Performance: Comparação de trechos de código

22/12/2011 08:00

Bom, quase todo ABAPer sabe que se você for na transação **SE30** e clicar em “Dicas e Sugestões” (Tips & Tricks em inglês), você vai cair em uma tela com diversas comparações prontas de códigos, onde você pode inclusive fazer as suas próprias comparações customizadas. Ahm, se você não sabia, **agora ficou sabendo** 😀

Mas vamos lá, quem aqui já passou por lugares onde eles bloqueiam o editor para você fazer suas comparações levanta a mãooooo….

E adivinhem só: eu também não gosto muito desse editor meio estranho ae não 😛

Pensando em uma forma um pouco mais didática para fazer comparações em cursos de performance, tive a idéia de fazer um programa bem simples que faça comparações de tempo de execução entre dois códigos custom (Zs).

Criei o programa e me ajudou pra caramba no curso. Nada mais justo do que compartilhar o código com você, caro leitor! 🙂

Bem simples, selecione o teste que você que fazer e execute!

O tempo de cada teste é mostrado em microsegundos.

Além de já ter algumas comparações prontas, você pode também criar seus próprios testes, utilizando o método *command_14* . **Para criar uma nova comparação,** procure por todos os lugares que tenha o “14” no código, e duplique as declarações, chamadas e textos. A única restrição é manter o código do método da seguinte forma:

```
METHOD command_XX.
*-------------------------------------------------------------------*
*  New Test
```

```
*--------------------------------------------------------------------*
CASE 'X'.

*     TEST 1
      WHEN test1.

*     TEST 2
      WHEN test2.

    ENDCASE.

  ENDMETHOD.                                                "command_XX
```

Espero que vocês gostem e que seja útil nos projetos. Pelo menos agora você vai poder **esfregar na cara** daquele seu amigo idiota que o seu SELECT é melhor que o dele 😈

Para baixar o código fonte, [clique aqui]() ou faça o CTRL+C, CTRL+V do código abaixo.

