03

Faça o que eu fiz na aula

Pensando sempre em como tornar o nosso código mais limpo, optamos então por substituir a utilização de uma série de variáveis que representavam as colunas da tabela SFLIGHT (cia_aerea, numero_voo, data e preco) por uma estrutura, deixando o nosso código da seguinte maneira:

```
* Declaração
TYPES: BEGIN OF estrutura_relatorio,
         cia_aerea(2) TYPE c,
         numero_voo   TYPE n,
         data         TYPE d,
         preco        TYPE p DECIMALS 2.
TYPES:END OF estrutura_relatorio.
```

Substituição no select

```
SELECT carrid connid fldate price
FROM sflight
INTO campos_relatorio.

  PERFORM escreve_nova_linha_no_corpo USING campos_relatorio-cia_aerea campos_relatorio-numero_voo c

ENDSELECT.
```

Com o ajuste realizado fizemos análise de performance do nosso código utilizando a transação SAT, e percebemos que o acesso ao banco de dados está sendo impactado pois ao consultarmos os dados estamos mantendo a conexão aberta por muito tempo devido o nosso select ser um loop e estarmos manipulando dados enquanto realizamos a seleção.

Então para solucionar esse problema vamos passar a primeiro selecionar os dados, e apenas após a seleção ser concluída vamos realizar a manipulação. Para isso iremos utilizar Tabela Interna para guardar as informações do banco. A declaração de uma tabela interna é da seguinte maneira:

```
DATA tabela_voo TYPE STANDARD TABLE OF estrutura_relatorio .
```

E então temos o nosso SELECT da seguinte maneira:

```
SELECT carrid connid fldate price
  FROM sflight
  INTO TABLE tabela_voo.
```

Desta forma já conseguimos salvar os dados do banco de dados dentro de uma tabela que funciona apenas em tempo de execução da aplicação. Após o select ser realizado com sucesso necessitamos realizar um loop para poder manipular os dados

```
LOOP AT tabela_voo INTO estrutura_voo .

  PERFORM escreve_nova_linha_no_corpo USING estrutura_voo-cia_aerea estrutura_voo-numero_voo estrutu

ENDLOOP .
```

Neste loop estamos percorrendo a nossa tabela interna e passando os valores para uma estrutura baseada na nossa tabela interna, sua declaração ficou da seguinte maneira:

```
DATA: tabela_voo    TYPE STANDARD TABLE OF estrutura_relatorio,
          estrutura_voo LIKE LINE OF tabela_voo.
```

Como resultado temos o nosso código da seguinte maneira:

```
TYPES: BEGIN OF estrutura_relatorio,
         cia_aerea(2) TYPE c,
         numero_voo   TYPE n,
         data         TYPE d,
         preco        TYPE p DECIMALS 2.
TYPES:END OF estrutura_relatorio.

CONSTANTS: posicao_preco_cabecalho TYPE i VALUE 36,

           posicao_n_voo_corpo     TYPE i VALUE 12,
           posicao_data_corpo      TYPE i VALUE 25.

DATA: tabela_voo    TYPE STANDARD TABLE OF estrutura_relatorio,
      estrutura_voo LIKE LINE OF tabela_voo.

WRITE: 'Cia Aerea',
        'Numero do Voo',
        'Data',
        AT posicao_preco_cabecalho 'Preço'.

SKIP .

SELECT carrid connid fldate price
  FROM sflight
  INTO TABLE tabela_voo.

LOOP AT tabela_voo INTO estrutura_voo .

  PERFORM escreve_nova_linha_no_corpo USING estrutura_voo-cia_aerea estrutura_voo-numero_voo estrutu

ENDLOOP .

SKIP.
WRITE 'Relatório gerado em: '.
WRITE sy-datum .

FORM escreve_nova_linha_no_corpo USING cia_aerea numero_cia_aerea data preco.

  WRITE:  / cia_aerea,
```

```
            AT posicao_n_voo_corpo numero_cia_aerea,
            AT posicao_data_corpo data DD/MM/YYYY,
            preco LEFT-JUSTIFIED.

  ENDFORM.
```

Assim podemos executar novamente a transação SAT e analisar que a performance da nossa aplicação melhorou, além de ficarmos com o nosso código ainda mais limpo.