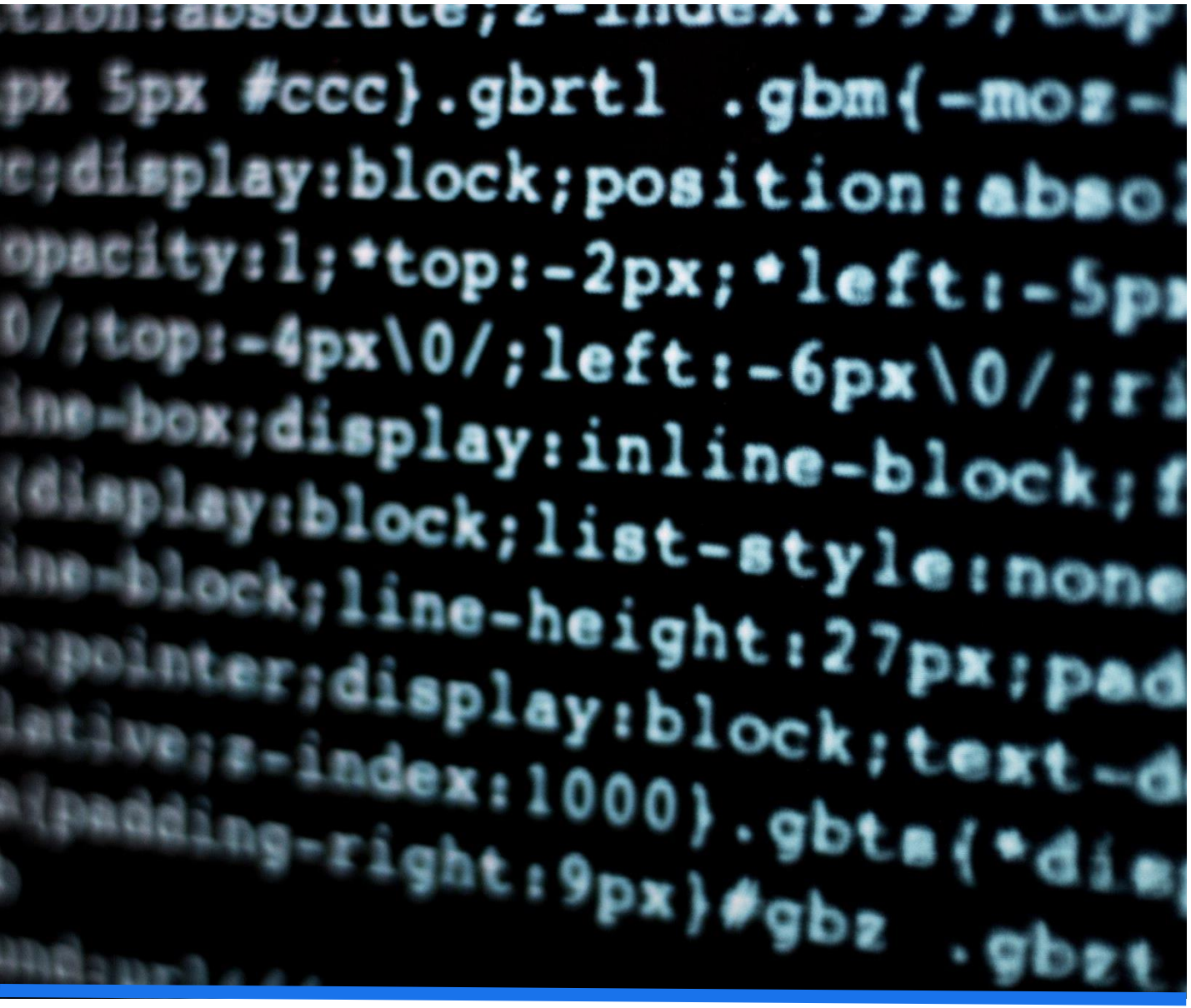


Consultas Avançadas com SQL: Técnicas Avançadas para Reforçar o SELECT

EXPLORAÇÃO DE RECURSOS SOFISTICADOS PARA CONSULTAS EFICIENTES

Resumo da Agenda

- Filtros avançados para seleção de dados
- Ordenação e limitação de resultados
- Operadores de conjunto em SQL
- Subconsultas (subqueries) em SELECT
- Funções de agregação para análise de dados
- Cláusula HAVING: filtrando resultados agregados

Filtros avançados para seleção de dados

Utilizando IN para múltiplos critérios de seleção

Função do operador IN

O operador IN permite selecionar registros que correspondem a qualquer valor em uma lista pré-definida, facilitando consultas complexas.

Aplicação em múltiplos critérios

Utilizar o operador IN simplifica a seleção de dados que atendem a vários critérios em uma única coluna de forma eficiente.

Explorando BETWEEN para intervalos de valores

Definição do BETWEEN

BETWEEN seleciona valores dentro de um intervalo, incluindo os valores extremos do intervalo.

Aplicações Comuns

É especialmente útil para filtrar datas, números e dados com faixas definidas em consultas.

