**Internal Tables**

**I. Internal Tables**

**1. Basic Statements**

**1.1 Data types**

**1.1.1 Elementary types**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Keyword** |
| Fixed length data types | Text/Character field | C |
| Numeric text | N |
| Integers | I |
| Packed number | P |
| Floating point | F |
| Date | D |
| Time | T |
| Byte/Hexadecimal field | X |
| Variable length data types | Text string | STRING |
| XSTRING | XSTRING |

" Khai báo các kiểu dữ liệu trong ABAP

" Alphanumeric character, độ dài cố định

DATA: gv\_char TYPE c LENGTH 50.  " Ban đầu là khoảng trắng

" Số nguyên từ 0-9, độ dài cố định

DATA: gv\_num TYPE n LENGTH 10.  " Ban đầu là ký tự số không

" Ngày định dạng YYYYMMDD, độ dài cố định

DATA: gv\_date TYPE d.  " Ban đầu là '00000000'

" Thời gian định dạng HHMMSS, độ dài cố định

DATA: gv\_time TYPE t.  " Ban đầu là '000000'

" Trường hệ thập lục phân, độ dài cố định

DATA: gv\_hex TYPE x LENGTH 5.  " Ban đầu là các byte không xác định

" Số thập phân đóng gói, độ dài cố định

DATA: gv\_packed TYPE p LENGTH 8.  " Ban đầu là 0

" Số nguyên, độ dài cố định

DATA: gv\_int TYPE i.  " Ban đầu là 0

" Số thực, độ dài cố định

DATA: gv\_float TYPE f.  " Ban đầu là 0.0

" Chuỗi ký tự, độ dài thay đổi

DATA: gv\_string TYPE string.  " Ban đầu là chuỗi rỗng

" Chuỗi thập lục phân, độ dài thay đổi

DATA: gv\_xstring TYPE xstring.  " Ban đầu là chuỗi rỗng

**1.1.2 Complex types (Structure and Internal Table)**

**💡 Simple Structures**: Chứa các kiểu dữ liệu cơ bản có độ dài cố định hoặc thay đổi, không có cấu trúc lồng nhau.

" Simple Structure

TYPES: BEGIN OF simple\_structure,

         field1 TYPE c LENGTH 10,  " Ký tự, độ dài cố định

         field2 TYPE string,       " Ký tự, độ dài thay đổi

       END OF simple\_structure.

**💡 Nested Structures**: Chứa ít nhất một cấu trúc khác bên trong.

" Nested Structure

TYPES: BEGIN OF nested\_structure,

         field1 TYPE c LENGTH 10,

         sub\_structure TYPE simple\_structure,  " Cấu trúc lồng nhau

       END OF nested\_structure.

**💡 Flat Structures**: Chỉ chứa các kiểu dữ liệu cơ bản có độ dài cố định, không có internal table, kiểu tham chiếu, hoặc chuỗi.

" Flat Structure

TYPES: BEGIN OF flat\_structure,

         field1 TYPE c LENGTH 10,  " Chỉ ký tự có độ dài cố định

         field2 TYPE i,

       END OF flat\_structure.

**💡 Deep Structures**: Chứa ít nhất một Internal table, kiểu tham chiếu, hoặc chuỗi.

" Deep Structure

TYPES: BEGIN OF deep\_structure,

         field1 TYPE c LENGTH 10,

         internal\_table TYPE TABLE OF simple\_structure,  " Internal table

         string\_field TYPE string,                       " Chuỗi

       END OF deep\_structure.

*Internal table sẽ được đề cập sau*

**1.2.3 Reference (Data Reference và Object Reference)**

" Data Reference Variable

DATA: gv\_data TYPE i VALUE 10.         " Biến dữ liệu

" (bắt buộc, điểm khác so với C++/Jv)

DATA: gr\_data TYPE REF TO data.      " Tham chiếu dữ liệu bát kì

gr\_data = REF #( gv\_data ).          " Gắn tham chiếu dữ liệu vào biến dữ liệu

WRITE: gr\_data->\*.                   " In giá trị của biến dữ liệu (10)

" Class Reference Variable

CLASS lcl\_example DEFINITION.

ENDCLASS.

DATA: go\_instance TYPE REF TO lcl\_example.   " Tham chiếu class

CREATE OBJECT go\_instance.                  " Tạo đối tượng từ class

" Interface Reference Variable

INTERFACE lif\_example.

ENDINTERFACE.

DATA: go\_instance TYPE REF TO lif\_example.   " Tham chiếu interface

CREATE OBJECT go\_instance.                  " Tạo đối tượng từ interface

" Biến iref có thể trỏ tới các đối tượng của bất kỳ class nào implement i1

INTERFACE i1.

...

ENDINTERFACE.

CLASS c1 DEFINITION.

  PUBLIC SECTION.

    INTERFACES i1.

ENDCLASS.

TYPES: iref TYPE REF TO i1.  " Định nghĩa kiểu dữ liệu cho biến tham chiếu interface

" Biến cref có thể trỏ tới các đối tượng của lớp c1 và các lớp con của nó

CLASS c1 DEFINITION.

    ...

ENDCLASS.

TYPES: cref TYPE REF TO c1.  " Định nghĩa kiểu dữ liệu cho biến tham chiếu class

" Biến dref có thể trỏ tới các đối tượng dữ liệu mà iref có thể trỏ tới

"  tức là các đối tượng implement interface i1

TYPES: dref TYPE REF TO iref.  " Định nghĩa kiểu dữ liệu cho biến tham chiếu dữ liệu

**1.2 Data Objects**

Tutorialscampus.com

**1.3 Variables**

**1.4 Processing Data**

**1.5 Operators**

**1.6 Decicion Control Statements**

**1.7 Loop Statements**

**1.8 Loop Control Statements**

**2. Introduction**

**💡** Internal table trong ABAP là cấu trúc dữ liệu động, chứa các dòng cùng kiểu, được xác định bởi loại dòng, loại bảng, và khóa, cho phép truy cập qua chỉ mục hoặc khóa, dùng để xử lý tập dữ liệu đa dạng một cách có cấu trúc.

**💡** Giúp xử lý dữ liệu trong bộ nhớ lúc đang chạy (runtime), không lưu trữ dữ liệu vĩnh viễn như bảng cơ sở dữ liệu, bao gồm thêm, xóa, tìm kiếm và sắp xếp. Thường được sử dụng để tạm thời lưu trữ dữ liệu từ cơ sở dữ liệu hoặc để xử lý dữ liệu.

**💡** Chỉ tồn tại trong suốt quá trình thực thi chương trình. Chúng được tạo và điền dữ liệu khi chương trình chạy và bị loại bỏ khi chương trình kết thúc.

**💡** Là túi thần kì, khoogn cần khai báo (có thể khai báo) kích thước ban đầu, SAP cho phép nó tự co giãn.

**3. Basic Properties**

**3.1 Line Type**

**💡** Line Type trong SAP ABAP là định nghĩa cấu trúc của mỗi dòng trong internal table, xác định các cột và có thể là loại dữ liệu ABAP elementary hoặc complex, bao gồm cả deep structure và internal table hoặc [reference](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abenreference_type_glosry.htm).

**3.2 Table Category**

Declaration of a structure:

TYPES:  BEGIN OF ty\_sflight,

          carrid TYPE sflight-carrid,

          connid TYPE sflight-connid,

          fldate TYPE sflight-fldate,

        END OF ty\_sflight.

DATA(lt\_sflight) = VALUE sflight\_t(

                                    carrid = 'LH'

                                    connid = '0400'

                                    fldate = '20201231' ).

WRITE:  |{ lt\_sflight[ 0 ]-carrid }|,

        |{ lt\_sflight[ 1 ]-connid }|,

        |{ lt\_sflight[ 2 ]-fldate }|.

" Output: LH, 0400, 20201231

**3.2.1 Standard Table**

TYPES:  BEGIN OF ty\_sflight,

          carrid TYPE sflight-carrid,

          connid TYPE sflight-connid,

          fldate TYPE sflight-fldate,

        END OF ty\_sflight.

DATA: lt\_sflight TYPE STANDARD TABLE OF ty\_sflight.

**3.2.2 Sorted Table**

TYPES:  BEGIN OF ty\_sflight,

          carrid TYPE sflight-carrid,

          connid TYPE sflight-connid,

          fldate TYPE sflight-fldate,

        END OF ty\_sflight.

DATA: lt\_sflight TYPE SORTED TABLE OF ty\_sflight

                      WITH UNIQUE KEY carrid connid.

**3.2.3 Hashed Table**

TYPES:  BEGIN OF ty\_sflight,

          carrid TYPE sflight-carrid,

          connid TYPE sflight-connid,

          fldate TYPE sflight-fldate,

        END OF ty\_sflight.

DATA: lt\_sflight TYPE HASHED TABLE OF ty\_sflight

                      WITH UNIQUE KEY carrid connid.

**Deep internal table**

An internal table inside an internal table. E.g: An internal table with a hashed key, but multiple rows for each key:

TYPES: BEGIN OF ty\_your\_structure,

        field1 TYPE c LENGTH 10,

        field2 TYPE i,

       END OF ty\_your\_structure.

TYPES: BEGIN OF ty\_your\_table\_structure,

        key TYPE string,

        rows TYPE STANDARD TABLE OF ty\_your\_structure WITH EMPTY KEY,

       END OF ty\_your\_table\_structure.

TYPES tt\_your\_table TYPE HASHED TABLE OF ty\_your\_table\_structure WITH UNIQUE KEY key.

DATA(lt\_your\_table) = VALUE tt\_your\_table(

  ( key = 'TEST'  rows = VALUE #( ( field1 = 'X' field2 = 1 )

                                  ( field1 = 'Y' field2 = 2 ) ) )

  ( key = 'TES2T' rows = VALUE #( ( field1 = 'A' field2 = 1 )

                                  ( field1 = 'B' field2 = 2 ) ) )

).

DATA(lt\_rows) = VALUE #( lt\_your\_table[ KEY = 'TEST' ]-rows OPTIONAL ).

**Table Category**

**💡** Quản lý và lưu trữ: Xác định cách quản lý và lưu trữ bảng, ảnh hưởng đến hiệu suất truy cập. Chọn loại bảng phù hợp tối ưu hóa truy cập, đặc biệt với bảng lớn.

**💡** Access by table index: cách nhanh nhất, đánh số từng record

**💡** Access by table key: Tìm giá trị cụ thể trong các cột, có thể là key hoặc không.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Category** | **Internally Managed By** | **Access** | **Primary Table Key** | **When to Use** | **Hints** |
| Standard | Index | By index (sequential) | Non-unique (can be empty) | General use with non-unique entries | Flexible, supports duplicates, slower for large data |
| Sorted | Index | By index or key | Unique or non-unique | Sorted access required | Automatically sorted, efficient for range selections |
| Hashed | Hash algorithm | By key (fast) | Always unique | Fast key access | Best for single record access, no duplicates, unsorted |

**3.3 Key Attributes**

**Thuộc tính của khóa:**

**Duy nhất (Unique)** hoặc **Không duy nhất (Non-Unique)**:

* + Khóa duy nhất không cho phép các dòng trùng lặp.
  + Khóa không duy nhất cho phép các dòng trùng lặp.

**Bảng tiêu chuẩn (Standard Tables)**:

* + Khóa chính có thể trống (không chứa cột khóa nào).
  + Truy cập tối ưu chỉ có thể thực hiện được với khóa phụ.

**Loại khóa cụ thể:**

**Khóa sắp xếp (Sorted Keys):**

* + **Khóa chính** của bảng sắp xếp hoặc **khóa phụ** của bất kỳ bảng nào.
  + **Quản lý** bằng chỉ số bảng:
    - Bảng sắp xếp: chỉ số chính.
    - Khóa phụ: thêm chỉ số phụ.
  + **Truy cập** bằng khóa sắp xếp cho phép tìm kiếm nhị phân tối ưu.

**Khóa băm (Hashed Keys):**

* + **Khóa chính** của bảng băm hoặc **khóa phụ** của bất kỳ bảng nào.
  + **Quản lý** bằng thuật toán băm.
  + **Không** có chỉ số bảng cho khóa băm.

**4. Table Keys in Internal Tb (Primary, Secondary, Standard, Empty)**

**4.1 Primary table key**

**💡** Each internal table has a primary table key. Can be either a self-defined key or the standard key. **Self-Defined Key** là khóa do người dùng tự định nghĩa để xác định một hoặc nhiều cột trong internal table.

**💡** The primary table key is: sorted for sorted tables, hashed for hashed tables. The key fields in sorted and hashed tables are read-only. This is not valid for standard tables.

**💡** Việc chỉ định khóa chính có thể được bỏ qua chỉ đối với các Standard Table. Khi đó, khóa chính của bảng sẽ tự động được định nghĩa là một non-unique standard key.

**💡** Primary key thường được chỉ định mặc định là ‘primary\_key’. Tuy nhiên có thể không cần chỉ rõ hoặc có thể đặt cho nó một cái tên khác tùy ý.

**💡** Khi access vào một internal tables thông qua table key thì primary\_key mặc định sẽ được sử dụng một cách ngầm định, trừ khi có chỉ định một khóa phụ khác. Tuy nhiên cái gì thì cũng cần chỉ định rõ ràng, không nên lạm dụng các trường hợp ngầm định như vậy.

**💡** Key có thể là một vài field riêng lẽ hoặc cả dòng (table\_line 🡪 một khái niệm được hệ thống ABAP hiểu và xử lý).

Vd: DATA: itab TYPE SORTED TABLE OF i WITH UNIQUE KEY table\_line.

**💡** Đối với các dòng không có cấu trúc (ví dụ, các dòng chỉ chứa giá trị đơn lẻ không phải là một cấu trúc hoặc bảng), table\_line là cách duy nhất để định nghĩa khóa. **Dòng không có cấu trúc** là dòng dữ liệu không được tổ chức thành các trường riêng lẻ, ví dụ, một giá trị số nguyên hoặc chuỗi đơn giản.

**4.2** **Standard key**

**💡** The standard key is a special primary table key. Có thể được khai báo rõ ràng hoặc ngầm định.

**💡** Standard key của internal table mà head line là structure 🡪 Primary key là các trường character và byte (trừ trường số, …). Đối với non-structured/elementary line type 🡪 key là toàn bộ line.

**💡** Một internal table không có đặc tả khóa rõ ràng thì ngầm định standard key là primary key.

**💡** Standard key cần phải được xem xét và để ý kỹ lưỡng: Việc sắp xếp trên Standard key có thể gây ra kết quả không mong muốn. Các trường của primary key là read-only, tức là sử dụng Standard key với các danh mục bảng này và sau đó các trường sửa đổi (vô tình) có thể gây ra lỗi runtime. Chỉ định khóa rõ rafngg sẽ dễ đọc, tránh nhầm lẫn, lộn xộn.

**4.3** **Empty key**

**💡** The primary table key of a standard table can be empty, it does not contain any key fields. An empty key is not possible for sorted and hashed tables.

**💡** Dùng khi việc khởi tạo một table key là không cần thiết, thay vì chỉ định là không có key. Nếu không, sử dụng Standard key sẽ dẫn đến kết quả không mong muốn như ở trên.

**💡** Chỉ định gõ ràng bằng cách dùng từ khóa EMPTY KEY. Nếu sử dụng standard key mà các field của dòng là structure và đều không phải là non-numeric hoặc line không có cấu trúc dạng bảng thì Empty key được chỉ định một cách ngầm định (dùng standard, nhưng không có field nào khác số hoặc cả line đều không phải structure).

**💡** Lưu ý rằng khi sử dụng một khai báo dạng inline declaration như INTO TABLE @DATA(itab) trong các câu lệnh SELECT trong ABAP, bảng kết quả sẽ là một standard table và có Empty key.

**4.4** **Secondary table keys**

**💡** Secondary table keys are optional for all table categories, can be unique/non-unique sorted keys or unique hash keys. It have a self-defined name. An alias name can also be specified.

**💡** Bí danh dùng cho trường hợp tên tự định nghĩa cần biểu thị một ý nghĩa nào đó nên quá dài làm mã khó viết và khó đọc hoặc các tên tự định nghĩa bị trùng nhau gây khó hiểu 🡪 Dùng bí danh.

**💡** Khi định nghĩa một sorted secondary key cho một internal table, hệ thống sẽ tạo ra một secondary table index để quản lý và sắp xếp dữ liệu dựa trên khóa này.

**💡** Thông thường, bảng băm (hashed tables) không có chỉ số trực tiếp vì chúng sử dụng thuật toán băm để quản lý dữ liệu. Tuy nhiên, khi sử dụng sorted secondary key, hệ thống sẽ tạo ra secondary index, cho phép truy cập dữ liệu theo chỉ số. Lúc này biến hệ thống sy-tabix (được sử dụng để lưu chỉ số dòng hiện tại trong internal table) sẽ được thiết lập.

