ABAP Fundamentals
Course Objectives

Topic Name	When you have completed this topic, you should be able to
ABAP and the Development Workbench	recognize the basic concepts of ABAP and access the ABAP Workbench.
The ABAP Workbench Tools	recognize which ABAP Workbench tool to use in a given scenario.
The ABAP Data Dictionary	recognize the characteristics of the ABAP data dictionary objects.
Using ABAP, Workbench Tools, and ABAP Dictionary	recognize the features of ABAP and the functions of the ABAP tools and data dictionary objects.
Introducing Tables in ABAP	recognize the components of a table definition.
Creating Domains and Data Elements in ABAP	create a domain and a data element in a given scenario.
Creating ABAP Tables	recognize how to create a table in a given scenario.
Creating ABAP Domains, Data Elements, and Tables	create domains and data elements and use them to create Data Dictionary tables for a given scenario.

ABAP and the Development Workbench

Learning objective

After completing this topic, you should be able to recognize the basic concepts of ABAP and access the ABAP Workbench.

1. Introducing ABAP

Advanced Business Applications Programming (ABAP) is an SAP fourth generation (4GL) programming language. All modules of SAP R/3 – Sales and Distribution (SD), Materials Management (MM), Financial Accounting (FI), and Production Planning and Control (PP) – are written in ABAP.

All SAP modules consist of a set of transactions. A transaction is made up of programs that enable you to create, change, or display information stored in SAP databases.

Based on the requirements of a company, you can customize existing programs or create new programs using ABAP.

Say you are working as an ABAP programmer for Award Sportswear, which specializes in casual and formal sportswear.

When the management wants to find out the amount receivable from its customers, the finance officer in the Finance department generates a Customer Balance report.

In addition to the current information displayed in the Customer Balance report, the management wants this report to display information about the products sold to each customer during the current fiscal year.

In this case, as an ABAP programmer, you need to customize the ABAP program for the Customer Balance report. You can also create a new program that meets the requirements of Award Sportswear.

The management at Award Sportswear wants to streamline the after sales service of the company. For this, management has decided to integrate the customers' non-SAP procurement applications with the company's SAP R/3 system. To integrate SAP R/3 applications with non-SAP applications, you decided to upgrade the SAP R/3 system to the NetWeaver platform. After integrating the applications with the NetWeaver platform, Award Sportswear can use ABAP to electronically perform business transactions with its customers.

Question

What can you do using ABAP?

Options:

1. Create new programs in SAP R/3
2. Customize existing programs depending on the requirements of a company
3. Integrate SAP and non-SAP applications

Answer

ABAP enables you to create new programs in SAP R/3 and customize an existing SAP R/3 program.

Option 1 is correct. You can create new programs from scratch using ABAP. For example, there is a business requirement to generate a certain report. SAP doesn't provide any report for the given requirement. In this case, you need to create new programs that help you in generating the required report.

Option 2 is correct. Using ABAP, you can customize SAP modules. For example, to view the total amount the company owes to all its vendors and to view information related to the products purchased from each vendor in a common report, you can customize the existing Vendor Balance report using ABAP.

Option 3 is incorrect. By upgrading the SAP R/3 system to the NetWeaver platform, you can integrate SAP R/3 applications with non-SAP applications. After integrating the applications with the NetWeaver platform, you can use ABAP to perform business transactions.

2. ABAP programs

When you perform a transaction in SAP R/3, an ABAP program runs in the background. This program extracts relevant information from SAP databases based on the parameters specified for the transaction and displays the output. Each screen you view in SAP R/3 is created using an ABAP program.

There are two types of ABAP programs, report and dialog.

The two types of report programs are

- Traditional report program
- Interactive report program

Traditional report program

A traditional report program is an input/output screen program. This program is activated and executed when you enter specific input parameters in the input fields and run the report. Based on the input parameters, SAP R/3 displays the output in list format on the screen. After this, the program concludes and you can't interact on the output screen.

A traditional report program consists of only two screens. The input parameter screen is called the selection/input screen, and the second screen is called the output screen. The output screen doesn't have any input fields.

Suppose you want to find out the balance of a customer who has purchased products from your company. For this purpose, you may need to generate the Customer Balance

report. This report provides you with an input screen where you can specify the code of the customer.

When you execute the Customer Balance report, the report program activates and displays the amount receivable information of the customer on the output screen. You cannot interact further with the information displayed on the output screen.

Interactive report program

An interactive report program offers more flexibility than the traditional report program. Instead of two screens, an interactive program consists of multiple screens and you can drill down through several screens to view the desired information. Unlike the traditional report program, you can interact with the output screens, but you cannot manipulate the information displayed on the output screen.

An interactive report program enables you to

- sort the information displayed on the output screen
- print selective information
- control the number of fields displayed on the output screen

Suppose you create an interactive report program to find the contact information of all customers. The first screen of the program enables you to specify input parameters, and the second screen lists the customers of the company. When you click the customer code on the second screen, the third screen opens, displaying contact information for that customer. You cannot manipulate the information displayed on the output screen.

In a nutshell, a traditional report program extracts information from SAP databases and displays the output on the screen.

An interactive report program, as the name suggests, offers user interaction on the output screen by enabling you to sort and print selective information, and control the number of fields on the output screen.

A dialog program consists of multiple screens and enables you to manipulate the information displayed on the output screen. Using this program, the output screen can contain input fields, push buttons, and multiple scrollable areas.

Suppose you want to change the contact information of a customer. In this case, you specify the customer code on the first screen. The next screen displays information related to the specified customer. You modify the contact information of the customer, and save it. The contact information of the customer is updated in the SAP database.

An interpreter interprets the ABAP programs, report and dialog. An interpreter is a program that executes other programs. The interpreter reads each line of code written to build an ABAP program. If an error, such as a syntax error, is encountered, the interpreter reports the line number at which the error was encountered and stops reading the rest of the code in the program.

When an ABAP program is executed, a runtime object of the program is automatically created. The runtime object, which is also called the generated form of the program,

displays the output of the program. If you make any changes to the program, the runtime object is automatically regenerated the next time you execute the program.

Question

Match each ABAP program to its description.

Options:

1. Dialog program
2. Interactive report program
3. Traditional report program

Targets:

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
nables you to manipulate information

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
nables you to only view information

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
nables you to sort information

Answer

The dialog program enables you to manipulate information. The traditional report program enables you to only view information. An interactive report program enables you to sort information.

The dialog program consists of multiple screens and enables you to manipulate the information displayed on the output screen.

An interactive report program offers more flexibility than the traditional program report. This report program consists of multiple screens and enables you to sort the information, print selective information, and control the number of fields on the output screen.

A traditional program is a basic program that displays information based on input parameters. This report consists of two screens, selection/input and output. You cannot have any user interaction on the output screen.

3. Components of an ABAP program

An ABAP program consists of several components.

The components of an ABAP program are

- source code
- attributes
- text elements

- documentation
- variants

source code

Source code is a set of instructions specified in a program that are interpreted by an interpreter. For example, the code you write in an ABAP program is considered as a source code.

attributes

An attribute is a property that you define for a program. The properties are

- title and language of the program
- creation and the last modified date of the program
- status of the program: active or inactive

text elements

A text element is a short description of report headers. For example, you create a text element CBR for the report heading, Customer Balance Report. Now in the program, you can specify CBR as the report heading. During program execution, the text element CBR is replaced with Customer Balance Report.

documentation

The documentation of a program helps developers understand the flow of the program. The documentation clearly states what a program does, when the program will be executed, and which format the output of the program will be displayed in.

variants

A variant is used to save a set of input values that are used frequently. For example, to generate a monthly sales report you always require all the employee codes of sales persons and all the regions. You save these input values as a variant. When you generate a monthly sales report, the respective fields are populated with the variant values.

Question

Match each component of an ABAP program to its description.

Options:

1. Attribute
2. Documentation
3. Source code
4. Text element

Targets:

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
elps the developer to understand the program flow

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
s a property that you define for a program

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
s a set of instructions specified in a program

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
s a short description of the text specified in a program

Answer

Documentation helps the developer understand program flow. An attribute is a property you define for a program. Source code is a set of instructions specified in a program. A text element is a short description of the text specified in a program.

An attribute is a property that you define for a program. The various properties are program title, program status, program creation date, and last modified date of the program.

Documentation is used to record all information about a program. This helps the developer understand the flow of the program.

Source code is a set of instructions specified in a program that is interpreted by an interpreter. For example, the instructions that you write in an ABAP program are considered source code.

A text element is a short description specified in a program. A text element can be created for titles, headers, or any other text you want to replace after the program executes.

4. ABAP Workbench

ABAP Workbench is a development environment that SAP R/3 offers. Using ABAP Workbench, you can

- create, modify, and test ABAP programs
- create user interfaces
- create and access databases
- create Web services

You can access ABAP Workbench from the SAP Easy Access screen. The screen is already opened for you.

To access ABAP Workbench, you double-click **Tools** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**.

Alternatively, you can type `SE80` in the Command field on the standard toolbar and press **Enter**.

Using the Object Navigator screen of ABAP Workbench that appears, you can develop and modify ABAP programs.

Question

Suppose you have created a program to find the status of all the materials lying in all the warehouses in the company. You want to test the functionality of the program. For this, you need to access ABAP Workbench. The SAP Easy Access screen is opened for you.

What are the most appropriate steps required to access ABAP Workbench?

Options:

1. Double-click **Administration** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**.
2. Double-click **Customizing** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**.
3. Double-click **Tools** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**.

Answer

You double-click **Tools** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator.**

Summary

Advanced Business Applications Programming (ABAP) is an SAP programming language that enables you to customize the existing modules of SAP R/3.

There are two types of ABAP programs, report and dialog. A report program is of two types, traditional report program and interactive report program. A traditional report program extracts information from SAP databases and displays the output on the screen. An interactive report program offers more interactivity than a traditional report program. A dialog report helps manipulate the information displayed on the output screen.

The components of an ABAP program include source code, attributes, text elements, documentation, and variants.

ABAP Workbench is a graphical user interface (GUI) that consists of tools that enable you to create and modify ABAP programs.

Table of Contents

| Top of page |

| Learning objective |

| 1. Introducing ABAP |

| 2. ABAP programs |

| 3. Components of an ABAP program |

| 4. ABAP Workbench |

| Summary |

The ABAP Workbench Tools

Learning objective

After completing this topic, you should be able to recognize which ABAP Workbench tool to use in a given scenario.

1. Development tools

SAP R/3 applications consist of programs that are written using Advanced Business Applications Programming (ABAP). ABAP Workbench is a programming environment that offers tools to

- write and modify ABAP programs
- design screens, menu bars, and toolbars
- test and debug ABAP programs for efficiency
- track versions of ABAP programs

To work effectively with ABAP Workbench, SAP has broadly categorized the ABAP Workbench tools into

- debugging
- development
- navigation
- organizing

debugging

Debugging tools enable you to debug ABAP programs.

development

Development tools enable you to create and modify ABAP programs.

navigation

Navigation tools enable you to navigate through programs that have been created using other ABAP Workbench tools.

organizing

Organizing tools enable you to avoid the creation of duplicate objects and move them from one server to another. In SAP R/3, there are three types of servers: development, testing, and production.

