Constructor operators. Corresponding & friends

Overview and best practices

Marcos del Puerto García, SAP
Month 12, 2021

PUBLIC

THE BEST RUN SAP

Follow us

www.sap.com/contactsap

THE BEST RUN SAP

SAP folgen auf

www.sap.com/germany/contactsap

THE BEST RUN SAP

Welcome!

Marcos del Puerto García
marcos.del.puerto.garcia@sap.com

Follow: Horst Keller
https://people.sap.com/horst.keller

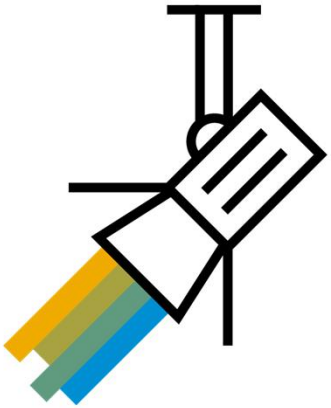
Agenda

§ Good ol’ MOVE-CORRESPONDING overview

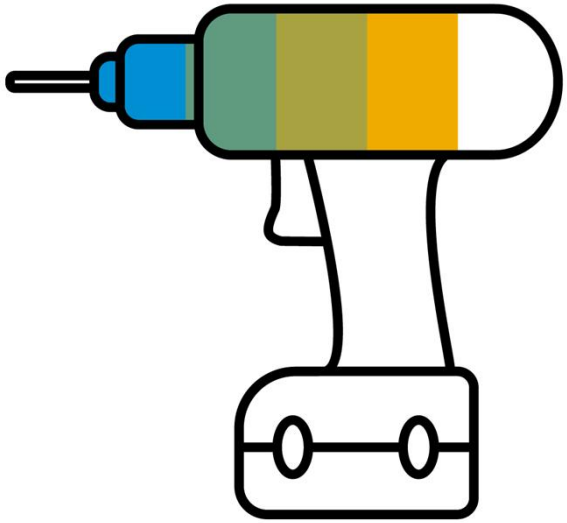
§ Corresponding

§ CL_ABAP_CORRESPONDING

§ FILTER



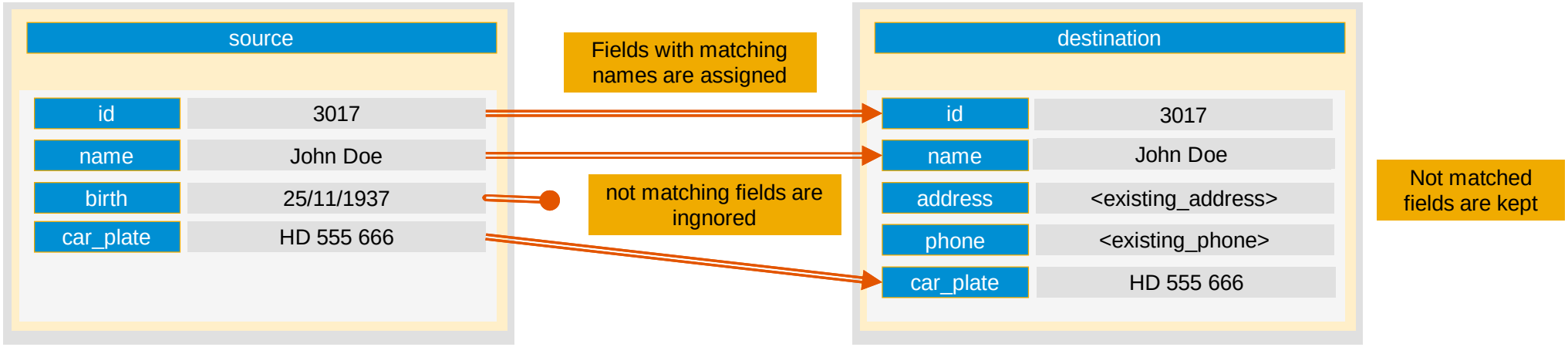
Move-corresponding



Good ol' MOVE-CORRESPONDING

Syntax

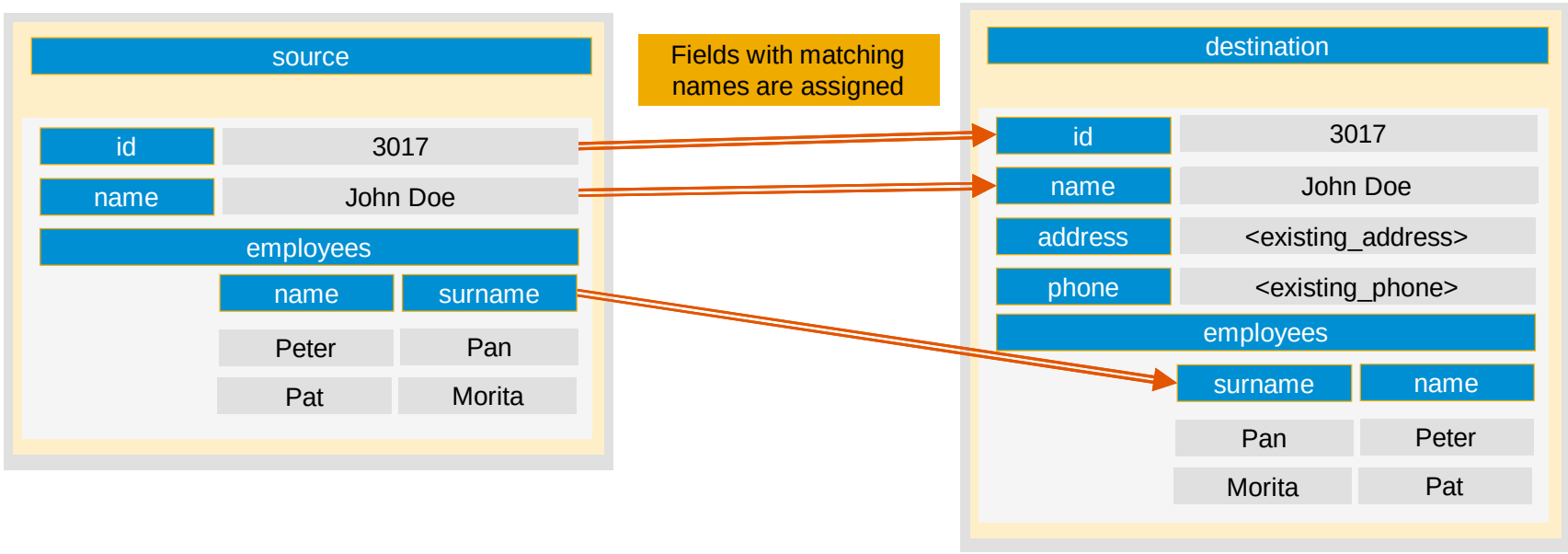
```
MOVE-CORRESPONDING source_struct TO destination_struct.
MOVE-CORRESPONDING EXACT source_struct TO destination_struct.
```



New Good ol' MOVE-CORRESPONDING

Expanding nested tables (since 7.42)