DATA: itab TYPE HASHED TABLE OF line WITH UNIQUE KEY primary\_key

           WITH NON-UNIQUE SORTED KEY secondary\_key COMPONENTS col1 col2.

READ TABLE itab WITH TABLE KEY secondary\_key = key\_value INTO wa.

IF sy-subrc = 0.

  WRITE: / 'Found at index', sy-tabix.

ENDIF.

**💡** Khi sử dụng Secondary table keys để truy cập internal table, cần phải chỉ rõ tên khóa hoặc bí danh được dùng. Secondary table keys không được chọn tự động bởi hệ thống. Nếu không chỉ rõ, hệ thống sẽ sử dụng khóa chính hoặc primary table index.

DATA: itab TYPE SORTED TABLE OF line

            WITH NON-UNIQUE KEY col1 col2.

READ TABLE itab WITH KEY col1 = 'value1' col2 = 'value2' INTO wa.

**💡** Sử dụng Secondary table keys để tăng hiệu suất đọc dữ liệu từ internal table. Khi có nhiều truy vấn đọc, việc sử dụng Secondary table keys sẽ giúp tìm kiếm nhanh hơn so với việc sử dụng khóa chính không tối ưu.

DATA: itab TYPE HASHED TABLE OF spfli

          WITH UNIQUE KEY carrid connid.

READ TABLE itab WITH KEY carrid = 'LH' connid = '0400' INTO wa.

**💡** Đối với các bảng chứa hàng triệu bản ghi, được điền dữ liệu một lần và thay đổi thường xuyên, việc sử dụng khóa phụ giúp tăng tốc độ truy cập và cập nhật dữ liệu.

DATA: large\_itab TYPE HASHED TABLE OF spfli

                 WITH UNIQUE KEY carrid connid

                 WITH NON-UNIQUE SORTED KEY city\_keys COMPONENTS cityfrom cityto.

LOOP AT large\_itab USING KEY city\_keys WHERE cityfrom = 'FRANKFURT'.

  " Xử lý các bản ghi có cityfrom là 'FRANKFURT'

ENDLOOP.

**💡** Sử dụng để đảm bảo tính read-only và đảm bảo chỉ đọc các mục không trùng lặp của những trường dữ liệu mà khóa chính là non-unique.

DATA: std\_itab TYPE STANDARD TABLE OF spfli

               WITH NON-UNIQUE KEY carrid connid

               WITH UNIQUE SORTED KEY unique\_keys COMPONENTS cityfrom cityto.

READ TABLE std\_itab WITH KEY unique\_keys cityfrom = 'NEW YORK' cityto = 'FRANKFURT' INTO wa.

**Header Line**

**💡** Khi tạo một internal table, có thể khai báo một header line có cùng tên với table. Header line này có thể được sử dụng như một work area khi xử lý internal table (work area có tên là header line). Các câu lệnh ABAP sử dụng với internal table có thể có dạng ngắn nếu internal table có header line. Các câu lệnh này tự động giả định rằng header line là vùng làm việc ngầm định.

DATA: BEGIN OF CUSTOMER\_TAB OCCURS 5,

         KUNNR TYPE KNA1-KUNNR,

         NAME1 TYPE KNA1-NAME1,

      END OF CUSTOMER\_TAB.

🡪 This declaration creates an internal table and a structure using the same name: CUSTOMER\_TAB.

|  |  |  |
| --- | --- | --- |
| **Không sử dụng Header Line** | **Sử dụng Header Line** | **Áp dụng cho tất cả các loại table** |
| INSERT wa INTO TABLE itab. | INSERT TABLE ITAB. | Chèn dữ liệu |
| COLLECT wa INTO itab. | COLLECT itab. | Thu thập dữ liệu |
| READ TABLE itab... INTO wa. | READ TABLE itab... | Đọc dữ liệu |
| MODIFY TABLE itab FROM wa... | MODIFY TABLE itab... | Sửa đổi dữ liệu |
| MODIFY itab FROM wa...WHERE... | MODIFY itab... WHERE... | Sửa đổi với điều kiện |
| DELETE TABLE itab FROM wa. | DELETE TABLE itab. | Xóa dữ liệu |
| LOOP AT ITAB INTO wa... | LOOP AT ITAB... | Vòng lặp |
| **Chỉ áp dụng cho bảng có chỉ số** | | |
| APPEND wa TO itab. | APPEND itab. | Thêm dữ liệu vào cuối bảng |
| INSERT wa INTO itab... | INSERT itab... | Chèn dữ liệu vào bảng |

**💡** Không được phép sử dụng header lines trong các lớp (classes). Câu lệnh này đã lỗi thời và chỉ còn được duy trì để đảm bảo tính tương thích với các phiên bản trước 4.6 và 6.10.

**💡** Header lines có thể xuất hiện trong các câu lệnh cũ nhưng không nên tiếp tục sử dụng. Vì nó dễ gây nhầm lẫn, do tạo work area và intenal table cùng tên. Việc tạo các work area để làm việc với internal table nên được thực hiện riêng lẻ.

**💡** Header Line cho phép sử dụng các câu lệnh ngắn hơn, nhưng không dễ hiểu hơn vì không thể nhận ra ngay nguồn gốc và mục tiêu của việc gán giá trị. Dễ gây nhầm lẫn trong các thao tác với internal table, do đó nên sử dụng internal table với **các work area** có tên khác nhau.

Có header line thì có thể truy cập, thêm sửa xóa record trước tiếp của internal table thông quan header line. Không khai báo header line thì muốn làm gì phải khai báo và chỉ định 1 work area và truy cập vào bảng thông qua nó, không thể truy cập trực tiếp vào internal table.

" With HEADER LINES

TYPES: BEGIN OF line,

         col1 TYPE i,

         col2 TYPE i,

       END OF line.

DATA itab TYPE HASHED TABLE OF line WITH UNIQUE KEY col1

               WITH HEADER LINE.

DO 4 TIMES.

  itab-col1 = sy-index.

  itab-col2 = sy-index \*\* 2.

  INSERT TABLE itab.

ENDDO.

itab-col1 = 2.

READ TABLE itab FROM itab.

itab-col2 = 100.

MODIFY TABLE itab.

itab-col1 = 4.

DELETE TABLE itab.

LOOP AT itab.

  WRITE: / itab-col1, itab-col2.

ENDLOOP.

" WITHOUT HEADER LINES

\* Định nghĩa cấu trúc dữ liệu cho các hàng trong bảng

TYPES: BEGIN OF line,

         col1 TYPE i,

         col2 TYPE i,

       END OF line.

\* Khai báo internal table và work area với tên khác nhau

DATA: itab TYPE HASHED TABLE OF line WITH UNIQUE KEY col1,  " Internal table

      wa   TYPE line.                                      " Vùng làm việc

START-OF-SELECTION.

\* Thêm dữ liệu vào internal table

DO 4 TIMES.

  wa-col1 = sy-index.

  wa-col2 = sy-index \*\* 2.

  INSERT wa INTO TABLE itab.

ENDDO.

\* Đọc và hiển thị dữ liệu từ internal table

LOOP AT itab INTO wa.

  WRITE: / 'Column 1:', wa-col1, 'Column 2:', wa-col2.

ENDLOOP.

\* Hiển thị thông báo kết thúc chương trình

WRITE: / 'Program completed.'.

" output:

1         1

2       100

3         9

**5. Creating Internal Tables and Types**

**💡** Declare internal tables and internal table types in ABAP programs using the TYPES and DATA statements. The relevant syntax elements for internal tables are TABLE OF in combination with the additions TYPE or LIKE.

**💡 Note**: Internal tables can also be created dynamically. 🡪 Để tính sau

Steps:

1. Define a structured data type (locally or globally). This is not necessary if you use, for example, the name of a database table or [CDS view](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abencds_view_glosry.htm) in the internal table declaration. In these cases their line type is used automatically.
2. Define an internal table type.
3. Create the internal table, i.e. a data object that uses this type.

TYPES itab\_type1 TYPE STANDARD TABLE OF data\_type ...   "Standard table type

TYPES itab\_type2 LIKE SORTED   TABLE OF data\_object ... "Sorted table type

DATA  itab1      TYPE          TABLE OF data\_type ...   "Standard table by default

DATA  itab2      TYPE HASHED   TABLE OF data\_type ...   "Hashed table

DATA  itab3      TYPE                   itab\_type1 ...  "Based on an existing internal table type

DATA  itab4      LIKE                   itab1 ...       "Based on an existing internal table

**💡** If the table category is not specified (... TYPE TABLE OF ...), it is automatically ... TYPE STANDARD TABLE OF ....

"------------------ Standard table key ------------------

"Standard table without explicit primary table key specification. Note that STANDARD

"is not explicitly specified.

"Implicitly, the standard key is used; all non-numeric table fields

"make up the primary table key.

DATA it1 TYPE TABLE OF zdemo\_abap\_fli.

"Explicitly specifying STANDARD for a standard table.

"Explicitly specifying the standard table key. It is the same as it1.

DATA it2 TYPE STANDARD TABLE OF zdemo\_abap\_fli WITH DEFAULT KEY.

"Hashed table with unique standard table key

DATA it3 TYPE HASHED TABLE OF zdemo\_abap\_fli WITH UNIQUE DEFAULT KEY.

"Sorted table with non-unique standard table key

DATA it4 TYPE SORTED TABLE OF zdemo\_abap\_fli WITH NON-UNIQUE DEFAULT KEY.

"Elementary line type; the whole table line is the standard table key

DATA it5 TYPE TABLE OF i.

"------------------ Primary table key ------------------

"Specifying the primary table key

"Standard tables: only a non-unique key possible

"The following two examples are the same. NON-UNIQUE can be ommitted but is implicitly added.

DATA it6 TYPE TABLE OF zdemo\_abap\_fli WITH NON-UNIQUE KEY carrid.

DATA it7 TYPE TABLE OF zdemo\_abap\_fli WITH KEY carrid.

"Sorted tables: both UNIQUE and NON-UNIQUE possible

DATA it8 TYPE SORTED TABLE OF zdemo\_abap\_fli WITH UNIQUE KEY carrid connid.

DATA it9 TYPE SORTED TABLE OF zdemo\_abap\_fli WITH NON-UNIQUE KEY carrid connid cityfrom.

"Hashed tables: UNIQUE KEY must be specified

DATA it10 TYPE HASHED TABLE OF zdemo\_abap\_fli WITH UNIQUE KEY carrid.

"Explicitly specifying the predefined name primary\_key and listing the components.

"The example is the same as it6 and it7.

DATA it11 TYPE TABLE OF zdemo\_abap\_fli WITH KEY primary\_key COMPONENTS carrid.

"The following example is the same as it9.

DATA it12 TYPE SORTED TABLE OF zdemo\_abap\_fli

  WITH NON-UNIQUE KEY primary\_key COMPONENTS carrid connid cityfrom.

"Specifying an alias name for a primary table key.

"Only possible for sorted/hashed tables.

DATA it13 TYPE SORTED TABLE OF zdemo\_abap\_fli

  WITH NON-UNIQUE KEY primary\_key

  ALIAS p1 COMPONENTS carrid connid cityfrom.

"Specifying a key that is composed of the entire line using the predefined table\_line.

"In the example, an alias name is defined for a primary table key.

DATA it14 TYPE HASHED TABLE OF zdemo\_abap\_fli

  WITH UNIQUE KEY primary\_key

  ALIAS p2 COMPONENTS table\_line.

"------------------ Empty key ------------------

"Empty keys only possible for standard tables

DATA it15 TYPE TABLE OF zdemo\_abap\_fli WITH EMPTY KEY.

"Excursion: The inline declaration in a SELECT statement produces a standard table with empty key.

SELECT \* FROM zdemo\_abap\_fli INTO TABLE @DATA(it16).

"------------------ Secondary table key ------------------

"The following examples demonstrate secondary table keys that are possible for all table categories.

DATA it17 TYPE TABLE OF zdemo\_abap\_fli                      "standard table

  WITH NON-UNIQUE KEY carrid connid                         "primary table key

  WITH UNIQUE SORTED KEY cities COMPONENTS cityfrom cityto. "secondary table key

DATA it18 TYPE HASHED TABLE OF zdemo\_abap\_fli               "hashed table

  WITH UNIQUE KEY carrid connid

  WITH NON-UNIQUE SORTED KEY airports COMPONENTS airpfrom airpto.

DATA it19 TYPE SORTED TABLE OF zdemo\_abap\_fli              "sorted table

  WITH UNIQUE KEY carrid connid

  WITH UNIQUE HASHED KEY countries COMPONENTS countryfr countryto.

"Multiple secondary keys are possible

DATA it20 TYPE TABLE OF zdemo\_abap\_fli

  WITH NON-UNIQUE KEY primary\_key COMPONENTS carrid connid

  WITH NON-UNIQUE SORTED KEY cities COMPONENTS cityfrom cityto

  WITH UNIQUE HASHED KEY airports COMPONENTS airpfrom airpto.

"Alias names for secondary table keys (and also for the primary table key in the example)

DATA it21 TYPE SORTED TABLE OF zdemo\_abap\_fli

  WITH NON-UNIQUE KEY primary\_key ALIAS k1 COMPONENTS carrid connid city

  WITH NON-UNIQUE SORTED KEY cities ALIAS s1 COMPONENTS cityfrom cityto

  WITH UNIQUE HASHED KEY airports ALIAS s2 COMPONENTS airpfrom airpto.

"Example for using table keys and alias names using a LOOP AT statement.

"All of the statements below are possible.

"Note that if the secondary table key is not specified (and if the USING KEY addition is not

"used in the example), the primary table key is respected by default.

"Further ABAP statements, such as READ or MODIFY, are available in which the key can be

"explicitly specified to process internal tables.

LOOP AT it21 INTO DATA(wa) USING KEY primary\_key.

"LOOP AT it21 INTO DATA(wa) USING KEY k1.

"LOOP AT it21 INTO DATA(wa) USING KEY cities.

"LOOP AT it21 INTO DATA(wa) USING KEY s1.

"LOOP AT it21 INTO DATA(wa) USING KEY airports.

"LOOP AT it21 INTO DATA(wa) USING KEY s2.

  ...

ENDLOOP.

Can also create an internal table by ...

* combining the data object creation and table type definition in one step.
* using an [inline declaration](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abeninline_declaration_glosry.htm). Such inline declarations are possible at suitable [declaration positions](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abendeclaration_positions.htm) if the operand type can be fully determined, for example, using a DATA statement (or [FINAL](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abenfinal_inline.htm) for immutable variables).

"1. Defining line type locally

TYPES: BEGIN OF ls\_loc,

        key\_field TYPE i,

        char1     TYPE c LENGTH 10,

        char2     TYPE c LENGTH 10,

        num1      TYPE i,

        num2      TYPE i,

      END OF ls\_loc.

"2. Defining internal table types

"All of the examples use the short form:

TYPES:

  "Standard table type based on locally defined structure type.

  tt\_loc\_str TYPE TABLE OF ls\_loc WITH NON-UNIQUE KEY key\_field,

  "Based on global structure type

  tt\_gl\_str TYPE TABLE OF zsome\_global\_struc\_type WITH NON-UNIQUE KEY key\_field,

  "Based on database table (could also be, e. g. a CDS view)

  "In this case, the line type of the table is automatically used.

  tt\_gl\_tab TYPE TABLE OF zdemo\_abap\_fli WITH NON-UNIQUE KEY carrid,

  "Based on an elementary type

  tt\_el\_type TYPE TABLE OF i.

"3. Creating internal tables ...

"... from locally defined table types

DATA: itab\_a1 TYPE tt\_loc\_str,

      itab\_a2 TYPE tt\_gl\_str,

      itab\_a3 TYPE tt\_gl\_tab,

      itab\_a4 TYPE tt\_el\_type.

"... from global internal table types

DATA itab\_a5 TYPE string\_table.

"Combining data object and table type definition

DATA itab\_a6 TYPE TABLE OF ls\_loc WITH NON-UNIQUE KEY key\_field.

"Internal table based on an already existing internal table using LIKE.

"Here, an internal table is created containing internal tables of the

"type of itab\_a6.

DATA itab\_a7 LIKE TABLE OF itab\_a6.

"Creating internal tables by inline declarations

"Table declared inline in the context of an assignment

"The examples show the copying of a table including the content on the fly

"and creating the table in one step. The data type of the

"declared variable is determined by the right side.

DATA(it\_inline1) = itab\_a1.

DATA(it\_inline2) = it\_inline1.

"Using FINAL for creating immutable variables

FINAL(it\_final) = it\_inline1.

"Using the VALUE operator and an internal table type

DATA(it\_inline3) = VALUE tt\_loc\_str( ( ... ) ).

"Not providing any table lines means the table is initial

"and has the same effect as the declaration of

"itab\_a1 above.

DATA(it\_inline4) = VALUE tt\_loc\_str( ).

"Table declared inline in the context of a SELECT statement;

"a prior extra declaration of an internal table is not needed.

