Banco de Dados Relacionais e Não Relacionais

Bancos de Dados Relacionais

Um banco de dados relacional (ou RDB, do inglês *Relational Database*) é um tipo de banco de dados que organiza e armazena informações em tabelas estruturadas.

- Os dados são armazenados em tabelas, compostas por linhas e colunas.
- Cada linha representa uma instância ou registro, enquanto cada coluna representa um atributo ou característica do dado.
- Podemos pensar neles como uma planilha Excel.

Exemplo: Tabela Clientes

ID_Cliente (PK)	Nome	Email	Telefone
1	João Silva	joao@email.com	99999-1234
2	Maria Oliveira	maria@email.com	98888-5678
3	Pedro Santos	pedro@email.com	97777-9101

Design de Bancos de Dados Relacionais

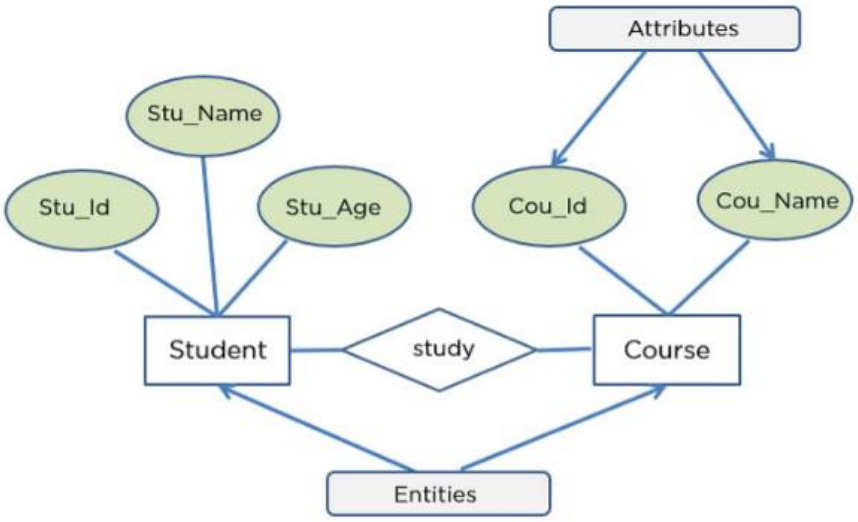
Modelo Conceitual:

- É a etapa inicial e mais abstrata do design de banco de dados.
- Focado em representar os dados e suas relações de maneira independente de qualquer sistema ou tecnologia específica.
- Representa o que será armazenado no banco de dados, com base nos requisitos do negócio.
- **Ao criar um modelo conceitual, é frequentemente útil visualizá-lo em um diagrama de entidade-relacionamento (ER), uma ferramenta padrão para visualizar as relações entre várias entidades.**

    - **MER (Modelo Entidade-Relacionamento):**
        - É a base do modelo conceitual.
        - Define as entidades, atributos, relacionamentos e cardinalidades de forma abstrata e alinhada aos requisitos do negócio.
        - Não se preocupa com detalhes técnicos ou implementação, mas com o que será armazenado e como esses elementos se conectam.
        - **Esse modelo ajuda a organizar a estrutura do banco de dados.**

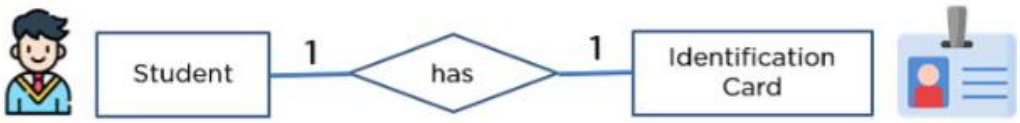
    - **DER (Diagrama Entidade Relacionamento):**
        - É a **representação gráfica do MER**, usada para ilustrar o modelo conceitual.
        - Representa o relacionamento entre entidades em um banco de dados.

- Facilita a compreensão visual das entidades, atributos e relacionamentos.



