

Evolução e Maestria

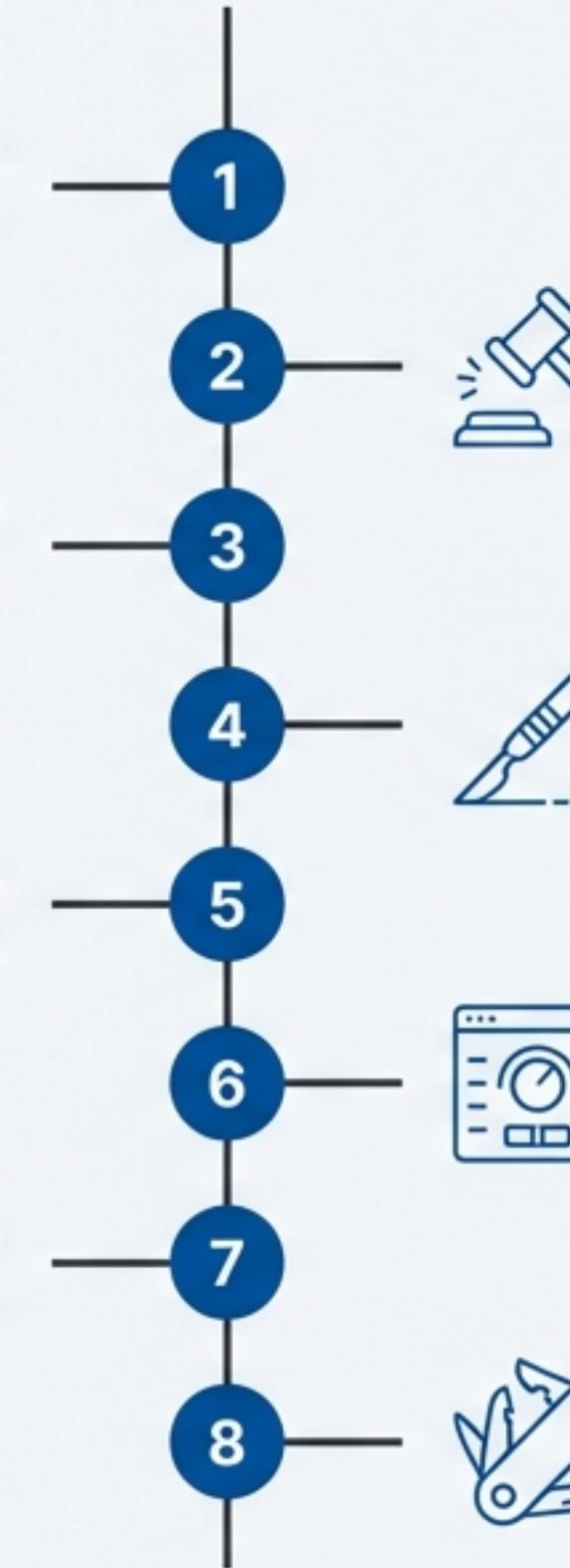
Dominando a Leitura de Dados
com o ABAP SQL Moderno

Baseado no ABAP Platform 7.40+ e otimizado para SAP HANA.

Nossa Rota para a Maestria



A Revolução do "Code Pushdown":
Entendendo o "porquê" da mudança.



Leitura de Múltiplas Linhas:
O padrão com 'INTO TABLE'.



O Detalhe Crucial: O papel do
símbolo '@' (Host Variables).



Código em Ação: Um exemplo
prático de ponta a ponta.



As Novas Regras do Jogo:
Sintaxe e o "Strict Mode" em detalhes.



A Leitura Cirúrgica:
'SELECT SINGLE' vs. 'UP TO 1 ROWS'.



