



Dr.SNS RAJALAKSHMI COLLEGE OF ARTS AND SCIENCE
(Autonomous)

Coimbatore -641049

Accredited by NAAC (Cycle- III)with 'A+' Grade
(Recognised by UGC, Approved by AICTE, New Delhi and
Affiliated to Bharathiar University, Coimbatore



UNIT – I

DR.A.DEVI

Associate Professor

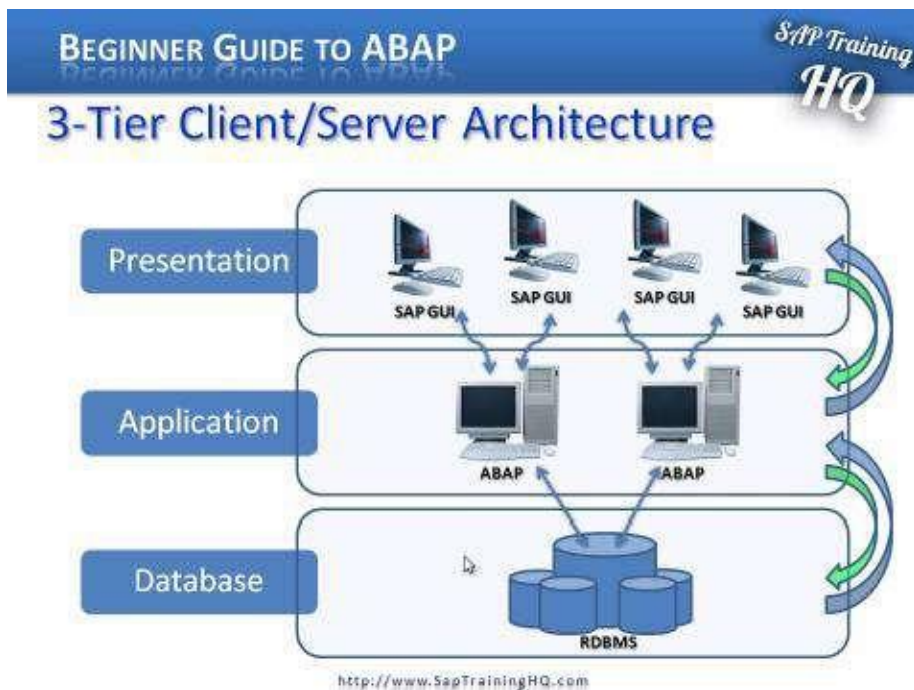
Department of Computer Applications

DRSNSRCAS

SAP System Architecture

First, the Technical Architecture of a typical SAP system will be discussed, before moving on to the Landscape Architecture, and a discussion of why the landscape should be broken into multiple systems.

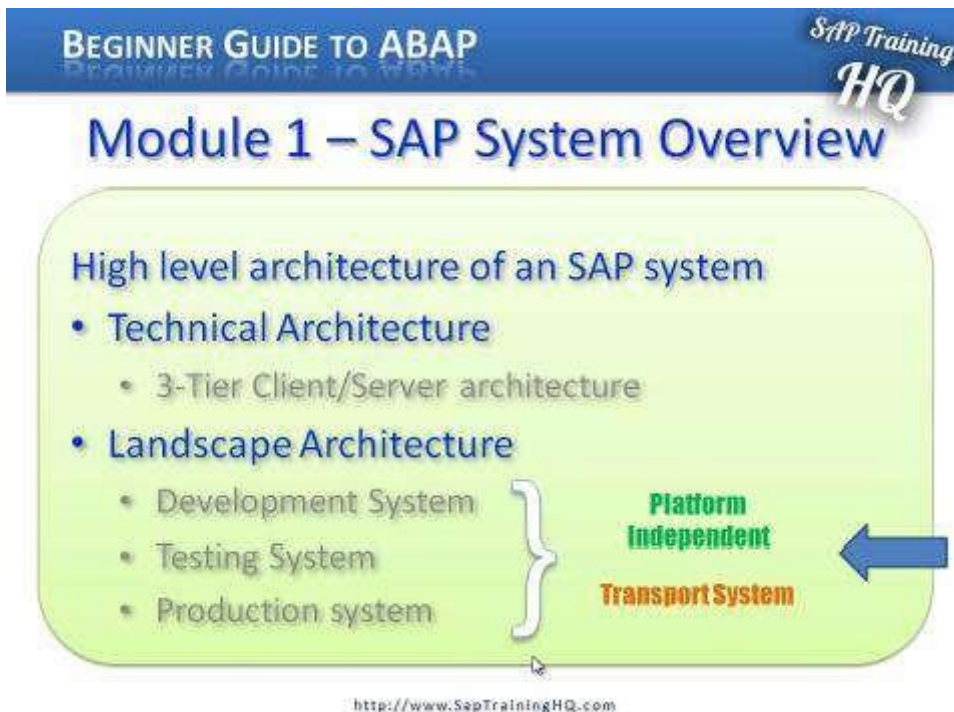
This diagram shows the 3-tier Client/Server architecture of a typical SAP system:



At the top is the Presentation server, which is any input device that can be used to control an SAP system (the diagram shows the SAP GUI, but this could equally be a web browser, a mobile device, and so on). The Presentation layer communicates with the Application server, and the Application server is the 'brains' of an SAP system, where all the central processing takes place. The Application server is not just one system in itself, but can be made up of multiple instances of the processing system. The Application server, in turn, communicates with the Database layer.

The Database is kept on a separate server, mainly for performance reasons, but also for security, providing a separation between the different layers of the system.

Communication happens between each layer of the system, from the Presentation layer, to the Application server, to the Database, and then back up the chain, through the Application server again, for further processing, until finally reaching the Presentation layer.



A typical Landscape Architecture - Typical here is subjective, in practical terms there is not really any such thing as a standard, 'typical' landscape architecture which most companies

use. However, it is common to find a Development system, a Testing system and a Production system:

The reason for this is fairly simple. All the initial development and testing is done on a Development system, which ensures other systems are not affected. Once developments are at a stage where they may be ready to be tested by an external source, or someone within the company whose role is to carry out testing, the developments are moved, using what is called a Transport System, to the next system (here, the Testing system).

Normally, no development at all is done on the testing system; it is just used for testing the developments from the development system. If everything passes through the Testing system, a Transport system is used again to move the developments into the Production environment. When code enters the Production environment, this is the stage at which it is turned on, and used within the business itself.

The landscape architecture is not separated just for development purposes; the company may have other reasons. This could be the quantity of data in the Production system, which may be too great to be used in the development environment (normally the Development and Testing systems are not as large as the Production system, only needing a subset of data to test on). Also, it could be for security reasons. More often than not, companies do not want developers to see live production data, for data security reasons (for example, the system could include employee data or sales data, which a company would not want people not employed in those areas to see). Normally, then, the Development and Testing systems would have their own set of data to work with.

The three systems described here, normally, are a minimum. It can increase to four systems, perhaps with the addition of a Training system, or perhaps multiple projects are running simultaneously, meaning there may be two separate Development systems, or Testing systems, even perhaps a Consolidation system before anything is passed to the Production environment. This is all, of course, dependent on the company, but commonly each system within the Landscape architecture will have its own Application server and its own Database server, ensuring platform independence.

Environment for Programs

Next, we have the environment which programs run in, the Work Processes, and the structure of an ABAP program.

The slide features a blue header with the text 'BEGINNER GUIDE TO ABAP' and 'SAP Training HQ' in the top right corner. The main title is 'Module 1 – SAP System Overview'. Below this, a rounded blue box contains the sub-heading 'Environment for our Programs' and a bulleted list of components.

- 2 Types of Programs
 - Reports & Dynpros
- Work Processes
 - *Dispatcher*
 - Dynpro Processor
 - ABAP Processor
 - Database Interface

At the bottom of the slide, there is a small logo on the left and the URL <http://www.SapTrainingHQ.com> on the right.

