019
1	2	Manoel Oliveira	13/12/2018
2	1	João Silva	15/02/2017
2	2	José Maciel	15/07/2016

Relacionamento entre FUNCIONARIO e DEPENDENTE.

— Chave estrangeira

“

Uma chave estrangeira é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela.

(HEUSER, 2009)

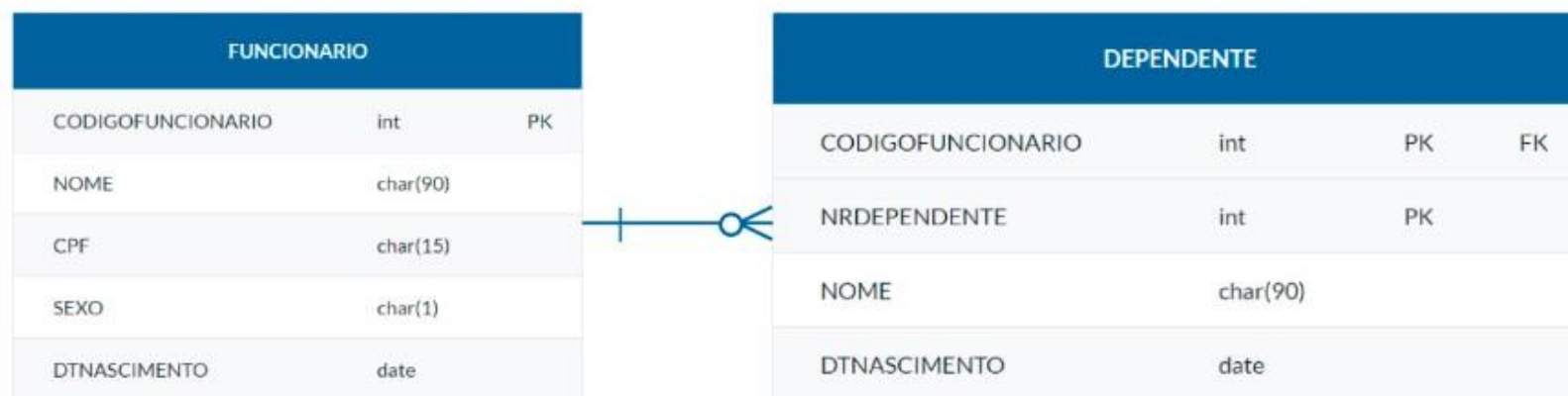
— Chave estrangeira

Restrições impostas pela chave estrangeira

- Chave estrangeira implementa relacionamentos entre tabelas e impõe restrições para manter integridade no banco de dados.
- Restrições incluem garantir que valores da chave estrangeira existam na chave primária referenciada.
- **Inclusão, alteração e exclusão** de linhas estão sujeitas a essas restrições para preservar integridade referencial.

— Esquema diagramático de banco de dados relacional

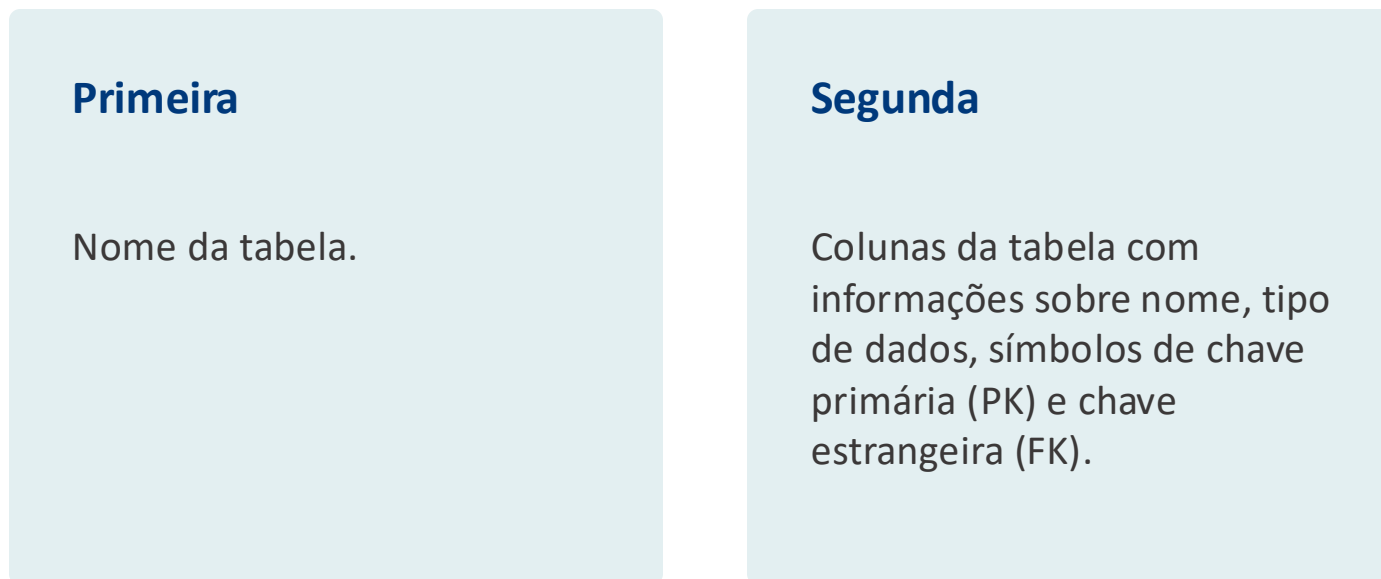
Diagrama construído a partir de uma ferramenta comercial:



Esquema diagramático “pé de galinha” envolvendo as tabelas FUNCIONARIO e DEPENDENTE.

— Esquema diagramático de banco de dados relacional

- Cada tabela é representada por um retângulo com duas subdivisões.




- Símbolo semelhante a um "pé de galinha" na tabela DEPENDENTE representa o lado N do relacionamento entre as tabelas.

— Esquema diagramático de banco de dados relacional

Esquema textual de banco de dados relacional

O banco de dados declarado sob o formato textual:



```
FUNCIONARIO (CODIGOFUNCIONARIO, NOME, CPF, SEXO, DTNASCIMENTO)  
  
DEPENDENTE (CODIGOFUNCIONARIO, NRDEPENDENTE, NOME, DTNASCIMENTO)  
  
CODIGOFUNCIONARIO REFERENCIA FUNCIONARIO
```

— Normalização

- Modelar um banco de dados relacional envolve mais do que adicionar tabelas e relacionamentos sem critério.
- Normalização é um processo que avalia se um banco de dados foi bem projetado, baseado em formas normais (FN).
- Para a maioria dos projetos, normalizamos até a 3FN.
- Processo de normalização: identificar origem dos dados, construir tabela não normalizada, aplicar regras da 1FN, 2FN e 3FN.

— Normalização

Tabela não normalizada

- Não segue as regras de normalização.
- Tabelas não normalizadas podem conter redundância desnecessária, levando a anomalias de atualização, exclusão e inserção.
- A aplicação da normalização é uma prática recomendada para garantir eficiência e consistência nos dados.

— Normalização

Primeira forma normal (1fn)

- Uma tabela está na 1FN se todos os seus atributos contêm valores atômicos (indivisíveis) e não existem grupos repetitivos de colunas.
- Serve para eliminar a redundância de dados e garantir que cada valor em uma tabela seja indivisível.

— Normalização

Segunda forma normal (2FN)

- Uma tabela está na 2FN se estiver na 1FN e todos os seus atributos não primários (não fazem parte da chave primária) são totalmente dependentes da chave primária.
- Serve para eliminar dependências parciais, garantindo que todos os atributos dependam completamente da chave primária.

— Normalização

Terceira forma normal (2FN)

- Uma tabela está na 3FN se estiver na 2FN e não existirem dependências transitivas entre os atributos não primários.
- Serve para eliminar dependências transitivas, garantindo que não haja informações redundantes.

— Mapeamento conceitual-lógico

- No projeto de banco de dados, o modelo conceitual usa o Diagrama de Entidade e Relacionamento (DER).
- DER representa entidades, atributos e relacionamentos no negócio modelado.
- Componentes do modelo relacional são essenciais para a implementação de bancos de dados relacionais.

— Mapeamento conceitual-lógico

Regras de mapeamento

Quatro etapas do mapeamento conceitual-lógico:

- Entidades;
- Relacionamentos;
- Atributos multivalorados;
- Especialização/generalização.

— Mapeamento de entidades

Entidade forte ou independente

- Cada entidade E vira uma tabela T.
- Cada atributo simples da entidade E vira uma coluna na tabela T.
- Atributo identificador da entidade E vira chave primária na tabela T.

Entidade fraca ou dependente

- Cada entidade fraca F vira uma tabela T.
- Cada atributo simples da entidade F vira uma coluna na tabela T.
- Tabela T possui chave estrangeira originada da chave primária da tabela proprietária.
- Tabela T possui chave primária composta pela coluna chave estrangeira e pela coluna mapeada do atributo identificador da entidade F.

— Mapeamento de relacionamento

Depende da cardinalidade máxima:

1:1

- $(0,1):(0,1)$: adicionar coluna(s) ou tabela própria.
- $(0,1):(1,1)$: fusão de tabelas ou adição de colunas.
- $(1,1):(1,1)$: fusão de tabelas.

1:N

- Adicionar chave estrangeira na tabela do lado N referente à chave primária da tabela do lado 1.
- Cada atributo simples do relacionamento vira coluna na tabela do lado N.

