



**Laissa Beatriz Soares da Silva
Vinícius de Oliveira Cruz
Maria de Fátima Ramos Brandão
Jonathan Rosa Moreira**

DE CONJUNTOS A CONSULTAS

**CONSTRUINDO
PONTES ENTRE
A MATEMÁTICA
E O SQL.**



Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Licenciatura em Computação

De conjuntos a consultas [livro eletrônico] :
construindo pontes entre a matemática e o SQL /
Laissa Beatriz Soares da Silva...[et al.]. --
1. ed. -- Brasilia, DF : Ed. dos Autores, 2025.
PDF

Outros autores: Vinícius de Oliveira Cruz,
Maria de Fátima Ramos Brandão, Jonathan Rosa
Moreira.

Bibliografia.
ISBN 978-65-987795-2-8

1. Banco de dados - Gerenciamento - Programas
de computador 2. Ciência da computação 3. Linguagem
de programação para computadores 4. Matemática -
Estudo e ensino I. Silva, Laissa Beatriz Soares da.
II. Cruz, Vinícius de Oliveira. III. Brandão, Maria
de Fátima Ramos. IV. Moreira, Jonathan Rosa.

25-281597

CDD-005.75

EXPEDIENTE

Paulo Lima Júnior

Diretor de Planejamento e Acompanhamento Pedagógico das Licenciaturas
Universidade de Brasília

Ricardo Ruviaro

Diretor do Instituto de Ciências Exatas
Universidade de Brasília

Alba Cristina Magalhães Alves de Melo

Chefe do Departamento de Ciência da Computação
Universidade de Brasília

Professor Li Weigang

Subchefe do Departamento de Ciência da Computação
Universidade de Brasília

Jorge Henrique Cabral Fernandes

Coordenador do Curso de Licenciatura em Computação
Universidade de Brasília

Edison Ishikawa

Vice-Cordenador do Curso de Licenciatura em Computação
Universidade de Brasília

Maria de Fátima Ramos Brandão

Docente do Curso de Licenciatura em Computação
Universidade de Brasília

Jonathan Rosa Moreira

Docente do Curso de Licenciatura em Computação
Universidade de Brasília



EDITORIA DISRUPTIVA



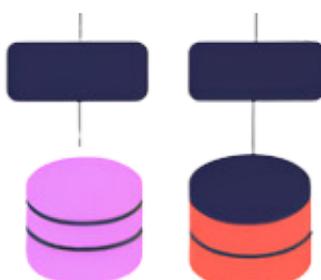
DE CONJUNTOS A CONSULTAS

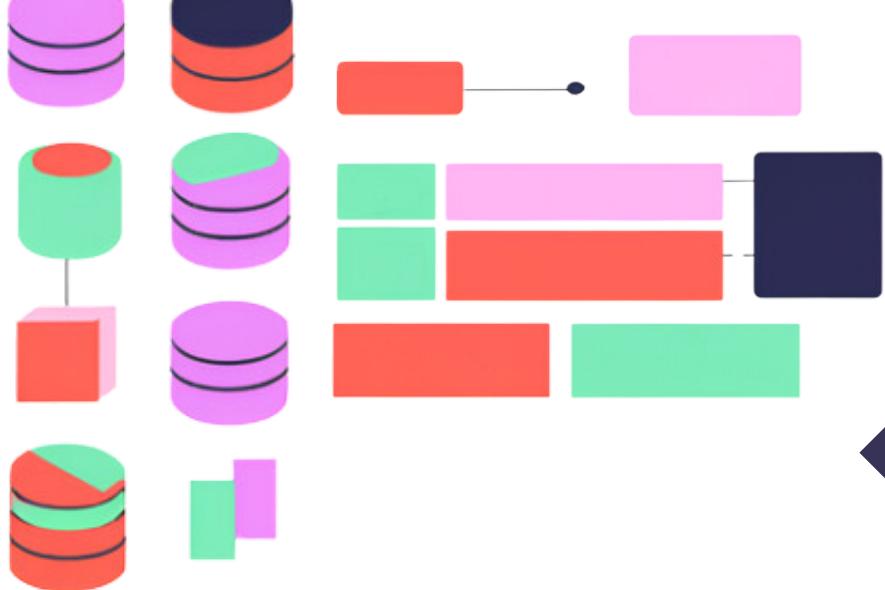
CONSTRUINDO
PONTES ENTRE
A MATEMÁTICA
E O SQL.



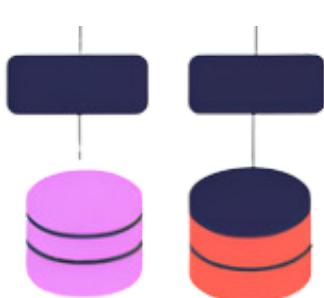
Sobre o projeto

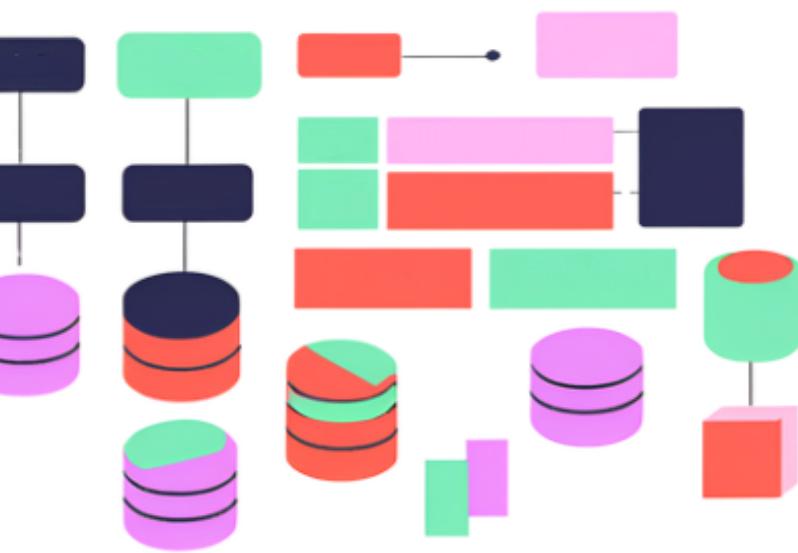
Este projeto foi desenvolvido para a componente curricular de **Produção de Material Didático em Computação** do curso de **Licenciatura em Computação** da Universidade de Brasília. "Produção de Material Didático" tem como objetivo principal desenvolver habilidades e competências para a elaboração e desenvolvimento de materiais didáticos e educativos voltados para o ensino de computação. Aborda-se o processo de construção de material didático, desde a concepção até a execução, considerando aspectos pedagógicos, metodológicos e legais, como o direito autoral e de avaliação. Além disso, a disciplina promove a produção de materiais que contribuam para a solução de problemas específicos de comunidades. Busca-se compreender as bases tecnológica, de comunicação e pedagógico-didática que suportam o material didático e aspectos de qualidade do processo de produção de material didático (seleção e avaliação).





O projeto visa oportunizar recursos pedagógicos que apoiem professores no ensino interdisciplinar de Computação e Matemática, alinhado às habilidades da BNCC, mais especificamente nos conteúdos relativos à Banco de Dados e Linguagem SQL bem como conceitos básicos de Teoria dos Conjuntos.





Sumário

Capítulo 1 - Introdução ao universo dos Conjuntos	01
Capítulo 2 - Conceitos Matemáticos em Ação	19
Capítulo 3 - Da Matemática ao Computador	29
Capítulo 4 - SQL - A Linguagem dos Dados	45
Capítulo 5 - Exercícios Práticos	73
Chegamos ao fim...	90



Olá, professor!

Este livro foi desenvolvido para auxiliar sua prática docente no ensino de Banco de Dados tanto na Educação Básica quanto na Educação Profissional Tecnológica.

A proposta é integrar os conceitos de Teoria de Conjuntos, da Matemática, com relações presentes no contexto de Banco de Dados, utilizando a linguagem SQL como ferramenta prática.

Vale destacar que o material não se aprofunda nos conceitos matemáticos, utilizando-os apenas como base para facilitar o entendimento das consultas em banco de dados. Por isso, os primeiros capítulos e exercícios são propostos apenas como retomada de conceitos matemáticos anteriores.



Ao longo do livro, criamos um canal de comunicação por meio das notas de sugestões posicionadas à esquerda da página. Nessas notas, apresentamos aspectos relacionados à prática docente, incluindo objetivos de ensino, objetivos de aprendizagem, além de sugestões de exercícios e metodologias.

Sinta-se à vontade para expandir o material, adicionando exemplos e exercícios conforme necessário.

Espero que você goste e aproveite ao máximo.

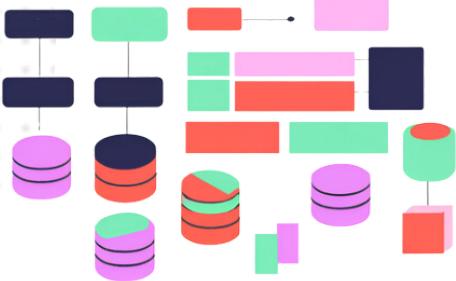
Boa aula! An icon of an open book with a single glowing lightbulb positioned above the center of the pages.



O ensino da Teoria de Conjuntos é fundamental para o desenvolvimento do pensamento lógico-matemático e da capacidade de abstração dos estudantes. Ele proporciona uma base conceitual sólida para diversas áreas da matemática e da computação.

Objetivo geral desse capítulo:

“Compreender e aplicar os conceitos da Teoria de Conjuntos como base para a organização, estruturação e manipulação de informações.”



CAPÍTULO 1

Introdução ao universo dos Conjuntos

1.1. O que são Conjuntos?

Para iniciar, precisa-se considerar três noções básicas da teoria de conjuntos. São elas: conjunto, elemento e pertinência entre um elemento e um conjunto.

- **Conjunto**

Na Matemática, um **Conjunto** é definido como uma coleção bem definida de objetos distintos que são agrupados com base em uma propriedade comum ou critério específico.



Nessa página,
temos como
objetivos
principais:

- Reconhecer e definir conjuntos e elementos;
- Diferenciar conjuntos e elementos, compreendendo a relação de pertinência.

Para isto,
professor, apoie-
se no recurso
visual e faça a
leitura da
sentença
matemática.

• Elemento

Um **elemento** é cada membro ou objeto que entra na formação do conjunto.

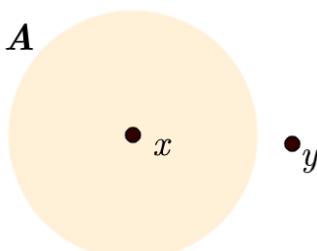
• Pertinência

Dado um conjunto A qualquer e um elemento x qualquer, dizemos que x pertence a A quando x é um dos elementos do conjunto A.

$$x \in A$$

Analogamente, se y não é um elemento do conjunto A, dizemos que y não pertence a A.

$$y \notin A$$





Aqui, é importante que o estudante entenda que conjunto universo é essencial porque define o limite do nosso estudo em um determinado contexto. Além disso, perceber que em um conjunto universo bem definido, poderíamos ter ambiguidades.

Uma boa estratégia é explicar o conjunto universo é usando exemplos do dia a dia, como o exemplo b).

Traga outros exemplos não numéricos, de preferência, para auxiliar.

CONJUNTO UNIVERSO

O conjunto universo é o conjunto que contém todos os elementos relevantes para um determinado contexto ou problema.

- **Exemplos:**

a) O conjunto universo relativo ao contexto dos números inteiros maiores ou iguais que 1 e menores que 10 é:

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

b) O conjunto universo no contexto dos meses do ano é:

$$U = \left\{ \begin{array}{l} Jan, Fev, Mar, \\ Abr, Mai, Jun, \\ Jul, Ago, Set, \\ Out, Nov, Dez \end{array} \right\}$$





O conhecimento da formalidade ajuda na leitura e escrita matemática uma vez que os estudantes que aprendem a notação formal conseguem interpretar textos matemáticos e resolver problemas com mais facilidade.

Além disso, em computação, o uso correto da notação é visto como uma boa prática, já que ajuda na compreensão de algoritmos e na escrita de código eficiente.

FORMALIDADE

Um conjunto é representado entre chaves {}, e os elementos podem ser descritos explicitamente (por enumeração) ou implicitamente (por propriedade).

Por exemplo:

- Por numeração:

$$A = \{0, 1, 2, 3, 4, \dots\}$$

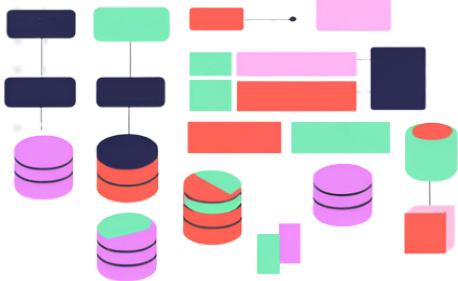
Lê-se: “A é o conjunto dos elementos de 0, 1, 2, 3, 4, infinitamente.”.

- Por propriedade:

$$B = \{x \mid x \text{ é um número natural}\}$$

Lê-se: “B é o conjunto dos elementos x, tal que x é um número natural”.





Professor, compreender como os números estão organizados ajuda os estudantes a desenvolverem um raciocínio estruturado, fundamental para a resolução de problemas.

Um programador precisa saber quando usar inteiros (int) ou números reais (float) para evitar erros em cálculos financeiros ou científicos.

CONJUNTOS NUMÉRICOS

Conjuntos Numéricos são **agrupamentos de números** que compartilham características em comum. Esses conjuntos ajudam a organizar e classificar os números de acordo com **suas propriedades**.