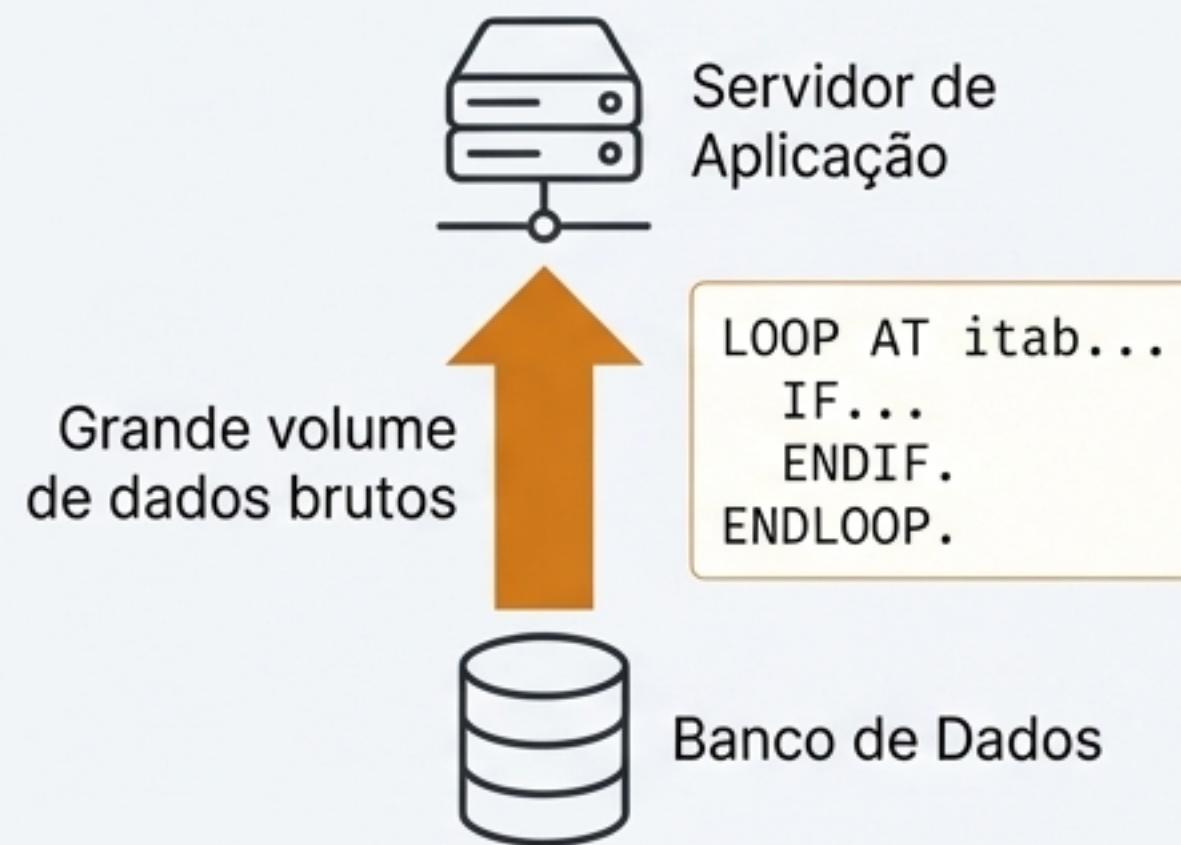
O Feedback do Banco:
Interpretando 'sy-subrc' e 'sy-dbcnt'.



Guia do Expert: Boas práticas e um
glossário essencial.