— Mapeamento de relacionamento

Depende da cardinalidade máxima:

N:N e N-ários (análogo a N:N)

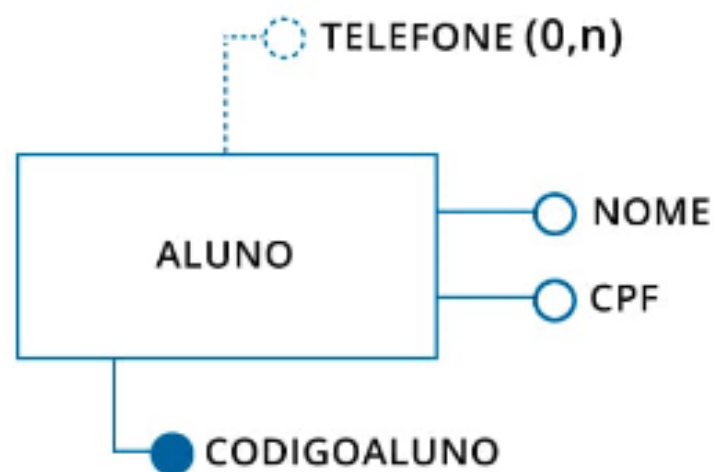
- Criar tabela para o relacionamento (T).
- Adicionar chaves estrangeiras de tabelas participantes.
- Chave primária composta pelas chaves estrangeiras.
- Cada atributo simples do relacionamento vira coluna na tabela T.

— Mapeamento de atributos multivalorados

- Criar uma tabela (T) para cada atributo multivalorado.
- Criar coluna(s) para o(s) atributo(s) multivalorado(s).
- Tabela T possui chave estrangeira originada da chave primária da tabela original.
- Tabela T possui chave primária composta pela chave estrangeira e pela(s) coluna(s) referente(s) ao(s) atributo(s) multivalorado(s).

— Mapeamento de atributos multivalorados

DER parcial contendo atributo multivalorado.



DER parcial contendo atributo multivalorado.

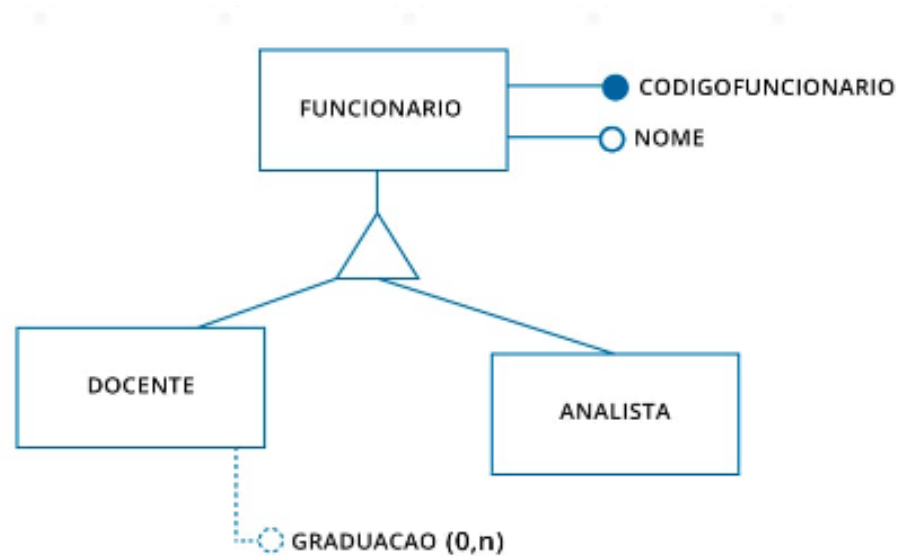
— Mapeamento de especialização/generalização

Tabela única

- Criar uma tabela única com todos os atributos das entidades genérica e especializadas.
- Criar coluna TIPO para identificar a entidade especializada.

— Mapeamento de especialização/generalização

Tabela única



FUNCIONARIO		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	
TIPO	char(40)	

— Mapeamento de especialização/generalização

Uma tabela para cada entidade

- Criar tabela para entidade genérica e para cada entidade especializada.
- Tabelas especializadas possuem chave estrangeira da genérica, sendo também sua chave primária.

— Mapeamento de especialização/generalização

Uma tabela para cada entidade



— Mapeamento de especialização/generalização

Subdivisão da entidade genérica

- Criar tabelas separadas para entidades especializadas.
- Incluir colunas dos atributos da entidade genérica nas tabelas especializadas, usando a chave primária originada do identificador da entidade genérica.
- Se a hierarquia for parcial, criar tabela adicional para entidades genéricas não presentes nas especializadas, contendo apenas seus atributos.

— Mapeamento de especialização/generalização

Subdivisão da entidade genérica

DOCENTE		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	

ANALISTA		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	

OUTROSFUNCIONARIOS		
CODIGOFUNCIONARIO	int	PK
NOME	char(90)	

— Consultas

- Compreender as consultas e transações planejadas para o banco de dados.
- Utilizar índices em colunas frequentemente usadas em cláusulas WHERE.
- Evitar índices em colunas com baixa seletividade.
- Limitar o uso de JOINS complexos e buscar alternativas como desnormalização.
- Utilizar partições para gerenciar grandes volumes de dados.
- Monitorar e ajustar o desempenho conforme necessário.

— Transações

Uma transação é uma unidade lógica de trabalho em um SGBD.

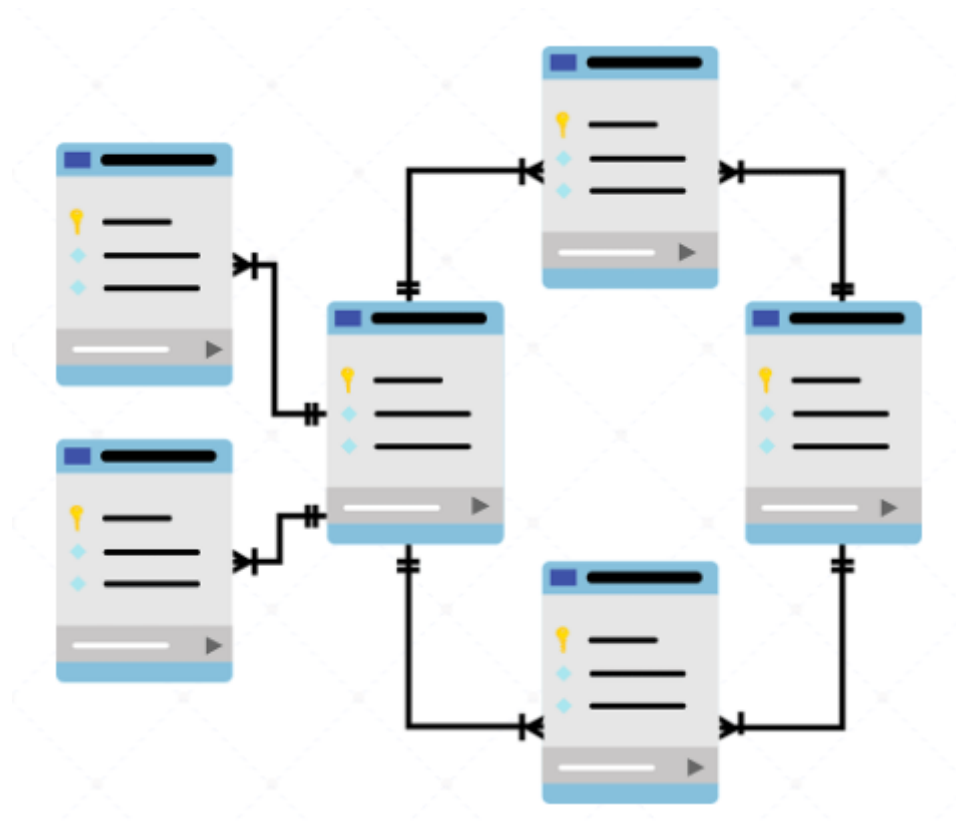
Características

- Consiste em uma série de operações.
- Todas as operações precisam ser executadas ou canceladas.
- Garante a atomicidade, onde todas as operações são confirmadas ou nenhuma é realizada.

— Indexação em banco de dados

- Estruturas auxiliares para otimizar a recuperação de registros.
- Similar à consulta de um índice remissivo em um livro.
- No banco de dados, são frequentemente utilizados para acelerar a busca por valores-chave em tabelas.
- Um índice é criado automaticamente para a chave primária de uma tabela, facilitando consultas por esse valor.

— Indexação em banco de dados



— Indexação em banco de dados

Projeto físico em bancos de dados relacionais



— Indexação em banco de dados

Consultas e transações de banco de dados

Para cada consulta de recuperação de dados, é necessário conhecer:

- **Tabelas acessadas;**
- **Colunas para condições de seleção;**
- **Natureza da condição de seleção;**
- **Colunas para operações de junção;**
- **Colunas nos resultados.**

— Indexação em banco de dados

Consultas e transações de banco de dados

- Consulta SQL avalia registros na tabela CURSO com NOME "Medicina" ou "Nutrição."
- A cláusula WHERE envolve a coluna NOME, sugerindo a criação de índice.
- Consulta II busca recuperar nome do curso e seu nível.
- Necessidade de otimização para melhorar desempenho.

— Indexação em banco de dados

Consultas e transações de banco de dados

- Uso de comando JOIN na consulta, com condição avaliada repetidamente.
- Recomendação de criar índices nas colunas CODIGONIVEL para otimização.
- Para operações de atualização, é essencial conhecer tabelas, categoria e colunas envolvidas.
- No contexto de indexação, colunas usadas em condições de seleção são ideais para índices; evitar índices para colunas de atualização.

— Indexação em banco de dados

Consultas envolvendo mais de uma tabela

- Consulta SQL recupera código do docente e nome do município de nascimento.
- Utiliza JOIN implícito entre tabelas DM_DOCENTE_2 e MUNICIPIO.
- Gera tabela temporária com mais de dois bilhões de linhas.
- Consulta frequente pode causar lentidão no sistema.

— Desnormalizar para ganhar desempenho

- Normalização (até 3FN) minimiza redundância, mas consultas podem ser custosas.
- Desnormalização busca melhorar desempenho, consolidando dados em uma tabela.
- Consulta simplificada com uma tabela desnormalizada.
- Desvantagens: redundância e necessidade de atualizações adicionais para consistência.

Criação e manipulação de objetos no Postgresql



Breve histórico

PostgreSQL originou-se do projeto POSTGRES, vinculado ao INGRES, na Universidade da Califórnia em Berkeley.

1986 - 1989

Implementação iniciada em 1986, tornou-se operacional em 1987, com versões públicas lançadas a partir de 1989.

1995

Postgres95 trouxe revisões no código e adoção do SQL como interface padrão.

1996

Renomeado para PostgreSQL em 1996 (versão 6), destacando-se como um SGBD de código aberto para Windows, Mac OS e Linux.

