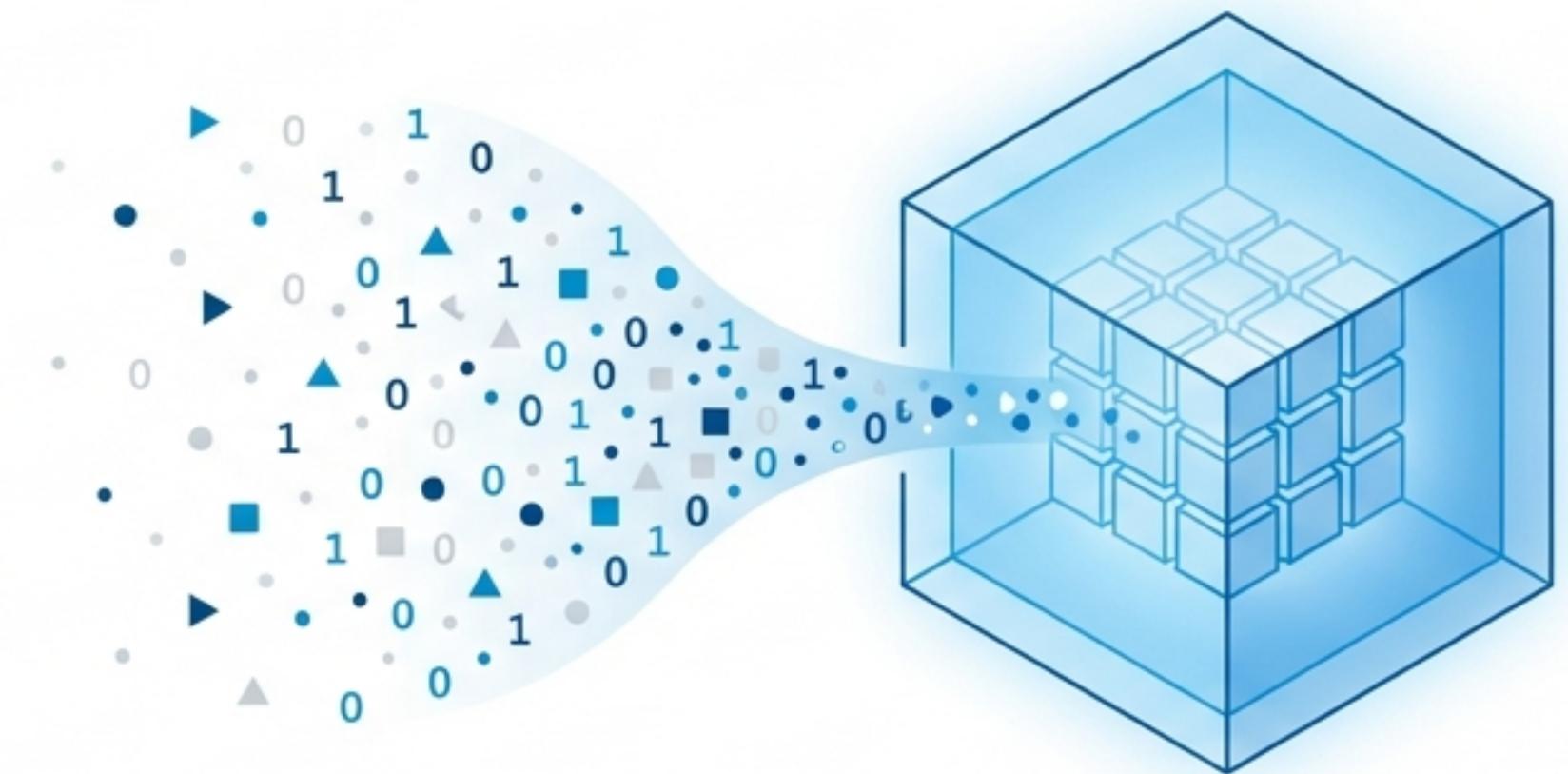


Usando Code Pushdown no ABAP SQL

Módulo 04 | Aula 04: Otimizando a Lógica no Banco de Dados

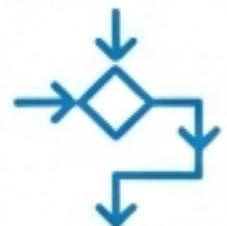
Senior Solution Architect Series



Objetivos de Aprendizagem



Aplicar expressões aritméticas e de string diretamente no SELECT, eliminando pós-processamento.