There are five development tools available in ABAP Workbench. They are

- ABAP Data Dictionary
- ABAP Editor
- Function Builder
- Menu Painter
- Screen Painter

ABAP Data Dictionary

The ABAP Data Dictionary tool contains definitions of all objects created in SAP. The ABAP Data Dictionary objects are table, view, data element, domain, structure, table type, lock object, search help, and online help.

The ABAP Data Dictionary is also called ABAP Dictionary.

All runtime objects, such as programs, collect information from the ABAP Data Dictionary to execute the program.

ABAP Editor

ABAP Editor enables you to

- create and edit programs
- verify syntax
- test the functionality of programs
- navigate through online ABAP help

Say you want to develop a program to create a sales report for the first quarter in a specific format. The report should consist of four columns and a header. The header should display the report name, the first column should display the names of all customers, and the remaining three columns should display the months April, May, and June. To create a program for this sales report, you need to use the ABAP Editor tool.

Function Builder

Function Builder acts as a central repository for all the predefined and user-defined functions. A function is a sequence of code that performs a specific task.

Before you call a function in an ABAP program, you need to ensure that the function is working properly. To ensure this, you need to execute the function using the Function Builder tool. When you execute a function in Function Builder, the tool checks for syntax errors.

Suppose you have created the DIS_RATE_GETLIST function, which checks the discount rates offered for a product in different states. If you want to find the discount rates offered for a product in Texas, you can use the DIS_RATE_GETLIST function.

Menu Painter

Menu Painter enables you to design user interfaces in SAP R/3. A user interface consists of a graphical user interface (GUI) status and a GUI title.

A GUI status consists of a menu bar, a standard toolbar, an application toolbar, and function keys. A GUI title consists of a title bar. Based on requirements, you program interface elements.

Say you have already designed a Customer Satisfaction form. The form consists of input fields and push buttons. You use input fields to specify information, and you use buttons to save and view customer feedback. You want the form to offer alternate ways to save and view customer information. In this case, you need to design a user interface that consists of a menu bar. To design a menu bar, you need to use the Menu Painter tool.

Screen Painter

Screen Painter enables you to create screens for all transactions that you perform in SAP R/3. A screen consists of various elements such as input fields, push buttons, checkboxes, radio buttons, and scroll bars.

Suppose you want to design a screen for the Customer Satisfaction form, which consists of screen elements, such as input fields and push buttons. To create the screen elements and design the Customer Satisfaction form, you need to use the Screen Painter tool.

Question

Say you are working as an ABAP programmer for Marine Computer Solutions. You want to design a user interface in SAP R/3. Based on requirements, you program the interface elements.

Which tool would you use to achieve this?

Options:

1. ABAP Editor
2. Function Builder
3. Menu Painter
4. Screen Painter

Answer

You use the Menu Painter tool to design a user interface.

Option 1 is incorrect. The ABAP Editor tool is used for developing and modifying ABAP programs. ABAP Editor is not used to design a user interface.

Option 2 is incorrect. The Function Builder tool provides you with the predefined SAP R/3 functions. Based on requirements, you can create new functions using this tool. Function Builder is not used to design a user interface.

Option 3 is correct. The Menu Painter tool is used to a design user interface. A user interface consists of a menu bar, a title bar, a standard toolbar, an application toolbar, and function keys.

Option 4 is incorrect. The Screen Painter tool is used to design a screen. The screen elements placed on a screen are input fields, push buttons, checkboxes, radio buttons, and scroll bars. Screen Painter Builder is not used to design a user interface.

2. Navigation tools

Navigation tools enable you to navigate objects created using other tools. You can also modify and save the objects using navigation tools. For example, you have written a program to design a screen using the Screen Painter tool. Using the navigation tools, you can modify and save the program in ABAP Editor without using Screen Painter.

You can also search particular objects and applications that SAP R/3 provides for SAP modules.

Note

Objects are small programs related to screen design, menu design, and other functions.

There are four navigation tools available in ABAP Workbench.

- Application Hierarchy
- Data Browser
- Object Navigator
- Repository Information System

Application Hierarchy

The Application Hierarchy tool displays all the applications written in ABAP. The applications are arranged hierarchically under respective modules that SAP R/3 offers.

For example, if you want to find the applications that SAP R/3 provides for the Financial Accounting (FI) module, you should navigate through the applications under the FI module.

Data Browser

The Data Browser tool enables you to navigate through data stored in tables. Using this tool, you can

- find selective rows/columns
- add data to rows/columns
- modify existing data in rows/columns
- delete existing data from rows/columns

However, to create and update table records with the Data Browser, the **Table maintenance allowed** option must be set for the table.

Object Navigator

The Object Navigator tool enables you to navigate through the objects stored in ABAP Workbench. The Object Navigator tool enables you to create, change, and manage objects. You can edit the object programs directly using the Object Navigator tool. You don't need to access tools related to the program.

For example, you have designed a screen using the Screen Painter tool. You can modify the program related to screen design in the ABAP Editor tool by using the Object Navigator tool.

Repository Information System

The Repository Information System tool enables you to search an object in SAP R/3. The objects are placed in the order of the ABAP Data Dictionary objects – tables, data elements, and domains – and development objects – packages, programs, and functions.

Question

Match each navigation tool to the corresponding situation where you would use it.

Options:

1. Application Hierarchy
2. Data Browser
3. Object Navigator
4. Repository Information System

Targets:

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
ou want to delete data from an existing row in a table

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
ou want to view the components that SAP R/3 provides for the Human Resources module

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
ou want to locate a table stored in SAP R/3

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
ou want to modify a program written for screen elements from ABAP Workbench

Answer

The Data Browser is used to delete data from an existing row in a table. Application Hierarchy is used to view the components that SAP R/3 provides for the Human Resources module. Repository Information System is used to locate a table stored in SAP R/3. Object Navigator is used to modify a program written for screen elements from ABAP Workbench.

All SAP R/3 applications offered for SAP R/3 modules can be viewed using the Application Hierarchy tool. The applications are grouped and displayed under their respective SAP R/3 modules. For example, all the applications related to human resource are grouped and displayed below the Human Resource module of SAP R/3.

When you want to add, modify, and delete data from a table, you need to use the Data Browser tool.

Object Navigator provides you with the functionality of modifying and saving programs, which are written using other tools, in ABAP editor. For example, you can modify and save screen elements, created using Screen Painter, in ABAP Editor.

All the objects you create using different tools are listed in Repository Information System. Table is one of the ABAP Data Dictionary objects. You can search the desired object using the Repository Information System tool.

3. Debugging tools

To track the exact location of errors in an ABAP program, you use debugging tools. After identifying errors, you can debug them.

Debugging tools also help find which program has load on the central processing unit (CPU) and needs to be optimized to enhance the performance of the program.

There are four debugging tools available in ABAP Workbench. They are

- On Line Debugger
- Run Time Analysis
- SQL Trace
- System Log

On Line Debugger

The On Line Debugger tool is used during the development phase of a program when you fail to obtain the desired output from the program. Using this tool, you can start the program in debug mode, view the data stored in variables or tables at runtime, and analyze the execution of each statement written in the program.

Say you have written an ABAP program to generate a purchase report. When you execute the report, the report fails to provide the desired output. In this case, you use the On Line Debugger tool and start the program in debug mode. You then analyze the execution of each statement written in the program.

Run Time Analysis

The Run Time Analysis tool analyzes the performance of a program. This tool is useful when you want to know the efficiency and performance of the program you developed. The performance criteria are

- execution time
- which part of the program has extensive load on the CPU
- the amount of data that can be retrieved from the database
- the usage of system resources

You have written an ABAP program to generate a sales report for a company with around 50,000 customers. During report execution, the report connects to SAP databases and accesses the Customer table. The amount of customer data stored in the table is huge. As a result, the process of report execution takes longer to display the information. To avoid this, you need to optimize the ABAP program and improve the performance of the program.

The Run Time Analysis tool displays information related to the performance of the program. Based on the information, you can analyze and improve the performance of the program.

SQL Trace

The SQL Trace tool tracks the attempts that the ABAP programs make to access the database. The tool enables you to view and analyze which SQL statements – DECLARE, PREPARE, OPEN, and FETCH – written in ABAP programs are executed.

SQL Trace records the number of inserts, deletes, and retrievals in a database. All database activity, either for a user or for an entire SAP R/3, is recorded in SQL Trace. In addition, you can track the time that SQL statements take to execute a query.

System Log

The System Log tool stores detailed information of all events – errors and warnings – that occur during program execution and determine the health of the SAP R/3 system. You use this tool when you want a detailed view of the events that have occurred during program execution.

For example, when performing a transaction, you encountered an error. And you want to track the cause of the error. So you want to view the system log messages associated with the errors. To view the system log messages associated with the errors, you need to use the System Log tool.

Question

Say you are working as an ABAP programmer for Cargoflow, a courier company. You have developed a program to create a purchase report. The vendor information stored in SAP databases is enormous. The purchase report takes around 2 minutes to show the output of the program. You want to improve the performance of the program.

Which tool would you use to achieve this?

Options:

1. On Line Debugger
2. Run Time Analysis
3. SQL Trace
4. System Log

Answer

You use the Run Time Analysis tool to improve the performance of a program that is required to create a purchase report.

Option 1 is incorrect. You use the On Line Debugger tool when the desired output is incorrect. After the program is executed, you start the program in debug mode and analyze the root cause of the program displaying incorrect output.

Option 2 is correct. When the purchase report delays the desired output, you use the Run Time Analysis tool. Using this tool, you can analyze the performance of the program that is developed for the purchase report.

Option 3 is incorrect. You use the SQL Trace tool to trace all attempts that the program makes to access vendor information from SAP databases.

Option 4 is incorrect. You use the System Log tool for detailed information about all events that occur in SAP R/3. The tool displays the errors, warnings, and information associated with the events.

4. Organizing tools

In SAP R/3, programs are developed, tested, and deployed on different servers. This is done to ensure that before a program is made available to users, the program is bugfree and functioning properly.

The organizing tools move ABAP programs from one server to another. However, you can move only those programs for which you have generated a request in SAP R/3.

The organizing tools available in ABAP Workbench are

- Transport System
- Workbench Organizer

Transport System

In SAP R/3, there are three types of servers – development, testing, and production. An ABAP programmer uses a development server to modify ABAP programs. The testing team uses the test server to test the program before releasing the changes to the production server. Finally, you perform all day-to-day business transactions on the production server. To move ABAP programs from one server to another, you need to use the Transport System tool.

Workbench Organizer

Workbench Organizer avoids the creation of duplicate objects and generates requests for moving objects from one server to another. For example, you want to develop a program named *Zprogr1*. If a program with the same name exists in SAP R/3, the tool won't allow you to create a program with that name. In addition, if you want to move a program from the development server to the production server, you need to generate a request using the Workbench Organizer tool.

Question

You have modified and tested a program on the testing server. The modified program enhances the functionality of the purchase order report. You want to make this program available to all users so that they can take advantage of the new functionality.

Which tool would you use to achieve this?