— Arquitetura do PostgreSQL

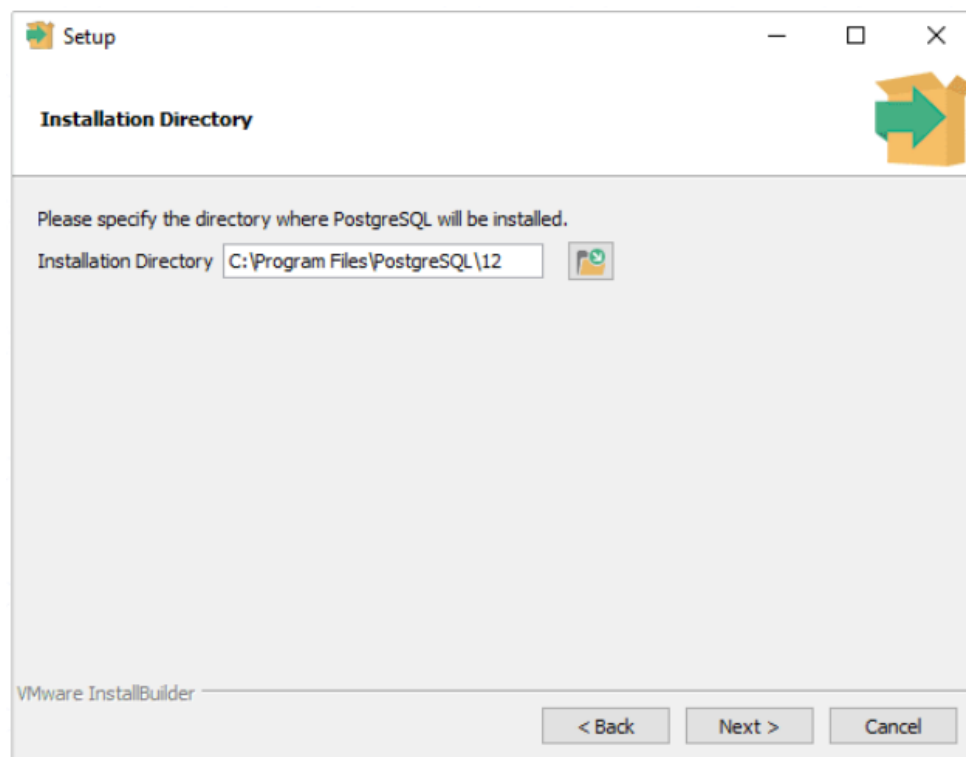
- Processo servidor gerencia arquivos, conexões e executa comandos do banco.
- Aplicativo cliente solicita acesso, manipula dados e consulta tabelas.
- **Comunicação Remota:** ambos podem estar em máquinas diferentes, conectando-se via TCP/IP.
- **Instalação do PostgreSQL:** escolher versão adequada ao sistema operacional, seguindo as atualizações na página oficial.

— Instalação do PostgreSQL no Linux

- Fazer download do arquivo .tar.gz do site PostgreSQL.
- Obter o código fonte no prompt do Linux com comandos específicos.
- Acessar o diretório criado pelo tar após a extração.
- Checar a instalação por meio de comandos específicos no prompt Linux.

— Instalação do PostgreSQL no Windows

- Deve-se executar o arquivo como usuário administrador.
- Diretório padrão no Windows: C:\Program Files\PostgreSQL\12\data.



— Instalação do PostgreSQL no Windows

- São instalados:

pgAdmin

Interface gráfica de administração.

psql

Ferramenta de linha de comando para administração.

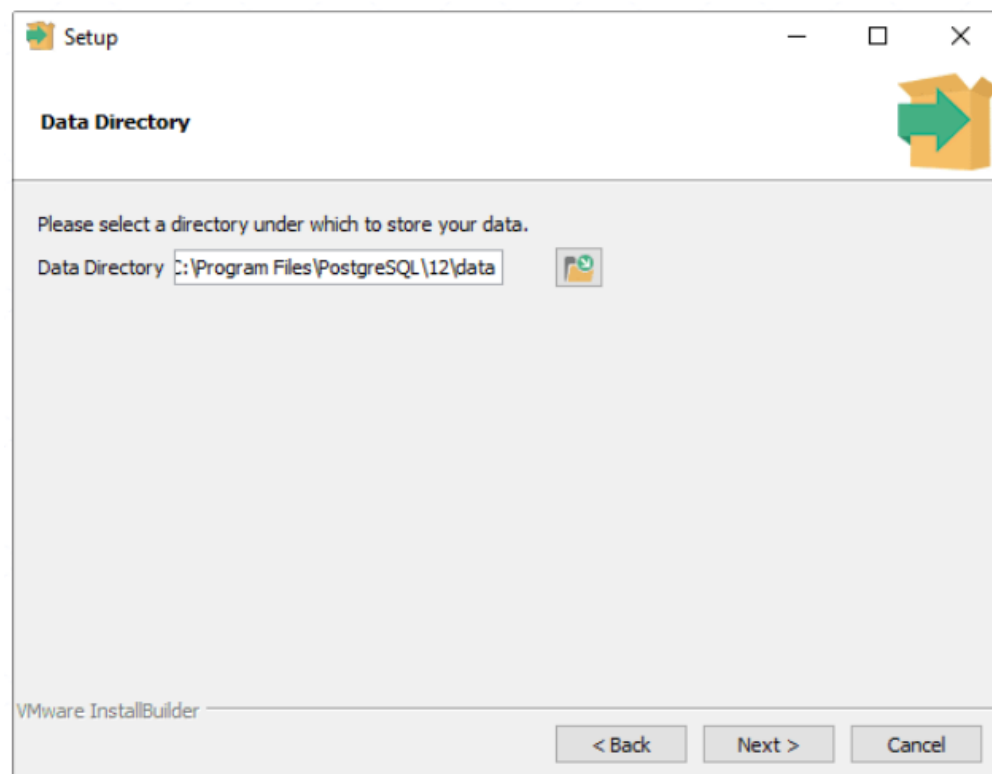
Stack Builder

Ferramenta útil para gerenciar a instalação de módulos complementares, tais como utilitários, drivers e extensões.

- Cada pasta armazena metadados e arquivos numerados para tabelas.
- Visualização no pgAdmin 4: árvore de diretórios exibe PostgreSQL 12, database "postgres," e suas tabelas no servidor.

— Instalação do PostgreSQL no Windows

Cada pasta armazena metadados e arquivos numerados para tabelas.



— Instalação do PostgreSQL no Windows

Visualização no pgAdmin 4: árvore de diretórios exibe PostgreSQL 12, database "postgres," e suas tabelas no servidor.


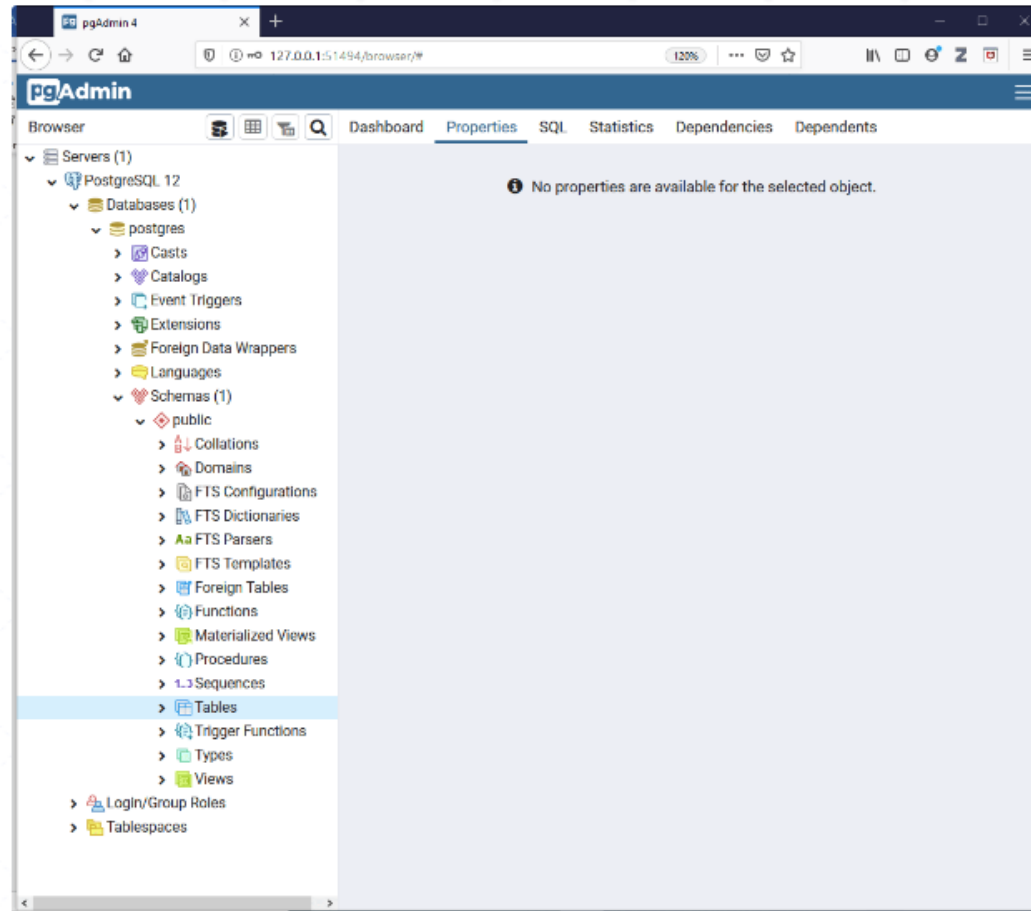


Diagrama da árvore de diretórios do pgAdmin 4:

- Servers (1) → PostgreSQL 12
 - Databases (1) → postgres
 - Schemas (1) → public
 - Tables

— Instalação do PostgreSQL no Windows

Tela do pgAdmin 4 com a árvore de diretórios da instalação padrão do PostgreSQL.



— Interfaces para interagir com o PostgreSQL

Oferece duas interfaces de acesso:

pgAdmin 4

Interface gráfica via navegador para acesso aos recursos do SGBD.

psql

Interface de linha de comando para submissão interativa de comandos ao SGBD via terminal.