Utilizar lógica condicional com CASE para transformar códigos técnicos em regras de negócio na fonte.



Realizar agregações (SUM, AVG, MIN, MAX) e agrupamentos obrigatórios (GROUP BY).



Combinar seleções heterogêneas com UNION e UNION ALL, entendendo impactos de performance.

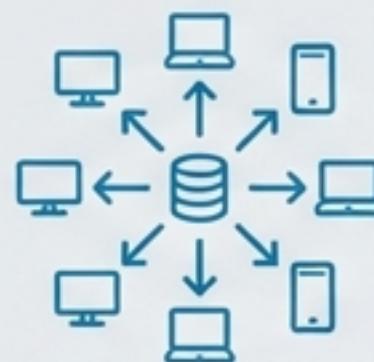


Garantir robustez em cálculos matemáticos usando tratamento de nulos (COALESCE).

Decisão Arquitetural: ABAP SQL ou CDS Views?

O dilema da reutilização versus especificidade.

CDS Views



- Single Source of Truth
- Ideal para modelos reutilizáveis.
- Exemplo: Cálculo de impostos consumido por 3 relatórios e 1 Fiori App.
- Regra: Se a lógica é compartilhada, use CDS.

ABAP SQL Moderno

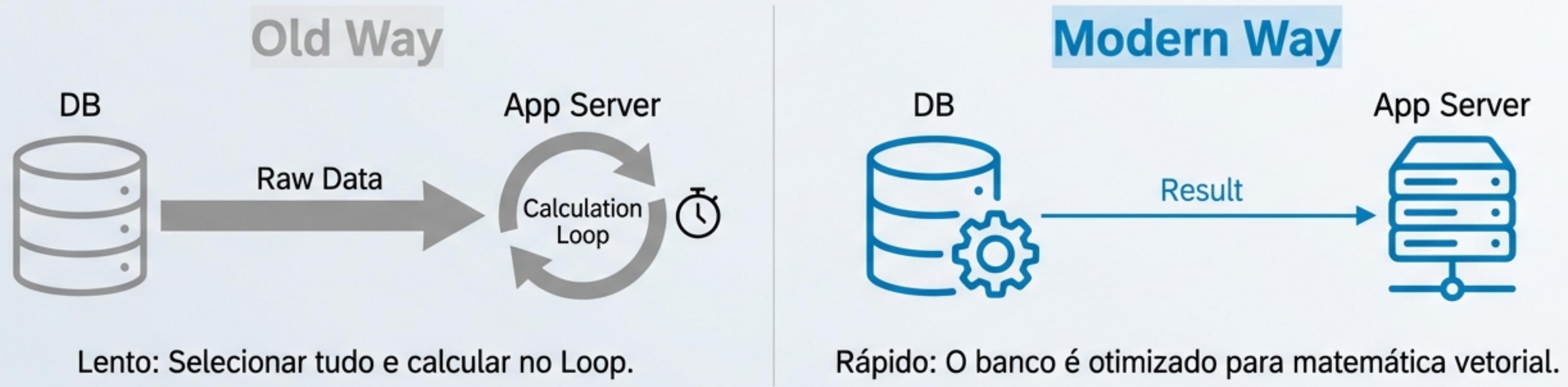


- Lógica Ad-hoc e Específica
- Ideal para processamento batch único ou lógica exclusiva de um método.
- Regra: Não polua o dicionário de dados com Views de uso único.



Nota do Arquiteto: O ABAP SQL (7.50+) herdou os ‘superpoderes’ das CDS Views. Prototipar no ABAP SQL facilita a refatoração futura para CDS se necessário.

Aritmética no SELECT (Code Pushdown)



Soma Simples

```
( total_price + booking_fee )  
AS grand_total
```



Casting & Tipagem

```
CAST( total_price * '0.9' AS  
CURR( 15, 2 ) )
```

Literal '0.9' (10% desconto). CAST garante formato moeda.