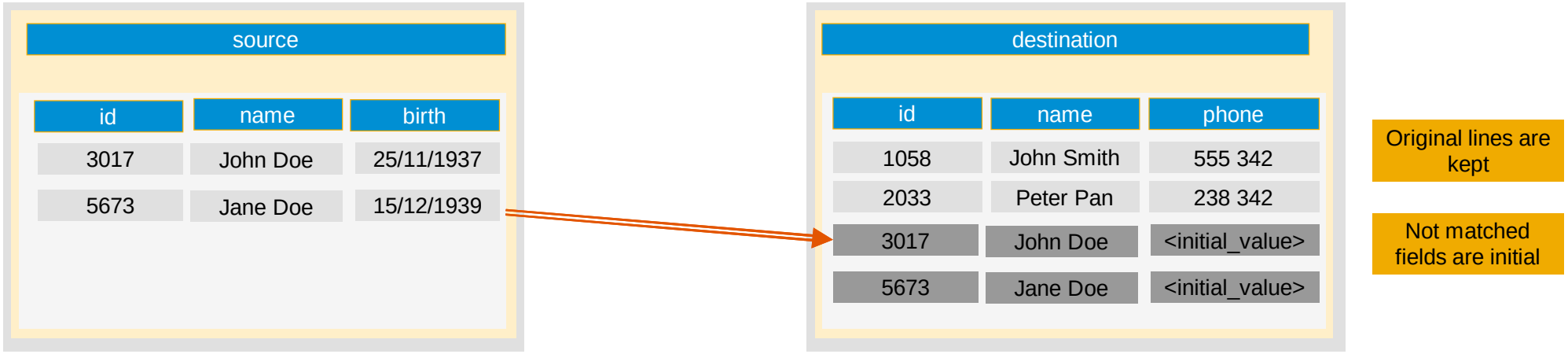
```
MOVE-CORRESPONDING source_table TO destination_table EXPANDING NESTED TABLES.
MOVE-CORRESPONDING source_struct TO destination _struct EXPANDING NESTED TABLES.
```



New Good ol' MOVE-CORRESPONDING

Keeping target lines (since 7.42)

```
MOVE-CORRESPONDING source_table TO destination_table KEEPING TARGET LINES.
```



New Good ol' MOVE-CORRESPONDING

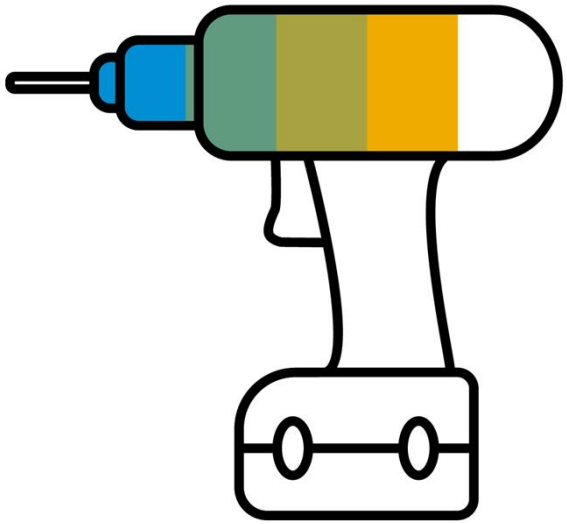
Keeping target lines (since 7.83)

```
MOVE-CORRESPONDING source_struct TO destination_struct KEEPING TARGET LINES.
```

```
MODIFY ENTITIES OF entity … FAILED DATA(ls_failed).

APPEND LINES OF ls_failed-constraint TO failed-constraint.
APPEND LINES OF ls_failed-constraintproduct TO failed-constraintproduct.
APPEND LINES OF ls_failed-periods TO failed-periods.
APPEND LINES OF ls_failed-text TO failed-text.
```

```
MOVE-CORRESPONDING ls_failed TO failed KEEPING TARGET LINES.
```

Corresponding



New operators design considerations

- Write less code to solve common coding patterns
- Offer enhanced type safety
- Avoid endless loops
- Optimized for big amounts of data
- Avoid hidden performance traps

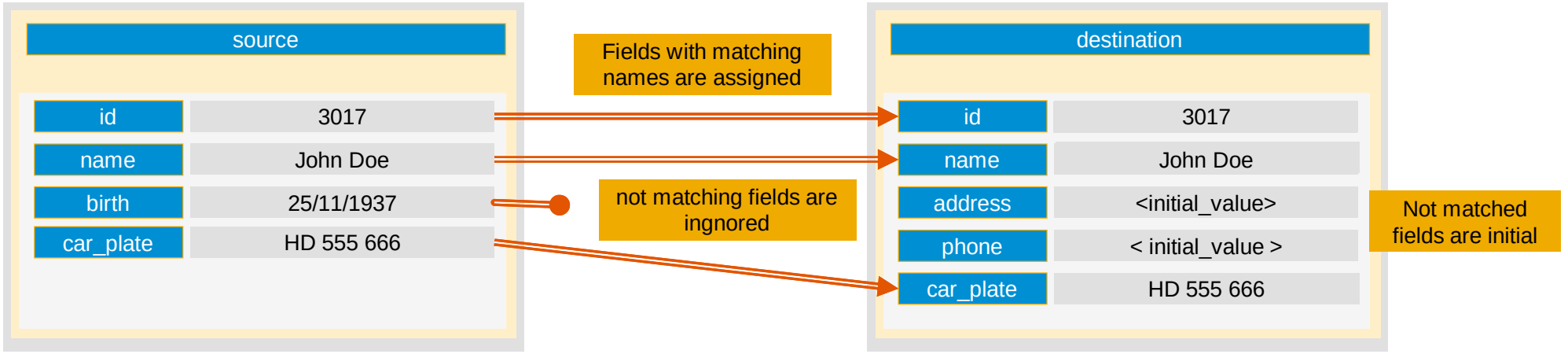
New operators general properties

- They are constructor operators, not just operators
- Block oriented operations à Stable, atomic

Basic CORRESPONDING

Syntax (since 7.42 exact since 7.54/7.76)

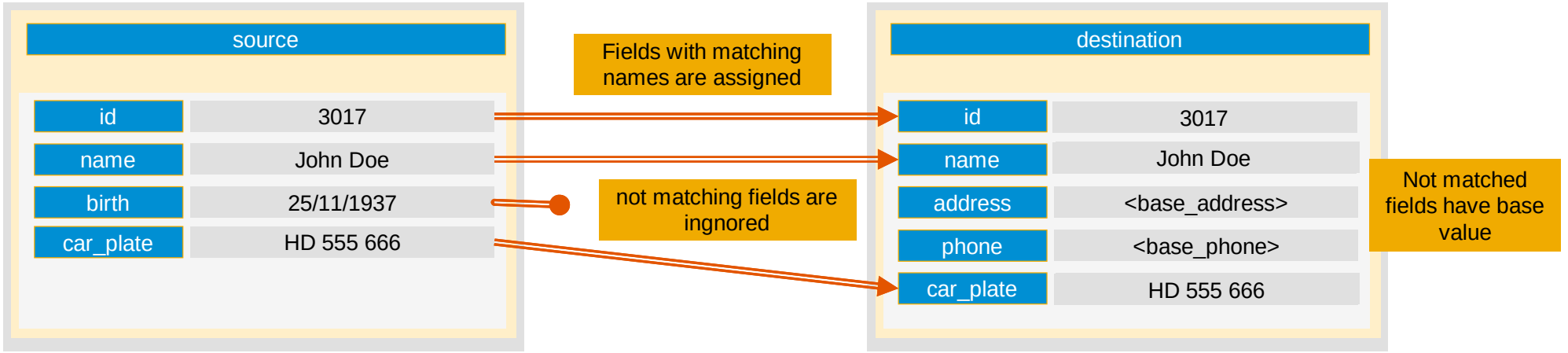
```
destination =  CORRESPONDING #( source ).
destination =  CORRESPONDING #( EXACT source ).
```