— Criando databases com o pgAdmin 4 e com o PSQL

Teste de criação de databases

Usando pgAdmin 4

- Abrir Query Tool, digitar `CREATE DATABASE BDTESTEPGADMIN;` e executar.
- Alternativa: criar pelo pgAdmin 4 interface.

Usando psql (terminal)

- Digitar `CREATE DATABASE BDTESTEPSQL;` e pressionar <enter>.

Verificação: pgAdmin 4 exibe os databases criados.

— Criando databases com o Pgadmin 4 e com o psql

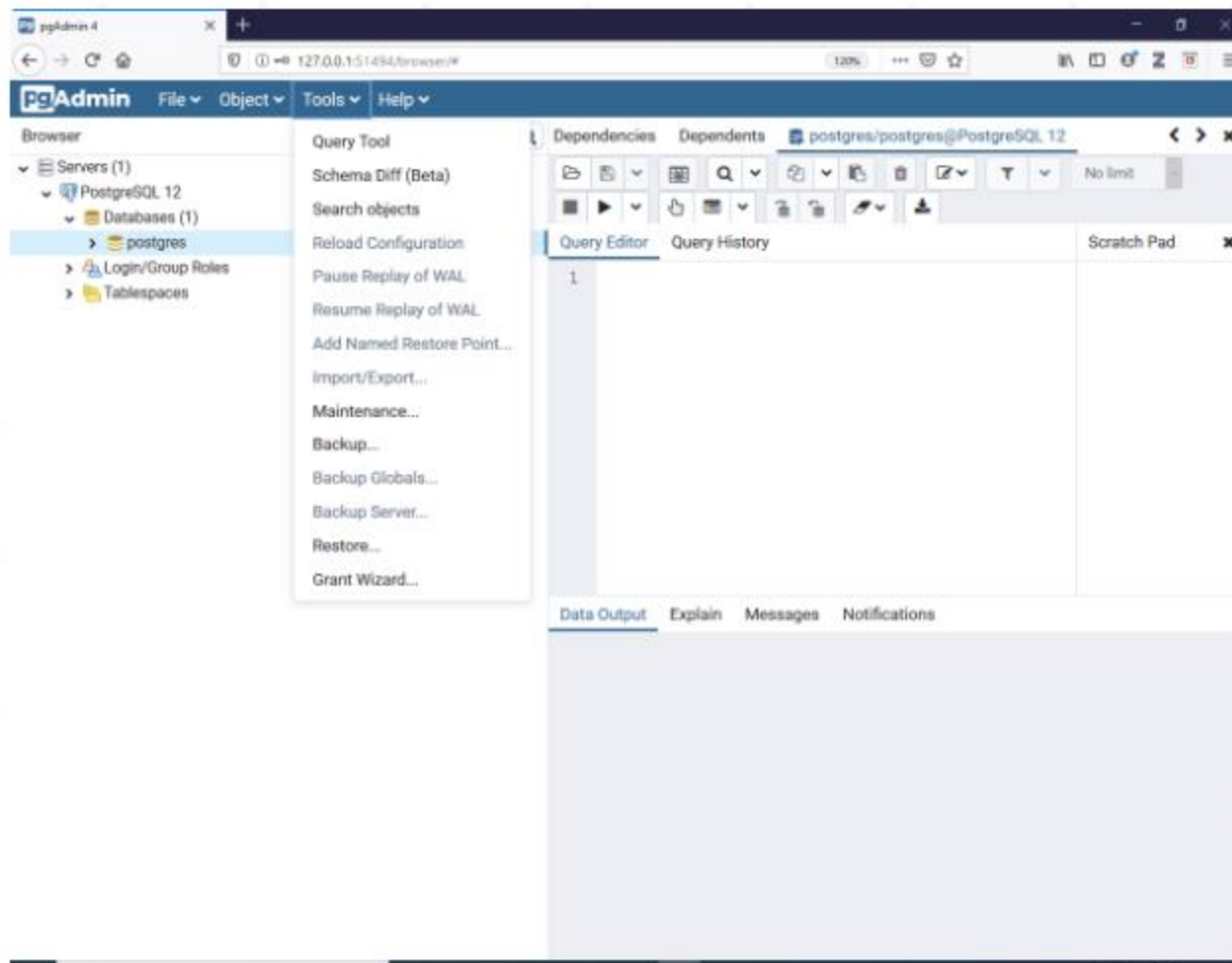
Teste de criação de databases

- *database* BDTESTEPGADMIN, a ser criado usando o pgAdmin 4.
- *database* BDTESTEPSQL, a ser criado usando o psql

— Criando databases com o Pgadmin 4 e com o psql

- No Windows, abra pgAdmin 4 via botão Iniciar.
- Navegador exibirá ambiente com database "postgres".
- Este database é criado automaticamente pelo instalador.

— Criando databases com o Pgadmin 4 e com o psql



— Criando databases com o Pgadmin 4 e com o psql

Criação de Database BDTESTPGADMIN no pgAdmin 4:

- No Query Editor, digitar: **CREATE DATABASE BDTESTPGADMIN;**
- Pressionar tecla F5 ou botão correspondente para executar o comando.



— Criando databases com o Pgadmin 4 e com o psql

Criação Alternativa no pgAdmin 4:

- Interface gráfica: Botão direito em Databases, selecione Create.
- Criação de Database BDTESTEPSQL no psql (terminal):
- Digitar **CREATE DATABASE BDTESTEPSQL;** no terminal e pressionar <enter>.

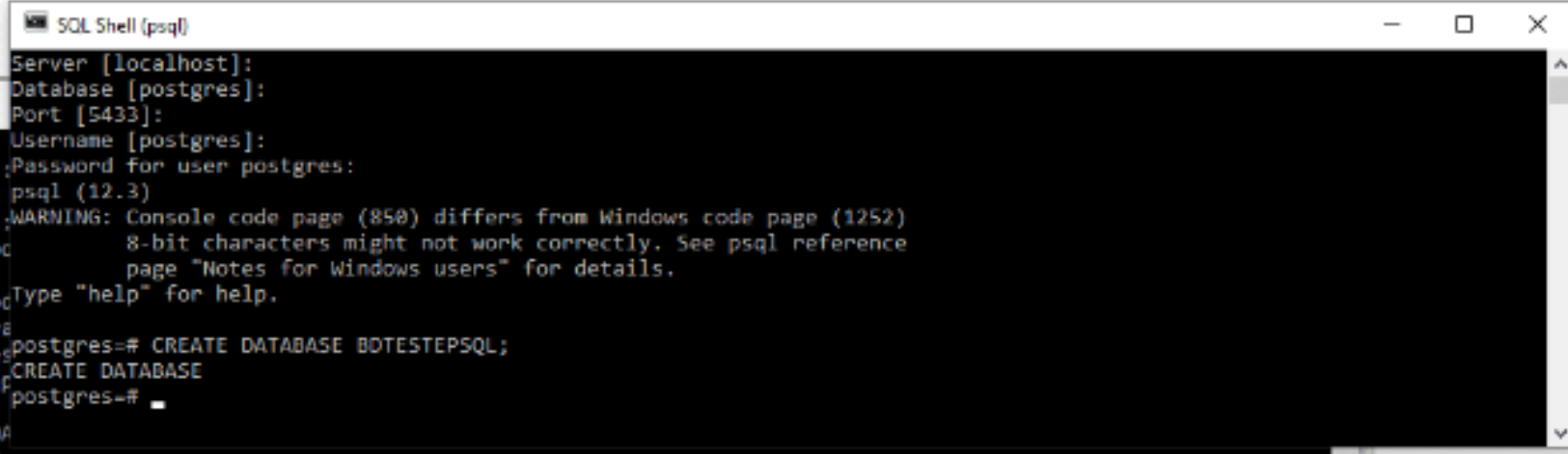
A terminal window with a dark gray title bar containing three colored window control buttons (red, yellow, green). The main area of the terminal is light gray and contains the text `CREATE DATABASE BDTESTEPSQL;` in a monospaced font, with the word `CREATE` in red and `BDTESTEPSQL;` in black.

```
CREATE DATABASE BDTESTEPSQL;
```

— Criando databases com o Pgadmin 4 e com o psql

Verificação no psql:

- Após comando, tela do psql exibe resultado da criação do BDTESTEPSQL.

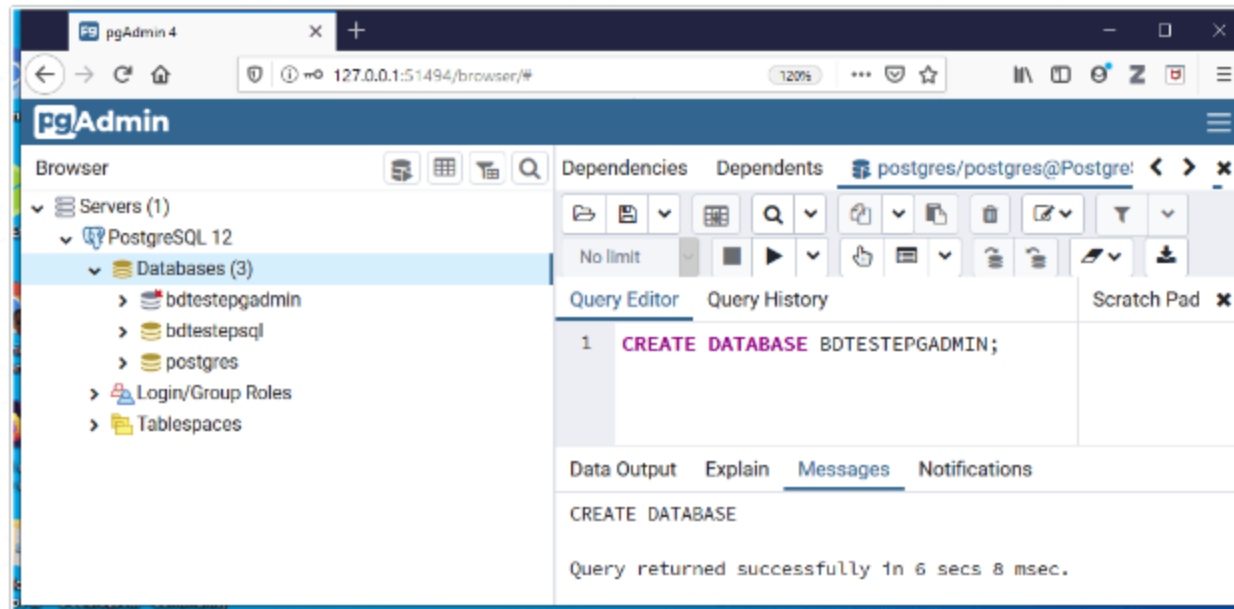


```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5433]:
Username [postgres]:
Password for user postgres:
psql (12.3)
WARNING: Console code page (850) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.
postgres=# CREATE DATABASE BDTESTEPSQL;
CREATE DATABASE
postgres=#
```


— Criando databases com o Pgadmin 4 e com o psql

Confirmação no pgAdmin 4:

- Ambos os databases criados são exibidos na interface.