Divirta-se! 😀

```
*----------------------------------------------------------------------*
*                     === ABAP ZOMBIE PRESENTS ===                     *
*----------------------------------------------------------------------*
*                      Selection Screen Examples                       *
*----------------------------------------------------------------------*
* Description -> Compare different code snipets and check wich one is  *
*                faster!                                               *
* Date        -> Dez 13, 2011                                          *
* SAP Version -> 6.0                                                   *
*----------------------------------------------------------------------*
* ABAP Zombie Staff: Mauricio Roberto Cruz                             *
*                    Mauro Cesar Laranjeira                            *
*                    Priscila Silva
*----------------------------------------------------------------------*
* Please, visit us at http://abapzombie.com/ and drop a Comment!   *
*----------------------------------------------------------------------*
REPORT  zombie_performance_examples.

*--------------------------------------------------------------------*
* Report Main Screen
*--------------------------------------------------------------------*
SELECTION-SCREEN BEGIN OF BLOCK bl01 WITH FRAME TITLE text-001.

PARAMETERS: p_both  RADIOBUTTON GROUP rb01,
            p_test1 RADIOBUTTON GROUP rb01,
            p_test2  RADIOBUTTON GROUP rb01.

SELECTION-SCREEN END OF BLOCK bl01.

SELECTION-SCREEN BEGIN OF BLOCK bl02 WITH FRAME TITLE text-002.

PARAMETERS: p_01 RADIOBUTTON GROUP rb02,
            p_02 RADIOBUTTON GROUP rb02,
            p_03 RADIOBUTTON GROUP rb02,
            p_04 RADIOBUTTON GROUP rb02,
            p_05 RADIOBUTTON GROUP rb02,
            p_06 RADIOBUTTON GROUP rb02,
            p_07 RADIOBUTTON GROUP rb02,
            p_08 RADIOBUTTON GROUP rb02,
            p_09 RADIOBUTTON GROUP rb02,
            p_10 RADIOBUTTON GROUP rb02,
            p_11 RADIOBUTTON GROUP rb02,
            p_12 RADIOBUTTON GROUP rb02,
            p_13 RADIOBUTTON GROUP rb02,
            p_14 RADIOBUTTON GROUP rb02.
```

```
SELECTION-SCREEN END OF BLOCK bl02.
*--------------------------------------------------------------------*
* Class - Examples Handler
*--------------------------------------------------------------------*
CLASS lcl_compare DEFINITION.

  PUBLIC SECTION.
    METHODS constructor IMPORTING comptype TYPE char5.
    METHODS compare_command IMPORTING compnum TYPE char2.

  PRIVATE SECTION.
    TYPES: BEGIN OF ty_sbook,
            carrid   TYPE sbook-carrid,
            connid   TYPE sbook-connid,
            fldate   TYPE sbook-fldate,
            bookid   TYPE sbook-bookid,
            passname TYPE sbook-passname,
           END OF ty_sbook.

    TYPES: BEGIN OF ty_sflight,
            carrid TYPE sflight-carrid,
            connid TYPE sflight-connid,
           END OF ty_sflight.

    DATA: v_test1 TYPE char1,
          v_test2  TYPE char1.

    DATA: t_sflight   TYPE TABLE OF ty_sflight,
          t_sbook     TYPE TABLE OF ty_sbook,
          t_sbook_aux TYPE TABLE OF ty_sbook,
          lwa_sflight LIKE LINE OF t_sflight,
          lwa_sbook   LIKE LINE OF t_sbook.

    METHODS command_01 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_02 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_03 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_04 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_05 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_06 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_07 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_08 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_09 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_10 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_11 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_12 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_13 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.
    METHODS command_14 IMPORTING test1 TYPE char1 OPTIONAL
                                 test2 TYPE char1 OPTIONAL.

ENDCLASS.                    "lcl_compare DEFINITION

*----------------------------------------------------------------------*
```

```
*       CLASS lcl_compare IMPLEMENTATION
*----------------------------------------------------------------------*
*
*----------------------------------------------------------------------*
CLASS lcl_compare IMPLEMENTATION.

  METHOD constructor.

    CASE comptype.
      WHEN 'TEST1'.