CORRESPONDING

Base clause

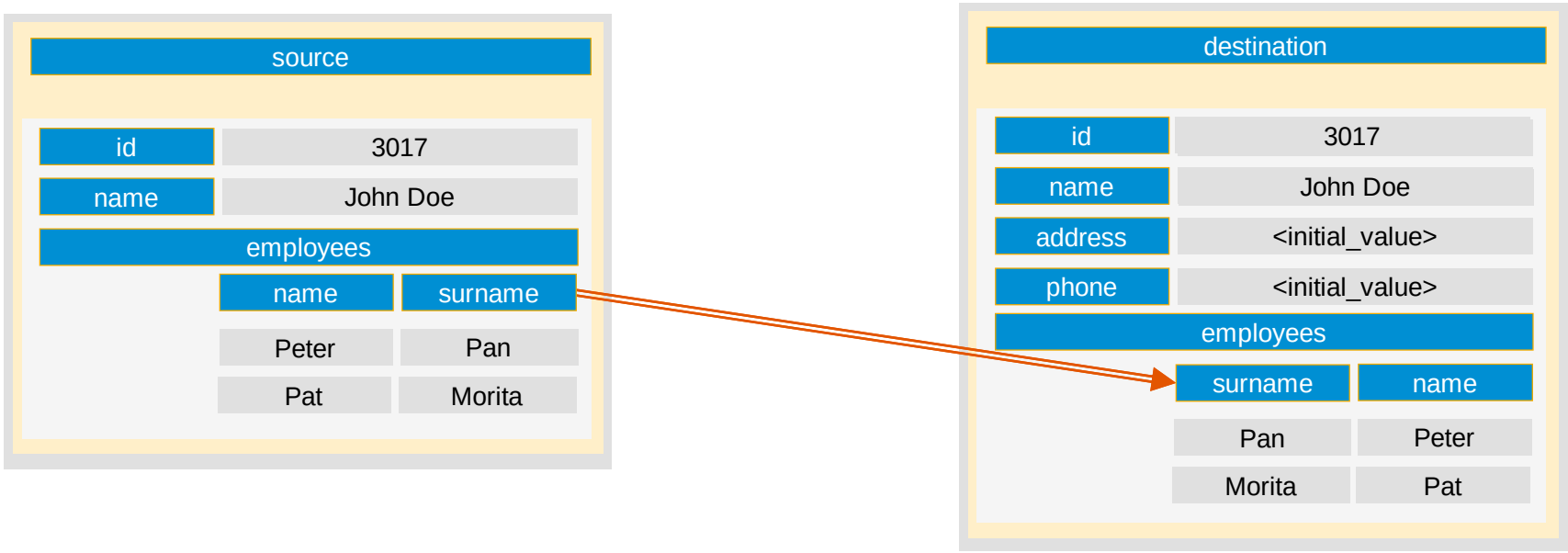
```
destination =  CORRESPONDING #( BASE ( destination ) source ).
```



CORRESPONDING

Deep clause

```
destination =  CORRESPONDING #( DEEP source ).
```



CORRESPONDING

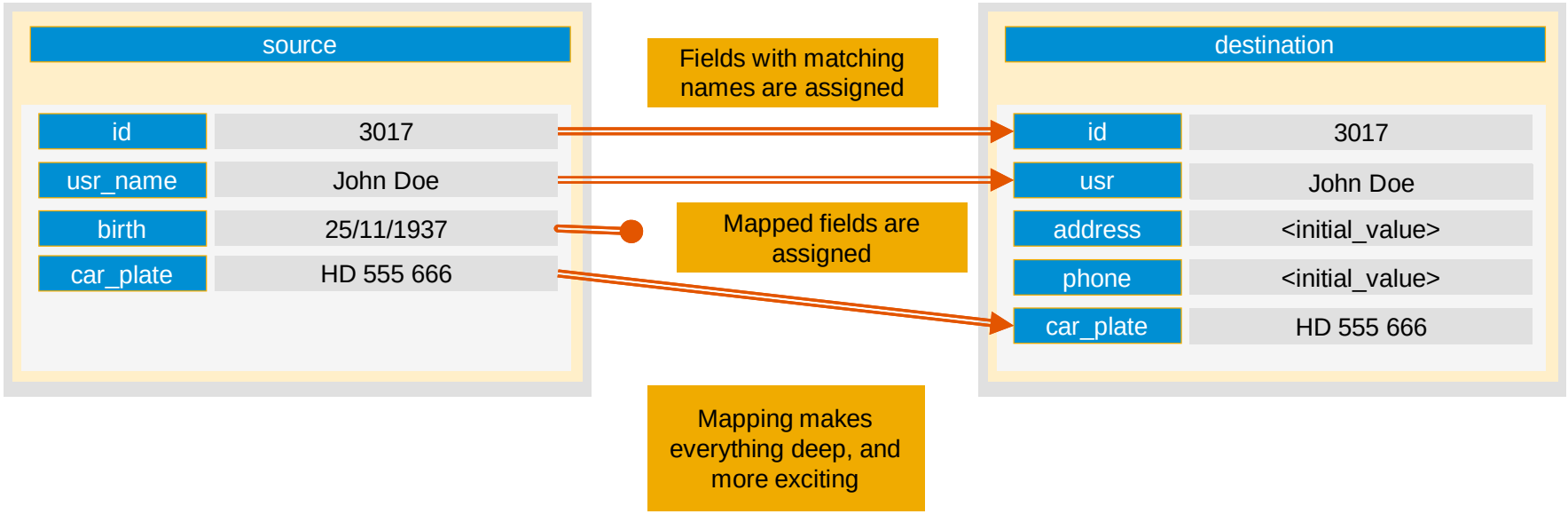
Deep + Base clause

```
destination =  CORRESPONDING #( DEEP BASE ( destination ) source ).
```