Within an SAP system, or at least the example used here, there are two types of programs, Reports and Dynpro's.

Reports, as the name would suggest, are programs which generate lists of data. They may involve a small amount of interactivity, but mainly they supply data to the front-end interfaces, the SAP GUI and so on. When a user runs a report, they typically get a selection screen. Once they enter their selection parameters and execute the report, they normally cannot intervene in the execution of the program. The program runs, and then displays the output.

Dynpro's are slightly different. They are dynamic programs, and allow the user to intervene in the execution of the program, by processing a series of screens, called

Dialogue screens. The user determines the flow of the program itself by choosing which buttons or fields to interact with on the screen. Their action then triggers different functions which have been coded within the flow logic of the program. While reports are being created, interfaces are also to be generated which are classed as Dynpro's, for all the selection criteria.

Most of the work done by people involved with ABAP is done within Report programs, and even though these programs are labelled 'Reports', they do not always generate output. The Report programs are there to process the logic, reading and writing to the Database, in order to make the system work.

Work Processes

Every program that runs in an SAP system runs on what are called Work Processes, which run on the Application server. Work Processes themselves work independently of the computer's operating system and the Database that it interacts with, giving the independence discussed earlier with regard to the Technical architecture. When an SAP system is initially set up, the basis consultants (who install the system, keep it running, manage all the memory and so on) configure SAP in such a way that it automatically sets the number of Work Processes programs use when they start, the equivalent of setting up a pre-defined number of channels or connections to the Database system itself, each of which tend to have their own set of properties and functions.

The Dispatcher

You might come across something referred to as the Dispatcher. The SAP system has no technical limits as to the number of users who can log on and use it, generally the number of users who can access an SAP system is much larger than the number of available Work Processes the system is configured for. This is because not everybody is sending instructions to the Application server at exactly the same time. Because of this, users cannot be assigned a certain number of processes while they are logged on.

The Dispatcher controls the distribution of the Work Processes to the system users. The Dispatcher keeps an eye on how many Work Processes are available, and when a user triggers a transaction, the Dispatcher's job is to provide that user with a Work Process to use. The Dispatcher tries to optimise things as far as possible, so that the same Work Process receives the sequential Dialogue steps of an application. If this is not possible, for example because the user takes a long time between clicking different aspects of the

screen, it will then select a different Work Process to continue the processing of the Dialogue program. It is the Work Process which executes an application, and it is the Work Process which has access to the memory areas that contain all of the data and objects an application uses. It also makes three very important elements available.

The first is the Dynpro processor. All Dynpro programs have flow and processing logic, and it is the Dynpro processor's job to handle the flow logic. It responds to the user's interactions, and controls the further flow of the program depending on these interactions. It is responsible for Dialogue control and the screen itself, but it is important to remember that it cannot perform calculations; it is purely there to manage the flow logic of a program.

The next important element is the ABAP processor, which is responsible for the processing logic of the programs. It receives screen entries from the Dynpro processor, and transmits the screen output to the program. It is the ABAP processor which can perform the logical operations and arithmetical calculations in the programs. It can check authorisations, and read and write to the Database, over the Database Interface.

The Database Interface

The Database Interface is the third important element. It is a set of ABAP statements that are Database independent. What this means is that a set of ABAP statements can be used that, in turn, can communicate with any type of Database that has been installed when the system was set up. Whether this is, for example, a Microsoft SQL server or an Oracle Database, you can use the same ABAP statements, called Open SQL, to control the entire Database reading and writing over the Database Interface. The great advantage of this is that the ABAP statements have encapsulation, meaning the programmers do not need to know which physical Database system the ABAP system they are using actually supports.

There are times when you may want to use a specific SQL statement native to the database which is installed. ABAP is designed in such a way that if this type of coding is necessary, this facility is available. It is possible to directly access the Database through the programs using native SQL statements, but this is not encouraged. Normally, when systems are set up, the system administrator will forbid these practices, due to the security and stability risks to the system which may be introduced. If you are going to be programming ABAP, make sure Open SQL is used, because then anyone subsequently looking at the programs will understand what is trying to be achieved.

First look at the ABAP Workbench

It is now time to take a first look at an SAP ABAP program. The following section will look at the SAP System and introduce the ABAP Workbench. But before doing so, let's take a look at the structure of an ABAP program.

The slide features a blue header with the text 'BEGINNER GUIDE TO ABAP' and 'SAP Training HQ' in the top right corner. The main title is 'Module 1 – SAP System Overview'. Below the title, a light orange rounded rectangle contains the text 'Lets get running with the system.' followed by a bulleted list of two items: '• How An ABAP Program Is Structured' and '• First Look At The ABAP Workbench'. At the bottom of the slide, the URL 'http://www.SapTrainingHQ.com' is displayed.

Like many other programming languages, ABAP programs are normally structured into two parts.

Part 1 - Declaration Section
Part 2 - Processing Blocks.

The first is what is considered to be the Declaration section. This is where you define the data types, structures, tables, work area variables and the individual fields to be used inside the programs. This is also where you would declare global variables that will be available throughout the individual subsections of the program. When creating an ABAP program, you do not only declare global variables, but you also have the option to declare

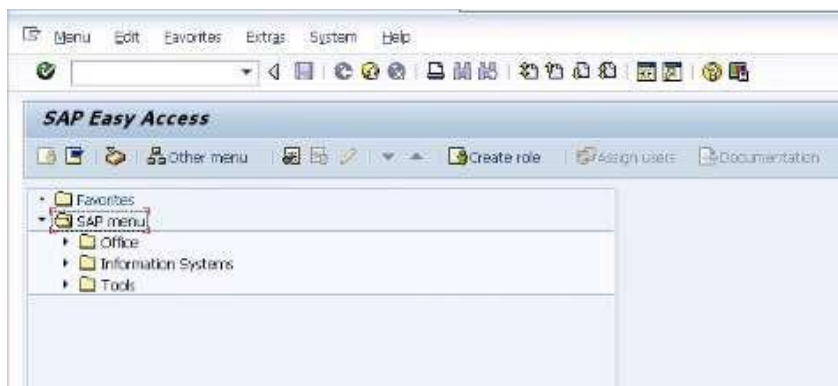
variables that are only valid within specific sections inside the programs. These sections are commonly referred to as internal Processing Blocks.

The Declaration part of the program is where you define the parameters used for the selection screens for the reports. Once you have declared tables, global variables and data types in the Declaration section of the program, then comes the second part of the ABAP program, where all of the logic for the program will be written. This part of an ABAP program is often split up into what are called Processing Blocks.

The Processing Blocks defined within programs can be called from the Dynpro processor, which were discussed previously, depending on the specific rules created within the program. These Processing Blocks are almost always just small sections of programming logic which allow the code to be encapsulated.

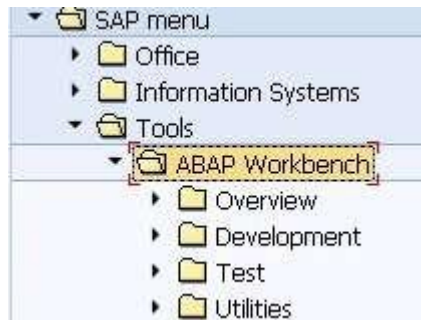
First Look

When logged into an SAP system it will look something similar to the image below.

