



# Desenvolvimento de relatórios

# Índice

## 01 Leitura de Bases de Dados

3

## 02 Ecrãs de selecção

33

## 03 SAP List Viewer – ALV's

61

## 04 Unicode

138

**01**

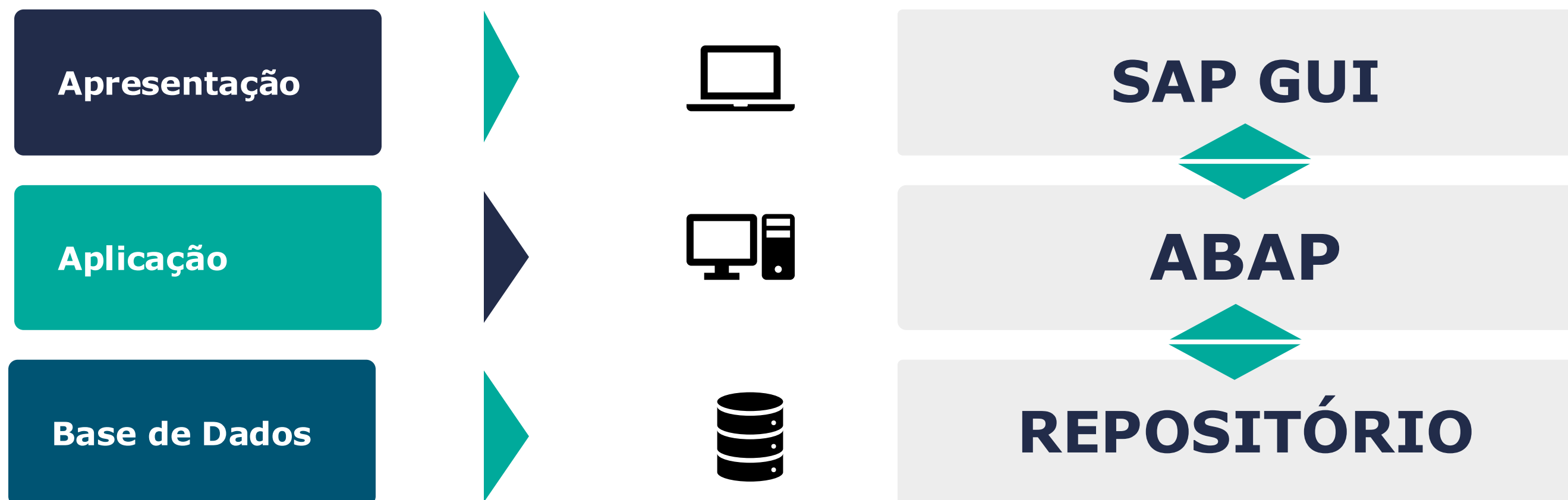
# Leitura de base de dados

# Leitura de base de dados - objetivos

No final deste módulo os formandos deverão ser capazes de:

- Extrair dados das tabelas do SAP, bem como atualizá-las, eliminar ou acrescentar novos registos.
- Identificar as bases de dados lógicas bem como saber efetuar programas que façam uso desta funcionalidade

# Leitura de base de dados



# Leitura de base de dados

## Open SQL

- Consiste num conjunto de instruções Abap que efectuam um determinado número de operações nas bases de dados do SAP
- Os programas Abap que utilizem Open SQL funcionarão em qualquer base de dados sobre a qual o SAP reside, independentemente de ser SAP DB, Oracle, ou outra.
- As instruções de Open SQL apenas podem ser utilizadas sobre bases de dados criadas no Abap Dictionary ( DDIC ).
- Tal como no SQL standard, podem-se utilizar JOINS, bem como efectuar a leitura directamente sobre vistas ( Views ) entre diferentes tabelas.

# Leitura de base de dados

## Open SQL

Palavra Chave	Funcionalidade
SELECT	Lê os dados das tabelas das bases de dados
INSERT	Insere linhas (registos) nas tabelas
UPDATE	Atualiza o conteúdo das tabelas
MODIFY	Atualiza ou insere linhas nas tabelas
DELETE	Apaga linhas das tabelas
OPEN CURSOR, FETCH, CLOSE CURSOR	Lê as tabelas utilizando cursores

# Leitura de base de dados

## Open SQL

### Códigos de Retorno

#### **SY-SUBRC**

Zero, se o acesso teve sucesso, outro valor caso não tenha conseguido ler, actualizar ou eliminar linhas da base de dados, de acordo com a instrução.

#### **SY-DBCNT**

Devolve o número de linhas processadas na base de dados pela instrução SQL que foi executada.



# Leitura de base de dados

## Open SQL

### Dependência de Mandante

Os acessos às tabelas do SAP são CLIENT-DEPENDENT, o que significa que a informação obtida para a mesma tabela num mandante será diferente da obtida pelo mesmo acesso noutro mandante.

Caso seja necessário efectuar uma leitura a um mandante específico, tem de se recorrer à adição de CLIENT SPECIFIED indicando o nº de mandante a ser lido.

# Leitura de base de dados

## Open SQL

### Sintaxe de Leitura

#### SELECT MÚLTIPLO:

```
SELECT result
      FROM source
      INTO|APPENDING target
      [[FOR ALL ENTRIES IN itab] WHERE sql_cond]
      [GROUP BY group] [HAVING group_cond]
      [ORDER BY sort_key].

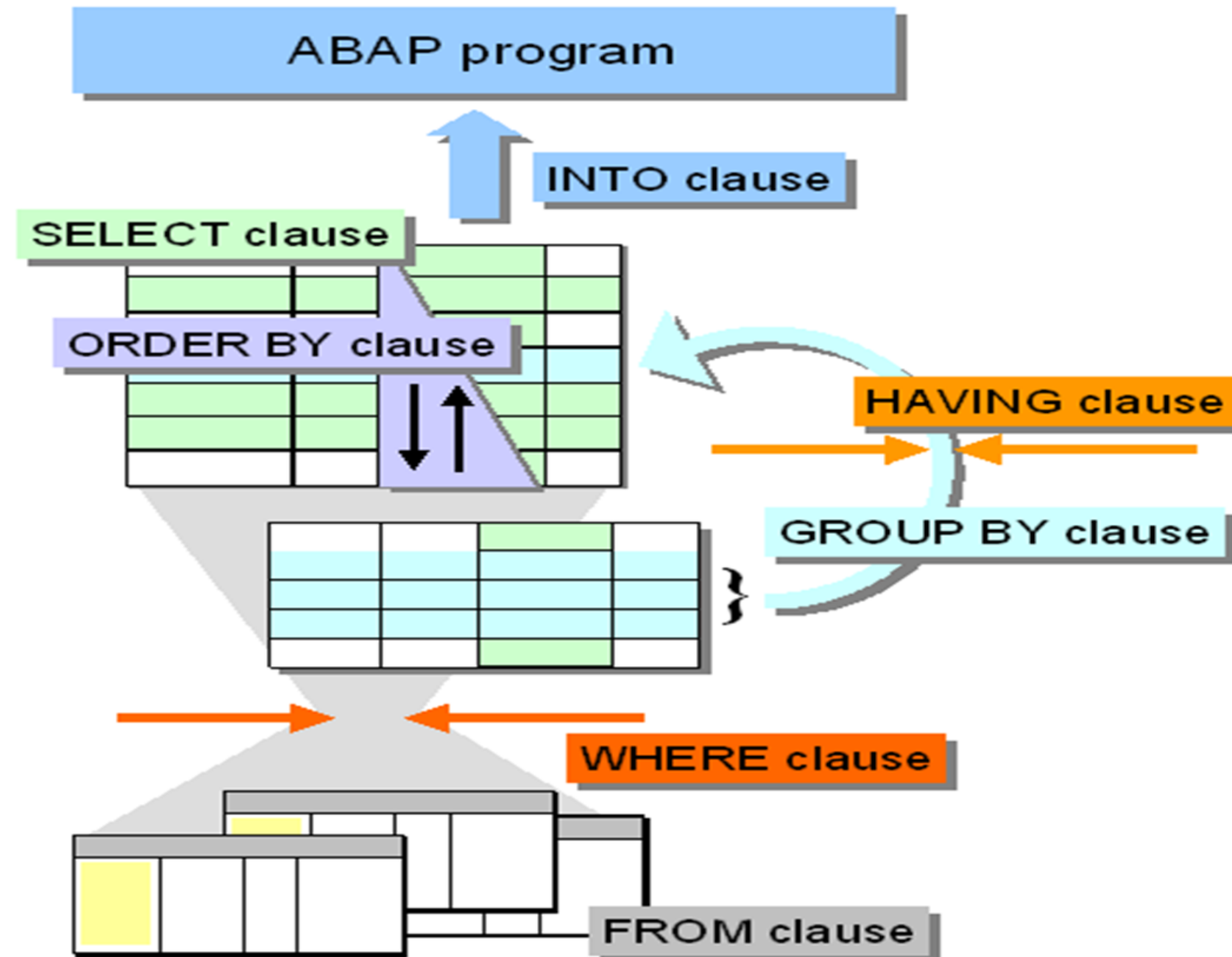
...
[ENDSELECT].
```

#### SELECT SINGLE:

```
... { SINGLE [FOR UPDATE] }
    | { [DISTINCT] { } } ... .
```

# Leitura de base de dados

## Open SQL



# Leitura de base de dados

## Open SQL

### Instrução SELECT

Cláusula	Descrição
SELECT <result>	Define a estrutura de dados a ler
INTO <target>	Determina a área para a qual os dados serão lidos
FROM <source>	Indica a tabela ou visão onde os dados são lidos
WHERE <condição>	Especifica as linhas lidas de acordo com condições
GROUP BY <fields>	Agrupa a leitura por linhas c/valores iguais p/ campo
HAVING <cond>	Lê as tabelas utilizando cursores
ORDER BY <cond>	Define a ordem pela qual as linhas são obtidas

# Leitura de base de dados

## Open SQL – inserir linhas em tabelas

### INSERT dbtab

Permite inserir um ou mais registos (linhas) na tabela <Target>

**INSERT { {INTO target VALUES source }  
| { target FROM source } }.**

Permite inserir um registo <wa> na tabela <Target>

**INSERT INTO <target> VALUES <wa> .**

Permite inserir vários registos/linhas de uma tabela interna ITAB na tabela <Target>

**INSERT <target> FROM TABLE <itab> [ACCEPTING DUPLICATE KEYS] .**

# Leitura de base de dados

## Open SQL – actualizar linhas/registos

### UPDATE

Permite modificar um registo (linha) na tabela <Target>

**UPDATE <target> <lines>.**

Permite indicar quais os campos a serem modificados, bem como as respectivas condições através da cláusula WHERE.

**UPDATE <target> SET <set1> <set 2> ... [WHERE <cond>].**

Permite modificar vários registos/linhas da tabela transparente <Target> a partir de uma tabela interna ITAB.

**UPDATE <target> FROM TABLE <itab> .**

# Leitura de base de dados

## Open SQL – eliminar linhas/registos

### DELETE

Elimina uma ou mais linhas da tabela transparente <Target> de acordo

Com as condições indicadas.

**DELETE FROM <target> [WHERE <cond>] .**

Elimina uma linha a partir de uma área de trabalho <wa>

**DELETE <target> FROM <wa> .**

Apaga um conjunto de linhas/registos da tabela <Target> a partir de uma tabela interna itab.

**DELETE <target> FROM TABLE itab <wa> .**

# Leitura de base de dados

## Open SQL – criar/modificar linhas/registos

### MODIFY

Se a tabela não contiver um único registo com a mesma chave primária, é criado um novo registo na tabela <Target>, caso contrário este será modificado.

**MODIFY <target> <lines>.**

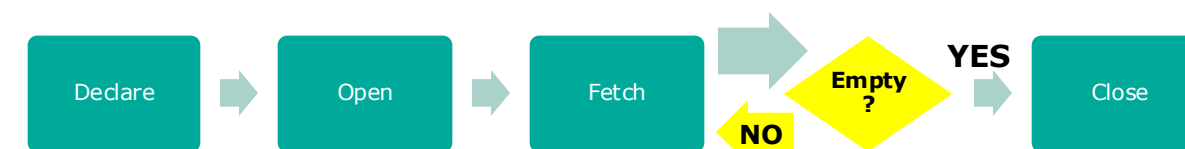
O mesmo que o anterior mas para os registos contidos numa tabela Interna ITAB.

**MODIFY <target> FROM TABLE <itab> .**



# Leitura de base de dados

## Open SQL - cursores



Numa instrução SELECT, os dados selecionados são lidos diretamente para uma “Work Area” específica durante a leitura; quando se utiliza um Cursor para ler dados, dividimos o processo efetuado pela instrução Select.

Para tal, tem de se abrir um cursor para uma instrução Select e só depois se poderão colocar os registos lidos numa área de trabalho.

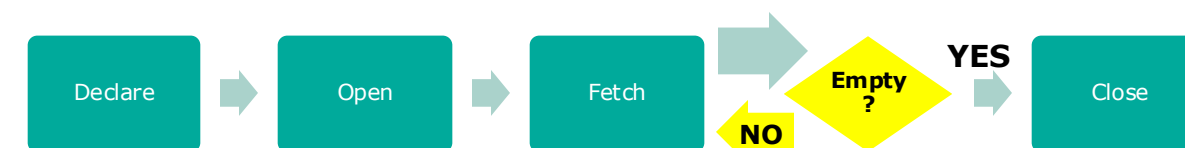
### Sintaxe:

```

OPEN CURSOR [WITH HOLD] <c> FOR SELECT    <result>
        FROM      <source>
        [WHERE    <condition>]
        [GROUP BY <fields>]
        [HAVING   <cond>]
        [ORDER BY <fields>].
  
```

# Leitura de base de dados

## Open SQL - cursores



Para ler os dados com Cursores recorre-se à intrução FETCH :

Sintaxe:

**FETCH NEXT CURSOR <c> INTO <target>.**

Escreve um registo/linha na área de trabalho <target>, movendo o cursor para a linha seguinte do conjunto a seleccionar.

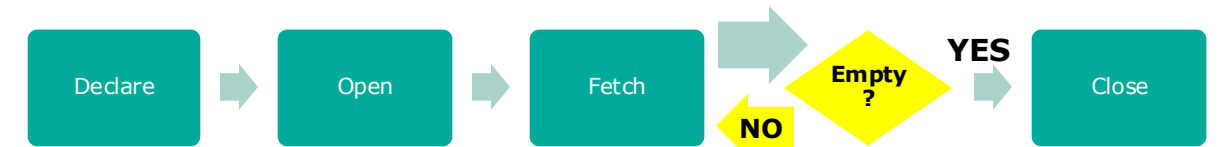
Para terminar a leitura com um Cursor, é necessário fechá-lo, cuja sintaxe é:

**CLOSE CURSOR <c>.**

O sufixo WITH HOLD acrescentado na OPEN CURSOR impede que o cursor seja encerrado quando ocorre um COMMIT WORK no Native SQL.

# Leitura de base de dados

## Open SQL - cursores



```
DATA: C1 TYPE CURSOR,  
      C2 TYPE CURSOR.
```

```
DATA: WA1 TYPE SPFLI,  
      WA2 TYPE SPFLI.
```

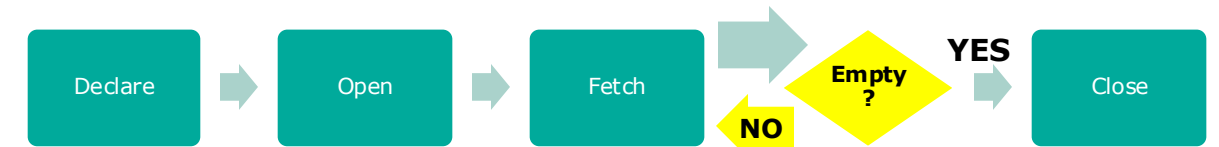
```
DATA: FLAG1,  
      FLAG2.
```

```
OPEN CURSOR: C1 FOR SELECT CARRID CONNID  
              FROM   SPFLI  
              WHERE  CARRID = 'LH',
```

```
              C2 FOR SELECT CARRID CONNID CITYFROM CITYTO  
              FROM   SPFLI  
              WHERE  CARRID = 'AZ'.
```

# Leitura de base de dados

## Open SQL - cursores



```

DO.
  IF FLAG1 NE 'X'.
    FETCH NEXT CURSOR C1 INTO CORRESPONDING FIELDS OF WA1.
    IF SY-SUBRC <> 0.
      CLOSE CURSOR C1.
      FLAG1 = 'X'.
    ELSE.
      WRITE: / WA1-CARRID, WA1-CONNID.
    ENDIF.
  ENDIF.
  IF FLAG2 NE 'X'.
    FETCH NEXT CURSOR C2 INTO CORRESPONDING FIELDS OF WA2.
    IF SY-SUBRC <> 0.
      CLOSE CURSOR C2.
      FLAG2 = 'X'.
    ELSE.
      WRITE: / WA2-CARRID, WA2-CONNID,
              WA2-CITYFROM, WA2-CITYTO.
    ENDIF.
  ENDIF.
  IF FLAG1 = 'X' AND FLAG2 = 'X'.
    EXIT.
  ENDIF.
ENDDO.
  
```

# Leitura de base de dados

## Open SQL - cursores

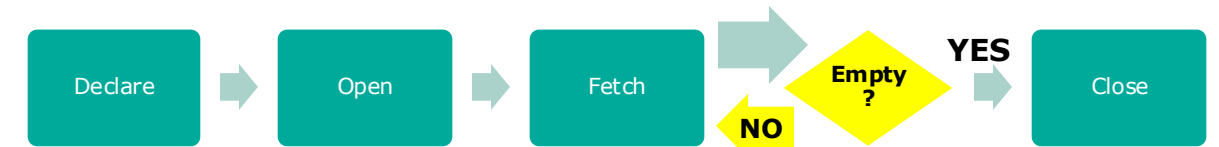
### OUTPUT

LH	0400		
AZ	0555	ROME	FRANKFURT
LH	0402		
AZ	0788	ROME	TOKYO
LH	2402		
AZ	0789	TOKYO	ROME
LH	2407		
AZ	0790	ROME	OSAKA

# Leitura de base de dados

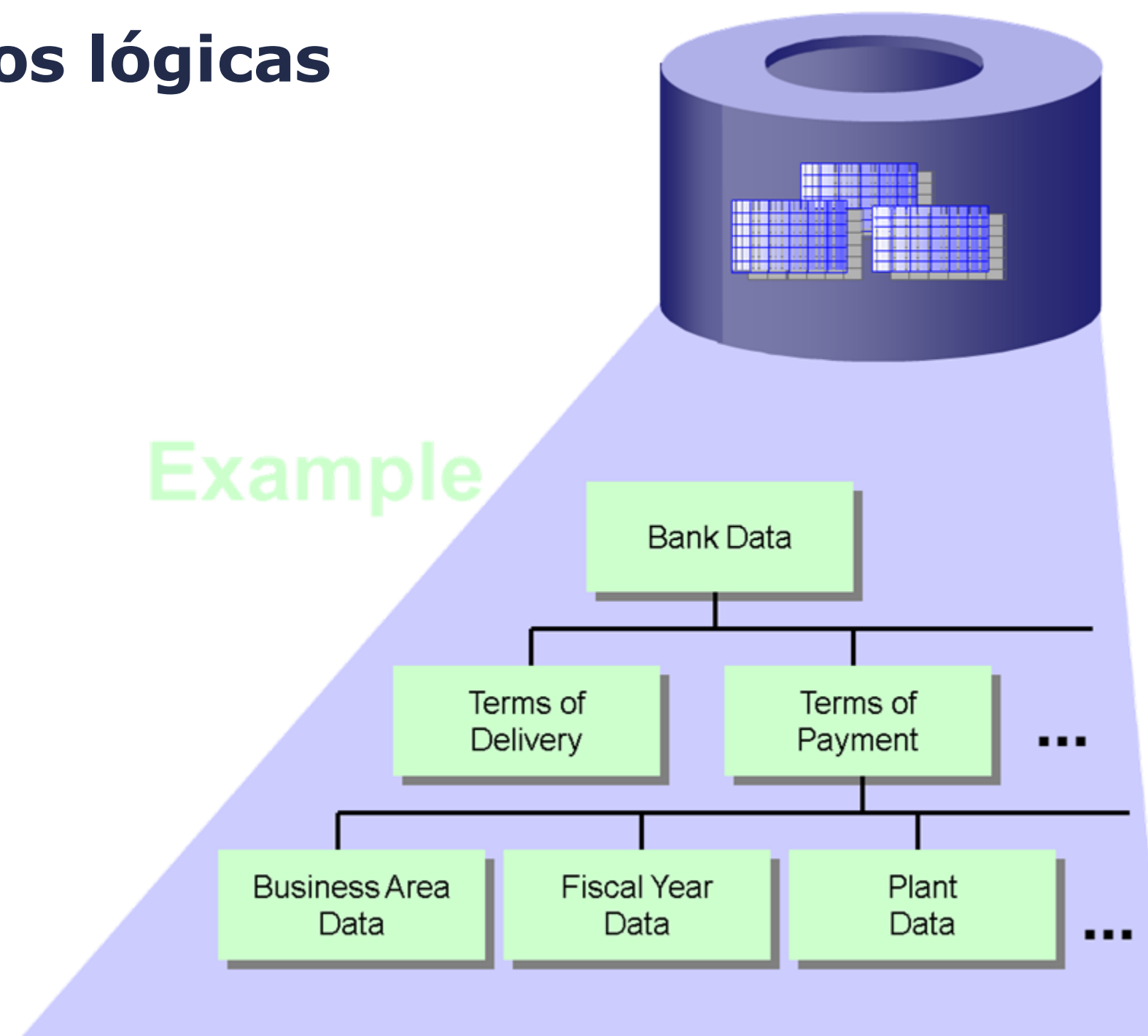
## Open SQL - cursores

- São programas especiais de Abap que obtêm dados de tabelas transparentes de modo a serem utilizados em aplicações.
- Geralmente são utilizadas para ler dados de tabelas e ligá-los a programas Abap



# Leitura de base de dados

## Bases de dados lógicas



# Leitura de base de dados

## Bases de dados lógicas

Exemplo de Banco de Dados Lógico (KDF – Fornecedores):

The screenshot displays the SAP KDF (Logical Data Bank) interface. The title bar reads 'Banco de dados Processar Ir para Suplementos Ambiente Sistema Ajuda'. The main window is titled 'Exibir banco de dados lógico KDF'. Below the title bar, there are tabs for 'Administr.', 'Estrutura', 'Ajuda para pesquisa', and 'Campos moeda/quantidade'. The 'Estrutura' tab is active, showing a tree view of the logical data bank structure. The tree view lists various tables and their descriptions, organized into a hierarchy starting with 'LFA1'.

Nome do nó	Tabela / catego...	Tipo de nó	Texto breve
▼ LFA1	LFA1	Tabela	Mestre de fornecedores (parte geral)
LFA1	ADDR1_VAL	Tabela	Dados de endereço
LFA1	LFAS	Tabela	Mestre forneceds.(parte geral nº ident.fiscal CE)
LFA1	LFBK	Tabela	Mestre de fornecedores (coordenadas do banco)
LFA1	LFB1	Tabela	Mestre de fornecedores (empresa)
LFA1	LFB5	Tabela	Mestre de fornecedores (dados de advertência)
LFA1	LFC1	Tabela	Mestre de fornecedores: movimentação no período
LFA1	LFC3	Tabela	Mestre fornec.: movim.período opers.Razão Espec.
LFA1	BSIK	Tabela	Contabilidade: índice secundário para fornecedores
LFA1	ADMI_FILES	Tabela	Files de arquivo
LFA1	BSIKEXT	Tabela	Índice secundário & extensão (BSEGA)
LFA1	BKPF	Tabela	Cabeçalho do documento contábil
LFA1	BSEG	Tabela	Segmento do documento p/contabilidade financeira
BSEG	WITH_ITEM	Tabela	Info IRF por categoria IRF e item de doc.FI
BSEG	GSEG	Tabela	Itens de contrapartida para BSEG no reporting

The bottom status bar shows 'GCD (2) (333)' and 'sapdeveu INS'.