CORRESPONDING

Mapping

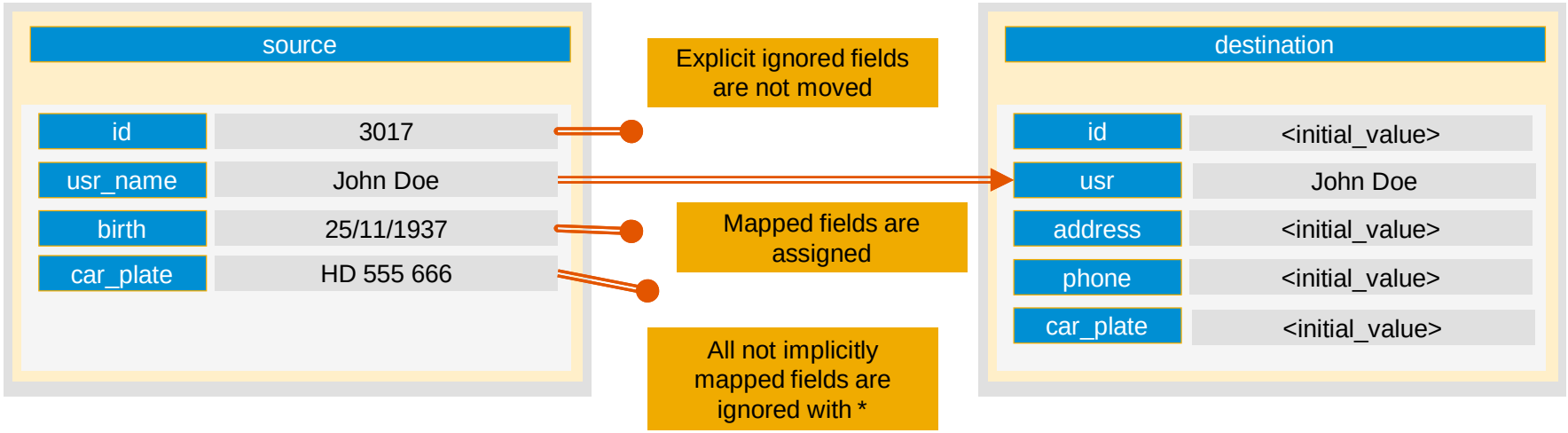
```
destination =  CORRESPONDING #( source MAPPING usr = user_name ).
```



CORRESPONDING

Mapping + Except

```
  destination =  CORRESPONDING #( source MAPPING usr = user_name EXCEPT id ).
destination =  CORRESPONDING #( source MAPPING usr = user_name EXCEPT id code ).
   destination =  CORRESPONDING #( source MAPPING usr = user_name EXCEPT * ).
```



CORRESPONDING

Complex map

```
destination =  CORRESPONDING #( source MAPPING usr = user_name
                                                (nested = nested MAPPING
                                                          origin      = cityfrom
                                                          destination = cityto
                                                          EXCEPT  price )
                                                 EXCEPT * ).
```

Operator CORRESPONDING

Mapping Example

```
FIELD-SYMBOLS <src_line> LIKE LINE OF src_tab.
DATA dst_line LIKE LINE OF dst_tab.


LOOP AT src_itab ASSIGNING <src_line>.
  CLEAR dst_line.
  MOVE-CORRESPONDING <src_line> TO dst_line.
  dst_line-dst_comp = <src_line>-src_comp.
  CLEAR dst_line-ignored.
  INSERT dst_line INTO TABLE dst_itab.
ENDLOOP.
```

```
dst_tab = CORRESPONDING #( src_tab
                           MAPPING dst_comp = src_comp
                           EXCEPT  ignored ).
```

CORRESPONDING

Appending (since 7.83)

```
destination =  CORRESPONDING #( APPENDING ( destination ) source ).
```

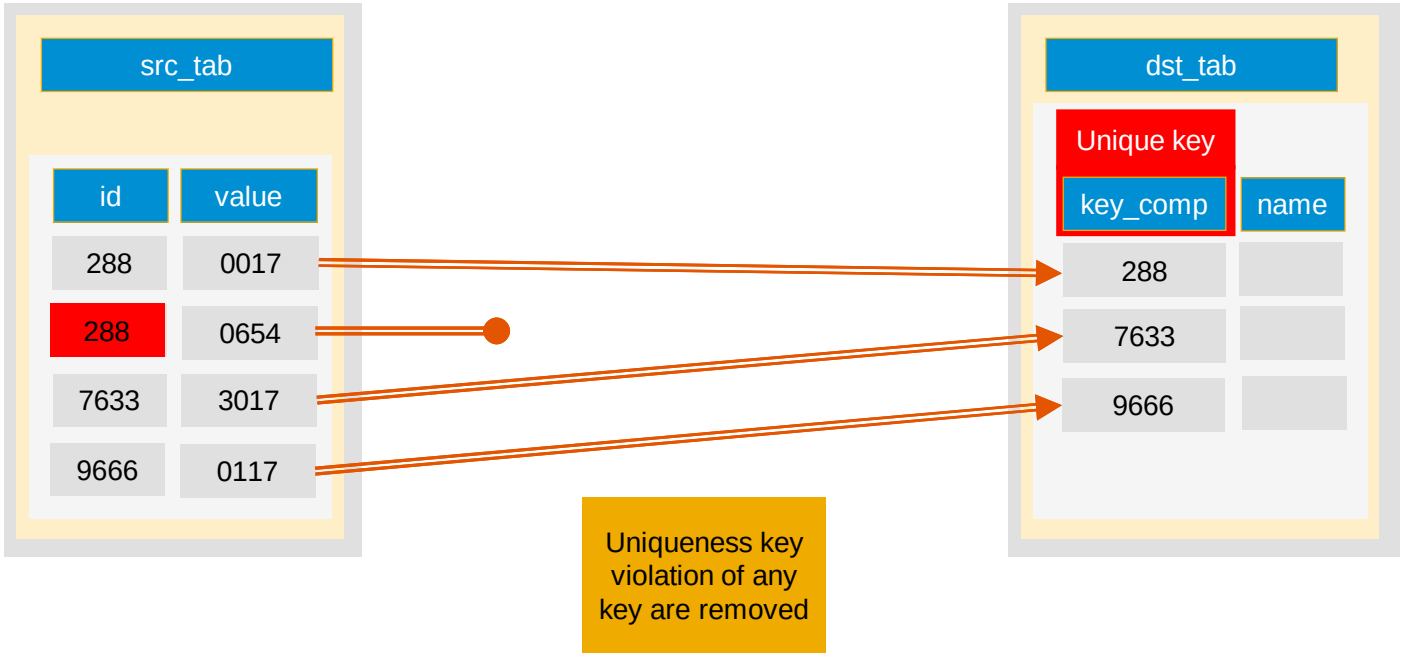
```
MODIFY ENTITIES OF entity … FAILED DATA(ls_failed).

APPEND LINES OF ls_failed-constraint TO failed-constraint.
APPEND LINES OF ls_failed-constraintproduct TO failed-constraintproduct.
APPEND LINES OF ls_failed-periods TO failed-periods.
APPEND LINES OF ls_failed-text TO failed-text.
```

```
failed  = CORRESPONDING #( APPENDING [BASE] ( failed ) ls_failed ).
failed  = CORRESPONDING #( DEEP APPENDING [BASE] ( failed ) ls_failed ).
```

CORRESPONDING

Discarding duplicates (since 7.81)

```
destination =  CORRESPONDING #( source DISCARDING DUPLICATES ).
```



Operator CORRESPONDING

Equivalences

```
MOVE-CORRESPONDING struct1 TO struct2.
```

```
struct2 = CORRESPONDING #( BASE ( struct2 ) struct1 ).
```

```
MOVE-CORRESPONDING struct1 TO struct2 EXPANDING NESTED TABLES.
```

```
struct2 = CORRESPONDING #( DEEP BASE ( struct2 ) struct1 ).
```

```
MOVE-CORRESPONDING table1 TO table2 KEEPING TARGET LINES.
```

```
table2 = CORRESPONDING #( BASE (table2 ) table1 ).
```

```
MOVE-CORRESPONDING struct1 TO struct2 KEEPING TARGET LINES.
```

```
struct2 = CORRESPONDING #( APPENDING BASE ( struct2 ) struct1 ).
```