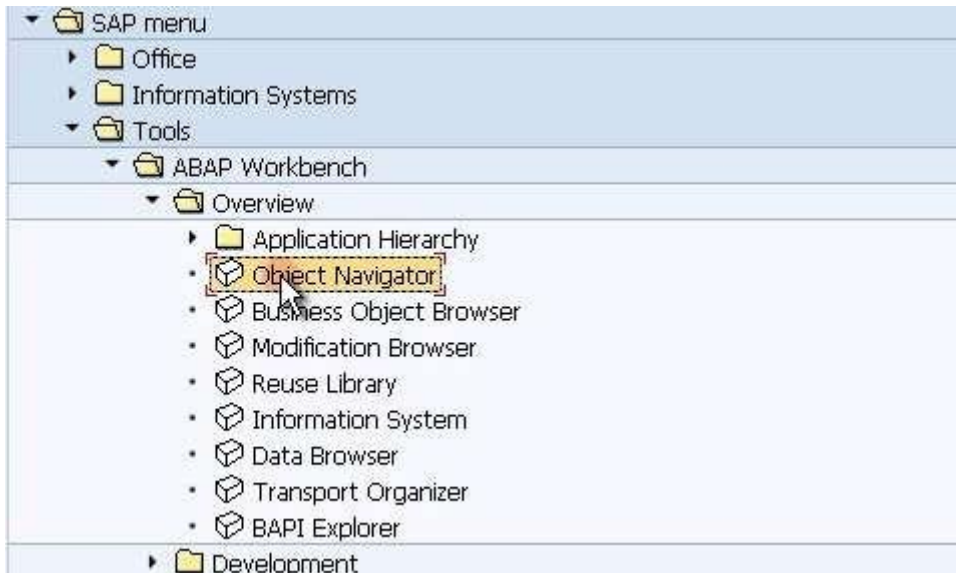


The way the SAP GUI looks may vary, the menu to the side may be different, but here the display shows a minimal menu tree which will be used throughout this book.

The first thing to do here is look at the ABAP Workbench. To access this, you use the menu on the left hand side. Open the SAP menu, choose Tools and open the ABAP Workbench, where there will be four different options.




The first thing to look at is a quick overview of how to run a transaction in SAP. There are two ways to do this. Firstly, if the overview folder is opened, any item which does not look like a folder itself, is a transaction which can be run. In this instance, we can see the Object Navigator:



Double click this, and the transaction will open:

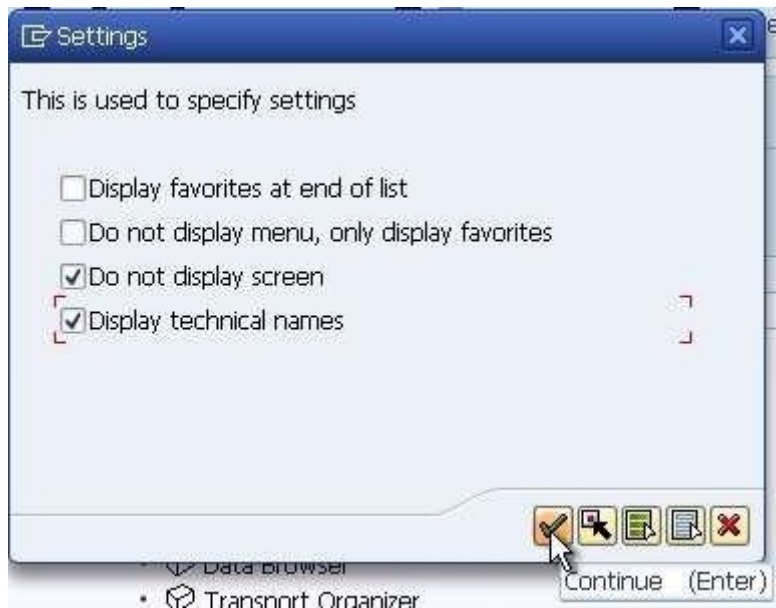


To exit out of the transaction, click the Back button: 

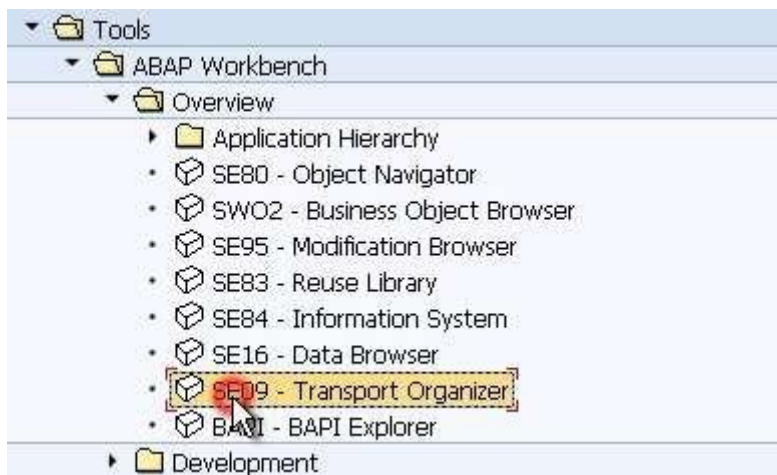
The second way of running a transaction is to enter the transaction code into the transaction code input area:



A useful tip to become familiar with the names of transactions is to look at the Extras menu --> open Settings and in the dialogue box which appears, select the option 'Display technical names' and click the 'Continue' icon:



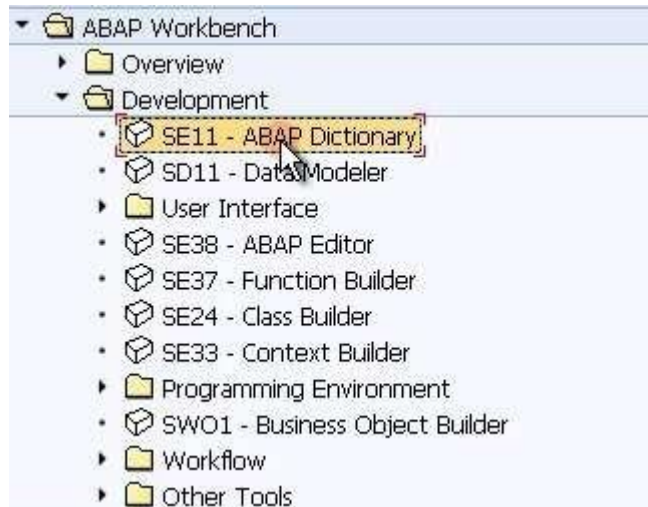
The menu tree will be refreshed, and when the 'Overview' folder is opened, the transaction codes will be made visible. It is now possible to become familiar with them, and enter them directly into the transaction code input area:



Now, a step-by-step look will be taken through the major transactions of the ABAP Workbench to become familiar with, and use, as an ABAP developer.

ABAP Dictionary

One thing most programs will have in common is that they will read and write data to and from the Database tables within the SAP system. The ABAP Workbench has a transaction to allow the creation of Database tables, view the fields which make up these tables and browse the data inside. This is called the ABAP Dictionary. The ABAP Dictionary can be found by expanding the ABAP Workbench menu tree --> 'Development'. The transaction code to run the ABAP dictionary directly is SE11:



ABAP Editor

The next and probably most commonly used part of the ABAP Workbench is the ABAP Editor, which much of this course will focus upon. The ABAP Editor is where all of the code is created, the logic built and, by using forward navigation (a function within an SAP system which will be discussed later), function modules defined, screens created and so on. The ABAP Editor can be found under the 'Development' menu, as shown above and with transaction code SE38.

Function Builder

The next important part of the Workbench is the Function Builder, which is similar to the ABAP Editor. Its main function is to define specific tasks that can be called from any other program. Interfaces are created in the Function Builder, where the different data elements and different types of tables are defined, that can be passed to and from the Function which is built. The Function Builder will be discussed a little later on, when the programs created are encapsulated into function modules. The Function Builder can be

called with transaction code SE37.

Menu Painter

The next item to look at here is called the Menu Painter, which can be found in the 'User Interface' folder inside the 'Development' menu, or with transaction code SE41. This is a tool which can be used to generate menu options, buttons, icons, menu bars, transaction input fields, all of which can trigger events within the program. You can define whether events are triggered using a mouse click, or with a keyboard-based shortcut. For example, in the top menu bar here, the 'Log off' button can be seen, which can be triggered by using (Shift + F3):



Screen Painter

While the Menu Painter is used for building menu items, menu bars and so on, the next item on the list is the Screen Painter, transaction code SE51, which allows you to define the user input screen, meaning that you can define text boxes, drop-down menus, list boxes, input fields, tabbed areas of the screen and so on. It allows you to define the whole interface which the user will eventually use, and behind the initial elements that are put on the screen, you can also define the individual functions which are called when the user interacts with them.

