ABAP Programing Cheat Sheet

by louis_lee via cheatography.com/210062/cs/45284/

Data Types

★Define technical properties of all data objects that have these data types, such as the maximum length of a text field.

★Are descriptions only, with no data memory attached except for administrative information.

★Can occur in ABAP programs as bound data types, that is, the type is a property of a data object, or as a standalone data type, that is, the data type is defined independently.

★Can be defined locally in an ABAP program or globally in classes, interfaces and in the ABAP Dictionary (DDIC).

Data types declared in interfaces and in the public visibility section of global classes are also globally visibile. Global classes and interfaces as such are global types to refer to. In classic ABAP, you may stumble on the option to create global data types in type pools, which is not possible in ABAP for Cloud Development. However, the predefined type pool abap can be used in ABAP for Cloud Development.

Reference Types

Describe data objects that contain references to other objects (data objects and instances of classes), which are known as reference variables.

There are two kinds of references: *Data references* (references to data objects) and *object references* (references to objects as instances of classes).

A reference type must be defined either in the ABAP program or in the ABAP Dictionary. There are no built-in reference types in ABAP.

The typical syntax element is ... REF TO ....

Type Conversions, Compatibility and Assignments

A value assignment means that the value of a data object is transferred to a target data object. If the data types of the source and target are compatible, the content is copied unchanged. If they are incompatible and a suitable conversion rule exists, the content is converted. The following cases must be distinguished with regard to the data type:

Type Conversions, Compatibility and Assignments (cont)

★The source and target data types are compatible, i.e. all technical type properties match. The content is transferred from the source to the target without being converted.

★The source and target data types are incompatible, but can be converted. The content of the source is converted according to the conversion rules and then transferred to the target. Two data types are convertible if a conversion rule exists for them. An exception is raised if the content of the source cannot be handled according to the conversion rules.

★If the data objects are neither compatible nor convertible, no assignment can take place. If the syntax check detects this state, a syntax error is raised, otherwise an uncatchable exception is raised when the program is executed.

📘Notes:

Type Conversions, Compatibility and Assignments (cont)

The operands of many ABAP statements are assigned internally according to the assignment rules.

Typically, assignements are made using the assignment operator =. If necessary and applicable, the type is converted implicitly. However, you can also use the conversion operator CONV to convert types explicitly.

For lossless assignments, the lossless operator EXACT can be used to perform checks before the conversion is performed to ensure that only valid values are assigned and that no values are lost in assignments.

In general, no checks are performed on assignments between compatible data objects. If a data object already contains an invalid value, for example, an invalid date or time in a date or time field, it is passed a valid value when the assignment is made to a compatible data object.

Type Conversions, Compatibility and Assignments (cont)

The applies_to_data method of the RTTI class cl_abap_datadescr can be used to check type compatibility.

Non-Admissible Values of Literals

Note recent syntax warnings when using literals that represent invalid values for target types. The following example demonstrates the assignment of literals using admissible and non-admissible values. You can copy and paste the code into a demo class in your SAP BTP ABAP Environment to explore the syntax warnings.

```
DATA char3 TYPE c LENGTH 3.

"Value is admissable and convertible
char3 = 'abc'.

"Non-admissable value assigned to the
target (type c length 6)
char3 = 'defghi'.

DATA date TYPE d.

"Value is admissable and convertible
date = '20250101'.

"Non-admissable value assigned to the
target
"Type i
date = 20250101.
"More characters than type d
date = '20250101234'.
"Fewer characters than type d
date = '202511'.
```

Creating Structured Types

★The type name is preceded by BEGIN OF (which marks the beginning of the structured type definition) and END OF (the end of the definition).
★The components - at least one must be defined - are listed in between.
★Such structured type definitions are usually grouped together in a chained statement, i.e. TYPES is followed by a colon, and the components are separated by commas.
★The simplest structures and structured types have elementary components.
★As mentioned above, the components can be of any type, i.e. they can be of structured types themselves, internal table types, or reference types.
★You can use the TYPE and LIKE additions for the types of the components. You can use the LINE OF addition to refer to a table type or an internal table.

Outside of classes, you can also refer to DDIC types using LIKE (... comp11 LIKE ddic_type, ...). If you actually want to refer to an existing data object, but due to typing errors you inadvertently specify a name that exists as DDIC type, errors may be unavoidable.