SELECT \* FROM zdemo\_abap\_fli INTO TABLE @DATA(it\_inline5).

"Instead of

DATA it\_sel TYPE TABLE OF zdemo\_abap\_fli WITH EMPTY KEY.

SELECT \* FROM zdemo\_abap\_fli INTO TABLE @it\_sel.

"Using FINAL

SELECT \* FROM zdemo\_abap\_fli INTO TABLE @FINAL(it\_inline6).

**Inline Data Declarations**

|  |  |  |
| --- | --- | --- |
| **Context** | **Explicit Data Declaration** | **Inline Data Declaration** |
| Variables | DATA lv\_number\_of\_authors TYPE i.  lv\_number\_of\_authors = 2. | DATA(lv\_number\_of\_authors) = 2. |
| Variables | DATA lv\_var2.  lv\_var2 = lv\_var1. | DATA(lv\_var2) = lv\_var1. |
| Looping on Internal Tables | DATA ls\_work\_area LIKE LINE OF lt\_itab.  LOOP AT lt\_itab INTO ls\_work\_area.  "Processing steps  ENDLOOP. | LOOP AT lt\_itab INTO DATA(ls\_work\_area).  "Processing steps  ENDLOOP. |
| Looping on Internal Tables | FIELD-SYMBOLS <fs\_row> LIKE LINE OF lt\_itab.  LOOP AT lt\_itab ASSIGNING <fs\_row>.  "Processing steps  ENDLOOP. | LOOP AT lt\_itab ASSIGNING FIELD-SYMBOL(<fs\_row>).  "Processing steps  ENDLOOP. |
| Temporary Internal Table | DATA: lt\_tmp\_table LIKE lt\_itab.  lt\_tmp\_table = lt\_itab. | DATA(lt\_tmp\_table) = lt\_itab. |
| Reading Internal Table | DATA ls\_work\_area LIKE LINE OF lt\_itab.  READ TABLE lt\_itab INTO ls\_work\_area INDEX 1.  IF sy-subrc EQ 0.  "Code for successful READ  ELSE.  "Code for failed READ  ENDIF. | READ TABLE lt\_itab INTO DATA(ls\_work\_area) INDEX 1.  IF sy-subrc EQ 0.  "Code for successful READ  ELSE.  "Code for failed READ  ENDIF. |
| Reading Internal Table | FIELD-SYMBOLS <fs\_row> like LINE OF lt\_itab.  READ TABLE lt\_itab ASSIGNING <fs\_row> INDEX 1.  IF sy-subrc EQ 0.  "Code for successful READ  ELSE.  "Code for failed READ  ENDIF. | READ TABLE lt\_itab ASSIGNING FIELD-SYMBOL(<fs\_row>) INDEX 1.  IF sy-subrc EQ 0.  "Code for successful READ  ELSE.  "Code for failed READ  ENDIF. |
| Reading Internal Table | Note:  Old READ statement is used for demonstration.  New READ syntax can be used as shown here. | TRY.  DATA(ls\_work\_area1) = lt\_itab[ 1 ].  "Code for successful READ  CATCH cx\_sy\_itab\_line\_not\_found.  "Code for failed READ  ENDTRY. |
| SELECT Query | TYPES:  BEGIN OF ty\_fls,  carrid TYPE sflight-carrid,  connid TYPE sflight-connid,  fldate TYPE sflight-fldate,  END OF ty\_fls.  DATA lt\_flights TYPE STANDARD TABLE OF ty\_fls.  SELECT carrid connid fldate  FROM sflight  INTO TABLE lt\_flights. | SELECT carrid, connid, fldate  FROM sflight  INTO TABLE @DATA(lt\_flights). |
| Method Call | DATA lv\_result TYPE char5.  sample\_method(  EXPORTING iv\_input = iv\_test  IMPORTING iv\_result = lv\_result ). | sample\_method(  EXPORTING iv\_input = iv\_test  IMPORTING iv\_result = DATA(lv\_result) ). |
| Support Variable | DATA lv\_count TYPE i.  FIND 'J' IN 'JAGDISH' MATCH COUNT lv\_count. | FIND 'J' IN 'JAGDISH' MATCH COUNT DATA(lv\_count). |
| Objects | DATA lo\_obj TYPE REF TO lcl\_abap.  CREATE OBJECT lo\_obj. | DATA(lo\_obj) = NEW lcl\_abap( ). |
| Factory ALV, Object, and Exceptions | DATA :    lt\_out\_tab  TYPE STANDARD TABLE OF scarr,    lo\_table    TYPE REF TO cl\_salv\_table,    lo\_function TYPE REF TO cl\_salv\_functions\_list,    lx\_msg      TYPE REF TO cx\_salv\_msg,    lx\_alv      TYPE REF TO cx\_salv\_not\_found.  SELECT \* FROM scarr INTO TABLE lt\_out\_tab.  TRY.      cl\_salv\_table=>factory(        IMPORTING r\_salv\_table = lo\_table        CHANGING  t\_table      = lt\_out\_tab ).      lo\_function = lo\_table->get\_functions( ).      lo\_function->set\_all( ).    CATCH   cx\_salv\_msg INTO lx\_msg.      "Handle Error    CATCH cx\_salv\_not\_found INTO lx\_alv.      "Handle Error  ENDTRY. | SELECT \* FROM scarr INTO TABLE @DATA(lt\_out\_tab).  TRY.      cl\_salv\_table=>factory(        IMPORTING r\_salv\_table = DATA(lo\_table)        CHANGING  t\_table      = lt\_out\_tab ).      DATA(lo\_function) = lo\_table->get\_functions( ).      lo\_function->set\_all( ).    CATCH cx\_salv\_msg INTO DATA(lx\_msg).      "Handle Error    CATCH cx\_salv\_not\_found INTO DATA(lx\_alv).      "Handle Error  ENDTRY. |

**6. Populating Internal Tables**

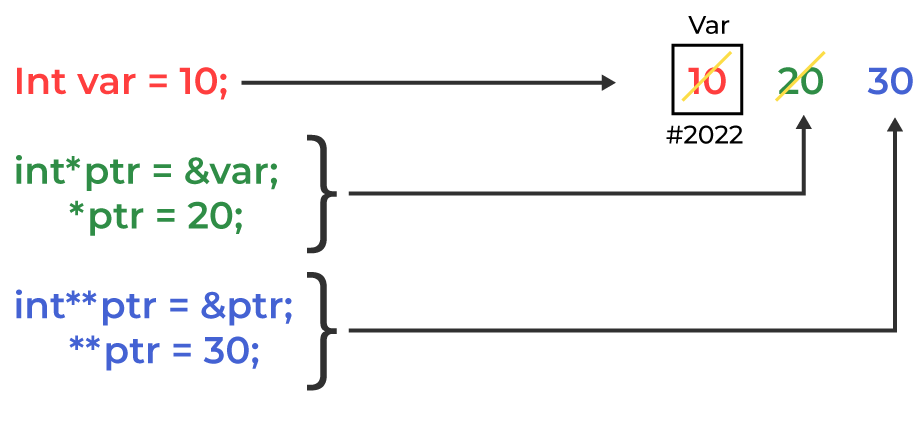
💡 Trước tiên cần tìm hiểu FIELD SYMBOLs và DATA References (Sẽ được giới thiệu trong Dynamic Programming).

💡 Field symbols and data references allow u to access data objects dynamically in ABAP programs. Unlike static access to a data object, where u need to specify the name of the object, field symbols and data references allow u to access and pass data objects whose name and attributes u do not know until runtime.

💡 Field Symbols là các con trỏ đặc biệt đã được dereference trong ABAP, cho phép truy cập "tượng trưng" vào một đối tượng dữ liệu hiện tại.

💡 Khi gán một đối tượng dữ liệu cho một field symbol, mọi truy cập vào field symbol sẽ thao tác trên dữ liệu của đối tượng đó. Điều này giúp giảm thiểu việc sao chép dữ liệu vào một work area và tối ưu hóa hiệu suất.

💡 Khi đánh địa chỉ một FIELD SYMBOL, hệ thống sẽ làm việc với nội dung của đối tượng dữ liệu được gán cho nó chứ không phải với nội dung của chính field symbol.



Int\*ptr = &var => ptr = #2022 / \*ptr = 10

💡 DATA References tương tự như con trỏ trong C++, chỉ khác là không thể thao tác di chuyển ++ hoặc – trên bộ nhớ. Có nhiều ý kiến cho rằng Reference sẽ lưu địa chỉ của vùng nhớ được tham chiếu (như ptr), nhưng khi debugging thì thấy rằng nó đại diện cho giá trọ thay vì đại chỉ (giống \*ptr), vậy có thể hiểu nó là con trỏ nhưng cắt bỏ các tính năng quản lý và thao tác trên địa chỉ.

💡 Khi truy cập vào biến tham chiếu, chúng ta đang xử lý chính tham chiếu dữ liệu đó. Để truy cập nội dung của đối tượng dữ liệu mà tham chiếu dữ liệu đang trỏ tới thì phải hủy đăng ký (dereference) nó.

**6.1 Copying Internal Tables**

💡 Sao chép thông tin từ một internal table đên một internal table khác (cùng kiểu) mà không dùng constructor, các dữ liệu trong itab sẽ bị xóa trước khi sao chép.

💡 Dùng cú pháp Inline để tránh phải khái báo một internal table khác với các kiểu dữ liệu tương ứng (để inline lo).

💡 itab4 sau khi được gán thì sẽ không thể thay đổi dữ liệu

itab = itab2.

DATA(itab3) = itab.

FINAL(itab4) = itab.

💡 Quy tắt gán / chuyển đổi trong internal table:

* Chỉ internal table mới gán được cho internal table. Việc gán chỉ phụ thuộc vào kiểu dòng (line type) chứ không phụ thuộc vào loại bảng (table type), khóa bảng (table key) hay số lượng dòng.
* Line type giữa các tables phải đảm bảo Compatible (Tương Thích) hoặc Convertible (Có Thể Chuyển Đổi) 🡪 Cùng kiểu hoặc là có quy tắc chuyển đổi giữa các kiểu (vd chuyển string 🡪 int nếu string là chuỗi số).
* Quy trình gán: Xóa các dòng của table đích 🡪 Tạo dòng mới trong table đích và sao chép từng dòng từ table nguồn sang 🡪 Lưu trữ theo loại table: sorted table (Nội dung sẽ tự động được sắp xếp), hashed table (Nội dung sẽ được lưu trữ theo thuật toán băm).
* Chuyển đổi từng dòng cũng giống như chuyển đổi riêng lẽ từng biến, phạm vi giá trị của record nhận phải đủ để chứa nội dung được nhận.
* Khi gán một internal table rỗng cho một internal table đã có dữ liệu, bảng đích sẽ được khởi tạo lại giống như lệnh CLEAR, giải phóng bộ nhớ trừ dung lượng bộ nhớ ban đầu.
* Uncatchable Exceptions có thể xảy ra khi gán các record bị trùng vào table đích có Unique Primary hoặc Secondary Keys. Ngoài ra có thể gặp catchable exceptions khi gán các table có elementary line types, khi bị **ném ngoại lệ vô mặt** thì dữ liệu trong table đích sẽ chứa tất cả các dòng trước khi gặp dòng gây ngoại lệ.

\* Định nghĩa kiểu dữ liệu

TYPES:

  text       TYPE c LENGTH 20.

\* Khai báo biến

DATA:

  text\_tab   TYPE STANDARD TABLE OF text WITH EMPTY KEY,

  string\_tab TYPE SORTED TABLE OF string WITH NON-UNIQUE KEY table\_line.

\* Gán giá trị cho bảng chuẩn

text\_tab = VALUE #(

  ( 'Perl' ) ( 'C' ) ( 'Visual Basic' ) ( 'Java' )

  ( 'ABAP' ) ( 'Pascal' ) ( 'Python' )  ( 'Lisp' ) ).

\* Gán standard table sang sorted table

string\_tab = text\_tab.

\* Hiển thị kết quả

cl\_demo\_output=>new( )->write( text\_tab )->display( string\_tab ).

**6.2 Using INSERT and APPEND Statements to Populate Internal Tables**

💡 APPEND:

* Luôn thêm dòng ở cuối table.
* Vì các dòng được quản lý bởi một chỉ số (index). Khi câu lệnh này được sử dụng, biến hệ thống sy-tabix được đặt thành index của dòng mới vừa được thêm vào. sy-tabix luôn được đặt theo primary table index.
* Không được dùng cho hashed table.
* Đối với sorted table, các dòng chỉ được thêm vào nếu chúng khớp với thứ tự được sắp xếp, nếu primary key là unique thì không được trùng. Vì vậy trường hợp này nên dùng INSERT cho so9rrted table.

💡 INSERT:

* Có thể dùng để thêm các dòng vào một vị trí cụ thể thông qua chỉ mục. Lúc này các dòng bên dưới sẽ tự dịch xuống 1 chỉ số.
* Kèm index chỉ dùng được với table có index, dòng sẽ được chèn vào ngay trước line thứ n của table trong primary table index.
* Nếu không chỉ định index thì sẽ thêm vào cuối bảng đối với standard table. Đối với Sorted table thì tự sort theo thứ tự và tự chèn vào vị trí hợp lý. Đối với hashed table thì tự chèn vào vị trí hợp lý theo thuật toán băm.
* sy-tabix sẽ không được set.
* Recommentdation sử dụng ví nó cover được tất cả trường hợp. Cần lưu ý khi đổi loại table/key types và khi dùng lộn xọn với APPEND.

💡 Adding a line to an internal table. The example shows both a structure that is created using the VALUE operator as well as an existing structure that is added.

APPEND VALUE #( comp1 = a comp2 = b ... ) TO itab.

APPEND lv\_struc TO itab.

INSERT VALUE #( comp1 = a comp2 = b ... ) INTO TABLE itab.

INSERT lv\_struc INTO TABLE itab.

💡 Adding an initial line to an internal table without providing any field values.

APPEND INITIAL LINE TO itab.

INSERT INITIAL LINE INTO TABLE itab.

💡 Adding a line and assigning the added line to a field symbol or data reference variable.

"When inserting single lines, u can specify the optional additions

"ASSIGNING and REFERENCE INTO. If the insertion is successful, the

"line is assigned to a field symbol or a data reference variable.

"The targets can also be created inline.

APPEND VALUE #( comp1 = a comp2 = b ... ) TO itab ASSIGNING FIELD-SYMBOL(<fs>).

APPEND INITIAL LINE TO itab ASSIGNING <fs>.

INSERT INITIAL LINE INTO TABLE itab REFERENCE INTO DATA(dref).

💡 Adding all lines from another internal table.

APPEND LINES OF itab2 TO itab.

INSERT LINES OF itab2 INTO TABLE itab.

💡 Adding multiple lines from another internal table with a specified index range.

* Both FROM and TO are not mandatory in one statement. it is possible to use only one of them.
* If use only ...
  + FROM, all lines up to the last table entry are respected.
  + TO, all lines starting with the first table entry are respected.

"i1/i2 represent integer values

APPEND LINES OF itab2 FROM i1 TO i2 TO itab.

APPEND LINES OF itab2 FROM i1 TO itab.

APPEND LINES OF itab2 TO i2 TO itab.

INSERT LINES OF itab2 FROM i1 TO i2 INTO itab.

💡 Inserting one line or multiple lines from another internal table at a specific position. FROM and TO can be used here, too.

INSERT lv\_struc INTO itab2 INDEX i.

INSERT LINES OF itab2 INTO itab INDEX i.

**6.3 Creating and Populating Internal Tables Using Constructor Expressions**

**6.3.1 VALUE operator**

💡 Khi sử dụng toán tử VALUE để tạo một internal table, cần phải sử dụng cặp dấu ngoặc đơn ( ... ) để bao quanh các giá trị của từng dòng trong bảng.

DATA: itab TYPE TABLE OF some\_structure.

itab = VALUE #( ( field1 = 'value1' field2 = 'value2' )

                ( field1 = 'value3' field2 = 'value4' ) ).

💡 Ký tự # được sử dụng trong ABAP để biểu thị rằng loại dữ liệu (data type) của dòng hoặc biến có thể được suy ra từ ngữ cảnh hiện tại. Không cần chỉ định loại dữ liệu cho itab vì nó đã được định nghĩa trước đó. Ký tự # giúp ABAP tự động nhận biết loại dữ liệu của itab từ ngữ cảnh.

DATA: itab TYPE TABLE OF some\_structure.

itab = VALUE #( ( field1 = 'value1' field2 = 'value2' )

                ( field1 = 'value3' field2 = 'value4' ) ).

💡 Khi gán một internal table mới cho một internal table hiện có, nội dung hiện có của internal table sẽ bị xóa và thay thế bằng nội dung mới. Sau phép gán thứ hai, itab chỉ chứa dòng mới ( field1 = 'value3' field2 = 'value4' ), và dòng trước đó bị xóa.