Aplicando LIKE para buscas por padrão

Uso do caractere %

O caractere % representa múltiplos caracteres em buscas, permitindo maior flexibilidade na pesquisa de texto.

Uso do caractere _

O caractere _ substitui um único caractere em buscas, facilitando encontrar variações específicas em texto.

Busca flexível com LIKE

LIKE permite criar buscas por padrão em colunas textuais, ideal para consultas dinâmicas e adaptáveis.

Tratamento de valores nulos com IS NULL

Identificação de valores nulos

O comando IS NULL permite localizar registros com valores nulos em colunas específicas para análise precisa.

Tratamento de dados ausentes

Usar IS NULL ajuda a lidar com dados incompletos, garantindo qualidade na seleção e manipulação de informação.

Verificação de existência com EXISTS

Função do EXISTS

EXISTS confirma se subconsulta retorna pelo menos um registro, validando dados relacionados.

Uso em consultas SQL

Permite executar ações na consulta principal somente quando dados relacionados existem.

Ordenação e limitação de resultados

Classificando dados com ORDER BY

Organização dos Resultados

ORDER BY organiza os dados retornados de forma ordenada, melhorando a análise dos resultados.

Ordenação por Múltiplas Colunas

É possível ordenar os resultados usando uma ou mais colunas para maior precisão na visualização.

Ordem Crescente e Decrescente

ORDER BY permite definir ordem crescente (ASC) ou decrescente (DESC) conforme a necessidade.

Restringindo o número de resultados com LIMIT

Função do LIMIT

LIMIT restringe o número máximo de registros retornados por uma consulta SQL.

Visualização de Amostras

Usar LIMIT permite visualizar amostras de dados sem retornar grandes volumes.

Paginação de Resultados

LIMIT é útil para paginar resultados em aplicações web ou sistemas de dados.

Navegação por resultados com OFFSET

Função do OFFSET

OFFSET pula registros iniciais em um conjunto de dados para controlar os resultados exibidos.

Uso combinado com LIMIT

OFFSET é usado com LIMIT para permitir paginação e navegação eficiente nos dados.

Operadores de conjunto em SQL

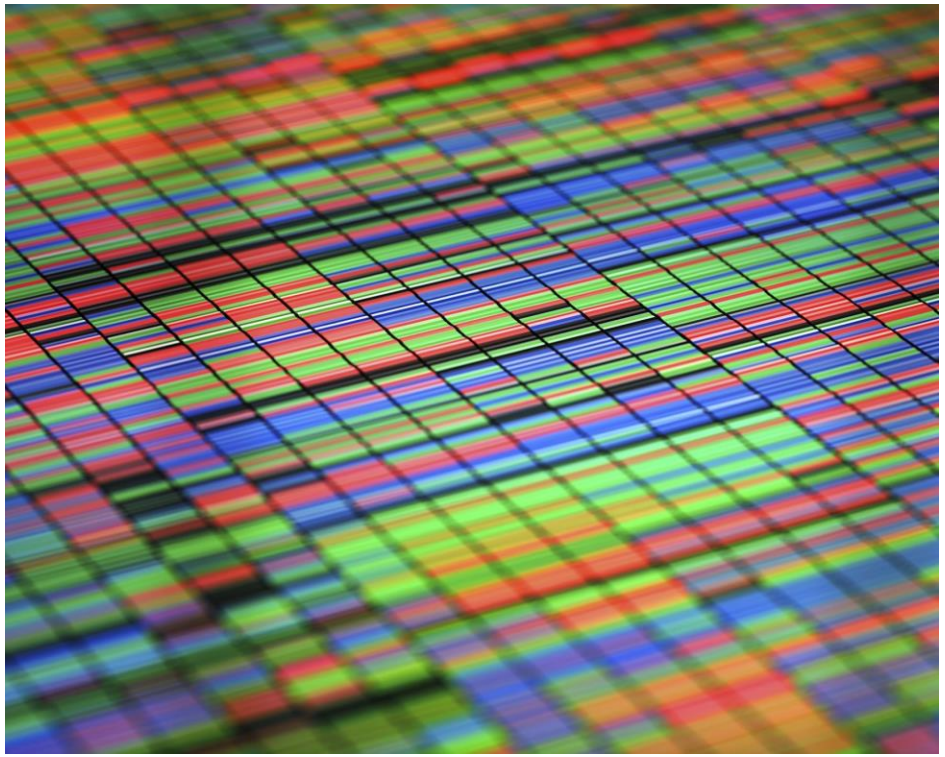
Unindo conjuntos de dados com UNION

Função do UNION

UNION combina resultados de múltiplas consultas SQL eliminando linhas duplicadas para consolidar dados.

Consolidação de Dados

Permite juntar dados relacionados de diferentes fontes em uma única saída clara e organizada.



Encontrando interseções com INTERSECT

Função do INTERSECT

INTERSECT retorna registros que aparecem em múltiplas consultas SQL distintas, mostrando apenas dados comuns.

Utilidade Prática

Útil para identificar dados compartilhados entre várias tabelas ou condições de busca complexas.