— Breve histórico da SQL

- Criada na IBM na década de 1970, inicialmente chamada SEQUEL.
- Inspirada na facilidade do comando SELECT para consultas em bancos de dados relacionais.
- Evolução resultou em problemas de compatibilidade entre SGBDs.
- Instituto ANSI definiu padrões para SQL, conhecida como ANSI-SQL.
- Padrão inclui sublinguagens para definição e manipulação de dados.
- Diversos SGBDs compatíveis e extensões para facilitar o desenvolvimento.

— Acesso ao Postgresql


Acesso ao PostgreSQL para Criação de Tabelas:

- Abra o pgAdmin 4 para acessar o ambiente de manipulação de objetos.
- Escolha um database na hierarquia e clique com o botão direito.
- Selecione a opção Query Tool para iniciar a criação de tabelas.

O pgAdmin é a interface Web padrão, mas você pode escolher outro utilitário para praticar comandos SQL.

— Criando um banco de dados

Criar database "bdestudo": **CREATE DATABASE BDESTUDO;**


A terminal window with a dark gray title bar containing three colored window control buttons (red, yellow, green). The main area is light gray and contains two lines of text: a comment and a SQL command.

```
-- Comando para criar um database.
```

```
CREATE DATABASE BDESTUDO;
```

— Criando um banco de dados

Remover database "bdestudo": **DROP DATABASE BDESTUDO;**



```
-- Comando para remover um database.
```

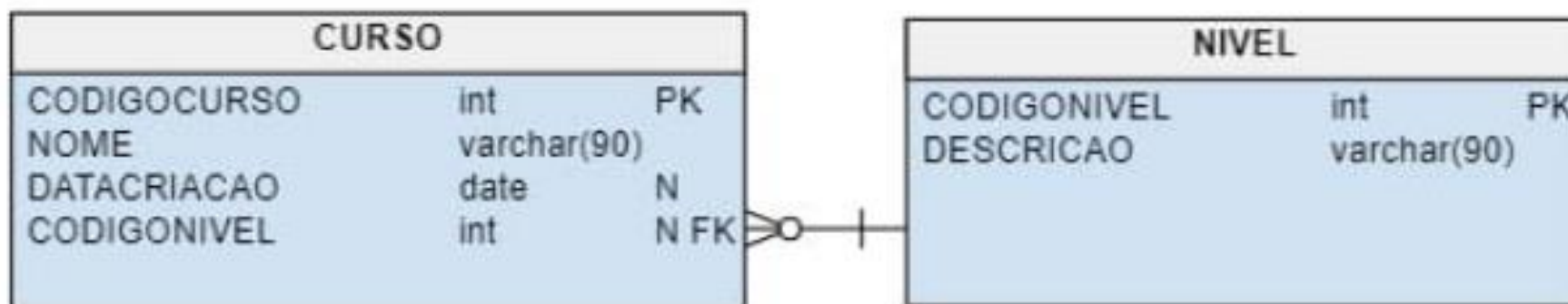
```
DROP DATABASE BDESTUDO;
```

— Criando um banco de dados

- Todo database possui o schema padrão "public" para armazenar tabelas.
- Para usar um schema diferente, criar com: **CREATE SCHEMA** esquema.
- Tabelas em schemas diferentes devem ser especificadas como: **esquema.tabela**.

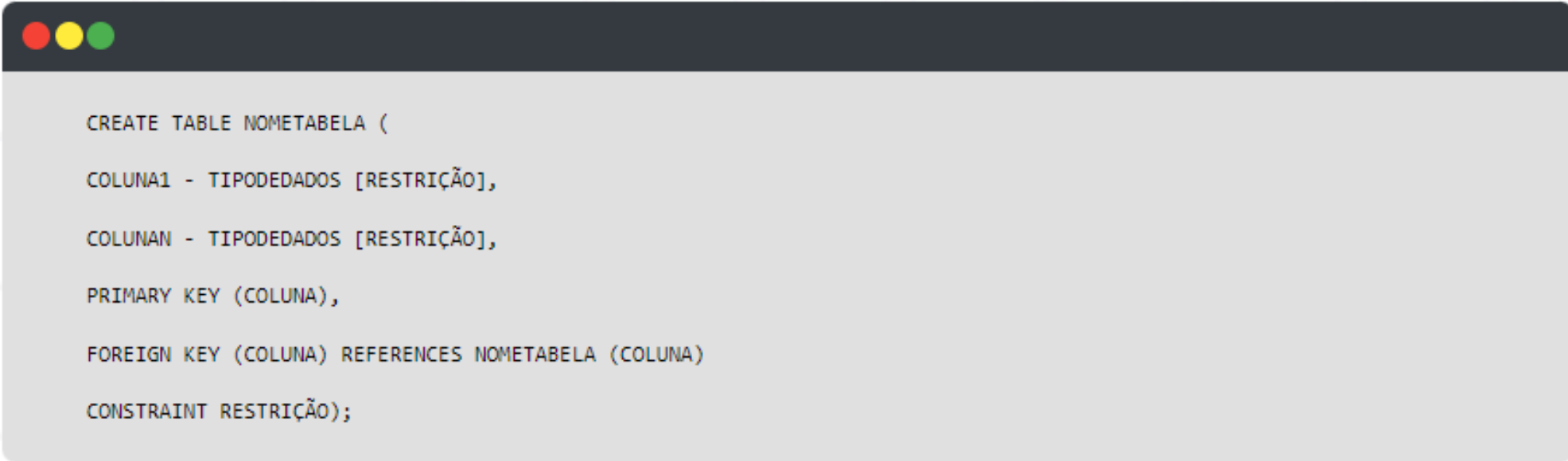
— Criando tabelas

Tabelas são criadas com auxílio do comando **CREATE TABLE**:



— Criando tabelas

Sintaxe **básica** do comando **CREATE TABLE**:



```
CREATE TABLE NOMETABELA (  
  COLUNA1 - TIPODEDADOS [RESTRIÇÃO],  
  COLUNAN - TIPODEDADOS [RESTRIÇÃO],  
  PRIMARY KEY (COLUNA),  
  FOREIGN KEY (COLUNA) REFERENCES NOMETABELA (COLUNA)  
  CONSTRAINT RESTRIÇÃO);
```


— Criando tabelas

Itens da tabela:

- **NOMETABELA**: nome da tabela a ser criada.
- **COLUNA1 e COLUNAN**: colunas da tabela.
- **TIPODEDADOS**: tipo de dados ou domínio da coluna.
- **RESTRIÇÃO**: propriedades associadas à coluna (opcional).
- **PRIMARY KEY**: chave primária da tabela.
- **FOREIGN KEY**: chave estrangeira referenciando outra tabela.
- **CONSTRAINT RESTRIÇÃO**: restrições adicionais (opcional).

— Tipos de dados

Tipos de Dados no PostgreSQL:

- **bigint**: inteiros entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807.
- **char(comprimento)**: sequência de caracteres de tamanho fixo com preenchimento de espaços.
- **date**: data no formato AAAA-MM-DD.
- **decimal**: precisão do valor de casas decimais.
- **double**: precisão do valor de até 15 casas decimais.
- **int ou integer**: inteiros entre -2.147.483.648 e 2.147.483.647.

— Tipos de dados

Tipos de Dados no PostgreSQL:

- **money**: valores monetários.
- **numeric**: precisão do valor de casas decimais.
- **real**: precisão do valor de até seis casas decimais.
- **serial**: valor único inteiro sequencial.
- **smallint**: valores entre 32.768 e 32.767.
- **time**: horário no intervalo de 00:00:00 a 24:00:00.
- **varchar**(comprimento): sequência de dados de caracteres com comprimento variável.

— Gerenciamento de *scripts* na prática


- Projetos reais podem ter dezenas de tabelas, tornando manual a gestão difícil.
- Comandos DDL são essenciais, mas ferramentas automatizadas são práticas.
- Ferramentas visuais como DBeaver, Vertabelo, SQL Power Architect, Toad for SQL, Erwin, etc.
- Permitem criar tabelas, relacionamentos e restrições visualmente.
- Facilitam a geração de código DDL para o projeto ou partes específicas dele.

— SGBD PostgreSQL nos bastidores

- Comando `CREATE DATABASE` gera etapas gerenciadas pelo servidor.
- No Windows, cria uma pasta com OID em `C:\Program Files\PostgreSQL\12\data\base`.
- PostgreSQL mantém informações em um database especial, o catálogo.
- Tabelas do catálogo têm nomes iniciados com o prefixo `PG_`.
- Exemplo: Informações de databases em `PG_DATABASE`.
- Consulta: `SELECT OID, DATNAME FROM PG_DATABASE;`
- Resultado pode variar em seu computador, dependendo das operações realizadas no SGBD.

— Alteração de tabela

- Comando ALTER TABLE útil para modificar estrutura de tabela existente.
- Sintaxe **básica**.



```
ALTER TABLE <NOMETABELA> ADD <COLUNA><TIPODEDADOS> ;
```

— Alteração de tabela

Na sintaxe significa:

<NOMETABELA>

Nome da tabela a ser modificada.