DATA: itab TYPE TABLE OF some\_structure.

itab = VALUE #( ( field1 = 'value1' field2 = 'value2' ) ). " Bảng có 1 dòng

itab = VALUE #( ( field1 = 'value3' field2 = 'value4' ) ). " Nội dung cũ bị xóa, chỉ còn dòng mới

**Creating an internal table by inline declaration and adding lines with a constructor expression.**

"Internal table type

TYPES it\_type LIKE itab.

"Inline declaration

"The # character would not be possible here since the line type

"cannot be derived from the context.

DATA(it\_in) = VALUE it\_type( ( comp1 = a comp2 = b ... )

                             ( comp1 = c comp2 = d ...  ) ).

"Creating string tables

DATA(str\_tab\_a) = VALUE string\_table( ( `Hallo` ) ( `World` ) ).

DATA(str\_tab\_b) = VALUE string\_table( ).

"In the previous statement, the internal table is declared

"inline, however, no content, no table lines are provided.

"This means that an initial string table was created. This

"way, the statement has the same effect as the following

"statement.

DATA str\_tab\_c TYPE string\_table.

💡 When use the above assignments (itab = ...), the internal table is initialized and the existing content is deleted. To add new lines without deleting existing content, use the BASE addition.

itab = VALUE #( BASE itab ( comp1 = a comp2 = b ... )

                          ( comp1 = c comp2 = d ... ) ).

💡 Without the BASE addition, the existing content is deleted. It is assumed that the line type of the source table is compatible with that of the target table.

itab = VALUE #( ( comp1 = a comp2 = b ...)

                ( comp1 = a comp2 = b ...)

                ( LINES OF itab2 )

                ... ). " --> itab chỉ chứa các dòng của itab2

**6.3.2 CORRESPONDING operator**

💡 Copying the content of another internal table using the CORRESPONDING operator. Trước khi sao chép thì dữ liệu hiện tại của table sẽ bị xóa.

💡 Có thể dùng cú pháp MOVE-CORRESPONDING tường minh hơn. Vd đối với table có kiểu dòng không tương thích, tuy nhiên nếu các kiểu dòng có tương thích thì vẫn chạy bình thường.

itab = CORRESPONDING #( itab3 ).

MOVE-CORRESPONDING itab3 TO itab.

💡 Sao chép nội dung và giữ lại nội dung hiện có:

itab = CORRESPONDING #( BASE ( itab ) itab3 ).

MOVE-CORRESPONDING itab3 TO itab KEEPING TARGET LINES.

💡 Assigning components using mapping relationships.

💡 Sử dụng ánh xạ để chỉ định rằng thành phần c của itab3 sẽ được gán cho thành phần a của itab và thành phần d của itab3 sẽ được gán cho thành phần b của itab:

itab = CORRESPONDING #( itab3 MAPPING a = c b = d ).

💡 Excluding components from the assignment using the EXCEPT addition to the CORRESPONDING operator.

💡 EXCEPT giúp loại bỏ các thành phần cụ thể khỏi phép gán giữa bảng nguồn và bảng đích, đặc biệt hữu ích khi có các thành phần trùng tên nhưng không tương thích hoặc không convertable.

💡 Thay vì liệt kê từng thành phần cụ thể, có thể sử dụng EXCEPT\* để loại bỏ tất cả các thành phần không được đề cập trong ánh xạ trước đó.

💡 Nếu sử dụng EXCEPT\* mà không có từ khóa MAPPING để ánh xạ các thành phần, tất cả các thành phần trong bảng đích sẽ giữ nguyên giá trị ban đầu (initial).

TYPES: BEGIN OF source\_structure,

         a TYPE i,

         b TYPE i,

         c TYPE i,

         d TYPE i,

         e TYPE i,

       END OF source\_structure.

TYPES: BEGIN OF target\_structure,

         a TYPE i,

         b TYPE i,

         c TYPE i,

         d TYPE i,

       END OF target\_structure.

DATA: itab1 TYPE TABLE OF source\_structure,

      itab2 TYPE TABLE OF target\_structure.

itab2 = CORRESPONDING #( itab1 EXCEPT e ).

💡 Preventing runtime errors when duplicate lines are assigned

💡 DISCARDING DUPLICATES là một bổ sung của toán tử CORRESPONDING giúp tránh lỗi runtime khi gán các dòng trùng lặp vào một internal table chỉ chấp nhận các khóa duy nhất. Có thể dùng kết hợp với MAPPING.

itab = CORRESPONDING #( itab2 DISCARDING DUPLICATES ).

itab = CORRESPONDING #( itab2 MAPPING a = c b = d DISCARDING DUPLICATES ).

Copying data from deep internal tables. 🡪 Internal table inside an internal table.

itab\_nested2 = CORRESPONDING #( DEEP itab\_nested1 ).

itab\_nested2 = CORRESPONDING #( DEEP BASE ( itab\_nested2 ) itab\_nested1 ).

MOVE-CORRESPONDING itab\_nested1 TO itab\_nested2 EXPANDING NESTED TABLES.

MOVE-CORRESPONDING itab\_nested1 TO itab\_nested2 EXPANDING NESTED TABLES KEEPING TARGET LINES.

**6.3.3 FILTER Operator**

💡 Sao chép table và lọc các dòng không đáp ứng điều kiệu. Trước khi chuyển hàng (sao chép) từ chỗ này sang chỗ kia thì phải kiểm tra xem hàng có thỏa mãn một điều kiệu nào đó không.

💡 FILTER sẽ tạo một internal table mới theo một kiểu được chỉ định (có thể là kiểu bảng cụ thể hoặc ký tự # để suy ra từ kiểu toán hạng trước dấu ngoặc đầu tiên).

💡 Các dòng của internal table mới được lấy từ internal table hiện có dựa trên các điều kiện được chỉ định trong câu lệnh WHERE.

💡 Ví dụ 1: Lọc đơn giản: Giả sử chúng ta có một internal table chứa các hãng hàng không và chúng ta muốn lọc ra các hãng hàng không có mã carrid là 'AA' hoặc 'LH'.

DATA: itab TYPE TABLE OF scarr,

      itab\_filtered TYPE TABLE OF scarr.

" itab chứa dữ liệu từ bảng scarr.

SELECT \* FROM scarr INTO TABLE @itab.

" itab\_filtered sẽ chỉ chứa các dòng từ itab mà carrid là 'AA' hoặc 'LH'.

itab\_filtered = FILTER #( itab WHERE carrid = 'AA' OR carrid = 'LH' ).

💡 Ví dụ 2: Sử dụng FILTER với Bảng Lọc: Nếu có một bảng lọc (ftab) chứa danh sách các giá trị muốn lọc, có thể sử dụng nó cùng với toán tử FILTER.

DATA: itab TYPE TABLE OF scarr,

      itab\_filtered TYPE TABLE OF scarr,

      filter TYPE SORTED TABLE OF scarr-carrid WITH UNIQUE KEY table\_line.

" itab chứa dữ liệu từ bảng scarr.

SELECT \* FROM scarr INTO TABLE @itab.

" filter chứa danh sách các carrid để lọc.

filter = VALUE #( ( 'AA' ) ( 'LH' ) ( 'UA' ) ).

" itab\_filtered sẽ chỉ chứa các dòng từ itab mà carrid khớp với bất kỳ giá trị nào trong filter.

itab\_filtered = FILTER #( itab IN filter WHERE carrid = table\_line ).

💡 Lưu ý rằng kiểu bảng của internal table hiện có phải có thể chuyển đổi thành kiểu đích được chỉ định.

" itab là internal table hiện có. itab\_filtered là internal table mới được tạo.

" FILTER sao chép các dòng từ itab vào itab\_filtered nhưng chỉ các dòng thỏa mãn điều kiện WHERE field1 = 'value'.

DATA: itab TYPE TABLE OF some\_structure,

      itab\_filtered TYPE TABLE OF some\_structure.

itab\_filtered = FILTER #(

    itab

    WHERE field1 = 'value'

).

💡 Các phần mở rộng của FILTER Operator:

|  |  |
| --- | --- |
| **Phần mở rộng** | **Chi tiết** |
| USING KEY | Chỉ định khóa bảng được sử dụng để đánh giá điều kiện WHERE. Nếu internal table không có khóa này, phải có khóa bảng phụ được chỉ định sau USING KEY. |
| EXCEPT | Sử dụng các dòng không thỏa mãn điều kiện WHERE. Nếu không có EXCEPT, sử dụng các dòng thỏa mãn điều kiện. |

"FILTER on conditions based on single values

"Assumption: The component num is of type i.

DATA itab1 TYPE SORTED TABLE OF struc WITH NON-UNIQUE KEY num.

DATA itab2 TYPE STANDARD TABLE OF struc WITH NON-UNIQUE SORTED KEY sec\_key COMPONENTS num.

DATA itab3 TYPE HASHED TABLE OF struc WITH UNIQUE KEY num.

"The lines meeting the condition are respected.

"Note: The source table must have at least one sorted or hashed key.

"Here, the primary key is used

DATA(f1) = FILTER #( itab1 WHERE num >= 3 ).

"USING KEY primary\_key explicitly specified; same as above

DATA(f2) = FILTER #( itab1 USING KEY primary\_key WHERE num >= 3 ).

"EXCEPT addition

DATA(f3) = FILTER #( itab1 EXCEPT WHERE num >= 3 ).

DATA(f4) = FILTER #( itab1 EXCEPT USING KEY primary\_key WHERE num >= 3 ).

"Secondary table key specified after USING KEY

DATA(f5) = FILTER #( itab2 USING KEY sec\_key WHERE num >= 4 ).

DATA(f6) = FILTER #( itab2 EXCEPT USING KEY sec\_key WHERE num >= 3 ).

"Note: In case of a hash key, exactly one comparison expression for each key component is allowed;

"only = as comparison operator possible.

DATA(f7) = FILTER #( itab3 WHERE num = 3 ).

"Using a filter table

"In the WHERE condition, the columns of source and filter table are compared.

"Those lines in the source table are used for which at least one line in the filter

"table meets the condition. EXCEPT and USING KEY are also possible.

DATA filter\_tab1 TYPE SORTED TABLE OF i

  WITH NON-UNIQUE KEY table\_line.

DATA filter\_tab2 TYPE STANDARD TABLE OF i

  WITH EMPTY KEY

  WITH UNIQUE SORTED KEY line COMPONENTS table\_line.

DATA(f8) = FILTER #( itab1 IN filter\_tab1 WHERE num = table\_line ).

"EXCEPT addition

DATA(f9) = FILTER #( itab1 EXCEPT IN filter\_tab1 WHERE num = table\_line ).

"USING KEY is specified for the filter table

DATA(f10) = FILTER #( itab2 IN filter\_tab2 USING KEY line WHERE num = table\_line ).

"USING KEY is specified for the source table, including EXCEPT

DATA(f11) = FILTER #( itab2 USING KEY sec\_key EXCEPT IN filter\_tab2 WHERE num = table\_line ).

**6.3.4 NEW Operator**

💡 Toán tử NEW trong ABAP được sử dụng để tạo anonymous data objects. Điều này đặc biệt hữu ích để tạo và điều khiển các internal table và structure mà không cần định nghĩa chúng một cách rõ ràng từ trước.

Cú pháp và Sử dụng

* **<dref>**: Đây là một tham chiếu dữ liệu (dref), nắm giữ tham chiếu đến đối tượng dữ liệu ẩn vừa được tạo.
* **<type>**: Xác định loại của đối tượng dữ liệu đang được tạo, ví dụ như một cấu trúc (s trong ví dụ) hoặc một loại bảng (tab\_type trong ví dụ).

DATA(<dref>) = NEW <type>( ( ... ) ( ... ) ).

💡 Vd đầy đủ:

TYPES: BEGIN OF s,

          a TYPE c LENGTH 3,

          b TYPE i,

        END OF s,

        tab\_type TYPE TABLE OF s WITH EMPTY KEY.

"Creating and populating an anonymous data object

DATA(dref\_tab) = NEW tab\_type( ( a = 'aaa' b = 1 )

                               ( a = 'bbb' b = 2 ) ).

"Access by derefencing

DATA(copy\_deref\_itab) = dref\_tab->\*.    " Sao chép toàn bộ anonymous internal tables

DATA(read\_line) = dref\_tab->\*[ 2 ].     " Đọc dòng thứ hai từ anonymous internal tables

DATA(read\_comp) = dref\_tab->\*[ 1 ]-a.   " Đọc thành phần 'a' từ dòng thứ nhất của anonymous internal tables

dref\_tab->\*[ 1 ]-a = 'zzz'.             " Thay đổi giá trị của thành phần 'a' trong dòng thứ nhất

ASSERT dref\_tab->\*[ 1 ]-a = 'zzz'.      " Kiểm tra rằng giá trị của thành phần 'a' trong dòng thứ nhất là 'zzz'

INSERT VALUE s( a = 'yyy' b = 3 ) INTO TABLE dref\_tab->\*.   " Chèn một dòng mới vào anonymous internal tables

*Toán tử new để tạo Anonymous có rất nhiều trò, nhưng khả năng là ít dùng đến (NEW Initial Value for All Types, NEW Single Value for All Data Types, NEW Structures, NEW Internal TablesNEW Classes) 🡪 Đề cập sau*

**7. Reading Single Lines from Internal Tables**

There are three different ways to specify the line to read:

* by index (only index tables)
* by table keys (only tables for which a key is defined)
* by free key

💡 Có thể truy cập các dòng riêng lẽ của internal table bằng các dùng READ TABLE hoặc table expressions, hoặc đọc tuần tự bằng việc sử dụng LOOP AT. Cả 2 trường hợp đều có thể sử dụng các câu lệnh bổ sung:

* INTO sao chép nội dung của line vào trong data object.
* ASSIGNING cho phép gán dòng đọc được cho field-symbol, cho phép dòng đó được đánh địa chỉ trực tiếp, không có sự copy nào diễn ra cả.
* REFERENCE INTO tham chiếu đến dòng được đọc.

💡 Rule: Choose appropriate output behavior:

* Sao chép vào work area nếu loại dòng là các dòng có cấu trúc đơn giản, ít thành phần dữ liệu và không cần thay đổi read line.
* Gán cho một field symbol nếu loại dòng là các dòng có cấu trúc phức tạp hơn, có nhiều thành phần dữ liệu hoặc các dòng có thành phần dữ liệu lồng nhau, chẳng hạn như các internal table trong internal table và read line cần được thay đổi.
* Đặt một tham chiếu dữ liệu nếu loại dòng là các dòng có cấu trúc phức tạp hơn, có nhiều thành phần dữ liệu hoặc các dòng có thành phần dữ liệu lồng nhau, chẳng hạn như các internal table trong internal table và cần truyền tham chiếu đến read line.

**7.1 Determining the Target Area when Reading Single Lines**

💡 Khi đọc một dòng từ internal table vào một data object sử dụng từ khóa INTO, sau khi sao chép, line được tìm thấy sẽ tồn tại riêng biệt trong internal table và data object. Điều này có nghĩa là nếu thay đổi data object hoặc line trong table, thay đổi này sẽ không ảnh hưởng đến đối tượng kia. Tuy nhiên, có thể sửa đổi table line đã sao chép và sử dụng lệnh MODIFY để cập nhật table dựa trên table line đã sửa đổi.

💡 Sao chép toàn bộ dòng vào một đối tượng dữ liệu:

DATA: itab TYPE TABLE OF scarr,

      wa   TYPE scarr.

" đọc dòng từ bảng itab với khóa carrid bằng 'LH' và sao chép toàn bộ dòng vào wa

READ TABLE itab WITH KEY carrid = 'LH' INTO wa.

" wa có cùng kiểu với cấu trúc của itab

IF sy-subrc = 0.

  WRITE: / wa-carrid, wa-carrname.

ENDIF.

💡 Sử dụng inline declaration:

DATA: itab TYPE TABLE OF scarr.

" Dòng được đọc và sao chép vào wa\_inl.

READ TABLE itab WITH KEY carrid = 'LH' INTO DATA(wa\_inl).

IF sy-subrc = 0.

  WRITE: / wa\_inl-carrid, wa\_inl-carrname.

ENDIF.

💡 Chỉ sao chép các thành phần cụ thể:

DATA: itab TYPE TABLE OF scarr,

      wa   TYPE scarr.

" Chỉ các thành phần carrid và carrname được sao chép vào wa

READ TABLE itab WITH KEY carrid = 'LH' INTO wa TRANSPORTING carrid carrname.

" Các thành phần khác của wa không bị thay đổi

IF sy-subrc = 0.

  WRITE: / wa-carrid, wa-carrname.

ENDIF.

💡 Sao chép là tạo bản sao để làm việc, lúc nào sẽ tạo 2 đối tượng riêng với dữ liệu giống nhau. Còn nếu dùng FIELD-SYMBOL (pointer) thì chỉ tham chiếu đến dữ liệu trong table 🡪 Không có sự sao chép nào, thay đổi thông qua FIELD-SYMBOL tức là thay đổi dữ liệu trong table.