        me->v_test1 = 'X'.
      WHEN 'TEST2'.
        me->v_test2  = 'X'.
      WHEN 'BOTH'.
        me->v_test1 = 'X'.
        me->v_test2  = 'X'.
    ENDCASE.

  ENDMETHOD.                    "constructor

  METHOD compare_command.

    DATA: l_methname TYPE string.

    DATA: l_before TYPE timestampl,
          l_after  TYPE timestampl,
          l_result TYPE timestampl.

    CONCATENATE 'COMMAND_' compnum INTO l_methname.

    IF me->v_test1 = 'X'.
      SKIP 1.
      GET TIME STAMP FIELD l_before.
      CALL METHOD me->(l_methname)
        EXPORTING
          test1 = 'X'.
      GET TIME STAMP FIELD l_after.
      l_result = l_after - l_before.
      WRITE: 'Test 1 Duration: ' COLOR COL_GROUP,
             50 l_result.
      SKIP 1.
    ENDIF.

    CLEAR: l_result, l_after, l_before.

    IF me->v_test2 = 'X'.
      GET TIME STAMP FIELD l_before.
      CALL METHOD me->(l_methname)
        EXPORTING
          test2 = 'X'.
      GET TIME STAMP FIELD l_after.
      l_result = l_after - l_before.
      WRITE: 'Test 2 Duration: ' COLOR COL_POSITIVE,
             50 l_result.
    ENDIF.

  ENDMETHOD.                    "compare_command

  METHOD command_01.
*--------------------------------------------------------------------*
*  Select..EndSelect vs. Array Fetch
*--------------------------------------------------------------------*
    CASE 'X'.

*     SELECT...ENDSELECT
      WHEN test1.
```

```
        SELECT carrid
               connid
               fldate
               bookid
          FROM sbook
          UP TO 30000 ROWS
          INTO lwa_sbook.
          APPEND lwa_sbook TO t_sbook.
        ENDSELECT.

*     ARRAY FETCH
      WHEN test2.

        SELECT carrid
               connid
               fldate
               bookid
          FROM sbook
          INTO TABLE t_sbook
          UP TO 30000 ROWS.

    ENDCASE.

  ENDMETHOD.                                                "command_01

  METHOD command_02.
*---------------------------------------------------------------------*
*  RANGEs - CAUTION!
*---------------------------------------------------------------------*
    DATA: rg_carrid TYPE RANGE OF sbook-carrid,
          lwa_carrid LIKE LINE OF rg_carrid.

    CASE 'X'.

*     Not Equal! Caution!
      WHEN test1.

        lwa_carrid-low    = 'JP'.
        lwa_carrid-option = 'NE'.
        lwa_carrid-sign   = 'I'.
        APPEND lwa_carrid TO rg_carrid.

*     Equal - OK
      WHEN test2.

        lwa_carrid-low    = 'JP'.
        lwa_carrid-option = 'EQ'.
        lwa_carrid-sign   = 'I'.
        APPEND lwa_carrid TO rg_carrid.

    ENDCASE.

    SELECT carrid
           connid
           fldate
           bookid
      FROM sbook
      INTO TABLE t_sbook
      UP TO 50000 ROWS
       WHERE carrid IN rg_carrid.

  ENDMETHOD.                                                "command_02

  METHOD command_03.
*---------------------------------------------------------------------*
```

```
*  FOR ALL ENTRIES vs. INNER JOIN
*----------------------------------------------------------------------*
    CASE 'X'.

*     FOR ALL ENTRIES
      WHEN test1.

        SELECT carrid
               connid
          FROM sflight
          INTO TABLE t_sflight
          WHERE carrid = 'AA'.

        DELETE ADJACENT DUPLICATES FROM t_sflight
                              COMPARING carrid connid.

        SELECT carrid
               connid
               fldate
               bookid
          FROM sbook
          INTO TABLE t_sbook
          UP TO 10000 ROWS
          FOR ALL ENTRIES IN t_sflight
            WHERE carrid = t_sflight-carrid
              AND connid = t_sflight-connid.

*     INNER JOIN
      WHEN test2.

        SELECT a~carrid
               a~connid
               b~fldate
               b~bookid
         FROM sflight AS a INNER JOIN sbook AS b
         ON   a~carrid = b~carrid AND
              a~connid = b~connid
         INTO TABLE t_sbook
         UP TO 10000 ROWS
           WHERE a~carrid = 'AA'.

    ENDCASE.

  ENDMETHOD.                                                "command_03

  METHOD command_04.
*----------------------------------------------------------------------*
*  READ TABLE WITHOUT BINARY SEARCH
*----------------------------------------------------------------------*
    SELECT carrid
           connid
      FROM sflight
      INTO TABLE t_sflight
      WHERE carrid = 'AA'.

    SELECT carrid
           connid
           fldate
           bookid
      FROM sbook
      INTO TABLE t_sbook
      UP TO 500000 ROWS
        WHERE carrid = 'AA'.

    CASE 'X'.
```

```
*     READ TABLE
      WHEN test1.
