

Rojo: sintaxis antigua u obsoleta

Verde: equivalente actual

Válido en: ECC 7.40 SP08+, S/4HANA, BTP ABAP Environment (salvo lo marcado)

Antes y ahora	
ANTES	AHORA
DATA lv TYPE i. lv = lines(lt).	DATA(lv) = lines(lt).
MOVE a TO b.	b = a.
ADD 1 TO lv_n.	lv_n += 1. (7.54+)
CREATE OBJECT lo.	lo = NEW #().
CALL METHOD lo->run EXPORTING iv = 1.	lo->run(iv = 1).
REFRESH lt.	CLEAR lt.
DESCRIBE TABLE lt LINES lv.	lv = lines(lt).
READ TABLE lt INTO ls WITH KEY id = 1.	ls = lt[id = 1].
READ TABLE ... TRANSPORTING NO FIELDS. IF sy-subrc = 0.	IF line_exists(lt[id = 1]).
CONCATENATE a b INTO c SEPARATED BY space.	c = { a } { b } .
MOVE-CORRESPONDING a TO b.	b = CORRESPONDING #(a).
RANGES r FOR mara-matnr.	DATA r TYPE RANGE OF matnr.
DATA lt TYPE TABLE OF x WITH HEADER LINE.	DATA lt TYPE STANDARD TABLE OF x WITH EMPTY KEY.
FORM / PERFORM	CLASS lcl ... METHODS
CONVERSION_EXIT_ALPHA_INPUT	{ lv ALPHA = IN }

```
DATA(lv_name) = `Pedido`.           " string
DATA(lv_count) = 0.                 " i
LOOP AT lt INTO DATA(ls).          " estructura
LOOP AT lt ASSIGNING FIELD-SYMBOL(<ls>). " field-symbol
LOOP AT lt REFERENCE INTO DATA(lr). " referencia
SELECT ... INTO TABLE @DATA(lt_res). " tabla de
resultados
CATCH cx_root INTO DATA(lx).        " excepción
```

FINAL() (7.57+) declara una variable inmutable: FINAL(lv_max) = 100.

Expresiones de constructor

```
DATA(ls_addr) = VALUE ty_addr( city = 'Sevilla' zip =
'41001' ).
DATA(lt_ids) = VALUE int4_table( ( 1 ) ( 2 ) ( 3 ) ).
lt_range = VALUE #( sign = 'I' option = 'EQ'
( low = 'A' ) ( low = 'B' ) ).
lt_all = VALUE #( BASE lt_all ( ls_new ) ). " añade sin
perder
```

NEW, CONV, REF, CAST, EXACT

```
DATA(lo) = NEW zcl_app( iv_mode = 1 ).
DATA(lr) = NEW ty_struct( id = 1 ).      " referencia
a dato
lv_dec = CONV decfloat34( lv_text ).
DATA(lo_d) = CAST zif_doc( lo_obj ).
lv_p = EXACT #( lv_float ).             " error si
se pierde precisión
```

COND y SWITCH

```
DATA(lv_txt) = COND string( WHEN lv_q > 100 THEN `Alto`
                             WHEN lv_q > 0 THEN `Bajo`
                             ELSE `Sin stock` ).
DATA(lv_day) = SWITCH string( lv_wd WHEN 1 THEN `Lunes`
                                WHEN 2 THEN `Martes`
                                ELSE THROW zcx_invalid(
) ).
```

CORRESPONDING

```
ls_out = CORRESPONDING #( ls_in ).
ls_out = CORRESPONDING #( BASE ( ls_out ) ls_in ). "
conserva
ls_out = CORRESPONDING #( ls_in MAPPING customer = kunnr
                           EXCEPT ernam ).
lt_out = CORRESPONDING #( lt_in DISCARDING DUPLICATES ).
```

LET: variables auxiliares

```
DATA(lv_full) = VALUE string( LET f = ls-first l = ls-last
                               IN |{ l }, { f }| ).
```

Tablas internas y cadenas

Tipos de tabla

TIPO	ACCESO POR CLAVE	CUÁNDO
STANDARD	Lineal (O(n))	Recorrer, añadir, pocas búsquedas
SORTED	Binario (O(log n))	Búsquedas y LOOP WHERE por la clave
HASHED	Constante (O(1))	Búsquedas por clave única completa

```
TYPES tt_mat TYPE SORTED TABLE OF ty_mat
  WITH UNIQUE KEY matnr
  WITH NON-UNIQUE SORTED KEY by_type COMPONENTS mtart.

LOOP AT lt_mat INTO DATA(ls) USING KEY by_type
  WHERE mtart = 'FERT'.
```