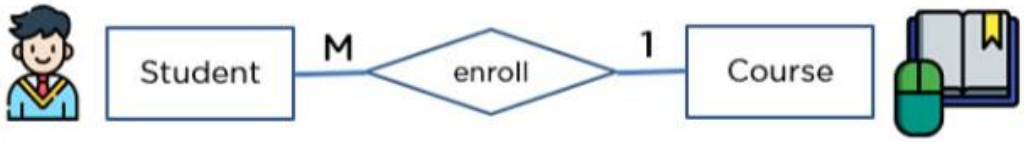
- **Elementos do DER**
  - **Entidades**: São os principais componentes do modelo conceitual e aparecem no DER como **retângulos**. Representam os objetos ou conceitos do mundo real que precisam ser modelados no banco de dados.
  - **Atributos:** Representam as características ou propriedades das entidades. São exibidos como **elipses** no DER, conectadas à entidade correspondente.
  - **Relacionamentos**: Definem as associações entre duas ou mais entidades e são representados como **losangos** no DER.

- **Cardinalidades:**
  - Representam a quantidade de ocorrências de uma entidade associadas a outra. Elas são definidas durante a criação do modelo conceitual
    - 1:1 (um para um): Cada registro de uma entidade está associado a exatamente um registro de outra. Por exemplo, um aluno tem apenas um cartão de identificação e um cartão de identificação é dado a uma pessoa.



    - 1:N (um para muitos): Um registro de uma entidade está associado a mais de um elemento da outra. Por exemplo, um cliente pode fazer muitos pedidos.

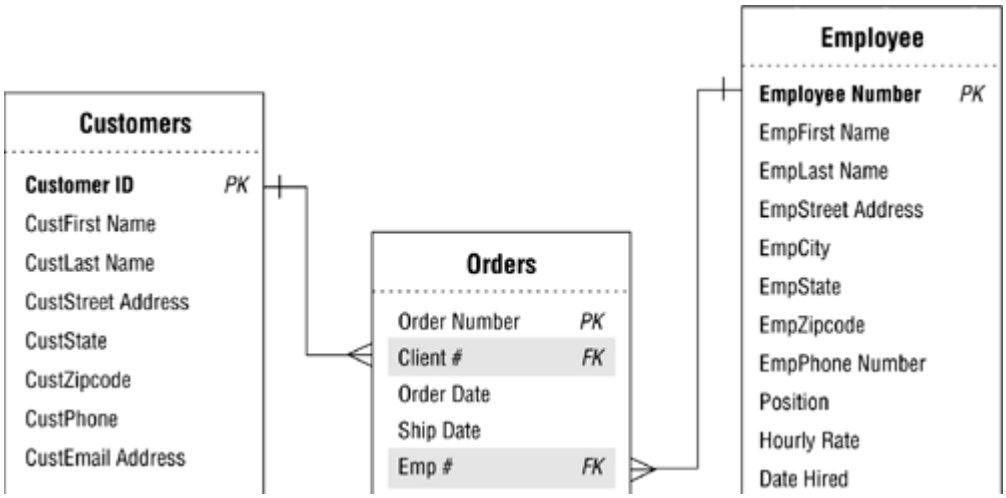
- N:1 (muitos para um): Quando mais de um elemento de uma entidade está relacionado a um único elemento de outra entidade. Por exemplo, os alunos têm que optar por um único curso, mas um curso pode ter muitos alunos



- N:N (muitos para muitos): Quando mais de um elemento de uma entidade é associado a mais de um elemento de outra entidade. Por exemplo, você pode atribuir um funcionário a muitos projetos e um projeto pode ter muitos funcionários.

Modelo Lógico:

- É a segunda etapa no processo de design de banco de dados, que traduz o modelo conceitual em uma estrutura mais detalhada e próxima da implementação prática, mas ainda independente de um SGBD específico.
- Elementos do Modelo Lógico:
  - **Tabelas:** As entidades do modelo conceitual são transformadas em tabelas no modelo lógico. Cada tabela é composta por linhas e colunas.
  - **Chaves:** As chaves são essenciais no modelo lógico para garantir a unicidade dos registros e as relações entre as tabelas.
    - **Chaves Candidatas**: Colunas (ou combinações de colunas) que podem identificar **de forma única** os registros de uma tabela. Exemplo: Em uma tabela de pessoas, tanto o CPF quanto a CNH podem ser chaves candidatas.
    - **Chave Primária**: Entre as chaves candidatas, uma é escolhida como **chave primária** (*Primary Key*). Ela identifica de forma única cada registro da tabela.
    - **Chaves Alternadas**: São as chaves candidatas não selecionadas como chave primária. Continuam sendo únicas, mas não são usadas como principal identificador.
    - **Chave Estrangeira**: É uma coluna que estabelece um relacionamento entre duas tabelas. Refere-se à chave primária de outra tabela.