Números Naturais:

$$\mathbb{N} = \{0, 1, 2, 3, 4, 5, \dots\}$$

Números Inteiros:

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

Números Racionais:

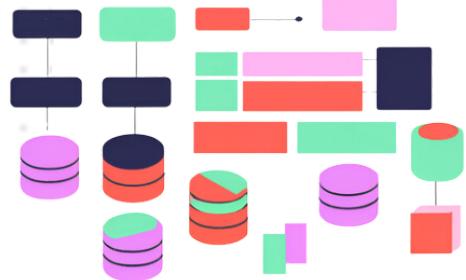
$$\mathbb{Q} = \left\{ \frac{a}{b} \mid a \in \mathbb{Z}, b \in \mathbb{Z}^* \right\}$$

Existem, também, o conjunto dos números irracionais e o conjunto dos números reais, entretanto nos limitaremos aos conjuntos acima.



No contexto de computação, a habilidade de identificar subconjuntos e verificar relações de inclusão é essencial para a organização, filtragem, análise e modelagem de informações. Por exemplo, relacionar dados como subconjuntos permite criar categorias e filtros para análises mais precisas.

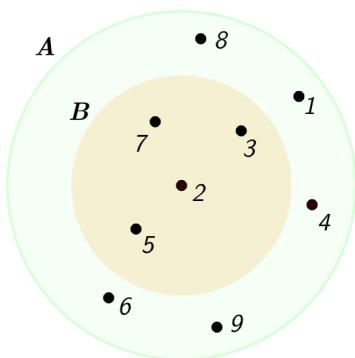
Professor, utilize as cores distintas para diferenciar os conjuntos e mostre que a propriedade geradora conjunto B, cria um subconjunto do conjunto A.



SUBCONJUNTOS

Um conjunto B é subconjunto de A se todo elemento de B pertence também a A.

Observe:



$$A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$B = \{x \mid x \text{ é primo menor que } 10\}$$

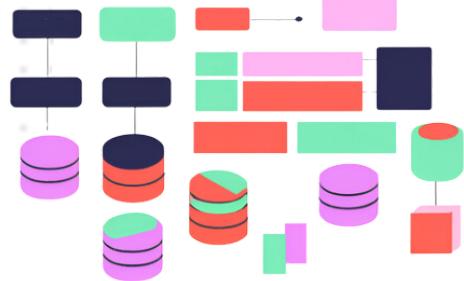
Portanto, dizemos que “B está contido em A” e/ou “B é subconjunto de A”.

$$B \subset A$$



Compreender e realizar operações entre conjuntos é, sem dúvida, uma das habilidades mais poderosas para o desenvolvimento da linguagem SQL. Isso porque, no coração do SQL, estão operações fundamentais entre conjuntos, como união, interseção, diferença e complemento, que permitem manipular, combinar e filtrar dados da melhor maneira possível.

Aqui, professor, deixe claro que o resultado de operações de conjuntos é um novo conjunto.



1.2. Operações básicas entre Conjuntos

As operações básicas entre conjuntos são operações fundamentais na Teoria dos Conjuntos que permitem **combinar, comparar e manipular conjuntos de forma estruturada**. Essas operações são essenciais para entender como diferentes conjuntos podem se relacionar entre si.

As principais operações básicas entre conjuntos incluem a **união**, a **interseção**, a **diferença** e o **complementar**.



Aqui, professor, é importante deixar evidente que as operações são base teórica para desenvolver habilidades de manipulação de dados em seguida.

É importante evidenciar que na operação de UNIÃO não existe elementos duplicados no conjunto resultante. Portanto, a operação de união REMOVE DUPLICATAS.

UNIÃO ENTRE CONJUNTOS

A **união entre conjuntos** é uma **operação** que resulta em um **novo conjunto** contendo todos os elementos que pertencem a **pelo menos um** dos conjuntos originais.

Sejam A e B dois conjuntos. A união entre A e B, denotada por $A \cup B$ é o conjunto definido como:

$$A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$$



UNIÃO ENTRE CONJUNTOS

- **Exemplos:**

a) Seja $A = \{1, 2, 3\}$ e $B = \{3, 4, 5\}$ então:

$$A \cup B = \{1, 2, 3, 4, 5\}$$



Utilize os exemplos a seguir para apoiar o entendimento de que no resultado não há elementos duplicados.

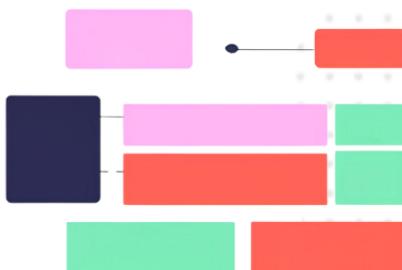
Além disso, deixe claro que $A \cup B$ é um conjunto.

b) Seja $A = \{\text{maçã, banana, laranja}\}$ e $B = \{\text{kiwi, banana, morango}\}$, então:

$$A \cup B = \{\text{maçã, banana, laranja, kiwi, morango}\}$$

c) Seja $A = \{\text{Brasil, Argentina, Chile}\}$ e $B = \{\text{França, Argentina, Japão}\}$, então:

$$A \cup B = \{\text{Brasil, Argentina, Chile, França, Japão}\}$$





Professor, mostre que na operação de **INTERSEÇÃO**, apenas os elementos que pertencem a ambos os conjuntos serão incluídos no conjunto resultante. Portanto, a operação de interseção RETÉM SOMENTE OS ELEMENTOS COMUNS entre os conjuntos.

INTERSEÇÃO ENTRE CONJUNTOS

A **interseção** entre conjuntos é uma **operação** que resulta em um **novo conjunto** contendo todos os elementos que **são comuns** a ambos os conjuntos originais.

Sejam A e B dois conjuntos. A interseção entre A e B, denotada por $A \cap B$ é o conjunto definido como:

$$A \cap B = \{x \mid x \in A \text{ e } x \in B\}$$



INTERSEÇÃO ENTRE CONJUNTOS

- **Exemplos:**



Utilize os exemplos a seguir para apoiar o entendimento de que no resultado foram mantidos apenas os elementos comuns aos dois conjuntos iniciais.

Além disso, deixe claro que $A \cap B$ é um conjunto.

- a) Seja $A = \{1, 2, 3\}$ e $B = \{2, 3, 4\}$ então:

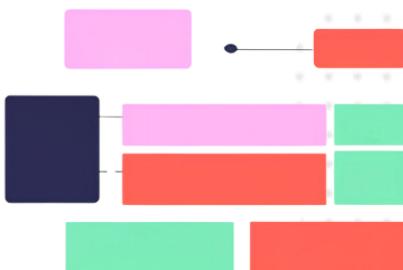
$$A \cap B = \{2, 3\}$$

- b) Seja $A = \{\text{maçã, banana, laranja}\}$ e $B = \{\text{kiwi, banana, morango}\}$, então:

$$A \cap B = \{\text{banana}\}$$

- c) Seja $A = \{\text{Brasil, Argentina, Chile}\}$ e $B = \{\text{França, Argentina, Japão}\}$, então:

$$A \cap B = \{\text{Argentina}\}$$





Na operação de DIFERENÇA entre dois conjuntos, o conjunto resultante contém somente os elementos que pertencem ao primeiro conjunto, mas não ao segundo.

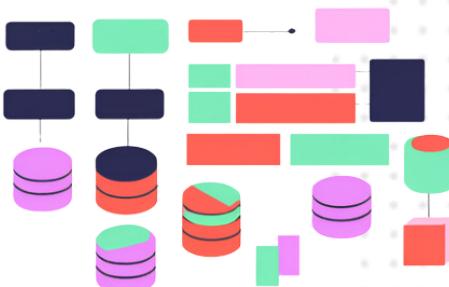
Portanto, a operação de diferença REMOVE OS ELEMENTOS DO PRIMEIRO CONJUNTO QUE ESTÃO NO SEGUNDO CONJUNTO.

DIFERENÇA ENTRE CONJUNTOS

A **diferença** entre conjuntos é uma operação que resulta em um **novo conjunto** contendo todos os elementos que **pertencem** a um conjunto, **mas não ao outro**.

Sejam A e B dois conjuntos. A diferença entre A e B, denotada por $A - B$ é definida como:

$$A - B = \{x \in A \mid x \notin B\}$$



DIFERENÇA ENTRE CONJUNTOS



Utilize os exemplos a seguir para apoiar o entendimento de que no resultado foram mantidos apenas os elementos de A que NÃO ESTÃO em B.

Além disso, deixe claro que A-B é um conjunto.

- **Exemplos:**

a) Seja $A = \{1, 2, 3, 5\}$ e $B = \{2, 3, 4\}$ então:

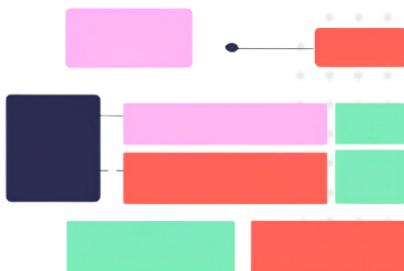
$$A - B = \{1, 5\}$$

b) Seja $A = \{\text{maçã, banana, laranja}\}$ e $B = \{\text{banana, morango}\}$, então:

$$A - B = \{\text{maçã, laranja}\}$$

c) Seja $A = \{\text{Ana, João, Carlos}\}$ e $B = \{\text{João, Maria}\}$, então:

$$A - B = \{\text{Ana, Carlos}\}$$





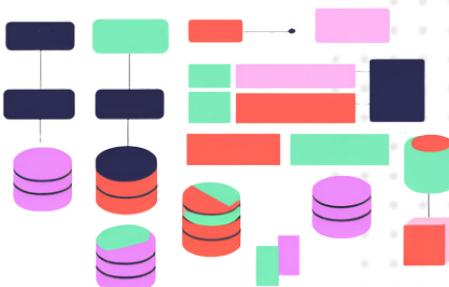
Na operação de **COMPLEMENTO**, os elementos do conjunto resultante são aqueles que pertencem ao conjunto universo, mas não pertencem ao conjunto original. Portanto, a operação de complemento **REMOVE OS ELEMENTOS DO CONJUNTO ORIGINAL DO CONJUNTO UNIVERSO.**

CONJUNTO COMPLEMENTAR

O **conjunto complementar** de outro é a operação que identifica todos os elementos que pertencem ao conjunto universal e que não pertencem ao conjunto em questão.

Seja U o conjunto universal e A um subconjunto de U . O complemento de A , denotado por A^c ou \bar{A} , é definido como:

$$A^c = \{x \in U \mid x \notin A\}$$



CONJUNTO COMPLEMENTAR

- **Exemplos:**



Utilize os exemplos a seguir para apoiar o entendimento de que no resultado foram mantidos apenas os elementos do universo que NÃO estão nos conjuntos iniciais.

Além disso, deixe claro que o resultado é um conjunto.

- a) Seja $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ e $A = \{2, 4, 6, 8, 10\}$ então:

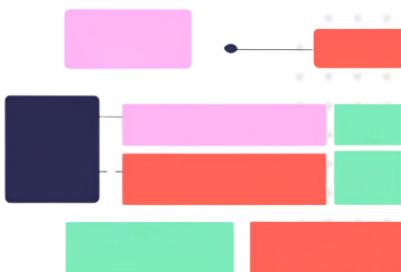
$$A^c = \{1, 3, 5, 7, 9\}$$

- b) Seja $U = \{\text{vermelho, azul, amarelo, verde, roxo}\}$ e $B = \{\text{vermelho, azul}\}$, então:

$$B^c = \{\text{amarelo, verde, roxo}\}$$

- c) Seja $U = \{\text{Ana, João, Carlos, Maria, Pedro}\}$ e $C = \{\text{João, Maria}\}$, então:

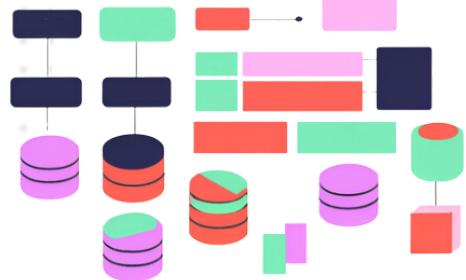
$$C^c = \{\text{Ana, Carlos, Pedro}\}$$





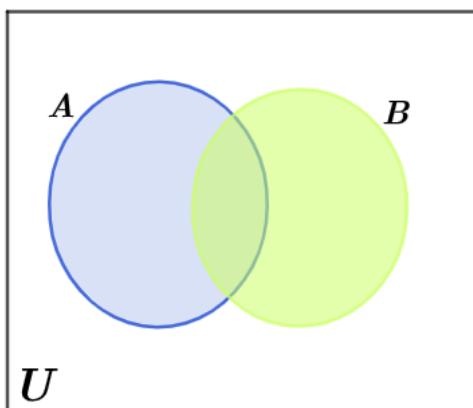
Explicar o diagrama de Venn para estudantes é extremamente importante, pois ele oferece uma visão visual e intuitiva das relações entre conjuntos, um conceito fundamental que se aplica ao processamento e manipulação de dados.

Ele oferece uma maneira simples de entender como dados podem ser combinados ou filtrados para resolver problemas como encontrar itens comuns entre conjuntos, por exemplo.



1.3. Representação Gráfica - Diagramas de Venn

Os **Diagramas de Venn** são representações gráficas utilizadas para mostrar relações entre conjuntos. Eles foram introduzidos por John Venn, um lógico britânico, em 1880. Esses diagramas facilitam a visualização de **operações e interseções** entre conjuntos, sendo assim amplamente utilizados.





Explicar os elementos do diagrama é fundamental porque ajuda os estudantes a entender como os conjuntos e suas relações são representados visualmente. Ao compreender esses elementos, os estudantes podem visualizar operações entre conjuntos de forma mais clara.

No próximo capítulo, faremos a aplicação dos diagramas às operações.

ELEMENTOS DO DIAGRAMA DE VENN

- **RETÂNGULO:** Muitas vezes, o diagrama inclui um retângulo que representa o Conjunto Universo (U)—o conjunto de todos os elementos em consideração.
- **CÍRCULOS/ELIPSES:** Cada círculo/elipse representa um conjunto específico.
- **ÁREAS:** As sobreposições ou regiões distintas mostram as relações entre os conjuntos.