# Leitura de base de dados

## Bases de dados lógicas

Exemplo com Banco de Dados Lógico de Documentos Contábeis BRM, obtendo cabeçalho e linhas:

**Editor ABAP: 1ª tela** ABAP: características do programa ZEXEMPLO26 modificar

Programa: ZEXEMPLO26

Título: Programa ZEXEMPLO26 - Exemplo para formação Abap

Idioma original: PT Português

Criado: 04.11.2009 NSEARA

Última modific.: 04.11.2009 NSEARA

Status: ativo

**Atributos**

Tipo: 1 Programa executável

Status:

Aplicação:

Grupo autorizações:

Pacote: ZFORMNOV2009 Formação - Novembro 2009

Banco de dados lógico: BRM Documentos contábeis

Versão tela seleção:

☐ Bloqueio do editor ☒ Aritmética em ponto fixo

☒ Verifs.unicode ativas ☐ Início via variante

Gravar

# Leitura de base de dados

## Bases de dados lógicas

Banco de Dados Lógico (KDF – Fornecedores):

Para efetuar a leitura dos registos recorrendo a bases de dados lógicas, utilizam-se as instruções:

GET node [LATE] [FIELDS f1 f2 ...].

1. GET node [FIELDS f1 f2 ...].
2. GET node LATE [FIELDS f1 f2 ...].

# Leitura de base de dados

## Bases de dados lógicas

```
REPORT demo_get.
* Base de dados lógica F1S
NODES: spfli, sflight, sbook.
```

```
DATA: weight          TYPE p LENGTH 8 DECIMALS 4,
      total_weight TYPE p LENGTH 8 DECIMALS 4.
```

```
INITIALIZATION.
  carrid-sign = 'I'.
  carrid-option = 'EQ'.
  carrid-low = 'AA'.
  carrid-high = 'LH'.
  APPEND carrid TO carrid.
```

```
START-OF-SELECTION.
  WRITE 'Luggage weight of flights'.
```

```
GET spfli FIELDS carrid connid cityfrom cityto.
```

```
SKIP.
ULINE.
```

```
WRITE: / 'Carrid:', spfli-carrid,
        'Connid:', spfli-connid,
        / 'From: ', spfli-cityfrom,
        'To: ', spfli-cityto.
```

```
ULINE.
```

```
GET sflight FIELDS fldate.
SKIP.
WRITE: / 'Date:', sflight-fldate.
```

```
GET sbook FIELDS luggweight.
  weight = weight + sbook-luggweight.
```

```
GET sflight LATE FIELDS carrid .
  WRITE: / 'Luggage weight =', weight.
  total_weight = total_weight + weight.
  weight = 0.
```

```
END-OF-SELECTION.
```

```
ULINE.
WRITE: / 'Sum of luggage weights =', total_weight.
```

# Leitura de base de dados

## Bases de dados lógicas

Exemplo com Banco de Dados Lógico de Documentos Contábeis BRM, obtendo cabeçalho e linhas:

```
*&-----*
*& Report ZEXEMPLO26
*&
*&-----*
*&
*&
*&-----*
```

REPORT zexemplo26.

NODES: bkp, bseg.

START-OF-SELECTION.

GET bkp.

GET bseg.

WRITE:/ bseg-belnr, bseg-buzei, bseg-matnr, bseg-dmbtr.

# Leitura de base de dados

## Bases de dados lógicas

Exemplo com Banco de Dados Lógico de Documentos Contábeis BRM, obtendo cabeçalho e linhas:

```
REPORT z_luggage_info.
```

```
CLASS flight_luggage DEFINITION.
```

```
  PUBLIC SECTION.
```

```
    METHODS: get_luggage_info.
```

```
ENDCLASS.
```

```
CLASS flight_luggage IMPLEMENTATION.
```

```
  METHOD get_luggage_info.
```

```
    DATA: lt_luggages TYPE TABLE OF sbook,
```

```
          lv_flight_date TYPE sflight-fldate,
```

```
          lv_passenger_name TYPE sbook-passname,
```

```
          lv_luggage_weight TYPE sbook-luggwgt.
```

```
  GET EVENT sflight.
```

```
  IF sflight-fldate IS NOT INITIAL.
```

```
    lv_flight_date = sflight-fldate.
```

```
    SELECT * FROM sbook INTO TABLE lt_luggages
```

```
      WHERE fldate = lv_flight_date.
```

```
    LOOP AT lt_luggages INTO DATA(luggage).
```

```
      lv_passenger_name = luggage-passname.
```

```
      lv_luggage_weight = luggage-luggwgt.
```

```
      WRITE: / 'Flight Date:', lv_flight_date,
```

```
              / 'Passenger Name:',
```

```
              lv_passenger_name,
```

```
              / 'Luggage Weight:',
```

```
              lv_luggage_weight,
```

```
              /.
```

```
    ENDLOOP.
```

```
  ENDIF.
```

```
  ENDMETHOD.
```

```
ENDCLASS.
```

```
START-OF-SELECTION.
```

```
  DATA: lo_flight_luggage TYPE REF TO  
         flight_luggage.
```

```
  CREATE OBJECT lo_flight_luggage.
```

```
  lo_flight_luggage->get_luggage_info().
```

# Leitura de base de dados

## Bases de dados lógicas

Exemplo BD Lógico de Documentos Contábeis BRM, em programação OO:

```
REPORT z_luggage_info.

CLASS flight_luggage DEFINITION.
  PUBLIC SECTION.
    METHODS: constructor,
              get_flight_info IMPORTING flight_date TYPE sflight-fldate,
              display_luggage_info.

  PRIVATE SECTION.
    TYPES: BEGIN OF ty_luggage_info,
            flight_date TYPE sflight-fldate,
            passenger_name TYPE sbook-passname,
            luggage_weight TYPE sbook-LUGGWEIGHT,
          END OF ty_luggage_info.

    DATA: lt_luggage_info TYPE TABLE OF ty_luggage_info.
  ENDCLASS.

CLASS flight_luggage IMPLEMENTATION.

  METHOD constructor.
    CLEAR lt_luggage_info.
  ENDMETHOD.

  METHOD get_flight_info.
    DATA: lv_flight_date TYPE sflight-fldate,
           lt_passengers TYPE TABLE OF sbook.

    lv_flight_date = flight_date.
    SELECT * FROM sbook INTO TABLE lt_passengers WHERE fldate = lv_flight_date.
```

```
    LOOP AT lt_passengers INTO DATA(luggage).
      DATA(luggage_info) = VALUE ty_luggage_info(
        flight_date = lv_flight_date
        passenger_name = luggage-passname
        luggage_weight = luggage-LUGGWEIGHT
      ).

      APPEND luggage_info TO lt_luggage_info.
    ENDLOOP.
  ENDMETHOD.

  METHOD display_luggage_info.
    LOOP AT lt_luggage_info INTO DATA(luggage).
      WRITE: / 'Flight Date:', luggage-flight_date,
              / 'Passenger Name:', luggage-passenger_name,
              / 'Luggage Weight:', luggage-luggage_weight,
              /.
    ENDLOOP.
  ENDMETHOD.

ENDCLASS.

START-OF-SELECTION.
  DATA: lo_flight_luggage TYPE REF TO flight_luggage,
        lv_flight_date TYPE sflight-fldate.

  lv_flight_date = '20180925'. "Replace with your desired flight date
  CREATE OBJECT lo_flight_luggage.
  lo_flight_luggage->get_flight_info( lv_flight_date ).
  lo_flight_luggage->display_luggage_info( ).
```

# Leitura de base de dados

## Bases de dados lógicas

Exemplo BD Lógico de Documentos Contábeis BRM, em programação OO versão 7.4:

```
CLASS flight_luggage DEFINITION.
  PUBLIC SECTION.
    METHODS: constructor,
              get_flight_info IMPORTING flight_date TYPE sflight-fdate,
              display_luggage_info.

  PRIVATE SECTION.
    TYPES: BEGIN OF ty_luggage_info,
            flight_date TYPE sflight-fdate,
            passenger_name TYPE sbook-passname,
            luggage_weight TYPE sbook-luggwgt,
          END OF ty_luggage_info.

    DATA: lt_luggage_info TYPE TABLE OF ty_luggage_info.
  ENDClass.

CLASS flight_luggage IMPLEMENTATION.

  METHOD constructor.
    CLEAR lt_luggage_info.
  ENDMETHOD.

  METHOD get_flight_info.
    DATA(lt_passengers) = VALUE #( FOR wa IN sbook
      WHERE ( fdate = flight_date ) ( wa ) ).
```

```
    LOOP AT lt_passengers INTO DATA(luggage).
      DATA(luggage_info) = VALUE ty_luggage_info(
        flight_date = flight_date,
        passenger_name = luggage-passname,
        luggage_weight = luggage-luggwgt
      ).

      APPEND luggage_info TO lt_luggage_info.
    ENDLOOP.
  ENDMETHOD.

  METHOD display_luggage_info.
    LOOP AT lt_luggage_info INTO DATA(luggage).
      WRITE: / 'Flight Date:', luggage-flight_date,
              / 'Passenger Name:', luggage-passenger_name,
              / 'Luggage Weight:', luggage-luggage_weight,
              /.
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.

START-OF-SELECTION.
  DATA(lo_flight_luggage) = NEW flight_luggage( ).
  DATA(lv_flight_date) = 'YOUR_FLIGHT_DATE_HERE'. "Replace with your desired flight date
  lo_flight_luggage->get_flight_info( lv_flight_date ).
  lo_flight_luggage->display_luggage_info( ).
```

# Leitura de base de dados

## Bases de dados lógicas

Exemplo BD Lógico de Documentos Contábeis BRM, em programação OO usando REDUCE:

```
REPORT z_luggage_info.
```

```
CLASS flight_luggage DEFINITION.
```

```
  PUBLIC SECTION.
```

```
    METHODS: constructor,
```

```
             get_flight_info IMPORTING flight_date TYPE sflight-fldate,
```

```
             display_luggage_info.
```

```
  PRIVATE SECTION.
```

```
    TYPES: BEGIN OF ty_luggage_info,
```

```
            flight_date TYPE sflight-fldate,
```

```
            passenger_name TYPE sbook-passname,
```

```
            luggage_weight TYPE sbook-luggwgt,
```

```
    END OF ty_luggage_info.
```

```
    DATA: lt_luggage_info TYPE TABLE OF ty_luggage_info.
```

```
ENDCLASS.
```

```
CLASS flight_luggage IMPLEMENTATION.
```

```
  METHOD constructor.
```

```
    CLEAR lt_luggage_info.
```

```
  ENDMETHOD.
```

```
  METHOD get_flight_info.
```

```
    DATA(lt_passengers) = REDUCE #( INIT lt_luggages = VALUE ty_luggage_info_table( )
```

```
      FOR wa IN sbook WHERE ( fldate = flight_date )
```

```
      NEXT lt_luggages = VALUE #( BASE lt_luggages (
```

```
        flight_date = flight_date
```

```
        passenger_name = wa-passname
```

```
        luggage_weight = wa-luggwgt
```

```
      ) ).
```

```
    lt_luggage_info = lt_passengers.
```

```
  ENDMETHOD.
```

```
  METHOD display_luggage_info.
```

```
    LOOP AT lt_luggage_info INTO DATA(luggage).
```

```
      WRITE: / 'Flight Date:', luggage-flight_date,
```

```
             / 'Passenger Name:', luggage-passenger_name,
```

```
             / 'Luggage Weight:', luggage-luggage_weight,
```

```
            /.
```

```
    ENDLOOP.
```

```
  ENDMETHOD.
```

```
ENDCLASS.
```

```
START-OF-SELECTION.
```

```
  DATA(lo_flight_luggage) = NEW flight_luggage( ).
```

```
  DATA(lv_flight_date) = 'YOUR_FLIGHT_DATE_HERE'. "Replace with your desired flight date
```

```
  lo_flight_luggage->get_flight_info( lv_flight_date ).
```

```
  lo_flight_luggage->display_luggage_info( ).
```



02

# Ecrãs de selecção

# Ecrãs de selecção - objectivos

No final deste módulo os formandos deverão ser capazes de criar diversos ecrans de selecção de modo a estarem aptos a criarem programas com apresentação de parâmetros ou intervalos de selecção bem como botões e tratamento de eventos.

# Ecrãs de selecção

**Existem 3 formas de definir ecrãs de selecção:**

**PARAMETERS** – para campos isolados

**SELECT-OPTIONS** – para selecções complexas

**SELECTION-SCREEN** – para formatar o ecran de selecção e definir ecrans específicos para o utilizador

# Ecrãs de selecção

## Ecrãs De Selecção / Sintaxes

```
SELECTION-SCREEN BEGIN OF SCREEN dynnr [TITLE title]
                        [AS WINDOW].
```

...

```
SELECTION-SCREEN END OF SCREEN dynnr.
```

Dynnr é o número de ecran a ser criado.

### EXEMPLO:

```
SELECTION-SCREEN BEGIN OF SCREEN 500 TITLE title
                        AS WINDOW.
```

```
PARAMETERS name TYPE sy-uname.
```

```
SELECTION-SCREEN END OF SCREEN 500.
```

title = 'Input name'.

```
CALL SELECTION-SCREEN '0500' STARTING AT 10 10.
```

# Ecrãs de selecção

## Ecrãs de Selecção / Sintaxes

```
SELECTION-SCREEN BEGIN OF SCREEN dynnr AS SUBSCREEN  
[NO INTERVALS]  
[NESTING LEVEL n].
```

...

```
SELECTION-SCREEN END OF SCREEN dynnr.
```

Dynnr é o número de subecran a ser criado.

# Ecrãs de selecção

## Ecrãs de Selecção / Sintaxes

### Exemplo:

REPORT ...

```
SELECTION-SCREEN BEGIN OF SCREEN 100 AS SUBSCREEN.
PARAMETERS: p1 TYPE c LENGTH 10,
              p2 TYPE c LENGTH 10,
              p3 TYPE c LENGTH 10.
SELECTION-SCREEN END OF SCREEN 100.
```

```
SELECTION-SCREEN BEGIN OF SCREEN 200 AS SUBSCREEN.
PARAMETERS: q1 TYPE c LENGTH 10,
              q2 TYPE c LENGTH 10,
              q3 TYPE c LENGTH 10.
SELECTION-SCREEN END OF SCREEN 200.
```

```
SELECTION-SCREEN: BEGIN OF TABBED BLOCK mytab FOR 10 LINES,
                  TAB (20) button1 USER-COMMAND push1
                      DEFAULT SCREEN 100,
                  TAB (20) button2 USER-COMMAND push2
                      DEFAULT SCREEN 200,
                  END OF BLOCK mytab.
```

```
INITIALIZATION.
  button1 = 'Pasta 1'.
  button2 = 'Pasta 2'.
```

# Ecrãs de selecção

## Resultado da Execução (Ecrã de Selecção)

Programa Processar Ir para Sistema Ajuda

**Exemplo 2 de ecrans de selecção**

Pasta 1 Pasta 2

Campo P1	
Campo P2	
Campo P3	

GCD (2) (333) sapdeveu INS

# Ecrãs de selecção

## Elementos de Ecrã / Formas de Sintaxe

### Filas vazias:

SELECTION-SCREEN SKIP [n] [ldb\_additions].

### Sublinhado:

SELECTION-SCREEN ULINE [[/][pos](len)] [MODIF ID modid]  
[ldb\_additions].

### Blocks:

SELECTION-SCREEN BEGIN OF BLOCK block  
[WITH FRAME [TITLE title]]  
[NO INTERVALS].

...

SELECTION-SCREEN END OF BLOCK block.



# Ecrãs de selecção

## Elementos de Ecrã / Formas de Sintaxe

### Botões:

```
SELECTION-SCREEN PUSHBUTTON [/][pos](len) button_text
      USER-COMMAND ucom
      [VISIBLE LENGTH vlen]
      [MODIF ID modid]
      [ldb_additions].
```

### Filas com vários elementos:

```
SELECTION-SCREEN BEGIN OF LINE.
...
[SELECTION-SCREEN POSITION pos [ldb_additions]].
...
SELECTION-SCREEN END OF LINE.
```

# Ecrãs de selecção

## Elementos de Ecrã / Exemplo

REPORT zselection\_screens03.

SELECTION-SCREEN COMMENT /2(50) text-001 MODIF ID sc1.

SELECTION-SCREEN SKIP 2.

SELECTION-SCREEN COMMENT /10(30) comm1.

SELECTION-SCREEN ULINE.

PARAMETERS: r1 RADIOBUTTON GROUP rad1,  
                  r2 RADIOBUTTON GROUP rad1,  
                  r3 RADIOBUTTON GROUP rad1.

SELECTION-SCREEN ULINE /1(50).

SELECTION-SCREEN COMMENT /10(30) comm2.

SELECTION-SCREEN ULINE.

PARAMETERS: s1 RADIOBUTTON GROUP rad2,  
                  s2 RADIOBUTTON GROUP rad2,  
                  s3 RADIOBUTTON GROUP rad2.

SELECTION-SCREEN ULINE /1(50).

INITIALIZATION.

comm1 ='Radio Button Grupo 1'.

comm2 ='Radio Button Grupo 2'.

LOOP AT SCREEN.

IF screen-group1 = 'SC1'.

screen-intensified = '1'.

MODIFY SCREEN.

ENDIF.

ENDLOOP.

# Ecrãs de selecção

## Output

Programa
Processar
Ir para
Sistema
Ajuda

☒

Linhas em branco, sublinhado e comentários

☒

|

Radio Button Grupo 1

☒ R1
☐ R2
☐ R3

Radio Button Grupo 2

☒ S1
☐ S2
☐ S3

# Ecrãs de selecção

## Estrutura SSCRFIELDS - Campos em imagens de selecção

Estrutura
Processar
Ir para
Utilitários
Suplementos
Ambiente
Sistema
Ajuda

## Dictionary: exibir estrutura

Representação de hierarquia
Estrutura append...

Estrut.
SSCRFIELDS
ativo

Descrição breve
Campos em imagens de seleção

Características
Componentes
Entres.possíveis/verificação
Campos moeda/quantidade

Tipo incorporad
1 / 9

Componente	Tp...	Tipo componente	Categoria ...	Compr	Casa...	Descrição breve
UCOMM	<input type="checkbox"/>	SYUCOMM	CHAR	70	0	Código de função que acionou o PAI
FROM TEXT	<input type="checkbox"/>	TEXT12	CHAR	12	0	Texto com 12 dígitos
TO TEXT	<input type="checkbox"/>	TEXT12	CHAR	12	0	Texto com 12 dígitos
FUNCTXT 01	<input type="checkbox"/>	RSFUNC TXT	CHAR	144	0	Tela de seleção: texto para botões
FUNCTXT 02	<input type="checkbox"/>	RSFUNC TXT	CHAR	144	0	Tela de seleção: texto para botões
FUNCTXT 03	<input type="checkbox"/>	RSFUNC TXT	CHAR	144	0	Tela de seleção: texto para botões
FUNCTXT 04	<input type="checkbox"/>	RSFUNC TXT	CHAR	144	0	Tela de seleção: texto para botões
FUNCTXT 05	<input type="checkbox"/>	RSFUNC TXT	CHAR	144	0	Tela de seleção: texto para botões
SEARCH BTN	<input type="checkbox"/>	RSSELPUSH	CHAR	40	0	Telas de seleção: botão para conjunto de valores

# Ecrãs de selecção

A estrutura SSCRFIELDS permite-nos definir e testar os eventos dos ecrãs de selecção

## **AT USER-COMMAND.**

Testa o evento / código de função digitado pelo utilizador.

Sintaxe: AT USER-COMMAND.

## **AT LINE-SELECTION.**

Testa uma ação sobre uma linha específica de uma lista

Sintaxe: AT LINE-SELECTION.

# Ecrãs de selecção

## AT SELECTION-SCREEN.

Trata os blocos de eventos a serem tratados durante a execução do programa ABAP. Estes eventos podem ocorrer antes de se efectuar uma selecção de ecrã e depois de o utilizador ter dado alguma instrução sobre um ecrã de selecção.

### Sintaxe:

#### AT SELECTION-SCREEN... { OUTPUT }

```
| { ON {para|selcrit} }  
| { ON END OF selcrit }  
| { ON BLOCK block }  
| { ON RADIOBUTTON GROUP radi }  
| { }  
| { ON {HELP-REQUEST|VALUE-REQUEST}  
|   FOR {para|selcrit-low|selcrit-high} }  
| { ON EXIT-COMMAND }.
```

# Ecrãs de selecção

## Exemplo com RadioButtons / At Selection-Screen:

REPORT zselection\_screens06.

TABLES sscrfields.

PARAMETERS: rad1 RADIOBUTTON GROUP rad USER-COMMAND radio,  
                    rad2 RADIOBUTTON GROUP rad,  
                    rad3 RADIOBUTTON GROUP rad.

PARAMETERS check AS CHECKBOX USER-COMMAND check.

AT SELECTION-SCREEN.