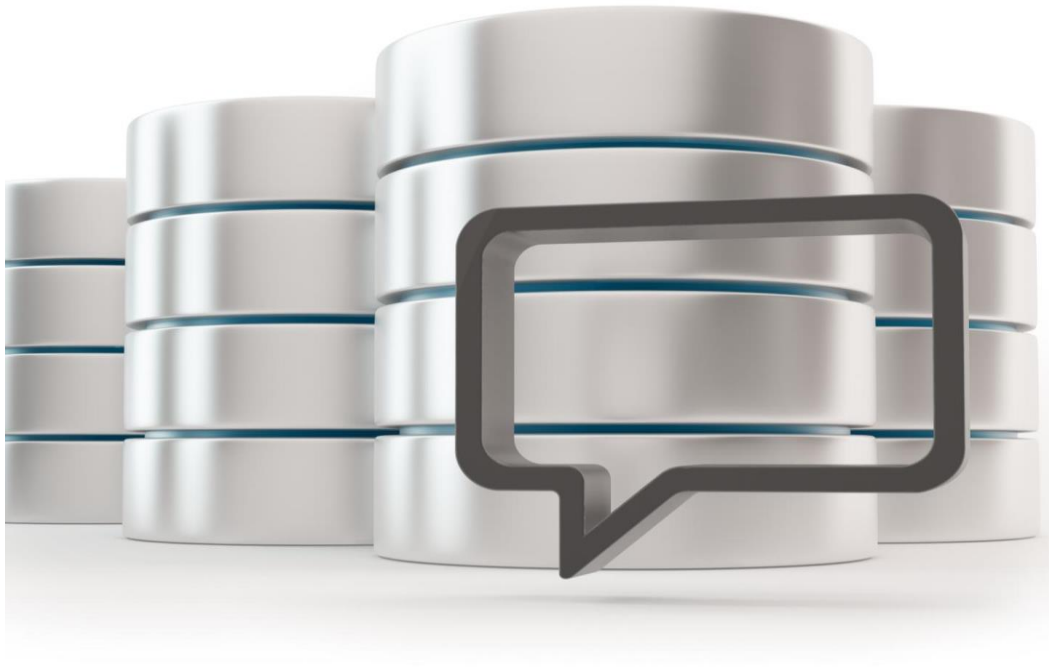
Exemplo:

```
CREATE TABLE t1 (
id INT PRIMARY KEY
);

CREATE TABLE t2 LIKE t1;

INSERT INTO t1(id) VALUES (1), (2), (3);
INSERT INTO t2(id) VALUES (2), (3), (4);

SELECT id FROM t1
INTERSECT
SELECT id FROM t2;
```



Diferenças entre EXCEPT e implementações no MySQL

Função do EXCEPT

EXCEPT retorna registros únicos de uma consulta que não existem em outra consulta.

Limitação no MySQL

MySQL não suporta EXCEPT nativamente, exigindo métodos alternativos para resultados semelhantes.

Simulação com LEFT JOIN e WHERE NULL

LEFT JOIN combinado com WHERE NULL permite simular a funcionalidade do EXCEPT no MySQL.