Modelo Físico:

- O modelo físico é a última etapa no design de um banco de dados e define como o modelo lógico será implementado em um sistema de gerenciamento de banco de dados (SGBD) específico. Ele se concentra nos detalhes técnicos e práticos, como tipos de dados, índices, configurações de armazenamento e segurança.
- Ele traduz as tabelas e relacionamentos definidos no **modelo lógico** para um banco de dados real em um SGBD específico, como MySQL, PostgreSQL, Oracle, SQL Server, etc

Normalização em Banco de Dados

A normalização é um processo utilizado na modelagem de dados em bancos de dados relacionais para eliminar redundâncias, garantir integridade referencial e minimizar anomalias de inserção, atualização e exclusão. Esse processo divide os dados em tabelas menores e bem estruturadas, utilizando chaves primárias e estrangeiras para estabelecer relacionamentos entre elas.

Dependências Funcionais: são o fundamento central para compreender e aplicar a normalização em bancos de dados relacionais. Representam uma relação entre os atributos de uma tabela, na qual o valor de um atributo (ou de um conjunto de atributos) determina de forma única o valor de outro atributo (ou conjunto de atributos). Essa relação é essencial para garantir a integridade dos dados e eliminar redundâncias, facilitando a organização e o gerenciamento das informações no banco de dados.

CPF	Nome
123.456.789-00	João
987.654.321-00	Maria

CPF → Nome: Neste caso, nome depende funcionalmente de CPF, porque cada CPF único corresponde a um único Nome.

- **Dependência Funcional Parcial**: Ocorre em tabelas com **chave composta**, quando um atributo depende apenas de **uma parte da chave** e não da chave inteira.
    - **Chave Composta**: (VendaID, ProdutoID)

VendaID	ProdutoID	NomeProduto	Quantidade	PrecoUnitario
1	101	Camisa	2	50
2	102	Calça	1	100
3	101	Camisa	3	50

    - NomeProduto depende apenas de ProdutoID, não da chave composta (VendaID, ProdutoID).
    - Isso significa que NomeProduto não precisa de VendaID para ser identificado unicamente, pois ProdutoID já determina unicamente NomeProduto.
    - Isso é o que caracteriza uma dependência funcional parcial: NomeProduto depende parcialmente apenas de ProdutoID e não de toda a chave composta (VendaID, ProdutoID).

- **Dependência Funcional Transitiva**: Ocorre quando um atributo depende de outro atributo **indiretamente através de um terceiro**.

EmpregadoID	Nome	DepartamentoID	NomeDepartamento	LocalDepartamento
1	João	10	Vendas	São Paulo
2	Maria	20	RH	Rio de Janeiro
3	Pedro	10	Vendas	São Paulo

    - EmpregadoID: é a chave primária, e determina unicamente todos os outros atributos.
    - A dependência funcional transitiva ocorre aqui porque:
        - NomeDepartamento e LocalDepartamento são dependentes de DepartamentoID, e não diretamente de EmpregadoID.
        - No entanto, como DepartamentoID é determinado por EmpregadoID, temos uma dependência indireta (transitiva) de NomeDepartamento e LocalDepartamento com EmpregadoID.

- **Atributos Multivalorados e Compostos:**
    - **Atributos Multivalorados**: Atributos que podem conter mais de um valor para uma única entidade. Exemplo: Um aluno pode ter múltiplos números de telefone.