Demo Example Data Type And Data Object

Demo

Creating Anonymous Structures

Using the instance operator NEW and CREATE DATA statements, you can create anonymous data objects, such as anonymous structures. The NEW addition of the INTO clause of an ABAP SQL SELECT statement also creates an anonymous data object. As outlined below, you can access the components or the entire data objects by dereferencing.

Populating Structures

★You can copy the content of a structure to another using the assignment **operator** =. In general, note that special conversion and comparison rules apply to value assignments involving structures.

★Using the **VALUE Operator**:The VALUE operator can be used to construct the content of complex data objects such as structures or internal tables.

Populating Structures (cont)

★Using the **NEW Operator**:Using the instance operator NEW, you can create anonymous data objects, such as anonymous structures. You can access the components or the entire data objects by dereferencing.

★You can use statements with **MOVE-CORRESPONDING and the CORRESPONDING operator** to assign values to structure components, especially when assigning values from a source structure to a target structure which have incompatible types and/or differently named components.

The CL_ABAP_CORRESPONDING system class is available for making assignments. The INTO clause of ABAP SQL statements has the CORRESPONDING addition. There, the following basic rule applies, which affects the value assignment: Without the CORRESPONDING ... addition, column names do not matter, only the position. With the CORRESPONDING ... addition, the position of the columns does not matter, only the name.

Including Structures

★INCLUDE TYPE and INCLUDE STRUCTURE statements are used in the context of local structures.

★Structured data objects and types created with ... BEGIN OF... END OF ... can use this syntax to include components of another structure, whether it is a locally defined or global structure, without creating substructures.

★INCLUDE TYPE can be used to include a structured type.

★You can use INCLUDE STRUCTURE to include a structure.

They are not additions of ... BEGIN OF ... END OF ... but individual ABAP statements. If you use a chained statement with a colon to declare the structure, the inclusion of other structures with these statements interrupts the chained statement, that is, the components of the included structures are included as direct components of the superstructure.

Including Structures (cont)

By using the optional AS addition and specifying a name, the included components can be addressed by this common name as if they were actually components of a substructure.
The optional RENAMING WITH SUFFIX addition, followed by a name, gives the included components a suffix name to avoid naming conflicts with other components.

Demo ABAP Structure

Executable Example

Generic Types

★Generic types are available with which formal parameters of methods or field symbols can be specified.
★At runtime, the actual data type is copied from the assigned actual parameter or memory area, i.e. they receive the complete data type only when an actual parameter is passed or a memory area is assigned.

The TYPE REF TO addition types as a reference variable. A generic type cannot be specified after REF TO. A typing with TYPE REF TO data and TYPE REF TO object is considered as completely typing.

Elementary data types

Elementary (or scalar) data types are based directly on a set of built-in ABAP types.
Numeric types: Integers (b, s, i, int8), decimal floating point numbers (decfloat16, decfloat34), binary floating point numbers (f), and packed numbers (p)
Character-like types: text fields (c) and numeric text fields (n)
Byte-like type: byte fields (x)
Character-like date and time types: date fields (d) and time fields (t)
Time stamp type for time stamp fields (utclong).
Character-like type for text strings (string)
Byte-like type for byte strings (xstring)

The data types c, n, x, and p are incomplete, i.e., generic data types, with respect to their length. The type definition syntax has a special addition for this (LENGTH). In addition, p is also generic with respect to the number of decimal places (DECIMALS addition). See more about generic types in the following sections.

Creating Types/Data Objects at Runtime

Using Runtime Type Services (RTTS) you can ...
★get type information on data objects, data types or instances at runtime (Runtime Type Identification (RTTI)).
★define and create new data types as type description objects at runtime (Runtime Type Creation (RTTC)).

ABAP Enumerated Types and Objects

★ABAP supports the concept of enumerations.
★Enumerations are a mixture of types and constants.
★An enumerated type specifies a value set in addition to the actual type properties.
★Enumerated objects - data objects with an enumerated type - are mainly used to check allowed values. This usually restricts the actual parameters passed to methods to the enumerated values defined in the class. Enumerated variables are variable enumerated objects. They can only contain the associated enumerated values.
★CDS enumerated types are also available.