Options:

1. ABAP Editor
2. Data Browser
3. Transport System
4. Workbench Organizer

Answer

You use the Transport System tool to move an ABAP program from one server to another.

Option 1 is incorrect. ABAP Editor is a development tool that is used to develop and modify programs. You don't use the ABAP Editor tool to move programs from one server to another.

Option 2 is incorrect. Data Browser is a navigation tool that is used to view and modify the data in a table. This tool is not used to transfer programs from one server to another.

Option 3 is correct. Transport System is an organizing tool that is used to move a program developed for the purchase order report from one server to another.

Option 4 is incorrect. Workbench Organizer is an organizing tool that is used to avoid the creation of duplicate objects and generate requests for moving objects from one server to another.

5. Accessing Workbench tools

You can access ABAP Workbench tools in two ways – using a transaction code or using SAP menu.

You want to access the Data Browser tool using a transaction code. The SAP Easy Access screen is opened for you.

To access the Data Browser tool using a transaction code, you type `SE16` in the Command field on the standard toolbar and click **Enter** (this is the button with the checkmark).

Alternatively, you can type `SE16` in the Command field on the standard toolbar and press **Enter**.

Using the Data Browser: Initial Screen that appears, you can view and modify the data stored in tables.

Similarly, you can access the Data Browser tool within ABAP Workbench using SAP menu. The SAP Easy Access screen is opened for you.

To access the Data browser tool, you double-click **Tools** - **ABAP Workbench** - **Overview** - **SE16 - Data Browser**.

Alternatively, you can type `SE16` in the Command field on the standard toolbar and press **Enter**.

Using the Data Browser: Initial Screen that appears, you can view and modify data stored within tables.

Supplement

Selecting the link title opens the resource in a new browser window.

Launch window

View the transaction codes of all tools available in ABAP Workbench.

Question

Say you are working as an ABAP programmer for Accounts Now, a finance company. Customer-related information is stored in the ZCustomer table. You want to find data related to a specific customer. For this, you need to retrieve data from the ZCustomer table. To retrieve the data, you need to access the Data Browser tool. The SAP Easy Access screen is opened for you.

What are the most appropriate steps required to access the Data Browser tool using a transaction code?

Options:

1. Type `SE09` and click **Enter**.
2. Type `SE16` and click **Enter**.
3. Type `SE83` and click **Enter**.
4. Type `SE95` and click **Enter**.

Answer

You type `SE16` and click **Enter**.

Summary

ABAP Workbench is a programming environment that offers development, navigation, debugging, and organizing tools to create and maintain SAP R/3 applications. You use development tools – ABAP Data Dictionary, ABAP Editor, Function Builder, Menu Painter, and Screen Painter – to create or modify ABAP programs.

You use navigation tools – Application Hierarchy, Data Browser, Object Navigator, and Repository Information System – to navigate through ABAP programs.

You use debugging tools – On Line Debugger, Run Time Analysis, SQL Trace, and System Log – to debug ABAP programs and optimize them for efficiency.

You use organizing tools – Transport System and WorkBench Organizer – to avoid the creation of duplicate objects and to move programs from one server to another.

You can access ABAP Workbench tools using either a transaction code or using SAP menu.

Table of Contents

| Top of page |

| Learning objective |

| 1. Development tools |

| 2. Navigation tools |

| 3. Debugging tools |

| 4. Organizing tools |

| 5. Accessing Workbench tools |

| Summary |

The ABAP Data Dictionary

Learning objective

After completing this topic, you should be able to recognize the characteristics of the ABAP data dictionary objects.

1. Overview of ABAP Dictionary

The ABAP Dictionary, one of the Workbench development tools, enables you to define data objects, including tables, views, and structures. In fact, the ABAP Dictionary is a repository that contains definitions of all the data objects that you create in SAP R/3.

Before you create an object in SAP R/3, you need to define the structure of the object. For example, before you create a table, you need to define the table by describing its structure in the ABAP Dictionary. The table structure consists of components such as table fields, foreign keys, and indexes.

All runtime objects, for example programs, collect information from the ABAP Dictionary to execute the program.

When you modify the definition of an object in the ABAP Dictionary, the modified definition is automatically applied to the associated object in SAP R/3. This ensures data integrity.

Suppose you have defined the structure of the ZCustomer table. The table has CName as one of its fields. The length of the CName field is set to 15 characters, and the field may exist in several other tables in the SAP database.

You can change the length of the CName field from 15 to 25 characters in the ABAP Dictionary without making changes in all the tables in the SAP database that refer to this field. The new setting is automatically applied to the CName field in all the tables.

ABAP Dictionary ensures data consistency. Suppose the ZCustomer table includes a telephone field. This field may be present in several other tables in the SAP database. You want this field to accept only numbers.

Using the ABAP Dictionary to define the telephone field, you can ensure that the format for this field is consistent in all the tables that include it.

Question

For what do you use the ABAP Dictionary?

Options:

1. Creating new programs
2. Defining the structure of an object
3. Ensuring data consistency

4. Ensuring data integrity
5. Integrating SAP and non-SAP applications

Answer

ABAP Dictionary enables you to define the structure of an object. In addition, it ensures data consistency and integrity.

Option 1 is incorrect. You cannot create programs using the ABAP Dictionary. However, all the runtime objects, for example programs, retrieve information about the required objects specified in the program from the ABAP Dictionary.

Option 2 is correct. Before you create an object in SAP R/3, you need to define its structure. For example, before you create a table, you need to define the table structure in the ABAP Dictionary. Any changes made to the object are replicated to the associated object in SAP R/3.

Option 3 is correct. The ABAP Dictionary ensures data consistency. Suppose you have defined that a particular table field should accept only numbers in the ABAP Dictionary. Then irrespective of tables where the field is defined, the field will accept only numbers.

Option 4 is correct. The ABAP Dictionary ensures data integrity. Suppose you change the length of a particular table field in the ABAP Dictionary. The new setting will be applied to the field in all the tables in SAP databases.

Option 5 is incorrect. By upgrading the SAP R/3 system to the NetWeaver platform, you can integrate SAP R/3 applications with non-SAP applications. After integrating the applications with the NetWeaver platform, you can use ABAP to manage the ABAP Dictionary.

2. ABAP Dictionary objects

The basic ABAP Dictionary objects are tables, data elements, and domains.

In SAP R/3, data is stored in tables. A table is a multidimensional object and consists of rows and columns.

Each table has a unique name that helps to identify the content of the table. SAP recommends that you prefix the letter Z to a table name to differentiate the user-defined tables from the tables provided by SAP R/3.

A column in a table represents a field and a row represents a record within a table. Each field in a table has these characteristics:

- technical characteristics – including data type, length, and decimal values
- short description – describing the purpose of the field
- field label – specifying the label for the field
- documentation – specifying the help text for the field

You can either provide the characteristics of a table field directly in the table definition, or you can define a data element and assign it to the field. A data element defines the characteristics of a field.

Using a data element ensures reusability. For example, the Qualification field exists in several tables in SAP databases. Defining the characteristics of the Qualification field in each table is repetitive. Using data elements, you can define the characteristics of the Qualification field, and assign the data element to the Qualification field. This ensures reusability.

As well as tables, domains are another main ABAP Dictionary object. A domain is used to define a value range. A value range is a set of valid values that can be specified for a field.

The characteristics of a domain include data type, length, and decimal length. These characteristics correspond to the technical characteristics of a data element. Using a domain, you can restrict the values entered in a table field.

You can create domains independently in the ABAP Dictionary. It is not necessary to define the table structure or a data element before creating a domain. Based on requirements, you can create domains and assign them to data elements.

There are six objects that are created in the ABAP Dictionary.

- Structure
- View
- Lock object
- Type group
- Online help
- Search help

Structure

A structure is a group of related components with a common name. For example, an address structure can consist of street name, city, region, country, and postal code.

The naming convention of a structure is similar to a table. However, you cannot have a common name for a structure and a table. You can include a maximum of 20 structures in a table.

View

A view is a virtual table that provides access to a set of fields or columns from one or more tables. Using views, you can ensure data security by hiding any table fields that are not required. In other words, views can hide certain confidential fields.

Lock object

A lock object locks database objects to ensure data security. You need to specify the database object name to be locked and begin the lock object name with the character E.

You can lock an object so that the data can be viewed and modified by only a single user (exclusive), the data can be just viewed (shared), or one user can modify data while others can just view the information (exclusive but not cumulative).

Type group

SAP R/3 provides you with several built-in data types such as CHAR, INT, and NUMC. In addition, you can create user-defined data types.

To create a user-defined data type, you need to define the structure of a type group in the ABAP Dictionary.

Online help

An online help facility enables you to define the documentation for the table fields. This documentation explains the meaning of the table field. You create and assign the documentation to a data element and associate the data element to a table field.

The online help is reusable. You can assign the same documentation to similar fields in other tables.

Search help

A search help object displays a list of input values for a field. When you press **F4**, the search help object for a field appears. You define the structure of the search help in the ABAP Dictionary and assign it to a data element. The data element is then assigned to a field.

Say you have created a search object, which contains a list of countries. You can assign the search object to a data element, and then assign the data element to the country field within a table. When you press **F4** for the country field, a list of input values – a list of countries – appears.

Question

Match the characteristics of each ABAP Dictionary object to its description.

Options:

1. Data element
2. Domain
3. Search help
4. Type group

Targets:

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ontains the characteristics of a field

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
reates user-defined data types

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
efines a value range

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
isplays a list of input values for a field

Answer

Data element contains the characteristics of a field. Type group creates user-defined data types. Domain defines a value range. Search help displays a list of input values for a field.

The characteristics of a field are technical characteristics, short description, field labels, and documentation. You define a data element using these characteristics and assign it to a table field.

A domain is used to define a value range. The characteristics of a domain include data type, length, and decimal length. Using domains, you can restrict the values that can be entered in a table field.

Search help is used for displaying a list of input values for a field. The user can select the desired input value from the list.

SAP R/3 provides you with several data types such as CHAR, INT, and NUMC. However, you can create user-defined data types. To create a user-defined data type, you need to define the structure of a type group in the ABAP Dictionary.

Summary

ABAP Dictionary is seamlessly integrated with ABAP Workbench and contains definitions for all the objects created in SAP R/3. All runtime programs refer to ABAP Dictionary to collect information to execute the program. The ABAP Dictionary ensures data consistency and data integrity.

The ABAP Dictionary objects include tables, views, data elements, structures, table types, lock objects, a search help facility, and online help.

Table of Contents

| **Top of page** |

| **Learning objective** |

| **1. Overview of ABAP Dictionary** |

| **2. ABAP Dictionary objects** |

| **Summary** |

Using ABAP, Workbench Tools, and ABAP Dictionary

Learning objective

After completing this topic, you should be able to recognize the features of ABAP and the functions of the ABAP tools and data dictionary objects.

Exercise overview

In this exercise, you're required to recognize the features of ABAP, functions of the ABAP tools, and ABAP Dictionary objects.

This involves the following tasks:

- using ABAP
- using ABAP Workbench tools
- using ABAP Dictionary

Task 1: Using ABAP

Say you are working as an ABAP programmer for Award Sportswear, which specializes in casual and formal sportswear. To find the amount of money that the company owes to its vendors, the finance clerk generates the Vendor Balance report.