        LOOP AT t_sbook INTO lwa_sbook.
          READ TABLE t_sflight INTO lwa_sflight WITH KEY
            carrid = lwa_sbook-carrid
            connid = lwa_sbook-connid.
        ENDLOOP.

*     READ TABLE BINARY SEARCH
      WHEN test2.
*       Do not forget to SORT the table before BINARY SEARCH
        LOOP AT t_sbook INTO lwa_sbook.
          READ TABLE t_sflight INTO lwa_sflight WITH KEY
            carrid = lwa_sbook-carrid
            connid = lwa_sbook-connid
          BINARY SEARCH.
        ENDLOOP.
    ENDCASE.

  ENDMETHOD.                                                  "command_04

  METHOD command_05.
*--------------------------------------------------------------------*
*  Select inside LOOP Statement
*--------------------------------------------------------------------*
    SELECT carrid
           connid
           fldate
           bookid
      FROM sbook
      INTO TABLE t_sbook
      UP TO 200000 ROWS
        WHERE carrid = 'AA'.

    CASE 'X'.

*     SELECT inside LOOPs
      WHEN test1.

        LOOP AT t_sbook INTO lwa_sbook.
          SELECT SINGLE carrid
                        connid
                   FROM sflight
                   INTO lwa_sflight
              WHERE carrid = lwa_sbook-carrid
                AND connid = lwa_sbook-connid.

        ENDLOOP.

*     SELECT without LOOPs
      WHEN test2.

        t_sbook_aux[] = t_sbook[].
        SORT t_sbook_aux BY carrid connid.
        DELETE ADJACENT DUPLICATES FROM t_sbook_aux
                              COMPARING carrid
                                        connid.

        SELECT carrid
               connid
          FROM sflight
        INTO TABLE t_sflight
          FOR ALL ENTRIES IN t_sbook_aux
          WHERE carrid = t_sbook_aux-carrid
            AND connid = t_sbook_aux-connid.
        LOOP AT t_sbook INTO lwa_sbook.
          READ TABLE t_sflight INTO lwa_sflight WITH KEY
            carrid = lwa_sbook-carrid
            connid = lwa_sbook-connid
          BINARY SEARCH.
        ENDLOOP.

    ENDCASE.

  ENDMETHOD.                                                "command_05

  METHOD command_06.
*--------------------------------------------------------------------*
*  Massive Update to DB Tables
*  The same concept can be applied to INSERT, DELETE and UPDATE
*--------------------------------------------------------------------*

    DATA: t_sbook TYPE TABLE OF sbook,
          lwa_sbook LIKE LINE OF t_sbook.

    SELECT *
      FROM sbook
      INTO TABLE t_sbook
      UP TO 100000 ROWS
        WHERE carrid = 'AA'.

*   MODIFY inside LOOPs
    CASE 'X'.
      WHEN test1.
        LOOP AT t_sbook INTO lwa_sbook.
          MODIFY sbook FROM lwa_sbook.
        ENDLOOP.

*     MODIFY From Table
      WHEN test2.
        MODIFY sbook FROM TABLE t_sbook.

    ENDCASE.

  ENDMETHOD.                                                "command_06

  METHOD command_07.
*--------------------------------------------------------------------*
*  SELECT INTO CORRESPODING FIELDS
*--------------------------------------------------------------------*
    CASE 'X'.

*     SELECT INTO CORRESPONDING FIELDS
      WHEN test1.
        SELECT *
          FROM sbook
          INTO CORRESPONDING FIELDS OF TABLE t_sbook
       UP TO 100000 ROWS.

*     SELECT ARRAY FETCH
      WHEN test2.
        SELECT carrid
               connid
               fldate
               bookid
          FROM sbook
          INTO TABLE t_sbook
           UP TO 100000 ROWS.

    ENDCASE.
  ENDMETHOD.                                                "command_07

  METHOD command_08.
*--------------------------------------------------------------------*
*  LOOP WHERE vs. LOOP with BINARY SEARCH
*--------------------------------------------------------------------*
    SELECT carrid
           connid
           fldate
           bookid
      FROM sbook
      INTO TABLE t_sbook
      UP TO 100000 ROWS.

    CASE 'X'.

*     LOOP WHERE
      WHEN test1.
        LOOP AT t_sbook INTO lwa_sbook WHERE carrid = 'AA'.
        ENDLOOP.

*     LOOP BINARY SEARCH
      WHEN test2.
        SORT t_sbook BY carrid.
        READ TABLE t_sbook INTO lwa_sbook WITH KEY
          carrid = 'AA'
        BINARY SEARCH.

        LOOP AT t_sbook INTO lwa_sbook FROM sy-tabix.
          IF lwa_sbook-carrid <> 'AA'.
            EXIT.
          ENDIF.
        ENDLOOP.

    ENDCASE.

  ENDMETHOD.                                                "command_08

  METHOD command_09.
*--------------------------------------------------------------------*
*  MOVE CORRESPONDING
*--------------------------------------------------------------------*