Divisão

```
DIVISION( total_price, 100, 2  
)
```

DIVISION para inteiros, operador / para decimais.

Ponto de Atenção: O Perigo dos Nulos

5 + NULL = NULL

Um campo vazio pode anular todo o cálculo de uma linha.

A Solução: **COALESCE(campo, 0)**

Retorna o primeiro valor não nulo da lista de argumentos.

Indispensável em operações aritméticas para garantir a integridade da query.

(total_price + **COALESCE(booking_fee, 0)**) AS final_amount

Manipulação de Strings no Banco

Elimine Loops ABAP para tratamento de texto. Use funções nativas.

Concatenação

```
first_name && last_name
```

```
concat_with_space(  
    first_name,  
    last_name,  
    last_name, 1  
)
```

João
Silva

Normalização

```
upper( last_name )
```

Padronização para buscas
insensíveis a maiúsculas.

Extração

```
substring( last_name, 1, 3 )
```

Recorte de texto posicional.

Lógica Condicional (CASE)

Transforme códigos técnicos em informações de negócio na fonte.

Simple CASE (Tradução Direta)

Compara um campo contra valores fixos.

Complex CASE (Lógica de Valor)

Condições baseadas em comparadores (<, >, BETWEEN).

```
CASE overall_status
  WHEN 'A' THEN 'Aceito'
  WHEN 'X' THEN 'Rejeitado'
  ELSE 'Pendente'
END AS status_text
```

```
CASE
  WHEN total_price < 1000 THEN 'Econômica'
  WHEN total_price > 5000 THEN 'Primeira Classe'
END AS category
```

Benefício: Elimina IF/ELSE no Loop. A informação chega pronta.

Aggregações e Agrupamento (GROUP BY)

Toolbox

SUM

AVG

MIN

MAX

COUNT

Raw Data

customer_id	currency_code	total_price
101	USD	100
101	USD	150
101	EUR	80
102	USD	200
102	USD	100



customer_id	currency_code	SUM(total_price)
101	USD	250
102	USD	300

Aggregated Data

Regra de Ouro

Se usar função de agregação, **TODO campo no FIELDS que não for agregado deve estar no GROUP BY**.

O banco precisa saber “somar o quê por quem”.

```
FIELDS customer_id, currency_code, SUM( total_price )  
... GROUP BY customer_id, currency_code
```