Subconsultas (subqueries) em SELECT

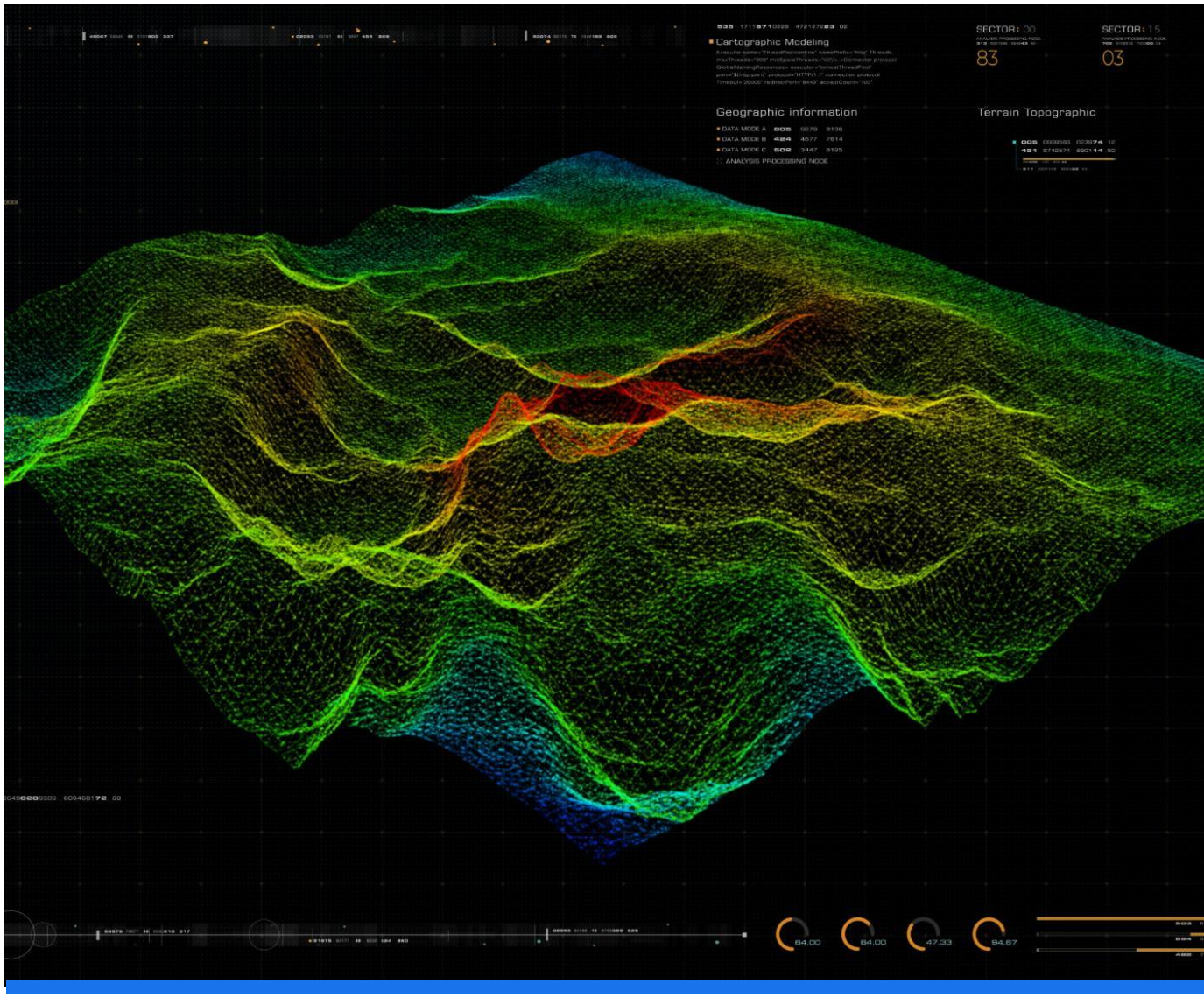
Subconsultas não correlacionadas

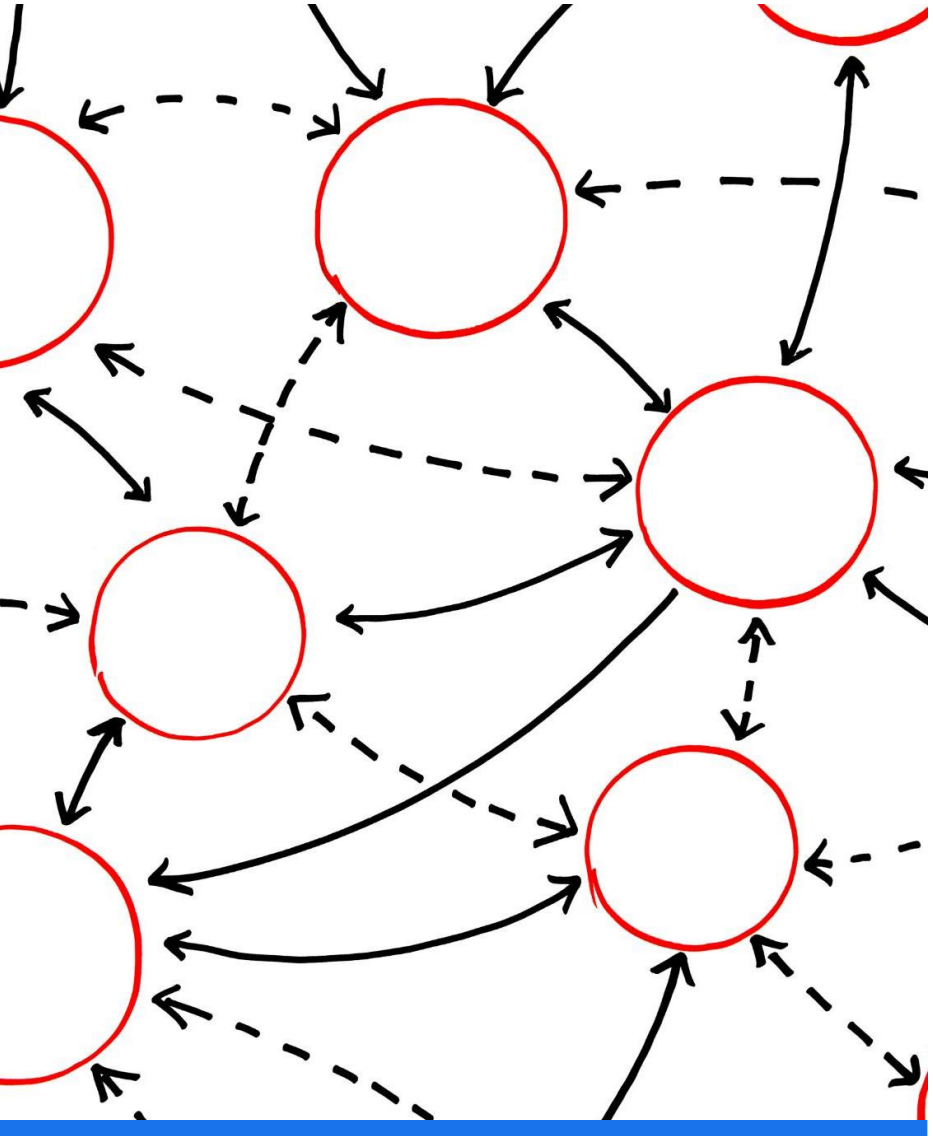
Definição de Subconsulta Não Correlacionada

São consultas internas independentes da consulta principal, executadas uma única vez.