    DATA: t_sbook   TYPE TABLE OF sbook,
          lwa_sbook TYPE sbook.

    SELECT *
      FROM sbook
      INTO TABLE t_sbook
      UP TO 200000 ROWS.

*   MOVE CORRESPODING
    CASE 'X'.
      WHEN test1.

        LOOP AT t_sbook INTO lwa_sbook.
          MOVE-CORRESPONDING lwa_sbook TO lwa_sflight.
        ENDLOOP.

*     MOVE SPECIFYING FIELDS
      WHEN test2.

        LOOP AT t_sbook INTO lwa_sbook.
          lwa_sflight-carrid = lwa_sbook-carrid.
          lwa_sflight-connid = lwa_sbook-connid.
        ENDLOOP.
    ENDCASE.

  ENDMETHOD.                                                  "command_09

  METHOD command_10.
*----------------------------------------------------------------------*
*  INDEX EXAMPLES
*----------------------------------------------------------------------*

    DATA: rg_buspart TYPE RANGE OF s_buspanum,
          lwa_buspart LIKE LINE OF rg_buspart.

    lwa_buspart-sign   = 'I'.
    lwa_buspart-option = 'EQ'.
    lwa_buspart-low    = '00003640'.
    APPEND lwa_buspart TO rg_buspart.

    CASE 'X'.

*     INDEX USAGE EXAMPLE
      WHEN test1.

        SELECT carrid
               connid
               fldate
               bookid
          FROM sbook
          INTO TABLE t_sbook
           UP TO 200000 ROWS
            WHERE agencynum IN rg_buspart.

*     ANOTHER INDEX USAGE EXAMPLE
      WHEN test2.

        SELECT carrid
               connid
               fldate
               bookid
          FROM sbook
          INTO TABLE t_sbook
           UP TO 200000 ROWS
            WHERE customid IN rg_buspart.

    ENDCASE.

  ENDMETHOD.                                                  "command_10

  METHOD command_11.

*----------------------------------------------------------------------*
*  INNER JOIN PARTIAL KEY vs INNER JOIN FULL PRIMARY KEY
*----------------------------------------------------------------------*

    CASE 'X'.

*     INNER JOIN PARTIAL KEY
      WHEN test1.

        SELECT a~carrid
               a~connid
               b~fldate
               b~bookid
         FROM sflight AS a INNER JOIN sbook AS b
         ON  a~carrid = b~carrid" AND
*             a~connid = b~connid
         INTO TABLE t_sbook
         UP TO 100000 ROWS
           WHERE a~carrid = 'AA'.

*     INNER JOIN FULL PRIMARY KEY
      WHEN test2.

        SELECT a~carrid
               a~connid
               b~fldate
               b~bookid
         FROM sflight AS a INNER JOIN sbook AS b
         ON   a~carrid = b~carrid AND
              a~connid = b~connid
         INTO TABLE t_sbook
         UP TO 100000 ROWS
           WHERE a~carrid = 'AA'.

    ENDCASE.

  ENDMETHOD.                                                 "command_11

  METHOD command_12.

*---------------------------------------------------------------------*
*  LOOP ASSIGNING
*---------------------------------------------------------------------*

    DATA: lwa_sbook LIKE LINE OF t_sbook.

    FIELD-SYMBOLS:  LIKE LINE OF t_sbook.

    SELECT a~carrid
           a~connid
           b~fldate
           b~bookid
     FROM sflight AS a INNER JOIN sbook AS b
     ON   a~carrid = b~carrid AND
          a~connid = b~connid
     INTO TABLE t_sbook
     UP TO 200000 ROWS
       WHERE a~carrid = 'AA'.

    CASE 'X'.

*     TEST 1
      WHEN test1.

        LOOP AT t_sbook INTO lwa_sbook.
          lwa_sbook-carrid = 'BB'.
          MODIFY t_sbook FROM lwa_sbook INDEX sy-tabix.
        ENDLOOP.

*     TEST 2
      WHEN test2.

        LOOP AT t_sbook ASSIGNING .
          -carrid = 'BB'.
        ENDLOOP.

    ENDCASE.

  ENDMETHOD.                                                 "command_12

  METHOD command_13.
*---------------------------------------------------------------------*
*  SubQuery
```

```
*---------------------------------------------------------------------*
CASE 'X'.

*     TEST 1
      WHEN test1.

        SELECT carrid
               connid
          FROM sflight
          INTO TABLE t_sflight
          WHERE carrid = 'AA'.

        DELETE ADJACENT DUPLICATES FROM t_sflight
                              COMPARING carrid connid.

        SELECT carrid
               connid
               fldate
               bookid
          FROM sbook
          INTO TABLE t_sbook
          UP TO 10000 ROWS
          FOR ALL ENTRIES IN t_sflight
            WHERE carrid = t_sflight-carrid
              AND connid = t_sflight-connid.

*     TEST 2
      WHEN test2.

        SELECT carrid
               connid
               fldate
               bookid
         FROM sbook
          INTO TABLE t_sbook
          UP TO 10000 ROWS