💡 FIELD-SYMBOL phải có kiểu tương thích với cấu trúc của bảng. Không dùng được với TRANSPORTING.

READ TABLE itab ASSIGNING <fs1> ...         "The field symbol must have an appropriate type.

READ TABLE itab ASSIGNING FIELD-SYMBOL(<fs2>) ...   "The field symbol is created inline.

" Ví dụ chi tiết

" Cú pháp tường minh

DATA: itab TYPE TABLE OF some\_structure,

      wa   TYPE some\_structure.

FIELD-SYMBOLS: <fs> TYPE some\_structure.

READ TABLE itab ASSIGNING <fs>.

IF sy-subrc = 0.

  <fs>-field1 = 'New Value'.

ENDIF.

" Inline declaration

DATA: itab TYPE TABLE OF some\_structure.

READ TABLE itab ASSIGNING FIELD-SYMBOL(<fs\_inl>).

IF sy-subrc = 0.

  <fs\_inl>-field1 = 'New Value'.

ENDIF.

" Inline declaration with key

DATA: itab TYPE TABLE OF scarr.

READ TABLE itab WITH KEY carrid = 'LH' ASSIGNING FIELD-SYMBOL(<fs\_inl>).

IF sy-subrc = 0.

  <fs\_inl>-carrname = 'New Name'.

ENDIF.

💡 REFERENCE INTO: đặt một tham chiếu dữ liệu đến dòng đọc được. không có sự copy nào diễn ra, không dùng cùng với TRANSPORTING. Nếu muốn sờ đến dữ liệu để làm việc thì phải dereference the data reference (toán tử tham chiếu ->\*).

READ TABLE itab REFERENCE INTO dref ...

READ TABLE itab REFERENCE INTO DATA(dref\_inl) ...

💡 Hiệu suất có thể bị ảnh hưởng bởi việc sao chép dữ liệu hoặc không. Việc sao chép có thể tốn kém về mặt thời gian và tài nguyên, đặc biệt khi bảng có nhiều cột hoặc các thành phần lồng nhau.

💡 Dùng TRANSPORTING Để hạn chế số lượng trường được sao chép và giảm thiểu chi phí sao chép dữ liệu.

**7.2 Reading a Single Line by Index**

💡 Sử dụng lệnh READ TABLE với index. Câu lệnh trên đọc dòng thứ i từ internal table itab và lưu vào work area wa.

💡 Có thể chỉ định key để sử dụng. Nếu table có secondary sorted key, có thể chỉ định khóa này để tìm dòng cần đọc.

READ TABLE itab INTO wa INDEX i.

READ TABLE itab INTO wa INDEX i USING KEY primary\_key.

💡 Có thể dùng table expression ở dạng inline. Khi sờ phải dòng not found có thể gây lỗi runtime, bồ xung khối TRY-CATCH để bắt lỗi:  
"In the examples, integer values are specified for the index.

DATA(lv1) = itab[ 1 ].

TRY.

  DATA(lv2) = itab[ 2 ].

  CATCH cx\_sy\_itab\_line\_not\_found.

  ...

ENDTRY.

DATA(lv3) = itab[ KEY primary\_key INDEX 3 ].

"Copying a table line via table expression and embedding in constructor expression

DATA(lv4) = VALUE #( itab[ 4 ] ).

"Reading into data reference variable using the REF operator

DATA(lv5\_ref) = REF #( itab[ 5 ] ).

💡 Có thể sử dụng key trong table expression để đọc line. Điều này có thể ảnh hưởng đến cách line được tìm kiếm và hiệu suất của việc đọc dữ liệu.

DATA(lv3) = itab[ KEY primary\_key INDEX 3 ].

💡 Nhúng dòng copy được vào biểu thứ khởi tạo giá trị, dùng inline:

DATA(lv4) = VALUE #( itab[ 4 ] ).

💡 Nhúng dòng copy được vào toán tử tham chiếu:

DATA(lv5\_ref) = REF #( itab[ 5 ] ).

💡 Xử lý dòng không tồn tại mà không ném lỗi: Để tránh lỗi khi đọc dòng không tồn tại, có thể sử dụng OPTIONAL hoặc DEFAULT.

" OPTIONAL: Trả về dòng với các giá trị khởi tạo nếu dòng không tồn tại.

DATA(line1) = VALUE #( itab[ 6 ] OPTIONAL ).

" DEFAULT: Trả về dòng mặc định nếu dòng không tồn tại.

DATA(line2) = VALUE #( itab[ 7 ] DEFAULT itab[ 1 ]  ).

" Nếu dòng thứ 7 không tồn tại, dòng đầu tiên itab[ 1 ] sẽ được trả về.

💡 Chi tiết về table expression:

Cú pháp tổng quát:

... itab[ itab\_line\_index ][-comp|[ ... ]|->comp] ...

" itab: Tên của internal table.

" itab\_line: Chỉ định dòng trong internal table.

" -comp: Truy cập vào thành phần của dòng cấu trúc.

" [ ... ]: Liên kết nhiều table expression.

" ->comp: Truy cập thành phần của đối tượng được tham chiếu bởi biểu thức trước đó.

Ví dụ đầy đủ:

TYPES: BEGIN OF nested\_type,

         value TYPE i,

       END OF nested\_type.

TYPES: BEGIN OF line\_type,

         id           TYPE i,

         name         TYPE string,

         nested\_table TYPE TABLE OF nested\_type,

       END OF line\_type.

" itab có các dòng là các cấu trúc chứa các thành phần khác nhau.

" Mỗi dòng của itab chứa một bảng lồng khác (nested\_table).

DATA: itab TYPE TABLE OF line\_type,

      nested\_tab TYPE TABLE OF nested\_type,

      wa   TYPE line\_type.

" Tạo dữ liệu mẫu cho internal table

nested\_tab = VALUE #( ( value = 10 ) ( value = 20 ) ( value = 30 ) ).

wa = VALUE #( id = 1 name = 'First' nested\_table = nested\_tab ).

APPEND wa TO itab.

nested\_tab = VALUE #( ( value = 40 ) ( value = 50 ) ( value = 60 ) ).

wa = VALUE #( id = 2 name = 'Second' nested\_table = nested\_tab ).

APPEND wa TO itab.

nested\_tab = VALUE #( ( value = 70 ) ( value = 80 ) ( value = 90 ) ).

wa = VALUE #( id = 3 name = 'Third' nested\_table = nested\_tab ).

APPEND wa TO itab.

==============

" Sử dụng table expression

" Đọc dòng đầu tiên của itab và truy cập thành phần name

DATA(first\_name) = itab[ 1 ]-name.

" Đọc dòng thứ hai của itab và truy cập thành phần đầu tiên của nested\_table

DATA(first\_nested\_value) = itab[ 2 ]-nested\_table[ 1 ]-value.

" Đọc dòng thứ ba của itab, truy cập thành phần nested\_table và sau đó truy cập giá trị

DATA(third\_nested\_value) = itab[ 3 ]-nested\_table[ 3 ]-value.

" Đọc dòng thứ hai của itab và tạo tham chiếu đến nested\_table

DATA(nested\_ref) = REF #( itab[ 2 ]-nested\_table ).

" Đọc giá trị từ nested\_table của dòng thứ hai và truy cập thành phần value

DATA(second\_nested\_value) = itab[ 2 ]-nested\_table[ 2 ]-value.

" In ra các giá trị để kiểm tra

cl\_demo\_output=>display(

  VALUE #(

    ( first\_name )

    ( first\_nested\_value )

    ( third\_nested\_value )

    ( second\_nested\_value )

  )

).

💡 Đọc dòng bằng table expression và xử lý ngoại lệ:

" Ví dụ đọc một dòng từ internal table carrier\_tab dựa trên secondary key

" Sau đó sử dụng dữ liệu từ dòng đã đọc để lấy thông tin từ phương thức get\_spfli.

DATA carrier\_tab TYPE HASHED TABLE OF scarr

                 WITH UNIQUE KEY carrid

                 WITH NON-UNIQUE SORTED KEY name COMPONENTS carrname.

" Lấy tất cả các dòng từ bảng cơ sở dữ liệu scarr và lưu vào internal table carrier\_tab

SELECT \* FROM scarr INTO TABLE @carrier\_tab.

" nội dung của thành phần carrid của dòng trong bảng carrier\_tab được truyền vào phương thức

" get\_spfli. Table expression tìm dòng có carrname là 'United Airlines' và trả về carrid

TRY.

    FINAL(flight\_tab) = cl\_demo\_spfli=>get\_spfli(

      carrier\_tab[ KEY name

                   COMPONENTS carrname = 'United Airlines' ]-carrid ).

    cl\_demo\_output=>display( flight\_tab ).

CATCH cx\_sy\_itab\_line\_not\_found.

    cl\_demo\_output=>display( `Nothing found` ).

ENDTRY.

💡 Tránh chọn dòng nhiều lần:

DATA itab TYPE TABLE OF i.

itab = VALUE #( ( 3 ) ( 5 ) ).

"Bad example: Chọn dòng ba lần

itab[ table\_line = 3 ] =

  itab[ table\_line = 3 ] \* itab[ table\_line = 3 ].

"Good example: Dùng FIELD-SYMBOL

ASSIGN itab[ table\_line = 5 ] TO FIELD-SYMBOL(<fs>).

<fs> = <fs> \* <fs>.

**7.3 Reading a Single Line Using Table Keys**

💡 Lines can be read by explicitly specifying the table keys or the alias names, if any.

"Example internal table with primary and secondary table key and alias names

"Assumption: All components are of type i

DATA it TYPE SORTED TABLE OF struc

  WITH NON-UNIQUE KEY primary\_key ALIAS pk COMPONENTS a b

  WITH NON-UNIQUE SORTED KEY sec\_key ALIAS sk COMPONENTS c d.

"Table expressions

"Key must be fully specified

line = it[ KEY primary\_key COMPONENTS a = 1 b = 2 ].

"The addition COMPONENTS is optional; same as above

line = it[ KEY primary\_key a = 1 b = 2 ].

"Primary key alias

line = it[ KEY pk a = 1 b = 2 ].

"Secondary table key

line = it[ KEY sec\_key c = 3 d = 4 ].

"Secondary table key alias

line = it[ KEY sk c = 3 d = 4 ].

"READ TABLE statements

"Primary table key

READ TABLE it INTO wa WITH TABLE KEY primary\_key COMPONENTS a = 1 b = 2.

"Alias

READ TABLE it INTO wa WITH TABLE KEY pk COMPONENTS a = 1 b = 2.

"Secondary table key

READ TABLE it INTO wa WITH TABLE KEY sec\_key COMPONENTS c = 3 d = 4.

"Alias

READ TABLE it INTO wa WITH TABLE KEY sk COMPONENTS c = 3 d = 4.

"Reading a line based on keys specified in a work area

"Work area containing primary and secondary table key values; the line type

"must be compatible to the internal table

DATA(pr\_keys) = VALUE struc( a = 1 b = 2 ).

DATA(sec\_keys) = VALUE struc( c = 3 d = 4 ).

READ TABLE it FROM pr\_keys INTO wa.

"If USING KEY is not specified, the primary table key is used.

"If it is used, the specified table key is used.

READ TABLE it FROM pr\_keys USING KEY primary\_key INTO wa.

READ TABLE it FROM sec\_keys USING KEY sec\_key INTO wa.

"Alias

READ TABLE it FROM sec\_keys USING KEY sk INTO wa.

**7.4 Reading a Single Line Using a Free Key**

💡 Free Key là các thành phần được sử dụng làm key để tìm kiếm trong internal table, nhưng chúng không nhất thiết phải là một phần của primary key của table đó.

" Tìm thành phần b trong bản it có giá trị 2 và gán cho line, không cần dùng READ TABLE

line = it[ b = 2 ].

" Sử dụng thành phần b với giá trị 2 làm điều kiện tìm kiếm, gán kq cho wa

READ TABLE it INTO wa WITH KEY b = 2.

💡 READ TABLE without BINARY SEARCH: Table is accessed linearly. READ TABLE with BINARY SEARCH: Table is accessed using a binary search.

💡 Sử dụng BINARY SEARCH đặc biệt hiệu quả hơn cho các bảng lớn khi cần truy cập dữ liệu thường xuyên. Ở đây chúng ta đánnh đổi chi phí để sort để lấy tốc độ truy cập nhanh, dùng được nhiều lần.

💡 Bảng phải được sắp xếp theo thứ tự tăng dần dựa trên các khóa đang được tìm kiếm. Phù hợp với các standard table không có secondary key và khi cần thực hiện nhiều quyền truy cập đọc vào bảng (tuy nhiên, hãy lưu ý chi phí của việc sắp xếp trước đó).

💡 BINARY SEARCH chỉ hoạt động với các bảng có index, không hoạt động với các hashed tables.

Ví dụ về trường hợp sử dụng sort trên khóa gồm 3 fieled:

" Ví dụ hợp lệ

DATA: itab TYPE TABLE OF some\_structure,

      wa   TYPE some\_structure.

" Bảng itab phải được sắp xếp theo thứ tự của a

READ TABLE itab INTO wa WITH KEY a = value\_a BINARY SEARCH.

" Bảng itab phải được sắp xếp theo thứ tự của a, sau đó là b

READ TABLE itab INTO wa WITH KEY a = value\_a b = value\_b BINARY SEARCH.

" Bảng itab phải được sắp xếp theo thứ tự của a, b, và c

READ TABLE itab INTO wa WITH KEY a = value\_a b = value\_b c = value\_c BINARY SEARCH.

" Ví dụ không hợp lệ

DATA: itab TYPE TABLE OF some\_structure,

      wa   TYPE some\_structure.

" Không thể bỏ qua a

READ TABLE itab INTO wa WITH KEY b = value\_b c = value\_c BINARY SEARCH.

" Không thể sử dụng thành phần không phải khóa

READ TABLE itab INTO wa WITH KEY d = value\_d BINARY SEARCH.

💡 Nếu cần truy cập nhiều lần: Nên sử dụng sorted tables hoặc bảng có secondary keys. Nếu chỉ cần đọc một hoặc vài dữ liệu: Nên xem xét chi phí quản trị của việc thiết lập index.

💡 Lưu ý Biểu thức bảng (table expressions) không hỗ trợ việc thêm BINARY SEARCH. Khi chỉ định KEY ... trong table expressions, hệ thống tự động thực hiện tìm kiếm tối ưu. Không có sự khác biệt về hiệu suất giữa việc sử dụng câu lệnh READ TABLE và biểu thức bảng khi chỉ định KEY ....

💡 Ví dụ so sánh hiệu năng READ TABLE trên standard table với 3 trường hợp: Không dùng BINARY SEARCH (truy cập tuyến tính), dùng BINARY SEARCH, dùng secondary keys:

CLASS zcl\_demo\_test DEFINITION

  PUBLIC

  FINAL

  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if\_oo\_adt\_classrun.

  PROTECTED SECTION.

  PRIVATE SECTION.

ENDCLASS.

CLASS zcl\_demo\_test IMPLEMENTATION.

  METHOD if\_oo\_adt\_classrun~main.

    "Line type and internal table declarations

    TYPES: BEGIN OF demo\_struc,

             idx TYPE i,

             str TYPE string,

             num TYPE i,

           END OF demo\_struc.

    DATA: "Tables with empty primary table key

      itab\_std1 TYPE STANDARD TABLE OF demo\_struc WITH EMPTY KEY,

      itab\_std2 LIKE itab\_std1,

      "Table with empty primary table key, secondary table key specified

      itab\_sec  TYPE STANDARD TABLE OF demo\_struc

                   WITH EMPTY KEY

                   WITH NON-UNIQUE SORTED KEY sk COMPONENTS str num.

    "Populating internal tables

    DO 1000 TIMES.

      INSERT VALUE #( idx = sy-index

                      str = |INDEX{ sy-index }|

                      num = sy-index ) INTO TABLE itab\_std1.

    ENDDO.

    itab\_std2 = itab\_std1.

    itab\_sec = itab\_std1.

    "---- Reading without the BINARY SEARCH addition ----

    DATA(ts1) = utclong\_current( ).

    DO 1000 TIMES.

      READ TABLE itab\_std1

        WITH KEY str = `INDEX500` num = 500

        REFERENCE INTO DATA(dref1).

    ENDDO.

    DATA(ts2) = utclong\_current( ).

    cl\_abap\_utclong=>diff( EXPORTING high     = ts2

                                     low      = ts1

                            IMPORTING seconds = DATA(seconds) ).

    out->write( `Elapsed time for the reads using READ TABLE without the BINARY SEARCH addition:` ).

    out->write( seconds ).

    out->write( repeat( val = `-` occ = 70  ) ).

    "---- Reading with the BINARY SEARCH addition ----

    ts1 = utclong\_current( ).