In addition to the information displayed in the Vendor Balance report, the finance clerk wants to display information about the products purchased from each vendor.

The company also wants to generate a billing report for each vendor. SAP doesn't provide any report to meet this requirement.

Award Sportswear also wants to streamline its after-sales service. For this, the company has decided to integrate the vendors' non-SAP procurement applications with the company's SAP R/3 system.

Step 1 of 3

Which tasks can you perform using ABAP to meet the business requirements of Award Sportswear?

Options:

1. Create a program to generate the vendor billing report
2. Customize existing programs to generate the Vendor Balance report
3. Integrate Award Sportswear's SAP system with the vendors' non-SAP procurement applications

Result

Using ABAP, you can create a program to generate the vendor billing report and customize the existing programs to generate the Vendor Balance report.

Option 1 is correct. Using ABAP, you can create a program from scratch to generate the vendor billing report because SAP doesn't provide any report to meet the business requirement.

Option 2 is correct. Using ABAP, you can customize existing SAP modules. To view the total amount of money that the company owes to all its vendors and to view the information related to the products purchased from each vendor in a common report, you can customize the existing Vendor Balance report using ABAP.

Option 3 is incorrect. To integrate the company's R/3 applications with the vendors' non-SAP procurement applications, you need to upgrade the SAP R/3 system with the NetWeaver platform. After integrating the applications with the NetWeave platform, you can use ABAP to perform business transactions.

All reports in SAP R/3 are made up of programs that enable you to create, change, or display information stored in SAP databases.

Step 2 of 3

You have decided to create a program to generate the vendor billing report for each vendor. You only want to view the vendor information displayed on the output screen. You do not want to interact with or manipulate the vendor information displayed on the output screen.

Which ABAP program would you use to only view the output of the vendor billing report?

Options:

1. Dialog program
2. Traditional report program
3. Interactive report program

Result

You would use the traditional report program to only view the output of the vendor billing report.

Option 1 is incorrect. You use the dialog program when you want to manipulate the vendor information displayed on the output screen.

Option 2 is correct. You use the traditional report program to only view the output of the Vendor Balance report. This program displays information based on the input parameters you provide. You cannot interact with the vendor information displayed on the output screen.

Option 3 is incorrect. You use an interactive report program when you want to interact with the vendor information displayed on the output screen. You can sort the information, print selective information, and control the number of fields on the output screen.

Step 3 of 3

To create a traditional program for the vendor billing report, you need to access the ABAP Workbench. The SAP Easy Access screen is opened for you.

What are the steps required to access the ABAP Workbench?

Options:

1. Double-click **Administration** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**
2. Double-click **Customizing** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**
3. Double-click **Tools** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**

Result

You double-click **Tools** - **ABAP Workbench** - **Overview** - **SE80 - Object Navigator**.

Task 2: Using ABAP Workbench tools

To work effectively with ABAP Workbench, SAP has broadly categorized the ABAP Workbench tools.

Step 1 of 3

Match each ABAP Workbench tool with the appropriate category.

Options:

1. Data Browser
2. Menu Painter
3. Run Time Analysis
4. Screen Painter
5. System Log
6. Transport System

Targets:

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ebugging tools

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
evelopment tools

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
avigation tools

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
rganizing tools

Result

Run Time Analysis and System Log map to Debugging tools. Menu Painter and Screen Painter map to Development tools. Data Browser maps to Navigation tools. Transport System maps to Organizing tools.

Step 2 of 3

You have created a program to generate the vendor billing report. The vendor information stored in the SAP databases is enormous. The report takes around two minutes to show output related to vendors. As a result, you want to improve the performance of the program.

Which tool would you use to achieve this?

Options:

1. On Line Debugger
2. Run Time Analysis
3. SQL Trace
4. System Log

Result

You use the Run Time Analysis tool to improve the performance of the program created for the billing report.

Option 1 is incorrect. You use the On Line Debugger tool when the desired output is incorrect. If the vendor billing report fails to show the desired output, you would use this tool to debug the program.

Option 2 is correct. When there is a delay in displaying the desired output using the vendor billing report, you use the Run Time Analysis tool. Using this tool, you can analyze the performance of the program that is developed for the vendor billing report.

Option 3 is incorrect. You use the SQL Trace tool to trace all attempts that the program makes to access vendor information from the SAP databases.

Option 4 is incorrect. You use the System Log tool to obtain detailed information about all events that occur in SAP R/3. This tool displays the errors, warnings, and information associated with the events.

Step 3 of 3

You use the Run Time Analysis tool to improve the performance of the program created for the billing report. The SAP Easy Access screen is opened for you.

What are the steps required to access the Run Time Analysis tool using a transaction code?

Options:

1. Type `SE09` and click **Enter**
2. Type `SE16` and click **Enter**
3. Type `SE30` and click **Enter**
4. Type `SE83` and click **Enter**

Result

You type `SE30` and click **Enter**.

Task 3: Using ABAP Dictionary

The vendor billing report extracts the data from the ZVendor table. Before you store the data in the ZVendor table, you need to define the table structure.

You want to ensure that the values entered in the ZVendor table are consistent in SAP R/3. In addition, you want to ensure that when you modify the structure of the ZVendor table, the new settings should automatically be applied to the ZVendor table in the SAP databases.

Step 1 of 2

Which tasks can you perform using the ABAP Dictionary to meet the reporting requirements of Award Sportswear?

Options:

1. Create programs to generate the vendor billing report
2. Define the structure of the ZVendor table
3. Ensure consistency of data stored in the ZVendor table
4. Ensure integrity of data stored in the ZVendor table
5. Integrate Award Sportswear's SAP system with the vendors' non-SAP procurement applications

Result

ABAP Dictionary enables you to define the structure of the ZVendor table. In addition, it ensures consistency and integrity of data stored in the ZVendor table.

Option 1 is Incorrect. You can't create programs using ABAP Dictionary. However, all the runtime objects, for example programs, retrieve information about the required objects specified in the program from the ABAP Dictionary.

Option 2 is correct. Before you create an object in SAP R/3, you need to define the structure of the object. For example, before you create the ZVendor table, you need to define the structure of the ZVendor table in ABAP Dictionary.

Option 3 is correct. ABAP Dictionary ensures data consistency. Suppose you have defined in the ABAP Dictionary that a particular field in the ZVendor table should accept only numbers. Then regardless of tables in which the field is defined, the field will accept only numbers.

Option 4 is correct. ABAP Dictionary ensures data integrity. Suppose you change the length of a particular field in the ZVendor table in the ABAP Dictionary. The new settings will be applied to the field in all the tables in the SAP databases.

Option 5 is incorrect. ABAP Dictionary doesn't integrate SAP and non-SAP applications. To integrate the company's SAP R/3 applications with the vendors' non-SAP procurement applications, you need to upgrade the SAP R/3 system to the NetWeaver platform. After integrating the applications with the NetWeaver platform, you can use ABAP to manage ABAP Dictionary.

Step 2 of 2

Match the characteristic of each ABAP Dictionary object with its description.

Options:

1. Data element
2. Domain
3. Search help
4. Type group

Targets:

ontains the characteristics of a field

reates user-defined data types

efines a value range that includes data type and length

isplays a list of input values for a field

Result

Data element contains the characteristics of a field. Type group creates user-defined data types. Domain defines a value range that includes data type and length. Search help displays a list of input values for a field.

The characteristics of a field are technical characteristics, short description, field labels, and documentation. You define a data element using these characteristics and assign it to a table field.

A domain is used to define a value range. The characteristics of a domain include data type, length, and decimal length. Using a domain, you can restrict the values entered in a table field.

Search help is used to display a list of input values for a field. The user can select the desired input value from the list.

SAP R/3 provides you with several data types such as CHAR, INT, and NUMC. However, you can also create user-defined data types. To create a user-defined data type, you need to define the structure of a type group in the ABAP Dictionary.

Table of Contents

| **Top of page** |

| **Learning objective** |

| **Exercise overview** |

| **Task 1: Using ABAP** |

| **Task 2: Using ABAP Workbench tools** |

| **Task 3: Using ABAP Dictionary** |

Introducing Tables in ABAP

Learning objective

After completing this topic, you should be able to recognize the components of a table definition.

1. Table types

A table in SAP R/3 is an array of data arranged into rows and columns.

The rows in a table display data records that are separated into records of a similar nature using columns. A typical table has the same number of columns in each row.

In SAP R/3, each table has a unique name.

Tables in SAP R/3 hold persistent data. This means that the data entered in a table by a program remains there after the program closes.

This data remains in the table until it is deleted or changed by the same program or another program.

The ABAP Dictionary helps you to create database tables. You can create three types of tables using the ABAP Dictionary.

- Transparent table
- Pooled table
- Cluster table

Transparent table

A transparent table is a single table that is first defined in the ABAP Dictionary and then physically created in the database.

After you create a transparent table, its logical definition is available in the ABAP Dictionary and the physical table is available in the database.

Transparent tables are used to store application data in SAP R/3. Application data includes master data and transactional data.

Application data is used most frequently in the day-to-day functioning of SAP R/3. As a result, you will need to define transparent tables frequently using the ABAP Dictionary.

Pooled table

A pooled table is a table created in the ABAP Dictionary for being stored in a table pool.

A table pool, in turn, is a physical table in the database that stores data coming from a large number of pooled tables.

Once created, a pooled table is available in the ABAP Dictionary; however, in the database, it appears as a part of a table pool.

Normally, small tables with 10 to 100 rows are defined as pooled tables, and then these are placed in a table pool.

The table pool helps you make efficient use of database resources and lets you access data from a number of pooled tables simultaneously.

Here is a simple example. Suppose *Vendor* and *Account* are two pooled tables.

The *Vendor* table has two key fields, *Name* and *Code*, and a data field, *Address*. The *Account* table has one key field, *Account Number,* and a data field, *Bank*.

The *Payment* table pool brings the records of the *Vendor* and *Account* pooled tables together.

In the *Payment* table pool, the first field shows the table names, the second field shows the records in the key fields of the pooled tables, and the third field shows the records in the data fields of the pooled tables in a continuous string.

For instance, the first row of the *Payment* table pool consists of three fields. The first field in this row shows a pooled table name, *Vendor*. The second field shows records from the *Name* and *Code* fields of the *Vendor* table. The third field shows records from the *Address* field of the *Vendor* table in a continuous string.

Cluster table

A cluster table is created in the ABAP Dictionary and is stored in a table cluster.

A table cluster, in turn, is a physical table in the database that stores data from a few cluster tables.

Normally, large tables with common key fields are defined as cluster tables, and then these are put in a table cluster.

The table cluster helps you simultaneously access data from tables with common key fields.

Here is a simple example. Suppose *Hotels* and *Rooms* are two cluster tables.

The *Hotels* cluster table has two fields, *Hotel code* and *Hotel name*. *Hotel code* is the key field in this table. Each row in this table shows a hotel code and its corresponding hotel name. For example, the first row in this table shows hotel code as 01 and hotel name as Skyways.