Object Navigator

The last item to look at here is the Object Navigator, a tool which brings together all the previous tools, providing a highly efficient environment in which to develop programs. When building large programs, with many function modules, many screens, the Object Navigator is the ideal tool to use to navigate around the development. It can be found in the 'Overview' menu of the ABAP Workbench, with transaction code SE80.

These are the main features of the ABAP Workbench interacted with during this course. In the SAP menu tree, there are evidently many more transactions which can be used to help develop programs, but these cover the vast majority of development tools which will be used.

Data Dictionary

This is the main tool used to look at, understand and enhance the Database and Database tables which are used by the SAP system. You can view standard tables delivered by SAP using this tool, create new tables and enhance the existing tables delivered by SAP with new fields. There are many other features involved in the Data Dictionary, but the focus here will be on the basic ones so as to build on this later on when creating ABAP programs.


First, a database table will be created, involving the creation of fields, data elements and domains. An explanation of what each of these is, and why they are necessary to the tables built will be given. During the building of the tables, the tools used to check for errors will be shown. Once these errors are eradicated, the tables can be activated so that they can be used within the system.

After this, a look will be taken at maintaining the technical settings of the table created, which will allow the entry of data, before finally looking at the data which has been entered using standard SAP transactions available in the SAP system.

Creating a Table

With the SAP GUI open, you will be able find the Data Dictionary in the SAP menu tree. This is done via the Tools menu. Open the ABAP Workbench and click the 'Development' folder, where the ABAP Dictionary can be found and double-clicked. Alternatively, use the transaction code SE11:

Now, the initial screen of the ABAP Dictionary will appear:



The screenshot shows a software interface for creating a database object. At the top, there is a toolbar with icons for home, search, navigation, and other functions. Below the toolbar, there are several radio button options for selecting the object type: 'Database table' (selected), 'View', 'Data type', 'Type Group', 'Domain', 'Search help', and 'Lock object'. To the right of these options are several text input fields. The first field, corresponding to the 'Database table' option, is highlighted with a yellow background and has a small icon to its right. Below the input fields, there are three buttons: 'Display' (with a magnifying glass icon), 'Change' (with a pencil icon), and 'Create' (with a document icon).

To create a table, select the 'Database table' option. In this exercise a transparent table will be created. Other types of table do exist (such a cluster tables and pool tables), but at this early stage the transparent table variety is the important one to focus upon.

The table name must adhere to the customer-defined name space, meaning that the name must begin with the letter Z or Y, most commonly this will be Z. In this example, the table will show a list of employees within a company, so, in the 'Database table' area, type 'ZEMPLOYEES' and click the 'Create' button.

Once this is done, a new screen will appear:

Database table: ZEMPLOYEES
 View:
 Data type:
 Type Group:
 Domain:
 Search help:
 Lock object:

Dictionary: Maintain Table www.SapTrainingHQ.com

Transp. table: ZEMPLOYEES New(Revised)
 Short text:

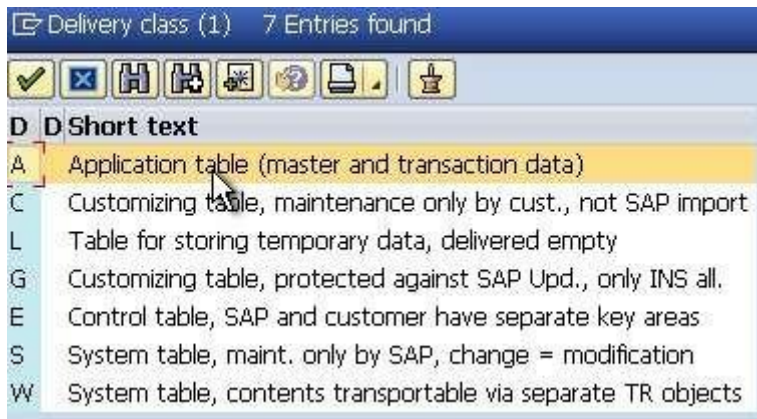
Delivery Class:

Data Browser/Table View Maint.:

In the 'Short text' field, a description for the table must be included, enter 'Employees':

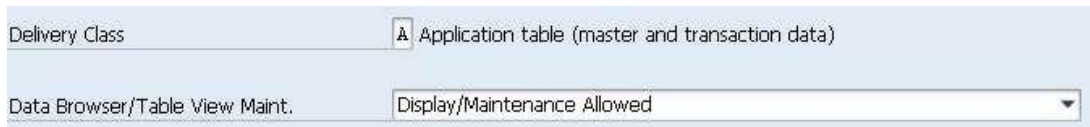
Transp. table	ZEMPLOYEES	New(Revised)
Short text	Employees	


In the 'Delivery and Maintenance' tab (which opens by default), look at the 'Delivery class' section, select the field and then click the drop-down button, where a list of Delivery classes will be shown and selected:



For the table being created here, choose 'Application table', as the data held in the table fits the description 'master and transaction data'.

In the field below this, labelled 'Data Browser/Table View Maint.', choose the 'Display/Maintenance allowed' option, which will allow for data entry directly into the table later on. It should look like this:



Before going any further, click the 'Save' button: 

A window appears titled 'Create Object Directory Entry'.

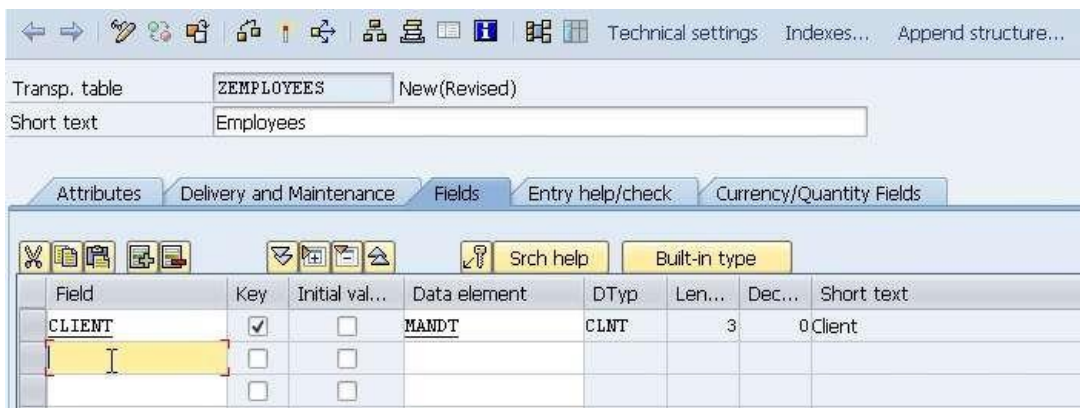
Nearly all development work done with SAP is usually done within a development environment, before being moved on to, for example, a quality assurance environment and on further to production. This window allows you to choose the appropriate Development class which is supported by other systems where the work may be moved on to. In this example scenario, though, developments will not be moved on to another system, so click '**Local object**', so as to indicate to the system (via the phrase '\$TMP' which appears) that the object is only to exist within the development system and not to be transported elsewhere. Once this is done, the status bar at the bottom will show that the object has been saved:



To check everything has worked as we want, select the 'Go to' menu and selects the 'Object directory entry' option, a similar pop-up box to the previous one will appear, where the 'Development class' field will show '\$TMP', confirming this has been done correctly.

Creating Fields

The next step is to begin creating Field names for the table, in the 'Fields' tab:



Fields, unlike the name of the table, can begin with any letter of the alphabet, not just Z and Y and can contain up to 16 characters.

Tables must include at least one Key field, which is used later for the searching and sorting of data, and to identify each record as being unique.