AlunoID	Nome	Telefones
1	João	(11) 99999-1234, (11) 3333-5678
2	Maria	(21) 88888-4321
3	Pedro	(31) 77777-7654, (31) 2222-9876

- **Atributos Compostos:** São atributos que podem ser divididos em subcomponentes menores que representam informações mais detalhadas. Exemplo: EnderecoCompleto pode ser dividido em Rua, Número, Cidade, Estado, CEP.

ClienteID	Nome	EnderecoCompleto
1	João	Rua A, 123, São Paulo, SP, 01000-000
2	Maria	Rua B, 456, Rio de Janeiro, RJ, 02000-000
3	Pedro	Av. C, 789, Belo Horizonte, MG, 03000-000

Formas Normais

O processo de normalização é dividido em etapas chamadas **formas normais** (NF - Normal Forms). Cada forma normal tem critérios específicos:

- **Primeira Forma Normal (1NF):**
  - Cada coluna deve conter apenas valores indivisíveis (sem atributos compostos ou multivalorados).
    - **Atributos multivalorados:**

ClienteID	Nome	Telefones
1	João	(11) 99999-1234, (11) 3333-5678
2	Maria	(21) 88888-4321
3	Pedro	(31) 7 -7654, (31) 2222-9876

    - Após a 1NF:

1. Tabela `Clientes`:

ClienteID	Nome
1	João
2	Maria
3	Pedro

2. Tabela `Telefones`:

TelefoneID	ClienteID	Telefone
1	1	(11) 99999-1234
2	1	(11) 3333-5678
3	2	(21) 88888-4321
4	3	(31) 77777-7654
5	3	(31) 2222-9876

- **Atributos compostos:**

ClienteID	Nome	Endereco
1	João	Rua A, 123, São Paulo, SP

- Tabela após 1NF:

ClienteID	Nome	Rua	Numero	Cidade	Estado
1	João	Rua A	123	São Paulo	SP

- **Segunda Forma Normal (2NF)**:
  - Eliminar dependências funcionais parciais.
  - A tabela deve estar na 1NF
  - Exemplo: Separar NomeProduto de uma tabela (VendaID, ProdutoID) para colocá-lo em uma tabela de Produtos, já que NomeProduto depende apenas de ProdutoID.

PedidoID	ProdutoID	NomeProduto	Quantidade	PrecoUnitario
1	101	Camisa	2	50
1	102	Calça	1	100
2	101	Camisa	3	50
2	103	Sapato	2	150

  - Após a 2NF:

PedidoID	ProdutoID	Quantidade
1	101	2
1	102	1
2	101	3
2	103	2

2. **Tabela `Produtos`** (contém atributos que dependem apenas de `ProdutoID`):

ProdutoID	NomeProduto	PrecoUnitario
101	Camisa	50
102	Calça	100
103	Sapato	150

- **Terceira Forma Normal (3NF)**:
    - Eliminar dependências funcionais transitivas.
    - A tabela deve estar na 2NF.
    - Exemplo: Mover NomeDepartamento e LocalDepartamento para uma tabela separada de Departamentos, já que essas colunas dependem de DepartamentoID e não diretamente de EmpregadoID.

FuncionarioID	Nome	DepartamentoID	NomeDepartamento	LocalDepartamento
1	João Silva	10	Vendas	São Paulo
2	Maria Costa	20	RH	Rio de Janeiro
3	Pedro Alves	10	Vendas	São Paulo

    - Após a 3NF:

FuncionarioID	Nome	DepartamentoID
1	João Silva	10
2	Maria Costa	20
3	Pedro Alves	10