The *Rooms* cluster table has three fields: *Hotel code*, *Room type*, and *Room rent*. *Hotel code* is the key field in this table. Each row in this table shows a hotel code and its corresponding room type and room rent. For example, the first row in this table shows hotel code as 01, room type as single, and room rent as 100.

Because the *Hotels* and *Rooms* cluster tables have a common key field, *Hotel code*, these are put together in a table cluster, *Hotel rooms*.

The *Hotel rooms* table cluster joins the corresponding data rows from the *Hotels* and *Rooms*

cluster tables to form one row. For example, the first row in the table cluster shows hotel code as 01, hotel name as Skyways, room type as single, and room rent as 100.

Out of the ABAP Dictionary table types, the pooled and cluster table types are SAP proprietary formats. Because of this, you cannot use third-party tools to access data from these tables. Accessing data from these tables requires you to use SAP interfaces, such as BAPI.

These two table types also have some ABAP programming restrictions.

These reasons make the day-to-day uses of the pooled and cluster tables limited.

Unlike transparent tables, pooled tables and cluster tables are not used to store application data in SAP R/3.

Pooled tables are generally used to store control data such as screen sequences or temporary data.

Cluster tables are generally used to store continuous text such as documentation.

Question

Which of these statements about tables in SAP R/3 are correct?

Options:

1. Activating a table definition creates the table in the ABAP Dictionary
2. Data entered in a table by a program remains there after the program closes
3. Table definitions can be created independently of the database
4. Tables in SAP R/3 can have common names

Answer

In SAP R/3, data entered in a table by a program remains in the table after the program closes. And you can create table definitions independently of the database.

Option 1 is incorrect. A table definition is created first in the ABAP Dictionary, and then this definition is activated to create the physical table in the database.

Option 2 is correct. Tables in SAP R/3 hold persistent data. This means that the data entered in a table by a program remains there after the program closes.

Option 3 is correct. You can create table definitions in the ABAP Dictionary. These definitions are created in the ABAP Dictionary independently of the database.

Option 4 is incorrect. Tables in SAP R/3 need to have unique names.

Question

Match each table type available in the ABAP Dictionary to the situation where you can use it.

Options:

1. Cluster table
2. Pooled table
3. Transparent table

Targets:

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ou need to bring together data from a few large tables with common key fields

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ou need to bring together data from a large number of small tables

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ou need to store application data

Answer

You can use the cluster table type to bring together data from a few large tables with common key fields. You can use the pooled table type to bring together data from a large number of small tables. You can use the transparent table type to store application data.

Cluster tables are the tables assigned to a table cluster. A table cluster brings together data from a few large tables with common key fields. Cluster tables store continuous text.

Pooled tables are the tables assigned to a table pool. A table pool brings together data from a large number of small tables. Pooled tables store control data.

Transparent tables store application data in SAP R/3. Application data includes master data and transactional data.

2. Table definition

To create a table in an SAP R/3 database, you need to first create its table definition – also called the table object – in the ABAP Dictionary.

A table definition represents the structure of a database table. This acts as a logical depiction of the database table and is not the physical database table. This definition is created in the ABAP Dictionary independently of the database.

Because transparent tables are the most commonly used tables in SAP R/3, the table definition for transparent tables is discussed here.

The table definition for a transparent table in the ABAP Dictionary has these components.

- Table fields

- Foreign keys
- Technical settings
- Indexes

Table fields

Table fields are used to define the names, data types, and other characteristics of the fields in a table. Table fields are represented as columns in a table.

To define a table field in the ABAP Dictionary, you need to specify these characteristics for the field:

The field name can have a maximum of 16 characters. You can use letters, digits, and underscores in a field name.

The data type refers to the data format used for the field. You can use one of the data types available in the ABAP Dictionary for a table field; for example, character string, date, and floating-point number.

The field length defines the number of characters that the field can hold.

The short text provides a brief description of the table field.

The decimal length defines the number of characters after a decimal point that the field can hold.

You can directly enter all the characteristics of a table field when creating a table definition.

Alternatively, you can enter the field name and then assign an existing data element to a table field. In this case, the data type, field length, and description of the data element are assigned to the table field.

Apart from creating individual table fields, you can also include the fields of a structure in a table.

Foreign keys

Foreign keys are used to link the fields in a table to the primary key fields in another table. The field in a table that you link to a primary key field in another table is called the *foreign key field*.

This link maintains data consistency between the two tables. This means that the data entered in the foreign key field can be checked against the primary key field of another table.

A primary key field has a unique value in each table row.

The table that contains a foreign key field is called the *foreign key table*. The table whose primary key field is linked to the foreign key field is called the *check table*.

In this example, the *Airline* table contains two fields, *Flight* and *Airport*. You want to ensure that a user can only enter a valid airport code in the *Airport* field.

To do this, you create another table called *Codes* that has a primary key field called *Airport code*, which lists valid airport codes.

Now you make the *Airport* field in the *Airline* table a foreign key field and link it to the *Airport code* field in the *Codes* table.

The *Airline* table now becomes the foreign key table and the *Codes* table becomes the check table.

In this case, when a user enters a value in the *Airport* field of the *Airline* table, this value is checked against the valid airport codes in the *Airport code* field of the *Codes* table.

This ensures that the user can only enter a valid airport code in the *Airport* field of the *Airline* table, which is a foreign key field.

Technical settings

Technical settings optimize the storage requirements and accessing behavior of database tables.

The technical settings for a table include

You can choose a data class to select a specific database area to create the table. Data classes include master data, transactional data, and organizational data. Each data class has a specific physical area marked for it in the database.

You can choose a size category to define the space in the database that the table is expected to occupy. Each size category is assigned a specific memory size in the database.

You can select a buffering permission to allow or deny buffering for a table.

You need to select a buffering type for a table if you have allowed buffering for it. A buffering type defines how many table records are loaded into the buffer when a table entry is accessed.

You can enable logging to record each change in table entries. These changes are listed in a log table.

Indexes

An index extracts specific fields of a table and sorts the data records in them. This makes the search for data records quicker because the index directs the search to specific data records that are sorted.

When you create a table, the *primary index* for the table is created automatically. The primary index takes the data records from the primary key fields of the table and sorts them. It also contains pointers to the row numbers of the data records in the actual table.

In this example, the primary index for the *Employee info* table alphabetically sorts the employee roles that are taken from the primary key field of the table. It also provides pointers to the row numbers of each role in the *Employee info* table.

Description of the primary index example follows:

Employee info is a database table with two fields, *Role* and *Location*. The *Role* field lists employee roles and the *Location* field lists employee locations. The *Role* field is the primary key field in the table.

The primary index for this table sorts the employee roles, taken from the *Role* field, in alphabetical order. In addition, pointers corresponding to the row numbers of the employee roles in the actual *Employee info* table are listed.

Description ends.

The primary index has its limitations. It may not meet all search requirements for a table.

For example, if you search a specific employee role, the primary index speeds up the search as the roles are sorted. However, if you search a specific employee location, the primary index does not speed up the search because the locations are not sorted in the index.

You can create *secondary indexes* to address the search requirements for a table that are not met by the primary index.

In this example, you can create a secondary index for the *Location* field to speed up searches based on employee location.

This secondary index for the *Location* field sorts employee locations in alphabetical order and provides pointers to the row numbers of the locations in the *Employee info* table.

After you have created a table definition in the ABAP Dictionary, which contains table fields, foreign keys, technical settings, and indexes, you need to activate the definition.

The physical table in the database is created when you activate the table definition in the ABAP Dictionary.

Question

Match each component of a table definition to its use.

Options:

1. Foreign keys
2. Indexes
3. Table fields
4. Technical settings

Targets:

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
efine the data types and length of the fields in a table
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
xtract specific fields of a table and sort the data records in them

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
ink the fields in a table to the key fields in another table

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
ptimize the storage requirements of a table

Answer

You can use table fields to define the data types and lengths of the fields in a table. You can extract specific fields of a table and sort the data records in them using indexes. You can use foreign keys to link the fields in a table to the key fields in another table. You can optimize the storage requirements of a table using technical settings.

Foreign keys link the fields in a table to the primary key fields in another table. This link maintains data consistency between the two tables.

Indexes extract specific fields of a table and sort the data records in them. This makes the search for data records quicker.

Table fields define the names, data types, lengths, and descriptions of the fields in a table.

Technical settings optimize the storage requirements and accessing behavior of database tables.

Summary

A table in an SAP R/3 database is an array of data arranged into rows and columns. You can define transparent tables, pooled tables, and cluster tables using the ABAP Dictionary. A transparent table is a single table that is the same in the database as it appears in its table definition. These are used most frequently because they store application data. Pooled tables are a part of a table pool. Table pools bring together data from a large number of small tables. Cluster tables are a part of a table cluster. A table cluster brings together data from a few large tables with common key fields.

To create a transparent table, you first need to create its table definition – independently of the database – in the ABAP Dictionary. A table definition consists of table fields, foreign keys, technical settings, and indexes. The physical table in the database is created when you activate the table definition.

Table of Contents

| Top of page |

| Learning objective |

| 1. Table types |

| 2. Table definition |

| **Summary** |

Creating Domains and Data Elements in ABAP

Learning objective

After completing this topic, you should be able to create a domain and a data element in a given scenario.

1. Overview of domains and data elements

Database tables in SAP R/3 consist of fields. Each field in a table has a name, a data type, a field length, a decimal length, and a short text.

You can either provide these characteristics directly in the table definition or assign the field a data element.

A data element is a data type in the ABAP Dictionary used to store the characteristics of a table field. The important characteristics that a data element includes are

- technical characteristics
- short description
- field labels
- documentation

technical characteristics

The technical characteristics of a data element include

- data type
- length
- decimal length

short description

Short description is the text that describes the purpose of the data element.

field labels

Field labels define column headings for the fields that use the data element.

documentation

Documentation for the data element is displayed as help text for the fields using the data element.

You can define the technical characteristics of a data element directly in the ABAP Dictionary or you can assign a domain to the data element.

A domain is an object that has these characteristics:

- data type
- length
- decimal length

These characteristics correspond to the technical characteristics of a data element.

When you create a domain, you obtain a data object that consists of a data type, a length, and a decimal length.

If you create a data element using a domain, the data type, length, and decimal length of the domain is assigned to the data element. You can add the short description and, if required, the field labels and documentation to complete the data element.

When you assign a data element to a table field, the data type, the length, and the decimal length are pulled from the domain to which the data element belongs and assigned to the table field. In addition, the short description of the data element is assigned to the table field as its short text.

Domains and data elements save you the time and effort spent in creating table fields because domains act as templates for data elements and data elements act as templates for table fields.

Note

Domains can be created independently in the ABAP Dictionary and then assigned to various data elements. Similarly, data elements can be defined independently and then assigned to various table fields that need similar characteristics.

Here is an example to describe the relationships among domains, data elements, and table fields. Suppose you need to create a table called *zemployee* that lists the first, middle, and last names of employees in a company.

To define the fields in this table, you can create a domain called *zname*. This domain has the "char" data type and its length is 30.

Next you can define three data elements called *zfirst*, *zmiddle*, and *zlast*. You can assign the *zname* domain to these data elements to define their technical characteristics. In addition, you need to add different short descriptions, field labels, and documentation for each data element.