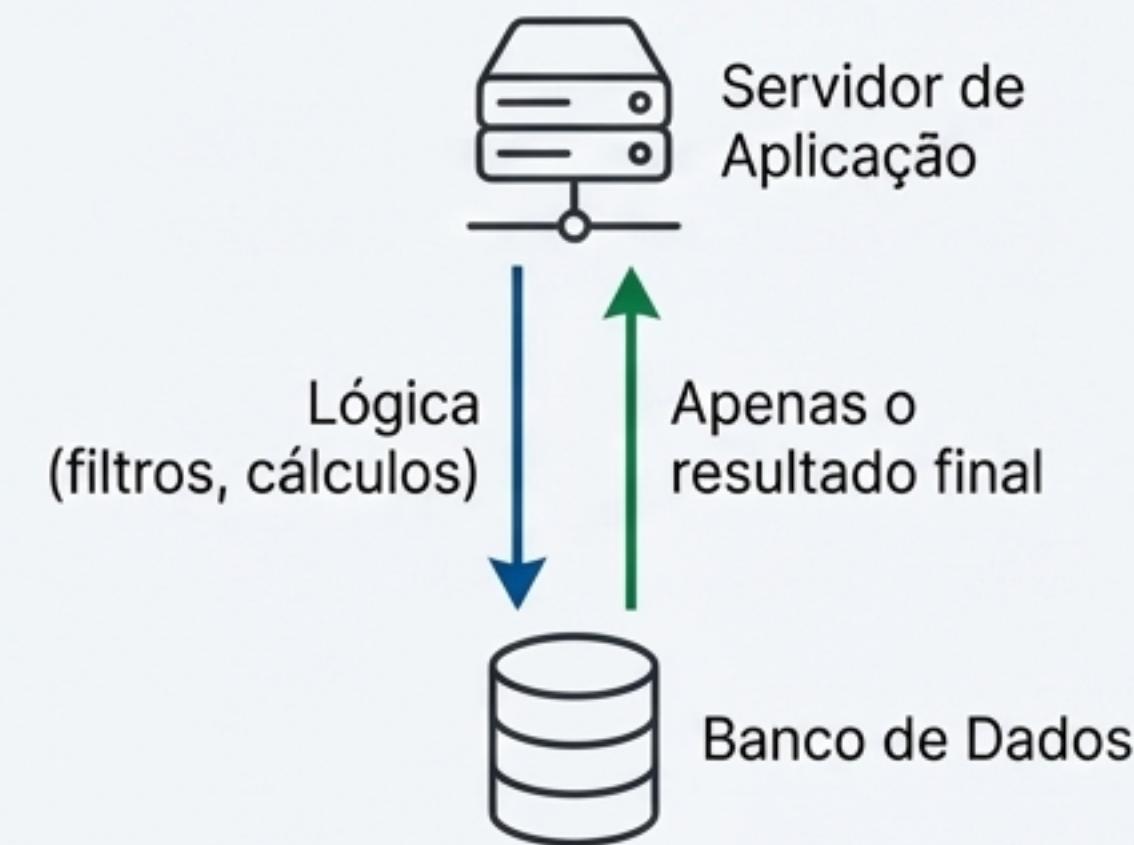
A Mudança de Paradigma: De "Trazer Dados" para "Processar na Fonte"

ABAP Clássico / Open SQL



Grande volume
de dados brutos

ABAP Moderno / Code Pushdown



O servidor ABAP puxava todos os dados para processá-los localmente.

A lógica é “empurrada” para o banco de dados (HANA), que processa e retorna apenas o necessário.

Com o ABAP 7.40+, o paradigma mudou para **Code Pushdown**. Em vez de trazer a montanha de dados até nosso código, levamos nosso código até os dados.

As Novas Regras do Jogo: Conheça o "Strict Mode"

O Modo Estrito não é uma opção. Ele é ativado automaticamente pelo compilador quando você usa qualquer funcionalidade moderna. Estas são as novas regras:

Campos Separados por Vírgula

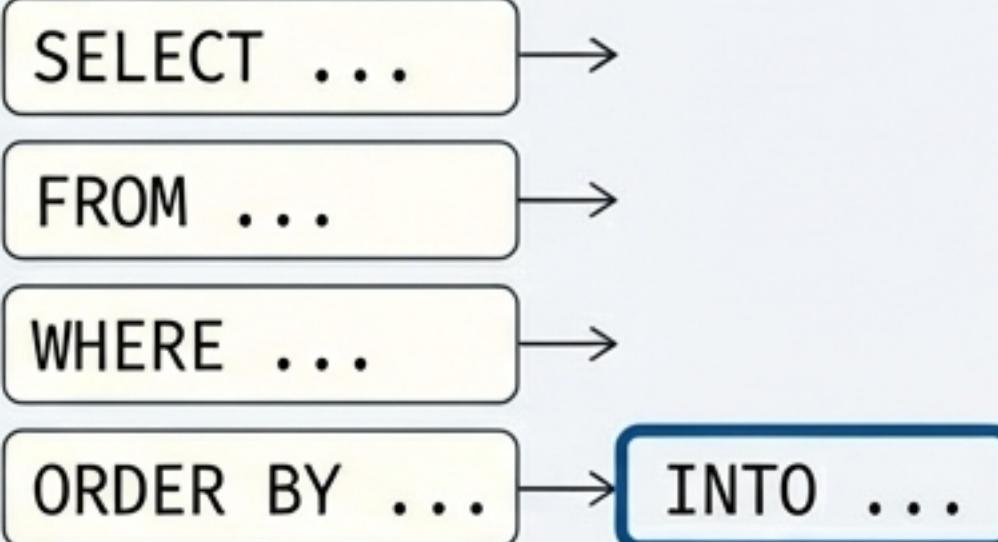
Exemplo Correto

```
SELECT f1, f2, f3...
```