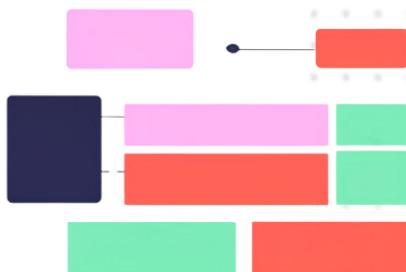
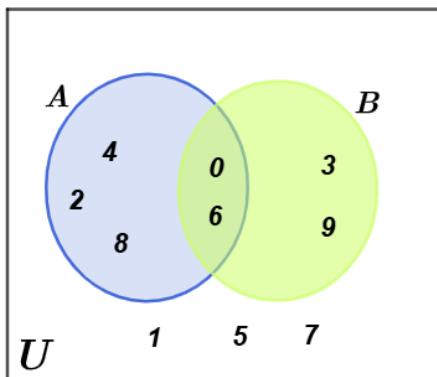


- **Exemplo:**

Seja o conjunto $U = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
considere $A = \{0, 2, 4, 6, 8\}$ e
 $B = \{0, 3, 6, 9\}$. Sendo assim, temos
o diagrama de Venn:



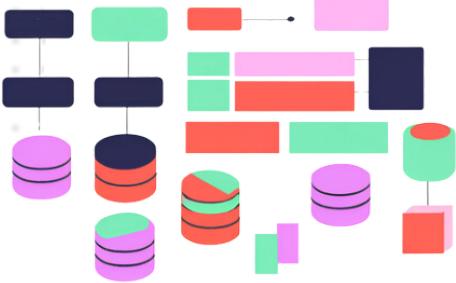
Nesse momento,
professor, tente
construir a
representação
com os
estudantes antes
de apresentar o
resultado.





Neste capítulo, conectaremos os conceitos anteriores por meio da visualização e da categorização de elementos em conjuntos, através do Diagrama de Venn.

Em seguida, serão propostos exercícios sobre conjunto, elemento, pertinência e representação gráfica de situações que envolvam conjuntos. Desta forma, sistematizaremos as habilidade basilares para a manipulação de dados.



CAPÍTULO 2

Conceitos Matemáticos em Ação

2.1. Operações de Conjuntos e o Diagrama de Venn

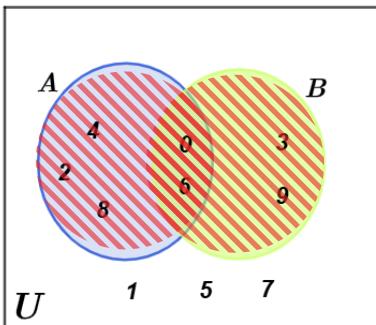
O Diagrama de Venn é uma ferramenta visual poderosa para compreender as operações entre conjuntos. Ao representar conjuntos como regiões sobrepostas, ele facilita a identificação de elementos comuns ou exclusivos, tornando conceitos abstratos mais concretos e intuitivos. Essa representação gráfica ajuda tanto no aprendizado inicial quanto na resolução de problemas.



OPERAÇÕES E O DIAGRAMA DE VENN

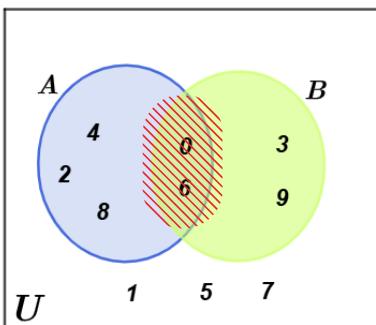
Estudantes de computação, especialmente os que têm uma lacuna em matemática ou estão começando sua formação, podem achar difícil conectar conceitos teóricos com aplicações práticas. O diagrama de Venn serve como uma ponte visual entre teoria de conjuntos e prática computacional, permitindo que vejam de maneira simples como essas operações abstratas se aplicam no contexto da máquina.

• UNIÃO ($A \cup B$)

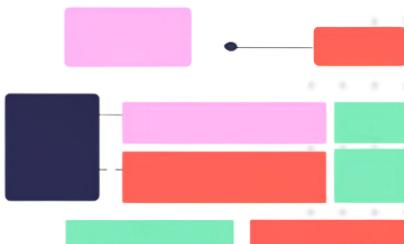


$$A \cup B = \{0, 2, 3, 4, 6, 8, 9\}$$

• INTERSEÇÃO ($A \cap B$)



$$A \cap B = \{6\}$$

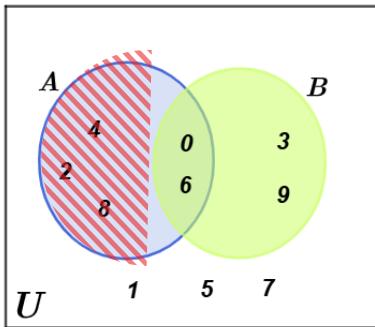


OPERAÇÕES E O DIAGRAMA DE VENN

- DIFERENÇA ($A - B$)

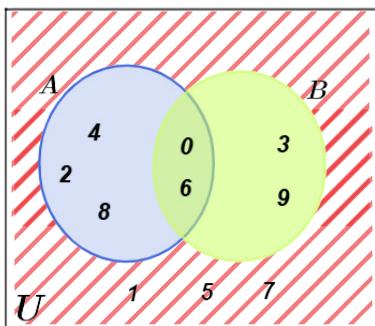


Aqui, professor, deixe claro que cada círculo no diagrama representa um conjunto, e as áreas de interseção, união, diferença e complemento mostram como esses conjuntos se relacionam.

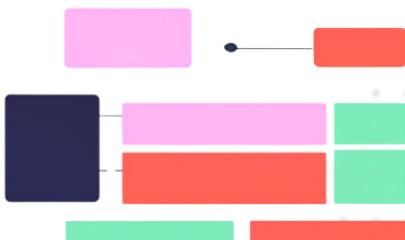


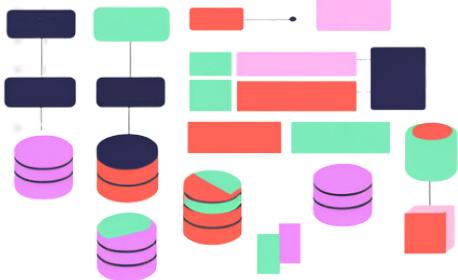
$$A - B = \{2, 4, 8\}$$

- COMPLEMENTO (A^c)



$$A^c = \{1, 3, 5, 7, 9\}$$





Professor, é hora de praticar. Até aqui falamos muito de teoria e conceitos densos para alguns.

Pensando nisso, essa seção irá propor uma sequência de 7 exercícios, nos quais o objetivo principal é o estudante desenvolver a habilidade de aplicar as propriedades das operações de conjuntos para resolver problemas envolvendo.

2.2. Aplicações Práticas dos Conjuntos

Os diagramas de Venn e o pertencimento de elementos a conjuntos permitem compreender operações como união, interseção e diferença de conjuntos, fundamentais na formulação de consultas estruturadas em SQL.

Por essa razão, esta seção abordará aplicações práticas da Teoria de Conjuntos com o objetivo de desenvolver essas habilidades de raciocínio lógico e matemático, facilitando futuramente a construção de consultas eficientes em bancos de dados.

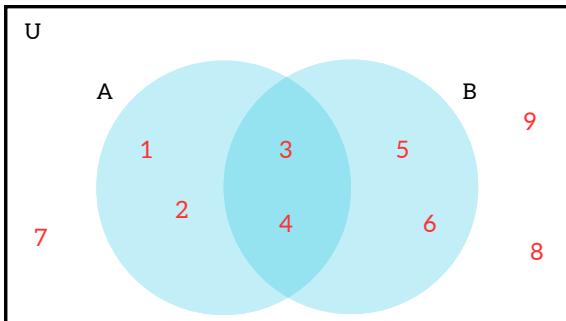
• Exercícios Propostos:

- 1) Dados os conjuntos: $U=\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $A=\{1,2,3,4\}$ e $B=\{3,4,5,6\}$. Preencha o diagrama a seguir e determine: $A \cup B$.



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!



$$(A \cup B) = \{1, 2, 3, 4, 5, 6\}$$

- a) Existem elementos que estão em A e B simultaneamente? Se sim, quais?

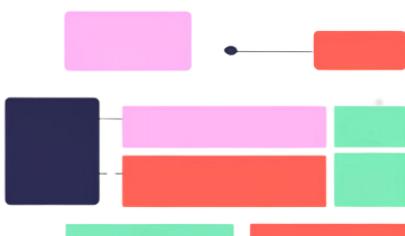
Sim. Os elementos 3 e 4.

- b) Existem elementos que não estão nem em A e nem em B? Se sim, quais?

Sim. Os elementos 7, 8 e 9.

- c) Determine o conjunto complementar de $A \cup B$.

$$(A \cup B)^c = \{7, 8, 9\}$$



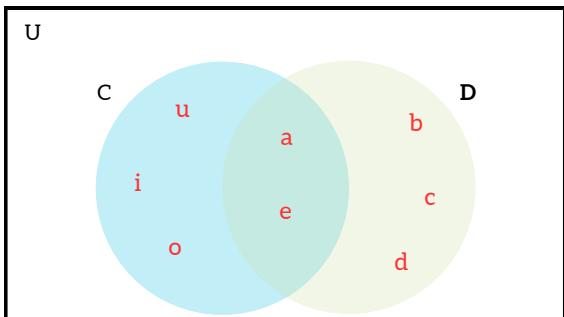
• Exercícios Propostos:

2) Sejam os conjuntos: $U = \{\text{letras do alfabeto português}\}$, $C=\{a,e,i,o,u\}$ e $D=\{a,b,c,d,e\}$. Preencha o diagrama a seguir e determine: $C \cap D$.



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!



$$C \cap D = \{a, e\}$$

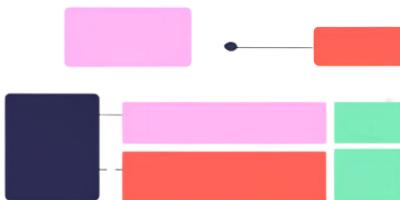
3) Sobre a questão anterior, determine os conjuntos:

$$C - D = \underline{\{i, o, u\}}$$

$$D - C = \underline{\{b, c, d\}}$$

Os conjuntos resultantes são iguais? Por quê?

Não, porque a diferença de conjuntos não é comutativa.



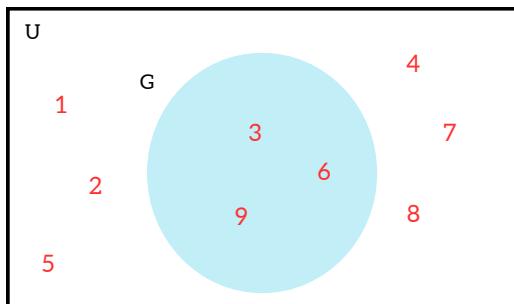
• Exercícios Propostos:

4) Seja o conjunto universo dado por $U=\{1,2,3,4,5,6,7,8,9\}$ e o conjunto dado por $G=\{3,6,9\}$. Determine o conjunto que representa o complemento de G .



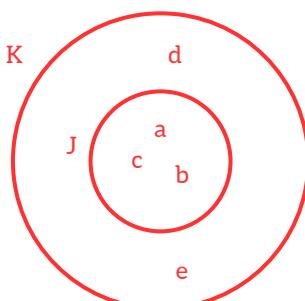
Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

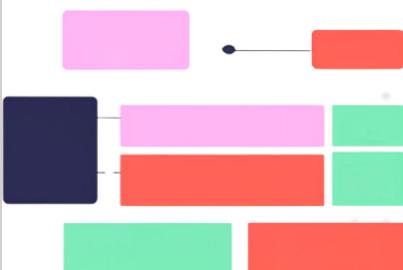


$$G^c = \{1, 2, 4, 5, 7, 8\}$$

5) Dados os conjuntos $J=\{a,b,c\}$ e $K=\{a,b,c,d,e\}$. Represente o diagrama de Venn e explique o que ocorre entre J e K .



O que ocorre é que J é subconjunto de K .



• Exercícios Propostos:

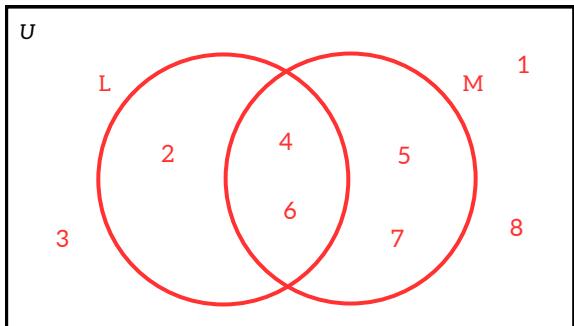
6) No universo $U=\{1,2,3,4,5,6,7,8\}$, considere os conjuntos:

$$L=\{2,4,6\} \text{ e } M=\{4,5,6,7\}.$$



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!



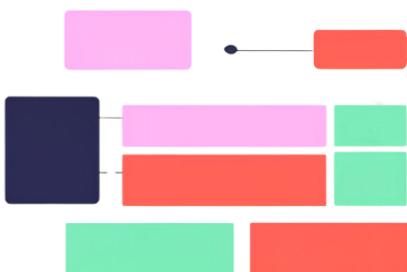
Determine:

a) $L - M = \{2\}$ _____

b) $L \cap M = \{4, 6\}$ _____

c) $L \cup M = \{2, 4, 5, 6, 7\}$ _____

d) $L^C = \{1, 3, 5, 7, 8\}$ _____



• Exercícios Propostos:

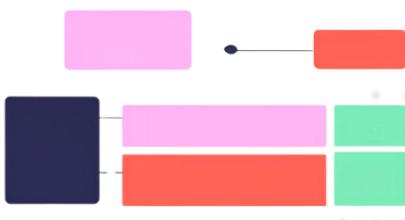
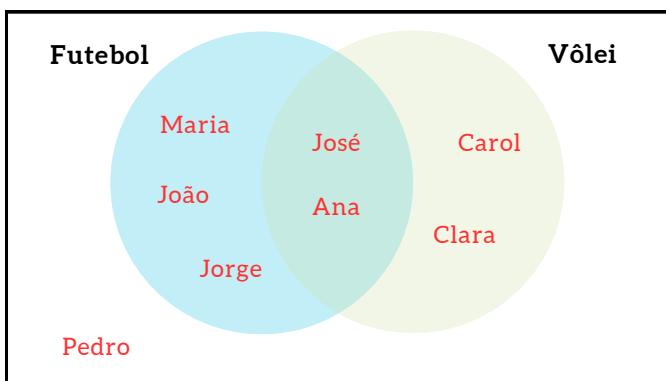
7) Uma pesquisa escolar com o tema: “você prefere futebol ou vôlei?” foi realizada com um grupo de 8 estudantes. São eles: Maria, João, José, Ana, Jorge, Carol, Clara e Pedro.