<COLUNA>

Nome da nova coluna a ser adicionada.


<TIPODEDADOS>

Tipo de dados ou domínio da nova coluna.

— Alteração de tabela

Remoção de tabela

Comando **DROP TABLE** remove uma tabela do banco de dados:


A terminal window with a dark gray title bar containing three colored window control buttons (red, yellow, green). The main area of the terminal is light gray and contains the text 'DROP TABLE <NOMETABELA>;'.

```
DROP TABLE <NOMETABELA>;
```


— Alteração de tabela

Remoção de tabela

Sintaxe: **DROP TABLE** <NOMETABELA>;



```
-- Comando para remover a tabela CURSO
```

```
DROP TABLE CURSO;
```

- Cuidados ao manipular estrutura de tabelas relacionadas.

— Criação e alteração de tabelas relacionadas

- Criação inicial das tabelas sem chave estrangeira (linhas 1 a 12).
- Modificação da tabela CURSO para adicionar a restrição de chave estrangeira (linha 14).

— Criação e alteração de tabelas relacionadas

Cuidados ao manipular tabelas relacionadas

- A criação de uma chave estrangeira estabelece uma dependência entre tabelas.
- O SGBD prioriza a integridade dos dados, inibindo operações que possam gerar inconsistência.
- Tentativa de remover tabela NIVEL com dependência gera erro.
- A remoção em cascata (CASCADE) permite a remoção automática de dependências.

— Manipulação de linhas nas tabelas

- Termo "manipulação" refere-se a operações de inserção, atualização e eliminação de dados.
- Mapeamento dos comandos SQL para CRUD:

Create: INSERT

Read: SELECT

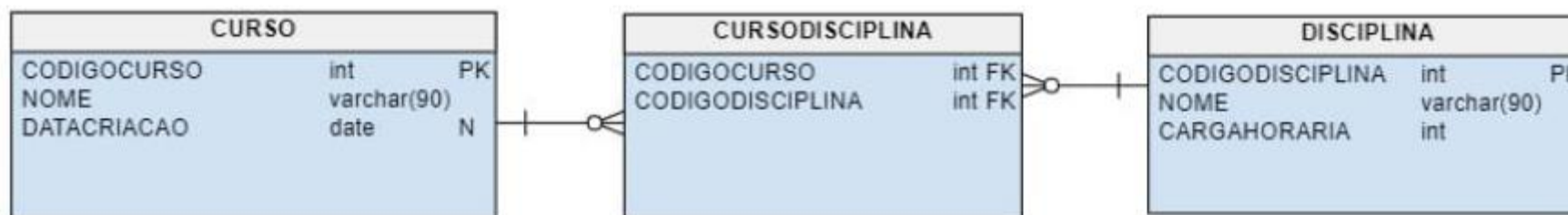
Update: UPDATE

Delete: DELETE

- Comandos SQL para Manipulação de Dados (DML): usados para inserir, modificar e remover dados.

— Manipulação de linhas nas tabelas

Modelo para os exemplos



Tabelas CURSO, CURSODISCIPLINA e DISCIPLINA.

— Inserção de linhas em tabela

- É realizada com o auxílio do comando INSERT da SQL.
- Sintaxe **básica**.



```
INSERT INTO <NOMETABELA> (COLUNA1, COLUNA2,...,COLUNAn) VALUES (VALOR1, VALOR2,...,VALORn);
```

— Mecanismo de chave primária em ação

- SGBD mantém integridade dos dados ao longo do ciclo de vida do banco de dados.
- Tentar inserir um registro com valor já existente na chave primária gera um erro.
- Valores duplicados em chave primária são identificados, evitando inconsistências.
- Chave primária é obrigatória e deve ser única.
- Tentativa de inserir registro sem valor para chave primária resulta em erro de valor nulo.
- SGBD garante consistência e validação dos dados durante as operações de inserção.

— Atualização de linhas em tabela

- Comando UPDATE é usado para atualizar linhas em uma tabela.
- Sintaxe **básica**.

```
UPDATE <NOMETABELA>
```

```
SET COLUNA1=VALOR1, COLUNA2=VALOR2,...,COLUNAn=VALORn
```

```
WHERE <CONDIÇÃO>;
```


— Atualização de linhas em tabela


- <NOMETABELA> representa a tabela a ser atualizada.
- Lista de colunas e valores a serem atualizados.
- Condição lógica (opcional) define o conjunto de linhas a ser alterado.
- Pode-se usar condições para especificar linhas a serem atualizadas.
- Atualizações podem ser feitas com base em expressões ou fórmulas.

— Atualização de coluna chave primária

- Cautela ao alterar o valor de uma coluna com papel de chave primária.
- Possível quando não há vínculo na tabela relacionada.
- Importância de manter integridade dos dados.
- Tentativa de UPDATE para alterar CODIGOCURSO para 10 onde CODIGOCURSO é 1.
- Erro devido à existência de registros na tabela relacionada CURSODISCIPLINA.
- Solução: alterar a tabela CURSODISCIPLINA para permitir a atualização em cascata.

— Remoção de linhas em tabela

- O comando DELETE é utilizado para remover linhas de uma tabela no PostgreSQL.
- A sintaxe **básica** do comando DELETE envolve a especificação da tabela alvo após "DELETE FROM" e uma condição opcional para determinar quais linhas serão removidas, indicada pela cláusula WHERE.

A terminal window with a dark header bar containing three colored circles (red, yellow, green). The main area is light gray and contains two lines of SQL syntax: 'DELETE FROM <NOMETABELA>' and 'WHERE <CONDIÇÃO>;'.

```
DELETE FROM <NOMETABELA>
```

```
WHERE <CONDIÇÃO>;
```

— Transações em banco de dados

- Comandos SQL desde a criação de tabelas até operações de manipulação de dados no PostgreSQL.
- Em ambientes de produção, o SGBD gerencia várias requisições de diversas aplicações simultaneamente.
- O usuário de uma aplicação geralmente está mais preocupado com os resultados dos processos de negócio automatizados do que com os detalhes internos de como o SGBD gerencia os dados.

— Transações em banco de dados

- Sistema acadêmico - um aluno pode se inscrever em várias disciplinas, e espera-se que a inscrição seja bem-sucedida para todas as disciplinas escolhidas.
- SGBD - a inscrição em várias disciplinas envolve a execução de várias instruções de inserção de dados, com considerações especiais, como verificar a disponibilidade de vagas.
- O conceito de transação em sistemas de banco de dados refere-se a um programa em execução que representa uma unidade lógica de trabalho, garantindo que todas as operações sejam realizadas com sucesso ou revertidas em caso de falha.

— Transações em banco de dados

- Limites de transação: Begin Transaction e End Transaction no programa de aplicação.
- Transações: **leitura** (não atualiza) e **leitura-gravação** (atualiza o banco de dados).
- Modelo simplificado de banco de dados: coleção de itens nomeados.
- SGBDs multiusuários (possíveis problemas):
 - **Atualização perdida;**
 - **Atualização temporária;**
 - **Resumo incorreto;**
 - **Leitura não repetitiva.**

— Transações em banco de dados

- Processamento de transação: operações devem ser completadas com sucesso para gravação permanente.
- Em caso de **falha**, todas as operações são canceladas, rollback é executado.
- Commit: confirmação das atualizações após o sucesso da transação.
- Falhas possíveis: falha do computador, erro de transação, condições de exceção, falha de disco e catástrofes.

— Transações em banco de dados

Propriedades de uma transação

ACID significa:

- **Atomicidade:** transação completa ou não realizada.
- **Consistência:** transição de estado consistente no banco de dados.
- **Isolamento:** execução sem interferência de outras transações.
- **Durabilidade:** mudanças persistem após confirmação da transação.

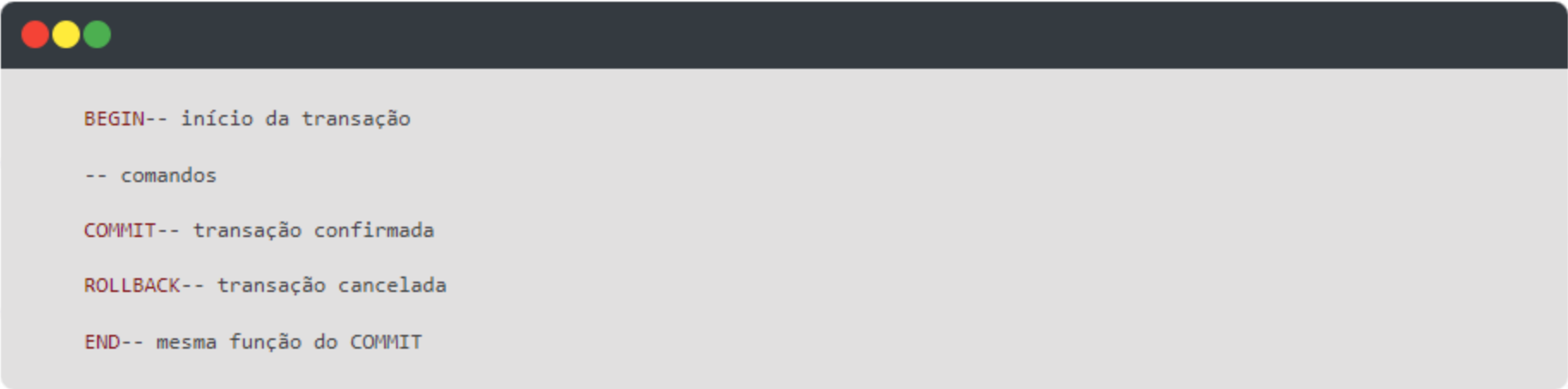
— Transações em banco de dados

Estados de uma transação

- Estados da transação: ativo, parcialmente confirmado, confirmado, falha, confirmado (saída do sistema).
- Recuperação de falhas: SGBD mantém log com operações de transação e informações sobre valores antigos e novos.
- Log de transação: registra operações que afetam itens do banco de dados, incluindo sucesso ou aborto da transação.