Uso do Resultado

O resultado da subconsulta é utilizado para filtrar ou calcular dados na consulta principal.





Subconsultas correlacionadas

Dependência de Valores Externos

Subconsultas correlacionadas usam valores da consulta externa para execução de forma dependente.

Avaliação por Linha

São avaliadas para cada linha da consulta principal, permitindo análises detalhadas e específicas.

Comparações Complexas

Permitem realizar comparações complexas e específicas entre dados relacionados.

Boas práticas e eficiência na utilização de subconsultas

Evitar Subconsultas Desnecessárias

Evitar subconsultas pode reduzir a complexidade e melhorar o desempenho das consultas SQL.

Preferir JOINs Eficientes

Usar JOINs em vez de subconsultas geralmente resulta em execuções mais rápidas e eficientes.

Garantir Índices Adequados

Índices apropriados aceleram a recuperação de dados e melhoram o desempenho geral das consultas.

Funções de agregação para análise de dados

Somando valores com SUM

Função SUM Básica

SUM agrega valores de uma coluna numérica para obter o total de um grupo de registros.

Importância para Análise

SUM é fundamental em análises financeiras e quantitativas para calcular totais precisos.

Calculando médias com AVG

Função AVG

AVG calcula a média aritmética dos valores contidos em uma coluna específica de dados.

Análise de Tendências

O uso de AVG facilita a identificação de tendências e padrões em conjuntos de dados complexos.

Identificando valores extremos com MAX e MIN

Função MAX

MAX identifica o maior valor em um conjunto de dados para destacar limites superiores claros.

Função MIN

MIN encontra o menor valor, ajudando a definir limites inferiores e detectar valores extremos baixos.

Identificação de Outliers

MAX e MIN ajudam a localizar valores fora do padrão, auxiliando na análise e limpeza dos dados.

Concatenando resultados com GROUP_CONCAT

Função GROUP_CONCAT

GROUP_CONCAT combina valores de várias linhas em uma única string separada por vírgulas.

Aplicação prática

Essa função é útil para criar listas compactas e resumidas de informações relacionadas.

Exemplo do operador IN

- Seleciona registros com valores múltiplos específicos em uma coluna.
- Consulta: SELECT * FROM tabela WHERE coluna IN ('valor1', 'valor2', 'valor3');
- Útil para filtrar dados sem usar múltiplos OR.

Exemplo do operador BETWEEN

- Seleciona valores dentro de um intervalo definido.
- Consulta: SELECT * FROM tabela WHERE data BETWEEN '2023-01-01' AND '2023-12-31';
- Muito usado para filtrar datas, números ou faixas contínuas.