As respostas obtidas foram:

- **Futebol:** Maria, João, José, Ana, Jorge.
- **Vôlei:** José, Ana, Carol, Clara.
- **Ambos os esportes:** José, Ana.

Sabendo disso, desenhe o diagrama de Venn que representa as respostas obtidas com a pesquisa.



a) Quantos estudantes gostam apenas de futebol?

3 estudantes.



b) Quantos estudantes gostam apenas de vôlei?

2 estudantes.

Os exercícios
foram todos
respondidos
nesta versão.

Fique à vontade
para adaptar as
questões e
adicionar outros
tópicos conforme
achar
conveniente!

c) Quantos estudantes não gostam de
nenhum dos dois esportes?

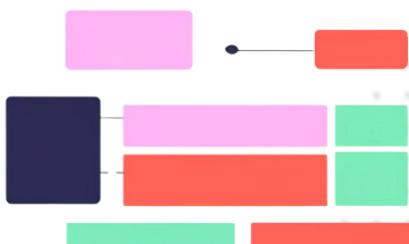
1 estudante.

d) Quantos estudantes gostam de futebol
E vôlei?

2 estudantes.

e) Quantos estudantes gostam de futebol
OU vôlei?

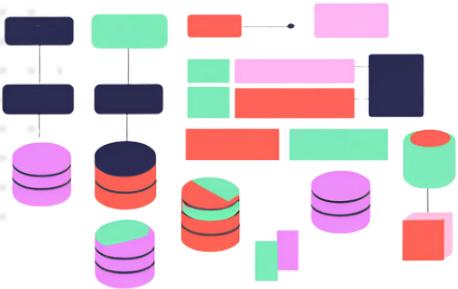
7 estudantes.





Este capítulo foi criado com o objetivo de estabelecer uma ponte entre os conceitos matemáticos e a linguagem SQL.

Reconhecendo que a compreensão dos fundamentos da computação – especialmente os que emergem das linguagens formais – é essencial para uma prática eficaz no uso de bancos de dados, este capítulo se dedica a apresentar, de forma clara e teórica, as bases que fundamentam essa interseção.



CAPÍTULO 3

Da Matemática ao Computador

As **linguagens humanas** seguem regras definidas por suas gramáticas, mas essas regras não são suficientes para os sistemas computacionais. Enquanto nós conseguimos interpretar frases com múltiplos significados ou contextos, os computadores exigem instruções claras e livres de ambiguidades.



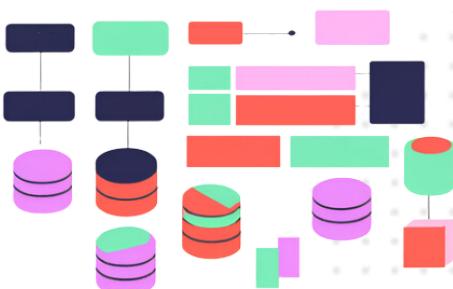
Por ser essencial para a construção de um entendimento sólido sobre como os conceitos matemáticos se traduzem em operações e estruturas na linguagem de máquina, o conteúdo é majoritariamente expositivo e teórico.

Por isso, professor, utilize metodologias baseadas em debates, discussões e/ou pesquisas direcionadas visando maior engajamento nesse capítulo.

Para garantir essa precisão, utilizamos as Linguagens e Gramáticas Formais, que são modelos matemáticos usados na maioria dos sistemas digitais. De forma simples, uma Linguagem é um conjunto de sentenças criadas a partir de um alfabeto, que é um grupo finito de símbolos.

Dependendo das regras de formação das sentenças, a linguagem resultante pode ser finita ou infinita, mesmo que o alfabeto utilizado seja limitado.

Um exemplo disso é o conjunto dos números naturais, que é infinito, mas formado por um conjunto finito de algarismos.



3.1. Linguagem Formal



Por isso, abordar a linguagem formal neste contexto é fundamental porque ela é a base para entender como os sistemas de computação funcionam.

Incentive a pesquisa sobre linguagem formal e linguagem natural para que os estudantes entendam a verdadeira importância no contexto computacional.

Uma *linguagem formal* é um sistema de **símbolos e regras bem definidos**, projetado para expressar informações de maneira precisa e não ambígua, como a Matemática, a Lógica e as Linguagens de Programação, por exemplo.

A linguagem formal possui os seguintes aspectos:

- **Alfabeto:** símbolos;
- **Sintaxe:** regras de sentenças válidas;
- **Semântica:** significado das sentenças.





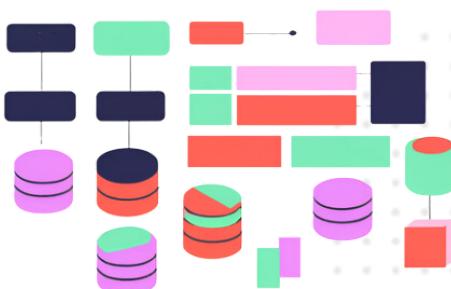
IMPORTÂNCIA PARA MÁQUINAS

Essa é uma excelente oportunidade para os estudantes explorarem a conexão entre a matemática e a computação, compreendendo como as máquinas interpretam e executam comandos com base em regras precisas e sem ambiguidade.

Além disso, pode estimular o pensamento crítico sobre como a formalização e a estruturação das informações são essenciais para a comunicação entre humanos e máquinas.

As máquinas interpretam comandos de maneira literal, sem a capacidade de deduzir significados implícitos ou lidar com ambiguidades. Por isso, as linguagens formais são fundamentais para a comunicação com computadores, garantindo que as instruções sejam **claras, previsíveis e compreensíveis** para a máquina.

Por isso, as Linguagens de Programação seguem regras rígidas de sintaxe e semântica para assegurar que os computadores processem as informações de maneira exata.



3.2. Lógica Proposicional



A Lógica Proposicional é fundamental para entender como as máquinas tomam decisões e processam informações de forma lógica e estruturada. Ela serve como base para a construção de algoritmos, condições e fluxos de controle em programação.

Para esta versão do livro, não entraremos em detalhes dessa linguagem, apenas explicitaremos seus conectivos e como a máquinas os utilizam.

A lógica proposicional é uma linguagem formal que estuda proposições declarativas, ou seja, sentenças que podem ser classificadas como verdadeiras ou falsas.

Ela utiliza conectivos lógicos, como:

- E (\wedge)
- OU (\vee)
- NÃO (\neg)

para combinar ou modificar proposições, permitindo a criação de expressões claras e precisas, essenciais para que máquinas interpretem comandos sem ambiguidades.





É importante destacar que, neste contexto, o resultado é baseado em uma atribuição de valor lógico (**verdadeiro ou falso**).

Além disso, cada um desses conectivos tem um papel específico.

O conectivo E permite que os estudantes construam condições complexas, como verificar múltiplos filtros ao mesmo tempo.

Conejativo lógico E (\wedge):

- Para uma proposição ser verdadeira com o conectivo lógico **E (conjunção)**, as duas partes da operação precisam ser verdadeiras.

Por exemplo:

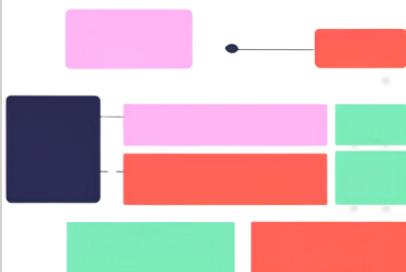
“O número 2 é primo **E** o número 2 é par.”

A proposição acima é **verdadeira**, porque as duas partes da conjunção são **verdadeiras**.

Contra-exemplo:

“O número 4 é primo **E** o número 4 é par.”

A proposição acima é **falsa**, porque o número 4 não é primo, portanto toda a frase é **falsa**.



Conectivo lógico OU (V):

- Para uma proposição ser verdadeira com o conectivo lógico **OU** (*disjunção*), apenas uma das partes da operação precisa ser verdadeira.



O conectivo OU permite a criação de condições alternativas, oferecendo flexibilidade ao permitir que os estudantes busquem registros que atendem a pelo menos uma das várias condições possíveis.

Por exemplo:

“O número 2 é primo **OU** o número 3 é par.”

A proposição acima é **verdadeira**, porque o número 2 de fato é primo. Isto já basta para tornar a frase verdadeira.

Contra-exemplo:

“O número 4 é primo **OU** o número 3 é par.”

A proposição acima é **falsa**, porque o número 4 não é primo e também o número 3 não é par. Por isso, toda a frase é **falsa**.





O conectivo NÃO é essencial para inverter o valor lógico de uma proposição, permitindo que os estudantes excluam ou modifiquem condições. Por exemplo, futuramente em consultas SQL, será possível excluir registros que atendem a uma condição específica, utilizando a negação para filtrar resultados indesejados.

Conectivo lógico NÃO (\neg):

- O conectivo NÃO (negação) inverte a valoração da proposição, ou seja, o que é verdadeiro torna-se falso e o que é falso torna-se verdadeiro.

Por exemplo:

“O número 2 é primo”

A proposição acima é verdadeira.

Entretanto, a negação:

“O número 2 NÃO é primo” é falsa.

Contra-exemplo:

Afirmiação:

“O número 4 é primo” → falso.

Negação da afirmação:

“O número 4 NÃO é primo” → verdadeiro.





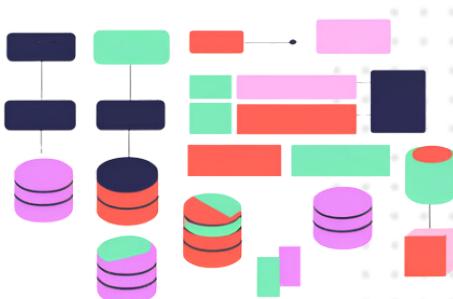
Professor, aqui o seu estudante deve compreender que o inglês é a língua predominante na área de tecnologia e estar familiarizado com esses termos é crucial para o desenvolvimento de habilidades que podem ser aplicadas em diversas ferramentas e em diversos contextos.

Por isso, conhecer os conectivos em inglês é uma vantagem prática e estratégica no aprendizado e futuro profissional.

CONECTIVOS EM INGLÊS

As linguagens de programação utilizam os conectivos E, OU e NÃO em inglês (AND, OR e NOT) porque o inglês se tornou a língua padrão da computação desde os primeiros dias da área. Como muitas das primeiras linguagens e ferramentas de programação foram desenvolvidas em países de língua inglesa, o vocabulário técnico foi definido em inglês.

PORTUGUÊS	INGLÊS
E	AND
OU	OR
NÃO	NOT





Professor, nesta seção, nosso objetivo principal é estabelecer uma relação clara entre as operações matemáticas, a lógica proposicional e a linguagem de máquina. Para facilitar a compreensão do extenso volume de informações apresentado até agora, elaboramos as tabelas a seguir, que sintetizam e sistematizam os conceitos abordados.

3.3. Conjuntos e Lógica

As máquinas utilizam os operadores matemáticos, os conectivos lógicos e as funções de comparação numérica para realizarem cálculos e comparações.

Daí, é importante relembrar que:

SÍMBOLO COMPUTACIONAL	OPERAÇÃO MATEMÁTICA
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo
pow() ou ^	Potenciação
sqrt()	Raiz Quadrada

SÍMBOLO COMPUTACIONAL	CONECTIVO LÓGICO
and	Conjunção (E)
or	Disjunção (OU)
not	Negação (NÃO)





Alguns conceitos a mais foram relembrados, como por exemplo a operação de módulo.

Fique à vontade para abordar ou não conforme seu planejamento!

Além disso, a tabela a seguir estabelece a relação entre as operações com conjuntos e os conectivos lógicos, ilustrando suas equivalências por meio do Diagrama de Venn.

Professor, essa é uma tabela muito poderosa!

IMPORTANTE:

- A operação de **Módulo (%)** tem como resultado o resto da divisão indicada.

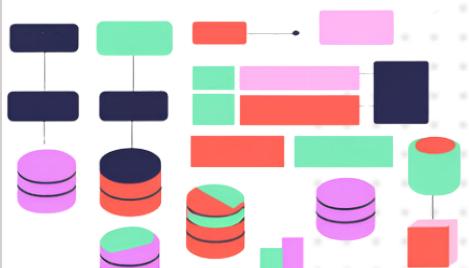
Portanto:

$$5 \% 2 = 1$$

$$15 \% 4 = 3$$

- Os conectivos lógicos serão sempre usados em inglês.

OPERAÇÃO DE CONJUNTOS	OPERADOR LÓGICO
União ($A \cup B$) 	'OU' lógico → OR ($A \vee B$)
Interseção ($A \cap B$) 	'E' lógico → AND ($A \wedge B$)
Complemento (A^C) 	NÃO lógico → NOT ($\neg A$)





Professor, neste momento, é importante destacar que os valores booleanos são o equivalente aos valores lógicos para a máquina.

Deixe claro que um valor booleano é um tipo de dado que somente pode assumir duas possibilidades: verdadeiro (True) ou falso (False).

Neste caso, utilize as funções de comparação como base para este tipo de dado.

Já as funções de comparação comparam valores numéricos conforme estamos habituados em matemática e retornam valores booleanos, ou seja, verdadeiros (true) ou falsos (false).