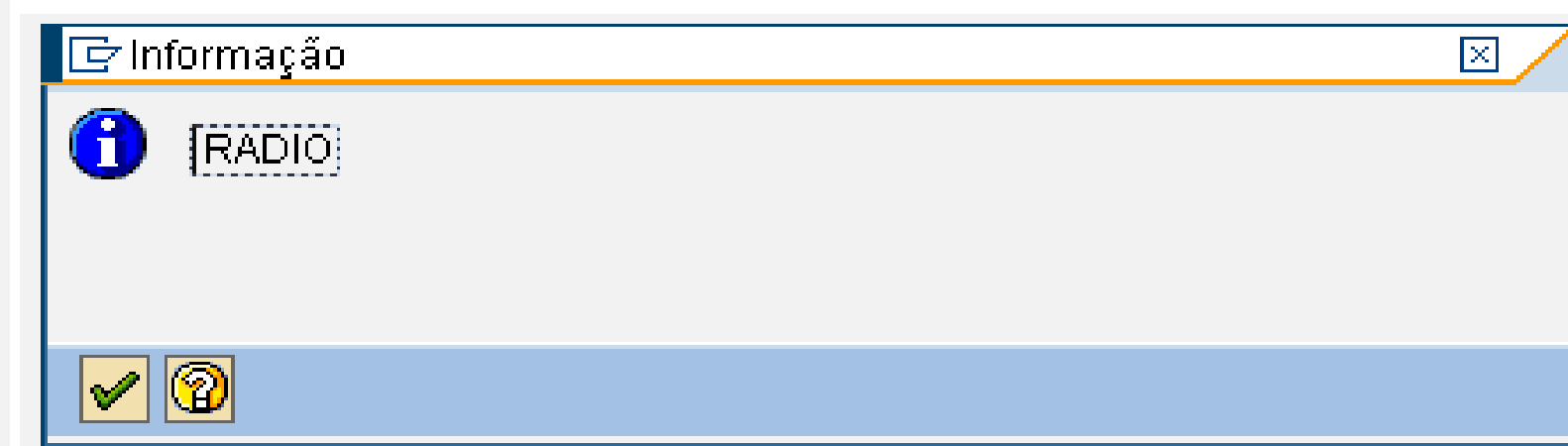
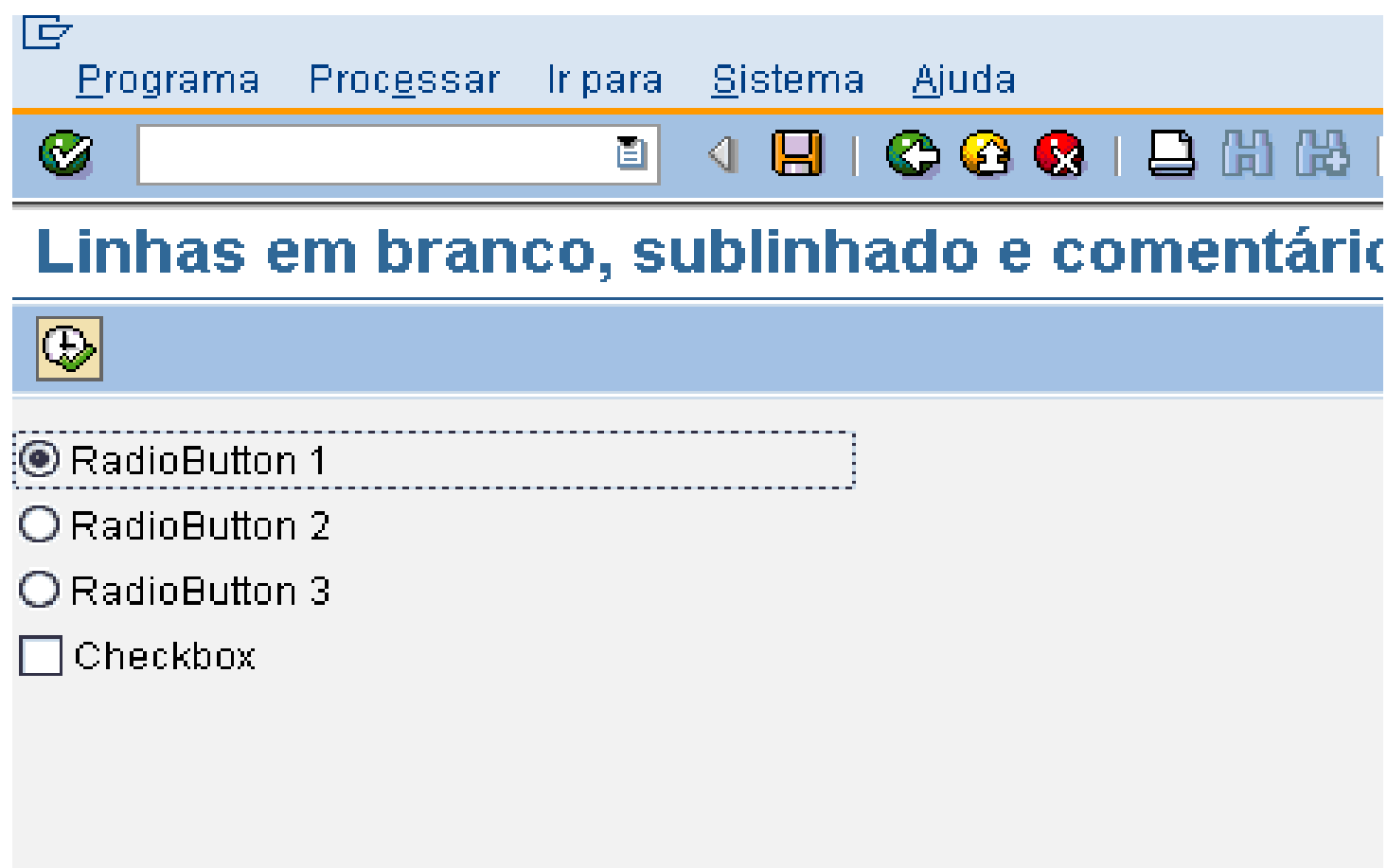
MESSAGE i888(sabapdocu) WITH text-001 sscrfields-ucomm.

START-OF-SELECTION.

WRITE text-002.

# Ecrãs de selecção

## Resultado





# Ecrãs de selecção

## Exemplo com Botões e Teclas de Função com a estrutura SSCRFIELDS

```
REPORT zselection_screens05.
```

```
TABLES sscrfields.
```

```
PARAMETERS: p_carrid TYPE s_carr_id,  
             p_cityfr TYPE s_from_cit.
```

```
SELECTION-SCREEN: FUNCTION KEY 1,  
                  FUNCTION KEY 2.
```

```
INITIALIZATION.
```

```
    sscrfields-functxt_01 = 'LH'.  
    sscrfields-functxt_02 = 'UA'.
```

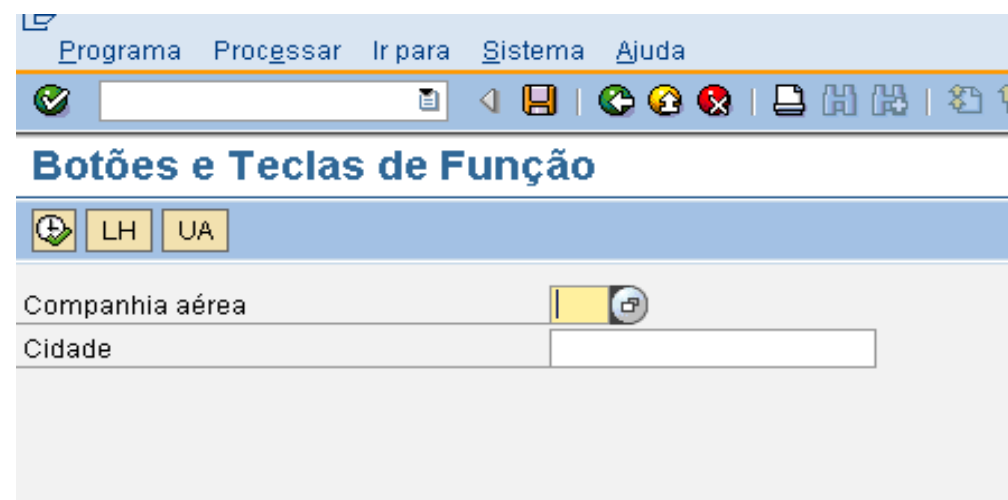
```
AT SELECTION-SCREEN.
```

```
    CASE sscrfields-ucomm.  
        WHEN 'FC01'.  
            p_carrid = 'LH'.  
            p_cityfr = 'Frankfurt'.  
        WHEN 'FC02'.  
            p_carrid = 'UA'.  
            p_cityfr = 'Chicago'.  
    ENDCASE.
```

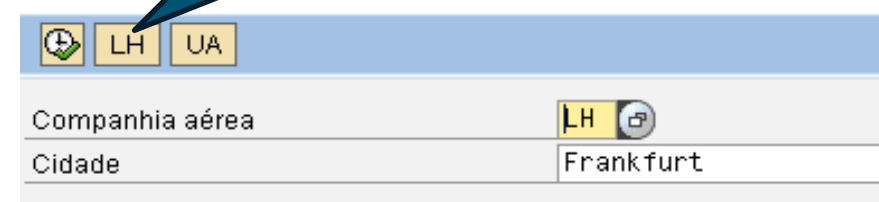
```
START-OF-SELECTION.
```

```
    WRITE / 'START-OF-SELECTION'.
```

# Ecrãs de selecção



Pressionando o botão LH, preenche os parâmetros com os dados de Frankfurt



Pressionando o botão UA, preenche os parâmetros com os dados de Chicago

# Ecrãs de selecção

## **Ajuda F1 na tela de selecção / Sintaxe:**

AT SELECTION-SCREEN ON HELP-REQUEST FOR <parameter>.

## **Ajuda F4 na tela de selecção:**

AT SELECTION-SCREEN ON VALUE-REQUEST FOR <parameter>.

# Ecrãs de selecção

REPORT demo\_selection\_screen\_f1.

PARAMETERS: p\_carr\_1 TYPE s\_carr\_id,  
p\_carr\_2 TYPE spfli-carrid.

AT SELECTION-SCREEN ON HELP-REQUEST FOR p\_carr\_2.

CALL SCREEN 100 STARTING AT 10 5  
ENDING AT 60 10.



# Ecrãs de selecção

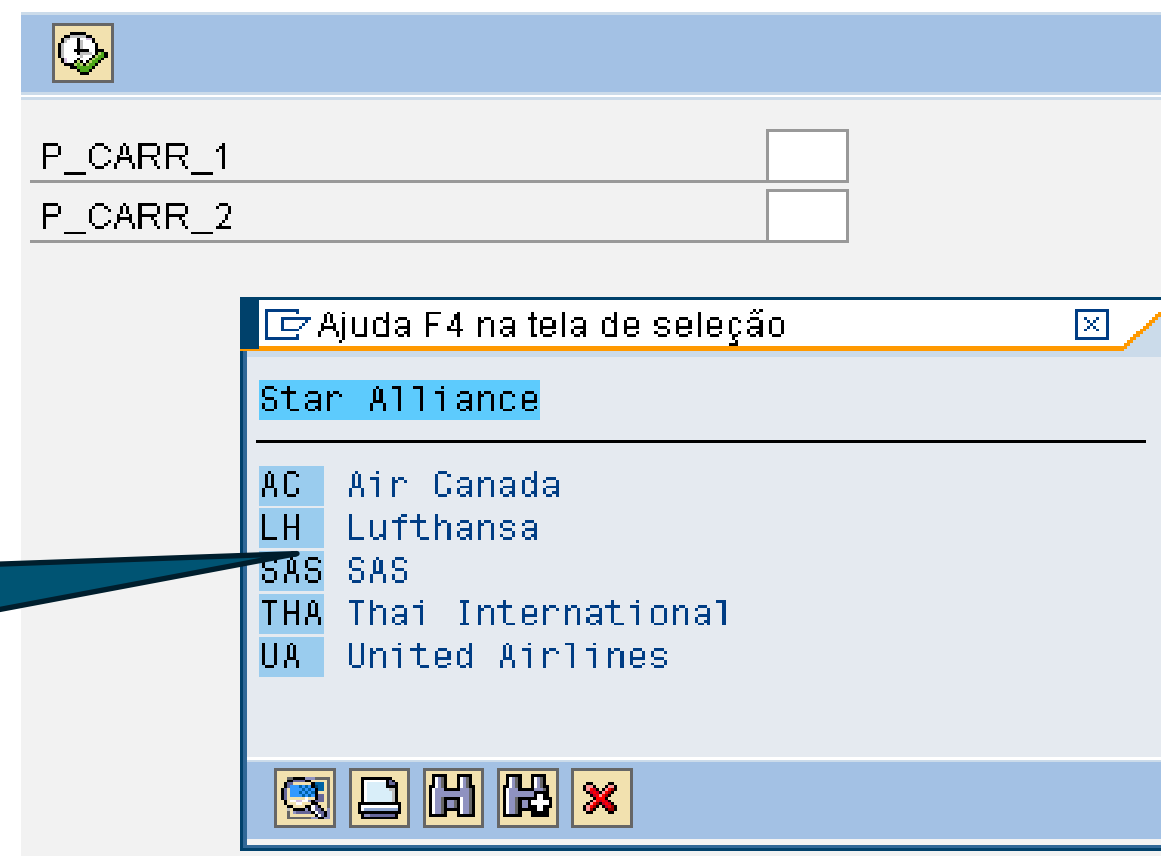
REPORT demo\_selection\_screen\_f4.

PARAMETERS: p\_carr\_1 TYPE spfli-carrid,  
p\_carr\_2 TYPE spfli-carrid.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR p\_carr\_2.  
CALL SCREEN 100 STARTING AT 10 5  
ENDING AT 50 10.

[...]

Após pressionar a tecla F4, é chamado o ecrã de selecção que disponibiliza o conteúdo.



# Ecrãs de selecção

## Exemplo com AT LINE-SELECTION e variáveis de sistema (SYST)

```
REPORT zlineselection01 NO STANDARD PAGE HEADING
      LINE-COUNT 12 LINE-SIZE 40.
```

```
DATA: tam TYPE i.
```

```
START-OF-SELECTION.
```

```
DO 100 TIMES.
  WRITE: / 'Iteração do Ciclo :', sy-index.
ENDDO.
```

```
END-OF-SELECTION.
```

```
TOP-OF-PAGE.
```

```
WRITE: 'Listagem Inicial, Pág.', sy-pagno.
ULINE.
```

```
TOP-OF-PAGE DURING LINE-SELECTION.
```

```
WRITE 'Listagem após seleccção de linha'.
ULINE.
```

# Ecrãs de selecção

## Exemplo com AT LINE-SELECTION e variáveis de sistema (SYST)

AT LINE-SELECTION.

DESCRIBE FIELD sy-lisel LENGTH tam IN BYTE MODE.

```
WRITE: 'SY-LSIND:',      sy-lsind,
      / 'SY-LISTI:',      sy-listi,
      / 'SY-LILLI:',      sy-lilli,
      / 'SY-CUROW:',      sy-curow,
      / 'SY-CUCOL:',      sy-cucol,
      / 'SY-CPAGE:',      sy-cpage,
      / 'SY-STARO:',      sy-staro,
      / 'SY-LISEL:',      sy-lisel,
      / 'Comprimento = ', tam.
```

# Ecrãs de selecção

## Output

Ao fazer duplo clique com o rato em cima de uma linha de iteração note-se a mudança da mensagem no topo da página, bem como a descrição das variáveis de sistema contendo os valores de posição da linha no ecrã.

**Initial Screen: Uso do line selection / Determinar**

Listagem Inicial, Pág. 1	
Iteração do Ciclo :	1
Iteração do Ciclo :	2
Iteração do Ciclo :	3
Iteração do Ciclo :	4
Iteração do Ciclo :	5
Iteração do Ciclo :	6
Iteração do Ciclo :	7
Iteração do Ciclo :	8
Iteração do Ciclo :	9
Iteração do Ciclo :	10

**Selected Screen: Uso do line selection / Determinar**

Listagem após selecção de linha

```

SY-LSIND: 1
SY-LISTI: 0
SY-LILLI: 7
SY-CUROW: 7
SY-CUCOL: 14
SY-CPAGE: 1
SY-STARO: 1
SY-LISEL:
Iteração do Ciclo : 5
Comprimento = 255
  
```



# Ecrãs de selecção

## Exemplo com AT LINE-SELECTION e HIDE

REPORT zlineselection02 NO STANDARD PAGE HEADING.

TABLES: bkpj, bseg.

SELECT-OPTIONS: s\_bukrs FOR bkpj-bukrs,  
                  s\_belnr FOR bkpj-belnr,  
                  s\_gjahr FOR bkpj-gjahr.

START-OF-SELECTION.

FORMAT HOTSPOT.

SELECT \* FROM bkpj  
      WHERE bukrs IN s\_bukrs  
          AND belnr IN s\_belnr  
          AND gjahr IN s\_gjahr.

WRITE: / bkpj-belnr, bkpj-budat, bkpj-xblnr.  
HIDE:   bkpj-belnr, bkpj-budat, bkpj-xblnr.

ENDSELECT. " BKPF

END-OF-SELECTION.

# Ecrãs de selecção

## Exemplo com AT LINE-SELECTION e HIDE (cont)

TOP-OF-PAGE.

WRITE 'Lista de Documentos'.

ULINE.

WRITE 'Número      Data Lnc. Doc.Refer.'.

ULINE.

AT LINE-SELECTION.

WINDOW STARTING AT 5 10 ENDING AT 60 20.

WRITE: / 'Linha: ', sy-curow, 'Coluna: ', sy-cucol.

SKIP 1.

WRITE: / 'Documento: ', bkpfbelnr.

SKIP 1.

WRITE: / 'Conteúdo linha seleccionada:', sy-lisel.

# Ecrãs de selecção

## Output

Lista Processar Ir para Sistema

**Exemplo de line-selection**

Lista de Documentos

Número	Data Lnc.	Doc.Refer.
6000000	2007.02.01	06080696
6000001	2007.03.27	2006/2131
6000002	2007.03.27	2006/2131
6000003	2007.03.27	2006/2131
6000004	2007.03.27	2006/2131
6000005	2007.03.27	2006/2131
6000006	2007.03.27	2006/2131
6000007	2007.03.27	2006/2131
6000008	2007.03.27	2006/2131
6000009	2007.03.27	2006/2131
6000010	2007.03.27	2006/2131
6000011	2007.03.27	2006/2131
6000012	2007.03.27	2006/2131
6000013	2007.03.01	2006/2131
6000014	2007.04.18	COLLOCADOS
6000015	2007.03.01	2006/2131
6000016	2007.03.31	2006/2131

Dando um duplo clique sobre a 1ª linha, note-se a posição 5 em vez da 1 atendendo a já terem sido escritas 4 linhas anteriormente (cabeçalho, underline, cabeçalho das colunas e novo linha de sublinhado e só depois o número de documento em si seguido da data e doc. Referência.  
SY-CUROW / SY-CUCOL – posição da linha e da coluna; SY-LISEL – conteúdo da linha seleccionada

Exemplo de line-selection com HIDE

Linha:	5	Coluna:	16
Documento:	6000000		
Conteúdo linha seleccionada:			
6000000	2007.02.01	06080696	

# Ecrãs de selecção

## Tabela SYST – Campos de Sistema ABAP

SAP

Estrutura Processar Ir para Utilitários Suplementos Ambiente Sistema Ajuda

Dictionary: exibir estrutura

Representação de hierarquia Estrutura append...

Estrut. SYST ativo

Descrição breve Campos de sistema ABAP

Características Componentes Entrs.possíveis/verificação Campos moeda/quantidade

Tipo incorporad 1 / 171

Componente	Tp...	Tipo componente	Categoria ...	Compr	Casa...	Descrição breve	Grupo
INDEX	<input type="checkbox"/>	SYINDEX	INT4	10	0	Contador de loops	
PAGNO	<input type="checkbox"/>	SY PAGNO	INT4	10	0	Página da lista atual	
TABIX	<input type="checkbox"/>	SYTABIX	INT4	10	0	Índice de tabelas internas	
TFILL	<input type="checkbox"/>	SYTFILL	INT4	10	0	Número de linhas de tabelas internas	
TLOPC	<input type="checkbox"/>	SYTLOPC	INT4	10	0	Campo do sistema interno	
TMAXL	<input type="checkbox"/>	SYTMAXL	INT4	10	0	Campo do sistema obsoleto	
TOCCU	<input type="checkbox"/>	SYTOCCU	INT4	10	0	Campo do sistema obsoleto	
TTABC	<input type="checkbox"/>	SYTTABC	INT4	10	0	Campo do sistema obsoleto	
TSTIS	<input type="checkbox"/>	SYTSTIS	INT4	10	0	Campo do sistema interno	
TTABI	<input type="checkbox"/>	SYTTABI	INT4	10	0	Campo do sistema obsoleto	
DBCNT	<input type="checkbox"/>	SYDBCNT	INT4	10	0	Entradas de tabela de banco de dados processadas	
FDPOS	<input type="checkbox"/>	SYFDPOS	INT4	10	0	Ocorrência em cadeia de bytes ou caracteres	
COLNO	<input type="checkbox"/>	SYCOLNO	INT4	10	0	Coluna atual na lista	
LINCT	<input type="checkbox"/>	SYLINCT	INT4	10	0	Comprimento das páginas da lista	
LINNO	<input type="checkbox"/>	SYLINNO	INT4	10	0	Linha atual na lista	
LINSZ	<input type="checkbox"/>	SYLINSZ	INT4	10	0	Largura da linha da lista	
PAGCT	<input type="checkbox"/>	SY PAGCT	INT4	10	0	Campo do sistema obsoleto	
MACOL	<input type="checkbox"/>	SYMACOL	INT4	10	0	Nº de colunas na margem esquerda de uma lista de impressão	
MAROW	<input type="checkbox"/>	SYMAROW	INT4	10	0	Nº de colunas na margem superior de uma lista de impressão	

GCD (3) (333) sapdeveu INS

03

# SAP List Viewer – ALV's

# SAP List Viewer - objetivos

No final deste módulo os formandos deverão ser capazes criar relatórios recorrendo à funcionalidade do SAP List Viewer (ALV).

Deverão estar aptos a utilizarem todos os módulos de função e grupos de tipos necessários à construção de relatórios com ALV's.

No final do módulo deverão ser capazes de criar ALV's simples, hierárquicas, editar colunas, tratar eventos, adicionar botões bem como utilizar o menu painter para o tratamento das teclas de função.

# SAP List Viewer

SAP List Viewer (ALV) é uma ferramenta flexível do SAP que permite a exibição de relatórios enriquecidos com um conjunto de funções standard que permitem ao utilizador o tratamento dos dados de acordo com as necessidades de visualização.

**Title**

**Generic functions of the toolbar**

**Output table in the grid control**

Book. no.	Cust. no.	...	...	Lug.weight	Unit	...	...	Amount
1	6	P		0,000	KG	Y		2.419,74 DEM 1485,56
2	38	P		50,000	KG	F		1.455,24 USD
3	3	P	X	16,000	KG	Y		1.485,56 USD
4	30	B	X	0,000	KG	C		2.395,540 ITL
5	9	P		0,000	KG	Y		1.424,93 USD
6	18	P		28,000	KG	F		1.515,88 USD
7	13	P	X	0,000	KG	Y		2.320,98 DEM 1424,93
8	42	P		0,000	KG	Y		1.515,88 USD 1515,88
9	8	P		0,000	KG	F		1.485,56 USD 1485,56
10	24	P		0,000	KG	Y		2.371,094 ITL 1470,40
11	5	P		0,000	KG	Y		1.485,56 USD 1485,56
12	25	P	X	0,000	KG	F		1.394,61 USD 1394,61
13	35	B		38,000	KG	C		2.197,52 DEM 1349,13
14	21	P		0,000	KG	Y		1.379,45 USD 1379,45

Return to main screen

# SAP List Viewer

## Características

Uma ALV permite:

- Exibição de listas não hierárquicas com um design moderno;
- Utilizar uma lista de funções típica sem quaisquer necessidades adicionais de programação (ordenação, somatórios, filtros diversos, etc.);
- Adaptar funções pré-definidas a outras necessidades;
- Responder a eventos despoletados pelo utilizador (click do rato nas linhas e colunas, nos botões, etc.);
- Efectuar um interface entre um relatório normal;
- Exibir relatórios no formato folha de cálculo, SAP Standard ou em árvore.