Non-aggregated fields
must be in GROUP BY

Filtragem Avançada: WHERE vs. HAVING

WHERE

Filtra dados **ANTES** de agrupar.

```
begin_date >= '20230101'
```

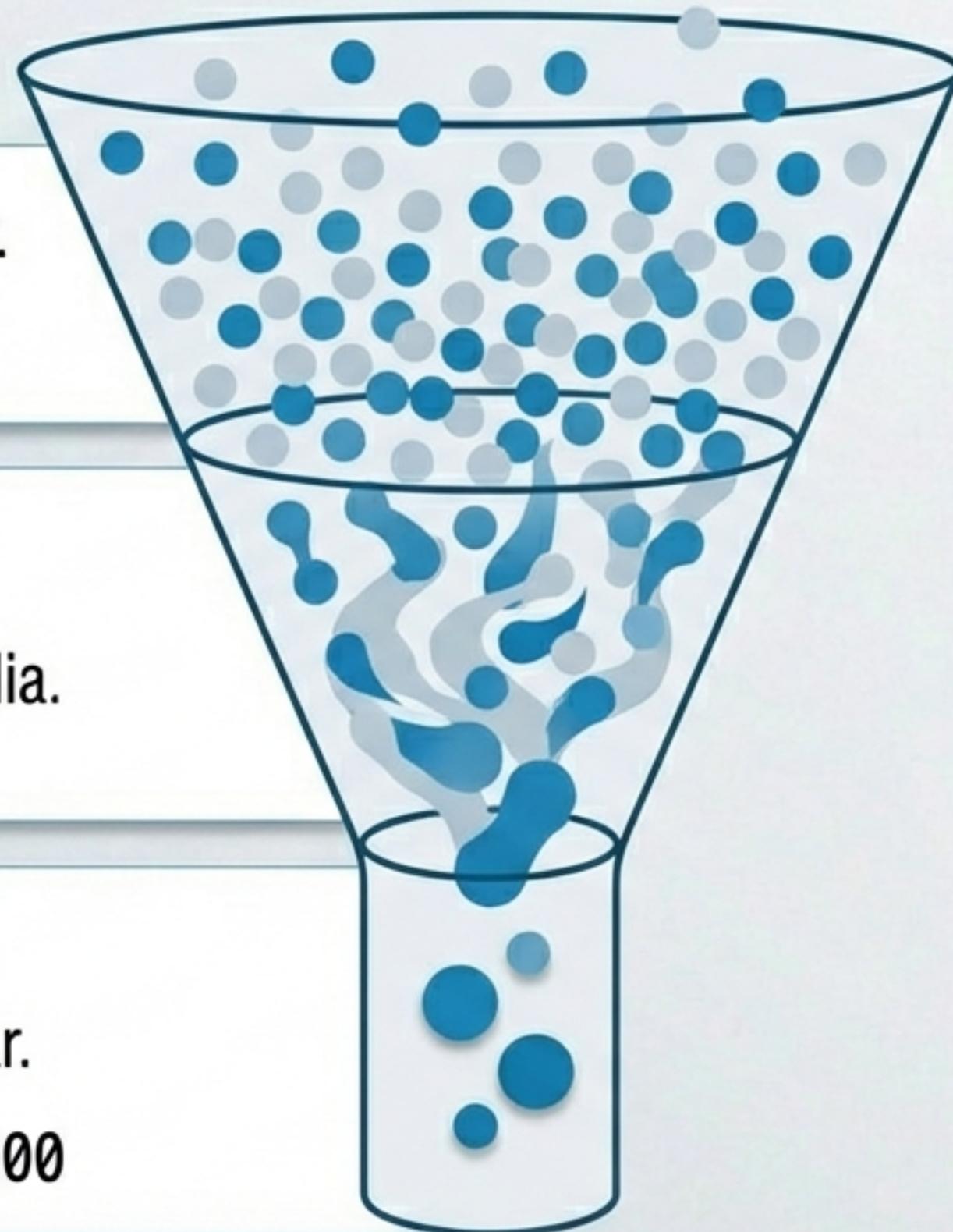
Aggregation

O banco processa a soma/média.

HAVING

Filtra dados **DEPOIS** de agrupar.

```
SUM( total_price ) > 50000
```



Dica de Performance:

Filtre o máximo possível no **WHERE** para reduzir o trabalho do agrupador.

Combinando Resultados (UNION)

Juntar dados de tabelas diferentes em uma única ida ao banco.

UNION ALL (Recomendado)