Creating Structures

To create a structure (i.e. a structured data object) in an ABAP program, you can use the DATA keyword.
It works in the same way as the TYPES statement above.
Unlike the TYPES statement, you can use the VALUE addition to set default values.

The keywords CLASS-DATA and CONSTANTS can also be used to create structures. In principle, they represent special cases of the general statement shown above.
Structures can also be created inline using DATA(...) or FINAL(...).

Structures

Structures ...
★are data objects with structured data types (which is a complex data type because it is composed of other data types).
★consist of a sequence of components of any data type, that is, the components of a structure can be, for example, elementary data objects, structures themselves, internal tables or references.

Structures (cont)

★are used to combine different data objects that belong together. A typical example is an address. It has several components, such as name, street, city, and so on, that belong together.
play an important role in the context of internal tables and database tables. Structured types serve as line types for these tables. Most internal tables across ABAP programs may have structured line types. For database tables, there is no alternative to structured line types.
★can be created locally in an ABAP program and globally.

Variants of Structures

★Flat structures contain only elementary types that have a fixed length, that is, there are no internal tables, reference types or strings as components. Nesting does not matter in this context. Even a nested structure is considered flat unless a substructure contains a deep component.
DATA: BEGIN OF struc,
comp1 TYPE i,

Variants of Structures (cont)

comp2 TYPE c LENGTH 15,
comp3 TYPE p LENGTH 8 DECIMALS 2,
...,
END OF struc.
★Nested structures: At least one component of a structure is a substructure, that is, it refers to another structure. The following example has multiple substructures.
DATA: BEGIN OF address_n,
BEGIN OF name,
title   TYPE string VALUE `Mr.`,
prename TYPE string VALUE `Duncan`,
surname TYPE string VALUE `Pea`,
END OF name,
BEGIN OF street,
name TYPE string VALUE `Vegetable Lane`,
num TYPE string VALUE `11`,
END OF street,
BEGIN OF city,
zipcode TYPE string VALUE `349875`,
name   TYPE string VALUE `Botanica`,

Variants of Structures (cont)

END OF city,
END OF address_n.
★Deep structures: Contains at least one internal table, reference type, or string as a component.
DATA: BEGIN OF address_d,
name TYPE string VALUE `Mr. Duncan Pea`,
street TYPE string VALUE `Vegetable Lane 11`,
city TYPE string VALUE `349875 Botanica`,
details TYPE TABLE OF some_table WITH EMPTY KEY,
END OF address_d.

The data types of DDIC types are all flat (not nested) structures. Exception: Components of type string can be contained.
Work areas of ABAP SQL statements cannot contain any deep components other than strings among others.
Especially for assignments and comparisons of deep structures, the compatibility of the source and target structure must be taken into account.

sy Structure

The sy (or syst) structure is a built-in data object.
The components of the structure represent ABAP system fields.
These fields, filled by the ABAP runtime framework, can be used to query system information and more.
Typically, they should only be read, and not overwritten.
Prominent system fields are the following
 sy-subrc: Return code of many ABAP statements; typically, the value 0 indicates success
 sy-tabix: Row index of internal tables
 sy-index: Loop pass index
These ones and others can be used in ABAP for Cloud Development. However, most of the fields should not be used in ABAP for Cloud Development (indicated by a syntax warning) because they refer to Standard ABAP contexts (e.g. classic dynpros and lists), or their values are not relevant in a cloud context.

Clearing Structures

You can reset individual components to their initial values and clear the entire structure using the CLEAR keyword. Note that FREE statements also deletes the content, but they also release the initially allocated memory. space.

Complex Data Types

Are composed of other types.
Structured types: Represent a sequence of arbitrary data types (i.e., they can be elementary, reference, or complex data types). The typical syntax element for the local definition of a structure is ... BEGIN OF ... END OF ....
Table types: Consist of a sequence of any number of lines of the same data type. It can be any elementary type, reference type, or complex data type. The type definition includes other properties such as the table category (defines how tables can be accessed) and table key (to identify the table lines). The typical syntax element is ... TABLE OF ....