# SAP List Viewer

## Tratamento ABAP

O SAP List Viewer gira em torno do grupo de tipos SLIS, declarado através da instrução:

**TYPE-POOLS: SLIS.**

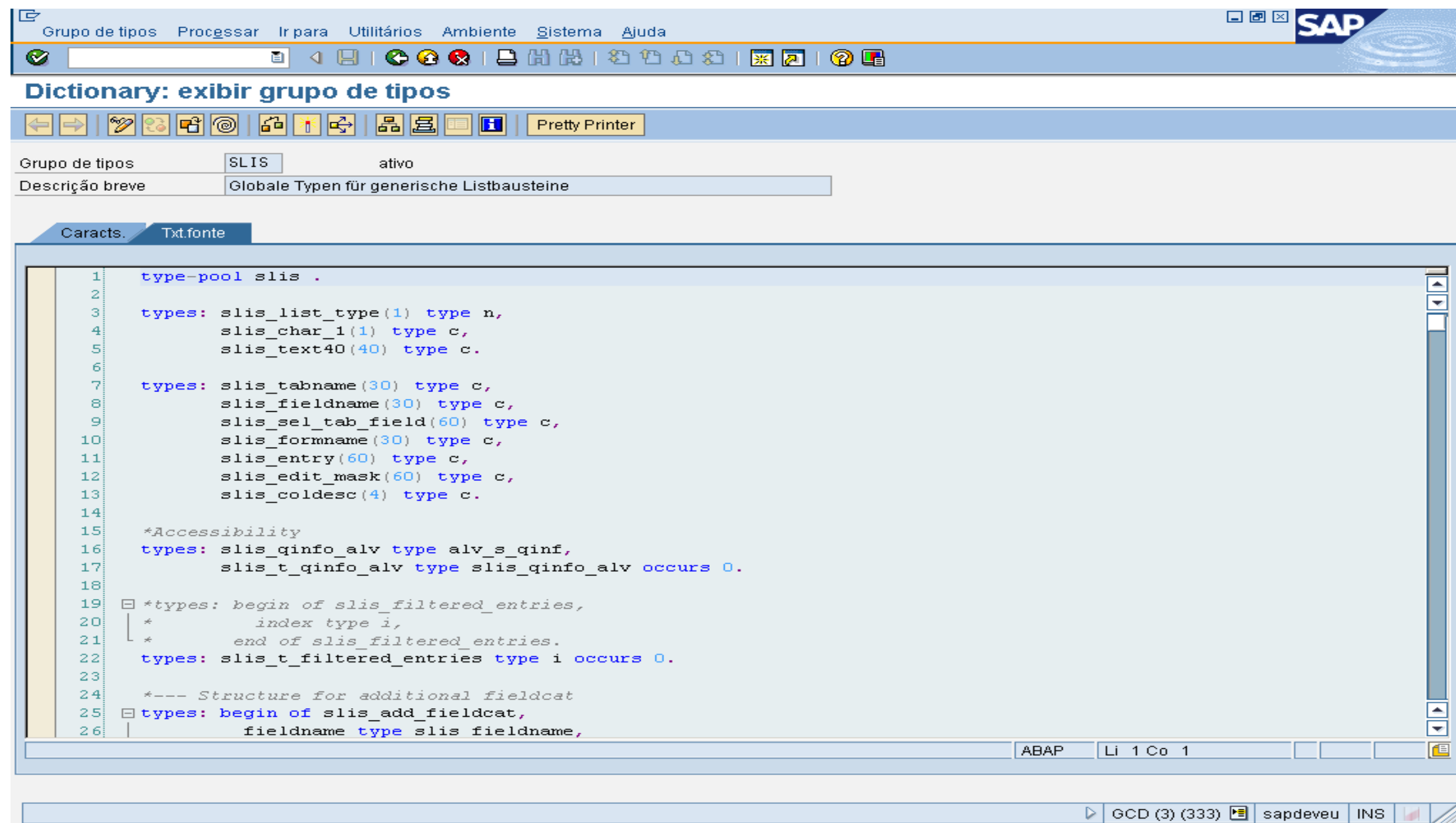
Objeto Dictionary Processar Ir para Utilitários Ambiente Sistema Ajuda

**ABAP Dictionary: entrada**

☐ Tab.banco dados  
☐ Visão  
☐ Categoria dados  
☒ Grupo de tipos  
☐ Domínio  
☐ Ajuda p/pesquisa  
☐ Objeto de bloqueio

Exibir Modificar Criar

# SAP List Viewer



# SAP List Viewer

**Para construir uma ALV são necessários 4 passos fundamentais:**

- Declaração de dados (estruturas, tabelas internas, variáveis)
- Leitura das bases de dados (SELECT, etc.)
- Construção do catálogo de colunas do relatório (FIELDATALOG)
- Preparação do Layout final, chamando o módulo de função:

**REUSE\_ALV\_GRID\_DISPLAY**

**Opcionalmente podemos ter passos adicionais para o tratamento de eventos, criação de botões, operações em linha e em coluna, etc.**

# SAP List Viewer

Para construir a funcionalidade da ALV recorre-se a tipos do pool de tipos SLIS, sendo os mais comuns:

SLIS\_LAYOUT\_ALV

SLIS\_T\_FIELDCAT\_ALV

SLIS\_SELFIELD

# SAP List Viewer

Preenchimento das colunas que dão origem ao FIELDCATALOG:

DATA: ls\_fieldcat TYPE slis\_fieldcat\_alv.

CLEAR pos.

\* *Field Catalog*

CLEAR ls\_fieldcat.

pos = pos + 1.

ls\_fieldcat-col\_pos = pos.

ls\_fieldcat-fieldname = 'KUNNR'.

ls\_fieldcat-reptext\_ddic = 'Nº Cliente'.

ls\_fieldcat-outputlen = '10'.

ls\_fieldcat-hotspot = 'X'.

APPEND ls\_fieldcat TO rt\_fieldcat.

CLEAR ls\_fieldcat.

pos = pos + 1.

ls\_fieldcat-col\_pos = pos.

ls\_fieldcat-fieldname = 'NAME1'.

ls\_fieldcat-reptext\_ddic = 'Nome'.

ls\_fieldcat-outputlen = '35'.

ls\_fieldcat-hotspot = ' '.

APPEND ls\_fieldcat TO rt\_fieldcat.

[...]

# SAP List Viewer

Preenchimento das colunas que dão origem ao FIELDATALOG:

O tipo SLIS\_FIELDCAT\_ALV compõe-se de dois subtipos:

- SLIS\_FIELDCAT\_MAIN

Que por sua vez se compões de dois subtipos:

SLIS\_FIELDCAT\_MAIN0

SLIS\_FIELDCAT\_MAIN1

- SLIS\_FIELDCAT\_ALV\_SPEC

É através dos tipos que se definem as propriedades e as características de apresentação das colunas.

# SAP List Viewer

Preenchimento das colunas que dão origem ao FIELDATALOG:

- É sempre necessário indicar a posição do campo na ALV, que deverá ser colocada no campo COL\_POS do respectivo tipo.
- O campo FIELDNAME conterá o nome da variável a ser visualizada
- O campo REPTXT\_DDIC conterá o título da coluna
- Se pretendermos controlar o tamanho da coluna, este será indicado no campo OUTPUTLEN
- Podem-se efectuar somatórios automáticos por coluna através da activação do campo DO\_SUM
- O campo DATATYPE permite indicar o tipo de dados; particularmente útil no tratamento de valores em moeda

# SAP List Viewer

Exemplo de ALV com uma listagem simples de clientes:

É gerada automaticamente uma barra de ferramentas com diversas funcionalidades, nomeadamente a integração directa do relatório com o MsExcel, bem como a criação de layouts específicos do utilizador.

Podem-se ordenar colunas, ocultá-las ou até disponibilizar funcionalidades de modo a que ao seleccionar-se uma coluna específica o utilizador possa transportar os dados seleccionados para visualização numa transacção específica.

**Exemplo de uma ALV - Formato Simples**

**Endereços de Clientes**

Nº Cliente	Nome	Morada	Cód.Postal	Localidade	Nº Fiscal
0006001123	ADELAIDE GRAÇA DA COSTA FRANCISCO		9999	Sem localidade	PT169029255
0006001124	MANUEL JESUS DA SILVA		9999	Sem localidade	PT169097633
0006001125	MARIA DA SILVA	" RUA DE BELOI, 783"	9999	RO DA COVA-GONDOMAR	PT169518221
0006001126	MARIA ALICE DE SOUSA MAIA	ED.17B PISO O/A CID.NOVA	2670	LOURES	PT169614328
0006001127	JOSE JOAQUIM FARIA CRUZ		9999	Sem localidade	PT169683591
0006001128	LUIS MANUEL SOARES CASTILHO MACEDO	" EST A-DOS-LOUCOS,LT02-2.B"	2600	VILA FRANCA DE XIRA	PT169861589
0006001129	CARLOS MARQUES DIAS	SARCENTO MOR - SOUSELAS	3000	COIMBRA	PT169870235
0006001130	MANUEL JOSE CHORA		2600	VILA FRANCA DE XIRA	PT170062058
0006001131	ADERITO DA SILVA CAR		2600	VILA FRANCA DE XIRA	PT170160475
0006001132	DOMINGOS RODRIGUE		2600	VILA FRANCA XIRA	PT170162052
0006001133	FERNANDO DA SILVA B		2600	VILA FRANCA XIRA	PT170164721
0006001134	ANTONIO MANUEL SIMO		2600	VILA FRANCA DE XIRA	PT170632679
0006001135	JOSE GUERREIRO CAS		9999	Sem localidade	PT170891216
0006001136	LUIS NUNES MARTINS		3100	LOULE	PT170891232
0006001137	CACILDA MARIA G. P. O.		2745	QUELUZ	PT170934500
0006001138	ALBINO DAS NEVES LO		3080	FIGUEIRA DA FOZ	PT171010531
0006001139	ANTONIO OLIVEIRA DUA		3000	COIMBRA	PT171057724
0006001140	JULIO MACEDO MATOS		3050	PAMPILHOSA	PT171057740
0006001141	JOAO ANTONIO FONTES		3130	SOURE	PT171057767
0006001142	AUGUSTO JOSE PINTO		9999	Sem localidade	PT171057775
0006001143	JOAQUIM SERRANO VENTURA SANTOS		9999	Sem localidade	PT171057783
0006001144	JOAQUIM FERNANDES P. MARTINS	ESTRADA DE LOGO DE DEUS	3000	COIMBRA	PT171057805
0006001145	JOAQUIM GONCALVES MONTEIRO	R CENTRAL	3020	SOUSELAS	PT171057813
0006001146	ARISTIDES SIMOES MARQUES	BRASFEMES	3000	COIMBRA	PT171057821
0006001147	JOSE AUGUSTO PEREIRA		9999	Sem localidade	PT171058046
0006001148	VALDEMAR FERREIRA MENDES	URB VALE FLORES LT.2-6.ES	3000	COIMBRA	PT171058054
0006001149	JOAQUIM LOPES PRIOR DE ALMEIDA	MARMELEIRA - SOUSELAS	3000	COIMBRA	PT171058062

GCD (2) (333) sapdeveu INS



# SAP List Viewer

Exemplo de ALV com uma listagem simples de clientes:

Lista Processar Ir para Visões Configurações Sistema Ajuda

Exemplo de uma ALV - Formato Simples

Endereços de Clientes

Nº Cliente	Nome	Morada	Cód.Postal	Localidade	Nº Fiscal
0006001123	ADELAIDE GRAÇA DA COSTA FRANCISCO		9999	Sem localidade	PT169029255
0006001124	MANUEL JESUS DA SILVA		9999	Sem localidade	PT169097633
0006001125	MARIA DA SILVA	" RUA DE BELOI, 783"	9999	RO DA COVA-GONDOMAR	PT169518221
0006001126	MARIA ALICE DE SOUSA MAIA	ED.17B PISO 0/A CID.NOVA	2670	LOURES	PT169614328
0006001127	JOSE JOAQUIM FARIA CRUZ		9999	Sem localidade	PT169683591
0006001128	LUIS MANUEL SOARES CASTILHO MACEDO	" EST A-DOS-LOUCOS,LT02-2.B"	2600	VILA FRANCA DE XIRA	PT169861589
0006001129	CARLOS MARQUES DIAS	SARGENTO-MOR - SOUSELAS	3000	COIMBRA	PT169870235
0006001130	MANUEL JOSE CHORA	R PASSOS MANUEL 107-1 DTO	2600	VILA FRANCA DE XIRA	PT170062058
0006001131	ADERITO DA SILVA CARRELHAS	" QUINTA DA FE, LT.3-3F"	2600	VILA FRANCA DE XIRA	PT170160475
0006001132	DOMINGOS RODRIGUES BERTOLO	BECO DO SAPAL 8 R/C	2600	VILA FRANCA XIRA	PT170162052
0006001133	FERNANDO DA SILVA BARREIROS	R PALHA BLANCO LT 4-2 ESQ	2600	VILA FRANCA XIRA	PT170164721
0006001134	ANTONIO MANUEL SIMOES BRANDAO	PQUE RESIDENCIAL LAMEIRAS	2600	VILA FRANCA DE XIRA	PT170632679
0006001135	JOSE GUERREIRO CASANOVA		9999	Sem localidade	PT170891216
0006001136	LUIS NUNES MARTINS	R JOSE COSTA GUERREIRO 79	8100	LOULE	PT170891232
0006001137	CACILDA MARIA G. P. O. FONTES	URB.CIDADE DESPORTIVA	2745	QUELUZ	PT170934500
0006001138	ALBINO DAS NEVES LOUREIRO	BOM SUCESSO	3080	FIGUEIRA DA FOZ	PT171010531
0006001139	ANTONIO OLIVEIRA DUARTE	PACO - BOTAO	3000	COIMBRA	PT171057724
0006001140	JULIO MACEDO MATOS	R.25 DE ABRIL	3050	PAMPILHOSA	PT171057740
0006001141	JOAO ANTONIO FONTES MOREIRA	GRANJA DO ULMEIRO	3130	SOURE	PT171057767
0006001142	AUGUSTO JOSE PINTO		9999	Sem localidade	PT171057775
0006001143	JOAQUIM SERRANO VENTURA SANTOS		9999	Sem localidade	PT171057783
0006001144	JOAQUIM FERNANDES P. MARTINS	ESTRADA DE LOGO DE DEUS	3000	COIMBRA	PT171057805
0006001145	JOAQUIM GONCALVES MONTEIRO	R CENTRAL	3020	SOUSELAS	PT171057813
0006001146	ARISTIDES SIMOES MARQUES	BRASFEMES	3000	COIMBRA	PT171057821
0006001147	JOSE AUGUSTO PEREIRA		9999	Sem localidade	PT171058046
0006001148	VALDEMAR FERREIRA MENDES	URB VALE FLORES LT.2-6.ES	3000	COIMBRA	PT171058054
0006001149	JOAQUIM LOPES PRIOR DE ALMEIDA	MARMELEIRA - SOUSELAS	3000	COIMBRA	PT171058062

GCD (2) (333) sapdeveu INS

# SAP List Viewer – código fonte

REPORT zalvsimples01.

TYPE-POOLS: slis.

TABLES: kna1. " *Mestre de clientes*

TYPES: BEGIN OF t\_cliente,  
         kunnr LIKE kna1-kunnr,  
         name1 LIKE kna1-name1,  
         stras LIKE kna1-stras,  
         pstlz LIKE kna1-pstlz,  
         ort01 LIKE kna1-ort01,  
         stceg LIKE kna1-stceg,  
 END OF t\_cliente.

DATA: gt\_outtab TYPE STANDARD TABLE OF t\_cliente.

DATA: gs\_layout TYPE slis\_layout\_alv,  
       gt\_fieldcat TYPE slis\_t\_fieldcat\_alv,  
       ls\_selfield TYPE slis\_selfield,  
       g\_repid LIKE sy-repid,  
       pos TYPE i,  
       ls\_outtab TYPE t\_cliente.

SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE text-000.  
 SELECTION-SCREEN SKIP 1.  
 SELECT-OPTIONS: s\_kunnr FOR kna1-kunnr.  
 SELECTION-SCREEN SKIP 1.  
 SELECTION-SCREEN END OF BLOCK b1.

START-OF-SELECTION.

g\_repid = sy-repid.

REFRESH: gt\_outtab, gt\_fieldcat.

SELECT kunnr  
       name1  
       stras  
       pstlz  
       ort01  
       stceg FROM kna1  
       INTO CORRESPONDING FIELDS OF TABLE gt\_outtab  
       WHERE kunnr IN s\_kunnr.  
 PERFORM fieldcat\_init USING gt\_fieldcat[].

PERFORM display\_alv\_clientes.

END-OF-SELECTION.

# SAP List Viewer – código fonte (cont.)

```
FORM fieldcat_init USING rt_fieldcat TYPE slis_t_fieldcat_alv.
  DATA: ls_fieldcat TYPE slis_fieldcat_alv.
```

```
  CLEAR pos.
```

```
  * Field Catalog
```

```
  CLEAR ls_fieldcat.
  pos = pos + 1.
  ls_fieldcat-col_pos      = pos.
  ls_fieldcat-fieldname    = 'KUNNR'.
  ls_fieldcat-reptext_ddic = 'Nº Cliente'.
  ls_fieldcat-outputlen   = '10'.
  ls_fieldcat-hotspot     = 'X'.
  APPEND ls_fieldcat TO rt_fieldcat.
```

```
  CLEAR ls_fieldcat.
  pos = pos + 1.
  ls_fieldcat-col_pos      = pos.
  ls_fieldcat-fieldname    = 'NAME1'.
  ls_fieldcat-reptext_ddic = 'Nome'.
  ls_fieldcat-outputlen   = '35'.
  ls_fieldcat-hotspot     = ' '.
  APPEND ls_fieldcat TO rt_fieldcat.
```

```
  CLEAR ls_fieldcat.
  pos = pos + 1.
  ls_fieldcat-col_pos      = pos.
  ls_fieldcat-fieldname    = 'STRAS'.
  ls_fieldcat-reptext_ddic = 'Morada'.
  ls_fieldcat-outputlen   = '35'.
  ls_fieldcat-hotspot     = ' '.
  APPEND ls_fieldcat TO rt_fieldcat.
```

```
  CLEAR ls_fieldcat.
  pos = pos + 1.
  ls_fieldcat-col_pos      = pos.
  ls_fieldcat-fieldname    = 'PSTLZ'.
  ls_fieldcat-reptext_ddic = 'Cód.Postal'.
  ls_fieldcat-outputlen   = '10'.
  ls_fieldcat-hotspot     = ' '.
  APPEND ls_fieldcat TO rt_fieldcat.
```

```
  CLEAR ls_fieldcat.
  pos = pos + 1.
  ls_fieldcat-col_pos      = pos.
  ls_fieldcat-fieldname    = 'ORT01'.
  ls_fieldcat-reptext_ddic = 'Localidade'.
  ls_fieldcat-outputlen   = '35'.
  ls_fieldcat-hotspot     = ' '.
  APPEND ls_fieldcat TO rt_fieldcat.
```

```
  CLEAR ls_fieldcat.
  pos = pos + 1.
  ls_fieldcat-col_pos      = pos.
  ls_fieldcat-fieldname    = 'STCEG'.
  ls_fieldcat-reptext_ddic = 'Nº Fiscal'.
  ls_fieldcat-outputlen   = '20'.
  ls_fieldcat-hotspot     = ' '.
  APPEND ls_fieldcat TO rt_fieldcat.
```

```
ENDFORM.                " fieldcat_init FORM display_alv_clientes .
```

# SAP List Viewer – código fonte (cont.)

```

DATA: w_lvc_title TYPE lvc_title.

MOVE 'Endereços de Clientes' TO w_lvc_title.

CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
    i_callback_program      = g_repid
    it_fieldcat              = gt_fieldcat[]
    i_callback_pf_status_set = 'STANDARD_FULLSCREEN'
    i_callback_user_command = 'USER_COMMAND'
    is_layout                = gs_layout
    i_grid_title             = w_lvc_title
  TABLES
    t_outtab                = gt_outtab.

ENDFORM.                " display_alv_clientes FORM user_command USING r_ucomm TYPE sy-ucomm
                        rs_selfield TYPE slis_selfield.

CLEAR ls_outtab.

CASE r_ucomm.
  WHEN '&IC1'.
    CASE rs_selfield-fieldname.
      WHEN 'KUNNR'.
        READ TABLE gt_outtab INDEX rs_selfield-tabindex
          INTO ls_outtab.
        SET PARAMETER ID 'KUN' FIELD ls_outtab-kunnr.

        CALL TRANSACTION 'FD03'.
      ENDCASE.
    ENDCASE.
ENDFORM. " user_command

```

# SAP List Viewer – código fonte (cont.)

```
FORM standard_fullscreen USING extab TYPE slis_t_extab.
```

```
    SET PF-STATUS 'STANDARD_FULLSCREEN' EXCLUDING extab.
```

```
ENDFORM.                "STANDARD_FULLSCREEN
```

# SAP List Viewer

## Preenchimento automático das colunas da ALV (Fieldcatalog)

Efectua-se recorrendo ao módulo de função:

### **REUSE\_ALV\_FIELDATALOG\_MERGE**

Se o corpo da ALV for definido através de uma estrutura ou de uma tabela transparente, podem-se obter automaticamente as descrições dos elementos de dados, as quais surgirão como título das colunas, sem termos de preencher toda a tabela do fieldcatalog.

Posteriormente, caso sejam necessárias algumas alterações ao formato standard que é obtido directamente a partir das estruturas/tabelas, pode-se percorrer a tabela interna para modificar as propriedades das colunas.

# SAP List Viewer

## Reuse\_ALV\_Fieldcatalog\_Merge – Exemplo:

```
REPORT zalvsimples02.
TYPE-POOLS: slis.
DATA:it_fieldcat TYPE slis_fieldcat_alv OCCURS 0 WITH HEADER LINE,
      gs_layout TYPE slis_layout_alv,
      v_repid TYPE sy-repid.
```

```
DATA: BEGIN OF itab OCCURS 0.
      INCLUDE STRUCTURE T001.
DATA: END OF itab.
```

```
START-OF-SELECTION.
  v_repid = sy-repid.
  *--- Selecção dos dados das empresas
  SELECT * FROM t001
  INTO TABLE itab UP TO 100 ROWS.
  PERFORM field_catalog.
```

```
CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
    i_callback_program      = v_repid
    i_callback_user_command = 'USER_COMMAND'
    it_fieldcat              = it_fieldcat[]
    is_layout               = gs_layout
  TABLES
    t_outtab                = itab.
```

```
FORM field_catalog.
  CALL FUNCTION 'REUSE_ALV_FIELDCATALOG_MERGE'
  EXPORTING
    i_program_name      = v_repid
    i_internal_tabname  = 'ITAB'
    i_inclname          = v_repid
  CHANGING
    ct_fieldcat         = it_fieldcat[].
ENDFORM.                " field_catalog
```

As colunas da ALV assumem o mesmo título que está na definição do Dicionário de Dados para a tabela T001 (Empresas):



The screenshot shows the SAP List Viewer interface. The title bar reads "Exemplo de uma ALV - Uso do REUSE\_ALV\_FIELDCATALOG\_MERGE". The table displayed has the following columns: Empresa, Nome da firma, Local, País, Moeda, L..., PICT, ..., VE, C, N°so, and Endereço. The data rows are as follows:

Empresa	Nome da firma	Local	País	Moeda	L...	PICT	...	VE	C	N°so	Endereço
0001	Division España	Barcelona	ES	EUR	ES			K4			50204344
B000	Cimpor Betão,SGPS, SA	Lisboa	PT	EUR	PT	POC	10	K4	2	PT004	50201315
B001	Cimpor Betão Indústria,SA	Lisboa	PT	EUR	PT	POC	10	K4	2	PT005	50201316
B002	Betão Liz, SA	Lisboa	PT	EUR	PT	POC	10	K4	2	PT006	50201317
B003	Sopab (Dissol. B.Liz)	Lisboa	PT	EUR	PT	POC	10	K4	2	PT007	50201335
B004	Betasa (Fusão CBIInd.)	Lisboa	PT	EUR	PT	POC	10	K4	2	PT008	50201336
B005	Betazul (Fusão CBIInd.)	Lisboa	PT	EUR	PT	POC	10	K4	2	PT009	50201337
B006	Jomatel, SA	Lisboa	PT	EUR	PT	POC	10	K4	2	PT010	50201338
B007	Betão (Fusão CBIInd.)	Lisboa	PT	EUR	PT	POC	10	K4	2	PT011	50201339



# SAP List Viewer

## STANDARD\_FULLSCREEN

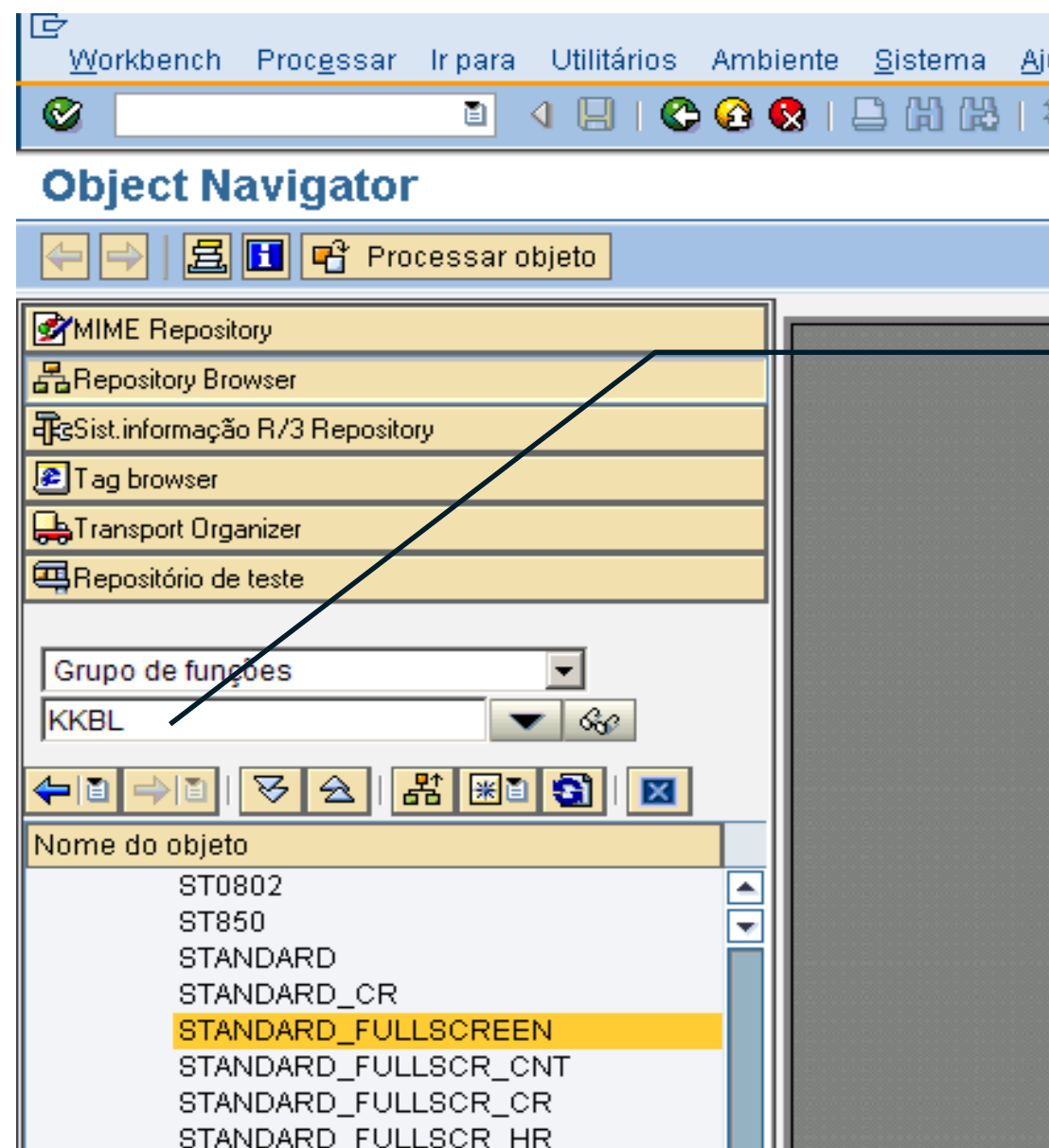
- O standard fullscreen é um interface de utilizador standard no qual residem os botões e funções disponibilizadas pelo SAP List Viewer;
- Este interface standard pode ser copiado de uns programas para os outros, não sendo necessário criá-lo de raiz, podendo visualizar ou omitir funcionalidades (Botões);
- Para que a barra de ferramentas funcione, deverá ser chamada no módulo de função e declarada uma subrotina com exactamente o mesmo nome que foi dado ao interface de utilizador;
- Para associar este Gui Status ao programa que exhibirá relatório via ALV, pode-se por exemplo aceder à transacção SE41 para copiar o Interface de utilizador de um já existente para este;
- No caso de não existir nenhum programa feito à medida que já contenha este Gui Status, pode-se recorrer a um standard existente em todas as máquinas SAP, copiando-o para o relatório onde se pretende utilizar a barra de ferramentas standard do SAP List Viewer.