An Initial value can be assigned to each field, for example, in the case of a field called *Employee Class* you could say the majority of employees are Regular Staff ('S'), but some are Directors, with a code of 'D'. The standard initial value would be 'S', but the user could change some of these to a 'D' later on.

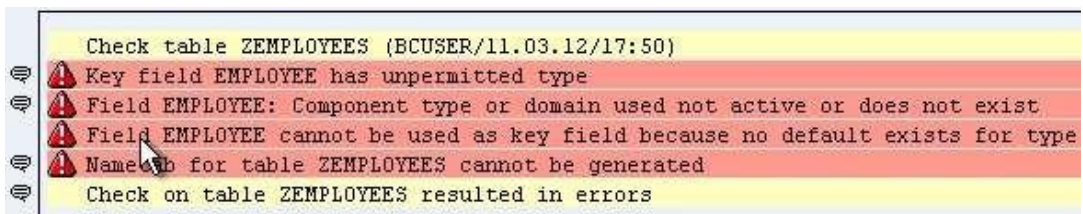
Data Elements

Every Field in the table is made up of what is called a Data Element, which defines specific attributes of each field.

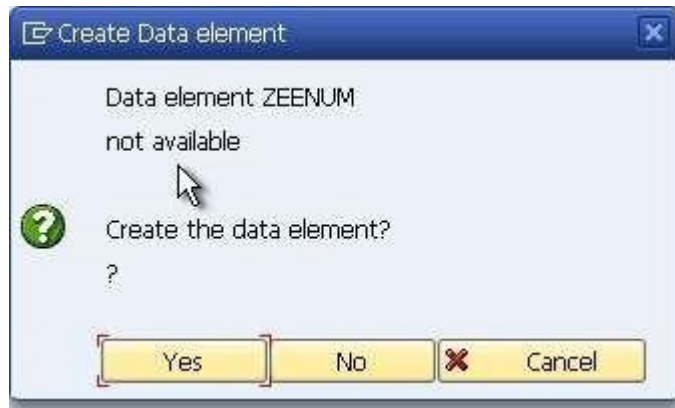
The first Field to be created here is an important one within an SAP system, and identifies the client which the records are associated with. In the Field name, enter 'Client', and in the Data Element, type 'MANDT'. This Data Element already exists in the system, and after entering it, the system automatically fills in the Data Type, the Length, Number of Decimals and Short text for the Data Element itself. Ensure that the 'Client' field is made a Key field in the table by checking the 'Key' box.

The next field will be called 'Employee'. Again, check the box to make this a Key field, and enter the new Data Element 'ZEENUM' (Data Elements broadly must adhere to the customer name space by beginning with Z or Y). Once this is done, click the save button.

Next, because the Data Element 'ZEENUM' does not yet exist, it must be created. If you try to activate or even check the table (via the 'Check'  button), an error message is displayed:



Until the Data Element 'ZEENUM' is created, it cannot be used within the system. To do this, forward navigation is used. Double-click the new Data Element, and a window labelled 'Create Data element' appears. Answer 'Yes' to this, and the 'Maintain Data Element' window comes up.



Dictionary: Maintain Data Element www.SapTraining.com

← → ✎ ↻ 📁 🔒 🔑 🔄 📦 📄 📖 Documentation Supplementary d

Data element New(Revised)

Short text

Attributes Data Type Further Characteristics Field label

Elementary type

Domain

📄

Data Type

Length Decimal Places

Built-in type

Data type

Length Decimal places

Reference type

Name of Ref. Type

Reference to Predefined Type

Data Type

Length Decimal Places

In the 'Short text' area, enter 'Employee Data Element'. Next, the Elementary data type, called the 'Domain', must be defined for the new Data element. Domains must adhere to the customer name space, so in this instance the same name as the Data element will be given: 'ZEENUM', (though giving both the same name is not imperative). Again, forward navigation will be used to create the Domain.

Data Domains

Double-click the entry ('ZEENUM') in the Domain area, and agree to save the changes made. Now, the 'Create Object Directory Entry' window will re-appear and again it is important to save this development to the '\$TMP' development class, via the 'save' or 'local object' button visible in this window.

After doing this, a window will appear stating that the new Domain 'ZEENUM' does not exist. Choose 'Yes' to create the Domain, and in the window which appears, type into the 'Short text' box a description of the Domain. In this example, 'Employee Domain':

Dictionary: Maintain Domain www.SapTraining

Domain: ZEENUM New(Revised)

Short text: Employee Domain

Attributes | **Definition** | Value range

Formatting

Data type	NUMC	Character string with only digits
No. characters	8	
Decimal places	0	

Output characteristics

Output length	8
Convers. routine	
<input type="checkbox"/> Sign	
<input type="checkbox"/> Lowercase	

The 'Definition' tab, which, as shown above, opens automatically. The first available field here is 'Data type', click inside the box and select the drop-down menu, and a number of generic data types already existing within the ABAP dictionary will appear.

The 'NUMC' type is the one to be used here for the Employee data, a “character string with only digits”. Once this selection is double-clicked, it will appear in the 'Data type' area in the 'Definition' tab.

Next, in the 'No. characters' field, enter the number 8, indicating that the field will contain a maximum of 8 characters, and in the 'Decimal places' area, enter 0. An Output length of 8 should be selected, and then press Enter.

The 'NUMC' field's description should re-appear, confirming that this is a valid entry.

Next, select the 'Value range' tab, which is visible next to the 'Definition' tab just used:


The screenshot shows the 'Value range' tab in the ABAP Data Dictionary. It features three main sections: 'Single vals', 'Intervals', and 'Value table'. The 'Single vals' section contains a table with columns 'Fix.val.' and 'Short text'. The first row in this table is highlighted in yellow. The 'Intervals' section contains a table with columns 'Lower limit', 'UpperLimit', and 'Short text', which is currently empty. The 'Value table' section at the bottom has a text input field that is also empty.

This is where you set valid value ranges for the Domain created. Once this is set, any subsequent user entering values outside the valid value range will be shown an error

message and be requested to enter a valid entry. Here, there are three options.

- First, where you can see 'Single values', it is possible to enter a list of individual valid values which can be entered by the user.
- Second, 'Intervals', where you can enter a lower and upper limit for valid values, for example 1 and 9, which saves the effort of entering 9 individual single values in the 'Single values' section.
- Last, the 'Value table' box visible at the bottom. When there are a large number of possible entries, this is a common method (to do this you must specify a complete valid value table entry list, in which case it is also necessary to introduce foreign keys to the table, to ensure the user's entries are tested against the value stored in the value table created).

This example Domain, however, does not require any Value range entry, so just click the save button and, again, assign it as a 'Local object'.

The next step is to Activate the object, allowing other Data elements to use this domain going forward. In the toolbar click the small matchstick icon  (also accessible by pressing CTRL +F3).

A pop-up window appears, listing the 3 currently inactive objects:

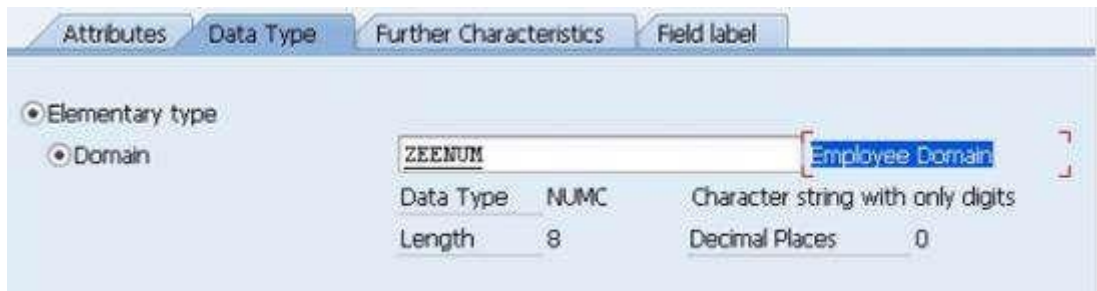
Transportable objects		Local objects	
Object name			
C	Obj...	Obj. name	User
<input checked="" type="checkbox"/>	DOMA	ZEENUM	BCUSER
<input type="checkbox"/>	DTEL	ZEENUM	BCUSER
<input type="checkbox"/>	TABL	ZEMPLOYEES	BCUSER

It may be possible to activate all of the objects together, but this is not advised. In a typical development environment, a number of people will be creating developments simultaneously, and quite often, others' objects will appear in this list.