Complex Data Types (cont)

Enumerated types: Specify a set of values in addition to the actual type properties. The typical syntax element is ... BEGIN OF ENUM ... END OF ENUM ....
Mesh types: Special structured type that contains only table types with structured line types as components that can be linked using mesh associations. The typical syntax element is ... BEGIN OF MESH ... END OF MESH ....
BDEF derived types: RAP-specific structured and table types. The typical syntax elements are ... TYPE STRUCTURE FOR ... and ... TYPE TABLE FOR .... More information can be found here and in the ABAP cheat sheet on ABAP EML.

Data objects

★Are objects (or instances) of a data type (similar to objects/instances of classes in ABAP Objects).
★Occupy memory space and exist in different forms, for example, numeric or textual data can be contained in data objects.

Data objects (cont)

★The type of data that a data object can receive is determined by its data type.
★Like data types, their existence and visibility depend on the declaration context.
★Are usually used in ABAP statements by specifying them in the operand position.

Ranges Tables

★Internal tables that have the predefined columns SIGN, OPTION, LOW, and HIGH
★Declared with the TYPE RANGE OF addition in DATA and TYPES statements
★Used to store range conditions that can be evaluated in expressions using the IN operator (each row in the table represents a separate comparison)

Typed Literals in ABAP SQL

Literal whose data types is defined by specifying a built-in dictionary type explicitly.
★Available for most but not all ABAP Dictionary data types.
★Can be used in ABAP SQL and in ABAP CDS.

Typed Literals in ABAP SQL (cont)

★Advantages of typed literals over untyped literals
★Allow type-safe use of literals
★Eliminate the need for (implicit type) conversions and casts, which can lead to surprising or erroneous results; also consider the conversion costs in terms of performance (typed literals are passed to the database and evaluated there without ABAP-specific type conversions).
★For better readability (you can immediately see what type is being used)

Globally Available Structures and Structured Types

★Apart from the local declaration of a structured type, you can create such a type, for example, as global DDIC structure in the ABAP Dictionary. Such a DDIC structure defines a globally available structured type (DDIC type).

★There are other structured types available globally, which may be the structured types most commonly used in ABAP programs:

Database tables defined in the ABAP Dictionary can be used as data types just like DDIC structures in an ABAP program.

Globally Available Structures and Structured Types (cont)

This means that when you create a structure in your ABAP program, for example, you can simply use the name of a database table to address the line type of the table. The structure you created will then have the same structured type as the database table. Typically, you use the database tables to create structures of such a type, or internal tables of such a structured line type, to process data read from the database table in structures or internal tables.
. **Various CDS entities** are globally available structured types. For example, a CDS view entity represents a structured data type and can be used as such in ABAP programs (but not in the ABAP Dictionary).

Structures and structured data types can be defined in the public visibility section of global classes or in global interfaces and then used globally.

Creating Structures by Inline Declaration

★This is particularly useful for declaring data objects at the operand positions where you actually need them.
★In this way, you can avoid an extra declaration of the structure in different contexts.
★You can use the declaration operator using DATA(...). The FINAL declaration operator is used to create immutable variables.
★You can also create structures using the VALUE operator (and also fill them as shown below). Without specifying component values in the parentheses, you create an initial structure.

Accessing (Components of) Structures

★Structures can be accessed as a whole. You can also address the individual components of structures at the appropriate operand positions.
★To address the components, use the structure component selector -.

Accessing (Components of) Structures (cont)

★For variables with reference to a structured data object, the object component selector -> can be used: ...dref->comp .... The following syntax also works, but is less convenient: ... dref->*-comp ....
★ADT and the ABAP Editor provide code completion for structure components after the component selectors.

Processing Structures

Structures in ABAP SQL Statements
Structures in Statements for Processing Internal Tables

Boxed Components

In structures, boxed components represent nested structures managed by an internal reference.
Currently, static boxes are supported as boxed components, enabling initial value sharing. Find more information here.
When used:

Boxed Components (cont)

Optimize memory consumption for structures used repeatedly, such as in internal tables with nested structures. Without boxed components, memory increases line by line, even if the nested structure is initial. With boxed components, memory does not increase when nested structures are initial, and only reads are performed.

Enhance runtime performance since assignments for components with active initial value sharing require only the internal reference, not additional data to be copied. Boxed components allocate memory when there is write access to at least one component or when a field symbol is assigned or data reference points to at least one component.