          WHERE EXISTS ( SELECT *
                          FROM sflight
                          WHERE carrid = 'AA'
                            AND connid = sbook~connid ).
    ENDCASE.

  ENDMETHOD.                                                 "command_13

 METHOD command_14.
*---------------------------------------------------------------------*
*  New Test
*---------------------------------------------------------------------*
CASE 'X'.

*     TEST 1
      WHEN test1.

*     TEST 2
      WHEN test2.

    ENDCASE.

  ENDMETHOD.                                                 "command_14

ENDCLASS.                    "lcl_compare IMPLEMENTATION

DATA: o_comp TYPE REF TO lcl_compare.
DATA: v_comm TYPE char2.

*---------------------------------------------------------------------*
```

```
* Event INITIALIZATION
*---------------------------------------------------------------------*
INITIALIZATION.

  PERFORM f_create_texts.

*&---------------------------------------------------------------------*
*&      Form  F_CREATE_TEXTS
*&---------------------------------------------------------------------*
FORM f_create_texts .
  %_p_both_%_app_%-text  = 'Run Both Examples'.
  %_p_test1_%_app_%-text = 'Run Test 1'.
  %_p_test2_%_app_%-text = 'Run Test 2'.
  %_p_01_%_app_%-text    = '01: SELECT... ENDSELECT'.
  %_p_02_%_app_%-text    = '02: RANGEs'.
  %_p_03_%_app_%-text    = '03: F.A.E. vs INNER JOIN'.
  %_p_04_%_app_%-text    = '04: READ TABLE Binary Search'.
  %_p_05_%_app_%-text    = '05: SELECT inside LOOPs'.
  %_p_06_%_app_%-text    = '06: Massive Update to DB  '.
  %_p_07_%_app_%-text    = '07: SELECT Into Corresponding F.'.
  %_p_08_%_app_%-text    = '08: LOOP WHERE vs BINARY LOOP'.
  %_p_09_%_app_%-text    = '09: MOVE-CORRESPONDING'.
  %_p_10_%_app_%-text    = '10: Usage of Indexes'.
  %_p_11_%_app_%-text    = '11: Inner Join Full vs Partial'.
  %_p_12_%_app_%-text    = '12: Loop Assigning'.
  %_p_13_%_app_%-text    = '13: Subquery'.
  %_p_14_%_app_%-text    = '14: Place your test HERE!'.
ENDFORM.                    " F_CREATE_TEXTS

*---------------------------------------------------------------------*
* Event Start-Of-Selection
*---------------------------------------------------------------------*
START-OF-SELECTION.

  CASE 'X'.
    WHEN p_test1.
      CREATE OBJECT o_comp
        EXPORTING
          comptype = 'TEST1'.
    WHEN p_test2.
      CREATE OBJECT o_comp
        EXPORTING
          comptype = 'TEST2'.
    WHEN p_both.
      CREATE OBJECT o_comp
        EXPORTING
          comptype = 'BOTH'.
  ENDCASE.

  CASE 'X'.
    WHEN p_01.
      v_comm = '01'.
    WHEN p_02.
      v_comm = '02'.
    WHEN p_03.
      v_comm = '03'.
    WHEN p_04.
      v_comm = '04'.
    WHEN p_05.
      v_comm = '05'.
    WHEN p_06.
      v_comm = '06'.
    WHEN p_07.
      v_comm = '07'.
    WHEN p_08.
      v_comm = '08'.
    WHEN p_09.
      v_comm = '09'.
    WHEN p_10.
      v_comm = '10'.
    WHEN p_11.
      v_comm = '11'.
    WHEN p_12.
      v_comm = '12'.
    WHEN p_13.
      v_comm = '13'.
    WHEN p_14.
      v_comm = '14'.
  ENDCASE.

  o_comp->compare_command( v_comm ).
*----------------------------------------------------------------------*
*                           === DISCLAIMER ===                          *
*----------------------------------------------------------------------*
* This code is made only for study and reference purposes. It was not  *
* copied from any running program and it does not make references     *
* to any functional requirement. All code here was created based on   *
* the authors experience and creativity! Enjoy!                        *
*----------------------------------------------------------------------*
```

Comentários

Vinícius Andrade — *09/03/2016 20:02*

Excepcional. Me ajudou muito.

Rafael Paes — *19/09/2013 11:03*

Ae Maurício e Priscila,

Ajudaram muito!
Abraços

Cláudio Rico — *18/07/2012 16:14*

Estava procurando um código para medir performance de dois trexos de código, semelhante ao da SE30…Eis que me depara com um código do Mauricio….kkkkk
Valeu mano …

Cláudio Rico

> **Mauricio Cruz** — *18/07/2012 16:16*
>
> Se eu pudesse, mandaria um e-mail pra você com “DE NADA CLAUDIÃO” escrito em fonte 89. 😀
>
> Abraços!

Flávio Furlan — *14/01/2012 12:50*

> Excelente! Parabéns pelo post, com certeza será muito útil para todos os programadores. Usaremos essas informações no futuro para fazer a análise das classes do nosso Campeonado de ABAP.
>
> Abs!
> Furlan

>> **Mauricio Cruz** — *23/01/2012 07:24*
>>
>> Valeu Furlan!
>>
>> Quem sabe no futuro eu não uma melhorada nesse código?
>>
>> Abraços 🙂