At this point, it is only the Domain which is to be activated, the top entry labelled 'DOMA', with the name 'ZEENUM'. When this is highlighted, click the green tick continue button. The window should disappear, and the status bar will display the message 'Object(s)

activated'

Now it is possible to proceed with the creation of the table. Forward navigation was used for generating the Domain, so click the 'Back' button, or press F3 to return to the 'Maintain Data Element' screen. As the domain is active, the description entered previously should appear by the area where 'ZEENUM' was typed, along with other Domain properties which have been created:



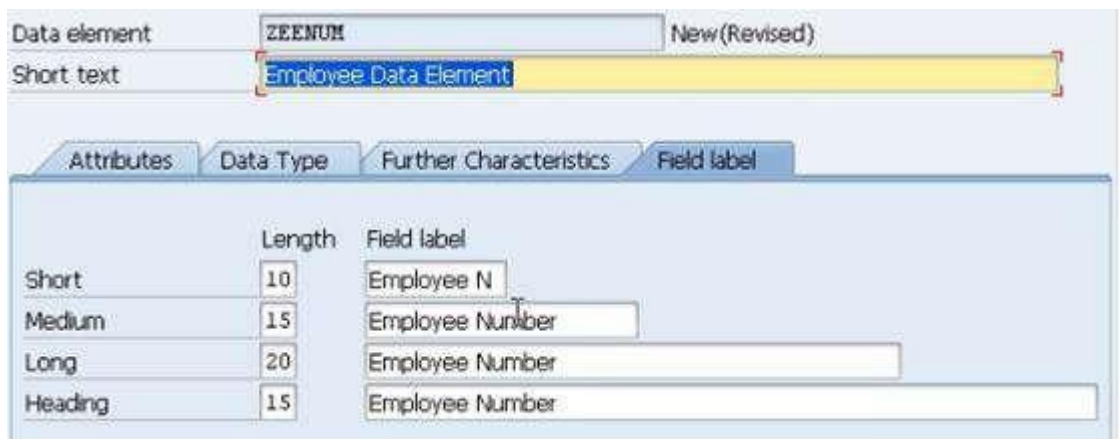
Attributes | **Data Type** | Further Characteristics | Field label

Elementary type
 Domain

ZEENUM Employee Domain

Data Type NUMC Character string with only digits
 Length 8 Decimal Places 0


Next, the Field labels must be created, so click that tab. The Field labels entered here will appear as field labels in the final table. In this example they should read 'Employee', or better, 'Employee Number'. If this does not fit within the area given, just tailor it so that it still makes sense, for example typing 'Employee N' into the 'Short' Field label box. Once the text has been put into the Field label spaces, press enter, and the 'Length' section will automatically be filled in:



Data element ZEENUM New (Revised)
 Short text Employee Data Element

Attributes | Data Type | Further Characteristics | **Field label**

	Length	Field label
Short	10	Employee N
Medium	15	Employee Number
Long	20	Employee Number
Heading	15	Employee Number

Once this is complete, Save and Activate the element via the toolbar at the top. The inactive objects window will reappear, where two inactive objects will remain. Highlight the Data element (labelled 'DTEL') and click the green tick 

Continue button at the bottom.

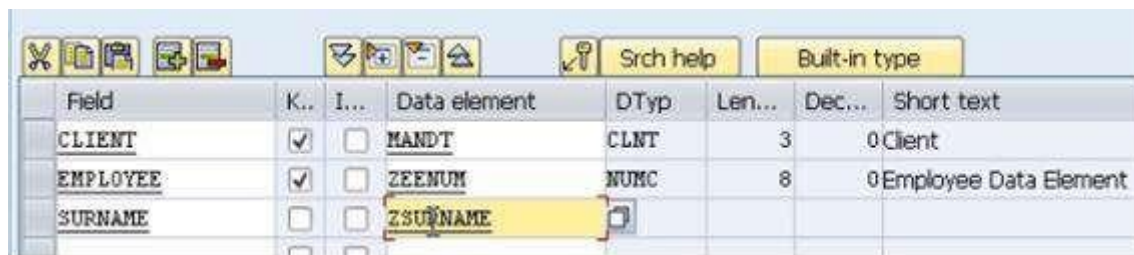
Again, the status bar should display 'Object(s) activated'.

Press the back button to return to the Table maintenance screen. Here you will now see that the 'EMPLOYEE' column has the correct Data Type, Length, Decimals and Short text, thus indicating the successful creation of a Data element and Domain being used for this Field.



Field	K..	I...	Data element	DTyp	Len...	Dec...	Short text	Group
CLIENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	MANDT	CLNT	3		Client	
EMPLOYEE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ZEENUM	NUMC	8		Employee Data Element	

The next field to create should be titled 'SURNAME'. This time it should *not* be selected as a Key field, so do not check the box. The Data element, in this instance, is labelled 'ZSURNAME':



Field	K..	I...	Data element	DTyp	Len...	Dec...	Short text	Group
CLIENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	MANDT	CLNT	3		Client	
EMPLOYEE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ZEENUM	NUMC	8		Employee Data Element	
SURNAME	<input type="checkbox"/>	<input type="checkbox"/>	ZSURNAME					

Now, forward navigation will again be used. Double-click 'ZSURNAME'; choose 'Yes' to save the table and 'Yes' again to create the new Data element. The 'Maintain Data Element' window will appear which will be familiar from the previous steps.

In the 'Short text' box this time type 'Surname Data Element' and title the new domain

'ZSURNAME':

The screenshot shows the 'Data element' screen for 'ZSURNAME'. The 'Short text' field contains 'Surname Data Element'. The 'Data Type' tab is selected, showing 'ZSURNAME' as the domain, 'Data Type' as 'CHAR', 'Length' as '0', and 'Decimal Places' as '0'. The 'Domain' radio button is selected under 'Elementary type'.

Double-click the new domain and save the Data element, assigning it a 'Local object' and then choose 'Yes' to create the new Domain.

The Domain maintenance screen will reappear. Enter the short text 'Surname' and, this time; the Data type to select is 'CHAR', a Character string.

The number of characters and output length should both be set to 40, then press enter to be sure everything has worked, and click the Activate button.

The screenshot shows the 'Dictionary: Maintain Domain' screen for 'ZSURNAME'. The 'Short text' field contains 'Surname'. The 'Definition' tab is selected, showing 'Data type' as 'CHAR' (Character string), 'No. characters' as '40', and 'Decimal places' as '0'. Under 'Output characteristics', 'Output length' is set to '40'. The 'Activate (Ctrl+F3)' button is highlighted.

Note that the Save button has not been pressed this time, as the Activate button will also save the work automatically. Ensure you assign the object to the \$TMP development class as usual.

In the Activate menu, select the object (the domain (labelled 'DOMA') named 'ZSURNAME') to be activated, and click the green tick continue button. The status bar should read 'Object saved and activated'.