— Transações no Postgresql

Estrutura de uma transação no PostgreSQL:



```
BEGIN-- início da transação  
  
-- comandos  
  
COMMIT-- transação confirmada  
  
ROLLBACK-- transação cancelada  
  
END-- mesma função do COMMIT
```

— Um pouco mais sobre atualização temporária

- Execução paralela: consulta durante transação (linha 3) veria dados originais.
- Motivo: evitar atualização temporária; se transação for desfeita, UPDATE da linha 2 também seria revertido.

— Um pouco mais sobre transação de leitura


- Tipos de transação: **READ ONLY** (somente leitura) e **READ WRITE** (leitura-gravação).
- Comando SET TRANSACTION: especifica o tipo de transação no PostgreSQL.
- Mensagem de erro: SGBD informa impossibilidade de atualização em transação READ ONLY.
- Conceitos aprendidos: sequência de comandos em uma transação, geração implícita de transação no PostgreSQL, gerenciamento de transações com comandos específicos.

Consultas em uma tabela no postgresql



— Estrutura básica de um comando select

- Comando SELECT: exibe dados resultantes de uma consulta.
- Dados exibidos: colunas físicas, colunas calculadas, expressões e funções.
- Sintaxe **básica**:



```
SELECT COLUNA1 [[AS] APELIDOCOLUNA1],  
       COLUNA2 [[AS] APELIDOCOLUNA2],  
       ...  
       COLUNAN [[AS] APELIDOCOLUNAN]  
FROM TABELA;
```

— Estrutura básica de um comando select

Sintaxe simplificada

Entendimento dos exemplos ao longo do módulo.



Sintaxe complexa

Envolvendo várias cláusulas e recursos para consultas mais complexas.

— Estrutura básica de um comando select

- Versatilidade do SELECT: pode ser usado de diferentes formas para obter o mesmo resultado.
- Cláusula SELECT: realiza a operação de projeção na Álgebra Relacional.
- Exibição de todas as colunas: `SELECT * FROM TABELA;` ou `TABLE tabela` (forma simplificada no PostgreSQL).

— Funções de data e hora

Dados representativos de datas:

- **current_date**: retorna a data de hoje;
- **current_time**: retorna a hora do dia;
- **current_timestamp**: retorna data e hora;
- **extract (campo from fonte)**: obtém subcampos de data e hora.

— Funções de data e hora

Exibindo o nome do dia da semana

- Utilização do comando CASE para lógica condicional.
- Cada linha com cláusula WHEN avalia expressão e retorna o dia da semana em texto.
- Equivalente ao comando IF: em outras linguagens.

— Funções de data e hora

Calculando idade e faixa etária

- Uso da função AGE na linha 3 para retornar frase representativa da idade.
- Uso da função EXTRACT na linha 4 para obter a idade real.
- Utilização do comando CASE para avaliar a idade e determinar a faixa etária.
- Cada linha com cláusula WHEN retorna a faixa etária correspondente.
- Resultado da consulta: nome, idade e faixa etária dos alunos.

— Funções de resumo ou de agregação

Funções úteis para obter resumo dos dados de alguma tabela.

- **COUNT(*)** - número de linhas da consulta
- **MIN(COLUNA/EXPRESSÃO)** - menor de uma coluna ou expressão
- **AVG(COLUNA/EXPRESSÃO)** - valor médio da coluna ou expressão
- **MAX(COLUNA/EXPRESSÃO)** - maior valor de uma coluna ou expressão
- **SUM(COLUNA/EXPRESSÃO)** - soma dos valores de uma coluna ou expressão
- **STDDEV(COLUNA/EXPRESSÃO)** - desvio padrão dos valores de uma coluna ou expressão
- **VARIANCE(COLUNA/EXPRESSÃO)** - variância dos valores de uma coluna ou expressão

— Funções de resumo ou de agregação

Listando resumos em uma linha

Consultas para obter quantidades de cursos, disciplinas e alunos.

Formato individual:

CURSO

```
SELECT COUNT(*) NCURSOS  
FROM CURSO;
```

DISCIPLINA

```
SELECT COUNT(*)  
NDISCIPLINAS FROM  
DISCIPLINA;
```

ALUNO

```
SELECT COUNT(*) NALUNOS  
FROM ALUNO;
```

— Criando tabela a partir de consulta

Comando CREATE TABLE <CONSULTA>.

```
1 CREATE TABLE TTESTE AS
2 SELECT NOME,
3        EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) AS "Idade do Aluno",
4        CASE WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) <=20 THEN '1. até 20 anos'
5              WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 21 AND 30 THEN '2. 21 a 30 anos'
6              WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 31 AND 40 THEN '3. 31 a 40 anos'
7              WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 41 AND 50 THEN '4. 41 a 50 anos'
8              WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) BETWEEN 51 AND 60 THEN '5. 51 a 60 anos'
9              WHEN EXTRACT(YEAR FROM AGE(DTNASCIMENTO)) > 60 THEN '6. mais de 60 anos'
10       END AS "Faixa Etária"
11 FROM ALUNO;
```

— Criando *view* a partir de consulta

- Comando: `CREATE VIEW <NOME_VIEW> AS <CONSULTA>;`
- A view encapsula a complexidade da consulta SQL.
- Ao executar `SELECT * FROM VTESTE`, o SGBD realiza a consulta associada à view.

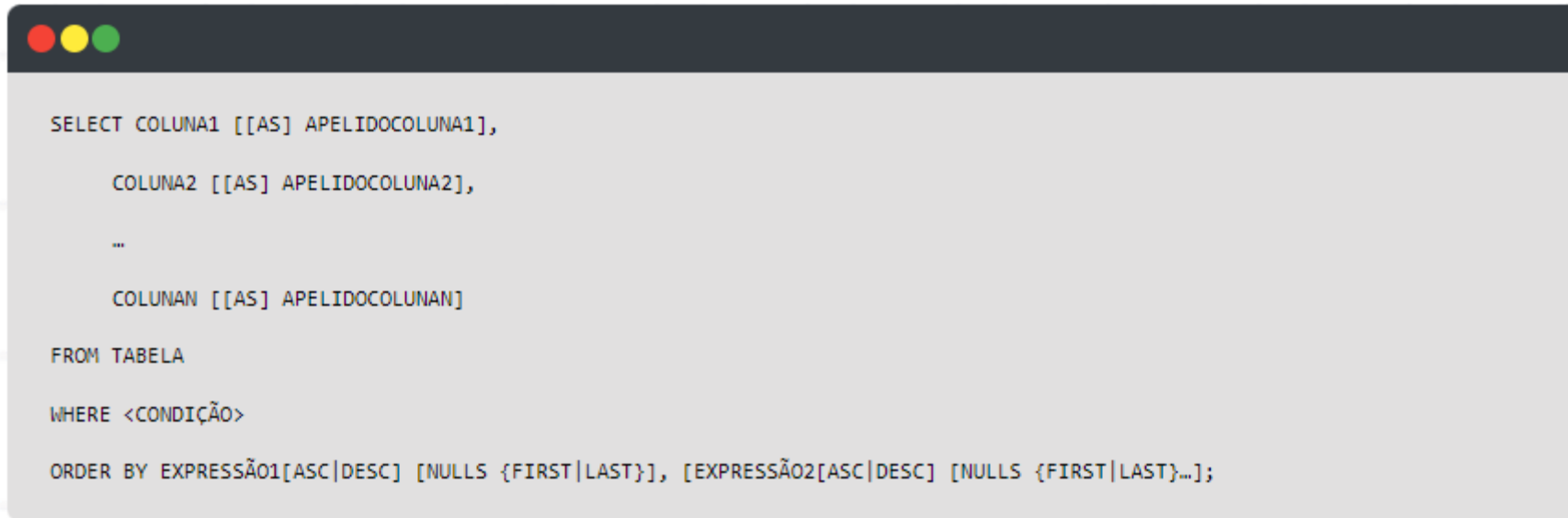
— Cláusula where e operadores da SQL

É possível criar a tabela e algumas linhas, usando o script. É necessário estar conectado ao PostgreSQL e acessando algum *database* criado.

```
1 CREATE TABLE ALUNO (  
2     CODIGOALUNO int NOT NULL,  
3     NOME varchar(90) NOT NULL,  
4     SEXO char(1) NOT NULL,  
5     DTNASCIMENTO date NOT NULL,  
6     EMAIL varchar(30) NULL,  
7     CONSTRAINT ALUNO_pk PRIMARY KEY (CODIGOALUNO));  
8  
9 INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (1, 'JOSÉ FRANCISCO TERRA', 'M', '28/10/1989', 'JFT@GMAIL.COM');  
10 INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (2, 'ANDREY COSTA FILHO', 'M', '20/10/1999', 'ANDREYCF@HOTMAIL.COM');  
11 INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (3, 'PATRÍCIA TORRES LOUREIRO', 'F', '20/10/1980', 'PTORRES@GMAIL.COM');  
12 INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (4, 'CARLA MARIA MACIEL', 'F', '20/11/1996', NULL);  
13 INSERT INTO ALUNO (CODIGOALUNO, NOME, SEXO, DTNASCIMENTO, EMAIL) VALUES (5, 'LEILA SANTANA COSTA', 'F', '20/11/2001', NULL);
```


— Recuperando dados com SELECT/FROM/WHERE/ORDER BY

Sintaxe **básica** para o comando SELECT, com o uso das cláusulas WHERE e ORDER BY



```
SELECT COLUNA1 [[AS] APELIDOCOLUNA1],  
       COLUNA2 [[AS] APELIDOCOLUNA2],  
       ...  
       COLUNAN [[AS] APELIDOCOLUNAN]  
FROM TABELA  
WHERE <CONDIÇÃO>  
ORDER BY EXPRESSÃO1[ASC|DESC] [NULLS {FIRST|LAST}], [EXPRESSÃO2[ASC|DESC] [NULLS {FIRST|LAST}...];
```

— Recuperando dados com SELECT/FROM/WHERE/ORDER BY

- Propósito do **SELECT**:
 - Declarar colunas da consulta.
 - FROM especifica tabela alvo da consulta.
 - WHERE realiza a operação de seleção (restrição da Álgebra Relacional).
- Condições envolvem operadores relacionais: <, <=, >, >=, =, <> (ou !=).