Alta Performance

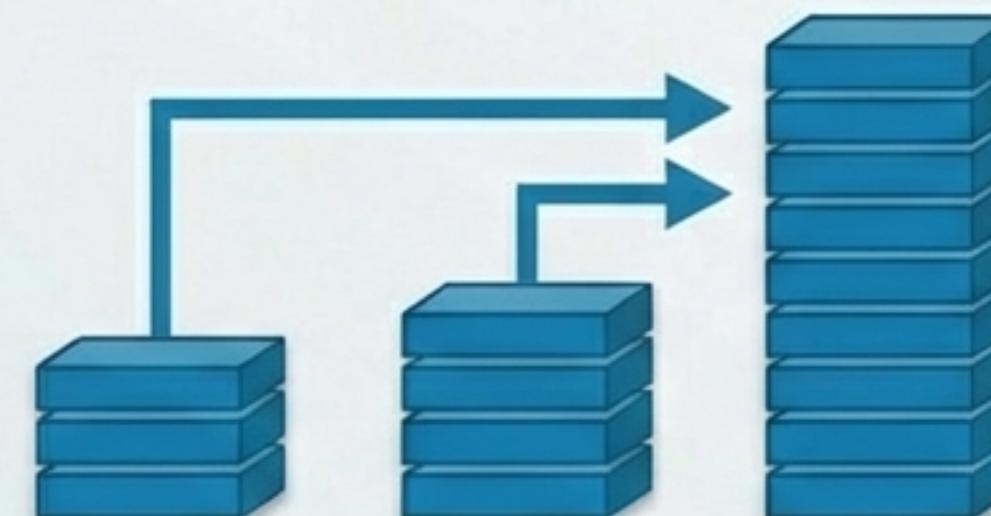


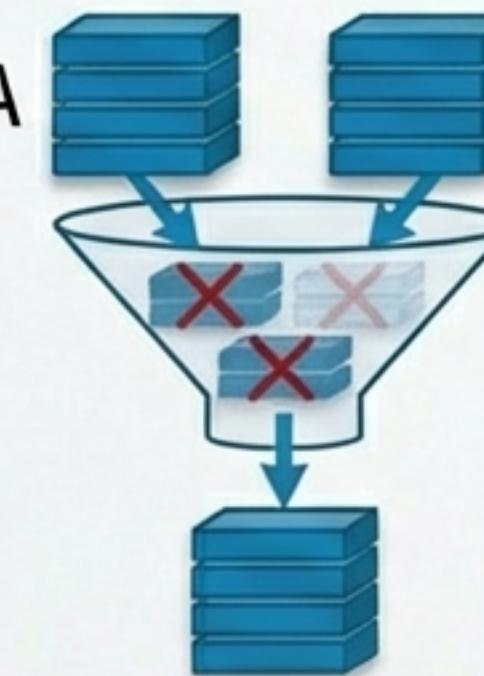
Tabela A Tabela B

Junta os resultados “como estão”.

UNION (Cuidado)

Processamento Custoso

Tabela A Tabela B



Junta, ordena e remove duplicatas.

```
SELECT ... FROM table_A  
UNION ALL  
SELECT ... FROM table_B
```

Exemplo Prático: Relatório Analítico

Análise de performance de agências (zcl_sql_pushdown)

```
SELECT FROM zrap_travel
FIELDS agency_id,
      currency_code,
      COUNT( * ) AS number_of_travels,
      SUM( total_price ) AS total_amount,
      AVG( total_price AS DEC( 15,2 ) ) AS average_ticket,
      CASE WHEN SUM( total_price ) > 100000
            THEN 'Platinum'
            ELSE 'Standard'
        END AS partner_category
GROUP BY agency_id, currency_code
HAVING SUM( total_price ) > 1000
ORDER BY total_amount DESCENDING
```

- 1 Agregação (Soma/Contagem)
- 2 Code Pushdown Lógico Lógico (Classificação)
- 3 Casting Interno para média precisa
- 4 Filtro Pós-Agregação (Remove irrelevantes)

Tabela Comparativa: Loop ABAP vs. Code Pushdown

Cenário	ABAP Clássico	ABAP SQL Moderno
Soma de Totais	Loop / Variável acumuladora (Lento)	SELECT SUM (Rápido/Índices)
Status Texto	Loop / IF-ELSE / Modificar tabela	SELECT CASE (Centralizado)
Juntar Tabelas	2 Selects / Loop / Append	UNION (Single Roundtrip)
Filtro de Soma	Loop / Delete se < X (Desperdício)	HAVING (Retorno Preciso)

Glossário Técnico

Aggregation Function

Operam em conjuntos para retornar valor resumido (SUM, AVG).

GROUP BY

Define os ‘baldes’ de agrupamento. Obrigatório ao misturar colunas e agregações.

COALESCE

Retorna o primeiro valor não nulo. Vital para evitar propagação de NULL.

Literals

Valores fixos (‘Ativo’, 100) usados diretamente na query.

Roundtrip

A viagem de ida e volta entre o Servidor de Aplicação e o Banco de Dados.

Quiz de Fixação

Q1: Qual a diferença de performance entre WHERE e HAVING?

Q3: Ao usar **SUM()**, o que fazer com os campos não somados?

Q2: Por que UNION ALL é mais rápido que UNION?

Q4: Em que cenário a função **COALESCE** é indispensável?

Deixe o banco de dados fazer o trabalho pesado.

Otimize seu ABAP.

Revise seus códigos antigos. Identifique onde Loops de cálculo
podem ser substituídos por SQL Moderno hoje mesmo.