Finally, you define three fields in the *zemployee* table: First, Middle, and Last. You assign the data elements *zfirst*, *zmiddle*, and *zlast* to the first, middle, and last fields, respectively.

The *zemployee* table now has three fields with different uses, field labels, and documentation but with the same data type, "char," and length, 30.

Question

Match the given features with the appropriate ABAP Dictionary objects.

Options:

1. Can be assigned to various data elements
2. Can be assigned to various table fields
3. Defines a value range that includes data type and length
4. Includes labels and short description for a field

Targets:

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
ata element

zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
omain

Answer

A data element includes labels and a short description of a field, and it is assignable to various table fields. A domain defines a value range that includes data type and length, and it is assignable to various data elements.

2. Creating a domain

A domain can be assigned to a data element, which in turn can be assigned to a table field.

You can create a domain using the ABAP Dictionary.

However, before creating a domain, you must check whether any domains that provide the characteristics you need are available in SAP R/3.

If an existing domain meets your needs, you should use it instead of creating a new one.

Suppose you want to create a domain called *zguide* that defines the value range for storing the names of tour guides working for Easy Nomad.

This domain should allow fields to accept up to 25 characters.

The SAP Easy Access screen is opened for you. You need to access the ABAP Dictionary: Initial Screen to create a domain.

To access the ABAP Dictionary: Initial Screen, you perform these steps:

- type `SE11` in the Command field on the standard toolbar
- click the **Enter** button

The ABAP Dictionary: Initial Screen appears. This screen enables you to create various ABAP Dictionary objects, including domains.

After reaching the ABAP Dictionary: Initial Screen you can begin creating the *zguide* domain.

You select the **Domain** option, type `zguide` in the Domain field, and then click **Create**. Alternatively, you can press **F5** after selecting the **Domain** option and typing `zguide` in the Domain field.

The Dictionary: Maintain Domain screen appears. This screen lets you define a domain.

Here are the two tabs on the Dictionary: Maintain Domain screen that you can use to specify the characteristics of a domain.

- Definition
- Value range

Definition

The **Definition** tab lets you define the data type, length, and the decimal length of the domain.

Value range

You can use the **Value range** tab if you need to use fixed values or value tables to restrict the range of values that can be entered in a field that refers to the domain.

On the Dictionary: Maintain Domain screen, you first need to specify a short description of the domain.

In this example, you type `Tour guides` in the Short Description field.

After providing the short description, you need to provide the data type and length of the domain. In this example, the domain should have the "char" data type.

You type `char` in the Data type field and press **Enter** to do this.

The data type for the domain is set to "char". This data type allows fields to accept a string of characters.

After specifying the data type, you need to provide 25 as the length of this domain.

You type `25` in the No. characters field and press **Enter** to do this.

The length of the domain is set to 25. This means that the domain will allow fields to accept 25 characters.

Some data types may require a user to enter numbers after a decimal in a field. For example, if you use the "quan" data type, the user can enter numeric quantities in the fields using this domain.

In such cases, you use the Decimal places field to specify the number of decimal places that a field that refers to this domain can accept.

In this example, you leave the Decimal places field blank.

The required characteristics of the *zguide* domain have been defined.

Now you can save this domain. To save the domain, you click **Domain** - **Save**.

The Create Object Directory Entry dialog box is displayed. This dialog box asks you to assign a package to the ABAP Dictionary object you created.

Packages are used to

- group together objects of the same type
- transport objects from one SAP system to another

In this example, you need to assign a temporary package to the domain. So you type `$TMP` in the Package field.

After assigning a package, you need to click the **Save** button in the Create Object Directory Entry dialog box to finish saving the domain.

The domain is saved and you return to the Dictionary: Maintain Domain screen. Now you should activate the domain to make it available in the runtime environment.

You click the **Activate** button on the application toolbar to activate the domain. Alternatively, you can press **Ctrl+F3** to activate the domain.

The *zguide* domain is now activated and available to be assigned to data elements.

A domain can also be used to restrict the values that can be entered in a field with the help of

- fixed values
- value tables

Fixed values for a domain help you define a fixed set of values that can be entered in a field referring to the domain.

These can be either single values or a range of values.

For example, a domain *zroom* defines the room types available in a motel. Its data type is "char" and length is 1. The fixed values for the domain are "S" (for single room) and "D" (for double room). If a table field refers to this domain, a user will be able to enter only "S" or "D" in the field.

Suppose you are creating the *zroom* domain. To define the fixed values for this domain, you open the Value range tabbed page by clicking the **Value range** tab on the Dictionary: Maintain Domain screen.

Then you type the fixed values `S` and `D` in the first two rows of the Fix.val. column. You also provide the short descriptions for these fixed values in the respective rows in the Short text column.

A domain also enables you to check the values entered in a field against a table. You can do this by assigning a value table to the domain.

For example, domain *zport* is referred by table fields where a user can enter airport codes. You want the user to enter only valid airport codes in such fields.

Suppose *zcode* is a table that lists valid airport codes. If you assign the *zcode* table as a value table to the *zport* domain, any values entered in the fields referring to the *zport* domain would be checked against the *zcode* table.

Note

To activate the checking of a field's values against a value table, you need to define this field as a foreign key referring to the value table.

Suppose you are creating the *zport* domain. You have opened the Value range tabbed page on the Dictionary: Maintain Domain screen.

You type `zcode` in the Value table field and press **Enter** to assign the *zcode* table as a value table to this domain.

The *zcode* table will now be the value table for the *zport* domain.

Note

After assigning fixed values or a value table to a domain, you need to save and activate the domain.

Question

You are creating a domain called *zmaterials*. The domain needs to have a short description, Material types. The domain should allow fields to accept up to 15 characters. The domain should check values entered in fields against the *ztypes* table. You need to assign the `$TMP` package to the domain.

You have reached the Dictionary: Maintain Domain screen.

What are the appropriate steps to finish creating this domain?

Options:

1. Type `Material types` in the Short Description field, type `char` in the Data type field, and type `15` in the No. characters field. Then click the **Value range** tab. On the Value range tabbed page, type `ztypes` in the Value table field and then press **Enter**. Click **Domain** - **Save**. In the Create Object Directory Entry dialog box, type `$TMP` in the Package field, and then click the **Save** button. Click the **Activate** button.
2. Type `Material types` in the Short Description field, type `char` in the Data type field, and type `15` in the No. characters field. Then click the **Value range** tab. On the Value range tabbed page, type `ztypes` in the first row in the Fix.val. column. Click **Domain** - **Save**. Click the **Activate** button.

Answer

You type `Material types` in the Short Description field, `char` in the Data type field, `15` in the No. characters field, and click the **Value range** tab. On the Value range tabbed page, you type `ztypes` in the Value table field and press **Enter**. Then you click **Domain** - **Save**. In the Create Object Directory Entry dialog box, you type `$TMP` in the Package field and click **Save**. Next you click **Activate**.

3. Creating a data element

You can now use the *zguide* domain to create a data element called *zguide_first*.

This data element will store the characteristics of table fields where you can enter the first names of the tour guides working for Easy Nomad Limited.

You can begin creating the *zguide_first* data element from the ABAP Dictionary: Initial Screen.

You select the **Data type** option, type `zguide_first` in the Data type field, and then click **Create**. Alternatively, you can press **F5** after selecting the **Data type** option and typing `zguide_first` in the Data type field.

The Create Type ZGUIDE_FIRST dialog box appears. It enables you to select the data type you want to create.

Note

The user-defined data types in the ABAP Dictionary include data elements, structures, and table types.

The **Data element** option is selected by default in the Create Type ZGUIDE_FIRST dialog box. So, in this case, you only need to confirm the selection.

You click the **Continue** button in the Create Type ZGUIDE_FIRST dialog box to confirm that you are creating a data element. Alternatively, you can press **Enter**.

The Dictionary: Maintain Data Element screen appears. This screen lets you define a data element.

The **Data Type** tab on this screen is open by default.

On the Dictionary: Maintain Data Element screen, you first need to specify a short description of the data element.

In this example, you type `Tour guide's first name` in the Short Description field.

Note

When you assign this data element to a field, the data element's short description will be used as the short text of the field.

You can now assign the *zguide* domain to the data element. Notice that the **Domain** option is selected by default on the screen.

You type `zguide` in the Domain field, and then press **Enter** to assign the *zguide* domain to the data element.

The Dictionary: Maintain Data Element screen is updated. The data type and length of the *zguide* domain are now assigned to the *zguide_first* data element.

After assigning the domain, you want to define field labels for the fields that will use this domain.

To define these, you click the **Field label** tab of the Dictionary: Maintain Data Element screen.

The Field label tabbed page opens. On this tabbed page, you can specify the short, medium, long, and heading field labels and their corresponding lengths.

The short, medium, and long field labels act as the column headings of the fields that use the data element. These different sizes are required because not all tables may have the same amount of space available to show the column headings.

The heading field label acts like the column heading when you extract the contents of a field in the form of a list.

Note

> *The length of each label can be more than the actual number of characters that you provide as its field label. This extra length can be of use if the labels are translated to any other language.*

You type `First` as the short field label, and specify its maximum length as `6`.

Next you type `First name` as the medium field label, and specify its maximum length as `12`.

Then you type `Guide's first name` as the long field label, and specify its maximum length as `20`.

Then you type `First name` as the heading label, and specify its maximum length as `12`.

The required characteristics of the *zguide_first* data element have now been defined. Now you need to save it.

You click **Data element** - **Save** to save the data element. Alternatively, you can press **Ctrl+S**.

The Create Object Directory Entry dialog box is displayed.

This dialog box asks you to assign a package to the ABAP Dictionary object you have created.

In this example, you need to assign a temporary package to the data element. So you type `$TMP` in the Package field.

After assigning a package, you need to click the **Save** button in the Create Object Directory Entry dialog box to finish saving the data element.

The data element is saved and you return to the Dictionary: Maintain Data Element screen. Now you want to set the documentation status of the data element.

You click **Goto** - **Documentation** - **Status**.

The Change Documentation Status dialog box appears. You can select one of the options in this dialog box as the documentation status of the data element.

In this example, you want to add documentation to the data element. So you accept the **Object requires documentation** option, which is selected by default.

To confirm this option, you click the **Save** button.

After saving the documentation status, you return to the Dictionary: Maintain Data Element screen. Now you want to add documentation to the data element. This documentation is displayed when a user working on a field that is assigned this data element presses **F1** to seek help related to the field.

To add documentation to the data element, you click the **Documentation** button on the application toolbar.

The Change Data element: ZGUIDE_FIRST Language EN screen appears. You can type the documentation text in the blank fields on this screen.

You type `Guide's first name field` as the help document's heading and type `This field is used to enter the travel guide's first name.` as the paragraph text for the help document.

Next you click the **Activate** button on the application toolbar to activate and save the help document.

After saving the help document, you need to go back to the Dictionary: Maintain Data Element screen.

To do this, you click the **Back** button on the standard toolbar.

You are back to the Dictionary: Maintain Data Element screen. You can now activate the data element.