SÍMBOLO COMPUTACIONAL	OPERAÇÃO MATEMÁTICA
=	Igualdade
<> ou !=	Diferença
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a

Por exemplo:

- $1 = 2$: Falso
- $265.7 > 200.3$: Verdadeiro
- $-59 \leq 0$: Verdadeiro



3.4. Banco de Dados



Professor, nesta seção, abordaremos os conceitos fundamentais de banco de dados, incluindo o banco de dados relacional, para fornecer a base teórica antes de introduzir a linguagem SQL, objetivando, assim, que os estudantes compreendam a estrutura e a lógica por trás da organização e manipulação dos dados.

Até agora, vimos como a Matemática organiza seus dados por meio da Teoria de Conjuntos e como a Lógica Proposicional relaciona esses dados com base na veracidade das proposições que os definem.

Em ambos os casos, busca-se uma organização clara das informações, independentemente de sua apresentação.

Da mesma forma, um banco de dados tem o objetivo de organizar e estruturar dados, mas agora sob a ótica das necessidades computacionais, facilitando a manipulação e o acesso eficiente pela máquina.



DEFINIÇÃO



Um banco de dados é um sistema organizado para armazenar, gerenciar e manipular dados de forma eficiente e estruturada.

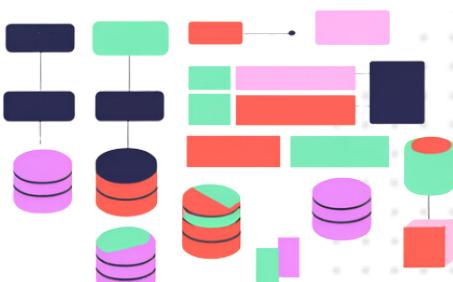
BANCO DE DADOS RELACIONAL

Aqui é fundamental enfatizar que o banco de dados relacional é um modelo facilita a integridade, a consistência e a flexibilidade na recuperação das informações, sendo essas características essenciais para a compreensão e aplicação da linguagem SQL.

É um tipo de banco de dados* que organiza os dados em tabelas interligadas, chamadas de relacionamentos, que podem ser manipuladas por meio de operações.

Cada tabela é composta por linhas (registros) e colunas (atributos), e os dados são armazenados de forma estruturada, estabelecendo relações entre as tabelas.

*Sempre que houver a menção "banco de dados" ao longo deste livro, estaremos nos referindo especificamente a bancos de dados relacionais, que serão aqui o objeto de estudo.





SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS - SGBD

Professor, deixe claro que o SGBD **viabiliza** a criação, a manipulação e a administração dos bancos de dados.

É importante compreender que, ao entender o funcionamento do SGBD, os estudantes conseguem contextualizar melhor o uso do SQL, pois passam a saber como as consultas são processadas e como os dados são armazenados.

Um SGBD é um *software* que **abstrai** a complexidade do armazenamento e manipulação dos dados, fornecendo uma interface simplificada para os usuários e aplicativos.

Ele organiza os dados de maneira estruturada e permite operações como inserção, atualização, exclusão e consulta, sem que o usuário precise se preocupar com os detalhes de implementação, como alocação de memória ou indexação.

*Abstrair, em computação, é sinônimo de “tornar simples”. Abstração é o fenômeno de deixar mais entendível para o ser humano.





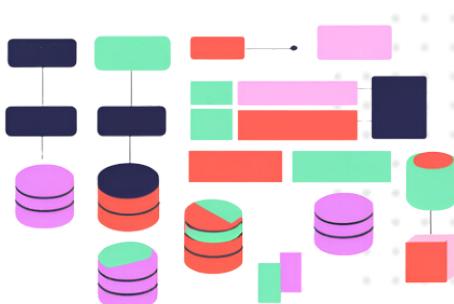
SQL - STRUCTURED QUERY LANGUAGE

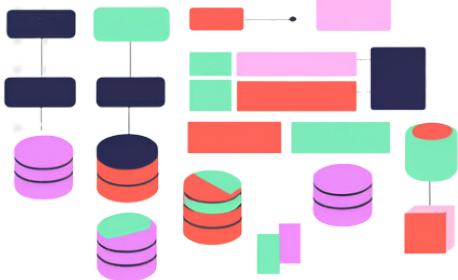
A linguagem SQL é usada para comunicar-se com bancos de dados, permitindo a criação, consulta, atualização e exclusão de dados, além da definição de estruturas e controle de acesso.

- Espaço para diálogo com o professor.

É uma forma padronizada e simples de manipular dados em bancos de dados, permitindo que usuários consultem e gerenciem informações usando *comandos declarativos** em vez de linguagens complexas.

*A definição de linguagem declarativa será dada no capítulo seguinte, o que facilitará no entendimento do termo “comando declarativos”.





Explicar o conceito de linguagem declarativa é fundamental para que os estudantes compreendam a diferença entre dizer o que deve ser feito (como no SQL) e especificar como isso deve ser feito (como em linguagens procedurais).

CAPÍTULO 4

SQL - A Linguagem dos Dados

4.1. Linguagem Declarativa

A linguagem SQL é uma linguagem declarativa, uma vez que, ao utilizá-la, o foco está nos resultados desejados e não como isso será feito* (*algoritmos para alcançar esses resultados*).

- **Exemplo:**

SQL

```
SELECT nome, idade  
FROM clientes  
WHERE idade > 30;
```

*É o SGBD que cuida do processo de busca e execução do dados solicitados.



Utilize a consulta anterior e leia-a os estudantes (assim como faz o primeiro parágrafo desta página).

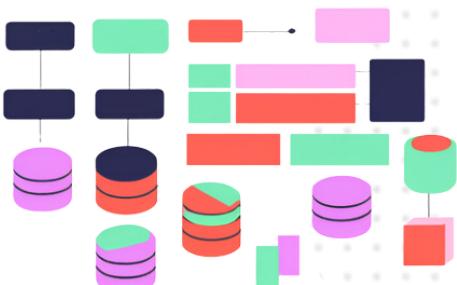
É importante que eles entendam o que se espera como resultado para então construir o comando.

A consulta está solicitando ao SGBD o equivalente a:

- “Quero os nomes e idades da tabela clientes;
- Somente dos clientes com idade maior que 30.”

Não há necessidade de preocupações de detalhes como:

- Qual índice será usado.
- Como os registros serão acessados.
- A ordem dos passos para processar a consulta.



É IMPORTANTE SABER...



Professor, esta página é essencial para estabelecer a base de como trabalharemos com SQL ao longo das próximas páginas.

Primeiro, defina o escopo de estudo, deixando claro que o foco será o comando SELECT em seguida explique a estrutura de leitura dos comandos SQL, enfatizando que a execução ocorre de forma sequencial até que o ponto e vírgula (;) seja encontrado.

1) O escopo de estudo desse livro limita-se ao uso do comando **SELECT** (veremos mais à frente). Portanto, todos os blocos de códigos de SQL mostrados aqui poderão ser chamados de: “**consulta**” ou “**query**” (termo em inglês).

2) Os comandos em SQL são lidos sequencialmente até que haja um ponto e vírgula. Este indica o fim do bloco de código escrito.

Portanto, é **obrigatório** o uso de **ponto e vírgula (;)** para marcar o **fim de uma consulta**.



4.2. PRINCIAL COMANDO



O principal comando da linguagem SQL é o comando **SELECT - FROM - WHERE**, como vimos no exemplo anterior.

Esta seção é fundamental para introduzir a estrutura básica do SQL, que é a base para consultas em bancos de dados. Ao compreender essa estrutura, os estudantes desenvolvem a capacidade de recuperar informações, filtrando dados conforme necessário.

Vejamos suas funções:

- **SELECT:** lista os campos desejados no resultado de uma consulta.
- **FROM:** indica as tabelas a serem utilizadas na consulta.
- **WHERE:** filtra os registros, especificando as condições que devem ser satisfeitas.



• Exemplo 1:

Tabela clientes



ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

Aqui, o estudante precisa compreender que:

SELECT → Define quais colunas ou dados queremos recuperar.

Ensine os estudantes a especificarem exatamente as informações desejadas, evitando retornos desnecessários.

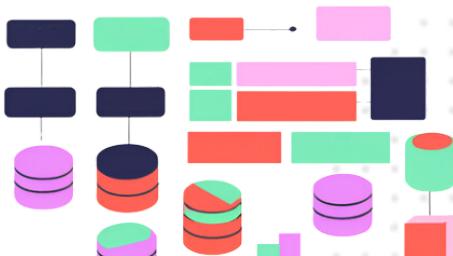
Caso queira-se selecionar as todas as informações dos clientes que moram em Brasília, deve-se:

SQL

```
SELECT *  
FROM clientes  
WHERE CIDADE = 'Brasília';
```

le-se: 'seleciona tudo'
le-se: 'da tabela'
le-se: 'onde/cujo'

1. SELECT *: seleciona todas as colunas;
2. FROM: indica a tabela da consulta;
3. WHERE: determina a condição para aparecer no resultado.



Nesse caso, a consulta...



SQL

```
SELECT *
FROM clientes
WHERE CIDADE = 'Brasília';
```

Aqui, o estudante precisa compreender que:

FROM → Indica a tabela de onde os dados serão extraídos.

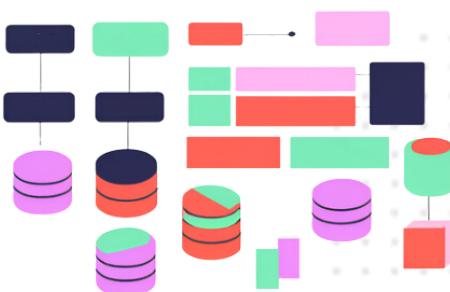
Explique essa cláusula é essencial para que os alunos compreendam a estrutura dos bancos relacionais e a organização dos dados.

... quer selecionar todas as colunas da tabela 'clientes' que possuem 'Brasília' como valor da coluna 'CIDADE'.

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasilia
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

Portanto, essa consulta terá como resultado a tabela:

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasilia
3	Maria	28	Brasília



• Exemplo 2:



Tabela clientes

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

Aqui, o estudante precisa compreender que:

WHERE →
Permite filtrar os registros com condições específicas.

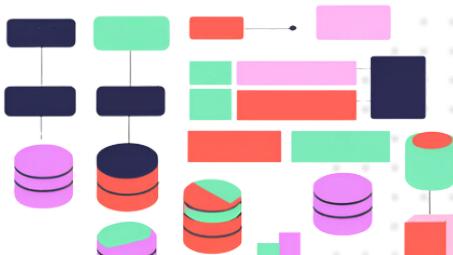
Essa cláusula é essencial para criar consultas mais direcionadas e evitar a sobrecarga de dados desnecessários.

Agora, não há necessidade de retornar todas as colunas da tabela, somente nome e idade. Para isto:

SQL

```
SELECT NOME, IDADE           le-se: 'seleciona COL1, COL2, ...'  
FROM clientes                 le-se: 'da tabela'  
WHERE CIDADE = 'São Paulo';  le-se: 'onde/cujo'
```

1. SELECT NOME, IDADE: seleciona as colunas especificadas;
2. FROM: indica a tabela da consulta;
3. WHERE: determina a condição para aparecer no resultado.



Nesse caso, a consulta...



SQL

```
SELECT NOME, IDADE  
FROM clientes  
WHERE CIDADE = 'São Paulo';
```

Professor, para facilitar a compreensão da estrutura do comando SELECT-FROM-WHERE, apresentaremos exemplos que ilustram seu uso prático.

A ideia é mostrar aos estudantes como consultar um banco de dados.

... quer selecionar as colunas ‘NOME’ e ‘IDADE’ da tabela ‘clientes’ que possuem ‘São Paulo’ como valor da coluna ‘CIDADE’.

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

Portanto, essa consulta terá como resultado a tabela:

NOME	IDADE
João	35



• Exemplo 3:



Tabela compras

ID_COMPRA	ID_CLIENTE	PRODUTO	VALOR
101	1	Notebook	3500.00
102	2	Smartphone	2500.00
103	3	Tablet	1500.00
104	4	Smartwatch	800.00
105	5	Fone de Ouvido	200.00

Professor, para facilitar a compreensão da estrutura do comando SELECT-FROM-WHERE, apresentaremos exemplos que ilustram seu uso prático.

A ideia é mostrar aos estudantes como consultar um banco de dados.

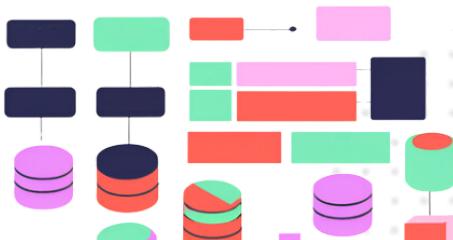
Agora analise a consulta a seguir:

SQL

```
SELECT PRODUTO, VALOR  
FROM compras  
WHERE VALOR > 2000;
```

Nesse caso temos uma comparação matemática envolvida. Não se assuste! É exatamente a relação a qual você está habituado.

A cláusula do WHERE determina que o resultado precisa conter os registros que contém a coluna 'VALOR' com números maiores que 2000 .





Daí, a consulta...

SQL

```
SELECT PRODUTO, VALOR  
FROM compras  
WHERE VALOR > 2000;
```

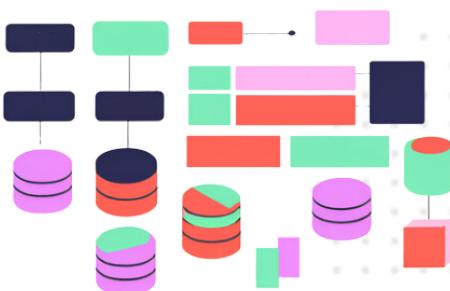
Professor, para facilitar a compreensão da estrutura do comando SELECT-FROM-WHERE, apresentaremos exemplos que ilustram seu uso prático.

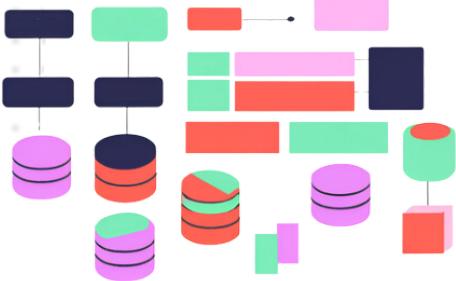
A ideia é mostrar aos estudantes como consultar um banco de dados.