2. Tabela `Departamentos` (contém atributos que dependem de `DepartamentoID`):

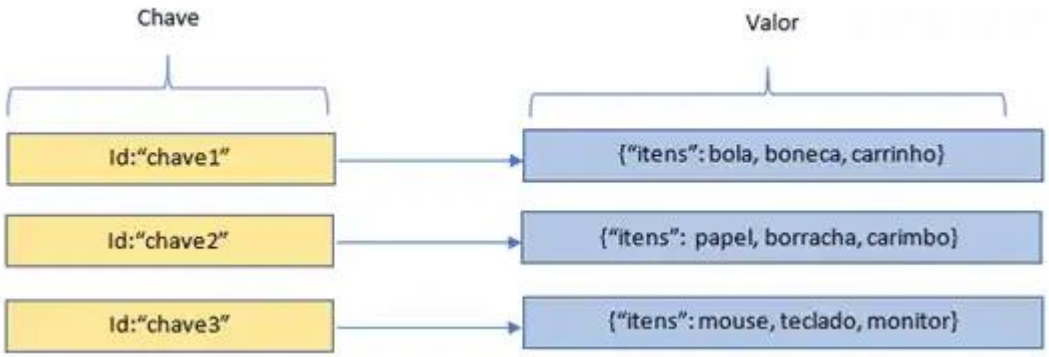
DepartamentoID	NomeDepartamento	LocalDepartamento
10	Vendas	São Paulo
20	RH	Rio de Janeiro

Bancos de Dados Não Relacionais: NoSQL

NoSQL, que significa "não apenas SQL", representa uma classe de bancos de dados que se distanciam do paradigma relacional tradicional, oferecendo maior flexibilidade e escalabilidade. Ele não usa o esquema de tabela de linhas e colunas encontrado na maioria dos sistemas de banco de dados tradicionais. Em vez disso, os bancos de dados não relacionais usam um modelo de armazenamento otimizado para os requisitos específicos do tipo de dados que está sendo armazenado. Na prática, "NoSQL" significa "banco de dados não relacionais", mesmo que muitos desses bancos de dados deem suporte a consultas compatíveis com SQL. Vamos explorar os principais tipos de bancos de dados NoSQL: chave-valor, documento, coluna larga e grafo.

1. Armazenamento Chave-Valor

Um banco de dados chave-valor armazena dados como pares de chave e valor, onde a chave é única e usada para localizar rapidamente os dados correspondentes. Esses bancos são ideais para cenários de alto desempenho e baixa complexidade. Exemplo: DynamoDB (AWS): Oferece armazenamento chave-valor escalável, amplamente utilizado para aplicações web.



2. Armazenamento de Documentos

O armazenamento de documento é um modelo de banco de dados NoSQL que organiza os dados como documentos, em vez de linhas e colunas (como nos bancos relacionais). Os dados armazenados são estruturas semelhantes a JSON (ou BSON), permitindo uma organização flexível e hierárquica. Cada registro é um documento que pode conter objetos aninhados.

Vantagens:

- Elimina a necessidade de normalização.
- Ideal para aplicações que exigem estruturação de dados flexível.

Exemplo: MongoDB: Muito popular para aplicações web, permite que dados estruturados e semiestruturados sejam armazenados e consultados com eficiência.

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

3. Bancos de Dados de Coluna Larga

Os **bancos de dados de coluna larga** (ou **wide-column databases**) são uma categoria de bancos de dados NoSQL projetados para armazenar grandes volumes de dados de forma distribuída e escalável. Eles são baseados em tabelas, mas diferem dos bancos de dados relacionais tradicionais porque organizam os dados em colunas em vez de linhas. Essa abordagem oferece alta eficiência para consultas que requerem leitura ou agregação de valores específicos em colunas. Os bancos de dados de coluna larga armazenam os dados em um formato onde:

- **Famílias de colunas** são coleções de colunas relacionadas que compartilham características semelhantes.
- Não é obrigatório que todas as linhas tenham o mesmo conjunto de colunas.

A estrutura é altamente flexível porque não exige que todas as linhas de uma tabela tenham as mesmas colunas. Exemplo: Apache Cassandra, excelente para armazenar dados em larga escala.

CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

4. Bancos de Dados de Grafo

Bancos de dados de grafo são uma categoria especializada de bancos de dados projetada para armazenar, gerenciar e consultar informações estruturadas em forma de grafos. Os bancos de dados de grafos são desenvolvidos especificamente para facilitar a criação e execução de aplicações que funcionam com conjuntos de dados altamente conectados. Eles usam nós para armazenar entidades de dados e bordas para armazenar os relacionamentos entre as entidades. Esses bancos são especialmente úteis para modelar e explorar dados interconectados, como redes sociais, sistemas de recomendação, análise de fraudes e redes de transporte. Exemplo: Neo4j e Amazon Neptune.