Exemplo do operador LIKE

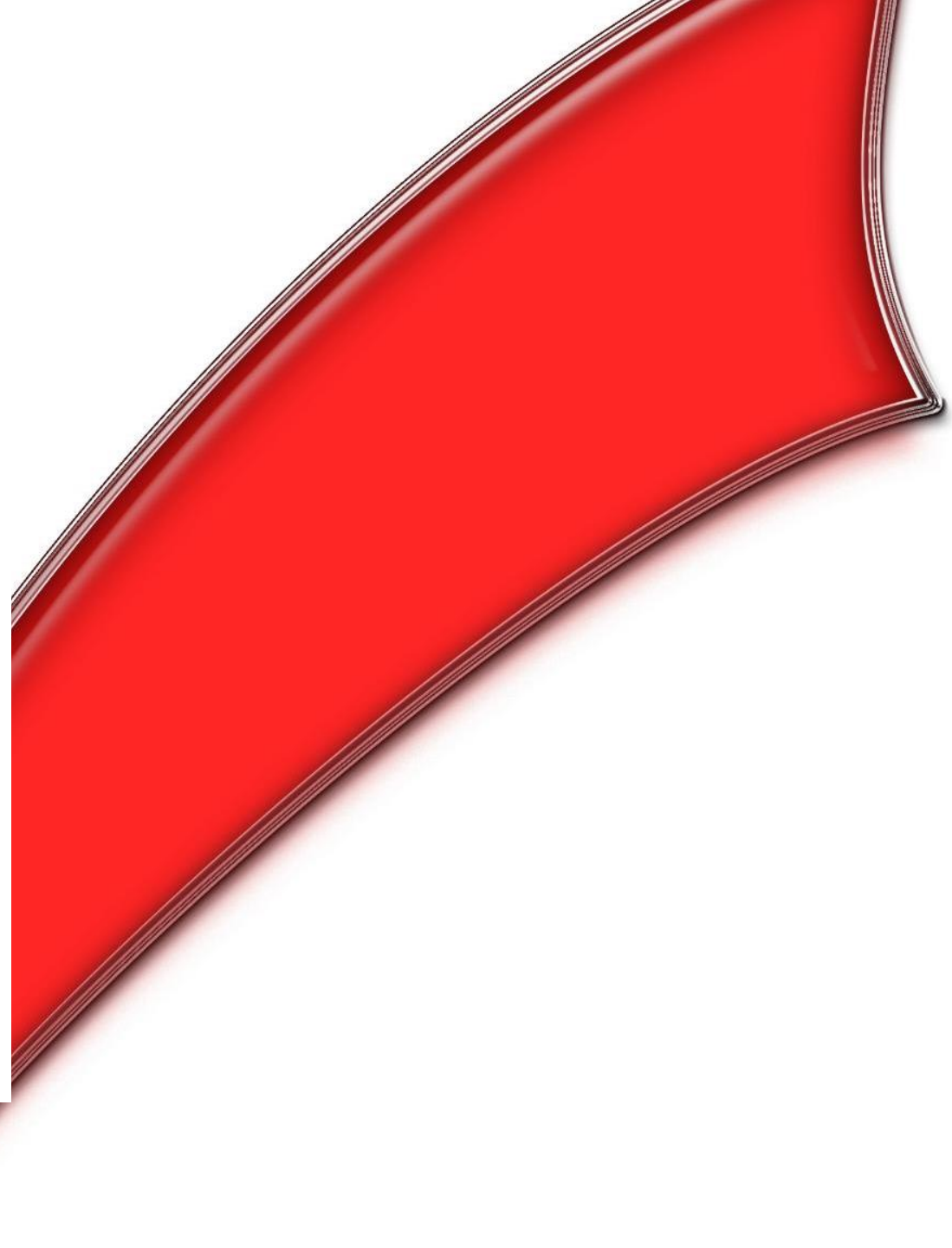
- Usa caracteres curinga para busca por padrão em texto.
- Consulta: SELECT * FROM tabela WHERE nome LIKE 'Jo%';
- Permite buscas flexíveis por prefixos, sufixos ou padrões.

Exemplo do tratamento IS NULL

- Seleciona registros onde um campo está vazio ou nulo.
- Consulta: SELECT * FROM tabela WHERE coluna IS NULL;
- Importante para identificar dados ausentes ou incompletos.

Exemplo do EXISTS

- Verifica se subconsulta retorna algum registro.
- Consulta: SELECT * FROM tabela1 WHERE EXISTS (SELECT 1 FROM tabela2 WHERE condição);
- Usado para validar existência antes de executar operações.