Leer y comprobar

```
" Excepción CX_SY_ITAB_LINE_NOT_FOUND si no existe:
ls = lt[ matnr = lv ].
" Valor inicial si no existe:
ls = VALUE #( lt[ matnr = lv ] OPTIONAL ).
ls = VALUE #( lt[ matnr = lv ] DEFAULT ls_def ).
lv = lt[ 1 ]-matnr. " por índice
lv = lt[ KEY by_type mtart = 'FERT' ]-matnr.
IF line_exists( lt[ matnr = lv ] ).
DATA(lv_idx) = line_index( lt[ matnr = lv ] ).
```

FOR: construir una tabla desde otra

```
DATA(lt_ids) = VALUE tt_ids( FOR ls IN lt_orders
  WHERE ( status = '0' )
  ( ls-order_id ) ).

DATA(lt_out) = VALUE tt_out(
  FOR ls IN lt_in INDEX INTO i
  ( pos = i name = to_upper( ls-name ) ) ).

" Anidado: posiciones de todas las cabeceras
DATA(lt_flat) = VALUE tt_flat(
  FOR h IN lt_head
  FOR p IN h-items
  ( vbeln = h-vbeln posnr = p-posnr ) ).
```

REDUCE: acumular

```
DATA(lv_sum) = REDUCE netwr( INIT s = CONV netwr( 0 )
  FOR ls IN lt_pos
  NEXT s = s + ls-netwr ).

DATA(lv_csv) = REDUCE string(
  INIT t = ``
  FOR ls IN lt_mat
  NEXT t = |{ t }{ ls-matnr };| ).
```

FILTER

```
" Requiere clave SORTED o HASHED sobre los campos del WHERE
DATA(lt_fert) = FILTER #( lt_mat USING KEY by_type
  WHERE mtart = CONV #( 'FERT' ) ).

" Con tabla de filtro
DATA(lt_sel) = FILTER #( lt_mat IN lt_keys
  WHERE matnr = matnr ).
DATA(lt_rest) = FILTER #( lt_mat EXCEPT IN lt_keys
  WHERE matnr = matnr ).
```

Agrupar

```
LOOP AT lt_pos INTO DATA(ls_pos)
  GROUP BY ( vbeln = ls_pos-vbeln n = GROUP SIZE )
  ASCENDING INTO DATA(ls_grp).
DATA(lv_tot) = REDUCE netwr( INIT s = CONV netwr( 0 )
  FOR m IN GROUP ls_grp
  NEXT s = s + m-netwr ).

ENDLOOP.
```

Operaciones habituales

```
SORT lt BY matnr werks DESCENDING.
" Tras SORT por los mismos campos:
DELETE ADJACENT DUPLICATES FROM lt COMPARING matnr.
DELETE lt WHERE status = 'X'.
INSERT LINES OF lt_new INTO TABLE lt.
APPEND INITIAL LINE TO lt ASSIGNING FIELD-SYMBOL(<ls_new>).
MODIFY lt FROM ls TRANSPORTING status WHERE vbeln = lv.
```

String templates y funciones

Pedido { lv_ebeln ALPHA = OUT } { lv_date DATE = USER } { lv_time TIME = USER } { lv_amount NUMBER = USER DECIMALS = 2 } { lv_text WIDTH = 10 ALIGN = RIGHT PAD = '0' } { lv_ts TIMESTAMP = ISO } Línea 1\nLínea 2\tcon tabulador	
condense(val = s)	Quita espacios sobrantes
to_upper(s) / to_lower()	Mayúsculas / minúsculas
substring(val = s off = 2 len = 3)	Subcadena
replace(val = s sub = 'a' with = 'b' occ = 0)	Reemplazar todas
find(val = s sub = 'x')	Posición (-1 si no)
matches(val = s pcre = '^d+\$')	Expresión regular
segment(val = s index = 2 sep = ';')	Trozo n de un CSV
strlen(s), numofchar(s)	Longitud
concat_lines_of(table = lt sep = `, `)	Unir líneas
shift_left(val = s sub = '0')	Quitar ceros a la izquierda

SELECT con sintaxis estricta

```
SELECT FROM ekko AS h
  INNER JOIN ekpo AS p ON p~ebeln = h~ebeln
  FIELDS h~ebeln, h~lifnr, p~ebel, p~matnr, p~netwr
  WHERE h~bukrs IN @s_bukrs
    AND p~loekz = @abap_false
  ORDER BY h~ebeln, p~ebel
  INTO TABLE @DATA(lt_po)
  UP TO 500 ROWS.
```

Comas entre campos + @ en variables ABAP = sintaxis estricta.
Necesaria para expresiones, declaración en línea y ABAP Cloud.