## USER\_COMMAND

- Quando chamado no módulo de função de display da ALV, permite controlar as acções do utilizador;
- Para que seja possível controlar a acção do utilizador sobre as teclas de função, é necessário criar uma subrotina denominada USER\_COMMAND onde serão tratados os respectivos eventos.



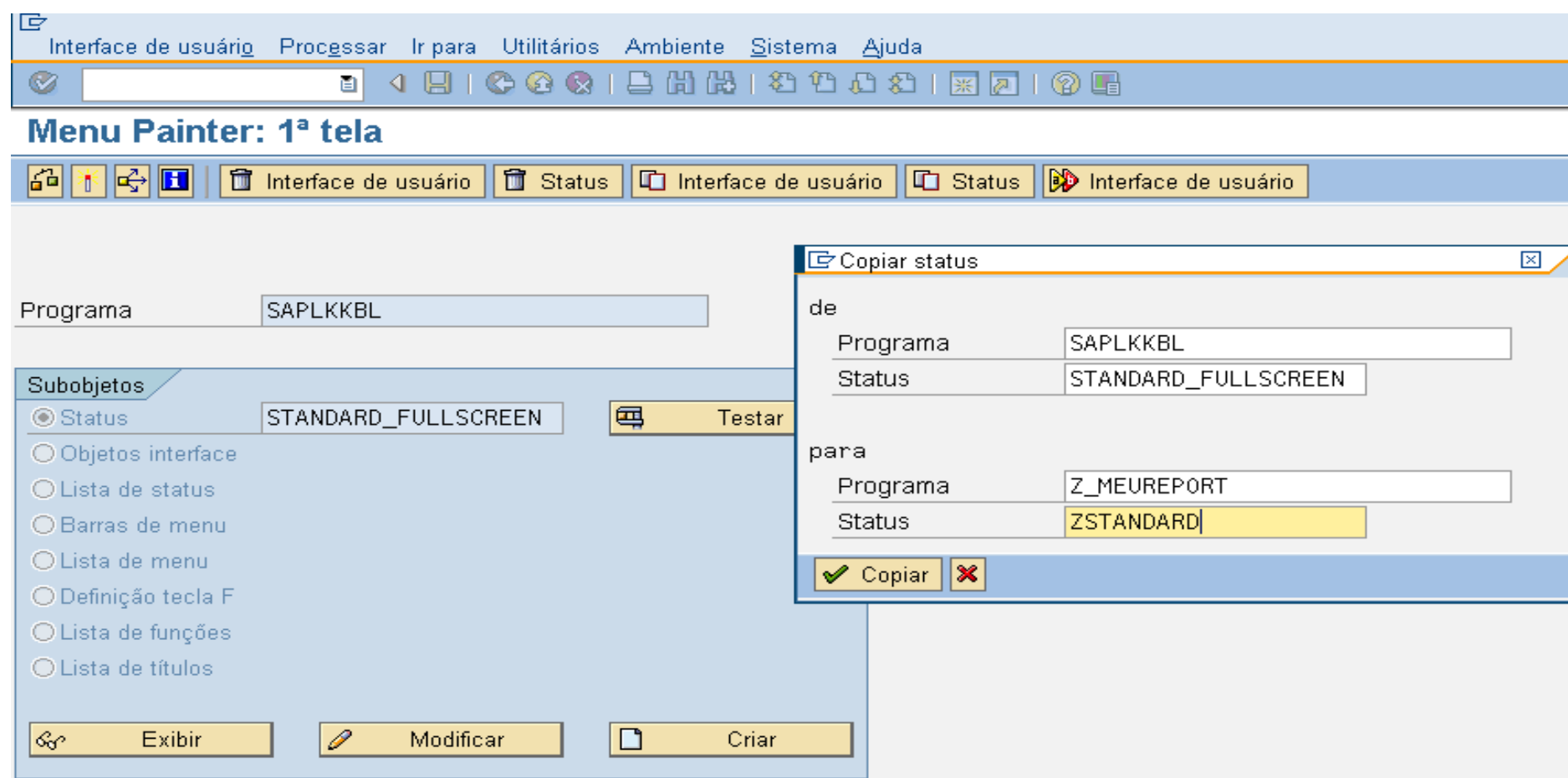
# SAP List Viewer



O programa SAPLKKBL do grupo de funções KKBL contém a barra de ferramentas standard que pode ser copiada para o novo relatório

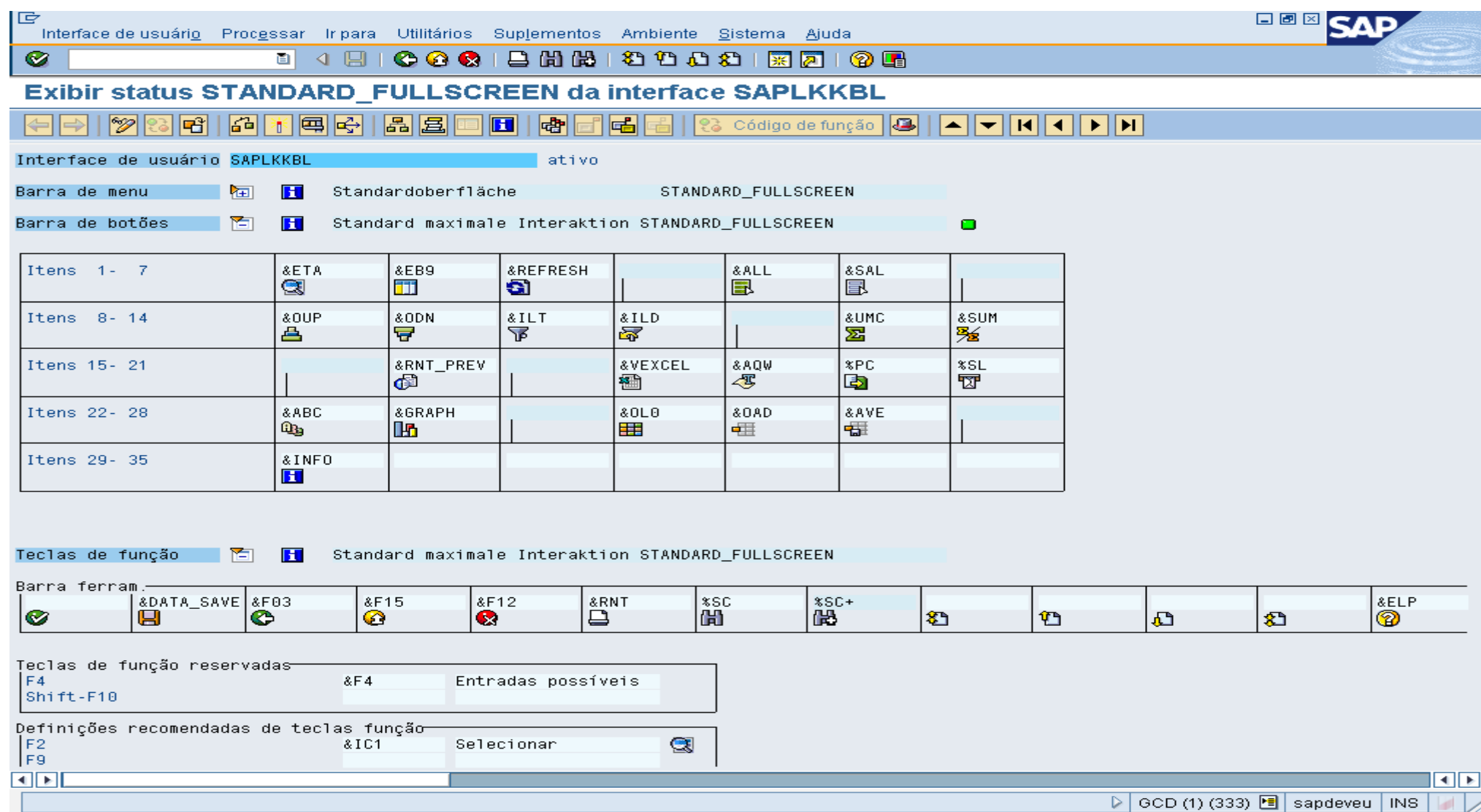
# SAP List Viewer

No Menu Painter (Transacção SE41) podemos copiar o Status de um programa já existente para um outro programa desenvolvido à medida:



# SAP List Viewer

Aspecto geral das funcionalidades do STANDARD\_FULLSCREEN:



# SAP List Viewer

## Funcionalidades do Layout do SAP List Viewer

Pressionando o botão Modificar Layout da barra de ferramentas podemos modificar o aspecto do Layout que é produzido pelo relatório, gravando-o como um Template que pode ser reutilizado sempre que necessário e até chamado directamente pelo programa de modo a que após a selecção dos dados, estes sejam automaticamente apresentados de acordo com o layout estipulado pelo utilizador.

Lista Processar Ir para Visões Configurações Sistema Ajuda

Exemplo de uma ALV - Formato Simples

Endereços de Clientes

Nº Cliente	Nome	Morada
0006001123	ADELAIDE GRAÇA DA COSTA FRANCISCO	
0006001124	MANUEL JESUS DA SILVA	
0006001125	MARIA DA SILVA	" RUA DE BELOI,
0006001126	MARIA ALICE DE SOUSA MAIA	ED.17B PISO O/A

**Pressionando o botão Modificar Layout permite alterar o modo de visualização do relatório, gravando-o sob a forma de template reutilizável**

Lista Processar Ir para Visões Configurações Sistema Ajuda

Exemplo de um ALV - Formato Tabelar

Modificar layout

SeleçãoColunas Ordenação Filtro Visão Representação

Colunas exibidas		Lista colunas
Nome coluna		Nome coluna
Nº Cliente		
Nome		
Morada		
Cód.Postal		
Localidade		
Nº Fiscal		

Endereços de Clientes

Nº Cliente	Nome	Morada
0006001123	ADELAIDE GRAÇA DA COSTA FRANCISCO	
0006001124	MANUEL JESUS DA SILVA	
0006001125	MARIA DA SILVA	" RUA DE BELOI,
0006001126	MARIA ALICE DE SOUSA MAIA	ED.17B PISO O/A

PACO - BOTAO 3000 COIMBRA

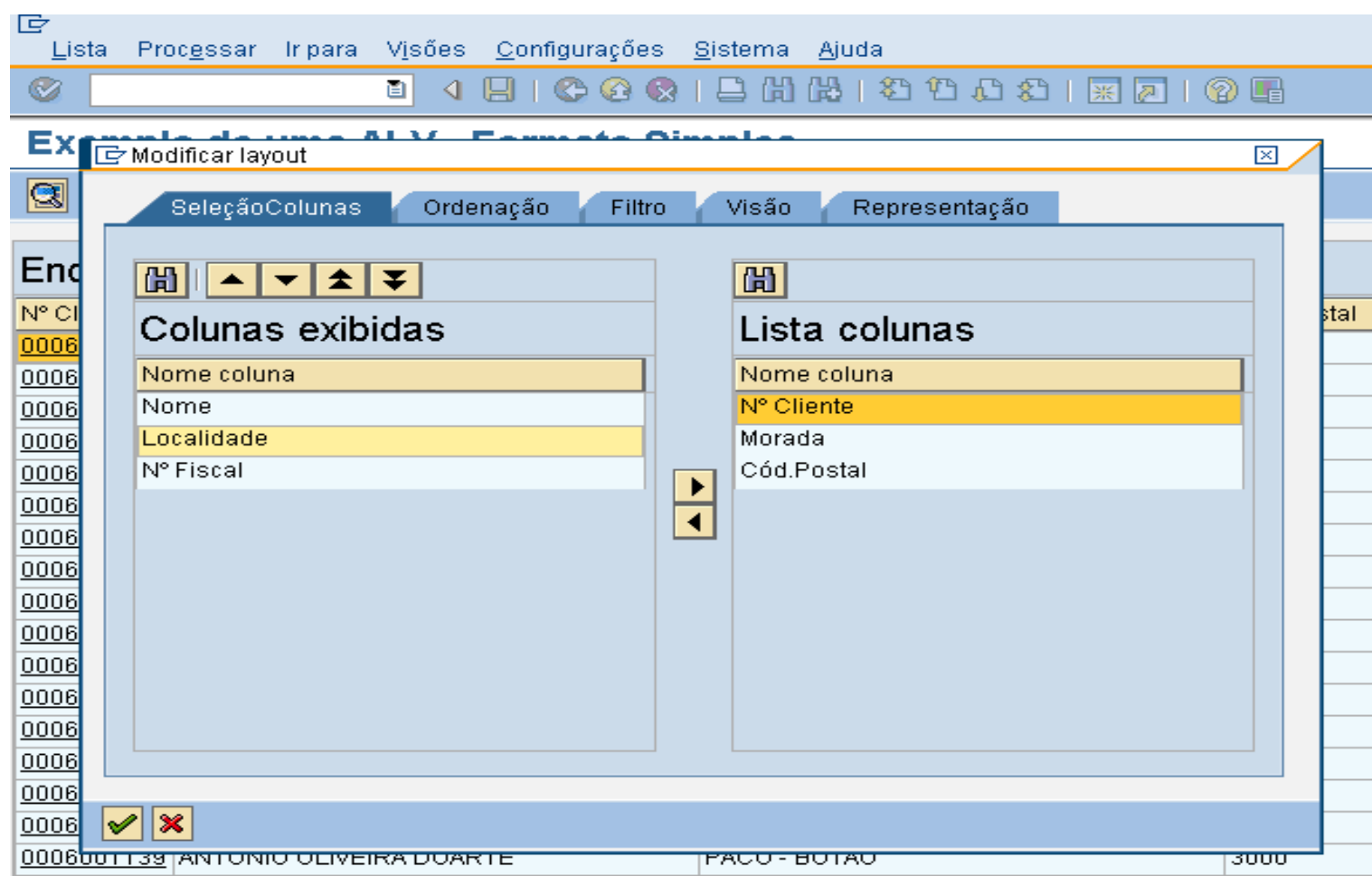
# SAP List Viewer

O SAP List Viewer disponibiliza diversas funcionalidades, destacando-se:

- Selecção das colunas a exibir;
- Ordenação por coluna(s);
- Critérios de Filtro;
- Visões;
- Modos de Representação ( títulos de coluna, separadores, etc. );
- Gravação do Layout;
- Possibilidade de associação de Layouts ao username;

# SAP List Viewer

Mostrar apenas o Nome, Morada e N° Fiscal na ALV do Exemplo:



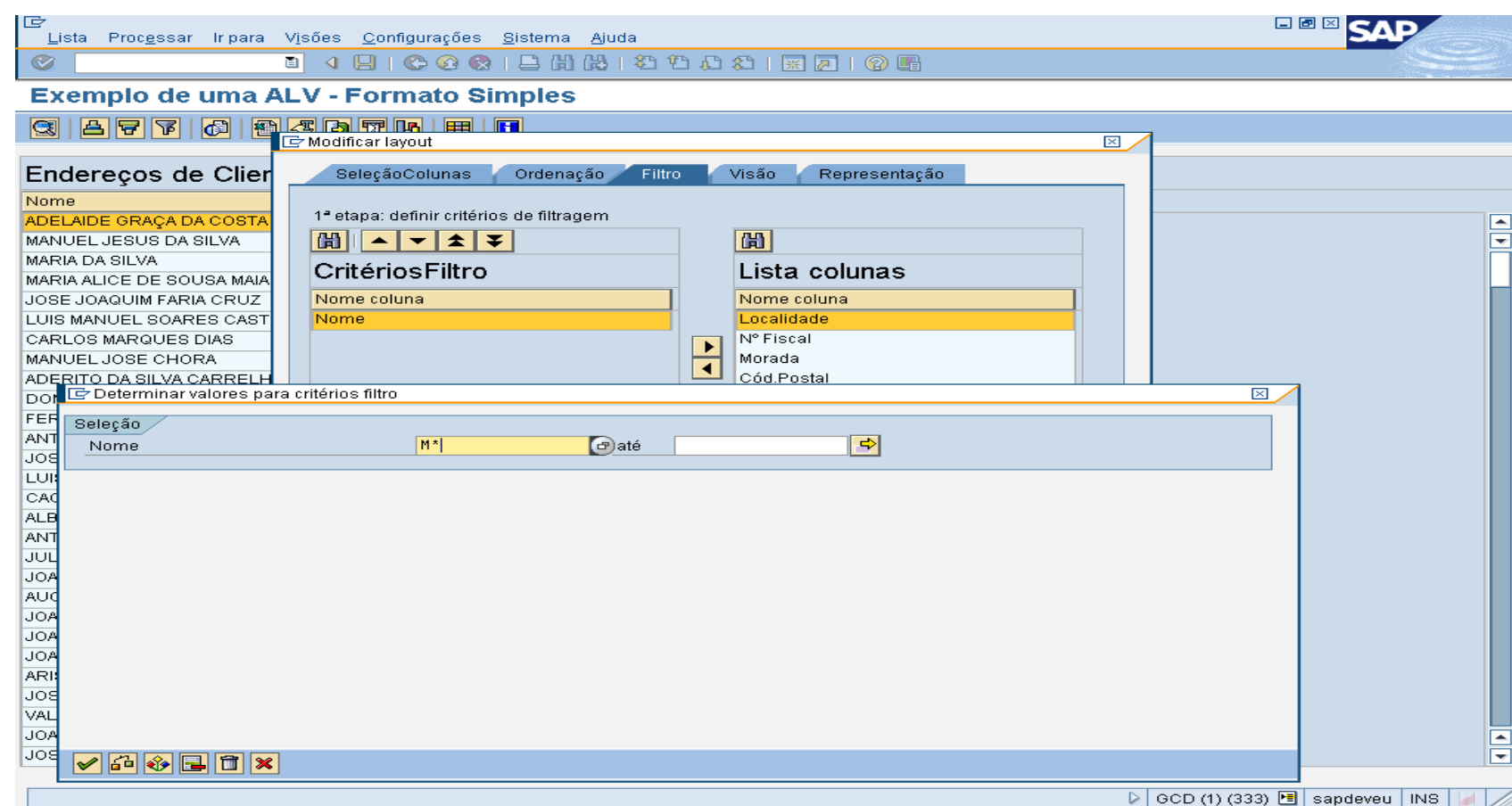
## Resultado

Exemplo de uma ALV - Formato Simples

Nome	Localidade	N° Fiscal
ADELAIDE GRAÇA DA COSTA FRANCISCO	Sem localidade	PT169029255
MANUEL JESUS DA SILVA	Sem localidade	PT169097633
MARIA DA SILVA	RO DA COVA-GONDOMAR	PT169518221
MARIA ALICE DE SOUSA MAIA	LOURES	PT169614328
JOSE JOAQUIM FARIA CRUZ	Sem localidade	PT169683591
LUIS MANUEL SOARES CASTILHO MACEDO	VILA FRANCA DE XIRA	PT169861589
CARLOS MARQUES DIAS	COIMBRA	PT169870235

# SAP List Viewer

Criar Filtro para exibir apenas os clientes cujo nome começa pela letra M:



## Resultado

**Exemplo de uma ALV - Formato Simples**

**Endereços de Clientes**

Nome	Localidade	Nº Fiscal
MANUEL JESUS DA SILVA	Sem localidade	PT169097633
MARIA DA SILVA	RO DA COVA-GONDOMAR	PT169518221
MARIA ALICE DE SOUSA MAIA	LOURES	PT169614328
MANUEL JOSE CHORA	VILA FRANCA DE XIRA	PT170062058
MANUEL ALBERTO RODRIG.FERREIRA	SOUSELAS	PT171162064
MARIO MANUEL PRATAS DA CRUZ	COIMBRA	PT171917120
MANUEL AIRES DA COSTA	COIMBRA	PT172162203
MANUEL OLIVEIRA TEIXEIRA	PENACOVA	PT172278058

# SAP List Viewer

## O mesmo exemplo com o parâmetro I\_SAVE = 'X' :


Para activar o botão SAVE de modo a possibilitar a gravação de layouts bem como orientá-los ao utilizador ou pré defini-los, é necessário preencher com 'X' o parâmetro de Input I\_SAVE do módulo de função:

### REUSE\_ALV\_GRID\_DISPLAY

```
CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
    i_callback_program      = g_repid
    it_fieldcat              = gt_fieldcat[]
    i_callback_pf_status_set = 'STANDARD_FULLSCREEN'
    i_callback_user_command = 'USER_COMMAND'
    is_layout                = gs_layout
    i_grid_title             = w_lvc_title
    i_save                   = 'X'
  TABLES
    t_outtab                = gt_outtab.
```