Cláusula HAVING: filtrando resultados agregados

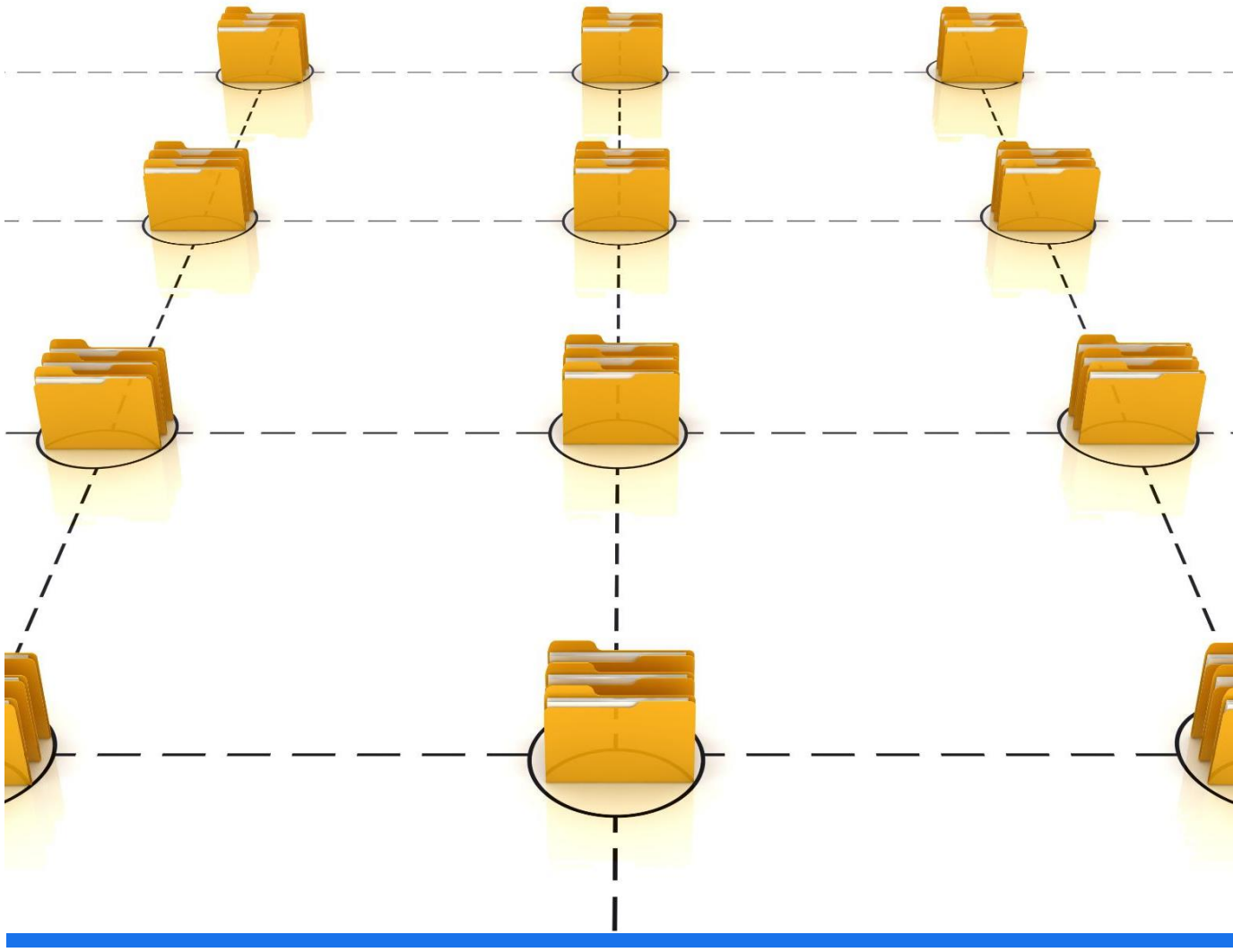
Diferenciando WHERE e HAVING

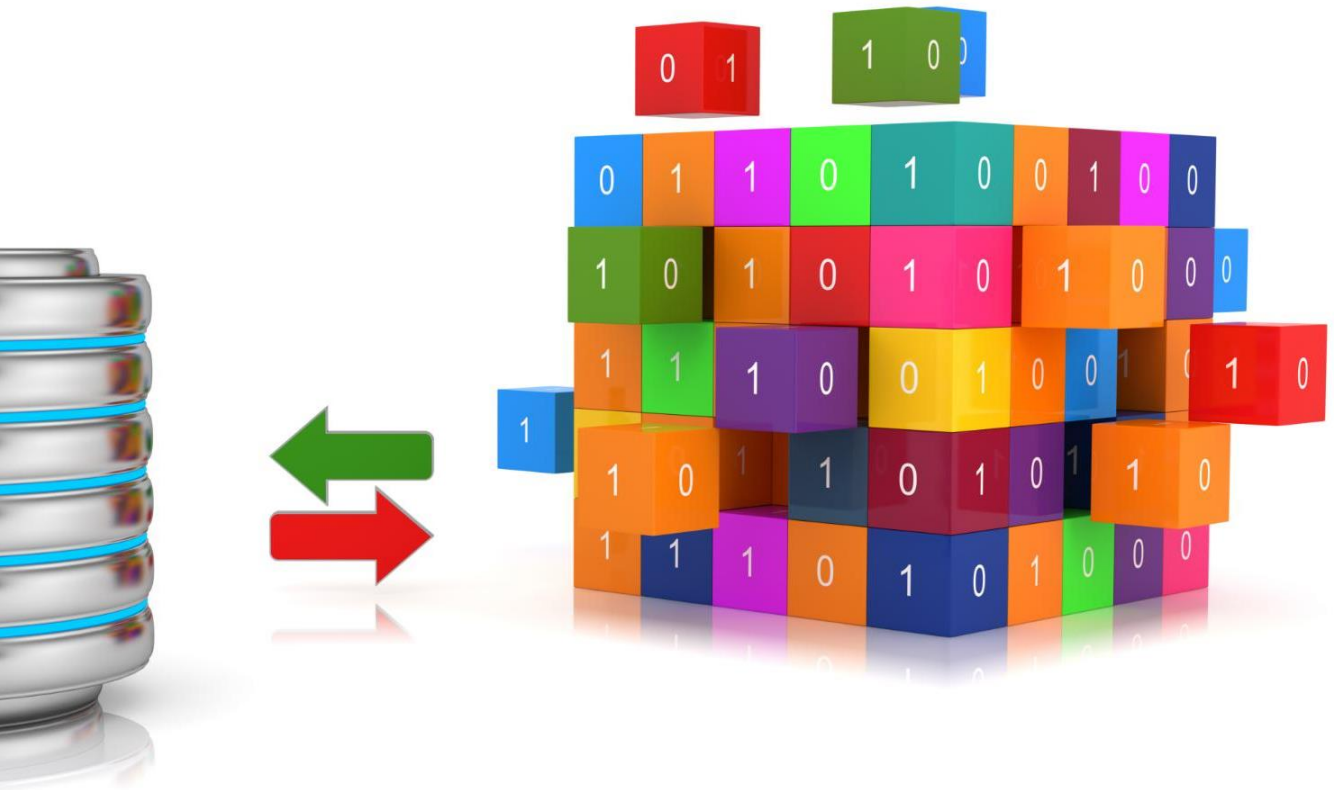
Função do WHERE

WHERE filtra linhas individualmente antes da agregação ser aplicada nos dados, limitando os registros processados.