Coresponding performance considerations

- Grouping fields:

  try to arrange fields wich belong together and which are likely to be moved next to another and in the same order

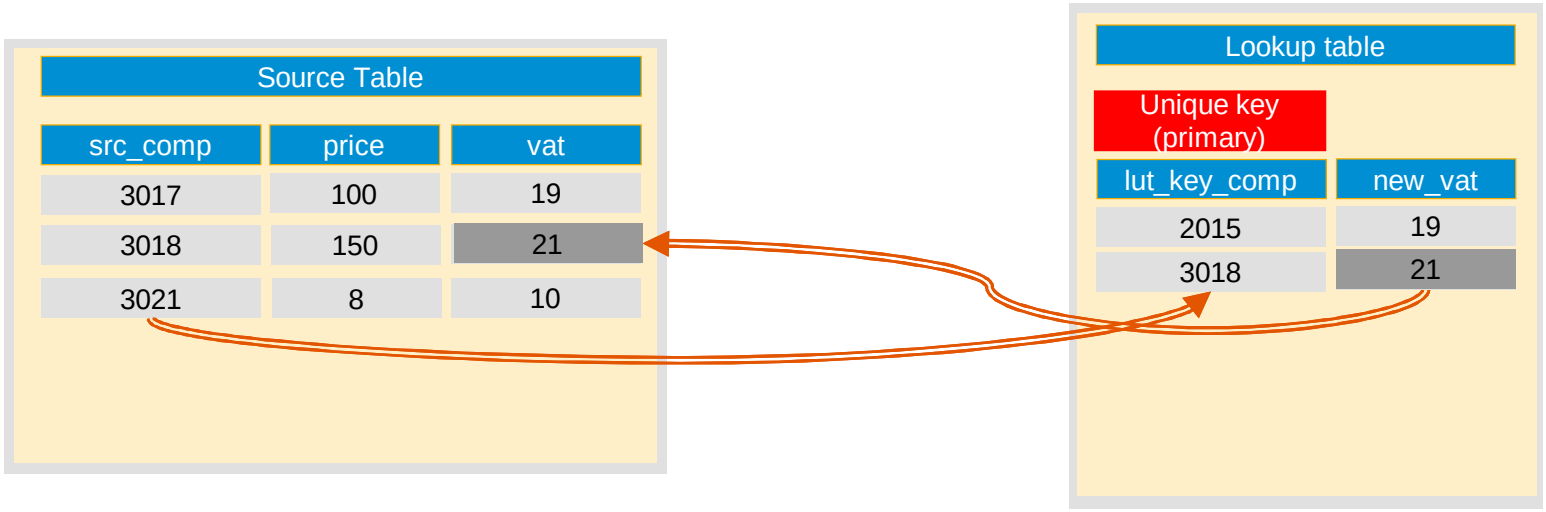
- ABAP compiler does not optimize nested calls

```
dst = CORRESPONDING final_t( CORRESPONDING inter_t( source CORRESPONDING MAPPING id = code ). ).
```

Operator CORRESPONDING

Lookup Example

```
dst_tab = CORRESPONDING #( src_tab FROM lut_tab [USING KEY key]
                           USING src_comp = lut_key_comp
                          MAPPING vat = new_vat ).
```



Operator CORRESPONDING

Lookup Example

```
FIELD-SYMBOLS <dst_line> LIKE LINE OF dst_tab.

dst_tab = src_tab.
LOOP AT dst_itab ASSIGNING <dst_line>.
  READ TABLE fil_tab WITH TABLE KEY foo_key
     COMPONENTS lut_key_comp = <dst_line>-src_comp
     ASSIGNING FIELD-SYMBOL(<fil_line>).
  CHECK sy-subrc = 0.
  <dst_line>-vat = <fil_line>-new_vat.
ENDLOOP.
```

```
dst_tab = CORRESPONDING #( src_tab FROM lut_tab
                           USING src_comp = lut_key_comp
                           MAPPING vat = new_vat ).
```

Operator CORRESPONDING

Syntax

```
target = CORRESPONDING type( source ).
```

- Applies name-matching to structures, and recursively to tables with the option `DEEP`.
- Exact assignment semantic using `EXACT`.
- Start values can be defined using `BASE`.
- Components with distinct names can be mapped explicitly using `MAPPING`.
- Components with equal names can be excluded from mapping using `EXCEPT`.
- Allows unique key denormalization using `DISCARDING DUPLICATES`
- Nested tables can be kept with `APPENDING`
- For dynamic mapping use class `CL_ABAP_CORRESPONDING`
- Lookup version with the clauses `USING` and `FROM`.

CORRESPONDING for RAP

From/to entity and using changing control

```
legacy_obj =  CORRESPONDING #( entity MAPPING FROM ENTITY ).
 entity =  CORRESPONDING #(legacy_type MAPPING TO ENTITY ).
```

```
define behavior for ROOT_ENTITY
 ...
 [deep] mapping for legacy_type [control control_type]
                      [corresponding [except comps] ]
 {
   EntityField      = key_field;
   EntityFieldChild = key_field_child;
   EntityDataField  = data_field;
   EntityCharField  = char_field;
 }
```

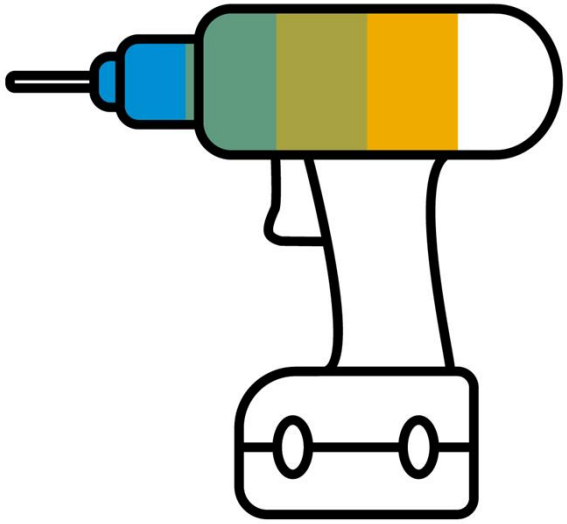
CORRESPONDING for RAP

From/to entity and using changing control (%control)

```
entity =  CORRESPONDING #(legacy_type CHANGING CONTROL ).
  legacy_obj =  CORRESPONDING #(entity USING CONTROL ).
legacy_obj =  CORRESPONDING #( entity MAPPING FROM ENTITY USING CONTROL).
```

```
legacy_obj =  CORRESPONDING #( BASE( legacy_obj ) entity MAPPING FROM ENTITY ).
```

Dynamic corresponding



Dynamic CORRESPONDING

Basic (since 7.43)

```
data(corr) =  cl_abap_corresponding=>create(   source = src
                                               destination = dst
                                               mapping = map ).
corr->execute( exporting source = scnd_src changing destination = scnd_dst ).
```

map = mapping_table(

(level = 0 kind = cl_abap_corresponding=>mapping_component srcname = ‘srccomp’ dstname = ‘dstcmp’)

(level = 0 kind = cl_abap_corresponding=>mapping_except_component srcname = ‘igncomp’)

(level = 0 kind = cl_abap_corresponding=>mapping_except_all)).