Following this, click Back or F3 to return to the Maintain Data element screen. Ensure the domain attributes have appeared (Short text, Data type, Length and so on). In the Field Label tab, enter 'Surname' in each box and press Enter to automatically fill the 'Length' boxes and then activate the Data element (in the Activate menu, the 'DTEL' object named 'ZSURNAMES'), checking the status bar to ensure this has occurred with any errors:

Dictionary: Maintain Data Element www.SapTrainingHQ.co

← → Documentation Supplementary documentat

Data element: ZSURNAM Activate (Ctrl+F3) New(Revised)

Short text: Surname Data Element

Attributes Data Type Further Characteristics **Field label**

	Length	Field label
Short	10	Surname
Medium	15	Surname
Long	20	Surname
Heading	40	Surname

Again, press Back to return to the Maintain Table screen, where the new Data element will be visible:

Transp. table	ZEMPLOYEES		New				
Short text	Employees						
<div style="display: flex; justify-content: space-between;"> Attributes Delivery and Maintenance Fields Entry help/check Currency/Quantity Fields </div>							
<div style="display: flex; justify-content: space-between;"> ✂ 📄 📁 🔄 📄 📁 🔑 Srch help Built-in type </div>							
Field	K..	I...	Data element	DTyp	Len...	Dec...	Short text
CLIENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	MANDT	CLNT	3		0Client
EMPLOYEE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ZEENUM	NUMC	8		0Employee Data Element
SURNAME	<input type="checkbox"/>	<input type="checkbox"/>	ZSURNAME	CHAR	40		0Surname Data Element

The next field to be created is titled 'FORENAME', and the data element 'ZFORENAME'. Click to create the Data element and follow the steps above again.

In the Maintain Data Element screen, the Short text should read 'Forename Data Element' and the domain 'ZFORENAME'. Save this and choose 'Yes' to create the domain.

The domain's short text should read 'Forename'. Use the CHAR data type again and a Length and Output length of 40. Next, Activate the Domain as before.

Return to the Maintain Data Element screen. Type 'Forename' into the four Field label boxes. Press *enter* to fill the length boxes and then Activate the Data Element named 'ZFORENAME' as before. Go back again to see the table:

Transp. table	ZEMPLOYEES		New				
Short text	Employees						
<div style="display: flex; justify-content: space-between;"> Attributes Delivery and Maintenance Fields Entry help/check Currency/Quantity Fields </div>							
<div style="display: flex; justify-content: space-between;"> ✂ 📄 📁 🔄 📄 📁 🔑 Srch help Built-in type </div>							
Field	K..	I...	Data element	DTyp	Len...	Dec...	Short text
CLIENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	MANDT	CLNT	3		0Client
EMPLOYEE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ZEENUM	NUMC	8		0Employee Data Element
SURNAME	<input type="checkbox"/>	<input type="checkbox"/>	ZSURNAME	CHAR	40		0Surname Data Element
FORENAME	<input type="checkbox"/>	<input type="checkbox"/>	ZFORENAME	CHAR	40		0Forename Data Element

The next field will be called 'Title' and the Data Element 'ZTITLE', follow the steps above again to create this field with the following information:

The Data element short text should read 'Title Data Element' and the domain should be named 'ZTITLE'.

The Domain Short text should be 'Title' and the Data type is again 'CHAR'. This time the Length and Output length will be 15.

The Field labels should all read 'Title'.

Activate all of these and go back to view the new, fifth field in the Table.

The final field which will be created for this table is for Date of Birth. In the Field box type 'DOB' and create the Data element 'ZDOB' using the steps from the previous section and this information:

The Data element short text should read 'Date of Birth Data Element' and the domain should be named 'ZDOB'.

The Domain Short text should be 'Date of Birth' and the Data type is, this time, 'DATS', after which an information box will appear to confirm this. Click the green tick to continue:

DTyp	DT...	Short text
ACCP		Posting period YYYYMM
CHAR		Character string
CLNT		Client
CUKY		Currency key, referenced by CURR fields
CURR		Currency field, stored as DEC
DATS		Date field (YYYYMMDD) stored as char(8)
DEC		Counter or amount field with comma and sign
FLTP		Floating point number, accurate to 8 bytes
INT1		1-byte integer, integer number <= 255
INT2		2-byte integer, only for length field before LCHR or LRAW
INT4		4-byte integer, integer number with sign



For the DATS data type, the Length and Output lengths are set automatically at 8 and 10 (the Output length is longer as it will automatically output dividers between the day, month and year parts of the date).

The Field labels should all read 'Date of Birth', except the 'Short' label where this will not fit, so just type 'DOB' here. Activate the Domain and Data element, and return to the table.

Field	K..	I..	Data element	DTyp	Len...	Dec...	Short text
CLIENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	MANDT	CLNT	3	0	Client
EMPLOYEE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ZEENUM	NUMC	8	0	Employee Data Element
SURNAME	<input type="checkbox"/>	<input type="checkbox"/>	ZSURNAME	CHAR	40	0	Surname Data Element
FORENAME	<input type="checkbox"/>	<input type="checkbox"/>	ZFORENAME	CHAR	40	0	Forename Data Element
TITLE	<input type="checkbox"/>	<input type="checkbox"/>	ZTITLE	CHAR	15	0	Title Data Element
DOB	<input type="checkbox"/>	<input type="checkbox"/>	ZDOB	DATS	8	0	Date of Birth Data Element

Technical Settings

Once this has been saved, the next step is to move on to maintaining the technical settings of the Table. Before creating the final Database table, SAP will need some more information about the table being created.

Select 'Technical settings' via the toolbar above the table, through the 'Go to' menu, or

with the shortcut CTRL+SHIFT+F9.

Here, it is important to tell the system what Data class is to be used, so select the drop down button. There are five different options, with accompanying descriptions. For this table, select the first, labelled 'APPL0', and double-click it:

Dictionary: Maintain Technical Settings www.SapTrainingHQ.com

Revised<-> active

Name: ZEMPLOYEES Transparent Table

Short text: Employees

Last changed: BCUSER 11.03.2012

Status: New Not saved

Logical storage parameters

Data class: 

Size category:

Possible Entries: Data Class

Data class	Description
APPL0	Master data, transparent tables
APPL1	Transaction data, transparent tables
APPL2	Organization and customizing
USER	Customer data class
USER1	Customer data class

System data types

For the 'Size category' field, again click the drop-down button. Here, you have to make an estimate as to the amount of data records which will be held within the table so that the system has some idea of how to create the tables in the underlying database. In this instance, it will be a relatively small amount of information, so select the first size category, labelled 0:



SzCat	Number of data records of table expected	
0	0 to	5.300
1	5.300 to	21.000
2	21.000 to	86.000
3	86.000 to	340.000
4	340.000 to	1.300.000
5	1.300.000 to	2.700.000
6	2.700.000 to	110.000.000

Below this are the Buffering options. Here, 'Buffering not allowed' should be selected:



Buffering

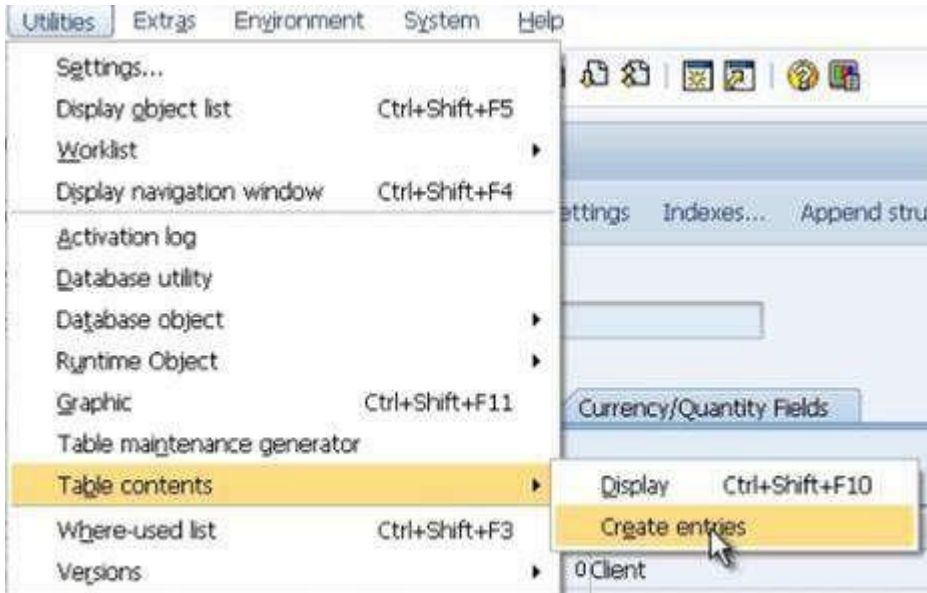
- Buffering not allowed
- Buffering allowed but switched off
- Buffering switched on

This prevents the table contents from being loaded into memory for reading, stopping the table from being read in advance of the selection of the records in the program. You may, correctly, point out that it may be advantageous to hold the table in the memory for speed efficiency, but in this example, this is not necessary. If speed was an issue in a development, buffering would then be switched on, ensuring the data is read into memory. In the case of large tables which are accessed regularly but updated infrequently, this is the option to choose.