Exemplo Incorreto

```
SELECT f1 > f2 f3...
```

Cláusula de Destino no Final



A cláusula '**INTO**' deve vir sempre após a seleção, filtragem e ordenação.

Escape para Variáveis ABAP

Exemplo Correto

```
WHERE campo =  
@minha_variavel
```

Toda variável do programa ABAP usada na query deve ter o prefixo '@'.

Lendo Múltiplas Linhas com `INTO TABLE`

Antes (Sintaxe Obsoleta)

```
DATA: lt_connections TYPE TABLE OF /dmo/connection.
```

```
SELECT * FROM /dmo/connection  
INTO TABLE lt_connections  
WHERE carrier_id = 'LH'.
```

Declaração prévia obrigatória e
ordem de cláusulas rígida.

Agora (Sintaxe Moderna)

```
DATA(lv_airline) = 'LH'.
```

```
SELECT FROM /dmo/connection  
FIELDS
```

carrier_id,

airport_from AS departure_airport,

airport_to AS arrival_airport

```
WHERE carrier_id = @lv_airline
```

```
ORDER BY connection_id DESCENDING
```

```
INTO TABLE @DATA(lt_flights).
```

Lista de campos
explícita

Aliases para
clareza

Devolução Inline: a
tabela é criada e tipada
automaticamente!

A Leitura Cirúrgica: Quando Usar `SINGLE` vs. `UP TO 1 ROWS`

`SELECT SINGLE`

Para Leituras por Chave Primária

Ideal quando você tem a chave primária completa e quer validar a existência de um registro ou ler seus atributos específicos.

```
SELECT SINGLE FROM /dmo/connection  
FIELDS airport_from, airport_to  
WHERE carrier_id      = 'AA'  
      AND connection_id = '0017'  
INTO @DATA(ls_flight_info).
```

`SELECT ... UP TO 1 ROWS`

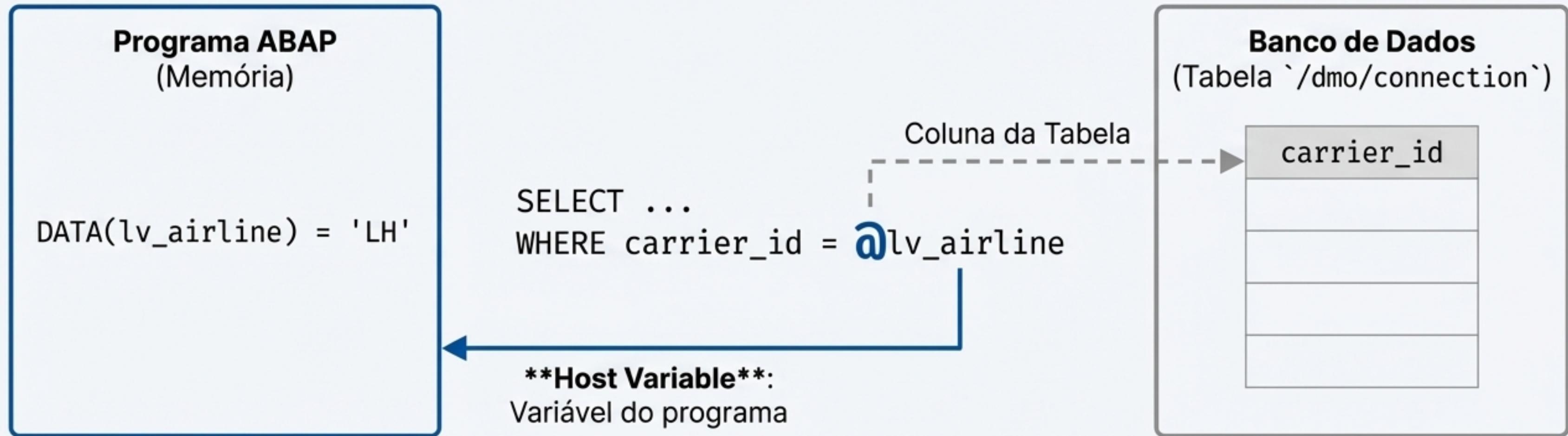
Para Amostragem ou 'Top 1'