Dynamic CORRESPONDING

Advanced mapping example

```
MAPPING usr = user_name
        (nested = nested MAPPING
                    origin = cityfrom
                    destination = cityto
                    EXCEPT  price )
      EXCEPT * ).
```

```
map = mapping_table(
        ( level = 0 kind = cl_abap_corresponding=>mapping_component srcname = 'usr_name' dstname = 'usr' )
        ( level = 0 kind = cl_abap_corresponding=>mapping_component srcname = 'nested' dstname = 'nested' )
        ( level = 1 kind = cl_abap_corresponding=>mapping_component srcname = 'cityfrom' dstname = 'origin' )
        ( level = 1 kind = cl_abap_corresponding=>mapping_component srcname = 'cityto' dstname = 'destination' )
        ( level = 1 kind = cl_abap_corresponding=>mapping_except_component srcname = 'price' )
        ( level = 0 kind = cl_abap_corresponding=>mapping_except_all ) ).
```

Dynamic CORRESPONDING

Keeping target lines (since 7.84)

```
corr->execute( exporting source = scnd_src
                         keeping_lines = abap_true
               changing destination = scnd_dst ).
```

Dynamic CORRESPONDING

Lookup table (since 7.45)

```
data(corr) =  cl_abap_corresponding=>create_using( using = lut
                                                   destination = dst
                                                   mapping = map ).
corr->execute_using( exporting using = lut
                               changing destination = scnd_dst ).
```

```
map = mapping_table(
    ( level = 0 kind = cl_abap_corresponding=>using_key srcname = 'primary_key' )
    ( level = 0 kind = cl_abap_corresponding=>mapping_using_component srcname = 'src_comp'
        dstname = 'lut_key_comp' )
    ( level = 0 kind = cl_abap_corresponding=>mapping_component srcname = 'usr_name' dstname = 'usr' )
....
).
```

Dynamic CORRESPONDING

Mapping values (since 7.52)

```
data(corr) =  cl_abap_corresponding=>create_with_value( source = src
                                                        destination = dst
                                                        mapping = map ).
corr->execute( exporting source = scnd_src changing destination = scnd_dst ).
```

```
map = mapping_table_value(
  ( level = 0 kind = cl_abap_corresponding=>mapping_value dstname = 'usr' value = ref #( 'fixed_value' ) )
  ( level = 0 kind = cl_abap_corresponding=>mapping_default_value srcname = 'cityfrom' dstname = 'origin'
    value = ref #( 'default_value' ) ) ).
```

Dynamic CORRESPONDING

Discarding duplicates (since 7.49)

```
data(corr) =  cl_abap_corresponding=>create(   source = src
                                               destination = dst
                                               discarding_duplicates = abap_true
                                               mapping = map ).
  corr->execute( exporting source = scnd_src changing destination = scnd_dst ).
```

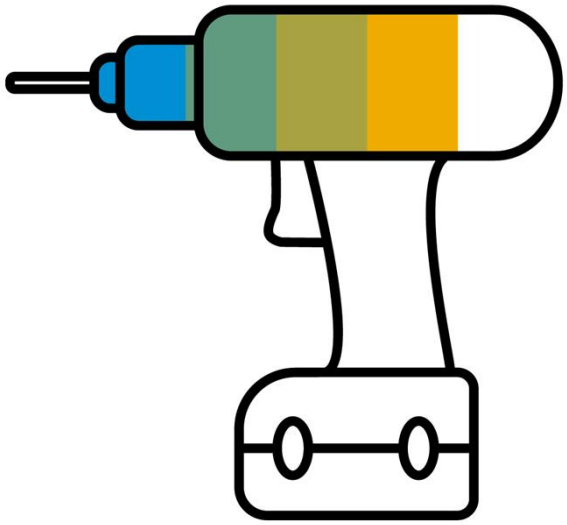
map = mapping_table(
(level = 0 kind = cl_abap_corresponding=>mapping_component srcname = ‘srccomp’ dstname = ‘dstcmp’)
(level = 0 kind = cl_abap_corresponding=>mapping_discarding_duplicates srcname = ‘nested’ dstname = ‘nested’)).

CL_ABAP_CORRESPONDING Performance considerations

- Parsing and matching string maps can be time consuming

```
loop at list_of_tables assigning <nested_table>.
  cl_abap_corresponding=>create( source = src
                                 destination = <nested_table>
                                 mapping = same_map )->execute(
                                           exporting source = src
                                           changing destination = <nested_table> ).
endloop.
```

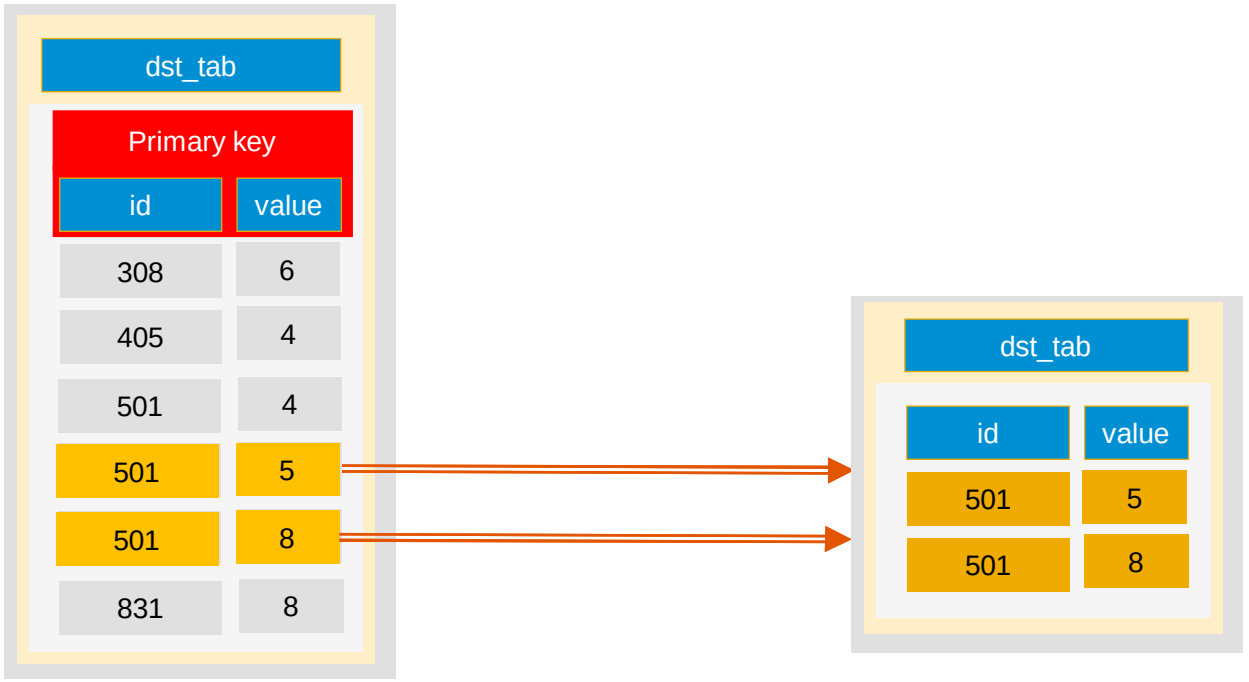
Filter



Filters

Filter by value - WHERE (since 7.42)