ID_COMPRA	ID_CLIENTE	PRODUTO	VALOR
101	1	Notebook	3500.00
102	2	Smartphone	2500.00
103	3	Tablet	1500.00
104	4	Smartwatch	800.00
105	5	Fone de Ouvido	200.00

... tem como resultado:

PRODUTO	VALOR
Notebook	3500.00
Smartphone	2500.00





Nessa seção, o estudante precisa compreender que:

AND → Permite combinar múltiplas condições, retornando apenas os registros que satisfazem todas elas.

4.3. Operadores Lógicos

O uso de operadores lógicos na linguagem SQL é essencial para a construção de consultas mais refinadas e poderosas dentro do comando **WHERE**.

Vejamos suas funções:

- **AND:** combina duas ou mais condições e retorna os registros que atendem a todas elas.
- **OR:** combina duas ou mais condições e retorna os registros que atendem a pelo menos uma delas.
- **NOT:** inverte o resultado de uma condição, retornando os registros que não a atendem.

• Exemplo 1:

Tabela clientes



ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

Suponha que queremos selecionar os clientes que possuam o campo CIDADE = ‘Brasília’ E o campo IDADE > 30.

Aqui, o estudante precisa compreender que:

OR → Permite que um registro seja retornado se pelo menos uma das condições for verdadeira.

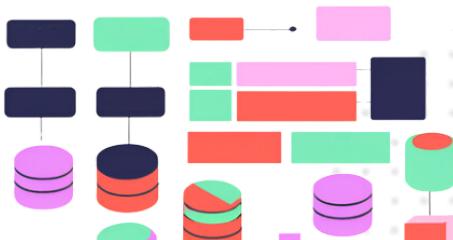
SQL

```
SELECT NOME, IDADE, CIDADE  
FROM clientes  
WHERE CIDADE = ‘Brasília’ AND IDADE > 25;
```

Nesse caso, as duas condições:

- CIDADE = ‘Brasília’; **E**
- IDADE > 25

ambas precisam ser verdadeiras.





Por isso, a consulta...

SQL

```
SELECT NOME, IDADE, CIDADE  
FROM clientes  
WHERE CIDADE = 'Brasília' AND IDADE > 25;
```

Aqui, o estudante precisa compreender que:

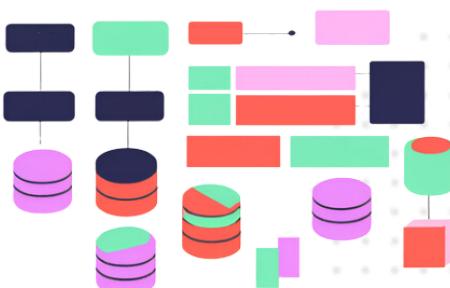
NOT → Inverte a lógica de uma condição, retornando os registros que não atendem ao critério especificado.

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

... tem como resultado:

NOME	IDADE	CIDADE
Maria	28	Brasília

Porque, nesse caso, é a única linha que possui das duas condições simultaneamente.



• Exemplo 2:



Tabela clientes

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

Para facilitar a compreensão da estrutura dos OPERADORES LÓGICOS, apresentaremos exemplos que ilustram seu uso prático.

A ideia é mostrar aos estudantes como consultar um banco de dados filtrando informações conforme necessário.

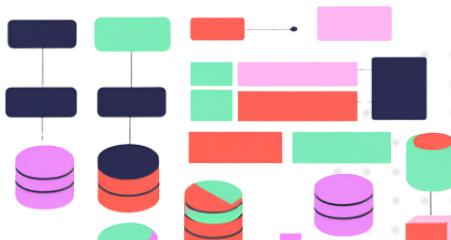
Suponha que queremos selecionar os clientes que possuam o campo CIDADE = 'São Paulo' OU o campo CIDADE = 'Salvador'.

SQL

```
SELECT NOME, CIDADE  
FROM clientes  
WHERE  
CIDADE = 'Brasília' OR CIDADE = 'Salvador';
```

Nesse caso:

- CIDADE = 'Brasília'; **OU**
 - CIDADE = 'Salvador'
- uma delas** podem ser verdadeiras.





Por isso, a consulta...

SQL

```
SELECT NOME, CIDADE  
FROM clientes  
WHERE  
CIDADE = 'Brasília' OR CIDADE = 'Salvador';
```

Para facilitar a compreensão da estrutura dos OPERADORES LÓGICOS, apresentaremos exemplos que ilustram seu uso prático.

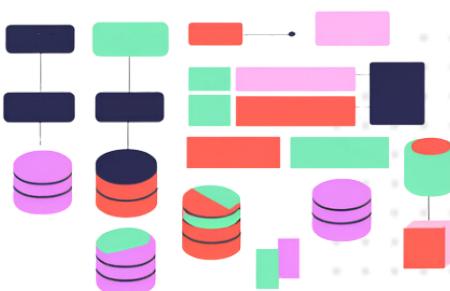
A ideia é mostrar aos estudantes como consultar um banco de dados filtrando informações conforme necessário.

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

... tem como resultado:

NOME	CIDADE
Ana	Brasília
Maria	Brasília
Fernanda	Salvador

Porque, essas linhas são aquelas que possuem CIDADE = 'Brasília' OU CIDADE = 'Salvador'.





• Exemplo 3:

Tabela clientes

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

Para facilitar a compreensão da estrutura dos OPERADORES LÓGICOS, apresentaremos exemplos que ilustram seu uso prático.

A ideia é mostrar aos estudantes como consultar um banco de dados filtrando informações conforme necessário.

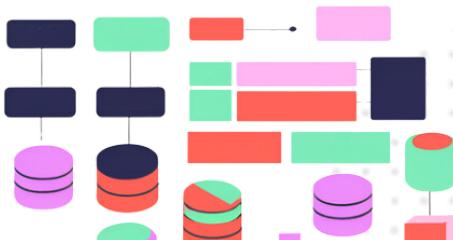
Agora que queremos selecionar os clientes que não possuam o campo CIDADE = 'Belo Horizonte'.

SQL

```
SELECT IDADE, CIDADE  
FROM clientes  
WHERE NOT CIDADE = 'Belo Horizonte'
```

Nesse caso queremos todas as ocorrências que NÃO são:

- CIDADE = 'Belo Horizonte'.





Por isso, a consulta...

SQL

```
SELECT IDADE, CIDADE  
FROM clientes  
WHERE NOT CIDADE = 'Belo Horizonte'
```

Para facilitar a compreensão da estrutura dos OPERADORES LÓGICOS, apresentaremos exemplos que ilustram seu uso prático.

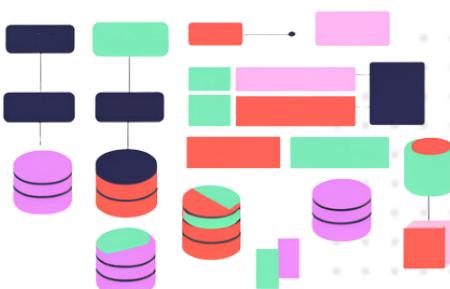
A ideia é mostrar aos estudantes como consultar um banco de dados filtrando informações conforme necessário.

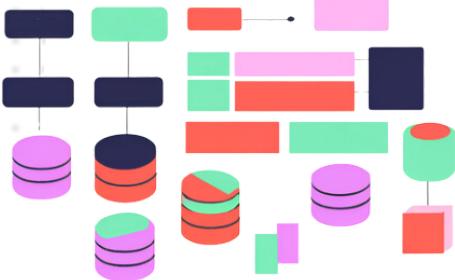
ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasilia
2	João	35	São Paulo
3	Maria	28	Brasilia
4	Pedro	40	Belo Horizonte
5	Fernanda	30	Salvador

... tem como resultado:

IDADE	CIDADE
25	Brasilia
35	São Paulo
28	Brasilia
30	Salvador

Porque essas são as ocorrências da tabela que possui linhas que não são CIDADE = 'Belo Horizonte'.





4.4. Operadores de JOIN

Professor, esta seção é fundamental para que os estudantes compreendam como JOINs são utilizados para combinar dados de diferentes tabelas em uma única consulta.

Um *JOIN* em SQL é uma operação que combina dados de duas ou mais tabelas com base em uma condição de relacionamento, geralmente uma coluna em comum. Ele permite vincular registros relacionados entre tabelas diferentes, facilitando a obtenção de informações completas e conectadas.

O uso de *joins* permite a criação de consultas mais completas e detalhadas.

Os *joins* mais utilizados em SQL são o INNER e o LEFT. Entretanto, existe o RIGHT que também pode ser usado em algumas situações.

Vejamos seus tipos:



Aqui, o estudante precisa compreender que:

INNER JOIN →
Retorna apenas os registros que possuem correspondência em ambas as tabelas.

- **INNER JOIN:** retorna apenas os registros que possuem correspondência **em ambas** as tabelas envolvidas na junção.
- **LEFT JOIN:** retorna todos os registros da tabela **à esquerda** e os registros correspondentes da tabela à direita; onde não houver correspondência, os valores serão nulos.
- **RIGHT JOIN:** retorna todos os registros da tabela **à direita** e os registros correspondentes da tabela à esquerda; onde não houver correspondência, os valores serão nulos.



• Exemplo 1:



Tabela clientes compras

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte

Tabela compras

ID_COMPRA	ID_CLIENTE	PRODUTO	VALOR
101	1	Notebook	3500.00
102	2	Smartwatch	800.00
103	2	Smartphone	2500.00
104	3	Tablet	1500.00
105	4	Smartphone	2500.00
106	4	Smartwatch	800.00
107	5	Fone de Ouvido	200.00

Observe que tanto na tabela cliente quanto na tabela compras existe a coluna ID_CLIENTE.

A partir dessa coluna, podemos juntar as tabelas, ou seja, realizar um join.



Aqui, o estudante precisa compreender que:

LEFT JOIN →
Retorna todos os registros da tabela à esquerda e os correspondentes da tabela à direita. Se não houver correspondência, os valores aparecem como NULL.

Vejamos o *inner join*:



SQL

```
SELECT  
    cl.NOME,  
    cl.CIDADE,  
    co.ID_COMPRA,  
    co.PRODUTO  
FROM clientes AS cl  
INNER JOIN compras AS co  
ON cl.ID_CLIENTE = co.ID_CLIENTE
```

A palavra reserva 'AS' apelida a tabela. Por isso, agora, a tabela clientes pode ser referenciada como 'cl' e a tabela compras como 'co'.

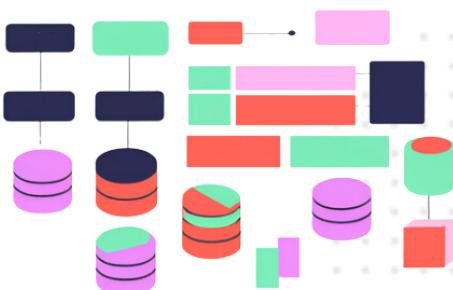
Aqui, o estudante precisa compreender que:

RIGHT JOIN → O oposto do LEFT JOIN: retorna todos os registros da tabela à direita e os correspondentes da tabela à esquerda.

Aqui estamos dizendo:

- Selecione as colunas NOME, CIDADE da tabela clientes e as colunas ID_COMPRA e PRODUTO da tabela compras.

O **INNER JOIN** retornará apenas os registros onde há correspondência entre as tabelas clientes e compras na coluna ID_CLIENTE.



Por isso, a consulta...



Para facilitar a compreensão da estrutura dos JOINs, apresentaremos exemplos que ilustram seu uso prático.

Aqui, especificamente, é importante explicar porque o registro de ID_CLIENTE = 5 não foi mostrado no resultado.

SQL

```
SELECT
    cl.NOME,
    cl.CIDADE,
    co.ID_COMPRA,
    co.PRODUTO
FROM clientes AS cl
INNER JOIN compras AS co
    ON cl.ID_CLIENTE = co.ID_CLIENTE
```

Precisamos colocar o "apelido" da tabela antes da coluna para evitar ambiguidades quando duas ou mais tabelas possuem colunas com o mesmo nome em uma consulta.

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte

ID_COMPRA	ID_CLIENTE	PRODUTO	VALOR
101	1	Notebook	3500.00
102	2	Smartwatch	800.00
103	2	Smartphone	2500.00
104	3	Tablet	1500.00
105	4	Smartphone	2500.00
106	4	Smartwatch	800.00
107	5	Fone de Ouvido	200.00

... tem como resultado:

cl.NOME	cl.CIDADE	co.ID_COMPRA	co.PRODUTO
Ana	Brasília	101	Notebook
João	São Paulo	102	Smartwatch
João	São Paulo	103	Smartphone
Maria	Brasília	104	Tablet
Pedro	Belo Horizonte	105	Smartphone
Pedro	Belo Horizonte	106	Smartwatch





• Exemplo 2:

Para facilitar a compreensão da estrutura dos JOINs, apresentaremos exemplos que ilustram seu uso prático.

Compreender esses conceitos permite que os estudantes realizem consultas mais avançadas e extraiam dados de múltiplas tabelas.

Tabela clientes compras

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte

Tabela compas

ID_COMPRA	ID_CLIENTE	PRODUTO	VALOR
101	1	Notebook	3500.00
102	2	Smartwatch	800.00
103	2	Smartphone	2500.00
104	3	Tablet	1500.00
105	4	Smartphone	2500.00
106	4	Smartwatch	800.00
107	5	Fone de Ouvido	200.00

Usando as mesmas tabelas do exemplo anterior, iremos exibir todos os clientes, incluindo aqueles que não realizaram compras (todas as ocorrências da tabela à esquerda, *left join*).



Vejamos:



Para facilitar a compreensão da estrutura dos JOINs, apresentaremos exemplos que ilustram seu uso prático.