**Adicionados automaticamente os botões Seleccionar Layout e Gravar Layout.**

## Resultado



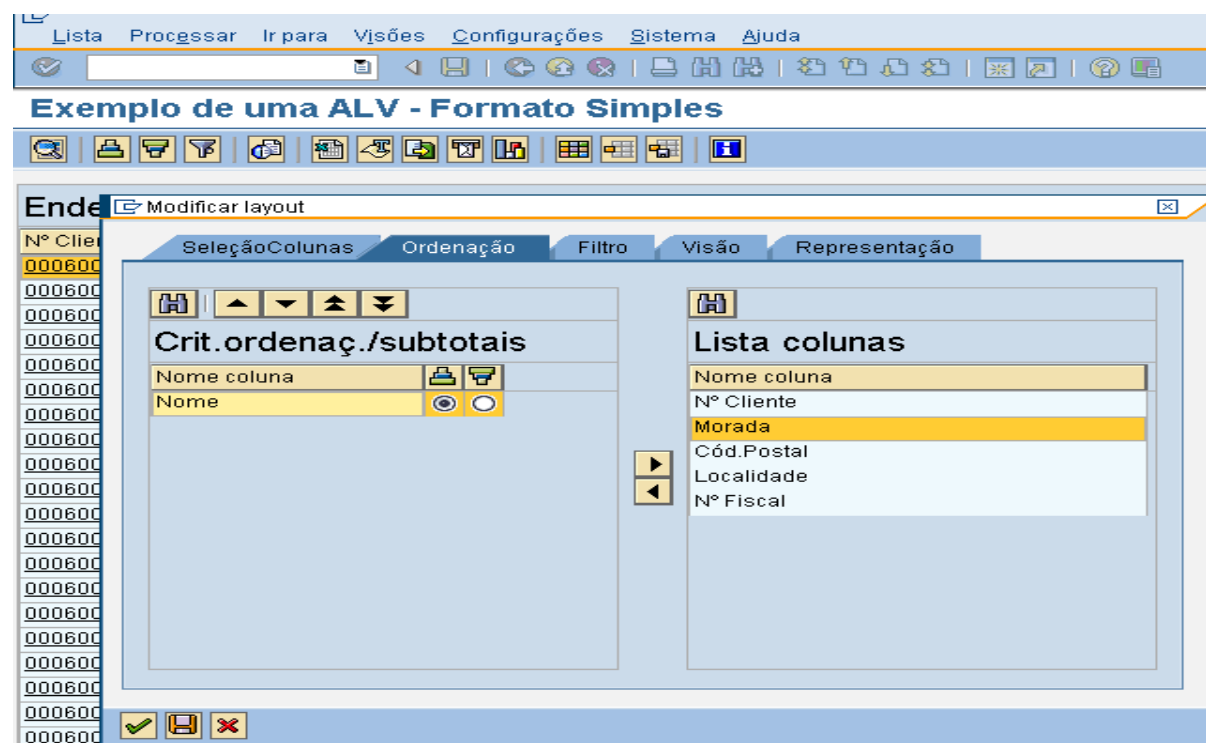
**Exemplo de uma ALV - Formato Simples**

**Endereços de Clientes**

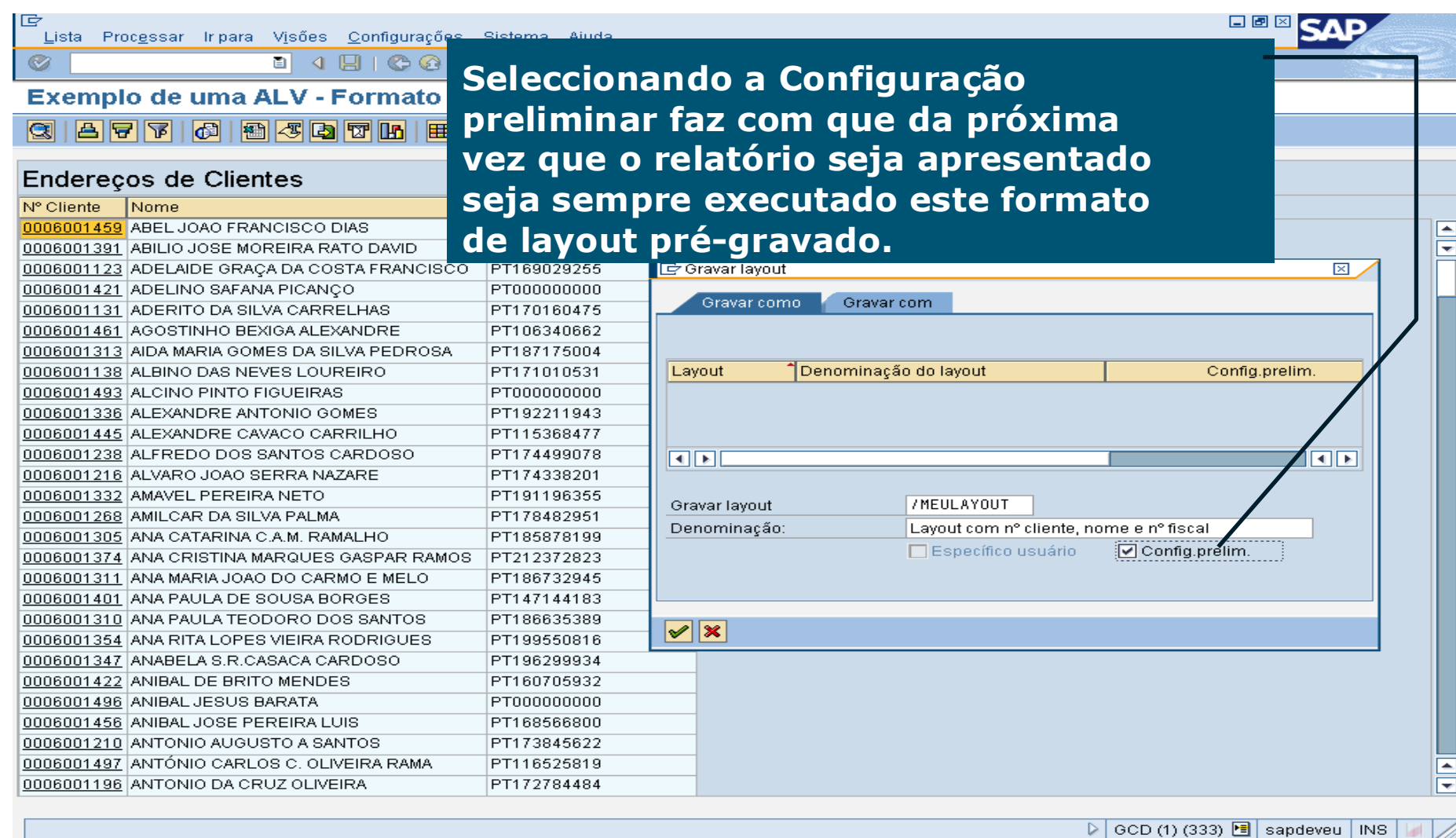
Nº Cliente	Nome	Morada	Cód.Postal	L
0006001123	ADELAIDE GRAÇA DA COSTA FRANCISCO		9999	E
0006001124	MANUEL JESUS DA SILVA		9999	E
0006001125	MARIA DA SILVA	" RUA DE BELOI, 783"	9999	F
0006001126	MARIA ALICE DE SOUSA MAIA	ED.17B PISO 0/A CID.NOVA	2670	L
0006001127	JOSE JOAQUIM FARIA CRUZ		9999	E



# SAP List Viewer



## Resultado



# SAP List Viewer

Lista Processar Ir para Visões Configurações Sistema Ajuda

**Exemplo de uma ALV - Formato Simples**

Endereços de Clientes

Nº Cliente	Nome	Nº Fiscal
0006001459	ABEL JOAO FRANCISCO D	
0006001391	ABILIO JOSE MOREIRA RA	
0006001123	ADELAIDE GRAÇA DA CO	
0006001421	ADELINO SAFANA PICANQ	
0006001131	ADERITO DA SILVA CARR	
0006001461	AGOSTINHO BEXIGA ALEX	
0006001313	AIDA MARIA GOMES DA SI	
0006001138	ALBINO DAS NEVES LOUF	
0006001493	ALCINO PINTO FIGUEIRAS	
0006001336	ALEXANDRE ANTONIO GO	
0006001445	ALEXANDRE CAVACO CA	
0006001238	ALFREDO DOS SANTOS C	
0006001216	ALVARO JOAO SERRA NA	
0006001332	AMAVEL PEREIRA NETO	PT191196355
0006001268	AMILCAR DA SILVA PALMA	PT178482951
0006001305	ANA CATARINA C.A.M. RAMALHO	PT185878199
0006001374	ANA CRISTINA MARQUES GASPAR RAMOS	PT212372823
0006001311	ANA MARIA JOAO DO CARMO E MELO	PT186732945

Dando um clique com o botão direito do rato em cima de uma coluna da ALV também permite aceder a algumas funcionalidades.



# SAP List Viewer

## Adicionar um cabeçalho à ALV:

Necessário indicar o parâmetro `I_CALLBACK_TOP_OF_PAGE` no módulo de função `REUSE_ALV_GRID_DISPLAY`.

A estrutura utilizada para o preenchimento de cabeçalhos na ALV é a `SLIS_T_LIST_HEADER`.


Necessário criar uma subrotina denominada `TOP_OF_PAGE` ou com o mesmo nome que a que irá proporcionar essa funcionalidade e se coloca a preencher o parâmetro `I_CALLBACK_TOP_OF_PAGE`.

Uso do Work Area `SLIS_LISTHEADER` para preenchimento da tabela interna que irá ser preenchida com os dados de cabeçalho.

Após preenchimento da tabela, transfere-se para a ALV com o módulo de função `REUSE_ALV_COMMENTARY_WRITE`.

# SAP List Viewer

Exemplo de um cabeçalho com Título ( tipo H ), Data do Dia ( Tipo S ) e número total de registos lidos ( Tipo A )



The screenshot shows the SAP List Viewer interface. At the top is a menu bar with options: Lista, Processar, Ir para, Visões, Configurações, Sistema, Ajuda. Below the menu bar is a toolbar with various icons. The main area displays a title 'Listagem de Clientes', a date 'Data: 20.09.2007', and a status 'Nº Total de registos seleccionados 355'. Below this is a table titled 'Endereços de Clientes' with columns 'Nº Cliente', 'Nome', and 'Nº Fiscal'.

Nº Cliente	Nome	Nº Fiscal
0006001459	ABEL JOAO FRANCISCO DIAS	PT122281632
0006001391	ABILIO JOSE MOREIRA RATO DAVID	PT000000000
0006001123	ADELAIDE GRAÇA DA COSTA FRANCISCO	PT169029255
0006001421	ADELINO SAFAIA RIGANCO	PT000000000

**Tipos:**

**H – Header**

**S – Selection**

**A – Action**

CALL FUNCTION 'REUSE\_ALV\_GRID\_DISPLAY'

EXPORTING

i_callback_program	= g_repid
it_fieldcat	= gt_fieldcat[]
i_callback_pf_status_set	= 'STANDARD_FULLSCREEN'
i_callback_user_command	= 'USER_COMMAND'
<i>i_callback_top_of_page</i>	<i>= 'TOP_OF_PAGE'</i>
is_layout	= gs_layout
i_grid_title	= w_lvc_title
i_save	= 'X'

TABLES

t_outtab	= gt_outtab.
----------	--------------

# SAP List Viewer

## Codificação adicional para impressão do TOP\_OF\_PAGE

FORM top\_of\_page.

### \* Declarações de Cabeçalho da ALV

```
DATA: t_header TYPE slis_t_listheader,
      wa_header TYPE slis_listheader,
      t_line LIKE wa_header-info,
      ld_lines TYPE i,
      ld_linesc(10) TYPE c.
```

### \* Título

```
wa_header-typ = 'H'.
wa_header-info = 'Listagem de Clientes'.
APPEND wa_header TO t_header.
CLEAR wa_header.
```

### \* Data

```
wa_header-typ = 'S'.
wa_header-key = 'Data: '.
CONCATENATE sy-datum+6(2) '.'
              sy-datum+4(2) '.'
              sy-datum(4) INTO wa_header-info. "data do sistema
APPEND wa_header TO t_header.
CLEAR: wa_header.
```

### \* Total de registos lidos

```
DESCRIBE TABLE gt_outtab LINES ld_lines.
ld_linesc = ld_lines.
CONCATENATE 'Nº Total de registos seleccionados ' ld_linesc
              INTO t_line SEPARATED BY space.
wa_header-typ = 'A'.
wa_header-info = t_line.
APPEND wa_header TO t_header.
CLEAR: wa_header, t_line.
```

```
CALL FUNCTION 'REUSE_ALV_COMMENTARY_WRITE'
EXPORTING
  it_list_commentary = t_header.
```

```
ENDFORM. "top_of_page
```

# SAP List Viewer

## Ordenação e Subtotais

O tipo através do qual se podem especificar previamente a ordenação das colunas e a determinação de subtotais é o SLIS\_SORTINFO\_ALV.

Para enviar as definições de ordenação e subtotais para o módulo de função de visualização da ALV é necessário preencher o parâmetro IT\_SORT.

Exemplo: numa listagem de encomendas, ordenar por nº de encomenda e por item, dando os subtotais quer pelo primeiro quer pelo segundo.

# SAP List Viewer

## Ordenação e Subtotais

```
* ALV data declarations
data: it_sortcat  type slis_sortinfo_alv occurs 1,
      wa_sort like line of it_sortcat.
```

```
perform build_sortcat.

*&-----*
*&      Form  build_sortcat
*&-----*
*      Build Sort catalog
*-----*
FORM build_sortcat .
  wa_sort-spos      = 1.
  wa_sort-fieldname = 'EBELN'.
  wa_sort-SUBTOT    = 'X'. "subtotals any totals column by this field
*  gd_sortcat-tabname
  APPEND wa_sort TO it_sortcat.

  wa_sort-spos      = 2.
  wa_sort-fieldname = 'EBELP'.
*  gd_sortcat-tabname
  APPEND wa_sort TO it_sortcat.
ENDFORM.                " build_sortcat
```

```
call function 'REUSE_ALV_GRID_DISPLAY'
  exporting
    i_callback_program      = gd_repid
    i_callback_top_of_page  = 'TOP-OF-PAGE'
    is_layout               = gd_layout
    it_fieldcat             = fieldcatalog[]
    it_sort                 = it_sortcat
    i_save                  = 'X'
  tables
    t_outtab               = it_ekko
  exceptions
    program_error          = 1
    others                  = 2.
```

# SAP List Viewer

```
REPORT Z_ALV_WITH_SORT_AND_SUBTOTAL_NO_OO.
DATA: lt_purchase_orders TYPE TABLE OF ty_purchase_order,
      lt_fieldcat        TYPE lvc_t_fcat,
      lv_subtotal        TYPE abap_bool.
```

```
*&-----*
*&      Form  BUILD_FIELDCATALOG
*&-----*
```

```
FORM build_fieldcatalog.
  CLEAR lt_fieldcat.
  ls_fieldcat-fieldname = 'EBELN'.
  ls_fieldcat-tabname = 'LT_PURCHASE_ORDERS'.
  ls_fieldcat-ref_field = 'EBELN'.
  APPEND ls_fieldcat TO lt_fieldcat.
```

```
  ls_fieldcat-fieldname = 'EBELP'.
  ls_fieldcat-tabname = 'LT_PURCHASE_ORDERS'.
  ls_fieldcat-ref_field = 'EBELP'.
  APPEND ls_fieldcat TO lt_fieldcat.
```

```
*&-----*
*&      Form  DISPLAY_ALV
*&-----*
```

```
FORM display_alv.
  CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
    EXPORTING
      it_fieldcat = lt_fieldcat[]
    TABLES
      t_outtab    = lt_purchase_orders.
```

```
IF sy-subrc <> 0.
  MESSAGE e003 WITH 'Error displaying ALV grid.'.
ENDIF.
ENDFORM.
```

Lista Processar Ir para Visões Configurações Sistema Ajuda

Header e Footer na ALV - Print Preview + Ordenações

Relatório da tabela EKKO

Data: 24.09.2007

Total de registos seleccionados: 10

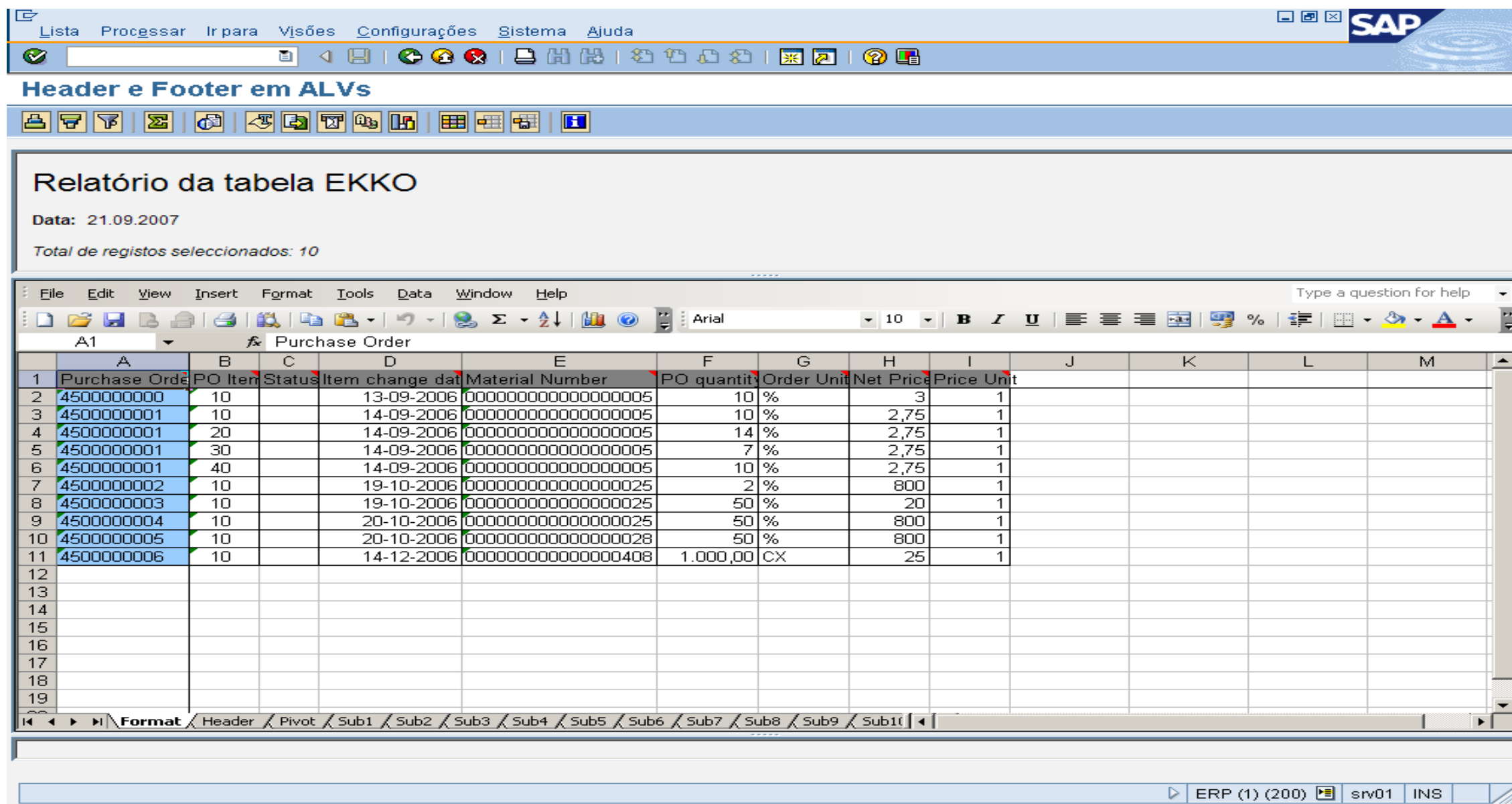
Nº Encomenda	Item	Status	Data modificação	Material	Quantidade	Unid. Medida	Preço	Preço Unit.
1010000000	10		2002.07.08	0000000000007000333	2,000	EA	792,00	1
	10						792,00	
1010000000							792,00	
1010000001	10		1999.02.18	0000000000007000336	1,000	EA	1.013,50	1
	10						1.013,50	
1010000001							1.013,50	
1010000002	10		1999.02.18	0000000000007000350	1,000	EA	402,00	1
	10						402,00	
1010000002							402,00	
1010000003	10		1999.02.18	0000000000007000323	1,000	EA	11,40	1
	10						11,40	
1010000003							11,40	
1010000004	10		2002.11.26	0000000000007000337	1,000	EA	1.293,95	10
	10						1.293,95	
1010000004							1.293,95	
1010000005	10		1999.02.18	0000000000007000350	1,000	EA	74,40	1
	10						74,40	
1010000005							74,40	
1010000006	10		1999.02.17	0000000000007000778	1.000,000	EA	0,80	1
	10						0,80	
1010000006							0,80	
1010000011	10		1999.05.06	0000000000000000072	34.000	BAQ	485,85	100

GCD (1) (333) sapdeveu INS



# SAP List Viewer

Para abrir o relatório automaticamente no Excel, pressionar o botão da barra de ferramentas com a etiqueta Microsoft Excel:



**Header e Footer em ALVs**

**Relatório da tabela EKKO**

Data: 21.09.2007

Total de registos seleccionados: 10

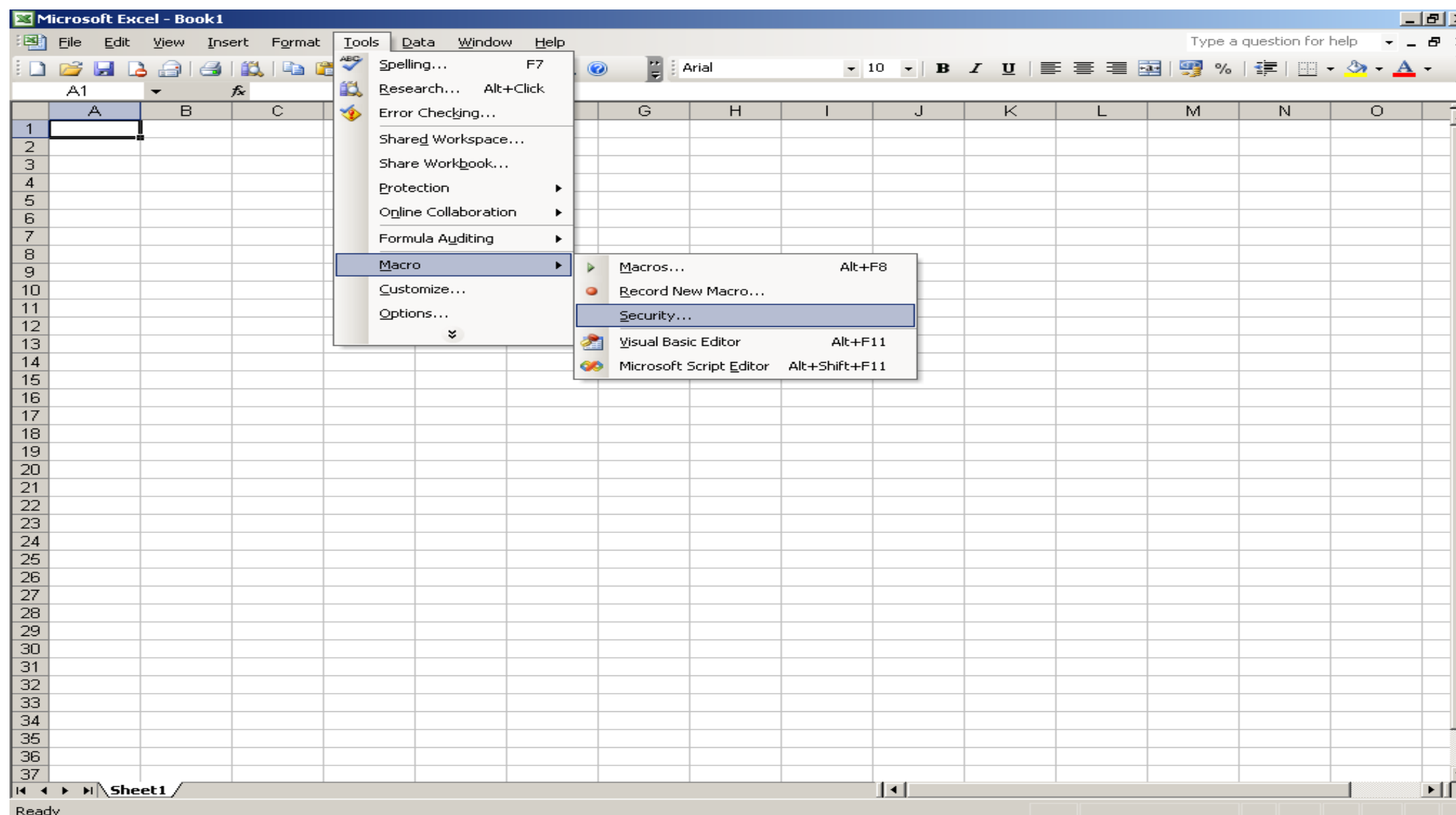
	A	B	C	D	E	F	G	H	I	J	K	L	M
	Purchase Order	PO Item	Status	Item change date	Material Number	PO quantity	Order Unit	Net Price	Price Unit				
1	4500000000	10		13-09-2006	00000000000000000005	10	%	3	1				
2	4500000001	10		14-09-2006	00000000000000000005	10	%	2,75	1				
3	4500000001	20		14-09-2006	00000000000000000005	14	%	2,75	1				
4	4500000001	30		14-09-2006	00000000000000000005	7	%	2,75	1				
5	4500000001	40		14-09-2006	00000000000000000005	10	%	2,75	1				
6	4500000002	10		19-10-2006	00000000000000000025	2	%	800	1				
7	4500000003	10		19-10-2006	00000000000000000025	50	%	20	1				
8	4500000004	10		20-10-2006	00000000000000000025	50	%	800	1				
9	4500000005	10		20-10-2006	00000000000000000028	50	%	800	1				
10	4500000006	10		14-12-2006	000000000000000000408	1.000,00	CX	25	1				