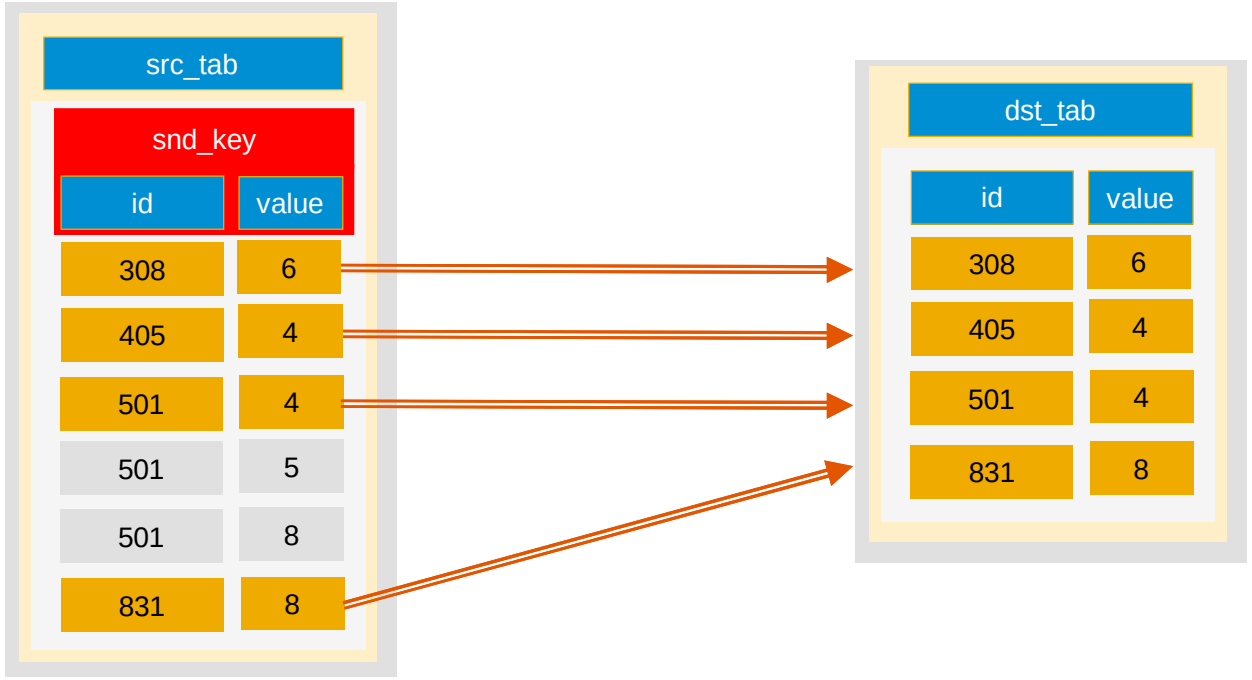
```
dst_table =  FILTER #( src_table WHERE id = 501 AND value >= 5 ).
```



Filters

Filter by value - EXCEPT + KEY (since 7.42)

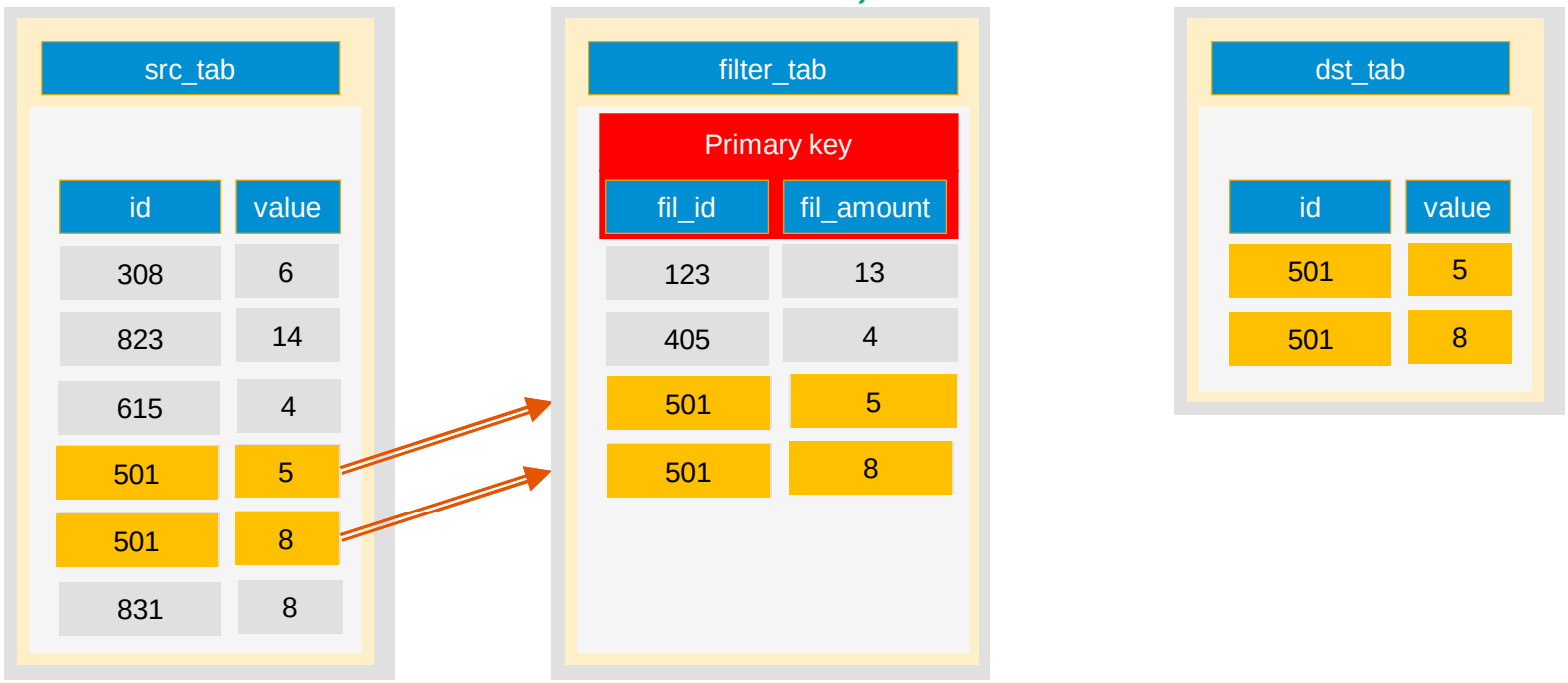
```
dst_table =  FILTER #( src_table EXCEPT USING KEY snd_key
                                  WHERE id = 501 AND value >= 5 ).
```



Filters

Filter by table – WHERE (since 7.42)