Compreender esses conceitos permite que os estudantes realizem consultas mais avançadas e extraiam dados de múltiplas tabelas.

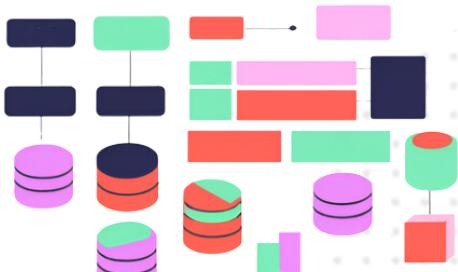
SQL

```
SELECT  
    cl.NOME,  
    cl.CIDADE,  
    co.PRODUTO,  
    co.VALOR  
FROM clientes AS cl  
LEFT JOIN compras AS co  
    ON cl.ID_CLIENTE = co.ID_CLIENTE
```

Aqui estamos dizendo:

- Selecione as colunas NOME, CIDADE da tabela clientes e as colunas PRODUTO e VALOR da tabela compras.

O **LEFT JOIN** retorna todos os registros da tabela clientes, mesmo que não haja correspondência na tabela compras. Onde não houver compras, os valores das colunas da tabela compras serão exibidos como **NULL**.





Por isso, a consulta...

SQL

SELECT

```
cl.NOME,  
cl.CIDADE,  
co.PRODUTO,  
co.VALOR  
FROM clientes AS cl  
LEFT JOIN compras AS co  
ON cl.ID_CLIENTE = co.ID_CLIENTE
```

Para facilitar a compreensão da estrutura dos JOINs, apresentaremos exemplos que ilustram seu uso prático.

Compreender esses conceitos permite que os estudantes realizem consultas mais avançadas e extraiam dados de múltiplas tabelas.

... tem como resultado:

cl.NOME	cl.CIDADE	co.PRODUTO	co.VALOR
Ana	Brasília	Notebook	3500.00
Ana	Brasília	NULL	NULL
João	São Paulo	Smartwatch	800.00
João	São Paulo	Smartphone	2500.00
Maria	Brasília	Tablet	1500.00
Maria	Brasilia	NULL	NULL
Pedro	Belo Horizonte	Smartphone	2500.00
Pedro	Belo Horizonte	Smartwatch	800.00
Maria	Brasília	NULL	NULL





• Exemplo 3:

Tabela clientes compras

ID_CLIENTE	NOME	IDADE	CIDADE
1	Ana	25	Brasília
2	João	35	São Paulo
3	Maria	28	Brasília
4	Pedro	40	Belo Horizonte

Para facilitar a compreensão da estrutura dos JOINs, apresentaremos exemplos que ilustram seu uso prático.

Compreender esses conceitos permite que os estudantes realizem consultas mais avançadas e extraiam dados de múltiplas tabelas.

Tabela compas

ID_COMPRA	ID_CLIENTE	PRODUTO	VALOR
101	1	Notebook	3500.00
102	2	Smartwatch	800.00
103	2	Smartphone	2500.00
104	3	Tablet	1500.00
105	4	Smartphone	2500.00
106	4	Smartwatch	800.00
107	5	Fone de Ouvido	200.00

Usando as mesmas tabelas do exemplo anterior, iremos exibir todas as compras, incluindo aquelas realizadas por clientes que não estão na tabela clientes (todas as ocorrências da tabela à esquerda, *right join*).



Vejamos:



Para facilitar a compreensão da estrutura dos JOINs, apresentaremos exemplos que ilustram seu uso prático.

Compreender esses conceitos permite que os estudantes realizem consultas mais avançadas e extraiam dados de múltiplas tabelas.

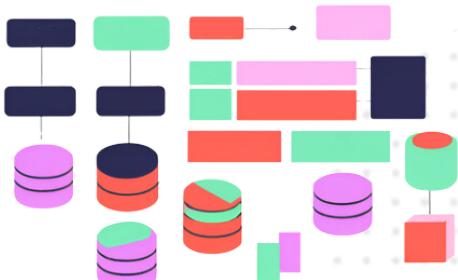
SQL

```
SELECT  
    cl.NOME,  
    cl.IDADE,  
    co.ID_COMPRA,  
    co.VALOR  
FROM clientes AS cl  
RIGHT JOIN compras AS co  
    ON cl.ID_CLIENTE = co.ID_CLIENTE
```

Aqui estamos dizendo:

- Selecione as colunas NOME, IDADE da tabela clientes e as colunas ID_COMPRA e VALOR da tabela compras.

O **RIGHT JOIN** retorna todos os registros da tabela compras, mesmo que não haja correspondência na tabela clientes. Onde não houver clientes, os valores das colunas da tabela clientes serão exibidos como **NULL**.





Por isso, a consulta...

SQL

```
SELECT  
    cl.NOME,  
    cl.IDADE,  
    co.ID_COMPRA,  
    co.VALOR  
FROM clientes AS cl  
RIGHT JOIN compras AS co  
    ON cl.ID_CLIENTE = co.ID_CLIENTE
```

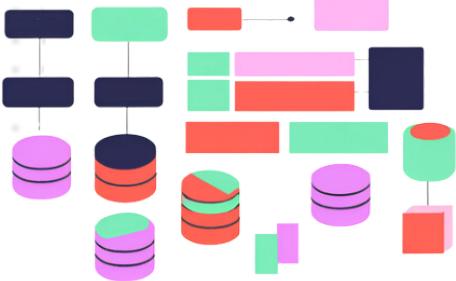
Para facilitar a compreensão da estrutura dos JOINs, apresentaremos exemplos que ilustram seu uso prático.

Compreender esses conceitos permite que os estudantes realizem consultas mais avançadas e extraiam dados de múltiplas tabelas.

... tem como resultado:

cl.NOME	cl.IDADE	co.ID_COMPRA	co.VALOR
Ana	25	101	3500.00
João	35	102	800.00
João	35	103	2500.00
Maria	28	104	1500.00
Pedro	40	105	2500.00
Pedro	40	106	800.00
NULL	NULL	107	200.00





Professor, os exercícios propostos são a ponte entre a teoria e a prática – é aqui que os estudantes exercem o aprendizado, experimentam, erram e evoluem.

Cada desafio resolvido fortalece a compreensão e aproxima a SQL do dia a dia deles.

Incentive!

CAPÍTULO 5

Exercícios Práticos

5.1. Conceitos gerais de SQL

Considere as seguintes afirmações sobre a linguagem SQL e julgue-as como Certo (C) ou Errado (E):

1. (E) A linguagem SQL é procedural, pois o usuário deve especificar exatamente como os dados devem ser buscados, incluindo os algoritmos utilizados pelo SGBD.

2. (C) O comando SELECT é utilizado para realizar consultas em um banco de dados e, neste livro, todos os blocos de código SQL apresentados podem ser chamados de "consulta" ou "query".

3. (C) Em SQL, a execução de comandos ocorre de maneira sequencial até encontrar um ponto e vírgula (;), que marca o fim da instrução.

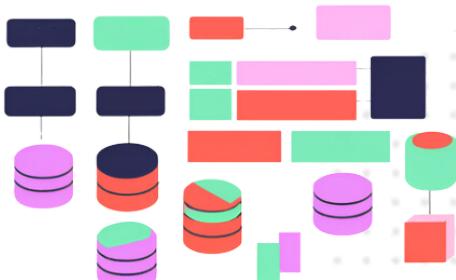


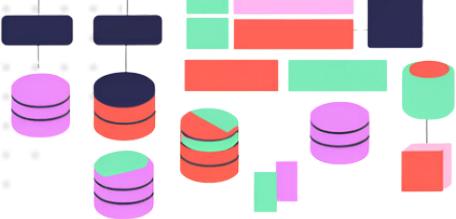
4. (E) Uma consulta SQL precisa especificar qual índice será utilizado e como os registros devem ser acessados, pois essas informações não são gerenciadas pelo SGBD.

Questões que aparecerem com o ícone de fogo (🔥) serão questões em que o estudante precisará extrapolar o material e pesquisar sobre a resolução.

5. (C) O foco da linguagem SQL está nos resultados desejados e não na forma como a consulta será executada internamente pelo SGBD.

6. (C) Diferente de algumas linguagens de programação tradicionais, em SQL a ordem dos comandos dentro de uma consulta pode ser diferente da ordem em que são processados pelo SGBD. 🔥





5.2. Principal Comando (SELECT - FROM - WHERE)

Para as questões seguintes, considere a tabela **clientes**:



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

id	nome	idade	cidade	renda
1	Ana Souza	25	Brasília	3500.00
2	Carlos Lima	40	São Paulo	7200.00
3	Júlia Melo	32	Rio de Janeiro	5000.00
4	Marcos Silva	45	São Paulo	8500.00
5	Fernanda Costa	29	Salvador	4200.00

1) Complete a consulta para selecionar os nomes e idades dos clientes que moram na cidade de São Paulo.

SQL

```
SELECT _____, _____  
FROM _____  
WHERE _____ = 'São Paulo';
```

Tabela Resultado Esperado:

nome	idade
Carlos Lima	40
Marcos Silva	45

2) Complete a consulta para selecionar todos os dados dos clientes cuja renda seja maior que 5000.



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

SQL

```
SELECT _____ *
FROM   _____
WHERE  _____ > 5000;
```

Tabela Resultado Esperado:

id	nome	idade	cidade	renda
2	Carlos Lima	40	São Paulo	7200.00
4	Marcos Silva	45	São Paulo	8500.00



3) Complete a consulta para selecionar todos os dados dos clientes cuja idade é menor que 40 anos.



Os exercícios foram todos respondidos nesta versão.

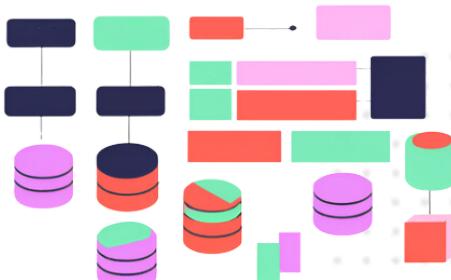
Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

SQL

```
SELECT _____ *
FROM _____
WHERE _____ < _____;
```

Complete a Tabela Resultado Esperado:

id	nome	idade	cidade	renda
1	Ana Souza	25	Brasília	3500.00
3	Júlia Melo	32	Rio de Janeiro	5000.00
5	Fernanda Costa	29	Salvador	4200.00



4) Complete a consulta para selecionar o id e os nomes dos clientes cujo nome começa com a letra "A". 🔥



Os exercícios
foram todos
respondidos
nesta versão.

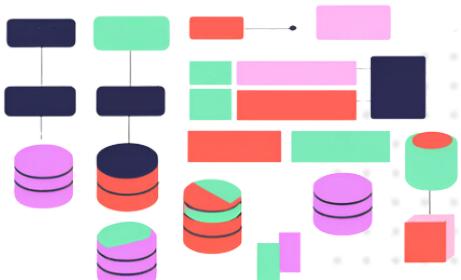
Fique à vontade
para adaptar as
questões e
adicionar outros
tópicos conforme
achar
conveniente!

SQL

```
SELECT _____, _____  
FROM _____  
WHERE _____ LIKE _____;
```

Tabela Resultado Esperado:

id	nome
1	Ana Souza



5) A tabela **produtos** armazena informações sobre os produtos de uma loja.

id	nome	categoria	preco
1	Notebook	Eletrônicos	4500.00
2	Smartphone	Eletrônicos	2500.00
3	Geladeira	Eletrodomésticos	3200.00
4	Fogão	Eletrodomésticos	1800.00
5	TV 50"	Eletrônicos	3700.00



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

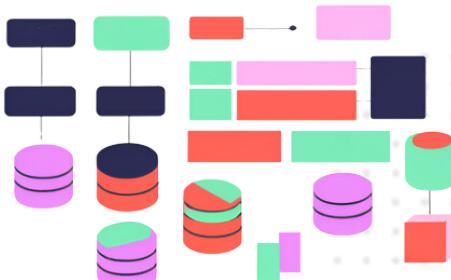
Uma consulta no banco de dados da empresa, gerou a seguinte Tabela Resultado:

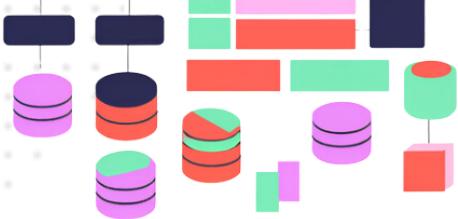
nome	preco
Notebook	4500.00
Smartphone	2500.00
TV 50"	3700.00

Qual foi a *query* executada?

SQL

```
SELECT nome, preco  
FROM produtos  
WHERE categoria = 'Eletrônicos';
```





5.3. Operadores lógicos (AND, OR e NOT)



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

id	nome	categoria	preco
1	Notebook	Eletrônicos	4500.00
2	Smartphone	Eletrônicos	2500.00
3	Geladeira	Eletrodomésticos	3200.00
4	Fogão	Eletrodomésticos	1800.00
5	TV 50"	Eletrônicos	3700.00

- 1) Complete a consulta abaixo para selecionar apenas produtos da categoria "Eletrônicos" e com preço maior que 3000.

SQL

```
SELECT nome, preco  
FROM produtos  
WHERE categoria = 'Eletrônicos' AND  
      preco > 3000;
```

Para as questões seguintes, considere a tabela **clientes**:

id	nome	idade	cidade	renda
1	Ana Souza	25	Brasília	3500.00
2	Carlos Lima	40	São Paulo	7200.00
3	Júlia Melo	32	Rio de Janeiro	5000.00
4	Marcos Silva	45	São Paulo	8500.00
5	Fernanda Costa	29	Salvador	4200.00



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

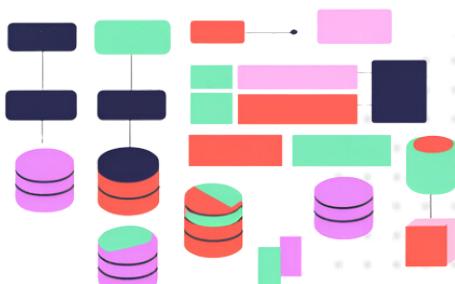
2) Complete a consulta abaixo para selecionar clientes que moram em São Paulo ou Salvador.