You click the **Activate** button on the application toolbar to activate the data element. Alternatively, you can press **Ctrl+F3**.

The *zguide_first* data element is now activated and available to be assigned to table fields.

Question

Suppose you are creating a data element called *zguide_last* using the *zguide* domain. The short description of the data element should be "Tour guide's last name."

You are currently on the ABAP Dictionary: Initial Screen.

Which sequence of steps will help you reach the Dictionary: Maintain Data Element screen, provide the short description, and assign the zguide domain to the zguide_last data element?

Options:

1. Select the **Data type** option, type `zguide_last` in the Data type field and then click **Create**. In the Create Type ZGUIDE_LAST dialog box, click **Continue**. On the Dictionary: Maintain Data Element screen, type `Tour guide's last name` in the Short Description field. Next type `zguide` in the Domain field and then press **Enter**.
2. Select the **Domain** option, type `zguide_last` in the Data type field and then click **Create**. In the Create Type ZGUIDE_LAST dialog box, click **Data element**. On the Dictionary: Maintain Data Element screen type `zguide` in the Domain field and then press **Enter**. Next type `Tour guide's last name` in the Short Description field.

Answer

You select the **Data type** option, type `zguide_last` in the Data type field, and then click **Create**. In the Create Type ZGUIDE_LAST dialog box, you click **Continue**. On the Dictionary: Maintain Data Element screen, you type `Tour guide's last name` in the Short Description field. Next you type `zguide` in the Domain field and then press **Enter**.

Summary

A data element is used to store the characteristics of a table field. A domain, in turn, is used to store the technical characteristics of a data element.

To create a domain, you need to provide it with a short description and assign it a data type, length, and decimal length. When saving the domain, you need to assign it a package. After the domain is saved, you need to activate it to make the domain object available in the runtime environment.

To create a data element using a domain, you first need to specify a short description for the data element and then assign it the required domain. Next you need to specify the field labels for the data element. When saving the data element, you need to assign it a package. You also need to specify a documentation status for the data element and, if required, you need to provide the documentation text. After saving the data element and creating documentation, you need to activate it.

Table of Contents

| Top of page |

| Learning objective |

| 1. Overview of domains and data elements |

| 2. Creating a domain |

| 3. Creating a data element |

| Summary |

Creating ABAP Tables

Learning objective

After completing this topic, you should be able to recognize how to create a table in a given scenario.

1. Creating table fields

You can use the ABAP Dictionary to create database tables in SAP R/3. To create a database table, you first create its table object - also called Table definition - in the ABAP Dictionary. This object is created independently of the database. The actual database table is created when you activate this object.

Here are the major tasks involved in creating a table object:

- creating the table fields
- assigning foreign keys to table fields, if required
- maintaining the technical settings of the table

Table fields can be created using existing data elements. Data elements store the characteristics for table fields.

Data elements, in turn, are created using domains. Domains define the technical characteristics stored in a data element.

Suppose you want to create a table called *ztourguide* that can store data related to the tour guides working for Easy Nomad.

This data includes the employee code, the first name, the last name, and the location of each tour guide.

The ABAP Dictionary: Initial Screen is open for you. You can begin creating the *ztourguide* table by providing its name and creating its database table object.

You type `ztourguide` in the Database table field, and then click **Create**. Alternatively, you can press **F5** after typing `ztourguide` in the Database table field.

The Dictionary: Maintain Table screen appears. You can define the required characteristics for the table using this screen.

On the Dictionary: Maintain Table screen, you first you need to provide a short description for the table you are creating.

In this example, you type `Easy Nomad tour guides` in the Short Description field.

After providing the short description, you need to assign a delivery class to the table. Delivery class controls the behavior of table data when it is copied, upgraded, or

transported between SAP systems. Some of the delivery classes available in SAP R/3 include

- A
- C
- L
- S

A

The delivery class "A" is assigned to application tables. These tables store master and transaction data.

C

The delivery class "C" is assigned to customizing tables. These tables store data used to customize the SAP system.

L

The delivery class "L" is assigned to tables that store temporary data.

S

The delivery class "S" is assigned to the system tables used in SAP R/3.

In this example, you want to assign the delivery class "A" to this table because this table stores transaction data. To do this, you type `A` in the Delivery Class field.

Note

You can also select an indicator that specifies whether the display and maintenance of the table should be allowed through the Data Browser and Maintain Table Views tools. This indicator can be selected using the Data Browser/Table View Maint. field.

After selecting the delivery class, you can create the fields in the table using the Fields tabbed page.

To open the Fields tabbed page, you click the **Fields** tab.

You use the Fields tabbed page to define the fields that the table contains.

You first want to add the field to store employee codes. You can use an existing data element named *zempid* to create this field.

To add this field, you first need to specify its name.

To do this, you type `Employee_code` in the first row in the Field column and press **Enter** to specify the field's name.

You have now provided a name for the table field.

Since the Employee_code field will store employee codes, which will be unique for each record in the database table, you can make this field the primary key of the table.

To make this field the primary key of the table, you click the corresponding checkbox in the **Key** column.

Note

A primary key field stores data that uniquely identifies each record in a table.

To define the data type, length, decimal length, short description, and field labels for the field, you can now assign the *zempid* data element to the Employee_code field.

You type `zempid` in the first row of the Data element column, and then press **Enter** to assign the *zempid* data element to this field.

The screen is updated. Notice that the first row in the Data type, Length, Decimal Places, and Short Description columns is now filled.

You have created the Employee_code field for the table. You can use the same procedure to add other fields to the table.

Suppose you add three more fields to the table using existing data elements. These fields are

- First_name; this field is assigned the *zguide_first* data element
- Last_name; this field is assigned the *zguide_last* data element
- Location; this field is assigned the *zlocations* data element

Question

Suppose you need to add a field called "Room_types" to the *zhotel* table. You want to create this field using the *zrooms* data element. This field needs to be the key field in the *zhotel* table.

The Fields tabbed page on the Dictionary: Maintain Table screen is already open for you.

What are the appropriate steps to add the "Room types" field to the zhotel table?

Options:

1. Type `Room_types` in the first row in the **Fields** column. Click the corresponding checkbox in the **Key** column. Type `zrooms` in the first row of the **Data element** column, and then press **Enter**.
2. Type `Room_types` in the first row in the **Data element** column. Click the corresponding checkbox in the **Key** column. Type `zrooms` in the first row of the **Data type** column, and then press **Enter**.

Answer

You type `Room_types` in the first row in the Field column. Then you click the corresponding checkbox in the Key column. Next you type `zrooms` in the first row of the Data element column, and then you press **Enter**.

2. Assigning foreign keys

Foreign keys are used to link the fields in a table to the primary key fields in another table. This link maintains data consistency between the two tables.

This means that the data entered in a foreign key field of a table can be checked against the primary key field of another table.

The specific primary key field that is linked to the foreign key field is called the *check field*. The table whose primary key field is linked to the foreign key field is called the *check table*.

Note

You can link together two fields with a foreign key only when they have the same domain.

Suppose you want to ensure that a user can only enter a valid employee code in the Employee_code field of the *ztourguide* table.

To do this, you can make the Employee_code field a foreign key field and check the values entered in this field against the primary key field, validcodes, of the *zemployeecode* table.

The validcodes key field of the *zemployeecode* table stores the valid employee codes for Easy Nomad.

You can use the Fields tabbed page on the Dictionary: Maintain Table screen to make the Employee_code field a foreign key field.

This page is already opened for you.

You first need to select the field that you want to make the foreign key field, in this case Employee_code.

After selecting the Employee_code field you need to assign the foreign key status to this field.

You click the **Foreign Keys** button (this button shows a key icon) on the Fields tabbed page to do this.

The Create Foreign Key ZTOURGUIDE-EMPLOYEE_CODE dialog box is displayed. This dialog box helps you define the foreign key relationship.

On the Create Foreign Key ZTOURGUIDE-EMPLOYEE_CODE dialog box, you first need to specify a short description for the foreign key relationship in the Short text field.

In this example, you type `Employee code validation` in the Short text field.

After specifying the short description, you need to specify the check table's name. In this example the check table is *zemployeecode*.

You type `zemployeecode` in the Check table field to specify the check table's name.

The *zemployeecode* table is now specified as the check table. The primary key of the check table will be used to check the values entered in the Employee_Code field of the *ztourguide* table.

After you have specified the check table name, you can let the SAP system propose the appropriate check field of the *zemployeecode* table that can be assigned to the Employee_code foreign key field of the *ztourguide* table.

You click the **Generate proposal** button to generate a proposal for the check field.

The Create Foreign Key ZTOURGUIDE-EMPLOYEE_CODE dialog box is updated. The system automatically suggests *zemployeecode* as the check table, validcodes as the check field, *ztourguide* as the foreign key table, and Employee_code as the foreign key field.

To generate the proposal, the system compares the foreign key field with the key fields of the check table; and proposes that key field as the check field which has the same domain as the foreign key field.

The proposal generated by the system suits your requirements, so you can accept it.

You click the **Copy** button to accept the proposal. Alternatively, you can press **Enter**.

The foreign key relationship is created, and you return to the Dictionary: Maintain Table screen.

After you have assigned the required foreign key, you can save the table.

You click **Table** - **Save** to begin saving the table.

The Create Object Directory Entry dialog box is displayed. In this dialog box, a temporary package "$TMP" is assigned by default to the table object. Because this table is a local object you have created for your use, this package is suitable.

You need to click the **Save** button on the Create Object Directory Entry dialog box to finish saving the table.

After the table is saved, you can maintain its technical settings.

Question

Suppose you want to define the Location field in the *ztourguide* table as a foreign key field. This field must be checked against the appropriate key field of the *zlocate* table. You can provide the description of this foreign key relationship as "Location

validation".

The Fields tabbed page on the Dictionary: Maintain Table screen is already open for you.

What are the appropriate steps to define this foreign key relationship?

Options:

1. Select the **Location** field. Click the **Foreign Keys** button. On the dialog box that appears, type `Location validation` in the Short text field. Next type `zlocate` in the Check table field, and then click the **Generate proposal** button. Finally, click the **Copy** button.
2. Select the Location field, and then click the **Foreign Keys** button. On the dialog box that appears, type `ztourguide` in the Short text field. Next type `zlocate` in the Check table field. Finally, click the **Save** button.

Answer

You select the **Location** field and click the **Foreign Keys** button. On the dialog box that appears, you type `Location validation` in the Short text field and `zlocate` in the Check table field, and then click the **Generate proposal** button. Then you click the **Copy** button.

3. Maintaining technical settings

Technical settings for a table help you specify the table's storage requirements and buffering behavior; and whether changes made to the table records should be logged.

These are the important technical settings for a table.

- Data class
- Size category
- Buffering
- Logging

Data class

You can select a data class to assign the table the appropriate physical area of the database where it should be created.

In the database, specific physical areas are demarcated to store different types of tables. Each of these areas corresponds to a data class.

The important data classes in the database include

The APPL0 data class is used to store transparent tables that contain master data. The master data in SAP does not change frequently.

The APPL1 data class is used to store transparent tables that contain transaction data. Data in such tables is frequently changed.