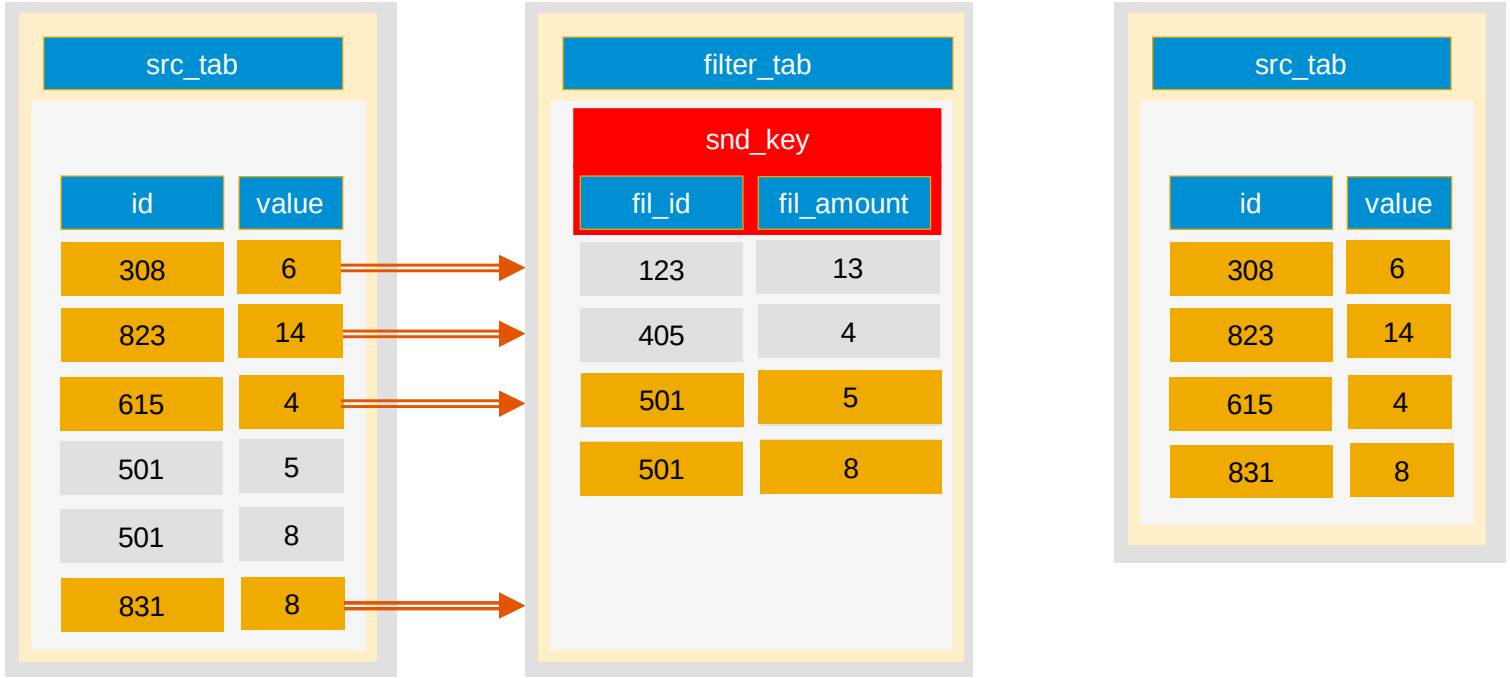
```
dst_table =  FILTER #( src_table IN filter_tab WHERE id = fil_id AND value =
                                                    fil_amount ).
```



Filters

Filter by table - EXCEPT + KEY (since 7.42)

```
dst_table =  FILTER #( src_table EXCEPT IN filter_tab USING KEY snd_key
                       WHERE id = fil_id AND value = fil_amount ).
```



Filters

Filter by table - EXCEPT + SRC KEY (since 7.42)

```
dst_table =  FILTER #( src_table USING KEY snd_key EXCEPT IN filter_tab
                       WHERE id = fil_id AND value = fil_amount ).
```

Filters

Example

Get lines of sorted table src_tab where key_comp is greater than filter_val

```
DATA dst_itab LIKE src_tab.
FIELD-SYMBOLS <line> LIKE LINE OF src_tab

LOOP AT src_tab ASSIGNING <line> WHERE key_comp > filter_val.
  INSERT <line> INTO TABLE dst_tab.
ENDLOOP.
```

```
DATA(dst_tab) = FILTER #( src_tab WHERE key_comp > filter_val ).
```

Filtering by arbitrary non-key components requires the general solution: VALUE operator

```
DATA(dst_tab) = VALUE tab_type( FOR <line> IN src_tab
                                  WHERE any_comp > filter_val
                                ( <line> ) ).
```

Filters

Syntax

```
dst_itab = FILTER table_type( src_itab [USING key]
                                [EXCEPT] [IN ftab] [USING key]
                                WHERE keycomp1 rel_op value1 AND … ).
```

- Building Tables Selecting Multiple Lines by Table Key
- Selects multiple lines of internal tables with sorted or hashed key.
- Addition `EXCEPT` selects all lines that do not fulfill the specified key.
- Secondary keys can be specified with addition `USING`.
- Multiple key values can be specified using a filter table and addition `IN`.

FILTER performance considerations

- ABAP compiler does not optimize nested calls

```
dst = CORRESPONDING #( FILTER #( source . . . ). ).
```

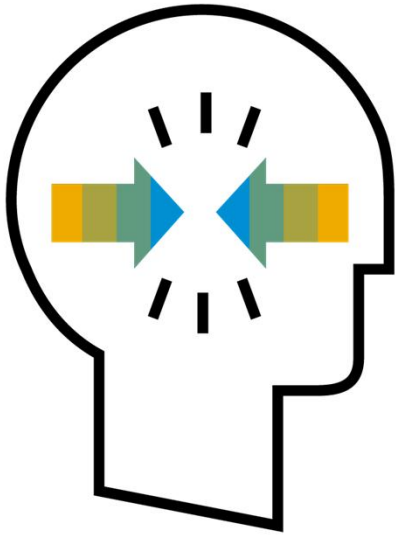
- Make sure you need all the resulting lines

```
dst = CORRESPONDING #( FILTER #( source . . . ). ). )
DESCRIBE TABLE dst LINES number_of_lines.
```

VS.

```
LOOP AT src_tab ASSIGNING <line> WHERE key_comp = 5.
   number_of_lines = number_of_lines + 1.
ENDLOOP.
```

Conclusion

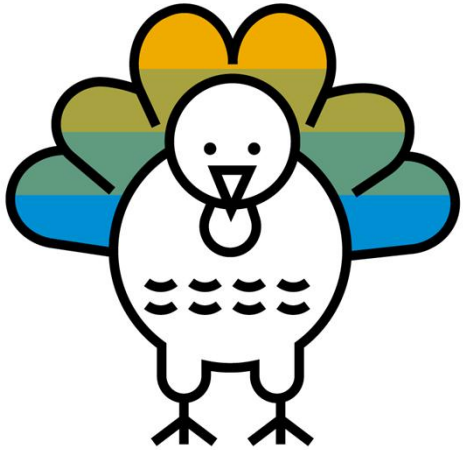


General performance considerations

- The three pillars of ABAP Perfomance

1. Knowing when to push down
2. Choosing the right keys
3. Choosing the right component layout

Thank you.



THE BEST RUN SAP