— Recuperando dados com SELECT/FROM/WHERE/ORDER BY

Operadores **relacionais** na cláusula WHERE:

Operador	Significado
<	menor
<=	menor ou igual a
>	maior
>=	maior ou igual a
=	igual
<> ou !=	> diferente

— Recuperando dados com SELECT/FROM/WHERE/ORDER BY

Operadores **lógicos** na cláusula WHERE:

Operador	Significado
AND	conjunção
OR	disjunção
NOT	negação

— Recuperando dados com o uso do operador IN

Pode ser utilizado em consultas que envolvam comparações usando uma lista de valores.

— Recuperando dados com o uso do operador BETWEEN

Verifica se determinado valor encontra-se no intervalo entre dois valores.

— Recuperando dados com o uso do operador LIKE

- Uso do [NOT] LIKE: realiza buscas em cadeias de caracteres.
- **Símbolos especiais:**
 - **_ (underline)** - para ignorar qualquer caractere específico;
 - **% (percentual)** - para ignorar qualquer padrão.

— Recuperando dados com o uso do operador NULL

Coluna opcional:

- Possibilidade de alguns registros não possuírem valor para a coluna.
- Valor pode ser "desconhecido" ou "não aplicável".

Teste de valor cadastrado:

- Expressão: COLUNA IS NOT NULL.
- Verifica se a coluna possui valor cadastrado (não é NULL).

— Recuperando dados usando ordenação dos resultados

Especifica critérios de ordenação para organizar os resultados de uma consulta.

Consultas com GROUP BY e HAVING

- Código utilizado para exibir todo o conteúdo da tabela:

```
1 SELECT *  
2 FROM FUNCIONARIO;
```

- Resultado:

	123 codigofuncionario	nome	cpf	sexo	dtascimento	123 salario
1	1	ROBERTA SILVA BRASIL	[NULL]	F	1980-02-20	7000.0
2	2	MARIA SILVA BRASIL	[NULL]	F	1988-09-20	9500.0
3	3	GABRIELLA PEREIRA LIMA	[NULL]	F	1990-02-20	6000.0
4	4	MARCOS PEREIRA BRASIL	[NULL]	M	1999-02-20	6000.0
5	5	HEMERSON SILVA BRASIL	[NULL]	M	1992-12-20	4000.0

— Grupo de dados

- Projetando consultas com agrupamento de dados.
- Utilização dos comandos GROUP BY e HAVING.
- Associação com funções de resumo: SUM, AVG, MIN, MAX.
- Funções representam soma, média, mínimo e máximo.
- Consultas úteis para construir relatórios e análises gerenciais.
- Os valores podem formar grupos sobre os quais podemos ter interesse em recuperar dados.

— Grupo de dados com GROUP BY

- Usada para exibir resultados de consulta de acordo com um grupo especificado.
- Declarada após FROM ou WHERE, caso exista.
- Uso com funções de agregação: associada a funções como COUNT, MIN, MAX, AVG.

— Grupo de dados com GROUP BY e HAVING

Cláusula HAVING:

- Utilizada para filtrar com base em cálculos de funções de agregação.
- Situações em que a cláusula WHERE não é adequada.
- Objetivo: Retornar registros com quantidade > 1.
- Quantidade é calculada por **função de agregação**.

Consulta com várias tabelas no PostgreSQL



— Operação de junção de tabelas

- Fundamental no modelo relacional de bancos de dados.
- Resultado é um subconjunto do produto cartesiano entre as tabelas.
- Condição de junção especificada por uma condição de igualdade entre colunas.

— Operação de junção de tabelas

Tipos de junção:

Interna (INNER JOIN)

Retorna apenas as linhas que têm correspondência nas duas tabelas.

Externa (OUTER JOIN)

Retorna todas as linhas quando há uma correspondência em uma das tabelas, e as linhas correspondentes da outra tabela.

— Operação de junção de tabelas

Junção externa pode ser:

- **LEFT OUTER JOIN** (ou **LEFT JOIN**);
- **RIGHT OUTER JOIN** (ou **RIGHT JOIN**);
- **FULL OUTER JOIN** (ou **FULL JOIN**).

— Operação de produto cartesiano

- Originada da operação de produto cartesiano.
- SGBD combina cada linha da tabela CURSO com cada registro da tabela NIVEL.
- Quantidade de colunas na tabela resultante é a soma das colunas das tabelas envolvidas.

— Operação de produto cartesiano

Interpretação de linhas em uma tabela:

- Cada linha representa um fato registrado no banco de dados.
- Correspondência à realidade do contexto do negócio modelado.

— Operação de produto cartesiano

Junção interna

Tipo específico de operação de junção que retorna apenas as linhas com correspondências em ambas as tabelas.

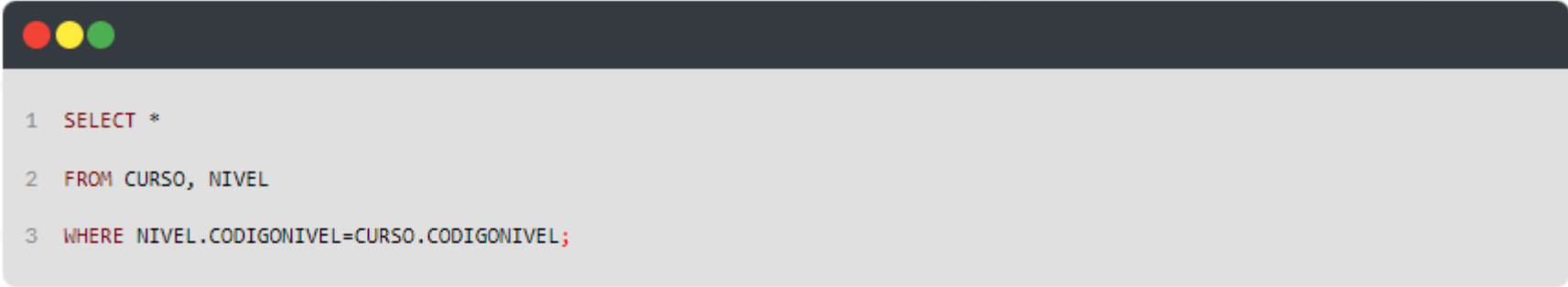
Sintaxe **básica**:

```
SELECT *  
FROM TABELA1 [INNER] JOIN TABELA2 ON (CONDIÇÃOJUNÇÃO) [USING (COLUNA_DE_JUNÇÃO)]
```

— Operação de produto cartesiano

Junção interna

Comando para recuperação das linhas corretas:



```
1 SELECT *  
2 FROM CURSO, NIVEL  
3 WHERE NIVEL.CODIGONIVEL=CURSO.CODIGONIVEL;
```

— Operação de produto cartesiano

Junção interna

- Declaração da cláusula de junção entre as tabelas.
- Preposição "ON" e condição da junção (geralmente, uma igualdade envolvendo a chave primária e a chave estrangeira).

— Operação de produto cartesiano

Junção externa

Inclusão de todos os registros da tabela CURSO:

- Uso da cláusula LEFT JOIN (junção à esquerda).

```
1 SELECT C.CODIGOCURSO, C.NOME,  
2        N.CODIGONIVEL, N.DESCRICAO  
3 FROM CURSO C LEFT JOIN NIVEL N ON (N.CODIGONIVEL=C.CODIGONIVEL) ;
```

— Operação de produto cartesiano

Junção externa

- Resultado contém todos os registros da tabela declarada à esquerda, mesmo que não haja correspondência na tabela da direita.
- Em casos sem correspondência, os resultados são retornados com valor NULL.

— Operação de produto cartesiano

Junção externa

Uso da cláusula RIGHT JOIN (junção à direita):

```
1 SELECT C.CODIGOCURSO, C.NOME,  
2       N.CODIGONIVEL, N.DESCRICAO  
3 FROM CURSO C RIGHT JOIN NIVEL N ON  
4 (N.CODIGONIVEL=C.CODIGONIVEL);
```

— Subconsultas

- Consulta sobre o resultado de outra consulta.
- Existem dois tipos:

Aninhadas

Consulta externa realiza teste com dados originados da consulta interna.

Correlatas

Subconsulta utiliza valores da consulta externa.

Subconsulta é executada para cada linha da consulta externa.

— Subconsultas

Subconsultas aninhadas

- Obtenção de dados que dependem do resultado de uma ou mais consultas internas.
- Criação de condição na cláusula WHERE, envolvendo o resultado da subconsulta em algum tipo de teste.

— Subconsultas

Subconsultas correlatas

- Tipo especial de consulta aninhada.
- Obtenção de dados que dependem do resultado de uma ou mais consultas internas.
- Criação de condição na cláusula WHERE, envolvendo o resultado da subconsulta em algum tipo de teste.

— Subconsultas

Consulta correlacionada com uso de [NOT] EXISTS

- Uso do operador EXISTS em uma consulta correlacionada.
- O operador EXISTS testa a existência de linha(s) retornada(s) por alguma subconsulta.

— Operadores de conjunto

- Combinam resultados de diversas consultas em um único conjunto de dados.
- Incluem: **UNION**, **INTERSECT** e **EXCEPT**.
- Cada tabela pertence a um banco de dados diferente e um domínio de aplicação diferente.

— Consultas com o operador UNION

- Serve para consolidar linhas resultantes de consultas.
- Todas as consultas envolvidas devem possuir a mesma quantidade de colunas e compatibilidade de tipo de dados.
- Linhas repetidas são eliminadas do resultado.

— Consultas com o operador INTERSECT

- Exibe linhas que aparecem em ambos os resultados das consultas envolvidas.
- Todas as consultas devem possuir a mesma quantidade de colunas e compatibilidade de tipo de dados.
- Linhas repetidas são eliminadas do resultado.

— Consultas com o operador EXCEPT

- Implementa a operação de subtração da Teoria dos Conjuntos.
- Exibe linhas que aparecem em uma consulta e não aparecem na outra.
- Todas as consultas devem possuir a mesma quantidade de colunas e compatibilidade de tipo de dados.
- Linhas repetidas são eliminadas do resultado.
- O operador de subtração possui a seguinte forma geral:
CONSULTASQL EXCEPT [ALL|DISTINCT] CONSULTASQL