    "Sorting the internal table when using BINARY SEARCH

    "In this simple example, the internal table is populated by having the free key components

    "to be searched in ascending order anyway. This is to emphasize the requirement to

    "sort the (standard) internal table when using BINARY SEARCH.

    SORT itab\_std2 BY str num.

    DO 1000 TIMES.

      READ TABLE itab\_std2

       WITH KEY str = `INDEX500` num = 500

       BINARY SEARCH

       REFERENCE INTO DATA(dref2).

    ENDDO.

    ts2 = utclong\_current( ).

    cl\_abap\_utclong=>diff( EXPORTING high     = ts2

                                     low      = ts1

                           IMPORTING seconds = seconds ).

    out->write( `Elapsed time for the reads using READ TABLE ... BINARY SEARCH ...:` ).

    out->write( seconds ).

    out->write( repeat( val = `-` occ = 70  ) ).

    "---- Excursion: Reading with READ TABLE using a secondary table key ----

    ts1 = utclong\_current( ).

    DO 1000 TIMES.

      READ TABLE itab\_sec

        WITH TABLE KEY sk COMPONENTS str = `INDEX500` num = 500

        INTO DATA(dref3).

    ENDDO.

    ts2 = utclong\_current( ).

    cl\_abap\_utclong=>diff( EXPORTING high     = ts2

                                     low      = ts1

                           IMPORTING seconds = seconds ).

    out->write( `Elapsed time for the reads using READ TABLE and a secondary table key:` ).

    out->write( seconds ).

  ENDMETHOD.

ENDCLASS.

💡 Thời gian dự đoán là: 1 > 3 > 2

**7.5 Addressing Individual Components of Read Lines**

💡 When reading single lines in general, u can also address individual components of the line using the component selector - (or the object component selector -> or the dereferencing operator ->\* in the case of data reference variables).

💡 Các kí hiệu Selector:

* **Structure Component Selector (-)** for structured data types or structures.
* **Class Component Selector (=>)** for static components of classes or constants/types of interfaces.
* **Object Component Selector (->)** for object-oriented access via reference variables.
* **Dereferencing Operator (->\*)** for accessing data through data reference variables.

💡 Structure Component Selector (-)

DATA(comp1) = it[ b = 2 ]-c.

READ TABLE it INTO DATA(wa) WITH KEY b = 2.

DATA(comp2) = wa-c.

READ TABLE it ASSIGNING FIELD-SYMBOL(<fs>) WITH KEY b = 2.

DATA(comp3) = <fs>-c.

💡 Interface component selector (~)

INTERFACE i1.

  TYPES t1 TYPE string.

ENDINTERFACE.

INTERFACE i2.

  INTERFACES i1.

  METHODS m2 IMPORTING p1 TYPE i1~t1.

ENDINTERFACE.

CLASS c1 DEFINITION.

  PUBLIC SECTION.

    INTERFACES i2.

ENDCLASS.

...

DATA oref TYPE REF TO c1.

oref->i2~m2( ... ).

...

CLASS c1 IMPLEMENTATION.

  METHOD i2~m2.

    ...

  ENDMETHOD.

ENDCLASS.

💡 Class component selector (=>)

CLASS factory DEFINITION CREATE PRIVATE.

    PUBLIC SECTION.

      CLASS-DATA oref TYPE REF TO factory.

      CLASS-METHODS class\_constructor.

      METHODS do\_something.

  ENDCLASS.

  CLASS factory IMPLEMENTATION.

    METHOD class\_constructor.

      CREATE OBJECT oref.

    ENDMETHOD.

    METHOD do\_something.

      "Do something

    ENDMETHOD.

  ENDCLASS.

  " factory=>oref truy cập thành phần tĩnh oref của lớp factory.

  " Sau đó, oref->do\_something() gọi phương thức do\_something trên đối tượng mà oref trỏ đến.

  factory=>oref->do\_something( ).

💡 Object component selector (->)

" của một đối tượng thông qua một biến tham chiếu.

READ TABLE it REFERENCE INTO DATA(dref) WITH KEY b = 2.

DATA(comp4) = dref->c.

" Truy cập thuộc tính public của class

CLASS c1 DEFINITION.

    PUBLIC SECTION.

      DATA a1 TYPE string READ-ONLY.

ENDCLASS.

  ...

DATA oref TYPE REF TO c1.

  ... oref->a1 ...

" Biến tham chiếu dữ liệu dref tham chiếu đến structure và carrid thành phần của structure đó." Biểu thức dref->carrid có cùng ý nghĩa với dref->\*-carrid.

DATA dref TYPE REF TO sflight.

...

... dref->carrid ...

💡 Dereferencing Operator (->\*): được sử dụng để dereference một tham chiếu dữ liệu và truy cập nội dung của nó

" Toán tử ->\* dereferences dref và cho phép truy cập thành phần c.

" Trong trường hợp này, dref->\*-c tương đương với dref->c.

DATA(comp5) = dref->\*-c.

" Khởi tạo i với giá trị 111 và gán cho dref.

DATA(dref) = NEW i( 111 ).

" Toán tử ->\* dereferences dref và cho phép truy cập toàn bộ ({ dref->\* }) thành phần i.

cl\_demo\_output=>write( |dref->\* { dref->\* }| ).

" Xóa dữ liệu của dref.

CLEAR dref.

" Gán nội dung của dref->\* cho FIELD-SYMBOL <fs>.

ASSIGN dref->\* TO FIELD-SYMBOL(<fs>).

" Hiển thị kết quả của biến hệ thống sy-subrc sau lệnh ASSIGN

" sy-subrc là một hệ thống biến chứa kết quả của lệnh ASSIGN

cl\_demo\_output=>write( |sy-subrc { sy-subrc }| ).

cl\_demo\_output=>display( ).

**8. Getting Information about Internal Tables, Table Lines, Table Types**

**8.1 Checking the Existence of a Line in an Internal Table**

💡 Kiểm tra một dòng có tồn tại hay không mà không quan tâm nó có nội dung như thế nào, dựa vào index hoặc thông số nào đó. Dùng TRANSPORTING NO FIELDS ám chỉ không có dòng nào tự sự được đọc, nó chỉ được kiểm tra và nếu tồn tại thì biến sy-subrc = 0.

💡 Một cách kiểm tra khác là dùng hàm line\_exists( ), hàm này dùng table expression như là một đối số và không thay đổi biến sy-subrc.

"Read using the key

READ TABLE it WITH KEY b = 2 TRANSPORTING NO FIELDS.

IF sy-subrc = 0.

  ...

ENDIF.

"Read using the index

READ TABLE it INDEX 1 TRANSPORTING NO FIELDS.

IF sy-subrc = 0.

  ...

ENDIF.

"Read using the key

IF line\_exists( it[ b = 2 ] ).

  ...

ENDIF.

"Read using the index

IF line\_exists( it[ 1 ] ).

  ...

ENDIF.

💡 Hàm line\_exists() trả về giá trị true/false:

DATA flight\_tab TYPE HASHED TABLE OF spfli

                     WITH UNIQUE KEY carrid connid.

SELECT \*

       FROM spfli

       INTO TABLE @flight\_tab.

IF line\_exists( flight\_tab[ carrid = 'LH'

                            connid = '0400' ] ).

  ...

ENDIF.

**8.2 Checking the Index of a Line in an Internal Table**

💡 Để kiểm tra sự tồn tại của một chỉ mục theo điều kiện, có thể dùng lệnh READ TABLE, nếu index được tìm thấy thì sy-tabix được set thành index đó.

💡 Cũng có thể dùng hàm line\_index( ), hàm này trả về index nếu tìm thấy, trả về 0 nếu không tìm thấy.

DATA(itab) = VALUE string\_table( ( `aaa` ) ( `bbb` ) ).

READ TABLE itab WITH KEY table\_line = `bbb` TRANSPORTING NO FIELDS.

"2

DATA(tabix) = sy-tabix.

"1

DATA(idx) = line\_index( itab[ table\_line = `aaa` ] ).

**8.3 Checking How Many Lines Exist in an Internal Table**

💡 Hàm lines( ) trả về sô lượng dòng tồn tại trong internal table, luôn trả về 1 số nguyên.

DATA(itab) = VALUE string\_table( ( `a` ) ( `b` ) ( `c` ) ( `d` ) ( `e` ) ).

"5

DATA(number\_of\_lines) = lines( itab ).

**8.4 Getting Table (Type) Information and Creating Internal Tables and Types at Runtime**

Using [Runtime Type Services (RTTS)](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abenrun_time_type_services_glosry.htm) u can ...

* get type information on internal tables and table types at runtime ([Runtime Type Identification (RTTI)](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abenrun_time_type_identific_glosry.htm)).
* define and create new internal tables and table types as [type description objects](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abentype_object_glosry.htm) at runtime ([Runtime Type Creation (RTTC)](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abenrun_time_type_creation_glosry.htm)).

For more information, see the [Dynamic Programming](https://github.com/SAP-samples/abap-cheat-sheets/blob/main/06_Dynamic_Programming.md) cheat sheet.

**9 Processing Multiple Internal Table Lines Sequentially**

💡 Nếu không quan tâm đến những dòng đơn lẽ mà muốn xử lý một vài dòng, toàn bộ dòng hoặc vài phần nào đó một cách tuần tự thì có thể dùng LOOP AT. Target area có nhiều sự lựa chọn: work area, field symbol, data reference.

💡 Có nhiều bổ sung cho câu lệnh LOOP AT để hạn chế hơn nữa nội dung bảng cần xử lý.

Simple form:

"The target is an existing work area.

DATA wa LIKE LINE OF it.

LOOP AT it INTO wa.

  "No addition of the loop statement; all lines are processed

  "Statements in this block are relevant for each individual table line.

  ...

ENDLOOP.

"Work area declared inline

LOOP AT itab INTO DATA(wa\_inl).

  ...

ENDLOOP.

"Field symbols

FIELD-SYMBOLS <fs> LIKE LINE OF it.

LOOP AT it ASSIGNING <fs>.

  ...

ENDLOOP.

LOOP AT it ASSIGNING FIELD-SYMBOL(<fs\_inl>).

  ...

ENDLOOP.

"Data reference variables

DATA dref TYPE REF TO dbtab.

LOOP AT it REFERENCE INTO dref.

  ...

ENDLOOP.

LOOP AT it REFERENCE INTO DATA(dref\_inl).

  ...

ENDLOOP.

💡 Thứ tự duyệt qua bảng: Duyệt qua theo thứ tự tăng dần của index đối với Index tables, Duyệt qua theo thứ tự các dòng được thêm vào đối với hashed tables, có thể sắp xếp lại bảng trước khi lặp.

💡 Có thể kiểm soát bước nhảy trong vòng lặp bằng ‘STEP’. sy-tabix phản ánh số dòng hiện tại đối với index tables, luôn bằng 0 đối với hashed table. Cần sử dụng sy-tabix ngay sau câu lệnh LOOP để tránh bị ghi đè.

**9.1 Restricting the Area of a Table to Be Looped Over**

💡 Có thể giới hạn vùng được lặp hoặc thay đổi thứ tự lặp, thay đổi step.

"FROM/TO: Only for index tables

"Specifying an index range

LOOP AT it INTO wa FROM 2 TO 5.

  ...

ENDLOOP.

"From specified line until the end

LOOP AT it INTO wa FROM 2.

  ...

ENDLOOP.

"From first line until the specified line

LOOP AT it INTO wa TO 5.

  ...

ENDLOOP.

"WHERE clause: Restricting lines based on logical expression

LOOP AT it INTO wa WHERE a > 1 AND b < 4.

  ...

ENDLOOP.

"No interest in the table content; only relevant system fields are populated

"Mandatory WHERE clause

LOOP AT it TRANSPORTING NO FIELDS WHERE a < 5.

  ...

ENDLOOP.

"Table key specification (snippet uses example table from above)

"The specified table key affects the order in which the table lines

"are accessed and the evaluation of the other conditions.

LOOP AT it INTO wa USING KEY primary\_key.

"LOOP AT it INTO wa USING KEY pk.            "primary key alias

"LOOP AT it INTO wa USING KEY sec\_key.       "secondary key

"LOOP AT it INTO wa USING KEY sk.            "secondary key alias

  ...

ENDLOOP.

"STEP addition for defining the step size and the direction of the loop

"- Step size: Specified by the absolute value of an integer

"- Direction: Specified by a positive (forward loop) or negative

"  (loop in reverse order) integer

"Reversing the loop order using a negative integer

"Each line is read indicated by the absolute value 1

LOOP AT it INTO wa STEP -1.

  ...

ENDLOOP.

"Forward loop by specifiying a positive integer

"In the example, every second line is read.

"Note: Omitting STEP means STEP 1 by default.

LOOP AT it INTO wa STEP 2.

  ...

ENDLOOP.

"STEP with other additions

"The example uses the additions FROM and TO.

"Note: If the value after STEP is negative, the value

"after FROM must be greater than the value after TO.

LOOP AT it INTO wa FROM 6 TO 3 STEP -2.

  ...

ENDLOOP.

**9.2 Iteration Expressions**

💡 Iteration expressions with [FOR](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abenfor.htm) as part of certain constructor expressions allow u to create content of an internal table by evaluating one or more source tables.

The expressions are covered in the cheat sheet [Constructor Expressions](https://github.com/SAP-samples/abap-cheat-sheets/blob/main/05_Constructor_Expressions.md):

* [Iteration Expressions Using FOR](https://github.com/SAP-samples/abap-cheat-sheets/blob/main/05_Constructor_Expressions.md#iteration-expressions-using-for)
* Special reduction operator REDUCE that is based on iteration expressions: [REDUCE](https://github.com/SAP-samples/abap-cheat-sheets/blob/main/05_Constructor_Expressions.md#reduce)

**9.3 Interrupting and Exiting Loops**

💡 ABAP keywords such as CONTINUE, CHECK, and EXIT, are available to exit and interrupt loops. Find more information in the [Program Flow Logic](https://github.com/SAP-samples/abap-cheat-sheets/blob/main/13_Program_Flow_Logic.md#interrupting-and-exiting-loops) cheat sheet.

💡 In the following example, the loop is exited using the EXIT statement when a certain condition is met.

DATA(str\_table) = VALUE string\_table( ( `a` ) ( `b` ) ( `c` ) ( `d` ) ( `e` ) ( `f` ) ).

LOOP AT str\_table INTO DATA(wa).

  DATA(tabix) = sy-tabix.

  IF wa = `e`.

    EXIT.

  ENDIF.

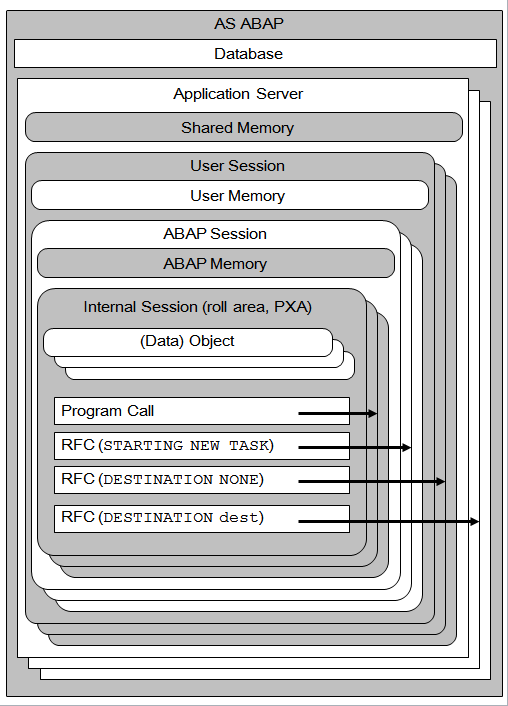
ENDLOOP.

ASSERT tabix = 5.

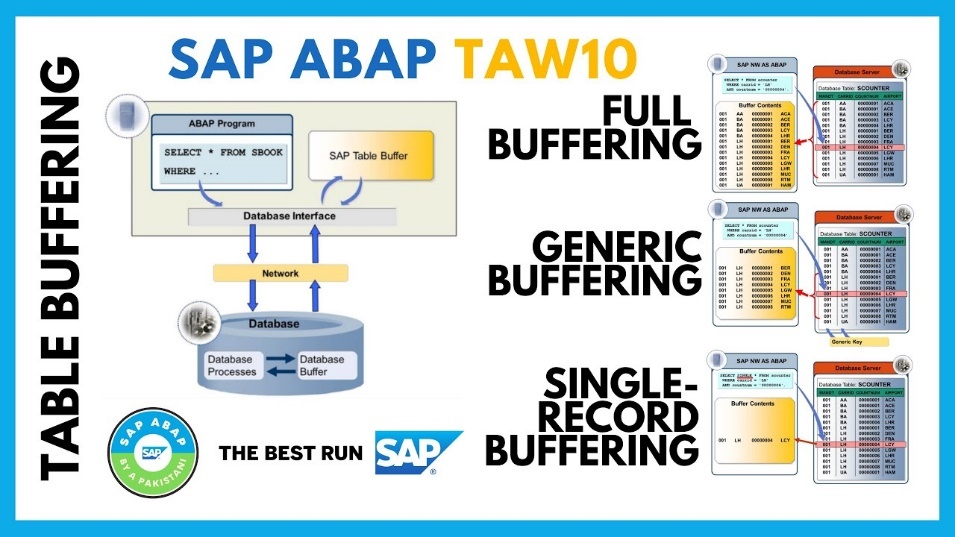
**10 Operations with Internal Tables Using ABAP SQL SELECT Statements**

💡 In ABAP, database data is buffered in a table buffer (internally, this happens in internal tables in the shared memory of the ABAP server).