Ideal quando você não tem a chave completa e quer 'o primeiro que encontrar' ou um registro específico baseado em uma ordenação (o mais recente, o mais barato, etc.).

```
SELECT FROM /dmo/connection  
FIELDS connection_id  
WHERE airport_from = 'JFK'  
ORDER BY distance ASCENDING " Pega o voo mais curto  
UP TO 1 ROWS  
INTO TABLE @DATA(lt_one_flight).
```

O Detalhe Crucial: Por que Usar o Símbolo `@`?

Separando o Código ABAP do Código do Banco com 'Host Variables'



O `@` diz ao compilador: "Isto é uma variável do meu programa, não uma coluna da tabela". Isso previne ambiguidades, melhora a segurança e otimiza a performance, pois o banco de dados sabe que está recebendo um parâmetro externo.

O Painel de Controle: Interpretando `sy-subrc` e `sy-dbcnt`

`sy-subrc` (Código de Retorno)



0

Sucesso! Pelo menos uma linha foi encontrada.



4

Aviso. Nenhuma linha encontrada com os filtros especificados.

Regra de Ouro: Sempre verifique o valor de `sy-subrc` imediatamente após a instrução `SELECT`.

`sy-dbcnt` (Contador de Registros)

Informa o **número exato** de linhas processadas/retornadas pela última operação.

$sy-subrc = 0$ \rightarrow $sy-dbcnt > 0$
(ex: 500 linhas)

$sy-subrc = 4$ \rightarrow $sy-dbcnt = 0$

Código em Ação (Parte 1): Preparação e Execução da Query

Vamos analisar um exemplo completo: buscar voos de uma companhia com distância acima de um limite, ordenados do maior para o menor.

```
CLASS zcl_read_data IMPLEMENTATION.  
METHOD if_oo_adt_classrun~main.  
  
    " 1. Preparação dos Filtros  
    DATA(lv_carrier) = 'UA'.  
    DATA(lv_min_distance) = 1000.  
  
    " 2. SELECT com Filtros Complexos e Ordenação  
    SELECT FROM /dmo/connection  
        FIELDS  
            carrier_id,  
            connection_id,  
            distance,  
            distance_unit  
        WHERE carrier_id = @lv_carrier _____ → Uso de Host Variable  
              AND distance > @lv_min_distance _____ → Condição lógica complexa  
        ORDER BY distance DESCENDING _____ → Ordenando do maior para o menor  
        INTO TABLE @DATA(lt_results).  
  
    " ... continua no próximo slide  
  
ENDMETHOD.  
ENDCLASS.
```