Nothing else on the 'Maintain Technical Settings' screen needs to be filled at this point, so click Save and then go back to the table itself. If all of this is successful, then the table should now be in a position to be activated and the entry of records can begin. Click the Activate icon to activate the table and check the status bar, which should again read 'Object Activated'.

Entering Records into a Table

Now that the table has been created, data can be entered. To do this, enter the 'Utilities' menu, scroll to 'Table contents', and then 'Create Entries':



A Data-entry screen will appear which has automatically been generated from the table created. The field names correspond here to the technical names given when we created them. To change these to the Field labels which we set up, enter the 'Settings' menu and select 'User Parameters'. This facility allows you to tailor how tables look for your own specific user ID. Select the 'Field label' radio button and click 'Continue':



The Field labels created will now appear as they were defined when creating the table:



The screenshot shows a web form titled "Table ZEMPLOYEES Insert". At the top left is a "Reset" button. Below it are six input fields, each with a label to its left: "Client" (a small square), "Employee Number" (a yellow rectangular box), "Surname" (a long horizontal box), "Forename" (a long horizontal box), "Title" (a medium horizontal box), and "Date of Birth" (a medium horizontal box). A mouse cursor is pointing at the "Forename" field.

The Employee Number field is limited to 8 characters, and the data type was set to NUMC, so only numerical characters can be entered. Create a record with the following data:

- Employee Number: '10000001'
- Surname: 'Brown'
- Forename: 'Stephen'
- Title: 'Mr'
- Date of Birth: '16.02.1980':



The screenshot shows the same "Table ZEMPLOYEES Insert" form, but now with data entered into the fields. The "Employee Number" field contains "10000001". The "Surname" field contains "Brown". The "Forename" field contains "Stephen". The "Title" field contains "Mr". The "Date of Birth" field contains "16.02.1980" and has a small calendar icon to its right. A mouse cursor is pointing at the "Reset" button.

Press Enter and the system will automatically put the names in upper case, and validate each field to ensure the correct values were entered:

Table ZEMPLOYEES Insert

Reset

Client

Employee Number

Surname

Forename

Title

Date of Birth

Click Save and the status bar should state 'Database record successfully created'. Next, click the 'Reset' button above the data entry fields to clear the fields for the next entry.

Create another record with the following data:

- Employee Number '10000002'
- Surname 'Jones'
- Forename 'Amy'
- Title 'Mrs'
- Date of Birth '181169'.

Note that this time the Date of Birth has been filled in without the appropriate dividers. When Enter is pressed, the system automatically validates all fields, correcting the Date of Birth field to the correct formatting itself:

Client

Employee Number

Surname

Forename

Title

Date of Birth

Client	
Employee Number	10000002
Surname	JONES
Forename	AMY
Title	MRS
Date of Birth	18.11.1969

Save, Reset, and then further records can be entered following the same steps:

Client	
Employee Number	10000003
Surname	MICHAELS
Forename	ANDREW
Title	MR
Date of Birth	01.01.1977

Note that if dates are entered in the wrong format, an error message will appear in the status bar:

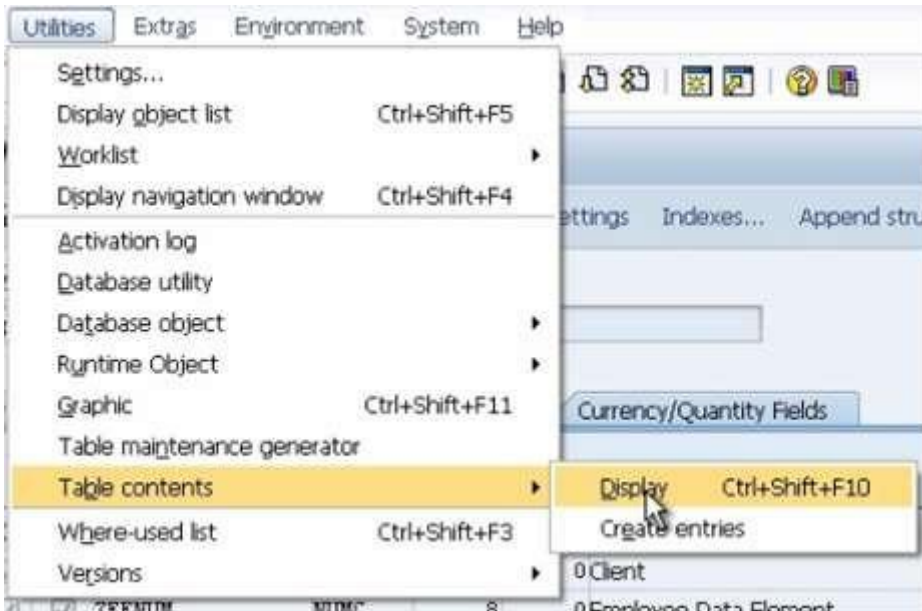
Date of Birth	16.08/2000
---------------	------------

❗ Enter date in the format __.__.__

Viewing the Data in a Table

Now that data has been entered into the table, the final few steps will allow this data to be viewed.

Having entered several data records in the manner discussed previously, click the Back key to return to the 'Dictionary: Display Table' screen. To view the table created with the data entered, from the 'Utilities' menu, select 'Table contents' and then 'Display':



A selection screen will then appear, allowing you to enter or choose filter values for the fields you created. The selection screen is very useful when you have lots of data in your table. In this case, though, only five records have been entered, so this is unnecessary. However, for example if you were to only want to focus on a single employee number, or a small range, these figures can be selected from this screen:




To view all of the records, do not enter any data here. Just click the 'Execute' button, which is displayed in the top left corner of the image above, or use the shortcut F8. You will now see a screen showing the data records you entered in the previous section:

Data Browser: Table ZEMPLOYEES Select Entries 5

Table: ZEMPLOYEES
Displayed fields: 6 of 6 Fixed columns: List width 0250

Client	Employee Number	Surname	Forename	Title	Date of Birth
000	10000001	BROWN	STEPHEN	MR	16.02.1980
000	10000002	JONES	AMY	MRS	18.11.1969
000	10000003	MICHAELS	ANDREW	MR	01.01.1977
000	10000004	NICHOLS	BRENDAN	MR	02.12.1958
000	10000005	MILLS	ALICE	MRS	16.08.2000

If further fields were to exist, the screen would scroll further to the right, meaning not all fields could be displayed simultaneously due to field size properties.

If you want to see all of the data for one record, double-click on the record and this will be shown. Alternatively, several records can be scrolled through by selecting the desired records via the check-boxes to the left of the 'Client' column and then clicking the 'Choose' icon on the toolbar: 

These can then be individually viewed and scrolled through with the 'Next entry' button:



Table ZEMPLOYEES Display



Client: 000

Employee Number: 10000001

Surname: BROWN

Forename: STEPHEN

Title: MR

Date of Birth: 16.02.1980

To return to the full table then, simply click the Back button, or press F3.

Experiment with the table created, using the toolbar's range of options to filter and sort the information in a number of ways:



For example, to organise alphabetically by forename, click to select the 'Forename' field, and then click the 'Sort ascending' button.

There are a number of things which can be achieved in this table view, and it can be a useful tool for checking the data within an SAP system without going through the transaction screens themselves.