💡 Trong quá trình read access, nó sẽ được kiểm tra xem dữ liệu có nằm trong buffer hay không và nếu có thì quá trình đọc sẽ diễn ra trực tiếp từ đó. Nếu không, dữ liệu đầu tiên sẽ được tải vào buffer.



💡 Table buffer trong SAP ABAP là một cơ chế giúp tăng hiệu suất hệ thống bằng cách lưu trữ tạm thời dữ liệu của bảng trong buffer cục bộ của máy chủ ứng dụng. Khi truy cập table, dữ liệu được lấy trực tiếp từ buffer thay vì phải truy cập cơ sở dữ liệu, giúp giảm thời gian truy xuất dữ liệu.



💡 Khi sử dụng Full buffering, tất cả các bản ghi của bảng được tải vào bộ đệm khi truy cập vào bất kỳ bản ghi nào của bảng đó. Điều này có nghĩa là hoặc toàn bộ bảng được lưu trong bộ đệm hoặc không lưu chút nào. Đối với các bảng nhỏ được đọc thường xuyên và ít khi được ghi, sử dụng full buffering là lựa chọn tối ưu.

A diagram of a server

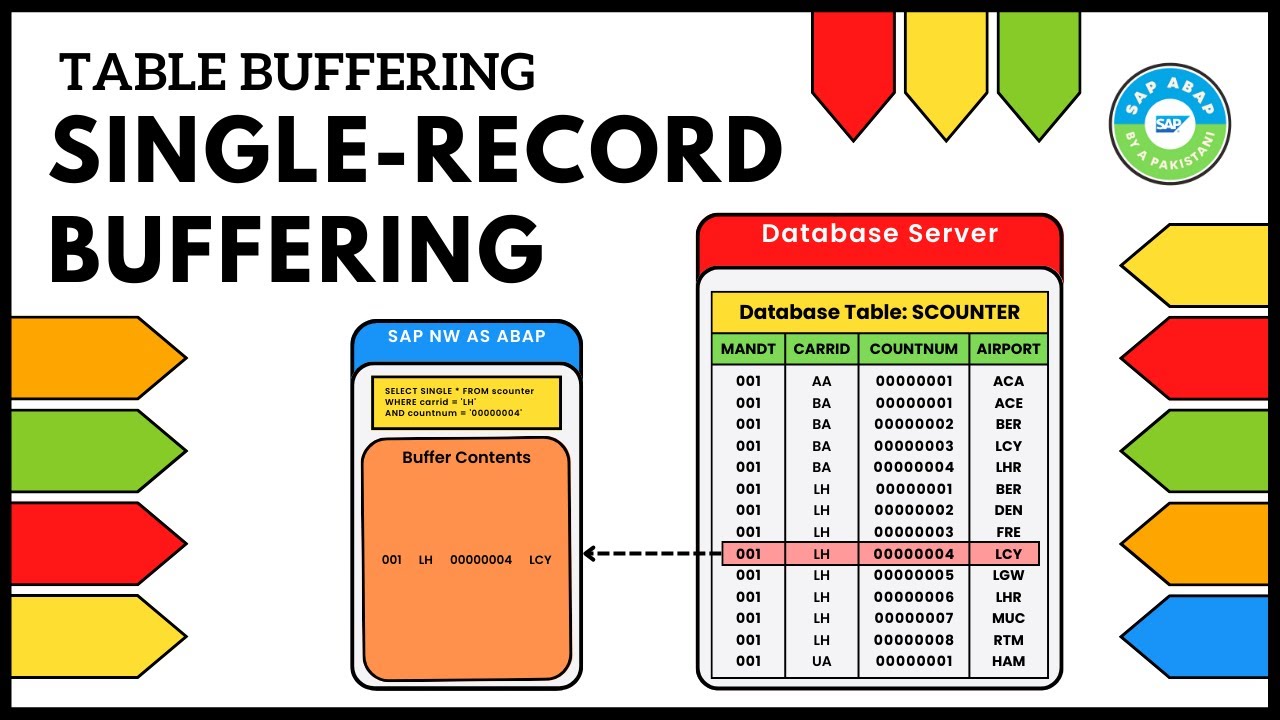
Description automatically generated

💡 Generic Buffering trong SAP là cơ chế cho phép tải các bản ghi vào bộ đệm dựa trên các trường khóa chung khi truy cập vào một bản ghi cụ thể. Khi truy cập vào một bản ghi, các bản ghi khác có trường khóa chung khớp cũng được tải vào bộ đệm. Chọn generic key phù hợp để tránh tối ưu hóa không hiệu quả do kích thước quá lớn. Dùng full buffering cho các bảng ít bản ghi trên mỗi khu vực generic để tối ưu hiệu suất.

A computer screen with a diagram

Description automatically generated with medium confidence

💡 Single-record buffering là quá trình tải các bản ghi được truy cập thông qua SELECT SINGLE vào bộ đệm. Được khuyến khích đối với các bảng lớn mà chỉ có một vài bản ghi được truy cập thường xuyên. Các truy cập không phải SELECT SINGLE sẽ bỏ qua bộ đệm và truy vấn trực tiếp vào cơ sở dữ liệu. Nếu một bản ghi chưa được lưu vào bộ đệm và không tìm thấy trong cơ sở dữ liệu, nó sẽ được đánh dấu là không tồn tại trong bộ đệm để ngăn chặn các truy cập cơ sở dữ liệu lặp lại. Tiết kiệm không gian bộ đệm so với generic và full buffering nhưng đòi hỏi nhiều lần truy cập cơ sở dữ liệu hơn để tải các bản ghi.



**10.1 Internal Tables as Target Data Objects**

💡 Giả sử chúng ta có hai bảng db1 và db2 với các cột và dữ liệu như sau:

**Bảng db1**

|  |  |
| --- | --- |
| **comp1** | **comp2** |
| 1 | A |
| 2 | B |
| 3 | C |

**Bảng db2**

|  |  |  |
| --- | --- | --- |
| **comp1** | **comp\_abc** | **comp\_xyz** |
| 1 | X | Y |
| 2 | W | Z |
| 4 | V | U |

💡 Adding multiple lines from a database table to an internal table using SELECT, for example, based on a condition. In the case below, the internal table is created inline.

SELECT FROM dbtab

  FIELDS comp1, comp2 ...

  WHERE ...

  INTO TABLE @DATA(itab\_sel).

💡 Có thể dùng FIELDS\* để chọn tất cả các fields, dùng APPENDING CORRESPONDING FIELDS INTO TABLE để giữa lại những dòng hiện có trong internal table và thêm vào cuối, dùng INTO CORRESPONDING FIELDS OF TABLE để xóa thông tin hiện có trong internal table và thêm mới.

SELECT FROM dbtab2

  FIELDS \*

  WHERE ...

  APPENDING CORRESPONDING FIELDS OF TABLE @itab.

SELECT FROM dbtab2

  FIELDS \*

  WHERE ...

  INTO CORRESPONDING FIELDS OF TABLE @itab.

💡 Combining data from multiple database tables into one internal table using an inner join. The following example uses the INNER JOIN addition. Note that the field list includes fields from both tables. The fields are referred to using ~.

SELECT db1~comp1, db1~comp2, db2~comp\_abc, db2~comp\_xyz ...

  FROM db1

  INNER JOIN db2 ON db1~comp1 = db2~comp1

  INTO TABLE @DATA(it\_join\_result).

**Kết quả:**

|  |  |  |  |
| --- | --- | --- | --- |
| **db1~comp1** | **db1~comp2** | **db2~comp\_abc** | **db2~comp\_xyz** |
| 1 | A | X | Y |
| 2 | B | W | Z |

💡 Populating an internal table from a database table using [subqueries](https://help.sap.com/doc/abapdocu_cp_index_htm/CLOUD/en-US/index.htm?file=abensubquery_glosry.htm). The following two examples populate an internal table from a database table. In the first example, a subquery is specified in the WHERE clause with the NOT IN addition. It checks whether a value matches a value in a set of values specified in parentheses. The second example populates an internal table depending on data in another table. A subquery with the EXISTS addition is specified in the WHERE clause. In this case, the result of the subquery, which is another SELECT statement, is checked to see if an entry exists in a table based on the specified conditions.

SELECT comp1, comp2, ...

  FROM dbtab

  WHERE comp1 NOT IN ( a, b, c ... )

  INTO TABLE @DATA(it\_subquery\_result1).

💡 Giả sử dbtab là db1 và a, b, c là 1, 3:

**Kết quả:**

|  |  |
| --- | --- |
| **comp1** | **comp2** |
| 2 | B |

SELECT comp1, comp2, ...

  FROM db1

  WHERE EXISTS ( SELECT 'X' FROM db2

                 WHERE comp1 = db1~comp1 )

  INTO TABLE @DATA(it\_subquery\_result2).

**Kết quả:**

|  |  |
| --- | --- |
| **comp1** | **comp2** |
| 1 | A |
| 2 | B |

💡 Populating an internal table from a table based on the existence of data in another table using the FOR ALL ENTRIES addition.

💡 Make sure that the internal table u are reading from is not initial. Therefore, it is recommended that u use a subquery as shown above: ... ( SELECT ... FROM ... WHERE ... )

IF itab IS NOT INITIAL.

  SELECT dbtab~comp1, dbtab~comp2, ...

    FROM dbtab

    FOR ALL ENTRIES IN @itab

    WHERE comp1 = @itab-comp1

    INTO TABLE @DATA(it\_select\_result).

ENDIF.

💡 Giả sử dbtab là db1:

**Kết quả:**

|  |  |
| --- | --- |
| **dbtab~comp1** | **dbtab~comp2** |
| 1 | A |
| 3 | C |

**10.2 Querying from Internal Tables**

💡 Trong ABAP, có thể sử dụng internal table làm nguồn dữ liệu cho câu lệnh SELECT. Điều này có nghĩa là có thể chọn (SELECT) dữ liệu từ một internal table và thêm nó vào một internal table khác.

DATA: itab TYPE TABLE OF some\_structure,

      itab\_sel TYPE TABLE OF some\_structure.

SELECT comp1, comp2, ...

  FROM @itab AS it\_alias

  INTO TABLE @DATA(itab\_sel).

💡 **@itab** là internal table nguồn muốn chọn dữ liệu, dấu @ được sử dụng để biểu thị rằng đây là một internal table đã được khai báo trước đó trong chương trình. **AS it\_alias** định nghĩa một tên gọi thay thế (alias) cho internal table **itab** trong câu lệnh **SELECT**, Alias giúp dễ dàng tham chiếu tới internal table này trong các câu lệnh phức tạp hơn.

Dùng toán tử LIKE trong mệnh đề WHERE:

TYPES: BEGIN OF s,

          a TYPE c LENGTH 3,

          b TYPE i,

        END OF s,

        tab\_type TYPE TABLE OF s WITH EMPTY KEY.

DATA(itab) = VALUE tab\_type( ( a = 'abc' b = 1 ) ( a = 'zbc' b = 2 )

                              ( a = 'bde' b = 3 ) ( a = 'yde' b = 4 ) ).

SELECT a, b

  FROM @itab AS it\_alias

  WHERE a LIKE '%bc'

  INTO TABLE @DATA(itab\_sel\_like).

\*A      B

\*abc    1

\*zbc    2

Dùng toán tử IN trong mệnh đề WHERE:

SELECT a, b

  FROM @itab AS it\_alias

  WHERE a IN ('bde', 'yde', 'zde')

  INTO TABLE @DATA(itab\_in).

\*A      B

\*bde    3

\*yde    4

💡 Combining data from multiple internal tables into one internal table using an inner join. See above.

TYPES: BEGIN OF s,

          a TYPE c LENGTH 3,

          b TYPE c LENGTH 3,

          c TYPE i,

        END OF s,

        tab\_type TYPE TABLE OF s WITH EMPTY KEY.

DATA(it1) = VALUE tab\_type( ( a = 'aaa' b = 'bbb' c = 1 )

                            ( a = 'ccc' b = 'ddd' c = 1 )

                            ( a = 'eee' b = 'fff' c = 2 ) ).

DATA(it2) = VALUE tab\_type( ( a = 'ggg' b = 'hhh' c = 1 )

                            ( a = 'iii' b = 'jjj' c = 1 )

                            ( a = 'kkk' b = 'lll' c = 3 ) ).

SELECT it\_alias1~a, it\_alias2~b

  FROM @it1 AS it\_alias1

  INNER JOIN @it2 AS it\_alias2 ON it\_alias1~c = it\_alias2~c

  INTO TABLE @DATA(it\_join\_result).

\*A      B

\*aaa    hhh

\*aaa    jjj

\*ccc    hhh

\*ccc    jjj

**11 Sorting Internal Tables**

💡 Các sorted table được lưu trữ trong bộ nhớ theo thứ tự được sắp xếp tự động, do đó, chúng không thể được sắp xếp một cách rõ ràng bằng SORT.

💡 For standard and hashed tables, the order can be changed.

💡 Khi dùng SORT thì thứ tự được sắp xếp được dựa trên primary key hoặc trên các fields được chỉ định, không thể dùng SORT trên secondary key.

💡 Cần đặc tả rõ ràng bằng câu lệnh để code dễ đọc và tránh nhầm lẫn, đặc biệt là table có standard key.

💡 Sorting by primary table key

"Implicit sorting by primary table key and in ascending order by default

SORT itab.

"Optional additions to determine the sort order

"As mentioned above, ASCENDING is used implicitly. Here, specifying it explicitly.

SORT itab ASCENDING.

SORT itab DESCENDING.

💡 Khi sử dụng câu lệnh sắp xếp đơn giản mà không chỉ định rõ ràng các key, ABAP sẽ sử dụng standard key. Standard key bao gồm tất cả các thành phần ký tự và byte của structure line type. Điều này có thể dẫn đến kết quả sắp xếp không như mong muốn nếu không chú ý đến các thành phần trong khóa chuẩn.

"Is basically the same as it2

DATA it1 TYPE TABLE OF zdemo\_abap\_fli.

DATA it2 TYPE STANDARD TABLE OF zdemo\_abap\_fli WITH DEFAULT KEY.

"Respecting the standard key when sorting

SORT it1.

💡 Nếu một internal table có structured line type chỉ chứa các thành phần số nguyên (numeric components), việc sắp xếp table này sẽ không có hiệu quả. Điều này xảy ra vì primary table key được coi là rỗng (empty).

Sorting by explicitly specifying components. Các câu lệnh sort khá giống SQL.

DATA it3 TYPE TABLE OF struc WITH NON-UNIQUE KEY a.

"Sorting by primary table key a

SORT itab.

"Specifying the component to sort for; here, it is the same as the key;

"this way, the sorting is easier to understand

SORT itab BY a.

"Syntax showing multiple component sorting with component-wise sort order

SORT itab BY a b ASCENDING c DESCENDING.

"Sorting respecting the entire line (e. g. in the context of tables with

"empty or standard keys)

SORT itab BY table\_line.

**12 Modifying Internal Table Content**

Có thể sửa đổi nội dung của internal table bằng các câu lệnh READ TABLE và LOOP AT thông qua field symbols hoặc data reference variables. Ngoài ra cũng có thể dùng table expressions.

Lưu ý là primary key của sorted table và hashed table là read-only. Nếu cố gắng sửa đổi chúng sẽ xảy ra lỗi runtime error, lỗi này không thể phát hiện lúc code mà chỉ có lỗi khi chạy.

"Table declarations

DATA it\_st TYPE TABLE OF struc WITH NON-UNIQUE KEY a.

DATA it\_so TYPE SORTED TABLE OF struc WITH UNIQUE KEY a.

"Reading table line into target area

READ TABLE it\_st ASSIGNING FIELD-SYMBOL(<fs>) INDEX 1.

READ TABLE it\_so REFERENCE INTO DATA(dref) INDEX 2.

"Modification examples

"Modifying the entire table line while keeping the values of other components;

"this way is not possible for it\_so because of key value change.

<fs> = VALUE #( BASE <fs> a = 1 b = 2 ).

"Modifying a single component via field symbol

<fs>-c = 3.

"Modification via dereferencing

ref->b = 4.

"Table expressions

it\_st[ 1 ] = VALUE #( a = 1 b = 2 ).

it\_st[ 2 ]-c = 3.