Função do HAVING

HAVING filtra grupos de dados após a aplicação de funções de agregação em consultas com GROUP BY.





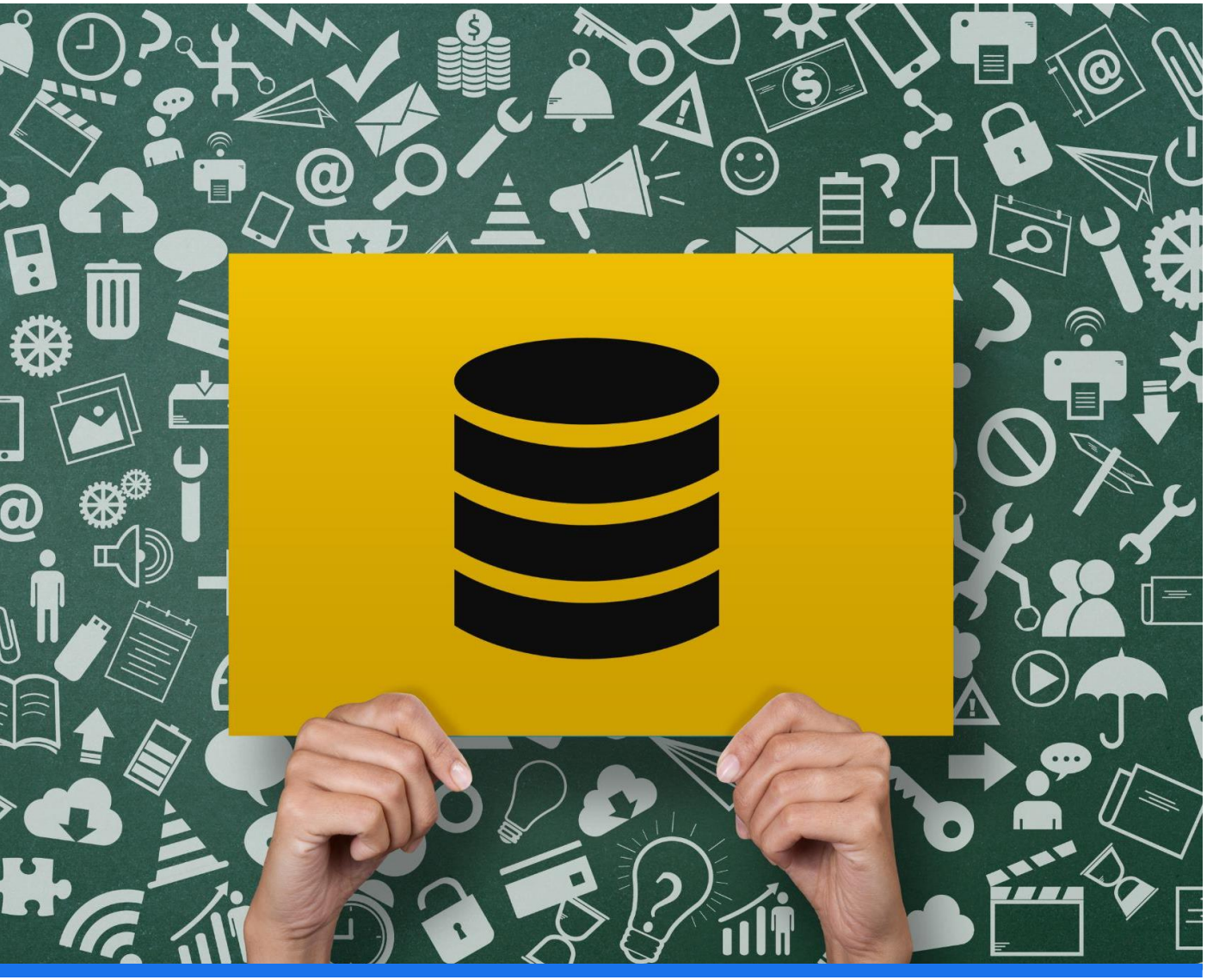
Aplicando filtros após agregações

Uso do HAVING

HAVING permite filtrar grupos após a agregação baseada em condições específicas, como soma ou média.

Filtragem de Grupos

É utilizado para selecionar grupos que atendem a critérios, por exemplo, soma maior que um valor definido.



Exemplos práticos de uso da cláusula HAVING

Filtragem de dados agregados

A cláusula HAVING permite filtrar agrupamentos de dados após a aplicação de funções agregadas em consultas SQL.

Aplicação em relatórios

HAVING é útil para refinar relatórios, mostrando apenas grupos que atendem a critérios específicos após agregação.

Conclusão

Consultas SQL Poderosas

Técnicas avançadas permitem criar consultas SQL eficientes para manipulação de dados complexos.

Aplicação de Filtros e Ordenação

Filtros e ordenações facilitam a extração de informações específicas dentro dos dados.

Funções de Agregação e HAVING

Funções de agregação combinadas com HAVING ajudam a resumir dados e identificar padrões relevantes.