Una fila, existencia y recuento

```
SELECT SINGLE FROM t001 FIELDS butxt
  WHERE bukrs = @lv_bukrs INTO @DATA(lv_name).

SELECT SINGLE @abap_true FROM mara
  WHERE matnr = @lv_matnr INTO @DATA(lv_exists).

SELECT COUNT(*) FROM vbak WHERE kunnr = @lv_kunnr
  INTO @DATA(lv_count).
```

Expresiones en el SELECT

```
SELECT FROM vbak
  FIELDS vbeln,
    CASE WHEN netwr > 10000 THEN 'A' ELSE 'B' END AS
clase,
    CAST( netwr AS DEC( 15, 2 ) ) AS importe,
    concat_with_space( vbeln, kunnr, 1 ) AS clave,
    coalesce( vkorg, '0000' ) AS org,
    dats_days_between( erdat, @lv_hoy ) AS antigüedad
  INTO TABLE @DATA(lt).
```

Agregados

```
SELECT FROM vbap
  FIELDS matnr, SUM( kwmeng ) AS cantidad, COUNT(*) AS
líneas
  WHERE erdat >= @lv_desde
  GROUP BY matnr
  HAVING SUM( kwmeng ) > 100
  INTO TABLE @DATA(lt_tot).
```

FOR ALL ENTRIES

```
IF lt_head IS NOT INITIAL.      " si está vacía, lee
  TODO
  SELECT FROM ekpo FIELDS ebeln, ebelp, matnr
    FOR ALL ENTRIES IN @lt_head
    WHERE ebeln = @lt_head-ebeln
    INTO TABLE @DATA(lt_items).
ENDIF.
```

Subconsulta y tabla interna como fuente

```
SELECT FROM kna1 FIELDS kunnr, name1
  WHERE kunnr IN ( SELECT kunnr FROM vbak WHERE erdat >=
@lv_desde )
  INTO TABLE @DATA(lt_cli).

SELECT FROM @lt_mat AS m FIELDS matnr      " 7.52+
  WHERE mtart = 'FERT' INTO TABLE @DATA(lt_fert).
```

CTE (WITH)

```
WITH +ventas AS ( SELECT FROM vbak FIELDS kunnr, SUM( netwr
) AS total
                  GROUP BY kunnr )
SELECT FROM +ventas FIELDS kunnr, total
  WHERE total > 50000
  INTO TABLE @DATA(lt_top).
```

Escritura

```
INSERT ztable FROM @ls_row.
INSERT ztable FROM TABLE @lt_rows ACCEPTING DUPLICATE KEYS.
UPDATE ztable SET status = 'C' WHERE id = @lv_id.
MODIFY ztable FROM TABLE @lt_rows.
DELETE FROM ztable WHERE created_at < @lv_limite.
```

CDS: vista de interfaz

```
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Pedido de venta'
define view entity ZI_SalesOrder
  as select from vbak
    association [0..*] to ZI_SalesOrderItem as _Item
      on $projection.SalesOrder = _Item.SalesOrder
    association [1..1] to I_Customer as _Customer
      on $projection.Customer = _Customer.Customer
  {
    key vbeln as SalesOrder,
      kunnr as Customer,
      @Semantics.amount.currencyCode: 'Currency'
      netwr as NetAmount,
      waerk as Currency,
      erdat as CreatedOn,
      _Item,
      _Customer
  }
}
```

Parámetros, sesión y control de acceso

```
define view entity ZI_OrdersFrom
  with parameters p_from : abap.dats
  as select from vbak
  { key vbeln, erdat }
  where erdat >= $parameters.p_from
    and ernam = $session.user

@MappingRole: true
define role ZI_SalesOrder_DCL {
  grant select on ZI_SalesOrder
    where ( SalesOrganization ) = aspect pfeg_auth(
V_VBAK_VKO, VKORG, ACTVT = '03' );
}
```