"Sorted table: no key field change

it\_so[ 2 ]-d = 4.

If u want to modify recently read lines in a work area, for example, within a loop (LOOP AT INTO dobj), u can modify the line and then use a MODIFY statement to modify the internal table based on this line.

MODIFY statements provide multiple ways of changing the content of single and multiple table lines by specifying the table key or a table index, without first reading the lines into a target area.

DATA: itab TYPE TABLE OF zdemo\_abap\_fli,

      wa   TYPE zdemo\_abap\_fli.

LOOP AT itab INTO wa.

  " Sửa đổi dòng trong vùng làm việc

  wa-comp1 = new\_value.

  " Cập nhật lại bảng nội bộ với dòng đã sửa đổi

  MODIFY itab FROM wa.

ENDLOOP.

" More about MODIFY statement

"Addition FROM ...; specified key values determine the line to be modified

"line: existing line including key values

MODIFY TABLE it FROM line.

"line constructed inline

MODIFY TABLE it FROM VALUE #( a = 1 b = 2 ... ).

"Respecting only specified fields with the addition TRANSPORTING

"In case of sorted/hashed tables, key values cannot be specified.

MODIFY TABLE it FROM line TRANSPORTING b c.

"Modification via index

"Note that it is only MODIFY, not MODIFY TABLE.

"Example: It modifies the line with number 1 in the primary table index.

MODIFY it FROM line INDEX 1.

"Without the addition TRANSPORTING, the entire line is changed.

"Example: It modifies specific values.

MODIFY it FROM line INDEX 1 TRANSPORTING b c.

"USING KEY addition

"If the addition is not specified, the primary table key is used;

"otherwise, it is the explicitly specified table key that is used.

"Example: It is the same as MODIFY it FROM line INDEX 1.

MODIFY it FROM line USING KEY primary\_key INDEX 1.

"The statement below uses a secondary key and an index specification

"for the secondary table index. Only specific fields are modified.

MODIFY it FROM line USING KEY sec\_key INDEX 1 TRANSPORTING c d.

"Modifying multiple lines in internal tables

"All lines matching the logical expression in the WHERE clause are modified

"as specified in line.

"The additions TRANSPORTING and WHERE are both mandatory; USING KEY is optional.

MODIFY it FROM line TRANSPORTING b c WHERE a < 5.

The system field sy-subrc is set to 0 if at least one line was changed. It is set to 4 if no lines were changed.

MODIFY, DELETE, and INSERT nếu đi kèm từ khóa TABLE thì truy cập thông qua index còn nếu không có từ khóa TABLE thì truy cập thông quan table key.

**13 Deleting Internal Table Content**

U can use DELETE statements to delete single and multiple lines in internal tables. The following additions can be used: USING KEY (for specifying a table key), FROM/TO (for specifying row ranges), STEP (for specifying the step size), and WHERE (for specifying conditions).

"-------------- Deleting via index --------------

"Example: The first line in the table is deleted.

DELETE it INDEX 1.

"If USING KEY is not used, INDEX can only be used with index tables.

"If doing so, it determines the line from the primary table index.

"If a secondary key is specified, the secondary table index is respected

"Example: same as above

DELETE it INDEX 1 USING KEY primary\_key.

"Deleting an index range; FROM or TO alone can also be specified

DELETE it FROM 2 TO 5.

"-------------- Deleting via keys --------------

"The line must have a compatible type to the tables line type and

"include key values. The first found line with the corresponding keys

"is deleted.

"If the key is empty, no line is deleted.

DELETE TABLE it FROM line.

"Instead of specifying the keys using a data object ("line" above),

"the keys can be specified separately. All key values must be specified.

"Example: Respects keys from primary table index.

DELETE TABLE it WITH TABLE KEY a = 1.

"U can also specify secondary keys.

"Example: Same as above

DELETE TABLE it WITH TABLE KEY primary\_key COMPONENTS a = 1.

DELETE TABLE it\_sec WITH TABLE KEY sec\_key COMPONENTS ...

"---------- Deleting multiple lines based on a WHERE condition ----------

"Specifying the additions USING KEY, FROM, TO is also possible.

DELETE it WHERE a < 6.

"Excursion: Deleting in a LIKE-like fashion u may know from

"ABAP SQL statements.

"The LIKE addition is not available for the WHERE clause in DELETE

"statements for internal tables as is the case for ABAP SQL DELETE statements.

DATA(str\_table) = VALUE string\_table( ( `abcZ` ) ( `Zdef` ) ( `gZhi` )

                                      ( `Zjkl` ) ( `Zmno` ) ( `pqrZ` ) ).

"U can, for example, use logical operators such as CP (conforms to pattern)

"All lines that begin with Z are to be deleted.

DELETE str\_table WHERE table\_line CP `Z\*`.

"Result: abcZ / gZhi / pqrZ

"---------- Deleting the current line inside a LOOP statement ----------

"The following example illustrates deleting the current table line

"using a DELETE statement within a LOOP statement. Lines with even

"numbers are deleted.

"Note:

"- The short form of the DELETE statement always deletes the

"  current first line implicitly. It is only possible within a LOOP

"  statement and the delete operation is performed on the same internal

"  table.

"- The field symbol (or reference variable) should not be used after

"  the DELETE statement any more.

DATA itab1 TYPE TABLE OF i WITH EMPTY KEY.

itab1 = VALUE #( ( 1 ) ( 2 ) ( 3 ) ( 4 ) ( 5 ) ( 6 ) ( 7 ) ( 8 ) ( 9 ) ( 10 ) ).

LOOP AT itab1 ASSIGNING FIELD-SYMBOL(<fs>).

  IF <fs> MOD 2 = 0.

    DELETE itab1.

  ENDIF.

ENDLOOP.

\*Table content:

\*1

\*3

\*5

\*7

\*9

"The following, similar example (uneven numbers are deleted) uses a

"table which is looped over by specifying the addition USING KEY.

"In this case (using LOOP ... USING KEY ...), the short form of the

"DELETE statement cannot be used. Use the DELETE statement with the

"addition USING KEY loop\_key to delete the current first line.

"loop\_key is a predefined name to be used with DELETE and within

"loops that specify LOOP ... USING KEY .... No other key name is

"possible here.

DATA itab2 TYPE TABLE OF i WITH NON-UNIQUE KEY table\_line.

itab2 = VALUE #( ( 1 ) ( 2 ) ( 3 ) ( 4 ) ( 5 ) ( 6 ) ( 7 ) ( 8 ) ( 9 ) ( 10 ) ).

LOOP AT itab2 USING KEY primary\_key REFERENCE INTO DATA(dref2).

  IF dref2->\* MOD 2 <> 0.

    DELETE itab2 USING KEY loop\_key.

  ENDIF.

ENDLOOP.

\*Table content:

\*2

\*4

\*6

\*8

\*10

**13.1 Deleting Adjacent Duplicate Lines (liền kề trùng lặp)**

DELETE ADJACENT DUPLICATES statements allow u to delete all adjacent lines except for the first line that have the same content in certain components (dòng đầu tiên trong đám giống nhau được giữ lại). U usually need to perform some appropriate sorting before using these statements.

"Implicitly using the primary table key

DELETE ADJACENT DUPLICATES FROM it.

"Deletion respecting the values of the entire line

DELETE ADJACENT DUPLICATES FROM it COMPARING ALL FIELDS.

"Only lines are deleted with matching content in specific fields

DELETE ADJACENT DUPLICATES FROM it COMPARING a c.

"Deletion respecting a specified table key

"Same as first example above

DELETE ADJACENT DUPLICATES FROM it USING KEY primary\_key.

DELETE ADJACENT DUPLICATES FROM it USING KEY sec\_key.

The system field sy-subrc is set to 0 if at least one line has been deleted. It is set to 4 if no lines were deleted.

**13.2 Deleting the Entire Internal Table Content**

The CLEAR and FREE statements allow to delete the entire table content.

**CLEAR** xóa nội dung của bảng nội bộ nhưng không giải phóng không gian bộ nhớ đã được cấp phát ban đầu. Nếu sau đó bảng nội bộ được điền lại dữ liệu, không gian bộ nhớ vẫn có sẵn, giúp cải thiện hiệu suất.

**FREE** xóa nội dung của bảng nội bộ và đồng thời giải phóng không gian bộ nhớ đã được cấp phát. Nếu bảng nội bộ được điền lại dữ liệu sau khi sử dụng FREE, nó cần phải yêu cầu lại không gian bộ nhớ, điều này có thể ảnh hưởng đến hiệu suất.

Gán một bảng nội bộ bằng toán tử **VALUE** mà không có mục nào trong dấu ngoặc đơn cũng sẽ làm sạch bảng nội bộ. Điều này tương đương với **CLEAR**, không giải phóng không gian bộ nhớ.

Nếu có một biến tham chiếu dữ liệu trỏ đến bảng nội bộ, có thể làm tương tự bằng cách tạo một đối tượng mới mà không có mục nào. Điều này tương đương với **CLEAR** cho biến tham chiếu.

CLEAR it.

"This statement additionally releases memory space.

FREE it.

"Assignment using the VALUE operator without entries in the parentheses

it = VALUE #( ).

"The same applies to data reference variables pointing to internal tables.

it\_ref = NEW #( ).

**14 Grouping Internal Tables**

Đọc mà đếch hiểu mẹ gì cả 😊

CLASS zcl\_some\_class DEFINITION

  PUBLIC

  FINAL

  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if\_oo\_adt\_classrun.

  PROTECTED SECTION.

  PRIVATE SECTION.

ENDCLASS.

CLASS zcl\_some\_class IMPLEMENTATION.

  METHOD if\_oo\_adt\_classrun~main.

    " Khai báo kiểu dữ liệu cho cấu trúc demo\_struct và kiểu bảng tab\_type

    TYPES: BEGIN OF demo\_struct,

             comp1 TYPE c LENGTH 1,        " Ký tự độ dài 1

             comp2 TYPE i,                 " Số nguyên

             comp3 TYPE abap\_boolean,      " Kiểu boolean

             comp4 TYPE string,            " Chuỗi

           END OF demo\_struct,

           tab\_type TYPE TABLE OF demo\_struct WITH EMPTY KEY. " Bảng không có khóa

    " Khai báo bảng string\_table để lưu trữ các khóa nhóm

    DATA str\_table TYPE string\_table.

    " Khởi tạo bảng nội bộ it với dữ liệu mẫu

    DATA(it) = VALUE tab\_type(

      ( comp1 = 'd' comp2 = 0 comp3 = abap\_false )

      ( comp1 = 'a' comp2 = 1 comp3 = abap\_true )

      ( comp1 = 'a' comp2 = 2 comp3 = abap\_false )

      ( comp1 = 'e' comp2 = 11 comp3 = abap\_true )

      ( comp1 = 'b' comp2 = 5 comp3 = abap\_true )

      ( comp1 = 'b' comp2 = 6 comp3 = abap\_false )

      ( comp1 = 'a' comp2 = 3 comp3 = abap\_false )

      ( comp1 = 'b' comp2 = 4 comp3 = abap\_true )

      ( comp1 = 'c' comp2 = 10 comp3 = abap\_true )

      ( comp1 = 'e' comp2 = 1 comp3 = abap\_false )

      ( comp1 = 'd' comp2 = 7 comp3 = abap\_true )

      ( comp1 = 'a' comp2 = 4 comp3 = abap\_true )

      ( comp1 = 'e' comp2 = 111 comp3 = abap\_true ) ).

    " Ví dụ không sử dụng vòng lặp lồng và không có ràng buộc khóa nhóm

    " Vùng làm việc (work area) chứa dòng đầu tiên của mỗi nhóm (đại diện cho nhóm)

    " comp4 được gán giá trị sy-tabix, là số thứ tự của dòng trong bảng mà không có nhóm

    DATA ita LIKE it.

    LOOP AT it INTO DATA(waa) GROUP BY waa-comp1.

      waa-comp4 = sy-tabix.

      APPEND waa TO ita.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = ita name = `ita` ).

    " Chỉ định thứ tự sắp xếp tăng dần cho các nhóm

    DATA itb LIKE it.

    LOOP AT it INTO DATA(wab) GROUP BY wab-comp1 ASCENDING.

      wab-comp4 = sy-tabix.

      APPEND wab TO itb.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = itb name = `itb` ).

    " Sử dụng tùy chọn WITHOUT MEMBERS, cần ràng buộc khóa nhóm

    LOOP AT it INTO DATA(wac) GROUP BY wac-comp1 WITHOUT MEMBERS INTO DATA(keyc).

      " Đảm bảo rằng vùng làm việc (wac) trống

      ASSERT wac IS INITIAL.

      " Thêm khóa nhóm vào bảng str\_table

      APPEND keyc TO str\_table.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = str\_table name = `str\_table` ).

    " Sử dụng khóa nhóm có cấu trúc để nhóm theo nhiều tiêu chí

    DATA itd LIKE it.

    LOOP AT it INTO DATA(wad) GROUP BY ( key1 = wad-comp1 key2 = wad-comp2 ) DESCENDING.

      APPEND wad TO itd.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = itd name = `itd` ).

    " Nhóm theo thứ tự tăng dần và sử dụng GROUP INDEX, GROUP SIZE để lưu trữ thông tin nhóm

    CLEAR str\_table.

    LOOP AT it INTO DATA(wae) GROUP BY ( key = wae-comp1 gi = GROUP INDEX gs = GROUP SIZE ) ASCENDING INTO DATA(keye).

      " Thêm thông tin khóa nhóm vào bảng str\_table

      APPEND |Key component: '{ keye-key }', group index: '{ keye-gi }', group size: '{ keye-gs }'| TO str\_table.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = str\_table name = `str\_table` ).

    " Sử dụng LOOP AT GROUP để thực hiện vòng lặp lồng qua các thành viên của nhóm

    DATA itf LIKE it.

    LOOP AT it INTO DATA(waf) GROUP BY ( key = waf-comp1 gi = GROUP INDEX gs = GROUP SIZE ) ASCENDING INTO DATA(keyf).

      LOOP AT GROUP keyf INTO DATA(memberf).

        " Thêm các thành viên nhóm vào bảng itf

        APPEND VALUE #( comp1 = memberf-comp1 comp2 = memberf-comp2 comp3 = memberf-comp3

        comp4 = |Key component: '{ keyf-key }', group index: '{ keyf-gi }', group size: '{ keyf-gs }'| ) TO itf.

      ENDLOOP.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = itf name = `itf` ).

    " Trích xuất dòng có giá trị cao nhất trong một cột cụ thể trong nhóm

    DATA itg LIKE it.

    LOOP AT it INTO DATA(wag) GROUP BY wag-comp1 ASCENDING.

      LOOP AT GROUP wag INTO DATA(memberg) GROUP BY memberg-comp2 DESCENDING.

        " Thêm thành viên nhóm vào bảng itg và thoát khỏi vòng lặp

        APPEND memberg TO itg.

        EXIT.

      ENDLOOP.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = itg name = `itg` ).

    " Ví dụ tương tự như trên nhưng sử dụng ràng buộc khóa nhóm sau LOOP AT GROUP

    DATA ith LIKE it.

    LOOP AT it INTO DATA(wah) GROUP BY wah-comp1 ASCENDING.

      LOOP AT GROUP wah INTO DATA(memberh) GROUP BY memberh-comp2 DESCENDING.

        " Thêm thành viên nhóm vào bảng ith và thoát khỏi vòng lặp

        APPEND memberh TO ith.

        EXIT.

      ENDLOOP.

    ENDLOOP.

    " Đảm bảo rằng kết quả itg và ith giống nhau

    ASSERT itg = ith.

    " Ghi kết quả ra màn hình

    out->write( data = ith name = `ith` ).

    " Sử dụng điều kiện WHERE trong vòng lặp để lọc các dòng

    DATA iti LIKE it.

    LOOP AT it INTO DATA(wai) GROUP BY wai-comp1 ASCENDING.

      LOOP AT GROUP wai INTO DATA(memberi) WHERE comp3 = abap\_true.

        " Thêm các dòng thỏa điều kiện vào bảng iti và lưu trữ giá trị sy-tabix

        APPEND VALUE #( comp1 = memberi-comp1 comp2 = memberi-comp2 comp3 = memberi-comp3

        comp4 = |sy-tabix: '{ sy-tabix }'| ) TO iti.

      ENDLOOP.

    ENDLOOP.

    " Ghi kết quả ra màn hình

    out->write( data = iti name = `iti` ).

  ENDMETHOD.

ENDCLASS.

**15 Excursions**

**15.1 Improving Read Performance with Secondary Table Keys**

The following example creates two demo internal tables. One without a secondary table key and the other with a secondary table key.

**15.2 Example: Exploring Read Access Performance with Internal Tables**

**15.3 Searching and Replacing Substrings in Internal Tables with Character-Like Data Types**

**15.4 Ranges Tables**

**15.5 Comparing Content of Compatible Internal Tables**

**16 More Information**

**17 Executable Example**