Format Header Pivot Sub1 Sub2 Sub3 Sub4 Sub5 Sub6 Sub7 Sub8 Sub9 Sub10

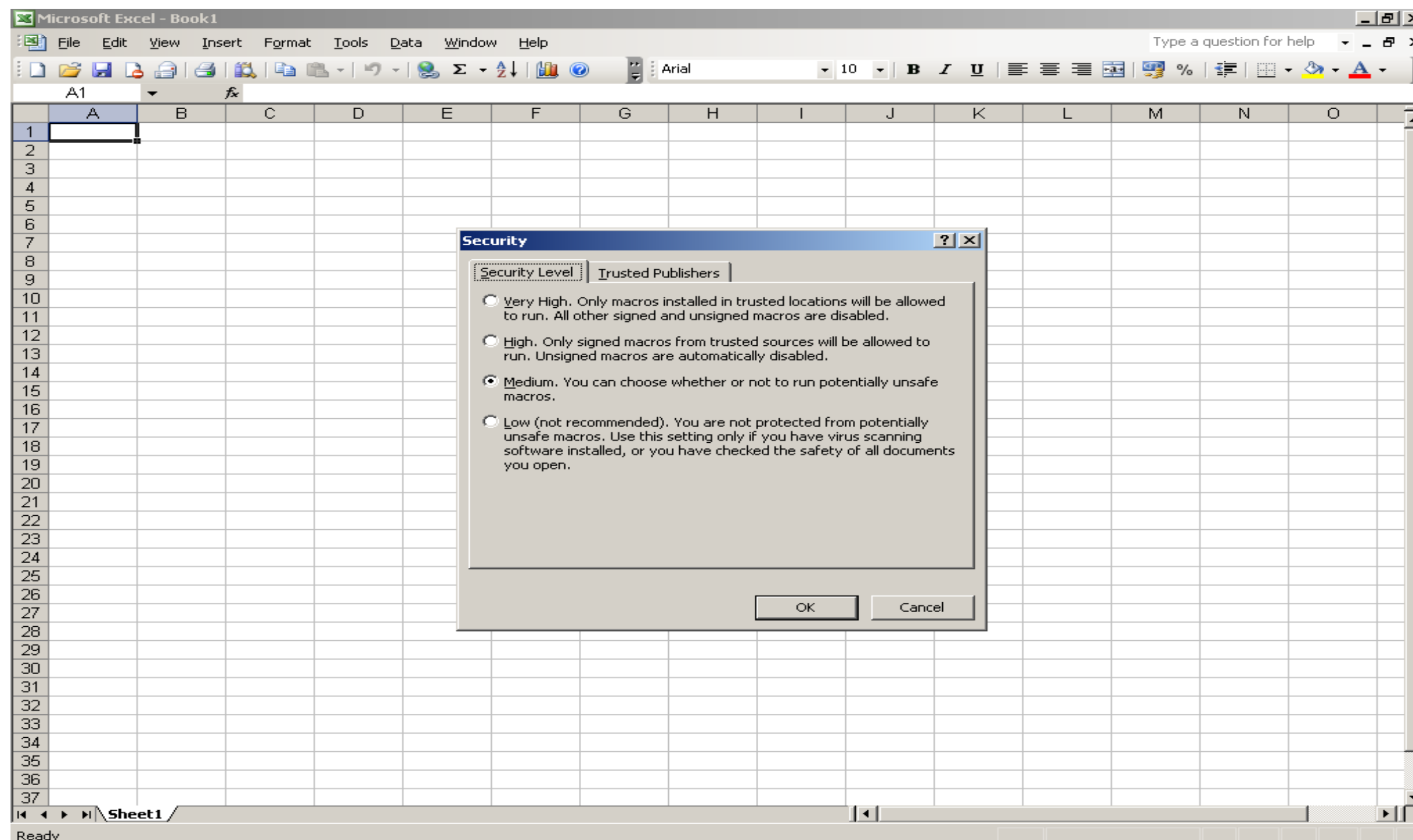
ERP (1) (200) srv01 INS

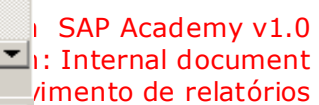
# SAP List Viewer

Para que o Excel possa receber os dados do SAP é necessário dar permissão:



# SAP List Viewer





# SAP List Viewer

Para tornar as colunas editáveis, basta indicá-lo no preenchimento do catálogo de campos:

```
fieldcatalog-fieldname = 'NETPR'.
fieldcatalog-seltext_m = 'Preço'.
fieldcatalog-col_pos = 7.
fieldcatalog-outputlen = 15.
fieldcatalog-datatype = 'CURR'.
fieldcatalog-do_sum = 'X'.

fieldcatalog-edit = 'X'.

APPEND fieldcatalog TO fieldcatalog.
CLEAR fieldcatalog.
```

# SAP List Viewer

## Seleccção múltipla:

- Necessário criar um campo de controlo de selecção dentro da definição da estrutura a ser visualizada pela ALV ( no exemplo, sel );
- Na definição do layout, marcar com 'X' o campo BOX\_FIELDNAME de modo a permitir a selecção da linha.

```
TYPES: BEGIN OF t_ekko,
    sel,
    ebeln  TYPE ekpo-ebeln,
    ebelp  TYPE ekpo-ebelp,
    statu  TYPE ekpo-statu,
    aedat  TYPE ekpo-aedat,
    matnr  TYPE ekpo-matnr,
    menge  TYPE ekpo-menge,
    meins  TYPE ekpo-meins,
    netpr  TYPE ekpo-netpr,
    peinh  TYPE ekpo-peinh,
END OF t_ekko.
```

*" Grava a linha seleccionada*

# SAP List Viewer

```
| *&-----*
*&      Form  BUILD_LAYOUT
*&-----*
*      Construção do Layout
*-----*
| FORM build_layout.
  gd_layout-box_fieldname    = 'SEL'.
  gd_layout-edit             = 'X'. "Torna toda a ALV editável
  gd_layout-zebra            = 'X'.
ENDFORM.                    " BUILD_LAYOUT
```

**Selecção múltipla**

Purchase ...	PO Item	Status	Item cha...	Material Nu...	PO quantity	Order Unit	Net Price	Price Uni
1010000000	10		2002.07....	0000000000...	2,000	EA	792,00	1
1010000001	10		1999.02....	0000000000...	1,000	EA	1.013,50	1
1010000002	10		1999.02....	0000000000...	1,000	EA	402,00	1
1010000003	10		1999.02....	0000000000...	1,000	EA	11,40	1
1010000004	10		2002.11....	0000000000...	1,000	EA	1.293,95	10
1010000005	10		1999.02....	0000000000...	1,000	EA	74,40	1
1010000006	10		1999.02....	0000000000...	1.000,000	EA	0,80	1
1010000011	10		1999.05....	0000000000...	34,000	BAG	485,85	100
1010000011	20		1999.05....	0000000000...	42,900	TO	97,17	1
1010000012	10		1999.05....	0000000000...	1.756,000	TO	77,59	1
							<b>4.248,66</b>	



# SAP List Viewer

## Exercício

Aproveitar o programa referente ao exercício da gravação das tabelas ZFTAB\_CAB\_01 e ZFTAB\_ITM\_01, mas desta vez recorrendo ao SAP List Viewer.

Após indicação do intervalo de documentos, serão visualizados os campos correspondentes da VBRK na ALV com a estrutura ZFTAB\_CAB\_01 que a irá suportar. Se o utilizador pressionar o botão &DATA\_SAVE, o programa deverá gravar quer a tabela de itens ZFTAB\_ITM\_01 quer a de cabeçalho ZFTAB\_CAB\_01.

A coluna FKDAT deverá ser editável para que caso seja modificada, ao pressionar &DATA\_SAVE é esta data que será gravada.

Ao dar 1 clique no campo VBELN, o utilizador saltará para a transacção VF03 mostrando as linhas do documento real, saltando o primeiro ecran.

# SAP List Viewer

## ALV Hierárquica (ou em árvore):

Lista Processar Ir para Configurações Sistema Ajuda

ALV Hierárquica ( em árvore )

Doc.vendas	Data doc.	TpDV	Valor líquido
Doc.vendas	Item	Material	CtgI Lote
20000001	21.06.2007	CO	6,00
* SUBTOTAL			6,00
20000000	21.06.2007	CO	60,00
* SUBTOTAL			60,00
55	03.01.2007	BV	0,00
57	03.01.2007	BV	0,00
84	05.01.2007	BV	0,00
* SUBTOTAL			0,00
12	22.12.2006	BV	4,50
* SUBTOTAL			4,50
24	29.12.2006	BV	30,00
26	29.12.2006	BV	30,00

# SAP List Viewer

ALV Hierárquica (ou em árvore):

- Utilizam-se as mesmas estruturas de fieldcatalog
- Recorre-se ao módulo de função REUSE\_ALV\_HIERSEQ\_LIST\_DISPLAY
- É utilizada uma nova estrutura do tipo SLIS\_KEYINFO\_ALV contendo as estruturas Header e Item

**Como exemplo suponhamos que pretendemos um report que liste as ordens de venda e respectivos itens em árvore**

# SAP List Viewer

```
REPORT zalv_teste

NO STANDARD PAGE HEADING LINE-COUNT 65(3) LINE-SIZE 220.

TYPE-POOLS: slis.
TABLES: vbak, "Cabeçalho de ordens de venda
        vbap. "Itens de ordens de venda
DATA: BEGIN OF it_vbak OCCURS 0,
        vbeln LIKE vbak-vbeln,
        audat LIKE vbak-audat,
        "N" Ordem
        "Data Documento
        auart LIKE vbak-auart, "Tipo de Ordem de Venda
        netwr LIKE vbak-netwr, "Valor Liquido
        expand TYPE c,
END OF it_vbak.
DATA: BEGIN OF it_vbap OCCURS 0,
        vbeln LIKE vbap-vbeln,
        posnr LIKE vbap-posnr,
        "N° Ordem "Item
        matnr LIKE vbap-matnr, "No Material
        pstyv LIKE vbap-pstyv, "Categoria de Item
        charg LIKE vbap-charg, "No Lote
END OF it_vbap.
```

# SAP List Viewer

```
DATA: it_header TYPE slis_tabname, it_item TYPE slis_tabname.
DATA: x_sort  TYPE slis_sortinfo_alv, it_sort
      TYPE slis_t_sortinfo_alv.
DATA: l_layout TYPE slis_layout_alv.
SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME
  TITLE TEXT-001.
  SELECT-OPTIONS: s_vbeln FOR vbak-vbeln,
    s_auart FOR vbak-auart.
SELECTION-SCREEN END OF BLOCK b1.

INITIALIZATION.
  it_header = 'IT_VBAK'. it_item = 'IT_VBAP'.
  CLEAR it_keyinfo.
  it_keyinfo-header01 = 'VBELN'.
  it_keyinfo-item01 = 'VBELN'.
  it_keyinfo-header02 = space.
  it_keyinfo-item02 = 'POSNR'.
```

# SAP List Viewer

START-OF-SELECTION.

*\* Carregar os dados nas tabelas internas*

PERFORM get\_data.

*\* Preencher o Fieldcatalog para IT\_VBAP e IT\_VBAK*

PERFORM fieldcatalog\_merge.

*\* Modificações ao layout da ALV*

PERFORM layout\_chg.

*\* Ordenar, obter subtotais e o total*

PERFORM sort\_func.

*\* Mostrar ALV Hierárquica ( em árvore )*

PERFORM final\_display.

\*&—

\*&

# SAP List Viewer

```
FORM get_data.
  SELECT vbeln
    audat
    auart
    netwr FROM vbak
  INTO TABLE it_vbak WHERE vbeln IN s_vbeln
  AND auart IN s_auart.
  SORT it_vbak BY vbeln.
  SELECT vbeln
    posnr
    matnr
    pstyv
    charg FROM vbap
  INTO TABLE it_vbap
  FOR ALL ENTRIES IN it_vbak
  WHERE vbeln = it_vbak-vbeln.
ENDFORM. "GET DATA
```

```
FORM fieldcatalog_merge.
  x_fieldcat_vbak-fieldname = 'NETWR'.
  x_fieldcat_vbak-tabname = 'IT_VBAK'.
  x_fieldcat_vbak-do_sum = 'X'.
  APPEND x_fieldcat_vbak TO it_fieldcat_vbak.
  CLEAR x_fieldcat_vbak.
  x_fieldcat_vbak-fieldname = 'AUART'.
  x_fieldcat_vbak-tabname = 'IT_VBAK'.
  APPEND x_fieldcat_vbak TO it_fieldcat_vbak.
  CLEAR x_fieldcat_vbak.
```

# SAP List Viewer

```
CALL FUNCTION 'REUSE_ALV_FIELDATALOG_MERGE'
  EXPORTING
    i_program_name      = sy-repid
    i_internal_tabname  = 'IT_VBAK'
    i_inclname          = sy-repid
  CHANGING
    ct_fieldcat         = it_fieldcat_vbak.

CALL FUNCTION 'REUSE_ALV_FIELDATALOG_MERGE'
  EXPORTING
    i_program_name      = sy-repid
    i_internal_tabname  = 'IT_VBAP'
    i_inclname          = sy-repid
  CHANGING
    ct_fieldcat         = it_fieldcat_vbak.
ENDFORM. "FIELDATALOG MERGE

FORM layout_chg.
*
  1_layout-zebra = 'X'.
  1_layout-subtotals_text = 'SUBTOTAL'.
  1_layout-totals_text = 'TOTAL'.
  1_layout-expand_fieldname = 'EXPAND'.

*1_layout-
expand_all *= 'X'. 77 Expandir tudo (ou encolher se vazio )
ENDFORM. "LAYOUT CHG
```



# SAP List Viewer

```

FORM sort_func.
  x_sort-spos = 1.
  x_sort-fieldname = 'AUART'.
  x_sort-tabname = 'IT_VBAK'.
  x_sort-up = 'X'.
*
*
  APPEND x_sort TO it_sort. CLEAR x_sort.
  x_sort-spos = 2.
  x_sort-fieldname = 'NETWR'.
  x_sort-tabname = 'IT_VBAK'.

  x_sort-up = 'X'.
  x_sort-subtot = 'X'.
  APPEND x_sort TO it_sort.
  CLEAR x_sort.
ENDFORM. "SORT FUNC
  
```

```

FORM final_display.
  CALL FUNCTION 'REUSE_ALV_HIERSEQ_LIST_DISPLAY'
    EXPORTING
      i_callback_program = sy-repid
      is_layout           = 1_layout
      it_fieldcat         = it_fieldcat_vbak
      it_sort             = it_sort
      i_tabname_header   = it_header
      i_tabname_item     = it_item
      is_keyinfo         = it_keyinfo
    TABLES
      t_outtab_header   = it_vbak
      t_outtab_item     = it_vbap
    EXCEPTIONS
      program_error     = 1
      OTHERS             = 2.
ENDFORM.
  
```

# SAP Programação por Objectos - ALVs

O novo modelo de objectos do SAP List Viewer consiste numa encapsulação orientada aos objectos para a ferramenta ALV que existe actualmente, permitindo criar:

- Tabelas simples ou bidimensionais
- Estruturas de sequências hierárquicas
- Estrutura em árvore

# SAP Programação por Objectos - ALVs

## Vantagens:

- Código unificado, orientado a objectos aplicável em todas as ALV's
- API coerente (Application Programming Interface)
- Permite a detecção de erros mais cedo durante a programação (o recurso às excepções permite saber antecipadamente se os métodos utilizados não são possíveis em determinadas situações).
- As funções que permitem o acesso já estão integradas na própria estrutura da ALV, pelo que não têm de ser codificadas, bastando serem invocadas quando necessárias.

# SAP Programação por Objectos - ALVs

**Na prática, vão ser utilizadas essencialmente três classes:**

- **CL\_SALV\_TABLE** – para listagens simples, bidimensionais
- **CL\_SALV\_HIERSEQ\_TABLE** – para sequências hierárquicas
- **CL\_SALV\_TREE** – para a estrutura em árvore

# SAP Programação por Objectos - ALVs

Para construir a ALV teremos que:

- Instanciar a classe principal da ALV:
  1. Definir a tabela interna que fornecerá a estrutura e conteúdo a ser visualizado pela ALV
  2. Definir o tipo de ALV que se pretende para a visualização ( normal, em árvore, etc.)

**Opcionalmente, poder-se-ão acrescentar funcionalidades, recorrendo aos métodos disponibilizados pelas respectivas classes.**

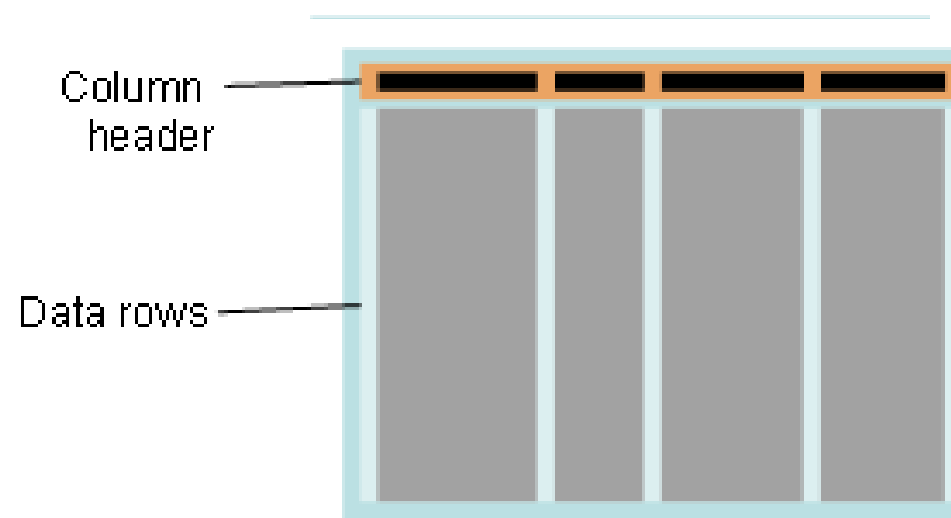
# SAP Programação por Objectos - ALVs

## Relativamente aos tipos de ALV temos:

### Tabelas simples, bi-dimensionais

Contêm um número de indeterminado de linhas, todas elas possuindo a mesma estrutura e sem relações hierárquicas entre si.

A estrutura é definida pelas colunas; cada coluna tem propriedades específicas que são automaticamente transferidas quando a ALV é visualizada.



# SAP Programação por Objectos - ALVs

## Lista sequencial hierárquica

- Consiste numa sequência de linhas divididas exactamente em 2 e só 2 níveis hierárquicos.
- Temos linhas de cabeçalho e de posição, estando as segundas dependentes das primeiras.
- Para cada linha de cabeçalho podem existir várias linhas de posição.



# SAP Programação por Objectos - ALVs

## Exibição em Árvore

- Contrariamente à lista sequencial hierárquica, esta pode apresentar um número indeterminado de níveis hierárquicos.
- As linhas da estrutura designam-se por nós. Cada nó tem a mesma estrutura hierárquica, independentemente do nível hierárquico a que pertença.
- Os nós podem ter relações ao mesmo nível ou terem uma relação pai-filho.





# SAP Programação por Objectos - ALVs

## Visualizar o output no formato ALV

Para visualizar o output no formato ALV mais simples basta recorrer a dois métodos:

### ✓ **FACTORY**

- Instancia-se a classe, define-se a tabela interna que vai servir de estrutura e que conterá os registos a serem visualizados
- Define-se o tipo de visualização que se pretende como output

### ✓ **DISPLAY**

- Invoca-se o método DISPLAY para exibir o conteúdo da tabela interna no ecrã

# SAP Programação por Objectos - ALVs

**Exemplo de programa para uma ALV simples, bi-dimensional:**

## Declaração de dados:

```
data: gt_outtab type table of sflight,  
      gr_table  type ref to cl_salv_table.
```

A tabela interna gt\_outtab conterá os dados a serem visualizados; gr\_table servirá para efectuar a referência à classe utilizada na construção da ALV.

## Extracção de dados:

```
select * from SFLIGHT up to 50 rows  
into corresponding fields of table gt_outtab.
```

Seleccionar os dados a serem visualizados para dentro da tabela interna gt\_outtab

# SAP Programação por Objectos - ALVs

**CALL METHOD cl\_salv\_table=>factory**

**IMPORTING**

**r\_salv\_table = gr\_table**

**CHANGING**

**t\_table = gt\_outtab.**

Instanciar à respectiva classe

**gr\_table->display( ).**

Mostrar o conteúdo no formato SAP List Viewer (ALV).

**(\*) Nas notas encontra-se um exemplo completo.**

# SAP Programação por Objectos - ALVs

## Exemplo de Output do programa exemplo contido nas notas:

Lista Processar Ir para Configurações Sistema Ajuda

Exemplo de ALV ( Factory ) número 1

Ma...	Empr.	Nome da firma	Local	País	Moeda	Id...	Plan...	D...	V...	Nº soci...	Endereço	Nº ID fiscal de IVA	Área...
2...	0001	SAP Iberoamerica S.A.	Amadora	PT	EUR	PT	CAPT	10	K4	2		PT171598610	0001
200	0MB1	IS-B Musterbank Deutschl.	Walldorf	DE	EUR	DE	0MB1		K4	1			
200	1000	Aurilis	France	FR	EUR	FR	CAFR	10	K4	2		FR14331048132	
200	2000	Carpan Supermercados	Portugal	PT	EUR	PT	CAPT	10	K4	2			
200	3000	Fundação D. Pedro IV	Portugal	PT	EUR	PT	CAPT	10	K4	2			
200	3001	Fundação D. Pedro IV	Portugal	PT	EUR	PT	CAPT	10	K4	2			
200	4000	Emp. NonTrade	Portugal	PT	EUR	PT	CAPT	10	K4	2			
200	7654	Tester	Portugal	PT	EUR	PT	CAPT	10	K4		23931		
200	9990	ROFF	ROFF	PT	EUR	PT	CAPT	10	K4	2			
200	9999	Direcção Nacional	Direcção Nacional	PT	EUR	PT	CAPT	10	K4	2		PT123456789	Z001
200	AM01	AXA Assurance Maroc	Casablanca	MA1	MAD	FR	CAFR	10	K4	2	22981		
200	AM02	SGS	Casablanca	MA	MAD	FR	CAFR	10	K4	2	22982		

# SAP Programação por Objectos - ALVs

## Programa exemplo de lista sequencial hierárquica:

Como cabeçalho vamos ter a tabela **ZNSTAB\_CAB\_01** que contém cabeçalho de documentos de facturação e a tabela **ZNSTAB\_ITM\_01** como posição, contendo o detalhe de cada factura.