Behavior definition (gestionado con draft)

```
managed implementation in class zbp_r_order unique;
strict ( 2 );
with draft;
define behavior for ZR_Order alias Order
persistent table zorder
draft table zorder_d
lock master total etag LastChangedAt
authorization master ( instance )
etag master LocalLastChangedAt
{
  create; update; delete;
  field ( readonly, numbering : managed ) OrderUUID;
  field ( mandatory ) Customer;
  action ( features : instance ) approve result [1] $self;
  determination setStatus on modify { create; }
  validation checkCustomer on save
    { create; field Customer; }
  draft action Edit; draft action Activate optimized;
  draft action Discard; draft action Resume;
  draft determine action Prepare
    { validation checkCustomer; }
}
```

EML: leer

```
READ ENTITIES OF zr_order IN LOCAL MODE
ENTITY Order
  FIELDS ( Status Customer ) WITH CORRESPONDING #( keys )
  RESULT DATA(lt_orders)
  FAILED DATA(ls_failed).

READ ENTITIES OF zr_order
ENTITY Order BY \_Item ALL FIELDS
WITH VALUE #( ( %tky = ls_order-%tky ) )
RESULT DATA(lt_items).
```

EML: crear, modificar, borrar, ejecutar

```
MODIFY ENTITIES OF zr_order
ENTITY Order
  CREATE FIELDS ( Customer Description )
  WITH VALUE #( ( %cid = '01'
    Customer = '1000'
    Description = `Nuevo` ) )
  CREATE BY \_Item FIELDS ( Material Quantity )
  WITH VALUE #( ( %cid_ref = '01'
    %target = VALUE #( ( %cid = 'I1'
      Material = 'M-01'
      Quantity = 5 ) ) )
) )
  MAPPED DATA(ls_mapped)
  FAILED DATA(ls_failed)
  REPORTED DATA(ls_reported).

MODIFY ENTITIES OF zr_order ENTITY Order
  UPDATE FIELDS ( Status )
  WITH VALUE #( ( %tky = ls-%tky Status = 'A' ) ).

MODIFY ENTITIES OF zr_order ENTITY Order
  DELETE FROM VALUE #( ( %tky = ls-%tky ) ).

MODIFY ENTITIES OF zr_order ENTITY Order
  EXECUTE approve FROM VALUE #( ( %tky = ls-%tky ) ).

" Fuera del behavior pool:
COMMIT ENTITIES RESPONSE OF zr_order
  FAILED DATA(ls_cf) REPORTED DATA(ls_cr).
```

Componentes %

%tky	Clave de transacción (clave + %is_draft)
%cid / %cid_ref	Id temporal en creación / referencia a él
%param	Parámetros de acción o su resultado
%control	Qué campos se informan en un UPDATE
%msg, %element	Mensaje y campo erróneo en REPORTED
%state_area	Grupo de mensajes que se limpian juntos
%is_draft	Instancia borrador (mk-on) o activa

Mensajes en validaciones

```
APPEND VALUE #( %tky = ls-%tky ) TO failed-order.
APPEND VALUE #(
  %tky = ls-%tky
  %state_area = 'CHECK_CUSTOMER'
  %element-Customer = if_abap_behv=>mk-on
  %msg = new_message(
    id = 'ZORDER'
    number = '001'
    v1 = ls-Customer
    severity = if_abap_behv_message=>severity-error
  ) )
  TO reported-order.
```

ABAP Cloud: qué usar en su lugar

CLÁSICO	ABAP CLOUD
sy-datum / sy-uzeit	cl_abap_context_info=>get_system_date() / _time()
sy-uname	cl_abap_context_info=>get_user_technical_name()
MARA, VBAK, EKKO	I_Product, I_SalesDocument, I_PurchaseOrderAPI01
BAL_LOG_CREATE	cl_bali_log
SO_NEW_DOCUMENT_ATT_SEND_API1	cl_bcs_mail_message
NUMBER_GET_NEXT	cl_numberrange_runtime
JOB_OPEN / SUBMIT	Application Jobs (cl_apj_rt_api)
cl_http_client	cl_web_http_client_manager
ALV, WRITE	Fiori Elements, if_oo_adt_classrun
GUI_UPLOAD, OPEN DATASET	RAP large objects, xco_cp_xlsx
BAPI de creación	EML sobre BO liberado (I_*TP)

Estado de liberación de un objeto: ADT, pestaña Properties > API State, o el Cloudification Repository (github.com/SAP/abap-atc-cr-cv-s4hc). Variante de ATC: ABAP_CLOUD_READINESS.