The APPL2 data class is used to store tables that contain organizational and customization data. Such tables are usually created when the SAP system is set up, and the data in these tables does not change frequently.

Size category

You can choose a size category to define the space in the database that the table is expected to occupy. Each size category is assigned a specific memory size in the database.

Selecting the correct size category for tables helps you optimize the use of database space.

You can choose a size category from 0 to 6; 0 being the smallest size and 6 being the largest.

For instance, a table with size category as 0 can hold 0 to 40,000 data records. On the other hand, a table with size category as 6 can hold 6,900,000 to 270,000,000 data records.

The exact size of the size categories may vary across different SAP installations, depending on the database system used.

Buffering

Buffering refers to loading table data in the buffer of the application server when the data is first accessed.

When you access this data the next time, the data is read directly from the application server memory. This helps you avoid accessing the database again, which saves you time and improves database performance.

The buffering of table data can be controlled by selecting a buffering permission. The available buffering permissions include

- Buffering not allowed
- Buffering allowed but switched off
- Buffering switched on

You should switch on buffering for the tables whose data is not changed very frequently. This helps you avoid accessing the database again to get the same data.

However, if the data in a table is updated quite frequently, buffering may not be beneficial. This is because the application server would need to constantly synchronize data in its buffer with the database, which would affect system performance.

If you choose to switch on buffering for a table, you need to select a buffering type. The buffering type determines which table records are loaded in the buffer of the application server. These are the available buffering types for a table:

- single-record buffering
- generic buffering
- full buffering

If you select single-record buffering, only those table records that are read by a program are loaded in the buffer.

For example, you select single-record buffering for the *zhotel* table. Then you use a program to read the first row in the table. In this case only the records from the first row of the table will be loaded in the buffer of the application server.

If you select generic buffering, only those records whose generic key is same as the generic key of the table record being read by a program are loaded in the buffer.

For example, you select single-record buffering for the *zhotel* table. Then you use a program to read the first row in the table. This row has the generic key as 001.

In this case, all the table records whose generic key is 001 are loaded in the buffer of the application server.

If you select full buffering, all the records of the table are loaded in the buffer when one record of the program is read.

For example, you select full buffering for the *zhotel* table. Then you use a program to read the first row in the table. In this case all the records of the table will be loaded in the buffer of the application server.

Logging

Logging refers to allowing the changes to the table records to be logged. If you allow logging for a table, all the changes made to the data records of the table, by a user or a program, are recorded in a log table.

The technical settings for a table allow you to switch on or switch off logging.

Switching on logging allows you to monitor the changes made to a table. However it may slow down the access to the database table, because every time a change to the table is made it will be logged in the log table.

Continuing with the *ztourguide* table example, now you need to maintain the technical settings for the table.

The Dictionary: Maintain Table screen is already open.

You click the **Technical Settings** button on the application toolbar to maintain the technical settings of the table. Alternatively, you can press **Ctrl+Shift+F9**.

The Dictionary: Maintain Technical Settings screen appears. You can maintain all the technical settings of the table using this screen.

Suppose the data that you store in this table will be used as transaction data. In this case, you can assign the APPL1 data class to the table.

To do this, you type `APPL1` in the Data class field.

After providing the data class, you need to select the size category for the table. Suppose you expect this table to store 0 to 40,000 data records. In this case, you can assign the size category 0 to this table.

To do this, you type `0` in the Size category field.

After specifying the size category, you need to select the buffering behavior for the table. Suppose you want to allow single-record buffering for this table.

You select the **Buffering switched on** option and then select the **Single records buff.** checkbox to do this.

The single-record buffering for the table will now be allowed. This means that only those records of the *ztourguide* table that are read by a program will be loaded in the buffer.

After selecting the buffering behavior, you want to allow logging for the table.

To do this, you select the **Log data changes** checkbox.

You have selected the technical settings for the table. Now you save the technical settings by clicking **Settings** - **Save**.

After saving the technical settings, you can go back to the Dictionary: Maintain Table screen by clicking the **Back** button on the standard toolbar.

You return to the Dictionary: Maintain Table screen. The table is ready to be used. You can now activate the table object to create the physical table in the database.

You click the **Activate** button on the application toolbar to activate the table. Alternatively, you can press **Ctrl+F3** on the keyboard to activate the table.

The *ztourguide* table is activated and created in the database. You can now enter the required data in this table.

Question

Suppose you are creating a transparent table that will contain master data. This table is expected to store 0 to 40,000 data records. The buffer for this table should only load those table records that are read by a program.

What are the appropriate technical settings for this table?

Options:

1. Data class = APPL0
2. Data class = APPL1
3. Buffering type = Full buffering
4. Buffering type = Single-record buffering
5. Size category = 0
6. Size category = 6

Answer

This table's data class should be APPL0, it should allow single-record buffering, and its size category should be 0.

Option 1 is correct. The APPL0 data class is used to store transparent tables that contain master data. The master data in SAP does not change frequently.

Option 2 is incorrect. The APPL1 data class is used to store transparent tables that contain transaction data. Data in such tables is frequently changed.

Option 3 is incorrect. Full buffering means that all the records of the table are loaded in the buffer when one record of the table is read.

Option 4 is correct. Single-record buffering means that only those table records that are read by a program are loaded in the buffer.

Option 5 is correct. A table with the size category 0 can hold 0 to 40,000 data records. Size category defines the database space that a table is expected to occupy.

Option 6 is incorrect. A table with the size category 6 can hold 6,900,000 to 270,000,000 data records.

Question

Suppose you want to maintain the technical settings for the *zhotel* table. This table will store transaction data and is expected to store 0 to 40,000 records. You want to allow full buffering for this table. The changes to the table records do not need to be logged.

The Dictionary: Maintain Table screen is opened for you.

What are the correct steps to maintain and save the technical settings of this table?

Options:

1. Click the **Technical Settings** button on the application toolbar. On the Dictionary: Maintain Technical Settings screen, type `APPL1` in the Data class field, type `0` in the Size category field, and press **Enter**. Next select the **Buffering switched on** option and then select the **Fully buffered** checkbox. Finally, click **Settings** - **Save**.
2. Click the **Technical Settings** button on the application toolbar. On the Dictionary: Maintain Technical Settings screen, type `APPL0` in the Data class field and type 1 in the Size category field. Next select the **Buffering switched on** option and then select the **Fully buffered** checkbox. Finally, click **Settings** - **Save**.

Answer

You click the **Technical Settings** button on the application toolbar. On the Dictionary: Maintain Technical Settings screen, you type `APPL1` in the Data class

field. Then you type 0 in the Size category field and press **Enter**. Next you select the **Buffering switched on** option and then you select the **Fully buffered** checkbox. Finally, you click **Settings** - **Save**.

Summary

You can use the ABAP Dictionary to create database tables in SAP R/3. To create a table, you first need to provide it a short description and assign it a delivery class. Next you need to create the fields in the table. You can use existing data elements to create the fields.

If you need to check the values entered in a field in the table against the primary key field of another table, you can define foreign key relationships. After defining the required foreign key relationships, you can save the table.

After saving the table you can maintain its technical settings. The important technical settings for a table include: data class, size category, buffering, and logging. After you have saved these settings, you can activate the table.

Table of Contents

| Top of page |

| Learning objective |

| 1. Creating table fields |

| 2. Assigning foreign keys |

| 3. Maintaining technical settings |

| Summary |

Creating ABAP Domains, Data Elements, and Tables

Learning objective

After completing this topic, you should be able to create domains and data elements and use them to create Data Dictionary tables for a given scenario.

Exercise overview

In this exercise, you're required to create an ABAP object to define a value range; assign this object to an ABAP data type that stores the characteristics of a table field; and then use this data type to create a table.

This involves the following tasks:

- Defining a value range
- Defining table field characteristics
- Creating a table field
- Maintaining technical settings

Suppose you work in the marketing department of Awards Sportswear. You are creating a database table in SAP R/3 to store the details of the products your company retails.

Task 1: Defining a value range

In the table, you need to create a field where you can store product codes.

To create this table field, you first want to create an ABAP object that defines a value range suitable for product codes.

You have reached the ABAP Dictionary: Initial Screen.

Create the data object that will define the required value range. Name the data object "zcodes" and provide its short description as "Product codes." The object should allow fields to accept up to 25 characters. You also want the object to check the values entered in fields against the *zproducts* table.

Steps list

Instructions
1. Select the **Domain** option
2. Type `zcodes` in the Domain field and click **Create**
3. Type `Product codes` in the Short Description field, type `char` in the Data type field, and type `25` in the No. characters field; and then press **Enter**
4. Click the **Value range** tab
5. Type `zproducts` in the Value table field and press **Enter**

Task 2: Defining table field characteristics

After defining the value table for the *zcodes* domain, you saved and activated the domain.

Now you want to create an ABAP data type to store the characteristics of the table field you require.

You have accessed the ABAP Dictionary: Initial Screen.

Now access the screen where you can define the appropriate data type to store the characteristics of the required table field (Name the data type "zproduct_codes"). On the screen, provide "Product codes field" as the short description, and assign the *zcodes* domain to the data type.

Steps list

Instructions
1. Select the **Data type** option
2. Type `zproduct_codes` in the Data type field and click **Create**
3. Click **Continue**
4. Type `Product codes field` in the Short Description field and press **Enter**
5. Type `zcodes` in the Domain field and press **Enter**

Task 3: Creating a table field

After assigning the *zcodes* domain to the *zproduct_codes* data element, you provided the field labels and documentation to the data element and activated it.

After you created the *zproduct_codes* data element, you begin creating a database table called *zretail* to store the details of the products Award Sportswear retails.

While creating the table, you need to create the table field to store product codes. You want to use the *zproduct_codes* data element to create this field.

You have opened the Fields tabbed page on the Dictionary: Maintain Table screen.

Now add a field called "Product_code" to the table, make it a key field for the table, and assign the *zproduct_codes* data element to the field.

Steps list

Instructions
1. Type `Product_code` in the first row in the **Fields** column
2. Select the corresponding checkbox in the **Key** column
3. Type `zproduct_codes` in the first row of the Data element column and press **Enter**

Task 4: Maintaining technical settings

After adding the Product_code field, you added two more fields, called Product_name and Product_price, to the table. Then you saved the table in the ABAP Dictionary.

Now you need to maintain its technical settings and create the physical table in the database.

You are on the Fields tabbed page on the Dictionary: Maintain Table screen.

The *zretail* table will contain transaction data and is expected to store 0 to 5,100 records. You want to allow single-record buffering for this table. The changes to the table records do not need to be logged.

Maintain the technical settings for the *zretail* table and then create the physical table in the database.

Steps list

Instructions
1. Click the **Technical Settings** button
2. Type `APPL1` in the Data class field and type `0` in the Size category field; and then press **Enter**
3. Select the **Buffering switched on** option
4. Select the **Single records buff.** checkbox
5. Click **Settings** - **Save**
6. Click the **Back** button
7. Click the **Activate** button

Table of Contents

| Top of page |

| Learning objective |

| Exercise overview |

| Task 1: Defining a value range |

| Task 2: Defining table field characteristics |

| Task 3: Creating a table field |

| Task 4: Maintaining technical settings |