As colunas serão transferidas para a aplicação através das tabelas internas GT\_PARENT e GT\_CHILD, respectivamente (a primeira para cabeçalho e a segunda para detalhe).

```
DATA: gt_parent TYPE TABLE OF znstab_cab_01,
      gt_child   TYPE TABLE OF znstab_itm_01,
      gr_table   TYPE REF TO cl_salv_hierseq_table,
      lt_binding TYPE salv_t_hierseq_binding,
      ls_binding TYPE salv_s_hierseq_binding.
```

**GR\_TABLE** contém a referência à classe através da qual se efectuará a lista sequencial hierárquica; **LT\_BINDING** servirá para definir o elo de ligação entre as duas hierarquias, ou seja, os campos que ligarão a tabela ZNSTAB\_CAB\_01 e **ZNSTAB\_ITM\_01**, ou seja, como ambas estão ligadas pelo campo Nº de Factura (VBELN), será este campo a servir de elo de ligação.

# SAP Programação por Objectos - ALVs

```
SELECT * FROM znstab_cab_01
  INTO TABLE gt_parent
 WHERE vbeln IN s_fact.

IF gt_parent[] IS NOT INITIAL.

  SELECT * FROM znstab_itm_01
    INTO TABLE gt_child
  FOR ALL ENTRIES IN gt_parent
  WHERE vbeln = gt_parent-vbeln.

ENDIF
```

Extracção de dados de acordo com os parâmetros indicados em S\_FACT. Se contiver cabeçalho então irá ler as respectivas entradas na tabela de itens.

# SAP Programação por Objectos - ALVs

```
ls_binding-master = 'VBELN'.
ls_binding-slave   = 'VBELN'.
APPEND ls_binding TO lt_binding.
```

A ligação MASTER/SLAVE que indica a dependência entre hierarquias, ou seja, a segunda está dependente da primeira, sendo a ligação efectuada através de um mesmo campo, o nº de factura ( VBELN ).

```
CALL METHOD cl_salv_hierseq_table=>factory
EXPORTING
    t_binding_level1_level2 = lt_binding
IMPORTING
    r_hierseq               = gr_table
CHANGING
    t_table_level1          = gt_parent
    t_table_level2          = gt_child.
```

```
gr_table->display( ).
```

Instanciar a classe e chamar o método para visualizar a ALV.

# SAP Programação por Objectos - ALVs

**Exemplo de output com o exemplo indicado:**

Lista Processar Ir para Configurações Sistema Ajuda

Exemplo de ALV ( Factory ) número 2

Man	Doc.fatur.	TpDo	C	Moeda	Valor líquido			
Man	Doc.fatur.	Item	Material	Denominação	Qtd.faturada	UW	Valor líquido	
200	900000000	F2	L EUR	8.625,00				
200	900000000	10	408	Disque de Frein	0,000	CX	0,00	
200	900000000	11	408	Disque de Frein	50,000	CX	3.750,00	
200	900000000	12	408	Disque de Frein	50,000	CX	3.750,00	
200	900000000	13	408	Disque de Frein	15,000	CX	1.125,00	
200	900000001	F2	L EUR	75,00				
200	900000001	10	408	Disque de Frein	1,000	CX	75,00	
200	900000002	F2	L EUR	75,00				
200	900000002	10	408	Disque de Frein	1,000	CX	75,00	



# SAP Programação por Objectos - ALVs

## Exemplo 1 de ALV em árvore (formato mais simples):

O programa deverá exibir a estrutura da tabela ZNSTAB\_CAB\_01 em formato árvore. Deste modo, cada registo será exibido ao expandir-se uma pasta a qual por sua vez visualizará o seu conteúdo.

```
DATA: gt_outtab1 TYPE TABLE OF znstab_cab_01,  
      gt_outtab2 TYPE TABLE OF znstab_cab_01,  
      ls_outtab  TYPE znstab_cab_01.
```

```
DATA: gr_tree TYPE REF TO cl_salv_tree,  
      nodes  TYPE REF TO cl_salv_nodes,  
      node   TYPE REF TO cl_salv_node,  
      key    TYPE salv_de_node_key.
```

GT\_OUTTAB1 conterá os registos a serem exibidos; GT\_OUTTAB2 irá servir apenas para definir a estrutura de visualização da árvore. GR\_TREE servirá para referenciar a respectiva classe.

# SAP Programação por Objectos - ALVs

Seleccção dos dados para a tabela interna GT\_OUTTAB1.

```
CALL METHOD cl_salv_tree=>factory
    IMPORTING
        r_salv_tree = gr_tree
    CHANGING
        t_table     = gt_outtab2.
```

Cria uma instância com a tabela vazia.

# SAP Programação por Objectos - ALVs

```

nodes = gr_tree->get_nodes( ).
LOOP AT gt_outtab1 INTO ls_outtab.
  TRY.
    node = nodes->add_node( related_node = key
    relationship = cl_gui_column_tree=>relat_first_child ).
    node->set_data_row( ls_outtab ).
    key = node->get_key( ).
  CATCH cx_salv_msg.
  ENDTRY.
ENDLOOP.

```

Adiciona nós à árvore. Por cada registo adiciona um novo nó. Nesta estrutura básica, cada novo nó será como filho do anterior, até ao final do carregamento de todos os nós.

```
gr_tree->display( ).
```

Exibe a ALV em árvore.

# SAP Programação por Objectos - ALVs

## Exemplo do Output após execução:

SAP						
	Mandante	Doc.fatur.	TpDocFat.	CtgDocFat	Moeda	Líquido
▼	200	90000000	F2	L	EUR	8.625,00
▼	200	90000001	F2	L	EUR	75,00
▼	200	90000002	F2	L	EUR	75,00
▼	200	90000003	F2	L	EUR	27,00
▼	200	90000004	F2	L	EUR	37,00
▼	200	90000005	F2	L	EUR	38,00
▼	200	90000006	F2	L	EUR	2.500,00
▼	200	90000007	F2	L	EUR	5,00
▼	200	90000008	F2	L	EUR	3.750,00
▼	200	90000009	F2	L	EUR	7.500,00
▼	200	90000010	F2	L	EUR	500,00
▼	200	90000011	F2	L	EUR	30,00
▼	200	90000012	F2	L	EUR	10,00
▶	200	90000013	S1	L	EUR	10,00

# SAP Programação por Objectos - ALVs

## Exemplo 2 de ALV em árvore (formato mais completo):

Pretende-se visualizar os códigos de voo para cada companhia gravados numa tabela de modo que por cada companhia ( nó pai ) sejam visualizados todos os voos existentes ( nó filho ).

DATA: lr\_columns TYPE REF TO cl\_salv\_columns\_tree.

LR\_COLUMNS irá referenciar a classe cl\_salv\_columns\_tree.

```
TRY.
  cl_salv_tree=>factory(
    IMPORTING
      r_salv_tree = gr_tree
    CHANGING
      t_table      = gt_outtab ).
CATCH cx_salv_no_new_data_allowed cx_salv_error.
  EXIT.
ENDTRY.
```

Definir a árvore invocando a classe factory.

# SAP Programação por Objectos - ALVs

DATA: settings TYPE REF TO cl\_salv\_tree\_settings.

```
settings = gr_tree->get_tree_settings( ).
settings->set_hierarchy_header( text-hd1 ).
settings->set_hierarchy_tooltip( text-ht1 ).
settings->set_hierarchy_size( 30 ).
```

DATA: title TYPE salv\_de\_tree\_text.

```
title = sy-title.
settings->set_header( title ).
```

Construção do cabeçalho estabelecendo as propriedades da árvore ( nome da hierarquia, título da árvore, etc. )

DATA: lt\_outtab LIKE gt\_outtab.

```
SELECT * FROM alv_t_t2 INTO CORRESPONDING FIELDS OF TABLE lt_outtab
      UP TO gs_test-amount ROWS.
```

Extracção dos registos de acordo com o número indicado em GS\_TEST-AMOUNT.

# SAP Programação por Objectos - ALVs

```
DATA: ls_data    TYPE alv_t_t2.
```

```
DATA: l_carrid_key TYPE lvc_nkey,  
      l_connid_key TYPE lvc_nkey,  
      l_last_key   TYPE lvc_nkey.
```

```
LOOP AT lt_outtab INTO ls_data.
```

```
  ON CHANGE OF ls_data-carrid.  
    PERFORM add_carrid_line USING  ls_data  
                                ""  
                                CHANGING l_carrid_key.
```

```
  ENDON.  
  ON CHANGE OF ls_data-connid.  
    PERFORM add_connid_line USING  ls_data  
                                l_carrid_key  
                                CHANGING l_connid_key.
```

```
  ENDON.  
  PERFORM add_complete_line USING ls_data  
                                l_connid_key  
                                CHANGING l_last_key.
```

```
ENDLOOP.
```

Preencher a ALV e construir a hierarquia, tendo em atenção a quebra de companhia aérea, e de número de voo, assim como o preenchimento do detalhe completo linha a linha.

# SAP Programação por Objectos - ALVs

```
FORM add_carrid_line USING p_ls_data TYPE alv_t_t2
                        p_key
                        CHANGING p_l_carrid_key.
```

```
DATA: nodes TYPE REF TO cl_salv_nodes,
      node  TYPE REF TO cl_salv_node,
      item  TYPE REF TO cl_salv_item.
```

```
*... §0 working with nodes
nodes = gr_tree->get_nodes( ).
```

```
TRY.
```

```
* ... §0.1 add a new node
```

```
* ... §0.3 set the data for the nes node
```

```
node = nodes->add_node( related_node = p_key
                        data_row     = p_ls_data
                        relationship = cl_gui_column_tree=>relat_last_child ).
```

```
p_l_carrid_key = node->get_key( ).
```

```
CATCH cx_salv_msg.
```

```
ENDTRY.
```

```
ENDFORM.                " add_carrid_line
```

Preenchimento de cada novo nó por quebra de companhia aérea.



# SAP Programação por Objectos - ALVs

**Exemplo do Output com ALV em árvore para este caso:**

Lista Processar Ir para Configurações Sistema Ajuda

Exemplo de ALV ( Factory ) número 4

Hierarquia	C.aérea	Nº voo	Data voo	Cia.aérea	C.aérea	Preço	Moeda	TpAvião	Capacid.	Ocupados	Total	País	Partida	País	Destino	D
▼	AA	17	11.12.2000	American Airlines	American Airlines	1.191,11	USD	146-200	112	40	53.361,73	US	JFK	US	SFO	
▼	AA	17	11.12.2000	American Airlines	American Airlines	1.191,11	USD	146-200	112	40	53.361,73	US	JFK	US	SFO	
	AA	17	11.12.2000	American Airlines	American Airlines	1.191,11	USD	146-200	112	40	53.361,73	US	JFK	US	SFO	
	AA	17	16.04.2001	American Airlines	American Airlines	9.506,17	USD	146-200	112	93	990.162,67	US	JFK	US	SFO	
	AA	17	20.08.2001	American Airlines	American Airlines	8.080,00	USD	146-200	112	1	9.049,60	US	JFK	US	SFO	
	AA	17	24.12.2001	American Airlines	American Airlines	98,77	USD	146-200	112	6	663,73	US	JFK	US	SFO	
▶	AA	26	09.12.2000	American Airlines	American Airlines	5.967,41	USD	146-300	128	77	514.629,44	DE	FRA	US	JFK	
▶	AA	64	07.12.2000	American Airlines	American Airlines	2.111,60	USD	737-200...	13	5	11.824,96	US	SFO	US	JFK	
▶	AB	17	05.12.2000	Air Berlin	Air Berlin	8.880,00	DEM	146-200	112	82	815.539,20	US	JFK	US	SFO	
▼	AC	17	29.11.2000	Air Canada	Air Canada	5.725,43	CAD	146-200	112	111	711.785,46	US	JFK	US	SFO	
▶	AC	17	29.11.2000	Air Canada	Air Canada	5.725,43	CAD	146-200	112	111	711.785,46	US	JFK	US	SFO	
▼	AC	26	27.11.2000	Air Canada	Air Canada	2.536,30	CAD	146-300	128	96	272.702,98	DE	FRA	US	JFK	
	AC	26	27.11.2000	Air Canada	Air Canada	2.536,30	CAD	146-300	128	96	272.702,98	DE	FRA	US	JFK	
	AC	26	02.04.2001	Air Canada	Air Canada	577,78	CAD	146-300	128	31	20.060,52	DE	FRA	US	JFK	

04

# Unicode



# Unicode - objetivos

No final deste módulo os formandos deverão estar aptos a descrever o que é o Unicode bem como a sua finalidade, sendo capazes de identificar as necessidades inerentes ao seu uso na linguagem ABAP.

# Unicode

Cada «code page» standard suporta apenas um grupo de linguagem restrito ( por exemplo, o Western European, o Japanese, etc.)

Dentro de um mesmo sistema só se consegue trabalhar de modo eficiente com um «code page».

Tornou-se, por conseguinte, necessária a criação de um «code page» especial o mais abrangente possível, capaz de abranger todos os símbolos, pontuação, sinais, letras, de todos os idiomas.

O Unicode é um superconjunto de todos os conjuntos de caracteres existentes. Define caracteres, não representações visuais, unificando os caracteres utilizado em diferentes scripts (União CJK\* - Chinese, Japanese, Korean).

# Unicode

## Codificações Unicode

### UTF-8

Esquema de codificação orientado ao byte; cada caracter é codificado com 1 a 4 bytes; compatível com o código ASCII de 7 bits.

### UTF-16

Esquema de codificação de 16 bits; cada caracter ocupa geralmente uma unidade de 16 bits; caracteres adicionais são codificados com 2 unidades de 16 bits.

### UTF-32

Unidades de 32 bits; tamanho fixo para todos os caracteres.

# Unicode

## Codificações Unicode / SAP

### UTF-8

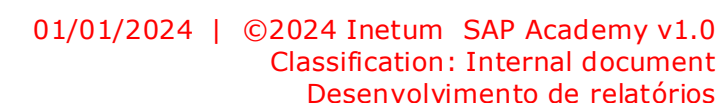
Utilizado para comunicação com o exterior ( ex. ficheiros, dados em rede ); ocupa o menor espaço em média mas tem uma limitação de compatibilidade com sistemas não-Unicode.

### UTF-16

- Melhor performance entre o uso de memória e a complexidade dos algoritmos;
- Compatível com ambientes Java e Microsoft
- A melhor forma de migrar programas Abap e C

# Unicode e ABAP

Todos os programas deverão ter a check-box "Verif.Unicode Activas com o visto de activação, de modo a que o interpretador Abap detecte a compatibilidade das sintaxes com as normas Unicode. Esta check-box é colocada automaticamente quando se cria um novo programa pelo que não deverá ser removida.



# Unicode

## Unicode e ABAP

Relativamente às versões de SAP R/3 anteriores, a introdução do Unicode significou alguma alteração no modo de programar. Em Unicode, a transferência de dados entre estruturas obriga não só a que sejam compatíveis em termos de tamanho como de características. Por exemplo, struct1 = struct2 deixou de ser permitido após a introdução do Unicode:

```
BEGIN OF struc1,  
  a(1) TYPE C,  
  x(1) TYPE X,  
END OF struc1.
```

```
BEGIN OF struc2,  
  a(1) TYPE C,  
  b(1) TYPE C,  
END OF struc2.
```



# Unicode

## Unicode e ABAP

```
BEGIN OF struc3,  
  a(2) TYPE C,  
  n(6) TYPE N,  
  i TYPE I,  
END OF struc3.
```

```
BEGIN OF struc4,  
  a(8) TYPE C,  
  i TYPE I,  
  f TYPE F,  
END OF struc4.
```

Struc3 = struc4 continua a ser permitido dado que as estruturas são compatíveis, ainda que no caso o campo f ficasse de fora mediante a atribuição.

Os campos a e n de struc3 guardariam o valor de a de struc4, o campo i de struc3 assumiria o homólogo i de struc4, não havendo espaço para o campo f em struc3.

No entanto, a atribuição era possível dado que existe uma compatibilidade entre os campos de ambas as estruturas, modo CHAR e NUMC, bem como Integer.

Para uma abordagem em maior profundidade, consultar o site da SAP:

[http://help.sap.com/saphelp\\_nw04/helpdata/en/79/c5546ab3dc11d5993800508b6b8b11/frameset.htm](http://help.sap.com/saphelp_nw04/helpdata/en/79/c5546ab3dc11d5993800508b6b8b11/frameset.htm)

# Unicode

## Unicode e ABAP

De igual modo, a introdução do Unicode veio provocar uma alteração nas instruções de Abap em diversas áreas, nomeadamente no tratamento de strings, ficheiros, etc.

Por exemplo, para abrir um ficheiro sequencial de texto passou a ser necessária uma instrução adicional de modo a suportar o Unicode:

**Antes do Unicode:**

**OPEN dataset FICH for OUTPUT IN TEXT MODE.**

**Depois do Unicode:**

**OPEN dataset FICH for OUTPUT IN TEXT MODE **ENCODING DEFAULT.****

# Unicode

## Unicode e ABAP

Para se atribuir toda uma estrutura a um único campo é necessário recorrer quer a técnicas diferentes das utilizadas antes do Unicode:

**REPORT ZCH\_UNIEXP\_8.**

**data: begin of STRUC,**

**F1(3) type x,**

**F2(8) type p,**

**end of STRUC,**

**CONTAINER(1000) type c.**

**\* Instrução inválida:**

**CONTAINER = STRUC. <---- Erro Unicode**

# Unicode

## Unicode e ABAP

Como alternativa poder-se-à recorrer à classe CL\_ABAP\_CONTAINER\_UTILITIES do seguinte modo:

```
class CL_ABAP_CONTAINER_UTILITIES definition load.
```

```
call method CL_ABAP_CONTAINER_UTILITIES =>FILL_CONTAINER_C  

exporting IM_VALUE = STRUC  

importing EX_CONTAINER = CONTAINER  

exceptions ILLEGAL_PARAMETER_TYPE = 1  

others = 2.
```

```
call method cl_abap_container_utilities=>read_container_c  

exporting IM_CONTAINER = CONTAINER  

importing EX_VALUE = STRUC  

exceptions ILLEGAL_PARAMETER_TYPE = 1  

others = 2.
```

Fill\_container\_c colocará a estrutura dentro da variável container, read\_container\_c permite ler uma string container com 1000 posições distribuindo-as pela estrutura STRUC.

# Unicode

## Unicode e ABAP

- As perguntas e respostas mais frequentes sobre Unicode podem ser vistas na nota de OSS 1322715 disponibilizada pela SAP para o devido efeito.
- Um programa ABAP adaptado ao Unicode (UP) é um programa no qual as verificações Unicode estão activas e produzirá os mesmos resultados se executado num sistema não Unicode.
- Para que os programas Abap sejam compatíveis dever-se-à recorrer à transacção SAMT para validar a sintaxe a fim de verificar se está de acordo com o Unicode.
- Como alternativa, pode executar-se o report RSUNISCAN\_FINAL para determinar as sintaxes relevantes para o Unicode ( tratamento de erros, etc. )

# Unicode

## Unicode e ABAP

### Verificar a compatibilidade Unicode num programa Abap

Programa Processar Ir para Sistema Ajuda

**Pesquisa em uma quantidade de programas**

Docum.ABAP + unicode Docum.UCCHECK

**Seleção de objeto**

Nome do objeto	ZALV_FACTORY01			
Tipo de objeto		até		→
Autor (TADIR)		até		→
Pacote		até		→
Sistema de origem		até		→

☐ Pesquisar só programas com flag unicode não ativado  
☒ Apenas objetos com entrada TADIR  
☒ Excluir pacotes \$\*  
☐ Exibir também objetos SAP modificados

**Restrição da quantidade de programas p/evitar timeout:**

Núm.máximo de programas

**Posições não analisáveis estaticamente**

☐ Exibir pontos não analisáveis de modo estático  
☐ Exibir também posições ocultas com "#EC"  
 Includes a exibir ☒ LSVIM\* até  →

**Verificações específicas aplicação**

☒ Atualização de visão  
☒ Módulos de função obsoletos UPLOAD/DOWNLOAD

Executar o programa RSUNISCAN\_FINAL e indicar o nome do(s) programa(s) cuja sintaxe se pretende avaliar relativamente à compatibilidade com o Unicode.

# Unicode

## Unicode e ABAP

### Verificar a compatibilidade Unicode

Se o programa Abap estiver de acordo com as normas Unicode, o resultado deverá ser um ecrã análogo ao representado em baixo.

**SAP**

Resultado da verificação sintaxe unicode

Exceção	UC	Orig.	Programa	Include	Linha	Código	Mensagem
	X	ERP	ZALV_FACTORY01	ZALV_FACTORY01	0	OK	Não foram encontrados erros de sintaxe unicode

# Unicode

## Unicode e ABAP

### Transacção SAMT

Permite processar um conjunto de programas em simultâneo, validando a compatibilidade Unicode e detectando anomalias diversas como esquecimento de instruções break-point nos programas, etc.)

Tarefa Processar Ir para Sistema Ajuda				
Processamento de conjunto de programas ABAP: 1ª tela				
Conj.programas				
Texto breve para a tarefa	Conj.programa	Programa	Subprograma	Última modificação
ABAP generation after transport	0	RSAMTIP	GENERATION_AFTER_IMPORT	NSEARA 30.06.20
Teste de migração dos programas 00	0	RSAMT00	00_MIGRATION_CHECK	NSEARA 30.06.20
Verif.ampliada programa (todos testes)	0	RSAMTSTD	EXTENDED_PROGRAM_CHECK	NSEARA 30.06.20
Verif.ampliada programa (testes import.)	0	RSAMTSTD	EXTENDED_PROGRAM_CHECK_GEN_POS	NSEARA 30.06.20
Geração de todas as telas	0	RSAMTSTD	GENERATE_DYNPROS	00.00.00
Geração de programa	1	RSAMTSTD	GENERATE_PROGRAM	00.00.00
Geração de progr.se necessário	0	RSAMTSTD	GENERATE_PROGRAM_IF_NECESSARY	00.00.00
Verificação de sintaxe	0	RSAMTSTD	SYNTAX_CHECK	00.00.00
Procurar BREAK-POINTS	0	RSAMTSTU	BREAK_POINTS	00.00.00
Título do programa	0	RSAMTSTU	PROGRAM_TITLE	00.00.00
Campos tipo D, D em expressão aritmética	0	SEARCH_D_T_IN_ARITHMETIC	SAMT_TEST	NSEARA 30.06.20
SQLN: eliminar todas as entradas	0	SQLSAMT	DELETE_ABAPS	NSEARA 30.06.20
SQLN: eliminar entradas antigas	0	SQLSAMT	DELETE_OLD_ABAPS	NSEARA 30.06.20
SQLN: Exec SQL Scan	0	SQLSAMT	SCAN_ABAPS	NSEARA 30.06.20



# Referências:

<http://www.sap.com>

<http://sdn.sap.com>

<http://help.sap.com>

<http://www.sapdevelopment.co.uk>

<http://www.erpgenie.com>

<http://www.sapdb.info>





inetumcom

FRANCE | SPAIN | PORTUGAL | BELGIUM | SWITZERLAND | LUXEMBOURG | ENGLAND |  
POLAND | ROMANIA | MOROCCO | TUNISIA | SENEGAL | CÔTE D'IVOIRE | ANGOLA |  
CAMEROON | USA | BRAZIL | COLOMBIA | MEXICO | RP OF PANAMA | PERU | CHILE |  
COSTA RICA | DOMINICAN REPUBLIC | ARGENTINA | SINGAPORE | UAE