SQL

```
SELECT nome, cidade  
FROM clientes  
WHERE cidade = 'São Paulo' OR  
cidade = 'Salvador';
```

Tabela Resultado Esperado:

nome	cidade
Carlos Lima	São Paulo
Marcos Silva	São Paulo
Fernanda Costa	Salvador



3) Complete a consulta abaixo para selecionar todos os clientes **exceto** os que moram no Rio de Janeiro.

SQL

```
SELECT nome, cidade  
FROM clientes  
WHERE NOT cidade = 'Rio de Janeiro';
```



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

Preencha a Tabela Resultado Esperado:

nome	cidade
Ana Souza	Brasília
Carlos Lima	São Paulo
Marcos Silva	São Paulo
Fernanda Costa	Salvador



4) Complete a consulta abaixo para selecionar clientes que moram em São Paulo e têm mais de 30 anos.

SQL

```
SELECT nome, cidade, idade  
FROM _____  
WHERE _____ = ' _____ ' , AND  
_____ > _____ ;
```

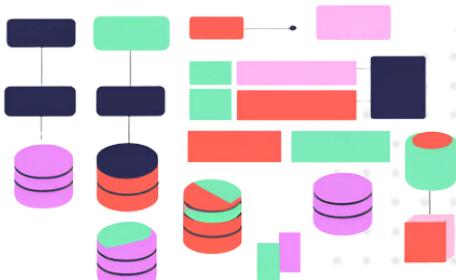


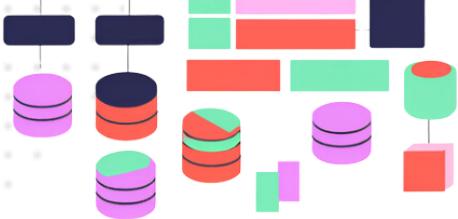
Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

Determine a Tabela Resultado Esperado.

nome	cidade	idade
Carlos Lima	São Paulo	40
Marcos Silva	São Paulo	45





5.4. Operadores JOIN (INNER, LEFT e RIGHT)

Para a questão a seguir, considere a tabela **pedidos**:



Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

id_pedido	id_cliente	data_pedido	valor
1	101	2024-01-10	500.00
2	102	2024-02-15	1200.00
3	103	2024-03-05	300.00
4	101	2024-04-20	450.00
5	104	2024-05-10	700.00

E a tabela **clientes**:

id_cliente	nome	cidade
101	Ana Souza	Brasília
102	Carlos Lima	São Paulo
103	Júlia Melo	Rio de Janeiro
105	Marcos Silva	São Paulo

- 1) Complete a consulta seguinte para selecionar apenas os pedidos que possuem clientes cadastrados, aplicando o *join* nas tabelas de forma correta.

SQL

```
SELECT p.id_pedido,  
       c.nome,  
       p.data_pedido,  
       p.valor  
  FROM pedidos AS p  
INNER JOIN clientes AS c  
    ON p.id_cliente = c.id_cliente;
```

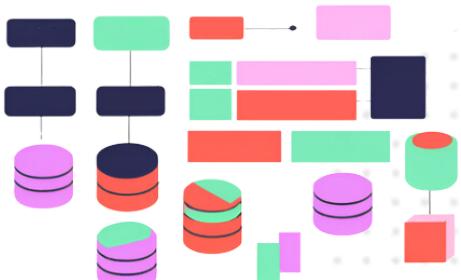


Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

Tabela Resultado Esperado:

id_pedido	nome	data_pedido	valor
1	Ana Souza	2024-01-10	500.00
2	Carlos Lima	2024-02-15	1200.00
3	Júlia Melo	2024-03-05	300.00
4	Ana Souza	2024-04-20	450.00



2) Complete a consulta abaixo para exibir todos os clientes, incluindo aqueles que não possuem pedidos registrados.

SQL

```
SELECT p.id_pedido,  
       c.nome,  
       p.data_pedido,  
       p.valor  
  FROM pedidos AS p  
LEFT JOIN clientes AS c  
    ON p.id_cliente = c.id_cliente;
```

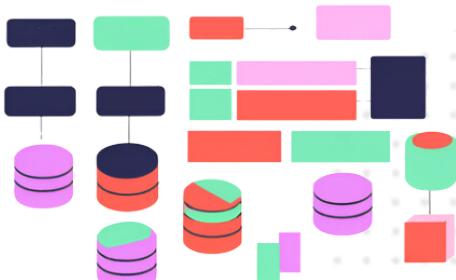


Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

Determine a Tabela Resultado Esperado.

id_pedido	nome	data_pedido	valor
1	Ana Souza	2024-01-10	500.00
2	Carlos Lima	2024-02-15	1200.00
3	Júlia Melo	2024-03-05	300.00
4	Ana Souza	2024-04-20	450.00
NULL	Marcos Silva	NULL	NULL



3) Complete a consulta abaixo para selecionar os pedidos feitos por clientes que moram em São Paulo ou Brasília.

SQL

```
SELECT p.id_pedido,  
       c.nome,  
       c.cidade,  
       p.valor  
FROM pedidos AS p  
INNER JOIN clientes AS c  
ON p.id_cliente = c.id_cliente  
WHERE c.cidade = 'São Paulo'  
      OR c.cidade = 'Brasília';
```

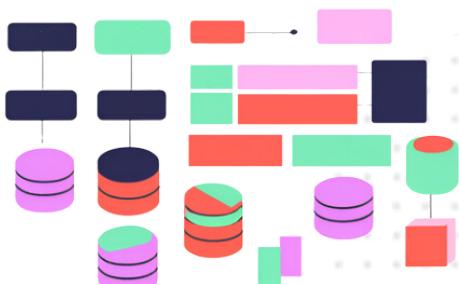


Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

Determine a Tabela Resultado Esperado.

id_pedido	nome	cidade	valor
1	Ana Souza	Brasília	500.00
2	Carlos Lima	São Paulo	1200.00
4	Ana Souza	Brasília	450.00



4) Complete a consulta abaixo para exibir todos os pedidos, incluindo aqueles sem clientes cadastrados.

SQL

```
SELECT p.id_pedido,  
       c.nome,  
       p.data_pedido,  
       p.valor  
FROM pedidos AS p  
RIGHT JOIN clientes AS c  
ON p.id_cliente = c.id_cliente;
```

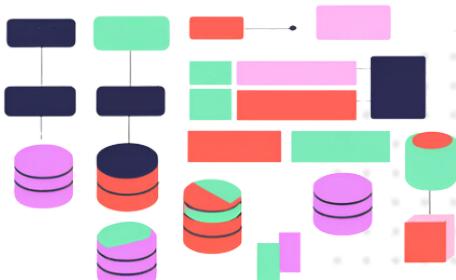


Os exercícios foram todos respondidos nesta versão.

Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

Tabela Resultado Esperado.

id_pedido	nome	data_pedido	valor
1	Ana Souza	2024-01-10	500.00
2	Carlos Lima	2024-02-15	1200.00
3	Júlia Melo	2024-03-05	300.00
4	Ana Souza	2024-04-20	450.00
5	NULL	2024-05-10	700.00



5) Complete a consulta abaixo para selecionar os pedidos com valor acima de 500 reais e de clientes que moram em São Paulo.



Os exercícios foram todos respondidos nesta versão.

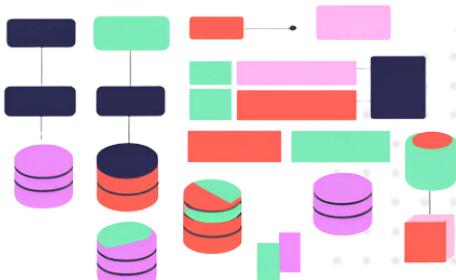
Fique à vontade para adaptar as questões e adicionar outros tópicos conforme achar conveniente!

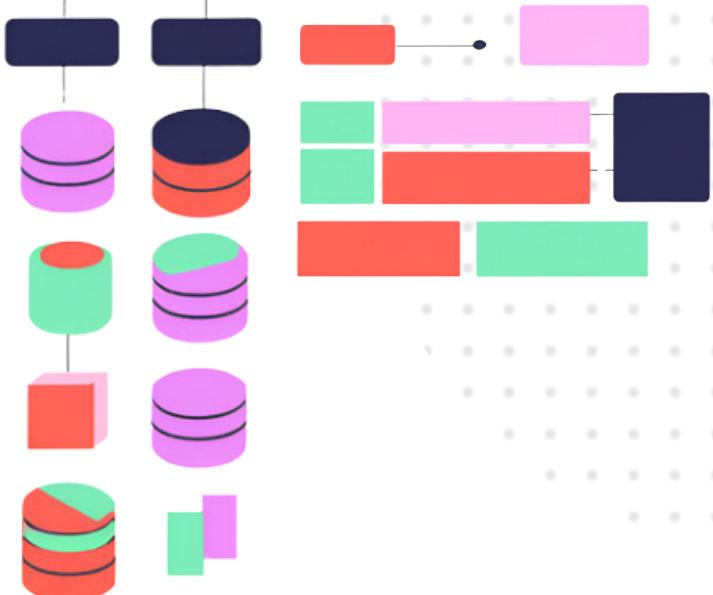
SQL

```
SELECT p.id_pedido,  
       c.nome,  
       c.cidade,  
       p.valor  
  FROM pedidos AS p  
INNER JOIN clientes AS c  
    ON p.id_cliente = c.id_cliente  
 WHERE p.valor > 500  
   AND c.cidade = 'São Paulo';
```

Tabela Resultado Esperado.

id_pedido	nome	cidade	valor





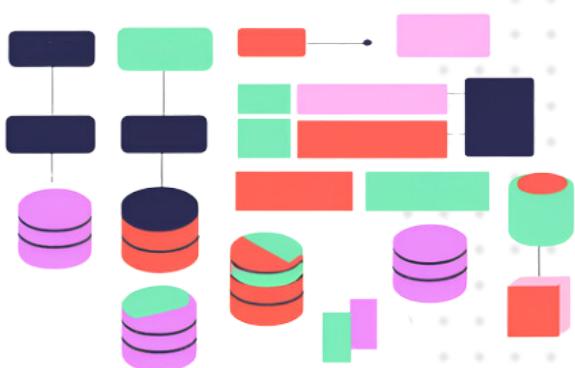
Chegamos ao fim...

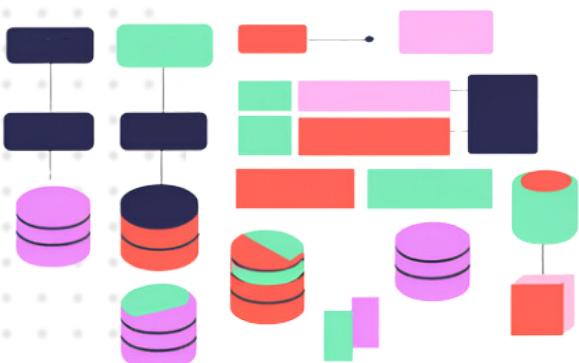
Chegamos ao fim da jornada deste livro, *De Conjuntos a Consultas: Construindo Pontes entre a Matemática e o SQL*, e espero que tenha sido uma experiência tão enriquecedora para você quanto foi para mim!

Ao longo dessas páginas, sistematizamos a teoria dos conjuntos como base para compreender a construção de consultas em SQL, estabelecendo conexões entre a Matemática e o mundo de Bancos de Dados.

Agradeço imensamente por dedicar seu tempo e interesse a este estudo. Espero que este livro tenha proporcionado não apenas conhecimento técnico, mas também inspiração para aprofundar-se ainda mais na fascinante interseção entre a Matemática e a Ciência de Dados.

Meus sinceros agradecimentos, também, aos professores Maria de Fátima Ramos Brandão e Jonathan Rosa Moreira, que orientaram a elaboração deste trabalho e contribuíram significativamente para sua produção, com rigor técnico e sempre preservando o objetivo pedagógico.





Que possamos continuar estudando, questionando e aprimorando essa produção.

“Ensinar inexiste sem aprender e vice-versa e foi aprendendo socialmente que, historicamente, mulheres e homens descobriram que era possível ensinar.” - Paulo Freire

Até a próxima versão!

Com gratidão,
Laíssa Soares.

REFERÊNCIA BIBLIOGRÁFICA

BRASIL. Ministério da Educação. *Base Nacional Comum Curricular*. Brasília, DF: MEC, 2017. Disponível em: <http://portal.mec.gov.br/>. Acesso em: 10 dez. 2024.

BRASIL. Ministério da Educação. *Base Nacional Comum Curricular da Computação*. Brasília, DF: MEC, 2022. Disponível em: <http://portal.mec.gov.br/docman/fevereiro-2022-pdf/236791-anexo-ao-parecer-cneceb-n-2-2022-bncc-computacao/file>. Acesso em: 10 dez. 2024.

NIELD, Thomas. *Introdução à Linguagem SQL: Abordagem prática para iniciantes*. 1. ed. Rio de Janeiro: Alta Books, 2016. 240 p. ISBN 9788575225014.

GONÇALVES, Eduardo. *SQL: Uma abordagem para banco de dados Oracle*. 1. ed. Curitiba: Editora Appris, 2016. 150 p. ISBN 9788555190556.

IEZZI, Gelson; MURAKAMI, Carlos. *Fundamentos de Matemática Elementar - Volume 1: Conjuntos e Funções*. 9. ed. São Paulo: Saraiva, 2013. 328 p. ISBN 9788535716804.

UNIVERSIDADE DE SÃO PAULO. *Introdução à Lógica e Conjuntos - PLC0001*. São Paulo: Universidade de São Paulo, [s.d.]. Disponível em: https://midia.atp.usp.br/plc/plc0001/impressos/plc0001_01.pdf. Acesso em: 10 dez. 2024.



DE CONJUNTOS A CONSULTAS

CONSTRUINDO
PONTES ENTRE
A MATEMÁTICA
E O SQL.