Código em Ação (Parte 2): Análise dos Resultados

Após a execução do `SELECT`, o próximo passo é sempre analisar o resultado.

```
" 3. Análise do Resultado
IF sy-subrc = 0. ← Verificando se a operação teve sucesso

    " Sucesso: Mostra a contagem de registros
    out->write( |Sucesso! Foram encontrados { sy-dbcnt } voos.| ). ← Usando sy-dbcnt para feedback

    LOOP AT lt_results INTO DATA(ls_flight).
        " Processa os dados retornados
        out->write( |Voo { ls_flight-connection_id }: { ls_flight-distance }| ).
    ENDLOOP.

ELSE.
    " Falha: Trata o caso de não encontrar dados
    out->write( |Nenhum voo encontrado com os critérios.| ).
ENDIF.

" 4. Bônus: Verificando a existência com @abap_true
SELECT SINGLE @abap_true ← A forma mais
    FROM /dmo/connection
    WHERE carrier_id = 'AA' AND connection_id = '0017'
    INTO @DATA(lv_exists).

IF lv_exists = abap_true.
    out->write( 'O voo AA 0017 existe.' ).
ENDIF.

ENDMETHOD.
ENDCLASS.
```

Guia do Expert: Escrevendo Código de Alta Performance

Prática	Evite (Legado / Ineficiente)	Prefira (Moderno / Performático)	Motivo
Seleção de Colunas	<code>SELECT *</code>	<code>SELECT field1, field2, ...</code>	Reducz tráfego de rede e consumo de memória. Você só transporta o que precisa.
Verificar Existência	<code>SELECT * ... UP TO 1 ROWS</code>	<code>SELECT SINGLE @abap_true ...</code>	Extremamente leve. Não transporta nenhum dado da tabela, apenas confirma a existência.
Leituras em Loop	<code>SELECT</code> dentro de um <code>LOOP</code>	<code>FOR ALL ENTRIES</code> ou <code>JOIN</code>	Reducz drasticamente o número de chamadas ao banco de dados (evita o problema "N+1").
Destino dos Dados	<code>INTO CORRESPONDING FIELDS</code>	Lista explícita de campos com <code>INTO TABLE</code>	É mais rápido, pois não precisa comparar nomes de campos, e menos propenso a erros.

Desvendando o Jargão Técnico



Code Pushdown

O paradigma de mover a lógica de processamento de dados (filtros, cálculos) para a camada de banco de dados, em vez de processar na camada de aplicação.



Host Variable (@)

Uma variável declarada no programa ABAP e utilizada dentro de uma instrução SQL. O prefixo `@` é obrigatório no modo estrito.



Inline Declaration (@DATA)

Recurso que permite criar a variável de destino (tabela ou estrutura) no momento da leitura, com o tipo inferido automaticamente a partir da projeção da consulta.



sy-subrc / sy-dbcnt

As variáveis de sistema que informam, respectivamente, o status da operação (0 = sucesso) e a quantidade de linhas afetadas.

Teste seu Conhecimento

Por que esquecer o `@` antes de uma variável ABAP em um SELECT moderno causa um erro de sintaxe?

Porque o compilador, no 'Strict Mode', exige o `@` para diferenciar inequivocamente uma variável do programa (Host Variable) de uma coluna da tabela no banco de dados.

Qual a principal razão para sempre evitar `SELECT *` em código de produção?

Porque ele transporta todas as colunas da tabela, consumindo memória e tráfego de rede desnecessariamente. A prática correta é selecionar explicitamente apenas os campos necessários.

Quando você deve escolher `SELECT SINGLE` em vez de `SELECT ... UP TO 1 ROWS`?

`SELECT SINGLE` é ideal quando se tem a chave primária completa para buscar um registro único. `UP TO 1 ROWS` é usado para pegar uma amostra ou o "primeiro" registro de um conjunto, geralmente combinado com `ORDER BY`.

Sua Evolução como Desenvolvedor Começa Agora

O ABAP SQL Moderno é sua nova ferramenta para criar aplicações mais rápidas, limpas e robustas.



Pense em "Code Pushdown": Antes de escrever, pergunte-se: "Posso delegar este processamento ao banco de dados?"



Abrace a Sintaxe Estrita: Use sempre `@` para variáveis, vírgulas entre campos e `INTO` no final. Clareza é performance.



Seja Explícito e Verifique Sempre: Selecione apenas os campos que precisa e sempre verifique o `sy-subrc` após cada operação.

Pratique! O próximo passo é refatorar um programa antigo seu utilizando a nova sintaxe. A maestria vem com a prática.